Государственное областное бюджетное
профессиональноеобразовательное учреждение
«Усманский многопрофильный колледж»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ И ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

по учебной дисциплине ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем

Программы подготовки специалистов среднего звена (ППССЗ)

по специальности: 09.02.04Информационные системы (по отраслям)

по программе базовой подготовки

Усмань 2020

Методические указания по организации и проведению практических работ по учебной дисциплине ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем по специальности09.02.04Информационные системы (по отраслям).

Организация-разработчик: Государственное областное бюджетное профессиональное образовательное учреждение «Усманский многопрофильный колледж»

Разработчики:
Мотин И.А. преподаватель информатики

Рассмотрены и утверждены на заседании предметно-цикловой комиссии естественнонаучных дисциплин

Протокол № 6 от 30.06.2020 г.

Председатель предметно-цикловой комиссии естественнонаучных дисциплин _____ Коровина Т.В.

УТВЕРЖДАЮ

Заместитель директора Лаува О.А.

по учебно-методической работе

Введение

Практические занятия, как вид учебных занятий, направлены на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных практических умений и составляют важную часть теоретической и профессиональной практической подготовки.

В процессе практического занятия обучающиеся выполняют одно или несколько практических заданий в соответствии с изучаемым содержанием учебного материала.

Содержание практических занятий по учебной дисциплине ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем должно охватывать весь круг профессиональных умений, на подготовку к которым ориентирована данная дисциплина, а в совокупности охватывать всю профессиональную деятельность, к которой готовится специалист.

При разработке содержания практических занятий следует учитывать, что наряду с формированием умений и навыков в процессе практических занятий обобщаются, систематизируются, углубляются и конкретизируются теоретические знания, вырабатывается способность и готовность использовать теоретические знания на практике, развиваются интеллектуальные умения.

Выполнение обучающимися практических занятий проводится с целью:

- формирования практических умений в соответствии с требованиями к уровню подготовки обучающихся, установленными ФГОС и рабочей программой учебной дисциплины ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем по конкретным разделам и темам дисциплины;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- совершенствования умений применять полученные знания на практике, реализации единства интеллектуальной и практической деятельности;
- развития интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработки таких профессионально значимых качеств, как самостоятельность,

ответственность, точность, творческая инициатива при решении поставленных задач при освоении общих и профессиональных компетенций.

Соответственно в процессе освоения учебной дисциплины Психология обучающиеся должны овладеть:

умениями:

- с помощью программных средств организовывать управление ресурсами вычислительных систем;

- осуществлять поддержку функционирования информационных систем.

знаниями:

- построение цифровых вычислительных систем и их архитектурные особенности;
- принципы работы основных логических блоков систем;
- классификацию вычислительных платформ и архитектур;
- параллелизм и конвейеризацию вычислений;
- основные конструктивные элементы средств вычислительной техники, функционирование, программно-аппаратная совместимость.

Вышеперечисленные умения и знания направлены на формирование следующих профессиональных и общих компетенций студентов:

Профессиональные компетенции:

ПК 1.1. Собирать данные для анализа использования и функционирования информационной системы, участвовать в составлении отчетной документации, принимать участие в разработке проектной документации на модификацию информационной системы.

ПК 1.2. Взаимодействовать со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности

ПК 1.9. Выполнять регламенты по обновлению, техническому сопровождению и восстановлению данных информационной системы, работать с технической документацией.

Общие компетенции:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и

способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

Данные методические указания по организации и проведению практических работ составлены в соответствии с содержанием рабочей программы учебной дисциплины ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем специальности 09.02.04Информационные системы (по отраслям) по программе базовой подготовки.

Учебная дисциплина ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем изучается в течение одного семестра. Общий объем времени, отведенный на выполнение практической работы по учебной дисциплине ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем, составляет в соответствии с учебным планом и рабочей программой– 16 часов.

Методические указания призваны помочь студентам правильно организовать работу и рационально использовать свое время при овладении содержанием учебной

дисциплины ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем, закреплении теоретических знаний и умений.

Распределение часов на выполнение практической работы студентов по разделам и темам учебной дисциплины ОП.01. Основы архитектуры, устройства и функционирования вычислительных систем

Наименование раздела, темы	Количество часов
Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем	**16**
Тема 3.1. Логические основы ЭВМ, элементы и узлы	1
Тема 3.2. Основы построения ЭВМ	4
Тема 3.3. Внутренняя организация процессора	2
Тема 3.7 Основы программирования процессора	8
Тема 3.8 Современные процессоры	1

Перечень рекомендуемой литературы

Основные источники:

1. Колдаев, В.Д. Архитектура ЭВМ: учебное пособие для учрежд. СПО/В.Д.Колдаев, С.А.Лупин С.А. - М.: ФОРУМ: Инфра-М, 2014.
2. Максимов, Н. В. Архитектура ЭВМ и вычислительных систем: учебник для учрежд. СПО/Н.В. Максимов, Т. Л. Партыка, И. И. Попов. - М.: ФОРУМ, 2015.
3. Чекмарев, Ю. В. Вычислительные системы, сети и телекоммуникации. - М.: ДМК-Пресс, 2016.

Дополнительные источники:

1. Сенкевич, А.В. Архитектура ЭВМ и вычислительные системы: учебник для студ. учрежд. СПО. - М.: Академия, 2014.
2. Таненбаум, Э. Архитектура компьютера/Э.Таненбаум, Т.Остин. - СПб. : Питер, 2013.

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.1. Логические основы ЭВМ, элементы и узлы(1 час)

Практическая работа №1

«Работа и особенности логических элементов ЭВМ»

Задачи обучающегося:

1. Изучить основные понятия алгебры логики.
2. Изучить назначение логических элементов и их обозначение на схемах
3. Выполнить задания по теме (решение задач).

Опорные понятия: логические элементы ЭВМ

Планируемый результат:

Студент должен

Знать понятия «логические элементы»

Уметь реализовывать логическую функцию заданную выражением

Необходимое оборудование: учебная литература

Порядок выполнения работы:

Изучите теоретический материал

При записи тех или иных логических выражений используется специальный язык, который принят в математической логике. Основоположником математической логики является великий немецкий математик Готфрид Вильгельм Лейбниц(1646 - 1716 гг.). Он сделал попытку построить универсальный язык, с помощью которого споры между людьми можно было бы разрешать посредством вычислений. На заложенном Лейбницем фундаменте ирландский математик Джордж Буль построил здание новой науки - математической логики, - которая в отличие от обычной алгебры оперирует не числами, а высказываниями. В честь Д.Буля логические переменные в языке программирования Паскаль впоследствии назвали булевскими.

Высказывание- это любое утверждение, относительно которого можно сказать истинно оно или ложно, т.е. соответствует оно действительности или нет. Таким образом по своей сути высказывания фактически являются двоичными объектами и поэтому часто истинному значению высказывания ставят в соответствие 1, а ложному - 0. Например, запись A = 1 означает, что высказывание A истинно.

Высказывания могут быть простыми и сложными. Простые соответствуют алгебраическим переменным, а сложные являются аналогом алгебраических функций. Функции могут получаться путем объединения переменных с помощью логических действий.

Самой простой логической операцией является операция НЕ (по-другому ее часто называют отрицанием, дополнением или инверсией и обозначают NOTX или $\overline{X}$. Результат отрицания всегда противоположен значению аргумента.

Логическая операция НЕ является унарной, т.е. имеет всего один операнд. В отличие от нее, операции И (AND) и ИЛИ (OR) являются бинарными, так как представляют собой результаты действий над двумя логическими величинами.

Операцию НЕ можно задать в виде таблицы

X	$\overline{X}$
0	1
1	0

Логическое И еще часто называют конъюнкцией, или логическим умножением.

Операция И имеет результат «истина» только в том случае, если оба ее операнда истинны. Принято обозначать значком «&»либо «^»

Операцию И можно задать в виде таблицы

X	Y	X^Y
0	0	0
0	1	0
1	0	0
1	1	1

Операция ИЛИ -дизъюнкцией, или логическим сложением. Она дает «истину», если значение «истина» имеет хотя бы один из операндов. Принято обозначать значком «»либо «+». Разумеется, в случае, когда справедливы оба аргумента одновременно, результат по-прежнему истинный. Действительно, когда студентка просит друга подарить ей на день рождения букет цветов или пригласить в кафе, можно без опасении сделать и то, и другое одновременно (впрочем, на практике в таком случае можно ограничиться чем-то одним).
Операцию ИЛИ можно задать в виде таблицы

X	Y	X∨Y
0	0	0
0	1	1
1	0	1
1	1	1

Приведенные выше таблицы значений переменных для логических операций называются таблицами истинности. В них указываются все возможные комбинации логических переменных X и Y, а также соответствующие им результаты операций. Таблица истинности может рассматриваться в качестве одного из способов задания логической функции.

Операции И, ИЛИ, НЕ образуют полную систему логических операций, из которой можно построить сколь угодно сложное логическое выражение.

В вычислительной технике также часто используется операция исключающее ИЛИ (XOR), которая отличается от обыкновенного ИЛИ только при X=1 и Y=l.

Как видно из табл. 1.2, операция XOR фактически сравнивает на совпадение два двоичных разряда. Хотя теоретически основными базовыми логическими операциями всегда называют именно И, ИЛИ, НЕ, на практике по технологическим причинам в качествеосновного логического элемента используется элемент И-НЕ(последняя колонка в табл. 1.2).

X	Y	X XORY	NOT(XANDY)
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0

Таблица 1.2. Дополнительные логические операции
Можно проверить, что на базе элементов И-НЕ могут быть скомпонованы все базовые логические

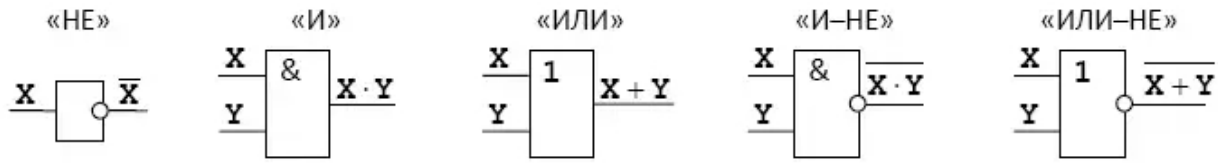
элементы (И, ИЛИ, НЕ), а значит и любые другие, более сложные.

Для упрощения логических выражений используют законы алгебры логики.

Таблица 1.3. Законы алгебры логики

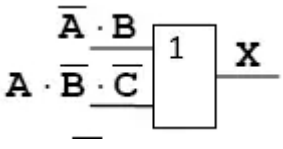
Закон	для «И»	для «ИЛИ»
двойного отрицания	$\overline{\overline{A}} = A$	
исключения третьего	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
операции с константами	$A \cdot 1 = A,\ A \cdot 0 = 0$	$A + 1 = 1,\ \ A + 0 = A$
повторения	$A \cdot A = A$	$A + A = A$
переместительный	$A \cdot B = B \cdot A$	$A + B = B + A$
сочетательный	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
распределительный	$A + B \cdot C = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$
поглощения	$A + A \cdot B = A$	$A \cdot (A + B) = A$
законы де Моргана	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A \cdot B} = \overline{A} + \overline{B}$

В компьютерах все вычисления выполняются с помощью логических элементов (электронных схем), выполняющих логические операции. Обозначения простейших элементов приводиться в таблице (ГОСТ 2.743-91). Обратите внимания, что небольшой кружок на выходе обозначает операцию НЕ (инверсию).

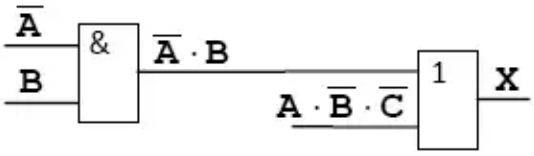


Если нужно составить схему по известному логическому выражению, ее начинают строить с конца.
Находят операцию, которая будет выполняться последней, и ставят на выходе соответствующий логический элемент.
Затем повторяют то же самое для сигналов, поступающих на вход этого элемента. В конце концов, должны остаться только исходные сигналы – переменные в логическом выражении.

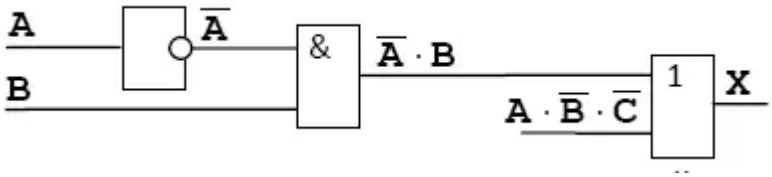
Составим схему, соответствующую выражению
X= $\overline{A}$^B∨A^ $\overline{B}$^ $\overline{C}$



Добавляем элемент И:



Ставим элемент НЕ:



Аналогично разбираем вторую ветку:

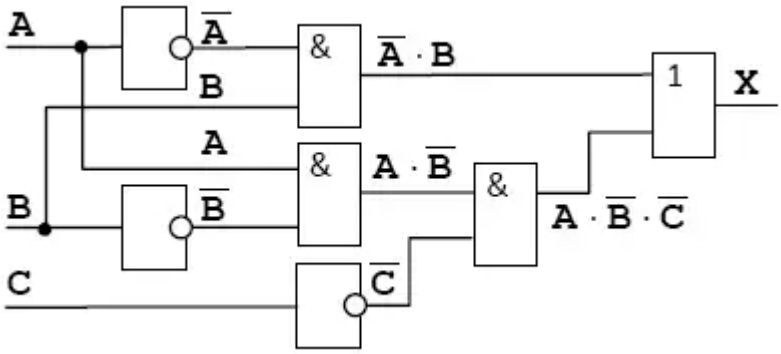


Схема составлена, ее входами являются сигналы A, B и C, а выходом X.

