

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ В.Ф. УТКИНА»

Кафедра «Электронные вычислительные машины»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

по дисциплине

«Машинно-зависимые языки программирования»

Направление подготовки

09.03.01 Информатика и вычислительная техника

Направленность (профиль) подготовки

Вычислительные машины, комплексы, системы и сети

Уровень подготовки

Академический бакалавриат

Квалификация (степень) выпускника — бакалавр

Форма обучения — очная, заочная

Рязань

1 ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов (контрольных заданий, описаний форм и процедур), предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной профессиональной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций, обучающихся целям и требованиям основной профессиональной образовательной программы в ходе проведения текущего контроля и промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности общекультурных, общепрофессиональных и профессиональных компетенций, приобретаемых обучающимся в соответствии с этими требованиями.

Контроль знаний проводится в форме текущего контроля и промежуточной аттестации.

Текущий контроль успеваемости проводится с целью определения степени усвоения учебного материала, своевременного выявления и устранения недостатков в подготовке обучающихся и принятия необходимых мер по совершенствованию методики преподавания учебной дисциплины (модуля), организации работы обучающихся в ходе учебных занятий и оказания им индивидуальной помощи.

К контролю текущей успеваемости относятся проверка знаний, умений и навыков, приобретенных обучающимися в ходе выполнения индивидуальных заданий на практических занятиях и лабораторных работах. При оценивании результатов освоения практических занятий и лабораторных работ применяется шкала оценки «зачтено – не зачтено». Количество лабораторных и практических работ и их тематика определена рабочей программой дисциплины, утвержденной заведующим кафедрой.

Результат выполнения каждого индивидуального задания должен соответствовать всем критериям оценки в соответствии с компетенциями, установленными для заданного раздела дисциплины.

Промежуточный контроль по дисциплине осуществляется проведением экзамена и теоретического зачета.

Форма проведения экзамена – письменный ответ по утвержденным экзаменационным билетам, сформулированным с учетом содержания учебной дисциплины. В экзаменационный билет включается два теоретических вопроса и одна задача. После выполнения письменной работы обучаемого производится ее оценка преподавателем и, при необходимости, проводится теоретическая беседа с обучаемым для уточнения экзаменационной оценки.

2 ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Сформированность каждой компетенции (или ее части) в рамках освоения данной дисциплины оценивается по трехуровневой шкале:

- 1) пороговый уровень является обязательным для всех обучающихся по завершении освоения дисциплины;
- 2) продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;
- 3) эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

Уровень освоения компетенций, формируемых дисциплиной:

Описание критериев и шкалы оценивания тестирования:

| Шкала оценивания | Критерий |
|----------------------------------|---|
| 3 балла (эталонный уровень) | Уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 85 до 100% |
| 2 балла (продвинутый уровень) | Уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 70 до 84% |
| 1 балл (пороговый уровень) | Уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 50 до 69% |
| 0 баллов | Уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 0 до 49% |

Описание критериев и шкалы оценивания теоретического вопроса:

| Шкала оценивания | Критерий |
|----------------------------------|--|
| 3 балла (эталонный уровень) | выставляется студенту, который дал полный ответ на вопрос, показал глубокие систематизированные знания, смог привести примеры, ответил на дополнительные вопросы преподавателя |
| 2 балла (продвинутый уровень) | выставляется студенту, который дал полный ответ на вопрос, но на некоторые дополнительные вопросы преподавателя ответил только с помощью наводящих вопросов |
| 1 балл (пороговый уровень) | выставляется студенту, который дал неполный ответ на вопрос в билете и смог ответить на дополнительные вопросы только с помощью преподавателя |
| 0 баллов | выставляется студенту, который не смог ответить на вопрос |

Описание критериев и шкалы оценивания практического задания:

| Шкала оценивания | Критерий |
|----------------------------------|---|
| 3 балла (эталонный уровень) | Задача решена верно |
| 2 балла (продвинутый уровень) | Задача решена верно, но имеются неточности в логике решения |
| 1 балл (пороговый уровень) | Задача решена верно, с дополнительными наводящими вопросами преподавателя |
| 0 баллов | Задача не решена |

На промежуточную аттестацию (экзамен) выносятся тест, два теоретических вопроса и 2 задачи. Максимально студент может набрать 15 баллов. Итоговый суммарный балл студента, полученный при прохождении промежуточной аттестации, переводится в традиционную форму по системе «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно».

