МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Благовещенский государственный педагогический университет»
ОСНОВНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА Рабочая программа дисциплины

УТВЕРЖДАЮ
И.о. декана физико-математический факультета ФГБОУ ВО «БГПУ»

_____ О.А.Днепровская
«22» мая 2019 г.

Рабочая программа дисциплины
АРХИТЕКТУРА КОМПЬЮТЕРА

Направление подготовки
44.03.05 ПЕДАГОГИЧЕСКОЕ ОБРАЗОВАНИЕ
(с двумя профилями подготовки)

Профиль
«ИНФОРМАТИКА»

Профиль
«МАТЕМАТИКА»

Уровень высшего образования
БАКАЛАВРИАТ

Принята на заседании кафедры
информатики и методики преподавания информатики
(протокол № _9_ от «15» мая 2019 г.)

Благовещенск 2019

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1 Цель дисциплины: формирование систематизированных знаний в области архитектуры компьютера, организации компьютерных систем, программирования на языке ассемблера.

1.2 Место дисциплины в структуре ООП: Дисциплина «Архитектура компьютера» относится к обязательным дисциплинам вариативной части дисциплин (модулей) (Б1.О.31). Для освоения дисциплины используются знания, умения и виды деятельности, сформированные в процессе изучения дисциплин: «Программирование», «Информатика», «Теоретические основы информатики». Изучение дисциплины закладывает основы для дальнейшего освоения студентами курсов по выбору профессионального цикла, дальнейшей профессиональной деятельности.

1.3 Дисциплина направлена на формирование следующих компетенций: УК-1, УК-6, ОПК-8, ПК-2:

- **УК-1.** Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач, **индикаторами** достижения которой является:

- УК-1.2 Находит и критически анализирует информацию, необходимую для решения поставленной задачи.
- УК-1.3 Аргументировано формирует собственное суждение и оценку информации, принимает обоснованное решение.

- **УК-6.** Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни, **индикаторами** достижения которой является:

- УК-6.3. Критически оценивает эффективность использования времени и других ресурсов при решении поставленных целей и задач.

- **ОПК-8.** Способен осуществлять педагогическую деятельность на основе специальных научных знаний.

- ОПК-8.3. Демонстрирует специальные научные знания в т.ч.в предметной области

- **ПК-2.** Способен осуществлять педагогическую деятельность по профильным предметам (дисциплинам, модулям) в рамках программ основного общего и среднего общего образования, индикаторами достижения которой является:

- ПК-2.3 Применяет методологии программирования и современные информационно-коммуникационные технологии для решения практических задач получения, хранения, обработки и передачи информации.

1.4 Перечень планируемых результатов обучения. В результате изучения дисциплины студент должен

- **знать**:

- классификацию компьютеров по различным признакам, характеристики и особенности различных классов ЭВМ, тенденции развития вычислительных систем;
- структурную и функциональную схему персонального компьютера, назначение, виды и характеристики центральных и внешних устройств ПЭВМ;
- формы представление информации в ЭВМ;
- принципы фон Неймана и классическую архитектуру современного компьютера, структуру микропроцессора, понятие о языке ассемблера (макроассемблера) и основных методах программирования с его использованием;

- **уметь**:

- использовать знания архитектуры компьютера, организации компьютерных систем, программирования на языке ассемблера в профессиональной деятельности;

- владеть:

- навыками программирования на языке ассемблера и макроассемблера.

1.5 Общая трудоемкость дисциплины «Архитектура компьютера» составляет 4 зачетных единицы (далее – ЗЕ) (144 часа):

№	Наименование раздела	Курс	Семестр	Кол-во часов	ЗЕ
1.	Архитектура компьютера	1	2	144	4

Общая трудоемкость дисциплины «» составляет 4 зачётные единицы.

Программа предусматривает изучение материала на лекциях и лабораторных занятиях. Предусмотрена самостоятельная работа студентов по темам и разделам. Проверка знаний осуществляется фронтально, индивидуально.

1.6 Объем дисциплины и виды учебной деятельности

Объем дисциплины и виды учебной деятельности (очная форма обучения)

Вид учебной работы	Всего часов	Семестр 2
Общая трудоемкость	144	144
Аудиторные занятия	54	54
Лекции	22	22
Лабораторные занятия	32	32
Самостоятельная работа	54	54
Вид итогового контроля	36	экзамен

УЧЕБНО-ТЕМАТИЧЕСКОЕ ПЛАНИРОВАНИЕ

2.1 Очная форма обучения

Учебно-тематический план

№	Наименование тем (разделов)	Всего часов	Аудиторные занятия		Самостоятельная работа
			Лекции	Лабораторные занятия	
1.	1. Многоуровневая компьютерная организация	6	2	-	4
2.	2. Цифровой логический уровень. Основные схемы	12	2	6	4
3.	3. Принципы организации памяти. ОЗУ. ПЗУ. КЭШ-память. Внешняя память	12	2	-	10
4.	4. Структурная организация микропроцессора.	14	2	6	6
5.	5. Шины. Работа шин. Организация ввода-вывода. Механизм прерываний	8	4	-	4
6.	6.Архитектура команд. Форматы команд. Введение в язык Ассемблер	34	4	20	10

7.	7. Уровень ОС. Виртуальная память	12	4	-	8
8.	8. Внешние устройства ввода-вывода	10	2	-	8
	Экзамен	36			
ИТОГО		**144**	**22**	**32**	**54**

Интерактивное обучение по дисциплине

№	Наименование тем (разделов)	Вид занятия	Форма интерактивного занятия	Кол-во часов
1.	Цифровой логический уровень. Основные схемы	ЛК	Обсуждение решения проблемных задач	2ч.
2.	Архитектура команд. Форматы команд. Введение в язык Ассемблер	ЛК	Обсуждение решения проблемных задач	2ч.
3.	Цифровой логический уровень. Основные схемы.	ЛБ	Работа в парах	2ч.
4.	Машинное исполнение команд	ЛБ	Работа в парах	2ч.
5.	Введение в Ассемблер. Основные команды.	ЛБ	Работа в парах	2ч.
ИТОГО				**10**

3 СОДЕРЖАНИЕ ТЕМ (РАЗДЕЛОВ)

Тема 1. Многоуровневая компьютерная организация.

Понятие многоуровневой компьютерной организации. Современные многоуровневые машины. Понятие архитектуры компьютера. Классификации ЭВМ по разным признакам. Характеристики ЭВМ. Принципы фон Неймана (классическая архитектура).

Тема 2. Цифровой логический уровень. Основные схемы.

Вентили и их виды. Интегральные схемы. Комбинационные схемы: декодер, мультиплексор, демультиплексор, компаратор. Арифметические схемы: сумматор, полусумматор, АЛУ. Тактовый генератор. Триггеры, защелки. Регистры.

Тема 3. Принципы организации памяти.

Логическая блок-схема памяти 4 на 3 (принцип работы). Понятие адреса. Микросхемы памяти. ОЗУ и ПЗУ. Виды, назначение. КЭШ-память (3-х уровневая модель). Внешняя память. Виды. Характеристики памяти. Принципы работы различных видов внешней памяти.

Тема 4. Структурная организация микропроцессора.

Микропроцессор (МП) i 8086. Его структура. Исполнительный блок. Устройство сопряжения с системной магистралью. Формирование физического адреса ОЗУ. Взаимодействие элементов при работе микропроцессора. Микросхема процессора. Алгоритм работы МП. Тракт данных. Классификация МП. RISC и CISC процессоры. Цоколевка типичного МП.

Тема 5. Шины. Работа шин. Организация ввода-вывода. Механизм прерываний.

Понятие шины. Характеристики шины. Способы передачи данных. Группы и виды

шин. Устройства, работающие с шиной. Ширина шины. Арбитраж шины. Синхронизация шины. Два способа организации реакции процессора на события: опрос, прерывание. Понятие прерывания. Цикл осуществления прерываний. Схема контролера прерывания. Назначение ее основных элементов. Взаимодействие схемы с процессором и внешними устройствами при прерывании. Таблицы векторов прерываний. Возврат к программе после прерывания. Виды прерываний. Понятие стека. Работа стека при прерывании. Программные прерывания.