Задание. Используя логические элементы реализовать логическую функцию заданную выражением.

Отчет должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод по работе.

Вопросы для самоконтроля

1. На чём основано выполнение логических операций в ЭВМ?
2. Что такое высказывание?
3. Назовите основные логические операции и приведите их таблицы истинности.
4. Что такое логическое выражение?
5. Каков порядок выполнения операций при вычислении значения логического выражения?
6. Что такое таблица истинности?

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.2. Основы построения ЭВМ узлы (4 часа)

Практическая работа №2

«Изучение внутреннего устройства персонального компьютера. Определение аппаратной конфигурации ПК»

Задачи обучающегося:

Закрепление, углубление и расширение знаний студентов о составе ПК.

– Приобретение умений и навыков работы по сборке и разборкеПК.

– Выработка способности логического мышления, осмыслениюполученных результатов при изучении состава ПК.

Опорные понятия:аппаратная конфигурация вычислительных систем

Планируемый результат:

Студент должен

Знать состав современных вычислительных систем

Уметь производить сборку и разборку ПК

Необходимое оборудование:ПК

Порядок выполнения работы :

Изучите теоретический материал

Компьютер (от англ. computer - вычислитель) представляет собой программируемое электронное устройство, способное обрабатывать данные и производить вычисления, а также выполнять другие задачи манипулирования символами.

Современный компьютер - это программно-аппаратный комплекс, который состоит из аппаратной *(hardware)* и программной *(software)* частей.

Аппаратная часть построена, в основном, с использованием электронных и электромеханических элементов и устройств.

Программная часть необходима для выполнения программ, т.е. заранее заданных, четко определенных последовательностей арифметических, логических и других операций.

Архитектурой компьютера называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.п.

Архитектура компьютера обычно определяется совокупностью ее свойств, существенных для пользователя. Основное внимание при этом уделяется структуре и функциональным возможностям компьютера, которые можно разделить на основные и дополнительные.

Основные функции определяют его назначение: обработка и хранение информации, обмен информацией с внешними объектами.

Дополнительные функции повышают эффективность выполнения основных функций: обеспечивают эффективные режимы работы, диалог с пользователем, высокую надежность и др.

Структура компьютера - это некоторая модель, устанавливающая состав, порядок и принципы взаимодействия входящих в нее компонентов.

Структура основана на общих логических принципах, позволяющих выделить в любом компьютере следующие главные устройства:

Память (запоминающее устройство), состоящую из перенумерованных ячеек;

Процессор, включающий в себя *устройство управления* (УУ) и *арифметико-логическое устройство* (АЛУ);

Устройство ввода;

Устройство вывода.

Эти устройства соединены каналами связи, по которым передается информация.

Персональным компьютером (ПК) называют сравнительно недорогой универсальный микрокомпьютер, рассчитанный на одного пользователя (рис. 1)

Рис. 1. Состав основных блоков ПК

Основные блоки ПК:

Сеть - электрическая сеть, обычно 220 вольт.

Системный блок - содержит основное аппаратное обеспечение (элементы) компьютера.

Монитор - средство для визуального отображения информации.

Клавиатура (мышь) - устройство ввода данных.

Принтер - устройство для вывода текстовой или графической информации на твердый физический носитель

Основные составляющие системного блока ПК 3.1.1.

Процессор (CPU)

Микропроцессор – устройство, отвечающее за выполнение арифметических, логических и операций управления, записанных в машинном коде, реализованный в виде одной микросхемы. В настоящее время наиболее широкое распространение получили платы компаний Intel (рис.2) и AMD, которые различаются по стоимости, производительности, и количеству ядер (1, 2, 4, 6, 8).

Рис. 2. Примеры процессоров

Системная (материнская) плата (англ. Motherboard или mainboard – главная плата) – это сложная многослойная печатная плата, на которой устанавливаются основные компоненты персонального компьютера (рис.3). Как правило, материнская плата содержит разъемы для подключения дополнительных контроллеров.

В типичный состав системной платы обычно входят:

1. Разъемы для подключения питания.
2. Разъем (socket) для подключения процессора.
3. Слоты для установки оперативной памяти.
4. Разъемы для подключения жестких дисков и оптических при- водов (IDE, SATA).
5. USB – разъемы.
6. Разъемы PCI (Peripheral component interconnect – дословно, взаимосвязь периферийных устройств).
7. Разъем для подключения видеокарты (PCI – EXPRESS).
8. Северный мост (Northbridge) - обеспечивает бесперебойную передачу данных в связке «Центральный процессор - Опера- тивная память – Графический адаптер».
9. Южный мост (в платах Intel называется ICH - I/O Controller Hub) обеспечивает передачу данных между портами USB, оп- тическими приводами и жесткими дисками, также отвечает за устройства ввода: клавиатуру, мышь.

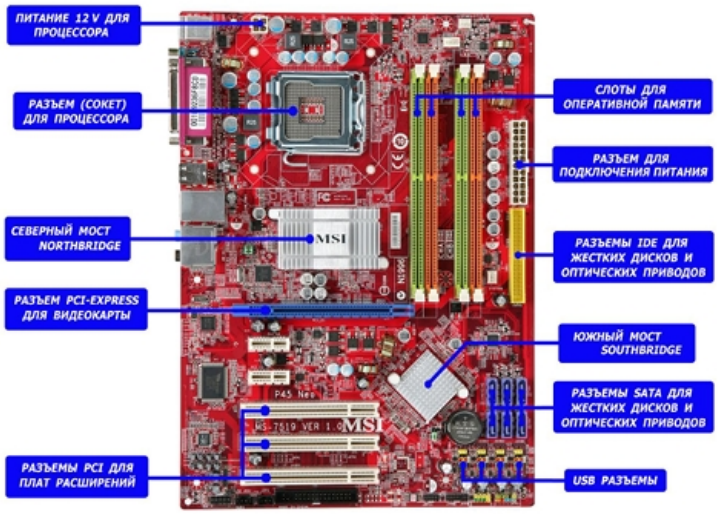


Рис. 3. Системная плата

Оперативная память (оперативное запоминающее устройство – ОЗУ) предназначена для временного хранения данных и команд, необходимых процессору для выполнения им операций.

Виды оперативной памяти:

1. DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, атакже является самым старым видом оперативной памяти, которую можно еще сегодня купить (рис.4).

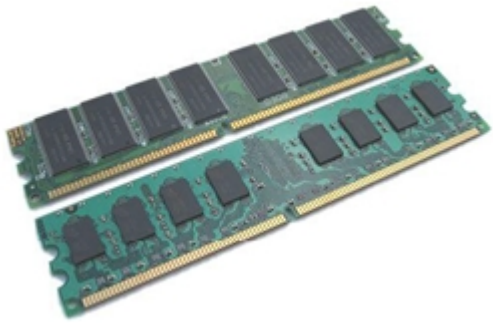


Рис. 4. DDR

2. DDR2 SDRAM (double-data-rate two synchronous dynamic random access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, второе поколение (рис. 5). Тоже устаревший вид оперативной памяти.

Рис. 5. DDR2

3. DDR3 SDRAM (double-data-rate three synchronous dynamic random access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, третье поколение (рис. 6). Широко используется в современных компьютерах.

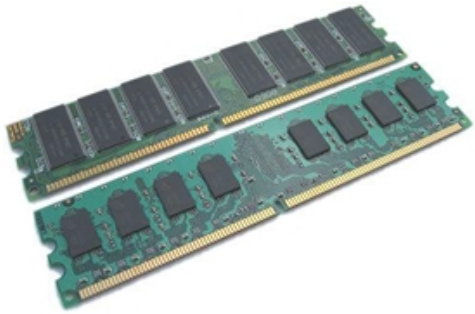


Рис. 6. DDR3

4. DDR4 SDRAM (double-data-rate four synchronous dynamic ran- dom access memory) – новый тип оперативной памяти, являющийся эволюционным развитием предыдущих поколений DDR.

Рис. 7. DDR4

Все виды памяти не совместимы друг с другом

Видеокарта (от англ. Videocard – графическая плата, графический адаптер, видеоадаптер) – устройство, преобразующее изображение, находящееся в памяти компьютера, в видеосигнал для монитора (рис. 8). Дополнительная видеокарта необходима лишь при условии, что ПК используется для обработки сложной графики.

Рис. 8. Видеокарта

Накопитель на жестких магнитных дисках (НЖМД), жесткий диск, винт, хард, харддиск, HDD (Hard Disk Drive) – энергонезависимое, перезаписываемое компьютерное запоминающее устройство.

Является основным накопителем данных практически во всех современных компьютерах. При выборе жесткого диска необходимо учитывать его интерфейс. В настоящее время наиболее распространен интерфейс SATA(Serial ATA)(рис. 9), но для модернизации старых ПК необходимо использовать жесткий диск с интерфейсом IDE (Integrated Drive Electronics), либо позднее он стал называться ATA(Advanced Technology Attachment) (рис. 10).

Рис. 9. HDD с интерфейсом SATA

Рис. 10. HDD с интерфейсом IDE

Звуковая плата (англ. sound card – звуковая карта или музыкальная плата) – это плата,

которая позволяет работать со звуком на компьютере. В настоящее время звуковые карты бывают как встроенными в материнскую плату, так и отдельными платами расширения или как внешними устройствами.

Рис. 11. Звуковая карта (плата расширения)

Оптический привод – устройство чтения, записи дисков (рис.12).

Можно выделить несколько основных типов данных устройств:

1. CD-ROM – данный привод способен читать только CD.

2. CD-RW – позволяет не только считывать информацию с обычных компакт-дисков, но и записывать ее на CD-R и CD-RW.

3. DVD-ROM - устройство, способное читать компакт-диски DVD.

4. DVD-CD-RW Combo - так называемый Combo-драйв, который сочетает в себе функции таких устройств, как DVD-ROM и CD- RW и, соответственно, может записывать диски CD-R и CD-RW, считывать, как обычные CD, так и DVD.

5. DVD-RW –позволяет не только читать диски CD/DVD, но и за- писывать как обычные CD-R/CD-RW-носители, так и куда более ёмкие DVD-R/DVD-RW/DVD+R/DVD+RW.

6. Blu-Ray, BD (blue ray — синий луч, и Disk - диск) — формат оптического носителя, используемый для записи и хранения цифровых данных, включая видео высокой чёткости с повышенной плотностью.

Рис. 12. Оптический привод

Блок питания предназначен для снабжения элементов системного блока компьютера электрической энергией (рис. 13). Мощность блока питания – это одно из важнейших свойств его параметров. Чем мощнее система, тем больше ее электропитание.

Рис. 13. Компьютерные блоки питания

Задание на лабораторную работу

Выполнить разборку ПК с целью изучения каждого элемента (устройства) и далее осуществить сборку ПК. Номер тренировочного стенда выдается преподавателем.

1. Произвести разборку ПК. Выложить все элементы на рабочий стол.

2. Изучить каждый элемент ПК, записать в таблицу 1 название, фирму производителя, основные характеристики элементов и разъем для подключения.

Таблица 1 – элементы системного блока

Название элемента	Фирма производитель и модель	Основные характеристики	Разъемы

3.Зарисовать схему соединения элементов ПК.

4. Осуществить сборку ПК.

5. Показать преподавателю собранный ПК.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

а) титульный лист;

б) описание хода выполнения работы;

в) таблицу с описание всех элементов ПК;

г) схему соединения элементов ПК;

д) заключение по выполненной работе;

е) ответы на контрольные вопросы.

Контрольные вопросы

1. Что понимается под персональным компьютером?
2. Какие основные блоки содержит ПК?
3. Что называется архитектурой компьютера?

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.2. Основы построения ЭВМ узлы (4 часа)

Практическая работа №3

«Подключение сетевых устройств, установка и настройка оборудования, работа с вычислительной сетью»

Задачи обучающегося:

1.Изучить процесс подключение сетевого адаптера

2.Исследовать процесс настройки адаптера в ОС

Опорные понятия: сетевые устройства

Планируемый результат:

Студент должен

Уметь подключать сетевые адаптеры и настраивать их.
Необходимое оборудование: ПК, сетевое оборудование.

Порядок выполнения работы :

Выключить компьютер.

Отключить питание.

Снять крышку системного блока и установить адаптер в соответствующий разъем на материнской плате.

Включить компьютер дождаться загрузки операционной системы.

При загрузке Windows должна обнаружить новый адаптер. Если с устройством поставляется диск с драйверами, вставить его в дисковод.

Если адаптер автоматически не определяется, необходимо воспользоваться инструкциями по установке оборудования из лабораторной работы № 3.

