СЕВЕРО-КАЗАХСТАНСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. КОЗЫБАЕВА

А. Л. ГУЛЯЕВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «АРХИТЕКТУРА КОМПЬЮТЕРА»

Петропавловск
2013

УДК 004
ББК 32.973
Г 94

Издается по решению учебно-методического совета Северо-Казахстанского государственного университета им. М. Козыбаева (протокол №4 от 26.12.2012 г.)

Рецензент:

старший преподаватель кафедры «Информационные системы», магистр информационных систем *Е. Л. Кадочникова*

Гуляева А. Л.

Г 94 Методические указания для выполнения лабораторных работ по дисциплине «Архитектура компьютера». - Петропавловск: СКГУ им. М. Козыбаева, 2013. - 187 с.

Методические указания представляют собой лабораторный практикум по предметам «Архитектура компьютера» и «Архитектура компьютерных систем». Они содержат задания к работам, необходимый теоретический материал и примеры.

Для студентов специальностей «Информатика» и «Информационные системы». Указания могут быть использованы при подготовке к практическим и лабораторным занятиям, СРС, СРСП и в дистанционном обучении.

УДК 004
ББК 32.973

© Гуляева А.Л., 2013
© СКГУ им.М.Козыбаева, 2013

Система менеджмента качества СКГУ им. М. Козыбаева сертифицирована на соответствие требованиям ISO 9001:2008

Содержание

Введение

Данные методические указания предназначены для студентов специальностей «Информатика» и «Информационные системы».

Методические указания выполнены в соответствии с государственным стандартом РК и учебным планом специальности «Информатика» и «Информационные системы» и рассчитано на учебный курс, в котором лабораторные занятия проводятся 2-4 раза в неделю. На каждую лабораторную работу отводится 1-4 занятия.

В методических указаниях содержится теоретический и практический материал, необходимый студентам для выполнения лабораторных работ по данному предмету. Методические указания разделены на несколько частей, каждая из которых соответствует конкретной лабораторной работе.

Для выполнения лабораторных работ необходимо программное обеспечение: MS Word, MS Excel, Paint. Debug.

Для выполнения лабораторных работ студент должен:

- иметь навыки работы с компьютером;
- иметь навыки работы с программной и технической документацией ПК.
- иметь представление о принципах построения компьютера;
- знать основные составляющие компьютера.

Для наилучшего освоения дисциплины студенту необходимо изучить дополнительную литературу.

Правила и порядок выполнения лабораторных работ: **лабораторные работы выполняются в течение семестра в соответствии с методическими рекомендациями и представляются преподавателю для защиты в электронном виде в соответствии с графиком работ.**

Лабораторная работа № 1-4 «Архитектура компьютера и система команд»

Цель работы: ознакомиться со структурой персонального компьютера.

Задание:

1. Ознакомиться с представленным в тексте теоретическим материалом.
2. Составить конспект, в который включить выборочно новую для вас информацию.
3. Для защиты работы необходимо подготовиться к ответам на контрольные вопросы (в конце текста)
4. Изучить основные составляющие ПК.
5. Изучить структуру и составные части операционной системы MS-DOS.
6. Изучить строение клавиатуры.
7. Изучить составляющие рабочего ПК, описать его основные характеристики в отчёте.

Теоретический материал:

Электронная вычислительная машина (компьютер) – комплекс технических средств, предназначенных для автоматической обработки информации в процессе решения вычислительных и информационных задач.

Компьютеры могут быть классифицированы по ряду признаков, в частности: по принципу действия, назначению, способам организации вычислительного процесса, размерам и вычислительной мощности, функциональным возможностям, способности к параллельному выполнению программ и др.

Классификация ЭВМ по принципу действия. По принципу действия вычислительные машины делятся на три больших класса: ***аналоговые*** (АВМ), ***цифровые*** (ЦВМ) и ***гибридные.***

Критерием деления вычислительных машин на эти три класса является форма представления информации, с которой они работают.

Цифровые вычислительные машины – вычислительные машины дискретного действия, работают с информацией, представленной в дискретной, а точнее, в *цифровой* форме.

Аналоговые вычислительные машины – вычислительные машины непрерывного действия, работают с информацией, представленной в непрерывной (*аналоговой*) форме, т.е. в виде непрерывного ряда значений какой-либо физической величины (чаще всего электрического напряжения).

Гибридные вычислительные машины – вычислительные машины комбинированного действия, работают с информацией, представленной *и в цифровой, и в аналоговой форме*; они совмещают в себе достоинства АВМ и ЦВМ. Гибридные ЭВМ целесообразно использовать для решения задач управления сложными быстродействующими техническими комплексами.

*Классификация ЭВМ по назначению.*По назначению ЭВМ можно разделить на три группы: ***универсальные*** (общего назначения), ***проблемно-ориентированные*** и ***специализированные.***

Универсальные ЭВМ предназначены для решения самых различных инженерно-технических задач: экономических, математических, информационных и других задач, отличающихся сложностью алгоритмов и большим объемом обрабатываемых данных. Они широко используются в вычислительных центрах коллективного пользования и в других мощных вычислительных комплексах.

Характерными чертами универсальных ЭВМ являются:

– высокая производительность;

– разнообразие форм обрабатываемых данных: двоичных, десятичных, символьных, при большом диапазоне их изменения и высокой точности их представления;

– обширная номенклатура выполняемых операций, как арифметических, логических, так и специальных;

– большая емкость оперативной памяти; развитая организация системы ввода-вывода информации, обеспечивающая подключение разнообразных видов внешних устройств.

Проблемно-ориентированные ЭВМ служат для решения более узкого круга задач, связанных, как правило, с

– правлением технологическими объектами;

– регистрацией, накоплением и обработкой относительно небольших объемов данных;

– выполнением расчетов по относительно несложным алгоритмам.

Они обладают ограниченными по сравнению с универсальными ЭВМ аппаратными и программными ресурсами.

К проблемно-ориентированным ЭВМ можно отнести, в частности, всевозможные управляющие вычислительные комплексы.

Специализированные ЭВМ используются для решения узкого круга задач или реализации строго определенной группы функций. Такая узкая ориентация ЭВМ позволяет четко специализировать их структуру, существенно снизить их сложность и стоимость при сохранении высокой производительности и надежности работы.

Архитектура компьютера

Обычно персональные компьютеры внешне состоят из 4 основных частей (блоков):

– системного блока - основной блок компьютерной системы;

– клавиатуры, позволяющей вводить символы в компьютер;

– монитора (или дисплея) – для отображения текстовой и графической информации;

– мышь - устройство «графического» управления.

Кроме того, в состав компьютера могут включаться различные периферийные устройства, предназначенные для ввода-вывода информации. Такими устройствами являются:

– принтер – для вывода на печать текстовой и графической информации;

– мышь – устройство, облегчающее ввод информации в компьютер;

– сканер – устройство ввода графической информации и другие устройства.

Их подключение выполняется с помощью кабелей через специальные гнезда (разъемы), находящиеся обычно на задней стенке системного блока.

Некоторые периферийные устройства могут вставляться внутрь системного блока, например:

– модем – для обмена информацией с другими компьютерами через телефонную сеть;

– факс-модем – сочетает возможности модема и телефакса.

1. Системный блок представляет собой основной узел, внутри которого установлены наиболее важные компоненты. Устройства, находящиеся внутри системного блока, называют *внутренними,* а устройства, подключаемые к нему снаружи, называют *внешними.* Внешние дополнительные устройства, предназначенные для ввода, вывода и длительного хранения данных, также называют *периферийными.*

*Внутренние устройства системного блока***.**

Материнская плата – самая большая плата ПК, к которой подключается все то, что составляет сам компьютер. На ней размещаются:

– магистрали, связывающие процессор с оперативной памятью, - так называемые шины. К шинам материнской платы подключаются также все прочие внутренние устройства компьютера;

– микропроцессорный набор микросхем – так называемый чипсет, который управляет работой материнской платы;

– микропроцессор – основная микросхема ПК. Все вычисления выполняются в ней. Основная характеристика процессора – тактовая частота. Чем выше тактовая частота, тем выше производительность компьютера. Единственное устройство, о существовании которого знает процессор – оперативная память;

– оперативная память (ОП), предназначена для хранения и оперативного обмена информацией с прочими блоками машины. ОП содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) — микросхема предназначенная для длительного хранения данных, в том числе когда компьютер выключен и оперативное запоминающее устройство (ОЗУ) — набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;

– разъемы для подключения дополнительных устройств (слоты).

Жесткие диски (винчестеры) - для длительного хранения данных и программ. Выключение питания компьютера не приводит к очистке внешней памяти. Жесткий диск – это не один диск, а пакет (набор) дисков с магнитным покрытием, вращающихся на общей оси. Основным параметром является емкость, измеряемая в гигабайтах.

Видеоадаптер (видеокарта) – внутренне устройство, устанавливается в один из разъемов материнской платы, и служит для служит для обработки информации, поступающей от процессора или из ОЗУ на монитор, а также для выработки управляющих сигналов.

Звуковая карта - явилась одним из наиболее поздних усовершенствований персонального компьютера. Она подключается к одному из слотов материнской платы в виде дочерней карты и выполняет вычислительные операции, связанные с обработкой звука, речи, музыки. Звук воспроизводится через внешние звуковые колонки, подключаемые к выходу звуковой карты.

Специальный разъем позволяет отправить звуковой сигнал на внешний усилитель. Имеется также разъем для подключения микрофона, что позволяет записывать речь или музыку и сохранять их на жестком диске для последующей обработки и использования.

Сетевая карта (или карта связи по локальной сети) служит для связи компьютеров в пределах одного предприятия, отдела или помещения находящихся на расстоянии не более 150

метров друг от друга. При наличии специальных дополнительных устройств можно организовать связь компьютеров и на большие расстояния. Основным параметром сетевой карты является скорость передачи информации и измеряется она в мегабайтах в секунду. Типовая норма от 10 до 100 мегабайт в секунду.

2. Монитор – устройство визуального представления данных. Это не единственно возможное, но главное устройство вывода. Его основным потребительским параметром является размер экрана.

Размер монитора измеряется между противоположными углами трубки кинескопа по диагонали. Единица измерения – дюймы. Стандартные размеры: 14»; 15»; 17»; 19»; 20»; 21». В настоящее время наиболее универсальными являются мониторы размером 15 и 17 дюймов, а для операций с графикой желательны мониторы размером 19-21 дюйм.

3. Клавиатура - устройство ввода символьных данных. Клавиатура состоит из клавиш, которые подразделяются на следующие группы: алфавитные; цифровые; функциональные; управления курсором; индикаторы режимов; специальные клавиши.

Алфавитно-цифровые клавиши

При нажатии на эти клавиши в компьютер вводится алфавитно-цифровой символ. Какой именно — зависит от того, установлен ли режим ввода латинских или русских букв, и нажата или нет клавиша Shift.

Клавиша Пробел. При нажатии на эту клавишу вводится пробел, независимо от того установлен ли режим ввода латинских или русских букв, нажата или нет клавиша Shift.

Режимы ввода русских и латинских букв. В режиме ввода латинских букв при нажатии на любую алфавитно-цифровую клавишу вводится латинская буква или символ, изображенный в левой части клавиши (обычно эти буквы и символы нарисованы на клавишах черным цветом). А в режиме ввода русских букв при этом вводится русская буква или символ, изображенный в правой части клавиши (обычно эти буквы и символы

нарисованы на клавишах красным цветом в правой части клавиши). Переключение этих режимов выполняется с помощью клавиши или комбинации клавиш, определенных используемым драйвером клавиатуры (программой, осуществляющей ввод с клавиатуры).

Для ввода прописных букв и других символов, располагающихся на верхнем регистре клавиатуры, имеется клавиша Shift. Например, чтобы ввести строчную букву «d», надо нажать клавишу, на которой изображено «D», а чтобы ввести прописную букву «D», надо нажать клавишу Shift и, не отпуская ее, нажать на клавишу «D». Аналогично, ввод символа «=« осуществляется без нажатия клавиши Shift, а ввод символа «+» — нажатием на ту же клавишу при нажатой клавише Shift.

Клавиша Caps Lock служит для фиксации режима прописных букв. В этом режиме при обычном нажатии на буквенные клавиши вводятся прописные буквы, а при нажатой клавише Shift — строчные (это противоположно тому, что делается в обычном режиме). Режим прописных букв (часто его называют «режим Caps Lock») удобен при вводе текста, состоящего из таких букв. Повторное нажатие клавиши (Caps Lock) отменяет режим прописных букв.

На клавиатурах, оснащенных световыми индикаторами режимов, в режиме «Caps Lock» зажигается одноименный индикатор.

Расположение латинских букв на клавиатуре IBM PC, как правило, такое же, как на английской пишущей машинке, а букв кириллицы — как на русской пишущей машинке. А расположение клавиш со специальными символами (\, # и др.) не является строго фиксированным

На верхней части клавиатуры располагается блок функциональных клавиш: F1-F12. Порядок использования этих клавиш определяется программой и операционной системой, с которыми Вы в данный момент работаете. Часто программы устанавливают те или иные значения и для комбинаций функциональных клавиш с клавишами Ctrl, Alt и Shift (см.

ниже). Во многих программах при нажатии клавиши F1 на экран выводится встроенный справочник по программе.

Клавиши ←, ↑, →, ↓, Home, End, PgUp, PgDn называют *клавишами управления курсором.* Как правило, нажатие на них приводит к перемещению курсора (указателя текущего места в обрабатываемом документе) в соответствующем направлении или к «перелистыванию» изображаемого на экране текста. Впрочем, в других ситуациях эти клавиши (а также их комбинации с клавишами Ctrl, Alt и Shift) могут иметь другие значения. На 101-клавишной клавиатуре данные клавиши располагаются удобным блоком (в этом блоке имеются также клавиши Insert и Delete, см. ниже).

Названия клавиш PgUp и PgDn означают «страница вверх» Page Up и «страница вниз» Page Down Обычно нажатие на эти клавиши приводит к перелистыванию содержимого экрана (скажем, при редактировании документа) на страницу вверх и вниз.

Обычно нажатие на клавиши Home и End перемещает курсор в начало и в конец строки.

Цифровая клавиатура. Блок клавиш в правой части стандартной 101-клавишной клавиатуры - используется для двух целей. В режиме блокировки цифр (режим «Num Lock») этот блок удобен для ввода числовой информации и знаков арифметических действий. В этом режиме при нажатии на белые клавиши из этого блока вводятся цифры от О до 9 и точка. А если режим блокировки цифр (режим «Nurn Lock») выключен, то эти клавиши дублируют клавиши управления курсором, а также клавиши Insert и Delete. Включение и выключение режима блокировки цифр осуществляется нажатием клавиши Num Lock.

Клавиша Del или Delete обычно используется для удаления символов — выделенного фрагмента текста, справа от курсора (в Windows) или под курсором (в DOS-программах, работающих в текстом режиме). Клавиша Ins или Insert (Insert — вставка) обычно используется либо для вставки символов, либо для переключения между двумя режимами ввода

символов: ввода с раздвижкой символов (вставка) и ввода с замещением ранее выбранных символов (замена).

Клавиша Enter обычно используется для окончания ввода того или иного объекта. Например, при вводе команд DOS ввод каждой команды (командной строки) должен оканчиваться нажатием клавиши Enter. При наборе текста в редакторе документов нажатие клавиши Enter, как правило, оканчивает ввод абзаца.

Кроме перечисленных клавиш, на клавиатуре имеется еще несколько специальных клавиш.

Клавиша Backspace обычно удаляет символ, находящийся слева от курсора.

Клавиша Esc, как правило, используется для отмены какого-либо действия, выхода из режима программы и т.д.

Клавиша Tab (табуляция) при редактировании текстов обычно используется (для перехода к следующей позиции табуляции. В других программах ее значение может быть иным: переключение между полями запроса и т.д.

Клавиша Print Screen используется для печати содержимого экрана, копирования его в файл или буфер обмена Windows и т.д.

Клавиша Pause/Break – приостановка/прерывание текущего процесса.

Некоторые действия при работе с компьютером осуществляются нажатием не одной клавиши, а их комбинации. Для ввода комбинации клавиш надо нажать на первую клавишу затем, не отпуская ее, нажать на другую клавишу, после чего отпустить обе клавиши.

Индикаторы режимов. В правом верхнем углу 101-клавишной клавиатуры располагаются индикаторы режима блокировки цифр Num Lock, режима прописных букв Caps Lock и режима блокировки прокрутки Scroll Lock. Эти индикаторы зажигаются при включении соответствующих режимов, а при выключении этих режимов — погасают. Включение и выключение указанных режимов осуществляется нажатием на одноименные клавиши (Num Lock, Caps Lock, Scroll Lock).

4. Мышь - устройство командного управления. При перемещении мыши по столу или иной поверхности на экране соответствующим образом перемещается указатель мыши (обычно - стрелка). Когда необходимо выполнить то или иное действие, или, например, выполнить пункт меню, на который установлен указатель мыши, пользователь нажимает ту или иную кнопку мыши.

Стандартная мышь имеет только две кнопки, хотя существуют нестандартные мыши с тремя и более кнопками и вращающимся регулятором. Функции нестандартных органов управления определяются тем программным обеспечением, которое поставляется вместе с устройством.

К числу регулируемых параметров мыши относятся: чувствительность (выражает величину перемещения указателя на экране при заданном линейном перемещении мыши), функции левой и правой кнопок, а также чувствительность к двойному нажатию (максимальный интервал времени, при котором два щелчка кнопкой мыши расцениваются как один двойной щелчок).

Приемы управления с помощью мыши:

– щелчок – быстрое нажатие и отпускание левой кнопки мыши;

– двойной щелчок – два щелчка, выполненные с малым интервалом времени между ними;

– щелчок правой кнопкой – то же, что и щелчок, но с использованием правой кнопки;

– перетаскивание (drag-and-drop) – выполняется путем перемещения мыши при нажатой левой кнопке);

– протягивание мыши (drag) – выполняется, как и перетаскивание, но при этом происходит не перемещение экранного объекта, а изменение его формы;

– специальное перетаскивание – выполняется, как и перетаскивание, но при нажатой правой кнопке мыши, а не левой;

– зависание – наведение указателя мыши на значок объекта или на элемент управления и задержка его на некоторое

время (при этом обычно на экране появляется всплывающая подсказка, кратко характеризующая свойства объекта).

5. Сканеры– устройство считывания графической и текстовой информации в компьютер. Ручные, листовые, «планшетники» и рулонные сканеры. Сканеры могут распознавать шрифты букв для этого используются специальные программы - системы векторного распознавания образов.

6. Плоттер или графопостроитель - устройство для вывода чертежей на бумагу.

7. Цифровые фото и видео-камеры.

8. Принтеры. Принтер предназначен для вывода информации на бумагу. Как, правило, применяются принтеры следующих типов: матричные, струйные, лазерные.

Матричные принтеры. Принцип печати: печатающая головка содержит вертикальный ряд тонких металлических стержней (иголок). Головка движется вдоль печатаемой строки, а стержни в нужный момент ударяют по бумаге через красящую ленту. Это и обеспечивает формирование изображения. В дешевых моделях принтеров используется головка с 9 иглами. Более качественная печать обеспечивается принтерами с 24 иглами. Скорость печати матричных принтеров - от 60 до 10 секунд на страницу.

Струйные принтеры. В этих принтерах изображение формируется микрокаплями специальных чернил, выдуваемых на бумагу. Этот способ печати обеспечивает более высокое качество печати по сравнению с матричными принтерами, он очень удобен для цветной печати. Скорость печати струйных принтеров приблизительно такая же, как и у матричных.

Лазерные принтеры обеспечивают в настоящее время наилучшее качество печати. В этих принтерах для печати используется принцип ксерографии: изображение переносится на бумагу со специального барабана, к которому электрически притягиваются частички краски. Отличие лазерного принтера от обычного ксерокопировального аппарата состоит в том, что печатающий барабан электризуется с помощью лазера по

командам из компьютера. Лазерные принтеры, хотя и достаточно дороги, являются наиболее удобными устройствами для получения качественных печатных документов. Разрешающая способность лазерных принтеров, как правило, 600 точек на дюйм. Скорость печати лазерных принтеров - от 15 до 3 с на страницу.

9. Флэш-диски. Устройство хранения данных на основе энергонезависимой флэш-памяти. Имеет минимальные размеры и допускает «горячее» подключение через разъём USB, после чего распознаётся как жёсткий диск. Объэм флэш-диска может составлять от 32 Мб до нескольких Гб.

10. Модем - устройство, предназначенное для обмена информацией между удалёнными компьютерами по каналам связи. В зависимости от типа канала модемы подразделяют на радио-модемы, кабельные и т.д. Наиболее распространены модемы для телефонных линий.

Программные средства персональных ЭВМ. Возможности компьютера как технической основы системы обработки данных связаны с используемым программным обеспечением (программами).

Программа – упорядоченная последовательность команд (инструкций) компьютера для решения задачи.

Программное обеспечение (ПО) – совокупность программ обработки данных и необходимых для их эксплуатации документов

Все программные средства можно условно разделить на две группы: общее ПО и специальное ПО.

Основные функции общего ПО:

– автоматическое управление вычислительным процессом, прохождением заданий через вычислительную систему;

– повышение эффективности вычислительной системы за счет реализации различных режимов ее работы, рационального распределения ресурсов системы, минимального вмешательства оператора или программиста в вычислительный процесс;

– обеспечение взаимодействия вычислительной системы и пользователя в формах, удобных для пользователя;

– обеспечение контроля и надежности функционирования ЭВМ с помощью наладочных, контролирующих и диагностических программ.

Общее ПО, выполняя указанные функции, решает задачу применения ЭВМ как некоторой универсальной системы обработки информации.

Специальное ПО представляет собой комплекс программ, которые добавляются к общему ПО и решают задачу применения ЭВМ как некоторой специализированной системы обработки информации.

Общее ПО состоит из операционных систем (ОС), набора функциональных программных средств (или пакетов прикладных программ) и комплексов программ технического обслуживания (КПТО).

Операционные системы — это комплекс взаимосвязанных системных программ, назначение которого – организовать взаимодействие пользователя с компьютером и выполнение всех других программ. Операционная система играет роль связующего звена между аппаратурой компьютера, с одной стороны, и выполняемыми программами, а также пользователем, с другой стороны.

Операционная система обычно хранится во внешней памяти компьютера – на диске. При включении компьютера она считывается с дисковой памяти и размещается в ОЗУ. Этот процесс называется загрузкой операционной системы.

В функции операционной системы входит:

– осуществление диалога с пользователем;

– ввод-вывод и управление данными;

– планирование и организация процесса обработки программ;

– распределение ресурсов;

– запуск программ на выполнение;

– всевозможные вспомогательные операции обслуживания;

– передача информации между различными внутренними устройствами;

– программная поддержка работы периферийных устройств.

В различных моделях компьютеров используют операционные системы с разной архитектурой и возможностями. Для их работы требуются разные ресурсы. Они предоставляют разную степень сервиса для программирования и работы с готовыми программами.

Операционная система для персонального компьютера, ориентированного на профессиональное применение, должна содержать следующие основные компоненты:

– программы управления вводом/выводом;

– программы, управляющие файловой системой и планирующие задания для компьютера;

– процессор командного языка, который принимает, анализирует и выполняет команды, адресованные операционной системе.

Каждая операционная система имеет свой командный язык, который позволяет пользователю выполнять те или иные действия:

– обращаться к каталогу;

– выполнять разметку внешних носителей;

– запускать программы и другие действия.

Под функциональными программными средствами (часто их называют пакетами прикладных программ – ППП) будем понимать комплексы программ, которые предназначены для решения определенных задач или класса задач. Обычно такой комплекс (или «пакет») ориентируется на решение задач большого объема и высокой сложности и представляет собой обособленный элемент общего программного обеспечения. В настоящее время разработаны сотни тысяч таких программных комплексов для различного применения. Наиболее широко применяются текстовые редакторы, табличные процессоры (или электронные таблицы), системы управления базами данных

(СУБД), графические системы, системы программирования, средства телекоммуникации, интегрированные системы.

Текстовые редакторы предназначены для создания и изменения (редактирования) текстов программ и документов.

Табличные процессоры (или электронные таблицы) обеспечивают работу с большими таблицами чисел и других данных. Табличные процессоры позволяют вычислять значения элементов таблицы по заданным формулам, строить по данным в таблице различные графики, диаграммы и т. д. Поэтому их относят к средствам экспресс моделирования, обеспечивающим выполнение разовых расчетов с изменяющимися входными данными, причем в короткие сроки и с наименьшими усилиями.

Системы управления базами данных (СУБД) позволяют управлять большими информационными массивами – базами данных. Наиболее простые системы этого вида позволяют обрабатывать один массив информации, например, персональную картотеку. Они обеспечивают ввод, поиск, сортировку записей, составление отчетов и т.д. Однако часто необходимо решать задачи, в которых участвуют много различных видов объектов и, соответственно, много информационных массивов, связанных друг с другом различными соотношениями. В таких случаях требуется создавать специализированные информационные системы, в которых нужная обработка данных выполняется наиболее естественным для пользователя способом – с удобным представлением входных данных, выходных форм, графиков и диаграмм, запросов на поиск. Для решения таких задач используются более сложные СУБД, позволяющие с помощью специальных средств (обычно – языков программирования) описывать данные и действия с ними.

Графические системы обеспечивают создание и вывод на экран и на печать графических изображений. Так, системы деловой и научной графики позволяют наглядно представлять на экране различные данные и зависимости, дают возможность выводить различные виды графиков и диаграмм.

Графика – одно из наиболее интенсивно развивающихся направлений функциональных программных средств.

Системы программирования (или, как их иногда называют, инструментальные программные средства) предоставляют пользователю средства для разработки программ.

Средства телекоммуникации обеспечивают программную поддержку работы в локальной вычислительной сети либо возможности подключения к другой (удаленной) ЭВМ с помощью канала связи.

Интегрированные системы объединяют функции отдельных систем (например, могут сочетать в себе возможности СУБД, табличного процессора, текстового редактора, системы деловой графики и др.). При этом используются единые правила доступа ко всем составным частям системы, обеспечиваются единые формы данных, увеличивается скорость работы. Часто пользователю предоставляется встроенный язык, позволяющий создавать на базе интегрированной системы различные надстройки, выполняющие нужные пользователю функции.

Контрольные вопросы:

1. Что такое вычислительная машина?
2. Цифровые, аналоговые и гибридные ЭВМ.
3. Перечислите внутренние устройства системного блока, для чего они предназначены?
4. Что такое монитор, клавиатура, мышь?
5. Какова структура ПК?
6. Для каких целей нужна оперативная память?
7. Какие дисководы могут входить в комплектацию ПК?
8. Для чего нужен монитор?
9. Основные характеристики мониторов?
10. Какие типы принтеров известны?
11. Клавиатура: основные группы клавиш.
12. Для чего предназначены периферийные устройства компьютера?

13. На какие виды по значению подразделяются ПУ компьютера?

14. Устройство управляющее работой ПК.

15. Устройства ввода/вывода информации/

16. Способ формирования изображения на экране монитора.

17. Способы формирования изображения при использовании принтеров различных типов.

18. Принцип работы сканера.

19. Дайте определение программы, программного обеспечения (ПО)?

20. Что такое рабочий стол?

21. Как упорядочить значки на рабочем столе?

22. Как переместить окно с помощью мыши? с помощью клавиатуры?

23. Как изменить размеры окна с помощью мыши? с помощью клавиатуры?

24. Как закрыть, свернуть, восстановить окно с помощью мыши? с помощью клавиатуры?

25. Как найти необходимый файл? (по имени и расширению; по имени без расширения; по расширению; по некоторым символам имени)

26. Как создать ярлык? Как удалить ярлык?

27. Как восстановить удаленные объекты?

28. Что такое контекстное меню?

29. Как выбрать необходимую пиктограмму рабочего стола с помощью мыши? с помощью клавиатуры?

30. Как просмотреть свойства нужного объекта с помощью мыши? с помощью клавиатуры?

31. Что такое панель задач?

32. Как изменить размеры панели задач?

33. Как переместить панель задач?

34. Как скрыть панель задач?

35. Как отобразить/скрыть ярлык часов?

36. Как настроить часы?

37. Как отобразить на панели задач ярлык языка?

38. Как добавить (удалить) к указанному списку языков необходимый?

39. Как изменить клавиши перехода на другой язык?

40. Как изменить цвет заголовка всех окон сообщения?

41. Для чего используется заставка рабочего стола?

42. Как изменить заставку рабочего стола?

43. Как открыть главное меню с помощью мыши? с помощью клавиатуры?

44. Как добавить папку (файл) в главное меню?

45. Как удалить папку (файл) в главное меню?

46. Как добавить в главное меню ярлык для программы/папки (если это возможно)?

47. Как в «Блокноте» добавить дату и время?

48. Как в «Paint» создать копию созданного объекта? (например, одинаковые деревья)

49. Какие есть режимы калькулятора?

50. Как перенести/скопировать информацию из одного приложения в другое?

51. Как расположить каскадом на экране открытые окна?

52. Как развернуть нужное окно из нескольких открытых с помощью мыши? с помощью клавиатуры?

53. Как запустить проводник (3 способа)?

54. Какова структура проводника?

55. Как просмотреть содержимое папки?

56. Какие существуют режимы отображения содержимого папки?

57. Как найти нужный файл через «Проводник»?

58. Как просмотреть структуру некоторого каталога (папки)?

59. Как создать папку в «Проводнике»??

60. Как скопировать файлы/папки в «Проводнике»?

61. Как переместить файлы/папки в «Проводнике»?

62. Как удалить файлы/папки в «Проводнике»?

63. Как изменить способ отображения содержимого папок (в одном окне/в разных окнах)?

Лабораторная работа №5-6 «Системы счислений»

Цель: Научиться выполнять перевод заданных точных десятичных чисел в различные системы счисления.

Задание 1: Выполнить перевод заданных точных десятичных чисел в системы счисления: двоичную, восьмеричную, шестнадцатеричную.

Перевод осуществить двумя способами:

1. Применив общие алгоритмы при переводе дробных частей руководствоваться правилами:

– либо точный перевод

– либо выделить период бесконечной дроби

– либо перевести с «достаточной» точностью.

Выполнить «обратный» переход в десятичную систему счисления и вычислить погрешность преобразований числа.

2. Используя свойство оснований $\mathbf{P = Q^n}$

Варианты для задания №1.

№ вар.	число	№ вар.	число	№ вар.	число
1	123,45	11	395,76	21	151,66
2	273,96	12	269,94	22	167,71
3	642,08	13	664,84	23	517,27
4	758,09	14	436,78	24	790,03
5	693,06	15	368,38	25	673,79
6	932,54	16	956,24	26	278,89
7	170,58	17	703,24	27	686,88
8	763,97	18	146,77	28	177,89
9	461,82	19	572,26	29	572,26
10	521,21	20	447,44	30	447,44

Образец оформления задания: Осуществить перевод числа 2112,135 в системы счисления.

1. двоичную;
2. восьмеричную;
3. шестнадцатеричную.

Выполнить «обратный» переход в десятичную систему счисления и вычислить погрешность перевода. Использовать общие алгоритмы и свойство оснований $\mathbf{P = Q^n}$.

Решение: Используя общие алгоритмы при переводе числа из десятичной системы счисления в двоичную, восьмеричную, шестнадцатеричную удобнее применять алгоритмы раздельного перевода целой и дробной частей, для чего представим исходное число в виде их суммы:

$$2112{,}135 = 2112+0{,}135$$

1. *перевод в двоичную систему счисления*

а) целой части $2112_{10} \rightarrow X_2$

2112	2
2112	1056 | 2
0	1056 | 528 | 2
	0 528 | 2
	0 264 | 2
	264 | 132 | 2
	0 132 | 66 | 2
	0 66 | 33 | 2
	0 33 | 16 | 2
	1 16 | 8 | 2
	0 8 | 4 | 2
	0 4 | 2 | 2
	0 2 | **1**
	0

Итак, $2112_{10} = 100001000000_2$

b) дробной части $0{,}135_{10} \rightarrow 0{,}X_2$

$0{,}135\cdot 2\rightarrow$ **0**$,27\cdot 2\rightarrow$ **0**$,54\cdot 2\rightarrow$ **1**$,08\cdot 2\rightarrow$ **0**$,16\cdot 2\rightarrow$ **0**$,32\cdot 2\rightarrow$**0**$,64\cdot 2$
$\rightarrow$**1**$,28\cdot 2\rightarrow$**0**$,56\cdot 2\rightarrow$**1**$,12\cdot 2\rightarrow$**0**$,24\cdot 2\rightarrow$**0**$,48\cdot 2\rightarrow$**0**$,96\cdot 2\rightarrow$**1**$,92...$

Как видно, период дроби достаточно быстро (или вообще) выделить не удается. Определим достаточную точность изображения числа в двоичной системе счисления из условия:

$$2^{-s} > \frac{10^{-3}}{2} > 2^{-(s+1)}$$

откуда $s = 10$, т.е. в двоичном изображении дроби надо оставить десять дробных знаков, т.е. $0{,}135_{10} = 0{,}0010001010..._2 = 0{,}001000101_2$

Таким образом,

$$2112{,}135_{10}=100001000000{,}001000101_2$$

«Возврат» в десятичную систему счисления осуществим вычислением полинома:

$$1\cdot2^{11}+1\cdot2^{6}+1\cdot2^{-3}+1\cdot2^{-7}+1\cdot2^{-9} = 2048+64+1/8+1/128+1/512»$$

$$« 2112+0,125+0,0078+0,0020=2112,1348»2112,135_{10}$$

Погрешность преобразований в данном случае равна нулю:

$$\left|2112,135-2112,135\right| = 0,000$$

2. перевод в восьмеричную систему счисления

а) целой части $2112_{10} \rightarrow X_8$

2112	8		
2112	264	8	
0	264	33	8
	0	32	**4**
		1	

Итак, $2112_{10} = 4100_8$

b) дробной части $0,135_{10} \rightarrow 0,X_8$

$$0,135\cdot8\rightarrow \mathbf{1},08\cdot8\rightarrow \mathbf{0},64\cdot8\rightarrow \mathbf{5},12\cdot8\rightarrow \mathbf{0},96\cdot8\rightarrow \mathbf{7},68\cdot8\rightarrow \mathbf{5},44\cdot8\rightarrow \mathbf{3},52...$$

Определим достаточную точность изображения числа в восьмеричной системе счисления из условия:

$$8^{-s} > \frac{10^{-3}}{2} > 8^{-(s+1)}$$

откуда $s = 3$, т.е. в восьмеричном изображении дроби достаточно оставить три дробных знака, т.е. $0,135_{10} = 0,105_8$.

Таким образом,

$$2112,135_{10}=4100,105_8$$

«Возврат» в десятичную систему счисления осуществим вычислением полинома:

$$4100,105_8=4\cdot8^{3}+1\cdot8^{2}+1\cdot8^{-1}+5\cdot8^{-3} = 2112+1/8+5/512»$$

$$« 2112+0,125+0,0098=2112,1348»2112,135_{10}$$

Погрешность преобразований в данном случае равна:

$$\left|2112,135-2112,135\right| = 0,000$$

3. перевод в шестнадцатеричную систему счисления

а) целой части $2112_{10} \rightarrow X_{16}$.

