ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА АСЦТ

Учебно-методический комплекс по дисциплине (УМК).

«Архитектура высокопроизводительных вычислительных систем»

Профиль: 710100 «Информатика и вычислительная система» (магистр)

Составил: ст.преподаватель кафедрф АСЦТ
Саданов А.Д.

ОШ-2020

«Жогорку өндүрүмдүү эсептөө системасынын архитектурасы» дисциплинасынын аннотациясы (магистратура)

Максаты: Параллель эсептөө системаларын уюштуруу принциптерин, салттуу архитектура менен эсептөө системаларын түзүүдө пайда болгон маселелерди чечүү методдорун жана технологияларын өздөштүрүү.

Мазмуну: Архитектура түшүнүгү Система, эсептөө системасы түшүнүгү. Аппараттык жана программалык камсыздоо. Системалык жана колдонмо архитектурасы. Фон Нейман архитектурасы: салттуу эсептөө системасынын түзүмү жана уюштуруунун негизги принциптери, чектөөлөр. Ар кандай муундагы процессорлордун өзгөчөлүктөрү. Эсептөө системасын өркүндөтүү тенденциясы. Параллелдик деңгээлдер. Командалык жана маалымат агымында параллелизмдин болушуна негизделген ЭС архитектурасынын жалпы классификациясы. Конвейеризациялоо жөнүндө түшүнүк, конвейерлердин түрлөрү.

Дисциплинанын орду: 710100 «Информатика жана эсептөө техникасы» магистр программасынын НББПнын кесиптик циклинин тандоо курсу бөлүмүндө окутулат.

Пререквизиттер: “информатика”, “программалоо”, “объектке багытталган программалоо”.

Постреквизиттер: “программалык тиркемелерди иштеп чыгуу”, ошондой эле окуп үйрөнүү жана өндүрүштүк практикалар”

Дисциплинаны окутуудагы күтүлүүчү натыйжалар: Жаңы теорияларды, усулдарды жана ыкмаларды сындоого жана терең түшүнүүгө, жаңы билим алуу үчүн дисциплиналар аралык ыкманы колдонууга жана ар кандай илимдердин жетишкендиктерин интеграциялоого жөндөмдүү**(ЖИК-1);** Өздөштүрүлгөн билимдердин негизинде жыйынтык чыгарууга, материалдарды так жана айкын түшүндүрүүгө (адиске жана адис эмеске) жөндөмдүү. Өз алдынча билим деңгээлин өрчүтүүгө жөндөмдүү**(АК-4);** Жамаатты, анын ичинде дисциплиналар аралык долбоорлорду жетектөөгө, команданын максаттарынын түзүлүшүнө таасир берүүгө, максаттарга жетүү үчүн зарыл болгон багытта анын социалдык-психологиялык климатына таасир этүүгө, ишмердиктин жыйынтыктарынын сапатына туура баа берүүгө жөндөмдүү**(СИЖМК-4);**. Программалык жана аппараттык долбоорлорду ишке ашыруучу үчүн инструменталдык каражаттарды иштеп чыгууга жана колдонууга жөндөмдүү **(КК-13)**; Иштелип чыккан системалардын натыйжалуулугуна техникалык, экономикалык жана функционалдык, нарктык анализ жүргүзө алат **(КК-11);**

Компетенциялык көрсөткүчтөр: Дисциплинаны окуп-үйрөнүп, өздөштүрүүнүн натыйжасында магистрант төмөнкү компетенциялык көрсөткүчтөрдүн денгээлине ээ болуусу керек:

Билүүсү зарыл: эсептөө системасынын архитектурасынын аныктамасы, колдонмо жана система архитектурасы, компьютердик процессордун иштөө принциптери;

Билгичтиктер: машинага багытталган тилдерде жазылган программалардын баштапкы кодун окуп, талдоо

Ээ болуу: стандарттык типтердин негизинде жаңы типтерди куруунун каражаттарын колдонуу

Колдоно алуу: . Intel тутумунун архитектурасынын мүмкүнчүлүктөрүн практика жүзүндө колдонуу, системалык функцияларын колдоно алуу, системалык программалык камсыздалышты иштеп чыгууга даяр болуу.

Аннотация дисциплины «Архитектура высокопроизводительных вычислительных систем» (магистратура).

Цель: Овладеть принципами организации параллельных вычислительных систем, методами и технологиями решения задач, возникающих при создании компьютерных систем с традиционной архитектурой.

Содержание: Концепция архитектуры системы, концепция компьютерных систем. Железо и софт. Системная и прикладная архитектура. Архитектура фон Неймана: структура традиционной компьютерной системы и основные принципы организации, ограничения. Особенности процессоров разных поколений. Тенденции развития компьютерных систем. Параллельные уровни. Общая классификация архитектуры ВС, основанная на существовании параллелизма в потоках команд и информации. Понятие конвейеризации, типы конвейеров.

Дисциплина: 710100 «Архитектура высокопроизводительных вычислительных систем» преподается на элективном курсе профессионального цикла.

Пререквизиты: «информатика», «программирование», «объектно-ориентированное программирование».

Постреквизиты: «Разработка программных приложений", а также производственная практика.

Ожидаемые результаты в обучении дисциплине: способен глубоко понимать и критически оценивать новейшие теории, методы и способы, использовать междисциплинарный подход и интегрировать достижения различных наук для приобретения новых знаний; (ОК-1); способен делать выводы, четко и ясно объяснять (транслировать) материал на основе приобретенных знаний (как специалисту, так и не специалисту). способен к дальнейшему самостоятельному обучению. (ИК-4); способен руководить коллективом, в том числе междисциплинарными проектами, влиять на формирование целей команды, воздействовать на ее социально-психологический климат в нужном для достижении целей направлении, корректно оценивать качество результатов деятельности. (СЛК-4); способен проводить технико-экономический и функционально-стоимостный анализ эффективности проектируемых систем;(ПК-11); способен проектировать и применять инструментальные средства реализации программно-аппаратных проектов;(ПК-13);

Показатели компетентности: В результате изучения и усвоения дисциплины магистрант должен иметь следующий уровень компетентности:

Должен знать: определение архитектуры компьютерной системы, архитектуры приложений и системы, принципов работы процессора компьютера;

Навыки: чтение и анализ исходного кода программ, написанных на машинно-ориентированных языках.

Приобретение: использование инструментов для создания новых типов на основе стандартных типов.

Использование:. Практическое применение архитектуры системы Intel, умение использовать системные функции, готовность к разработке системного ПО.

Выписка из решений заседания кафедры АССТ

Протокол №__ от «____»__Сентябрь_____ 2020 г.

Согласно матрицы компетенций ОПОП по программе «710100 Информатика и вычислительная техника» (магистр)

дисциплина «Архитектура высокопроизводительных вычислительных систем» формирует следующие:

1) компетенции:

(ОК-1) способен глубоко понимать и критически оценивать новейшие теории, методы и способы, использовать междисциплинарный подход и интегрировать достижения различных наук для приобретения новых знаний;

(ИК-4) способен делать выводы, четко и ясно объяснять (транслировать) материал на основе приобретенных знаний (как специалисту, так и не специалисту). способен к дальнейшему самостоятельному обучению;

(СЛК-4) способен руководить коллективом, в том числе междисциплинарными проектами, влиять на формирование целей команды, воздействовать на ее социально-психологический климат в нужном для достижении целей направлении, корректно оценивать качество результатов деятельности;

(ПК-11) способен проводить технико-экономический и функционально-стоимостный анализ эффективности проектируемых систем;

(ПК-13) способен проектировать и применять инструментальные средства реализации программно-аппаратных проектов;

2) результаты обучения ООП:

РО 1	Проектная деятельность: разработка требований и особенностей к отдельным компонентам объектов профессиональной деятельности на основе анализа возможностей, образцов технических средств и требований пользователей; Проектирование аппаратных компонентов; Умеет использовать инструменты программирования и компьютерное оборудование для эффективной реализации оборудования.
РО 2	Технологическая деятельность: создание компьютерных систем, компонентов автоматизированных систем, программ, пакетов программного обеспечения с использованием передовых технологий в требуемом качестве и в срок; Тестирование и ремонт аппаратного и программного обеспечения; Разработка программ и методов их тестирования, обследование объектов профессиональной деятельности; Умеет анализировать и разрабатывать программы тестирования и методики тестирования для объектов профессиональной деятельности, умеет проводить тесты

Зав. кафедрой, доцент: **Молдояров У.Д.**

РЕЦЕНЗИЯ
на рабочую программу по дисциплине «Архитектура высокопроизводительных вычислительных систем»
преподавателя Саданова А.Д.

на рабочую программу по дисциплине «Архитектура высокопроизводительных вычислительных систем» для специальности 710100 «Информатика и вычислительная техника»-ИВТ (магистр).
Автор: А.Саданов, старший преподаватель кафедры АСЦТ.

Рабочая программа учебной дисциплины разработана на основе государственного образовательного стандарта профессионального образования по специальности 710100 «Информатика и вычислительная техника»(ИВТ) базисного учебного плана, примерной программы.

Рабочая программа включает обязательные компоненты: структуру и содержание, условия реализации, контроль и оценку результатов освоения дисциплины. Содержание рабочей программы охватывает весь материал, необходимый для обучения студентов высших учебных заведений. Раскрываются основные цели и задачи изучаемой, дисциплины- требования к результатам освоения дисциплины. В структуре и содержании учебной дисциплины программы определены темы и количество часов на их изучение, указывается объем часов максимальной, обязательной аудиторной учебной нагрузки, самостоятельной работы обучающихся, перечислены виды обязательной аудиторной учебной нагрузки, самостоятельной работы и форма итоговой аттестации по дисциплине.

Содержание программы направлено на приобретение обучающимися знаний, умений и навыков, направленных на формирование общих компетенций ОК3, и соответствует объему часов, указанному в рабочем учебном плане

Программа состоит из структуры и содержания учебной дисциплины, условий реализации программы дисциплины, контроля и оценки результатов освоения дисциплины.

Рабочая программа предполагает распределение тем и изучение материала по разделам:
-основы теории архитектуры высокопроизводительных вычислительных систем;
-виды архитектуры высокопроизводительных вычислительных систем;
-свойства архитектуры высокопроизводительных вычислительных систем;
-классификация архитектуры высокопроизводительных вычислительных систем;

Данная рабочая программа может быть рекомендована для изучения дисциплины «Архитектуры высокопроизводительных вычислительных систем» в ВУЗе.

Старший преподаватель
кафедры АСЦТ : Сулайманов А.А.

РЕЦЕНЗИЯ

на рабочую программу по дисциплине «Архитектура высокопроизводительных вычислительных систем»

преподавателя Саданова А.Д.

на рабочую программу по дисциплине «Архитектура высокопроизводительных вычислительных систем» для специальности 710100 «Информатика и вычислителная техника»-ИВТ (магистр). Автор: **А.Саданов**, старший преподаватель кафедры АСЦТ.

Рабочая программа учебной дисциплины разработана на основе государственного образовательного стандарта профессионального образования по специальности 710100 «Информатика и вычислительная техника»(ИВТ), базисного учебного плана, примерной программы. Программа состоит из структуры и содержания учебной дисциплины, условий реализации программы дисциплины, контроля и оценки результатов освоения дисциплины.

Рабочая программа предполагает распределение тем и изучение материала по разделам:

- основы теории архитектуры высокопроизводительных вычислительных систем;
- виды архитектуры высокопроизводительных вычислительных систем;
- свойства архитектуры высокопроизводительных вычислительных систем;
- вспомагательные программы архитектуры высокопроизводительных вычислительных систем;

Оформление соответствует всем предъявленным требованиям. Темы самостоятельной работы подобраны грамотно. Рабочая программа составлена грамотно, язык и стиль изложения соответствуют общепринятым нормам. Термины используются правильно.

Данная рабочая программа может быть рекомендована для изучения дисциплины «Архитектура высокопроизводительных вычислительных систем» в ВУЗе.

Директор ОсОО “ЭлКат” : У.Абдыкалыков

Министерство образования и науки Кыргызской Республики

Ошский государственный университет

Факультет математики и информационных технологий

Кафедра АСЦТ

«УТВЕРЖДАЮ»

Заведующий кафедрой,
к.ф.-м.н., доцент
_____ Молдояров У.Д.

«___» _сентябрь_2020г

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ

«Архитектура высокопроизводительных вычислительных систем»

Составил: **старший преподаватель кафедры АСЦТ**
Саданов А.Д.

Ош-2020

1. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ

Фонд оценочных средств (ФОС) для текущего контроля успеваемости и промежуточной аттестации по дисциплине «Архитектура высокопроизводительных вычислительных систем» разработан в соответствии с рабочей программой, входящей в ОПОП направления подготовки 710100 «Информатика и вычислительная техника»(ИВТ). В результате освоения учебной дисциплины Архитектура высокопроизводительных вычислительных систем обучающийся должен обладать предусмотренными ФГОС по профессии/специальности 710100 «Информатика и вычислительная техника»(ИВТ_магистр) следующими умениями, знаниями, которые формируют профессиональную компетенцию, и общими компетенциями:

Должен знать: определение архитектуры компьютерной системы, архитектуры приложений и системы, принципов работы процессора компьютера;

Дожен уметь: чтение и анализ исходного кода программ, написанных на машинно-ориентированных языках.

Дожен владеть: использование инструментов для создания новых типов на основе стандартных типов. Практическое применение архитектуры системы Intel, умение использовать системные функции, готовность к разработке системного ПО.

№№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
1	Начальные сведения по устройству компьютера	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
2	1.1. Данные и программы Представление данных в компьютере. Текстовые, графические данные	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
3	Числовые данные. Форматы представления чисел в компьютере	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
4	Логические основы обработки данных. Понятие такта.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
5	Схема памяти на базовых вентилях. Интегральные схемы	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
6	Архитектура компьютера на базе процессора xxxx86.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
7	Машинные команды процессора xxxx86	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
8	Развитие архитектуры и параллелизм вычислений.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
9	Многоуровневая организация памяти.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
10	Кеш память. Механизмы работы кэша.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
11	Шины, циклы шин,многошинная архитектура.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
12	Улучшение эффективности процессора.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
13	Внешняя память.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
14	Системный блок и периферийные устройства.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
15	Оценка производительности вычислительных систем.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
16	Классификация архитектур.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
17	Обзор основных семейств микропроцессоров.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов
18	Параллельные архитектуры. Неклассические архитектуры.	ОК-1,ИК-4, ПК-11, ПК-13,	Перечень вопросов

* Наименование тем (разделов) берётся из рабочей программы.

Оценка освоения учебной дисциплины

Формы и методы оценивания.

Предметом оценки служат умения и знания, предусмотренные ФГОС по дисциплине Архитектура вычислительных систем, направленные на формирование общих компетенций. Промежуточной формой контроля знаний, умений и навыков по дисциплине является **дифференцированный зачет.**

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ТЕСТИРОВАНИЯ

1. Память МПС - это...

А) **совокупность устройств, служащих для запоминания, хранения и выдачи информации;**

Б) память, предназначенная для долговременного хранения информации;

В) память, в которой хранится информация, присутствие которой постоянно необходимо в компьютере;

Г) память, в которой хранятся программы, предназначенные для обеспечения диалога пользователя с ЭВМ.

2. Важнейшими характеристиками ЗУ являются:

А) **емкость памяти (пропуская способность);** Б) тактовая частота;

В) **удельная емкость; Г) быстродействие.**

3. Основные операции памяти:

А) **запись информации в память;**

Б) тестирование узлов компьютера;

В) обработки информации;

Г) считывание информации из памяти.

4. Максимальное количество данных памяти, которые могут в ней храниться:

А) размер памяти; **Б) емкость памяти;**

В) резерв памяти;

Г) объем памяти.

5. В МПС содержатся:

А) сверхоперативная память; Б) оперативная память;

В) постоянная память; **Г) все ответы верны.**

6. В компьютере управление работой системной шины осуществляет:

А) **микропроцессор;** Б) оперативная память;

В) драйвер системной шины;

Г) контроллер системной шины.

7. Каждая ячейка основной памяти компьютера имеет свой

А) индекс; **Б) адрес;**

В) размер; Г) тип.

8. Оперативная память служит для ...

А) обработки информации;

Б) хранения информации, изменяющейся в ходе выполнения процессором операций по ее обработке;

В) запуска программ;

Г) тестирования узлов компьютера.

9. Что такое Кэш-память?

А) память, предназначенная для долговременного хранения информации, независимо от того, работает ЭВМ или нет;

Б) это сверхоперативная память, в которой хранятся наиболее часто используемые участки оперативной памяти;

В) память, в которой хранятся системные файлы операционной системы;

Г) память, в которой обрабатывается одна программа в данный момент времени.

10. ПЗУ - это память, в которой хранится...

А) информация, когда ЭВМ работает;
Б) исполняемая в данный момент времени программа и данные, с которыми она непосредственно работает;
В) программы, предназначенные для обеспечения диалога пользователя с ЭВМ;
Г) информация, присутствие которой постоянно необходимо в компьютере.
11. Укажите верное высказывание:
А) **внутренняя память - это память высокого быстродействия и ограниченной емкости;**
Б) внутренняя память предназначена для долговременного хранения информации;

В) внутренняя память производит арифметические и логические действия.
12. Оперативная память имеет следующую структуру:
А) состоит из ячеек, каждая ячейка имеет адрес и содержание. Б) разбита на сектора и дорожки, информация записана в виде намагниченных и не намагниченных областей; В) разбита на кластеры, информация записана в виде намагниченных и не намагниченных областей;
13. Вид организации памяти, при котором размещение и поиск информации в запоминающем массиве основан на использовании древа хранения слова:
А) **адресная;**
Б) стековая;
В) ассоциативная;
Г) внешняя;
14. Вид организации памяти, при котором поиск нужной информации производится не по адресу, а по ее содержанию:
А) адресная; Б)
стековая;
В) **ассоциативная;**
Г) внешняя;
15. Вид организации памяти, доступ к которой организован по принципу: "последним записан - первым считан" (Last Input First Output - LIFO):
А) адресная;
Б) стековая;
В) ассоциативная;
Г) внешняя;
16. К методам защиты памяти относят:
А) **метод граничных регистров;**
Б) метод управления паролями;
В) **защита отдельных ячеек памяти;**
Г) метод ключей защиты.
17. Перечислите уровни кэш-памяти:
А) **вторичный кэш (внешний); Б) кэш**
третьего уровня;
В) **первичный кэш (внутренний);** Г)
многоуровненвый кэш.
18. Часть оперативной памяти, в которую при запуске компьютера переписывается содержание постоянной памяти, и заменяющая эту постоянную память на время работы компьютера:
А) сверхоперативная; **Б)теневая;**
В) динамическая Г) статическая.
19. Тип памяти, предназначенный для хранения и считывания данных, которые никогда не изменяются:
А) внешняя; Б) внутренная;
В) **постоянная;**
Г) статичная
20. Что такое статическая память?
А) часть памяти ЭВМ, предназначенная для размещения временных наборов данных;
Б) вид памяти, в котором положение данных и их значение не изменяются в процессе хранения и считывания;
В) вид памяти, в которой все области поиска могут быть доступны одновременно; Г) память, записи в которых не стираются при снятии электропитания.

21. Разновидность энергозависимой полупроводниковой памяти, в которой хранимая информация с течением времени разрушается, поэтому для сохранения записей необходимо производить их периодическое восстановление (регенерацию), которое выполняется под управлением специальных внешних схемных элементов:

А) **динамическая;**

Б) ёмкостная;

В) магнитная;

Г) энергонезависимая.

22. При сравнении объемов оперативной и постоянной памяти:

А) **Объем оперативной памяти больше, чем постоянной памяти;** Б)

Объем оперативной памяти меньше, чем постоянной памяти;

В) Объем оперативной памяти равен объему постоянной памяти;

23. По способу организации доступа различают устройства памяти:

А) с непосредственным или произвольным доступом; Б) с прямым или циклическим доступом;

В) с последовательным доступом; **Г) все ответы верны.**

24. В зависимости от реализуемых в памяти операций обращения различают:

А) **память только для считывания информации;**

Б) полупроводниковая память;

В) **память с произвольным обращением, т.е. возможна и запись и считывание;**

Г) память последовательного действия.

25. Что такое память с последовательным доступом?

А) **Вид памяти, в котором последовательность обращенных к ним входных сообщений и выборок данных соответствует последовательности, в которой организованы их записи;**

Б) Вид памяти, в которой адресация, запись и выборка данных производится не побайтно, а пословно;

В) Память, содержащая управляющие программы или микропрограммы;

Г) Вид памяти, допускающий одновременное использование его несколькими процессорами.

Пакет преподавателя

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
правильный ответ	А	А,В ,Г	А, Г	Б	Г	А	Б	Б	Б	Г	А	А	А	В	Б	А,В ,Г	А,Б, В

№ задания	18	19	20	21	22	23	24	25
правильный ответ	Б	В	Б	А	А	Г	А, В	А

Перечень вопросов для контрольной работы.

1. Какие основные группы команд включает в себя система команд процессора?
2. Для чего предназначены команды пересылки данных?
3. Какие операции выполняют арифметические команды?
4. Каковы функции логических команд?
5. Перечислите логические операции, выполняемые логическими командами процессора?
6. Для чего предназначены команды переходов?
7. Какие функции выполняют команды пересылки данных?
8. Для чего в систему команд вводится специальная команда для строчной (или цепочечной) пересылки данных?
9. Для чего используется функция обмена с устройствами ввода/вывода?
10. Что относится к командам обмена информацией?
11. Как работают команды операций с фиксированной запятой?
12. Что используют команды операций с плавающей запятой?
13. Для чего предназначены команды очистки?

14. Что такое команды инкремента?
15. Для чего предназначены команды сравнения?
16. Что позволяют вычислять команды логических операций?
17. Что позволяют делать команды сдвигов?
18. Для чего нужны циклические сдвиги?
19. Для чего предназначены команды проверки битов и операндов?
20. Что позволяют сделать команды установки и очистки битов регистра состояния процессора?

21. На какие группы делятся команды переходов без возврата?
22. Для чего нужны команды безусловных переходов?
23. Для чего нужны команды условных переходов?
24. Для чего нужны команды переходов с дальнейшим возвратом?
25. Каково основное назначение команд прерываний?
26. Какие существуют методы адресации операндов?
27. Что предполагает непосредственная адресация?
28. Что предполагает абсолютная адресация?
29. Что предполагает регистровая адресация?
30. Что предполагает укороченная адресация?
31. Что предполагает косвенно-регистровая адресация?
32. Как работает автоинкрементная адресация?
33. Как работает автодекрементная адресация?
34. Как работает индексная адресация?
35. Как работает относительная адресация?
36. Как работает страничная адресация?

Перечень вопросов для контроля лабораторных и практических занятий.

Вариант 1

1. Через какие логические функции выражается функция Шеффера?
2. В чем особенность RS-триггера?
3. Каково условное обозначение синхронного RSC-триггера?
4. На каких логических элементах может быть построен триггер?
5. Почему взяты буквы R и S для входов RS-триггера?
6. В чем особенность JK-триггера?
7. Чем отличается синхронный триггер от асинхронного?
8. Как называют D-триггер и почему?
9. В какое состояние переходит JK-триггер при подаче на входы J и K логической 1 (C=1)?
10. Как называется состояние триггера, когда выход Q триггера не меняется?
11. Как называются входы R и S в синхронном триггере?
12. На базе каких триггеров можно построить D-триггер?
13. Что такое регистры?
14. Какие бывают типы регистров?
15. Что такое шифратор?
16. Сколько выходов имеет дешифратор с n входами?
17. Что такое мультиплексор?
18. Какие входы имеет мультиплексор?
19. Для чего предназначен цифровой компаратор?
20. Что такое сумматор?

Вариант 2

1. Через какие логические функции выражается функция Пирса?
2. В чем особенность синхронного RSC-триггера?
3. Приведите примеры синхронных триггеров?
4. В чем особенность D-триггера?
5. В чем особенность T-триггера?
6. Как называют T-триггер и почему?

7. В какое состояние переходит Т-триггер при подаче на вход Т логической 1 (С=1)
8. Почему состояние триггера называют запрещенным при подаче на входы R и S логической 1?
9. Какой триггер называют двухтактным?
10. Как называют вход С в синхронном триггере и какова его основная функция?
11. На базе каких триггеров можно построить Т-триггер?
12. Что определяет количество триггеров в регистре?
13. Для чего предназначены сдвигающие регистры?

14. Сколько входов имеет шифратор с n выходами?
15. Какое условное обозначение имеет шифратор?
16. Что такое демультиплексор?
17. Что такое преобразователь кодов?
18. Что такое счетчики?
19. Какие виды счетчиков вам известны?
20. Что такое полусумматор?

Перечень вопросов для дифференцированного зачета

1. Основные телекоммуникационные системы.
2. Представление информации в вычислительных системах.
3. Непозиционные системы счисления.
4. Позиционные системы счисления. Общий вид числа.
5. Недесятичная арифметика.
6. Правила перевода чисел в различные системы счисления.
7. Прямой, обратный и дополнительный коды числа.
8. Естественная и нормальная формы представления чисел.
9. Арифметические операции над числами с фиксированной точкой.
10. Арифметические операции над числами с плавающей точкой.
11. Классификация вычислительных машин.
12. Комплектация вычислительных машин.
13. Построение цифровых вычислительных систем. Особенности цифровых систем.
14. Специализированные и универсальные системы.
15. Основные логические узлы ЭВМ.
16. Фон Неймановская архитектура.
17. Гарвардская архитектура
18. Основные типы архитектур ЭВМ.
19. Микропроцессор.
20. Виды микропроцессоров.
21. Функции и характеристики микропроцессоров.
22. Система команд процессора.
23. Однопроцессорные вычислительные системы.
24. Многопроцессорные вычислительные системы.
25. Организация вычислений в вычислительных системах.
26. Параллелизм и конвейеризация вычислений.
27. Классификация ВС по М.Флинну.
28. Класс вычислительных систем SISD.
29. Класс вычислительных систем SIMD.
30. Класс вычислительных систем MISD.
31. Класс вычислительных систем MIMD.
32. Микропроцессоры с архитектурой CISC.
33. Микропроцессоры с архитектурой RISC.
34. Использование DSP-процессоров в вычислительной технике.

35. Режимы работы процессора.Реальный режим.
36. Режимы работы процессора.Защищённый режим.
37. Режимы работы процессора.Виртуальный режим.
38. Основы программирования процессора.
39. Основные команды процессора.
40. Внутримашинный системный интерфейс ЭВМ.
41. Набор микросхем системной логики (чипсет).

42. Системная шина ЭВМ, виды шин.
43. Шины расширения.
44. Локальные шины.
45. Организация оперативной памяти ЭВМ.
46. Использование кэш-памяти.
47. Типы современных микросхем оперативной памяти.
48. Типы современных модулей оперативной памяти.
49. Запоминающие устройства ЭВМ.
50. Внешняя и постоянная память ЭВМ.
51. Взаимодействие внутренних компонентов ЭВМ.
52. Система прерываний.
53. Тактовый генератор ЭВМ.
54. Организация прямого доступа к памяти.
55. Порты ввода вывода.
56. Принцип последовательной передачи информации.
57. Принцип параллельной передачи информации.
58. Коммуникационные порты ЭВМ.
59. Электропитание ЭВМ.
60. Защита оборудования ЭВМ.
61. Проблемы электропитания.

Условия проведения дифференцированного зачета.

Промежуточная аттестация освоения дисциплины **Архитектура высокопроизводительных систем** осуществляется на дифференцированном зачете. Условием допуска к дифференцированному зачету является положительная аттестация по практическим работам и формам текущего контроля. На выполнение итогового теста отводится 45 минут. Перечень вопросов доведен до сведения студентов для подготовки.

Информационное обеспечение обучения. Основные источники:

1. А.Н.Степанов., Архитектура вычислительных систем и компьютерных сетей. Учебник, «Питер Пресс», 2007г
2. С.Л. Сергеев., Архитектура вычислительных систем., БХВ-Петербург, 2010г.
3. Колдаев, В.Д. Архитектура ЭВМ: учебное пособие для учрежд. СПО (Гриф)/ В.Д.Колдаев, С.А.Лупин С.А. - М.: ФОРУМ, 2017. - 383с. ISBN: 978-5-8199-0689-7
4. Максимов, Н. В. Архитектура ЭВМ и вычислительных систем: учебник для учрежд. СПО/Н.В. Максимов, Т. Л. Партыка, И. И. Попов. - М.: ФОРУМ, 2015.
5. Чекмарев, Ю. В. Вычислительные системы, сети и телекоммуникации: учебное пособие [Электронный ресурс] / Ю. В. Чекмарев. - 2-е изд. испр. и доп. - М.: ДМК Пресс, 2009. - 184 с.:

Дополнительные источники:

1. Александров, Е.К. Микропроцессорные системы: учебное пособие [Электронный ресурс] : учебное пособие для вузов/ Е.К. Александров [и др.].— Электрон. текстовые данные.— СПб.: Политехника, 2016.— 936 с.— ЭБС «IPRbooks» / URL: http://www.iprbookshop.ru/59491html. (дата обращения 20.03.2018). - Режим доступа: ограничения по логину и паролю.
2. Таненбаум, Э. Архитектура компьютера./ Таненбаум Э., Остин Г. 6-е изд.- СПб.: Питер, 2017.- 816с. - ISBN 978-5-496-00337-7.
3. Сенкевич, А.В. Архитектура ЭВМ и вычислительные системы: учебник для студ. учрежд. СПО. / Сенкевич А.В. - М.: Академия, 2014. - 240с. ISBN 978-5-7695-6462-

5.Шкала оценки образовательных достижений по результатам работы.

Применяется накопительная система оценивания, соответствующая традиционной пятибалльной шкале. Во время дифференцированного контроля проверяются обязательные умения работать с информацией, представленной в тестовой форме.
оценка «отлично» - при выполнении 87-100% заданий
оценка «хорошо» - при выполнении 74-86% заданий
оценка «удовлетворительно» - при выполнении 61-73% заданий
оценка «неудовлетворительно» - при выполнении менее 60% заданий

Критерии оценки

Критерии оценки: оценивается как процесс выполнения задания, так и его результат:

Оценка процесса выполнения задания:

- обращение студента к информационным источникам, оптимальное использование найденной информации;
- рациональное распределение времени на выполнение задания.

Оценка подготовленного задания.

Например, при решении комплексной ситуационной задачи можно использовать следующие критерии оценки:

«отлично»
-дается комплексная оценка предложенной ситуации;
-демонстрируются глубокие знания теоретического материала и умение их применять;
- последовательное, правильное выполнение всех заданий;
-умение обоснованно излагать свои мысли, делать необходимые выводы.

«хорошо»
-дается комплексная оценка предложенной ситуации;
-демонстрируются глубокие знания теоретического материала и умение их применять;
- последовательное, правильное выполнение всех заданий;
-возможны единичные ошибки, исправляемые самим студентом после замечания преподавателя;
-умение обоснованно излагать свои мысли, делать необходимые выводы.

«удовлетворительно»
-затруднения с комплексной оценкой предложенной ситуации;
-неполное теоретическое обоснование, требующее наводящих вопросов преподавателя; -выполнение заданий при подсказке преподавателя;
- затруднения в формулировке выводов. **«неудовлетворительно»**
- неправильная оценка предложенной ситуации; -отсутствие теоретического обоснования выполнения заданий.

При выполнении заданий в тестовой форме обычно используются следующие критерии оценки

Процент результативности (правильных ответов)	Качественная оценка уровня подготовки	
	балл (отметка)	вербальный аналог
86 - 100	5	отлично
74 – 86	4	хорошо
61 – 73	3	удовлетворительно
менее 60	2	неудовлетворительно

Тема 1: Начальные сведения по устройству компьютера

Компьютер представляет собой универсальное устройство, которое используется для автоматической обработки информации, осуществляемой по заранее составленному плану. Такой план, обладающий необходимыми для получения требуемого результата свойствами, принято называть **алгоритмом,** а его запись в определенной, «понятной» компьютеру форме называется **программой.**

В отечественной литературе до 1985 г. в основном использовались термин **электронная вычислительная машина** и его аббревиатура **ЭВМ,** часто встречался также оборот **вычислительная машина.** После 1985 г. широкое распространение получил англоязычный термин «компьютер» (от computer — вычислитель). Мы будем использовать эти термины в основном как равноправные. Следует, однако, отметить, что в последнее время активно ведутся разработки компьютеров, работа которых основана на оптических, квантовых и некоторых других физических принципах. В связи с этим понятие «электронная вычислительная машина», в котором акцентируется, что машина построена на основе электронных устройств, становится более узким, чем понятие «компьютер».

> **Компьютер — это универсальное устройство, используемое для автоматизации процессов приема, хранения, обработки и передачи информации, которые осуществляются по заранее разработанным человеком программам.**

Обращаем внимание на важнейшие моменты этого определения:

1) компьютер представляет собой *устройство* (обычно электронное);
2) компьютер выполняет действия без вмешательства человека — *автоматически*;
3) для этого ему должен быть заранее передан разработанный человеком и записанный в специальной форме план действий — *программа*;
4) это устройство является *универсальным* в том смысле, что оно может выполнять *любую* обработку информации, для которой имеется соответствующая программа.

Существует множество автоматических устройств, которые так или иначе обрабатывают информацию. Это, например, всевозможные автоматы по продаже товаров, банкоматы (банковские автоматы), телефоны-автоматы, встроенные в бытовую технику устройства управления и т. д., но в подавляющем большинстве они имеют узкую специализацию и не могут выполнить произвольно заданную программу — именно в этом их коренное отличие от компьютеров, являющихся *универсальными устройствами обработки информации.*

Подчеркнем, что аппаратура компьютера принципиально не может выполнять никаких действий без программы, описывающей эти действия. Аппаратура компьютера без программы подобна автомобилю без водителя. Но и программы сами по

себе без аппаратуры не могут обработать данные, так же как водитель без автомобиля не может перевести пассажиров или груз. Таким образом, аппаратура компьютера и выполняемые программы образуют *систему.*

Совокупность устройств одного или нескольких компьютеров принято называть аппаратным обеспечением, или аппаратными ресурсами. Совокупность программ, которые могут быть выполнены на этих устройствах, называется программным обеспечением, или программными ресурсами. Вычислительной системой называется комплекс, который включает в себя совокупность взаимодействующих в процессе обработки данных аппаратного и программного обеспечения.

Напомним, что *системой* называется сложная структура, состоящая из взаимодействующих компонентов, каждый из которых в отдельности не обладает свойствами, присущими системе в целом. Свойства, которые присущи системе в целом и не присущи никакому ее компоненту, называют *системными свойствами.* В данном случае системным свойством является способность вычислительной системы обрабатывать данные. Вычислительная система — это общее понятие. Компьютер (вместе с программами) является частным случаем вычислительной системы.

1.1. Данные и программы

Из курса информатики известно, что **информация** представляет собой нематериальное содержание, которое извлекается человеком из некоторого материального сообщения. И следовательно, вести речь об обработке информации каким- либо «неодушевленным» устройством в принципе невозможно. Следует говорить об обработке сообщений, которая осуществляется такими устройствами. Но в силу укоренившейся традиции устной и письменной речи существенные различия между сообщением и информацией игнорируются, и даже в учебной и научной литературе почти всегда говорят о передаче информации, обработке информации, хранении информации и т. д.

Носителем сообщения называется любая материальная среда, служащая для его хранения или передачи. Примеры носителей сообщений: бумага, воздух, электромагнитные колебания, электронные схемы, магнитные и оптические диски и т. д.

В общем случае сообщение — это последовательность зафиксированных каким- либо образом сигналов. **Сигнал** представляет собой изменение во времени или в пространстве материального объекта — носителя сообщения или же некоторой его характеристики, при этом сама изменяющаяся характеристика называется **параметром сигнала.** Важными для практического использования примерами сигналов могут служить изменения амплитуды, фазы или частоты звуковых, а также электромагнитных колебаний. При этом амплитуда, фаза или частота являются возможными параметрами сигнала.

Дискретный сигнал Рис. 1.1.

Непрерывный и дискретный сигналы

Сигнал называется **непрерывным,** или **аналоговым,** если множество

значений его параметра бесконечно — точнее, более чем счетно. Например, звуковые и электромагнитные сигналы относятся к непрерывным, так как их параметры могут принимать любые значения из некоторого отрезка числовой оси. На рис. 1.1, *слева,* изображен пример непрерывного сигнала, параметр которого *A* может принимать любые значения из интервала E.

Сигнал называется **дискретным,** если его параметр может принимать лишь конечное число значений. На рис. 1.1, *справа*, изображен пример дискретного сигнала, параметр которого *A* может принимать только 10 различных значений, образующих конечное множество E. Эти значения изображены короткими горизонтальными штрихами на оси ординат. Дискретные сигналы используются в любых письменных текстах, в разнообразных устройствах — часах, информационных табло и т. д., в том числе в компьютерах. Отметим, что теоретически допускаются дискретные сигналы, имеющие счетное количество различных значений параметра, но на практике важны только сигналы с конечными наборами значений параметров.

Сообщения, основанные на непрерывных сигналах, называются непрерывными, а сообщения, основанные на дискретных сигналах, — дискретными. В то же время информация, в отличие от сообщений, не обладает ни свойством непрерывности, ни свойством дискретности.

Знаком называется элемент некоторого множества объектов, используемых для хранения или передачи дискретных сообщений. С обсуждавшейся ранее точки зрения, знак — это одно из возможных значений параметра дискретного сигнала. Знак вместе с его смыслом принято называть **символом.** Множество знаков, в котором определен линейный порядок, называется **алфавитом.** Для практических нужд человек использует множество различных алфавитов: алфавиты естественных языков, условные знаки на чертежах и топографических картах, знаки дорожного движения, нотные знаки и т. д. Для передачи или хранения сообщений иногда используются множества знаков, в которых отсутствует упорядоченность, например множество иероглифов китайского языка, набор знаков азбуки Брайля (специальной азбуки для восприятия текстов слепыми) и т. д.

Из теории кодирования информации известно, что при соблюдении некоторых условий сообщение, заданное в одном алфавите, можно без потери его смысла (без потери информации) задать в другом алфавите. Это создает возможность свести все разнообразие используемых человечеством алфавитов к одному, наиболее удобному в некотором смысле алфавиту. Теоретически и экспериментально было показано, что самым удобным и эффективным для представления сообщений в компьютерах является использование двоичных алфавитов, в которых фигурируют всего два различных знака.

Примеры двоичных алфавитов: {0, 1}, {true, false}, пара напряжений {1 В, 5 В} и т. д. Отметим, что используемые в двоичном алфавите {0, 1} знаки 0 и 1 часто называют также **двоичными цифрами.**

Когда говорят об обработке информации с помощью компьютеров, обычно в качестве термина, эквивалентного термину «информация» (следует понимать, что речь идет о «сообщениях»), используют термин «данные». Отметим, что план обработки данных — **алгоритм** — с общей точки зрения, *также является сообщением*, которое записано в некотором алфавите. А если этот алфавит и правила задания алгоритма воспринимаются вычислительной системой, тогда применяется термин «программа».

Обрабатываемую информацию, записанную в «понятной» компьютеру

форме, принято называть данными. План обработки данных — алгоритм, также записанный в специальной, «понятной» компьютеру форме, — принято называть программой.

Как следует из сказанного ранее, естественной, «понятной» для компьютера формой задания алгоритмов и обрабатываемых данных является их запись в двоичном алфавите {0, 1}. Переход к двоичному алфавиту в записи программ и данных принято называть **двоичным кодированием.** А сами программы и данные, записанные в этом алфавите, часто называют **двоичным кодом.** Поскольку двоичный код используется для хранения информации в компьютерах — вычислительных машинах, его называют также **машинным кодом.**

1.2. Понятие архитектуры компьютера

Термин «архитектура» в применении к компьютерам относительно нов. Почти до конца 70-х гг. применялись более узкие термины «устройство» или «структура» компьютера [20]. Но, как правило, после рассмотрения собственно устройства машины следовало изучение способов представления программ и данных в компьютере, особенностей различных устройств компьютера, организации обмена данными и т. д. Впоследствии всю эту совокупность сведений объединили под одним общим названием — **архитектура компьютера.**

Под архитектурой компьютера понимается совокупность взаимосвязанных сведений о способах представления в компьютере программ и данных, о назначении, структуре и особенностях функционирования отдельных его устройств, а также об организации и работе компьютера в целом.

Из сказанного следует, что основными функциями компьютера являются хранение, обработка, прием и передача данных. Для выполнения каждой из этих функций в компьютере предусмотрены специальные устройства:

- **память** — группа устройств, которые обеспечивают *хранение* программ **и** данных;
- **процессор** (от process — обработка) — одно или несколько устройств, которые обеспечивают задаваемую программой *обработку* данных;
- **устройства ввода/вывода** — группа устройств, которые обеспечивают обмен, то есть *прием и передачу* данных между пользователем и компьютером или между двумя или более компьютерами.

Различные устройства компьютера подсоединяют друг к другу с помощью стандартизированных и унифицированных (то есть единообразных) аппаратных средств — кабелей, разъемов и т. д. При этом устройства обмениваются друг с другом информацией и управляющими сигналами, которые также приводятся к некоторым стандартным формам. Совокупность этих стандартных средств **и** форм образует конкретный **интерфейс** (от interface — взаимный вид) того или иного устройства или программы.

Интерфейсом называется совокупность унифицированных стандартных соглашений, аппаратных и программных средств, методов и правил взаимодействия устройств или программ, а также устройств или программ **с пользователем.**

Изучение архитектуры компьютера естественно начать с обсуждения логического строения его памяти

1.3. Элементарные логические устройства памяти

Память компьютера имеет сложную многоуровневую структуру, реализованную в виде взаимодействующих устройств, которые могут использовать различные физические принципы для хранения данных. К этим устройствам относятся интегральные схемы, магнитные и оптические диски и т. д. Многоуровневый подход к реализации памяти вытекает из необходимости обеспечения эффективной работы компьютера при решении задач, точнее, при выполнении соответствующих им программ. Но в любом случае, при любой физической реализации памяти ее базовыми функциональными элементами являются **бит** и **байт.**

Элементарное устройство памяти компьютера, которое применяется для хранения одного из знаков двоичного алфавита, называется битом, или двоичным разрядом.

Термин «бит» произошел от английского bit, представляющего собой сокращение словосочетания Binary digiT — двоичная цифра (в некоторых источниках Binary elemenT — двоичный элемент). Способы физической реализации битов памяти в компьютерах обсуждаются в дальнейшем, а пока нас будут интересовать только возможности, точнее, функции бита.

- Бит может находиться только в одном из двух возможных устойчивых состояний, одно из которых принято считать изображением знака «О», а другое — изображением знака «1». Свое состояние бит сохраняет сколь угодно долго, пока оно не будет изменено принудительным способом. Следовательно, бит может хранить записанную в нем информацию.
- В любой момент времени можно узнать, в каком из двух состояний находится бит, — в состоянии «О» или в состоянии «1»; при этом текущее состояние бита останется неизменным. Другими словами, можно прочитать записанную в бит информацию (без ее потери).
- Всегда, когда в этом возникнет необходимость и вне зависимости от текущего состояния, можно перевести бит из одного состояния в другое. Иначе говоря, в бит можно записать новую информацию.

Итак, бит обеспечивает необходимую основу для реализации одной из важнейших функций компьютера — *хранения* данных.

Бит — это очень маленькая порция данных. Поэтому как для записи десятичных чисел используется несколько десятичных разрядов — разряд единиц, разряд десятков, сотен и т. д., так и для хранения двоичных машинных кодов используется несколько битов, совместно образующих устройство, которое принято называть **ячейкой памяти.** В общем случае ячейки различных компьютеров могут состоять из различного количества битов. Однако это создает значительные сложности для организации обмена данными между разными моделями компьютеров. Поэтому, начиная с машин третьего поколения, стандартными являются ячейки, состоящие из восьми битов.