При необходимости установить сетевые службы. Поддержка сети обычно настраивается при установке Windows. Вручную поддержка сети TCP/IP настраивается в компоненте Сетевые соединения. Для этого необходимо войти в систему компьютера под учетной записью Administrator (Администратор).

Выбрать Пуск – Программы – Стандартные - Связь и щелкнуть значок Сетевые подключения. Дважды щелкнуть значок нужного подключения.

В диалоговом окне Состояние, щелкнуть кнопку Свойства. Откроется диалоговое окно Подключение по локальной сети — Свойства. Если протокола TCP/IP нет в списке установленных компонентов, его надо установить. Щелкнуть кнопку Установить и выбрать в списке «Протокол» и щелкните кнопку «Добавить». В диалоговом окне «Выбор сетевого протокола», выбрать TCP/IP и щелкнуть ОК.

В диалоговом окне Подключение по локальной сети — Свойства, убедиться, что флажок напротив TCP/IP установлен, и щелкнуть ОК.

Настройка локальных подключений. Локальные подключения создаются автоматически, если компьютер подключен к сети и на нем установлен сетевой адаптер. Если на компьютере несколько адаптеров, для каждого создается отдельное подключение. Если сетевые подключения недоступны, надо подключить компьютер к сети или создать другой тип подключений.

Содержание отчета:

название работы, цель работы, оборудование, порядок подключения и установки сетевого адаптера.

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.3. Внутренняя организация процессора (2 часа)

Практическая работа №4
«Построение последовательности машинных операций для реализации простых вычислений»

Задачи обучающегося:
1.Ознакомиться с циклом работы процессора
Опорные понятия:логическая структура ЦПУ
Планируемый результат:
Уметь составлять программу в машинных кодах.
Необходимое оборудование:ПК, дополнительное ПО.

1 Структура ЭВМ

Моделируемая ЭВМ включает процессор, оперативную (ОЗУ) и сверхоперативная память, устройство ввода (УВв) и устройство вывода (УВыв). Процессор в свою очередь, состоит из центрального устройства управления (УУ), арифметического устройства (АУ) и системных регистров (CR, PC, и др.). Структурная схема ЭВМ показана на рис. 1.

В ячейках ОЗУ хранятся команды и данные. Емкость ОЗУ составляет 1000 ячеек. По сигналу MWr выполняется запись содержимого регистра данных (MDR) в ячейку памяти с адресом, указанным в регистре адреса (MAR) По сигналу MRd происходит считывание - содержимое ячейки памяти с адресом, содержащимся в MAR, передается в MDR.

Сверхоперативная память с прямой адресацией содержит десять регистров общего назначения R0-R9. Доступ к ним осуществляется (аналогично доступу к ОЗУ) через регистры RAR и RDR.

АЛУ осуществляет выполнение одной из арифметических операций, определяемой кодом операции (COP), над содержимым аккумулятора (Acc) и регистра операнда (DR). Результат операции всегда помещается в Acc. При завершении выполнения операции АУ вырабатывает сигналы признаков результата: Z (равен 1, если результат равен нулю); S (равен 1, если результат отрицателен); OV (равен 1, если при выполнении операции произошло переполнение разрядной сетки). В случаях, когда эти условия не выполняются, соответствующие сигналы имеют нулевое значение.

В модели ЭВМ предусмотрены внешние устройства двух типов. Во-первых регистры IR и OR, которые могут обмениваться с аккумулятором с помощью безадресных команд IN (Acc := IR) и OUT (OR := Acc). Во-вторых, это моделей внешних устройств, которые могут подключаться к системе и взаимодействовать с ней в соответствии с заложенными в моделях алгоритмами. Каждое внешнее устройство имеет ряд программно-доступных регистров, может иметь собственный *обозреватель* (окно видимых элементов).

УУ осуществляет выборку команд из ОЗУ в последовательности, определяемой естественным порядком выполнения команд (т. е. в порядке возрастания адресов команд в ОЗУ) или командами передачи управления; выборку из ОЗУ операндов, задаваемых адресами команды; инициирование выполнения операции, предписанной командой; останов или переход к выполнению с следующей команды.

В качестве сверхоперативной памяти в модель включены регистры общего назначения (РОН), и может подключаться модель кэш-памяти.

В состав УУ ЭВМ входят:

- PC - счетчик адреса команды, содержащий адрес текущей команды;
- CR - регистр команды, содержащий код команды;
- RB - регистр базового адреса, содержащий базовый адрес;

- SP - указатель стека, содержащий адрес верхушки стека;
- RA - регистр адреса, содержащий исполнительный адрес при косвенной адресации.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB - 3 разряда.

2 Представление данных в модели

Данные в ЭВМ представляются в формате, показанном на рис. 2. Это целые десятичные числа, изменяющиеся в диапазоне "-99 999 ... +99 999", содержащие знак и 5 десятичных цифр.

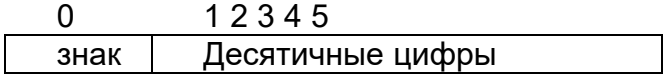


Рис.2 Форматы десятичных данных

Старший разряд слова данных используется для кодирования знака: плюс (+) изображается как О, минус (-) - как 1. Если результат арифметической операции выходит за пределы указанного диапазона, то говорят, что произошло переполнение разрядной сетки. АЛУ в этом случае вырабатывает сигнал переполнения OV = 1. Результатом операции деления является целая часть частного. Деление на ноль вызывает переполнение

3 Система команд

При рассмотрении системы команд ЭВМ обычно анализируют три аспекта: форматы, способы адресации и систему операций.

3. 1. Форматы команд

Большинство команд учебной ЭВМ являются одноадресными или безадресными, длиной в одно машинное слово (6 разрядов). Исключение оставляют двухсловные команды с непосредственной адресацией и команда MOV, являющаяся двухадресной.

В форматах команд выделяется три поля:

- два старших разряда [0:1] определяют код операции COP;
- разряд 2 может определять тип адресации (в одном случае (формат 5а) он определяет номер регистра);

- разряды [3:5] могут определять прямой или косвенный адрес памяти, номер регистра (В команде MOV номера двух регистров), адрес перехода или короткий непосредственный операнд. В двухсловных командах непосредственный операнд занимает поле [6: 11].

Полный список форматов команд показан на рис.3, где приняты следующие обозначения:
COP - код операции;
ADR - адрес операнда в памяти;
ADC - адрес перехода;
I - непосредственный операнд;
R, Rl, R2 - номер регистра;
TA - тип адресации;
X-разряд не используется.

Номер формата	0 1		3 4 5	
1	COP		X X X	
2	COP	TA	ADR	
3	COP	TA	X X R	
3a	COP	TA	XR1 R2	6 11
4	COP		X X X	I
5	COP		ADC	
5a	COP		ADC	

Рис. 3. Форматы команд учебной программы.

3.2. Способы адресации

В ЭВМ принято различать пять основных способов адресации: *прямая, косвенная, непосредственная, относительная, безадресная.*

Каждый способ имеет разновидности. В модели учебной ЭВМ реализованы семь способов в адресации, приведенные в табл1.

Таблица 1. Адресация в командах учебной ЭВМ.

Код TA	Тип адресации	Исполнительный адрес
0	Прямая (регистровая)	ADR(R)
1	Непосредственная	-
2	Косвенная	ОЗУ(ADR)[3:5]
3	Относительная	ADR+RB
4	Косвенно-регистровая	РОН(R)[3:5]
5	Индексная с постинкрементом	РОН(R)[3:5], R:= R + I
6	Индексная с преддекрементом	R:= R -1, РОН(R)[3 :5]

3.3. Система операций

Система команд учебной ЭВМ включает команды следующих классов:

- *арифметико-логические и специальные*: сложение, вычитание, умножение, деление;
- *пересылки и загрузки:* чтение, запись, пересылка (из регистра в регистр), помещение в стек, извлечение из стека, загрузка указателя стека, загрузка базового регистра;
- *ввода/вывода:* ввод, вывод;
- *передачи управления:* безусловный и шесть условных переходов, вызов подпрограммы, возврат из подпрограммы, цикл, программное прерывание, возврат из прерывания.

- системные: пустая операция, разрешить прерывание, запретить прерывание, стон.

Список команд учебной ЭВМ приведен в табл. 4 и 6.

4. Состояния и режимы работы ЭВМ

Ядром УУ ЭВМ является управляющий автомат (УА), вырабатывающий сигналы управления, которые инициируют работу АЛУ, РОН, ОЗУ и УВВ, передачу информации между регистрами устройств ЭВМ и действия над содержимым регистров УУ.

ЭВМ может находиться в одном из двух состояний: Останов и Работа.

В состояние **Работа** ЭВМ переходит по действию команд Пуск или Шаг. Команда Пуск запускает выполнение программы, представляющую собой последовательность команд, записанных в ОЗУ, в автоматическом режиме до команды HLT или точки останова. Программа выполняется по командам, начиная с ячейки ОЗУ, на которую показывает PC, причем изменение состояний объектов модели отображается в окнах обозревателей.

В состояние **Останов** ЭВМ переходит по действию команды Стоп или автоматически в зависимости от установленного режима работы.

Команда **Шаг**, в зависимости от установленного режима работы, запускает выполнение одной команды или одной микрокоманды (если установлен **Режим микрокоманд**), после чего переходит в состояние **Останов**.

В состоянии **Останов** допускается просмотр и модификация объектов модели: регистров процессора и РОН, ячеек ОЗУ, устройств ввода/вывода. В процессе модификации ячеек ОЗУ и РОН можно вводить данные для программы в ячейки ОЗУ - программу в кодах. Кроме того, в режиме **Останов** можно менять параметры модели и режимы ее работы, вводить и/или редактировать программу в мнемокодах, ассемблировать мнемокоды, выполнять стандартные операции с файлами.

5. Интерфейс ЭВМ

Для решения с помощью ЭВМ некоторой задачи должна быть разработана программа. Программа на языке ЭВМ представляет собой последовательность команд. Код каждой команды определяет выполняемую операцию, тип адресации и адрес. Выполнение программы, записанной в памяти ЭВМ, осуществляется последовательно по командам в порядке возрастания адресов команд или в порядке, определяемом командами передачи управления.

Для того чтобы получить результат выполнения программы, пользователь должен:

- ввести программу в память ЭВМ;
- определить, если это необходимо, содержимое ячеек ОЗУ и РОН, содержащих исходные данные, а также регистров IR и BR;
- установить в PC стартовый адрес программы;
- перевести модель в режим Работа.

Каждое из этих действий выполняется посредством ***интерфейса модели***.

Окна основных обозревателей системы

Окно Процессор

Окно Процессор обеспечивает доступ ко всем регистрам и флагам процессоров.

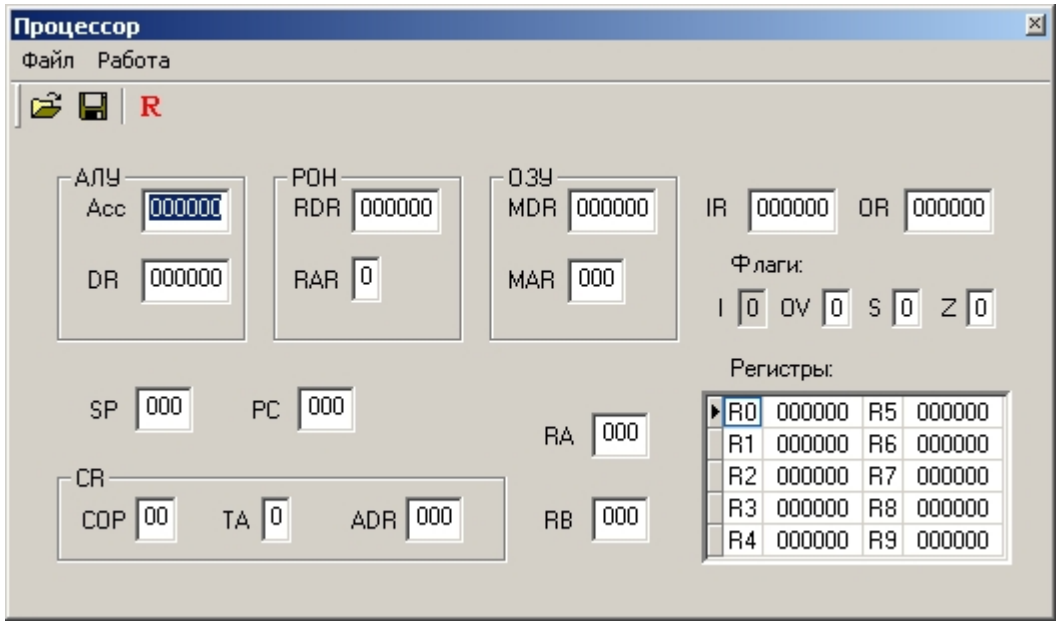


Рис. 8.4. Окно Процессор

Программно-доступные регистры и флаги:

Acc — аккумулятор;
PC — счетчик адреса команды, содержащий адрес текущей команды;
SP — указатель стека, содержащий адрес верхушки стека;
RB — регистр базового адреса, содержащий базовый адрес;
RA — регистр адреса, содержащий исполнительный адрес при косвенной адресации;
IR — входной регистр;
OR — выходной регистр;
I — флаг разрешения прерываний.