Тема 6. Архитектура команд. Форматы команд. Введение в язык «Ассемблер»

Типы данных. Форматы команд. Критерии разработки для форматов команд. Расширение кода операций. Определение языка Ассемблера, его особенности, области применения. Понятие мнемонического кода операции. Компоненты предложения языка Ассемблер. Понятия метки, мнемоники, операнда, комментарий, константы. Основные типы и группы команд. Типы операндов. Основные способы адресации. Понятие макропроцессора, макровызов (макрос), макроопределение, макрорасширение. Области применения макрокоманд. Команды пересылки данных. Арифметические и логические команды. Команды передачи управления. Команды int, call. Работа стека при вызове подпрограмм.

Тема 7. Уровень ОС. Виртуальная память.

Уровень операционной системы (ОС). Виртуализация памяти (способы). Распределение памяти. Виртуальная память (ВП). Страничная организация ВП и ее реализация. Сегментация и ее реализация. Виртуальная память в процессоре Pentium II. Сегментно-страничная ВП.

Тема 8. Внешние устройства ввода-вывода.

Понятие внешнего устройства. Их виды, характеристики. Принципы функционирования. Принципы управления внешними устройствами персонального компьютера. Базовая система ввода/вывода.

4 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ (УКАЗАНИЯ) ДЛЯ СТУДЕНТОВ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ

Для успешного усвоения дисциплины необходима самостоятельная работа студентов:

– регулярная проработка теоретического материала по конспектам лекций и учебникам;

– систематическая подготовка к лабораторным занятиям;

В случае появления каких-либо вопросов следует обращаться к преподавателю в часы его консультаций. Критерием качества усвоения знаний могут служить аттестационные оценки по дисциплине и текущие оценки, выставляемые преподавателем в течение семестра.

Рекомендации по изучению отдельных тем курса:

При изучении тем № 1-8 особое внимание следует обратить на то, что часть содержания этих темы оставлена на самостоятельное изучение и при подготовке к ним студент должен пользоваться основной и дополнительной литературой, представленной по данному курсу. Также при изучении тем № 2, № 4 и № 6 необходимо выполнить ряд лабораторных работ: при изучении темы № 2 лабораторную работу №1, 2 при изучении тем №4, 6 – лабораторные работы № 3-6. Эти лабораторные работы выставлены в электронном виде во внутренней сети БГПУ и системе электронного обучения и студенты имеют к ним свободный доступ.

Советы по подготовке к экзамену:

При подготовке к экзамену особое внимание следует обратить на следующее:

1. К моменту начала подготовки к экзамену студент должен проработать все вопросы, оставленные на самостоятельное изучение.

2. Процесс подготовки к экзамену будет проходить также легче, если к моменту начала подготовки студент пройдет весь курс лабораторных работ, поскольку в нем частично повторяются, расширяются и закрепляются некоторые темы теоретического курса.

Рекомендации по работе с литературой:

При изучении тем курса, оставленных на самостоятельное изучение, за основной источник подготовки можно взять литературные источники № 1-3. Для выполнения лабораторных работ целесообразно будет использовать литературный источник № 3.

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

№	Наименование раздела (темы)	Формы/виды самостоятельной работы	Количество часов, в соответствии с учебно-тематическим планом
1.	1. Многоуровневая компьютерная организация	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	4
2.	2. Цифровой логический уровень. Основные схемы	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	4
3.	3. Принципы организации памяти. ОЗУ. ПЗУ. КЭШ-память. Внешняя память	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	10
4.	4. Структурная организация микропроцессора.	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ. Подготовка к лабораторным работам.	6
5.	5. Шины. Работа шин. Организация ввода-вывода. Механизм прерываний	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	4
6.	6.Архитектура команд. Форматы команд. Введение в язык Ассемблер	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	10
7.	7. Уровень ОС. Виртуальная память	Проработка теоретического материала по конспектам лекций и в СЭО БГПУ.	8
8.	8. Внешние устройства ввода-вывода	Проработка теоретического материала по конспектам лекций и информационным источникам.	8
	Всего		**54**

5 ПРАКТИКУМ ПО ДИСЦИПЛИНЕ

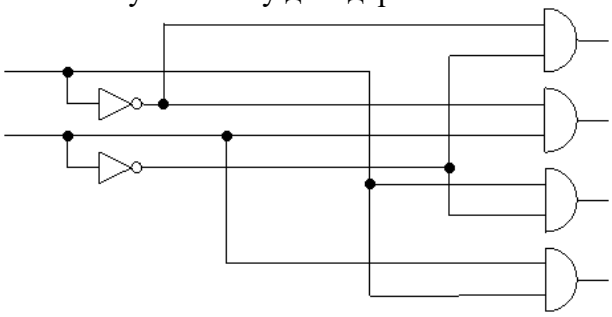
Тема 1. Схемы цифрового логического уровня

Содержание

Работа ведется в программе "Модель программируемой логической матрицы". Вышеуказанная программы находится по следующему адресу во внутренней сети: http://192.168.0.205/ - Подразделения – Кафедра информатики – Учебные материалы – Архив – Юрий Григорьевич Вахмянин – Архитектура компьютера - Лабораторная работа №

Задания:

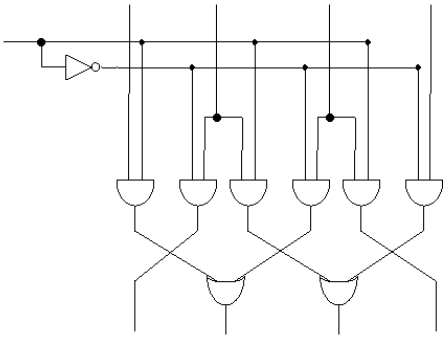
1. Реализуйте схему декодера на ПЛМ.



Для чего используется декодер?

2. На основе созданного декодера постройте демультиплексор. Для каких целей применяется демультиплексор?

3. Создайте схему арифметического сдвига.



Объясните как она работает.

4. Преобразуйте схему, чтобы она осуществляла циклический сдвиг.

5. Создайте схему для нижеприведенной таблицы истинности

1	**2**	**3**	**4**	**OUT**
0	0	0	0	0
и т.д.				0
1	0	0	1	0
1	0	1	0	1
и т.д.				1
1	1	1	1	1

6. Создайте схему для суммирования двух двухразрядных двоичных чисел (результат трехразрядный).

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).

2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info

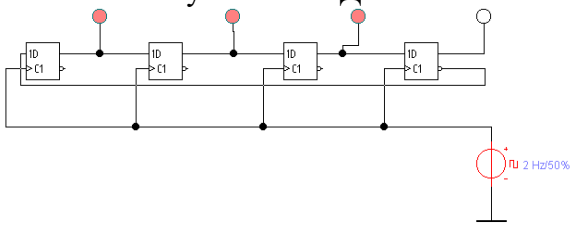
3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).

4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).

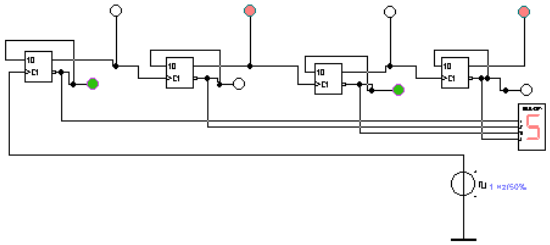
Тема 2. Схемы цифрового логического уровня

Содержание

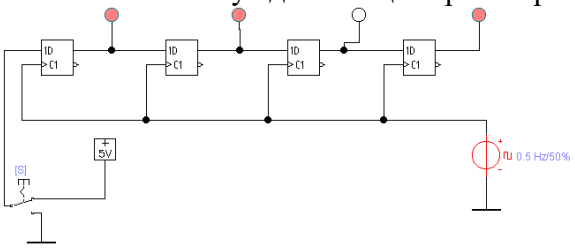
1. Составьте схему счетчика Джонсона



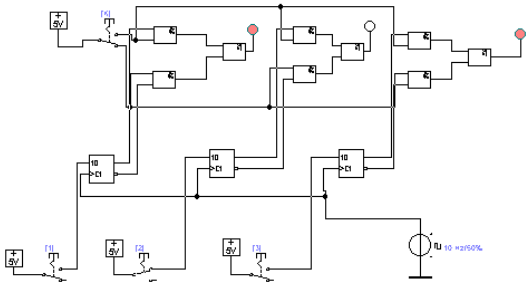
2. Составьте схему асинхронного суммирующего счётчика на D-триггерах



3. Составьте схему сдвигающего регистра.