Самостоятельный элемент памяти компьютера, состоящий из восьми битов, называется байтом.

Слово «байт» произошло от английского термина byte, представляющего собой сокращение словосочетания BinarY TErm — двоичный терм, двоичное выражение. Байт сохраняет все перечисленные выше свойства бита — он может хранить записанную в него информацию, ее можно прочитать, можно также записать в байт любую новую информацию.

Более подробно обсудим, что представляет собой содержимое байта. Каждый из восьми битов байта может содержать любую из двоичных цифр *независимо* от остальных. Следовательно, байт может содержать *произвольную комбинацию*, последовательность из восьми нулей или единиц, например последовательность 10110011. Такую последовательность называют также двоичным числом, или двоичным кодом. Количество различных кодов, различных комбинаций из восьми нулей и единиц, записываемых в один байт, равно $2^8 = 256$.

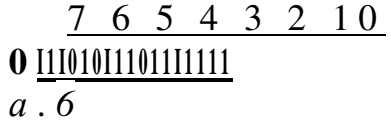
Запись находящегося в байте двоичного кода можно спутать с аналогичным по записи десятичным числом. Например, двоичный код 10110011 можно рассматривать и как «обычное» число «десять миллионов сто десять тысяч одиннадцать». В тех случаях, когда есть опасность спутать десятичное число и двоичный код, справа от двоичного кода записывают нижний индекс 2, а справа от десятичного числа указывают индекс 10. Таким образом, 10110011_2 — двоичный код, а 10110011_{10} — десятичное число. Для удобства восприятия десятичные числа в текстах на русском языке принято делить на группы по три цифры в каждой и отделять эти группы друг от друга пробелом — $10\ 110\ 011_{10}$. По аналогии с этим двоичные коды иногда также группируют, но по четыре цифры в группе — $1011\ 0011_2$.

Все байты одной из важнейших разновидностей памяти — оперативной — пронумерованы целыми числами. При этом номером начального байта всегда считается ноль. С помощью номера легко отличить один байт от другого, в любой момент можно «открыть» байт с нужным номером и посмотреть, какой код в нем находится, либо заменить его любым другим.

Байты, в отличие от отдельных битов, из которых они состоят, являются самостоятельными элементами памяти компьютера, так как именно с байтом в целом, а не с отдельным битом производятся такие операции, как запись и чтение.

Условно бит изображают в виде небольшого прямоугольника, содержащего либо цифру 0, либо цифру 1 (рис. 1.2, *а*), а байт рисуют в виде расположенных рядом восьми одинаковых прямоугольников, каждый из которых содержит какую-либо двоичную цифру. Биты, входящие в байт, принято нумеровать справа налево, начиная с нуля, — так, как показано на рис. 1.2, *б.*

Во многих случаях для записи некоторой неделимой порции информации, например для записи кода какого-либо числа восьми разрядов, одного байта не хватает. Тогда несколько соседних, смежных байтов объединяются в структуру, которую принято называть *полем.* Соседними считаются байты, номера которых образуют арифметическую прогрессию *n, n* + 1, *n* + 2,..., *n* + *k.* При этом значение выражения *k* + 1 считается длиной поля.



а . б

Рис. 1.2. Условные изображения: а — бита; *б* — байта

Группа смежных байтов памяти, которая используется для записи неделимой порции информации или для каких-либо других целей, называется полем. Количество байтов, из которых состоит поле, называется длиной поля.

1.4. Объем памяти

Важнейшей характеристикой любых устройств памяти компьютера является

объем. Объем памяти равен количеству байтов, из которых она состоит. Заметим, что объем памяти, который измеряется в байтах, не имеет ничего общего с физической величиной «объем тела», которая измеряется в литрах, кубических сантиметрах, кубических метрах и т. д. Это величина не кубической, а линейной природы. Она аналогична длине тела, измеряемой в сантиметрах, метрах и т. д. Впрочем, термин «объем» используется, когда говорят о памяти устройства в целом. Ранее установлено, что когда речь идет о характеристике некоторого участка (например, поля памяти), также используется термин «длина». Следовательно, *объем памяти* и *длина участка памяти* представляют собой одну и ту же характеристику — количество байт, из которых состоит обсуждаемый объект. В дальнейшем изложении эти термины используются как эквивалентные.

Байт является основной единицей измерения объема памяти. Вместе с тем байт как единица измерения представляет собой слишком маленькую величину. Поэтому для указания объемов памяти используется целый ряд кратных единиц, которые образуются с помощью так называемой **двоичной тысячи,** равной $2^{10} = 1024$.

Первая кратная единица называется **килобайт** (Кбайт, произносится «кабайт»): 1 Кбайт = 1024 байт. Собственно говоря, называть эту единицу килобайтом не совсем правильно, так как килобайт должен быть равным 1000 байт, а не 1024. Но обычно этой небольшой разницей пренебрегают. Следующая кратная единица — **мегабайт** (Мбайт): 1 Мбайт = 1024 Кбайт = 1 048 576 байт. Далее следуют **гигабайт** (Гбайт, около миллиарда байт) и **терабайт** (Тбайт, около триллиона байт). Можно упомянуть и еще более крупные единицы: **петабайт** (Пбайт), **эксабайт** (Эбайт), **зетабайт** (Збайт) и **йоттабайт** (Йбайт). Для наглядности значения кратных единиц сведены в табл. 1.1.

Таблица 1.1. **Кратные единицы объема памяти**

Единица	**Значение**	**Метрический**
1	**1024 байт (2^{10})**	**1000 (10^3)**
1	**1024 Кбайт = 1 048 576 байт (2^{20})**	**1 000 000**
1	**1024 Мбайт = 1 073 741 824 байт (2^{30})**	**ю^9**
1	**1024 Гбайт = 1 099 511 697 776 байт**	**ю^{12}**
1	**1024 Тбайт = 125 899 978 522 624 байт**	**ю^{15}**
1	**1024 Пбайт = 1 152 921 504 606 846 976**	**ю^{18}**
1	**1024 Эбайт = 1 180 591 620 717 411 303**	**ю^{21}**
1	**1021 Збайт = 1 208 925 819 614 629 174**	**ю^{24}**

Тема 2: Представление данных в компьютере

При компьютерной обработке информации приходится иметь дело с текстовыми, графическими, числовыми, звуковыми и другими данными. Для хранения данных различной природы применяются различные способы их представления в двоичном алфавите — различные способы кодировки. Кроме того, *для одной и той же* разновидности данных также могут использоваться *различные* способы кодировки, которые отличаются друг от друга эффективностью и различными требованиями к ресурсам компьютера.

Конкретный способ кодирования той или иной разновидности информации в компьютере принято называть *форматом данных.*

В общем *случае* термин «формат» понимается как строго определенный, исчерпывающе полный набор правил. Следовательно, в приведенном ранее определении речь идет об *исчерпывающем наборе правил кодирования* в компьютере той или иной разновидности данных.

2.1. Текстовые данные

При хранении в компьютере любой текст (документ, статья, книга) рассматривается как линейная последовательность символов. Причем промежуток между отдельными словами — пробел, переход на следующую строчку, переход на следующую страницу — также могут рассматриваться как некие специальные символы. Каждому символу из этой последовательности ставится в соответствие конкретный двоичный код, состоящий *ровно из восьми* двоичных разрядов. Таким образом, код каждого символа текста занимает *ровно один байт* памяти. Следовательно, текст целиком занимает столько байтов памяти компьютера, из скольких символов он состоит (включая все его символы — пробелы, знаки препинания, специальные знаки перехода на новую строчку, на новую страницу и т. д.).

Алфавит, используемый для представления текстов на естественном языке, должен содержать как минимум 52 латинские буквы (строчные и прописные), десятичные цифры, знаки препинания, математические знаки, специальные знаки и т. д., всего примерно 150 символов. Исходя из теоретических соображений, это требует при равномерном алфавитном двоичном кодировании для представления любого знака исходного алфавита $\log_2 150 \approx 7{,}2$, то есть восьми двоичных цифр [29]. Списки всех используемых при записи текстов символов и *однозначно* соответствующих им двоичных кодов образуют так называемые **кодовые таблицы.** В практике программирования применяются различные кодовые таблицы. Одной из наиболее часто используемых является кодовая таблица **ASCII** (American Standard Code for Information Interchange — американский стандартный код для обмена информацией). В этой таблице зафиксированы коды для 128 различных символов. Их список и соответствующие им восьмиразрядные (то есть состоящие из восьми двоичных цифр, разрядов) двоичные коды образуют основную (базовую) кодовую таблицу ASCII. Но один байт может содержать 256 различных восьмиразрядных кодов. Это означает, что в стандарте ASCII задействована только половина возможностей 8-битного кодирования. Имеются различные расширения основной кодовой таблицы ASCII, в которых задаются коды еще для 128 символов, в том числе и для символов различных национальных алфавитов. Фрагмент одного из расширений кодовой таблицы ASCII, включающий буквы русского алфавита —

кириллицы, приведен в табл. 2.1.

Таблица 2.1. **Фрагмент кодовой таблицы**

Символ	Двоичный	Символ	Двоичный	Символ	Двоичный	Символ	Двоичный
А	100000	И	100010	р	100100	ш	100110
Б	100000	й	100010	с	100100	щ	100110
В	100000	К	100010	Т	100100	ъ	100110
Г	100000	л	100010	У	100100	ы	100110
Д	100001	м	100011	ф	100101	ь	100111
Е	100001	н	100011	Х	100101	э	100111
Ж	100001	О	100011	ц	100101	ю	100111
З	100001	п	100011	ч	100101	я	100111

В качестве примера кодировки получим машинный код текста, состоящего из одного слова «КОМПЬЮТЕР». Этот текст состоит из 9 символов, следовательно, для его хранения требуется 9 байтов памяти. Используя табл. 2.1, для каждого символа легко получить соответствующий ему двоичный код. Остается только записать найденные коды в группу расположенных подряд байтов памяти. В табл. 2.2 приведен полученный таким образом машинный код этого текста. В первой строке таблицы указаны порядковые номера байтов памяти, в которых записан текст, во второй — символы, из которых текст состоит, а в третьей — машинные, двоичные коды символов. Таким образом, текст «КОМПЬЮТЕР» в вычислительной машине представлен двоичным кодом 1000 1010 1000 1110 1000 1100 1000 1111 1001 1100 1000 1110 1001 0010 1000 0101 1001 0000_2. Следует понимать, что пробелы между четверками двоичных цифр вставляются только для удобства их восприятия (чтения человеком), и в память компьютера они, естественно, не записываются.

Таблица 2.2. **Машинный код текста «КОМПЬЮТЕР»**

1	2	3	4	5	6	7	8	9
к	О	м	п	ь	ю	т	Е	Р

Обратите внимание на то, что в табл. 2.1 приведены коды прописных букв. Строчные буквы имеют другие коды. Например, код буквы «а» имеет вид 1010 0000_2, в то время как кол буквы «А» — 1000 0000_2. Не случайно рассматриваемое слово записано именно в таком виде — машинный код слова «КОМПЬЮТЕР» *отличается* от машинного кода слова «компьютер».

Если учесть все возможные буквы, входящие в национальные алфавиты различных стран, все возможные символы, которые встречаются **в** математических и других специальных текстах, то их общее количество окажется значительно больше 256 символов, кодировку которых обеспечивает один байт[1]. Поэтому было разработано несколько десятков различных кодовых таблиц. При этом в разных кодовых таблицах один и тот же код может соответствовать разным символам. Так, например, двоичный код 1000 1 010_2 соответствует символу «К» только в так называемой **ГОСТ-альтернативной** кодовой таблице. Именно ее фрагмент (она является одним из расширений базовой кодовой таблицы) приведен в табл. 2.1. А в другой популярной кодовой таблице, **Windows 1251,** этот же двоичный код служит для обозначения символа «Лэ». Следовательно, текст, записанный какой- либо программой в одной кодовой таблице, может быть полностью искажен при его

[1] **По некоторым оценкам, во всем мире в настоящее время используется около 200 000 различных символов.**

чтении с помощью другой программы, использующей другую кодовую таблицу. Если приведенный код слова «КОМПЬЮТЕР» попытаться прочитать с помощью программы, которая использует кодовую таблицу «Windows 1251», то он окажется соответствующим «слову» «Л>1)1Ы)н>}) ’...})».

Учитывая недостаточность возможностей обсуждавшихся кодовых таблиц, в последнее время все шире используется кодовая таблица с названием **UNICODE** (UNIversal CODE — универсальный код), в которой для кода одного символа отводится два байта, а не один, как в рассмотренных ранее таблицах. С помощью двух байтов, то есть 16 битов, можно закодировать уже $2^{16} = 65\,536$ различных символов. Такого количества кодов вполне достаточно, чтобы представить большинство из встречающихся во всевозможных текстах символов.

Если внимательно проанализировать действия, которые приходится выполнять над произвольными текстами, можно заметить, что любые их преобразования сводятся к замене одного символа текста другим, удалению символа из текста и вставке нового символа в выбранное место текста. В тех случаях, когда упомянутые действия по преобразованию текстов необходимо выполнить не над одним символом, а над некоторой их группой, соответствующие операции можно выполнить циклически несколько раз. Если включить в алфавит специальный «пустой» символ (не путать с пробелом!), который можно считать совпадающим с пустым текстом (то есть текстом, не содержащим ни одного символа), то окажется, что любые преобразования текстов сводятся к одной-единственной операции: замене одного символа алфавита другим. Естественно, для определения заменяемого или удаляемого символа либо места вставки нового символа нужно еще уметь сравнивать между собой любые два символа алфавита на совпадение или несовпадение.

Итак, основные элементарные действия, которые должен «уметь» выполнять компьютер над текстовыми данными, — это сравнение кодов двух символов текста и замена одного кода другим.

2.2. Графические данные

Для обсуждения общих принципов кодирования графических данных в качестве конкретного, достаточно общего случая графического объекта выберем изображение на экране телевизора или **дисплея** (от display — показывать, демонстрировать) компьютера. Это изображение состоит из некоторого количества горизонтальных линий — строк (рис. 2.1). А каждая строка, в свою очередь, состоит из элементарных мельчайших единиц изображения — точек, которые принято называть **пикселами** (от pixel — PICture’S ELement — элемент картинки). Весь массив элементарных единиц изображения называют **растром** (от лат. rastrum — грабли). Степень четкости изображения зависит от количества строк на весь экран и количества точек в строке, которые представляют **разрешающую способность** экрана, или просто **разрешение.** Чем больше строк и точек, тем четче и лучше изображение. Достаточно хорошим считается, например, разрешение 800 х 600, то есть 800 точек на строку и 600 строчек на экран.

Строки, из которых состоит изображение, можно просматривать сверху вниз друг за другом, как бы составив из них одну сплошную линию. После полного просмотра первой строки просматривается вторая, за ней третья, потом четвертая и т. д. до последней строки экрана. Этот процесс очень похож на принятый в большинстве стран мира способ чтения текстов, когда строчки просматриваются друг за другом слева направо и сверху вниз. Такой способ работы с изображением называется

строчной разверткой, или **сканированием** (от scan — бегло просматривать, развертывать изображение). Так как каждая из строк представляет собой последовательность пикселов, то все изображение, вытянутое в линию, также можно считать линейной последовательностью элементарных точек. В рассматриваемом случае эта последовательность состоит из 800 х 600 = 480 000 пикселов.

Рис 2.1. Структура растрового изображения

Вначале обсудим принципы кодирования монохромного изображения, то есть изображения, состоящего из любых двух контрастных цветов — черного и белого, зеленого и белого, коричневого и белого и т. д. Для определенности будем считать, что один из цветов — черный, а второй — белый. Тогда каждый пиксел изображения может иметь либо черный, либо белый цвет. Поставив в соответствие черному цвету двоичный код 0, а белому — код 1 (либо наоборот), мы сможем закодировать в одном бите состояние одного пиксела монохромного изображения. Так как 1 байт состоит из 8 битов, то на строчку, состоящую из 800 пикселов, потребуется 100 байтов памяти (рис. 2.2), а на все изображение — 60 000 байтов.

Восстановление изображения по известному его коду может осуществляться, например, с помощью луча электронно-лучевой трубки телевизора или монитора компьютера, который в процессе сканирования строк экрана под управлением специальной декодирующей схемы создает соответствующий коду цвет пиксела (белый или черный).

Однако полученное таким образом изображение будет чрезмерно контрастным. Реальное черно-белое изображение состоит не только из белого и черного цветов. В него входит множество различных промежуточных оттенков — серый, светло-серый, темно-серый и т. д. Если кроме белого и черного цветов использовать только две дополнительные градации, скажем, светло-серый и темно-серый цвета, то чтобы закодировать цветовое состояние одного пиксела, потребуется уже *два* бита. При этом кодировка может быть, например, такой: черный цвет — 00_2, темно-серый — 01_2, светло-серый — 10_2, белый — 11_2

Общепринятым на сегодняшний день и дающим достаточно реалистичные монохромные изображения считается кодирование состояния одного пиксела с помощью восьми битов, то есть одного байта, которое позволяет передавать 256 различных оттенков серого цвета, от полностью белого до полностью черного. В этом случае для передачи всего растра из 800 х 600 пикселов потребуется уже не 60 000, а 480 000 байтов.

Цветное изображение может формироваться различными способами. Один из них — метод **RGB** (Red, Green, Blue — красный, зеленый, синий), который опирается на то, что глаз человека воспринимает все цвета как сумму трех основных — красного, зеленого и синего. Например, сиреневый цвет — это сумма красного и синего, желтый цвет — сумма красного и зеленого и т. д. Для получения цветного пиксела в одну и ту же точку экрана направляется не один белый, а сразу три цветных луча. Опять упрощая ситуацию, будем считать, что для кодирования каждого из цветов достаточно одного бита, при этом нуль в бите означает, что в суммарном цвете данный основной отсутствует, а единица — присутствует.

Следовательно, для кодирования одного цветного пиксела потребуется три бита — по одному на каждый цвет. Пусть первый бит соответствует красному цвету, второй зеленому, а третий — синему. Тогда код 101_2 обозначает сиреневый цвет (красный есть, зеленого нет, синий есть), а код 110_2 — желтый цвет (красный есть, зеленый есть, синего нет). При такой схеме кодирования каждый пиксел может быть окрашен в один из восьми возможных цветов. Если каждый из цветов кодировать с помощью одного байта, как это принято для реалистического монохромного изображения, то появится возможность передавать по 256 оттенков каждого из основных цветов. Всего в этом случае обеспечивается передача 256 х 256 х 256 = 16 777 216 различных цветов, что довольно близко к реальной чувствительности

человеческого глаза. Таким образом, при данной схеме кодирования цвета на изображение одного пиксела требуется три байта, или 24 бита памяти. Этот способ представления цветной графики принято называть режимом **True Color** (true color — точный цвет), или **полноцветным** режимом. Отметим, что полноцветный режим требует очень много памяти. Так, для обсуждавшегося ранее растра 800 x 600 при использовании метода RGB требуется

1 440 000 байт « 1,37 Мбайт. В целях экономии памяти разрабатываются различные режимы и графические форматы, которые немного хуже передают цвет или само изображение, но требуют гораздо меньше памяти.

Большинство действий, которые приходится выполнять над графическими данными в пиксельном формате, сводятся к замене текущего цвета пиксела другим цветом. Например, стирание какого-либо участка изображения — это не что иное как замена цветов всех пикселов стираемого участка цветом фона рисунка. Следовательно, и для графических данных элементарные действия, которые должен «уметь» выполнять компьютер, сводятся к сравнению двух двоичных кодов и замене одного кода другим.

Тема 3: Числовые данные

2.3.1. Форматы представления чисел в компьютере

Напомним, что исчерпывающе полный набор правил кодирования той или иной разновидности информации в компьютере принято называть *форматом* данных. Для представления числовых данных в компьютерах используются два принципиально разных формата: формат **с фиксированной точкой** (запятой) и формат **с плавающей точкой** (запятой).

В названиях форматов речь идет о знаке, с помощью которого целая часть числа отделяется от его дробной части. В обычной практике записи чисел для этого используется запятая, а в программировании целая часть числа отделяется от дробной точкой. Поэтому в литературе в зависимости от предпочтений авторов используется как термин «фиксированная точка», так и термин «фиксированная запятая». Это же относится и ко второму формату.

Формат с фиксированной точкой предназначен для *абсолютно тачного* представления *целых* чисел. В программировании эти числа относятся *к целому типу*, в то время как формат с плавающей точкой используется для представления только нецелых, *приближенных* чисел. В программировании такие числа относятся к *вещественному* типу. Напомним, что вещественные числа возникают в задачах в результате различных измерений (например, измерений веса тела или его длины), которые, как известно, всегда выполняются с некоторой погрешностью, приближенно.

Как выяснится немного позже, возможности одного байта для кодирования чисел довольно малы, поэтому числа обычно занимают несколько соседних байтов, то есть поле, длина которого зависит от используемого формата.

2.3.2. Форматы целых чисел

Существуют две модификации формата с фиксированной точкой, которые принято называть его **беззнаковым** и **знаковым представлениями.** Беззнаковое представление формата используется для работы с целыми *неотрицательными* числами, а существующее в нескольких вариантах знаковое — для работы как с положительными, так и с отрицательными целыми числами.

Беззнаковое представление формата с фиксированной очкой

В беззнаковом представлении целого числа используется *прямой двоичный коду* который представляет собой запись этого числа в двоичной системе счисления. При этом все разряды занятого числом поля содержат его значащие цифры. Точка, отделяющая целую часть числа от дробной, считается расположенной, *фиксированной справа от крайнего правого разряда.* Следовательно, под дробную часть числа отводится *нулевое* количество разрядов, и в данном варианте кодировки возможна работа только с целыми числами. Постоянное расположение, фиксация позиции точки дала название формату — с фиксированной точкой.

Пусть N — длина используемого поля в битах, тогда в нем может быть записано ЛГ-разрядное двоичное число, и следовательнр, в этом поле могут быть представлены коды целых чисел *x,* максимальное из которых равно 2^-1. Пусть далее $Z°_N = \{x$ е $Z|0 < x < 2^N - 1\}$, где Z — *математическое* множество целых чисел. Тогда формально можно записать, что беззнаковым кодом могут быть представлены только числа x е $Z°_N$.

Для кодирования чисел в формате с фиксированной точкой используются поля длиной 1, 2 или 4 байта, поэтому N может быть равно 8, 16 или 32. В табл. 2.3 приведены обычно используемые в программировании названия соответствующих

этим полям целых типов и диапазоны их возможных значений.

На практике иногда возникают задачи определения машинного кода заданного числа, а также определения числа по его коду. При использовании беззнакового представления формата с фиксированной точкой эти задачи решаются довольно просто.

Получение беззнакового кода целого числа. Чтобы получить машинный код целого неотрицательного числа в беззнаковом представлении формата с фиксированной точкой, следует перевести заданное число в двоичную систему счисления и записать полученный код в выделенное для него поле.

Пусть, например, задано число 77_{10}. Переводя его в двоичную систему счисления, получим 1001101_2. Искомый код в однобайтном поле имеет вид $0100\ 1101_2$, в двухбайтном — $0000\ 0000\ 0100\ 1101_2$, а в четырехбайтном — $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 1101_2$.

Видно, что запись машинных кодов непосредственно в двоичном виде очень громоздкая. Гораздо компактнее выглядят те же самые машинные коды, но в шестнадцатеричном виде: $4D_{16}$, $00\ 4D_{16}$ и $00\ 00\ 00\ 4D_{16}$. Поэтому шестнадцатеричное представление используется для отображения содержимого полей памяти гораздо чаще, чем двоичное. При этом двоичные коды группируют по четыре цифры, а шестнадцатеричные — по две. Таким образом, содержимое одного байта изображается *двумя группами из четырех двоичных цифр* или *одной группой из двух шестнадцатеричных цифр*.

Отметим, что выбор длины поля, которое отводится под запись кода числа, зависит от ряда обстоятельств, и не всегда возможно выбрать поле любой допустимой длины, как в рассмотренном примере. При выборе длины поля самым важным фактором является значение числа, код которого должен быть записан в это поле. Из табл. 2.3 следует, что, например, для числа 155_{10}, имеющего код $9B_{16}$, при использовании обсуждаемого способа можно выбрать поле любой возможной длины. А, скажем, код числа 497_{10} в однобайтовое поле не поместится, так как оно больше, чем максимально допустимое для одного байта число 255_{10}. Число 497_{10} имеет беззнаковый код $1F1_{16}$, и для его записи могут быть использованы только двух- или четырехбайтовые поля. А если рассматривается число, большее чем $65\ 535_{10}$, например $567\ 398\ Ю4_{10}$, то для его кода $21\ D1\ CE\ D8_{16}$ потребуется уже четырехбайтовое поле. Из этой таблицы также следует, что числа большие, чем $4\ 294\ 967\ 295_{10}$, вообще не могут быть закодированы таким способом.

Определение значения числа по его беззнаковому коду. Чтобы определить значение числа, код которого задан в беззнаковом представлении формата с фиксированной точкой, достаточно, считая весь этот код двоичным (или шестнадцатеричным) числом, перевести его в десятичную систему счисления.

Пусть, например, задан занимающий два байта памяти машинный код $010B_{16}$ некоторого числа в обсуждаемом формате. Переходя в десятичную систему счисления, получим, что это код числа 267_{10}.

Знаковые представления формата с фиксированной очкой

В истории развития архитектуры компьютеров использовались четыре различных варианта представления знаковых чисел:

- система со знаком;
- обратный код, поразрядное дополнение или код с дополнением до единицы;
- дополнительный, комплементарный код или код с дополнением до двух;
- система со смещением.

В настоящее время первые две системы устарели и практически вышли из употребления. Тем не менее по ходу изложения материала мы затронем и эти устаревшие системы.

Очевидно, что для машинного представления знаковых чисел нужно определенным образом закодировать знак числа и его *модуль.* Так как существует всего два знака чисел, «+» и «-», то самое простое напрашивающееся решение состоит в том, чтобы представить код знака одной двоичной цифрой и выделить под него один из разрядов поля, а во всех остальных разрядах записывать *модуль* числа, кодируя его, например, с помощью *прямого двоичного кода*, так же как кодируются беззнаковые целые числа. Такой способ называется **системой кодирования со знаком.** Код знака числа принято размещать в *самом левом* разряде поля, который в связи с этим принято называть **знаковым битом.** По традиции знак «плюс» кодируется нулем, а знак «минус» — единицей. Таким образом, если в знаковом бите находится *нуль*, это означает, что остальные биты поля содержат модуль *положительного* числа, а если знаковый бит занят *единицей*, то в них находится модуль *отрицательного* числа. Заметим, что можно было бы договориться и о другом способе кодирования знака, но по ряду причин выбран именно этот способ.

На рис. 2.4 показано размещение знаковых битов в полях разной длины. В однобайтовом поле разряды с 0-го по 6-й содержат код модуля числа, а код знака занимает 7-й разряд. В двухбайтовом поле код модуля занимает разряды с 0-го по 14-й, код знака — 15-й разряд. В четырехбайтовом поле код модуля расположен в разрядах с 0-го по 30-й, а код знака находится в 31-м разряде.

Значащие цифры модуля

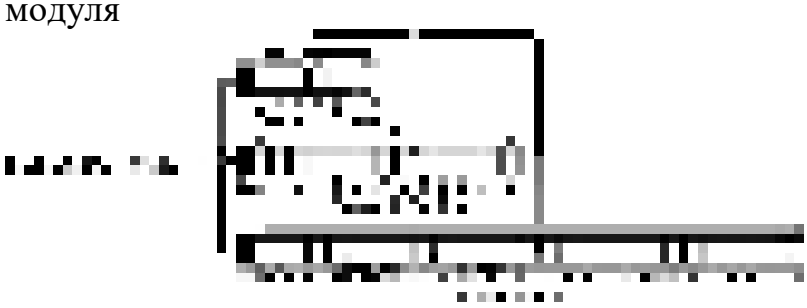


Рис. 2.4. Разрядная сетка системы кодирования со знаком

Обратите внимание на то, что в знаковом представлении различные биты кода играют различную роль, в то время как во всех рассматривавшихся ранее вариантах кодирования все цифры кода играют одну и ту же роль, например образуют код отдельного символа текста или являются значащими цифрами беззнакового кода целого числа. Закрепление за разрядами поля конкретных функций хранения различных элементов кода принято называть **разрядной сеткой.**

Итак, в рассматриваемом варианте кодирования для представления *модулей* целых чисел предлагается использовать *прямой двоичный код числа.* Возьмем, например, числа $+4_{10}$ и $^{-}4_{10}$ с прямым двоичным кодом модуля 100_2. Тогда однобайтовый код числа $+4_{10}$ есть $0|000\ 0100_2$, или 04_{16}, а такой же код числа -4,о — это $1|000\ 0100_2$, или 84_{16}. Еще пример: число $+127_{10}$ имеет в этой системе код $0|\ 111\ 1111_2$ ($7F_{16}$), а число -127_{10} — код $1|111\ 1111_2$ (FF_{16}). В приведенных двоичных кодах для наглядности знаковый бит отделен от битов модуля вертикальной чертой, которая, естественно, в реальных машинных кодах отсутствует.

Поскольку один из *N* битов поля отводится под код знака, под запись кода модуля остается *N* - 1 битов. Следовательно, в таком поле могут быть закодированы целые числа, модули которых удовлетворяют условию $|x|<2^{N-1}-1$. Пусть $Z^X_N = \{x$ е

Z| 1 - 2^{N}~^{x} < x < 2^{n}~^{1} - 1}. В системе кодирования со знаком могут быть представлены только числа x е Zjy.

Посмотрим теперь, к каким последствиям приведет принятие предлагаемого варианта кодирования знаковых целых чисел. Во-первых, оказывается, что числу $_{10}$ соответствуют два различных кода: код 0000 0000_2 ($+0_{10}$) и код 1000 0000_2 (-0_{10}). Такая неоднозначность крайне нежелательна, так как необходимо либо дополнительно учитывать ее при различных проверках аппаратными средствами компьютера, либо предусматривать отдельные проверки в программах. Во-вторых, и это самое главное, возникает специальная арифметика с совершенно непривычными правилами выполнения самых обычных действий. Так, по правилам обычной арифметики, сложение чисел $+4_{10}$ и -4_{10} дает в результате 0_{10}. А теперь выполним сложение для полученных ранее кодов этих чисел: 00000100_2 + 10000100_2 = 10001000_2. Как видим, получен совершенно неожиданный результат: вместо ожидавшегося кода числа 0_{10} сложение дало код числа -8_{10}. Этот результат является следствием неудачного выбора способа кодирования. Его нужно выбирать исходя из *логики использования кода*, а не из «напрашивающегося» или «очевидного» на первый взгляд подхода. Основное требование при выборе системы кодирования чисел состоит в том, что полученный код должен, удовлетворять правилам выполнения сложения и вычитания в двоичной системе счисления. В связи с этим код каждого следующего *положительного* числа должен получаться *прибавлением* единицы к коду текущего числа, а код каждого следующего *отрицательного* числа должен получаться *вычитанием* единицы из кода текущего числа. Построенный таким образом код принято называть **дополнительным, с дополнением до двух** или **комплементарным** (от complementary — дополняющий).

Для простоты рассуждений рассмотрим построение этого кода на примере четырех битов. Подчеркнем, что при построении кодов знаковых чисел знаковый бит ни при каких условиях не может быть занят значащими цифрами кода модуля, в него не должен выполняться перенос и в нем нельзя производить заем. Итак, код числа 0_{1O} — это 0000_2. Код следующего положительного числа $+1_{10}$ получаем прибавлением к нему единицы: 0001_2. Аналогичным образом получаем код числа $+2_{10}$: 0010_2. Дальнейший процесс получения кодов положительных чисел изображен на рис. 2.5, *слева.* Отметим, что при записи кода в четыре бита этот процесс можно продолжить только до числа $+7_{10}$, имеющего код 0111_2. Переход к следующему положительному числу $+8_{10}$ с кодом 1000_2 приведет к занятию знакового бита старшей цифрой кода модуля. Следовательно, числа, большие $+7_{10}$, таким способом изобразить нельзя.

Рассмотрим теперь получение кодов отрицательных чисел. Для получения кода числа -1_{10} вычтем из кода числа 0_{10} единицу: 00 00_2 - 1_2. Это возможно только если условиться о том, что необходимый для такого вычитания заем выполняет- ся из воображаемого дополнительного разряда: 10000_2 - 1_2 = 1111_2. Дальнейшее получение кодов отрицательных чисел не вызывает затруднений и показано на рис. 2.5, *справа.* Например, код числа -2_{10} получается так: 1111_2 - 1_2 = 1110_2. Очевидно, что получение кодов чисел, меньших, чем -8_{10}, в случае использования для кода всего четырех битов невозможно, так как, например, для получения кода ч!'сла -9_{10} нужно выполнить заем из знакового бита: 1000_2 - 1_2 = 0111_2. К тому же, получающийся при этом код 0111_2 уже используется как код числа $+7_{10}$.

Отметим, что название «дополнительный» код возникло потому, что в качестве кода отрицательного числа фактически выбирается *D* — дополнение я-разрядного прямого кода модуля *P* этого числа до числа *M = 2n: D = M - P.* Вновь возьмем *n* = 4 и найдем код числа -4. В данном случае $M = 10000_2$, прямой четырехразрядный код модуля обсуждаемого числа $P = 0100_2$, и, следовательно, его дополнение до *M* равно $D = 10000_2 - 0100_2 = 1100_2$, что совпадает с полученным прямыми рассуждениями дополнительным кодом.

Способ кодирования знаковых чисел, основанный на использовании *дополнительного* кода, устраняет все отмеченные ранее недостатки применения системы кодирования со знаком. Во-первых, необходимые арифметические свойства удовлетворяются автоматически по способу построения кода. Например, при сложении кодов 0100_2 ($+4_{10}$) и 1100_2 (-4_{10}) получается код 10000_2, старшая единица которого не помещается в используемые четыре разряда и *отбрасывается.* Таким образом, остается код 0000_2, который в точности соответствует нужному результату. Отметим, что это общий технический прием при выполнении действий со знаковыми числами. Он аналогичен приему, использованному в рассуждениях при получении дополнительного кода числа -1. Можно считать, что во время выполнения аналогичных операций возвращается заем, выполненный ранее из *воображаемого дополнительного разряда.*

Во-вторых, исчезла неоднозначность кодирования нуля. В самом деле, код 1000_2, который раньше вместе с кодом 0000_2 использовался для кодирования нуля, оказался закрепленным за кодом максимального по модулю отрицательного числа -8_{10}. Это привело к тому, что диапазоны представления положительных и отрицательных чисел стали несимметричными. В самом деле, если для записи кода используется только 4 бита, то при выборе *любого* способа кодирования возможно формирование всего $2^4 = 16$ различных кодов. Из них при применении системы со знаком 14 кодов закреплено за ненулевыми числами из симметричного диапазона от -7_{10} до $+7_{10}$, и еще два кода, 0000_2 и 1000_2, соответствуют нулю. Применение дополнительного кода позволяет изобразить шестнадцать целых чисел (вместе с нулем) из несимметричного диапазона от $\sim 8_{10}$ до $+7_{10}$. При этом под нуль занят всего один код 0000_2, а закрепление второго кода 1000_2 за числом -8_{10} как раз и приводит к появлению несимметричности диапазона.

Приведенные рассуждения не зависят от длины поля. Поэтому при использовании дополнительных кодов для полей длиной Л^бит в соотношении $\backslash -2^{N} \sim^{x} < x < 2^{N} \sim^{X}$ -1, описывающем диапазон кодируемых чисел в системе со знаком, левую границу нужно сместить на единицу в *отрицательную* сторону. Следовательно, при использовании *дополнительных* кодов в поле длиной *N* бит можно закодировать числа x е Z^{2}_{n}, где $Ъ\backslash = \{x$ е $Z| - 2^{N} \sim^{l} < x < 2^{ЛМ} - 1\}$

При сравнении диапазонов чисел из табл. 2.3 и 2.4 для беззнакового и знакового представлений может показаться, что диапазон изображаемых чисел в случае знакового представления уже, чем в случае беззнакового варианта. Однако это не так, поскольку общее количество различных кодов зависит только от длины используемого поля и никак не связано с используемым форматом.

Способы получения дополнительных кодов целых чисел. Понятно, что для практического применения рассмотренный способ получения дополнительного кода малопригоден. Поэтому рассмотрим два простых способа получения дополнительного кода *по известному прямому коду.* Первый способ можно применять,

когда используется двоичная система счисления.

1. Прямой код записывается в выбранное поле длиной 1, 2 или 4 байта.
2. Прямой код инвертируется (обращается), то есть каждая цифра 0 кода заменяется цифрой 1, и наоборот, каждая цифра 1 кода заменяется цифрой 0. Заметим, что полученный таким образом код называется **обратным, с дополнением до единицы** или **с поразрядным дополнением. К** обратному коду прибавляется единица.

Пусть, например, имеется прямой код $100\ 1101_2$ и выбрано однобайтовое поле. Тогда исходным будет код в виде $0100\ 1101_2$, его обратный код — $1011\ 0010_2$, а дополнительный — 1011 00 И $_2$.

Важным моментом в получении правильного конечного результата является первоначальная запись кода в поле подходящей длины. Возьмем, например, прямой код 100_2. Его инвертирование дает обратный код 011_2, а последующее добавление единицы — дополнительный 100_2. Совпадение прямого и дополнительного кодов свидетельствует об ошибке, так как в данном случае однозначность кодирования является одним из основных предъявляемых к коду требований. Ошибка состоит в том, что сделана попытка работать с несоответствующим значению числа количеством битов. В самом деле, прямой код 100_2 числа 4 занимает $N = 3$ бита. Но, как следует из вышеизложенного, обсуждаемым способом в этом количестве битов можно представить коды чисел только из диапазона от -4 до +3. Для получения правильного результата необходимо взять хотя бы 4 бита. Тогда получится правильный результат: прямой код 0100_2, обратный - 1011_2, а дополнительный — 1100_2. Итак, при выборе длины поля для заииси кода числа необходимо учитывать приведенные в табл. 2.4 соответствия между длиной поля и диапазоном представимых в нем чисел.

Отметим, что если используется прямой код числа, модуль которого принадлежит допустимому для данного поля диапазону, то в знаковом (крайнем слева) бите *обязательно* находится 0 (этого не было у кода 100_2!). Тогда переход к дополнительному коду автоматически приведет к формированию в знаковом бите нужной единицы.

Второй способ удобно применять и для двоичных, и для шестнадцатеричных кодов. Для получения дополнительного кода нужно прямой код вычесть из числа $M = 2W$ для двоичной системы или 16W для шестнадцатеричной, где N — длина используемого поля в байтах. Так, для однобайтовых $(N = 1)$ полей и двоичного кода вычитать прямой код следует из числа $2^8 = 1\ 0000\ 0000_2$. А для того же поля и шестнадцатеричного кода вычитание должно выполняться из числа $16^2 = 100_{16}$. Пусть, например, прямой код равен $100\ 1101_2$, его вычитание из $1\ 0000\ 0000_2$ дает в результате $1011\ 0011_2$. Очевидно, что, работая в шестнадцатеричной системе счисления, можно получить тот же результат, но в значительно более компактном виде: вычитая прямой код $4D_{16}$ из 100_{16}, получаем тот же дополнительный код в виде $B3_{16}$.

Получение знакового кода заданного числа. Чтобы получить машинный код целого числа в знаковом представлении формата с фиксированной точкой, следует:

1) перевести модуль этого числа в двоичную систему счисления;
2) по табл. 2.4 выбрать поле, длина которого обеспечивает возможность записи знакового кода рассматриваемого числа;
3) если число положительное, записать полученный код в выбранное поле.
4) если число отрицательное, получить в выбранном поле его дополнительный код.

Пусть, например, задано число $+77_{10}$. Перевод модуля этого числа в двоичную

систему счисления дает 77_{10} —» 1001101_2. Так как рассматривается положительное число, то осталось записать его код в выбранное поле. По табл. 2.4 находим, что для этого числа может быть использовано поле любой длины. Отсюда для однобайтового поля получаем: 0100 1101_2, для двухбайтового — 0000 0000 0100 1101_2, и для четырехбайтового — 0000 0000 0000 0000 0000 0000 0100 1101_2.

Попутно заметим, что, если число $x > 0$ принадлежит множеству Ъ\, то оно принадлежит и множеству Z, то есть и знаковый, и беззнаковый коды этого числа могут

быть записаны в поле выбранной длины *N*. В этом случае его *знаковый* и *беззнаковый* коды в выбранном поле *совпадают.* Это видно хотя бы на примере рассматривавшихся ранее кодов числа $+77_{10}$. А теперь возьмем, скажем, $x = +155_{10}$, прямой код модуля которого 1001 1011_2 или $9B_{16}$. Ясно, что для однобайтового поля x *G* Zg, но, с другой стороны, x е Zg. Это значит, что беззнаковый код числа может быть записан в однобайтовое поле, а знаковый — нет. То, что код 10011011_2 не может быть знаковым однобайтовым кодом положительного числа, видно и по занятому единицей знаковому биту. В этом случае, как известно, в поле находится код отрицательного числа. Как будет показано далее, знаковый код $9B_{16}$ соответствует числу -101_{10}, в то время как знаковые коды $009B_{16}$ и $00009B_{16}$ представляют число $+155_{10}$. С другой стороны все коды $9B_{16}$, $009B_{16}$ и $00009B_{16}$, рассматриваемые как беззнаковые, соответствуют числу $+155_{10}$. Аналогичная ситуация наблюдается для всех неотрицательных чисел, которые для поля выбранной длины *N* принадлежат $Z°_N$ и не принадлежат Z^2_n: \$X$ G$ Z^ л(X е Z^).

Теперь получим знаковое представление числа -77. Прямой код модуля этого числа уже найден, это 1001 101_2. Переходя любым из способов к дополнительному коду, для однобайтового поля получим 1011 0011_2 или $B3_{16}$. В случае использования двух- и четырехбайтовых полей работать с двоичными кодами неудобно. Поэтому перейдем к шестнадцатеричным кодам. Итак, для однобайтового поля прямой код модуля числа -77 равен $4D_{16}$, и тогда искомый дополнительный код есть $100_{16} - 4B_{16} = B3_{16}$. Для двухбайтового поля находим: $10000_{16} - 4D_{16} = FFB3_{16}$,

и,наконец, для четырехбайтового — $100000000_{16} - 4D_{16} = FFFFFFB3_{16}$.

В связи с тем, что для записи кодов знаковых чисел могут быть использованы поля различной длины, на практике возникает задача: по имеющемуся коду числа в поле некоторой длины найти его код в поле большей или меньшей длины. На рис. 2.7 показаны знаковые представления для чисел +77 и -77 в полях разной длины. В двоичных кодах вертикальной чертой отделен знаковый бит поля. Сравнение этих результатов показывает, что при переходе к полю большей длины свободные слева биты поля заполняются кодом знакового бита 0_2 (0_{16}) для положительных и 1_2 (F_{16}) — для отрицательных чисел. Это правило называется *правилом размножения знака.* Переход к полям меньшей длины в тех случаях, когда он возможен, осуществляется отбрасыванием двух, четырех или шести цифр 0_{16} или F_{16} или же соответствующего количества тетрад 0000_2 или 1111_2.

Коды числа **$+77_{1в}$** в стандартных полях

Определение значения числа по его знаковому коду. Чтобы определить значение числа, для которого задан его машинный код в знаковом представлении формата с фиксированной точкой, нужно сделать следующее.