Системные регистры и флаги:

DR — регистр данных АЛУ, содержащий второй операнд;
MDR — регистр данных ОЗУ;
MAR — регистр адреса ОЗУ;
RDR — регистр данных блока РОН;
RAR — регистр адреса блока РОН;
CR — регистр команд, содержащий поля:
COP — код операции;
TA — тип адресации;
ADR — адрес или непосредственный операнд;
Z — флаг нулевого значения Acc;
S — флаг отрицательного значения Асе;
OV — флаг переполнения.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB — 3 разряда. В окне **Процессор** отражаются текущие значения регистров и флагов, причем в состоянии **Останов** все регистры, включая регистры блока РОН, и флаги (кроме флага 1) доступны для непосредственного редактирования.

Элементы управления окна **Процессор** включают меню и кнопки, вызывающие команды:

- **Сохранить;**
- **Загрузить;**
- **Reset;**
- **ResetR0-R9** (только команда меню **Работа).**

Команды **Сохранить, Загрузить** позволяют сохранить текущее значение регистров и флагов процессора в файле и восстановить состояние процессора из файла. Команда **Reset**и кнопка R устанавливают все регистры (в т. ч. блок РОН) в начальное (нулевое) значение. Содержимое ячеек

памяти при этом не меняется. Выполняемая лишь из меню **Работа** команда ResetR0-R9 очищает только регистры блока РОН.

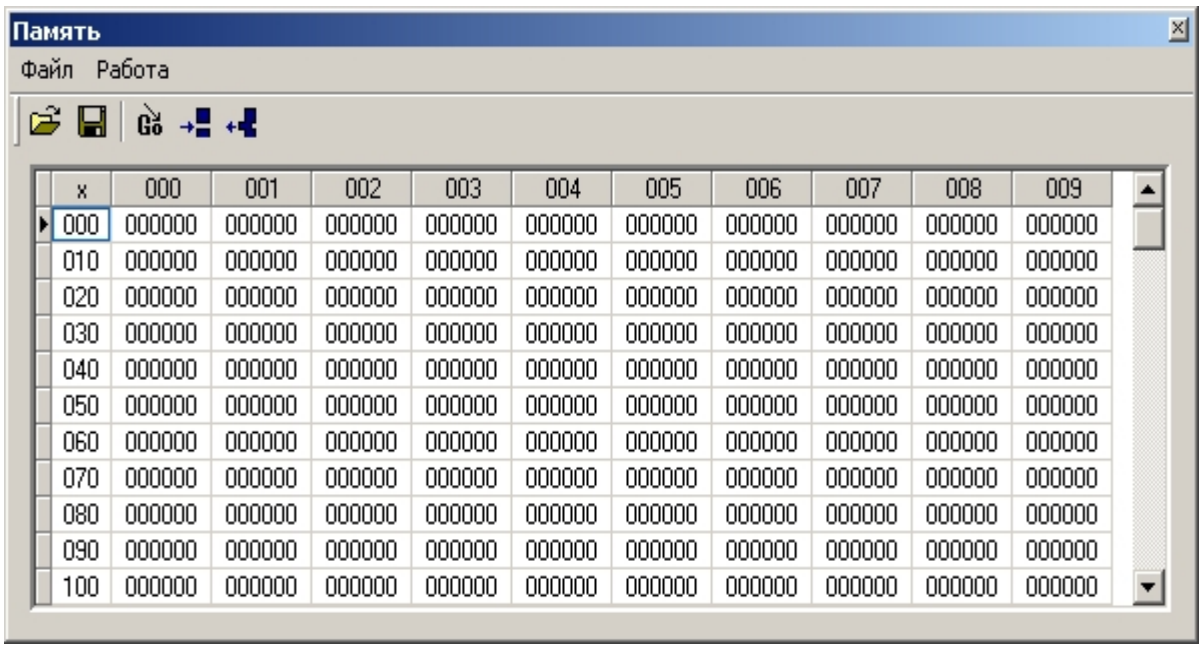
Окно *Память*

Окно **Память** (рис. 8.5) отражает текущее состояние ячеек ОЗУ. В этом окне допускается редактирование содержимого ячеек, кроме того, предусмотрена возможность выполнения (через меню или с помощью кнопок панели инструментов) пяти команд: **Сохранить, Загрузить, Перейти к, Вставить**, **Убрать.**

Команды **Сохранить, Загрузить** во всех окнах, где они предусмотрены, работают одинаково — сохраняют в файле текущее состояние объекта (в данном случае памяти) и восстанавливают это состояние из выбранного файла, причем файл в каждом окне записывается по умолчанию с характерным для этого окна расширением.

Команда **Перейти к** открывает диалоговое окно, позволяющее перейти на заданную ячейку ОЗУ.

Команда **Убрать** открывает диалог, в котором указывается диапазон ячеек с mпо n. Содержимое ячеек в этом диапазоне теряется, а содержимое ячеек [(n +1): 999] перемещается в соседние ячейки с меньшими адресами. Освободившиеся ячейки с адресами 999, 998, ... заполняются нулями.



x	000	001	002	003	004	005	006	007	008	009
000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
010	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
020	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
030	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
040	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
050	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
060	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
070	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
080	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
090	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
100	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000

Рис. 8.5. Окно Память

Команда **Вставить,** позволяющая задать номера ячеек, перемещает содержимое всех ячеек, начиная от *m* -й на *n* - *m* позиций в направлении больших адресов, ячейки заданного диапазона *[m:n]* заполняются нулями, а содержимое последних ячеек памяти теряется.

Окно *Текст программы*

Окно **Текст программы** (рис. 8.6) содержит стандартное поле текстового редактора, в котором можно редактировать тексты, загружать в него текстовые файлы и сохранять подготовленный текст в виде файла.

Команды меню **Файл:**

Новая — открывает новый сеанс редактирования;

Загрузить — открывает стандартный диалог загрузки файла в окно редактора;

Сохранить — сохраняет файл под текущим именем;

Сохранить как — открывает стандартный диалог сохранения файла;

Вставить — позволяет вставить выбранный файл в позицию курсора.

Все перечисленные команды, кроме последней, дублированы кнопками на панели инструментов окна. На той же панели присутствует еще одна кнопка — **Компилировать,** которая запускает процедуру ассемблирования текста в поле редактора.

Ту же процедуру можно запустить из меню **Работа.** Команда **Адрес вставки** позволяет задать адрес ячейки ОЗУ, начиная с которой программа будет размещаться в памяти. По умолчанию этот адрес принят равным 0.

Ниже области редактирования в строку состояния выводится позиция текущей строки редактора — номер строки, в которой находится курсор.

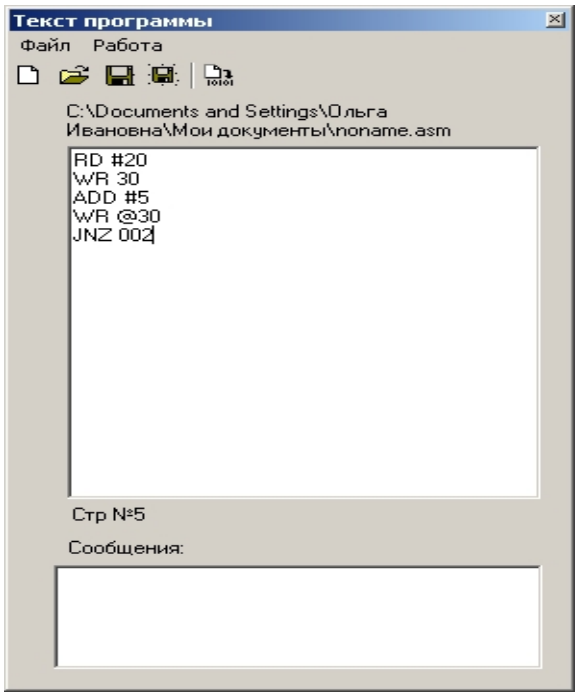


Рис. 8.6. Окно Текст программы

В случае обнаружения синтаксических ошибок в тексте программы диагностические сообщения процесса компиляции выводятся в окно сообщений, запись в память кодов (даже безошибочного начального фрагмента программы) не производится.

После исправления ошибок и повторной компиляции выдается сообщение об отсутствии ошибок, о расположении и размере области памяти, занятой под ассемблированную программу.

Окно ***Программа***

Окно **Программа** (рис. 8.7) отображает таблицу, имеющую 300 строк и 4 столбца. Каждая строка таблицы соответствует дизассемблированной ячейке ОЗУ. Второй столбец содержит адрес ячейки ОЗУ, третий — дизассемблированный мнемокод, четвертый — машинный код команды. В первом столбце может помещаться указатель — на текущую команду (текущее значение PC) и точка останова — красная заливка ячейки.

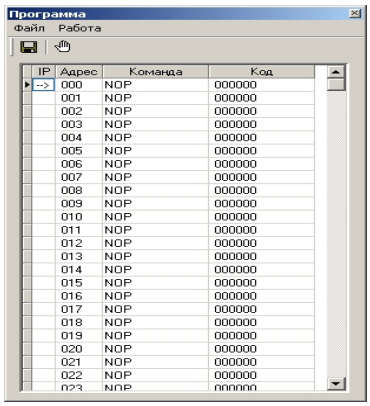


Рис. 8.7. Окно Программа

Окно **Программа** позволяет наблюдать процесс прохождения программы. В этом окне ничего нельзя редактировать. Органы управления окна позволяют сохранить содержимое окна в виде текстового файла, выбрать начальный адрес области ОЗУ, которая будет дизассемблироваться

(размер области постоянный — 300 ячеек), а также установить/снять точку останова. Последнее можно проделать тремя способами: командой **Точка останова** из меню **Работа,** кнопкой на панели инструментов или двойным щелчком мыши в первой ячейке соответствующей строки. Характерно, что прочитать в это окно ничего нельзя. Сохраненный текстовый asm-файл можно загрузить в окно **Текст программы,** ассемблировать его и тогда дизассемблированное значение заданной области памяти автоматически появится в окне **Программа.** Такую процедуру удобно использовать, если программа изначально пишется или редактируется непосредственно в памяти в машинных кодах. Начальный адрес области дизассемблирования задается в диалоге командой **Начальный адрес** меню **Работа.**

Окно *Микрокомандный уровень*

Окно **Микрокомандный уровень** (рис. 8.8) используется только в режиме микрокоманд, который устанавливается командой **Режим микрокоманд** меню **Работа.** В это окно выводится мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

Шаговый режим выполнения программы или запуск программы в автоматическом режиме с задержкой командного цикла позволяет наблюдать процесс выполнения программы на уровне микрокоманд.

Если открыть окно **Микрокомандный уровень,** не установив режим микрокоманд в меню **Работа,** то после начала выполнения программы в режиме **Шаг** (или в автоматическом режиме) в строке сообщений окна будет выдано сообщение "Режим микрокоманд неактивен".

Рис. 8.8. Окно Микрокомандный уровень

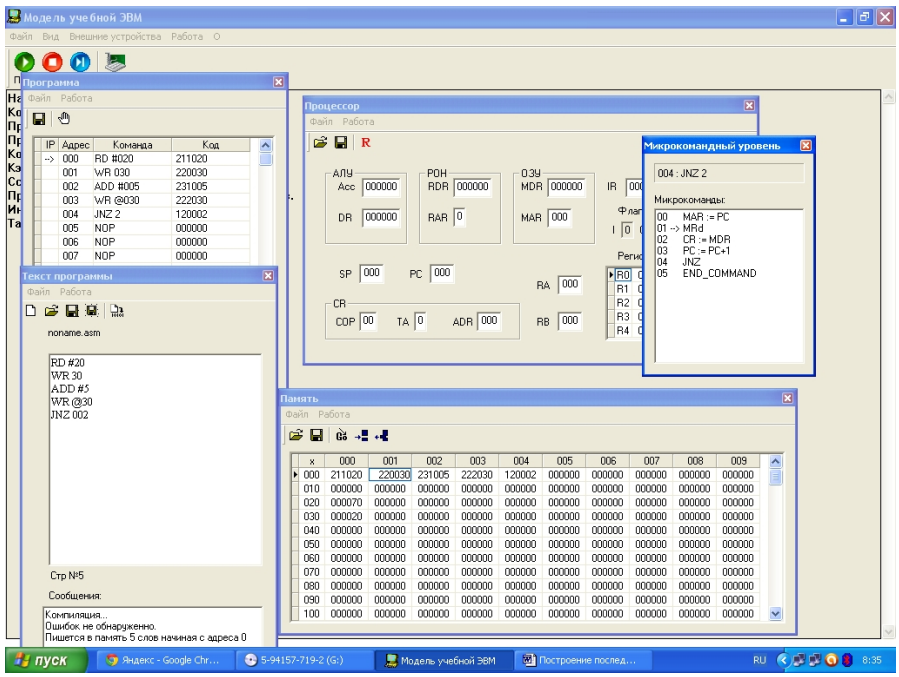


Рис. 8.9. Окна в режиме микрокомандного уровня

Вспомогательные таблицы

В данном разделе представлены вспомогательные таблицы (табл. 8.4—8.8) для работы с

моделью учебной ЭВМ.