2112	16		
2112	132	16	
0	128	**8**	
	4		

Итак, $2112_{10} = 840_{16}$

b) дробной части $0{,}135_{10} \rightarrow 0{,}X_{16}$

$0{,}135\cdot16 \rightarrow \mathbf{2}{,}\underline{16}\cdot16 \rightarrow \mathbf{2}{,}56\cdot16 \rightarrow \mathbf{8}{,}96\cdot16 \rightarrow \mathbf{15}{,}36\cdot16 \rightarrow \mathbf{5}{,}76\cdot16 \rightarrow \mathbf{12}{,}\underline{16}...$

Достаточно быстро выделен период (указан в скобках), т.е.

$$0{,}135_{10} = 0{,}2(28F5C)_{16}$$

Таким образом,

$$2112{,}135_{10}=840,\ 2(28F5C)_{16}$$

Для «обратного» преобразования определим достаточную точность изображения числа в шестнадцатеричной системе счисления из условия:

$$16^{-s} > \frac{10^{-3}}{2} > 16^{-(s+1)}$$

откуда $s = 2$, т.е. в шестнадцатеричном изображении дроби достаточно оставить два дробных знака: $2112{,}135_{10}=840{,}23_{16}$ (в шестнадцатеричной системе счисления округление цифр производится по обычной схеме, причем роль 5_{10} играет 8_{16}), и для «возврата» в десятичную систему счисления вычислим полином:

$840{,}23_{16}=8\cdot16^2+4\cdot16^1+2\cdot16^{-1}+3\cdot16^{-2} = 2112+2/16+3/256$»

« $2112+0{,}125+0{,}0117=2112{,}1367=2112{,}137_{10}$

Погрешность преобразований в данном случае равна:

$$|2112{,}135-2112{,}137| = 0{,}002.$$

Используя свойство оснований $\mathbf{P} = \mathbf{Q}^{\mathbf{n}}$, в нашем случае $8 = 2^3$ и $16 = 2^4$, можно выполнить перевод исходного числа $2112{,}135_{10}$ в двоичную, восьмеричную и шестнадцатеричную системы счисления, зная одно из указанных его изображений, например, $2112{,}135_{10}=4100{,}105_8$ (как правило, восьмеричное изображение проще всего получить «ручным способом» в смысле скорости выполнения

арифметических действий) и таблицы соответствий изображений необходимых символов:

восьмеричная запись	0	1	2	3	4	5	6	7
двоичная запись	000	001	010	011	100	101	110	111

Таким образом, имеем:

шестнадцатеричная запись	0	1	2	3	4	5	6	7
двоичная запись	0000	0001	0010	0011	0100	0101	0110	0111
шестнадцатеричная запись	8	9	A	B	C	D	E	F
двоичная запись	1000	0101	1010	1011	1100	1101	1110	1111

$4100{,}105_8=\underline{100}\underline{001}\underline{000}000{,}\underline{001}\underline{000}\underline{101}_2$ (здесь выделены триады)

далее: $\underline{1000}\underline{0100}\underline{0000}{,}\underline{0010}\underline{0010}\underline{1000}_2=840{,}228_{16}$

(здесь выделены тетрады и добавлены незначащие нули для формирования целой тетрады).

Итак,

$$2112{,}135_{10}=4100{,}105_8=100001000000{,}001000101_2=840{,}228_{16}$$

Задание 2: Решить уравнение в восьмеричной и шестнадцатеричной системах счисления (считаем, что исходные данные представлены в соответствующей системе счисления).

Варианты для задания № 2.

№ вар.	уравнение	№ вар.	уравнение
1	$32{,}74x+7{,}2=23{,}5$	16	$2{,}1x+31{,}22=10{,}36$
2	$32{,}74x-7{,}2=23{,}5$	17	$2{,}1x-31{,}22=10{,}36$
3	$7{,}5x+32{,}74=23{,}5$	18	$2{,}1x+31{,}22=-10{,}36$
4	$7{,}5x-32{,}74=23{,}5$	19	$72{,}01x+14{,}6=27{,}06$
5	$7{,}5x+32{,}74=-23{,}5$	20	$72{,}01x-14{,}6=27{,}06$
6	$32{,}74x+7{,}2=-23{,}5$	21	$72{,}01x+14{,}6=-27{,}06$
7	$30{,}41x+4{,}2=20{,}7$	22	$14{,}6x+72{,}01=27{,}06$
8	$30{,}41x-4{,}2=20{,}7$	23	$14{,}6x-72{,}01=27{,}06$
9	$30{,}41x+4{,}2=-20{,}7$	24	$14{,}6x+72{,}01=-27{,}06$

10	$4{,}2x + 30{,}41 = 20{,}7$	25	$44{,}6x + 36{,}11 = 70{,}23$
11	$4{,}2x - 30{,}41 = 20{,}7$	26	$44{,}6x - 36{,}11 = 70{,}23$
12	$4{,}2x + 30{,}41 = -20{,}7$	27	$44{,}6x + 36{,}11 = -70{,}23$
13	$31{,}22x + 2{,}1 = 10{,}36$	28	$36{,}11x + 44{,}6 = 70{,}23$
14	$31{,}22x - 2{,}1 = 10{,}36$	29	$36{,}11x - 44{,}6 = 70{,}23$
15	$31{,}22x + 2{,}1 = -10{,}36$	30	$36{,}11x + 44{,}6 = -70{,}23$

Образец оформления. Решить уравнение 1,05·x + 1,51 = 2,07 в восьмеричной и шестнадцатеричной системах счисления, считая, что исходные данные представлены в соответствующей системе.

а) *восьмеричная система счисления*: $1{,}05_8 \cdot x + 1{,}51_8 = 2{,}07_8$ используем таблицы сложения и умножения

+	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

×	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	10	12	14	16
3	3	6	11	14	17	22	25
4	4	10	14	20	24	30	34
5	5	12	17	24	31	36	43
6	6	14	22	30	36	44	52
7	7	16	25	34	43	52	61

2,07	360	105	1,05·x = 2,07- 1,51
– 1,51	– 317	0,336...	1,05·x = 0,36
0,36	410		x = 0,36 / 1,05
	– 317		x = 36 / 105
	710		x = 5·6 /(3·27) =
	– 636		5·2·3 /(3·27)= 5·2/ 27
	52		

Итак, $\mathbf{x} = \frac{12}{27}$ или в виде десятичной дроби **x**»0,336»0,34 (достаточно оставить два десятичных знака согласно исходным данным).

b) шестнадцатеричная система счисления: $1{,}05_{16} \cdot x + 1{,}51_{16} = 2{,}07_{16}$

используем таблицы сложения и умножения

+	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1C	1D	1D	1E

×	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87

A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

```
  2,07        B60      | 105      1,05·x + 1,51 = 2,07
– 1,51      – B37      | 0,B28... 1,05·x  = 2,07- 1,51
  0,B6        290      |          1,05·x = 0,B6
            – 20A                  x = 0,B6 / 1,05
              860                  x = B6 / 105
            – 828                  x « 0,B28 « 0,B3 (достаточно
              38
```

оставить два десятичных знака согласно исходным данным, округление по обычной схеме).

Арифметические действия в двоичной системе счисления.

Для двоичной системы счисления.

+	0	1
0	0	1
1	1	10

*	0	1
0	0	0
1	0	1

Пример:

```
    *  1  0  1  1
          1  0  1
       ----------
       1  0  1  1
 1  0  1  1
-------------------
 1  1  0  1  1  1
```

Лабораторная работа №7-8 «Представление информации в ЭВМ (прямом коде, обратном коде, дополнительном коде, соответствующих модифицированных кодах)».

Цель: Научится представлять числа в прямом, обратном, дополнительном кодах, соответствующих модифицированных кодах.

Задание: Согласно своему варианту представить числа в прямом, обратном, дополнительном и в соответствующих модифицированных кодах.

Теоретический материал:

Поскольку, процесс кодирования любого вида информации в компьютере это преобразование информации в двоичный код, то естественно начнем рассмотрение вопроса с представления числовых данных в ЭВМ. Числа различают:

- целые (со знаком и без знака), например: 567; 234; -56; -11; 0;
- вещественные числа с фиксированной запятой, например: 125,56; -34,123; -0,67 это естественная форма записи чисел;
- вещественные числа с плавающей запятой, например, $3*10^{-8}$; $-1,6*10^{-19}$ - нормальная форма записи числа;

В общем случае любое число А в системе счисления с основанием Q можно записать в виде $A=(\pm M)*Q^{\pm P}$ где М-мантисса числа, а Р - порядок числа.

Для записи, хранения и выдачи по запросу информации, обрабатываемой ЭВМ, предназначено запоминающее устройство или память. На этапе конструирования ЭВМ для размещения информации предусмотрена разрядная сетка, то есть устанавливается длина разрядной сетки и указывается количество разрядов для изображения целой и дробной части числа.

Представление положительных чисел при переходе от беззнаковых чисел к числам со знаком сохраняется, а для изображения отрицательных чисел применяются специально построенные коды прямой, обратный и дополнительный, а также соответствующие модифицированные коды (выделяется два знаковых разряда). Благодаря этому приему, действия над всеми числами выполняются единообразно.

Положительные числа во всех кодах представляется одинаково.

Представление отрицательного числа в прямом коде: знаковый разряд равен 1 , далее двоичная форма числа.

Представление отрицательного числа в обратном коде: знаковый разряд равен 1 , далее инвертированная двоичная форма числа.(0 заменяется на 1, а 1 на 0)

Представление отрицательного числа в дополнительном коде: знаковый разряд равен 1, далее инвертированная двоичная форма числа плюс 1 к младшему разряду числа

Представление отрицательного числа в соответствующих модифицированных кодах отличается только лишь выделением двух знаковых разрядов равных 11

Пример: Представить числа в прямом, обратном, дополнительном и в соответствующих модифицированных кодах

Дес. число	8-разряд. двоичная форма	прямой код	обратный код	дополни-тельный код	модифици-рованный прямой код	мод. обр	модифиц. доп
5	00000101	0,00000101	0,00000101	0,00000101	00,00000101	00,00000101	00,00000101
-5	-00000101	1,00000101	1,11111010	1,11111011	11,00000101	11,11111010	11,11111011
-78	-01001110	1,01001110	1,10110001	1,10110010	11,01001110	11,10110001	11,10110010

Инвертир. +1 в мл.разряд

Примеры:

1.Обратный код в прямой код (инвертируем)

0,101→0,101

0,01→0,01

1,0011→1,1100

2. Дано в дополнительном коде перевести в прямой код 1,00111

– В обратный код 1,00111-1 из младшего разряда =1,00110;

– Инвертируем и получаем 1,11001.

Далее будем оперировать числами, по модулю меньшими единицы и представленными с фиксированной запятой. Знак числа будет размещаться непосредственно перед запятой. В модифицированных кодах оба знаковых разряда будут размещаться также перед запятой.

Представим отрицательное число $A = -0,a_1a_2 \ldots a_{n-1}a_{n(B)}$ *в* различных кодах (a_i — возможные цифры при данном

основании *B, i* = 1, 2, ..., *n).* Положительное число $A' = + 0,a_1a_2 \ldots a_{n-1}a_{n(B)}$ во всех кодах имеет одинаковое представление, совпадающее с самим числом.

Прямой код: [A]пр = (B - 1), a1a2... ап-1ап{B).

Например: A = - 0,2578(10);

[A]пр = 9,2578(10);

A= - 0,101101(2);

[A]пр = 1,101101(2).

Обратный код:

[A]обр=(B-1),b1b2…..bn-1bn; bi=B-1-ai (I=1,2,…,n).

Например:

A = - 0,2570(10);

[A]об = 9,7429(10)

A = -0,101101(2);

[A]обр = 1,010010(2).

Дополнительный код:

[A]доп =(B-1), b1b2…..bn-1bn+0,00…01.

n- разрядов

где bi = B- 1 - ai (i= 1, 2, ..., n)

Например:

A= - 0,2578(10);

[A]доп = 9,7422(10)

A = - 0,101100(2);

[A]доп = 1,010100(2).

Модифицированный обратный код:

[A]м. обр = (B - 1) (B - 1), b1b2….bn-1bn,

где bi = B-1-ai (i = l, 2, ..., n).

Например: A = - 0,2578(10);

[A] обр = 99,7421(10);

A = -0,101100(2);

[A]м. обр. = 11,010011(2).

Модифицированный дополнительный код:

[A]м. доп =(B-1) (B-1), b1b2…..bn-1bn+0,00…01.

n- разрядов

где bi = B-1- ai (i = 1, 2…..,n)

Например:

A = - 0,2570(10); [A]м. доп = 99,7430(10);
A = -0,101101(2); [A]м. доп = ll,010011(2).

Задание 1. Перевести следующие числа из прямого кода в дополнительный.

1. $0,39256_{(10)}$;
2. $-0,02910_{(10)}$;
3. $-0,99999_{(10)}$;
4. $-0,110111_{(2)}$;
5. $0,010111_{(2)}$;
6. $-0,00001_{(2)}$;
7. $-0,000001_{(10)}$;
8. $0,32323_{(10)}$;
9. $-0,001111_{(2)}$;
10. $-0,010100_{(2)}$;
11. —$0,10101_{(2)}$.

Задание 2. Перевести следующие числа из дополнительного кода в прямой:

1. $9,72079_{(10)}$;
2. $0,03232_{(10)}$;
3. $9,90000_{(10)}$;
4. $1,010011_{(2)}$;
5. $0,001101_{(2)}$;
6. $1,111111_{(2)}$,
7. $9,999989_{(10)}$;
8. $0,55544_{(10)}$,
9. $1,110101_{(2)}$;
10. $1,010100_{(2)}$;
11. $1,1010011_{(2)}$.

Задание 3. Перевести следующие числа из прямого кода в обратный.

1. $0,27355_{(10)}$;
2. $-0,01910_{(10)}$;
3. $-0,99998_{(10)}$;
4. $-0,101111_{(2)}$;
5. $0,0110101_{(2)}$;
6. $-0,00001_{(2)}$;
7. $-0,000000_{(10)}$,
8. $0,47321_{(10)}$;
9. $-0,000000_{(2)}$;
10. $-0,011011_{(2)}$;
11. $-0,111101_{(2)}$.

Задание 4. Перевести следующие числа из обратного кода в прямой.

1. $0,75325_{(10)}$;
2. $9,90588_{(10)}$;
3. $9,11111_{(10)}$;
4. $1,000010_{(2)}$;
5. $0,0111011_{(2)}$;
6. $1,111110_{(2)}$;
7. $9,999898_{(10)}$;
8. $0,56123_{(10)}$;
9. $1,011110_{(2)}$,
10. $1,0101110_{(2)}$;
11. $1,1110010_{(2)}$.

Задание 5. Перевести следующие числа из прямого кода в модифицированный дополнительный код.

1. $-0,20790_{(10)}$;
2. $-0,33674_{(10)}$;
3. $0,87899_{(10)}$;
4. $-0,0111010_{(2)}$;
5. $-0,0011001_{(2)}$;
6. $0,0101101_{(2)}$;
7. $-0,1110111_{(2)}$;
8. $-0,1101111_{(2)}$;
9. $-0,95001_{(10)}$;
10. $0,1010111_{(2)}$.

Задание 6. Перевести следующие числа из модифицированного дополнительного кода в прямой код.

1. $00{,}12379_{(10)}$;
2. $99{,}37895_{(10)}$;
3. $00{,}1011001_{(2)}$;
4. $11{,}1010000_{(2)}$;
5. $11{,}1000001_{(2)}$;
6. $00{,}1110001_{(2)}$;
7. $11{,}1001101_{(2)}$;
8. $99{,}10059_{(10)}$;
9. $00{,}75063_{(10)}$;
10. $11{,}1111110_{(2)}$.

Задание 7. Перевести следующие числа из прямого кода в модифицированный обратный код.

1. $-0{,}38975_{(10)}$;
2. $-0{,}21218_{(10)}$;
3. $0{,}97979_{(10)}$;
4. $-0{,}0101110_{(2)}$;
5. $-0{,}0010111_{(2)}$;
6. $0{,}0101011_{(2)}$;
7. $-0{,}1111001_{(2)}$;
8. $0{,}1011011_{(2)}$;
9. $-0{,}12301_{(10)}$;
10. $-0{,}0001010_{(2)}$.

Задание 8. Перевести следующие числа из модифицированного обратного кода в прямой код.

1. $99{,}00279_{(10)}$;
2. $99{,}99999_{(10)}$;
3. $00{,}12099_{(10)}$;
4. $00{,}1011101_{(2)}$;
5. $11{,}0000011_{(2)}$;
6. $11{,}1111101_{(2)}$;
7. $11{,}1010111_{(2)}$;
8. $99{,}47365_{(10)}$;
9. $11{,}1111111_{(2)}$.

Лабораторная работа №9-10 «Представление информации в ЭВМ. Арифметические операции в различных кодах»

Цель: научиться оперировать числами, по модулю меньшими единицы.

Задание: согласно своему варианту провести операции над числами.

Теоретический материал:

Операции сложения и вычитания с использованием прямого, дополнительного и модифицированного дополнительного кодов.

При сложении чисел, представленных в дополнительном или модифицированном дополнительном кодах, осуществляется обычное сложение по всем разрядам (включая и знаковые разряды). Единица переноса, вышедшая за пределы знакового или старшего знакового (модифицированный код) разряда, отбрасывается. Если при выполнении операции сложения в модифицированном коде в полученной сумме знаковые разряды не совпадают, то это означает, что произошло переполнение, т.

е. полученная сумма по модулю больше или равна единице (в дополнительном коде определить переполнение не всегда возможно). Если знаковые разряды суммы совпадают, то это означает, что полученная сумма по модулю меньше единицы, за исключением случая, когда полученная сумма равна минус единице.

$_{+}$11,101 011
11,011001
1 | 11,0 0010 0

$_{+}$0,110100
1,101001
1 | 0,011101

При выполнении операции вычитания она заменяется на сложение с дополнением, т.е. инвертируем вычитаемое и прибавляем затем 1, знак дополнения становится противоположным знаку вычитаемого. Например: Выполнить операцию алгебраического вычитания над числами, представленными в дополнительном коде.

$C = (+\ 0{,}100101_{(2)}) - (0{,}110101_{(2)}) = 0{,}110000_{(2)}$.

Находим дополнение:

1,110101 = 0,001010+1= 0,001011

$_{+}$ 0,100101
0,001011
0,110000

Задание №1. Выполнить операцию сложения в прямом коде:

1. $0{,}110111_{(2)} + 0{,}000101_{(2)}$	7. $0{,}010110_{(2)} + 0{,}010101_{(2)}$
2. $0{,}010111_{(2)} + 0{,}001001_{(2)}$	8. $0{,}010101_{(2)} + 0{,}001011_{(2)}$
3. $0{,}101011_{(2)} + 0{,}001011_{(2)}$	9. $0{,}011111_{(2)} + 0{,}011111_{(2)}$
4. $0{,}011111_{(2)} + 0{,}011111_{(2)}$	10. $0{,}010101_{(2)} + 0{,}100100_{(2)}$
5. $0{,}001111_{(2)} + 0{,}001111_{(2)}$	11. $0{,}100101_{(2)} + 0{,}000101_{(2)}$
6. $0{,}010111_{(2)} + 0{,}001001_{(2)}$	12. $0{,}101011_{(2)} + 0{,}001011_{(2)}$

Задание №2. Выполнить операцию алгебраического сложения над числами, представленными в дополнительном коде (ответ дать также в дополнительном коде):

1. $0{,}010011_{(2)}+ 0{,}100101_{(2)}$	6. $1{,}101011_{(2)}+ 1{,}011010_{(2)}$
2. $0{,}110100_{(2)} +1{,}101001_{(2)}$	7. $0{,}110101_{(2)}+1{,}001011_{(2)}$
3. $0{,}110101_{(2)}+1{,}001011_{(2)}$	8. $0{,}001010_{(2)}+1{,}100101_{(2)}$
4. $0{,}001010_{(2)}+ 1{,}100101l_{(2)}$	9. $1{,}110111_{(2)}+0{,}011011_{(2)}$
5. $1{,}110111_{(2)}+0{,}011011_{(2)}$	10. $1{,}101011_{(2)}+1{,}011010_{(2)}$

Задание №3. Выполнить операцию алгебраического сложения над числами, представленными в дополнительном коде (ответ дать в прямом коде):

1. $1{,}001011_{(2)}+ 0{,}101101_{(2)}$	5. $0{,}110110_{(2)}+1{,}100001_{(2)}$
2. $1{,}011101_{(2)} +1{,}100110_{(2)}$	6. $0{,}110110_{(2)} +1{,}100001_{(2)}$
3. $1{,}100101_{(2)} +0{,}011011_{(2)}$	7. $0{,}1111000_{(2)} +1{,}1111000_{(2)}$
4. $1{,}100101_{(2)} +0{,}011011_{(2)}$	8. $0{,}1101001_{(2)} +1{,}1111000_{(2)}$
	9. $0{,}1101111_{(2)} +0{,}1111000_{(2)}$

Задание №4. Выполнить операцию алгебраического сложения в дополнительном коде над числами, представленными в прямом коде (ответ дать в дополнительном коде):

1. $0{,}010111_{(2)}+ 1{,}101101_{(2)}$
2. $1{,}111011_{(2)}+ 1{,}000011_{(2)}$
3. $1{,}110011_{(2)} + 0{,}011111_{(2)}$
4. $0{,}010111_{(2)}+1{,}101101_{(2)}$
5. $1{,}1110111_{(2)}+1{,}000011_{(2)}$
6. $1{,}110011_{(2)}+0{,}011111_{(2)}$
7. $0{,}010111_{(2)} + 1{,}101001_{(2)}$
8. $0{,}011001_{(2)} + 1{,}001100_{(2)}$
10. $1{,}111001_{(2)}+ 1{,}000011_{(2)}$

Задание №5. Выполнить операцию алгебраического сложения в дополнительном коде над числами, представленными в прямом коде (ответ дать в прямом коде):

1. $1{,}110110_{(2)} +0{,}110101_{(2)}$	5. $0{,}101110_{(2)}+0{,}010001_{(2)}$
2. $0{,}001001_{(2)} + 0{,}011011_{(2)}$	6. $1{,}100111_{(2)} + 0{,}101001_{(2)}$
3. $0{,}111011_{(2)}+1{,}111001_{(2)}$	7. $0{,}001001_{(2)} +0{,}011011_{(2)}$
4. $1{,}100111_{(2)} +0{,}101001_{(2)}$	8. $0{,}111011_{(2)} +1{,}111001_{(2)}$
	9. $0{,}101110_{(2)} +0{,}010001_{(2)}$

Задание №6. Выполнить операцию алгебраического вычитания над числами, представленными в дополнительном коде (ответ дать в прямом коде):

1. $0{,}100101_{(2)}$ -$1{,}110101_{(2)}$	6. $0{,}101101_{(2)}$ - $0{,}110011_{(2)}$
2. $0{,}001101_{(2)}$ -$1{,}010111_{(2)}$	7. $0{,}011010_{(2)}$ -$0{,}101011_{(2)}$
3. $0{,}100101_{(2)}$ -$1{,}110101_{(2)}$	8. $0{,}001101_{(2)}$ -$1{,}010111_{(2)}$
4. $0{,}011010_{(2)}$ -$0{,}101011_{(2)}$	9. $0{,}101101_{(2)}$ -$0{,}110011_{(2)}$
5. $0{,}1101001_{(2)}$-$1{,}111100_{(2)}$	

Задание №7. Выполнить операцию алгебраического вычитания в дополнительном коде над числами, представленными в прямом коде (ответ дать в прямом коде):

1. $0{,}111001_{(2)}$ -$0{,}111101_{(2)}$	6. $0{,}101011_{(2)}$ -(-$0{,}010011_{(2)}$)
2. (-$0{,}110001_{(2)}$) -$0{,}001011_{(2)}$	7. $0{,}001111_{(2)}$ - $0{,}100001_{(2)}$
3. (-$0{,}111011_{(2)}$) - (-$0{,}101101_{(2)}$)	8. (-$0{,}100101_{(2)}$) -(-$0{,}010111_{(2)}$)
4. $0{,}110110_{(2)}$ - $0{,}101111_{(2)}$	9. (-$0{,}110001_{(2)}$) -$0{,}001011_{(2)}$
5. $0{,}111001_{(2)}$-$0{,}111101_{(2)}$	

Задание №8. Выполнить операцию алгебраического сложения над числами, представленными в модифицированном дополнительном коде (ответ дать в прямом коде). В данных примерах возможно переполнение.

1. $00{,}101011_{(2)}$ +$00{,}011101_{(2)}$	5. $11{,}111011_{(2)}$+ $00{,}110101_{(2)}$
2. $00{,}101101_{(2)}$ + $11{,}001101_{(2)}$	6. $00{,}101011_{(2)}$ + $11{,}010101_{(2)}$
3. $00{,}010111_{(2)}$ +$00{,}100101_{(2)}$	7. $11{,}100011_{(2)}$ + $11{,}001101_{(2)}$
4. $11{,}101100_{(2)}$+ $11{,}011011_{(2)}$	8. $11{,}111011_{(2)}$ +$00{,}110111_{(2)}$
	9. $00{,}010111_{(2)}$ +$00{,}100101_{(2)}$

Задание №9. Выполнить операцию алгебраического сложения в модифицированном дополнительном коде над числами, представленными в прямом коде (ответ дать в прямом коде). В данных примерах возможно переполнение.

1. (-$0{,}100101_{(2)}$)+ $0{,}011111_{(2)}$	7. $0{,}010111_{(2)}$+ $0{,}111011_{(2)}$
2. $0{,}101110_{(2)}$+ $0{,}011011_{(2)}$	8. (-$0{,}100100_{(2)}$)+(-$0{,}111001_{(2)}$)
3. (-$0{,}101001_{(2)}$)+ $0{,}110011_{(2)}$	10. (-$0{,}101001_{(2)}$)+ $0{,}110011_{(2)}$
4. (-$0{,}011101_{(2)}$)+ (-$0{,}100010_{(2)}$)	11. (-$0{,}010101_{(2)}$)+(-$0{,}100111_{(2)}$)
5. (-$0{,}110110_{(2)}$) + (-$0{,}001101_{(2)}$)	

Лабораторная работа № 11-14 «Логические основы ЭВМ»

Цель: Научиться оперировать с логическими операциями.

Теоретическая информация:

Алгебра логики — это раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними. Алгебра логики возникла в середине XIX века в трудах английского математика Джорджа Буля. Ее создание представляло собой попытку решать традиционные логические задачи алгебраическими методами. Что же такое логическое высказывание?

Логическое высказывание - это любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Джордж Буль Так, например, предложение «6 — четное число» следует считать высказыванием, так как оно истинное. Предложение «Рим — столица Франции» тоже высказывание, так как оно ложное. Разумеется, не всякое предложение является логическим высказыванием. Высказываниями не являются, например, предложения «ученик десятого класса» и «информатика — интересный предмет». Вопросительные и восклицательные предложения также не являются высказываниями, поскольку говорить об их истинности или ложности не имеет смысла. Предложения типа «в городе А более миллиона жителей», «у него голубые глаза» не являются высказываниями, так как для выяснения их истинности или ложности нужны дополнительные сведения: о каком конкретно городе или человеке идет речь. Такие предложения называются высказывательными формами.

Высказывательная форма — это повествовательное предложение, которое прямо или косвенно содержит хотя бы одну переменную и становится высказыванием, когда все переменные замещаются своими значениями. Алгебра логики рассматривает любое высказывание только с одной точки зрения — является ли оно истинным или ложным.

Употребляемые в обычной речи слова и словосочетания «не», «и», «или», «если..., то», «тогда и только тогда» и

другие позволяют из уже заданных высказываний строить новые высказывания. Такие слова и словосочетания называются логическими связками. Высказывания, образованные из других высказываний с помощью логических связок, называются составными. Высказывания, не являющиеся составными, называются элементарными. Чтобы обращаться к логическим высказываниям, им назначают имена. Пусть через A обозначено высказывание «Тимур поедет летом на море», а через B — высказывание «Тимур летом отправится в горы». Тогда составное высказывание «Тимур летом побывает и на море, и в горах» можно кратко записать как A и B. Здесь «и» — логическая связка, A, B — логические переменные, которые могут принимать только два значения — «истина» или «ложь», обозначаемые, соответственно, «1» и «0». Каждая логическая связка рассматривается как операция над логическими высказываниями и имеет свое название и обозначение:

НЕ (отрицание) $\overline{A}$ Высказывание истинно, когда A ложно, и ложно, когда A истинно.

A	$\overline{A}$
1	0
0	1

И (конъюнкция) (лат. conjunctio — соединение) или логическим умножением $A \wedge B$ Высказывание $A \wedge B$ истинно тогда и только тогда, когда оба высказывания A и B истинны.

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

ИЛИ (дизъюнкцией) (лат. disjunctio — разделение) или логическим сложением и обозначается знаком v (или плюсом).

Высказывание A v B ложно тогда и только тогда, когда оба высказывания A и B ложны.

A	B	A v B
1	1	1
1	0	1
0	1	1
0	0	0

ЕСЛИ-ТО (импликацией) (лат. implico — тесно связаны) $A \Rightarrow B$ Высказывание ложно тогда и только тогда, когда A истинно, а B ложно.

A	B	$A \Rightarrow B$
1	1	1
1	0	0
0	1	1
0	0	1

РАВНОСИЛЬНО »тогда и только тогда», «необходимо и достаточно», «... равносильно (эквиваленция) $A \Leftrightarrow B$. Высказывание истинно тогда и только тогда, когда значения A и B совпадают.

A	B	$A \Leftrightarrow B$
1	1	1
1	0	0
0	1	0
0	0	1

Любая цифровая вычислительная машина состоит из *логических схем - таких схем, которые могут находиться только в одном из двух возможных состояний - либо «логический ноль», либо «логическая единица».* За логический 0 и логическую 1 можно принять любое выражение, в том числе и словесное, которое можно характеризовать как «истина» и «ложь». В вычислительной технике логические 0 и 1 - это состояние электрических схем с определенными параметрами.

Так, для логических элементов и схем, выполненных по технологии транзисторно-транзисторной логики (ТТЛ-схемы), логический 0 - это напряжение в диапазоне 0 … + 0,4 В, а логическая 1 - это напряжение в диапазоне + 2,4 … + 5 В. Работа логических схем описывается посредством специального математического аппарата, который называется логической (булевой) алгеброй или алгеброй логики. Булева алгебра была разработана Джорджем Булем (1815 - 1864 гг.), она является основой всех методов упрощения булевых выражений.

Логические переменные и логические функции - это такие переменные и функции, которые могут принимать только два значения - либо логический 0, либо логическая 1.

Основные логические функции и элементы

Логический элемент - графическое представление элементарной **логической функции**.

Логическое умножение (конъюнкция) - функция И

Рассмотрим ключевую схему представленную на рис. 1.1,а. Примем за логический 0:

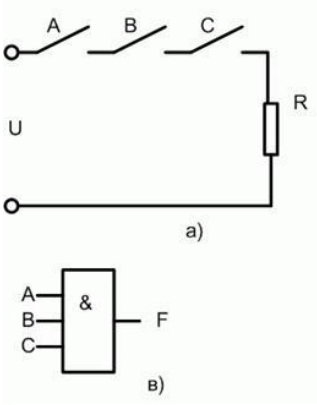
– на входе схемы разомкнутое состояние соответствующего ключа, например, $A = 0$;

– на выходе схемы ($F = 0$) - такое ее состояние, когда через сопротивление R ток не протекает.

Таблица истинности - это таблица, содержащая все возможные комбинации входных логических переменных и соответствующие им значения логической функции.

Таблица истинности для логической схемы, представленной на рис. 1.1,б, состоит из 8 строк, поскольку данная схема имеет три входа - A, B и C. Каждая из этих логических переменных может находиться либо в состоянии логического 0, либо логической 1. Соответственно количество сочетаний этих переменных равно $2^3 = 8$. Очевидно, что через сопротивление R ток протекает только тогда, когда замкнуты все три ключа - и A, и B, и C. Отсюда еще одно название логического умножения - логический элемент И. В логических

схемах этот элемент независимо от того, на какой элементной базе он реализован, обозначается так, как показано на рис. 1.1,в.



A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

б)

Рис. 1.1. Трёх-входовой логический элемент И

Правило **логического умножения** :если на вход логического элемента И подается хотя бы один логический 0, то на его выходе будет логический 0.

Уровень логического 0 является решающим для **логического умножения .**

В логических выражениях применяется несколько вариантов обозначения логического умножения. Так, для приведенного на рис. 1.1,в трёх-входового элемента И, логическое выражение можно представить в виде:

– либо $F = A \cdot B \cdot C$, но при этом из контекста должно быть ясно, что данное умножение именно логическое;

– либо $F = A\&B\&C$;

– либо $F = A \wedge B \wedge C$- с использованием знака конъюнкции;

– либо $F = ABC$, но при этом из контекста должно быть ясно, что между переменными A, B и C производится логическое умножение.

Логическое сложение (дизъюнкция) - функция ИЛИ

Рассмотрим ключевую схему, представленную на рис. 1.2,а. **Таблица истинности** для данной **логической схемы** (рис. 1.2,б) состоит из 4 строк, поскольку данная схема имеет

два входа - A и B. Количество сочетаний этих переменных равно $2^2 = 4$. Очевидно, что через сопротивление R ток протекает тогда, когда замкнуты или A, или B. Отсюда еще одно название логического сложения - логическое ИЛИ. **В логических схемах** соответствующий **логический элемент** независимо от того, на какой элементной базе он реализован, обозначается так, как показано на рис. 1.2,в.

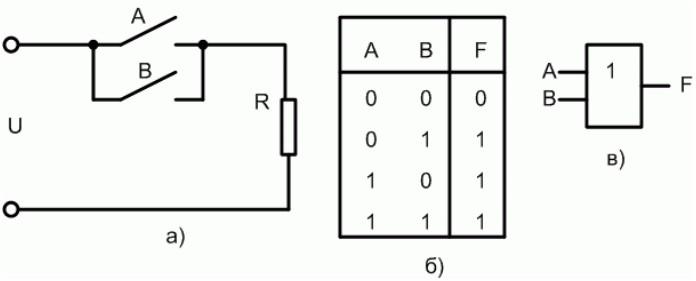


Рис. 1.2. Логический элемент ИЛИ на два входа

Правило **логического сложения**: если на вход логического элемента ИЛИ подается хотя бы одна логическая 1, то на его выходе будет логическая 1.

Для **логического сложения** решающим является уровень логической 1.

В логических выражениях применяется два варианта обозначения **логического сложения**. Так, для приведенного двух-входового элемента ИЛИ, логическое выражение можно представить в виде:

– либо $F = A + B$, но при этом из контекста должно быть ясно, что данное сложение именно логическое;

– либо $F = A \vee B$ - с использованием знака дизъюнкции.

Логическое отрицание (инверсия) - функция НЕ

Рассмотрим ключевую схему, представленную на рис. 1,а. Таблица истинности для данной схемы (рис. 1,б) самая простая и состоит всего из 2 строк, поскольку она (единственная из всех логических элементов) имеет только один вход - A. Количество вариантов для единственной **логической переменной** равно $2^1 = 2$. Очевидно, что через сопротивление R ток протекает (

$F = 1$) тогда, когда A не замкнут, т.е. $A = 0$. Еще одно название этой **логической функции** - отрицание, а соответствующий логический элемент называется **инвертором**. В **логических схемах** этот элемент независимо от того, на какой элементной базе он реализован, обозначается так, как показано на рис. 1.3,в. Поскольку он имеет только один вход, в его обозначении допустимым является и знак логического сложения, и знак логического умножения.