1. По знаковому биту определить знак числа. Для отрицательных чисел в двоичной кодировке крайняя слева цифра равна 1_2, а в шестнадцатеричной эта цифра должна быть больше 7_{16}. В противном случае поле содержит прямой код положительного

числа.

2. Если число отрицательное, то в поле находится дополнительный код, от которого по формуле $P = M - D$ следует перейти к прямому коду.

3. Перевести число из двоичной или шестнадцатеричной системы счисления в десятичную.

4. Приписать результату соответствующий знак числа.

Пусть, например, задан код $005A_{16}$ = 0000 0000 0101 1010_2 и известно, что это знаковое представление формата с фиксированной точкой. По первой цифре шестнадцатеричного кода 0_{16} или знаковому биту 0_2 определяем, что в поле находится код положительного числа, перевод которого в десятичную систему счисления дает $+90_{10}$. Еще пример. Пусть в том же формате задан код $FFE8_{16}$ или 1111 1111 1110 1000_2. Первая цифра кода $F_{16} > 8_{16}$ или равный 1_2 знаковый бит являются признаками отрицательного числа. Переход к прямому коду дает 10000_{16} - $FFE8_{16}$ = 18_{16} или 24_{10}. Окончательно находим, что рассматриваемый код $FFE8_{16}$ соответствует числу -24_{10}: Если же код $FFE8_{16}$ рассматривать как беззнаковое представление числа, то он соответствует числу 65 512_{10}.

Один и тот же код в разных представлениях и, тем более, разных форматах может соответствовать различным числам.

В заключение кратко обсудим систему кодирования **со смещением,** которая в англоязычной литературе называется **системой excess 2N - 1.** Эта система кодирования характеризуется целым положительным числом — константой смещения *K*, которое прибавляется к любому кодируемому числу *x* с тем, чтобы сумма исходного числа и константы смещения $x + K$ попала в диапазон допустимых для поля выбранной длины *N* беззнаковых чисел $Z°_N$: $x + K$ е $Z°_N$. В качестве *кода исходного числа* выбирается *беззнаковый код суммы.* Таким образом, множество кодируемых чисел в системе с константой смещения *K* есть $Z^* = \{x £ Z | - K < x < 2^N - K - 1\}$. Пусть, например, $N = 8$ и $K = 64$. Тогда Zjjf = $\{x$ е Z\- 64 < $x < 191\}$, а код, например, числа $x = -4_{10}$ определяется как беззнаковый код числа 60_{10}, равный

11 1100_2 или $3C_{16}$. В определенном смысле эта операция является обратной по отношению к изображенной на рис. 2.7: диапазон, включающий числа разного знака, с помощью константы *K* смещается как единое целое вдоль числовой оси в конечное положение, совпадающее с диапазоном беззнакового кодирования.

Если взять специальный случай, когда константа $K = 2N$, где *N* — длина поля в битах, то окажется, что система с таким смещением весьма близка к дополнительному коду. Возьмем, например $N = 8$, тогда $K = 128$ и $Zj^{\wedge 8} = \{x$ е Z\- 128 < $x < 127\}$. Как можно заметить, в данном случае диапазоны системы со смещением и дополнительного кода *совпадают.* Практически одинаковы и получаемые при этом коды чисел. Например, в качестве кода числа -4 в этой системе выбирается беззнаковый код числа 124_{10}, равный 0111 1100_2 или $7C_{16}$. Если сравнить полученный результат с *дополнительным* кодом 1111 1100_2 того же самого числа -4, то легко увидеть, что они различаются только в знаковом бите. Другими словами, код числа в системе со смещением с константой $K = 2N$ совпадет с *дополнительным* кодом, у которого инвертирован знаковый бит.

Как мы увидим в дальнейшем, в настоящее время система кодирования со смещением используется как один из элементов представления чисел в формате с плавающей точкой.

Тема 4: Логические основы обработки данных

После выяснения способов представления данных в памяти компьютера необходимо выяснить, как осуществляется их обработка. В данной главе обсуждаются некоторые вопросы, связанные с физическими и логическими основами обработки ***дискретных*** данных.

Понятие такта4.1. Основные устройства компьютера 38

В выполнении любых действий над данными участвуют несколько устройств компьютера. Совершенно очевидно, что действия, в которых участвуют несколько исполнителей (людей, устройств), должны быть согласованы и синхронизированы друг с другом. В этом смысле можно провести наглядную аналогию между компьютером и оркестром музыкальных инструментов. Чтобы оркестр мог исполнить какую-либо мелодию, музыканты должны действовать строго синхронно, несогласованная игра представляет собой какофонию, а не мелодию. За синхронизацию игры музыкантов в оркестре отвечает дирижер, который в определенном темпе делает взмахи дирижерской палочкой. В компьютере такую же роль играет специальное устройство — тактовый генератор, который через равные промежутки времени вырабатывает импульсы синхронизации — синхроимпульсы, служащие ориентирами во времени и используемые для координации всеми участвующими в выполнении действия устройствами (рис. 3.1). Длительности вырабатываемых импульсов также одинаковы.

Промежуток времени от начала одного импульса синхронизации до начала следующего за ним импульса называется тактом. Такты обладают равными длительностями с высокой степенью точности.

Выполнение процессором любых действий всегда происходит во время некоторой части такта, а за ней следует пауза, в течение которой ничего не происходит. Наличие такой паузы является принципиальным фактором, обусловленным физическими законами, которые управляют работой процессора. Аналогичным образом обстоят дела при забивании гвоздя молотком. Удары по шляпке гвоздя чередуются с периодами, во время которых молоток поднимается вверх для нанесения удара. Забить гвоздь без таких периодов невозможно. Хотя движение молотка происходит непрерывно, можно считать, что процесс забивания гвоздя в целом распадается на ряд дискретных действий — ударов, происходящих ***мгновенно*** в моменты соприкосновения молотка и шляпки гвоздя. Точно так же, обсуждая выполнение процессором определенной в программе последовательности действий, можно считать, что любые длящиеся внутри такта действия происходят ***мгновенно*** в моменты времени t_{Oy} t_u t_2..., соответствующие границам тактов. Строго говоря, синхроимпульсы, которые считаются границами тактов, также имеют отличную от нуля длительность, поэтому моменты времени t_0, t_u t_2... следует привязывать к началам тактовых импульсов.

С понятием такта связана ***тактовая частота*** — одна из важнейших технических характеристик различных устройств компьютера.

Тактовая частота представляет собой техническую характеристику отдельных устройств компьютера, которая равна количеству тактов, управляющих работой устройства, в единицу времени. Единицей измерения тактовой частоты является герц, равный одному такту в секунду.

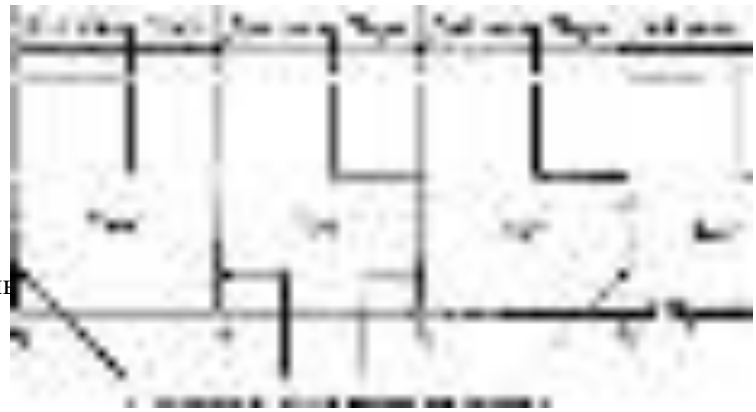


Рис. 3.1. К понятию такта

Тактовая частота является одним из главных факторов, определяющих скорость обработки данных компьютером. Первые персональные компьютеры работали на частотах 5-10 МГц. В настоящее время процессоры компьютеров работают с тактовыми частотами от сотен мегагерц до 3-4 ГГц. Возможно, в недалеком будущем появятся процессоры с тактовой частотой порядка 10 ГГц. Однако следует заметить, что, согласно теоретическим оценкам, процессоры, выполненные по современным технологиям, не смогут превзойти частот 30-40 ГГц.

Как известно, программа в общем случае представляет собой некоторый текст, определяющий последовательность действий по обработке данных. Такой текст, так же как и обрабатываемые данные, кодируется в компьютере с помощью двоичного кода. То есть программы, как и данные, представлены в компьютере в дискретном виде.

Для обеспечения задаваемого программой порядка обработки данных в компьютере формируются и используются различные вспомогательные значения и признаки, так или иначе характеризующие результаты обработки и используемые для определения дальнейшего порядка выполнения действий. Так, например, при вычислении любых числовых значений формируется признак нулевого результата. Этот признак имеет значение 1, если в результате вычислений получается 0, и значение 0, если результат ненулевой. Анализ этого признака, например, при вычислении знаменателя дроби, позволяет избежать бессмысленного деления на нуль. Упомянутые значения и признаки, так же как и выполняемая программа в целом, представлены в компьютере двоичным кодом, то есть являются дискретными.

> Совокупность, состоящая из двоичного кода выполняемой программы, значений обрабатываемых дискретных данных, а также набора дискретных значений и признаков, которые используются для управления процессом обработки, образует внутреннее состояние компьютера.

С точки зрения введенного понятия внутреннего состояния компьютера, можно считать, что в моменты времени t_0, t_u t_2... происходит ***мгновенный*** переход компьютера из одного внутреннего состояния в другое — из состояния, соответствующего значениям обрабатываемых данных и признаков до выполнения действия, в состояние, соответствующее значениям данных и признаков после выполнения этого действия. Таким образом, множество внутренних состояний компьютера также является дискретным. В связи с этим говорят, что компьютер в целом является ***дискретным устройством.***

> Устройства, обеспечивающие работу с дискретными данными или сигналами, обладающие дискретным множеством внутренних состояний, а также выполняющие действия в дискретные моменты времени, называются

дискретными. Компьютер в целом относится к группе дискретных устройств.

Тема 5: Вентили и комбинационные схемы

Основными, базовыми операциями, которые обязательно должен «уметь» выполнять процессор компьютера над двоичными кодами данных, являются логические операции отрицания, дизъюнкции, конъюнкции, арифметического сложения, а также сдвига кода. Используемые для реализации этих и других операций устройства принято называть **вентилями** (от нем. Ventil — клапан).

ВНИМАНИЕ Основные устройства компьютера 41

Вентилем называется физическое устройство, реализующее одну из базовых логических операций: отрицание, дизъюнкцию, конъюнкцию, исключающую дизъюнкцию и т. д. Вентили, входящие в состав процессоров компьютера, называют также логическими элементами.

Релейно-контактные вентили

В 1938 г. известный специалист в области теории информации Клод Шеннон предложил использовать для моделирования основных логических операций релейно-контактные электрические схемы. Этот подход был использован в электромеханических релейных вычислительных машинах Z-3 (Германия, Конрад Цузе, 1939), «Марк 2» (США, Говард Айкен, 1947), РВМ-1 (СССР, Н. И. Бессонов, 1951) и в целом ряде других машин.

В качестве примера рассмотрим изображенную на рис. 3.2 реализацию логических операций конъюнкции и дизъюнкции с помощью схем, которые принято называть вентилями «И» и «ИЛИ» соответственно. Логические операнды в этих схемах соответствуют релейно-контактным переключателям, которые на рисунке обозначены как ***p n q***. При этом логические значения 0 и 1 моделируются соответственно разомкнутым и замкнутым состояниями контакта. Результат операции отображается включенной в сеть лампочкой. Если лампочка не горит, результат равен 0; горящая лампочка соответствует результату, равному 1. Иначе говоря, знаки двоичного алфавита 0 и 1 моделируются ***отсутствием*** или ***наличием тока*** в цепи соответственно. Заметим: переключатели называются релейными потому, что управление ими обычно осуществляется с помощью электромагнитных реле.

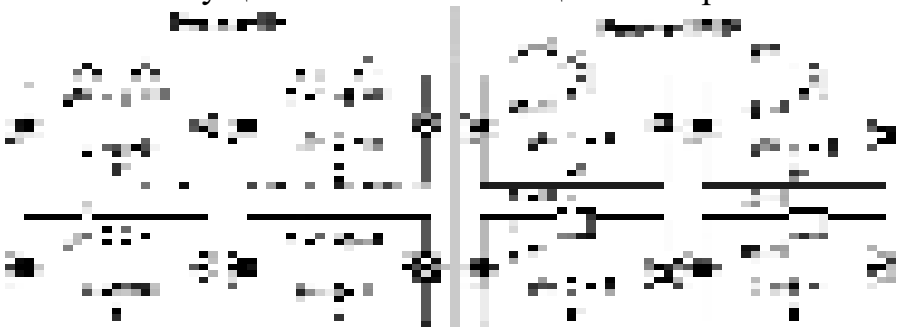


Рис. 3.2. Вентили «И» и «ИЛИ» на базе релейно-контактной схемы

На рис. 3.2, ***слева*** показаны четыре состояния вентиля «И», которые соответствуют различным строкам таблицы истинности (табл. 3.1) этой операции. Так, в состоянии ***а*** оба контакта разомкнуты, то есть ***p*** = 0 и ***q*** = 0, и, следовательно, ток в цепи не течет — лампочка не горит — ***p*** д ***q*** = 0. Состояния ***б*** и ***в*** моделируют другие строки таблицы истинности операции конъюнкции, в которых результат равен нулю: 1л0 = 0 и 0 л 1 = 0 соответственно. Ток в цепи течет и лампочка горит только в состоянии г, то есть в том случае, когда замкнуты оба контакта, что соответствует строке таблицы 1 д 1 = 1.

На том же рисунке ***справа*** изображена схема вентиля «ИЛИ», соответствующего

операции дизъюнкции. Лампочка не горит только в состоянии ***а***, когда разомкнуты оба контакта, то есть 0 v 0=0. Во всех остальных состояниях лампочка горит, и, следовательно, результат операции равен 1, то есть моделируются строки таблицы истинности **1 v 0 = 1 , 0 v 1 = 1 h 1 v 1 = 1 .** Релейная схема вентиля «НЕ», соответствующего логической операции отрицания, устроена более сложно, поэтому она на рисунке не приводится.

Основным недостатком описанного способа реализации операции над данными является ***наличие механических перемещений*** контактов релейной схемы, которые требуют значительных временных затрат. Время выполнения операции определяется временем, в течение которого производится замыкание или размыкание контакта. Даже в самых современных релейных устройствах на это требуются по крайней мере сотые доли секунды. Именно это обстоятельство привело к тому, что уже в 1950-е годы релейные вычислительные машины были вытеснены сначала ламповыми, а затем транзисторными компьютерами.

Все современные вентили реализуются на основе полупроводниковых устройств — транзисторов (рис. 3.3, ***а)*** или их аналогов в интегральных схемах, характерное время срабатывания которых в настоящее время приближается к долям наносекунды (1 не = 10^{-9} с).

Рис. 3.3. Транзисторы: а — внешний вид; б — изображение на схемах

В задачу курса не входит изучение физических процессов и явлений, происходящих внутри транзистора. Достаточно понимания его поведения в электрической цепи в нескольких простых ситуациях. Транзистор имеет три контактных вывода, которые принято называть эмиттером, базой и коллектором (рис. 3.3, ***б).*** Внешнее напряжение (например, +5 В) через резистор подается на коллектор,а эмиттер заземляется, что создает условия для протекания тока в цепи. Если напряжение на контакте базы отсутствует, то транзистор ведет себя в электрической цепи как резистор с большим внутренним сопротивлением. А если на контакт базы подать напряжение определенной величины, то транзистор ведет себя как проводник с очень маленьким сопротивлением. В первом случае говорят, что транзистор ***заперт***, а во втором — что транзистор ***открыт.*** Контактный вывод базы считается входом схемы. Начиная или прекращая подачу напряжения на этот вход, можно управлять поведением транзистора, открывать или закрывать его. Выходом схемы считается контактный вывод коллектора, на котором регистрируется результат управляющего воздействия на схему.

В вентильных схемах на базе транзисторов двоичному знаку 0 соответствует низкое напряжение с уровнем от 0 до 1 В, а двоичному знаку 1 — высокое напряжение с уровнем от 2 до 5 В. Могут применяться и другие конкретные значения напряжений, однако в любом случае используются ***два четко различимых его уровня.*** Подачу на базу транзистора низкого напряжения можно трактовать как поступление на вход

схемы бита со значением 0, а подача на базу высокого напряжения соответствует поступлению на вход схемы бита со значением 1. Аналогично регистрация низкого напряжения на коллекторе транзистора может, трактоваться как формирование на выходе значения 0, а регистрация на нем высокого напряжения — как формирование на выходе значения 1.

Вентиль «НЕ»

Рассмотрим поведение схемы, изображенной на рис. 3.4, ***а***, при различных значениях входного бита ***р.*** Пусть на вход схемы подано значение ***р*** = 0. Тогда транзистор заперт, он ведет себя в цепи как дополнительный резистор с сопротивлением, гораздо большим, чем сопротивление резистора, через который транзистор подключен к источнику питания схемы. В силу того что падение напряжения на участке цепи пропорционально сопротивлению этого участка, напряжение в точке выхода будет мало отличаться от высокого напряжения источника питания. Другими словами, на выходе схемы в этом случае формируется значение 1.

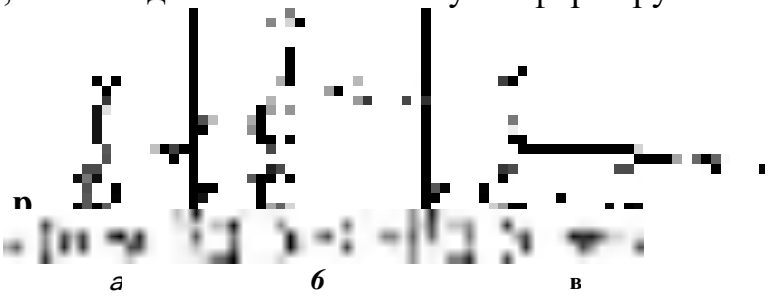


Рис. 3.4. Вентили и их обозначения: а — «НЕ»; *б* — «НЕ И»; *в* — «НЕ ИЛИ»

Пусть теперь на вход поступил ***6т р*** = 1. Тогда транзистор открыт, он ведет себя как проводник с очень маленьким сопротивлением, и все падение напряжения происходит на резисторе[2]. Выход схемы при этом оказывается как бы напрямую соединенным с землей, следовательно, напряжение на нем близко к нулю. Другими словами на выходе схемы формируется значение 0. Таким образом, обсуждаемая схема представляет собой вентиль «НЕ», реализующий одноместную операцию отрицания: при поступившем на вход значении ***р*** на выходе схемы формируется значение -i***р.*** Отметим, что время переключения вентиля из одного состояния в другое существенно зависит от физической реализации транзистора. Но в любом случае это переключение для электронных устройств происходит очень быстро, за время, измеряемое микро-, наносекундами или их долями.

Вентили «НЕ И» и «НЕ ИЛИ»

Рассмотрим логику работы схемы, изображенной на рис. 3.4, ***б.*** Она состоит из двух соединенных последовательно транзисторов. У этой схемы два входа, обозначенных на рисунке буквами ***р*** и ***q,*** и один выход. Если на входы поступают единичные значения (р=**1**и<**7=1**), то оба транзистора открыты, участок цепи с ними имеет очень маленькое сопротивление и, следовательно, как и в вентиле «НЕ», на выходе формируется значение 0. Во всех остальных случаях хотя бы один из транзисторов оказывается запертым и участок цепи с транзисторами обладает высоким сопротивлением, что приводит к формированию на выходе значения 1. Анализируя таблицу истинности работы этой схемы (табл. 3.1, четвертый столбец), приходим к выводу, что она описывается выражением -|(рд^) Поэтому такая схема называется вентилем «НЕ И». Эта операция известна также под названием «штрих

[2] Заметим, что это обстоятельство обусловливает наличие в схеме резистора, через который питание подается на коллектор. Отсутствие резистора в открытом режиме работы транзистора приведет к тому, что все падение напряжения произойдет на обладающем очень маленьким сопротивлением транзисторе и ток большой величины приведет к его перегоранию.

Шеффера». Ее обозначают значком «|»: п(рд 59#) з ***p\ q.***

Таблица 3.1. Таблицы истинностей базовых вентилей

p	*ч р л q,* **«И»**	**-1** *(р л q),* **И»,**	**«НЕ штрих**	*Pvq,* **«ИЛ**	*P¥Я>* **«Исключа**	**-ЧР** ˅ *Я)>* **«НЕ ИЛИ»,**
0	**0 0**	**1**		**0**	**0**	**1**
0	**1 0**	**1**		**1**	**1**	**0**
1	**0 0**	**1**		**1**	**1**	**0**
1	**1 1**	**0**		**1**	**0**	**0**

4.

В схеме, изображенной на рис. 3.4, в, транзисторы соединены параллельно. Следовательно, участок цепи с транзисторами обладает высоким сопротивлением только в том случае, когда оба транзистора закрыты одновременно. Поэтому если на оба входа поступают нулевые значения ***(p*** = 0 и ***q*** = 0), на выходе формируется значение 1. Во всех остальных случаях хотя бы один из транзисторов открыт и, следовательно, весь содержащий их параллельное соединение участок цепи обладает маленьким сопротивлением. Это значит, что на выходе схемы формируется значение 0. Анализируя таблицу истинности работы этой схемы (табл. 3.1, седьмой столбец), приходим к выводу, что она описывается выражением -|(p v#). Поэтому такая схема называется вентилем «НЕ ИЛИ». Эта операция известна также под названием «стрелка Пирса». Ее обозначают значком

«>U: (p vg) = ***piq.***

Вентили «НЕ», «НЕ И» и «НЕ ИЛИ», используемые для построения других вентилей и произвольных схем, считаются базовыми, а схемы, которые получаются с помощью всевозможных комбинаций базовых вентилей, принято называть **цифровыми логическими схемами.** Важным частным случаем цифровых схем являются **комбинационные схемы,** в которых значения, получаемые на выходах схемы, зависят только от значений, поступающих на ее входы. Такие схемы классифицируются также как **схемы без памяти.**

Использование в комбинационных схемах стандартных обозначений транзистора и других показанных на рис. 3.4, ***б*** обязательных элементов (значков земли, питания и т. д.) приводит к излишней громоздкости и затрудненному восприятию схем. Поэтому в комбинационных схемах используются условные обозначения для каждого из базовых вентилей в целом. Условные обозначения трех рассмотренных ранее вентилей приведены в нижней части рис. 3.4 под соответствующими им схемами.

Вентили «И» и «ИЛИ»

Теоретически для задания ***любой*** логической функции можно обойтись только одной операцией — стрелкой Пирса или штрихом Шеффера. Таким образом, вентили «НЕИ»и«НЕ ИЛИ» могут рассматриваться как ***универсальные***, из которых можно составить схему, соответствующую любому логическому выражению. Но получаемые при этом логические выражения и соответствующие им цифровые схемы оказываются чрезвычайно громоздкими и малопонятными. В то же время известно, что логические функции удобно задавать, используя три основные логические операции: отрицание, дизъюнкцию и конъюнкцию. В связи с этим целесообразно использовать в комбинационных схемах вентили, соответствующие этим операциям.

Рис. 3.5. Вентили и их условные обозначения: а — «И»; *б* — «ИЛИ»

Способ построения вентилей для операций конъюнкции и дизъюнкции вытекает из очевидных соотношений -.(-.(р д #)) = рд<**7**и-.(-|(ру $\boldsymbol{q}$)) **=** $\boldsymbol{pvq}$**.** Следовательно, соединив выходы вентилей «НЕ И» и «НЕ ИЛИ» со входом вентиля «НЕ», получим удобные для построения любых цифровых схем вентили «И» и «ИЛИ» операций конъюнкции и дизъюнкции соответственно. Схемы этих вентилей и их обозначения приведены на рис. 3.5. Вентили «И» и «ИЛИ» также относятся к базовым. Отметим, что для реализации вентиля «НЕ» достаточно одного транзистора, для вентилей «НЕ И» и «НЕ ИЛИ» требуется по два транзистора, а для вентилей «И» и «ИЛИ» необходимо уже по три транзистора на каждую схему.

Тема 6: Архитектура компьютера на базе процессора i8086

Различные модели вычислительных машин существенно отличаются друг от друга по своему устройству. Для более подробного начального изучения выбрана архитектура персонального компьютера IBM PC (от International Business Machines Personal Computer) на базе микропроцессора i8086 (или просто 8086). Буква i в маркировке отображает название производителя процессора — фирмы Intel (от Integrated Electronics). В литературе для этого процессора иногда встречается обозначение iAPX8086, где iAPX образовано от Intel Advanced Processor Architecture — продвинутая процессорная архитектура фирмы Intel.

Модель микропроцессора i8086 является в некотором смысле базовой для всех остальных моделей IBM PC. По сути дела, с появлением этой модели микропроцессоров персональные компьютеры начали стремительными темпами завоевывать и изменять мир.

Основные устройства компьютера

Для обсуждения принципов работы компьютера необходимо более подробно обсудить некоторые особенности его важнейших устройств: оперативной памяти, процессора, шины и т. д.

Оперативная память

Память в компьютере, обеспечивающая одну из важнейших его функций — ***хранение*** данных и программ, представлена группой устройств, которые отличаются друг от друга физической реализацией, способом использования, объемом, стой- мостью и некоторыми другими характеристиками. Из всех видов памяти компьютера наиболее важное значение имеет **оперативная память** (ОП). Компьютер принципиально не в состоянии работать, если в нем отсутствует оперативная память, в то время как отсутствие других видов памяти только снижает эффективность его работы, но возможность выполнять программы и обрабатывать данные остается.

Уровень оперативной памяти компьютера подобен кратковременной памяти человека. Когда человек сосредоточен на выполнении какого-либо дела — играет на музыкальном инструменте, управляет автомобилем, — он хорошо помнит все детали, подробности текущей ситуации, а также план выполняемой работы. После перехода к другой деятельности все это забывается, но в памяти возникают другой план и другие подробности.

> Оперативная память представляет собой устройство компьютера, которое предназначено для хранения программ, находящихся на стадии выполнения, а также обрабатываемых этими программами данных.

Согласно определению, в оперативной памяти на стадии выполнения может одновременно находиться несколько программ. Кроме того, в ней могут находиться как обрабатываемые, так и уже обработанные или же ожидающие обработки данные для всех выполняющихся процессором программ.

Важнейшим отличительным свойством оперативной памяти является то, что процессор компьютера имеет непосредственный доступ ко всей информации, которая в ней находится. Программы, находящиеся в оперативной памяти, могут быть выполнены процессором, а данные, находящиеся в ней, могут быть по этим программам обработаны. Кстати, именно потому что процессор может оперировать данными и программами, этот вид памяти называется оперативной памятью.

Оперативная память ***энергозависима.*** Это значит, что при отключении электропитания компьютера все программы и данные, которые в этот момент находятся в

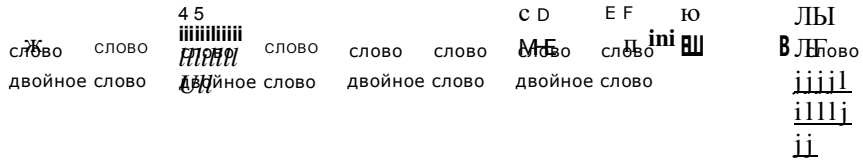
оперативной памяти, ***безвозвратно теряются.*** Наряду с другими устройствами компьютера память характеризуется важным показателем — **быстродействием,** которое в дальнейшем рассматривается более подробно. Пока будем считать, что эта характеристика определяется временем, которое требуется для получения из памяти запрошенных данных, — чем меньше промежуток времени от момента запроса данных до момента их получения из памяти, тем больше скорость памяти.

Ранее было выяснено, что элементарными устройствами любого вида памяти компьютера являются биты, которые предназначены для хранения одной двоичной цифры. Биты оперативной памяти могут быть реализованы различными цифровыми логическими схемами. Одна из них — это рассмотренный ранее D-тригтер. Другие возможные реализации битов оперативной памяти рассматриваются в 5.3 и 7.1.

Следующим уровнем в организации памяти являются байты. С логической точки зрения, байт состоит из восьми бит, каждый из которых независимо от остальных может содержать любую двоичную цифру, — следовательно, байт служит для хранения восьмиразрядного двоичного кода.

Все байты в пределах оперативной памяти пронумерованы идущими подряд целыми числами, при этом номер начального байта всегда считается равным нулю. Номер байта, заданный в обсуждаемой в дальнейшем форме (см. 4.1.3), принято называть его **адресом.** Нумеруются байты памяти в двоичной системе счисления, но, как правило, номера записываются в шестнадцатеричном виде (рис. 4.1, ***сверху).***

Номера байтов оперативной памяти



Учетверенное слово
Учетверенное слово
Траница параграфа

Рис. 4.1. Нумерация байтов и стандартные поля оперативной памяти

Содержимое любого байта памяти может задаваться и обрабатываться независимым от остальных байтов образом. После получения адреса любого байта оперативной памяти процессор компьютера может прочитать код, который в нем записан, или записать в этот байт какой-либо другой код. В то же время процессор не может получить доступ к отдельному биту байта. Для этого он должен сначала обратиться к байту, который содержит нужный бит.

Из-за того что есть возможность напрямую обращаться к любому байту, оперативную память называют еще **прямоадресуемой памятью,** или **памятью с прямым доступом,** и используют для нее обозначение RAM (от Random Access Memory — память произвольного доступа). Кроме того, для оперативной памяти применяются и другие названия и обозначения: **оперативное запоминающее устройство** (ОЗУ), **основная оперативная память** (ООП), а иногда просто **основная память** (ОП).

В 1.3 введено понятие поля, которое представляет собой рассматриваемую как единое целое группу смежных байтов памяти. Байт поля с наименьшим номером считается младшим, а байт поля с наибольшим номером считается его старшим байтом. Поле памяти характеризуется адресом и длиной. Адресом поля считается номер его младшего байта, а длиной поля считается количество байтов, из которых оно состоит (рис. 4.2). Байты внутри поля нумеруются слева направо. Напомним, что биты внутри полей памяти и ее отдельных байтов принято нумеровать справа налево, также начиная с нуля. На рис. 4.2, ***внизу***, в увеличенном виде показаны пример содержимого и нумерация байтов и битов поля длиной 4 байта.

Принято различать стандартные и нестандартные поля. Нестандартные поля могут иметь произвольную длину и любой адрес, в то время как на длину и адрес стандартных полей накладываются определенные ограничения, которые зависят от архитектуры компьютера. Нестандартные поля используются в основном для хранения текстовых данных, а стандартные — для хранения числовых данных различных форматов.

Длина поля 4 байта, номер начального байта 5

0 12 3 4	S 6 7 8	9 N
jiliffiSffip MIIII	jiiiiiiiiii	iiiiiii 4 ЯШМ
	Поле	

Нумерация битов внутри поля

Нумерация байтов внутри поля

Рис. 4.2.Поле оперативной памяти

Различают следующие типы стандартных полей: **полуслово, слово, двойное сло-**

во, учетверенное слово. Базовым стандартным полем считается слово, длина которого является одной из основных характеристик архитектуры компьютера. Длины всех остальных разновидностей полей очевидным образом связаны с их названиями. В архитектуре процессора i8086 слово памяти имеет длину 2 байта, то есть 16 бит. Соответственно полуслово имеет длину 1 байт, двойное слово — 4 байта, а учетверенное слово — 8 байтов. Существуют и другие варианты длины слова памяти. Укажем, например, на еще один распространенный в настоящее время вариант, в котором слово состоит из четырех байтов. Тогда полуслово имеет длину 2 байта, а двойное и учетверенное слова — 8 и 16 байтов соответственно.

Адреса стандартных полей не могут быть произвольными, они обязательно должны быть кратными длине поля: адрес слова должен без остатка делиться на 2, адрес двойного слова — на 4, адрес учетверенного слова — на 8. Адрес состоящего из одного байта полуслова может быть любым. Слова памяти всегда начинаются с четных адресов: 0, 2, 4, 6..., двойные слова — с адресов 0, 4, 8, C, 10..., а учетверенные слова — с адресов 0, 8, 10, 18... Примеры стандартных полей приведены на рис. 4.1. Отметим, что изображенное на рис. 4.2 поле не относится к стандартным полям. Несмотря на то что оно имеет длину 4 байта, это поле нельзя считать двойным словом, так как его адрес не кратен четырем. На указанном рисунке изображено нестандартное поле длиной 4 байта.

Процессор

Следующая основная функция компьютера — обработка данных, осуществляемая по заранее заданной человеком программе. Как мы уже выяснили, эта функция выполняется устройством, которое называется процессором, иногда **центральным процессором** (ЦП), или CPU (от Central Processing Unit — центральный обрабатывающий блок, устройство), а в персональных компьютерах еще и микропроцессором.

Рассматривая комбинационные схемы, мы выяснили, каким образом могут быть в принципе реализованы устройства, обеспечивающие выполнение базовых операций над двоичными данными, таких как сравнение, сложение, сдвиг и т. д. Напомним, что рассматривались упрощенные варианты этих устройств. Реальные процессоры содержат их аналоги в той или иной усовершенствованной форме. Для повышения удобства составления программ, а также для управления последовательностью выполнения действий возможности процессоров по обработке данных существенно расширены. Реальные процессоры могут выполнять сотни различных действий, не входящих в обсуждавшийся базовый набор операций. Все множество действий, которые могут быть выполнены процессором, называется его **системой команд.** Отметим, что процессоры разных моделей компьютеров обладают различными системами команд. Поэтому система команд процессора фактически определяет модель компьютера. Можно также утверждать, что модель компьютера определяет его систему команд.

> Системой команд процессора называется набор действий над данными, которые могут быть им выполнены. Система команд процессора взаимнооднозначно связана с его моделью.

Машинной командой (или машинной инструкцией) называется закодированное в двоичном алфавите указание процессору на выполнение отдельного действия, принадлежащего к его системе команд.

Конкретная последовательность машинных команд, которая обеспечивает необходимую обработку данных, называется программой, записанной на уровне машинного языка, или просто машинной программой.

Итак, машинная программа представляет собой некоторую последовательность двоичных кодов, поэтому ее часто называют также **программным кодом.** Заметим, что команды программы расположены в оперативной памяти не обязательно подряд на сплошном ее участке. Запись программы в виде двоичных кодов — это единственный способ задания программы, в котором она непосредственно* воспринимается, «понимается» и выполняется процессором компьютера. Все остальные способы записи программ являются промежуточными или вспомогательными.

Процессоры компьютеров характеризуются рядом параметров. Основными считаются **тактовая частота** и **длина машинного слова.** Понятие тактовой частоты рассмотрено в 3.1. Отметим, что тактовая частота различных процессоров, даже одной и той же модели, может изменяться в широких пределах. Так, например, тактовая частота процессора i8086 в зависимости от его модификации колеблется в пределах от 5 до 10 МГц.

Процессор выполняет каждую машинную команду программы за определенное количество тактов. Скажем, существуют процессоры, в которых операция сложения двух чисел в формате с фиксированной точкой выполняется за два такта. А операция деления может занять и 25 тактов такого процессора. Таким образом, можно сделать следующий вывод: чем выше тактовая частота, тем быстрее работает компьютер. Другими словами, скорость работы, или **быстродействие** компьютера, которое может характеризоваться количеством машинных команд, выполняемых компьютером за одну секунду, связано с его тактовой частотой. Одновременно мы выяснили, что быстродействие зависит и от выполняющейся программы, от того, какие команды — сложения или, скажем, деления — в ней преобладают. Если взять программу, в которой имеются только команды типа сложения, выполняющиеся за два такта, тогда быстродействие процессора с тактовой частотой 500 МГц оценивается в 250 млн. машинных команд в секунду. Если же в программе преобладают действия типа деления, то быстродействие оказывается равным только 20 млн. команд в секунду. Поэтому быстродействие определяют на специальных тестовых программах, которые дают представление как о наивысшей возможной скорости вычислений, так и о предполагаемой средней скорости обработки данных.

Вычислительная мощность компьютера определяется также количеством байтов, которые могут быть одновременно обработаны процессором. Чем больше это количество, тем больше данных в единицу времени может быть обработано.

> Машинным словом называется наибольшая группа байтов, которая может быть обработана процессором за один машинный такт. Количество байтов в машинном слове называется длиной машинного слова.

Разные модели машин имеют машинные слова, содержащие различное количество байтов. Первые персональные компьютеры могли за такт переслать или обработать всего один байт, это значит, что машинное слово состояло из одного байта.

В процессоре i8086 машинное слово состоит из двух байтов, то есть длина машинного слова совпадает с длиной слова памяти. Другими типичными для настоящего времени длинами машинных слов являются 4 и 8 байтов. Длина машинного слова

обычно выступает в качестве основной характеристики архитектуры компьютера. Так, если машинное слово состоит из одного байта, то есть из 8 битов, то говорят: «восьмибитная архитектура», «восьмибитный компьютер», а если из двух байтов, то есть из 16 битов, то говорят: «шестнадцатибитная архитектура», «шестнадцатибитный компьютер» и т. д.

Тема 7: Машинные команды процессора i8086

Прежде чем обсуждать систему команд процессора, следует более подробно выяснить, как устроена машинная команда, представляющая собой исчерпывающе точное указание процессору на выполнение отдельного действия, принадлежащего системе его команд.: этапы развития 52

Структура машинной команды

Вначале обратим внимание на то, каким образом задаются аналогичные действия, когда их исполнителем является человек. Типичными указаниями на выполнение действий, подобных командам компьютера, являются записи арифметических и алгебраических операций вида 4 + 3 = 7 или ***c = ab.*** Такое указание испол- нителю-человеку содержит:

- обозначающий действие значок (+, / и т. д.), который указывает, какое именно действие из всех доступных исполнителю (из тех, которые он умеет выполнять, из системы команд исполнителя) ему следует выполнить;
- числа, над которыми требуется выполнить действие. Они могут быть заданы непосредственно (3, 4) либо некоторым условным названием (я, ***b***). Могут быть также заданы некоторые правила их определения;
- способ фиксации результата (7, ***c)*** выполненного действия.

Вполне очевидно, что указание для выполнения действия исполнителю — процессору компьютера — должно содержать не менее исчерпывающую информацию о действии. Естественным для компьютера способом его задания является двоичное кодирование всех элементов команды.

Каждому действию, принадлежащему системе команд процессора, ставится во взаимно однозначное соответствие двоичный код, который принято называть кодом операции (КОП). Данные (число, логическое значение, символ, строка бит, адрес поля памяти), участвующие в выполнении действия, обычно называются операндами. Операнд может быть задан непосредственно к команде, он может находиться в регистре процессора или в поле оперативной памяти. В любом случае операнд представлен некоторым двоичным кодом. Результат может быть помещен на хранение в один из регистров процессора или же в поле оперативной памяти. Если для операнда или результата используется регистр процессора, то необходимо указать, какой именно. В машинной команде это делается с помощью задания соответствующего регистру двоичного кода. Если используется поле памяти, необходимо указать его адрес, который также представляет собой двоичный код.

> Машинная команда в целом представляет собой некоторый двоичный код, который так же, как и код операнда, может быть помещен в регистр процессора или в поле оперативной памяти. Для сокращения записи машинные команды обычно задаются в шестнадцатеричном виде.

Каждая машинная команда характеризуется ***длиной***, которая равна количеству байтов, занятых ее кодом. Если машинная команда находится в поле оперативной памяти, то она характеризуется еще и ***адресом***, которым считается адрес этого поля.

Для описания структуры кода команды, так же как и для описания кодов данных, используется понятие формат команды, являющееся разновидностью введенного в главе 2 общего понятия формата. Формат машинной команды представляет собой

исчерпывающе полный набор правил ее кодирования. Он определяет длину команды, распределение разрядов поля зд отдельными ее элементами, количество и способы задания операндов, всевозможные признаки и параметры, уточняющие режим выполнения команды, и т. д.

Адресность машинных команд

Единственным обязательным элементом машинной команды является ее код. Кроме того, в составе команды обычно имеется несколько операндов, которые задаются либо непосредственно, либо указанием их местоположения (поле памяти или регистр процессора). Поскольку операнды в машинных командах чаще всего задаются определением их местоположения, соответствующий элемент кода команды принято называть адресом. В связи с этим при описании структуры команды мы будем обозначать ее операнды буквой А с каким-либо номером. Например, А1 — это обозначение первого операнда команды.

Как уже отмечалось, у разных машинных команд может быть различное количество операндов. Наибольшее количество операндов имеют ***четырехадресные*** команды со структурой КОП А1 А2 А3 А4. Первый (А1) и второй (А2) адреса задают операнды, например слагаемые. Третий адрес (А3) определяет место, куда следует записать результат, а четвертый операнд (А4) определяет адрес команды, которую процессор должен выполнить следующей.

Команды этого формата не очень распространены, так как в большинстве компьютеров принят естественный порядок выполнения команд программы, в соответствии с которым они выполняются последовательно друг за другом в том порядке, в каком находятся в оперативной памяти. В связи с этим адрес следующей команды задавать не нужно, так как он определяется простым сложением адреса и длины текущей команды. Этот принцип позволяет уменьшить количество операндов в команде и, следовательно, уменьшить ее длину.

- Наиболее естественную структуру имеют ***трехадресные*** команды — КОП А1 А2 А3. Такая команда содержит адреса операндов А1 и А2, а также адрес А3, по которому следует разместить результат ее выполнения. Систему трехадресных команд имели широко распространенная в свое время советская машина БЭСМ 4, ее аналоги М 220, М 222 и некоторые другие машины.
- С целью уменьшения длины команды из ее структуры можно исключить определяющий местоположение результата третий адрес. Результат при этом можно помещать по первому или по второму адресу команды, заменяя тот операнд, который в дальнейших вычислениях не требуется. Так получаются ***двухадресные*** команды со структурой вида КОП А1 А2. Двухадресной системой команд обладали, например, машины семейства «Минск».

Если пойти по пути дальнейшего упрощения структуры команды, то можно прийти к ***одноадресным*** командам вида КОП А1. Очевидно, что в команде с такой структурой может быть определен только один операнд. Положение другого операнда в одноадресных системах команд всегда фиксировано — он должен находиться в специальном регистре сумматора, который является составной частью арифметико-логического устройства. Отсюда вытекает следующая схема задания такого, например, действия, как сложение. Вначале с помощью специальной команды одно из слагаемых заносится в регистр сумматора. Адрес другого слагаемого определяется в команде сложения. Результат остается в регистре сумматора, подготавливая тем самым выполнение следующей команды. Одноадресной

была, например, система команд у лучшей в Европе в конце 1960-х гг. советской машины БЭСМ 6.

И наконец, следует упомянуть еще об одном возможном варианте — о ***безадресных*** командах, которые состоят только из кода операции. Операнды таким командам либо вообще не нужны, либо всегда строго фиксированы и не могут быть изменены ни при каких обстоятельствах. К безадресным относится, например, упоминавшаяся ранее команда остановки, прекращающая выполнение программы.

Способы адресации

Способы задания операндов в команде принято называть адресацией. Существует несколько десятков различных способов адресации, но, в принципе, все эти способы являются различными вариантами четырех основных:

- ***непосредственная адресация*** означает, что сам операнд (точнее, его код) включается в машинную команду как ее составная часть;
- ***регистровая адресация*** означает, что операнд находится в регистре процессора, а в команде указывается код этого регистра;
- ***прямая адресация*** означает, что операнд находится в поле оперативной памяти, а в команде указан адрес этого поля;
- ***косвенная адресация*** означает, что операнд также находится в поле оперативной памяти, а в команде содержатся некоторые элементы, по которым однозначно определяется адрес этого поля.