4. Составьте схему регистра параллельного занесения двоичного кода



5. Составьте схему трехбитного регистра на D-триггерах.
6. Дополните схему трехбитного регистра с возможностью сброса всех триггеров.

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).
2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info
3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).
4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).

Тема 3. Машинное исполнение программ

Содержание

Работа ведется в программе Debug. Для открытия программы нажмите кнопку "Пуск" - Выполнить - в текстовом поле "Открыть" напишите debug.exe.

Краткая справка по программе Debug

Программа DEBUG позволяет просматривать память, вводить программы и осуществлять трассировку (пошаговое выполнение программы).

Команда DEBUG	**Описание действия команды**	**Формат записи**	**Пример записи команды**
Q	выход из программы	Q	Q
D	Команда просмотра дампа памяти	D сегмент:смещение	D 400:30 или D DS:06
E	Команда ввода программы в ячейки, начиная с адреса сегмент: смещение	E сегмент:смещение байт байт… *Где последовательность байт - есть программа в машинных кодах*	E CS:100 B8 23 01… *B8 23 01… - есть программа в машинных кодах*
R	Отображение состояния всех регистров (или просмотр/изменение состояния одного регистра)	R или R регистр	R или R AX
T	Пошаговое выполнение программы (один шаг выполнения программы)	T	T
A	Режим приема команд ассемблера и преобразование их в машинные коды	A 100 *Переход в этот режим и установка*	A 110

		начального смещения в сегменте кодов 100	
U	Показывает машинные коды для команд ассемблера	U xxxx,yyyy *xxxx-начальное смещение в CS, yyyy-конечное смещение в CS*	U 100,106

1. Чтобы установить регистр указателя команд на нужное смещение относительно начала сегмента кодов, необходимо ввести

```
R IP <нажать Enter>
```

Ввести нужное смещение и нажать Enter

Обычно начальным смещение в сегменте кодов является `100h`, то есть при вводе программы в память пишут:

```
E CS:100 …
```

Если допущена ошибка в программе, то надо заново переписать команду

```
E CS:100 …
```

Если надо дописать программу, то надо вычислить соответствующее смещение от начала сегмента так, чтобы дописываемая часть стыковалась с уже написанной.

Например, была введена программа `E CS:100 AB C8 00 05`, тогда чтобы дописать еще какую-либо команду (например `C1 C2`) надо записать `E CS:104 C1 C2`.

2. Команда отладчика `A xxxx` позволяет вводить в память машины команды на языке ассемблера начиная с адреса `CS:xxxx`.

3. Команда `U 100,106` - показывает соответствие машинных команд, находящихся начиная со смещения `100` в сегменте кодов и заканчивая смещением `106`, их ассемблерным аналогам. Например, `B025  MOV AL,25`

4. Ввод в сегмент кода: `E CS:...` , в сегмент данных: `E DS:...`. Соответственно, просмотр содержимого сегмента кода: `D CS:...` , сегмента данных: `D DS:...`.

Задания:

Задание 1. *(сегмент кодов)* Ввести данную программу в машинных кодах в память машины в сегмент кодов со смещением 100h, запустить ее на выполнение и проследить выполнение каждой команды в режиме трассировки. Записать (на бумаге) последовательность мнемонических кодов операций соответствующую последовательности машинных команд и записать значения регистров AX,BX,DX после выполнения каждой команды.

Команда	**Назначение**
B82301	Переслать значение 0123h в AX
052500	Прибавить значение 0025h к AX
8BD8	Переслать содержимое AX в BX
03D8	Прибавить содержимое AX к BX
8BCB	Переслать содержимое BX в CX
2BC8	Вычесть содержимое AX из CX
2BC0	Вычесть содержимое AX из AX (очистка AX)
90	Нет операции

Задание 2. *(сегмент кодов)* По аналогии с предыдущим заданием ввести в память машины следующую программу в машинных кодах, запустить ее на выполнение и проследить ее выполнение в режиме трассировки. Записать (на бумаге) последовательность мнемонических кодов операций, соответствующую последовательности машинных команд и записать значения регистров AX, BX после выполнения каждой команды. Назвать результат выполнения программы, который находится в регистре BX.

```
Переслать значение 05FFh в AX
Прибавить значение 0001h к AX
Переслать содержимое AX в BX
Переслать значение 066h в AX
```

```
Прибавить содержимое AX к BX
Нет операции
```

Задание 3. *(сегмент данных)* Ввести в память машины два байта данных (41h 42h) в **сегмент данных** со смещением 0. Ввести данную программу в машинных кодах в память машины в **сегмент кодов** со смещением 100h, запустить ее на выполнение и проследить выполнение каждой команды в режиме трассировки. Записать (на бумаге) последовательность мнемонических кодов операций соответствующую последовательности машинных команд и записать значения регистров AX, BX после выполнения каждой команды. В режиме просмотра дампа просмотреть сегмент данных до выполнения программы и после. Что произошло с данными?

Команда	**Назначение**
A1 00 00	Переслать слово (два байта), начинающееся в сегменте данных DS по адресу 0000, в регистр AX
05 20 20	Прибавить значение 2020h к AX
A3 00 00	Переслать содержимое регистра AX в слово, начинающееся в DS по адресу 0000
90	Нет операции

Задание 4. *(сегмент данных)* По аналогии с заданием 3 записать в память машины в **сегмент данных** два байта (4Dh 41h) со смещением 0, ввести в память машины программу в машинных кодах по копированию этих двух байт в сегменте данных по адресу со смещением 2 относительно начала сегмента данных. Запустить ее на выполнение и проследить выполнение каждой команды в режиме трассировки. Записать (на бумаге) последовательность мнемонических кодов операций, соответствующую последовательности машинных команд и записать значения регистра AX после выполнения каждой команды. В режиме просмотра дампа просмотреть сегмент данных до выполнения программы и после. Что произошло с данными?

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).
2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info
3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).
4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).
5. Майко, Г.В. Assembler для IBM PC / Г.В. Майко. – М. : Бизнес-Информ, Сирин, 1999. – 212 с.
6. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с.

Тема 4. Введение в язык Ассемблер. Основные команды

Содержание

Работа ведется в программе Debug. Для открытия программы нажмите кнопку "Пуск" - Выполнить - в текстовом поле "Открыть" напишите debug.exe.

Краткая справка по программе Debug

Программа DEBUG позволяет просматривать память, вводить программы и осуществлять трассировку (пошаговое выполнение программы).

Команда DEBUG	Описание действия команды	Формат записи	Пример записи команды
Q	выход из программы	Q	Q
D	Команда просмотра дампа памяти	D сегмент:смещение	D 400:30 или D DS:06
E	Команда ввода программы в ячейки, начиная с адреса сегмент: смещение	E сегмент:смещение байт байт… *Где последовательность байт - есть программа в машинных кодах*	E CS:100 B8 23 01… *B8 23 01… - есть программа в машинных кодах*
R	Отображение состояния всех регистров (или просмотр/изменение состояния одного регистра)	R или R регистр	R или R AX
T	Пошаговое выполнение программы (один шаг выполнения программы)	T	T
A	Режим приема команд ассемблера и преобразование их в машинные коды	A 100 *Переход в этот режим и установка начального смещения в сегменте кодов 100*	A 110
U	Показывает машинные коды для команд ассемблера	U xxxx,yyyy *xxxx-начальное смещение в CS, yyyy-конечное смещение в CS*	U 100,106

1. Чтобы установить регистр указателя команд на нужное смещение относительно начала сегмента кодов, необходимо ввести

```
R IP <нажать Enter>
```

Ввести нужное смещение и нажать Enter

Обычно начальным смещение в сегменте кодов является `100h`, то есть при вводе программы в память пишут:

```
E CS:100 …
```

Если допущена ошибка в программе, то надо заново переписать команду

```
E CS:100 …
```

Если надо дописать программу, то надо вычислить соответствующее смещение от начала сегмента так, чтобы дописываемая часть стыковалась с уже написанной.