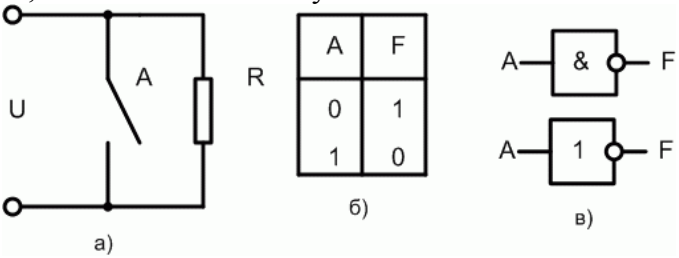


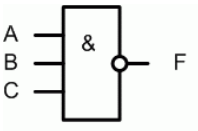
Рис. 1 Логический элемент НЕ

Правило инверсии: проходя через инвертор, сигнал меняет свое значение на противоположное.

В логических выражениях применяется единственный вариант обозначения инверсии: $F = \overline{A}$

К основным **логическим элементам** относятся еще два элемента, которые являются комбинацией элементов И, ИЛИ и НЕ: элемент И-НЕ и ИЛИ-НЕ.

Логическая функция и элемент И-НЕ. Данная функция производит **логическое умножение** значений входных сигналов, а затем инвертирует результат этого умножения. В **логических схемах** этот элемент независимо от того, на какой элементной базе он реализован, обозначается так, как показано на рис. 2,а. **Таблица истинности** приведена на рис. 2,б.



а)

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

б)

Рис. 2. Логический элемент И-НЕ на три входа

Если на вход логического элемента И-НЕ подается хотя бы один логический 0, то на его выходе будет логическая 1.

В логических выражениях применяются обозначения:

– либо $F = \overline{A \cdot B \cdot C}$, но при этом из контекста должно быть ясно, что данное умножение именно логическое;

– либо $F = \overline{ABC}$;

– либо $F = \overline{A\&B\&C}$;

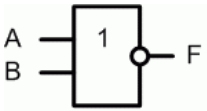
– либо $F = \overline{A \vee B \vee C}$.

Логическая функция и элемент ИЛИ-НЕ. В **логических схемах** этот элемент независимо от того, на какой элементной базе он реализован, обозначается так, как показано на рис. 3,а. **Таблица истинности** приведена на рис. 3,б.

Если на вход логического элемента ИЛИ-НЕ подается хотя бы одна логическая 1, то на его выходе будет логический 0.В логических выражениях применяются обозначения:

либо $F = \overline{A + B}$, но при этом из контекста должно быть ясно, что данное сложение именно логическое;

либо $F = \overline{A \vee B}$.



а)

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

б)

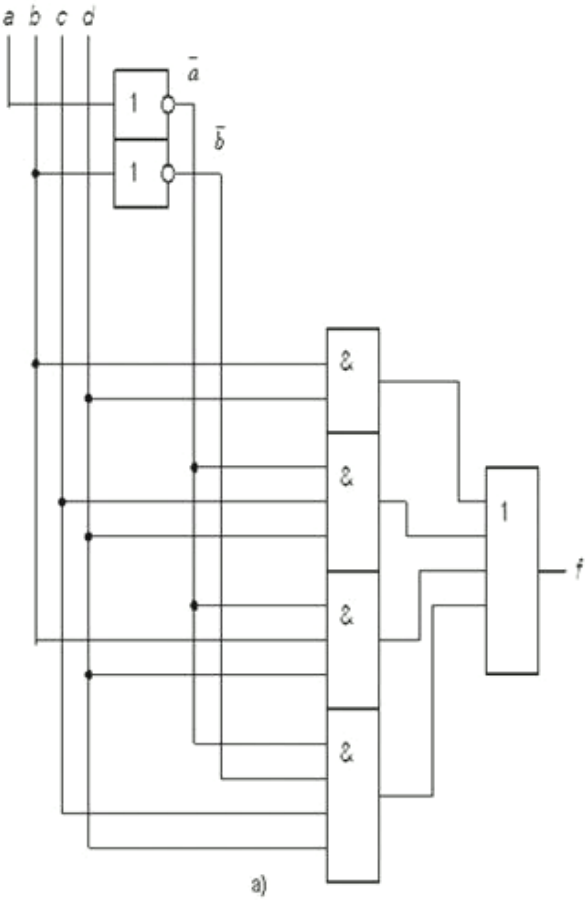
Рис. 3. Логический элемент ИЛИ-НЕ на два входа

Логические схемы. Составление таблиц истинности для логических схем. Для **логических схем**, представляющих собой соединение нескольких **логических элементов**, в левой части таблицы перечисляются все возможные комбинации входных сигналов, а в правой части - соответствующие значения на выходе логической схемы. Очевидно, что левые части таблицы будут одинаковыми для всех функций двух переменных, для всех функций трёх переменных и т.д. Традиционно комбинации сигналов в них располагают в порядке возрастания соответствующих двоичных кодов. На рис. 4 приведен пример логической схемы и таблица истинности, полностью описывающая ее работу.

Вероятность ошибки уменьшается, если не решать задачу «в лоб», а проанализировать её работу с точки зрения уже известных нам правил логического сложения, умножения и инверсии. Очевидно, что в рассматриваемой схеме осуществляется логическое сложение нескольких логических произведений.

Можно записать логическое выражение, соответствующее данной схеме:

$$f = bd + \overline{a}cd + \overline{a}bd + \overline{a}\overline{b}cd \qquad (1.1)$$



a	b	c	d	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

б)

Рис. 4. Логическая схема и соответствующая ей таблица истинности

Булево выражение в виде суммы произведений называется **дизъюнктивно нормальной формой (ДНФ)**.

Булево выражение в виде произведения сумм **называется конъюнктивной нормальной формой (КНФ).**

По **правилу логического сложения** выражение (1.1) имеет на выходе логическую 1 $f = 1$ только в том случае, если равно 1 хотя бы одно из четырех произведений, входящих в сумму. По **правилу логического умножения** каждое

произведение будет равно 1 только в том случае, когда все входящие в произведение переменные равны 1. Рассмотрим все эти возможности отдельно и по порядку.

Произведение *bd* будет равно 1 только тогда, когда будет выполняться условие: и *b=1*, и *d=1*. При этом от значений остальных входных переменных - *a* и *c* - значение данного произведения не зависит. Поэтому логические 1 будут в строках, соответствующих полным произведениям *abcd*, в которых *b=d=1*, а переменные *a* и *c* перечисляются во всех четырех возможных комбинациях: *abcd=0101, 0011,* 1101 и 1111.

Произведение $\overline{a}cd$ будет равно 1 только тогда, когда будет выполняться условие: и $\overline{a}=1$ (т.е. $a=0$), и $c=1$, и $d=1$. От значения не вошедшей в данное произведение переменной b произведение $\overline{a}cd$ не зависит. Поэтому логические 1 будут в строках таблицы истинности, соответствующих полным произведениям $abcd$, в которых $c=d=1$ и одновременно $a=0$, а переменная b перечисляется во всех двух возможных комбинациях: $abcd=0011$ и 0111.

Произведение $\overline{a}bd$ будет равно 1 только тогда, когда будет выполняться условие: и $\overline{a}=1$ (т.е. $a=0$), и $b=1$, и $d=1$. От значения не вошедшей в данное произведение переменной c произведение $\overline{a}bd$ не зависит. Поэтому логические 1 будут в строках таблицы истинности, соответствующих полным произведениям $abcd$, в которых $b=d=1$ и одновременно $a=0$, а переменная c перечисляется во всех двух возможных комбинациях: $abcd=0101$ и 0111.

Произведение $\overline{a}\overline{b}cd$ будет равно 1 только тогда, когда будет выполняться условие: и $\overline{a}=1$(т.е. $a=0$), $\overline{b}=1$ (т.е. $b=0$), и $c=1$ и $d=1$. Поэтому логическая 1, соответствующая данному полному произведению всех переменных, будет только в той строке таблицы истинности, где $abcd=0011$.

Анализ всех этих возможностей показывает, что они могут совпадать для нескольких произведений. Например,

комбинация входных переменных 0011 встречается в произведениях $\bar{a}cd$ и $\bar{a}\bar{b}cd$. А сочетание 0111 встречается даже в трех произведениях: и в bd, и в $\bar{a}cd$, и в $\bar{a}bd$. Это говорит о том, что для данного логического выражения есть возможности минимизации.

Ключевые термины

ДНФ - дизъюнктивно-нормальная форма - представление логического выражения в виде суммы произведений.

Инверсия - операция НЕ- логическое действие, при котором появление хотя бы одного логического нуля на входе даёт логический нуль на выходе.

Инвертор - логический элемент, реализующий операцию НЕ.

КНФ - конъюктивно-нормальная форма - представление логического выражения в виде произведения сумм.

Логическая переменная - переменная, значение которой может быть равно либо логическому нулю, либо логической единице.

Логическая схема - схема, состоящая из логических элементов.

Логическая функция - функция, включающая в себя логические переменные, значение которой может быть равно либо логическому нулю, либо логической единице.

Логический элемент - графическое представление элементарной логической функции.

Логическое отрицание - операция НЕ, инверсия - логическое действие, при котором происходит изменение состояния на противоположное.

Логическое сложение - операция ИЛИ, дизъюнкция - логическое действие, при котором появление хотя бы одной логической единицы на входе даёт логическую единицу на выходе.

Логическое умножение - операция И, конъюнкция - логическое действие, при котором появление хотя бы одного логического нуля на входе даёт логический нуль на выходе.

Таблица истинности - таблица, содержащая все возможные комбинации входных логических переменных и соответствующие им значения логической функции.

Логическая формула. С помощью логических переменных и символов логических операций любое высказывание можно формализовать, то есть заменить логической формулой.

Опр: *Всякая логическая переменная и символы «истина» («1») и «ложь» («0») — формулы.*

Если A и B — формулы, то их отрицание, конъюнкция, дизъюнкция, имплекация и эквивалентность — формулы. Никаких других формул в алгебре логики нет.

Для упрощения записи формул алгебры высказываний принято соглашение о скобках.

Опр: *Формула алгебры высказываний называется* ***тождественно-истенной*** *, если она принимает значение истина, при любых значениях переменных.*

Опр: *Формула алгебры высказываний называется* ***тождественно-ложной*** *, если она принимает значение ложь, при любых значениях переменных.*

Опр: *Тождественно – истинные формулы алгебры высказываний наз. законами логики.*

Значение высказанной переменной или формулы обозначается $|A|$ (0 или 1).

Опр: Две формулы алгебры высказываний будут называться равносильными $A \equiv B$, тогда и только тогда, когда для любых значений переменных значения этих формул совпадает.

Тождественные формулы или законы логики:

1	Переместительный закон	A∧B≡B∧A.	A∨B≡B∨A.
2	Сочетательный закон	(A∧B) ∧C≡A∧(B∧C)	(A∨B)∨C≡A∨(B∨C)
3	Распределительный закон	A∧(B∨C)≡(A∧B)∨(B∧C)	A∨(B∧C)≡(A∨B)∧(B∨C)

4	Закон де Моргана	$\overline{A \wedge B} \equiv \overline{A} \vee \overline{B}$	$\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$
5	Закон двойного отрицания	$\overline{\overline{A}} \equiv A$	
6	Закон поглощения	A∧A≡A	A∨A≡A
		A∧1≡A	A∨1≡1
		A∧0≡0	A∨0≡A
		$A \wedge \overline{A} \equiv 0$	$A \vee \overline{A} \equiv 1$

Математический аппарат алгебры логики очень удобен для описания того, как функционируют аппаратные средства компьютера, поскольку основной системой счисления в компьютере является двоичная, в которой используются цифры 1 и 0, а значений логических переменных тоже два: «1» и «0».

Из этого следует два вывода:

- одни и те же устройства компьютера могут применяться для обработки и хранения как числовой информации, представленной в двоичной системе счисления, так и логических переменных;

- на этапе конструирования аппаратных средств алгебра логики позволяет значительно упростить логические функции, описывающие функционирование схем компьютера, и, следовательно, уменьшить число элементарных логических элементов, из десятков тысяч которых состоят основные узлы компьютера.

Логический элемент компьютера — *это часть электронной логической схемы, которая реализует элементарную логическую функцию.* Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И—НЕ, ИЛИ—НЕ и другие (называемые также вентилями), а также триггер. С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у вентилей бывает от двух до восьми входов и один или два выхода. Чтобы представить два логических состояния — «1» и «0» в вентилях, соответствующие им входные и выходные сигналы имеют один из двух установленных уровней напряжения. Каждый

логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая именно электронная схема в нем реализована. Это упрощает запись и понимание сложных логических схем. Работу логических элементов описывают с помощью таблиц истинности.

Таблица истинности это табличное представление логической схемы (операции), в котором перечислены все возможные сочетания значений истинности входных сигналов (операндов) вместе со значением истинности выходного сигнала (результата операции) для каждого из этих сочетаний.

Триггер — *это электронная схема, широко применяемая в регистрах компьютера для надёжного запоминания одного разряда двоичного кода.* Триггер имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю. Термин триггер происходит от английского слова trigger — защёлка, спусковой крючок. Самый распространённый тип триггера — так называемый RS-триггер (S и R, соответственно, от английских set — установка, и reset — сброс).

Сумматор — *это электронная логическая схема, выполняющая суммирование двоичных чисел*. Сумматор служит, прежде всего, центральным узлом арифметико-логического устройства компьютера, однако он находит применение также и в других устройствах машины. Многоразрядный двоичный сумматор, предназначенный для сложения многоразрядных двоичных чисел, представляет собой комбинацию одноразрядных сумматоров, с рассмотрения которых мы и начнём.

Согласно определению, таблица истинности логической формулы выражает соответствие между всевозможными наборами значений переменных и значениями формулы. Для формулы, которая содержит две переменные, таких наборов значений переменных всего четыре: (0, 0), (0, 1), (1, 0), (1, 1).

Если формула содержит три переменные, то возможных наборов значений переменных восемь: (0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1).

Количество наборов для формулы с четырьмя переменными равно шестнадцати и т.д. Удобной формой записи при нахождении значений формулы является таблица, содержащая кроме значений переменных и значений формулы также и значения промежуточных формул.

Пример: Составить таблицу истинности для $\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$

A	B	A∨B	$\overline{A \vee B}$	$\overline{A}$	■	■■■	$\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$
1	1	1	0	0	0	0	1
1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1
0	0	0	1	1	1	1	1

Вывод: Выражение $\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$ является тождественно-истинным.

Равносильные преобразования логических формул имеют то же назначение, что и преобразования формул в обычной алгебре. Они служат для упрощения формул или приведения их к определённому виду путем использования основных законов алгебры логики.

Под упрощением формулы, не содержащей операций импликации и эквиваленции, понимают равносильное преобразование, приводящее к формуле, которая либо содержит по сравнению с исходной меньшее число операций конъюнкции и дизъюнкции и не содержит отрицаний неэлементарных формул, либо содержит меньшее число вхождений переменных.

Некоторые преобразования логических формул похожи на преобразования формул в обычной алгебре (вынесение общего множителя за скобки, использование переместительного и сочетательного законов и т.п.), тогда как другие преобразования основаны на свойствах, которыми не обладают операции обычной алгебры (использование распределительного закона

для конъюнкции, законов поглощения, склеивания, де Моргана и др.). Покажем на примере некоторые приемы и способы, применяемые при упрощении логических формул:

(законы алгебры логики применяются в следующей последовательности: правило де Моргана, сочетательный закон, правило операций переменной с её инверсией и правило операций с константами).

Любая цифровая вычислительная машина состоит из логических схем. Логические схемы, в свою очередь, состоят из логических элементов. Самыми простыми логическими элементами являются элементы И, ИЛИ и НЕ. Им соответствуют функции логического умножения, сложения и инверсии.

Задание 1 (Общее задание для всех):

1. Нарисуйте элементы И на два, четыре и пять входов, составьте для каждого из них таблицу истинности, напишите соответствующее каждому элементу логическое выражение.
2. Нарисуйте элементы ИЛИ на три, четыре и пять входов, составьте для каждого из них таблицу истинности, напишите соответствующее каждому элементу логическое выражение.
3. Нарисуйте элементы И-НЕ на два, четыре и пять входов, составьте для каждого из них таблицу истинности, напишите соответствующее каждому элементу логическое выражение.
4. Нарисуйте элементы ИЛИ-НЕ на три, четыре и пять входов, составьте для каждого из них таблицу истинности, напишите соответствующее каждому элементу логическое выражение.
5. Какой уровень сигнала является решающим для логического сложения? для логического умножения? для функции И-НЕ? для функции ИЛИ-НЕ?
6. Что такое таблица истинности?
7. Сколько строк в таблице истинности для 5-входовой логической схемы? для 4-входовой? для 2-входовой?

8. Функция скольких переменных описывается таблицей истинности длиной 4 строки? 64 строки? 512 строк?

Задание 2 (по вариантам): Построить таблицы истинности для выражений, упростить выражения и определить истинность высказываний. Отчет с выкладками оформить в MS Excel.

Вариант 1

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Если 2 кратно 5 и неверно, что 3 больше 5, то 8 делиться на 4 или 2 – нечетное число»

Вариант 2

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Если Париж столица Франции и вы живете в Париже, то вам не больше 20 лет и вы мужчина»

Вариант 3

1. Построить таблицу истинности для выражения: [redacted]

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Вы гражданин Казахстана или России, поэтому вы в совершенстве владеете русским языком и сочиняете стихи»

Вариант 4

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Вы девушка и не верно, что вам больше 20 лет, следовательно, ваш рост не более 170 сантиметров»

Вариант 5

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Вы являетесь гражданином Казахстана тогда и только тогда, когда ваш возраст не менее 16 лет и вы в совершенстве владеете казахским языком».

Вариант 6

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted].

3. Определить истинность высказывания: «Если неверно, что 8 кратно 4 и 1 – простое число, то неверно, что 2!=2 или любое число в нулевой степени равно 1»

Вариант 7

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted].

3. Определить истинность высказывания: «9 кратно 3 и 10-2=8 тогда и только тогда, когда 13 простое число или 2 – нечетное число»

Вариант 8

1. Построить таблицу истинности для выражения: [redacted]

2. Упростить выражение: [redacted].

3. Определить истинность высказывания: «Если 8 кратно 3 и 1 – простое число, то неверно, что 2!=2 или любое число в нулевой степени равно 1» .

Вариант 9

1. Построить таблицу истинности для выражения: [redacted]

2. Упростить выражение: [redacted].

3. Определить истинность высказывания: «2 меньше 5 или неверно, что 8 кратно 3 и 1 – простое число»

Вариант 10

1. Построить таблицу истинности для выражения: [redacted]

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Если вы любите музыку или танцы, то не верно, что вам 16 лет и вы учитесь на музыкально-педагогическом факультете»

Вариант 11

1. Построить таблицу истинности для выражения: [redacted].

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Если неверно, что 9 кратно 4 и 1 – четное число, то неверно, что любое число в нулевой степени равно 1»

Вариант 12

1. Построить таблицу истинности для выражения: [redacted]

2. Упростить выражение: [redacted]

3. Определить истинность высказывания: «Если Париж столица Франции и вы хоть раз бывали в Париже, то вы путешественник и вам больше 20 лет»

Лабораторная работа № 15-16 «Нахождение эффективного адреса при базово-индексной адресации со смещением. Определение объёма жёсткого диска».

Цель работы: Научиться находить эффективный адрес при базово-индексной адресации со смещением и определять объём жёсткого диска и расшифровывать характеристики НЖМД.

Теоретическая информация:

Методы адресации

В машинах с регистрами общего назначения метод (или режим) адресации объектов, с которыми манипулирует команда, может задавать константу, регистр или ячейку памяти. Для обращения к ячейке памяти процессор прежде всего должен вычислить действительный или эффективный адрес памяти, который определяется заданным в команде методом адресации.

Система команд 32-разрядных процессоров предусматривает 11 режимов адресации. При этом только в двух случаях операнды не связаны с памятью. Это операнд-содержимое регистра, которое берется из любого 8-, 16- или 32-битного регистра процессора, и непосредственный операнд (8, 16 или 32 бит), который содержится в самой команде. Остальные девять режимов так или иначе обращаются к памяти.

При обращении к памяти эффективный адрес вычисляется с использованием следующих компонентов.

– Смещение (Displacement или Disp) – 8-, 16- или 32-битное число, включенное в команду.

– База (Base) – содержимое базового регистра. Обычно используется для указания на начало некоторого массива.

– Индекс (Index) – содержимое индексного регистра. Обычно используется для выбора элемента массива.

– Масштаб (Scale) – множитель (1, 2, 4 или 8), указанный в коде инструкции. Этот элемент используется для указания размера элемента массива, доступен только при 32-битной адресации.

Эффективный адрес вычисляется по формуле:

EA=Base + Index + Scale + Disp.

Использование сложных методов адресации позволяет существенно сократить количество команд в программе, но при этом значительно увеличивается сложность аппаратуры. Важным вопросом построения любой системы команд является оптимальное кодирование команд. Оно определяется количеством регистров и применяемых методов адресации, а также сложностью аппаратуры, необходимой для декодирования. Именно поэтому в современных RISC архитектурах используются достаточно простые методы адресации, позволяющие резко упростить декодирование команд. Более сложные и редко встречающиеся в реальных программах методы адресации реализуются с помощью дополнительных команд, что приводит к увеличению размера программного кода. Однако такое увеличение длины программы с лихвой окупается возможностью простого увеличения тактовой частоты RISC-процессоров.

Устройство накопителей на жестких дисках

В настоящее время как основными производителями, так и дочерними фирмами выпускаются несколько десятков типов накопителей на жестких дисках. Зачастую используются оригинальные конструкционные материалы, имеются отличия в расположении узлов, но принципы работы большинства накопителей одинаковы (рис. 5).

Взглянув на накопитель на жестком диске, вы увидите только прочный металлический корпус. Он полностью герметичен и защищает дисковод от частичек пыли, которые при попадании в узкий зазор между головкой и поверхностью диска могут повредить чувствительный магнитный слой и вывести диск из строя. Кроме того, корпус экранирует накопитель от электромагнитных помех.

Рис. 5. Основные элементы накопителя на жестких дисках

Дисковод – устройство, которое содержит механизмы для вращения магнитного диска и перемещения головки чтения и записи по его поверхности.

Головка считывания/записи – магнитная головка, позволяющая осуществлять чтение и запись данных на диск.

Внутри корпуса находятся все механизмы и некоторые электронные узлы.

Механизмы – это сами диски, на которых хранится информация, головки, которые записывают и считывают информацию с дисков, а также двигатели, приводящие все это в движение.

Диск представляет собой круглую металлическую пластину с очень ровной поверхностью, покрытую тонким ферромагнитным слоем. Технология его нанесения близка к той, которая используется при производстве интегральных микросхем.

Количество дисков может быть различным, количество рабочих поверхностей, соответственно, вдвое больше (по две на каждом диске). Последнее (как и материал, использованный для магнитного покрытия) определяет емкость жесткого диска. Иногда наружные поверхности крайних дисков (или одного из них) не используются, что позволяет уменьшить высоту

накопителя, но при этом количество рабочих поверхностей уменьшается и может оказаться нечетным.

Магнитные головки считывают и записывают информацию на диски. Принцип записи в общем схож с тем, который используется в обычном магнитофоне. Цифровая информация преобразуется в переменный электрический ток, поступающий на магнитную головку, а затем передается на магнитный диск, но уже в виде магнитного поля, которое диск может воспринять и «запомнить».

Магнитное покрытие диска представляет собой множество мельчайших областей самопроизвольной (спонтанной) намагниченности. Для наглядности представьте себе, что диск покрыт слоем очень маленьких стрелок от компаса, направленных в разные стороны. Такие частицы-стрелки называются доменами. Под воздействием внешнего магнитного поля собственные магнитные поля доменов ориентируются в соответствии с его направлением. После прекращения действия внешнего поля на поверхности диска образуются зоны остаточной намагниченности. Таким образом сохраняется записанная на диск информация. Участки остаточной намагниченности, оказавшись при вращении диска напротив зазора магнитной головки, наводят в ней электродвижущую силу, изменяющуюся в зависимости от величины намагниченности.

Пакет дисков, смонтированный на оси-шпинделе, приводится в движение специальным двигателем, компактно расположенным под ним. Для того чтобы сократить время выхода накопителя в рабочее состояние, двигатель при включении некоторое время работает в форсированном режиме. Поэтому источник питания компьютера должен иметь запас по пиковой мощности.

Головки перемещаются с помощью прецизионного шагового двигателя и как бы «плывут» на расстоянии в доли микрона от поверхности диска, не касаясь его. Держатель головки представляет собой крыло, парящее над поверхностью, благодаря тому, что поверхность увлекает с собой частицы воздуха, создавая таким образом набегающий на крыло поток.

На поверхности дисков в результате записи информации образуются намагниченные участки в форме концентрических окружностей. Они называются магнитными дорожками.

Дорожка – концентрическое кольцо на поверхности магнитного диска, на которое записываются данные.

Сектор – деление дисковых дорожек, представляющее собой основную единицу размера, используемую накопителем. Секторы обычно содержат по 512 байтов.

Совокупность дорожек, расположенных друг под другом на всех поверхностях, называют **цилиндром**. Все головки накопителя перемещаются одновременно, осуществляя доступ к одноименным цилиндрам с одинаковыми номерами.

Число дисков, головок и дорожек накопителя устанавливается изготовителем исходя из свойств и качества дисков. Изменить эти характеристики нельзя. Количество секторов на диске зависит от метода записи. В одном секторе располагается 512 байт (в системе DOS). Зная эту величину, всегда можно рассчитать общий объем накопителя:

$$\mathbf{V = C \cdot H \cdot S \cdot B,}$$

где C – количество цилиндров; H – количество головок; S – количество секторов на дорожку; B – размер сектора.

Описанное выше разбиение называется низкоуровневым (LowLewel) форматированием. Такое форматирование нижнего уровня чаще всего выполняет изготовитель, используя специальные программные средства (например, Speed Store или Disk Manager) или команды DOS. Перед первым использованием дисков необходимо произвести их логическое форматирование – специальным образом инициализировать их (с помощью программы format).

Хранение и извлечение данных с диска требует взаимодействия между операционной системой, контроллером жесткого диска и электронными и механическими компонентами самого накопителя.

Краткая характеристика интерфейсов жестких дисков

Основная функция интерфейса – передача данных из системы в накопитель и обратно. От типа интерфейса зависит, с

какой скоростью будут осуществляться эти операции, а это и определяет производительность компьютера.

Со времени создания персональных компьютеров было разработано несколько типов интерфейсов: ST-506/412, ESDI, IDE, SCSI. Из них только первые два можно считать собственно интерфейсами для обмена информацией между контроллером и жестким диском. SCSI и IDE – интерфейсы системного уровня, в которых контроллер выполнен в виде микросхемы, установленной на плате накопителя. В интерфейсе SCSI между контроллером и системной шиной вводится еще один уровень организации данных и управления, а интерфейс IDE взаимодействует с системной шиной непосредственно.

Интерфейс ST-506/412 разработан фирмой Seagate Technologies в 1982 г. Впервые он был использован в накопителе размером 5,25» емкостью 12 Мбайт. Подобные накопители использовались в качестве стандарта для PC XT и AT 286. Самыми известными из них являются два устройства фирмы Seagate: ST225 объемом 21,4 Мбайт и средним временем доступа 65 мс и ST251 (42,8 Мбайт, 28 мс). В обоих случаях речь идет о накопителях 5,25» половинной высоты (2,6»). В литературе эти накопители иногда называют MFM-накопителями (по способу кодирования информации).

Интерфейс ESDI (Enhanced Small Device Interface – усовершенствованный интерфейс малых устройств) – специализированный интерфейс накопителей на жестких дисках, разработанный фирмой Maxtor. ESDI-накопители обычно бывают полной высоты и находятся в корпусе 5,25». ESDI-накопители имеют до 53 секторов на дорожку и принадлежат к первым накопителям, емкость которых достигла 100 Мбайт. Поэтому область их применения – в первую очередь, сетевые серверы и высокоскоростные устройства (по меркам прошлых лет). По сравнению с ST-506/412, в интерфейсе ESDI предприняты меры по сокращению числа ошибок считывания данных, в частности, шифратор/дешифратор расположен непосредственно на плате накопителя. Скорость передачи данных в этом стандарте может достигать 24

Мбайт/с, хотя на практике она составляет 10–15 Мбайт/с. При использовании интерфейса ESDI можно считывать с жесткого диска карту расположения поверхностных дефектов (соответствующая информация завода-изготовителя может храниться на диске в служебном файле).

Дальнейшее совершенствование интерфейсов шло по пути объединения контроллера и накопителя на жестких дисках, что позволило повысить тактовую частоту шифратора/дешифратора, плотность размещения данных на носителе и общее быстродействие системы. Официальное название интерфейса IDE, признанного ANSI в марте 1989 г., – ATA (AT Attachment). Главные достоинства IDE-накопителей – дешевизна и быстродействие. Стандарты ATA прошли долгий путь эволюции, пока не была решена проблема совместимости, возникающая при подключении к шинам ISA и EISA. В CMOS Setup первые IDE-накопители можно было устанавливать с их физическими параметрами (Nature Mode) или указывать логический тип. Современные IDE-накопители поддерживают универсальный режим трансляции, при котором основным критерием выбора параметров накопителя является общее количество секторов данной модели. Большинство BIOS персональных компьютеров имеют процедуру «Autodetect», которая позволяет считывать и устанавливать паспортные параметры накопителя. Некоторые накопители, например, фирмы Conner, используют адаптивный режим трансляции, в котором накопитель сигнализирует о неправильном использовании дискового пространства. При инициализации накопителю передаются два параметра: количество головок и секторов; затем накопитель подстраивает свою логическую структуру таким образом, чтобы общая емкость не изменилась, причем коррекция осуществляется за счет цилиндров.

Спецификация стандарта ATA-2 (EIDE) была разработана фирмой Western Digital. Аналогичные стандарты Fast-ATA и Fast-ATA-2 были приняты фирмами Seagate и Quantum. Можно выделить четыре области, в которых стандарт ATA-2 претерпел существенные изменения по сравнению с исходным:

– увеличение максимальной емкости накопителей;

– увеличение скорости обмена данными;

– появление вторичного канала для подключения двух устройств;

– использование интерфейса ATAPI.

Максимальная емкость накопителей ATA-2 значительно увеличена за счет разработки улучшенной BIOS (Enhanced BIOS), что позволило преодолеть барьер в 504 Мбайт емкости жесткого диска. Появление этого ограничения связано с физическими параметрами жесткого диска (количество цилиндров, головок, секторов).

SCSI-накопители имеют самую высокую скорость обмена данными. Но их достоинством является не столько скорость обмена информацией, сколько вся SCSI-система как таковая. SCSI Host-адаптер может управлять не только накопителем, но и всеми периферийными устройствами, которые подключены к нему и поддерживают протокол SCSI. Для SCSI-накопителей (как и для IDE-накопителей) ни в коем случае нельзя выполнять низкоуровневое форматирование, поскольку при этом теряется информация о важнейших эксплуатационных параметрах, необходимых для обеспечения работоспособности устройства. Стандарт SCSI за время своего развития претерпел существенные изменения. Совершенствование интерфейса SCSI происходит и в настоящее время.

При сравнении возможностей накопителей на жестких дисках IDE и SCSI учитываются несколько факторов. При тестировании IDE-накопители в большинстве случаев оказываются эквивалентными SCSI-устройствам. В IDE-накопителях при передаче данных из каждого сектора на вспомогательные операции затрачивается меньше времени, чем в SCSI (дополнительные задержки связаны с установлением соглашения о синхронизации, выбором жесткого диска-адреса, запросом данных, сигналом окончания передачи, преобразованием логических адресов в физические, выраженные в значениях цилиндров, головок и секторов). В результате интерфейс IDE имеет неоспоримые преимущества

при последовательном обмене данными, характерном для однозадачной операционной системы. При работе в многозадачной системе производительность SCSI-устройств выше. Архитектура SCSI-накопителей сложнее архитектуры накопителей IDE.

Характеристики накопителей на жестких дисках

При оценивании достоинств того или иного накопителя на жестких дисках (или семейства накопителей), а также возможных ограничений обычно пользуются набором критериев оценки качества устройств.

Рассмотрим параметры, характеризующие накопители на жестких дисках.

Скоростные параметры. *Среднее время доступа к данным* (Average Access Time) – это среднее время, за которое головка перемещается к нужной дорожке диска, устанавливается на нее и начинает считывать данные. Измеряется оно в миллисекундах (мс) и составляет в настоящее время 8–11 мс. Данный параметр улучшается медленно, так как совершенствовать механику трудно. Иногда время чтения меньше времени записи. Параметр обязательно сообщается в торговых предложениях. Для конкретного диска его можно оценить утилитами, например, Norton SI, Checkit.

Среднее время доступа имеет значение, например, когда архивируется целый набор файлов, так как в это время головки интенсивно перемещаются от файла к файлу.

Скорость вращения (Rotational Speed, Spindle Speed) – скорость вращения дисков, измеряемая в оборотах в минуту (RPM – Rotational Per Minute). Параметр относится к основным, так как пропорционально скорости вращения диска растет скорость обмена данными между винчестером и шиной данных системной платы. Для дисков пользовательских компьютеров сейчас скорость вращения составляет 5400 и 7200 об/мин. Более высокооборотные диски (10000 об/мин и более) имеют SCSI-интерфейс. Они очень дороги и предназначены для серверов.

С ростом скорости вращения появляются проблемы вибрации, шума и нагрева. Наилучшим решением в ближайшее время

будет использование гидродинамических подшипников, впервые внедренных фирмой Seagate.

При скорости 5400 об/мин никаких специальных мер по охлаждению применять не нужно. При скорости 7200 об/мин диск нужно устанавливать посередине хорошо вентилируемого корпуса и обеспечивать свободное пространство для лучшего теплоотвода. При скорости вращения 10 000 об/мин применяют обдув диска отдельным вентилятором.

Перегрев диска приводит к температурным расширениям механики и, как следствие, ухудшает распознавание дорожек. Это вызывает замедление работы (что недопустимо для работы с аудио- и видеоинформацией в реальном времени).

В связи с этим, например, компания Quantum постоянно принимает меры по увеличению плотности записи, что позволяет получить ту же скорость доступа к данным, но при меньших оборотах.

Внутренняя скорость обмена (Internal Data Rate) – скорость обмена между поверхностью диска и буфером (Media to Buffer). Измеряется в мегабитах в секунду. Порядок чисел – 200 Мбит/с, или 20 Мбайт/с. Однако это пиковая скорость, реальная – 10–12 Мбит/с. В эту скорость неявно входят как множители скорость вращения и линейная плотность записи. К сожалению, данный параметр редко указывается в предложениях, несмотря на то, что отражает реальную скорость жесткого диска. Измеряется он, например, утилитой Norton SI.

Внешняя скорость обмена (Data Transfer Rate Buffer-to-Host) – это скорость обмена между буфером и контроллером канала (Host). Определяется интерфейсом, поддерживаемым диском (а также чипсетом со стороны системной платы). Она с запасом превосходит скорость считывания данных с диска, поэтому не очень существенна.

Среднее время перехода на соседнюю дорожку (Track-To-Track Seek Time) имеет значение только при работе с большими (не фрагментированными) файлами, поэтому редко указывается. Измеряется в миллисекундах. Типичное значение – около 1,5–3 мс.

Параметры надежности. *Стойкость к ударам* (Shock resistance). В механике под ударом понимается кратковременное воздействие значительной внешней силы. Стойкость к ударам, после которых устройство остается работоспособным, определяется ускорением (g – 9,8 м/с^2), а также временем воздействия.