Отметим, что непосредственная адресация при всей ее простоте имеет ограниченные возможности применения, так как при любом изменении операнда приходится изменять и содержащую его код машинную команду. Это означает изменение и программы в целом, что, вообще говоря, весьма нежелательно, так как в большинстве случаев требует ее повторной подготовки к выполнению (например, повторной трансляции). Непосредственно в машинной команде могут быть заданы только такие операнды, которые не меняются при любых выполнениях программы.

Использование для задания операндов регистровой адресации является самым эффективным и самым простым способом. Однако, как мы знаем, процессор i8086 обладает малым количеством регистров. Поэтому неизбежны сложности при организации вычислений с большим количеством различных данных. Кроме того, в начале выполнения программы все ее данные размещаются в оперативной памяти. Следовательно, в любом случае необходимы способы адресации, обеспечивающие возможность выборки операндов из полей оперативной памяти. Такими возможностями обладают прямая и косвенная адресация. В прямой адресации адрес поля памяти прямо указывается в команде, а в косвенной его необходимо определить по заданной в команде информации. Косвенная адресация по сравнению с прямой обеспечивает большую гибкость, необходимую при работе с данными сложной структуры, такими как массивы и записи,

В связи с введением косвенной адресации приходится различать адрес операнда из команды $A_{ком}$ и адрес, по которому происходит фактическое обращение в память, $Л_{исп}$.

Адрес, заданный в команде, $Л_{ком}$, представляет собой указанный в команде адрес поля или элементы, по которым он определяется. Исполнительный, или физический, адрес $Л_{исп}$ представляет собой адрес, по которому производится фактическое обращение в оперативную память в момент выполнения

команды.

Тема 8: Развитие архитектуры и параллелизм вычислений

Целью развития вычислительных машин всегда было (да, наверно, и будет) улучшение эффективности обработки данных, выражающееся в повышении скорости обработки и увеличении объема обрабатываемых данных. Одновременно с этим разработчики добивались повышения надежности и уменьшения стоимости компьютеров, а также обеспечения удобства и упрощения работы пользователей.

В процессе развития вычислительных систем этот комплекс требований удовлетворялся за счет применения различных физических принципов хранения и обработки информации, а также усовершенствования технологий производства аппаратуры. Это приводило к желаемому повышению надежности и быстродействия отдельных устройств, росту емкости памяти, а также к уменьшению стоимости компьютеров. Кроме того, происходило уменьшение габаритов компьютеров, расширялись области использования компьютерных технологий обработки данных, упрощалось взаимодействие человека и компьютера. Одновременно с развитием аппаратных средств совершенствовались средства и методы разработки эффективного и надежного программного обеспечения.

Вместе с тем уже довольно давно стало ясно, что рано или поздно этот путь приведет к невозможности дальнейших улучшений в рамках используемого физического принципа работы компьютера. К настоящему времени возможности увеличения производительности отдельно взятого компьютера подходят к своим естественным границам, которые определяются конечностью скорости света и некоторыми другими физическими ограничениями.

Выход был найден в широком использовании параллелизма, который, по-видимому, является единственным способом дальнейшего роста производительности вычислительных систем, базирующихся на электронных устройствах.

> В информатике параллелизмом называется одновременное выполнение различными устройствами или различными частями одного и того же устройства каких-либо действий по обработке информации.

В приведенном ранее общем понятии параллелизма главным является одновременная работа нескольких устройств. При этом не уточняется характер выполняемых действий по обработке информации. Это уточнение приводит к выделению собственно параллелизма, когда несколько устройств одновременно выполняют различные команды одной и той же или разных программ, и конвейеризации, когда несколько одновременно работающих устройств последовательно выполняют одну сложную операцию и при этом на разных этапах обработки одновременно находится несколько операций.

Отметим, что конвейером (от convey — транспортировать) в промышленности называется непрерывно или периодически движущееся транспортное устройство для сборки машин, обработки какого-либо материала и т. д. с ***последовательным выполнением отдельных этапов несколькими одновременно работающими исполнителями (людьми, автоматами).*** Конвейер широко используется для повышения эффективности производства, так как каждый исполнитель, специализируясь на выполнении отдельной простой операции, выполняет ее быстрее, чем это делает один исполнитель, последовательно выполняющий разные операции.

ВНИМАНИЕ ----------

Одновременное выполнение несколькими специализированными устройствами различных этапов различных операций (команд, действий и т. д.), при котором этапы одной операции выполняются этими устройствами последовательно, называется конвейеризацией.

Для собственно параллельной работы требуется несколько универсальных устройств, которые одновременно выполняют несколько сложных действий, при этом каждое универсальное устройство выполняет все этапы одного действия. Для конвейерной обработки нужно несколько более простых специализированных устройств, также работающих одновременно, при этом каждое из них выполняет один и тот же этап разных действий.

Чтобы лучше понять разницу между собственно параллелизмом и конвейеризацией, рассмотрим бытовую аналогию. Пусть имеется бригада строительных рабочих, которой поручено построить несколько домов. Понятно, что четыре таких же бригады, работая одновременно, должны построить эти же дома в четыре раза быстрее, чем одна бригада (разумеется, при наличии для работы всех необходимых условий).

Организовать одновременную (параллельную) работу нескольких бригад можно по-разному. ***Собственно параллельной*** является такая организация, когда каждая бригада выполняет все работы на строительстве одного и того же дома от начала до конца. Таким образом, четыре бригады одновременно построят четыре дома.

С другой стороны, можно выделить четыре специализированные группы, одна из которых готовит фундамент, другая одновременно кладет стены другого здания, третья в это же время делает крышу на третьем, а четвертая выполняет окончательную отделку на четвертом. Тогда на строительстве дома будут последовательно заняты все четыре бригады, которые по конвейеру выполняют свои операции. Преимущество конвейерной организации по сравнению с собственно параллельной в том, что узкая специализация повышает производительность каждой бригады. При этом каждый работник может иметь более низкую квалификацию, так как ему не приходится выполнять все разновидности работ. В реальных строительных организациях всегда применяется конвейерная система.

В архитектуре компьютеров широко применяются и собственно параллелизм, и конвейеризация. В последнем случае преимущества возникают из-за упрощения устройства каждого специализированного узла, которые, следовательно, могут работать быстрее, чем универсальные, а значит, более сложные и медленные устройства.

Далее во второй части учебника кратко прослеживаются основные этапы развития архитектуры вычислительных систем, акцент делается на повышении их производительности и организации параллельных вычислений. При этом производительность (мощность) компьютера в первом приближении оценивается с помощью скорости обработки данных, которая задается либо тактовой частотой процессора, либо количеством выполняемых им за одну секунду арифметических операций.

Начальные этапы развития

Потребность в выполнении различных вычислений появилась, по-видимому, вместе с человеком, всегда желавшим упростить для себя эту «сложную» работу. Тысячелетиями для вычислений использовался счет на пальцах, с помощью ка-

мешков и разнообразных насечек. Затем появились узелковый счет в доколумбовой Америке, абак (глиняная пластинка с желобками, в которых размещались камешки) в Древнем Риме, русские счеты и другие аналогичные приспособления, целиком основанные на ручной работе человека. Очевидно, что говорить о высокой эффективности счета или параллельности в этот период не приходится.

Механический этап

Первой известной попыткой построения механизма, предназначенного для обработки числовой информации, является относящийся примерно к 1500 г. эскиз суммирующего устройства Леонардо да Винчи. К сожалению, построить по этому эскизу реальное счетное устройство не удалось.

Первое действующее устройство для выполнения сложения было создано в 1623 г. Вильгельмом Шиккардом. Он называл свое изобретение «суммирующими часами», так как оно было создано в единичном экземпляре на базе механических часов.

В 1641-1645 гг. Блез Паскаль разработал суммирующую машину, которая получила широкую известность и была выпущена целой серией в 50 машин, из них **8** экземпляров дошло до наших дней. В честь Паскаля назван один из популярнейших современных языков программирования.

Машина Паскаля могла выполнять только сложение и вычитание, а Готфриду Вильгельму Лейбницу в 1671-1674 гг. удалось построить ***арифмометр —*** машину для выполнения всех четырех арифметических операций (рис. 5.1, ***а).*** К заслугам Лейбница в этой области следует отнести также работы по развитию двоичной системы счисления — основного «языка» всех современных компьютеров.

Базовыми элементами, на которых основано действие машин Шиккарда, Паска- ля, Лейбница и множества последовавших за ними счетных устройств, были различные валики, зубчатые колеса и т. д., совершавшие в процессе вычислений механические перемещения.

> Основное устройство, на котором реализована аппаратура компьютера, называется его элементной базой.

Можно считать, что элементной базой машин этого периода является ***зубчатое колесо***, которому принадлежит одна из ведущих ролей в осуществляемых на этих машинах вычислениях.

Машины Паскаля и Лейбница являются дальними предками компьютера. Но в отличие от компьютера, их работа еще не была автоматической. Эти машины, а также широко использовавшиеся в XIX и в начале XX в. арифмометры (рис. 5.1, б и в) и нынешние микрокалькуляторы характеризуются тем, что человек непосредственно участвует в вычислительном процессе на всех его этапах. В частности, человек не только определяет последовательность выполняемых действий, но и напрямую управляет вычислениями.

Машины Чарльза Бэббиджа

В ходе промышленной революции XVIII-XIX вв. появились и стали широко использоваться бумажные ленты с отверстиями — ***перфоленты*** и листы из плотного картона с отверстиями — ***перфокарты*** (рис. 5.2), которые являются разновидностями долговременных носителей информации. С помощью определенных комбинаций отверстий на перфолентах и перфокартах задавался конкретный план работы различных устройств. Характерным примером такого рода устройств

является ткацкий станок, изобретенный Жозефом Жаккаром во Франции в 1801-1808 гг. Наличие или отсутствие отверстия в перфокарте, управлявшей работой станка, заставляло подниматься или опускаться нить при одном ходе челнока. Станок Жаккара был первым массовым промышленным устройством, автоматически работающим по заданному плану.

После появления автоматических ткацких станков естественным образом должна была возникнуть мысль о том, что машине можно поручить не только изготовление тканей. По-видимому, можно попытаться поручить ей и арифметические вычисления, потребность в выполнении которых в связи с бурным ростом мореплавания была в то время очень велика. Такая мысль возникла у английского математика, профессора Кембриджского университета Чарльза Бэббиджа. В 1822 г. он опубликовал статью с описанием так называемой ***разностной*** машины, предназначенной для автоматического вычисления и печати таблиц математических функций, используемых в морской навигации. Позже эта машина была построена и довольно долго и успешно работала. Фактически разностная машина «умела» выполнять только один алгоритм табулирования (то есть построения таблицы значений) полиномов до 4-й степени методом конечных разностей с точностью до 15 десятичных знаков.

Затем Бэббидж начал работать над проектом машины, которую впоследствии стали называть ***аналитической.*** По замыслу Бэббиджа, эта машина должна была «уметь» самостоятельно решать произвольные задачи с привлечением всех арифметических операций. Эта идея полностью исключала участие человека в вычислительном процессе, сводя его роль к подготовке необходимых числовых данных и,как и в случае с ткацким станком Жаккара, составлению плана выполнения вычислений, зафиксированного в некоторой специальной форме на перфокарте. Собственно процесс обработки информации должен был выполняться автоматически по заданной программе. Первый эскиз этой машины появился в 1834 г.

Машина Бэббиджа должна была содержать «склад», то есть память, из 1000 ячеек по 50 десятичных разрядов в каждой, вычислительное устройство — «мельницу», по терминологии Бэббиджа, а также устройство ввода обрабатываемых данных, которые планировалось хранить на перфокартах, и устройство вывода результатов на печать и на перфоратор. Программа обработки данных также должна была считываться с перфокарт.

Несмотря на несколько десятилетий работы и затраченные усилия, Бэббиджу не удалось реализовать свою идею, в основном из-за несовершенства технической базы того периода. Точность, с которой нужно было изготавливать детали этой машины, в то время была еще недостижима.

Опередивший свое время проект машины Бэббиджа содержал все основные компоненты вычислительных машин, появившихся почти через столетие после его работ. Основные идеи Бэббиджа не были забыты, они сыграли важную роль в дальнейшем развитии средств обработки информации.

Несмотря на то что аналитическая машина Бэббиджа существовала только в виде проекта, для нее была составлена первая в мире программа. В 1843 г. Ада Лавлейс, дочь английского поэта Джорджа Байрона, опубликовала работу, в которой были заложены основы современного программирования. Ею же для машины Бэббиджа была составлена программа вычисления чисел Фибоначчи. Впоследствии в ее

честь был назван алгоритмический язык Ада, один из самых мощных и сложных современных языков программирования.

Электромеханический этап

Следующий этап в развитии средств вычислений связан с использованием так называемых табуляторов (от лат. tabula — доска, таблица), которые представляют собой устройства для считывания и простейших видов обработки данных, нанесенных на перфокарты. Отличительной особенностью табулятора является неизменность алгоритма обработки данных, который определяется его конструкцией.

Первый табулятор был создан Германом Холлеритом в 1887 г. Основой этого устройства являлись простейшие ***электромеханические реле*** (см. 3.2.1), которые составляли элементную базу вычислительных устройств в течение последующих 50-60 лет.

Табуляторы широко использовались для выполнения расчетов статистического характера, например для проведения переписи населения в конце XIX в. в США, Канаде, России и в некоторых других странах. Для производства табуляторов Г. Холлерит в 1897 г. организовал фирму Tabulating Machine Company (компания по производству табуляторов), которая впоследствии преобразовалась в фирму IBM (от International Business Machines corporation — корпорация «Международные коммерческие машины»). В настоящее время эта компания является одним из мировых лидеров в сфере компьютерного производства.

В 30-х гг. XX в. в разных странах начались разработки принципиально иных устройств — программно-управляемых релейных вычислительных машин. Одна из первых таких машин под названием Z-3 была создана Конрадом Цузе в Германии в 1939-1941 гг. В ее конструкцию входило 2600 реле. Она могла «помнить» до шестидесяти четырех двадцатидвухбитовых чисел. В машине Z-3 использовалась одноадресная система команд. Сложение выполнялось за 0,3 с, а умножение — за 5 с. Предусматривались клавишный ввод данных и вывод результатов на световое табло.

Однако возможности и этой, и созданной позднее более совершенной модели Z-4 по составлению программ были довольно скромными. В частности, не было возможности осуществлять программный выбор одного из нескольких возможных вариантов действий. Это не позволяет считать Z-3 универсальной вычислительной машиной.

Полностью идеи Чарльза Бэббиджа впервые были реализованы в машине «Марк 1» (рис. 5.3, ***а***), разработанной в фирме IBM под руководством Говарда Айкена в 1937-1944 гг. Машина работала в десятичной системе счисления. Ее память состояла из 72 ячеек по 23 разряда каждая. Программа работы машины задавалась с помощью различных коммутационных устройств, которые соединялись, разъединялись и переключались вручную. Весила машина 5 т, а ее работу обеспечивало устройство мощностью 5 лошадиных сил.

«Марк 1» считается первой в мире программно-управляемой универсальной вычислительной машиной. Вместе с тем устройство для выполнения арифметических действий в ней было чисто механическим. Затем в 1947 г. была построена ***полностью электромеханическая*** машина «Марк 2». Она выполняла одну операцию умножения за 0,7 с.

Характерная для электромеханических машин скорость обработки данных не

удовлетворяла потребностям того периода. Так, например, самая быстродействующая в мире релейная машина РВМ-1, которая была построена в 1950-х гг. в СССР под руководством Н. И. Бессонова, выполняла операцию умножения только за 0,05 с, что соответствует выполнению 20 операций в секунду. Машина РВМ-1 была всего в 14 раз быстрее, чем «Марк 2». Дело в том, что механические перемещения — неотъемлемая часть реализации вычислительных операций в механических и электромеханических машинах — существенно ограничивали их быстродействие. Только полностью электронные, то есть исключающие механические перемещения в процессе вычислений (и, следовательно, безынерционные) устройства могли решить проблему быстродействия вычислительных машин.

Начало электронного этапа

Начало важнейшего на сегодняшний день электронного этапа в развитии средств обработки информации относится к 40-м гг. XX в. Элементной базой вычислительных систем в этот период стал ***триггер*** (см. 3.3), который может рассматриваться как ***электронное реле —*** функциональный эквивалент электромеханического реле. Триггер был изобретен М. Бонч-Бруевичем в 1913 г. на базе разработанного в 1906 г. триода — электровакуумной лампы накаливания с тремя выходными контактами.

В 1937-1942 гг. в США под руководством Дж. Атанасова и К. Берри была сконструирована первая полностью электродная машина ABC (от Atanasoff — Berry Computer), содержавшая около 600 электронных ламп накаливания. Но эта машина могла выполнять только операции сложения и вычитания и, к сожалению, так и не стала действующей. Характерными ее отличиями были применение двоичной системы счисления, а также периодически подзаряжаемые емкостные электрические элементы — конденсаторы, используемые для хранения данных. Этот принцип лежит в основе так называемой ***динамической*** разновидности современной оперативной памяти (см. 8.1).

Первой в мире работающей электронной цифровой машиной стал специализированный компьютер COLOSSUS, который с 1943 г. использовался англичанами для дешифровки радиосообщений, пересылаемых на немецкие подводные лодки. В создании этого компьютера принимал участие математик Алан Тьюринг, внесший значительный вклад в теоретическую информатику. В конструкцию машины входило 2000 электронных ламп.

Первая ***программно-управляемая универсальная электронная*** вычислительная машина была разработана в 1943-1945 гг. в Пенсильванском университете США под руководством Д. Моучли и Д. Эккерта. Эта машина называлась ENIAC (Electronic Numerical Integrator And Computer — электронно-цифровой интегратор и вычислитель) (рис. 5.3, ***б).*** Компьютер весил 30 т, его высота равнялась 6 м, ширина — 4 м, а общая площадь составляла 120 м^2. Машина состояла из 18 000 электронных ламп накаливания. Процессор машины содержал 20 регистров, способных хранить десятиразрядные десятичные числа. Он выполнял примерно 5000 арифметических операций в секунду (сравните с 20 операциями в секунду, выполняемыми электромеханической машиной РВМ-1). Программа для машины ENIAC задавалась вручную с помощью механических переключателей и гибких кабелей со штекерами, вставляемыми в нужные разъемы. Поэтому любые изменения в программе требовали много сил и времени.

В 1944 г. Д. Моучли и Д. Эккерт включились в разработку машины EDVAC (от Electronic Discrete Automatic Variable Computer — параметризуемый автоматический электронный вычислитель дискретного действия), работа над которой продолжалась до 1952 г. Машина работала в двоичной системе счисления, ее память содержала свыше 5000 ячеек по 44 бита каждая. Память была реализована на так называемых ртутных линиях задержки. Система команд у машины была четырехадресной. Сложение выполнялось за 0,001 с, а умножение — за 0,002 с. Отличительной особенностью этой машины было размещение программы внутри машины, в ее оперативной памяти.

Архитектура фон Неймана

Участник разработок машин ENIAC и EDVAC выдающийся математик Джон фон Нейман, анализируя в подготовленном им с соавторами отчете «Предварительный доклад о машине EDVAC» работу первых компьютеров, фактически заложил основы архитектуры, которая в той или иной форме до сих пор используется в подавляющем большинстве вычислительных систем. Базовые принципы, выдвинутые фон Нейманом:

- использование для кодирования программ и данных двоичной системы счисления;
- применение в конструкции машины электронной элементной базы;
- организация памяти в виде линейного адресного пространства;
- хранение выполняющейся программы и обрабатываемых данных внутри машины, в ее электронных схемах памяти, а не вне ее — на перфокартах, перфолентах или разъемах со штекерами;
- организация последовательного (естественного) порядка выполнения команд программы, что приводит к отказу от четвертого адреса в командах;
- организация параллельной, то есть одновременной обработки разрядов кодов данных;
- организация параллельной передачи по внутренним линиям компьютера всех разрядов кода.

Джон фон Нейман разработал собственный проект машины IAS (от Immediate Address Storage — память с прямой адресацией), которую сейчас принято называть фон-неймановской вычислительной машиной, а ее архитектура считается классической фон-неймановской архитектурой.

Память в машине фон Неймана, схема которой изображена на рис. 5.4, состоит из 4096 ячеек по 40 битов каждая. В одной ячейке размещается один 40-битовый код целого числа, или две 20-битовые команды. Система команд в машины IAS одноадресная. Код операции занимает 8 битов, а операнд — остальные 12 битов.

Последовательность обрабатываемых процессором кодов данных, выполняемых им команд и управляющих сигналов принято называть потоком данных, потоком команд и управляющим потоком соответственно.

В машине фон Неймана имеются потоки данных между оперативной памятью и устройством управления, между оперативной памятью и арифметико-логическим устройством, а также между устройствами ввода/вывода и арифметико-логическим устройством. При этом обмен данными между устройствами ввода/вывода и оперативной памятью происходит через процессор. Поэтому на время обмена вычисления по программе прекращаются. В машине IAS во внутренних линиях, а также в линиях связи с внешними устройствами данные и команды передаются в параллельном режиме.

Основу арифметико-логического устройства составляет единственный регистр, который называется аккумулятором. Типичная команда выбирает операнд из оперативной памяти и использует его в операции с аккумулятором, где при необходимости должен находиться второй операнд. Результат остается в аккумуляторе. Архитектура компьютеров, которая основана на этом принципе, называется аккумуляторной архитектурой.

Первой действующей машиной, в которой были реализованы базовые принципы фон-неймановской архитектуры, стал компьютер EDSAC (от Electronic Delay Storage Automatic Calculator — автоматический вычислитель с электронной памятью на линиях задержки), построенный М. Уилксом при участии Алана Тьюринга в Великобритании в 1949 г. В этой машине использовалась двоичная система счисления. Программа хранилась в оперативной памяти. Система команд была одноадресной. Тактовая частота машины составляла 0,5 МГц, то есть длительность такта равнялась 2 мкс. Но одна команда, занимая почти 5000 тактов, выполнялась в среднем за 0,01 с, что составляло примерно 100 арифметических операций за одну секунду. С машины EDSAC принято вести отсчет первого поколения компьютеров.

Развитие архитектуры первого поколения компьютеров происходило в основном в рамках фон-неймановской архитектуры. При этом наращивались количественные

показатели производительности компьютеров: увеличивались тактовая частота, длина машинного слова, скорость обмена у оперативной памяти. Повышалась плотность хранения информации, уменьшались размеры компьютеров. Появились новые долговременные носители программ и данных, такие как ***магнитные ленты***, ***барабаны*** и ***диски***, на которых научились хранить не только числовую, но и текстовую, звуковую, графическую информацию. Появились удобные средства для организации взаимодействия человека и машины, в том числе компактные и надежные клавиатуры, служащие для первичного ввода информации и управления работой компьютера, а также подобные телевизионным приемникам монохромные, а затем и цветные устройства для отображения информации — дисплеи. Так, например, первый дисплей с разрешением 512 х 512 пикселов появился в составе выпущенного в 1961 г. компьютера PDP-1.

В нашей стране первые компьютеры создавались примерно в тот же период. В 1947-1951 гг. под руководством академика С. А. Лебедева была пущена первая советская вычислительная машина — МЭСМ (малая электронно-счетная машина). Кроме того, выпускались машины «Стрела», «Минск», «Днепр», «Урал», БЭСМ (большая электронно-счетная машина), М-2, «Мир» и некоторые другие. Они разрабатывались под руководством крупных советских конструкторов и теоретиков И. С. Брука, М. А. Карцева, Б. И. Рамеева, В. М. Глушкова, Ю. А. Базилевского. Советская вычислительная техника того периода относилась к лидирующим в европейских странах.

Параллелизм в архитектуре начального периода

Практически сразу же после появления* компьютеров в их архитектуру стали внедрять параллелизм, затрагивавший в первый период развития в основном функционирование автономного компьютера.

Параллельная обработка разрядов кода

Оперативная память у машин EDSAC и EDVAC реализована ***на ртутных линиях задержки***, которые представляют собой тонкие герметичные металлические трубки с парами ртути внутри (рис. 5.5). На концах трубки находятся кристаллы кварца. По своим функциям они похожи на мембраны в телефонных аппаратах: один из кристаллов играет роль передающего, а другой — принимающего. К передающему кристаллу подсоединяется электрическое устройство, создающее колебания. Они, в свою очередь, возбуждают упругие колебания в парах ртути, которые с определенной временной задержкой доходят до другого конца трубки и заставляют колебаться приемный кристалл. Его колебания преобразуются в электрические импульсы и по цепочке обратной связи вновь подаются на передающий кристалл. Получается замкнутый контур, который в динамическом режиме сколько угодно долго сохраняет поступившие в трубку импульсы.

Биты, которые нужно сохранить в памяти, подаются в виде электрических импульсов на передающий кристалл. Замкнутый контур памяти принимает их и сохраняет в описанном ранее динамическом режиме. При необходимости выполнить чтение информация снимается с линии обратной связи.

По сравнению с реализацией памяти в виде триггеров на электронных лампах, такая система отличается большей плотностью хранения данных. Например, трубка длиной 1 м может сохранять до тысячи бит. Если на этой же площади разместить ламповые триггеры, то в них можно сохранить только десятки бит. Кроме того, память на ртутных линиях задержки дешевле, чем ламповая память.

Поэтому на ламповых триггерах были реализованы только самые ответственные узлы этих машин, в том числе их арифметико-логические устройства.

Однако память на ртутных линиях допускает только последовательную выборку записанных в нее битов. Следовательно, сложение, а также другие операции могут выполняться только поразрядно. Например, для сложения 16-битовых чисел требуется не менее 16 тактов работы процессора. Такая архитектура считается разрядно-последовательной.эша

В 1952 г. была пущена в эксплуатацию машина «Whirlwind 1», в которой оперативная память была построена на ферритовых магнитных сердечниках, представлявших собой колечки диаметром менее 1 мм. Материал, из которого они сделаны, обладает способностью длительное время сохранять одно из двух возможных состояний намагниченности (аналог северного и южного полюсов у магнита). Одно из этих состояний считается цифрой 0, а второе — цифрой 1 (рис. 5.6). Через каждое колечко проходит несколько проводов. Пропуская по ним ток нужных величины и направления, можно изменять состояние намагниченности или же определять текущее состояние, то есть осуществлять запись или чтение бита. Память на магнитных сердечниках оказалась гораздо компактнее, чем память на электронных лампах накаливания. Так, блок памяти объемом 1024 бита размещался на площади всего 12 х 12 см.

Отличительной особенностью памяти, построенной на ферритовых кольцах, является значительно более высокая скорость обмена, чем у памяти на ртутных линиях задержки. Машина «Whirlwind 1» установила рекордные по тому времени показатели быстродействия. Она выполняла в секунду 330 ООО операций сложения и 60 ООО операций умножения. С этого времени большинство компьютеров оснащались оперативной памятью на ферритовых кольцах. Ферритовая память была вытеснена только в конце 1960-х гг. памятью на транзисторах, а затем на интегральных схемах.

Тема 9: Многоуровневая организация памяти

Были рассмотрены структура и функции двух основных уровней памяти — регистров процессора и оперативной памяти. Упоминался еще и третий уровень, на котором находятся внешние запоминающие устройства, играющие роль информационных складов. Эти уровни выполняют в организации вычислений самостоятельные функциональные роли, и присутствие в составе компьютера по крайней мере первых двух уровней памяти является обязательным. Запоминающие устройства компьютера различных уровней существенно отличаются друг от друга своими базовыми характеристиками, такими как объем, скорость обмена и стоимость.

Этот набор характеристик внутренне противоречив. В самом деле, как правило, увеличение объема памяти сопряжено с увеличением стоимости и уменьшением скорости выборки, так как выбирать нужно из большего количества единиц памяти. Борьба за снижение стоимости приводит к ухудшению остальных характеристик.

Кроме противоречий между базовыми характеристиками одного и того же уровня, имеются противоречия между различными уровнями памяти, важнейшим из которых является существенная разница в их быстродействии, и по мере совершенствования аппаратных средств компьютера эта разница только нарастает.

Чтобы разрешить эти противоречия и повысить общую эффективность системы, память компьютера реализована в виде многоуровневой структуры, которая включает некоторое количество дополнительных уровней, сглаживающих разницу между соседними основными уровнями.

Кратко охарактеризуем свойства используемых в настоящее время уровней памяти компьютера.

□ Регистровая память процессора. Используется для промежуточного хранения данных в процессе их обработки. Энергозависимый вид памяти. Обладает наиболее высоким быстродействием, поскольку работает на тактовых частотах процессора. Время доступа составляет десятые доли наносекунд. Объем регистров — десятки и сотни машинных слов.

□Кэш-память. Дополнительный уровень памяти, играющий роль буфера между оперативной памятью и процессором. Служит для сглаживания разницы в скоростях их работы. Энергозависимый вид памяти. За исключением регистрового, это самый быстрый тип памяти, реализуемый на ***статических*** микросхемах памяти. Время доступа составляет единицы наносекунд. Объем кэша может достигать нескольких мегабайт. Отличается относительно высокой стоимостью и высоким энергопотреблением.

□ Оперативная память. Используется для хранения выполняющихся программ и необходимых им данных. Энергозависимый вид памяти. Время доступа составляет десятки наносекунд, а объем — несколько гигабайт. Более дешевая, чем кэш-память, так как реализуется с помощью динамических микросхем. Высокая надежность хранения данных и программ, расчетная вероятность ошибки — одна в десять лет.

□ Внешняя память — магнитные и оптические диски. Служит информационным складом. Самый медленный уровень памяти, время доступа порядка 30 ООО ООО не. Самая дешевая и самая емкая — объем современных дисков доходит до десятков терабайт.

Анализируя приведенный список, можно заметить, что уровни памяти компьютера

как бы образуют пирамиду, в основании которой находится внешняя память, а на вершине — регистровая память. Уровень, лежащий в основании, — самый медленный, но зато обладает огромным объемом. По мере продвижения к вершине пирамиды характерный объем уменьшается, а скорость обмена увеличивается, достигая максимального значения у регистрового уровня.

Кроме перечисленных выше уровней памяти в компьютере используются еще несколько разновидностей запоминающих устройств, которые играют, в общем-то, вспомогательную, но довольно важную роль. Это буферная, постоянная и полупостоянная память.

Буферная память используется в адаптерах, контроллерах, портах и т. д. для временного хранения данных в процессе ввода/вывода с целью организации асинхронной работы внешних устройств или сглаживания разницы в скоростях работы устройств, между которыми осуществляется обмен. Это энергозависимый вид памяти. Различные устройства имеют разную по скорости и объему буферную память. Самым высоким быстродействием, сравнимым с быстродействием кэш-памяти, и объемом до десятков мегабайт обладает буферная память контроллеров дисплеев.

Постоянная память (ПЗУ — постоянное запоминающее устройство, или ROM, от Read Only Memory — память только для чтения) используется для энергонезависимого хранения важной системной информации. В постоянной памяти всегда находится часть операционной системы, которая называется BIOS (от Base Input/Output System — базовая система ввода/вывода). BIOS — это набор программ проверки и обслуживания аппаратуры компьютера, который обеспечивает также выполнение простейших операции ввода с клавиатуры, вывода на дисплей и т. д. Постоянная память допускает только считывание. Типовое значение объема ПЗУ до 256 Кбайт при невысокой скорости обмена и времени доступа более 100 не. Для повышения производительности содержимое постоянной памяти копируется в оперативную, и в работе используется только копия BIOS, которую часто называют теневой памятью. В последние годы постоянная память вытесняется другими видами энергонезависимой памяти, такими как флэш-память (от flash — вспышка, мгновение).

Основными разновидностями постоянной памяти являются:

- PROM (от Programmable ROM — однократно программируемые ПЗУ). Эта разновидность памяти отличается тем, что для записи программ в микросхемы памяти используется специальное устройство. Модули памяти вынимаются из корпуса компьютера и вставляются в это устройство, в котором в модули заносится новое содержимое. Затем модули памяти вставляются назад в компьютер. Этот вид памяти считается устаревшим;
- EPROM (от Erasable Programmable ROM — перепрограммируемые ПЗУ). Микросхемы отличаются тем, что можно ***неоднократно*** стирать их содержимое и заносить новое. Удаление кодов из микросхем памяти осуществляется с помощью ультрафиолетового облучения в течение нескольких минут через специальные окошки, которые после облучения вновь заклеиваются. Перепрограммирование микросхем можно выполнять прямо на компьютере с помощью специально подключаемого устройства. Также устаревший вид памяти;
- EEPROM (от Electrically Erasable Programmable ROM — электрически стираемые перепрограммируемые ПЗУ). Стирание осуществляется прямо в компьютере с помощью высокого (порядка 30 В) напряжения;

□ FEPROM (от Flash EPROM), или просто флэш-память. Объединяет достоинства ROM и RAM, то есть допускает не только чтение, но и запись в обычном режиме работы компьютера, так же как и микросхемы DRAM, но при выключении электропитания содержимое памяти не уничтожается. В отличие от DRAM потребляет энергию не в течение всего времени работы, а только в периоды чтения/записи.

Полупостоянная память обеспечивает хранение относительно небольшого объема важных параметров конфигурации компьютера (характеристики процессора, системного диска и т. д.) и операционной системы даже при отключенном электропитании. Для поддержания внутреннего состояния такой памяти при отключенном электропитании используется батарейка или аккумулятор. Реализуется полупостоянная память с помощью CMOS-вентилей (от Complimentary Metal Oxide Semiconductor — комплементарный металл-оксид-полупроводник, соответствующее русское сокращение — КМОП). К полупостоянной памяти относится также CMOS RTC (от CMOS Real Time Clock — память часов реального времени) и ESCD (от Extended Static Configuration Data — расширенные статические данные конфигурации). Память CMOS RTC используется для хранения системных часов и календаря, а память ESCD — для хранения параметров устройств с автоматическим конфигурированием, которые обычно называют PnP-устройствами (от Plug and Play — вставь и работай).

Отметим, что под конфигурированием устройства понимается осуществляемый во время его подключения к компьютеру подбор параметров и режимов работы, номера прерывания и т. д., которые обеспечивают нормальное взаимодействие с остальными устройствами компьютера и с операционной системой. Такое конфигурирование, как правило, осуществляется один раз, а затем выбранные параметры должны сохраняться даже при отключенном электропитании. Объем полупостоянной памяти составляет несколько сотен байт, а время доступа к ней превышает 100 не.

Дополнительный материал по обсуждавшимся в главе вопросам можно найти в изданиях [18], [30], [34].

Контрольные вопросы и упражнения

1. Для чего потребовалась реализация многоуровневой подсистемы памяти компьютера?
2. Охарактеризуйте основные уровни памяти компьютера.
3. Охарактеризуйте вспомогательные уровни памяти компьютера.

Тема 10: Кэш память, механизмы работы кэша.

В любом случае взаимодействия с оперативной памятью процессор не может получить или передать данные со скоростью, превышающей возможности памяти. На современном уровне развития вычислительных систем скорость работы процессора в десятки и даже сотни раз выше скорости работы микросхем памяти. Поэтому во время обмена с оперативной памятью процессор теряет много тактов на ожидание данных из памяти.

В 1960-е гг. в составе компьютеров появилась так называемая сверхоперативная память, существенно повысившая общую производительность вычислительных систем. В персональных компьютерах аналогичную функциональную роль стал играть появившийся в 1985 г. высокоскоростной буферный уровень памяти, который назвали кэш-памятью (от cache — запас, тайник, наличные в кармане). Точнее всего способ использования этого уровня памяти характеризует сленговый перевод слова cache — карманные деньги, то есть находящиеся под рукой и которые быстро, в любой момент можно достать и использовать для любых нужд.

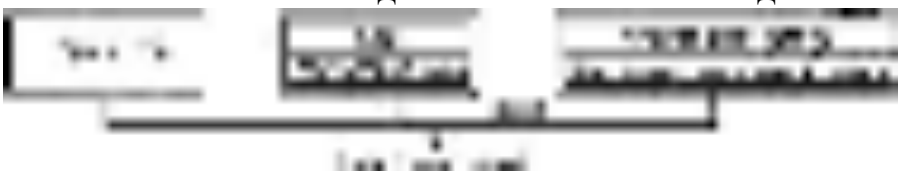


Рис. 9.1. Упрощенная схема взаимодействия процессора, кэша и оперативной памяти

Объем кэша значительно меньше, чем объем оперативной памяти, а скорость больше, чем у микросхем динамической памяти, и примерно такая же, как у процессора. В общем случае кэш помещается между оперативной памятью и процессором (рис. 9.1).

Экспериментально установлено, что после обработки некоторых данных процессор с большой вероятностью обращается к тем же самым данным или к находящимся в непосредственной близости от них. Это наблюдение, которое называется принципом пространственной локализации, уже обсуждалось при изучении пакетного режима функционирования микросхем памяти. Аналогом принципа пространственной локализации является также экспериментально полученный принцип временной локализации, который утверждает, что программе в ближайшее время, вероятнее всего, потребуются данные, которые недавно или только что были использованы.

С учетом принципов локализации данные, затребованные процессором, а также некоторая группа находящихся рядом кодов не только передаются в процессор, но и автоматически заносятся в кэш. Когда процессору нужны какие-либо данные, он вначале «заглядывает» в кэш «в надежде» обнаружить их там. Если нужные данные действительно находятся в кэше, то они с высокой скоростью передаются в процессор. Эта ситуация носит название попадание в кэш. Если данные в кэше отсутствуют, то говорят, что имеет место промах кэша, и процессор вынужден обращаться в оперативную память.

Если при попытке записи новых данных в кэш он оказывается до конца заполненным, необходимое для записи место освобождается за счет удаления из кэша некоторой группы уже находящихся в нем данных. Удаляться могут данные, к которым было меньше всего обращений, или, в соответствии с принципом

временной локализации, данные, которые дольше всего не использовались процессором. Применение описанного подхода приводит к тому, что по мере выполнения программы в кэше скапливаются наиболее часто используемые данные. Такое накопление принято называть кэшированием.

Очевидно, что при попадании в кэш происходит существенное ускорение работы системы. Количественно оценить общее ускорение от использования кэша можно следующим образом. Пусть время доступа к микросхемам оперативной памяти равно t_0, время доступа к микросхемам кэша — £к, h — доля попаданий в кэш от общего количества обращений к памяти. Тогда доля промахов в общем количестве обращений составляет 1 - А, а среднее время доступа при наличии кэша $t = t_0($ 1 - $K) + t_K h$. Например, для t_0 = 10 не, t_K = 1 не и h = 0,75 получим t = 3,25 не — более чем в три раза лучше, чем при отсутствии кэша.

Может показаться, что промахи кэша замедляют работу, так как процессор, безрезультатно «заглянув в кэш», делает лишнюю операцию. Однако эту проблему можно обойти, организовав параллельное обращение в оперативную память и в кэш. Если данные в кэше обнаружатся, то выборка из памяти прекращается досрочно. Если же данные в кэше отсутствуют, то выборка из памяти завершается стандартным образом. В любом случае дополнительные временные затраты очень малы.

Механизмы работы кэша

К настоящему времени разработано несколько различных вариантов организации кэша. Рассмотрим наиболее известные из них. В общем случае кэш состоит из области хранения данных, в которой находятся копии некоторых полей оперативной памяти — собственно кэш-памяти, или памяти данных (рис. 9.2), и области, которая содержит управляющую информацию в виде набора признаков, описывающих находящиеся в кэше копии. Эту область кэша называют памятью тегов (от tag — ярлык, этикетка, признак).

Кэш прямого отображения

Исторически первым и наиболее простым является кэш прямого отображения. Память для хранения копий делится на элементы, которые принято называть строками кэша. Строки кэша могут быть различной длины. Довольно часто используется длина 32 байта, соответствующая стандартно передаваемой из оперативной памяти в процессор группе байтов (пакету). Роль строки кэша состоит в том, что данные из оперативной памяти дублируются в кэш сразу целой строкой, которая содержит запрошенное процессором поле памяти. Таким образом, в кэш попадает не только копия требуемого поля, но и копия группы байтов, расположенных рядом с ним.

Количество строк в кэше зависит от имеющегося в компьютере физического объема кэша и от длины его строк. Так, например, кэш объемом 4 Кбайт может состоять из 128 тридцатидвухбайтных или из 256 шестнадцатибайтных строк. Встречаются и другие варианты. Каждой строке кэш-памяти однозначно соответствует элемент памяти тегов, который содержит значения рассматриваемых далее признаков этой строки.

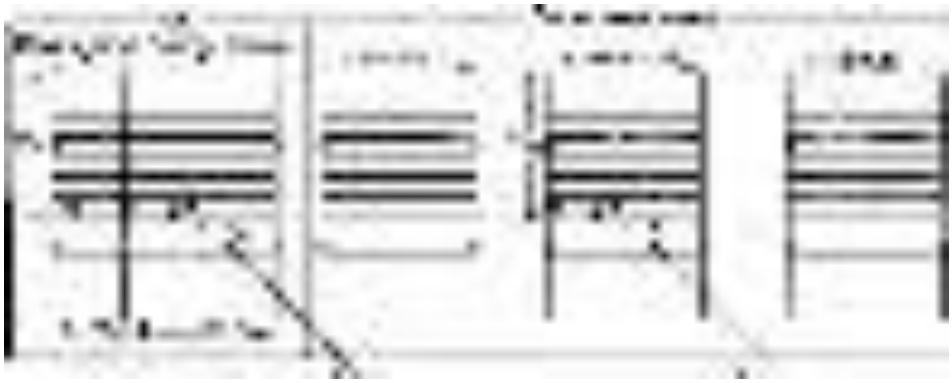
Для получения простого и быстродействующего механизма поиска контроллер кэша рассматривает всю имеющуюся оперативную память как совокупность страниц, которые равны по объему кэшу и также состоят из строк. Простота дублирования и поиска в кэше обеспечивается тем, что копия любой строки из любой страницы всегда занимает в кэше точно такое же положение, что и оригинал на соответствующей странице оперативной памяти. Это значит, что смещение копии

строки относительно начала кэш-памяти всегда равно смещению оригинала строки относительно начала ее страницы в оперативной памяти (рис. 9.2). Таким образом, страница оперативной памяти как бы полностью отображается на память данных в кэше — отсюда и название «кэш прямого отображения».

Каждая страница оперативной памяти имеет номер, который принято называть тегом. Когда копия строки из какой-либо страницы попадает в кэш, ее тег занимает одноименное поле соответствующего элемента памяти тегов. 71

Для определения положения запрошенного процессором байта (или поля памяти) контроллер кэша выделяет в его физическом адресе три участка. Находящиеся в этих участках коды считаются номером страницы оперативной памяти, номером строки на странице и номером байта в строке. Разрядность участков зависит от длины физического адреса, количества страниц и длины строк.

В качестве примера для обсуждения будем считать, что кэш объемом 4 Кбайт состоит из 128 тридцатидвухбитовых строк, а физический адрес состоит из 20 битов. Тогда для определения номера байта в строке нужно 5 битов ($2^5 = 32$), а для задания строки на странице — 7 битов ($2^7 = 128$). Из 20 битов адреса на номер страницы остается 8 битов, с помощью которых можно перенумеровать $2^8 = 256$ страниц. Таким образом, рассматриваемый кэш может использоваться для кэширования оперативной памяти объемом 4 Кбайт • 256 = 1 Мбайт.