Например, была введена программа `E CS:100 AB C8 00 05`, тогда чтобы дописать еще какую-либо команду (например, `C1 C2`) надо записать `E CS:104 C1 C2`.

2. Команда отладчика `A xxxx` позволяет вводить в память машины команды на языке ассемблера начиная с адреса `CS:xxxx`.

3. Команда `U 100,106` - показывает соответствие машинных команд, находящихся начиная со смещения `100` в сегменте кодов и заканчивая смещением `106`, их ассемблерным аналогам. Например, `B025 MOV AL,25`

4. Ввод в сегмент кода: `E CS:...` , в сегмент данных: `E DS:...`. Соответственно, просмотр содержимого сегмента кода: `D CS:...` , сегмента данных: `D DS:...`.

Задания:

Задание 1. Напишите программу для вычисления 12-и чисел Фибоначчи: 1, 1, 2, 3, 5, 8, 13,... (каждое число в последовательности представляет собой сумму двух предыдущих чисел). Для организации цикла используйте команду `LOOP`. Программу напишите и выполните трассировку с помощью отладчика DEBUG. Результат выполнения программы (последовательность из 12 чисел - байт) должен находиться в сегменте данных.

Задание 2. Пусть в сегменте данных по смещению 0 определено поле, содержащее слово `personal`. (Использовать команду ассемблера `DB 'строка'`).

Напишите программу

а) преобразующую строчные буквы данного слова в заглавные и записывающую результат в сегменте данных по смещению 10;

б) то же самое, только для преобразования заглавных букв в строчные и помещения результирующего слова в сегмент данных по смещению 20.

Примечание. Допускается использовать команды типа

`MOV [BX+10], AX` или `MOV AX,[BX+12]`

`INC AX` - увеличение содержимого AX на 1

`DEC AX` - уменьшение содержимого AX на 1

Задание 3. Используя команды сдвига (`SHL`, `SHR`), пересылки (`MOV`) и сложения (`ADD`),

а) умножьте содержимое регистра `AX` на 10.

б) разделите содержимое регистра `AX` на 8.

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).

2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info

3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).

4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).

5. Майко, Г.В. Assembler для IBM PC / Г.В. Майко. – М. : Бизнес-Информ, Сирин, 1999. – 212 с.

6. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с.

Тема 5. Введение в язык Ассемблер. Команды сравнения и условных и безусловных переходов

Содержание

Работа ведется в программе Debug. Для открытия программы нажмите кнопку "Пуск" - Выполнить - в текстовом поле "Открыть" напишите debug.exe.

Краткая справка по программе Debug

Программа DEBUG позволяет просматривать память, вводить программы и осуществлять трассировку (пошаговое выполнение программы).

Команда DEBUG	Описание действия команды	Формат записи	Пример записи команды
Q	выход из программы	Q	Q
D	Команда просмотра дампа памяти	D сегмент:смещение	D 400:30 или D DS:06
E	Команда ввода программы в ячейки, начиная с адреса сегмент: смещение	E сегмент:смещение байт байт… *Где последовательность байт - есть программа в машинных кодах*	E CS:100 B8 23 01… *B8 23 01… - есть программа в машинных кодах*
R	Отображение состояния всех регистров (или просмотр/изменение состояния одного регистра)	R или R регистр	R или R AX
T	Пошаговое выполнение программы (один шаг выполнения программы)	T	T
A	Режим приема команд ассемблера и преобразование их в машинные коды	A 100 *Переход в этот режим и установка начального смещения в сегменте кодов 100*	A 110
U	Показывает машинные коды для команд ассемблера	U xxxx,yyyy *xxxx-начальное смещение в CS, yyyy-конечное смещение в CS*	U 100,106

Задания:

Задание 1. Пусть в сегменте кодов определена строковая переменная с помощью команды `DB 'Personal Computer'`. Напишите программу преобразующую строчные буквы данного слова в заглавные и наоборот, и записывающую результат в сегменте данных по смещению 0;

Примечание. Допускается использовать команды типа, впрочем, другие тоже можно

- Условного и безусловного переходов (для организации подпрограмм);
- `CMP` - команду сравнения;
- `LOOPxx` - цикл;
- `PUSH, POP` (для сохранения значений регистра DX при вызове подпрограмм);
- `MOV [BX+10], AX` или `MOV AX,[BX+12];`
- `INC AX` - увеличение содержимого AX на 1;
- `DEC AX` - уменьшение содержимого AX на 1.

Задание 2. Напишите программу на ассемблере, реализующую алгоритм:

`AX=12h`

```
BX=16h
CX=05h
DX=ffh
Если (AX and BX) > (CX or DX) то AX=FFFFh иначе AX=0000
```

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).
2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info
3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).
4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).
5. Майко, Г.В. Assembler для IBM PC / Г.В. Майко. – М. : Бизнес-Информ, Сирин, 1999. – 212 с.
6. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с.

Тема 6. Механизм прерывания

Содержание

Работа ведется в программе Debug. Для открытия программы нажмите кнопку "Пуск" - Выполнить - в текстовом поле "Открыть" напишите debug.exe.

Выполнять код, использующий вызовы int нужно не с помощью команды t, а с помощью команды g (выполнить серию команд), которая имеет формат

g=начальный_адрес конечный_адрес

например, команда

g=100 120

выполнит все команды, начиная с адреса cs:0100 до адреса cs:0120. При этом не обязательно настраивать регистр ip.

Задание 1. Написать программу на ассемблере, вызывающую прерывание 10h для очистки фрагмента экрана:
а) весь экран;
б) начиная с позиции (x1,y1) до позиции (x2,y2).
Задание 2. Напишите программу на ассемблере, вызывающую прерывание 21h для вывода символьной строки "PC User Group" на экран.
Задание 3. Напишите программу на ассемблере, вызывающую прерывание 21h для ввода символьной строки с клавиатуры в память.
Задание 4. Напишите программу на ассемблере, использующую прерывание 21h для вывода на экран полного набора ASCII-символов (коды от 0 до 255).
Задание 5. Напишите программу на ассемблере, использующую прерывание 21h для вывода на экран полного набора ASCII-символов кроме управляющих кодов, находящихся в диапазоне от 08h до 0Dh.

Приложение 1

Очистка экрана (INT 10h)

Текст на экране остается до тех пор, пока не будет смещен в результате прокручивания ("скроллинга") или переписан на том же месте другим текстом. Экран может быть очищен. Очищаемая область экрана может начинаться в любой позиции и заканчиваться в любой другой позиции с большим номером.

Начальная позиция (строка и столбец) заносится в регистр CX, конечное в DX, значение 07 в регистр BH и 0600h в AX.

Например, очистка всего экрана:

```
MOV     AX,0600h ;AH 06 (прокрутка) AL 00 (весь экран)
MOV     BH,07    ;Нормальный атрибут (черно/белый)
MOV     CX,0000  ;Верхняя левая позиция
MOV     DX,184Fh ;Нижняя правая позиция
INT     10h      ;Передача управления в BIOS
```

Значение 06 в регистре AH указывает команде INT 10h на выполнение операции очистки экрана.

Операция очистки экрана заключается в заполнении указанных областей пробелами.

Обычно программы должны обеспечивать хотя бы минимальный диалог с пользователем: обмен данными, сообщения об ошибках, о завершении работы и т.п. Эти функции обычно осуществляются посредством экрана и клавиатуры.

Приложение 2

Экранные и клавиатурные операции (Прерывание DOS INT 21h)

При вызове прерывания ДОС INT 21h используется так называемый файловый номер (file handle). Другими словами, все возможные устройства ввода и вывода разбиты на классы и эти классы пронумерованы, эти классы носят наименование "файл". Существуют следующие стандартные файловые номера:

0	Ввод, обычно с клавиатуры (CON)
1	Вывод, обычно на экран (CON)
2	Вывод по ошибке, обычно на экран (CON)
3	Ввод-вывод на внешнее устройство (AUX)
4	Вывод на печать (LPT1 или PRN)

Для ввода-вывода используется прерывание DOS INT 21h.