Таблица 8.4**.** Таблица команд учебной ЭВМ

Мл/Ст.	0	1	2	3	4
0	NOP	JMP		MOV	
1	IN	JZ	RD	RD	RDI
2	**OUT**	JNZ	WR	WR	
3	IRET	JS	ADD	ADD	ADI
4	WRRB	JNS	SUB	SUB	SBI
5	WRSP	JO	MUL	MUL	MULI
6	PUSH	JNO	DIV	DIV	DIVI
7	POP	JRNZ	IN		
8	RET	INT	EI	OUT	
9	HLT	CALL	DI		

Таблица 8.5.Типы адресации, их коды и обозначение

Обозначение	Код	**Тип адресации**	**Пример команды**
	0	Прямая (регистровая)	ADD **23** (ADD **R3)**
#	1	Непосредственная	ADD **#33**
@	2	Косвенная	ADD **@33**
[ ]	3	Относительная	ADD [33]
@R	4	Косвенно-регистровая	ADD @R3
@R+	5	Индексная с постинкрементом	ADD @R3+
-@R	6	Индексная с преддекрементом	ADD -@R3

В табл. 8.6 приняты следующие обозначения:

DD— данные, формируемые командой в качестве (второго) операнда: прямо или косвенно адресуемая ячейка памяти или трехразрядный непосредственный операнд;

R* — содержимое регистра или косвенно адресуемая через регистр ячейка памяти;

ADR* — два младших разряда ADR поля регистра CR;

V — адрес памяти, соответствующий вектору прерывания;

M(*) — ячейка памяти, прямо или косвенно адресуемая в команде; I — пятиразрядный непосредственный операнд со знаком.

Таблица 8.6. Система команд учебной ЭВМ

КОП	**Мнемокод**	**Название**	**Действие**
00	NOP	Пустая операция	Нет
01	IN	Ввод	Acc<-IR
02	OUT	Вывод	OR <- Acc
03	I RET	Возврат из прерывания	FLAGS.PC <- M(SP); INC(SP)
04	WRRB	Загрузка RB	RB <- CR[ADR]
05	WRSP	Загрузка SP	SP <- CR[ADR]
06	PUSH	Поместить в стек	DEC(SP); M(SP) <- R
07	POP	Извлечь из стека	R ->M(SP); INC(SP)
08	RET	Возврат	PC -> M(SP); INC(SP)
09	HLT	Стоп	Конец командных циклов
10	JMP	Безусловный переход	PC <- CR[ADR]
11	JZ	Переход, если 0	if Acc = 0 then PC <- CR[ADR]

12	JNZ	Переход, если не 0	if Acc <>0 then PC <-CR[ADR]
13	JS	Переход, если отрицательно	if Acc < 0 then PC <- CR[ADR]
14	JNS	Переход, если положительно	if Acc > 0 then PC <- CR[ADR]
15	JO	Переход, если переполнение	if |Acc| > 99999 then PC <- CR[ADR]
16	JNO	Переход, если нет переполнения	if |Acc| < =99999 then PC <- CR[ADR]
17	JRNZ	Цикл	DEC(R); if R > 0 then PC <- CR[ADR]
18	INT	Программное прерывание	DEC(SP); M(SP) <- FLAGS.PC; PC <- M(V)
19	CALL	Вызов подпрограммы	DEC(SP); M(SP) <- PC; PC <- CR(ADR)
20	Нет		
21	RD	Чтение	Acc <- DD
22	WR	Запись	M(*) <- Acc
23	ADD	Сложение	Acc <- Acc + DD
24	SUB	Вычитание	Acc <- Acc – DD
25	MUL	Умножение	Acc <- Acc x DD
26	DIV	Деление	Acc <— Acc/DD
27	Нет		
28	EI	Разрешить прерывание	IF<- 1
29	DI	Запретить прерывание	IF<-0
30	MOV	Пересылка	Rl <-R2
31	RD	Чтение	Acc <- R*
32	WR	Запись	R* <- Acc
33	ADD	Сложение	Acc<-Acc + R*
34	SUB	Вычитание	Acc <- Acc - R*
35	MUL	Умножение	Acc <- Acc x R*
36	DIV	Деление	Acc <- Acc/R*
37	IN	Ввод	Acc <- ВУ(CR[ADR*])
38	OUT	Вывод	ВУ(CR[ADR*]) <- Acc
39	Нет		
40	Нет		
41	RDI	Чтение	Acc <-I
42	Нет		
43	ADI	Сложение	Acc<— Acc + I
44	SBI	Вычитание	Acc<- Acc - I
45	MULI	Умножение	Acc<- Acc x I
46	DIVI	Деление	Acc<- Acc/I

Ввод программы может осуществляться как в машинных кодах непосредственно в память модели, так и в мнемокодах в окно **Текст программы** с последующим ассемблированием.

Для этого необходимо ввести в память ЭВМ и выполнить в режиме **Шаг** некоторую последовательность команд (определенную вариантом задания) и зафиксировать все изменения на уровне программно-доступных объектов ЭВМ, происходящие при выполнении этих команд.

Практическая часть

В настоящей лабораторной работе будем программировать ЭВМ в машинных кодах.

Пример

Дана последовательность мнемокодов, которую необходимо преобразовать в машинные коды, занести в ОЗУ ЭВМ, выполнить в режиме **Шаг** и зафиксировать изменение состояний программно-доступных объектов ЭВМ (табл. 8.7)-

Таблица 8.7 . Команды и коды

Последовательность	**Значения**				
Команды	RD#20	WR30	ADD #5	WR@30	JNZ 002
Коды	21 1 020	22 0 030	23 1 005	22 2 030	12 0002

Введем полученные коды последовательно в ячейки ОЗУ, начиная с адреса 000. Выполняя команды в режиме Шаг, будем фиксировать изменения программно-доступных объектов (в данном случае это Асе, РС и ячейки ОЗУ 020 и 030) в табл. 8.8

Таблица 8.8. Содержимое регистров

PC	Ace	M(30)	M(20)	**PC**	**Ace**	M(30)	M(20)
000	**000000**	**000000**	**000000**	**004**			**000025**
001	**000020**			**002**			
002		**000020**		**003**	**000030**		
003	**000025**			**004**			**000030**

Задание

1. Ознакомиться с архитектурой ЭВМ.
2. Записать в ОЗУ "программу", состоящую из пяти команд— варианты задания выбрать из табл. 9.3. Команды разместить в последовательных ячейках памяти.
3. При необходимости установить начальное значение в устройство ввода IR.
4. Определить те программно-доступные объекты ЭВМ, которые будут изменяться при выполнении этих команд.
5. Выполнить в режиме **Шаг** введенную последовательность **команд (в режиме микрокоманд!!!!)**, фиксируя изменения значений объектов, определенных в п. 4.
6. Оформить отчет согласно указанным требованиям к нему.

Содержание отчета

1. Формулировка варианта задания.
2. Машинные коды команд, соответствующих варианту задания.
3. Последовательность микрокоманд для каждой команды.
4. Для каждой команды результаты выполнения последовательности команд в форме таблицы

Код микрокоманды	АЛУ		РОН		ОЗУ		SP	PC	RA	CR			RA	M(20)	M(30)
	Acc	DR	RDR	RAR	MDR	MAR				COP	TA	ADR			

Таблица 8.9. Варианты задания 1

№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5

1.	**000007**	**IN**	**MUL #2**	WR10	**WR** 010	JNS 001
2.	**X**	**RD #17**	SUB **#9**	**WR16**	**WR** 016	JNS 001
3.	**100029**	**IN**	ADD **#16**	**WR8**	WR08	**JS** 001
4.	**X**	**RD #2**	**MUL #6**	**WR 11**	**WR 011**	JNZ 00
5.	**000016**	**IN**	WR8	**DIV #4**	**WR 08**	JMP 002
6.	**X**	**RD #4**	**WR 11**	**RD 011**	ADD #330	**JS** 000
7.	**000000**	**IN**	**WR9**	**RD @9**	**SUB#1**	**JS** 001
8.	**X**	**RD 4**	SUB #8	**WR8**	**WR @8**	JNZ 001
9.	**100005**	**IN**	**ADD #12**	**WR 10**	**WR @10**	**JS** 004
10.	**X**	**RD 4**	ADD #15	**WR 13**	**WR @13**	JMP 001
11.	**000315**	**IN**	SUB #308	**WR11**	**WR @11**	JMP 001
12.	**X**	**RD** #988	ADD #19	**WR9**	**WR @9**	JNZ 001
13.	**000017**	**IN**	WR11	**ADD 11**	**WR @11**	**JMP 002**
14.	**X**	**RD #5**	MUL #9	**WR10**	**WR @10**	JNZ 001

Контрольные вопросы

1. Что такое система команд ЭВМ?
2. Перечислите регистры процессора.
3. Как проходит выполнение цикла команды в регистрах процессора?
4. Какие классы команд представлены в модели?
5. Какие действия выполняют команды передачи управления?
6. Какие способы адресации операндов применяются в командах ЭВМ?

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.7 Основы программирования процессора (8 часов)

Практическая работа №5

«Программирование арифметических и логических команд»

Задачи обучающегося:

1.Углублениезнаний по структуре и принципам работы процессора

Опорные понятия:логическая структура памяти

Планируемый результат:

Студент должен

Знать организацию и принципы работы процессора

Знать основы языка ассемблера, типы регистров процессора,

Уметь выстраивать последовательность машинных операций для реализации простых вычислений.

Необходимое оборудование:ПК, дополнительное ПО.

Порядок выполнения работы :

Изучите теоретический материал

Программа на уровне машинных команд может быть разработана в символическом виде. Такой вид реализуется с помощью языка ассемблера. Программы, написанные на языке ассемблера отличаются высокой эффективностью, т.е минимальным объемом и максимальным быстродействием.

Команды ассемблера можно разделить на следующие группы:

1. Команды пересылки данных

- MOVdst, src – пересылка данных (move пересылать, destination - приемник, source – источник). Пересылает один байт или одно слово (из src в dst) между регистрами или между регистром и памятью, а также заносит непосредственное значение в регистр или память. Н.Р.

  MOV AX, 156*10H

  MOV AX, BX

- PUSHsrc – значение слова в стек (puch – протолкнуть). Помещает в вершине стека содержимое src – любого регистра или ячеек памяти.
- POPdst – извлечение слова из стека (pop – вытолкнуть). Снимает слово с вершины стека и помещает его в любой регистр или ячейку памяти.

2. Арифметические команды

* ADDdst, scr – сложение двоичных чисел (add – сложить) прибавляет байт или слово из памяти, регистра непосредственно к содержимому регистра или прибавляет байт или слово из регистра непосредственно к памяти. Операнды должны иметь одинаковый формат и типы данных.
* SUBdst, scr – вычитание двоичных чисел (substract - вычесть). вычитает байт или слово, взятое из памяти, регистра или непосредственно из содержимого регистра, или вычитает байт или слово, взятое из регистра или непосредственно из памяти (содержимое scr вычитается из содержимого dst).
* CMPdst, scr – сравнение (compare – сравнить). Сравнивает содержимое двух полей данных; фактически команда вычитает второй операнд(scr) и первого (dst), но значение dst не изменяет, а лишь формирует флаги
* INCdst – инкремент (increment – нарастить). Прибавляет 1 к содержимому dst.
* DECdst – декремент (decrement – уменьшить). Вычитает 1 из содержимого dst.
* MULscr – умножение (multiply – умножение без знака). Выполняет умножение беззнакового множимого на беззнаковый множитель. Множитель указывается в scr. Если scr – байт, то множимое должно находится в регистре AL, результат будет в AX. Если scr – слово, то множимое должно находится в регистре AX, результат будет в DX:AX (старшие два байта в DX, младшие – в AX).
* IMULscr – целое умножение знаковых чисел (integermultiply – умножение целых со знаком). Выполняет умножение знакового множимого на знаковый множитель. Множитель указывается в scr. Если scr – байт, то множимое должно находится в регистре AL, результат будет в AX. Если scr – слово, то множимое должно находиться в регистре AX, результат будет в DX:AX (старшие два байта в DX, младшие – в AX).
* DIVscr – деление (divide – деление без знака). Выполняет деление беззнакового

делимого на беззнаковый делитель. Делитель указывается в scr. Если scr – байт, то делимое должно находится в регистре AX, частное от деления будет в AL, остаток от деления будет в AH. Если scr – слово, то делимое должно находится в DX:AX частное от деления будет в регистре AX, остаток от деления в DL. Команда формирует флаг IF (IF=1 при делении на ноль и при делении большого числа на очень малое, если частное вне диапазона).