Стойкость к ударам бывает двух типов: во время работы диска и в выключенном состоянии. Раньше диски были слабо защищены и любой удар приводил к тяжелым последствиям. В настоящее время они выдерживают удары не менее 10 g при работе и 100 g в выключенном состоянии. Падение диска на жесткий материал с высоты 10 см равнозначно воздействию в 70 g.

Существуют интересные фирменные технологии защиты. Примером является антиударная система Quantum Shock Protection System (SPS), защищающая диск при транспортировке.

Технология SMART. Название этой технологии часто записывают через точки: S.M.A.R.T. Сокращение от английского Self Monitoring Analysis Reporting Technology – самомониторинг и информирование о состоянии диска.

Это технология самоконтроля диска, и содержание ее заключается в том, что на основные компоненты (двигатели, головки, поверхности и т.д.) крепятся датчики. Информацию от датчиков постоянно обрабатывают процедуры из firmware-диска. В результате этого в самом диске накапливается и запоминается статистика. При включении компьютера программа из BIOS системной платы или ОС должна просмотреть статистику и сравнить с заранее установленными пороговыми значениями контролируемых параметров (например, число плохих секторов). Как только контролируемый параметр выходит за допустимые пределы, выдается сигнал на дисплей. В результате своевременно и точно выдаются предупреждение и диагностика, позволяющие принять меры (ремонт или замена) и не потерять драгоценные данные.

Технология была разработана компанией Compaq и первоначально называлась IntelliSafe. В настоящее время известна версия SMART II, которая является частью стандарта ATA -2 (EIDE).

Слабостью SMART является ее пассивность – она оповещает, но не лечит. Поэтому в настоящее время получили распространение фирменные расширения стандарта, позволяющие автоматизировать поддержку работоспособности жесткого диска. Примером является технология Data Lifeguard компании Western Digital. Через каждые несколько часов работы она тестирует поверхность диска в фоновом режиме и исправляет ошибки, вплоть до переписывания информации в резервный сектор.

Среднее время безотказной работы (Mean Time Between Failure, MTBF) – это среднее время между двумя соседними сбоями. В настоящее время данный показатель достигает 300, 400 и 500 тысяч часов, а у лучших моделей и 800 тысяч.

Параметр второстепенен для пользователя, так как предполагает, что диск включен постоянно. А такая ситуация бывает только на серверах. На самом деле время жизни диска на порядок меньше (около 5 лет), чему способствует операция включения/выключения.

Гарантированное число включений также не имеет особого значения для пользователя, так как их число достаточно велико – 40–50 тысяч.

Полезно понимать разницу между сроком гарантии и временем наработки на отказ – жесткий диск вам заменят, но бесценные данные пропадут.

Архитектурные параметры. *Число пластин.* Винчестер строится обычно на основе 1–4 пластин (реже больше). В принципе, чем меньше пластин при одинаковой емкости устройства, тем лучше: во-первых, выше плотность записи и не надо форсировать число оборотов; во-вторых, меньше деталей, а значит, выше надежность. У современных дисков емкость пластины превысила 2,5 Гбайт.

Размер кэша (Buffer Size). Кэш является аппаратным и выполняется обычно на модулях типа DRAM. Иногда называется буфером, но это настоящий кэш со своей таблицей.

Для получения требуемых данных в буфер считывается вся дорожка, где они располагаются, а затем из буфера извлекаются только нужные данные.

Размер кэша обязательно сообщается в торговых предложениях. До недавнего времени размер кэша был 128 Кбайт, сейчас используется кэш размером 512 Кбайт, причем для IDE-дисков (раньше – исключительно для SCSI).

Тип головок. В настоящее время для большинства жестких дисков применяют головки типа GR, а для более совершенных моделей используют головки типа MGR, которые способствуют более высокой плотности записи.

Задание 1: *Вычислите эффективный адрес при базово-индексной адресации со смещением.*

Пример выполнения: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 1E00h, индекс Index = 0002h, смещение Disp = 0100h.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Записать формулу для определения эффективного адреса.	Эффективный адрес: EA = Base + Index + Disp
2	Рассчитать эффективный адрес.	EA = 1E00h + 0002h + 0100h = 1F02h

Решите самостоятельно следующие задания.

Задание 1.1: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 1C00h, индекс Index = 0A00h, смещение Disp = 0602h.

Задание 1.2: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 0320h, индекс Index = 1800h, смещение Disp = 2020h.

Задание 1.3: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 0500h, индекс Index = 0030h, смещение Disp = 0D27h.

Задание 1.4: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 5400h, индекс Index = 0470h, смещение Disp = 0010h.

Задание 1.5: Вычислите эффективный адрес при базово-индексной адресации со смещением, если база Base = 1028h, индекс Index = 0250h, смещение Disp = 0060h.

Задание 2: *Рассчитайте объем жесткого диска. Данные для расчёта взять в соответствии с номером варианта.*

Пример выполнения: Рассчитайте объем жесткого диска, если количество цилиндров C = 1011, головок H = 15, секторов на дорожке S = 22, размер сектора B = 512 B.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Записать формулу для определения объема НЖМД.	Объем НЖМД: $V = C \cdot H \cdot S \cdot B$
2	Рассчитать объем НЖМД.	$V = 1011 \cdot 15 \cdot 22 \cdot 512$ B = 170818560 B « 170 MB

Решите самостоятельно следующие задания.

Задание 2.1: Рассчитайте объем жесткого диска, если количество цилиндров C = 809+n, головок H = 6+n, секторов на дорожке S = 26+n, размер сектора B = 512+n B.

Задание 2.2: Рассчитайте объем жесткого диска, если количество цилиндров C = 809+n, головок H = 6+n, секторов на дорожке S = 17+n, размер сектора B = 512+n B.

Задание 2.3: Рассчитайте объем жесткого диска, если количество цилиндров C = 1024+n, головок H = 5+n, секторов на дорожке S = 26+n, размер сектора B = 512+n B.

Задание 2.4: Рассчитайте объем жесткого диска, если количество цилиндров C = 918+n, головок H = 15+n, секторов на дорожке S = 17+n, размер сектора B = 512+n B.

Задание 2.5: Рассчитайте объем жесткого диска, если количество цилиндров C = 1024+n, головок H = 16+n, секторов на дорожке S = 17+n, размер сектора B = 512+n B.

Задание 3: *Расшифруйте следующее обозначения характеристик НЖМД.*

Пример выполнения: Расшифруйте следующее обозначение характеристик НЖМД:

HDD IBM 30.7 GB IDE 7200 rpm.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить производителя.	Фирма: IBM
2	Определить объем НЖМД.	Объем НЖМД: 30,7 Гб
3	Определить тип интерфейса.	Тип интерфейса: IDE
4	Определить скорость вращения.	Скорость вращения: 7200 об/мин.

Решите самостоятельно следующие задания.

Задание 3.1: Расшифруйте следующее обозначение характеристик НЖМД:HDD Seagate 20,4 GB IDE 7200 rpm.

Задание 3.2: Расшифруйте следующее обозначение характеристик НЖМД:HDD Quantum 30,0 GB IDE 7200 rpm.

Задание 3.3: Расшифруйте следующее обозначение характеристик НЖМД:HDD Fujitsu 10,2 GB IDE 5400 rpm.

Задание 3.4: Расшифруйте следующее обозначение характеристик НЖМД:HDD Seagate 10,2 GB IDE 5400 rpm.

Задание 3.5: Расшифруйте следующее обозначение характеристик НЖМД: HDD IBM 61,4 GB IDE 7200 rpm.

Лабораторная работа № 17-18 «Расчет параметров внутренней памяти компьютера»

Цель: Научиться решать задачи по нахождению параметров внутренней памяти компьютера.

Задание: решить задачи на расчет параметров внутренней памяти компьютера. Для их решения должны быть знакомы с понятиями: объем памяти, размер машинного слова, принципы адресации памяти, связь между единицами памяти бит, байт, килобайт, мегабайт.

Теоретическая информация:

Оценка информации, так же как вещества или энергии, может быть субъективной и объективной. В первом случае главное - смысл информации, а во втором - её измеримость. Смысл информации для машины, обрабатывающей её - это абсурд. Субъективная оценка информации не является универсальной. При объективной оценке информации следует отрешиться от содержания её человеческой «важности» для кого бы то ни было. Измеримость информации достигается использованием искусственных органов чувств -устройств, приборов, датчиков и т. д. Они не только расширяют пределы слышимого и видимого, но и могут обрабатывать сигналы, не доступные органам чувств человека. Информацию измеряют так же, как измеряют вещество и энергию, приняв некоторую единицу измерения за эталон. Что же принять за эталон? Компьютерная техника, как известно, работает, «питаясь» электрическим то - ком – явлением с двумя состояниями. Если обозначить эти состояния (ведь смысл самого явления не важен!), получится алфавит из двух символов или двоичный

алфавит. Количество информации, которое содержит символ двоичного алфавита, принято за единицу измерения информации. Так как речь идет о единицах и количестве, то в качестве символов используются цифры. Исторически случилось так, что этими цифрами стали 0 и 1. Единица измерения количества информации называется БИТ (от англ. bit, сокращенно от binary digit – двоичная цифра). Бит - самое короткое слово двоичного алфавита и самая маленькая единица измерения информации, причем символы 0 и 1 равноправны. Для удобства введена более крупная единица измерения количества информации, принятая Международной системой СИ за основную, - БАЙТ (англ. byte).

Первый способ измерения информации отражает вероятностный (содержательный) подход. Этот метод

называется субъективным Информация - сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределенности, неполноты знаний. Точнее раскрыть суть определения помогает понять то, какие задачи помогает решить информация (а не то, что собой она представляет). Информация устраняет неопределенность, предоставляет человеку сделать выбор в пользу какого-либо варианта исхода некоторого события. Таким образом, информация начинает играть роль в том случае, если, во-первых, имеется некоторый выбор вариантов и, во-вторых, если эти варианты требуется определенным способом оценить. Информация предоставляет человеку возможность дать такую оценку. Рассмотрим равновероятностные варианты. Подойдя к развилке дорог, человек, направляющийся в определенное место, неожиданно встает перед выбором, на какую же из них свернуть. Он выбирает ту дорогу, которая ведет к цели. Если он знает, по какой из дорог он доберется до места, то информация ему не нужна. С самого начала он способен оценить оба варианта. Если же вся обстановка совершенно незнакома и у него нет никаких исходных данных, то ему нужна информация. Объем информации, требующейся при полном отсутствии предварительных данных для выбора одного из двух равноценных и совершенно независимых вариантов, принято считать единицей информации и обозначать, как бит.1 бит - количество информации, позволяющее выбрать один вариант из двух равноценных, независимых вариантов. Чтобы иметь возможность сделать правильный выбор между четырьмя различными дорогами, требуется два бита информации. Поясним это на таком примере. Сначала две дороги делят на две группы по две дороги в каждой. Далее выбираем группу как один из двух равноценных вариантов, т. е. нам требуется один бит информации, После того, как будет выбрана группа, в ней опять делается выбор между двумя дорогами и нам требуется еще один бит. В сумме получаем два бита информации для выбора одного из четырех вариантов. Если бы нам

потребовалось сделать выбор из восьми вариантов, то нужно было бы три бита информации: один бит идет на выбор между двумя группами из четырех дорог, второй - в группе из четырех выбираем подгруппу, состоящую из двух дорог, третий бит - выбираем из двух дорог ту, по которой пойдем. Для выбора одного варианта из 16 требуется четыре бита информации и т.д.Здесь уже прослеживается определенная закономерность: при n битах информации нужный вариант выбирается из 2n возможных. И наоборот, указав один нужный вариант из 2n возможных и одинаково принимаемых в расчет, мы дадим информацию в n битов. Количество информации n, содержащееся в сообщении о том, что произошло одно из N равновероятных событий, определяется из решения показательного уравнения: 2^n = N = > i = log2N (формула Хартли). Если события имеют разные вероятности, то применяется формула Шеннона, имеющая вид

i= -Σ Pi log2 Pi = -(P1 log2 P1+ P2 log2 P 2 + ... + Pn log2 Pn),

где i – количество информации, N – количество возможных событий,

Pi – вероятность этих событий.

Тем самым мы получаем точное правило для определения объема информации, содержащейся в сведениях и сообщениях.

Задание 1.1 Объем оперативной памяти компьютера составляет 1/n часть мегабайта. Сколько машинных слов составляют оперативную память, если одно слово содержит 8*n бита? Где n- номер варианта по журналу.

Пример: *Объем оперативной памяти компьютера составляет 1/8 часть мегабайта. Сколько машинных слов составляют оперативную память, если одно слово содержит 64 бита?*

1. Нужно перевести объем памяти и размер машинного слова в одинаковые единицы. Удобнее всего — в байты. Обозначим объем памяти буквой M, а размер слова — W. Тогда:

M = 1/8 Мб = 1024 ·1024/8 = 131 072 байта, W = 64/8 = 8 байтов.

2. Вычислить число слов, составляющих память:

N = M/W = 131 072/8 = 16 384 слова.

Задание 1.2 Какой объем имеет оперативная память компьютера, если 3FF+n (в 16-ой сс) — шестнадцатеричный адрес последнего байта памяти?

Пример: *Какой объем имеет оперативная память компьютера, если 3FF — шестнадцатеричный адрес последнего байта памяти?*

Здесь подразумевается, что объем памяти нужно выразить десятичным числом, равным количеству байтов, составляющих ОЗУ. Адрес последнего байта задан в шестнадцатеричной системе. Поскольку нумерация байтов памяти начинается с нуля, то значит диапазон адресов от 0 до 3FF. Следовательно, число байтов памяти в шестнадцатеричной системе счисления равно $3FF + 1 = 400_{16}$.

Для получения ответа нужно перевести это число в десятичную систему счисления:

$$400_{16} = 4 \cdot 16^2 = 4 \cdot 256 = 1024 \text{ байт} = 1 \text{ Кбайт}.$$

Задание 1.3 Компьютер имеет объем оперативной памяти, равный 0,1+n Кбайт и содержит 100+n машинных слов. Укажите адрес последнего байта и адрес последнего машинного слова памяти в шестнадцатеричной форме.

Пример 3. *Компьютер имеет объем оперативной памяти, равный 0,5 Кбайт и содержит 128 машинных слов. Укажите адрес последнего байта и адрес последнего машинного слова памяти в шестнадцатеричной форме.*

Выразим размер памяти в байтах:

0,5 Кбайт = 512 байт.

Размер машинного слова определяется делением объема памяти на число слов в памяти:

512/128 = 4 байта.

Переведем величину объема памяти в шестнадцатеричную систему счисления:

$512_{10} = 2 \cdot 256 = 2 \cdot 16^2 = 200_{16}$.

Следовательно, диапазон шестнадцатеричных адресов байтов памяти: от 0 до 1FF. Отсюда адрес последнего байта

равен 1FF. Последнее машинное слово включает в себя 4 последних байта памяти: 1FC, 1FD, 1FE, 1FF. Значит, адрес последнего слова равен 1FC.

Далее приведены задачи на расчеты, связанные с параметрами информационной емкости дисков, для решения которых требуется знать связь между объемом диска и числом сторон (магнитных поверхностей), числом дорожек, числом секторов на дорожке, числом байтов в секторе. Эта связь выражается следующей формулой:

ОБЪЕМ = СТОРОНЫ × ДОРОЖКИ × СЕКТОРЫ × БАЙТЫ

Задание 1.4 Двусторонняя дискета имеет объем 120*n Кбайт. Сколько дорожек на одной стороне дискеты, если каждая дорожка содержит n+10 секторов (для вариантов с1-15) и n (для вариантов с 16 по 30) по n *1024 битов?

Пример 4. *Двусторонняя дискета имеет объем 1200 Кбайт. Сколько дорожек на одной стороне дискеты, если каждая дорожка содержит 15 секторов по 4096 битов?*

Как это уже делалось раньше, здесь нужно перейти к одной единице измерения информации. Переведем в килобайты размер сектора:

4096/8 = 512 байт =0,5 Кбайт.

Теперь вычислим информационный размер дорожки:

0,5 x 15 = 7,5 Кбайт.

Поскольку дискета двухсторонняя, то на одной стороне

1200/2 = 600 Кбайт.

Теперь можно получить окончательный ответ. Чтобы найти число дорожек на одной стороне дискеты, нужно информационный объем стороны разделить на информационный размер одной дорожки:

600/7,5 = 80 дорожек.

Задание 1.5 Односторонняя дискета имеет объем 18*n Кбайт. Сколько дорожек на диске, если каждая из них содержит n секторов, а в каждом секторе размещается по 1000+n символа из 16-символьного алфавита.

Пример 5. *Односторонняя дискета имеет объем 180 Кбайт. Сколько дорожек на диске, если каждая из них*

содержит 9 секторов, а в каждом секторе размещается по 1024 символа из 16-символьного алфавита.

Главная «хитрость» этой задачи состоит в том, что надо сообразить сколько памяти занимает 1 символ.

Этот вопрос относится к теме «Измерение информации». Один символ из 16-символьного алфавита несет 4 бита информации, поскольку $2^4 = 16$. Это значит, что и в памяти компьютера символы такого алфавита будут занимать по 4 бита. Следовательно, в одном байте помещается 2 таких символа.

Теперь можно определить размер сектора. Он равен:

1024/2 = 512 байт =0,5 Кбайт.

Поскольку на одной дорожке размещается 9 секторов, то информационный объем дорожки равен:

9 × 0,5 - 4,5 Кбайт.

Теперь можно определить число дорожек на всем диске:

180/4,5 = 40 дорожек.

Лабораторная работа № 19-20 «Определение частоты строк монитора. Расчёт объёма видеопамяти»

Цель работы: Научиться определять частоту строк монитора, расчитывать объем памяти, расшифровывать характеристики мониторов.

Теоретическая информация:

Характеристики мониторов

Монитор – устройство для наблюдения за результатами действий вычислительной системы при помощи дисплейного блока, на который поступает видеосигнал с видеоадаптера компьютера.

Все современные мониторы можно разделить на три класса по их физическому принципу действия:

1) мониторы на базе электронно-лучевой трубки (CRT);
2) жидкокристаллические дисплеи (LCD);
3) газоплазменные мониторы.

Мониторы на базе электронно-лучевой трубки. Принцип действия монитора на базе электронно-лучевой

трубки мало отличается от принципа действия обычного телевизора и заключается в том, что испускаемый катодом (электронной пушкой) пучок электронов, попадая на экран, покрытый люминофором, вызывает его свечение. На пути пучка электронов обычно находятся дополнительные электроды: модулятор, регулирующий интенсивность пучка электронов и связанную с ней яркость изображения, и отклоняющая система, позволяющая изменять направление пучка.

Любое текстовое или графическое изображение на экране монитора компьютера (так же, как и телевизора) состоит из множества дискретных точек люминофора, представляющих собой минимальный элемент изображения (растра) и называемых пикселами. Такие мониторы называют растровыми. Электронный луч в этом случае периодически сканирует весь экран, образуя на нем близко расположенные строки развертки. По мере движения луча по строкам видеосигнал, подаваемый на модулятор, изменяет яркость светового пятна и образует некоторое видимое изображение. Разрешающая способность монитора определяется числом элементов изображения, которые он может воспроизводить по горизонтали и вертикали, например 640x480 или 1024x768 пикселов.

Для формирования растра в мониторе используются специальные сигналы. В цикле сканирования луч движется по зигзагообразной траектории от левого верхнего угла до правого нижнего. Прямой ход луча по горизонтали осуществляется сигналом строчной (горизонтальной – H.Sync) развертки, а по вертикали – кадровой (вертикальной – V.Sync) развертки. Перевод луча из крайней правой точки строки в крайнюю левую точку следующей строки (обратный ход луча по горизонтали) и из крайней правой позиции последней строки экрана в крайнюю левую позицию первой строки (обратный ход луча по вертикали) осуществляется специальными сигналами обратного хода.

Таким образом, принцип действия монитора обусловливает важность следующих параметров: частота вертикальной (кадровой) развертки, частота горизонтальной (строчной) развертки и расстояние между точками.

Частота вертикальной развертки. Этот очень важный параметр, называемый также частотой регенерации изображения, определяет, как часто в течение одной секунды заново формируется изображение на экране монитора. Если, например, указывается частота 60 Гц, то это означает, что в течение одной секунды изображение формируется заново ровно 60 раз.

Названная в этом примере величина 60 Гц – очень посредственное значение для рассматриваемого параметра. При такой частоте регенерации изображение нельзя считать устойчивым (без видимого мерцания). Чем выше частота регенерации, тем меньше мерцание на экране монитора и, следовательно, меньше нагрузка на зрение. Эргономичной можно считать частоту регенерации 75 Гц. По мере увеличения диагонали экрана частота регенерации должна быть повышена до 80 или 85 Гц.

Частота горизонтальной развертки. Это второй параметр, значение которого нужно всегда находить в технических характеристиках монитора. Частота строк определяет, сколько строк формируется на экране монитора в течение одной секунды. Значение данного параметра указывается в килогерцах (кГц).

Расстояние между точками (величина «зерна»). Расстояние между точками или пикселами определяет удаление друг от друга соседних точек одного цвета (красного, зеленого, синего) на экране монитора. Чем меньше эта величина, тем резче представляемое изображение. С этим параметром обычно связывается разрешающая способность.

Разрешающая способность (разрешение) – степень точности воспроизведения изображения, часто указывается количеством пикселов, которыми можно управлять независимо.

В зависимости от используемого разрешения на экране монитора должно располагаться большее или меньшее число пикселов. Однако здесь имеется и физическое ограничение. Так, при расстоянии между точками 0,25 мм число пикселов на экране 14-дюймового монитора просто не может превысить

числа пикселов, соответствующего разрешению 800x600 точек. При выборе более высокого разрешения это будет неизбежно приводить к определенной нерезкости. Будет ли это допустимо и совместимо с частотой регенерации изображения, должен решать пользователь.

Кроме того, существуют другие крайне важные характеристики: диагональ экрана монитора; потребляемая мощность; антибликовое покрытие.

Диагональю экрана монитора, как и телевизора, называется расстояние между левым нижним и правым верхним углом экрана. Это расстояние измеряется в дюймах. Не следует путать этот параметр с диагональю рабочей области экрана, доступной для отображения информации. В отличие от телевизоров многочисленные производители под диагональю экрана понимают геометрический размер диагонали электронно-лучевой трубки и не учитывают размеры черного поля, расположенного по периметру экрана. Это черное поле не входит в рабочую область экрана. Размеры его определяются конструкцией электронно-лучевой трубки.

В качестве стандарта для PC выделились мониторы с диагональю 15». Для работы в Windows с более высоким разрешением, прежде всего, необходимо иметь монитор размером, по крайней мере, 17». Для профессиональной работы с настольными издательскими системами и САПР лучше иметь монитор с диагональю 20» или 21».

Значение *потребляемой мощности монитора* приводится в его технических характеристиках или, возможно, на стандартном шильдике с обратной стороны корпуса монитора.

У мониторов с диагональю 14» потребляемая мощность не должна превышать 60 Вт. Чем больше потребляемый монитором ток, тем выше его тепловой нагрев. Хотя еще есть мониторы, потребляющие 60–80 Вт, они должны заменяться более экономичными.

Все приведенные выше значения соответствуют мониторам с диагональю 14». Большие по размерам мониторы имеют, соответственно, большую потребляемую мощность.

Все мониторы должны иметь *антибликовое покрытие.* Вряд ли можно получить хорошую читаемость информации на мониторе без такого свойства его поверхности экрана.

При напылении поверхность экрана обрабатывается при помощи воздушного пистолета, в котором находятся песочные частицы. Такой метод характерен для дешевых мониторов. Его недостатком является то, что графика и картинки на таком экране не могут быть резкими, изображение становится смазанным и рыхлым.

Лучший способ покрытия кинескопа – это нанесение специального антибликового слоя. В этом случае на поверхность экрана электронно-лучевой трубки наносится химическое вещество, обеспечивающее эффект, в результате которого свет не может отражаться от поверхности. Этот метод применяется в таких высокочувствительных приборах, как фотоаппараты, микроскопы, очки и так далее. Этот слой можно узнать по пленке с голубым оттенком.

При подобной обработке поверхность не будет волнистой, как при напылении, а останется без изменения, поэтому контуры изображения будут совершенно четкими. Недостатком этого метода являются значительные затраты, необходимые для нанесения антибликового слоя.

Соответствие стандартам безопасности. В настоящее время достаточно распространены мониторы с низким уровнем излучения – так называемые LR-мониторы (Low Radiation). Они отвечают одной из спецификаций международного стандарта, устанавливающих предельные величины статических и низкочастотных полей, излучаемых мониторами.

Спецификация MPR I устанавливает нормы в основном для магнитных полей и определяет уровень излучения в полосе частот от 1 до 400 кГц. Спецификация MPR II, утвержденная в декабре 1990 года, была распространена и на электрические поля. Нормы MPR II значительно строже, чем MPR I. Об этом можно судить из того, что мониторы, удовлетворяющие MPR II, излучают настолько мало, что не могут оказать никакого вредного воздействия на здоровье.

В настоящее время широко распространены стандарты TCO’95 и TCO’99. Они включают более жесткие экологические и эргономические нормы. В частности по TCO’95 частота вертикальной развертки монитора должна быть не меньше 75 Гц, а по TCO’99 – не меньше 85 Гц.

Жидкокристаллические дисплеи.

Жидкокристаллические дисплеи – Liquid Crystal Display (LCD-дисплеи) – используются очень широко: от наручных часов до компьютера. Портативность, низкое потребление энергии, отсутствие мерцания изображения, прекрасные геометрико-оптические характеристики – все это было достигнуто благодаря технологии тонкопленочных транзисторов TFT в активной матрице. Благодаря этому ЖКД широко используются как высококачественные мониторы различного назначения (рис. 6).

Рис. 6. Устройство LCD-монитора

В процессе развития технология изготовления ЖК-дисплеев испытывала неоднократные принципиальные прорывы, направленные на улучшение качества изображения. В итоге, теперь можно сказать, что активноматричные ЖК-дисплеи обошли по качеству картинки своих «коллег» на вакуумных кинескопах.

Изображение на них формируется с помощью матрицы пикселов, как и в обычных мониторах; отличие же состоит в материале пикселов и в способе генерации излучения. Каждый элемент матрицы – так называемый жидкий кристалл, являющийся оптически активным материалом. Он способен в естественном состоянии поворачивать плоскость поляризации проходящего через него излучения. Второе важное его свойство – это способность изменять угол поворота плоскости поляризации в зависимости от приложенного внешнего электрического поля. Такие характеристики ЖК-ячейки позволяют манипулировать интенсивностью прошедшего света. На практике это делается следующим образом. С обеих сторон от ЖК-ячейки на пути распространения излучения устанавливаются скрещенные поляризаторы. Первый из них выделяет определенную компоненту поляризации падающего излучения. Далее это излучение попадает на жидкий кристалл, который поворачивает плоскость поляризации на определенный угол. Второй поляризатор служит для управления интенсивностью излучения: если его выделенное направление совпадает с направлением плоскости поляризации излучения, то для света он окажется абсолютно прозрачным, а если между ними будет угол 90°, то свет поглотится. Таким образом, можно изменять интенсивность излучения внешним электрическим полем. Однако при помощи подобной схемы можно сконструировать лишь черно-белый монитор. Для создания цветного дисплея необходимо наличие ячеек трех цветов – красного, синего и зеленого. На самом деле все ячейки одинаковые, а цвета генерируются за счет пропускания излучения сквозь светофильтры нужных цветов. Но проблема состоит в том, что отфильтрованное излучение очень сильно теряет в своей интенсивности, а это сказывается на общей яркости, уменьшает глубину контраста и, естественно, качество цветопередачи. В последнее время стал применяться альтернативный подход, основанный на интересном свойстве жидких кристаллов, а именно: для разных длин волн углы поворота плоскости поляризации излучения при одном и том же

внешнем поле отличаются. Реализация этого способа более технологична и сложна, но зато она позволяет достичь большей яркости, лучшей контрастности и в целом улучшить цветопередачу.

Плазменные мониторы. Одной из перспективных разработок плоских дисплеев является плазменная панель, которая используется уже довольно давно, но потребляемая мощность и габаритные размеры дисплеев позволяли использовать их разве что на улице в качестве гигантских рекламных видеощитов. Теперь многие ведущие производители электроники имеют в своем ассортименте качественные плазменные дисплеи для профессионального и бытового применения. По качеству изображения и масштабным характеристикам современные плазменные дисплеи не имеют себе равных (рис. 7).

Рис. 7. Устройство плазменного монитора

Принцип работы плазменной панели состоит в управлении холодным разрядом разреженного газа (ксенона или неона), находящегося в ионизированном состоянии (холодная плазма). Рабочим элементом (пикселом), формирующим отдельную точку изображения, является группа из трех подпикселов, ответственных за три основных цвета соответственно. Каждый подпиксел представляет собой отдельную микрокамеру, на стенках которой находится флюоресцирующее вещество одного из основных цветов. Пикселы находятся в точках пересечения прозрачных управляющих хром-медь-хромовых электродов, образующих прямоугольную сетку.

Для того чтобы «зажечь» пиксел, происходит приблизительно следующее. На два ортогональных друг другу электрода, питающий и управляющий, в точке пересечения которых находится нужный пиксел, подается высокое управляющее переменное напряжение прямоугольной формы. Газ в ячейке отдает большую часть своих валентных электронов и переходит в состояние плазмы. Ионы и электроны попеременно собираются у электродов по разные стороны камеры, в зависимости от фазы управляющего напряжения. Для «поджига» подаются синфазный импульс на сканирующий электрод, одноименные потенциалы складываются, вектор электростатического поля удваивает свою величину. Происходит разряд: часть заряженных ионов отдает энергию в виде излучения квантов света в ультрафиолетовом диапазоне (в зависимости от газа). В свою очередь флюоресцирующее покрытие, находясь в зоне разряда, начинает излучать свет в видимом диапазоне, который и воспринимает наблюдатель. 97% ультрафиолетовой составляющей излучения, вредного для глаз, поглощается наружным стеклом. Яркость свечения люминофора определяется величиной управляющего напряжения.

Достоинством плазменной панели являются следующие свойства. Как и в ЖК-панелях, в плазменных экранах отсутствуют мерцание изображения, несведение; картинка имеет одинаковую высокую четкость по всему рабочему полю; малая толщина панели (не более 6 дюймов), бытовые дисплеи можно вешать на стенку; прекрасная обзорность (под любым углом), высокая контрастность. А главное, что большеразмерные ЖК-панели создать пока трудно, а плазменные дисплеи легко «масштабируются».

Но такие уникальные возможности плазменной панели обусловлены высокой потребляемой мощностью (в десятки раз больше, чем у вакуумных кинескопов, и в сотни, чем у ЖКД). К тому же плазменная панель пока имеет небольшой срок службы (5–10 лет) по сравнению с остальными конкурирующими технологиями.

Видеопамять

Видеопамять – память, которая используется для хранения данных, выводимых на экран.

Глубина цвета. Чем большее количество цветов выбирается для представления изображения, тем большее число ячеек памяти требуется для каждого пиксела в таком изображении. Ниже указано число битов, необходимое для отображения различного количества цветов (или, как иногда говорят, для реализации различной глубины цвета):

Количество цветов	Необходимое число битов
2	1
4	2
16	4
256	8
32768	15
65536	16
16777216	24

При этом нужно осознавать, что современная графическая плата с 64-разрядной шиной за один тактовый цикл может передавать в четыре раза больше данных, чем устаревшая 16-разрядная плата.

Однако за быстродействие графической платы отвечает не только разрядность ее шины. Большую роль играют также тип микросхемы графического акселератора, быстродействие RAMDAC и используемый графический драйвер.

Необходимый объем видеопамяти. Чем больше разрешающая способность монитора, тем больше должна быть видеопамять, так как для каждой точки экрана в видеопамять записывается код ее цвета. Максимальный объем видеопамяти – важный параметр, который, однако, ничего не говорит о быстродействии графической платы. Он определяет лишь наивысшие реализуемые значения разрешения и глубины цвета.

Объем видеопамяти – не единственный критичный параметр. Очень важным параметром является также скорость работы графической платы. В конце концов, какой смысл в

представлении видеофайлов с наивысшим разрешением и максимальной глубиной цвета, если изображение на экране непрерывно подергивается.

Для обеспечения высокого разрешения требуется не только более быстрый ЦП, нужны также быстрые графические процессоры и видеопамять большего объема с меньшим временем доступа. Если несколько лет назад можно было выбирать только между DRAM- и VDRAM-микросхемами, то сегодня производители графических плат развивают и используют все более новые технологии микросхем видеопамяти.

Впереди всех в разработке новых RAM-микросхем для графических плат была фирма Matrox. На графической плате Matrox Millenium, которая все еще считается одной из лучших графических плат, установлены так называемые WRAM-микросхемы (Windows RAM). Имеется несколько различных направлений в разработке новых микросхем RAM для графических плат. Каждый производитель имеет здесь свои пристрастия.

DRAM. Эти микросхемы динамической RAM представляют самый медленный и самый дешевый тип памяти для графических плат. Данные могут или только записываться в память, или только считываться из памяти. Одновременный ввод и вывод невозможен.

VRAM (Video RAM). VRAM-микросхемы, называемые также двухпортовыми RAM-микросхемами, позволяют одновременно записывать и считывать данные. Благодаря этому RAMDAC может непрерывно считывать данные из памяти.

WRAM (Windows RAM). Эти RAM-микросхемы представляют еще один более быстрый вариант VRAM. Они имеют специальные дополнительные графические функции, обеспечивающие представление изображения даже в режиме True Color (т.е. для 16 миллионов цветов) при исключительно высоком разрешении. Запись и считывание могут осуществляться одновременно.

EDO RAM (Extended Data Output RAM). Микросхемы EDO RAM представляют улучшенный вариант DRAM. При

относительно невысоких значениях разрешения и глубины цвета они все же быстрее микросхем DRAM благодаря своей архитектуре. При использовании EDO RAM в каждый момент времени данные могут или только записываться, или только считываться.

SDRAM (Synchronous DRAM). Микросхемы SDRAM быстрее микросхем EDO RAM, но используются довольно редко. Эти дорогие синхронные микросхемы DRAM также не допускают одновременного выполнения записи и считывания данных.

SGRAM (Synchronous Graphics RAM). Более быстродействующая модификация SDRAM обозначается SGRAM. Микросхемы SGRAM имеют в своем распоряжении видеошину удвоенной разрядности, но тем не менее не допускают одновременного выполнения записи и считывания данных. Их преимущество в быстродействии обеспечивается за счет специальных дополнительных графических функций.

RDRAM (Rambus DRAM). Эти микросхемы RAM (Rambus – фирма-разработчик), поставляемые в трех различных исполнениях, настроены на максимальную разрядность видеошины и быстрое отображение данных. Несмотря на все это, речь по-прежнему идет о микросхемах DRAM, которые не допускают одновременного ввода и вывода данных.

MDRAM (Multibank DRAM – многобанковое ОЗУ). Это вариант DRAM, разработанный фирмой MoSys, организован в виде множества независимых банков объемом по 32 Кб каждый, работающих в конвейерном режиме и использующих распараллеливание операций доступа к данным между большим количеством банков памяти RDRAM (RAMBus DRAM). На сегодняшний момент этот тип памяти обеспечивает наивысшую пропускную способность на один чип памяти среди всех остальных типов памяти. Увеличение скорости обращения видеопроцессора к видеопамяти, помимо повышения пропускной способности адаптера, позволяет поднять максимальную частоту регенерации изображения, что снижает утомляемость глаз оператора.