Необходимая функция запрашивается через регистр AH (3Fh - для ввода,40h - для вывода)

В регистр CX заносится число байтов для ввода-вывода.

В регистр DX заносится адрес области ввода-вывода.

В регистр BX заносится файловый номер.

В результате выполнения операции ввода-вывода происходит следующее:

В случае успеха - очищается флаг переноса CF, в регистр AX устанавливается действительное число байт, участвующих в операции.

В случае ошибки - устанавливается флаг переноса CF, в регистр AX устанавливается код ошибки (в данном случае 6).

Таким образом, единственный способ проверить, произошла ли ошибка во время операции ввода-вывода - проверить флаг CF.

Рассмотрим пример, иллюстрирующий операцию вывода на экран:

```
MOV     AH,40h   ;Запрос на вывод
MOV     BX,01h   ;Выводное устройство
MOV     CX,20h   ;Максимальное число байтов
LEA     DX,00h   ;Адрес области данных
INT     21h      ;Вызов ДОС
```

Рассмотрим пример, иллюстрирующий операцию ввода с клавиатуры:

```
MOV     AH,3Fh   ;Запрос на ввод
MOV     BX,00h   ;Номер для клавиатуры
MOV     CX,0Ch   ;Максимальное число байт для ввода
LEA     DX,00h   ;Адрес области ввода
```

```
INT             21h                     ;Вызов ДОС
```

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).

2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info

3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).

4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).

5. Майко, Г.В. Assembler для IBM PC / Г.В. Майко. – М. : Бизнес-Информ, Сирин, 1999. – 212 с.

6. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с.

Тема 7. Введение в язык Ассемблер. Ассемблерные вставки в программы на Pascal и C++

Содержание

Откройте и прочитайте электронный документ "Ассемблерные вставки на Turbo Pascal" (место его нахождения в сети укажет преподаватель). Ознакомьтесь с помощью каких операторов создается ассемблерная вставка и в каких случаях они используются, ознакомьтесь с еще не изученными командами языка Ассемблер.

Наберите и выполните следующие программы на языке Turbo Pascal, исправьте, где это необходимо синтаксические ошибки.

Команды mov, add, sub.

1. Дана переменная типа integer. Присвоить ей значение 10 с помощью ассемблера. Затем вывести на экран стандартным способом.

```
var a:integer;
begin
asm
mov bx,10
mov a,bx
end;
writeln(a);
end.
```

2. Даны две переменных a (тип integer) и b (тип byte). Присвоить b – значение a. Вывести на экран.

```
var a:integer;
   b:byte;
begin
writeln('Введите a:');
readln(a);
asm
mov ax,a
mov bl,al
mov b,bl
end;
writeln(b);
end.
```

3. Даны две переменные a, b (тип byte). С клавиатуры вводятся их значения.В ассемблерной вставке поменять значения переменных местами. Далее вывести значения обоих переменных на экран.

```
var a,b:byte;
begin
write('Введите a:');
readln(a);
write('Введите b:');
readln(b);
asm
mov al,a
mov bl,b
mov ch,al
mov al,bl
mov bl,ch
mov a,al
mov b,bl
end;
writeln('A=',a);
writeln('B=',b);
end.
```

4. Дана переменная типа integer (word) – ее значение вводится с клавиатуры. Вывести байты старший и младший из которых она состоит.

```
var a:integer;
   h,l:byte;
begin
write('Введите a:');
readln(a);
asm
mov ax,a
mov h,ah
mov l,al
end;
writeln(a,'=256*',h,'+',l);
end.
```

5. Сложить два числа a,b.

```
var a,b:integer;
   sum:integer;
begin
writeln('Введите a и b:');
readln(a);
readln(b);
asm
mov ax,a
add ax,b
mov sum,ax
end;
writeln('Сумма:',sum);
end.
```

6. Сложить два числа.

Замечание:

Одно число имеет тип integer, второе число – тип byte.

```
var a:integer;
   b:byte;
```

```
   sum:integer;
begin
write('Введите a (значения от -32768 до 32768):');
readln(a);
write('Введите b (значения от 0 до 255):');
readln(b);
asm
mov ax,a
{складывать можно только AX и BX.
A BX состоит из BH+BL (старшая и младшая часть)
В младшую часть всегда помещаются числа размером byte.
А чтобы старшая чать не мешалась делаем ее равной нулю}
mov bl,b
mov bh,0
add ax,bx
mov sum,ax
end;
writeln('Сумма:',sum);
end.
```

7. Сложить три числа.

Замечание:

Числа имеют тип integer;

```
var a,b,c:integer;
   sum:integer;
   w:word;
begin
write('Введите a (значения от -32768 до 32768):');
readln(a);
write('Введите b (значения от -32768 до 32768):');
readln(b);
write('Введите c (значения от -32768 до 32768):');
readln(c);
asm
mov ax,a
mov bx,b
add ax,bx
add ax,c
mov w,ax
end;
writeln('Сумма:',w);
end.
```

8. К числу A (тип Integer) введенному с клавиатуры прибавить число 100.

```
var a:integer;
begin
write('Введите a (значения от -32768 до 32768):');
readln(a);
asm
mov ax,a
add ax,100
mov a,ax
end;
writeln('A=',a);
```

```
end.
```

9. От числа введенного с клавиатуры отнять число 10 с помощью команды сложения.

```
var a:integer;
begin
write('Введите a (значения от -32768 до 32768):');
readln(a);
asm
mov ax,a
add ax,-10
mov a,ax
end;
writeln('A=',a);
end.
```

После этих задач у вас может возникнуть вопрос: что будет если сложить числа сумма которых будет больше 32768, ведь это максимально допустимое число? Для этого нам будет необходимо понять саму суть сложения двоичных чисел – ведь именно в двоичной системе складывает компьютер числа.

К примеру сложим 2+1=3

В двоичной системе 10+1=11

2+2=4

В двоичной 10+10=100

2+3=5

11+11=101

и т.д. При увеличении разряда числа слева добавляется одна единица (см. системы счисления)

Тип integer занимает 2 байта памяти т.е. 16 битов (нулей и или единиц). Причем 16 бит (старший) является служебным и равен 1 (установлен) когда число отрицательное или 0 (сброшен) когда положительное. Таким образом реально мы можем использовать только 15 битов – отсюда и число 32768 = 215.

И если число получилось при сложении больше 15 ячеек, что лишняя часть (все что следует за 15-ым битом – сбрасывается) т.е. реально из числа вычитается 215.

Работа со стеком. Push и Pop.

Push – помещает значения в стек

Pop – вынимает значения из стека.

1. Поместить значение переменной a в стек. Затем взять из стека и вывести на печать.

```
uses crt;
var a:integer;
begin
clrscr;
a:=100;
asm
 push a
mov a,200
pop a
end;
writeln('a=',a);
readln;
end.
```

2. Поместить в стек число 255. Затем верхнее значение стека поместить в переменную w.

```
uses crt;
var a:integer;
begin
```

```
clrscr;
a:=100;
writeln('a=',a);
asm
push 255
pop a
end;
writeln('a=',a);
readln;
end.
```

3. Поменять значения двух переменных местами используя стек.

```
uses crt;
var a,b:integer;
begin
clrscr;
a:=10;
b:=20;
asm
push a
push b
{это для наглядности см. ниже}
mov a,49
mov b,23
pop a
pop b
end;
writeln('a=',a);
writeln('b=',b);
readln;
end.
```

4. Даны переменные a,b,c,d,e,f,g,h,j,k,l. Поменять из значения таким образом, чтобы значение a было в l, b – в k и т.д.

```
uses crt;
var a,b,c,d,e,f,g,h,k,l:integer;
begin
clrscr;
a:=1;b:=2;c:=3;d:=4;e:=5;f:=6;g:=7;h:=8;k:=9;l:=10;
asm
push a;push b;push c;push d;push e
push f;push g;push h;push k;push l
pop l;pop k;pop h;pop g;pop f
pop e;pop d;pop c;pop b;pop a
end;
writeln(a);
writeln(b);
writeln(c);
writeln(d);
writeln(e);
writeln(f);
writeln(g);
writeln(h);
writeln(k);
```

```
writeln(l);
readln;
end.
```