* IDIVscr – деление целых чисел со знаком (integerdivide – деление целых чисел со знаком). Выполняет деление знакового делимого на знаковый делитель. Делитель указывается в scr. Если scr – байт, то делимое должно находится в регистре AX, частное от деления будет в AL, остаток от деления будет в AH. Если scr – слово, то делимое должно находиться в DX:AX частное от деления будет в регистре AX, остаток от деления в DL. Команда формирует флаг IF (IF=1 при делении на ноль и при делении большого числа на очень малое, если частное вне диапазона).
* ORdst, scr – логическое сложение (or – или). Команда выполняет поразрядную дизъюнкцию битов двух операндов, устанавливает 1 в тех битах операнда dst в которых была 1 хотя бы у одного из исходных операндов. Операнды должны иметь одинаковый формат.
* ANDdst, scr – логическое умножение (and – и). команда выполняет поразрядную дизъюнкцию битов двух операндов, устанавливает 1 в тех битах операнда dst, в которых у обоих исходных операндов были 1. Операнды должны иметь одинаковый формат.

Директивы (псевдооператоры) – это инструкции ассемблеру, они обрабатываются только при ассемблировании (транслировании) программы.

1. Директивы определения идентификатора

= - выполняет текущее присваивание. Присваивает только числовое выражение, содержащее простые математические преобразования, которые и будут выполнены при трансляции.

2. Директивы определения данных

DB – определить байт (1 байт);

DW – определить слово (2 байта);

DD – определить двойное слово (4 байта);

DQ – определить 8 байтов;

DT – определить 10 байтов.

3. Директивы определения сегментов и процедур

Сегмент определяется псевдооператорами:

```
Имя_сегмента segment
…
Имя_сегмента ends
```

В программе можно использовать четыре сегмента и для каждого нужно указать соответствующий регистр сегмента псевдооператором ASSUME (assume - присвоить) Н.Р.

```
codeseg segment
assume CS:codeseg, DS:dataseg, SS:stackseg
…
codesegends
```

После директивы ASSUME следует явным образом загрузить адрес начала сегмента данных в регистр DS:

```
mov AX, dataseg
mov DS, AX
```

Инициализация сегментных регистров CS и SS производится автоматически.

4. Директивы управления трансляцией

Наиболее часто используется END. Директива END отмечает конец программы и указывает ассемблеру, где завершить трансляцию. Формат:

```
END [имя_программы]
```

Пример программы, вычисляющей выражение d=a*b+c, при заданных значениях a, b и c.

```
TITLE prog.ASM
STACK1 segment stack
DB 512 dup (?)
STACK1 ends
DATA segment
a DW 10 ;
b DW -7
c DW 120
d DW ?
DATA ends
CODE segment
Assume CS:code, DS:data, SS:stack1
Start:
push DS
sub AX,AX
push AX
mov AX, data
mov DS, AX
MOV AX, a
MOV BX, b
```

```
IMUL BX
ADD AX, c
MOV d, AX
ret
;завершение программы
movAX, 4c00h
int 21h
CODE ends
End Start
```

Исходная программа, составленная на языке ассемблер оформляется с использованием любого текстового редактора в виде файла с расширением ASM. После обработки исходной программы программой-ассемблером (MASM или TASM) формируется программа в машинных кодах – объектная программа с расширением OBJ. Объектная программа еще не является законченной исполняемой программой. Преобразование объектной программы в исполняемую (компоновка программы) выполняется редактором связи (компоновщиком LINK или TLINK). Исполняемый файл после загрузки имеет расширение EXE. Отладку исполняемой программы удобно выполнять с помощью отладчика DEBUG (TD).

Последовательность формирования программы

1 . Составить программу и ввести ее в текстовый редактор

2. Сохранить файл в корневом каталоге с расширением .asm. H.P.

P.asm

Где P – название программы

3. Ассемблирование программы

Запустить MSDOS или ее эмулятор. Ввести команду

D:\tasm\bin\tasm /z /zi /lP, P, P

4. Компоновка программы

Ввести команду

D:\tasm\bin\tlink /vP, P

5. При необходимости отладка программы.

D:\Tasm\TD

6. Выполнение программы

D:\Tasm\P[.exe]

Листинг

Листинг (распечатка) программы на ассемблере используется при создании и отладке программы.

Листинг состоит из двух частей: из листинга программы и сводной информации о сегментах

и идентификаторах программы.

Листинг формируется в процессе компоновки программы (файл с расширением LST).

Отладка программы

Программа DEBUG обеспечивает интерактивную отладку программ. Отладчик DEBUG умеет:

- Прослеживать выполнение программы и управлять выполнением программ;
- Вносить изменения в ход выполнения и данные отлаживаемой программы;
- Вводить небольшие программы на языке ассемблера, выполнять ассемблирование этих программ и создавать исполняемые программы с расширением COM;
- Выполнять преобразование шестнадцатеричных кодов команд в формат языка ассемблер;
- Отображать текстовые файлы в ASCII и шестнадцатеричном формате;
- Просматривать и изменять содержимое регистров и памяти МП и ячеек основной памяти;
- Считывать информацию с дисков и записывать на них;
- Осуществлять поиск конкретных данных в текстах сообщений и программ с выдачей адреса их хранения;
- Просматривать регистр флагов с отображением мнемокодов значений этих флагов.

Для запуска отладчика введите команду

D:\tasm\bin\td

В отладчике откройте свою программу File->Open->p1.exe

Для анализа работы программы и контроля содержимого регистров процессора нужно переключиться в режим процессора: VIEW->CPU. Затем следует выполнить программу в пошаговом режиме, нажимая клавишу F7.

Задания на лабораторную работу:

Задание 1.

Составьте набор команд на ассемблере, для вычисления значение выражения с использованием арифметических команд.

Исходные данные

1. a+b/(2-c)*d-1	**2.** (a-b)/(2+c)*d
3. a*b/(c*d)-5	4. a-b*c*(3+d)+2
5. (a-4)/(b+c)-d	6. a-b/(c+d*2)
7. a*5—(b+c+d)/2	8. (a-b*c)/(d+5)
9. a+b/c-d*5+1	10. a/(3*b-c)+d
11. (a-b)/2+(c+d)/3	12. (a+1)/(b-1)+c*d
13. (a-b*(c-d))/4	14. a*b+(c-1)/d+1

15. a/(b*(c+1))-d	16. (a+b)*(c+3)/(d-1)
17. (a-b+c*2)/(d+5)	18. (a-b/c)*(d-3)+1
19. a+2*b-3*(c-d)	20. a*(3+b)-2*(c+d)
21. (a+b/2-c*d-1	22. a-b/(2+c)*d
23. a*(b/c+3)*d-5	24. (a-b)*c/4+(d*2)
25. a-4/(b+c)*d	26. a-b/(c+d*2)
27. a*(5—b)+c+d/2	28. a-b*(c/d+5)
29. a+b/(c-d)*3+1	30. a/3*(b-c)+d
31. (a-b/2)*(c+d)/3	32. a+8/b-(1+c)*d
33. a-b*(c-d)/4	34. a*b+(c-2)/(d+3)
35. a/b*(c+1)-d	36.

Задание 2.

Скомпилируйте программу/

Задание 3.

Проверьте работу программы в отладчике.

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.7 Основы программирования процессора (8 часов)

Практическая работа №6

«Программирование переходов»

Задачи обучающегося:

1.Ознакомиться с распределением адресного пространства памяти.

2.Исследовать обмен данными процессора с памятью.

Опорные понятия:логическая структура памяти

Планируемый результат:

Студент должен

Знать понятия «теневая память» и «распределение адресного пространства»

Знать и рассчитывать адресное пространство в памяти

Уметь составлять программу и проводить отладку.

Необходимое оборудование:ПК, дополнительное ПО.

Порядок выполнения работы :

Приложение кодировки символов

Компьютер может обрабатывать только информацию, представленную в цифровом коде. При

вводе текстовой информации буквы и символы кодируются определенными числами, а при выводе их на экран или принтер по каждому коду символа строится изображение символа. Соответствие между набором символов и их кодами называется кодировкой символов.

Как правило, коды имеют длину один байт и принимают значения от 0 до 255 (в настоящее время приобретает распространение и двухбайтная кодировка Unicode). Программы, работающие под DOS, используют ту кодировку, которая имеется в знакогенераторах адаптеров мониторов. На импортных компьютерах IBM PC используется Американский национальный стандартный код для обмена информацией ASCII (American Standard Code for Information Interchange).

В нашей стране разработаны кодировки, имеющие символы кириллицы. При этом символы с кодами 0...127 обычно совпадают с кодами ASCII, так что программа, выводящая сообщение на английском языке, будет работать одинаково на любом персональном компьютере. Альтернативная кодировка ГОСТа (Табл. П1) имеет символы кириллицы в тех позициях, где в кодировке ASCII находятся относительно редко используемые символы национальных европейских алфавитов. Коды представлены в десятичной (D) и шестнадцатеричной (H) системах.

Первые 32 кода (1...32) имеют два значения: управляющие символы и изобразительные символы. Когда DOS пересылает эти коды на монитор или принтер, они выполняют управляющие функции, а не отображают символы, например:

8-возврат на одну позицию; 9-горизонтальная табуляция; 10- перевод строки; 13- возврат каретки; 32- пробел.

Для получения изображения символов эти коды необходимо занести в буфер экрана (начальный адрес 0B800:0000H).

Программы, работающие в Windows, не используют для вывода средства знакогенератора адаптера монитора. Windows предоставляет более удобные средства, поддерживая масштабируемые шрифты. В кодировке текстовых шрифтов для Windows отсутствуют символы псевдографики, т.к. Windows поддерживает настоящую графику, но имеется большое количество букв европейских языков и полиграфических символов. В русской версии кодировки для Windows символы русского алфавита имеют коды от 192 до 255.

Практическая часть

1. Изучить методические указания.
2. Подготовить ответы на контрольные вопросы.

Ниже приведена программа CHANGE , которая в заданной текстовой строке заменяет латинские строчные буквы заглавными.

Коды строчных и заглавных букв английского алфавита можно найти в Таблице кодировки символов (Приложение, с 23).

3. Проанализировать приведенную ниже программу CHANGE, дополнить каждую команду комментарием.
4. . Введите программу, используя текстовый редактор. Оттранслируйте и скомпонуйте программу в режимах TASM/ZI, TLINK/V.
5. Загрузите отладчик и программу. Произведите ее пошаговое выполнение. Наблюдайте результаты выполнения команд.
6. Установите ловушку на одной из команд подпрограммы. В точке останова отройте в окне CPU локальное меню и выберите пункт CALLER.Пронаблюдайте исполнение этой инструкции.
7. Пронаблюдайте результат выполнения программы в окне WINDOW (режим USER SCREEN).
8. Введите вариант программы из домашнего задания, обеспечивающий замену заглавных букв строчными.
9. Убедитесь в работоспособности второго варианта программы.

ПРИМЕР ПРОГРАММЫ

```
TITLE CHANGE - ЗАМЕНА СТРОЧНЫХ БУКВ ЗАГЛАВНЫМИ
DATASGSEGMENT PARA
MYTEXT DB 'Our Native Town' ,13,10, '$'
DATASGENDS
STACKSG SEGMENT 'Stack'
```

```
DB 12 DUP(?) STACKSG ENDS
CODESG SEGMENT PARA 'Code'
BEGIN PROC FAR
ASSUME SS:STACKSG, CS:CODESG, DS:DATASG
PUSH DS
SUB AX,AX
PUSH AX
MOV AX, DATASG
MOV DS, AX
LEA BX, MYTEXT
MOV CX, 10H MT1: MOV AH, [BX]
CMP AH, 61H
JB MT2
CMP AH, 7AH
JA MT2
CALL COR MT2: INC BX
LOOP MT1
LEA DX, MYTEXT
MOV AH, 09H INT 21H RET BEGIN ENDP
COR PROC NEAR
NOP
AND AH, 0DFH
MOV [BX], AH
RET COR ENDP CODESG ENDS END BEGIN
```

10. Ввести свой собственный текст на английском языке, содержащий строчные и заглавные буквы.

11. Изменить программу так, чтобы в соответствии с вариантом задания (Таб 5.1.) она обеспечивала:

	Таблица 5.1.
№варианта	Заменить
	а) ‘a’ на ‘A’ б) все заглавные строчными
	а) строчные от ‘a’ до ‘f’ заглавными б) все заглавные строчными
	а) строчные ‘b’и’c’ заглавными б) все заглавные строчными
	а) строчные от ‘f’ до’z’ заглавными б) все заглавные строчными
	а) символ ’(’ на символ ‘) ’ б) все заглавные строчными
	а) ‘Z’ на ‘z’ б) все заглавные строчными

Контрольные вопросы

1. Назовите три типа команды безусловного перехода.
2. Какой может быть длина перехода в разных типах команды JMP?
3. Содержимое каких регистров модифицируется при выполнении безусловных переходов разных типов?
4. Какова максимальная длина условного перехода?
5. Каким образом может быть указан адрес перехода?
6. Какие флаги могут быть использованы в командах условного перехода после выполнения команды сложения?
7. Приведите возможные команды условных переходов, если после сравнения беззнаковых чисел D1иD2 оказалось: а)D1=D2, б) D1< D2 , в) D1>D2.
8. Приведите возможные команды условных переходов, если после сравнения чисел со знаками P1иP2 оказалось: а) P1 *P2, б) P1<P2, в) P1 >P2.
9. Какие команды могут использоваться для организации циклов?
10. Какова максимальная длина переходов при организации циклов?