Задание 1: *Определите частоту строк монитора при заданном разрешении, если известны частота регенерации изображения равна и потери на синхронизацию.*

Пример выполнения: Определите частоту строк монитора при разрешении 1024x768 точек, если частота регенерации изображения равна 60 Гц, а потери на синхронизацию составляют 10%.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Рассчитать частоту строк без учета потерь на синхронизацию.	Частота строк без учета потерь на синхронизацию: $768 \cdot 60 = 46080$ Гц
2	Рассчитать частоту строк с учетом потерь на синхронизацию.	Частота строк с учетом потерь на синхронизацию: $46080 + 46080 \cdot 10\% = 50688$ Гц « 50,7 кГц

Решите самостоятельно следующие задания.

Задание 1.1: Определите частоту строк монитора при разрешении 640x480 точек, если частота регенерации изображения равна 80+n Гц, а потери на синхронизацию составляют 10%+n.

Задание 1.2: Определите частоту строк монитора при разрешении 800x600 точек, если частота регенерации изображения равна 75+n Гц, а потери на синхронизацию составляют 10%+n.

Задание 1.3: Определите частоту строк монитора при разрешении 1024x768 точек, если частота регенерации изображения равна 85 +n Гц, а потери на синхронизацию составляют 10%+n.

Задание 1.4: Определите частоту строк монитора при разрешении 1280x1024 точек, если частота регенерации

изображения равна 93 Гц + n, а потери на синхронизацию составляют 10%+ n.

Задание 1.5: Определите частоту строк монитора при разрешении 1280x1024 точек, если частота регенерации изображения равна 85+n Гц, а потери на синхронизацию составляют 10%+n.

Задание 2: *Рассчитайте объем видеопамяти, необходимый для реализации заданного разрешения и количества цветов. Если вариант в журнале чётный, то номера заданий выбирать чётные и наоборот.*

Пример выполнения: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 1024x768 точек и количества цветов 65536.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Рассчитать количество точек на экране.	Количество точек на экране: $1024 \cdot 768 = 786432$
2	Найти число битов, необходимых для представления одного пиксела (по табл. в теории).	Для 65536 цветов необходимо 16 бит = 2В.
3	Рассчитать объем видеопамяти.	Необходимый объем видеопамяти: $786432 \cdot 2$ В = 1572864 В $\approx$1,57 МВ

Решите самостоятельно следующие задания.

Задание 2.1: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 640x480 точек и количества цветов 256.

Задание 2.2: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 800x600 точек и количества цветов 65536.

Задание 2.3: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 1024x768 точек и количества цветов 16777216.

Задание 2.4: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 800x600 точек и количества цветов 256.

Задание 2.5: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 1280x1024 точек и количества цветов 65536.

Задание 2.6: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 640x480 точек и количества цветов 2.

Задание 2.7: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 800x600 точек и количества цветов 4.

Задание 2.8: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 1024x768 точек и количества цветов 32768.

Задание 2.9: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 800x600 точек и количества цветов 65536.

Задание 2.10: Рассчитайте объем видеопамяти, необходимый для реализации разрешения 1280x1024 точек и количества цветов 16777216.

Задание 3: Расшифруйте следующее обозначение характеристик монитора.

Пример выполнения: Расшифруйте следующее обозначение характеристик монитора:

LG 774FT 0,24 17» 1280x1024 60Hz TCO’95.

Решение: Предварительно заполните таблицу, подобрав к алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить производителя и марку.	Фирма: LG Марка: 774FT
2	Определить величину «зерна».	Величина «зерна»: 0,24 мм
3	Определить диагональ экрана.	Диагональ экрана: 17 дюймов
4	Определить разрешение и частоту регенерации.	Разрешение: 1280x1024 точек Частота регенерации: 60 Гц
5	Определить соответствие стандарту.	Соответствует стандарту: TCO’95

Решите самостоятельно следующие задания.

Задание 3.1: Расшифруйте следующее обозначение характеристик монитора:

Sony CPD-E100 0,25 15» 1280x1024 65Hz TCO’99.

Задание 3.2: Расшифруйте следующее обозначение характеристик монитора:

ViewSonic E50 0,28 15» 1024x768 60Hz MPR II.

Задание 3.3: Расшифруйте следующее обозначение характеристик монитора:

Hitachi CM772ET 0,25 19» 1600x1200 85Hz TCO’99.

Задание 3.4: Расшифруйте следующее обозначение характеристик монитора:

NEC FE750 + 0,25 17» 1600x1200 73Hz TCO’99.

Задание 3.5: Расшифруйте следующее обозначение характеристик монитора:

Samsung 775DF 0,28 17» 1600x1200 68Hz TCO’99.

Лабораторная работа № 21-22 «Определение производительности шины данных. Определение адреса памяти».

Цель работы: Научиться определять производительность шины данных, определять адрес памяти.

Теоретическая информация:

Шины

Многие компоненты ПК, к которым, в первую очередь, относятся процессор, память и периферийные устройства, должны быть соединены друг с другом линиями передачи электрических сигналов. В этом и заключается предназначение шины. Она состоит из определенного числа линий (проводов), которые в соответствии с выполняемыми ими функциями называются линиями данных, управления или адресными линиями. Группы линий, выполняющих одинаковые функции, обычно также называют шинами с указанием выполняемой ими функции (шины данных, управления, адресная шина), выделяя их для удобства из полной системной шины.

Системная шина – основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой.

Системная шина состоит из шины данных, адреса и управления.

Основной функцией системной шины является передача информации между микропроцессором и остальными электронными компонентами компьютера. По этой шине осуществляется также адресация устройств и происходит обмен специальными служебными сигналами.

Информация передается по шине в виде групп битов. В состав шины для каждого бита слова может быть предусмотрена отдельная линия (параллельная шина), или все биты слова могут последовательно во времени использовать одну линию (последовательная шина).

Передачей информации по шине управляет процессор или специально выделенный для этого узел, называемый *арбитром шины* (в многопроцессорных системах).

Важнейшими характеристиками шины являются *ширина (разрядность),* т.е. число *линий данных,* и *частота,* которые непосредственно влияют в совокупности на *производительность,* измеряемую в мегабайтах в секунду.

Шина данных. Разрядность шины

Шину данных образуют линии, служащие для передачи данных между отдельными структурными группами ПК. Исходным пунктом линий данных является центральный процессор. Он определяет разрядность шины данных, т.е. число линий, по которым передаются данные. Чем выше разрядность шины данных, тем больший объем данных можно передать по ней за некоторый определенный промежуток времени и тем выше быстродействие компьютера.

В первых ПК использовался процессор Intel 8088. Этот 16-разрядный процессор имел всего лишь 8 внешних линий данных (этим объясняется его низкая стоимость). Для внутренних операций было задействовано 16 линий данных, благодаря чему процессор мог одновременно обрабатывать два восьмиразрядных числа. Но на внешнем уровне к нему присоединялась дешевая восьмиразрядная шина данных. Эти 8 линий обеспечивали связь со всеми микросхемами на системной плате, выполняющими функции обработки данных, и всеми платами расширения, установленными в гнездах. Таким образом осуществлялась передача данных между платами расширения и процессором.

Современные процессоры допускают внешнее подключение большего числа линий данных: процессор 80286 - 16 линий данных, процессоры 80386 DX и 80486 DX - 32 линии, а процессор Pentium - 64 линии данных.

Адресная шина. Разрядность шины

Другая группа линий образует адресную шину. Эта шина используется для адресации. Каждая ячейка памяти и устройство ввода-вывода компьютера имеет свой собственный адрес.

При считывании или записи данных процессор должен сообщать, по какому адресу он желает прочитать или записать данные, для чего необходимо указать этот адрес.

В отличие от шины данных шина адреса является однонаправленной.

Разрядность адресной шины определяет максимальное число адресов, по которым может обратиться процессор, т. е. число линий в адресной шине показывает, каким объемом памяти может управлять процессор. Учитывая, что одна адресная линия обеспечивает представление одного разряда двоичного числа, формулу для максимального объема адресуемой памяти можно записать в следующем виде:

$$\text{максимальное число адресов} = 2^n,$$

где n - разрядность адресной шины.

Процессор 8088 имел 20 адресных линий, что в соответствии с приведенной формулой обеспечивало адресацию памяти объемом:

$$2^{20} = 1\,048\,576 \text{ байт} = 1024 \text{ Кбайт} = 1 \text{ Мбайт}.$$

Это тот самый предельный объем памяти, который все еще имеет силу в операционной системе DOS.

Совсем иная ситуация с процессором 80286. Он имеет 24 адресных линии и поэтому в состоянии управлять памятью объемом:

$$2^{24} = 16\,777\,216 \text{ байт} = 16 \text{ Мбайт}.$$

Для обеспечения связи с микросхемами памяти число адресных линий процессора должно равняться числу адресных линий на системной плате.

Процессоры 80386, 80486 и Pentium имеют 32 адресных линии, что обеспечивает адресацию свыше 4 млрд. ячеек памяти. На системной плате с такими процессорами должно быть 32 линии, обеспечивающие обмен адресами между центральным процессором и всеми важными периферийными микросхемами.

Шина управления

Линии шины управления на системной плате служат для управления различными компонентами ПК. По выполняемой

ими функции их можно сравнить с переводной стрелкой на железнодорожных путях.

С помощью небольшого числа линий шина управления обеспечивает такое функционирование системы, чтобы в каждый данный момент времени только одна структурная единица ПК пересылала данные по шине данных или осуществляла адресацию памяти.

К шине может быть подключено много приемных устройств. Сочетание управляющих и адресных сигналов определяет, для кого именно предназначаются данные на шине. Управляющая логика возбуждает специальные стробирующие сигналы, чтобы указать получателю, когда ему следует принимать данные.

Управляющая логика активизирует в каждый конкретный момент только одно устройство, которое становиться ведущим. Когда устройство активизировано, оно помещает свои данные на шину. Все другие микросхемы в этот промежуток времени должны блокироваться с помощью соответствующего сигнала на линии управления.

Цикл шины

Микропроцессор взаимодействует с внешней средой с помощью шины адреса/данных/состояния и нескольких управляющих сигналов. Собственно взаимодействие заключается в выполнении одной из двух операций: МП либо выводит (записывает) данные, либо вводит (считывает) данные или команды. В каждой из этих операций процессор должен указывать то устройство, с которым он будет взаимодействовать; другими словами, процессор должен адресовать ячейку памяти либо порт ввода или вывода.

Для передачи данных или выборки команды процессор инициирует так называемый *цикл шины*. Кроме процессора, цикл шины могут инициировать и другие устройства, например, арифметический сопроцессор.

Цикл шины представляет собой последовательность событий, в течение которой процессор выдает адрес ячейки памяти или периферийного устройства, а затем формирует

сигнал записи или считывания, а также выдает данные в операции записи. Выбранное устройство воспринимает данные с шины в цикле записи или помещает данные на шину в цикле считывания. По окончании цикла шины устройство фиксирует записываемые данные или снимает считываемые данные.

Системные и локальные шины

В вычислительной системе, состоящей из множества подсистем, необходим механизм для их взаимодействия. Эти подсистемы должны быстро и эффективно обмениваться данными. Например, процессор, с одной стороны, должен быть связан с памятью, с другой стороны, необходима связь процессора с устройствами ввода/вывода. Одним из простейших механизмов, позволяющих организовать взаимодействие различных подсистем, является единственная центральная шина, к которой подсоединяются все подсистемы. Доступ к такой шине разделяется между всеми подсистемами. Подобная организация имеет два основных преимущества: низкая стоимость и универсальность. Поскольку такая шина является единственным местом подсоединения для разных устройств, новые устройства могут быть легко добавлены, и одни и те же периферийные устройства можно даже применять в разных вычислительных системах, использующих однотипную шину. Стоимость такой организации получается достаточно низкой, поскольку для реализации множества путей передачи информации используется единственный набор линий шины, разделяемый множеством устройств.

Главным недостатком организации с единственной шиной является то, что шина создает узкое горло, ограничивая, возможно, максимальную пропускную способность ввода/вывода. Если весь поток ввода/вывода должен проходить через центральную шину, такое ограничение пропускной способности весьма реально. В коммерческих системах, где ввод/вывод осуществляется очень часто, а также в суперкомпьютерах, где необходимые скорости ввода/вывода очень высоки из-за высокой производительности процессора, одним из главных вопросов разработки является создание

системы нескольких шин, способной удовлетворить все запросы.

Одна из причин больших трудностей, возникающих при разработке шин, заключается в том, что максимальная скорость шины главным образом лимитируется физическими факторами: длиной шины и количеством подсоединяемых устройств (и, следовательно, нагрузкой на шину). Эти физические ограничения не позволяют произвольно ускорять шины. Требования быстродействия (малой задержки) системы ввода/вывода и высокой пропускной способности являются противоречивыми. В современных крупных системах используется целый комплекс взаимосвязанных шин, каждая из которых обеспечивает упрощение взаимодействия различных подсистем, высокую пропускную способность, избыточность (для увеличения отказоустойчивости) и эффективность.

Магистраль – совокупность всех шин.

Магистрально-модульный принцип организации ЭВМ – подход к организации ЭВМ, позволяющий без конструктивных изменений материнской платы подключать (отключать) дополнительные устройства.

Магистрально-модульная организация получила широкое распространение, поскольку в этом случае все устройства используют единый подход сопряжения модулей центральных процессоров и устройств ввода/вывода с помощью стандартных шин.

Традиционно шины делятся на шины, обеспечивающие организацию связи процессора с памятью, и шины ввода/вывода. Шины ввода/вывода могут иметь большую протяженность, поддерживать подсоединение многих типов устройств и обычно следуют одному из шинных стандартов. Шины процессор-память, с другой стороны, сравнительно короткие, обычно высокоскоростные и соответствуют организации системы памяти для обеспечения максимальной пропускной способности канала память-процессор. На этапе разработки системы для шины процессор-память заранее известны все типы и параметры устройств, которые должны

соединяться между собой, в то время как разработчик шины ввода/вывода должен иметь дело с устройствами, различающимися по задержке и пропускной способности.

Как уже было отмечено, с целью снижения стоимости некоторые компьютеры имеют единственную шину для памяти и устройств ввода/вывода. Такая шина часто называется системной. Персональные компьютеры, как правило, строятся на основе одной системной шины.

Необходимость сохранения баланса производительности по мере роста быстродействия микропроцессоров привела к двухуровневой организации шин в персональных компьютерах на основе локальной шины.

Локальной шиной называется шина, электрически выходящая непосредственно на контакты микропроцессора. Она обычно объединяет процессор, память, схемы буферизации для системной шины и ее контроллер, а также некоторые вспомогательные схемы. Типичными примерами локальных шин являются VL-Bus и PCI.

На современных компьютерах часто используется специальная магистраль обмена данными с процессором – *локальная видеошина*. AGP (Accelerated Graphics Port - ускоренный графический порт) – скоростная шина для связи с графической картой. Разработана Intel для высокоскоростной графики. Основное преимущество этой шины заключается в скорости.

Разработка шины связана с реализацией ряда дополнительных возможностей. Решение о выборе той или иной возможности зависит от целевых параметров стоимости и производительности. Такими возможностями являются:

- раздельные линии адреса и данных;
- более широкие (имеющие большую разрядность) шины данных;
- режим групповых пересылок (пересылки нескольких слов).

Выбор типа синхронизации определяет, является ли шина синхронной или асинхронной. Если шина синхронная, то она

включает сигналы синхронизации, которые передаются по линиям управления шины, и фиксированный протокол, определяющий расположение сигналов адреса и данных относительно сигналов синхронизации. Поскольку практически никакой дополнительной логики не требуется для того, чтобы решить, что делать в следующий момент времени, эти шины могут быть и быстрыми, и дешевыми. Однако они имеют два главных недостатка. Все на шине должно происходить с одной и той же частотой синхронизации, поэтому из-за проблемы перекоса синхросигналов синхронные шины не могут быть длинными. Обычно шины процессор-память синхронные.

Асинхронная шина, с другой стороны, не тактируется. Вместо этого обычно используется старт-стопный режим передачи и протокол «рукопожатия» (handshaking) между источником и приемником данных на шине. Эта схема позволяет гораздо проще приспособить широкое разнообразие устройств и удлинить шину без беспокойства о перекосе сигналов синхронизации и о системе синхронизации. Если может использоваться синхронная шина, то она обычно быстрее, чем асинхронная, из-за отсутствия накладных расходов на синхронизацию шины для каждой транзакции. Выбор типа шины (синхронной или асинхронной) определяет не только пропускную способность, но также непосредственно влияет на емкость системы ввода/вывода в терминах физического расстояния и количества устройств, которые могут быть подсоединены к шине. Асинхронные шины по мере изменения технологии лучше масштабируются. Шины ввода/вывода обычно асинхронные.

Стандарты шин

Обычно количество и типы устройств ввода/вывода в вычислительных системах не фиксируются, что позволяет пользователю самому подобрать необходимую конфигурацию. Шина ввода/вывода компьютера может рассматриваться как шина расширения, обеспечивающая постепенное наращивание устройств ввода/вывода. Поэтому стандарты играют огромную роль, позволяя разработчикам компьютеров и устройств

ввода/вывода работать независимо. Появление стандартов определяется разными обстоятельствами.

Иногда широкое распространение и популярность конкретных машин становятся причиной того, что их шина ввода/вывода становится стандартом де-факто. Примерами таких шин могут служить PDP-11 Unibus и IBM PC/AT Bus. Иногда стандарты появляются также в результате определенных достижений по стандартизации в некотором секторе рынка устройств ввода/вывода. Интеллектуальный периферийный интерфейс (IPI - Intelligent Peripheral Interface) и Ethernet являются примерами стандартов, появившихся в результате кооперации производителей. Успех того или иного стандарта в значительной степени определяется его принятием такими организациями как ANSI (Национальный институт по стандартизации США) или IEEE (Институт инженеров по электротехнике и радиоэлектронике). Иногда стандарт шины может быть прямо разработан одним из комитетов по стандартизации: примером такого стандарта шины является FutureBus.

Одной из популярных шин персональных компьютеров была системная шина *IBM PC/XT*, обеспечивавшая передачу 8 бит данных. Кроме того, эта шина включала 20 адресных линий, которые ограничивали адресное пространство пределом в 1 Мбайт. Для работы с внешними устройствами в этой шине были предусмотрены также 4 линии аппаратных прерываний (IRQ) и 4 линии для требования внешними устройствами прямого доступа к памяти (DMA). Для подключения плат расширения использовались специальные 62-контактные разъемы. При этом системная шина и микропроцессор синхронизировались от одного тактового генератора с частотой 4.77 МГц. Таким образом, теоретическая скорость передачи данных могла достигать немногим более 4 Мбайт/с.

Системная шина *ISA* (Industry Standard Architecture) впервые стала применяться в персональных компьютерах IBM PC/AT на базе процессора 80286. Для этой системной шины характерно наличие второго, 36-контактного дополнительного разъема для соответствующих плат расширения. За счет этого

количество адресных линий было увеличено на 4, а данных - на 8, что позволило передавать параллельно 16 бит данных и обращаться к 16 Мбайт системной памяти. Количество линий аппаратных прерываний в этой шине было увеличено до 15, а каналов прямого доступа - до 7. Системная шина ISA полностью включала в себя возможности старой 8-разрядной шины. Шина ISA позволяет синхронизировать работу процессора и шины с разными тактовыми частотами. Она работает на частоте 8 МГц, что соответствует максимальной скорости передачи 16 Мбайт/с.

С появлением процессоров 80386, 80486 и Pentium шина ISA стала узким местом персональных компьютеров на их основе. Новая системная шина *EISA* (Extended Industry Standard Architecture), появившаяся в конце 1988 года, обеспечивает адресное пространство в 4 Гбайта, 32-битовую передачу данных (в том числе и в режиме DMA), улучшенную систему прерываний и арбитраж DMA, автоматическую конфигурацию системы и плат расширения. Устройства шины ISA могут работать на шине EISA. Шина EISA предусматривает централизованное управление доступом к шине за счет наличия специального устройства - арбитра шины. Поэтому к ней может подключаться несколько главных устройств шины. Улучшенная система прерываний позволяет подключать к каждой физической линии запроса на прерывание несколько устройств, что снимает проблему количества линий прерывания. Шина EISA тактируется частотой около 8 МГц и имеет максимальную теоретическую скорость передачи данных 33 Мбайт/с.

Шина *MCA* также обеспечивает 32-разрядную передачу данных, тактируется частотой 10 МГц, имеет средства автоматического конфигурирования и арбитража запросов. В отличие от EISA она не совместима с шиной ISA и используется только в компьютерах компании IBM.

Шина *VL-bus*, предложенная ассоциацией VESA (Video Electronics Standard Association), предназначалась для увеличения быстродействия видеоадаптеров и контроллеров дисковых накопителей для того, чтобы они могли работать с

тактовой частотой до 40 МГц. Шина VL-bus имеет 32 линии данных и позволяет подключать до трех периферийных устройств, в качестве которых наряду с видеоадаптерами и дисковыми контроллерами могут выступать и сетевые адаптеры. Максимальная скорость передачи данных по шине VL-bus может составлять около 130 Мбайт/с.

Шина PCI (Peripheral Component Interconnect) также, как и шина VL-bus, поддерживает 32-битовый канал передачи данных между процессором и периферийными устройствами, работает на тактовой частоте 33 МГц и имеет максимальную пропускную способность 120 Мбайт/с. При работе с процессорами 80486 шина PCI дает примерно те же показатели производительности, что и шина VL-bus. Однако, в отличие от последней, шина PCI является процессорно независимой (шина VL-bus подключается непосредственно к процессору 80486 и только к нему). Ее легко подключить к различным центральным процессорам. В их числе Pentium, Alpha, R4400 и PowerPC.

Основные возможности шины PCI:

- синхронный протокол обмена, предусматривающий подтверждение о приеме, что позволяет пересылать данные со скоростью, максимально возможной для приемного устройства (в отличие от ISA);
- в PC используется 32-разрядная шина PCI с частотой 33 МГц. Однако шина PCI процессорно независимая, и спецификация допускает ширину в 64 разряда и частоту 66 МГц. Это используется, например, в компьютерах компании Sun, в серверах на базе ЦП Alpha и Intel Xeon. Слот 32-разрядной шины PCI имеет 124 контакта (у 64-разрядной есть еще дополнительная секция);
- вставляемые платы автоматически конфигурируются (при старте) по стандарту Plug and Play, т.е. не надо вручную устанавливать перемычки и переключатели, чтобы избежать конфликтов с другими устройствами;
- для пользователя шина предоставляет слоты (не более пяти). Это так называемый сегмент PCI;

- спецификация PCI допускает произвольное число сегментов, образующих древовидную топологию. Для соединения сегментов используются PCI-мосты;
- есть режим Bits mastering, дающий устройству возможность брать управление шиной на себя и тем самым разгрузить ЦП;
- начиная с версии 2.2 (внедрена в 1999 г.) в шину включена линия, управляющая пробуждением компьютера от PCI-устройства, например, модема, сетевой карты.

PCI Special Interest Group в октябре 1999 г. объявила о выходе спецификации PCI-X 1.0. Спецификация представляет собой окончательный вариант преемницы PCI-шины. Основным достоинством новой шины является увеличенная до 133 МГц частота при сохранившейся обратной совместимости с PCI-картами.

Одной из популярных шин ввода-вывода является шина *SCSI*.

Под термином SCSI - Small Computer System Interface (Интерфейс малых вычислительных систем) обычно понимается набор стандартов, разработанных Национальным институтом стандартов США (ANSI) и определяющих механизм реализации магистрали передачи данных между системной шиной компьютера и периферийными устройствами.

Первоначально SCSI предназначался для использования в небольших дешевых системах и поэтому был ориентирован на достижение хороших результатов при низкой стоимости. Характерной его чертой является простота, особенно в части обеспечения гибкости конфигурирования периферийных устройств без изменения организации основного процессора. Главной особенностью подсистемы SCSI является размещение в периферийном оборудовании интеллектуального контроллера.

Для достижения требуемого высокого уровня независимости от типов периферийных устройств в операционной системе основной машины, устройства SCSI представляются имеющими очень простую архитектуру. Например, геометрия дискового накопителя представляется в

виде линейной последовательности одинаковых блоков, хотя в действительности любой диск имеет более сложную многомерную геометрию, содержащую поверхности, цилиндры, дорожки, характеристики плотности, таблицу дефектных блоков и множество других деталей. В этом случае само устройство или его контроллер несут ответственность за преобразование упрощенной SCSI модели в данные для реального устройства.

Начальный стандарт 1986 года, известный теперь под названием SCSI-1, определял рабочие спецификации протокола шины, набор команд и электрические параметры. Шина данных SCSI-1 использовала 50-жильный экранированный кабель, имела разрядность 8 бит, а максимальная скорость передачи составляла 5 Мбайт/сек.

В 1992 году этот стандарт был пересмотрен с целью устранения недостатков первоначальной спецификации (особенно в части синхронного режима передачи данных) и добавления новых возможностей повышения производительности, таких как «быстрый режим» (fast mode), «широкий режим» (wide mode) и помеченные очереди. Этот пересмотренный стандарт получил название SCSI-2 и в настоящее время используется большинством поставщиков вычислительных систем.

Задание 1: Определить производительность шины данных с заданной разрядностью и частотой.

Пример выполнения: Определить производительность шины данных с разрядностью n=8 бит и частотой f=100 МГц.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Записать формулу для определения производительности шины.	Производительность шины: $P = (n /8) \cdot f$

2	Рассчитать производительность шины.	P = (8/8) байт · (100 · 10^6) Гц = 10^8 байт/с = 100 Мбайт/с

Выполните самостоятельно следующие задания, где N-номер варианта:

Задание 1.1: Определить производительность шины данных с разрядностью n=16+N бит и частотой f=150+N МГц.

Задание 1.2: Определить производительность шины данных с разрядностью n=8+N бит и частотой f=33+N МГц.

Задание 1.3: Определить производительность шины данных с разрядностью n=12+N бит и частотой f=90+N МГц.

Задание 1.4: Определить производительность шины данных с разрядностью n=32+N бит и частотой f=200 +N МГц.

Задание 1.5: Определить производительность шины данных с разрядностью n=8+N бит и частотой f=66+N МГц.

Задание 2: Определить адрес памяти, заданный как в микропроцессоре 8086.

Пример: Определить адрес памяти, заданный как 0001:00F0h в микропроцессоре 8086.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Умножить значение сегмента на 16.	Значение сегмента 0001h умножаем на 16, получаем 00010h.
2	Сложить полученное значение со смещением.	Полученное значение складываем со смещением: 00010h + 000F0h = 00100h

Выполните самостоятельно следующие задания, где к каждому значению адреса прибавить номер варианта, переведенного в 16-ую систему счисления:

Задание 2.1: Определить адрес памяти, заданный как 0011:03F0h в микропроцессоре 8086.

Задание 2.2: Определить адрес памяти, заданный как 0002:00E0h в микропроцессоре 8086.

Задание 2.3: Определить адрес памяти, заданный как 04A1:0077h в микропроцессоре 8086.

Задание 2.4: Определить адрес памяти, заданный как 1D00:1080h в микропроцессоре 8086.

Задание 2.5: Определить адрес памяти, заданный как B000:35F7h в микропроцессоре 8086.

Задание 2.6: Определить адрес памяти, заданный как 0B11:06A0h в микропроцессоре 8086.

Задание 2.7: Определить адрес памяти, заданный как 0002:00E0h в микропроцессоре 8086.

Задание 2.8: Определить адрес памяти, заданный как 11A1:CA77h в микропроцессоре 8086.

Задание 2.9: Определить адрес памяти, заданный как D1A0:1880h в микропроцессоре 8086.

Задание 2.10: Определить адрес памяти, заданный как ABC0:3511h в микропроцессоре 8086.

Лабораторная работа № 23-24 «Нахождение кода Хэмминга для кодовой комбинации. Определение время доступа к памяти при разных условиях»

Цель работы: Научиться находить код Хэмминга для различных кодовых комбинаций.

Теоретическая информация:

ЗАЩИТА ИНФОРМАЦИИ И ПАМЯТИ

Для повышения достоверности информации, хранимой на различных уровнях иерархии памяти, применяются дублирование, избыточное кодирование и ограничение доступа.

На практике часто используют комбинированное сочетание этих способов.

Дублирование информации заключается в следующем: вся информация (или отдельные ячейки) записывается несколько раз, обычно нечетное число раз (минимум три раза). Информация считывается специальным устройством и сравнивается. Суждение о правильности передачи выносится по совпадению большинства из считанной информации методами «два из трех», «три из пяти» и так далее. Например, кодовая комбинация 01101 при трехразовой записи была частично искажена, поэтому были считаны следующие комбинации: 10101, 01110, 01001. В результате проверки каждой позиции отдельно правильной считается комбинация 01101.

Среди методов защиты от ошибок наибольшее распространение получило *избыточное кодирование*, позволяющее получить более высокие качественные показатели работы устройств памяти. Его основное назначение - принятие всех возможных мер для того, чтобы вероятность искажений информации была достаточно малой, несмотря на наличие сбоев в работе ЭВМ. Широко применяется метод контроля циклическим избыточным кодом.

Контроль циклическим избыточным кодом – метод, использующий вычисление контрольной суммы и служащий для определения ошибок в передаваемых данных.

Помехоустойчивое кодирование предполагает разработку *корректирующих (помехоустойчивых) кодов*, обнаруживающих и исправляющих определенного рода ошибки, а также построение и реализацию кодирующих и декодирующих устройств и программных механизмов.

В помехоустойчивых кодах, кроме информационных элементов, всегда содержится один или несколько дополнительных элементов, являющихся *проверочными*. Наличие в кодах избыточной информации позволяет обнаруживать и исправлять (или только обнаруживать) ошибки.

Одной из простейших форм проверки ошибок является метод *контроля на четность*. Его суть заключается в том, что к

каждой кодовой комбинации добавляется один разряд, в который записывается единица, если число единиц в кодовой комбинации нечетное, или ноль, если четное. При декодировании подсчитывается количество единиц в кодовой комбинации. Если оно оказывается четным, то поступившая информация считается правильной, если нет, то ошибочной.

Кроме проверки по горизонтали, контроль на четность и нечетность может проводиться и по вертикали.

Преимущества контроля на четность заключаются в минимальном значении коэффициента избыточности К (для пятиэлементного кода К=0,17) и в простоте его технической реализации, а недостаток – в том, что обнаруживаются ошибки, имеющие только нечетную кратность.

Однако такая методика проверки не может обнаружить ошибки в случае двойного переброса (например, две единицы перебросились в ноль), что может привести к высокому уровню ошибок.

Двойная проверка на четность/нечетность является усовершенствованием одинарной проверки. В этой методике вместо бита четности в каждом символе определяется четность или нечетность целого блока символов. Проверка блока позволяет обнаруживать ошибки как внутри символа, так и между символами. Эта проверка называется также двумерным кодом проверки на четность. Она имеет значительное преимущество по сравнению с одинарной. С помощью такой перекрестной проверки может быть существенно улучшена надежность работы устройства хранения информации.

Еще одной формой проверки ошибок служит подсчет *контрольных сумм*. Это несложный способ, который обычно применяется вместе с проверкой на четность/нечетность. Сущность его состоит в суммировании численных значений всех ячеек блока памяти. Шестнадцать младших разрядов суммы помещаются в 16-разрядный счетчик контрольной суммы, который вместе с информацией пользователей записывается в память. При считывании выполняются такие же вычисления и сравнивается полученная контрольная сумма с

записанной. Если эти суммы совпадают, подразумевается, что блок без ошибок. При этом имеется незначительная вероятность того, что в результате такой проверки ошибочный блок может быть не обнаружен.

Код Хэмминга позволяет не только обнаруживать, но и исправлять ошибки.

Код Хэмминга - метод определения и исправления ошибок при передаче данных, использующий проверочные биты и проверочную сумму.

В этом коде каждая кодовая комбинация состоит из m информационных и k контрольных элементов, так, например, в семиэлементном коде Хэмминга n=7, m=4, k=3 (для всех остальных элементов существует специальная таблица). Контрольные символы 0 или 1 записываются в первый, второй и четвертый элементы кодовой комбинации, причем в первый элемент - в соответствии с контролем на четность для третьего, пятого и седьмого элементов, во второй - для третьего, шестого и седьмого элементов, и в четвертый - для пятого, шестого и седьмого элементов. В соответствии с этим правилом комбинация 1001 будет представляться в коде Хэмминга как 0011001, и в этом виде она будет записываться в ячейку памяти.

При декодировании в начале проверяются на четность первый, третий, пятый и седьмой элементы, результат проверки записывается в первый элемент контрольного числа. Далее контролируется четвертый – седьмой элементы – результат проставляется в младшем элементе контрольного числа. При правильно выполненной передаче контрольное число состоит из одних нулей, а при неправильной - из комбинации нулей и единиц, соответствующей при чтении ее справа налево номеру элемента, содержащего ошибку.

Для устранения этой ошибки необходимо изменить находящийся в этом элементе символ на обратный.

Код Хэмминга имеет существенный недостаток: при обнаружении любого числа ошибок он исправляет лишь одиночные ошибки.

Избыточность семиэлементного кода Хэмминга равна 0,43. При увеличении значности кодовых комбинаций увеличивается число проверок, но уменьшается избыточность кода.

На одном процессоре в защищенном режиме могут параллельно исполняться несколько задач. При этом использование памяти управляется механизмами сегментации и трансляции страниц. Защита памяти каждой выполняемой программы осуществляется с помощью ограничения доступа по чтению-записи. Попытки выполнения недопустимых команд, выхода за рамки отведенного пространства памяти и разрешенной области ввода/вывода контролируются системой защиты. Процессор предоставляет необходимые аппаратные средства поддержки защиты памяти, а их реальное использование зависит от операционной системы.

При страничной организации памяти для обеспечения целостности, секретности и взаимной изоляции выполняющихся программ предусматриваются различные режимы доступа к страницам, которые реализуются с помощью специальных индикаторов доступа в элементах таблиц страниц.

При сегментной организации памяти для защиты сегментов вводятся различные режимы доступа к сегментам. Из программных сегментов допускается только выборка команд и чтение констант. Запись в программные сегменты может рассматриваться как незаконная и запрещаться системой. Выборка команд из сегментов данных также может считаться незаконной и любой сегмент данных может быть защищен от обращений по записи или по чтению. При сегментной организации памяти наличие базово-граничных пар в дескрипторе таблицы дескрипторов и элементах таблицы дескрипторов предотвращает возможность обращения программы пользователя к таблицам дескрипторов и страниц, с которыми она не связана.

МНОГОУРОВНЕВАЯ ОРГАНИЗАЦИЯ ПАМЯТИ

Обычно памятью машины называют оперативное запоминающее устройство ОЗУ. ОЗУ используется для записи

программ, а также исходных данных, промежуточных и конечных результатов.

Внешние запоминающие устройства (ВЗУ) обладают практически неограниченным объемом памяти и наименьшим быстродействием.

ОЗУ не сохраняет информацию при отключении питания. Существуют ПЗУ, которые сохраняют информацию при отключении питания. ПЗУ работают только в режиме чтения, а ОЗУ в режиме чтения и записи. Существуют перепрограммируемые ПЗУ (ППЗУ), которые сохраняют информацию при отключении питания и допускают запись информации.

Характеристики устройств памяти

Важнейшими характеристиками отдельных устройств памяти являются емкость памяти, удельная емкость, время доступа.

Емкость памяти определяется максимальным количеством данных, которые могут в ней храниться. Емкость измеряется в двоичных единицах (битах), машинных словах, но большей частью в байтах.

Время доступа к памяти – время, которое занимает установка адреса на адресной шине и считывание данных с шины данных.