5. Даны переменные a,b,c,d,e,f,g,h,j,k,l. Поместить их значения в стек. Затем поместить обратно и вывести на печать.

```
uses crt;
var a,b,c,d,e,f,g,h,k,l:integer;
begin
clrscr;
a:=1;b:=2;c:=3;d:=4;e:=5;f:=6;g:=7;h:=8;k:=9;l:=10;
asm
push a;push b;push c;push d;push e
push f;push g;push h;push k;push l
pop l;pop k;pop h;pop g;pop f
pop e;pop d;pop c;pop b;pop a
end;
writeln(a);
writeln(b);
writeln(c);
writeln(d);
writeln(e);
writeln(f);
writeln(g);
writeln(h);
writeln(k);
writeln(l);
readln;
end.
```

6. Поместить число типа byte в стек, затем взять из стека и вывести на печать.

```
uses crt;
var b:byte;
begin
clrscr;
b:=20;
asm
mov al,b
mov ah,0
push ax
pop bx
mov b,bl
end;
writeln('b=',b);
readln;
end.
```

Указания: постарайтесь найти принципиальные различия в данных программах. Обратите внимание на последовательность помещения значений в стек и последовательность из изъятия из стека.

```
uses crt;
var a,b:integer;
begin
clrscr;
a:=10;
```

```
b:=20;
asm
push a
push b
mov a,49
mov b,23
pop a
pop b
end;
writeln('a=',a);
writeln('b=',b);
readln;
end.
```

Потренируйтесь с программами из примеров. Попробуйте изменить или переставить значения переменных.

Команда сравнения и перехода. Ветвления в ассемблере.

CMP и JMP

1. Найти минимум из 3 чисел.

```
var a,b,c,min:integer;
begin
a:=6;
b:=3;
c:=1;
  asm
mov ax,a
mov bx,b
mov cx,c
cmp ax,bx
jnc @lp1
mov dx,ax
jmp @cont1
@lp1:
mov dx,bx
@cont1:
cmp dx,cx
jc @cont2
mov dx,cx
@cont2:
mov min,dx
end;
writeln(min);
end.
```

2. Дан x. Найти y если

```
var x,y:integer;
begin
x:=10;
  asm
mov ax,x
cmp ax,10
{Дело в том, что условие x<=10
выполняться не будет по этому
необходима дополнительная
проверка - равно ли число
x десяти ли нет}
jz @cont1
```

```
jnc @cont2
@cont1:
add ax,100
@cont2:
add ax,2
mov y,ax
end;
        writeln(y);
end.
```

3. Даны числа a и b. Найти их сумму по модулю. Результат поместить в переменную sum.

```
var a,b,sum:integer;
begin
a:=-10;
b:=20;
  asm
mov ax,a
mov bx,b
cmp ax,0
jns @cont1
mov dx,65535
sub dx,ax
inc dx
mov ax,dx
@cont1:
cmp bx,0
jns @cont2
mov dx,65535
sub dx,bx
inc dx
mov bx,dx
@cont2:
add ax,bx
mov sum,ax
end;
        writeln(sum);
end.
```

4. Дано число a. Найти s, если s=sgn(a).

```
var a,s:integer;
begin
a:=-2;
   asm
mov ax,a
mov s,ax
cmp ax,0
jz @exit
js @cont
mov ax,1
mov s,ax
jmp @exit
   @cont:
mov ax,-1
mov s,ax
   @exit:
 end;
        writeln(s);
end.
end.
```

Организация циклов.Команды loop и jcxz.

Организация цикла с помощью оператора jnz:
1. Какому-то регистру присваивается начальное знаечение. Например, регистру CX.
2. Устанавливается метка. Например @lp1
3. Выполняется тело цикла.
4. Уменьшается на единицу регистр CX.
5. Пока CX не равен нулю выполняется переход на установленную метку. Jnz @lp1

Существуют еще два оператора которые могут пригодиться в программировании циклов. Рассмотрим следующий вариант цикла
1. Регистру CX присваивается начальное значение.
2. Устанавливается метка. Например @lp1
3. Выполняется тело цикла.
4. Loop @lp1
Что делает команда loop? Как видно из примера одна заменяет сразу два пункта из предыдущего примера (4-5). А именно автоматически уменьшает на единицу регистр CX и пока он не равен нулю переходит на указанную метку. Из чего следует, что loop может использовать в качестве счетчика цикла только регистр CX.
Команда jcxz – выполняется аналогично команде jz, только еще осуществляется проверка на то равен ли регистр CX нулю.

1. Посчитать n факториал. При этом дополнительно (в PASCAL) проверить чтобы вводимое n не выходило за пределы допустимых значений.

```
uses crt;
var n:integer;
begin
clrscr;
write('Введите N:');
readln(n);
if n>9 then begin
            writeln('Нельзя вводить N>9!');
            halt;
           end;
asm
mov cx,n
mov ax,1
@lp1:
mul cl
loop @lp1
mov n,ax
end;
writeln('Факториал равен:',n);
readln;
end.
```

2. Дано целое число a и натуральное (целое неотрицательное) число n. Вычислить a в степени n. Другими словами, необходимо составить программу, при исполнении которой значения переменных a и n не меняются, а значение некоторой другой переменной (например, b) становится равным a в степени n.

```
uses crt;
var a:integer;
   n:byte;
begin
clrscr;
write('A=');
```

```
readln(a);
write('N=');
readln(n);
asm
mov bl,0
mov ax,1
mov bh,n
mov cx,a
@lp1:
inc bl
mul cx
cmp bl,bh
jnz @lp1
mov a,ax
end;
writeln('a^n=',a);
readln;
end.
```

3. Даны натуральные числа a, b. Вычислить произведение a*b, используя в программе команду сложения.

```
uses crt;
var a,b:byte;
{мы не можем допустить чтобы умножались числа большие чем 255
так как 255*255=65535, что уже может вызвать переполнение}
   res:integer;
{а вот резуьтат может оказаться большим числом}
begin
clrscr;
write('A=');
readln(a);
write('b=');
readln(b);
asm
mov bl,0
mov ax,0
mov cl,a
mov ch,0
mov dl,b
@lp1:
inc bl
add ax,cx
cmp bl,dl
jnz @lp1
mov res,ax
end;
writeln('a*b=',res);
readln;
end.
```

Возьмите варианты индивидуальных задач для решения у преподавателя. Решите указанные задачи из нижеследующего списка в системе программирования Turbo Pascal.

После отчета по решенным заданиям, разделитесь на пары и решите указанные преподавателем из нижеизложенного списка задачи на языке C++.

Литература:

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника).

2. Гуров, В., Чуканов, В. Архитектура и организация ЭВМ. – Национальный открытый университет «Интуит», Национальный исследовательский ядерный университет «МИФИ». Режим доступа: http://www.intuit.ru/studies/courses/60/60/info

3. Новожилов, О.П. Архитектура ЭВМ и систем : учеб. пособие для бакалавров / О. П. Новожилов. – М. : Юрайт, 2012, 2015. – 527 с. – (Бакалавр. Базовый курс).

4. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science).

5. Майко, Г.В. Assembler для IBM PC / Г.В. Майко. – М. : Бизнес-Информ, Сирин, 1999. – 212 с.

6. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с.

6 ДИДАКТИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ КОНТРОЛЯ (САМОКОНТРОЛЯ) УСВОЕННОГО МАТЕРИАЛА

6.1 Оценочные средства, показатели и критерии оценивания компетенций

Индекс компетенции	**Оценочное средство**	**Показатели оценивания**	**Критерии оценивания сформированности компетенций**
УК-1, ОПК-8 ПК-2	Собеседование	Низкий (неудовлетворительно)	Студент отвечает неправильно, нечетко и неубедительно, дает неверные формулировки, в ответе отсутствует какое-либо представление о вопросе
		Пороговый (удовлетворительно)	Студент отвечает неконкретно, слабо аргументировано и не убедительно, хотя и имеется какое-то представление о вопросе
		Базовый (хорошо)	Студент отвечает в целом правильно, но недостаточно полно, четко и убедительно
		Высокий (отлично)	Ставится, если продемонстрированы знание вопроса и самостоятельность мышления, ответ соответствует требованиям правильности, полноты и аргументированности.
УК-1, УК-6, ОПК-8 ПК-2	Обсуждение	Низкий (неудовлетворительно)	Ставится, если: обнаружено незнание или непонимание большей, или наиболее важной части учебного материала; допущены ошибки в определении понятий, при использовании терминологии, которые не исправлены после нескольких наводящих вопросов; не сформированы компетенции, умения и навыки публичной речи, аргументации, ведения дискуссии и полемики, критического восприятия информации.