11. Какие признаки , кроме CX=0, могут быть использованы при организации циклов?
12. Как осуществляется переход к процедурам разных типов?
13. Назовите варианты команды возврата из процедуры

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.7 Основы программирования процессора (8 часов)

Практическая работа №7

«Программирование ввода-вывода»

Задачи обучающегося:

1.Углублениезнаний по структуре и принципам работы процессора.

Опорные понятия:программирование ввода-вывода

Планируемый результат:

Студент должен

Знать типы регистров процессора, структуру команд процессора

Уметь применять функции ввода-вывода средствами MS DOS.

Необходимое оборудование:ПК, дополнительное ПО.

Порядок выполнения работы :

Методические указания:

Для представления всех букв, цифр и знаков, появляющихся на экране компьютера, обычно используется всего один байт. Символы, соответствующие значениям от 0 до 127, то есть первой половине всех возможных значений байта, были стандартизованы и названы символами ASCII (хотя часто кодами ASCII называют всю таблицу из 256 символов). Сюда входят некоторые управляющие коды (символ с кодом 0Dh — конец строки), знаки препинания, цифры (символы с кодами 30h – 39h), большие (41h – 5Ah) и маленькие (61h – 7Ah) латинские буквы. Вторая половина символьных кодов используется для алфавитов других языков и псевдографики.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	f
0																
1																
2		!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п

B	▒	▒	▓	┃	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▉	▄	▌	▐	▀
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	Ё	ё	Є	є	Ї	ї	Ў	ў	°	∙	·	√	№	¤	■	

Команды вывода на экран средствами DOS

ФункцииDOS

Номер прерывания	Функция	Описание	При вызове	При возврате
Int 21h	01h	Ввод символа с эхом. Вводит символ с клавиатуры и отображает его на экране. При отсутствии символа ждет его ввода	AH=01h	AL=код введенного символа
	02h	Вывод символа на экран.	AH=02h	DL=код выводимого символа
	06h	Прямой ввод-вывод. Вводит с клавиатуры или выводит символ на экран	AH=06h DL=код выводимого символа (при выводе) DL=FFh (при вводе)	AL=код введенного символа (при вводе)
	09h	Вывод строки. Строка должна заканчиваться символом $	AH=09h DS:DX =адрес выводимой строки	
	25h	Установка вектора прерывания. позволяет заполнить вектор прерывания адресом обработчика прерываний	AH=25h AL=номер вектора прерывания DS:DX=адрес обработчика прерывания	
	2Ah	Получение текущей даты	AH=2Ah	CX=год DH=месяц DL=день AL=день недели
	2Ch	Получение текущего времени	Ah=2Ch	CH=часы CL=минуты DH=секунды
	2Dh	Установка текущего времени. Позволяет изменить текущее время системных часов	AH=2Dh CH=часы CL=минуты DH=секунды	AL=00h – успешное выполнение AL=FFh - недопустимое время, системное время не измнилось
	30h	Возвращает номер используемой версии DOS	AH=30h	AL=номер версии AH=номер подверсии
	35h	Получение вектора прерывания	AH=35h AL=номер вектора прерывания	ES:BX=адрес обработчика прерывания
	4Ch	Завершение процесса с кодом возврата	AH=4Ch AL= код возврата	

Функции DOS вывода на экран позволяют перенаправлять вывод в файл, но не позволяют

вывести текст в любую позицию экрана и не позволяют изменить цвет текста. DOS предполагает, что для более тонкой работы с экраном программы должны использоваться видеофункции BIOS.

Для того, чтобы вывести на экран строку текста нужно записать в сегмент данных эту строку

msg1 db ‘Иванов Иван Иванович$’

Затем в сегменте кода в регистр AH записываем номер функции вывода на экран , в регистр DX записывается адрес строки, затем вызывается прерывание int 21h

```
mov AH, 09h
mov DX, offset msg1
int 21h
```

Чтобы увидеть результат работы программы на экране рекомендуется использовать функцию 01h

```
movAH,01h
int 21h
```

Вывод нескольких символов можно осуществить с помощью цикла. . Количество повторений предварительно записывается в регистр CX (счетчик циклов).

```
movCX, 223
```

В сегменте данных следует объявить массив, в который будут помещены символы, выводимые на экран.

```
symbols db 223 dup ('*')                ;заполняем массив из 224 символов *
```

Затем нужно заполнить массив символами (в сегменте кода)

```
;подготовка цикла
 mov CX, 223        ;количество повторений цикла
 mov SI, 0  ;индекс элемента в заполняемом массиве
 mov AL, 32         ;код первого символа; цикл заполнения массива
begin: mov symbols[SI], AL ;очередной код записывается в массив
     inc AL  ;получаем код следующего символа
inc SI          ;переходим к следующему элементу массива
loop begin      ;повторяем цикл заданное число раз
```

После этого выводим массив на экран

```
mov AH, 40h         ;функция вывода записывается в регистр AH
 mov BX, 1 ;стандартный дескриптор экрана заносим в регистр BX
 mov CX, 223;число выводимых байтов заносим в регистр CX
 mov DX, offset symbols  ;адрес выводимого сообщения
int 21h ;вызов DOS
```

Ввод с клавиатуры

Как и в случае вывода на экран, DOS предоставляет набор функций для чтения данных с клавиатуры, которые используют стандартное устройство ввода STDIN, так что можно использовать в качестве источника данных файл или стандартный вывод другой программы.

Функции посимвольного ввода без эха можно использовать для интерактивного управления программой.

Задания на лабораторную работу:

Задание 1.

Напишите программу, которая выведет на экран Вашу фамилию, имя и отчество. Скомпилируйте программу. Проверьте работу программы.

Задание 2.

Напишите программу, которая выводит на экран символы согласно варианту

Вариант	Задание	Вариант	Задание	Вариант	Задание
1	Прописные буквы латинского алфавита ABCDEFGHIJKL	2	Строчные буквы латинского алфавита abcdefghij	3	Цифры 0 1 2 3 4 5 6 7 8 9
4	Первые десять прописных букв русского алфавита	5	Первые десять строчных букв русского алфавита	6	Символы ( ) * + , - . /
7	Символы ! " # $ % & ' ( )	8	Символы : ; < = > ? @	9	Символы [\ ] ^ _ `
10	Символы { | } ~ €	11	Символы %& ' ( ) * + ,	12	Символы >=<;:
13	Прописные буквы латинского алфавита MNOPQRSTU	14	Прописные буквы латинского алфавита OPQRSTUVWXYZ	15	Строчные буквы латинского алфавита klmnopqr
16	Буквы русского алфавита К Л М Н О П Р С Т У Ф	17	Буквы русского алфавита к л м н о п р с т у ф	18	Первые десять печатных символов кодовой таблицы
19	Вторые десять печатных символов кодовой таблицы	20	Третьи десять печатных символов кодовой таблицы	21	Цифры 9 8 7 6 5 4 3 2 1 0
22	Прописные буквы латинского алфавита LKJIHGFEDCBA	23	Строчные буквы латинского алфавита jihgfedcba	24	Последние десять прописных букв русского алфавита
25	Последние десять строчных букв русского алфавита	26	Последние десять прописных букв латинского алфавита	27	Последние десять строчных букв латинского алфавита
28	Символы, коды которых с 38 по 45	29	Символы, коды которых с 48 по 59	30	Символы, коды которых с 62 по 73
31	Символы, коды которых с 84 по 99	32	Символы, коды которых со 101 по 119	33	Символы, коды которых со 121 по 128
34	Символы, коды которых со 130 по 144	35	Символы, коды которых со 182 по 197		

Скомпилируйте программу. Проверьте работу программы.

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.7 Основы программирования процессора (8 часов)

Практическая работа №8

«Программирование и отладка программ»

Задачи обучающегося:

1.Изучить процесс отладки программного обеспечения ручным методом

Опорные понятия: отладка программ

Планируемый результат:

Студент должен

Знать понятия «Отладка» и «Локализация»

Знать и рассчитывать адресное пространство в памяти

Уметь составлять программу и проводить отладку.

Необходимое оборудование: ПК, дополнительное ПО.

Порядок выполнения работы :

Отладка – это процесс локализации и исправления ошибок, обнаруженных при тестировании программного обеспечения.

Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т.е. определить оператор или фрагмент, содержащие ошибку. Причины ошибок могут быть как очевидны, так и очень глубоко скрыты. В целом сложность отладки обусловлена следующими причинами:

– требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;

– психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;

– возможно взаимовлияние ошибок в разных частях программы, например, за счет затирания области памяти одного модуля другим из-за ошибок адресации;

– отсутствуют четко сформулированные методики отладки.

Классификация ошибок

В соответствии с этапом обработки, на котором появляются ошибки, различают:

синтаксические ошибки – ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и частично семантического анализа программы;

ошибки компоновки – ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы;

ошибки выполнения – ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

Методы отладки программногообеспечения

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации. При этом используют различные методы:

– ручного тестирования;
– индукции;
– дедукции;
– обратного прослеживания.

Метод ручного тестирования

Это – самый простой и естественный способ данной группы. При обнаружении ошибки

необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которыми была обнаружена ошибка. Метод очень эффективен, но не применим для больших программ, программ со сложными вычислениями и в тех случаях, когда ошибка связана с неверным представлением программиста о выполнении некоторых операций. Данный метод часто используют как составную часть других методов отладки.

Общая методика отладки программных продуктов, написанных для выполнения в операционных системах MS DOS и Win32:

1 этап – изучение проявления ошибки;
2 этап – определение локализации ошибки;
3 этап – определение причины ошибки;
4 этап – исправление ошибки;
5 этап – повторное тестирование.

Процесс отладки можно существенно упростить, если следовать основным рекомендациям структурного подхода к программированию:

– программу наращивать «сверху-вниз», от интерфейса к обрабатывающим подпрограммам, тестируя ее по ходу добавления подпрограмм;
– выводить пользователю вводимые им данные для контроля и проверять их на допустимость сразу после ввода;
– предусматривать вывод основных данных во всех узловых точках алгоритма (ветвлениях, вызовах подпрограмм).

Спецификация программы, программная спецификация (program specification) - точная и полная формулировка определенной задачи или группы задач, содержащая сведения, необходимые для построения алгоритма их решения. Содержит описание результата, который должен быть достигнут с помощью конкретной программы, а также действий, выполняемых программой для достижения конечного результата без упоминания того, как указанный результат достигается

Практическая часть

Задание 1. Запишите вариант в отчет.
Задание 2. Согласно поставленной задаче выполните ручную отладку:
- Опишите математическую модель задачи с указанием имен и назначения переменных;
- Опишите спецификацию программы;
- Запишите алгоритм программы;
- Выполните отладку логики программы методом «грубой силы» с помощью соседа;
- Составьте тестовые наборы для проверки функционала системы.

Задание 3. Результаты выполнения практического задания запишите в отчет.

Варианты заданий

Создать Windows-приложение, реализующие линейный и разветвляющийся алгоритмы, которые размещены на разных вкладках окна формы. На вкладке линейного алгоритма предусмотреть поля ввода значений переменных и поле вывода результата вычисления. На вкладке разветвляющегося алгоритма предусмотреть поля для ввода значений переменных, поле вывода результатов расчета по одной из трех формул в зависимости от результата выполнения условия. В качестве $f(x)$ использовать по выбору: $cos(x)$ или x^2 или e^x. Пример рабочей формы представлен на рисунке 1.

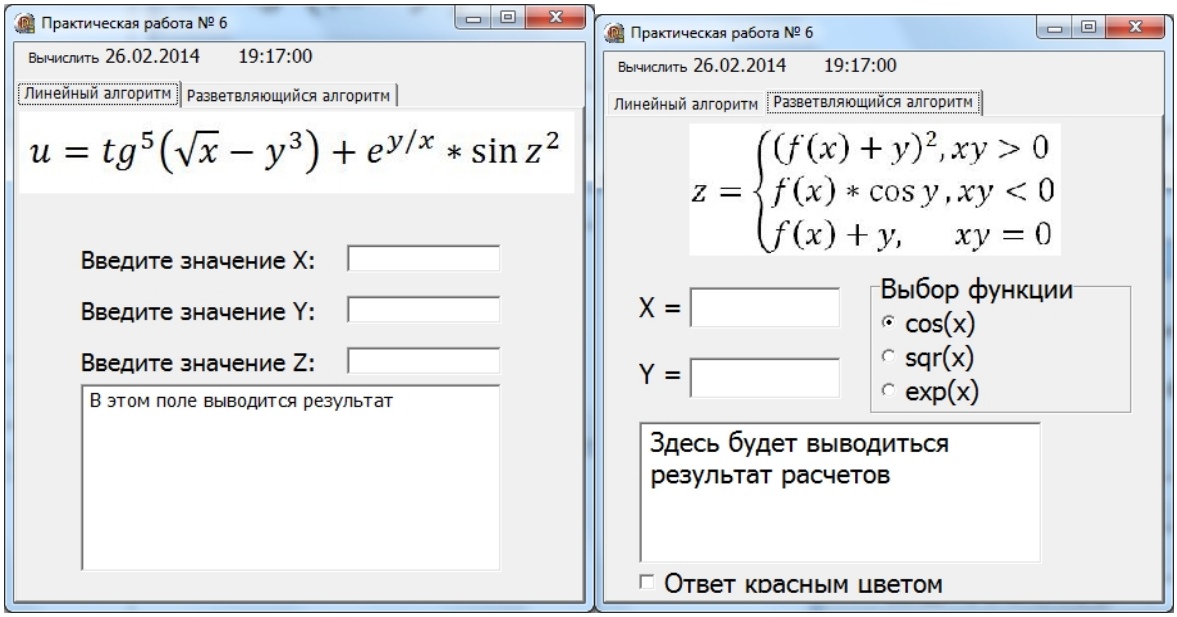


Рисунок 1 – Windows-приложение

Линейный алгоритм:

1. .
2. .
3. .
4.
5. .
6. .
7. .