В некоторых устройствах памяти считывание информации сопровождается ее разрушением (стиранием). В таком случае цикл обращения должен содержать операцию восстановления (регенерации) считанной информации на прежнем месте в памяти.

Таким образом, время доступа к памяти при считывании определяется по формуле

$$t_{обр\_с} = t_{дост\_с} + t_{счит} + t_{рег},$$

где $t_{дост\_с}$ - промежуток времени между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации;

$t_{счит}$ - продолжительность самого физического процесса считывания;

$t_{рег}$ - время, затрачиваемое на регенерацию информации (равно нулю для ЗУ, которым регенерация не требуется).

Время доступа при записи определяется по формуле

$$t_{обр\_з} = t_{дост\_з} + t_{подг} + t_{зап},$$

где $t_{дост\_з}$ - промежуток времени между моментом начала операции записи и моментом, когда становится возможным доступ к запоминающим элементам;

$t_{подг}$ - время подготовки, расходуемое на приведение в исходное состояние запоминающих элементов для записи заданной единицы информации;

$t_{зап}$ - время занесения информации.

В качестве продолжительности цикла обращения к памяти принимается величина

$$t_{обр} = \max(t_{обр\_с}, t_{обр\_з}).$$

Задание 1: Определить код Хэмминга для заданной кодовой комбинации.

Пример выполнения: Определить код Хэмминга для кодовой комбинации 1001.

Решение

Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Занести информационные биты в соответствующие элементы выходного кода.	Код Хэмминга для 4-разрядной кодовой комбинации имеет вид: $P = p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7$ **Информационные биты заносятся в 3, 5, 6, 7 элементы:** $P = p_1\ p_2\ 1\ p_4\ 0\ 0\ 1$, где p_1, p_2, p_4 – контрольные символы.
2	Рассчитать первый контрольный символ.	**Первый контрольный символ:** $p_1 = \mathrm{mod}_2(p_3, p_5, p_7) = \mathrm{mod}_2(1, 0, 1) = 0$

3	Рассчитать второй контрольный символ.	Второй контрольный символ: $p_2 = \text{mod}_2 (p_3, p_6, p_7) = \text{mod}_2 (1, 0, 1) = 0$
4	Рассчитать третий контрольный символ.	**Третий контрольный символ:** $p_4 = \text{mod}_2 (p_5, p_6, p_7) = \text{mod}_2 (0, 0, 1) = 1$
5	Сформировать кодовую комбинацию.	P = 0 0 1 1 0 0 1

Выполните самостоятельно следующие задания, где к каждому значению кодовой комбинации прибавить номер варианта, переведенного в 2-ую систему счисления:

Задание 1.1: Определить код Хэмминга для кодовой комбинации 1101.

Задание 1.2: Определить код Хэмминга для кодовой комбинации 0110.

Задание 1.3: Определить код Хэмминга для кодовой комбинации 0001.

Задание 1.4: Определить код Хэмминга для кодовой комбинации 1100.

Задание 1.5: Определить код Хэмминга для кодовой комбинации 0111.

Задание 1.6: Определить код Хэмминга для кодовой комбинации 1111.

Задание 1.7: Определить код Хэмминга для кодовой комбинации 0101.

Задание 1.8: Определить код Хэмминга для кодовой комбинации 0001.

Задание 1.9: Определить код Хэмминга для кодовой комбинации 1000.

Задание 1.10: Определить код Хэмминга для кодовой комбинации 0110.

Задание 2: Определить время доступа к памяти при считывании при заданных значениях.

Пример: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 45 нс, время считывания $t_{счит}$ = 100 нс, время регенерации $t_{рег}$ = 5 нс.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Записать формулу для определения времени доступа к памяти при считывании.	Время доступа к памяти при считывании: $t_{обр\_с} = t_{дост\_с} + t_{счит} + t_{рег}$
2	Рассчитать время доступа.	$t_{обр\_с}$ = 45 нс + 100 нс + 5 нс = 150 нс

Выполните самостоятельно следующие задания, где к каждому значению прибавить номер варианта:

Задание 2.1: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 25 нс, время считывания $t_{счит}$ = 80 нс, время регенерации $t_{рег}$ = 2 нс.

Задание 2.2: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 15 нс, время считывания $t_{счит}$ = 60 нс, время регенерации $t_{рег}$ = 3 нс.

Задание 2.3: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 60 нс, время считывания $t_{счит}$ = 150 нс, время регенерации $t_{рег}$ = 8 нс.

Задание 2.4: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 35 нс, время считывания $t_{счит}$ = 100 нс, время регенерации $t_{рег}$ = 6 нс.

Задание 2.5: Определить время доступа к памяти при считывании, если время между моментом начала операции считывания и моментом, когда становится возможным доступ к данной единице информации $t_{дост\_с}$ = 10 нс, время считывания $t_{счит}$ = 20 нс, время регенерации $t_{рег}$ = 4 нс.

Лабораторная работа № 25-26 «Принципы работы кэш-памяти. Нахождение объема кэшируемой памяти при архитектуре прямого отображения. Представление кода символа сигнальными уровнями ТТЛ для передачи по интерфейсу RS-232C.»

Цель работы: Научиться определять производительность шины данных, определять адрес памяти.

Теоретическая информация:

Кэширование памяти

Производительность ПК зависит не только от процессора. Большой потенциал производительности ПК связан с организацией и способом функционирования памяти. Важную роль при этом играет кэш-память.

Кэш-память выполняет функцию буфера между процессором и оперативной памятью.

Буфер – область памяти, используемая для временного хранения информации.

Данные, которые процессор уже получил из оперативной памяти, остаются в быстрой кэш-памяти, несмотря на то, что они уже обработаны. Подразумевается, что при обмене данными и при выполнении большого числа операций процессор будет часто запрашивать одни и те же данные и команды. При обращении микропроцессора к памяти сначала производится поиск нужных данных в кэш-памяти.

При отсутствии кэш-памяти данные каждый раз считываются из относительно медленной оперативной памяти компьютера. Процессор прерывает свою текущую работу и ждет, пока будут получены необходимые данные для дальнейшей обработки.

При наличии кэш-памяти данные находятся в специально предназначенной для процессора исключительно быстрой памяти, и при их запросе циклы ожидания отсутствуют.

При управлении кэш-памятью речь идет не о простой памяти, организованной по принципу стека, которая теряет старые данные, если в нее считываются новые. С помощью специальных алгоритмов, наиболее простым из которых является алгоритм LRU (Least Recently Used – алгоритм удаления наиболее давно использовавшихся данных), чаще используемые данные хранятся в этой памяти дольше, чем другие. Благодаря этому необходимость доступа к основной памяти сводится к минимуму, и компьютер в целом работает быстрей.

В современных компьютерах кэш обычно строится по двухуровневой схеме. *Первичный кэш (LI Cache)* встроен во все процессоры класса 486 и выше; это внутренний кэш. Объем этого кэша невелик (8-32 Кбайт). Чтобы повысить производительность, для данных и команд часто используется раздельный кэш (так называемая Гарвардская архитектура – противоположность Принстонской, использующей общую память для команд и данных). *Вторичный кэш (L2 Cache)* для процессоров 486 и Pentium является внешним (устанавливается на системной плате), а у P6 и Pentium 4 располагается в одной упаковке с ядром и подключается к специальной внутренней шине процессора, благодаря чему обеспечивается работа на полной тактовой частоте процессора.

Кэш-контроллер должен обеспечивать когерентность (coherency) – согласованность данных кэш-памяти обоих уровней с данными в основной памяти при том условии, что обращение к этим данным может производиться не только процессором, но и другими активными (bus master) адаптерами,

подключенными к шинам (PCI, VLB, ISA и т. д.). Следует также учесть, что процессоров может быть несколько, и у каждого может быть свой внутренний кэш.

Кэш-память первого уровня, интегрированная внутри процессора, работает на полной внутренней тактовой частоте процессора, кэш-память второго уровня обычно работает на внешней тактовой частоте процессора.

Архитектура современных 32-разрядных процессоров включает ряд средств кэширования памяти: два уровня кэша инструкций и данных (LI Cache и L2 Cache), буферы ассоциативной трансляции (TLB) блока страничной переадресации и буферы записи. Эти средства в разных вариациях (на кристалле, картридже процессора или на системной плате) представлены в системах с процессорами 486, Pentium, P6 и Pentium 4.

Процессор Pentium III имеет L2-кэш объемом 256 Кбайт, который работает на тактовой частоте, равной тактовой частоте ядра. У процессора Pentium 4 L2-кэш имеет объем 256 Кбайт, работает на тактовой частота ядра, обмен данными с ядром процессора идет по 256-битной шине.

Принципы кэширования

Основная память компьютеров реализуется на относительно медленной динамической памяти (DRAM), обращение к ней приводит к простою процессора – появляются такты ожидания (wait states). Статическая память (SRAM), построенная, как и процессор, на триггерных ячейках, по своей природе способна догнать современные процессоры по быстродействию и сделать ненужными такты ожидания (или хотя бы сократить их количество). Разумным компромиссом для построения экономичных и производительных систем явился иерархический способ организации оперативной памяти. Идея заключается в сочетании основной памяти большого объема на DRAM с относительно небольшой кэш-памятью на быстродействующих микросхемах SRAM.

Кэш является дополнительным быстродействующим хранилищем копий блоков информации из основной памяти,

вероятность обращения к которым в ближайшее время велика. Кэш не может хранить копию всей основной памяти, поскольку его объем во много раз меньше основной памяти. Он хранит лишь ограниченное количество блоков данных и каталог (cache directory) – список их текущего соответствия областям основной памяти. Кроме того, кэшироваться может не вся память, доступная процессору.

При каждом обращении к памяти контроллер кэш-памяти по каталогу проверяет, есть ли действительная копия затребованных данных в кэше. Если она там есть, то это случай *кэш-попадания* (cache hit), и данные берутся из кэш-памяти. Если действительной копии там нет, это случай *кэш-промаха* (cache miss), и данные берутся из основной памяти. В соответствии с алгоритмом кэширования блок данных, считанный из основной памяти, при определенных условиях заместит один из блоков кэша. От интеллектуальности алгоритма замещения зависит процент попаданий и, следовательно, эффективность кэширования. Поиск блока в списке должен производиться достаточно быстро, чтобы «задумчивостью» в принятии решения не свести «на нет» выигрыш от применения быстродействующей памяти. Обращение к основной памяти может начинаться одновременно с поиском в каталоге, а в случае попадания – прерываться (архитектура Look aside). Это экономит время, но лишние обращения к основной памяти ведут к увеличению энергопотребления. Другой вариант: обращение к внешней памяти начинается только после фиксации промаха (архитектура Look Through), при этом теряется, по крайней мере, один такт процессора, зато экономится энергия.

Контроллер кэша оперирует *строками* (cache line) фиксированной длины. Строка может хранить копию блока основной памяти, размер которого, естественно, совпадает с длиной строки. С каждой строкой кэша связана информация об адресе скопированного в нее блока основной памяти и ее состоянии. Строка может быть действительной (valid) – это означает, что в текущий момент времени она достоверно

отражает соответствующий блок основной памяти – или недействительной.

Информация о том, какой именно блок занимает данную строку (т.е. старшая часть адреса или номер страницы), и о ее состоянии называется *тегом* (tag) и хранится в связанной с данной строкой ячейке специальной *памяти тегов* (tag RAM). В операциях обмена с основной памятью обычно строка участвует целиком (несекторированный кэш), для процессоров 486 и выше длина строки совпадает с объемом данных, передаваемых за один пакетный цикл (для 486 – это 4x4=16 байт, для Pentium – 4x8=32 байт). Возможен и вариант секторированного (sectored) кэша, при котором одна строка содержит несколько смежных ячеек – секторов, размер которых соответствует минимальной порции обмена данных кэша с основной памятью. При этом в записи каталога, соответствующей каждой строке, должны храниться биты действительности для каждого сектора данной строки. Секторирование позволяет экономить память, необходимую для хранения каталога при увеличении объема кэша, поскольку большее количество бит каталога отводится под тег, и выгоднее использовать дополнительные биты действительности, чем увеличивать глубину индекса (количество элементов) каталога.

Строки кэша под отображение блока памяти выделяются при промахах операций чтения, в P6 строки заполняются и при записи. Запись блока, не имеющего копии в кэше, производится в основную память (для повышения быстродействия запись может производиться через буфер отложенной записи). Поведение кэш-контроллера при операции записи в память, когда копия затребованной области находится в некоторой строке кэша, определяется его алгоритмом, или *политикой записи* (Write Policy). Существует две основных политики записи данных из кэша в основную память: *сквозная запись WT* (Write Through) и *обратная запись WB* (Write Back).

Политика WT предусматривает одновременное выполнение каждой операции записи (даже однобайтной), попадающей в кэшированный блок, в строку кэша и в основную

память. При этом процессору при каждой операции записи придется выполнять относительно длительную запись в основную память. Алгоритм достаточно прост в реализации и легко обеспечивает целостность данных за счет постоянного совпадения копий данных в кэше и основной памяти. Для него не нужно хранить признаки присутствия и модифицированности – вполне достаточно только информации тега (при этом считается, что любая строка всегда отражает какой-либо блок, а какой именно – указывает тег). Но эта простота оборачивается низкой эффективностью записи. Существуют варианты этого алгоритма с применением отложенной буферизованной записи, при которой данные в основную память переписываются через FIFO-буфер во время свободных тактов шины.

Политика WB позволяет уменьшить количество операций записи на шине основной памяти. Если блок памяти, в который должна производиться запись, отображен в кэше, то физическая запись сначала будет произведена в эту действительную строку кэша, которая отмечается как грязная (dirty), или модифицированная, т.е. требующая выгрузки в основную память. Только после этой выгрузки (записи в основную память) строка станет чистой (clean), и ее можно будет использовать для кэширования других блоков без потери целостности данных. В основную память данные переписываются только целой строкой. Эта выгрузка контроллером может откладываться до наступления крайней необходимости (обращение к кэшированной памяти другим абонентом, замещение в кэше новыми данными) или выполняться в свободное время после модификации всей строки. Данный алгоритм сложнее в реализации, но существенно эффективнее, чем WT. Поддержка системной платой кэширования с обратной записью требует обработки дополнительных интерфейсных сигналов для выгрузки модифицированных строк в основную память, если к этой области производится обращение со стороны таких

контроллеров шины, как другие процессоры, графические адаптеры, контроллеры дисков, сетевые адаптеры и т. п.

В зависимости от способа определения взаимного соответствия строки кэша и области основной памяти различают три архитектуры кэш-памяти: *кэш прямого отображения* (direct-mapped cache), *полностью ассоциативный кэш* (fully associative cache) и их комбинация – *частично или наборно-ассоциативный кэш* (set-associative cache).

Кэш прямого отображения. В *кэш-памяти прямого отображения* адрес памяти, по которому происходит обращение, однозначно определяет строку кэша, в которой может находиться требуемый блок. Принцип работы такого кэша поясним на примере несекторированного кэша объемом 256 Кбайт с размером строки 32 байта и объемом кэшируемой основной памяти 64 Мбайт – типичный кэш системной платы для Pentium. Структуру памяти в такой системе иллюстрирует рис. 5.

Кэшируемая основная память условно разбивается на страницы (в данном случае по 256 Кбайт), размер которых совпадает с размером кэш-памяти (256 Кбайт). Кэш-память (и, условно, страницы основной памяти) делятся на строки (256 Кбайт/32= 8 Кбайт строк). Архитектура прямого отображения подразумевает, что каждая строка кэша может отображать из любой страницы кэшируемой памяти только соответствующую ей строку (на рис. 5 они находятся на одном горизонтальном уровне).

Поскольку объем основной памяти много больше объема кэша, на каждую строку кэша может претендовать множество блоков памяти с одинаковой младшей частью адреса (смещением внутри страницы). Одна строка в определенный момент может, естественно, содержать копию только одного из этих блоков. Номер (адрес) строки в кэш-памяти называется *индексом* (index). Тег несет информацию о том, какой именно блок занимает данную строку (то есть старшая часть адреса или номер страницы).

Рис. 5. Кэш прямого отображения

Память тегов должна иметь количество ячеек, равное количеству строк кэша, а ее разрядность должна быть достаточной, чтобы вместить старшие биты адреса кэшируемой памяти, не попавшие на шину адреса кэш-памяти. Кроме адресной части тега, с каждой строкой кэша связаны биты признаков действительности и модифицированности данных. В начале каждого обращения к кэшируемой памяти контроллер вначале считывает ячейку каталога с заданным индексом, сравнивает биты адреса тега со старшими битами адреса памяти и анализирует признак действительности. Этот анализ выполняется в специальном цикле слежения (snoop cycle), иногда его называют *циклом запроса* (inquire). Если в результате анализа выясняется, что требуемого блока нет в кэше, генерируется (или продолжается) цикл обращения к ос-

новной памяти (случай кэш-промаха). В случае попадания запрос обслуживается кэш-памятью. В случае промаха после считывания основной памяти приемником информации новые данные помещаются в строку кэша (если она чистая), а в ее тег помещаются старшие биты адреса и устанавливается признак действительности данных. Независимо от объема затребованных данных в кэш из основной памяти строка переписывается вся целиком (поскольку признак действительности относится ко всем ее байтам). Если контроллер кэша реализует упреждающее считывание (read ahead), то в последующие свободные циклы шины также обновится и следующая строка (если она была чистой). Чтение «про запас» позволяет при необходимости осуществлять пакетный цикл чтения из кэша через границу строки.

Такой кэш имеет самую простую аппаратную реализацию и применяется во вторичном кэше большинства системных плат. Однако ему присущ серьезный недостаток, вполне очевидный при рассмотрении рис. 5. Если в процессе выполнения программы процессору поочередно будут требоваться блоки памяти, смещенные относительно друг друга на величину, кратную размеру страницы (на рисунке эти блоки расположены на одной горизонтали в разных страницах), то кэш будет работать интенсивно, но вхолостую (cache trashing). Очередное обращение будет замещать данные, считанные в предыдущем и необходимые в следующем обращении, – т.е. будет сплошная череда кэш-промахов. Переключение страниц в многозадачных ОС также снижает количество кэш-попаданий, что отражается на производительности системы. Увеличение размера кэша при сохранении архитектуры прямого отображения даст не очень существенный эффект, поскольку разные задачи будут претендовать на одни и те же строки кэша. Не увеличивая объема, можно повысить эффективность кэширования изменением структуры кэша, о чем пойдет речь ниже.

Объем кэшируемой памяти (M_{cached}) при архитектуре прямого отображения определяется объемом кэш-памяти (V_{cache}) и разрядностью памяти тегов (N):

$$M_{cached} = V_{cache} \cdot 2^N,$$

в нашем случае $M_{cached} = 256$ Кбайт $\cdot 2^8 = 64$ Мбайт.

Иногда в описании кэша прямого отображения фигурирует понятие набор (set), что может ввести в заблуждение. Оно применяется вместо термина *строка* (line) в секторированном кэше прямого отображения, а сектор тогда называют *строкой*. С набором (как и строкой несекторированного кэша) связана информация о теге, относящаяся ко всем элементам набора (строкам или секторам). Кроме того, каждый элемент набора (строка или сектор) имеет собственный бит действительности в кэш-каталоге (рис. 8).

Рис. 8. Секторированный кэш прямого отображения

Наборно-ассоциативный кэш. *Наборно-ассоциативная архитектура* кэша позволяет каждому блоку кэшируемой памяти претендовать на одну из нескольких строк кэша, объединенных в *набор* (set). Можно считать, что в этой архитектуре есть несколько параллельно и согласованно работающих каналов прямого отображения, где контроллеру кэша приходится принимать решение о том, в какую из строк набора помещать очередной блок данных.

В простейшем случае каждый блок памяти может помещаться в одну из двух строк (Two Way Set-Associative Cache – двухканальный наборно-ассоциативный кэш). Такой кэш должен содержать два банка памяти тегов (рис. 9).

Рис. 9. Двухканальный наборно-ассоциативный кэш

Номер набора (индекс), в котором может отображаться затребованный блок данных, однозначно определяется средней частью адреса (как номер строки в кэше прямого отображения). Строка набора, отображающая требуемый блок, определяется сравнением тегов (как и в ассоциативном кэше), параллельно выполняемым для всех каналов кэша. Кроме того, с каждым набором должен быть связан признак, определяющий строку набора, подлежащую замещению новым блоком данных в случае кэш-промаха. Кандидатом на замещение обычно выбирается строка, к которой дольше всего не обращались

(алгоритм LRU– Least Recently Used). При относительно большом количестве каналов (строк в наборе) прибегают к некоторому упрощению – алгоритм Pseudo-LRU для четырех строк (Four Way Set Associative Cache) позволяет принимать решения, используя всего 3 бита. Возможно также применение алгоритма замещения FIFO (первым вошел – первым вышел) или даже случайного (random) замещения, что проще, но менее эффективно.

Наборно-ассоциативная архитектура широко применяется для первичного кэша современных процессоров. Объем кэшируемой памяти определяется так же, как и в предыдущем варианте, но здесь фигурируют объем одного банка (а не всего кэша) и разрядность относящихся к нему ячеек тега.

Ассоциативный кэш. В отличие от предыдущих у *полностью ассоциативного кэша* любая его строка может отображать любой блок памяти, что существенно повышает эффективность использования его ограниченного объема. При этом все биты адреса кэшированного блока, за вычетом битов, определяющих положение (смещение) данных в строке, хранятся в памяти тегов. В такой архитектуре для определения наличия затребованных данных в кэш-памяти требуется сравнение со старшей частью адреса тегов всех строк, а не одной или нескольких, как при прямом отображении или наборно-ассоциативной архитектуре. Естественно, последовательный перебор ячеек памяти тегов отпадает – на это может уйти слишком много времени. Остается параллельный анализ всех ячеек, что является сложной аппаратной задачей, которая пока решена только для небольших объемов первичного кэша в некоторых процессорах. Применение полностью ассоциативной архитектуры во вторичном кэше пока не предвидится.

Система ввода-вывода

Устройства ввода и вывода, такие как клавиатура, мышь, монитор, принтер, жесткий диск, накопитель на гибких магнитных дисках, CD-ROM, модем и другие, принадлежат к

стандартной комплектации системы, и их использование является само собой разумеющимся.

Все периферийные устройства ввода должны коммутироваться с PC таким образом, чтобы данные, вводимые пользователем, могли не только корректно поступать в компьютер, но и в дальнейшем эффективно обрабатываться. В случае вывода информация, поступающая на монитор, принтер или другие внешние периферийные устройства, должна быть предварительно обработана таким образом, чтобы соответствовать спецификации соответствующих выходных устройств.

Передача информации с периферийного устройства (ПУ) в ядро ЭВМ называется *операцией ввода*, а передача из ядра ЭВМ в периферийное устройство - *операцией вывода*.

Связь устройств ЭВМ друг с другом осуществляется с помощью средств сопряжения - интерфейсов.

Интерфейс представляет собой совокупность физических средств сопряжения (коннекторов, разъемов), линий и шин, сигналов, электронных схем и алгоритмов, предназначенную для осуществления обмена информацией между устройствами. От характеристик интерфейсов во многом зависят производительность и надежность вычислительной машины.

Время ответа – характеристика системы ввода-вывода – является одним из самых главных показателей производительности ЭВМ. Время ответа определяется временем, прошедшим с момента подачи запроса на ввод-вывод до момента начала передачи данных.

При разработке систем ввода-вывода должны быть решены следующие проблемы:

1. Должна быть обеспечена возможность реализации машин с переменным составом оборудования.

2. Для эффективного использования оборудования ЭВМ должны реализовываться параллельная во времени работа процессора над программой и выполнение периферийными устройствами процедур ввода-вывода.

3. Необходимо стандартизировать программирование операций ввода-вывода для обеспечения их независимости от особенностей периферийного устройства.

4. Необходимо обеспечить автоматическое распознавание и реакцию ядра ЭВМ на многообразие ситуаций, возникающих в ПУ (готовность устройства, различные неисправности и т.п.).

При конструировании ЭВМ широко применяются различные средства унификации.

Средства вычислительной техники проектируются на основе модульного принципа, который заключается в том, что отдельные устройства выполняются в виде конструктивно законченных модулей, из которых можно собирать ЭВМ в различных конфигурациях.

При обмене между ПУ и ЭВМ используются унифицированные форматы данных. Преобразование унифицированных форматов данных в индивидуальные, приспособленные для отдельных ПУ, производится в самих ПУ. Унификации также подвергают все компоненты интерфейса, а также формат и набор команд процессора для операций ввода-вывода. Унификация распространяется на семейство моделей ЭВМ.

Для обеспечения параллельной работы процессора и периферийных устройств схемы управления вводом-выводом отделяют от процессора. Выполнение общих функций возлагают на общие для групп периферийного оборудования унифицированные устройства - контроллеры прямого доступа к памяти, процессоры ввода-вывода.

BIOS (Basic Input Output System) - базовая система ввода-вывода. Содержит набор основных функций управления стандартными внешними устройствами PC.

Назначение ROM BIOS состоит в выполнении наиболее простых и универсальных услуг, связанных с осуществлением ввода-вывода.

Обозначение ROM расшифровывается как Read Only Memory (память только для чтения, ПЗУ), т.е. информация может только считываться, но не может быть записана. На

самом деле, все современные ROM BIOS позволяют в режиме обновления BIOS стирать и переписывать (обновлять) информацию BIOS.

ROM BIOS выполняет три основные функции:

- предоставляет операционной системе аппаратные драйверы и осуществляет сопряжение между материнской платой и остальным PC;
- содержит тест проверки системы, так называемый POST (Power On Self Test), который проверяет при включении PC все важнейшие компоненты;
- содержит программу установки параметров BIOS и аппаратной конфигурации PC – CMOS Setup.

Изменения конфигурации (например, информация о новом винчестере) записываются в специальную область памяти (и оттуда считываются ROM BIOS), называемую CMOS RAM. Эта область памяти (емкостью 100–129 байт) расположена в контроллере периферии 82C206. Для того чтобы записанные значения не были потеряны, контроллер обеспечивается питанием аккумуляторной батареи. Таким образом, информация о конфигурации PC остается в памяти, даже если долго не включать компьютер. Эта аккумуляторная батарея обеспечивает хранение установок CMOS Setup и работу системного таймера.

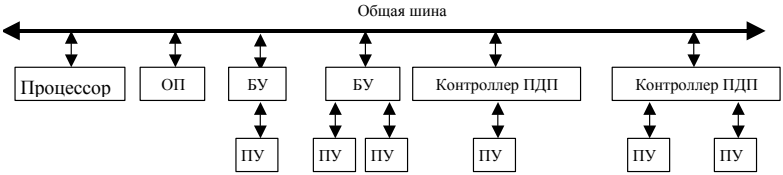
Таймер – устройство, служащее для отсчета текущего времени.

Принципы организации обменов данными

Можно выделить два характерных принципа построения систем ввода-вывода: ЭВМ с одним общим интерфейсом и ЭВМ с множеством интерфейсов и процессорами (каналами) ввода-вывода.

Структура с одним общим интерфейсом. Структура *с одним общим интерфейсом* предполагает наличие общей шины, к которой подсоединяются все модули, в совокупности образующие ЭВМ: процессор, оперативная и постоянная память и периферийные устройства. В каждый данный момент через общую шину может происходить обмен данными только между

одной парой присоединенных к ней модулей. Таким образом, модули ЭВМ разделяют во времени один общий интерфейс, причем процессор выступает как один из модулей системы (рис. 10).



БУ – блок управления

Рис. 10. Структура с одним общим интерфейсом

Периферийные устройства подсоединяются к общей шине с помощью блоков управления периферийными устройствами (контроллеров), осуществляющих согласование форматов данных периферийных устройств с форматом, принятым для передачи по общей шине.

Если в периферийном устройстве операции ввода-вывода производятся для отдельных байтов или слов, то используется программно-управляемая передача данных через процессор и под его управлением. Конструкция контроллера при этом сильно упрощается.

Для периферийных устройств с поблочной передачей данных (ЗУ на дисках, лентах и др.) применяется прямой доступ к памяти (ПДП) и контроллеры ПДП.

При общем интерфейсе аппаратура управления вводом-выводом рассредоточена по отдельным модулям ЭВМ. Процессор при этом не полностью освобождается от управления операциями ввода-вывода. Более того, на все время операции передачи данных интерфейс оказывается занятым, а связь процессора с памятью блокированной.

Структура с каналами ввода-вывода. Структура системы *с процессорами (каналами) ввода-вывода* применяется в высокопроизводительных ЭВМ. В таких ЭВМ система ввода-вывода строится путем централизации аппаратуры управления вводом-выводом на основе применения программно-

управляемых процессоров (каналов) ввода-вывода. Обмен информацией между памятью и периферийным устройством осуществляется через канал ввода-вывода (рис. 11).

Каналы ввода-вывода полностью освобождают процессор от управления операциями ввода-вывода.

В вычислительной машине с каналами ввода-вывода форматы передаваемых данных неоднородны, поэтому необходимо использовать в ЭВМ несколько специализированных интерфейсов.

Можно выделить четыре типа интерфейсов: интерфейс основной памяти, интерфейс процессор-каналы, интерфейсы ввода-вывода, интерфейсы периферийных устройств (малые интерфейсы).

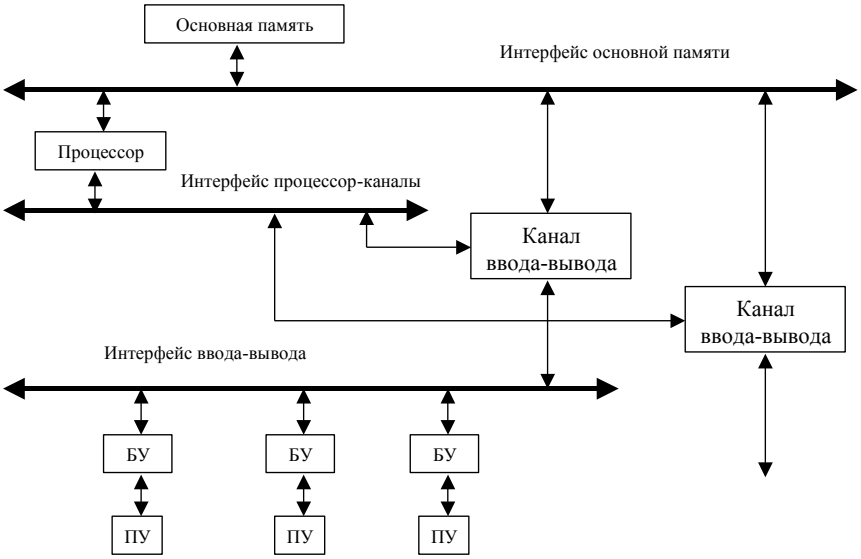


Рис. 11. Структура с каналами ввода-вывода

Через интерфейс основной памяти производится обмен информацией между памятью, с одной стороны, и процессором и каналами – с другой.

Интерфейс процессор-каналы предназначается для передачи информации между процессорами и каналами ввода-вывода.

Через интерфейс ввода-вывода происходит обмен информацией между каналами и блоками управления периферийных устройств.

Основные параметры интерфейсов. Интерфейсы характеризуются следующими параметрами:

1. Пропускная способность интерфейса - это количество информации, которое может быть передано через интерфейс в единицу времени (имеет диапазон от десятков байт до сотен мегабайт).

2. Максимальная частота передачи информационных сигналов через интерфейс лежит в диапазоне от десятков герц до тысяч мегагерц.

3. Максимально допустимое расстояние между соединяемыми устройствами имеет диапазон от десятков сантиметров до нескольких километров при использовании оптоволоконных линий.

4. Динамические параметры интерфейса – время передачи отдельного слова и блока данных с учетом продолжительности процедур подготовки и завершения передачи. Эти параметры особенно важны для систем реального времени.

5. Общее число линий (проводов) в интерфейсе.

6. Информационная ширина интерфейса - число бит данных, передаваемых параллельно через интерфейс. Различные интерфейсы имеют ширину 1, 8, 16, 32, 64, 128 или 256 бит.

7. Связность интерфейса: интерфейс может быть односвязным, когда существует лишь единственный путь передачи информации между парой устройств машины, и многосвязным, позволяющим устройствам обмениваться информацией по нескольким независимым путям. Многосвязность интерфейсов требует дополнительной аппаратуры, но повышает надежность и живучесть вычислительной машины, обеспечивает возможность

автоматической реконфигурации вычислительного комплекса при выходе из строя отдельных устройств.

Параллельная и последовательная передача данных. *Параллельная* и *последовательная* передачи данных хотя и служат одной цели - обмену данными и связи между периферией (устройствами ввода/вывода) и модулем обработки данных (материнской платой), но используют различные методы и принципы обмена информацией.

Параллельная связь означает, что биты передаются все одновременно (параллельно). При этом здесь принципиальным становится понятие разрядности шины.

В отличие от последовательной передачи данных параллельная передача, как правило, однонаправленная, т.е. данные передаются только в одном направлении.

В отличие от параллельной передачи данных отдельные биты пересылаются (или принимаются) последовательно друг за другом, при этом возможен обмен данными в двух направлениях. Уровень напряжения последовательного интерфейса изменяется в пределах от –25 В до +25 В. Благодаря этому относительно высокому значению напряжения повышается помехоустойчивость, и данные могут передаваться без потерь по кабелю длиной 50 м и более.

Поскольку данные обычно представлены на шине микропроцессора в параллельной форме (байтами, словами), их последовательный ввод-вывод оказывается несколько сложным. Для последовательного ввода потребуются средства преобразования последовательных входных данных в параллельные данные, которые можно поместить на шину. С другой стороны, для последовательного вывода необходимы средства преобразования параллельных данных, представленных на шине, в последовательные выходные данные. В первом случае преобразование осуществляется регистром сдвига с последовательным входом и параллельным выходом, а во втором – регистром сдвига с параллельным входом и последовательным выходом.

Методы передачи информации между устройствами ЭВМ. Последовательные данные передаются в *синхронном* или *асинхронном* режимах.

При *синхронном* методе передающее устройство устанавливает одно из двух возможных состояний сигнала (0 или 1) и поддерживает его в течение строго определенного интервала времени, по истечении которого состояние сигнала на передающей стороне может быть изменено.

Время передачи сигнала, которое складывается из времени передачи сигнала по линии и времени распознавания и фиксации сигнала в регистре приемного устройства, зависит от параметров линии связи и характеристик приемного и передающего устройств. Период синхронной передачи информации должен превышать максимальное время передачи сигнала. Он задается специальными тактовыми импульсами, как правило, поступающими от тактового генератора с кварцевым резонатором.

При *асинхронной* передаче передающее устройство устанавливает соответствующее передаваемому коду состояние сигнала на информационной линии, а принимающее устройство после приема сигнала информирует об этом передающее устройство изменением состояния сигнала на линии подтверждения приема. Передающее устройство, получив сигнал подтверждения, снимает передаваемый сигнал.

Конечно, второй режим сложнее, но у него есть серьезное преимущество: не нужен отдельный сигнал синхронизации.

Существуют специальные микросхемы ввода и вывода, решающие проблемы преобразования, описанные выше.

Обычно при передаче сигналов на короткие расстояния (десятки сантиметров) более быстрым оказывается синхронный метод, а при передаче на большие расстояния - асинхронный.

При передаче параллельного кода по параллельным линиям сигналы поступят в приемное устройство в разное время из-за разброса параметров цепей, формирующих сигналы, и линий интерфейса (так называемая проблема состязаний).

Используется два метода передачи параллельного кода по нескольким линиям: *со стробированием*, применяющим синхронную передачу, и *с квитированием*, в котором используется асинхронная передача.