		Пороговый (удовлетворительно)	Ставится, если: неполно или непоследовательно раскрыто содержание предложений по решению при обсуждении, но показано общее понимание вопроса и продемонстрированы умения, достаточные для дальнейшего усвоения материала; имелись затруднения или допущены ошибки в определении понятий, использовании терминологии, исправленные после нескольких наводящих вопросов; при неполном знании теоретического материала выявлена недостаточная сформированность компетенций, умений и навыков, студент не может применить теорию в новой ситуации.
		Базовый (хорошо)	Ставится, если: ответ удовлетворяет в основном требованиям на оценку «5», но при этом имеет один из недостатков: в усвоении учебного материала допущены небольшие пробелы, не исказившие содержание высказываний при обсуждении; допущены один – два недочета в формировании навыков публичной речи, аргументации, критического восприятия информации.
		Высокий (отлично)	Ставится, если студент полно усвоил учебный материал; проявляет навыки анализа, обобщения, критического осмысления, публичной речи, аргументации, критического восприятия информации; точно используется терминология; продемонстрировано усвоение ранее изученных сопутствующих вопросов.
		Низкий (неудовлетворительно)	Ответ студенту не зачитывается если: • задание выполнено менее, чем на половину; • студент обнаруживает незнание большей части соответствующего материала, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно излагает материал; • студент пропустил основные и дополнительные сроки сдачи задач и заданий.
УК-1, УК-6, ОПК-8 ПК-2	Задачи и задания	Пороговый (удовлетворительно)	Задание выполнено более, чем на половину. Студент обнаруживает знание и понимание основных положений задания, но:

			• излагает материал неполно и допускает неточности в определении понятий; • не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; • излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого; • студент пропустил основные сроки сдачи задач и заданий.
		Базовый (хорошо)	Задание в основном выполнено. Ответы правильные, но: • в ответе допущены малозначительные ошибки и недостаточно полно раскрыто содержание вопроса; • допущено 1-2 недочета в последовательности и языковом оформлении излагаемого; • студент сдал задачи, задания в назначенные основные сроки сдачи.
		Высокий (отлично)	Задание выполнено в максимальном объеме. Ответы полные и правильные. • студент полно излагает ответ на вопрос; • обнаруживает понимание материала, может обосновать свои суждения; • излагает материал последовательно и правильно с точки зрения норм литературного языка; • студент сдал задачи, задания в назначенные основные сроки сдачи или с опережением.
		Зачтено	• в отчете разработанные программы реализуют требуемую в заданиях функциональность; • студент самостоятельно выполнил все этапы решения задач лабораторной работы на ЭВМ; • разработанные программы лабораторной работы используют изучаемые механизмы и в исходном тексте программ соблюдаются выбранные правила оформления кода (способ именования переменных, выравнивание строк и другие соглашения в оформлении программ) исходный код программ понятен и ясно иллюстрирует принятые проектировочные решения;

			• программы или схемы лабораторной работы возвращают верный результат для различных исходных данных; • студент осмысленно и уверенно объясняет алгоритм решения задачи или работы и построения схемы; • студент осмысленно и уверенно поясняет смысл инструкций программы на Ассемблере; • студент при обсуждении алгоритма решения задач при необходимости может самостоятельно в присутствии преподавателя внести изменения в программу; • студент создает отчет о выполнении лабораторных работ, согласно предъявляемым требованиям, и выставляет в системе электронного обучения БГПУ.
УК-1, УК-6, ОПК-8 ПК-2	Лабораторные работы	Не зачтено	• значительная часть решения программ (схем) лабораторной работы на ЭВМ выполнена студентом не самостоятельно; • разработанные программы лабораторной работы используют изучаемые механизмы и в исходном тексте программы соблюдаются выбранные правила оформления кода (способ именования переменных, выравнивание строк и другие соглашения в оформлении программ), но при этом: программа возвращает не всегда верный результат для различных исходных данных или студент не может объяснить алгоритм решения задачи на Ассемблере и/или студент не может пояснить смысл инструкций программы на Ассемблере; • при обсуждении алгоритма решения задачи студент не может самостоятельно в присутствии преподавателя внести изменения в программу; • студент не создал отчет о выполнении лабораторных работ, согласно предъявляемым требованиям, и не выставил в системе • электронного обучения БГПУ.

6.2 Промежуточная аттестация студентов по дисциплине

Вопросы экзамена

1. Понятия ВМ и компьютера. Их виды, области применения.
2. Классификации ЭВМ и их характеристики.
3. Понятие многоуровневой компьютерной организации. Трансляция и интерпретация. Понятия «программа» и «архитектура».
4. Современные многоуровневые машины. Вентили и их обозначения. Уровни -1-3.
5. Современные многоуровневые машины. Уровни 4-6. Понятие архитектуры.
6. Классическая архитектура ЭВМ. Архитектура ЭВМ Дж. фон Неймана.
7. Функциональная структура ЭВМ. Канальная и шинная системотехника.
8. Цифровой логический уровень. Понятие защелки и триггера. SR-защелка-схема и принцип работы. Понятие регистра.
9. Цифровой логический уровень. Память 4 на 3. Регистр. Буферные элементы. Два способа организации памяти. Микросхемы памяти.
10. Цифровой логический уровень. Память. ОЗУ и ПЗУ. Виды, характеристика, назначение.
11. Кэш память. Кэш-память с тремя уровнями.
12. Внешние запоминающие устройства. Их классификация. Магнитные диски.
13. Внешние запоминающие устройства. Накопители на жестких магнитных дисках.
14. Внешние запоминающие устройства. Устройства флэш-памяти.
15. Накопители на оптических дисках.
16. Цифровые диски DVD.
17. Структурная схема микропроцессора. Исполнительный блок.
18. Структурная схема микропроцессора. Устройство сопряжения с системной магистралью.
19. Структурная схема микропроцессора. Взаимодействие элементов при работе микропроцессора. Тракт данных. Алгоритм выполнения команд.
20. Микросхема процессора. Цоколевка типичного центрального процессора.
21. Классификация микропроцессоров.
22. Шины. Определение шины и ее виды.
23. Вопросы разработки шин. Ширина шины. Мультиплексная шина. Синхронизация шины.
24. Вопросы разработки шин. Арбитраж шины.
25. Механизмы работы шин. Механизм опроса. Механизм прерываний. Вектор прерывания. Виды прерываний.
26. Архитектура команд. Форматы команд. Критерии разработки для форматов команд. Расширение кода операций.
27. Определение языка Ассемблера, его особенности, области применения. Понятие мнемонического кода операции. Компоненты предложения языка Ассемблер. Понятия метки, мнемоники, операнда, комментарий, константы. Основные типы и группы команд.
28. Типы операндов. Основные способы адресации. Понятие макропроцессора, макровызов (макрос), макроопределение, макрорасширение. Области применения макрокоманд.
29. Команды пересылки данных. Арифметические и логические команды. Команды передачи управления. Команды int, call. Работа стека при вызове подпрограмм.
30. Концепция виртуальной памяти.
31. Виртуализация памяти.

32. Функции ОС по управлению памятью в мультипрограммных системах.
33. Распределение памяти.
34. Виртуальная память. Страничная организация виртуальной памяти.
35. Виртуальная память. Сегментная организация виртуальной памяти Виртуальная память. Сегментно-страничная организация виртуальной памяти.
36. Цифровые и аналоговые мониторы. Характеристики мониторов.
37. Видеомониторы на плоских панелях. Мониторы на жидкокристаллических индикаторах. Плазменные мониторы.
38. Видеоконтроллеры.
39. Принтеры. Матричные принтеры.
40. Струйные и лазерные принтеры.
41. Сканеры и их основные характеристики.
42. Типы сканеров.