разряды адреса-» 19 __ 1211 54 О

Адрес поля памяти $18A72_{16}$-IQQQ1 10Q0llIQ1Q 01111 QQToL

18 53 12

Тег (номер страницы) Строка Байт

. Кэш прямого отображения

Распределим разряды физического адреса следующим образом. В пяти младших битах (4-0) будем кодировать номер байта в строке, следующие 7 битов (11-5) займем номером строки, а последние 8 битов (19-12) пусть содержат номер страницы (рис. 9.2, ***внизу).*** Допустим, что процессору потребовался байт оперативной памяти с физическим адресом $18A72_{16}$ = 0001 1000 1010 0111 0010_2. После выделения контроллером кэша в этом адресе соответствующих участков $oooiiooo \mid 1oioo1i \mid 10010_2$ окажется, что нужный байт расположен на странице оперативной памяти с номером 18_{16} в строке с номером 53_{16}, а его номер в строке 12_{16}. Во время передачи байта из оперативной памяти в процессор вся содержащая его 32-битная строка попадает и в кэш, контроллер которого запишет ее копию в строку с номером 53_{16}. Номер страницы оперативной памяти (18_{16}), из которой была выбрана строка, заносится в поле тега соответствующего строке элемента памяти тегов.

Если теперь процессору потребуется какой-либо байт из оперативной памяти, то кроме обращения в память процессор направляет запрос и контроллеру кэша. Далее по описанной схеме контроллер находит номер строки кэша, в которой

может находиться нужный байт. Если тег строки кэша совпадает с номером страницы физического адреса, то обращение в оперативную память прекращается, а нужный байт передается в процессор из кэша — говорят, что имеет место попадание в кэш. Такая ситуация наблюдается, если, например, процессор запросит данные из физического адреса $18A74_{16}$. А вот для адреса $20A72_{16}$, который обращается к байту в строке с тем же номером 53_{16}, но на другой странице, тег 18_{16} не совпадает с ее номером 20_{16}. В этом случае имеет место промах кэша. Тогда процессор осуществит выборку из оперативной памяти. Кроме того, произойдет замена всей строки кэша с номером 53_{16}, а также соответствующего строке элемента памяти тегов. В таких случаях говорят, что произошло вытеснение (выталкивание) строки из кэша.

Основным достоинством кэш-памяти прямого отображения является высокая скорость определения попадания или промаха. Кроме того, весьма просто решаются основные задачи управления кэшем, в частности, задача вытеснения строки, когда кэш уже заполнен.

Многовходовый ассоциативный кэш

Кэш прямого отображения имеет и существенный недостаток. В него могут попасть только копии строк с разными номерами (или, что то же самое, с разными смещениями относительно начала страницы) из одной и той же или из разных страниц оперативной памяти. Копии строк, которые имеют один и тот же номер, но расположены на разных страницах, не могут одновременно находиться в кэше, так как при любой попытке обратиться к строке на другой странице копия строки из первой страницы заменяется в кэше. Например, в кэше не могут одновременно находиться копии строки с номером 53_{16} из 18_{16}-й и 20_{16}-й страниц памяти, так как обращение к 20_{16}-й странице вытеснит из кэша копию этой строки из 18_{16}-й страницы. Такая замена требует дополнительных временных расходов, и при частых заменах весь выигрыш от наличия кэша может исчезнуть, а то и заменится снижением общей эффективности из-за необходимости вместо байта или слова считывать из оперативной памяти каждый раз целую строку.

Размещение в кэше двух и более копий «однономерных» строк из разных страниц оперативной памяти допускается в так называемом ассоциативном кэше с множественным доступом, или многовходовом ассоциативном кэше. Иногда такой вариант организации называют наборно-ассоциативным кэшем. Количество копий «однономерных» строк из разных страниц, которые могут быть помещены в кэш, может отражаться в названии кэша: я-входовый (или я-связный) ассоциативный кэш, например двухвходовый кэш. Понятно, что я-входовый кэш допускает наличие в нем ***n*** копий «однономерных» строк. В настоящее время наиболее распространены четырех- и восьмивходовые ассоциативные кэши.

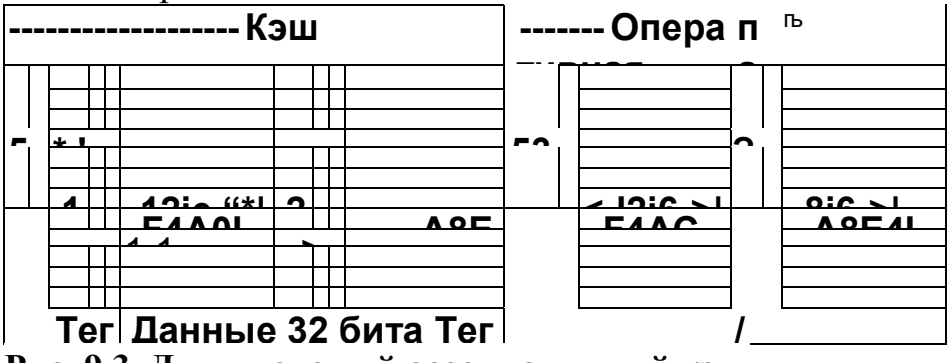


Рис. 9.3. Двухвходовый ассоциативный кэш

На рис. 9.3 изображен двухвходовый ассоциативный кэш. По сути дела он представляет собой два экземпляра кэша прямого отображения, которые принято называть банками кэша. Каждый из банков ассоциативного кэша может содержать копию строки с одним и тем же номером, но из разных страниц памяти. Кроме того, допускается независимый и одновременный просмотр тегов из разных банков. В изображенной на рисунке ситуации в кэше находятся копии двух строк с номером 53_{16}: одна из 18_{16}-й, а вторая — из 20_{16}-й страницы.

Ассоциативная память

Отметим, что ассоциативный кэш является разновидностью ассоциативной памяти. Выборка кода из обычной памяти, которую называют также адресуемой, осуществляется с помощью задания адреса, никак не связанного (не ассоциирующегося) со значением кода. В ассоциативной памяти выборка кода связана, т. е. ассоциируется (отсюда и название разновидности памяти) с его значением или со значением некоторой его части, которую принято называть ключом поиска, или просто ключом.

В общем случае в ассоциативной памяти каждый записываемый код сопровождается дополнительным уникальным (то есть не повторяющимся нигде в другом месте) кодом ключа. Можно даже считать, что ключ, в отличие от адреса, является неотделимой составной частью кода данного. Например, в случае ассоциативного кэша таким ключом поиска является тег страницы, сопровождающий копию любой строки в кэше. Для выполнения чтения значение ключа кода в памяти сравнивается с заданным значением ключа поиска. В случае совпадения сравниваемых значений код считывается из памяти.

Реализуется ассоциативная память значительно сложнее, чем адресная память, поскольку необходимо организовать аппаратный просмотр некоторого количества полей, занятых ключами. Такой просмотр может быть организован как последовательный, когда коды по очереди сравниваются друг с другом, или как параллельный, когда выполняется поразрядное сравнение сразу для всех ключей. Понятно, что второй способ выполнения поиска гораздо быстрее первого, но микросхемы памяти получаются более дорогими. Ассоциативная память используется в основном для решения задач, требующих быстрого поиска в относительно небольших объемах данных.

Управление ассоциативным кэшем

Для выявления наличия нужных данных в ассоциативном кэше уже недостаточно анализа физического адреса и сравнения тега с номером строки. В нем следует просмотреть ***n*** элементов памяти тегов в ***n*** различных банках кэша, причем лучше организовать их параллельный просмотр. Если тег в одном из банков совпадает с номером страницы из физического адреса, то поиск завершается попаданием в кэш. Если нет совпадения ни в одном из банков, фиксируется промах кэша.

Сложности возникают также при определении замещаемой строки в случае, когда нужная строка занята во всех банках. Пусть, например, в изображенной на рис. 9.3 ситуации процессором вызывается строка 53_{16} из страницы с номером 21_{16}. Оба банка заняты, поэтому нужно решить, какую из копий строк (из 18_{16}-й или из 20_{16}-й страницы) следует заменить копией строки из 21_{16}-й страницы. При использовании кэша прямого отображения такой проблемы нет занятая позиция кэша перезаписывается при любом новом поступлении на эту позицию. В многовходовом ассоциативном кэше кандидатов на вытеснение несколько, и желательно выбрать вытесняемую копию так, чтобы минимизировать количество

промахов кэша. Для такого выбора применяется несколько правил, которые обычно называют алгоритмами замещения. Чаще всего используется алгоритм LRU (от least recently used — наиболее давно использованный), в соответствии с ним из кэша вытесняется строка, к которой дольше всего не было обращений. Для его реализации может быть, например, составлен связный список копий «однономерных» строк из разных банков. При каждом обращении в кэш этот список модифицируется так, что в его начало попадает наиболее часто (а в конец — наименее часто) используемый элемент. При возникновении необходимости из кэша вытесняется последний элемент списка. Существуют и другие варианты реализации этого алгоритма.

Многоуровневый кэш

В современных вычислительных системах кэш, как правило, имеет сложную многоуровневую структуру. Это значит, что в состав системы включается несколько различающихся объемом и быстродействием устройств, выполняющих функции кэша.

Самый быстродействующий, но при этом самый маленький по объему уровень кэша встраивается в микросхему процессора. Он обозначается L1 и называется внутренним, или первичным, кэшем. В настоящее время внутренний кэш обычно реализуется в виде двух независимых банков объемом 8-32 Кбайт каждый. Один банк внутреннего кэша используется только для хранения кодов данных, а второй — только для хранения кодов команд. Такую структуру называют разделенной, или гарвардской, архитектурой кэша, поскольку впервые она появилась в машине «Марк 3», созданной Говардом Айкеном в Гарвардском университете. Раздельная структура кэша имеет ряд преимуществ. Во-первых, банки кэша допускают одновременное и независимое выполнение операций считывания/за- писи через отдельные каналы доступа к оперативной памяти. Это примерно вдвое увеличивает пропускную способность кэша. Во-вторых, разделенный кэш допускает упрощенную организацию банка с кодами команд, поскольку во время выполнения программы они обычно не меняются, в то время как находящиеся в кэше коды данных могут изменяться.

Отметим, что в связи с включением в микросхему процессора устройств, не связанных с его основными функциями, введено понятие ядро процессора, к которому относится только часть его микросхемы, содержащая арифметико-логическое устройство, устройство управления, регистры и т. д., то есть те элементы, которые и составляют собственно процессор. Любые дополнительные элементы,

включаемые в микросхему процессора для повышения эффективности системы, такие как, например, кэш первого уровня, не входят в ядро процессора.
Следующий уровень кэша обозначается L2 и называется внешним, или вторичным. Он реализуется в виде отдельной микросхемы статической памяти объемом 256-512 Кбайт. Если в системе используется всего два уровня кэша, то внешний кэш может иметь объем 2-4 Мбайт. Обычно второй уровень кэша содержит одновременно как коды команд, так и коды данных. Такую структуру называют объединенной, или принстонской, архитектурой кэша.
Довольно часто компьютеры имеют еще один внешний кэш — кэш третьего уровня L3. Его объем доходит до 16-32 Мбайт, а быстродействие ниже, чем у кэша первого и второго уровней, но все-таки существенно выше, чем у микросхем динамической оперативной памяти. Обычно содержимое кэша первого уровня целиком находится в кэше второго уровня. А при наличии кэша третьего уровня содержимое двух верхних уровней целиком помещается в третий.
Управлять многоуровневым кэшем значительно сложнее, чем одноуровневым, но получаемое от увеличения количества уровней повышение общей эффективности вычислительной системы привело к повсеместному использованию такой архитектуры.
Когерентность кэша
Выполнение операций считывания кодов при наличии кэша не вызывает особых проблем. Сложности начинаются при выполнении операций записи. Они вызваны наличием нескольких копий (хотя бы двух) одного и того же кода в разных местах. С течением времени одна из копий кода может быть обновлена, заменена другим значением. Тогда возникает несовпадение находящихся в разных местах значений одной и той же величины.
Вообще говоря, это общая для всей информатики (и не только информатики) проблема хранения информации. Данные приходится дублировать в разных случаях и с разными целями. Например, в базах данных для обеспечения надежности хранения создается несколько архивных копий базы, которые к тому же принято размещать в разных местах. При необходимости внести в базу какие-либо изменения должны быть заменены все ее копии, иначе нарушается целостность данных. Задача поддержания одинаковых значений у кодов, находящихся в оперативной памяти и в кэше, называется проблемой когерентности[3] (достоверности, целостности) кэша.
Когда исходный код находится в оперативной памяти, а его копия помещена в кэш, возможны два варианта нарушения когерентности. Во-первых, ко многовходовой оперативной памяти может обращаться не только процессор. Например, независимо могут выполнять операции записи в память устройства Bus Mastering (жесткие диски по каналам контроллера DMA). В процессе такой записи изменяется только значение кода в оперативной памяти, а его копия в кэше остается в неизменном виде. Во-вторых, процессор может изменить значение в кэше, а код в оперативной памяти при этом оставить без изменения. Для регистрации описанных вариантов несоответствия в элементах памяти тегов предусмотрены флаг достоверности **V** (от validity — действительность) и флаг модификации **M** (от modify — изменение). Флаг **V**

[3] Аналог физического понятия когерентности (от лат. cohaerentia), которое означает согласованное протекание во времени нескольких колебательных или волновых процессов. В данном случае имеется в виду согласованное изменение значений кода и всех его копий.

принимает значение 1 при записи копии строки в кэш и сбрасывается в **0,** если изменился соответствующий код в оперативной памяти. Флаг **M,** наоборот, принимает значение **0** при записи копии строки в кэш, а устанавливается в **1** при изменении значения в кэше.

В связи с введенным уточнением структуры элемента памяти тегов отметим, что для выборки кода из кэша (чтения кода) оказывается недостаточно только совпадения тега и выделенного из физического адреса номера страницы. Дополнительно требуется, чтобы флаг достоверности **V** был равен 1, сигнализируя о том, что находящийся в кэше код действителен, то есть совпадает с оригиналом в оперативной памяти. Если это не так, процессору следует выбирать код из оперативной памяти.

Существует несколько способов поддержания когерентности кэша. В схеме пря**мой** записи измененный процессором код заменяется в кэше и в оперативной памяти одновременно. Это надежный вариант, в котором несоответствие кодов не возникает. Вместе с тем его использование снижает быстродействие системы.

Более высокой эффективностью обладают схемы, которые выполняют запись в оперативную память с некоторой задержкой во времени относительно момента записи в кэш. При использовании **обратной** записи, которую называют также **буферизованной сквозной записью,** значение передается в память в первом же свободном такте работы процессора. А в схеме **отложенной** записи передача измененного кода в оперативную память выполняется только при окончательном заполнении кэша, то есть когда для помещения в кэш нового значения не оказывается свободной области. При этом в соответствии с алгоритмом LRU в оперативную память переносится наименее часто используемая строка кэша.

Схемы обратной и отложенной записи более эффективны, так как не требуют при выполнении каждой операции записи обращения к оперативной памяти. Но они и более сложны, так как нужно уметь определять, следует ли выталкиваемую из кэша строку переписывать в оперативную память. Для экономии времени строки, флаг **M** которых равен 0, в оперативную память не дублируются. Кроме того, нужно заботиться о поддержании соответствия содержимого кэша и основной памяти. При использовании этих схем существуют довольно длительные интервалы времени, в течение которых коды в памяти и кэше не когерентны. Следует исключить каким-либо способом доступ к некогерентным данным в течение этих интервалов.

Еще более сложной является проблема когерентности в многоуровневом кэше, в котором необходимо заботиться сразу о нескольких разноуровневых копиях одного и того же кода, а также в многопроцессорных системах, когда каждый из входящих в систему центральных процессоров имеет собственный кэш и потому может создать собственную копию одной и той же строки из оперативной памяти. Любой из процессоров может изменить значение кода в принадлежащем ему кэше, но копии кода в чужих кэшах ему недоступны. Так возникает еще одна причина появления некогерентных данных.

Тема 11: Шины, циклы шин, многошинная архитектура.

Организация связи между устройствами компьютера по единственной общей шине оказалась довольно дешевым и простым вариантом архитектуры компьютера. Но одна шина могла удовлетворительно справляться с обменом до тех пор, пока быстродействие процессора, микросхем памяти, магнитных дисков и других устройств было относительно низким. По мере роста тактовых частот и объемов передаваемых данных общая шина компьютера стала «узким местом» архитектуры. В связи с этим начались работы по улучшению эффективности шин компьютера.

Основными техническими характеристиками шин являются разрядность, тактовая частота и производительность (скорость обмена, или пропускная способность[4]). Напомним, что разрядность определяется количеством проводов шины, по каждому из которых передается один бит адреса, один бит данных или один управляющий бит. Тактовая частота определяется как количество управляющих работой шины синхроимпульсов в секунду. Производительность равна количеству битов (или байтов), передаваемых по шине в единицу времени. Если, например, 16-битовая шина работает на частоте 100 МГц и выполняет передачу кода за один такт, то ее производительность равна 2 байта • 10^8 тактов/с * 190 Мбайт/с. Не следует путать пропускные способности шины и памяти. Это согласованные, но не обязательно совпадающие величины.

Один из возможных путей повышения производительности шины состоит в увеличении ее разрядности и тактовой частоты. Повышение разрядности приводит к увеличению геометрических размеров шины и, как следствие, к увеличению общей стоимости компьютера. Для уменьшения стоимости и размеров шины были разработаны различные варианты **мультиплексных** шин, в которых нет отдельных линий данных и линий адреса. Коды адресов и данных в таких шинах передаются по одним и тем же линиям. Сначала по шине передается адрес передаваемого кода, затем по тем же самым проводам передаются его биты. Такие шины, однако, обладают относительно невысокой производительностью.

Увеличение тактовой частоты шины может нарушить ее совместимость со старыми устройствами и с существующим программным обеспечением. Кроме того, это увеличение может привести к так называемому **перекосу** шины, который возникает при передаче данных по разным линиям шины с различной скоростью. В силу ряда физических факторов перекос шины вносит ошибки в передаваемые по ее линиям коды, а также является причиной аппаратных сбоев, возникающих при передаче по ней управляющих сигналов. Повышение частоты вызывает особенно сильные помехи в многоразрядных шинах, поэтому одновременно увеличивать и разрядность шин и их частоту невозможно.

Таким образом, разработчикам шин для увеличения их эффективности приходится решать множество задач, которые выдвигают взаимно противоречивые требования. Компромиссные варианты построения шин были получены на пути организации параллельности и конвейеризации в работе шины. Заметим, что в последнее время

---

[4] Если быть точным, то пропускная способность — это максимально возможная скорость обмена. Реальная скорость обмена всегда ниже пропускной способности.

В частности, существуют циклы чтения, циклы записи, циклы прерывания и некоторые другие типы циклов шин. Некоторые разновидности циклов рассматриваются в дальнейшем.

наметилась тенденция к замене шин с большой разрядностью на шины с меньшей разрядностью, но с более высокой частотой.

Циклы шин

Как было выяснено ранее, работой шины управляют ее ***контроллер*** и ***арбитр.*** Существуют различные конструктивные варианты реализации управляющих функций этих устройств. В частности, функции арбитража могут быть возложены на отдельную микросхему, они могут быть также переданы процессору или же закреплены за контроллером шины.

Шинам компьютера приходится выполнять несколько различных функций. Поэтому каждый вид работы определенным образом стандартизируется в виде отдельного **цикла шины.**

> Направленная на выполнение определенной функции шины периодически повторяемая связная последовательность управляющих работой шины сигналов, а также передач кодов по ее линиям называется циклом шины.

По отношению к тактовым синхроимпульсам шины делятся на ***синхронные*** и ***асинхронные.*** У синхронных шин все действия цикла шины привязаны к определенным фазам синхроимпульса — началу, середине, концу и т. д. Они имеют строго определенную длительность, а цикл такой шины всегда занимает целое количество тактов. У асинхронных шин такая привязка отсутствует.

Цикл чтения синхронных шин

Рассмотрим изображенную на рис. 10.1 упрощенную временную диаграмму цикла чтения шины. В некоторый момент времени $£_a$ процессор выставляет на адресные линии шины адрес байта, который нужно прочитать в оперативной памяти. Так же, как в описанной в 7.3 диаграмме цикла памяти, на стабилизацию адреса уходит некоторое время. Когда адрес стабилизируется, процессор формирует нулевое значение управляющего сигнала MEMR# (от memory read — чтение из памяти). Этот сигнал сообщает всем присоединенным к шине устройствам, что процессор запрашивает данные из оперативной памяти. Напоминаем: знак # в названии управляющего сигнала означает, что активным, то есть приводящим в действие, является его нулевое значение.

Синхроимпульсы тактового генератора

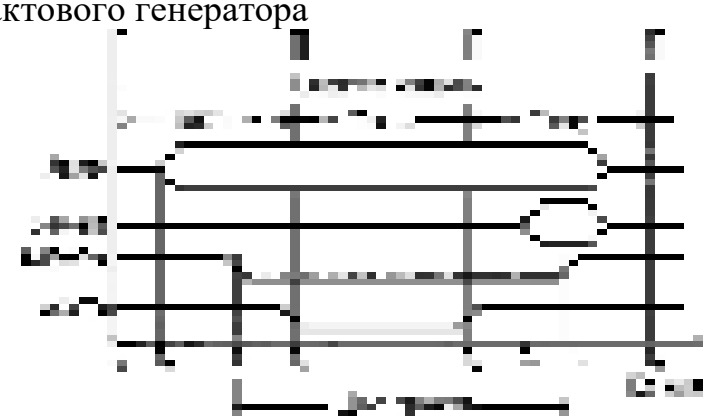


Рис. 10.1. Цикл чтения синхронной шины с одним тактом ожидания

В момент t_T стабилизации указанного сигнала контроллер оперативной памяти получает по шине сигнал о том, что есть запрос на чтение из памяти, а также адрес требуемого байта. С этого момента начинается цикл памяти — контроллер вместе с микросхемой начинают формировать ответ.

Предположим, что ответ (затребованный из оперативной памяти код) стабилизируется на шине данных только к моменту t_y в третьем такте цикла, то есть цикл памяти длится от t_T до £$_v$. Следовательно, в течение второго такта все устройства, участвующие в обмене, должны ждать получения ответа.

Контроллер памяти «знает» временные характеристики микросхем памяти. Чтобы процессор напрасно не ждал ответа и мог заняться выполнением других действий, **контроллер формирует на шине управляющий сигнал WAIT# (от wait — ожидание), занимающий столько тактов, сколько должен ждать ответа процессор. В ситуации, изображенной на рис. 10.1, сигнал WAIT# длится один такт.**

В момент времени t_d биты передаваемого кода появляются на шине, и после того как к моменту t_y закончится период их стабилизации, процессор переводит управляющий сигнал **MEMR#** в неактивное состояние. После окончания этого такта шина готова к выполнению нового цикла чтения или записи. Цикл записи не имеет принципиальных отличий от цикла чтения и потому здесь не рассматривается.

Обсуждаемый цикл шины занимает три такта. Если микросхема памяти обладает меньшим быстродействием и цикл памяти длится дольше, то цикл шины может занять четыре и более тактов процессора. Если микросхема памяти работает быстрее и цикл памяти укладывается в один такт, то цикл шины может занять всего два такта, так как вводить такт пропуска с помощью сигнала **WAIT#** не нужно. Еще более быстрые микросхемы памяти при рассмотренной структуре цикла шины не приведут к дальнейшему улучшению эффективности, так как цикл с такой структурой в любом случае не может занимать меньше двух тактов.

Преимуществом синхронных шин является более простая и дешевая их реализация. Поэтому они распространены довольно широко. Вместе с тем у синхронных шин имеется ряд недостатков. К ним относится снижение общей эффективности передач, вызванное тем, что цикл шины может занимать только целое количество тактов. Пусть, например, процессор и микросхема памяти способны закончить обмен за **2,1** такта, но так как цикл шины может занимать только целое количество тактов, он займет три такта. Следовательно, процессор и память потеряют 0,9 такта на вынужденное ожидание при выполнении каждой операции обмена, то есть общее снижение эффективности за счет этого простоя составляет примерно 43 %. Следует также упомянуть вынужденную подстройку быстродействия шины к скорости самого медленного из подсоединенных к ней передающих устройств.

Цикл чтения асинхронных шин

Для того чтобы избавиться от отмеченных недостатков синхронных шин, были разработаны асинхронные шины, у которых привязка к тактовым синхроимпульсам отсутствует. Любое действие выполняется столько времени, сколько ему фактически требуется. Один и тот же цикл шины для различных пар устройств может иметь различную длительность.

Рассмотрим изображенную на рис. 10.2 упрощенную временную диаграмму цикла чтения для асинхронной шины. Кроме использованного ранее сигнала **MEMR#,** асинхронной шине требуется еще два управляющих сигнала — **BEG#,** роль которого сводится к оповещению «заинтересованного» устройства о том, что оно может начать свое действие, и сигнал **FIN#,** который отправляется устройством

после завершения его части работы. Эти два сигнала в цикле шины играют своеобразную синхронизирующую роль.

Рис. 10.2. Цикл чтения асинхронной шины

Для пояснения причинно-следственных связей в цикле шины на временной диаграмме асинхронной шины используются стрелки, начинающиеся от графика сигнала (адреса, данных), который вызывает действие, и заканчивающиеся у графика устройства, которое это действие исполняет.

Пусть, как и ранее, в момент времени t_0 процессор выставляет на шину адрес требуемого байта. Когда он стабилизируется, процессор формирует нулевое значение управляющего сигнала **MEMR#** — на рисунке для отражения этой связки поясняющая стрелка идет от графика **Адрес** к графику **MEMR#** (далее пояснения по поводу стрелок отсутствуют).

После стабилизации указанного сигнала формируется сигнал **BEG#,** извещающий оперативную память о том, что она может начинать чтение запрошенного байта. Фактически цикл памяти начинается в момент t_v когда сигнал **BEG#** стабилизируется. После стабилизации кода на шине данных и окончания цикла контроллер памяти устанавливает сигнал **FIN#,** извещая тем самым об окончании своей части действий. Стабилизация сигнала **FIN#** вызывает отключение сигналов **BEG#** и **MEMR#,** а также освобождение адресных линий. Чтобы привести систему в исходное состояние, позволяющее начать новый цикл, нужно убрать сигнал **FIN#.** Для этого используется фаза отключения сигнала **BEG#,** которая одновременно вызывает освобождение линий данных, что означает завершение цикла шины в момент t_c.

Цепочка причинно-следственных связей цикла асинхронной шины в краткой форме выглядит следующим образом.

1. Стабилизация адреса вызывает формирование сигнала **MEMR#.**
2. Стабилизация сигнала **MEMR#** вызывает формирование сигнала **BEG#.**
3. Стабилизация сигнала **BEG#** запускает цикл памяти.
4. Стабилизация данных (завершение цикла памяти) вызывает формирование сигнала **FIN#.**
5. Стабилизация сигнала **FIN#** вызывает отключение сигналов **BEG#, MEMR#,** а также адресных линий.
6. Отключение сигнала **BEG#** вызывает отключение сигнала **FIN#** и освобождение линий данных.

Такая последовательность взаимосвязанных сигналов называется **полным квитированием** (от итал. quitanza — квитанция[5]), так как каждое участвующее в обмене устройство как бы посылает и принимает ***квитанцию***, заставляющую начать действие, или оповещающую о его завершении.

Асинхронная организация цикла шины не содержит связанных с ожиданием пауз,

[5] Квитанция — официальный документ, расписка установленной формы в принятии денег, ценностей и т. д.

требует минимальных временных издержек, а также обеспечивает каждому устройству такую длительность цикла шины, которая соответствует его возможностям, независимо от длительностей циклов для других устройств. Платой за повышение эффективности является большая стоимость асинхронных шин по сравнению с синхронными.

Блочные циклы шины

Количество битов, которые передаются по шине за один рассмотренный ранее цикл чтения или записи, равно разрядности шины данных. Это могут быть одиночные байты, слова, двойные слова и т. д. Однако, как было выяснено при обсуждении кэша, данные из оперативной памяти желательно передавать пакетами (блоками), длина которых совпадает с длиной строки кэша. Понятно, что передача пакетами эффективнее передачи одиночными байтами или словами, так как можно сэкономить время на получение доступа к шине во время арбитража (см. 4.1.3), на выставление адреса и т. д.

Блочные (пакетные) циклы шины аналогичны по организации и получаемым преимуществам режиму BEDO микросхем памяти (см. 7.4.4). Для организации пакетного режима контроллеру памяти передается количество байтов (слов), из которых должен состоять блок. Далее без освобождения шины из памяти передается по одному байту (слову) в течение каждого такта до полного формирования пакета. Таким образом, при передаче блока, состоящего, например, из 4 байтов (слов), требуется не **12** тактов, а всего **6**.

Многошинная архитектура

Несмотря на возросшие с течением времени разрядности и тактовые частоты, несмотря на увеличение производительности шин, достигнутое с помощью конвейеризации и пакетного режима, одна общая шина в составе компьютера оказалась не способной обеспечить все необходимые передачи кодов и сигналов. Поэтому довольно быстро в архитектуре компьютеров появились дополнительные специализированные шины, которые связывают между собой различные группы устройств с различными скоростями обмена и различными требованиями к производительности. Архитектура компьютеров, в составе которых имеется более одной шины, называется **многошинной.**

Вначале в компьютерах была выделена шина, обеспечивающая обмен с низкоскоростными внешними устройствами, такими как клавиатура, мышь и принтер. Затем появилась еще одна шина с более высокой производительностью, к которой оказалось удобно подсоединять дисковые накопители. Относительно недавно выделилась шина для передачи на дисплей — графическая шина (шина графического адаптера) и т. д.

Когда в состав компьютеров стали включать внешний кэш, он подсоединялся к системной шине, связывающей процессор и оперативную память (см. рис. 9.1). Эта архитектура оказалась неэффективной, потому что кэш простаивал из-за низкой скорости шины. В 1997 г. была предложена архитектура **двойной независимой шины,** или **архитектура DIB** (от Dual Independent Bus), подразумевавшая наличие двух отдельных шин, одна из которых — **шина кэша,** или **шина BSB** (от Back Side Bus), — связывает процессор с внешним кэшем, а вторая — **системная шина,** или **шина FSB** (от Front Side Bus), — связывает процессор с оперативной памятью. Наличие двух независимых шин, по которым процессор может получить доступ к

данным, передающимся по любой из шин одновременно и параллельно, более чем в три раза ускоряет работу внешнего кэша.

Итак, современные персональные компьютеры обычно содержат (рис. 10.4):

- шину низкоскоростных внешних устройств (клавиатура, мышь, принтер);
- шину высокоскоростных внешних устройств (магнитные, оптические диски);
- шину графического адаптера, для передачи высококачественных цветных движущихся изображений на экран дисплея;
- системную шину (шину памяти), связывающую процессор и оперативную память;
- шину кэша, связывающую процессор и внешний кэш.

1<1PE.

Основные типы шин

В процессе развития архитектуры и технологии производства было создано несколько типов шин. Наиболее старыми, но до сих пор используемыми являются шины **ISA** (от Industry Standard Architecture — стандартная промышленная архитектура) и **EISA** (от Extended ISA — расширенная ISA). Это традиционные, дешевые и универсальные шины, служащие для подключения медленных внешних устройств, например таких как мышь и клавиатура, с тактовой частотой до **8** МГц и малой скоростью обмена — от **8** до 33 Мбайт/с. Шины ISA имели разрядности **8** и 16 бит, а шины EISA — 32 бита.

Позднее появилась шина **PCI** (от Peripheral Component Interconnect — соединитель периферийных компонентов) — высокоскоростная шина для подсоединения дисковых и графических устройств со скоростью обмена до 500 Мбайт/с и тактовой частотой **66** МГц. Разрядности шин PCI — 64 и 128 бит.

С 2004 г. в качестве перспективного стандарта, который должен прийти на смену шине PCI, рассматривается шина **PCI Express.** Для нее введена единица скорости передачи данных, равная 256 Мбайт/с. Уже выпущены шины этого стандарта, имеющие скорость передачи 16x, то есть 4 Гбайт/с.

Шины **IDE** (от Integrated Drive Electronics — интегрированные электронные устройства) и **EIDE** (от Extended IDE — расширенная IDE). Это шины и одновременно стандарт (интерфейс) для подключения жестких, оптических дисков и мобильных дисководов со скоростью обмена до 20 Мбайт/с.

Шина **SCSI** (от Small Computer System Interface — интерфейс малых вычислительных систем). Это шина и стандарт, которые предназначены для подключения высокопроизводительных дисковых устройств со скоростью обмена до 80 Мбайт/с. Отметим, что шины SCSI дороже шин EIDE.

Шина **AGP** (от Advanced Graphic Port — улучшенный графический порт). Это шина для подключения видеоплат со скоростью обмена от 256 Мбайт/с до 1, 06 Гбайт/с. При этом скорость обмена 256 Мбайт/с считается условной едини

цей измерения. Например, скорость обмена, равную 528 Мбайт/с, принято обозначать AGP 2x, а 1,06 Гбайт/с — AGP 4x.

Шина **USB** (от Universal Serial Bus — универсальная последовательная шина). Допускается одновременное подключение к такой шине до 127 внешних устройств (принтеров, сканеров, переносных запоминающих устройств и т. д.). Скорость обмена данными у шины доходит до 12 Мбайт/с, а у ее модификации USB 2.0 — до 60 Мбайт/с.

Шина **FireWire** (от fire wire — огненный провод), или **IEEE 1394.** Это шина и стандарт, предназначенные для подключения высокоскоростных внешних устройств типа цифровых фото- и видеокамер. Обеспечивается скорость обмена до 50 Мбайт/с.

Чипсет

Для согласования операций обмена и управления в сложной иерархической системе шин в персональных компьютерах служит специальный набор микросхем компьютера, который принято называть **чипсетом** (от chip set — набор чипов).

> Чипсет представляет собой базовый набор специализированных микросхем, при подключении которых друг к другу формируется функциональный блок компьютера, обеспечивающий связь между основными компонентами компьютера — процессором, памятью, шинами, периферийными устройствами и т. д.

Чипсет фактически является связующим звеном между всеми компонентами материнской платы. Именно он определяет возможности модернизации компьютера путем замены отдельных устройств более эффективными.

Микросхемы чипсета содержат в себе контроллеры прерываний, контроллеры прямого доступа к памяти, контроллеры шин и т. д. В одну из микросхем набора обычно входят также часы реального времени со CMOS-памятью, а иногда и клавиатурный контроллер, однако эти блоки могут присутствовать в компьютере и в виде отдельных микросхем. В последних разработках в состав микросхем чипсета стали включаться контроллеры внешних устройств.

В «обязанности» чипсета, в частности, входит:

- обслуживание управляющих сигналов процессора;
- формирование управляющих сигналов для внешнего кэша;
- формирование адреса и управляющих сигналов для микросхем оперативной памяти;
- обеспечение когерентности данных для всех уровней кэша и оперативной памяти;
- обеспечение взаимосвязи между всеми типами шин компьютера ;
- арбитраж контроллеров шин и т. д.

Характеристики чипсета определяют большинство важнейших параметров компьютера, в том числе:

- возможные типы и частоты центрального процессора;
- скоростные характеристики внешнего кэша и его допустимый объем;
- типы, объемы и максимальное количество модулей динамической памяти;
- возможные частоты шин и т. д.

Тип используемого чипсета существенно влияет на производительность компьютера в целом. При одинаковых базовых компонентах компьютера — процессоре, оперативной памяти, кэше, графическом контроллере (контроллере дисплея) и жестком диске — производительность машин, собранных на разных чипсетах,

может различаться на 30 % и более.

Тема 12: Улучшение эффективности процессора

Рассмотренные ранее подходы к реализации памяти, кэша и шин компьютера существенно влияют на его эффективность. Однако самое большое влияние на производительность компьютера оказывают характеристики процессора, которые также могут быть существенно улучшены, и не только путем изменения его физических параметров. Основными техническими характеристиками процессоров в настоящее время считаются:

- архитектура системы команд процессора;
- модель процессора, связанная с его системой команд;
- внутренняя тактовая частота;
- разрядность или длина машинного слова;
- объем внутреннего кэша L1;
- производительность;
- плотность логических элементов;
- линейный размер логических элементов;
- форм-фактор, определяющий геометрические размеры, форму, количество контактов посадочного гнезда (сокета), а также способ подключения микросхемы процессора к материнской плате.

Архитектура системы команд и модели компьютеров рассматриваются в главах 15 и 16. Понятие быстродействия процессора или его производительности обсуждается в главе 14. А в данной главе описываются архитектурные подходы и решения, касающиеся логического устройства процессора.

Микроархитектура процессора

Уже на начальных этапах развития вычислительных систем стало ясно, что компьютер — это сложная система, которую следует рассматривать как ***многоуровневую структуру*** (рис. 11.1). На самом нижнем уровне этой структуры находятся ***физические элементы***, из которых изготовлена машина: электромагнитные реле, электронные лампы накаливания, интегральные схемы, резисторы, конденсаторы, соединительные провода, шины, линии связи и т. д. На этом уровне обсуждаются физические характеристики устройств: геометрические размеры (высота, ширина, толщина), электрические параметры (уровни напряжения и тока в цепи, сопротивление, емкость, индуктивность), временные последовательности прохождения импульсов тока и т. д.

Следующий уровень вычислительной системы состоит из различных ***вентилей***, которые реализуют все необходимые функции цифровой логики, из-за чего его называют ***цифровым логическим*** уровнем, или уровнем цифровой схемотехники. На этом уровне происходит абстрагирование от отдельных физических устройств и их характеристик. Обсуждение ведется в терминах двоичных цифр и связывающих их логических и арифметических операций. Этот уровень организации компьютера довольно подробно рассматривался в 3.2. Физический и цифровой логический уровень — это сфера деятельности разработчиков вычислительной техники, специалистов в области схемотехники.

Над указанными уровнями в машинах первого поколения располагался уровень машинных команд (рис. **11.1**, я), которые подобны командам, рассматривавшимся в главе 4. Пользователь взаимодействовал с аппаратурой компьютера на языке машинных команд. При этом в его поле зрения отсутствовали отдельные вентили и поступающие в них сигналы. Рассмотрение велось на уровне битов, байтов,

полей, кодов операций, адресов операндов, флажков процессора и т. д. Ясно, что для практического программирования хватает общего представления о первых двух уровнях, но зато необходимо детальное знание уровня машинных команд. Довольно быстро специалисты обнаружили, что выполнение процессором большинства машинных команд включает в себя практически одинаковые действия. Как правило, в различных машинных командах по одной и той же схеме выполняется выборка операндов из регистров процессора, из полей памяти или из самой команды. Точно так же одинаково отправляются на хранение результаты выполнения команд, и по одним и тем же правилам формируются значения флажков. Кроме того, имеются тесно связанные команды, реализующие родственные действия, например: сложение, инвертирование знака, вычитание, инкремент, декремент. Было бы неразумно для каждой машинной команды предусматривать отдельные аппаратные схемы выполнения, в которых дублируются выполняемые по общим правилам действия.

В связи с этим еще в 1951 г. М. Уилкс из Кембриджского университета с целью упрощения аппаратного обеспечения компьютера предложил подход, который впоследствии получил название **микропрограммирования.** Для его реализации в структурную организацию вычислительной системы между цифровым логическим уровнем и уровнем машинных команд включается дополнительный **микро**архитектурный уровень (рис. **11.1**, ***б).***

На этом рисунке кроме микроархитектурного уровня изображены еще два дополнительных уровня в организации вычислительной системы, которые также появились позднее. Это уровень операционной системы, автоматизирующий доступ программам и пользователям ко всем ресурсам компьютера, а также уровень приложений, которые обычно обеспечивают пользователям значительно более удобный интерфейс, чем работа на уровне машинных команд или на уровне операционной системы.

Основу микроархитектурного уровня образуют **микрокоманды.** Каждая микрокоманда, выдавая управляющие сигналы на отдельные вентили аппаратных схем, определяет некоторое простейшее действие процессора, например задает выбор кода из конкретного регистра процессора, установку флажка в соответствующее результату значение и т. д. Поэтому выполнение любой машинной команды фактически представляет собой выполнение соответствующей последовательности микрокоманд. Например, команда add ax, cx из листинга 4.2 может быть реализована как последовательность микрокоманд, изложенная ниже.

1. Микрокоманда, выбирающая операнд из регистра ax и перемещающая его в арифметико-логическое устройство.
2. Микрокоманда, выбирающая операнд из регистра cx и перемещающая его в арифметико-логическое устройство.
3. Микрокоманда, формирующая новое значение указателя команд.
4. Микрокоманда, выполняющая сложение двух операндов.
5. Микрокоманда, отправляющая полученную сумму в регистр ax.
6. Микрокоманда, формирующая значения флажков zf и sf.

Если же, например, нужно выполнить команду sub ax, cx, то для ее реализации может быть использована почти такая же последовательность микрокоманд, только после второй микрокоманды нужно включить микрокоманду, инвертирующую вычитаемое, и микрокоманду, выполняющую его инкремент. Две добавочные микрокоманды образуют дополнительный код вычитаемого и тем самым

сводят вычитание к сложению с отрицательны^ числом.
Последовательности микрокоманд, аналогичные рассмотренной, могут быть построены для каждой машинной команды, принадлежащей системе команд процессора. Понятно, что такие последовательности существенным образом зависят от базового набора микрокоманд, включенных в микроархитектурный уровень. В связи с этим в различных реализациях микроархитектурного уровня формируемые последовательности микрокоманд могут быть разными.
Из приведенных примеров видно, что введение микроархитектурного уровня значительно упрощает разработку аппаратных средств, которые теперь должны уметь выполнять значительно меньше различных действий, чем при аппаратной реализации каждой машинной команды в отдельности.
Отметим еще одну важную особенность микрокоманд. Любая микрокоманда выполняется за один такт работы процессора, в то время как выполнение машинной команды может потребовать нескольких его тактов.
С точки зрения микроархитектурного уровня отдельная машинная команда — это целая программа. Она выполняется либо специальной программой, которую принято называть **микропрограммой,** либо аппаратными средствами процессора. Основными функциями микропрограммы являются построение для каждой машинной команды из программы пользователя соответствующей последовательности микрокоманд и организация их выполнения. Эти действия выполняются микропрограммой для каждой машинной команды по отдельности, то есть программа пользователя фактически интерпретируется микропрограммой, поэтому микропрограмма считается ***интерпретатором*** выполняющейся программы.
Из-за огромного влияния микропрограммы на эффективность работы компьютера в целом она разрабатывается особенно тщательно. Обычно микропрограмма имеет небольшие размеры и хранится в постоянном запоминающем устройстве компьютера в виде, недоступном для внесения изменений.
С течением времени границы между функциями микропрограммы и аппаратной реализацией микрокоманд меняются. Появление новых аппаратных возможностей приводит к переносу операций, закрепленных за программным обеспечением, на аппаратный уровень, их встраиванию в аппаратные средства. Это приводит к увеличению скорости выполнения машинных команд. С другой стороны, программная реализация машинной команды удешевляет разработку и производство компьютера. Конструкторам вычислительной техники приходится постоянно искать компромисс между противоречивыми требованиями, когда улучшение одних параметров приводит к ухудшению других. Сложность в изучении микроархитектурного уровня связана с тем, что разработчиками компьютеров предложено огромное количество различных вариантов его реализации. Подробное изучение микроархитектурного уровня выходит за рамки данного учебника.

Конвейерная архитектура процессора

Впервые параллелизм на уровне машинных команд в виде совмещения во времени различных этапов выполнения команды был реализован в архитектуре выпущенной в 1959-1961 гг. машины IBM Stretch. В ее конструкции был использован ***буфер вызова команд с упреждением.*** Очередная команда программы попадала из оперативной памяти в этот буфер заранее, еще до завершения предыдущей команды. В момент, когда очередь доходила до выполнения следующей команды, она выбиралась уже из буфера, что требовало гораздо меньше времени, чем выборка из оперативной памяти. Кроме того, в машине был организован

опережающий просмотр находящихся в буфере команд для определения адресов и предварительной выборки операндов. К тому моменту, когда процессор заканчивал дешифрацию выбранной команды, ее операнды уже находились в арифметико-логическом устройстве. И сразу же в буфер переносилась следующая команда. В выпущенной в 1962 г. машине ATLAS этот подход был расширен до полноценной конвейерной архитектуры процессора.