При передаче *со стробированием* кроме N информационных линий используется линия «готовность данных»: вначале устанавливаются значения передаваемых сигналов на информационных линиях, затем на линии готовности устанавливается уровень 1. Через строго определенный период времени (превышающий максимальное время передачи) сигнал готовности сбрасывается в 0, процесс передачи завершается, после чего можно изменить сигналы на информационных линиях и передавать следующую порцию данных.

При передаче *с квитированием* кроме N информационных линий и линии готовности данных используется линия подтверждения приема: вначале устанавливаются значения передаваемых сигналов на информационных линиях, затем на линии готовности устанавливается уровень 1. Приняв фронт сигнала готовности, приемное устройство считывает сигналы с информационных линий и посылает передатчику сигнал подтверждения приема. Приняв фронт сигнала подтверждения, передатчик снимает сигнал готовности, после чего может приступать к передаче новой порции данных.

Порты ввода-вывода. Подсоединение периферийных устройств, таких как манипулятор типа мышь, внешний модем или принтер, к персональному компьютеру производится через так называемые *устройства сопряжения*, или *адаптеры*, на которых реализованы стандартные или специальные интерфейсы. До недавнего времени подобные адаптеры были выполнены в виде отдельных плат ввода-вывода – Input-Output (I/O) Card, вставляемых в разъемы расширения на системной плате. Современные системные платы, как правило, интегрируют все необходимые адаптеры.

Итак, взаимодействие периферийного устройства с адаптером происходит через один (возможно, один из двух)

интерфейс, определяющий, в частности, тип и «род» (розетка или вилка, female или male) соединителя, уровни и длительность электрических сигналов, протоколы обмена.

На практике стандартные последовательный и параллельный интерфейсы часто называют *портами ввода-вывода*.

Порт – электронная схема, использующаяся для передачи сигналов на другие устройства.

Собственно, до последнего времени в качестве последовательного стандартного интерфейса используется разновидность *RS-232C* (Recommended Standard), а в качестве параллельного – *Centronics*.

Порт называют *последовательным*, когда информационные биты передаются последовательно один за другим и *параллельным*, когда несколько бит данных передаются одновременно. Если несколько адаптеров (последовательного и параллельного портов, приводов флоппи- и жестких дисков) конструктивно выполнены на отдельной плате, она называется, как правило, многофункциональной платой ввода-вывода (Multi I/O Card).

Через порт процессор получает данные с устройств ввода и посылает данные на устройства вывода. В большинстве случаев к параллельному интерфейсу подключается принтер. Однако имеются еще и другие периферийные устройства, управление которыми осуществляется через этот интерфейс, в ряде случаев это плоттер или сканер. Иногда это внешние дисководы, но чаще внешние стриммеры. При этом возможна передача данных с максимальной скоростью 1 Мбайт/с. Также параллельные интерфейсы используются для обмена информацией между двумя PC.

Параллельный порт. Поскольку *параллельный порт* в IBM PC-совместимом компьютере чаще всего используется для подключения принтера, то его называют также *принтер-портом*. В MS-DOS компьютер работает максимум с тремя параллельными портами, которые имеют логические имена LPT1, LPT2 и LPT3. В адресном пространстве компьютера резервируются базовые адреса этих портов: 3BCh, 378h и 278h.

Первый адрес обычно используется, если принтер-порт находится, например, на плате графического адаптера Hercules или EGA. На плате Multi I/O Card адрес LPT1 – 378h, а LPT2 – 278h. Для принтерного порта LPT1 предусмотрено аппаратное прерывание IRQ7, а для LPT2 – IRQ5, хотя на практике они обычно не применяются. Установка базовых адресов портов и возможность использования прерываний настраиваются перестановкой перемычек (jumpers) на плате, описание которых приводится в технической документации для конкретного адаптера или в программе начальной установки SETUP.

Сначала интерфейс Centronics был конструктивно реализован на нескольких ТТЛ-микросхемах. Именно на них в этом случае выполняются декодирование адреса, промежуточное хранение и инвертирование отдельных сигналов. Затем широкое распространение получили адаптеры, в которых практически все функции отдельных ТТЛ-микросхем объединены в одной БИС типа 82С11, выполненной по КМОП-технологии. Теперь все микросхемы портов и адаптеров «спрятаны» обычно в одной СБИС.

Начиная с базового адреса, каждый порт принтера имеет в адресном пространстве три адреса. При этом первый адрес соответствует регистру данных, посылаемых от компьютера к принтеру. Чтение установленных битов данных можно осуществить по тому же адресу. Физически чтение данных происходит через специальный буфер данных.

Следующий адрес (базовый плюс единица) позволяет читать регистр статуса адаптера (расположенный в принтере) через буферную микросхему. Регистр статуса позволяет определить состояние некоторых сигналов интерфейса Centronics.Чтение регистра статуса имеет смысл при передаче данных на принтер для определения состояния принтера и процесса передачи данных.

Адрес третьего порта (базовый адрес плюс 2) соответствует регистру управления интерфейса. Этот регистр (read only – только для чтения) позволяет определить следующие состояния принтера:

- бит 0 = 0: сигнал Strobe активен (описание см. ниже),
- бит 1 = 0: сигнал Auto feed включен (описание см. ниже),
- бит 2 = 0: инициализация принтера,
- бит 3 = 1: принтер выбран,
- бит 4 = 1: прерывание разрешено.

Распределение сигналов. При минимальной конфигурации для реализации параллельного интерфейса было бы достаточно 11 проводов: один провод массы (корпус), два так называемых провода подтверждения (Handshake) и восемь проводов для передачи данных. Поскольку параллельный интерфейс часто отождествляют с разъемом кабеля принтера, приведем назначение выводов соответствующего 36-контактного разъема Centronics.

Данные (линия 2-9 и 18-25). Восемь линий данных передают восемь битов информации. Для каждой линии данных имеется отдельный проводник заземления (18-25)

Подтверждение (линия 1,10,11). При передаче данных обе стороны должны сообщать друг другу информацию о своем состоянии. Это осуществляется путем передачи по определенной линии сигнала низкого или высокого уровня (напряжения 0 В или 5 В).

Сигнал Strobe сообщает принтеру о том, что PC установил байт данных на линии данных и принтер может печатать символ.

Сигнал Busy сообщает PC, что принтер занят обработкой данных, которые накопились в его буфере. После обработки байта данных принтер сообщает, что обработка прошла корректно и он готов к приему новой информации.

Контроль принтера (линии 12-17). Оставшиеся проводники кабеля принтера предназначены для передачи контрольных сигналов о состоянии принтера и конфликтах в его работе, например об ошибках при передаче данных.

Paper End. По линии 12 передается сообщение PC о том, что в принтере нет бумаги. Интерфейс реагирует на это, как на срыв передачи данных, и сигнализирует об этом по другим линиям (Select и Error). В противном случае принтер печатал бы

без бумаги прямо по валику, что могло бы привести к серьезному повреждению головки принтера.

Select и Select Input. По линии 13 интерфейсу сообщается о том, в каком из состояний находится принтер, в состоянии On-line (готов) или Off-line (выключен или не готов). Уровень сигнала на линии 13 можно изменять вручную с помощью соответствующих переключателей на панели управления принтера. Сигнал Select Input, передаваемый по линии 17 или 36, устанавливает принтер в режим готовности к работе (On-line), например, после устранения какой-либо ошибки в его работе.

Error. Все ошибки, возникающие во время передачи данных, сообщаются по линии 15 или 32. Состояние линии Error влияет на состояние других линий и может остановить процесс печати.

Часто проявляющаяся ошибка при работе принтера – это так называемая ошибка времени выполнения (Time Out). Если принтер долгое время занят «изнурительной» работой с данными и не может сообщить сигналом Busy о том, что он больше не в состоянии принимать данные, то спустя некоторое время фиксируется ошибка Time Out. Регистрация этой ошибки влияет на все линии. При отсутствии сигнала об ошибке Time Out происходила бы непрерывная передача данных на принтер до тех пор, пока бесконечный цикл не привел бы к зависанию всей системы.

При инициализации (линия Init – 16 или 31) принтер переходит в свое исходное состояние. Для матричного принтера это означает, что его печатающая головка возвращается в исходное состояние. Кроме того, очищается буфер принтера, т. е. данные, обрабатываемые принтером, выгружаются из его памяти.

По линии Auto Feed (автоматический перевод строки, линия 14) передается сигнал, указывающий принтеру, как обрабатывать новую строку при поступлении команды возврата каретки.

Последовательный порт. *Последовательный порт* используется для подключения большинства периферийных

устройств, таких как плоттер, принтер, мышь, внешний модем, программатор ПЗУ и т.д. До настоящего времени для последовательной связи IBM PC-совместимых компьютеров используются адаптеры с интерфейсом RS-232C. Описание этого интерфейса было опубликовано Американской промышленной ассоциацией еще в 1969 году. Европейским аналогом RS-232 являются два стандарта, разработанных Международным союзом электросвязи (International Telecommunications Union, ITU): V.24 (механические характеристики) и V.28 (электрические характеристики). Хотя первоначально RS-232 был предназначен для связи центральной машины с терминалами, его простота и богатые возможности обеспечили ему более широкое применение. В современном IBM PC-совместимом компьютере, работающем под MS-DOS, может использоваться до четырех последовательных портов, имеющих логические имена соответственно COM1, COM2, COM3 и COM4.

Выбор устройств, подключаемых к последовательному порту, значительно шире, чем к параллельному, поэтому большинство PC одновременно оборудовано двумя интерфейсными разъемами для последовательной передачи данных. Обычно они различаются по внешнему виду. Разъемы последовательного интерфейса на PC имеют 9 и 25 контактов.

Главный элемент последовательного интерфейса – микросхема 8250 для старых и 16450 UART (Universal Asynchron Receiver Transmitter) для новых плат контроллеров. Контроллер на базе чипа 8250 обеспечивает максимальную скорость передачи данных 9600 бод, а чип 16450 – 115200 бод.

В адресном пространстве IBM PC-совместимых компьютеров последовательный адаптер занимает восемь последовательных адресов, включая базовый. Однако с помощью определенного «трюка» через эти восемь адресов происходит обращение к 11 регистрам, которые программируются соответствующим образом.

Адреса и прерывания последовательных портов

Обозначение	Диапазон адресов	Прерывание
COM1	3F8h-3FFh	IRQ4
COM2	2F8h-2FFh	IRQ3
COM3	3E8h-3EFh	IRQ10 (IRQ2)
COM4	2E8h-2EFh	IRQ11 (IRQ5)

Конфигурация. Последовательный интерфейс связывает два устройства. Для того чтобы «собеседники» при обмене данными не перебивали друг друга, они должны иметь единый протокол приема-передачи, которым определяется последовательность обмена данными.

Скорость передачи данных.

Номинальная скорость передачи – это скорость передачи данных, определяемая количеством элементов двоичной информации, передаваемых за 1 секунду.

Эффективная (реальная) скорость – скорость передачи с учетом необходимости передачи служебной информации (что уменьшает эффективную скорость по сравнению с номинальной) и сжатия данных (что увеличивает эффективную скорость).

Скорость передачи измеряется в *бодах.* Иногда вместо *бод* употребляют обозначение *bps* (bit per second) или бит/с.

Следует обратить внимание на то, что при передаче данных, например между двумя модемами, один из которых является высокоскоростным модемом со скоростью 19200 бод, а другой в состоянии обеспечить обмен данными лишь со скоростью 2400 бод, компьютеры все равно будут общаться со скоростью 2400 бод.

Стартовый бит, стоп-бит, биты данных

Данные при последовательной передаче разделяются между собой служебными посылками, такими как *стартовый бит* (Start bit) и *стоп бит* (Stop bit). Эти биты указывают на начало и конец передачи последовательности битов данных (Data bits). Данный метод передачи позволяет осуществить синхронизацию между приемной и передающей сторонами, а также выровнять скорость обмена данными.

Бит контроля четности

Для идентификации и распознавания ошибок при последовательной передаче в состав байта данных дополнительно включают *бит контроля четности* (Parity bit). Существует несколько различных вариантов использования бита контроля четности:

- бит контроля четности не посылается (No Parity),
- бит контроля четности четный (Even Parity);
- бит контроля четности нечетный (Odd Parity).

Значение бита контроля четности определяется бинарной суммой всех передаваемых битов данных.

Для установления связи между двумя последовательными интерфейсами предварительно необходимо сконфигурировать их соответствующим образом, т.е. указать, каким образом будет осуществляться обмен данными: скорость обмена, формат данных, контроль четности и т. п.

Обычно на практике параметры указывают только в краткой формулировке. Например, запись 8E1 обозначает протокол обмена данными с числом бит 8, битов контроля четности четный и один стоп-бит. Запись 7N2 указывает на 7 бит данных, без контроля четности и два стоп-бита.

Общие сведения об интерфейсе RS–232C. Интерфейс RS–232C является наиболее широко распространенной стандартной последовательной связью между микрокомпьютерами и периферийными устройствами. Интерфейс, определенный стандартом Ассоциации электронной промышленности (EIA), подразумевает наличие оборудования двух видов: терминального DTE и связного DCE.

Сигналы интерфейса RS–232C подразделяются на следующие классы:

Последовательные данные RXD - принимаемые данные (входные последовательные данные); TXD - передаваемые данные (выходные последовательные данные)). Интерфейс RS–232C обеспечивает два независимых последовательных канала данных: первичный (главный) и вторичный (вспомогательный).

Оба канала могут работать в дуплексном режиме, т.е. одновременно осуществляют передачу и прием информации.

Управляющие сигналы квитирования (CTS - сброс передачи. На этой линии периферийное устройство формирует сигнал низкого уровня, когда оно готово воспринимать информацию от процессора; RTS - запрос передачи. На эту линию микропроцессорная система выдает сигнал низкого уровня, когда она намерена передавать данные в периферийное устройство). Сигналы квитирования – средство, с помощью которого обмен сигналами позволяет DTE начать диалог с DCE до фактической передачи или приема данных по последовательной линии связи.

Сигналы синхронизации (TC, RC). В синхронном режиме (в отличие от более распространенного асинхронного) между устройствами необходимо передавать сигналы синхронизации, которые упрощают синхронизм принимаемого сигнала в целях его декодирования.

На практике вспомогательный канал RS–232C применяется редко, и в асинхронном режиме вместо 25 линий используется 9 линий.

Виды сигналов

В большинстве схем, содержащих интерфейс RS–232C, данные передаются асинхронно, т.е. в виде последовательности пакета данных. Каждый пакет содержит один символ кода ASCII, причем информация в пакете достаточна для его декодирования без отдельного сигнала синхронизации.

Символы кода ASCII представляются семью битами, например буква **А** имеет код 1000001. Чтобы передать букву **А** по интерфейсу RS–232C, необходимо ввести дополнительные биты, обозначающие начало и конец пакета. Кроме того, желательно добавить лишний бит для простого контроля ошибок по паритету (четности).

Наиболее широко распространен формат, включающий в себя один стартовый бит, один бит паритета и два стоповых бита. Начало пакета данных всегда отмечает низкий уровень стартового бита. После него следует 7 бит данных символа кода

ASCII. Бит четности содержит 1 или 0 так, чтобы общее число единиц в 8–битной группе было нечетным. Последним передаются два стоповых бита, представленных высоким уровнем напряжения. Эквивалентный ТТЛ–сигнал при передаче буквы **А** показан на рис. 12.

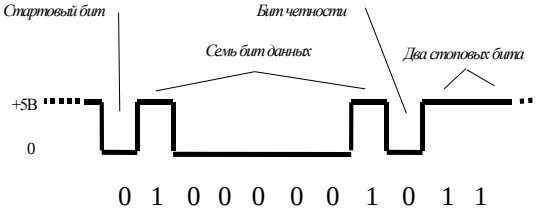


Рис. 12. Представление кода буквы А сигнальными уровнями ТТЛ

Таким образом, полное асинхронно передаваемое слово состоит из 11 бит (фактически данные содержат только 7 бит) и записывается в виде 01000001011.

Используемые в интерфейсе RS–232C уровни сигналов отличаются от уровней сигналов, действующих в компьютере. Логический 0 (SPACE) представляется положительным напряжением в диапазоне от +3 до +25 В, логическая 1 (MARK) – отрицательным напряжением в диапазоне от –3 до –25 В. На рис. 13 показан сигнал в том виде, в каком он существует на линиях TXD и RXD интерфейса RS–232C.

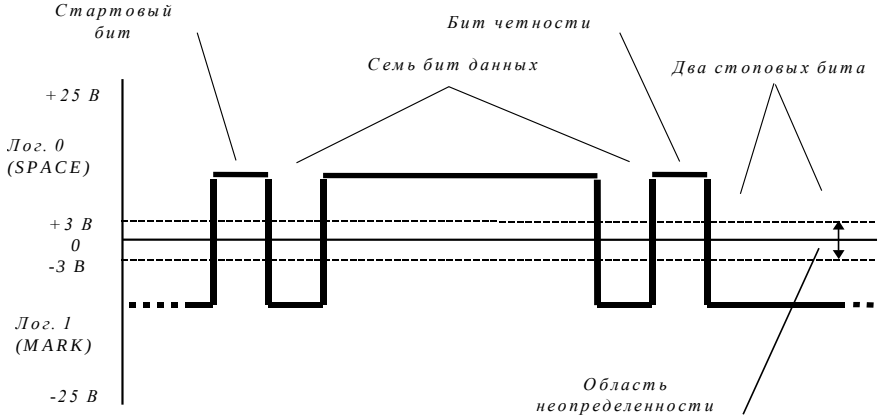


Рис. 13. Вид кода буквы А на сигнальных линиях TXD и RXD

Задание1: *Определить объем кэшируемой памяти при архитектуре прямого отображения, если известен объем кэш-памяти V_{cache} и разрядность памяти тегов N.*

Пример выполнения: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=128 Кбайт и разрядность памяти тегов N=8.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Записать формулу для определения объема кэшируемой памяти.	Объем кэшируемой памяти: $M_{cached} = V_{cache} \cdot 2^N$
2	Рассчитать объем кэшируемой памяти.	$M_{cached} = 128$ Кбайт $\cdot 2^8 = 32$ Мбайт.

Выполните самостоятельно следующие задания, где к каждому значению прибавить номер варианта по журналу:

Задание 1.1: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=256 Кбайт и разрядность памяти тегов N=4.

Задание 1.2: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=128 Кбайт и разрядность памяти тегов N=12.

Задание 1.3: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=64 Кбайт и разрядность памяти тегов N=8.

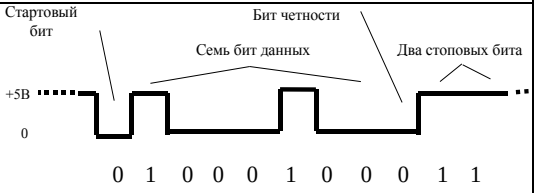
Задание 1.4: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=256 Кбайт и разрядность памяти тегов N=16.

Задание 1.5: Определить объем кэшируемой памяти при архитектуре прямого отображения, если объем кэш-памяти V_{cache}=512 Кбайт и разрядность памяти тегов N=12.

Задание 2: *Представить код символа сигнальными уровнями ТТЛ для передачи по интерфейсу RS-232C.*

Пример выполнения: Представить код символа **D** (код 44h) сигнальными уровнями ТТЛ для передачи по интерфейсу RS-232C.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить биты данных(перевести тз 16-ой системы счисления в 2-ую).	44h = 1 0 0 0 1 0 0
2	Определить бит четности.	$\mathrm{mod}_2\,(1, 0, 0, 0, 1, 0, 0) = 0$
3	Сформировать ТТЛ сигнал для передачи по интерфейсу RS-232C.	

Выполните самостоятельно следующие задания, символы взять в соответствии с таблицей по номеру варианта:

№варианта	Символ	Код	Символ	Код	Символ	Код
1	!	21h	?	3Fh	]	5Dh
2	«	22h	@	40h	^	5Eh
3	#	23h	{	7Bh	_	5Fh
4	$	24h	B	42h	`	60h
5	%	25h	C	43h	a	61h
6	&	26h	|	7Ch	b	62h
7	'	27h	E	45h	c	63h
8	(	28h	F	46h	d	64h
9	)	29h	G	47h	e	65h

10	*	2Ah	H	48h	f	66h
11	+	2Bh	I	49h	g	67h
12	,	2Ch	J	4Ah	h	68h
13	-	2Dh	K	4Bh	i	69h
14	.	2Eh	L	4Ch	j	6Ah
15	/	2Fh	M	4Dh	k	6Bh
16	0	30h	N	4Eh	l	6Ch
17	1	31h	O	4Fh	m	6Dh
18	2	32h	P	50h	n	6Eh
19	3	33h	Q	51h	o	6Fh
20	4	34h	R	52h	p	70h
21	5	35h	S	53h	q	71h
22	6	36h	T	54h	r	72h
23	7	37h	U	55h	s	73h
24	8	38h	V	56h	t	74h
25	9	39h	W	57h	u	75h
26	:	3Ah	X	58h	v	76h
27	;	3Bh	Y	59h	w	77h
28	<	3Ch	Z	5Ah	x	78h
29	=	3Dh	[	5Bh	y	79h
30	>	3Eh	\	5Ch	z	7Ah

Лабораторная работа № 27-28 «Определение времени выполнения программы на конвейерной машине. Составление графа выполнения вычислений в несколько процессорной системе. Прогнозирование перехода в n – битовой схеме прогнозирования со значением счётчика перехода».

Цель работы: Научиться определять время выполнения программы на конвейерной машине, составлять граф выполнения в несколько процессорной системе, научиться прогнозировать переход в n-битовой схеме прогнозирования со значением счётчика перехода.

Теоретическая информация:

Многомашинные и многопроцессорные ВС. Повышение быстродействия и надежности ЭВМ обеспечивается путем создания *многомашинных и многопроцессорных вычислительных систем*. Такие системы позволяют производить распараллеливание во времени выполнения программы или осуществлять многозадачность.

Параллельная обработка – метод увеличения скорости обработки, позволяющий одновременное выполнение нескольких команд.

Многозадачность – возможность выполнения нескольких программ вместе.

В системах с последовательной обработкой многозадачность может быть реализована только в режиме разделения времени.

Последовательная обработка – действия по обработке данных, которые осуществляются в порядке очередности их поступления.

Режим разделения времени – использование компьютера (процессора) несколькими пользователями (программами), обычно при помощи системы квантования времени.

Работа в системах обработки данных и управления, особенно при работе в режиме реального времени, требует высокой надежности и готовности, что решается на основе принципа избыточности, и ориентирует на построение такого рода комплексов.

Многомашинные и многопроцессорные ВС. Многомашинная вычислительная система (ММС) содержит несколько ЭВМ, каждая их которых имеет свою оперативную память (ОП) и работает под управлением своей операционной системы, а также средства обмена информацией между машинами. Реализация обмена информацией происходит в конечном итоге путем взаимодействия операционных систем разных машин между собой. Это ухудшает динамические характеристики процессов межмашинного обмена данными.

Применение многомашинных систем позволяет повысить надежность вычислительных комплексов. При отказе в одной машине обработку данных может продолжать другая машина комплекса. Однако можно заметить, что при этом оборудование комплекса недостаточно эффективно используется для этой цели. Достаточно в этой системе в каждой из машин выйти из строя хотя бы по одному устройству, как вся система становится неработоспособной.

Этих недостатков лишены многопроцессорные системы (МПС). В них процессоры обретают статус рядовых агрегатов вычислительной системы, которые подобно другим агрегатам, таким как модули памяти, каналы, ПУ, включаются в состав системы в нужном количестве. Многопроцессорная система содержит несколько процессоров, работающих с общей ОП, и управляется одной общей операционной системой.

В МПС по сравнению с ММС достигается более быстрый обмен информацией между процессорами через общую ОП, и поэтому может быть получена более высокая производительность, более быстрая реакция на ситуации, возникающие внутри системы и в ее внешней среде, и более высокая надежность и живучесть, так как система сохраняет работоспособность пока работоспособны хотя бы по одному модулю каждого типа устройства.

Однако построение ММС из стандартно выпускаемых ЭВМ с их стандартными операционными системами значительно проще, чем построение МПС, требующих преодоления определенных трудностей, возникающих при реализации общего поля ОП, и, главное, трудоемкой разработки специальной операционной системы.

Важной структурной особенностью ВС является способ организации связи между устройствами (модулями) системы. Он непосредственно влияет на быстроту обмена информацией между модулями системы, а следовательно, и на производительность, быстроту реакции на поступающие запросы, приспособленность к изменению конфигурации, и, наконец, на размеры аппаратурных затрат на осуществление

межмодульных связей. В частности, от организации межмодульных связей зависит частота возникновения конфликтов при обращении процессора к одним и тем же ресурсам и потери производительности из-за конфликтов. Используются следующие способы организации межмодульных связей:

- многоуровневые связи, соответствующие иерархии интерфейсов ЭВМ;
- общая шина;
- регулярные связи между модулями;
- коммутатор межмодульных связей.

Принципы организации МПС и ММС существенно отличаются в зависимости от их предназначения. Поэтому целесообразно различать:

1. ВС, ориентированные на достижение сверхвысокой производительности.
2. ВС, ориентированные на повышение надежности и живучести.

Организация параллелизма вычислений в современных процессорах

В процессорах пятого и шестого поколений — Pentium, Pentium Pro, Pentium MMX, Pentium II – реализованы различные способы конвейеризации и распараллеливания вычислительных процессов.

Конвейерная обработка – раздельное выполнение некоторой операции в несколько этапов (за несколько ступеней) с передачей данных одного этапа следующему.

Напомним, что суперскалярным называется процессор, который имеет конвейеры (более одного), способные обрабатывать инструкции параллельно. Pentium является двухпотоковым процессором (имеет два конвейера), Pentium Pro – трехпотоковым.

Блок-схема процессора Pentium представлена на рис. 14.

Рис. 14. Упрощенная блок-схема процессора Pentium

Микроархитектура этого процессора базируется на идее суперскалярной обработки. Основные команды распределяются по двум независимым исполнительным устройствам (конвейерам U и V). Конвейер U может выполнять любые команды семейства x86, включая целочисленные команды и команды с плавающей точкой. Конвейер V предназначен для выполнения простых целочисленных команд и некоторых команд с плавающей точкой. Команды могут направляться в каждое из этих устройств одновременно, причем при выдаче устройством управления в одном такте пары команд более сложная команда поступает в конвейер U, а менее сложная - в конвейер V. Такая попарная выдача команд возможна, правда, только для ограниченного подмножества целочисленных команд. Команды арифметики с плавающей точкой не могут запускаться в паре с целочисленными командами. Одновременная выдача двух команд возможна только при отсутствии зависимостей по регистрам. При остановке команды по любой причине в одном конвейере, как правило, останавливается и второй конвейер.

Остальные устройства процессора предназначены для снабжения конвейеров необходимыми командами и данными. В

процессоре Pentium используется раздельная кэш-память команд и данных емкостью по 8 Кбайт, что обеспечивает независимость обращений. За один такт из каждой кэш-памяти могут считываться два слова. При этом кэш-память данных построена на принципах двухкратного расслоения, что обеспечивает одновременное считывание двух слов, принадлежащих одной строке кэш-памяти. Кэш-память команд позволяет в одном такте считывать два командных слова, принадлежащих либо одной строке, либо смежным строкам для обеспечения попарной выдачи команд.

В современных процессорах используются различные способы увеличения степени параллелизма вычислений. Рассмотрим основные из них.

Динамическое прогнозирование переходов

Простейшей схемой динамического прогнозирования направления условных переходов является *буфер прогнозирования условных переходов* (branch-prediction buffer) или таблица «истории» условных переходов (branch history table). Буфер прогнозирования условных переходов представляет собой небольшую память, адресуемую с помощью младших разрядов адреса команды перехода. Каждая ячейка этой памяти содержит один бит, который говорит о том, был ли предыдущий переход выполняемым или нет. Это простейший вид такого рода буфера. В нем отсутствуют теги, и он оказывается полезным только для сокращения задержки перехода в случае, если эта задержка больше, чем время, необходимое для вычисления значения целевого адреса перехода. В действительности мы не знаем, является ли прогноз корректным (этот бит в соответствующую ячейку буфера могла установить совсем другая команда перехода, которая имела то же самое значение младших разрядов адреса). Но это не имеет значения. Прогноз – это только предположение, которое рассматривается как корректное, и выборка команд начинается по прогнозируемому направлению. Если же предположение окажется неверным, бит прогноза инвертируется. Конечно, такой буфер можно рассматривать как кэш-память, каждое

обращение к которой является попаданием, и производительность буфера зависит от того, насколько часто прогноз применялся и насколько он оказался точным.

Однако простая однобитовая схема прогноза имеет недостаточную производительность. Рассмотрим, например, команду условного перехода в цикле, которая являлась выполняемым переходом последовательно девять раз подряд, а затем однажды невыполняемым. Направление перехода будет неправильно предсказываться при первой и при последней итерации цикла. Неправильный прогноз последней итерации цикла неизбежен, поскольку бит прогноза будет говорить, что переход «выполняемый» (переход был девять раз подряд выполняемым). Неправильный прогноз на первой итерации происходит из-за того, что бит прогноза инвертируется при предыдущем выполнении последней итерации цикла, поскольку в этой итерации переход был невыполняемым. Таким образом, точность прогноза для перехода, который выполнялся в 90% случаев, составила только 80% (2 некорректных прогноза и 8 корректных). В общем случае, для команд условного перехода, используемых для организации циклов, переход является выполняемым много раз подряд, а затем один раз оказывается невыполняемым. Поэтому однобитовая схема прогнозирования будет неправильно предсказывать направление перехода дважды (при первой и при последней итерации).

Для исправления этого положения часто используется схема двухбитового прогноза. В двухбитовой схеме прогноз должен быть сделан неверно дважды, прежде чем он изменится на противоположное значение. На рис. 15 представлена диаграмма состояний двухбитовой схемы прогнозирования направления перехода.

Рис. 15. Диаграмма состояния двухбитовой схемы прогнозирования

Двухбитовая схема прогнозирования в действительности является частным случаем более общей схемы, которая в каждой строке буфера прогнозирования имеет n-битовый счетчик. Этот счетчик может принимать значения от 0 до 2^{n-1}. Тогда схема прогноза будет следующей:

- Если значение счетчика больше или равно 2^{n-1} (точка на середине интервала), то переход прогнозируется как выполняемый. Если направление перехода предсказано правильно, к значению счетчика добавляется единица (если только оно не достигло максимальной величины); если прогноз был неверным, из значения счетчика вычитается единица.
- Если значение счетчика меньше, чем 2^{n-1}, то переход прогнозируется как невыполняемый. Если направление перехода предсказано правильно, из значения счетчика вычитается единица (если только не достигнуто значение 0); если прогноз был неверным, к значению счетчика добавляется единица.

Исследования n-битовых схем прогнозирования показали, что двухбитовая схема работает почти так же хорошо, и поэтому в большинстве систем применяются двухбитовые схемы прогноза, а не n-битовые.

Буфер прогнозирования переходов может быть реализован в виде небольшой специальной кэш-памяти, доступ к которой осуществляется с помощью адреса команды во время стадии выборки команды, или как пара битов, связанных с каждым блоком кэш-памяти команд и выбираемых с каждой командой. Если команда декодируется как команда перехода, и если переход спрогнозирован как выполняемый, выборка команд начинается с целевого адреса, как только станет известным новое значение счетчика команд. В противном случае продолжается последовательная выборка и выполнение команд. Если прогноз оказался неверным, значение битов прогноза меняется в соответствии с рис. 15.

Разворачивание циклов. Разворачивание циклов представляет собой простой, но полезный метод увеличения размера линейного кодового фрагмента, который может эффективно оптимизироваться. Такое разворачивание выполняется путем многократной репликации (повторения) тела цикла и коррекции соответствующего кода конца цикла.

Базовое планирование конвейера. Для поддержания максимальной загрузки конвейера должен использоваться параллелизм уровня команд, основанный на выявлении последовательностей несвязанных команд, которые могут выполняться в конвейере с совмещением. Чтобы избежать приостановки конвейера, зависимая команда должна быть отделена от исходной команды на расстояние в тактах, равное задержке конвейера для этой исходной команды. Способность компилятора выполнять подобное планирование зависит как от степени параллелизма уровня команд, доступного в программе, так и от задержки функциональных устройств в конвейере.

Динамическое планирование с централизованной схемой управления. В конвейере с динамическим планированием выполнения команд все команды проходят через ступень выдачи строго в порядке, предписанном программой (упорядоченная выдача). Однако они могут приостанавливаться и обходить друг друга на второй ступени (ступени чтения операндов) и тем самым поступать на ступени выполнения

неупорядоченно. Централизованная схема обнаружения конфликтов представляет собой метод, допускающий неупорядоченное выполнение команд при наличии достаточных ресурсов и отсутствии зависимостей по данным.

Выдача нескольких команд в одном такте. Суперскалярные машины используют параллелизм на уровне команд путем посылки нескольких команд из обычного потока команд в несколько функциональных устройств. В типичной суперскалярной машине аппаратура может осуществлять выдачу от одной до восьми команд в одном такте. Обычно эти команды должны быть независимыми и удовлетворять некоторым ограничениям, например, таким, что в каждом такте не может выдаваться более одной команды обращения к памяти. Если какая-либо команда в потоке команд является логически зависимой или не удовлетворяет критериям выдачи, на выполнение будут выданы только команды, предшествующие данной. Предположим, что машина может выдавать на выполнение две команды в одном такте. Одной из таких команд может быть команда загрузки регистров из памяти, записи регистров в память, команда переходов, операции целочисленного АЛУ, а другой может быть любая операция плавающей точки.

Анализ зависимостей компилятором. Когда направление переходов может прогнозироваться во время компиляции, и компилятор может найти команды, которые он может безопасно перенести на место перед точкой перехода, решение, базирующееся на технологии компилятора, идеально. Эти два условия являются ключевыми ограничениями для выявления параллелизма уровня команд статически с помощью компилятора.

Выполнение по предположению. Машина может иметь в своем составе специальные аппаратные средства поддержки выполнения по предположению. Эта методика позволяет машине выполнять команду, которая может быть зависимой по управлению, и избежать любых последствий выполнения этой команды, если окажется, что в действительности команда не

должна выполняться. Механизмы выполнения по предположению позволяют выполнять команду даже до момента оценки условия командой условного перехода, что невозможно при условных командах. Аппаратная поддержка выполнения по предположению достаточно сложна и требует значительных аппаратных ресурсов.

Программная конвейеризация и планирование трасс. Компилятор может реорганизовать последовательность команд так, чтобы избежать приостановок конвейера. Техника, называемая планированием трасс, стала особой областью интереса в 1980-х годах, когда конвейерные машины стали более распространенными. Многие современные компиляторы используют технику планирования команд для улучшения производительности конвейера. В простейшем алгоритме компилятор просто планирует распределение команд в одном и том же базовом блоке. Базовый блок представляет собой линейный участок последовательности программного кода, в котором отсутствуют команды перехода, за исключением начала и конца участка. Планирование такой последовательности команд осуществляется достаточно просто, поскольку компилятор знает, что каждая команда в блоке будет выполняться, если выполняется первая из них, и можно просто построить граф зависимостей этих команд и упорядочить их так, чтобы минимизировать приостановки конвейера. Для простых конвейеров стратегия планирования на основе базовых блоков вполне удовлетворительна. Однако когда конвейеризация становится более интенсивной и действительные задержки конвейера растут, требуются более сложные алгоритмы планирования.