Критерии оценки устного ответа на экзамене:

Промежуточная аттестация является проверкой всех знаний, навыков и умений студентов, приобретённых в процессе изучения дисциплины. Формой промежуточной аттестации по дисциплине является экзамен.

При проведении междисциплинарного экзамена в устной форме устанавливаются следующие критерии оценки знаний выпускников:

Оценка «**отлично**» – глубокие, исчерпывающие знания всего программного материала, понимание сущности и взаимосвязи рассматриваемых процессов и явлений. Логически последовательные, содержательные, полные, правильные и конкретные ответы на все вопросы экзаменационного билета и дополнительные вопросы экзаменатора. Грамотное чтение и чёткое изображении схем; использование в необходимой мере в ответах на вопросы материалов всей рекомендованной литературы. Демонстрирует прикладную направленность полученных знаний и умений и не допускает терминологических ошибок и фактических неточностей.

Оценка «**хорошо**» – твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; грамотное чтение и четкое изображение схем. Демонстрирует прикладную направленность полученных знаний и умений и допускает незначительные терминологические ошибки и фактические неточности.

Оценка «**удовлетворительно**» – твердое знание и понимание основных вопросов программы; правильные и конкретные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах экзаменатора; наличие ошибок в чтении и изображении схем; при ответах на вопросы основная рекомендованная литература использована недостаточно. Не полностью демонстрирует прикладную направленность полученных знаний и умений и допускает терминологические ошибки и фактические неточности.

Оценка «**неудовлетворительно**» – неправильный ответ хотя бы на один из основных вопросов, грубые ошибки в ответе, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы. Не демонстрирует прикладную направленность полученных знаний и умений, допускает терминологические ошибки и фактические неточности.

7 ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ В ПРОЦЕССЕ ОБУЧЕНИЯ

Информационные технологии – обучение в электронной образовательной среде с целью расширения доступа к образовательным ресурсам, увеличения контактного взаимодействия с преподавателем, построения индивидуальных траекторий подготовки, объективного контроля и мониторинга знаний студентов.

В образовательном процессе по дисциплине используются следующие информационные технологии, являющиеся компонентами Электронной информационно-образовательной среды БГПУ:

- Внутренний сайт БГПУ;
- Система электронного обучения ФГБОУ ВО «БГПУ»;
- Мультимедийное сопровождение лекций и практических занятий;
- Программа «Модель программируемой логической матрицы»;
- Программа DosBox;
- Программа Debug;
- Система программирования TurboPascal.

8 ОСОБЕННОСТИ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ ИНВАЛИДАМИ И ЛИЦАМИ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

При обучении лиц с ограниченными возможностями здоровья применяются адаптивные образовательные технологии в соответствии с условиями, изложенными в раздел «Особенности организации образовательного процесса по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья» основной образовательной программы (использование специальных учебных пособий и дидактических материалов, специальных технических средств обучения коллективного и индивидуального пользования, предоставление услуг ассистента (помощника), оказывающего обучающимся необходимую техническую помощь и т.п.) с учётом индивидуальных особенностей обучающихся.

9 СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ

9.1 Литература

1. Горнец, Н.Н. Организация ЭВМ и систем : учеб. пособие для студ. вузов / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. – 2-е изд., стер. – М. : Академия, 2008. – 315, [1] с. – (Высшее профессиональное образование. Информатика и вычислительная техника). (16 экз.)

2. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 4-е изд. – СПб. [и др.] : Питер, 2003. – 698 с. – (Классика computer science). (5 экз.)

3. Цилькер, Б.Я. Организация ЭВМ и систем: учебник для вузов / Б. Я. Цилькер, С. А. Орлов. – М.; СПб. [и др.]: Питер, 2007. – 668 с. : ил.(10 экз.)

4. Новожилов, О. П. Архитектура ЭВМ и систем в 2 ч. Часть 1 : учебное пособие для вузов / О. П. Новожилов. – Москва : Издательство Юрайт, 2022. – 276 с. – (Высшее образование). – ISBN 978-5-534-07717-9. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/494314 (дата обращения: 10.10.2022).

5. Новожилов, О. П. Архитектура ЭВМ и систем в 2 ч. Часть 2 : учебное пособие для вузов / О. П. Новожилов. – Москва : Издательство Юрайт, 2022. – 246 с. – (Высшее образование). – ISBN 978-5-534-07718-6. – Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/494315 (дата обращения: 10.10.2022).

6. Толстобров, А. П. Архитектура ЭВМ : учебное пособие для вузов / А. П. Толстобров. – 2-е изд., испр. и доп. – Москва : Издательство Юрайт, 2022. – 154 с. – (Высшее образование). – ISBN 978-5-534-12377-7. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/496167 (дата обращения: 10.10.2022).

7. Юров, В. ASSEMBLER / В. Юров. – М.; Харьков; Минск; СПб. : Питер, 2001. – 622 с. (10 экз.)

8. Юров В. ASSEMBLER : Практикум / Юров В. - М.;Харьков;Минск;СПб. : Питер, 2001. - 395 с. (8 экз.)

9.2 Базы данных и информационно-справочные системы

1. Федеральный портал «Российское образование» – Режим доступа: http://www.edu.ru

2. Информационная система «Единое окно доступа к образовательным ресурсам» – Режим доступа: http://www.window.edu.ru

3. Федеральный центр информационно-образовательных ресурсов – Режим доступа: http://fcior.edu.ru

4. Сайт Федеральной службы по интеллектуальной собственности, патентам и товарным знакам (Роспатента). – Режим доступа: http://www.fips.ru/rospatent/index.htm

9.3 Электронно-библиотечные ресурсы

1. ЭБС «Юрайт». – Режим доступа: https://urait.ru
2. Полпред (обзор СМИ). – Режим доступа: https://polpred.com/news

10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА

Для проведения занятий лекционного и семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации используются аудитории, оснащённые учебной мебелью, аудиторной доской, компьютером(рами) с установленным лицензионным специализированным программным обеспечением, коммутатором для выхода в электронно-библиотечную систему и электронную информационно-образовательную среду БГПУ, мультимедийными проекторами, экспозиционными экранами, учебно-наглядными пособиями (мультимедийные презентации).

Самостоятельная работа студентов организуется в аудиториях оснащенных компьютерной техникой с выходом в электронную информационно-образовательную среду вуза, в специализированных лабораториях по дисциплине, а также в залах доступа в локальную сеть БГПУ.

Лицензионное программное обеспечение: операционные системы семейства Windows; офисные программы Microsoft Office, Libreoffice, OpenOffice; программа «Модель программируемой логической матрицы»; программа DosBox; программа Debug; система программирования TurboPascal.

Разработчик: Рокосей В.А., кандидат физико – математических наук, доцент

11 ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ

Утверждение изменений и дополнений в РПД для реализации в 2020/2021 уч. г.

РПД обсуждена и одобрена для реализации в 2020/2021 уч. г. на заседании кафедры информатики и методики преподавания информатики (протокол № 8 от «17» июня 2020 г.). В РПД внесены следующие изменения и дополнения:

№ изменения: 1 № страницы с изменением: на титульном листе	
Исключить:	Включить:
Текст: МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ	Текст: МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Утверждение изменений и дополнений в РПД для реализации в 2021/2022 уч. г.

РПД обсуждена и одобрена для реализации в 2021/2022 уч. г. без изменений на заседании кафедры информатики и методики преподавания информатики (протокол № 7 от 21.04.2021 г.).

Утверждение изменений и дополнений в РПД для реализации в 2022/2023 уч. г.

РПД пересмотрена, обсуждена и одобрена для реализации в 2022/2023 учебном году на заседании кафедры информатики и методики преподавания информатики (протокол №1 от 21 сентября 2022 г.).

В рабочую программу внесены следующие изменения и дополнения:

№ изменения: 1 № страницы с изменением: 35-36	
В Раздел 9 внесены изменения в список литературы, в базы данных и информационно-справочные системы, в электронно-библиотечные ресурсы. Указаны ссылки, обеспечивающие доступ обучающимся к электронным учебным изданиям и электронным образовательным ресурсам с сайта ФГБОУ ВО «БГПУ».	