На рис. 11.2 представлен порядок выполнения машинной команды процессором, не имеющим конвейера. По горизонтальной оси рисунка располагаются этапы выполнения команды процессором, а по вертикальной оси отложены такты его работы. Для упрощения обсуждения будем считать, что процессор осуществляет выполнение ***любой*** команды за пять этапов, а на выполнение каждого этапа требуется один такт. Тогда на выполнение одной команды процессору требуется пять тактов: на первом такте процессор выполняет выборку операнда, на втором — декодирование, на третьем — выборку операндов, на четвертом — исполняется действие команды, а на последнем пятом — записываются результаты. Точно по такой же схеме выполняется каждая следующая команда, причем процессор не начинает выполнение следующей команды, пока не завершит выполнение предыдущей. Поэтому промежуток времени между завершениями двух последовательных команд также равен пяти тактам (рис. **11.2**).

Обратите внимание на то, что схема процессора, которая выполняет, например, выборку команды, простаивает в течение четырех тактов, пока процессор полностью не завершит выполнение всей команды. Аналогичным образом после выполнения своих «обязанностей» простаивают все остальные схемы процессора.

Идея организации конвейерной обработки состоит в том, чтобы с помощью выделения этих схем процессора в относительно самостоятельные блоки организовать их непрерывную работу — сразу после обработки своего этапа текущей команды блок должен переключаться на выполнение того же самого этапа у следующей команды.

При наличии конвейера в микросхемах памяти, в шинах и т. д. архитектура компьютера в целом не считается конвейерной. Только организация конвейерного выполнения машинных команд в процессоре позволяет считать архитектуру компьютера конвейерной.

Группа блоков процессора, обеспечивающих одновременное выполнение различных этапов различных машинных команд, называется конвейером, блоки, из которых он состоит, называются ступенями конвейера, а количество ступеней — длиной конвейера. Архитектура компьютера, в конструкции процессора которого используется конвейер, называется конвейерной.

Многопроцессорные и многоядерные архитектуры

Дальнейшее наращивание производительности однопроцессорного компьютера, вероятнее всего, будет происходить с большими трудностями и вскоре станет невозможным.

Вместе с тем разработчики уже довольно давно начали создавать вычислительные системы, которые содержат несколько процессоров, обладающих возможностью одновременно выполнять разные участки одной и той же или различных программ. Такие ***многопроцессорные*** системы имеют ряд очевидных преимуществ перед однопроцессорными. Во-первых, несколько процессоров могут выполнить больше работы за фиксированный промежуток времени или же

выполнить ту же работу, но быстрее, чем один. Во-вторых, многопроцессорная система надежнее, чем однопроцессорная, так как вышедший из строя процессор может быть почти мгновенно заменен другим процессором без остановки работы системы. В-треть- их, такая система может одновременно обслужить большее количество пользователей с меньшим временем ожидания ответа для каждого из них. Имеются и другие выгодные моменты, поэтому создано множество различных вариантов многопроцессорных систем, в которых разработчики пытались реализовать эти преимущества. К настоящему времени построены многопроцессорные системы, состоящие из сотен тысяч процессоров и обладающие огромными вычислительными мощностями, в миллионы раз превосходящими производительность любых однопроцессорных систем.

Однако системы, содержащие несколько процессоров, стоят гораздо дороже, чем системы с одним процессором. Кроме того, у многопроцессорных систем существует множество конструктивных проблем, связанных с организацией электропитания, охлаждения, обслуживания, с геометрическими размерами и т. д. Многопроцессорным системам нужно специальное программное обеспечение, которое должно учитывать наличие нескольких процессоров. Операционные и инструментальные системы для них должны обеспечивать оптимальное распределение между процессорами потоков команд и данных, разрешать конфликты при одновременном запросе одного и того же ресурса, поддерживать когерентность данных в кэшах процессоров, а также решать множество других задач параллельной обработки данных. Более подробно особенности архитектуры многопроцессорных вычислительных систем обсуждаются в главе 17.

Современные процессоры достигли тактовых частот порядка нескольких гигагерц. На частоте 2 ГГц такт длится 0,5 не, за это время электромагнитное поле распространяется на расстояние всего 15 см. Данные оценки показывают одно из узких мест многопроцессорных систем. Существующие ограничения на минимальные конструктивные расстояния между процессорами, вызванные рядом технических и физических факторов, например необходимостью обеспечивать охлаждение процессоров и защищать от взаимного искажения передаваемые по шинам данные, начинают сказываться на скорости обработки данных. Если, например, процессоры находятся на расстоянии 0,5 м друг от друга, то обмен данными между ними приведет к задержке на четыре такта, в течение которых биты проходят по линиям шины от одного процессора к другому.

По этой причине разработчики в последние годы занимались разработкой систем, которые содержат два процессорных ядра в одной микросхеме процессора. Каждое ядро по своим функциональным возможностям полностью аналогично обычному одноядерному процессору. Другими словами, двухъядерный процессор соответствует вычислительной системе с двумя отдельными процессорами. Различие проявляется в том, что в двухъядерном варианте эти процессоры конструктивно размещаются ***предельно близко***, на площади примерно 200-300 мм^2, что обеспечивает значительную экономию на времени передачи данных по сравнению с двухпроцессорным вариантом. Одновременно снимается ряд других конструктивных проблем: например, теперь достаточно всего одной, но более мощной системы охлаждения.

Использование многоядерных процессоров обеспечивает более высокую производительность системы и по сравнению с одноядерным процессором, работающим с применением технологии Hyper Threading. На рис. 11.9, ***в*** изображена ситуация,

при которой одно ядро занято выполнением команд потока А, а второе выполняет команды потока Б. При этом общее время выполнения обоих потоков оказывается равным **10** тактам, то есть времени выполнения одного потока.

Отметим, что каждое из ядер процессора работает по технологии Hyper Threading, по этой причине работа одного двухъядерного процессора может рассматриваться как одновременная работа четырех логических процессоров.

Итак, многоядерные процессоры решают проблему ограничений на минимальные расстояния между процессорами и увеличивают общую производительность системы. Однако остальные проблемы многопроцессорных систем, которые связаны с программным обеспечением параллельных вычислений, характерны и для систем с многоядерными процессорами, и их еще предстоит решать. Предполагается, что в ближайшие годы это направление развития вычислительных систем станет одним из основных.

Тема 13: Внешняя память

Как отмечалось в 4.1.4, внешними запоминающими устройствами (ВЗУ), или просто внешней памятью (ВП), в компьютере считается группа устройств, которые предназначены для долговременного хранения больших массивов программ и данных. Отличительными особенностями внешней памяти являются:

- отсутствие доступа со стороны процессора;
- энергонезависимость;
- малая скорость обмена (относительно оперативной памяти);
- практически неограниченный объем;
- относительная дешевизна хранения данных.

В настоящее время основными разновидностями внешней памяти являются гибкие и жесткие магнитные диски, оптические диски, магнитные ленты и переносимые устройства памяти.

Жесткие магнитные диски

Жесткий магнитный диск (ЖМД) называют также винчестером (от Winchester — разновидность двуствольной винтовки) и **HDD** (от Hard Disk Drive — привод жесткого диска). Он представляет собой пакет, содержащий от 2 до 10 закрепленных на общей оси пластин. Пакет помещается в герметичный корпус, в котором создается разрежение воздуха, обеспечивающее высокую скорость вращения и передачи данных в процессе обмена. Во время вращения головки чтения/записи не касаются рабочих поверхностей дисков. Такая конструкция позволяет увеличить плотность записи кодов данных до $\mathbf{10^5}$ бит/см^2 и, следовательно, увеличить общий объем диска. Обычно жесткий диск является постоянным, несъемным устройством компьютера, поэтому доступ к нему без разбора корпуса компьютера обычно невозможен. Но существуют и варианты комплектации компьютера со сменными жесткими дисками.

Жесткие диски описываются следующим набором характеристик:

- объем диска;
- среднее время доступа;
- скорость вращения диска;
- скорость передачи данных;
- стандарт интерфейса;
- размер собственной кэш-памяти диска;
- форм-фактор;
- фирма-производитель.

Современные винчестеры имеют объем от нескольких десятков до нескольких сотен гигабайт. По-видимому, в ближайшее время этот показатель может возрасти до нескольких терабайт.

> Относительно недавно были созданы устройства, которые способны определять в сотни раз более слабое магнитное поле, чем это было возможно до сих пор. Одним из результатов применения подобных разработок может стать повышение емкости жестких дисков в компьютерах до нескольких сотен петабайт (то есть до сотен тысяч терабайт). Заметим, что, по некоторым подсчетам, объем всей имеющейся в мире на сегодняшний день информации (включая литературу, звукозаписи и видеозаписи) не превышает 100 Пбайт.

Для перемещения головок чтения/записи к различным дорожкам требуется раз-

личное время, так как они имеют различные радиусы. Кроме того, в процессе обмена головки перемещаются между дорожками, вообще говоря, случайным образом. Поэтому для характеристики временных затрат, связанных с позиционированием головок над нужной дорожкой, используется величина, равная среднему времени доступа к дорожке, вычисленному по всем дорожкам рабочей поверхности. Жесткие диски имеют среднее время доступа, равное 7-9 мкс.

Пакет пластин вращается в герметичном корпусе с большой скоростью. В настоящее время стандартные скорости вращения жесткого диска составляют 5400, 7200 и 10 800 об./мин. Высокая скорость вращения дисков приводит к их нагреванию. Вследствие расширения материала диска при нагревании увеличиваются его геометрические размеры. Так как линейный размер бита на рабочей поверхности не превышает **0,2** мкм, а ширина дорожки не превышает **10** мкм, даже малые изменения размеров пластины могут привести к неточности позиционирования головок диска над битом при чтении или записи. В связи с этим некоторые диски нуждаются в рекалибровке, то есть повторной калибровке механизмов перемещения головок, учитывающей изменение геометрических размеров пластин. На выполнение рекалибровки требуется некоторое время, в течение которого диск не может быть использован для операций обмена. Это может привести к трудностям при использовании мультимедийных программ, требующих непрерывного потока битов для качественного воспроизведения звука и видео. Поэтому для хранения мультимедийных данных используются специальные аудио- и видеодиски, не испытывающие термических расширений и тем самым обеспечивающие высокое качество записи и воспроизведения звука и изображения.

Скорость чтения/записи данных у жестких дисков зависит от скорости вращения и плотности записи. Для ее указания используется спецификация вида UDMA/n, которая определяет максимальную скорость передачи данных в режиме DMA (прямого доступа диска к оперативной памяти), равную ***n*** мегабайт в секунду. Например, спецификация UDMA/100 означает, что максимальная скорость передачи равна 100 Мбайт/с.

Так как данные из файла обычно разбросаны по разным секторам, дорожкам и поверхностям диска, существует большая разница между максимально возможной скоростью передачи и фактической или средней скоростью. В настоящее время средняя скорость передачи равна 10-15 Мбайт/с.

Работой каждого диска управляют соответствующие его модели контроллеры. Каждый контроллер по определенным правилам взаимодействует с управляемым диском и с шиной, к которой он подключен. Совокупность таких правил образует стандарт, или интерфейс, диска и его шины. В настоящее время стандартно используемыми являются интерфейсы IDE, EIDE, SCSI и RAID.

Диски стандарта IDE (от Integrated Drive Electronics — интегрированное электронное устройство) имеют встроенный в корпус устройства контроллер. Таким образом, приобретая устройство, потребитель одновременно приобретает и соответствующий контроллер. В базовом стандарте диск IDE может содержать до 16 рабочих поверхностей, до 1024 цилиндров и до 63 секторов на дорожке. Общий объем таких дисков не превышает 512 Мбайт. Интерфейс IDE допускает малую и среднюю скорости обмена от 2,1 до 8,3 Мбайт/с. По этому стандарту к контроллеру можно подключить только один монопольно работающий с шиной диск.

Интерфейс **EIDE** (от Extended IDE — усовершенствованный IDE) предусматривает возможность одновременного подключения к контроллеру до четырех дисков,

которые используют подключение монопольно. Это значит, что шина и контроллер этого стандарта работают аналогично селекторному каналу. К контроллеру может быть подключено несколько устройств, но любое из них захватывает шину и контроллер на весь период обмена. Контроллер переходит к обслуживанию следующего диска только после полного завершения обмена для текущего диска. Возможная скорость обмена для дисков EIDE увеличена до 20 Мбайт/с, а объем может достигать сотен гигабайт.

Интерфейс **SCSI** (от Small Computer System Interface — интерфейс малых вычислительных систем) предусматривает скорость обмена до 80 Мбайт/с при высокой стабильности передачи данных. Для дисков этого стандарта необходим отдельно приобретаемый и устанавливаемый контроллер, который допускает одновременное подсоединение к нему до семи дисковых устройств. Отличительной особенностью контроллера SCSI является возможность ***одновременной*** работы всех подключенных дисков. В этом смысле работа контроллера SCSI похожа на работу мультиплексного канала: несколько устройств могут начать и одновременно выполнять обмен.

Имеется несколько модификаций обсуждаемого стандарта. Например, простой стандарт SCSI предусматривает работу с шиной разрядностью **8** бит, тактовой частотой 5 МГц и скоростью обмена до 5 Мбайт/с, а в самом развитом стандарте с названием Wide Ultra2 SCSI предусматривается шина с разрядностью 16 бит, тактовой частотой 40 МГц и скоростью обмена 80 Мбайт/с.

Перспективным направлением развития интерфейса SCSI считается интерфейс **SAS** (от Serial Attached SCSI — последовательное соединение для интерфейса SCSI), обеспечивающий скорость обмена до 300 Мбайт/с.

Интерфейс **RAID** (от Redundant Array of Inexpensive Disks — избыточный массив недорогих дисков, позднее возникла трактовка Redundant Array of Independent Disks — избыточный массив независимых дисков), по сути, относится не к дискам, а к контроллеру, управляющему группой дисков. Компьютер комплектуется группой обычно SCSI-дисков, которые подсоединяются к RAID-контроллеру. Он обеспечивает восприятие операционной системой группы дисков как одного диска очень большой емкости — как диска типа **SLED** (от Single Large Expensive Disk — одиночный большой дорогой диск). Один контроллер RAID может управлять группой, содержащей до 16 дисков SCSI. Основное преимущество интерфейса RAID — возможность осуществления параллельного обмена со всеми дисками массива.

Существует несколько способов размещения данных на RAID дисках, которые обеспечивают, с одной стороны, высокую скорость передачи данных, а с другой — высокую надежность их хранения. Эти способы принято называть типами, или уровнями, RAID. По сути дела, многие из этих способов воспроизводят на уровне жестких дисков приемы, использованные для повышения эффективности оперативной памяти.

Рассмотрим особенности некоторых вариантов организации RAID-массивов. Нулевой тип (нулевой уровень) массива RAID аналогичен обсуждавшемуся ранее расслоению оперативной памяти (см. 7.4.1). На жестких дисках выделяются зоны, которые можно рассматривать как аналоги блоков оперативной памяти в механизме расслоения. Каждая зона включает некоторое количество секторов диска. Количество зон на диске и секторов в одной зоне может изменяться в широких пределах и подбирается программным обеспечением, исходя из скорости

передачи, объема дисков, количества дисков в массиве и т. д. На рис. 12.3 изображен массив из четырех дисков, на которых организовано 16 зон.

ДискО_	Диск 1	Диск 2	Диск 3
Зона 0	Зона1	Зона 2	Зона 3
Зона 4	Зона 5	Зона 6	Зона 7
Зона	Зона 9	Зона	Зона
Зона	Зона	Зона	Зона

Рис. 12.3. Массив дисков RAID нулевого типа

Каждая следующая зона располагается на следующем диске массива. Вся совокупность зон образует кольцо, то есть после записи последней зоны происходит возврат к первой. Такое распределение зон по нескольким дискам называется разметкой. Записываемый на диски поток байтов (или слов) направляется в разные зоны: первый байт (слово) записывается в первую зону, второй байт (слово) — во вторую и т. д. Это позволяет организовать параллельный высокоскоростной обмен с дисками группы. Но надежность всей системы дисков ниже, чем при использовании одного диска. В самом деле, вероятность выхода из строя одного из, скажем, четырех дисков в четыре раза выше, чем вероятность выхода из строя единичного диска. А при выбранной схеме разметки выход из строя любого из дисков означает, что все данные будут потеряны.

В массиве RAID первого типа все диски массива делятся на основные и дублирующие. Каждый основной диск массива имеет один дублирующий (рис. 12.4). Запись информации как в группу основных, так и в группу дублирующих производится по тому же принципу, что и для массива дисков нулевого уровня, — по разнесенным между дисками зонам. Но каждая зона записывается дважды: один раз на основной диск, а второй — на дублирующий. При этом запись зоны на дублирующий диск производится одновременно с ее записью на основной диск. Поэтому скорость передачи данных при записи такая же, как в обычном режиме. Но чтение может производиться с любого экземпляра диска, как основного, так и дублирующего. Так как обмен со всеми дисками может производиться одновременно, может быть получен выигрыш в скорости чтения в два раза.Добавим, что эта схема имеет высокую отказоустойчивость, поскольку возможно полное восстановление информации с уцелевшего экземпляра диска.

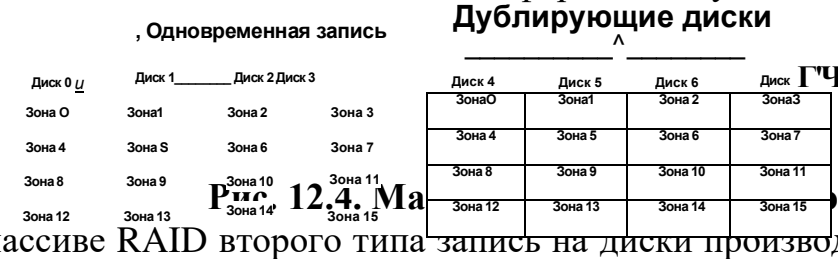


Рис. 12.4. Ма ого типа

В массиве RAID второго типа запись на диски производится не зонами, а полубайтами, байтами или словами, к которым добавляется необходимое количество контрольных разрядов кода Хемминга (см. 2.6), обеспечивающих необходимый уровень надежности записи данных. На рис. 12.5 показано, как к четырем битам полубайта добавляется три контрольных разряда кода Хемминга и каждый бит записывается на отдельный диск массива. Второй тип массива RAID, так же как и первый, обеспечивает высокую надежность, поскольку выход из строя любого диска не нарушает целостности информации. Но, в отличие от массива первого

типа, накладные расходы гораздо меньше, так как полное дублирование требует большего количества дисков, чем включение в код контрольных битов. В то же время метод требует высокой синхронизации дисков и высокопроизводительного контроллера.Рис. 12.5. Массив дисков RAID второго типа

В практике построения дисковых массивов RAID применяется несколько десятков увеличивающих скорость обмена и/или надежность хранения способов, являющихся модификациями и улучшениями описанных ранее.

Современный жесткий диск может комплектоваться собственным кэшем, объем которого доходит до 2 Мбайт. Роль кэша у жесткого диска полностью аналогична роли кэша оперативной памяти.

Форм-фактор диска описывает его геометрические размеры и форму. В настоящее время используются преимущественно три форм-фактора: стандартные (для настольных систем), для переносных компьютеров и для дисков типа Micro Drive.

Наиболее известными фирмами-производителями жестких дисков в настоящее время являются IBM, Fujitsu, Western Digital (WD), Seagate, Quantum.

Оптические диски

Оптические диски, так же как и гибкие диски, относятся к группе сменных носителей информации. Первые диски имели диаметр 30 см и использовались для хранения копий кинофильмов. В 1980 г. компаниями Philips и Sony были разработаны оптические диски меньшего диаметра. Они получили название **CD** (от Compact Disc — компактные, то есть небольшие диски). Немного позже был утвержден сохраняющийся до сих пор стандарт IS 10149 технических характеристик компакт-дисков. В частности, они должны иметь диаметр 117 мм, толщину 1,2 мм и диаметр центрального отверстия 15 мм. Предполагается, что компакт-диски ***смогут*** сохранять записанную на них информацию в течение **100** лет.

Компакт-диски CD-ROM

Первые компакт-диски изготавливались по следующей технологии. Вначале с помощью мощного лазера в контрольном стеклянном диске выжигаются углубления диаметром **0,8** мкм, которые рассматриваются как коды единичного значения бита. Отсутствие углубления считается кодом нулевого значения бита. Углубления принято называть впадинами, а ровные поверхности между ними — площадками. Запись кода начинается от центра и продвигается к краю, при этом впадины и площадки наносятся на поверхность диска не концентрическими полосами, как у магнитных дисков, а в виде спирали (рис. **12.6**), которая проходит около 22 000 оборотов вокруг диска. Общая длина спирали составляет более 5 км.

По этому диску делается шаблон, у которого вместо углублений образуются выступы. В процессе производства компакт-диска в шаблон вводится жидкая смола (поликарбонат), в которой углубления образуются там же, где и у контрольного диска. Далее на смолу наносится образующий жесткую основу диска тонкий слой алюминия, который покрывается защитным лаком. Очевидно, что таким способом можно получить диск, с которого можно только считывать однажды записанные на него коды. Изготовленные в условиях промышленного производства оптические диски, которые могут использоваться только для чтения, получили название CD-ROM (от Compact Disk Read Only Memory — память только для чтения на компакт-дисках).

Во время воспроизведения луч лазера небольшой мощности диаметром 0,78 мкм пробегает по спиральной канавке диска. Лазер светит с той стороны, гд$ находится слой смолы, поэтому впадины для луча — это выступы на ровной поверхности. Вообще говоря, при считывании воспринимаются не сами выступы, а переходы между ними и ровными участками. Это обеспечивает более высокую надежность считывания. Чтобы считывание выполнялось равномерно и надежно, луч лазера должен двигаться по спирали с постоянной линейной скоростью

1, 2 м/с. Поэтому угловая скорость вращения по мере считывания уменьшается от 530 до 200 об./мин. Отметим, что магнитные диски вращаются с постоянной угловой скоростью.

В связи с высокой плотностью записи информации на компакт-диски их разработчики уделяли много внимания надежности выполнения операций записи и чтения. В 1984 г. был принят международный стандарт, содержащий описание методов борьбы с ошибками хранения информации на CD-ROM. Предусмотренная в стандартах система кодирования данных включает трехуровневую защиту от ошибок.

На самом нижнем уровне каждые восемь информационных битов байта дополняются шестью контрольными битами кода Хемминга, которые позволяют исправлять двойные ошибки в одном байте. Таким образом, один байт компакт-диска фактически состоит из 14 битов.

На втором уровне защиты к каждой группе из 24 байтов данных (то есть 14-бит- ных кодов) добавляются 18 контрольных кодов той же самой длины. Эту последовательность из 42 кодов принято называть фреймом (от frame — каркас). Следовательно, фрейм, состоящий из 42 • 14 = 588 битов, содержит всего 24 • **8** = 192 информационных бита.

На верхнем уровне защиты от ошибок каждые 98 фреймов образуют сектор. Сектор начинается с занимающей 16 байтов преамбулы. Ее первые 12 байтов содержат специальный код, который обеспечивает синхронизацию во время обмена. Следующие три байта преамбулы содержат порядковый номер сектора, а в ее последнем байте находится код типа диска. За преамбулой располагаются 2048 информационных и 288 контрольных байтов кода Рида — Соломона. Под преамбулу и контрольный код занята некоторая часть информационных байтов фреймов. Сектор, состоящий из 98 фреймов, имеет общую длину 57 624 бита, из них к информационным относятся только 16 384 бита. То есть информационную нагрузку несут примерно 28 **%** площади канавки компакт-диска, а остальные обеспечивают правильность записи и чтения.

Стандартный диск вмещает 332 800 секторов и имеет объем 650 Мбайт. Отметим, что имеются оптические диски, у которых проверки верхнего уровня отсутствуют и

информационными являются не 2048, а 2336 байтов сектора, то есть все байты сектора, за исключением преамбулы. Существуют и другие варианты стандартов, в соответствии с которыми объем компакт-диска может доходить до 800 Мбайт.

Скорость 150 Кбайт/с, характерная для первых компакт-дисков, в настоящее время принята в качестве стандартной единицы измерения скорости. Фактическая скорость обмена указывается множителем, определяющим, во сколько раз допустимая скорость чтения больше, чем базовая скорость 150 Кбайт/с. Эта характеристика указывается в названии дисков или как их основная характеристика. Если, например, в маркировке диска указано 8x-12x, это означает, что он может использоваться для работы на дисководах со скоростями от 1200 Кбайт/с до 1800 Кбайт/с. Заметим, что в настоящее время уже имеются 52-скоростные дисководы (52x), что составляет примерно 7,6 Мбайт/с.

Компакт-диски однократной записи CD-R

Существенный недостаток CD-ROM — невозможность записи на диск при их использовании — был устранен в дисках WORM (от Write Once/Read Many — однократная запись, множественное считывание), которые имеют более распространенное название CD-R (от Compact Disk Recordable — записываемый компакт-диск). На диски этого типа можно записывать информацию, так же как и на обычную гибкую дискету, — прямо на компьютере, ***но только один раз***. Чтение такого диска может производиться произвольное количество раз. Для использования этой технологии требуются специальные заготовки дисков и специальные дисководы.

Запись двоичного кода на заготовки (в просторечии болванки) таких дисков производится с помощью воздействия высокотемпературного лазерного луча на особый светочувствительный слой диска из цианина зеленого цвета или пталоцианина желто-оранжевого цвета, который при этом как бы «выгорает». Упомянутые вещества сходны с красителями, используемыми в некоторых видах цветных фотопленок.

В исходном состоянии заготовки светочувствительный слой прозрачен. Запись кода выполняется с помощью луча лазера высокой мощности. Для записи кода единицы луч лазера, пробегающий по спирали над рабочей поверхностью, соприкасается со светочувствительным слоем, нагревает его и образует в месте контакта темное пятно. При записи кода нуля контакт между лучом и слоем отсутствует, и слой остается прозрачным.

Чтение кода осуществляется с помощью луча лазера, мощность которого уменьшена примерно в 20-30 раз. Пробегающий над поверхностью с закодированным темными и светлыми областями двоичным кодом луч отражается от нее и улавливается входящим в состав дисковода фотодетектором, который выявляет разницу между темными и прозрачными областями канавки, — тем самым записанный ранее код считывается.

В отличие от магнитных дисков, запись на которые производится отдельными кластерами, не обязательно размещенными последовательно друг за другом, запись файлов на оптические диски может производиться только сплошными участками. Группа последовательных секторов, записываемых за один раз, называется дорожкой. Еще раз подчеркнем, что дорожка компакт-диска — это не окружность, а состоящий из произвольного количества секторов участок спирали.

Для обеспечения надежной записи каждая дорожка должна записываться непрерывно без изменения скорости луча. Поэтому жесткий диск должен передавать

записываемые на оптический диск данные без задержек на поиск нужного файла. Чтобы не произошла остановка во время записи дорожки на компакт-диск, программы, записывающие информацию на CD-R, вначале формируют на жестком диске так называемый файл-образ всех объектов, которые должны быть записаны на компакт-диск. Все предназначенные к записи на компакт-диск файлы и папки собираются в буферной области на жестком диске, и из них формируется, создается временный единый файл, который и носит название файл-образ. После чего за один сеанс записи, который принято называть сессией, лазерный луч переносит его на оптический диск.

Во время сессии на оптическом диске формируется ***оглавление***, содержащее информацию о записанных на диск файлах и папках и их расположении. Это оглавление называется VTOC (от Volume Table Of Content — таблица содержания). Наличие на оптическом диске единственной VTOC позволяет организовывать единственную сессию записи. Если за время такой сессии используется не весь возможный объем диска, неиспользованный участок рабочей поверхности диска безвозвратно теряется. Немного позднее появился более эффективный стандарт CD-R XA, в котором определены средства организации ***многосессионной*** записи, обеспечивающей несколько последовательных сеансов записи на диск CD-R. Этот стандарт предусматривает создание во время каждой очередной сессии нового экземпляра VTOC, который может включать в себя нужные сведения из предыдущего экземпляра. Если сведения о ранее записанных на компакт-диск файлах не включаются в новый VTOC, эти файлы оказываются как бы удаленными. Но место, которое такие файлы занимают на поверхности диска, вновь использовать невозможно. Во время чтения всегда используется самый последний экземпляр VTOC. Отметим, что некоторые программы записи на CD-R не допускают многосессионную работу.

Компакт-диски многократной записи CD-RW

По своим размерам, объему и внешнему виду диски CD-RW (от Compact Disk ReWriteable — перезаписываемые компакт-диски) ничем не отличаются от дисков CD-R и CD-ROM. Но принцип записи информации на диски CD-RW совершенно другой: на металлическую или иную подходящую наносится рабочий слой из сплава серебра, индия, сурьмы и теллура, который под влиянием лазерного луча изменяет свое состояние, переходя из кристаллического состояния в аморфное или наоборот. Причем переход из одного состояния в другое может быть выполнен многократно. Поскольку кристаллические и аморфные состояния имеют различную отражающую способность, такие переходы создают возможность многократно записывать и читать двоичные коды данных.

Лазер дисковода CD-RW имеет три уровня мощности. При самой высокой мощности луч лазера расплавляет вещество рабочего слоя и переводит его в аморфное состояние. Так получается аналог впадины на CD-ROM. При средней мощности вещество переводится в кристаллическое состояние. Так получается аналог площадки. При самой низкой мощности лазера происходит считывание.

Как уже отмечалось, первые оптические дисководы, вошедшие в состав персональных компьютеров, выполняли обмен со скоростью 150 Кбайт/с. Исторически сложилось так, что эта скорость была выбрана в качестве базовой, основной единицы измерения скорости обмена не только для CD-ROM, но и для записываемых CD-R- и перезаписываемых CD-RW-дисков. Характеристика CD-R и CD-RW включает в себя два или три показателя: скорость обмена в режиме чтения (максимальный

коэффициент), скорость для режима многократной записи (минимальный коэффициент) и скорость в режиме однократной записи. Например, маркировка 12x8x32x означает, что чтение выполняется с 32-кратной базовой скоростью, запись на CD-RW — с **8**-кратной скоростью, а запись на CD-R — с 12-кратной.

Диски DVD

В настоящее время характерный для компакт-дисков объем 600-800 Мбайт уже считается довольно маленьким. Поэтому были разработаны другие способы записи информации, позволяющие при том же самом диаметре диска разместить на нем гораздо больше данных и программ. В частности, можно упомянуть предложенный в 2000 г. стандарт **DDCD** (от Double Density CD — компакт- диск двойной плотности), который предусматривает возможность записи на стандартный компакт-диск 1,3 Гбайт. Но и этот объем не может считаться значительным.

Наиболее соответствующими современным потребностям являются диски **DVD** (от Digital Versatile Disk — цифровой многосторонний универсальный диск). Они содержат несколько рабочих слоев, которые размещаются на одной и той же рабочей поверхности. Способ записи кода на каждый слой диска DVD такой же, как на рабочую поверхность у диска CD-ROM. Для кодирования данных используется сжимающий мультимедийный формат MPEG-2. В настоящее время используются четыре основных формата DVD-дисков:

- односторонние однослойные объемом 4,7 Гбайт;
- односторонние двухслойные объемом 8,5 Гбайт;
- двусторонние однослойные объемом 9,4 Гбайт;
- двусторонние двухслойные объемом до 17 Гбайт.

Относительно недавно фирма Constellation 3D представила свою новую разработку — диски FMD-ROM (от Fluorescent Multilayer Disk — многослойный флуоресцентный диск однократной записи), в которых предложено на одной рабочей поверхности стандартного диска размещать до ста рабочих слоев общей емкостью 150 Гбайт.

Основными отличиями дисков DVD от дисков CD являются следующие:

- впадины имеют размер 0,4, а не 0,8 мкм;
- плотность спирали 0,74 мкм между дорожками, а не 1,74 мкм;
- используется луч лазера с длиной волны 0,65 мкм вместо луча с длиной волны 0,78 мкм.

Для DVD-дисков базовой единицей измерения скорости обмена считается скорость 1,38 Мбайт/с. Таким образом, обозначение **8**x в маркировке DVD диска или характеристике дисковода означает, что скорость обмена равна 11,04 Мбайт/с.

В настоящее время преобладают диски DVD-ROM, хотя уже имеются довольно надежные однократно (DVD-R) и многократно записываемые (DVD-RW) диски и устройства для их записи.

Мобильные носители памяти

Уже довольно давно емкость стандартных дискет (1,44 Мбайт) считается недостаточной для выполнения основных функций — временного хранения информации и ее переноса между компьютерами. Поэтому разработчики предлагают различные варианты дешевых и надежных мобильных носителей информации.

Мобильные дисководы

Существует несколько вариантов внешних, ***мобильных дисководов*** большой емкости. Выражение «мобильный дисковод» означает, что переносить между

компьютерами нужно не только дискету, но и сам дисковод. В частности, можно упомянуть довольно популярные модели мобильных дисководов Iomega ZIP, Iomega JAZ и ORB Drive. Эти устройства используют сменные диски собственных, специализированных форматов, которые часто называют **ZIP-дис- ками.** Они имеют объем от 100 Мбайт до 2 Гбайт и допускают скорость обмена до 20 Мбайт/с.

Относительно недавно выпущены мобильные дисководы модели 2,5ᴹ Combo Portable HDD, которые используют диски диаметром 2,5 дюйма емкостью до 160 Гбайт и скоростью обмена 60 Мбайт/с.

Стоимость специализированных дисков для мобильных дисководов довольно высока. Но если учесть их высокую емкость, то использование внешних дисководов может оказаться выгоднее, чем использование оптических дисков и, тем более, стандартных трехдюймовых дискет.

Мобильные устройства флэш-памяти

В отличие от мобильных дисководов, мобильные устройства, основанные на флэш-памяти, не используют механических перемещений при выполнении операций чтения и записи. Кроме того, они не имеют сменных носителей. Фактически эти устройства представляют собой ***микросхему памяти***, снабженную защитным корпусом и разъемом USB. Однако работа с ними после подключения к порту выполняется так же, как и с жесткими дисками. В качестве примера можно упомянуть мобильное запоминающее устройство **Jet Flash** (от jet — реактивный) объемом до 4 Гбайт и скоростью обмена до **8** Мбайт/с. Это устройство выдерживает до **1 000 000** циклов чтения/записи и обеспечивает длительность хранения данных до **10** лет.

Тема 14: Системный блок и периферийные устройства

Практически все обсуждавшиеся до сих пор устройства — микросхемы памяти, кэша и процессора, шины, дисководы жестких и сменных дисков — находятся внутри корпуса компьютера, который принято называть **системным блоком.** Как правило, устройства, служащие для организации ввода/вывода, находятся вне системного блока, и потому их часто называют **периферийными** устройствами.

Системный блок

Системный блок компьютера представляет собой корпус, коробку из металла, предназначенную для установки и крепления основных устройств компьютера.

В стандартном системном блоке персонального компьютера находятся:

- системная (материнская) плата;
- платы расширений, которые используются для подключения различных внешних устройств, например сетевая плата, видеоплата, звуковая плата и т. д.;
- один или несколько жестких дисков;
- дисководы гибких дисков, дисководы CD-ROM, CD-R, CD-RW, DVD;
- тактовый генератор;
- блок питания, система вентиляторов или кулеров (от cooler — охладитель), встроенный динамик.

Системной, или **материнской,** платой (рис. 13.1) называется основная плата компьютера, по которой проходят его основные шины и на которой размещены слоты и сокеты для крепления основных внутренних устройств компьютера — микросхем процессора, чипсета, кэша, оперативной памяти, а также плат расширений. Платы расширений обычно содержат контроллер устройства и его буферную память. Они вставляются в слоты системной платы с помощью концевых разъемов, аналогичных концевым разъемам модуля памяти .

Размеры системных блоков регулируются стандартами, первым из которых считается стандарт AT (от Advanced Technology — продвинутая технология), использовавшийся для корпусов персональных компьютеров IBM PC/AT. В настоящее время большинство корпусов компьютеров выпускается в соответствии со стандартом ATX (от AT extended — улучшенный AT), хотя уже разработано несколько новых стандартов.

Ранее широко использовались системные блоки горизонтального расположения с длиной корпуса около 35 см. Сейчас в основном выпускаются системные блоки вертикального расположения, имеющие несколько разновидностей:

- minitower (мини-башня) — высота 35 см;
- miditower (средняя башня) — высота 40 см;
- bigtower (большая башня) — высота 60 см;
- superbigtower (сверхбольшая башня) — высота от **1** м.

Корпуса последнего типа используются как стойки для размещения серверов.

На передней панели системного блока компьютера находятся кнопка включения электропитания **Power** и кнопка перезагрузки **Reset.** Там же находятся две сигнальные лампочки: зеленая лампочка питания и желтая лампочка жесткого диска, — а также панели управления дисководами и выход встроенного динамика. У компьютеров, выпущенных в последние годы, на переднюю панель системного блока перенесен разъем шины USB.

На задней панели системного блока находятся: разъем для подключения электропитания, стандартные разъемы для подключения различных внешних устройств, выходы плат расширений, служащие для подключения звуковых устройств, видеоустройств (например, разъемы телевизионного входа/выхода TV In и TV Out), сетевых устройств и т. д.

В современных персональных компьютерах предусмотрены следующие разъемы, которые принято называть портами (см. 4.2.7):

- параллельный порт LPT, через который данные передаются байтами со скоростью около 2 Мбайт/с. Предназначен для подключения принтера, сканера, внешних дисководов и т. д.;
- последовательные порты COM, через которые данные передаются битами со скоростью около 100 Кбайт/с. Предназначены для подключения низкоскоростных внешних устройств, таких как мышь, клавиатура, модем и т. д.;
- последовательные порты PS/2, более современные, чем последовательный порт COM. Специально предназначены для подключения клавиатуры и мыши;
- порт USB, соответствующий унифицированному стандарту на шину со скоростью передачи данных до 12 Мбайт/с (у модификации 2.0 — до 60 Мбайт/с). К порту USB одновременно можно подключить до 127 внешних устройств. Предполагается, что со временем все устройства будут подключаться к разъемам типа USB;
- порт FireWire (или IEEE 1394), предназначенный для подключения высокоскоростных внешних устройств, таких как цифровые фото- и видеокамеры.

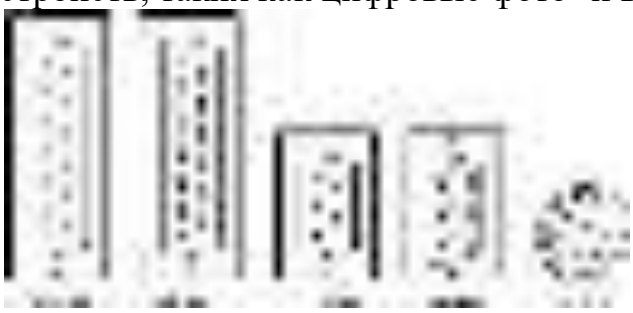


Рис. 13.2. Некоторые стандартные разъемы компьютера

В портах компьютера используются двух- и трехрядные разъемы типов S (розетка) и P (вилка) со стандартными геометрическими размерами. В частности, для LPT-порта используется двухрядный разъем DB-25S, который имеет 25 гнезд (рис. 13.2). Для COM-портов используется двухрядный разъем DB-25P, который имеет 25 штырьков, или двухрядный девятиштырьковый разъем DB-9P. А для подключения дисплея используется содержащий 15 гнезд трехрядный разъем DB-15S. Компьютер имеет два одинаковых разъема типа PS/2 круглой формы, причем разъем и штекер клавиатуры окрашены в фиолетовый цвет, а разъем и штекер мыши — в зеленый. Все эти разъемы имеют стандартные форму и размеры, исключающие случайное неправильное соединение.

Дисплей и графическая подсистема

Одним из важнейших устройств компьютера, применяющихся для вывода информации, является дисплей или монитор (от monitor — устройство для слежения, контроля). На экран дисплея выводятся данные, вводимые с клавиатуры, результаты их обработки, а также всевозможная служебная информация.

Дисплеи бывают монохромные (то есть одноцветные — черно-белые, с желтым или зеленоватым оттенком) и цветные. Кроме того, различают алфавитно-цифровые и графические дисплеи. У алфавитно-цифровых дисплеев группа пикселов,

занимающая небольшую прямоугольную область экрана и используемая для размещения изображения одного символа, образует знакоместо. Например, для растра размером 600 x 480 область, занимаемая знакоместом, образуется группой **8**x**8** пикселов. Изображение символа формируется примерно так же, как из группы точек на почтовом конверте получается изображение какой-либо цифры почтового индекса адресата. Подчеркнем, что у алфавитно-цифровых дисплеев не существует возможности работать с отдельным пикселом. Информация выводится на экран сразу целым знакоместом, символом. Поэтому такие дисплеи могут использоваться только для вывода различного рода текстов. Рисунки, графики, чертежи, картинки не могут быть выведены на алфавитно-цифровые дисплеи. В настоящее время алфавитно-цифровые дисплеи используются для управления различного рода серверами, то есть там, где отображение графики не является обязательным.

Графические дисплеи отличаются тем, что из программы можно управлять состоянием отдельного пиксела, и, следовательно, для них доступны все возможности формирования изображений.

Основными техническими характеристиками дисплеев являются:

- принцип действия;
- размер экрана по диагонали;
- разрешающая способность;
- размер «зерна» экрана;
- частота регенерации;
- форма экрана;
- класс защиты.

По принципу действия выделяют дисплеи на электронно-лучевой трубке (ЭЛТ, или CRT — от Cathode Ray Terminal, т. е. терминал на катодно-лучевой трубке), жидкокристаллические (ЖК, или LCD — от Liquid-Crystal Display, то есть жидкокристаллический дисплей) и плазменные дисплеи.

Принцип действия мониторов с электронно-лучевой трубкой в точности такой же, как у бытовых телевизоров. Электронная пушка, аналог катода в электронных лампах накаливания, вырабатывает луч — узконаправленный поток электронов, который с помощью системы отклоняющих пластин сканирует поверхность экрана дисплея. Точка пересечения луча с экраном представляет собой пиксел — элементарную единицу изображения. С помощью декодирующей схемы, на вход которой поступает закодированное изображение, пиксел переводится в одно из двух состояний — черное или белое: это позволяет формировать монохромные изображения. Для создания цветного изображения в мониторе устанавливаются три электронных пушки — красного, зеленого и голубого цвета. ЭЛТ-мониторы отличаются довольно большими габаритами, прекрасной цветопередачей и невысокой стоимостью.

Принцип действия жидкокристаллических дисплеев основан на свойствах жидких кристаллов, открытых еще в 1888 г. Они представляют собой вязкие органические молекулы, которые, с одной стороны, имеют структуру, аналогичную структуре кристалла, а с другой — ведут себя как молекулы жидкости. Оказалось, что оптические свойства жидких кристаллов зависят от ориентации молекул, а на ориентацию молекул жидкого кристалла можно воздействовать электрическим полем, что создает возможность для программно-управляемого построения изображения.

Экран LCD-дисплея состоит из двух стеклянных параллельных пластин, пространство между которыми заполнено жидкокристаллическим веществом. У жидкокристаллических дисплеев с пассивной матрицей на стеклянные пластины наносится сетка прозрачных электродов. Например, для обеспечения разрешающей способности экрана 800 х 600 сетка на задней пластине содержит 800 вертикальных проводов, а сетка на передней пластине — 600 горизонтальных. Источник света за задней пластиной освещает экран изнутри монитора. На провода сетки подается напряжение, которое различным образом ориентирует молекулы в разных точках экрана, определяя нужным образом цвет, яркость или контрастность в каждой его точке, в каждом пикселе. У жидкокристаллических дисплеев с активной матрицей вместо двух наборов сеток около каждого пиксела экрана находится крошечный элемент переключения напряжения электрического поля. Меняя соответствующим образом напряжение элемента в каждой точке, можно управлять изображением на экране.