Динамическое планирование с переименованием регистров. Еще одним аппаратным методом минимизации конфликтов по данным является метод переименования регистров (register renaming). При аппаратной реализации метода переименования регистров выделяются логические регистры, обращение к которым выполняется с помощью соответствующих полей команды, и физические регистры,

которые размещаются в аппаратном регистровом файле процессора. Номера логических регистров динамически отображаются на номера физических регистров посредством таблиц отображения, которые обновляются после декодирования каждой команды. Каждый новый результат записывается в новый физический регистр. Однако предыдущее значение каждого логического регистра сохраняется и может быть восстановлено в случае, если выполнение команды должно быть прервано из-за неправильного предсказания направления условного перехода.

В процессе выполнения программы генерируется множество временных регистровых результатов. Эти временные значения записываются в регистровые файлы вместе с постоянными значениями. Временное значение становится новым постоянным значением, когда завершается выполнение команды (фиксируется ее результат). В свою очередь, завершение выполнения команды происходит, когда все предыдущие команды успешно завершились в заданном программой порядке.

Организация вычислительного процесса. Организацию вычислений рассмотрим на примере вычисления быстрого преобразования Фурье (БПФ).

При спектральном анализе сигналов с широкой полосой частот (до десятков и сотен мегагерц) даже при использовании БПФ требования, предъявляемые к быстродействию вычислительной системы, оказываются весьма жесткими. Данная задача может быть решена только в рамках многопроцессорной организации вычислительной системы.

На рис. 16 показан граф, иллюстрирующий необходимую последовательность преобразований переменной x (в точках отсчета $x_0,...,x_7$) в коэффициенты дискретного ряда Фурье $A_0,...,A_7$.

Каждая вершина изображает переменную, а дуги исходят из тех вершин, которые вносят вклад в вершину, к которой направлена дуга. Все вклады аддитивны, и вес каждого вклада $W_j = \exp(-2jp/N)$ (где N – число точек отсчета

анализируемой переменной) указан рядом с соответствующей дугой.

Из графа видно, что вычисление $A_7 = B_3 + C_3W_7$ можно одновременно выполнить на N процессорах (в данном случае на восьми). Процессор P_i в исходном состоянии получает на вход одно из значений x (в порядке, указанном на рис. 16) и в конце вычисления выдает значение A_i. Таким образом, горизонтальные пути на графе соответствуют обработке внутри процессора, а промежуточные вершины — промежуточным этапам вычисления. Остальные дуги графа определяют межпроцессорные обмены. На каждом этапе вычисления число обменов равно N. Результаты, полученные на любом этапе, нужны только для следующего этапа.

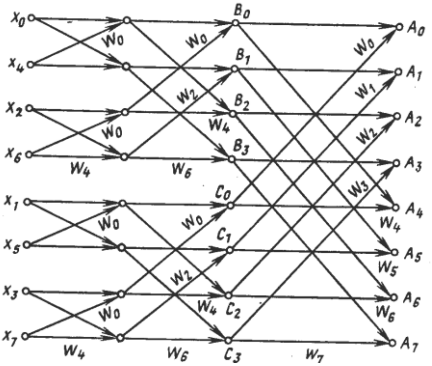


Рис. 16. Граф преобразования переменных x_i в коэффициенты дискретного преобразования Фурье

Если на одном процессоре решение может быть получено за $N \cdot log_2 N$ шагов, то на N процессорах требуется в N раз меньше шагов.

Отметим, что в рассмотренном примере все обмены между процессорами предопределены, и управление ими сводится к осуществлению коммутации по заранее составленной программе. Поэтому требования к системе коммутации здесь значительно упрощены по сравнению с общим случаем, когда решение о коммутации надо принимать, исходя из результатов вычислений.

В данном случае все процессоры, начавшие вычисления, заняты их выполнением до получения результата. На практике процессоры освобождаются постепенно, причем, как правило, в неизвестные заранее моменты времени. Если освобождающийся ресурс не используется (он выделен только для данной работы), это приводит к снижению параллелизма и соответственно к снижению эффективности использования ресурсов системы.

Как следует из рассмотренного примера, обмены информацией между процессорами достигают большого значения, что может привести к малой эффективности многопроцессорной обработки. Поэтому выгоднее распараллеливать обработку данных более крупными блоками. За счет наличия в блоках операций самого различного характера появляется возможность наиболее полного использования процессоров.

В рассматриваемом примере процессоры работают таким образом, что данные, полученные на предыдущем этапе, необходимы для выполнения последующих операций. При отказе одного из процессоров общее решение оказывается неверным. Во избежание этого вычисления, ранее проводившиеся на отказавшем процессоре, необходимо продолжить на другом (исправном).

Задание 1: Определить, за какое время T будет выполнена программа, состоящая из n команд, на конвейерной машине с n этапами выполнения операций, которые имеют разные длительности t_i, если известны накладные расходы на организацию конвейера.

Пример выполнения: Определить, за какое время T будет выполнена программа, состоящая из 10 команд, на конвейерной машине с пятью этапами выполнения операций, которые имеют длительности t_i, равные 50, 40, 40, 50, 30 нс соответственно, если накладные расходы на организацию конвейера составляют $t_{нс}$ = 3 нс.

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Написать формулу для определения среднего времени выполнения команды в конвейерном режиме.	Среднее время выполнения команды в конвейерном режиме: $t_{ср} = t_{i\ max} + t_{нр}$
2	Определить среднее время выполнения команды в конвейерном режиме.	$t_{ср} = 50 + 3 = 53$ нс
3	Написать формулу для определения времени выполнения программы.	Время выполнения программы: $T = T_{1команды} + \sum_{i=2}^{10} t_{ср} = t_{ср} \cdot 5 + t_{ср} \cdot (10-1) = t_{ср} \cdot 14$
4	Определить время выполнения программы.	T = 53 * 14 = 742 нс

Решите самостоятельно следующие задания, где n – номер варианта по журналу:

Задание 1.1: Определить, за какое время T будет выполнена программа, состоящая из 2*n команд, на конвейерной машине с пятью этапами выполнения операций, которые имеют длительности ti, равные 20+n, 40+n, 60+n, 50+n,

70+n нс соответственно, если накладные расходы на организацию конвейера составляют tнс = 5+n нс.

Задание 1.2: Определить, за какое время T будет выполнена программа, состоящая из 10*n команд, на конвейерной машине с четырьмя этапами выполнения операций, которые имеют длительности ti, равные 50+n, 50+n, 50+n, 50+n нс соответственно, если накладные расходы на организацию конвейера составляют tнс = 10+n нс.

Задание 1.3: Определить, за какое время T будет выполнена программа, состоящая из 7*n команд, на конвейерной машине с шестью этапами выполнения операций, которые имеют длительности ti, равные 60+n, 60+n, 40+n, 40+n, 60+n, 50+n нс соответственно, если накладные расходы на организацию конвейера составляют tнс = 4+n нс.

Задание 1.4: Определить, за какое время T будет выполнена программа, состоящая из 6+n команд, на конвейерной машине с пятью этапами выполнения операций, которые имеют длительности ti, равные 30+n, 40+n, 50+n, 30+n, 50+n нс соответственно, если накладные расходы на организацию конвейера составляют tнс = 8+n нс.

Задание 1.5: Определить, за какое время T будет выполнена программа, состоящая из 2*n команд, на конвейерной машине с шестью этапами выполнения операций, которые имеют длительности ti, равные 70+n, 50+n, 50+n, 40+n, 40+n, 40+n нс соответственно, если накладные расходы на организацию конвейера составляют tнс = 20+n нс.

Задание 2: Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно n?

Пример выполнения: Как будет спрогнозирован переход в 5-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 10?

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Определить среднее из возможных значений счетчика.	Среднее из возможных значений счетчика в n-битовой схеме прогнозирования: $2^{n-1} = 2^4 = 16$
2	Сравнить значение счетчика со средним значением и определить, как будет спрогнозирован переход	10 < 16, следовательно, переход будет спрогнозирован, как невыполняемый

Решите самостоятельно следующие задания, где n –номер варианта по журналу:

Задание 2.1. Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 8?

Задание 2.2. Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 20?

Задание 2.3. Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 3?

Задание 2.4. Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 130?

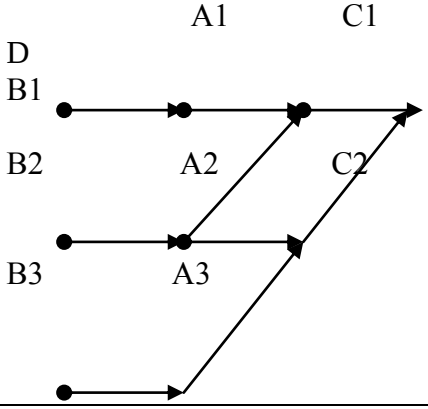
Задание 2.5. Как будет спрогнозирован переход в n-битовой схеме прогнозирования переходов, если значение счетчика для этого перехода равно 36?

Задание 3: Составить граф выполнения следующих вычислений в n-процессорной системе.

Пример выполнения: Составить граф выполнения следующих вычислений в трехпроцессорной системе:

1. A1=B1 · 2
2. A2=B2 · 4
3. A3=B3 · 8
4. C1=(A1+A2)/2
5. C2=(A2+A3)/2
6. D=C1+C2

Решение: Предварительно заполните таблицу, подобрав к каждому алгоритму конкретное соответствие из данного задания.

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1	Выделить группы операций, которые могут выполняться одновременно и определить количество этапов вычислений.	Выделяем три этапа, в которые входят операции: 1-й этап: 1, 2, 3 2-й этап: 4, 5 3-й этап: 6
2	Нарисовать граф.	

Решите самостоятельно следующие задания (для любого варианта выполнить все задания):

Задание 2.1: Составить граф выполнения следующих вычислений в двухпроцессорной системе:

1. A1=B1 · B1
2. B2=A1 · A1
3. C=2 · 100
4. D=B2+C

Задание 2.2: Составить граф выполнения следующих вычислений в трехпроцессорной системе:

1. A1=B1+C1
2. A2=B1·C1
3. A3=B1/C1
4. D=A1+A2
5. D= D+A3

Задание 2.3: Составить граф выполнения следующих вычислений в трехпроцессорной системе:

1. A1=B1 + C1
2. A2=B2 + C2
3. A3=A1 · A2
4. D=A1+A2
5. D= D +A3

Задание 2.4: Составить граф выполнения следующих вычислений в двухпроцессорной системе:

1. A1=B1·2
2. A2=B2·2
3. C1=A1·A2
4. C2=B3·B4
5. D=C1+C2

Задание 2.5: Составить граф выполнения следующих вычислений в двухпроцессорной системе:

1. A1=B1 + C1
2. A2=A1 + C2
3. A3=A2 + C3
4. D=A1·A2
5. D=D·A3

Лабораторная работа № 29-30 «Знакомство с программой разработки и отладки программ на языке Ассемблера - Debug».

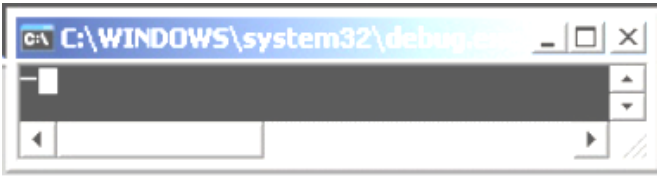
Цель работы: Познакомиться с программой Debug и командами языка Ассемблера. Выполнение работы предполагает изучение устройства работы центрального процессора и организации памяти ЭВМ на конкретных примерах с использованием отладчика Debug.

Теоретическая информация:

Debug – 16-разрядная утилита операционной системы MS DOS. Название программы происходит от Bugs – насекомые. На программистском слэнге Bugs означает «ошибки в программе». Используя Debug для пошаговой трассировки тестируемого кода, эти ошибки можно найти и исправить.

Располагается отладчик Debug в системной папке system32 папки WINDOWS (файл debug.exe). Запуск отладчика удобнее всего выполнять из меню Пуск/Выполнить.

Основной режим работы отладчика Debug (характерная особенность – присутствие чёрточки и мигающего справа от неё курсора) устанавливается автоматически после загрузки отладчика.



В этом режиме можно набирать любые макрокоманды отладчика от A до W. Выход из этого режима означает завершение работы отладчика.

Вход в режим ассемблирования из основного режима работы отладчика осуществляется набором команды A. Признак нахождения в этом режиме – наличие адреса (например, 0CC0:0100) перед знаком курсора:

```
-A 100
0CC0:0100 MOV DX,140
0CC0:0103
```

После набора команды необходимо нажатием клавиши Enter ввести её в память (пока не нажата клавиша Enter команда в память не введена).

Если команда набрана с ошибкой и нажата клавиша Enter, эта команда в память не вводится. Вместо этого отладчик предлагает ввести её повторно, по тому же самому адресу:

```
-A 100
0CFC:01

0CFC:01
0CFC:01
```

Возвращение из режима ассемблирования в основной режим работы отладчика – нажатие клавиши Enter в пустой

строке (когда не набрано ни одного символа), например, в строке 0CFC:0103.

Режим ввода данных может быть реализован во время выполнения макрокоманд E и R. В этом режиме вводятся данные в ячейки оперативной памяти или в регистры процессора:

Возвращение из этого режима в основной режим работы отладчика – нажатие клавиши Enter. Продолжение работы в режиме ввода данных – нажатие клавиши пробел. Нажимая клавишу пробел, можно попасть на любую ячейку памяти и ввести в неё нужное число. Ввод значений завершается нажатием клавиши Enter.

Режим выполнения кода из основного режима работы отладчика осуществляется набором команды G. Как правило, время нахождения в данном режиме – считанные мгновенья. Исключение – зависание программы или выполнение огромных циклов.

Возвращение в основной режим работы отладчика осуществляется автоматически, без участия работающего с этим отладчиком человека.

Режим ввода информации с клавиатуры во время работы в режиме выполнения кода. Завершение этого режима и продолжение выполнения кода осуществляется нажатием клавиши Enter.

Режим трассировки выполняемого кода. В данном режиме процессор выполняет заранее заданное ограниченное количество команд. Вход в режим трассировки осуществляется набором команды T. Возвращение в основной режим работы осуществляется автоматически, без участия работающего с отладчиком человека.

Аварийное прекращение работы отладчика. Имеет место в случае, когда процессор пытается исполнить запрещённый в режиме эмуляции работы микропроцессора i8086 программный

код. Например, пытается выполнить команды работы с портами ввода-вывода данных.

Debug.exe - программа отладчик, которую используют для проверки и отладки выполняемых файлов. Использовалась при операционной системе *MS-DOS*. Под более поздние версии операционных систем работает через эмулятор *MS-DOS* и имеет ограниченные возможности. Вызывается через командную сроку

```
DEBUG
```

или

```
'(%8*                >>^bkd@>fZjrjml@bfyBnZceZ
>iZjZf_lju@@
```

Например: `DEBUG C:\...\My.com`.

Данная программа является консольным приложением и предназначена для создания или изменения кода файлов. С помощью неё можно создавать простые приложение под *MS-DOS* и отслеживать их работу. Данный отладчик находится на самом низком уровне компиляторов *assembler*. Но обладает неплохими возможностями, такими как просмотр и изменение памяти, получение состояния регистров.

Команды debug.exe

Правила набора команд debug.exe:

- В debug.exe не различается регистр букв.
- Пробелы в командах используется только для разделения параметров.
- Вводимые числа должны быть в шестнадцатеричной системе счисления, причем без завершающей буквы h.
- Сегмент и смещение записываются с использованием двоеточия, в формате сегмент:смещение, например, CS:3C1 (смещение 3C1h в сегменте кода) или 40:17 (смещение 17h в сегменте, адрес начала которого - 40[0]h).

После загрузки отладчика на экране появится приглашение, выглядящее в виде дефиса. Регистры *CS, DS, ES, SS* в этот момент инициализированы адресом 256-байтного префикса сегмента программы, а рабочая область в памяти будет начинаться с адреса этого префикса + 100h.

Команды *debug.exe* вводятся сразу после приглашения на месте, которое отмечено курсором. Каждая команда состоит из идентификатора и параметров, идентификатор состоит из одной буквы.

Таблица 1. Краткая таблица всех команд debug.exe

Команда	Описание	Формат
A (Assemble)	Транслирование команд ассемблера в машинный код; адрес по умолчанию - CS:0100h.	*[illegible]*
C (Compare)	Сравнение содержимого двух областей памяти; по умолчанию используется DS. В команде указывается либо длина участков, либо диапазон адресов.	*[illegible]*
D (Display/Dump)	Вывод содержимого области памяти в шестнадцатеричном и ASCII-форматах. По умолчанию используется DS; можно указывать длину или диапазон.	*[illegible]*
E (Enter)	Ввод в память данные или инструкции машинного кода; по умолчанию используется DS.	*[illegible]*
F (Fill)	Заполнение области памяти данными из списка; по умолчанию используется DS. Использовать можно как длину, так и диапазон.	*[illegible]*
G (Go)	Выполнение отлаженной	*[illegible]*

	программы на машинном языке до указанной точки останова; по умолчанию используется CS. При этом убедитесь, что IP содержит корректный адрес.	>Z^j_kBhklZgh\Z!@
H (Hexadecimal)	Вычисление суммы и разности двух шестнадцатеричных величин.	+_ebqbgZB 1> _ebqbgZB!
I (Input)	Считывание и вывод одного байта из порта.	,Z^j_kBihjlZ!
L (Load)	Загрузка файла или данных из секторов диска в память; по умолчанию - CS:100h. Файл можно указать с помощью команды N или аргумента при запуске debug.exe.	L >Z^j _kB\BiZfylbB^eyBaZ]jmadb!@ L >Z^j_kB\BiZfylbB^eyBaZ]jmadb!>ghf_jB^bkdZ! gZqZevgucBk_dlhj! dhebq_kl\hBk_dlhjh\!@@
M (Move)	Копирование содержимого ячеек памяти; по умолчанию используется DS. Можно указывать как длину, так и диапазон.	0gZqZevgucBZ^j_ k! /^ebgZ! Z^j_kBgZagZq_gby! 0gZqZevgucBZ^j_k! dhg_qgucBZ^j_k! Z^j_kBgZagZq_gby!
N (Name)	Указание имени файла для команд L и W.	1bfyBnZceZ!
O (Output)	Отсылка байта в порт.	2Z^j_kBihjlZ![Zcl!
P (Proceed)	Выполнение инструкций CALL, LOOP, INT или повторяемой строковой инструкции с префиксами REPnn, переходя к следующей инструкции.	3> Z^j_kBgZqZeZ!@ >dhebq_kl\hBbgkljmdpb c!@
Q (Quit)	Завершение работы debug.exe.	Q
R (Register)	Вывод содержимого	5bfyBj_]b kljZ!

	регистров и следующей инструкции.	
S (Search)	Поиск в памяти символов из списка; по умолчанию используется DS. Можно указывать как длину, так и диапазон.	6gZqZevgucBZ^j_k! /^ebgZ!^Zggu_! 6gZqZevgucBZ^j_k! dhg_qgucBZ^j_k! ^Zggu_!
T (Trace)	Пошаговое выполнение программы. Как и в команде P, по умолчанию используется пара CS:IP. Замечу, что для выполнения прерываний лучше пользоваться командой P.	7> Z^j_kBgZqZeZ!@ >dhebq_kl\hB\uihegy_f uoBdhfZg^!@
U (Unassemble)	Дизассемблирование машинного кода; по умолчанию используется пара CS:IP. К сожалению, debug.exe некорректно дизассемблирует специфические команды процессоров 80286+, хотя они все равно выполняются корректно.	8>gZqZevgucBZ^j_k!@ 8>gZqZevgucBZ^j_k dhg_qgucBZ^j_k!@
W (Write)	Запись файла из debug.exe; необходимо обязательно задать имя файла командой N, если он не был загружен. А программы записываются только в виде файлов .COM!	:>Z^j_k! >ghf_jB^bkdZ! gZqZevgucBk_dlhj! dhebq_kl\hBk_dlhjh\!@]

Просмотр областей памяти

Рассмотрим подробно работу команды *D*, позволяющей просматривать содержимое отдельных областей памяти.

Этот пример использует команду *D* для просмотра области памяти, начиная с 0159:0240:

```
-d 0159:0240
0159:0240  00 00 00 00 00 00 00 00-00 00 6C 0D 00 00 00 00   ..........l.....
0159:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0159:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0159:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0159:0280  00 00 00 00 00 00 00 00-00 FF FF FF FF 00 00 00   ................
0159:0290  FF 00 00 00 00 00 00 00-00 00 4E 4F 20 4E 41 4D   ..........NO NAM
0159:02A0  45 20 20 20 20 00 26 81-4F 03 00 01 CB 00 00 00   E    .&.O.......
0159:02B0  00 00 00 00 00 00 00 00-00 00 00 01 07 04 FF 02   ................
-
```

Здесь на запрос просмотра участка памяти мы получили восемь строк, в которых указано содержимое выбранной области памяти. Каждая строка состоит из трех частей:

- Адрес первого слева показанного байта в формате *сегмент:смещение*.
- Шестнадцатеричное представление параграфа (16 байт), начинающегося с указанного в начале строки байта.
- Символы этого же параграфа в ASCII-формате.

Адрес, указанный в строке, относится исключительно к первому байту в параграфе, а адреса последующих байтов следует вычислять самостоятельно. Шестнадцатеричное представление содержит по два знака в каждом байте, а сами байты разделены пробелами для облегчения чтения. Кроме того, следует отметить, что восьмой и девятый байты разделены дефисом, разделяя тем самым параграф на две части и облегчая вычисление адресов байтов в параграфе.

Полезные приемы с командой D

Проверка параллельных и последовательных портов

Первые 16 байт области данных BIOS содержат адреса параллельных и последовательных портов. Поэтому с помощью следующей команды можно проверить эти порты:

```
-d 40:00
```

Первые выведенные восемь байтов указывают на адреса последовательных портов *COM1-COM4*. Следующие 8 байтов указывают на адреса параллельных портов *LPT1-LPT4*.

Например, если на вашем компьютере есть один параллельный порт, то первые два байта будут, скорее всего, такими: 7803. Адрес порта записывается в обращенной последовательности, т.е. 0378.

Проверка оборудования

Первые два байта, располагающиеся в BIOS по адресу 410h, содержат информацию об установленном в системе оборудовании. Находим эти байты командой:

```
-d 40:10
```

Предположим, что первые два байта окажутся 23 44. Расшифруем эти байты для получения информации об установленных устройствах. Для этого обратим эти байты (44 23), затем переведем их в двоичную систему счисления. Получаем:

Значение бита	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	1
Позиция бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Что означают эти биты? Продолжаем расшифровывать:

Биты	**Устройство**
15, 14	Число параллельных портов (01 = 1 порт, ...)
11, 10, 9	Число последовательных портов (..., 010 = 2 порта, ...)
7, 6	Число дисководов (00 = 1 дисковод, 01 = 2, 10 = 3, 11 = 4)
5, 4	Начальный видеорежим (01 = 40x25 цветной, 10 = 80x25 цветной, 11 = 80x25 монохромный)
1	Присутствие математического сопроцессора (0 = нет, 1 = есть)
0	Наличие привода для дискет (0 = нет, 1 = есть)

Проверка состояния регистра клавиатуры

В области данных *BIOS* по адресу *417h* находится первый байт, который хранит состояние регистра клавиатуры. Выключаем *Num Lock* и *Caps Lock*, затем набираем команду:

```
-d 40:17
```

Первый байт будет равен 00. Включив *Num Lock* и *Caps Lock*, снова выполняем команду. Теперь первый байт должен равняться 60. Опытным путем установлено, что при включенном *Num Lock* первый байт равен 20, а при *Caps Lock* - 40.

Проверка состояния видеосистемы

По адресу *449h* в *BIOS* находится первая область видеоданных. Для проверки набираем:

```
-d 40:49
```

Первый байт показывает текущий видеорежим (к примеру, 03 - цветной), а второй - число столбцов (например, 50 - режим с 80 столбцами). Число строк можно найти по адресу *484h* (40:84).

Проверка копирайта BIOS и серийного номера

Сведения об авторских правах на *BIOS* встроены в *ROM BIOS* по адресу *FE00:0*. Строку с копирайтом можно легко найти в ASCII-последовательности, а серийный номер - в виде шестнадцатеричного числа. Хотя, строка с указанием авторских прав может быть длинной и не умещаться в выведенную область памяти. В таком случае следует просто ввести еще раз *D*.

Проверка даты произвоства BIOS

Эта дата также записана в *ROM BIOS* начиная с адреса *FFFF:5*. После выполнения соответствующей команды в ASCII-последовательности будет находиться эта дата, записанная в формате *мм/дд/гг*.

Непосредственный ввод программы в память с помощью debug.exe

debug.exe позволяет вводить программу непосредственно в память машины, а затем следить и управлять ее выполнением. Мы будем вводить программу в машинных кодах, используя команду *E*. При этом будьте бдительны - *ввод ошибочных данных по ошибочному адресу чреват непредсказуемыми последствиями!* Хотя к серьезным проблемам в системе это вряд ли приведет, но потерять все данные, введенные в *debug.exe*, можно запросто.

Программа, которую необходимо ввести, использует данные, заложенные непосредственно в теле инструкций. Далее показан листинг программы на Ассемблере, в комментариях указаны аналоги команд языка в машинных кодах, а также объяснение каждой команды. Заметьте, что в числах нет символа *h*, поскольку, как было сказано выше, *debug.exe* понимает только числа в шестнадцатеричной системе.

```
029$;dh^%aZghkbfagZq_gb_
K\$;
 $ ' '   $ ;                  d h ^
 d   a g Z q _ g b x   $ ;
 0 2 9   % ;     $ ;     d h ^     % '       a Z
 \   % ;
 $ ' '   % ;     $ ;     d h ^       '       i _
 $ ;   d   % ;
 0 2 9   & ;     % ;     d h ^     % & %     a Z
 \   & ;
 6 8 %   & ;     $ ;     d h ^     % &       h l
 b a   & ;
 6 8 %   $ ;     $ ;     d h ^     % &       h q
 - 0 3               d h ^    ( % ( (      i _ j
 i j h ] j Z f f u
```

Как можно заметить, каждая машинная инструкция имеет длину от 1 до 3 байтов. Первый байт указывает операцию,

последующие - ее операнды. Исполнение программы начинается соответственно с первой инструкции и последовательно проходит через все инструкции одну за другой.

Теперь можно ввести программу в память. Разделим машинный код на три части по шесть байт и введем каждую, используя команду *E* и начиная с адреса *CS:100*.

```
-E CS:100 B8 23 01 05 25 00
-E CS:106 8B D8 03 D8 8B CB
-E CS:10C 2B C8 2B C0 EB EE
-
```

Теперь, когда программа введена в память, попробуем управлять ее выполнением. Для начала проверим текущее состояние регистров и флагов, для этого вводим команду *R*. Отладчик выведет содержимое регистров в шестнадцатеричной форме; на разных машинах содержимое регистров может различаться.

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B12  ES=0B12  SS=0B12  CS=0B12  IP=0100   NV UP EI PL NZ NA PO NC
0B12:0100 B82301        MOV     AX,0123
```

Итак, как можно видеть, *debug.exe* инициализировал сегменты *DS, ES, SS, CS* одним и тем же адресом. Регистр *IP* содержит *0100*, указывая на то, что инструкции выполняются со смещения *100h* относительно *CS* (а мы, вводя инструкции в память, как раз указали этот адрес).

Здесь же указаны и значения флагов переполнения, направления, прерывания, знака, нуля, дополнительного переноса, четности и переноса:

Значение	Описание
NV	Отсутствие переполнения
UP	Направление вверх или вправо
EI	Разрешение прерываний
PL	Положительный знак

NZ	Ненулевое значение
NA	Отсутствие дополнительного переноса
PO	Нечетное слово
NC	Отсутствие переноса

После регистров и состояния флагов debug.exe выводит информацию о первой инструкции, которая будет выполняться:

- Адрес инструкции, в нашем случае это 0B12:0100, где 0B12 - адрес сегмента кода.
- Машинный код, соответствующей этой инструкции (B82301).
- Собственно инструкция, записанная на ассемблере (MOV AX,0123).

Теперь, после анализа содержимого регистров и флагов, давайте перейдем к выполнению программы. Выполнять программу мы будем пошагово, используя команду *T*. Использовав в первый раз команду *T*, мы выполняем инструкцию *MOV*. Здесь машинный код операнда инструкции - *2301*. Операция помещает *23* в *AL* (младшая половина *AX*), а *01* - в *AH* (старшая).

После этого *debug.exe* снова выводит информацию о регистрах:

```
-T

AX=0123  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B12  ES=0B12  SS=0B12  CS=0B12  IP=0103   NV UP EI PL NZ NA PO NC
0B12:0103 052500        ADD     AX,0025
-
```

Теперь *AX* содержит *0123h, IP - 0103h* (следовательно, длина выполненной инструкции: *0103h - 0100h = 3* байта), а в качестве следующей инструкции указана операция *ADD*.

Так, раз за разом выполняя команду *T*, мы дойдем до последней инструкции *JMP 100*. Она установит регистр *IP* в *100h*, и *debug.exe* вернется к началу программы. Возвращаясь к началу программы, следует заметить, что в *DS, ES, SS* и *CS* содержится один и тот же адрес. Дело в том, что *debug.exe* рассматривает введенные программы исключительно как

программы *.COM*. А в программах *.COM*, в отличие от *.EXE*, стек, код и данные хранятся в одном сегменте.

Ассемблирование и дизассемблирование

В прошлом примере мы вводили программу в машинных кодах, однако, *debug.exe* вполне способен понимать инструкции, записанные на ассемблере. Для работы с такими программами в *debug.exe* используются команды *A* и *U*.

Команда *A* запрашивает инструкции на ассемблере и преобразовывает их в машинный код. Для начала инициализируем начальный адрес для ввода инструкций (100h):

```
-a 100
```

Отладчик выведет адрес сегмента кода и смещения (например, *13F2:0100*). Теперь мы должны ввести следующие инструкции на ассемблере в память, после каждой строки нажимая *Enter*:

```
MOV CL, 42
MOV DL, 2A
ADD CL, DL
JMP 100
```

После ввода последней инструкции нажимаем *Enter* дважды, чтобы указать отладчику, что мы закончили вводить текст программы. Теперь программу можно запускать, используя команды *R* для просмотра регистров и *T* для трассировки. Замечу, что в своих программах при наличии инструкций *INT* их следует обрабатывать не командой *T*, а командой *P*, которая обрабатывает все прерывание сразу.

Перейдем к процедуре дизассемблирования, а в качестве примера возьмем только что введенную программу. Используем адреса первой и последней инструкций для указания диапазона, который мы собираемся дизассемблировать, т.е. *100h* и *107h*.

```
-u 100, 107
```

После выполнения этой команды *debug.exe* выведет инструкции, находящиеся в указанном диапазоне, на ассемблере, в машинных кодах, а также адрес каждой инструкции:

```
0B12:0100 B142          MOV     CL,42
0B12:0102 B22A          MOV     DL,2A
0B12:0104 00D1          ADD     CL,DL
0B12:0106 EBF8          JMP     0100
-
```

Сохранение программы на диске

Сначала задается имя файла:

```
-N my_file.com
```

Затем в регистр *CX* необходимо поместить размер программы в байтах. Он будет равен разности конечного и начального смещений. Теперь остается только осуществить запись на диск командой *W* и в результате увидеть записанное количество байтов. В итоге мы получаем программу, готовую к исполнению. Выход осуществляется командой *q*. Пример:

```
-A
0B3B:0100 mov ax,1234
0B3B:0103 mov ah, 4c
0B3B:0105 int 21
0B3B:0107
-u CS:100, 106
0B3B:0100 B83412 MOV AX,1234
0B3B:0103 B44C MOV AH,4C
0B3B:0105 CD21 INT 21
-r cx
CX 0000
:7
-r
AX=0000  BX=0000  CX=0007  DX=0000  SP=FFEE
BP=0000 SI=0000 DI=0000
DS=0B3B ES=0B3B SS=0B3B CS=0B3B IP=0100 NV
UP EI PL NZ NA PO NC
```

```
0B3B:0100 B83412 MOV AX,1234
-N my.com
-W
 A Z i b k v                [ Z c l
-q
```

Для выполнения этой лабораторной работы понадобится несколько команд ассемблера:

- MOV AH — регистр *AH* числа *02* для указания системной функции - вывод символа на экран;
- MOV DL — регистр *DL* кода символа;
- INT 21 - основное прерывание DOS (процедура), реализующее много различных функций; номер функции записывается предварительно в регистр AH; для распечатки символа на экране - в регистре AH функция `02`, при этом в *DL* записывают предварительно код символа.
- INT 20 - прерывание DOS, осуществляющее выход из программы (из .COM-программы).

Пример:

Вывести символ "*" на экран.

```
mov AH,02 ; системная функция 02 - вывод символа на экран
mov DL,2A ; ASCII-код звездочки
int 21h ; прерывание для вывода "*"
int 20h ; выход из программы
```

Задание:

- Изучить теоретический материал.
- Проделать описанные по ходу текста команды.
- Найдите сумму и разность 2-х чисел: 1-е число - номер в группе (переведенное в шестнадцатиричную форму), 2-е - число, противоположное номеру первой буквы фамилии в алфавите (отрицательное число в дополнительном коде). Сумму и разность переведите в десятичную форму.

- Просмотрите содержимое регистров микропроцессора, а также флагов и выпишите их в протокол. Какую функцию выполняет каждый из регистров?
- Запишите в регистр AX первое число (из задания 3), а в регистр BX - второе (из задания 3). Введите в оперативную память в сегмент кода (смещение 100) машинную команду сложения регистров AX и BX. Просмотрите на экране ее ассемблерную форму. Выполните эту команду, результат переведите в десятичную форму.
- Введите в оперативную память в сегмент кода (смещение 100) набор команд ассемблера для распечатки символа на экране - первой буквы вашей фамилии. Проверьте программу в DEBUG. Затем запишите ее на диск в виде .COM-файла. Чему равен размер программы? Запустите ее на выполнение из DOS.
- Все действия опишите в отчете.

Вопросы к допуску

1. Для чего предназначена программа Debug?
2. Как запустить эту программу?
3. Каким образом вводятся команды в Debug?
4. Что такое идентификатор?
5. Как производится ввод программы в Debug?

Контрольные вопросы

1. Какая команда производит ввод в память данные или инструкции машинного кода?
2. Как сравнить содержимое двух областей памяти?
3. Каким образом заполнить область памяти данными из списка?
4. Какая команда производит выполнение отлаженной программы на машинном языке?
5. Какая команда записывает файл из Debug?
6. Как производится ассемблирование и дизассемблирование?

Рекомендуемая литература

1.Архитектура компьютера, Э. Таненбаум, С-Пб., Изд: «Питер», 2003.

2.Архитектура ЭВМ, А.П. Жмакин, С-Пб., Изд: «БВХ-Петербург», 2006.

3.Организация ЭВМ, К. Хамахер, З. Вранешич, С. Заки, Санкт-Петербург, «Питер», 2003.

4.IBM для пользователя, В.Э. Фигурнов, Москва, Изд: «Инфра-М», 2003.

5.Основы цифровой техники, О.П. Новожилов, Москва, ИП «РадиоСофт», 2004.

6.Вычислительные системы, сети и телекоммуникации, В.Л. Бройдо, Санкт-Петербург, «Питер», 2005.

7.Структурная организация и архитектура компьютерных систем, Вильям Столлингс (перевод В.Т. Терпышного), Москва, Изд: «Вильямс», 2002.

8.Технологии разработки программного обеспечения, Орлов С.А. – СПб.: Питер, 2004. – 527 с.

9.Организация ЭВМ и систем, Цилькер Б.Я. – СПб.: Питер, 2006. – 668 с.