Разветвляющийся алгоритм:

1. 2.

3. 4.

5. 6.

7. 8.

Содержание отчета

1. Тема. Цель.
2. Оборудование.
3. Результат выполнения практического задания.
4. Ответы на контрольные вопросы.
5. Вывод.

Отладка – это процесс локализации и исправления ошибок, обнаруженных при тестировании программного обеспечения.

Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т.е. определить оператор или фрагмент, содержащие ошибку. Причины ошибок могут быть как очевидны, так и очень глубоко скрыты. В целом сложность отладки обусловлена следующими причинами:

– требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;

– психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;

– возможно взаимовлияние ошибок в разных частях программы, например, за счет затирания области памяти одного модуля другим из-за ошибок адресации;

– отсутствуют четко сформулированные методики отладки.

Классификация ошибок

В соответствии с этапом обработки, на котором появляются ошибки, различают:

–*синтаксические ошибки* – ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и частично семантического анализа программы;

–*ошибки компоновки* – ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы;

–*ошибки выполнения* – ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

Методы отладки программного обеспечения

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации. При этом используют различные методы:

–ручного тестирования;

–индукции;

–дедукции;

–обратного прослеживания.

Метод ручного тестирования

Это – самый простой и естественный способ данной группы. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которыми была обнаружена ошибка. Метод очень эффективен, но не применим для больших программ, программ со сложными вычислениями и в тех случаях, когда ошибка связана с неверным представлением программиста о выполнении некоторых операций. Данный метод часто используют как составную часть других методов отладки.

Общая методика отладки программных продуктов, написанных для выполнения в операционных системах MS DOS и Win32:

1 этап – изучение проявления ошибки;

2 этап – определение локализации ошибки;

3 этап – определение причины ошибки;

4 этап – исправление ошибки;

5 этап – повторное тестирование.

Процесс отладки можно существенно упростить, если следовать основным рекомендациям структурного подхода к программированию:

- программу наращивать «сверху-вниз», от интерфейса к обрабатывающим подпрограммам, тестируя ее по ходу добавления подпрограмм;
- выводить пользователю вводимые им данные для контроля и проверять их на допустимость сразу после ввода;
- предусматривать вывод основных данных во всех узловых точках алгоритма (ветвлениях, вызовах подпрограмм).

Спецификация программы, программная спецификация (program specification) - точная и полная формулировка определенной задачи или группы задач, содержащая сведения, необходимые для построения алгоритма их решения. Содержит описание результата, который должен быть достигнут с помощью конкретной программы, а также действий, выполняемых программой для достижения конечного результата без упоминания того, как указанный результат достигается

Практическая часть

Задание 4. Запишите вариант в отчет.

Задание 5. Согласно поставленной задаче выполните ручную отладку:

- Опишите математическую модель задачи с указанием имен и назначения переменных;
- Опишите спецификацию программы;
- Запишите алгоритм программы;
- Выполните отладку логики программы методом «грубой силы» с помощью соседа;
- Составьте тестовые наборы для проверки функционала системы.

Задание 6. Результаты выполнения практического задания запишите в отчет.

Контрольные вопросы

1. В чем заключается ручная отладка ПО?
2. На каком этапе проводится ручная отладка?
3. Опишите методы отладки.

Варианты заданий

Создать Windows-приложение, реализующие линейный и разветвляющийся алгоритмы, которые размещены на разных вкладках окна формы. На вкладке линейного алгоритма предусмотреть поля ввода значений переменных и поле вывода результата вычисления. На вкладке разветвляющегося алгоритма предусмотреть поля для ввода значений переменных, поле вывода результатов расчета по одной из трех формул в зависимости от результата выполнения условия. В качестве *f(x)* использовать по выбору: *cos(x)* или x^2 или e^x. Пример рабочей формы представлен на рисунке 1.

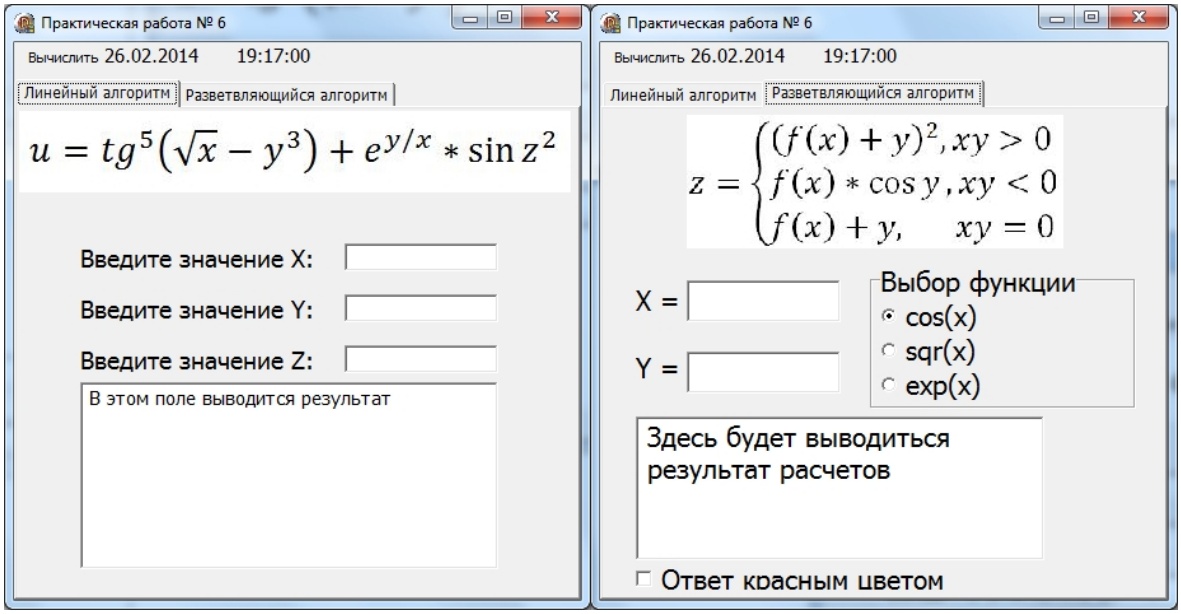


Рисунок 1 – Windows-приложение

Линейный алгоритм:

8. .

9. .

10. .

11.

12. .

13. .

14. .

Разветвляющийся алгоритм:

1. 2.

3. 4.

5. 6.

7. 8.

Содержание отчета

1. Тема. Цель.
2. Оборудование.
3. Результат выполнения практического задания.
4. Ответы на контрольные вопросы.
5. Вывод.

Раздел 3. Архитектура и принципы работы основных логических блоков вычислительных систем

Тема 3.8 Современные процессоры (1 час)

Практическая работа №9

«Идентификация и установка процессора»

Задачи обучающегося:

1.Изучить характеристики процессора.

Опорные понятия:центральный процессор

Планируемый результат:

Студент должен

Знать основные характеристики процессора

Уметь идентифицировать и устанавливать процессоры.

Необходимое оборудование:учебная литература.

Порядок выполнения работы :

Основа вычислительной системы – микропроцессор (МП). МП характеризуется следующими параметрами:

Тактовая частота

- Степень интеграции микросхемы (сколько транзисторов содержится в чипе).
- Внутренняя разрядность данных (количество бит, которые МП может обрабатывать одновременно);
- Внешняя разрядность данных (количество одновременно передаваемых бит в процессе обмена данными с памятью и другими устройствами);
- Адресуемая память (зависит от числа адресных бит).

Типы МП

Тип процессора	Поколение	Год выпуска	Разрядность шины данных	Разрядность шины адреса	Первичная кэш-память, Кбайт		Тактовая частота шины, МГц	Тактовая частота процессора, МГц	Количество транзисторов, млн	Размер минимальной структуры, мкм
					Команды	Данные				
8088	1	1979	8	20			4,77-8	4,77-8		
8086	1	1978	16	20			4,77-8	4,77-8	0,029	3,0
80286	2	1982	16	24			6-20	6-20	0,13	1,5
80386DX	3	1985	32	32			16-33	16-33	0,27	1,0
80386SX	3	1988	16	32	8		16-33	16-33	0,27	1,0
80486DX	4	1989	32	32	8		25-50	25-50	1,2	1,0-0,8
80486SX	4	1989	32	32	8		25-50	25-50	1,1	0,8
80486DX2	4	1992	32	32	8		25-40	25-40		
80486DX4	5	1994	32	32	8	8	25-40	75-120		
Pentium	5	1993	64	32	8	8	60-66	60-200	3.1-3.3	0.8-0.35
P-MMX	5	1997	64	32	16	16	66	166-233	4.5	0.6-0.35
Pentium Pro	6	1995	64	32	8	8	66	150-200	5.5	0.35
Pentium II	6	1997	64	32	16	16	66	233-300	7.5	0.35-0.25
Pentium II Celeron	6	1998	64	32	16	16	66/100	266-533	7.5-19	0.25

Pentium Xeon	6	1998					100	400-1700		0.18
Pentium III	6	1999	64	32	16	16	100	450-1200	9.5-44	0.25-0.13
AMD Athlon	7	1999	64	32	64	64	266	500-2200	22	0.25
Pentium 4	7	2000	64	32	12	8	400	1.4-3.4 ГГц	42-125	0,18-0,09
AMD Athlon 64	8	2003	64	64	64	64	400	2 ГГц	54-106	0,13-0,09
Pentium 4 Prescott	8	2004	64	32	16	16	800	2.8-3.4 ГГц	125	0.09

В сущности МП – плоский квадратный слой кремния со схемами, выгравированными на его поверхности. Этот элемент укрепляется на основе - керамической или пластмассовой – образуя пакет с контактами, выполненными или по плоской нижней стороне или по одному из краев. Пакет ЦП связан с системной платой через разъем формы гнездо (Soket) и слот (Slot).

Характеристики интерфейсов процессоров

Наименование	Разъем (число контактов)	Описание
Soket 1	169	Устанавливался на системных платах 486, напряжение 5 В, поддерживает 486 чипсет
Soket 2	238	Минимальное обновление Soket 1, поддерживает те же схемные элементы
Soket 3	237	Рабочее напряжение 5 В, но может работать также и при 3,3 В, переключатели размещены на системной плате
Soket 4	273	Первый разъем, спроектированный для Pentium. рабочее напряжение 5 В
Soket 5	320	Работает при 3,3 В поддерживает Pentium от 75 до 133 МГц
Soket 6	235	Разработан для процессора 486. напряжение 3,3 В
Soket 7	321	Разработан для Pentium-MMX, используется для всех Pentium с частотой шины 66 МГц
Soket 8	387	Используется только для Pentium Pro, из-за дороговизны быстро вышел из употребления
Slot 1	242	Поддерживает кэш-память L1 до 512 Кбайт, состоящая из двух по 256 Кбайт; оперирует на половинной частоте МП. Использовался для Pentium II, Pentium III и Celeron
Slot 2	330	Аналогичен разъему Slot 1, но поддерживает до 2 Мбайт кэш-памяти, работающей на частоте МП. Использовался для Pentium II, Pentium III, Pentium Xeon
Slot A	242	Разработан для AMD Athlon, механически совместим с разъемом Slot 1, но поддерживает совершенно другие электрические цепи
Soket 370	370	Заменяет Slot 1 для МП Celeron. Также используется для Pentium III

Soket A	462	Разработан для процессора AMD Athlon, который содержал на кристалле кэш-память L2
Soket 423	423	Введен для удовлетворения новых требований Pentium 4, который содержит совершенно новую системную шину (FSB). Включает теплорассеиватель
Soket 603	603	Предназначен для Pentium 4. дополнительные контакты ориентированы на МП, которые будут содержать на кристалле кэш-память, а также для подключения других процессоров в мультипроцессорных системах
Soket 478	478	Разработан для поддержки технологий 0,13-мкм для Pentium 4 Northwood.
Socket AM2 940	940	Выпущен AMD в 2006 году для Athlon 64 X2 на 5000+ и 4000+
Socket LGA 1155	1155	Предназначен для процессора Intel Core i7-3960X (Sandy Bridge-E). Процессор поддерживает четыре канала памяти и 40 линий PCI Express, что потребовало намного большего количества ножек для питания и передачи сигналов.

Задания на лабораторную работу:

Задание 1.

Выберите процессор, подходящий для установки на целевой системной плате. Установите процессор на целевую системную плату

Задание 2.

Идентифицируйте процессор целевого компьютера. Назовите его основные характеристики. Дайте рекомендации по модернизации целевого компьютера