Жидкокристаллические дисплеи отличаются малой толщиной и плоским экраном. Их стоимость пока выше, чем стоимость мониторов с электронно-лучевой трубкой. Причем мониторы с активной матрицей более качественные и более дорогие, а мониторы с пассивной матрицей имеют более бледное изображение, на них заметнее следы от смены кадров, но они и дешевле.

Самыми дорогими в настоящее время являются плазменные мониторы, которые обладают высоким качеством формируемого изображения и могут иметь значительные размеры — до **1** м и более по диагонали при толщине всего **10** см.

Перспективным направлением в развитии устройств отображения данных являются дисплеи, построенные по технологии OLED (от Organic Light Emitting Diodes — органические светодиоды). Во-первых, эти дисплеи не требуют дополнительной подсветки, так как вещество само испускает свет, а во-вторых, возможно размещение очень тонких экранов на гибкой основе.

Размер экрана дисплея по диагонали определяется в сантиметрах или дюймах. В настоящее время выпускаются мониторы с экранами от 9 до 42 дюймов или от 23 до 107 см. Наиболее распространенными являются экраны размером 15,
17, 19 и 21 дюйм. Для стандартных целей достаточно 17-дюймового экрана. При большом объеме работы с графикой желательно выбирать 19- или 21-дюймовые мониторы.

Важной характеристикой дисплеев является разрешающая способность экрана, определяющая ***степень четкости*** изображения. Разрешающая способность зависит от количества строк на весь экран и количества пикселов в строке. В настоящее время существует несколько стандартных разрешений, в частности: 800 х 600, 1024 х 768,1152 х 864,1280 х 1024,1600 х 1200,1600 х 1280,1920 х 1200,1920 х 1600, 2048 х 1536. Здесь первая цифра определяет количество пикселов в строке, а вторая — количество строк на экране. Возможное разрешение существенно зависит от фактического размера экрана. Например, для 17-дюймового монитора стандартным считается разрешение 1024 х 768, а максимальным может быть разрешение 1600 х **1200**.

Отметим, что у ЭЛТ-мониторов разрешающая способность лучше, она может достигать 2048 х 1536, в то время как у лучших ЖК-мониторов она пока значительно ниже — до 1280 х 1024. Попутно заметим, что у телевизионных приемников наилучшим на сегодняшний день считается разрешение 1024 х 768.

Качество изображения определяется не только разрешающей способностью, но и так называемой зернистостью экрана. Зернистость разными производителями определяется либо как фактический линейный размер пиксела, либо как расстояние между двумя соседними пикселами. В настоящее время этот параметр у большинства мониторов равен 0,18-0,28 мм. Чем меньше размер зерна, тем лучше, но и дороже монитор.

Частота регенерации (обновления) — это параметр, который показывает, сколько раз в секунду обновляется изображение на экране дисплея. Без такого обновления невозможно формирование нормального зрительного восприятия телевизионного изображения, а также невозможна передача движений. Если частота регенерации меньше 60 Гц, то есть если обновление происходит менее чем 60 раз в секунду, то появляется мерцание изображения, что отрицательно сказывается на зрении. В настоящее время частота регенерации большинства мониторов составляет 60-100 Гц, а стандартной считается частота 85 Гц.

Экраны мониторов бывают выпуклые и плоские. В настоящее время большинство экранов, в том числе и у бытовых телевизоров, выпуклые. Вместе с тем более перспективными моделями считаются мониторы с плоским экраном, например модель Trinitron, у которой экран абсолютно плоский по вертикали и лишь слегка искривлен по горизонтали.

С точки зрения техники безопасности работы с мониторами, необходимо учитывать класс защиты монитора, который определяется международными стандартами. В настоящее время действует стандарт под названием TCO-2004, выдвигающий самые жесткие требования к безопасному для человека уровню электромагнитных излучений, эргономическим и экологическим параметрам, а также к параметрам, определяющим качество изображения — яркости, контрастности, мерцанию, антибликовым и антистатическим свойствам покрытия экрана монитора.

Для создания изображения на экране дисплея необходим еще один компонент компьютера, который называют видеоплатой, видеокартой или видеоадаптером. Если быть точным, то это устройство следует называть графическим контроллером. Именно видеоадаптер определяет разрешающую способность монитора и количество передаваемых цветовых оттенков. Видеоадаптер вместе с дисплеем образуют видеоподсистему компьютера. В настоящее время в основном используются адаптеры типа SVGA (от Super Video Graphics Array — супервидео- графический массив), способные передавать 16,7 млн. цветовых оттенков.

Для обеспечения такого количества цветов, а также хорошего разрешения видеоадаптеры содержат собственную видеопамять довольно большого объема — 64 Мбайт и выше. Построение высококачественных изображений и, тем более, какие-либо их преобразования, как правило, требуют выполнения большого количества математических операций. Чтобы освободить процессор компьютера от действий с изображениями и тем самым существенно ускорить их построение, а также повысить общую эффективность работы компьютера, современные видеоадаптеры берут на себя значительную часть этих операций. При этом часть работы по формированию изображения возлагается на аппаратные средства адаптера — микросхемы видеоускорителя, которые могут входить в состав видеоадаптера или размещаться на отдельной плате, подсоединяемой к адаптеру. Различают два типа видеоускорителей: плоские, или 2D (от 2-dimension — двухмерный), и трехмерные, или 3D (от 3-dimension — трехмерный). Требования современных видеоадаптеров, особенно с

аппаратным ускорением, уже не удовлетворяются стандартными шинами компьютера. Поэтому для них были разработаны уже упоминавшиеся специализированные шины AGP.

Принтеры

Для получения результатов работы программ на бумаге в комплект персональных компьютеров включают печатающее устройство, которое обычно называют принтером (от print — печать). Основные технические характеристики принтеров:

- тип принтера — точечно-матричный, струйный или лазерный;
- возможность работы с цветом;ширина каретки;
- разрешение при печати;
- скорость печати;
- эксплуатационные расходы.

Существенной характеристикой любого принтера является ширина каретки, определяющая максимальные размеры, или формат документа, который может быть напечатан на данном принтере. Размер двух газетных разворотов, составляющий 840 х 1188 мм, принят в нашей стране как основа для стандартных форматов. Этот формат принято обозначать А0. Остальные форматы образуются из основного последовательным делением большего размера на два. Так, формату А1 соответствуют размеры 594 х 840 мм (две газетные страницы на одном листе бумаги), формату А2 — размеры 420 х 594 мм, формату А3 — 297 х 420 мм, А4 — 210 х 297 мм и т. д. до формата А10 с размерами 26 х 37 мм. В качестве стандартного размера листа бумаги для делопроизводства в нашей стране принят лист формата А4. Документы форматов А2 и А3 могут быть подготовлены только на принтерах с широкой кареткой. Документы формата А4 и меньше могут быть напечатаны и на принтере с узкой кареткой.

При печати графики важной характеристикой является разрешение при печати, которое измеряется в единицах dpi (от dots per inch — точки на дюйм). Один dpi равен одной точке, печатаемой на одном дюйме. Эта характеристика очень похожа на разрешающую способность мониторов, только речь идет о формировании изображения на бумаге, а не на экране монитора.

Скорость печати определяется как количество полностью отпечатанных листов или как количество печатаемых символов в единицу времени, обычно в минуту. Иногда скорость печати указывается как время, которое требуется для печати одного листа. При указании этой характеристики предполагается, что печатаемый материал не содержит графики.

Весьма важной характеристикой принтера является стоимость его эксплуатации, или эксплуатационные расходы. Современные принтеры обычно используют для выполнения печати специальное съемное устройство — картридж (от cartridge — патрон, заряд, катушка с пленками). Внутри картриджа размещается красящая лента или расходуемое при печати вещество: чернила или же специальный порошок, который принято называть тонером (от tone — тон, оттенок). Используемые при печати красящие ленты, чернила, тонеры, бумагу и т. д. называют расходными материалами. Эксплуатационные расходы включают в себя стоимость картриджа и стоимость заправки картриджа тонером или замены ленты в картридже, а также стоимость печати одного листа. Кроме того, в качестве характеристики экономичности указывается количество листов, которые можно напечатать на одной заправке картриджа.

По принципу действия используемые в настоящее время принтеры можно разделить на три группы — точечно-матричные, струйные и лазерные.

Печатающие головки точечно-матричных принтеров содержат матрицу — группу иголок, которые, выдвигаясь из головки принтера в определенных комбинациях и ударяя по красящей ленте, оставляют на бумаге изображение символа. Принтеры этой группы могут печатать со скоростью (при черновой печати) от 180 до 400 символов в секунду, при этом на печать одного листа текста требуется от 9 до 20 с. Они могут печатать не только текст, но и графику, при этом характерным является разрешение 360 x 360 dpi. Некоторые модели обеспечивают печать в цвете, а также вывод сразу нескольких копий документа. Однако качество цветной печати невысокое. Кроме того, матричные принтеры довольно сильно шумят при печати. Точечно-матричные принтеры имеют сравнительно низкую собственную цену и невысокую стоимость расходных материалов — красящих лент, картриджей, бумаги.

Печатающие головки струйных принтеров имеют несколько форсунок, капиллярных сопел, через которые на бумагу выстреливаются очень маленькие капельки чернил разных цветов. Количество сопел может быть большим — 64 и выше. Таким образом, на бумаге получается цветное точечное изображение высокого качества. Струйные принтеры обладают хорошим качеством вывода цветных и графических изображений. В настоящее время разрешение при печати достигает 2400 x 1200 dpi. Однако скорость печати струйных принтеров относительно невысока — порядка 10 страниц в минуту. Принтеры этой группы довольно сложны в эксплуатации, так как требуют специального ухода за соплами. Засыхание чернил приводит к полному выходу из строя пишущего узла принтера. Вместе с тем стоимость самих струйных принтеров относительно невысока.

В лазерных принтерах используется практически такая же технология, как и в фотокопировальных устройствах. Главной частью лазерного принтера является вращающийся барабан или лента. Перед печатью листа на барабан подается напряжение порядка 1000 В. Луч лазера, несущий закодированную строку пикселов, проходит вдоль образующей барабана. Участки, на которые попадает луч, теряют свой электрический заряд — в результате получается строка с переменным потенциалом, отображающая очередную строку пикселов печатаемого текста. После формирования будущей строки пикселов барабан поворачивается на определенный угол и входит в контакт с резервуаром, содержащим тонер, который представляет собой красящий порошок, чувствительный к электростатическому полю. Тонер притягивается к заряженным точкам барабана. После еще одного поворота барабан с подготовленной строкой пикселов прижимается к бумаге, оставляя на ней изображение. Затем лист бумаги проходит через горячие валики, частицы тонера расплавляются и внедряются в бумажный лист, закрепляя сформированное изображение. После этого участок барабана, который несет напечатанную строку, разряжается от статического электричества и очищается от остатков тонера.

Для хранения кодов всех строк пикселов печатаемой страницы лазерные принтеры имеют собственную память, объем которой доходит до нескольких сотен мегабайт. Для лазерных принтеров характерны: высокая скорость печати — до 30 и более страниц в минуту, высокое, сравнимое с полиграфическим, качество печати цвета и графики при разрешающей способности до 2400 x 2400 dpi. Лазерные принтеры, особенно цветные, имеют относительно высокие собственную стоимость и стоимость эксплуатации.

Недавно компания IBM представила лазерный принтер Infoprint 4100, который печатает со скоростью 330 страниц в минуту. Такой принтер всего за две минуты способен отпечатать целый том романа Л. Н. Толстого «Война и мир». Принтер печатает документы на бумажных рулонах, которые затем разрезаются на страницы с помощью другого устройства.

Другие устройства компьютера

К основным устройствам ввода относится клавиатура, которая используется для первичного ввода данных в компьютер, а также для управления его работой. В настоящее время клавиатуры подавляющего большинства персональных компьютеров унифицированы и выполнены в стандартах 101/102-, 104/105- или 108-клавишных клавиатур. Вместе с тем формы и дизайн клавиатур довольно разнообразны. Можно отметить распространение в последнее время беспроводных клавиатур с инфракрасной передачей данных, которые не требуют подсоединения к системному блоку с помощью кабеля. Заметим, что клавиатуру вместе с дисплеем (а иногда и только клавиатуру) называют консолью.

К группе устройств ввода относится манипулятор «мышь», который обеспечивает удобный уровень управления работой программы и используется для ввода простейших видов графической информации — рисунков, чертежей и т. д. Мыши бывают механические, оптические, проводные и беспроводные (инфракрасные).

Устройство, которое называется джойстик, по принципу действия похоже на мышь. Вместо коробочки мыши, произвольно перемещаемой по поверхности стола, у джойстика имеется вертикальная рукоятка, один конец которой зафиксирован в шарнирном механизме, позволяющем произвольным образом наклонять и вращать ручку. Изменения в положении ручки джойстика соответствуют изменениям в положении корпуса мыши. Как и у мыши, у джойстика имеются кнопки, с помощью которых осуществляется управление программой. Существуют и специализированные разновидности джойстиков, предназначенные для имитации в игровых программах действий по управлению автомобилем или самолетом. В частности, они могут быть выполнены в виде руля автомобиля или штурвала самолета в комплекте с несколькими педалями.

Существует и еще одно устройство, принцип действия которого аналогичен принципу действия мыши. Это устройство называется пенмаус. Внешне пенмаус похож на шариковую ручку, на рабочем конце которой находится узел, регистрирующий ее перемещения.

В ноутбуках широко используется трекбол, представляющий собой стационарно устанавливаемый корпус, на верхней панели которого имеется шарик, приводимый в движение рукой. Можно считать, что трекбол — это перевернутая неподвижная мышь, шарик которой вращается пользователем непосредственно, а не при перемещении корпуса устройства.

Для ввода в память машины изображений, более сложных, чем простые рисунки, которые, как карандашом на листе бумаги, можно рисовать на экране мышью, разработаны сканеры. Сканер используется для передачи в память машины уже имеющихся на бумажном носителе чертежей, фотографий, иллюстраций и т. д. Оптическое устройство сканирует изображение, просматривая его узкими горизонтальными полосками, и сформированный сканером машинный код этого изображения передается в память.

Основными техническими характеристиками сканеров являются возможный формат сканируемого листа бумаги — А2, А3 или А4, и разрешающая способность, равная количеству точек, различаемых сканером на одном дюйме, и измеряемая в единицах dpi. Разрешающая способность сканеров в зависимости от модели изменяется в диапазоне от 300 до 4800 dpi. В некоторых случаях разрешения по горизонтали и вертикали у сканера отличаются друг от друга, тогда разрешающая способность указывается двумя значениями, например 300 x 600 dpi.

Для работы со сканером разработаны программы, позволяющие не только передавать в память машины изображение печатного или рукописного текста, но и распознавать этот текст. С их помощью можно не только факсимильно (от лат. fac simile — делай подобное), то есть точно, без изменений, воспроизводить такой текст на экране или на бумаге, но и работать с ним, как с текстом, набранным с помощью клавиатуры. Например, можно проверить его синтаксическую правильность, отредактировать или преобразовать в печатный документ.

Для ввода графической информации в память компьютера применяются также дигитайзеры, или графические планшеты. В основе их действия лежит фиксация положения специального пера относительно планшета или экрана дисплея. В последнем случае перо называется световым. Дигитайзеры могут быть использованы художниками для создания всевозможных рисунков, иллюстраций ***без промежуточного нанесения*** на бумагу или иной традиционный носитель. В отличие от сканера, который используется для передачи в память компьютера уже готового изображения, дигитайзеры служат для сохранения изображения в процессе его создания.

Возможности вывода графической информации на современных принтерах довольно велики. Несмотря на это существуют и специализированные устройства — графопостроители, или плоттеры, которые используются для подготовки на бумаге различного рода конструкторских документов, чертежей, графиков, рисунков. Они обеспечивают работу с цветом и документами очень больших форматов.

Для работы со звуком в мультимедийной среде к аппаратуре компьютера предъявляются определенные требования. Эти требования сформулированы в стандарте **MPC** (от Multimedia Personal Computer). В частности, в комплект машины должны входить: акустические стереоколонки, микрофон и звуковой адаптер (звуковая плата, звуковая карта), который связывает различные акустические устройства с компьютером. Звук может быть введен как с микрофона, так и с любого создающего звук устройства — магнитофона, проигрывателя, телевизора и т. д.

Среди менее распространенных периферийных устройств можно упомянуть синтезаторы звука, которые используются для воспроизведения звуков различных музыкальных инструментов, а также музыкальные приставки, применяемые профессиональными музыкантами для создания и аранжировки музыкальных произведений.

Среди перспективных периферийных устройств компьютера следует указать на речевой ввод и вьюод. Они представляют собой устройства, распознающие голос человека и «понимающие» сказанные им фразы, а также устройства, синтезирующие человеческую речь.

Два класса устройств, ограничители напряжения и источники бесперебойного питания, используются для обеспечения надежного электропитания компьютеров. Устройства первой группы просто сглаживают резкие скачки напряжения в сети до

приемлемого уровня, а устройства второй группы на некоторое время обеспечивают компьютер автономным электропитанием при его внезапном отключении и позволяют в это время сохранить результаты текущей работы.

Компактная условная формула — характеристика компьютера

В стандартный комплект персонального компьютера при его продаже обычно входят системный блок, клавиатура, мышь и монитор. При необходимости отдельно приобретаются принтер, сканер, модем и другие периферийные устройства. В настоящее время для описания продаваемого компьютера используется краткая условная формула, дающая довольно полное представление об основных технических характеристиках машины. В эту формулу входят:

1) модель процессора, его тактовая частота, наличие и объем кэша;
2) объем и тип оперативной памяти;
3) используемые шины;
4) объем жесткого диска;
5) наличие флоппи-дисковода;
6) наличие и тип дисковода компакт-дисков;
7) наличие видеоплаты, ее тип и объем дополнительной памяти;
8) наличие звуковой платы, ее тип;
9) стандарт клавиатуры и наличие в комплекте мыши.

Технические характеристики устройств компьютера обычно перечисляются именно в таком порядке. Если какая-либо характеристика не указывается, то ее отсутствие не влияет на порядок задания остальных параметров. Возьмем, например, следующую «формулу» компьютера:

P4-2400, 512 L2/256 DIMM DDR/PCI, USB/80 Gb/FDD 3,5”/CD-RW52x/AGP4x 32МЬ/ SB AUDIGY 5.1 /Mouse/Keyboard 108

Формула всегда начинается с указания типа процессора и его тактовой частоты. В данном случае P4-2400 означает, что основу компьютера составляет процессор типа Pentium 4 с тактовой частотой 2400 МГц, или 2,4 ГГц, и внешним кэшем объемом 512 Мбайт (512 L2). Далее указываются объем и тип оперативной памяти: 256 DIMM DDR означает двухрядный модуль (DIMM) оперативной памяти типа DDR SDRAM объемом 256 Мбайт. Используемые в компьютере специализированные шины указываются не всегда. В нашем примере они указаны — это шины типа PCI и USB. Зато объем жесткого диска указывается всегда — это очень важный показатель. В рассматриваемом примере указано, что объем жесткого диска равен 80 Гбайт (80 Gb). Иногда добавляются тип интерфейса жесткого диска (EIDE или SCSI) и скорость его вращения (например, 7200 об./мин). Далее в формуле показано наличие в комплекте дисковода гибких дисков (FDD 3,5”) и дисковода оптических дисков с возможностью многократной перезаписи (CD-RW) и 52-кратной скоростью обмена (52x). В состав системного блока также входит видеоплата типа AGP со скоростью обмена

1, 06 Гбайт/с (4x) и дополнительной памятью объемом 32 Мбайт. Имеется звуковая плата модели SB (Sound Blaster) AUDIGY 5.1. Завершается формула указанием на наличие в комплекте поставки мыши (Mouse) и 108-клавишной клавиатуры (Keyboard 108). Заметим, что наличие двух последних компонентов обычно подразумевается и в формуле специально не оговаривается.

Монитор характеризуют отдельно указанием модели, типа, размера экрана, размера зерна, разрешения и частоты, а также наличием сертификата класса защиты, например: Samsung 763MB/CRT/17"/0,20/1280xl024@85/TC099. В данном случае

речь идет о мониторе на электронно-лучевой трубке (CRT) модели Samsung 763MB с 17-дюймовым экраном (17”), размером зерна 0,20 мм, разрешением 1280 x 1024, частотой регенерации 85 Гц (@85) и наличием сертификата класса защиты TCO-99.
Выбирая компьютер, необходимо обращать внимание на то, чтобы технические характеристики устройств были ***сбалансированы.*** Бессмысленно устанавливать мощный процессор на системную плату, шина памяти которой имеет низкую пропускную способность, с малым объемом оперативной памяти и без внешнего кэша. Не менее важную роль в общей производительности компьютера играют возможности чипсета и дисковой подсистемы.

Тема 15: Оценка производительности вычислительных систем

Ранее в качестве приближенной меры производительности (мощности) вычислительных систем мы использовали две важные характеристики: тактовую частоту центрального процессора и объем оперативной памяти. Подобный подход неплохо работал в вычислительных машинах, в которых не было многоуровневого кэша, конвейеров, «хитрых» методов улучшения загрузки конвейера, суперска- лярных процессоров и т. д. Но дальнейшее развитие архитектуры компьютера показало, что ситуация с измерением мощности компьютера далеко не так проста.

Проиллюстрируем это замечание наглядным примером, взятым из [11]. В 5.1 приведены данные машины EDSAC, считающейся первой машиной первого поколения (год выпуска 1949): тактовая частота 0,5 МГц, производительность 100 арифметических операций в секунду. Центральные процессоры вычислительной системы Hewlett-Packard Superdome в 2002 г. работали на частоте 770 МГц, а ее производительность оценивалась в 192 млрд. арифметических операций в секунду. То есть тактовая частота выросла «всего» в 1540 раз, в то время как производительность увеличилась почти в 2 миллиарда раз. «Дополнительный» прирост обеспечен не улучшением физических характеристик процессора и других центральных устройств компьютера, а широчайшим внедрением параллелизма и сопутствующих архитектурных решений, развитием математических и алгоритмических методов, а также соответствующего программного обеспечения.

Тщательный анализ показывает, что производительность вычислительной системы зависит от множества факторов, в том числе:

- скоростных характеристик и разрядностей системы шин компьютера;
- скоростных характеристик и объема внешних запоминающих устройств, особенно магнитных дисков;
- устройств, обеспечивающих обмен данными между входящими в вычислительную систему процессорами;
- возможностей используемой операционной системы, ее «умения распорядиться» возможностями аппаратуры, а особенно того, как организована параллельная работа центральных процессоров;
- «умения» трансляторов подготовить машинный код программы к исполнению в параллельной среде — на нескольких функциональных блоках, конвейерах, процессорах и т. д.;
- возможностей организации параллельного исполнения программы, имеющихся в используемых языках программирования;
- мощности применяемых алгоритмических и математических методов, того, насколько удачным оказался выбранный способ распараллеливания задачи, то есть способ выделения участков, предназначенных для одновременного, параллельного исполнения на нескольких процессорах или компьютерах вычислительной системы;
- степени соответствия имеющихся аппаратных средств и выбранного способа распараллеливания;
- и наконец, совсем уже плохо контролируемого фактора — от возможностей распараллеливания, которые заложены в «природу» самой решаемой задачи.

В связи с наличием такого огромного количества факторов, влияющих на производительность вычислительной системы, и настоятельной необходимостью каким-то образом все-таки ее оценивать, в настоящее время используется несколько

способов указания мощности компьютеров. Если отбросить детали, таких способов всего три:

- оценка с помощью тактовой частоты;
- оценка с помощью указания количества операций, выполняемых в единицу времени;
- тестирование на специально отобранных программах.

Оценка производительности тактовой частотой

Тактовая частота используется как характеристика процессора в тех случаях, когда требуется только приблизительная оценка мощности, например для описания персональных компьютеров, применяющихся для решения офисных задач, для развлечений и в других целях. Чем выше тактовая частота, тем быстрее выполняются команды, тем большее количество команд в единицу времени выполняет процессор, тем выше его производительность. Применение тактовой частоты для оценки мощности облегчается тем, что это довольно легко измеряемый и воспринимаемый параметр. В случае многопроцессорных систем тактовая частота используется как дополнительная характеристика отдельного процессора, входящего в систему.

Использовать тактовую частоту для реального представления о производительности компьютера сложно, так как нужно дополнительно знать множество факторов, например среднее количество тактов, приходящихся на одну машинную команду, количество ступеней конвейера, количество функциональных блоков в суперскалярном процессоре, параметры всех уровней кэша и т. д. Одновременный учет всех этих факторов — довольно сложная задача. Особенно слабое представление дает тактовая частота о производительности многопроцессорных вычислительных систем.

Пиковая и реальная производительность

Для многих способов оценки различают ***пиковую*** (от peak — высшая точка) и ***реальную*** производительность вычислительной системы. Пиковая производительность представляет собой полученную теоретическим путем верхнюю оценку мощности вычислительной системы, а реальная производительность определяется экспериментальным путем, во время выполнения реальных программ. Пиковую производительность рассчитывают в предположении, что при выполнении программы все устройства компьютера работают на максимальном уровне своих возможностей. К пиковой производительности можно подойти довольно близко, но достичь ее в реальных условиях невозможно. Пиковая производительность для любой вычислительной системы рассчитывается однозначно, однако она слабо связана с конкретными показателями, которые могут быть достигнуты для конкретных задач: одних задачах это может быть 90 % процентов от пиковой, а на других — только 5-10 %.

Единицы MIPS

Для более точной характеристики мощности вычислительных систем используется подход, основанный на указании количества машинных команд, выполняемых системой в единицу времени. Отметим, что эту характеристику можно использовать и для оценки производительности многопроцессорных машин, если учитывать количество команд, выполняемых всей системой.

В этом подходе оценка производительности вычислительных систем производится в единицах MIPS (от Million Instructions Per Second — миллион машинных команд в секунду), в которых мощность компьютера равна отношению количества выполненных машинных команд (инструкций) ко времени их выполнения. Отличие

этого способа оценки производительности в том, что в расчетах не различают формат данных, над которыми выполняет действие центральный процессор, то есть используется реальная смесь команд программы, состоящая из действий и над целочисленными, и над вещественными данными. Очевидное удобство этого способа — в его простоте и интуитивной понятности.

Основной недостаток использования единиц MIPS — в том, что результат зависит от системы команд процессора. Поэтому сложно сравнивать производительности компьютеров с разными системами машинных команд. Кроме того, известно, что различные команды выполняются процессором за разное время, а разные программы содержат «быстрые» и «медленные» команды в различных пропорциях. Следовательно, при выполнении на одном и том же компьютере разных программ можно получить разные оценки его производительности, что также препятствует широкому использованию этого показателя.

14.1. Единицы Flops

Альтернативной единицей измерения производительности вычислительных систем являются флопы, или единицы Flops (от Floating point operation per second — операции с плавающей точкой в секунду). В этом случае производительность системы равна отношению количества операций над вещественными данными (в формате с плавающей точкой) ко времени их выполнения. В современных условиях более часто используются кратные единицы: мегафлопы (1 Mflops = 10^6 Flops), гигафлопы (1 Gflops = 10^9 Flops), терафлопы (1 Tflops = 10^{12} Flops).

Эта единица измерения отличается от предыдущей двумя особенностями. Во-первых, при измерении в единицах Flops учитываются только операции над вещественными данными, а во-вторых, в оценке участвуют не машинные команды процессора, а выполненные операции над вещественными числами. Разница в том, что одна операция над вещественными числами (например, умножение или извлечение квадратного корня) может быть задана различными последовательностями машинных команд. Количество операций над вещественными числами зависит только от решаемой задачи и не зависит от реализующей ^вычисления машинной программы. Поэтому измерение в единицах Flops более объективно отражает производительность компьютера.

К сожалению, вне операций над вещественными данными эта система оценки производительности неприменима, так как для программ, слабо, использующих или вообще не использующих вычисления с вещественными данными (например, для программ-компиляторов), показатель производительности в единицах Flops оказывается очень малым.

У этого способа, так же как и у предыдущего, имеется недостаток, проявляющийся в существенной зависимости производительности системы от выполняемой программы. Как и в предыдущем случае, это объясняется различным соотношением между «быстрыми» и « медленными» операциями, но теперь уже не в программе, а в решаемой задаче. Кроме того, для программ с короткими циклами, когда все команды цикла могут одновременно находиться в кэше, производительность машины оказывается выше, чем для программы с циклами, в которых приходится обращаться к оперативной памяти. А для программ, в которых можно организовать много параллельных ветвей, например для программ, работающих с матрицами, производительность многопроцессорной системы окажется существенно выше, чем ее же производительность во время выполнения программы, не допускающей

распараллеливания.

Тесты UNPACK

Из-за отмеченных недостатков единиц MIPS и Flops для сравнения производительности компьютеров было предложено использовать в качестве критерия время выполнения специально подобранной эталонной программы или же связанные с этим временем показатели. Программы, на которых осуществляется тестирование, иногда называют ***бенчмарками*** (от bench-mark — отметка уровня). К настоящему времени создано довольно много различных тестовых и эталонных программ.

Одной из наиболее известных систем оценки является тест LINPACK, представляющий собой пакет программ на языке Фортран, предназначенных для решения систем линейных алгебраических уравнений больших размерностей (до нескольких миллионов неизвестных) с плотной матрицей методом Гаусса с выбором главного элемента. Имеется несколько разновидностей этого теста, например LINPACK TPP (от Toward Peak Performance — направляющийся к пиковой производительности) и HPL (от High-Performance LINPACK — высокопроизводительный LINPACK).

Для проведения тестирования формируется некоторая система линейных уравнений максимально возможной для имеющегося объема размерности и измеряется время ее решения на тестируемой вычислительной системе. Количество операций с вещественной точкой $\boldsymbol{K}$, которые должны быть выполнены для получения решения, равно $\boldsymbol{K} = 2гг^3/3 + 2гг^2$, оно однозначно зависит от заданной размерности матрицы гг, поэтому дальнейшая оценка производительности в единицах Flops не вызывает затруднений.

Лабораторные работы

Лабораторная работа №1
Системы счисления. Преобразование чисел.

Цель работы: приобрести навыки преобразования чисел различных систем счисления.

Задание: перевести числа согласно варианту задания в другие системы счисления (двоичная, восьмеричная, десятичная, шестнадцатеричная). Результат проверить на калькуляторе.

Вариант	$X_{(2)}$	$X_{(8)}$	$X_{(10)}$	$X_{(16)}$
1	110011,001	24,4	125,375	DA,3
2	111000,011	17,6	205,5	2C,A
3	100001,111	63,2	222,75	5A,6
4	100111,101	36,6	108,25	99,B
5	110111,010	26,7	89,125	8D,F
6	011111,111	42,1	166,5	C6,1
7	011010,001	47,3	148,875	E0,7
8	101111,100	32,4	133,625	8C,3
9	111111,010	72,7	244,25	3F,B
10	111001,101	50,5	176, 75	A2,8
11	101010,111	40,4	76,875	B8,9
12	010111,011	30,3	95,375	A0,C
13	011001,001	20,2	130,625	38,F
14	100100,100	76,1	270,5	4E,A

Лабораторная работа №2

Формы преобразования данных. Специальное кодирование.

Цель работы: научиться применять специальное кодирование чисел при выполнении арифметических операций.

Задание:

1) Записать в таблицу 1 свое фамилии и имя в коде ASCII
2) Выполнить сложение и вычитание чисел X_1 и X_2 таблицы 3 в двоичной системе счисления при помощи прямого, обратного и дополнительных кодов. Все необходимые вычисления предоставить в отчете. Результаты записать в таблицу 2.

Таблица 1

Номер байта	1	2	3	4	5	6	7	8	…
Символ									
Код ASCII									

Таблица 2

	Десятичная система счисления	Двоичная система счисления
X_1		
X_2		
$X_1 + X_2$		

$X_1 - X_2$		

Таблица 3

Вариант	X_1	X_2
1	73	40
2	62	25
3	28	57
4	35	80
5	15	115
6	97	38
7	121	26
8	55	79
9	99	43
10	134	22
11	67	127
12	108	52
13	101	114
14	17	72
15	140	27

Лабораторная работа №3

Работа и особенности логических элементов ЭВМ

Цель работы: Освоить работу логических элементов.

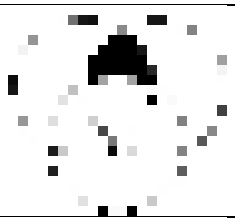
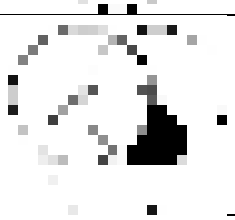
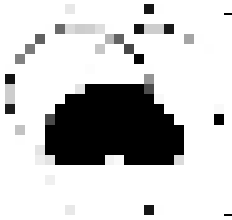
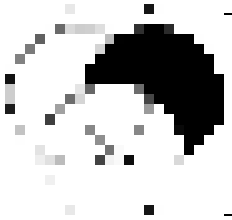
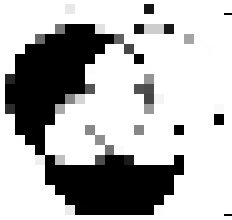
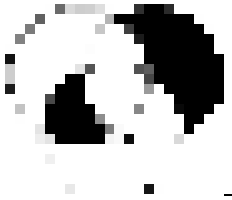
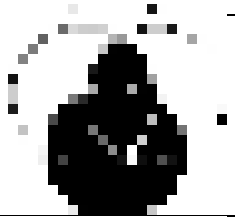
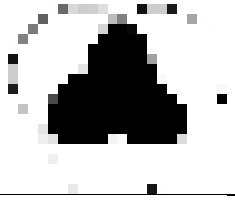
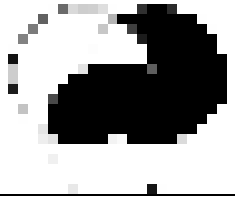
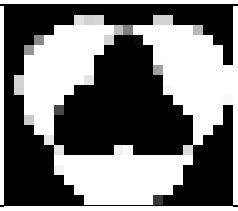
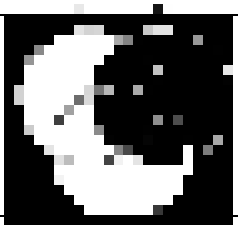
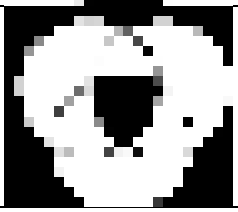
Задание:

3) Изобразить выражения согласно варианту (таблица 1) на диаграммах Эйлера.
4) Описать логическим выражением диаграмму Эйлера согласно варианту (таблица 2).
5) Построить временные диаграммы работы элементов И, ИЛИ, НЕ, исключающее ИЛИ. Записать таблицы истинности.

Таблица 1

Вариант	Выражения
1	A∨(B&C) A-(B∨$\overline{C}$)
2	A⊕$\overline{B-C}$ A∨$\overline{B}$∨C
3	A&$\overline{B}$&$\overline{C}$ A∨B-C
4	A∨B⊕C $\overline{A}$∨(B&C)
5	$\overline{A\&B\&C}$ A∨($\overline{B}$⊕$\overline{C}$)
6	A&B-C A⊕$\overline{B\vee C}$
7	A⊕B∨$\overline{C}$ A-(B&C)
8	$\overline{A\&\overline{B\vee C}}$ (A-B)∨C
9	A-B-C $\overline{A\vee\overline{B}\&\overline{C}}$
10	A-(B&C) $\overline{\overline{A}\&B}$⊕C
11	$\overline{A}$⊕B∨C A-$\overline{B-C}$
12	A&B&C $\overline{A\vee B}$-C
13	A⊕B⊕C $\overline{A}$⊕$\overline{B-C}$
14	(A⊕B)∨$\overline{C}$ $\overline{A\&B}$⊕$\overline{C}$
15	$\overline{A}$&B-$\overline{C}$ A⊕(B∨C)

Таблица 2

Вариант	Выражения
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Лабораторная работа №4

Работа логических узлов ЭВМ

Цель работы: Освоить работу логических узлов ЭВМ.

Задание:

6) Построить схему по заданной логической функции.
7) Преобразовать выражение согласно варианту (таблица 1) в базисы 2И-НЕ с помощью законов Де-Моргана и построить схему для полученной логической функции.
8) Нарисовать заданное устройство согласно варианту (таблица 2), построить временные диаграммы работы данного устройства.

Таблица 1

Вариант	$f(x_1, x_2, x_3, x_4)$
1	$\overline{x_1}\, x_2 \vee \overline{\overline{x_1}\,\overline{x_3}}$
2	$x_1 \overline{x_3} \vee \overline{x_2}\,\overline{x_3}$
3	$\overline{x_1\, x_3} \vee x_2\, \overline{x_4}$
4	$\overline{\overline{x_2}\,\overline{x_3}} \vee \overline{x_1\, x_2}$
5	$\overline{x_1}\, x_2 \vee x_3\, \overline{x_4}$
6	$\overline{x_1\, x_2} \vee x_2\, x_3$
7	$x_1\, \overline{x_4} \vee \overline{x_2\, \overline{x_4}}$
8	$x_2\, \overline{x_3} \vee x_2\, x_4$
9	$\overline{x_1\, x_4} \vee \overline{x_2\, x_3}$
10	$x_1\, x_3 \vee \overline{x_1}\, x_2$
11	$\overline{x_3\, \overline{x_4}} \vee x_2\, x_3$
12	$\overline{x_1}\, x_2 \vee \overline{x_1\, x_4}$
13	$\overline{x_1\, \overline{x_2}} \vee \overline{\overline{x_2}\,\overline{x_4}}$
14	$x_1\, \overline{x_3} \vee \overline{x_2}\, x_4$
15	$\overline{x_1}\,\overline{x_3} \vee \overline{x_1}\,\overline{x_4}$

Таблица 2

Вариант	Устройство
1	Дешифратор 3:8
2	Шифратор 8:3
3	Мультиплексор 8:1
4	Демультиплексор 1:8
5	RS-триггер
6	JK-триггер
7	D-триггер
8	T-триггер
9	Полусумматор
10	Мультиплексор 4:1
11	Демультиплексор 1:4
12	Сумматор (3 разряда)
13	Дешифратор 2:4
14	Шифратор 4:2
15	Одноразрядный сумматор

Лабораторная работа №5

Основные характеристики процессоров различных архитектур

Цель работы:

1) Выяснить области применения существующих процессоров на основе их архитектур.
2) Выделить основные характеристики существующих процессоров.

Задание:

9) Составить таблицу поколений ЭВМ по шаблону таблицы 1.
10) Составить таблицу характеристик процессоров Intel и AMD по шаблону таблицы 2.
11) Начертить график динамики роста тактовой частоты процессоров.
12) Начертить график динамики роста количества транзисторов. Сделать вывод (правило Мура).

Таблица 1

Показатель	Поколения ЭВМ			
	Первое	Второе	Третье	Четвертое
	1951-1954	1958-I960	1965-1966	1976-1979
Элементная база процессора				
Элементная база ОЗУ				
Максимальная емкость ОЗУ, байт				
Максимальное быстродействие процессора (оп/с)				
Языки программирования				
Средства связи пользователя с ЭВМ				

Таблица 2

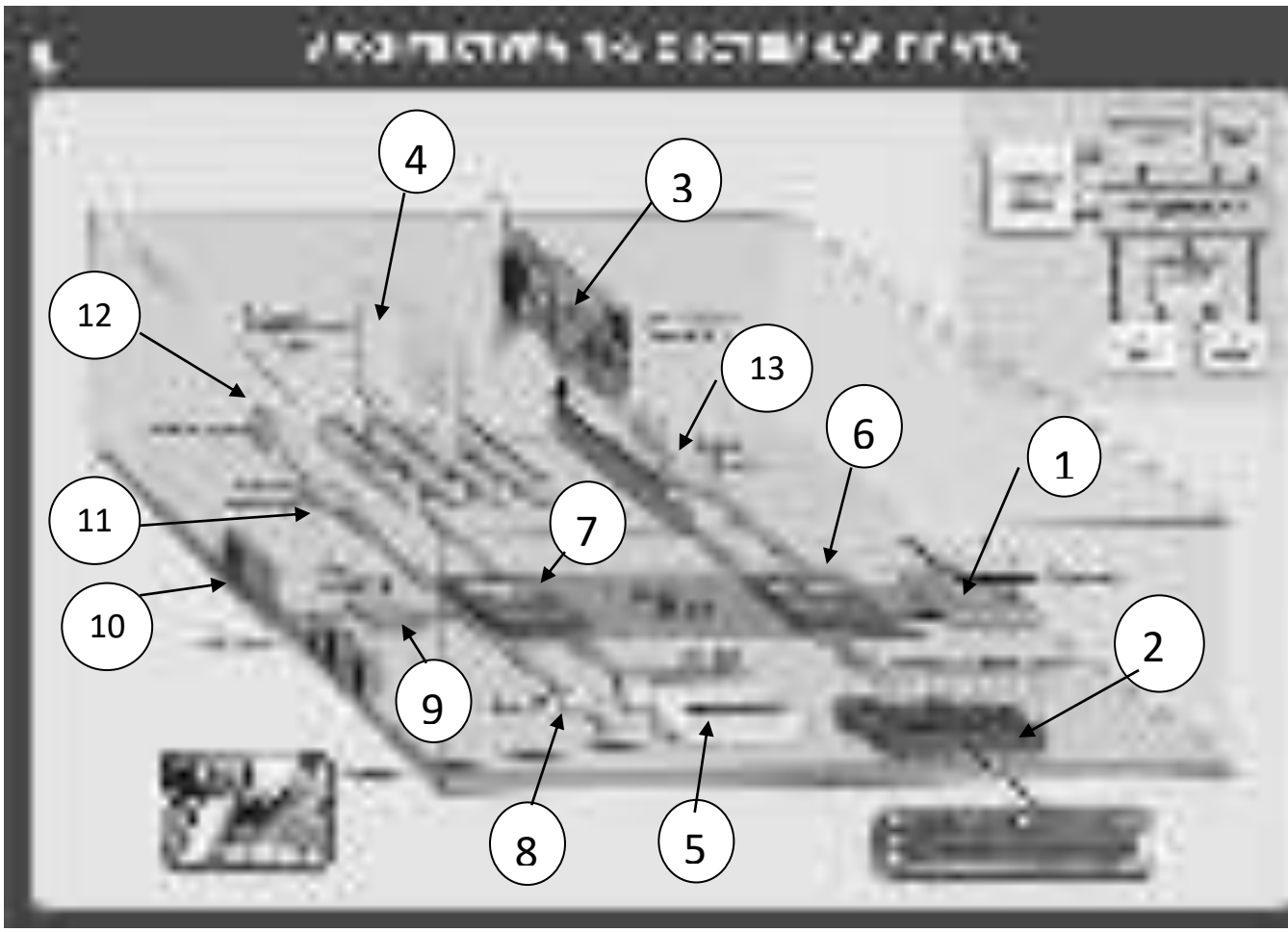
Тип процессора	Поколение	Год выпуска	Разрядность шины данных	Разрядность шины адреса	Первичная кэш-память, Кбайт		Тактовая частота шины, Мгц	Так част про Мг
					Команды	Данные		
4004	1	1971	4	12	Нет		0,75	
8088								
8086								
80286								
80386DX								
80386SX								
80486DX								
80486SX								
80486DX2								
80486DX4								
Pentium								
P-MMX								
Pentium Pro								

Pentium II							
Pentium II Celeron							
Pentium Xeon							
Pentium III							
AMD Athlon							
Pentium 4							
AMD Athlon 64							

Лабораторная работа №6
«Архитектура системной платы»

Цель работы: Изучение архитектуры системной платы.

Задание: рассмотреть представленную материнскую плату и указать основные компоненты а также их назначение.



Ход работы:

Системная плата – печатная плата, соединяющая все узлы компьютера в одно устройств Кроме термина "системная плата", используется название "материнская плата".

Основные компоненты системной платы:

1) разъем для подключения микропроцессора; Процессор – основная часть ЭВМ обеспечивающая выполнение процедур обработки данных и взаимодействие все устройств ЭВМ.
2)

Лабораторная работа №7
«Внутренние интерфейсы системной платы»

Цель работы: Изучение внутренних интерфейсов системной платы.

Задание 1

Идентифицируйте внутренние интерфейсы системной платы.

Задание 2

Дайте сравнительную характеристику внутренних интерфейсов целевой системной платы.

Методические указания:

Интерфейс – это совокупность средств сопряжения и связи, обеспечивающая эффективное взаимодействие систем или их частей. В интерфейсе обычно предусмотрены вопросы сопряжения на механическом (число проводов, элементы связи, типы соединений, разъемы, номера контактов и т.п.) и логическом (сигналы, их длительность, полярность, частоты и амплитуда, протоколы взаимодействия) уровнях.

Внутренний интерфейс – это система связи и сопряжения узлов и блоков компьютера между собой. Представляет собой совокупность электрических линий связи, схем сопряжения с компонентами компьютера, протоколов (алгоритмов) передачи и преобразования сигналов.

В современных компьютерах в качестве системного интерфейса обычно используется системная шина.

Шина (bus) – это совокупность линий связи, по которым информация передается одновременно. Под основной или системной шиной понимается шина между процессором и подсистемой памяти. Шины характеризуются разрядностью и частотой.

Разрядность или ширина шины (bus width) – количество линий связи в шине, т.е. количество битов, которое может быть передано по шине одновременно.

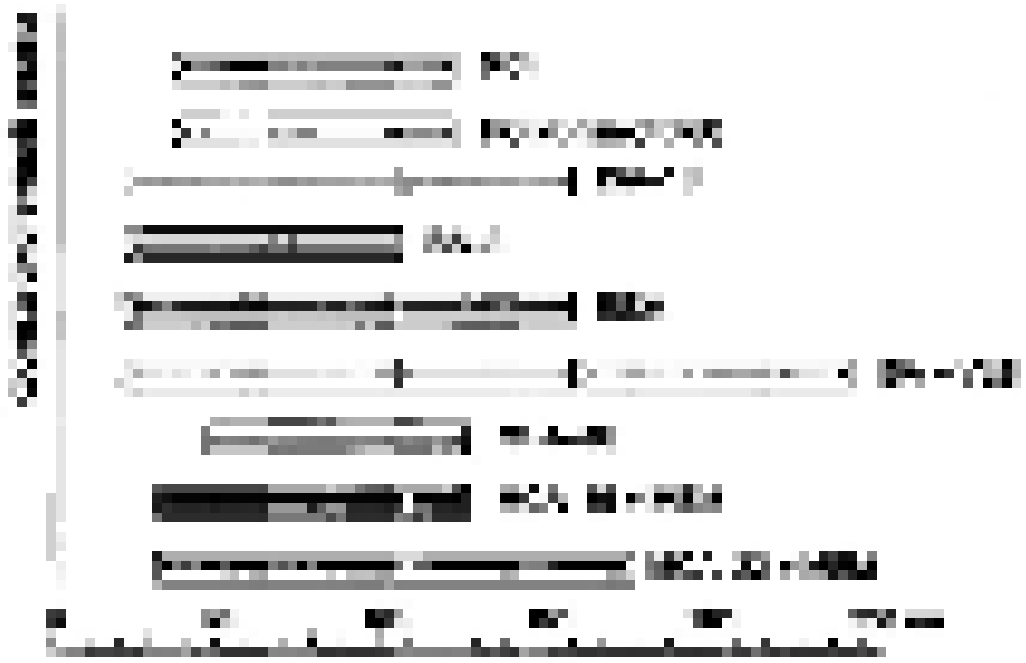
Тактовая частота шины (bus frequency) – частота с которой передаются последовательные биты информации по линиям связи.

В качестве системной шины в ПК могут использоваться шины расширений и локальные шины.

Шины расширений – шины общего назначения, позволяющие подключать большое количество самых разнообразных устройств.

Локальные шины специализируются на обслуживании небольшого количества устройств определенного класса.

Шины расширений



Шина расширения ISA (Industry Standard Architecture) – основная шина на устаревших материнских платах. Служит для подключения витдеокарт, модемов, звуковых карт и т.д. Конструктивно представляет собой разъем состоящий из двух частей – 62-контактного и примыкающего к

Лабораторная работа №8
«Интерфейсы периферийных устройств IDE, SCSI, SATA»

Цель лабораторной работы**:**

– Изучение интерфейсов периферийных устройств;

Методические указания:

Периферийные шины используются в основном для внешних запоминающих устройств.

Интерфейс IDE (Integrated Drive Electronics) – интерфейс устройств со встроенным контроллером. Поддерживает несколько способов обмена. Первый способ производит обмен данными через регистры процессора под его непосредственным управлением. Следствием этого является высокая загрузка процессора при операциях ввода/вывода. Вторым способом является использование режима прямого доступа к памяти, при котором контроллер интерфейса IDE и контроллер прямого доступа к памяти материнской платы пересылают данные между диском и оперативной памятью, не загружая центральный процессор. В целях развития возможностей интерфейса IDE была предложена его расширенная спецификация EIDE (синонимы ATA, ATA-2). Она поддерживает накопители емкостью свыше 504 Мбайт, поддерживает несколько накопителей IDE и позволяет подключать к одному контроллеру до четырех устройств, а также поддерживает периферийные устройства, отличные от жестких дисков. Расширение спецификации IDE для поддержки иных типов накопителей с интерфейсом IDE называют также ATAPI.

SATA (Serial ATA) – последовательный интерфейс обмена данными с накопителями информации. Для подключения используется 8-pin разъем. SATA является развитием параллельного интерфейса ATA (IDE), который после появления SATA был переименован в PATA (Parallel ATA).Стандарт SATA (SATA150) обеспечивал пропускную способность равную 150 МБ/с (1,2 Гбит/с).

SATA 2 (SATA300). Стандарт SATA 2 увеличивал пропускную способность в двое, до 300 МБ/с (2,4 Гбит/с), и позволяет работать на частоте 3 ГГц. Стандартны SATA и SATA 2 совместимы между собой, однако для некоторых моделей необходимо вручную устанавливать режимы, переставляя джамперы.

SATA 3, хотя про требованию спецификаций правильно называть **SATA 6Gb/s**. Этот стандарт в двое увеличил скорость передачи данных до 6 Гбит/с (600 МБ/с). Также к положительным нововведениям относится функция программного управления NCQ и команды для непрерывной передачи данных для процесса с высоким приоритетом.

Лабораторная работа №9

Параллельные и последовательные порты и их особенности работы

Цель лабораторной работы:

– Изучение особенностей работы параллельных и последовательных портов

Порт (персонального) компьютера предназначен для обмена информацией между устройствами, подключенными к шине внутри компьютера и внешним устройством.

Для связи с периферийными устройствами к шине компьютера подключены одна или несколько микросхем контроллера ввода-вывода.

Последовательный порт стандарта RS-232-C. Является стандартом для соединения ЭВМ с различными последовательными внешними устройствами. В операционных системах каждому порту RS-232 присваивается логическое имя COM1-COM4.

Параллельный порт используется для одновременной передачи 8 битов информации. В компьютерах этот порт используется главным образом для подключения принтера, графопостроителей и других устройств. Параллельные порты обозначаются LPT1-LPT4.

Интерфейс USB (Universal Serial Bus) – универсальная последовательная шина призвана заменить устаревшие последовательный (COM-порт) и параллельный (LTP-порт) порты. Шина USB допускает подключение новых устройств без выключения компьютера. Шина сама определяет, что именно подключили к компьютеру, какой драйвер и ресурсы понадобятся устройству, после чего выделяет их без вмешательства пользователя. Шина USB позволяет подключить до 127 устройств.

IEEE 1394 (Institute of Electrical and Electronic Engineers 1394 – стандарт Института инженеров по электротехнике и электронику 1394) - последовательный интерфейс, предназначенный для подключения внутренних компонентов и внешних устройств. Цифровой последовательный интерфейс **IEEE 1394** характеризуется высокой надежностью и качеством передачи данных, его протокол поддерживает гарантированную передачу критичной по времени информации, обеспечивая прохождение видео- и аудиосигналов в реальном масштабе времени без заметных искажений. При помощи шины **IEEE 1394** можно подключить до 63 устройств и практически в любой конфигурации, чем она выгодно отличается от трудноконфигурируемых шин SCSI. Этот интерфейс используется для подключения жестких дисков, дисководов CD-ROM и DVD-ROM, а также высокоскоростных внешних устройств, таких как видеокамеры, видеомагнитофоны и т.д.

Задание 1.

Найти рисунок разъёмов системной платы. Указать название разъёмов и для каких устройств они применяются. Найти теоретические сведения по этим разъёмам.

Задание 2.

Определить внешние интерфейсы целевого компьютера. Подключить к целевому компьютеру принтер, монитор, сканер, мышь, клавиатуру, колонки.

Наружные разъёмы материнской платы: PS/2 (1 - мышь, 2 - клавиатура), сетевой RJ-45 (3), USB (4), D-subminiature (9-контактный разъём COM-порта) (5), LPT порт (6), VGA порт (7), MIDI) (8) и 3.5 мм аудио входы-выходы (разъём TRS) (9)

Пример:

1, 2. PS/2 — компьютерный порт (разъём), применяемый для подключения клавиатуры и мыши. Впервые появился в 1987 году на компьютерах IBM PS/2 и впоследствии получил признание других производителей и широкое распространение в персональных компьютерах и серверах. Скорость передачи данных — от 80 до 300 Кб/с и зависит от производительности подключенного устройства и программного драйвера.

Лабораторная работа №10

Цель работы:

- Освоить команды программирования микропроцессора I8080

Теоретические сведения

Написание программ для микропроцессорной системы - важнейший и часто наиболее трудоемкий этап разработки такой системы. Для разработки программного обеспечения существуют всевозможные программные средства. Чаще всего применяются языки программирования высокого уровня, такие как Паскаль и Си. Самые компактные и быстрые программы и подпрограммы создаются на языке Ассемблер, представляющем собой символьную запись цифровых кодов машинного языка, кодов команд микропроцессора.

Основная функция любого микропроцессора - выполнение команд. Система команд, выполняемых процессором, представляет собой нечто подобное таблице истинности логических элементов или таблице режимов работы более сложных логических микросхем. Она определяет логику работы микропроцессора и его реакцию на те или иные комбинации внешних событий. Знание системы команд и языка Ассемблер позволяет в несколько раз повысить эффективность некоторых наиболее важных частей программного обеспечения любой микропроцессорной системы - от микроконтроллера до персонального компьютера. Язык машинных кодов понятен процессору, но неудобен для работы программисту, поскольку команды трудно запоминаются и читаются. Гораздо удобнее символьное представление команд, когда символы команды напоминают о ее действиях, а двухбайтные операнды и адреса в командах пишутся в нормальном виде (начиная слева, от старших байт). Такая форма представления команд используется в языке Ассемблер. Символы, представляющие команды, называются мнемокодами. Характерной особенностью языка Ассемблер, определяющей его принадлежность в языкам низкого уровня, является то, что каждой команде Ассемблера соответствует одна и только одна машинная команда.

Рассмотрим основные типы команд, имеющиеся у большинства микропроцессоров, и особенности их применения.

Каждая команда, выбираемая (читаемая) из памяти микропроцессором, определяет алгоритм его поведения на ближайшие несколько тактов. Код команды говорит о том, какую операцию предстоит выполнить микропроцессору и с какими операндами (то есть кодами данных), где взять исходную информацию для выполнения команды и куда поместить результат (если необходимо). Код команды может занимать от одного до нескольких байт, причем микропроцессор узнает о том, сколько байт команды ему надо читать из первого прочитанного им байта или слова. В микропроцессоре код команды расшифровывается и преобразуется в набор микроопераций, выполняемых отдельными узлами микропроцессора

Пример:
Адрес Число Мнемокод
0800 78 MOV A,B

MOV – сокращение от move (переместить). После пробела указывают-ся операнды: сначала – операнд-приемник (регистр А), а затем – операнд-источник (регистр В). Операнды разделяются друг от друга запятой.

Система команд микропроцессора КР580ИК80А – i8080

Таблица 1– Коды регистров и пар регистров, используемые в командах МП

Регистры				Пары регистров			
Код	Имя (r)	Код	Имя (r)	Код (RP)	Имя пары (rp)	Регистры пары	
						старший	младший
000	B	100	H	00	B	B	C
001	C	101	L	01	D	D	E
010	D	110	M (память)	10	H	H	L
011	E	111	A (аккумулятор)	11	PSW	A	PSW

Список команд. Команды МП КР580ИК80А приведены в табл. 3-6. Трехбайтовые поля адресации источника и приемника информации кодируются в машинных командах символами SSS и DDD соответственно.

В мнемонических изображениях двухадресных команд приемник указывается на первом месте, а источник — на втором. В описаниях команд для обозначения содержимого регистра или ячейки памяти используется запись вида: (rl), (r), (H), (M) и т. п.

Таблица 2 – Коды условий, используемые в командах условных переходов

Код (CCC)	Мнемо-ника (cc)	Условие	Код (CCC)	Мнемо-ника (cc)	Условие
000	NZ	Не нуль (Z=0)	001	Z	Нуль (Z = l)
010	NC	Нет переноса (C = 0)	011	C	Перенос (C = 1)
100	PO	Нечетность (P = 0)	101	PE	Четность (P = 1)
110	P	Плюс (S = 0)	111	M	Минус (S = l)

Таблица 3 – Список команд передачи данных

Таблица 4 – Арифметические команды МП

Мнемоника	Код	Длина	Тактов	Флажки CY Z M P AC	Функция
ADD R	10000SSS	1	4	+++++	A←A+R
ADC R	10001SSS	1	4	+++++	A←A+R+CY
SUB R	10010SSS	1	4	+++++	A←A–R
SBB R	10011SSS	1	4	+++++	A←A–R–CY
ADI data8	86	00DDD110	7	2	+++++	A←A+(HL)
ACI data8	8E	36	1	7	2	+++++	10	A←A+(HL)+CY
SUI data8	96	3A	1	7	3	+++++	13	A←A–(HL)
SBI data8	9E	32	1	7	3	+++++	13	A←A–(HL)–CY
INR R	00DDD100	0A	1	1	+++++	7	A←A+data8
DCR R	00DDD101	1A	2	7	1	+++++	7	A←A+data8+CY
INR M	34	02	2	7	1	+++++	7	A←A–data8
DCR M	35	12	2	7	1	+++++	7	A←A–data8–CY
DAA	27	F9	1	4	1	+++++	5	A←2/10 коррекция A
DAD B	09	C5	1	10	1	+- - - -	12(11)	HL←HL+BC
DAD D	19	D5	1	10	1	+- - - -	12(11)	HL←HL+DE
DAD H	29	E5	1	10	1	+- - - -	12(11)	HL←HL+HL
DAD SP	39	F5	1	10	1	+- - - -	12(11)	HL←HL+SP
INX B	03	C1	1	6	1	- - - - -	10	BC←BC+1
INX D	13	D1	1	6	1	- - - - -	10	DE←DE+1
INX H	23	E1	1	6	1	- - - - -	10	HL←HL+1
INX SP	33	F1	1	6	1	- - - - -	10	SP←SP+1
DCX B	0B	EB	1	6	1	- - - - -	10(4)	BC←BC–1
DCX D	1B	E3	1	6	1	- - - - -	16(18)	DE←DE–1
DCX H	2B	1	6	- - - - -	HL←HL–1
DCX SP	3B	1	6	- - - - -	SP←SP–1

Таблица 5 – Логические команды МП

Мнемоника	Код	Длина	Тактов	Флажки CYZMPAC	Функция
ANA R	10100SSS	1	4	0+++0	A←A AND R
XRA R	10101SSS	1	4	0+++0	A←A XOR R
ORA R	10110SSS	1	4	0+++0	A←A OR R
CMP R	10111SSS	1	4	+++++	A – R
ANA M	A6	1	7	0+++0	A←A AND (HL)
XRA M	AE	1	7	0+++0	A←A XOR (HL)
ORA M	B6	1	7	0+++0	A←A OR (HL)
CMP M	BE	1	7	+++++	A – (HL)
ANI data8	E6	2	7	0+++0	A←A AND data8
XRI data8	EE	2	7	0+++0	A←A AND data8
ORI data8	F6	2	7	0+++0	A←A AND data8
					A←A AND data8

CPI data8	FE	2	7	+++++	$A_7 \leftarrow A_6 \leftarrow \ldots A_0 \leftarrow A_7$
RLC	07	1	4	+- - - -	$A_0 \leftarrow A_1 \leftarrow \ldots A_7 \leftarrow A_0$
RRC	0F	1	4	+- - - -	$A_7 \leftarrow A_6 \leftarrow \ldots A_0 \leftarrow CY \leftarrow A_7$
RAL	17	1	4	+- - - -	$A_0 \leftarrow A_1 \leftarrow \ldots A_7 \leftarrow CY \leftarrow A_0$
RAR	1F	1	4	+- - - -	A←NOT A
CMA	2E	1	4	- - - - -	CY←NOT CY
CMC	3F	1	4	+- - - -	CY←1
STC	37	1	4	1- - - -	

Таблица 6 – Команды ВВ и управления процессором

Мнемоника	Код	Длина	Тактов	Флажки CYZMPAC	Функция
IN port	DB	2	10	- - - - -	A←(port)
OUT port	D3	2	10	- - - - -	(port) ←A
RST n	11NNN111	1	11	- - - - -	-(SP) ←PC←8 x n, n=0-7
EI	FB	1	4	- - - - -	Разрешение прерываний
DI	F3	1	4	- - - - -	Запрет прерываний
RIM	20	1	4	- - - - -	Чтение маски прерывания
SIM	30	1	4	- - - - -	Запрет маски прерывания
HLT	76	1	5(7)	- - - - -	Останов
NOP	00	1	4	- - - - -	Нет операции

Описание симулятора МП КР580ВМ80

Данная программа предназначена для изучения функционирования МП КР580ВМ80 (импортный аналог), программирования на языке Ассемблер.

Интерфейс программы приведен на рис. 1. В нем можно выделить следующие блоки:

- блок состояния памяти;
- блок состояния регистров;
- блок битов состояний;
- рабочая область;
- указатель стека;
- порты ввода и вывода

- панель инструментов.

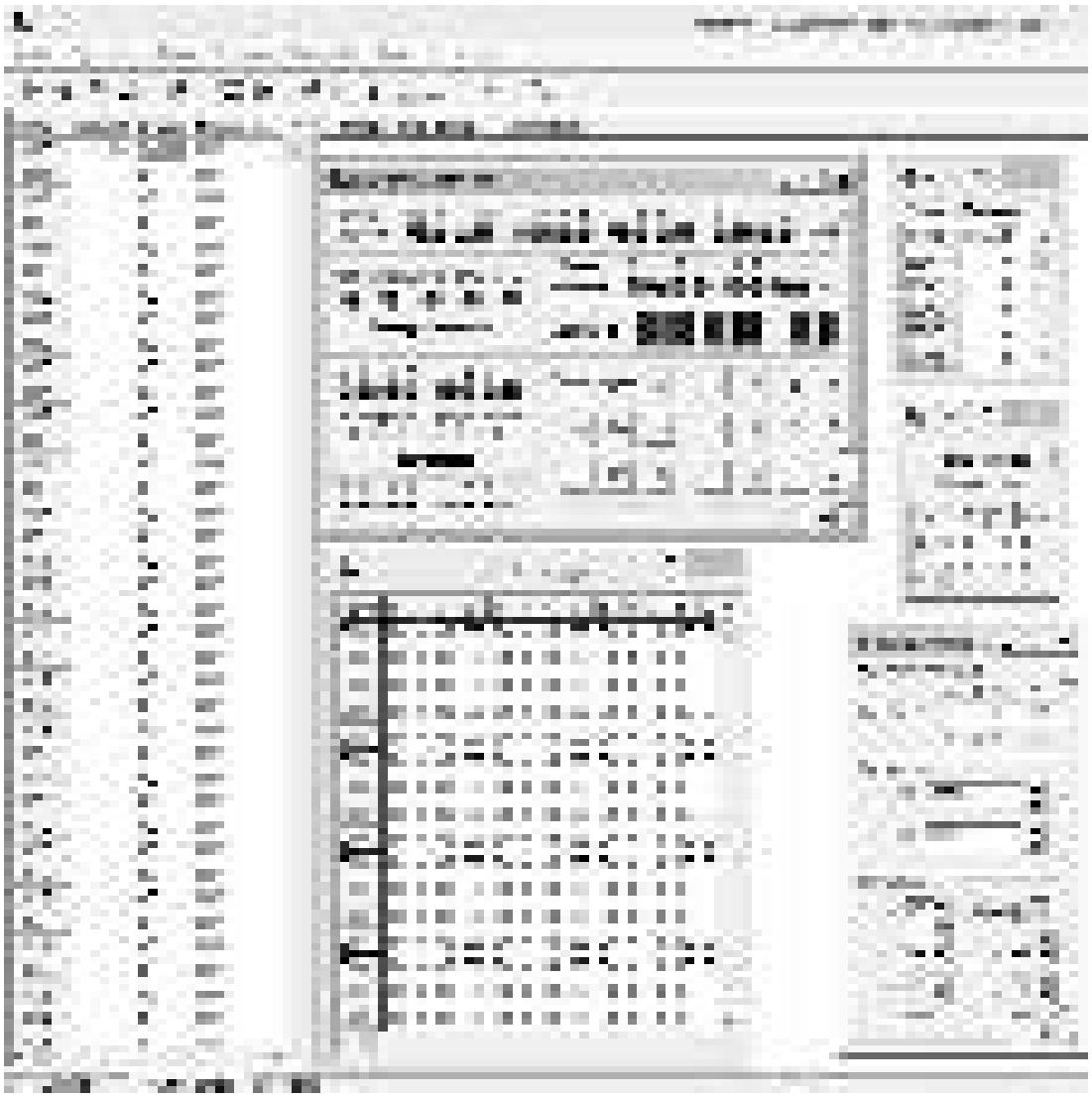


Рис. 1. Интерфейс симулятора микропроцессора КР580ВМ80 (I8080)

Программа может выполнять следующие функции:
- ввод информации и вызов директив с помощью клавиатуры;
- отображение вводимой информации в шестнадцатеричном коде на дисплее симулятора;
- запуск, выполнение и отладка программ пользователя.

Для ввода программы пользователю необходимо:

- в меню «Файл» выбрать «Новый» (рис. 2).

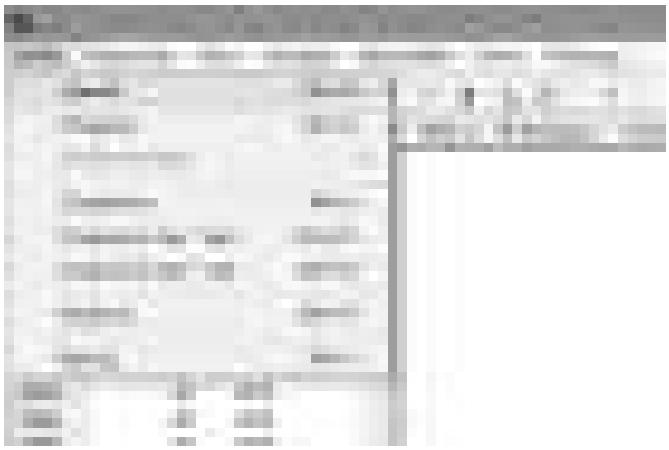


Рис. 2. Вид меню «Файл»

Теперь симулятор готов к записи программы пользователя. В блоке памяти по каждому адресу в исходном состоянии записываются пустые комманды NOP (рис. 3).

Рис. 3. Окно отображения состояния памяти

Блок памяти содержит три поля:
- поле адресов ячеек памяти;
- поле мнемокодов команд;
- шестнадцатеричный (машинный) код команды.

Новую команду можно ввести следующим образом:
- указать в области памяти на нужный адрес;
- перейти в окно ввода команд;
- записать команду, вводя его с клавиатуры компьютера (рис. 4).

Рис. 4. Рабочее поле

После занесения программы в память, необходимо выполнить ассемблирование. Ассемблирование – процесс трансляции программы с языка ассемблера в машинный код. Ассемблирование однозначный и обратимый процесс. В языке ассемблера каждой мнемонике соответствует одна машинная инструкция.

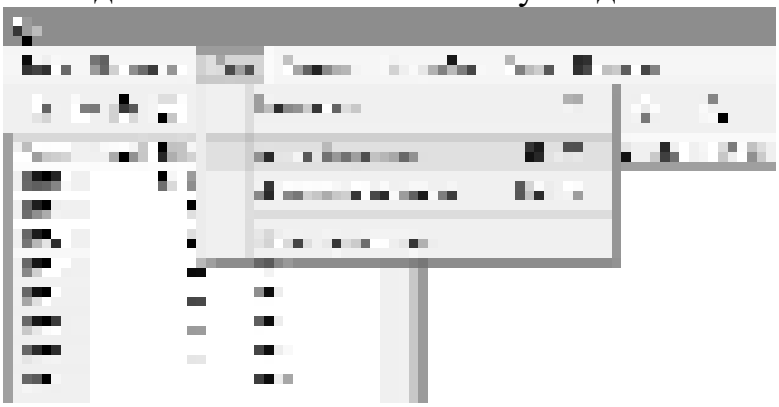


Рис. 5. Ассемблирование

После ассемблирования можно приступать к ее отладке и запуску. Сначала устанавливаем указатель памяти на начальный адрес. Пользователь может вызвать пошаговый режим выполнения программы, нажимая на клавиатуре кнопку F7 (одно нажатие – выполнение одной команды), либо воспользовавшись специальной кнопкой на панели инструментов  , либо выполнив следующее: Отладка→Шаг команды (рис. 6).

Рис. 6. Пошаговое выполнение программы

Так же запустить программу можно сразу, воспользовавшись специальной кнопкой на панели инструментов  .

Проследить выполнение и увидеть результаты работы программы можно, воспользовавшись окнами отображения регистров и флагов, состоянием портов ввода-вывода (см. рис. 7,8).

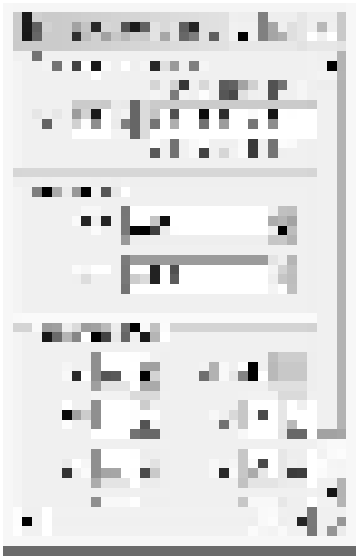


Рис. 7. Отображение состояния регистров

Рис. 8. Состояние портов ввода-вывода

В окне интерфейса также имеется специально выделенная область, в которую можно вносить различные записи о выполняемой программе на данном симуляторе: условие задания, комментарии.

Полученные результаты можно сохранить в удобном для пользователя виде, выбрав в меню «Файл» соответствующий формат файла: текстовый или файл памяти.

Выход из программы осуществляется нажатием комбинации клавиш Alt+X.

Контрольные вопросы:

1) Какие программные средства используются при написании программ для микропроцессорной системы?

2) Что представляет собой язык программирования Ассемблер?

3) Какая форма представления команд используется в языке Ассемблер?

4) Какими группами представлена система команд микропроцессора i8080 (привести примеры по каждой группе команд)?

5) Что такое командный цикл, машинный цикл?

ЛАБОРАТОРНАЯ РАБОТА №11

Цель работы:

- Написать программу для выполнения арифметических и логических действий над числами.

Теоретические сведения:

Возможности арифметических команд ограничиваются только операциями сложения и вычитания. Умножение, деление и более сложные арифметические операции можно организовать, составив соответствующие подпрограммы на основе имеющихся в распоряжении команд. Кроме того, возможности АЛУ позволяют одной команде оперировать лишь с однобайтными (и немного с двухбайтными) числами. Реализовать операции с многобайтными числами можно программным путем, понимая и используя признаки результатов однобайтных операций, которые устанавливаются командами этой группы в регистре признаков. К арифметическим командам относятся также команды сравнения, поскольку сравнение производится путем вычитания и установки признаков Z и CY, способных отразить равенство сравниваемых чисел или указать на большее из них.

Логические команды предоставляют возможности непосредственно выполнить следующие операции с однобайтными числами: И (конъюнкция), ИЛИ (дизъюнкция), исключающее ИЛИ (сложение по модулю два), НЕ (инверсия).

Схема выполнения практически всех арифметических и логических операций может быть представлена следующим образом:

(A) (A) <op> <2-й операнд>

где <op> – символ операции: +, —, & и т.д. Второй операнд может хранится в регистре процессора, в ячейке памяти или быть в составе самой команды. Первый операнд (или единственный операнд для операций с одним операндом) всегда хранится в аккумуляторе. Результат операции отправляется командой в аккумулятор, а признаки этого результата устанавливаются в регистре признаков.

Команды сложения

Команды сложения позволяют выполнять операции сложения однобайтных или двухбайтных операндов.

Команды сложения однобайтных операндов различаются по методам адресации второго операнда и по участию или неучастию в операции бита переноса CY. Общим для них является то, что первое слагаемое и результат хранятся в аккумуляторе. Команды сложения двухбайтных операндов работают с операндами только в регистровых парах: первое слагаемое и результат все такие команды хранят в регистровой HL (регистровая пара HL выступает в роли аккумулятора для однобайтных операций), второе слагаемое можно определять в любой из регистровых

трех пар процессора. Двоичное сложение выполняется в соответствии с правилами двоичной арифметики.

Таблица:

Правила двоичного сложения

0 + 0 = 0

0 + 1 = 1

1 + 1 = 0 + перенос 1 в следующий разряд

1 + 1 + 1 = 1 + перенос 1 в следующий разряд

Рассмотрим команды сложения однобайтных чисел.

Команды типа ADD r, ADC r обеспечивают выбор второго операнда регистровым методом. Выполняемое командой ADD r действие:

(A)← (A) + (r)

Эти команды предполагают, что исходные операнды будут предварительно записаны в аккумулятор и в регистр r.

Пример:

Требуется сложить числа 8Eh и C5h.

Адрес Число Мнемокод Комментарий

0800 3E 8E MVI A,8Eh ;поместить в аккумулятор 1-е слагаемое

0802 06 C5 MVI B,C5h ; поместить в регистр B 2-е слагаемое

0804 80 ADD B ; выполнить сложение: (A) ← (A) + (B)

Операция сложения должна дать такой результат:

```
   1 0 0 0 1 1 0 0     8 C
+  1 1 1 0 0 1 0 1  +  E 5
 1 0 1 1 1 0 0 0 1    1 7 1
```

Команды вычитания

Команды вычитания позволяют выполняют операции только с однобайтными операндами. По схеме выполнения и способам определения 2-го операнда, а также по участию или неучастию в операциях бита CY эти команды аналогичны командам однобайтного сложения. Команды выполняют вычитание по правилам двоичного вычитания из двоичной арифметики.

Правила двоичного вычитания

0 – 0 = 0

1 – 0 = 1

1 – 1 = 0

0 – 1 = 1 – заем 1 из следующего разряда

Команды вычитания SUB r и SBB r определяют второй операнд (вычитаемое) регистровым способом. Команды SUB r выполняется по схеме:

(A) ← (A) – (r)

Эти команды предполагают, что исходные операнды будут предварительно записаны в аккумулятор (уменьшаемое) и в регистр r (вычитаемое).

Пример:

Требуется вычесть из числа 8Eh число C5h.

Адрес Число Мнемокод Комментарий

0800 3E 8E MVI A,8E ; поместить в аккумулятор уменьшаемое

0802 06 C5 MVI B,C5 ; поместить в регистр B вычитаемое

0804 90 SUB B ; выполнить вычитание: (A) ← (A) – (B)

Операция вычитания должна дать такой результат:

```
  1 0 0 0 1 1 0 0     8 C
– 1 1 1 0 0 1 0 1   – E 5
1 1 0 1 0 0 1 1 1   1 A 7
```

Команды логических операций

Команды логических операций обеспечивают логические операции над соответствующими (имеющими одинаковый номер) битами однобайтных операндов. Влияния битов друг на друга типа переноса или заёма, имевшие место при арифметических операциях, в логических операциях отсутствуют. Поэтому, за счет влияния логических команд на все биты признаков, биты CY и AC после этих команд всегда будут сброшены в 0.

Основные логические команды обеспечивают логические операции –

И (конъюнкция), – ИЛИ (дизъюнкция), – исключающее ИЛИ(сложение по модулю два) в соответствии с правилами логики . Эти команды выполняются по общей схеме арифметически-логических команд: первый операнд и результат операции всегда хранятся в аккумуляторе, а второй операнд может быть выбран модификациями команд: либо в любом из регистров процессора; либо в ячейке памяти M, хранящей адрес в HL; либо непосредственно в составе самой команды.

Результаты логических операций

Исходные числа		&	V	⊕
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0

Команда CMA обеспечивает логическую операцию НЕ над операндом из аккумулятора. Выполняется инвертирование всех битов и результат остаётся в аккумуляторе. Если например, до выполнения команды в аккумуляторе было число 10110101, то после команды CMA там будет число 01001010.

Команды типа ANA r, ORA r, XRA r обеспечивают операции —И, ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется регистровым способом через выбор регистра r из регистров A, B, C, D, E, H, L.

Команды типа ANA M, ORA M, XRA M обеспечивают операции И,

ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется косвенно-регистровым способом через адрес подготовленный предварительно в регистровой паре HL.

Команды типа ANI d8, ORI d8, XRI d8 обеспечивают операции —И,

ИЛИ, исключающее ИЛИ соответственно, притом, что второй операнд адресуется прямым способом то есть является вторым байтом d8 в составе самой команды. Команды операций И используется когда требуется, например, оценить состояние какого-то одного из битов в составе байта.

Задание:

1) Выполнить действия над числами в соответствии с вариантом задания ручным счетом. При необходимости осуществить перевод чисел из десятичной в шестнадцатеричную систему счисления.
2) Написать программу для вычисления результата машинным счетом. Ввод чисел в регистры организовать программно. Предоставить листинг программы и соответствующий ему объектный код. Рассчитанное значение выражения сверить с данными ручного расчета.

Вариант	Действия	Начальные значения
1	C←(C+L+H) v E	C=1FH, L=10, H=25H, E=69
2	B←(B-C+D) v E	B=105, C=1BH, D=3EH, E=17
3	E←(B+L)+(D&E)	B=3AH, L=2BH, D=2DH, E=88
4	C←(B v H)+(2*L-C)	B=44, L=53, H=3CH, C=51H
5	H←(E&C)+(L⊕H)	E=4AH, L=4DH, H=62, C=21H
6	L←B⊕(C-H+L)	B=73H, L=22, H=34, C=89H
7	D←(B&D)+(C v D)*2	B=35H, D=45, C=4DH
8	B←(B+H)+(D⊕E)	B=93, H=2CH, D=3FH, E=110
9	E←(C-L)⊕(D v E)	C=5FH, L=3CH, D=15, E=240
10	C←(B&C)&(L-D)	B=7, L=101, D=21H, C=51H
11	H←(C&H+L)-(D⊕H)	D=1AH, L=6BH, H=30, C=28H
12	L←C&(D v L+B)	D=2AH, L=16, B=20, C=30H
13	D←(L+D)-(C v B)	B=74, L=37, D=3FH, C=4CH
14	B←(L⊕C+E) &B	B=2CH, L=3DH, C=28, E=59
15	E←(B⊕C) &(D v E)	B=2CH, C=35, D=3DH, E=70

Контрольные вопросы:

1) Какие арифметические операции может выполнять МП I8080?
2) Каковы правила сложения двоичных чисел?
3) Каковы правила вычитания двоичных чисел?
4) Назовите основные логические операции.

Пример:

Задание:

Выполнить действия:

D ← (L+D) - (C v B)

Начальные значения:

B = 74

L = 37

D = 3F*H*

C = 4C*H*

Расчет:

Ручной счет:

$L+D = 37_{10} + 3F_{16} = 25_{16} + 3F_{16} = 100_{10}$

$25_{16} = 100101_2 = 37_{10}$

$3F_{16} = 111111_2 = 63_{10}$
$37_{10} + 63_{10} = 100_{10} = 1100100_2 = 64_{16}$

$CvB = 4C_{16}$ v $74 = 4C_{16}$ v $4A_{16} = 78_{10}$
$4C_{16} = 1001100_2 = 76_{10}$
$4A_{16} = 1001010_2 = 74_{10}$
(1001100_2) v $(1001010_2) = 1001110_2 = 78_{10} = 4E_{16}$

	1	0	0	1	1	0	0
x	1	0	0	1	0	1	0
	1	0	0	1	1	1	0

$(L+D) - (CvB) = 64_{16} - 4E_{16} = 16_{16}$
$64_{16} = 1100100_2 = 100_{10}$
$4E_{16} = 1001110_2 = 78_{10}$
$100_{10} - 78_{10} = 22_{10} = 10110_2 = 16_{16}$

Машинный счет:
MVI B, 4A
MVI L, 25
MVI D, 3F
MVI C, 4C
MOV A, B
ORA C
MOV C, A
MOV A, L
ADD D
SUB C
MOV D, A

Результаты машинного и ручного счета совпадают.

Адрес	Машинный код	Мнемоника
0800	06	MVI B,4A
0801	4A	2-ой байт команды
0802	2E	MVI L,25
0803	25	2-ой байт команды
0804	16	MVI D,3F
0805	3F	2-ой байт команды
0806	0E	MVI E,4C
0807	4C	2-ой байт команды
0808	78	MOV A,B
0809	B1	ORA C
080A	4F	MOV C,A
080B	7D	MOV A,L
080C	82	ADD D
080D	91	SUB C
080E	47	MOV B,A

Лабораторная работа №12

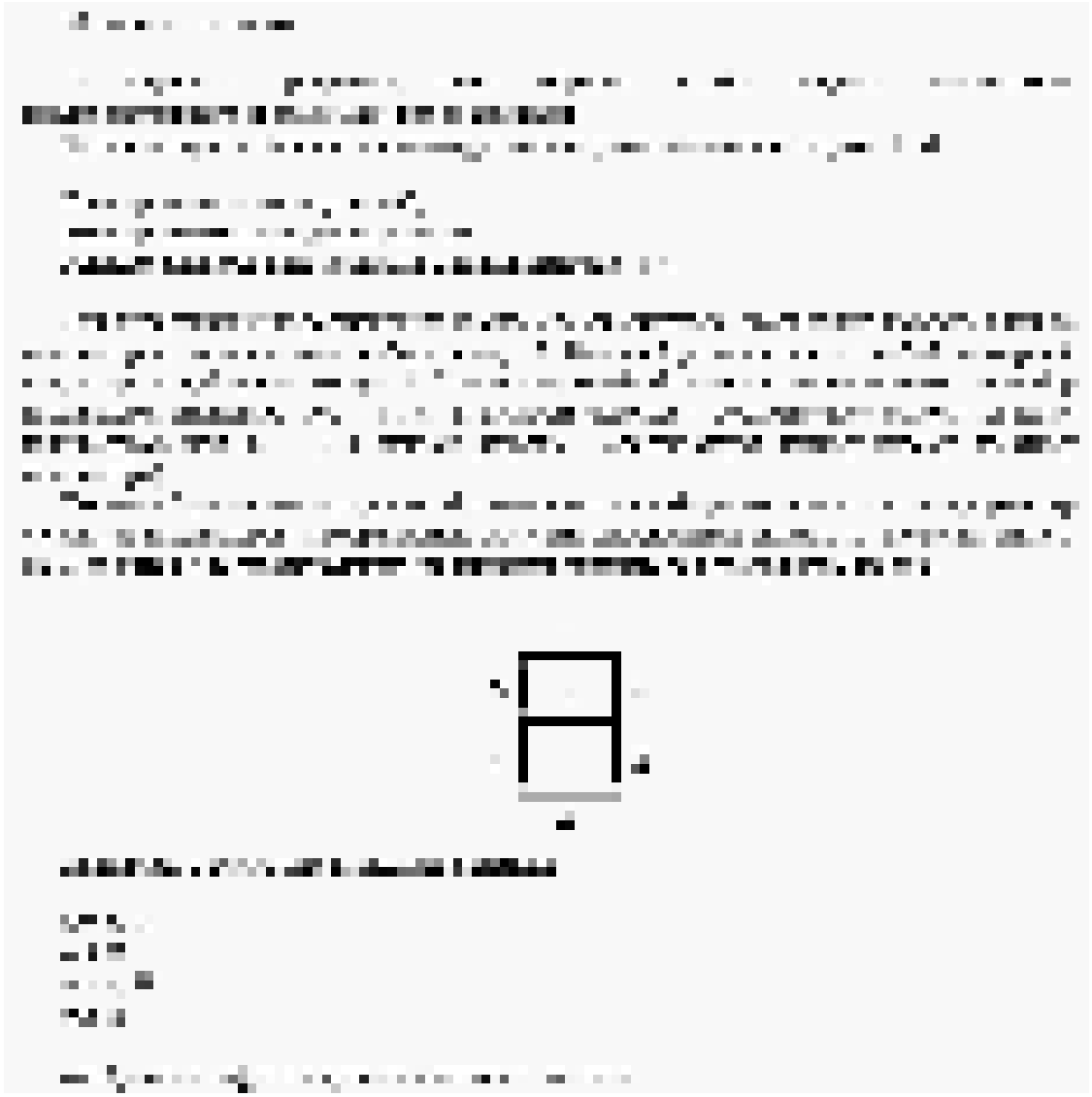
Программирование портов вывода

Цель работы:

1. Освоить команды программирования ввода-вывода и управления процессором.
2. Написать программу для выполнения вывода данных на индикаторы.
3. Запустив программу открыть средства стенда и зафиксировать результат.

Задание:

Вариант	Задание
1	Зажечь цифру «1» на 1, 3, 5 индикаторах.
2	Зажечь цифру «2» на 2, 3, 4 индикаторах.
3	Зажечь цифру «3» на 3, 6 индикаторах.
4	Зажечь цифру «4» на 1, 4, 6 индикаторах.
5	Зажечь цифру «5» на 1, 2, 5 индикаторах.
6	Зажечь цифру «6» на 1, 3, 6 индикаторах.
7	Зажечь цифру «7» на 1, 2, 3 индикаторах.
8	Зажечь цифру «8» на 2, 5 индикаторах.
9	Зажечь цифру «9» на 1, 5 индикаторах.
10	Зажечь цифру «0» на 3, 4 индикаторах.
11	Зажечь букву «A» на 1, 4 индикаторах.
12	Зажечь букву «b» на 3, 4, 6 индикаторах.
13	Зажечь букву «C» на 2, 4, 6 индикаторах.
14	Зажечь букву «d» на 2, 5, 6 индикаторах.
15	Зажечь букву «h» на 1, 2 индикаторах.



Пример:
Зажечь цифру «1» на 1 индикаторе.
Программа:

mvi a, 01
out 07
mvi a, 06
out 06