Оценка «отлично» выставляется студенту, который набрал в сумме 15 баллов (выполнил все задания на эталонном уровне). Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «хорошо» выставляется студенту, который набрал в сумме от 10 до 14 баллов при условии выполнения всех заданий на уровне не ниже продвинутого. Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «удовлетворительно» выставляется студенту, который набрал в сумме от 5 до 9 баллов при условии выполнения всех заданий на уровне не ниже порогового.

Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий.

Оценка «неудовлетворительно» выставляется студенту, который набрал в сумме менее 5 баллов или не выполнил всех предусмотренных в течение семестра практических заданий.

3 ПАСПОРТ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ДИСЦИПЛИНЕ

| Контролируемые разделы (темы) дисциплины | Код контролируемой компетенции (или её части) | Вид, метод, форма оценочного мероприятия |
|--|---|--|
| Тема 1. Общие принципы организации ЭВМ на примере ЭВМ семейства IBM PC. | ОПК-3 | экзамен |
| Тема 2. Основные элементы программирования на Ассемблере. | ОПК-3 | экзамен |
| Тема 3. Сложные типы данных в Ассемблере: массивы, строки, структуры и записи. | ПК-6 | экзамен |
| Тема 4. Макросредства в языке Ассемблер. | ПК-6 | экзамен |
| Тема 5. Работа с файлами и директориями в Ассемблере. | ОПК-3, ПК-6 | экзамен |

4 ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ

4.1. Промежуточная аттестация в форме экзамена

| Код компетенции | Результаты освоения ОПОП Содержание компетенций |
|-----------------|--|
| ОПК-3 | способен учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности |

а) типовые тестовые вопросы:

1. Предложения языка ассемблера состоят из следующих компонент:

- +а) метка или имя;
- +б) мнемоника;
- +в) операнды;
- +г) комментарии;
- д) константы;
- е) литералы;

2. Для указания ассемблеру того, что в программе используются числа в двоичной системе исчисления необходимо:

- +а) в конце каждого двоичного числа ставить букву «b»;
- б) в конце каждого двоичного числа ставить обозначение «bit»;
- в) в начале каждого двоичного числа ставить букву «b», а в конце 2;
- г) в начале каждого двоичного числа ставить цифру «2», а в конце букву «b»;
- д) в начале каждого двоичного числа ставить букву «b»;
- е) в конце каждого двоичного числа ставить цифру «2»;
- ж) ничего не ставить, ассемблер сам разберётся, где двоичная запись, а где шестнадцатеричная;

3. Для представления отрицательного числа в компьютере выполняются следующие операции:

- +а) инверсия положительного числа – прибавление 1 к результату инверсии = отрицательное число;
- б) прибавление 1 к положительному числу – инверсия результата = отрицательное число;
- в) побитовое сложение положительного числа с ним же самим – инверсия результата сложения плюс 1 = отрицательное число;
- г) инверсия положительного числа - побитовое сложение инвертированного результата с ним же самим плюс 1 = отрицательное число;

4. Процессор – это:

- +а) кремневая плата или подложка с логическими цепями, состоящими из транзисторов, скрытая в пластмассовом корпусе, снабжённом контактными ножками;
- б) кремневая плата, обеспечивающая механизм страничной организации памяти, которая необходима для любой многозадачной операционной системы;
- в) кремневая плата, хранящая инструкции и данные в виде двоичных сигналов в двоичной системе исчисления;

5. К регистрам общего назначения относят регистры:

- +а) EAX;
- +б) EBX;
- +в) ECX;
- +г) EDX;
- д) EES;
- е) EDS;
- ж) ESS;
- з) ECS;

6. Выберите правильные записи команд:

- а) `mov ah,123h;`
- б) `mov bx,12345h;`
- в) `mov dl,100h;`
- +г) `mov cx,1234h;`
- +д) `mov al,56h;`
- е) `mov es,ds;`
- +ж) `mov dx,0DEF0h;`

7. Сегментные регистры в архитектуре x86_32 имеют:

- +а) 16 разрядов;
- б) 20 разрядов;
- в) 8 разрядов;
- г) 32 разряда;
- д) 64 разряда;

8. Сегментные регистры:

- +а) хранят начальные адреса сегментов программы и обеспечивают возможность обращения к этим сегментам;
- б) используются для хранения данных. В эти регистры может быть записан адрес возврата в основную программу после завершения работы процедуры;
- в) хранят машинные коды команд после трансляции программы;
- г) хранят адрес инструкции, которая должна быть выполнена следующей;

9. Выберите правильные трактовки:

- +а) флаг ZF – признак нуля;
- +б) флаг CF – признак переноса;
- +в) флаг SF – признак знака;
- г) флаг TF – признак полупереноса;

10. КОП – это:

- +а) код операции;
- +б) мнемоническое обозначение соответствующей машинной команды, макрокоманды или директивы транслятора;
- в) часть команды, макрокоманды или директивы ассемблера, обозначающая объекты над которыми производятся командные операции;
- г) последовательность допустимых символов, обозначающих команду;

11. Когда ассемблер встречает в программе команду `jmp $+3` то:

- а) прибавляет к переменной `$` цифру 3;
- б) прибавляет к машинному коду операции цифру 3;
- +в) к текущему смещению прибавляет 3 и переходит к команде, имеющей полученный адрес;
- г) прибавляет к содержимому регистра `AX` цифру 3 и переходит к команде, имеющей полученный адрес;

12. Параметры модели памяти могут быть следующие:

- +а) TINY;
- +б) SMALL;
- +в) MEDIUM;
- +г) FLAT;
- д) LITTLE;

13. Тип данных `dq` резервирует в памяти:

- а) 2 байта;
- б) 4 байта;
- +в) 8 байтов;
- г) 10 байтов;

14. Атрибут выравнивания сегмента может принимать следующие значения:

- +а) BYTE;
- +б) WORD;
- +в) DWORD;
- г) PRIVATE;
- д) PUBLIC;
- е) COMMON;

15. Параметры модели памяти могут быть следующие:

- +а) TINY;
- +б) SMALL;
- +в) MEDIUM;
- +г) FLAT;
- д) LITTLE;

16. Прямая адресация делится на:

- +а) относительную прямую адресацию;

- +б) абсолютную прямую адресацию;
- в) косвенную прямую адресацию;
- г) базовую прямую адресацию;

17. К командам пересылки данных относятся:

- +а) mov;
- +б) xchg;
- +в) cmovcc;
- +г) bswap;
- д) shr;
- е) shl;
- ж) in, out;

18. Выберите неправильно записанные команды:

- +а) mov fld,fls;
- +б) mov ds,@data;
- +в) mov es,ds;
- +г) mov ax,bl;
- д) mov ax,dx;

19. Допустимыми операндами-источниками для команды IN являются:

- +а) регистр DX;
- +б) значение-константа меньше 255;
- в) регистр AX;
- г) регистр BX;

20. Команда in AX,DX :

- +а) загружает в AX слово из порта с номером из DX;
- б) загружает в AX байт из порта с номером из DX;
- в) загружает байт в порт, адресуемый регистром DX;
- г) загружает слово в порт, адресуемый регистром DX;
- д) загружает в AX содержимое порта DX;

б) типовые практические задания:

Задание 1

Написать программу пересылки данных из одного сегмента памяти в другой.

Критерии выполнения заданий 3

Задание считается выполненным, если обучающийся написал правильный код на ассемблере.

Задание 2

Написать программу транспонирования массива данных с помощью косвенной адресации.

Критерии выполнения заданий 3

Задание считается выполненным, если обучающийся написал правильный код на ассемблере.

| Код компетенции | Результаты освоения ОПОП Содержание компетенций |
|------------------------|---|
| ПК-6 | способен обосновывать разработку функциональной структуры и выбор принципов организации технического, программного и информационного обеспечения проектирования специальных организационно-технических систем |

а) типовые тестовые вопросы:

1. Основное предназначение регистров MMX – это:

- а) работа с одной целочисленной переменной, объём которой 8 байт;
- +б) работа с несколькими целочисленными переменными одинаковой длины, которые занимают в сумме 8 байт;
- в) работа с несколькими вещественными переменными одинаковой длины, которые занимают в сумме 8 байт;
- г) работа с одной вещественной переменной, объём которой 8 байт;
- д) нет таких регистров в архитектуре процессоров x86;
- е) работа с одной вещественной переменной, объём которой 10 байт;

2. Косвенный переход на метку в другом сегменте кода:

- +а) имеет модификатор dword ptr;
- б) имеет модификатор word ptr;
- в) имеет модификатор far ptr;
- г) имеет модификатор near ptr;
- +д) изменяет содержимое регистров cs и ip;
- е) изменяет содержимое только регистра cs;
- ж) изменяет содержимое только регистра ip;
- +з) предполагает использование адреса перехода, записанного в какой-либо из регистров;
- и) предполагает использование адреса перехода длиной в 4 байта;
- к) предполагает использование адреса перехода длиной в 3 байта;

3. Если при написании программы предполагается использование большого объёма данных, но малого объёма кода, то лучше всего использовать модель памяти:

- а) Tiny;
- б) Small;
- в) Medium;
- +г) Compact;
- д) Large;

4. Двумерный массив array в программе описан следующим образом `array dw 10 dup (?)`, где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 2x5. При этом при обращении к элементу массива используются регистры esi – столбцы в матрице и ebx – строки в матрице. Чтобы записать в регистр eax нулевой элемент третьей строки матрицы необходимо выполнить команды (см. рис. 1.1):

| | |
|--|---|
| A) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,12</code> <code>mov eax,array[ebx][esi]</code> | Б) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,3</code> <code>mov ax,array[ebx][esi]</code> |
| В) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,3</code> <code>mov ax,array[esi][ebx]</code> | Г) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,12</code> <code>mov ax,array[ebx][esi]</code> |
| Д) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,3</code> <code>mov ax,array[ebx+esi]</code> | Е) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,12</code> <code>mov ax,array[ebx,esi]</code> |
| Ж) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,6</code> <code>add ebx,6</code> <code>mov ax,array[ebx][esi]</code> | З) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,3</code> <code>mov ax,array[ebx][esi]</code> |
| И) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>add ebx,3</code> <code>mov ax,array[esi,ebx]</code> | |

Рис. 1.1.

- а) а);
- б) б);
- в) в);
- +г) г);
- д) д);
- е) е);
- +ж) ж);
- з) з);
- и) и);

5. Двумерный массив `array` в программе описан следующим образом `array dw 15 dup (?)`, где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 3x5. При этом при обращении к элементу массива используются регистры `esi` – столбцы в матрице и `ebx` – строки в матрице. Чтобы записать в регистр `eax` второй элемент третьей строки матрицы необходимо выполнить команды (см. рис. 1.2):

| | |
|---|---|
| А) mov ebx,0 mov esi,0 xor eax,eax mov esi,4 add ebx,18 mov ax,array[ebx][esi] + | Б) mov ebx,0 mov esi,0 xor eax,eax mov esi,2 add ebx,3 mov ax,array[ebx][esi] |
| В) mov ebx,0 xor eax,eax mov esi,0 add ebx,22 mov ax,array[ebx][esi] + | Г) mov ebx,0 mov esi,0 xor eax,eax mov esi,18 add ebx,4 mov eax,array[ebx][esi] |
| Д) mov ebx,0 mov esi,0 xor eax,eax mov esi,3 add ebx,2 mov ax,array[ebx+esi] | Е) mov ebx,0 mov esi,0 xor eax,eax mov esi,18 add ebx,4 mov ax,array[ebx][esi] + |
| Ж) mov ebx,0 mov esi,0 xor eax,eax mov esi,4 add ebx,18 mov ax,array[ebx,esi] | З) mov ebx,0 mov esi,0 xor eax,eax mov esi,4 add ebx,18 mov ax,array[esi,ebx] |

Рис. 1.2.

+а) а);
б) б);
+в) в);
г) г);
д) д);
+е) е);
ж) ж);
з) з);

6. Двумерный массив array в программе описан следующим образом array dw 20 dup (?), где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 4x5. При этом при обращении к элементу массива используются регистры esi – столбцы в матрице и ebx – строки в матрице. Чтобы записать в регистр eax третий элемент третьей строки матрицы необходимо выполнить команды (см. рис. 1.3):

| | |
|---|--|
| А) mov ebx,0 mov esi,0 mov esi,6 add ebx,24 mov eax,array[ebx][esi] | Б) mov ebx,0 mov esi,0 mov esi,24 add ebx,6 mov ax,array[ebx][esi] |
| В) mov ebx,0 mov esi,0 mov esi,0 add ebx,30 mov ax,array[ebx][esi] | Г) mov ebx,0 mov esi,0 mov esi,0 add ebx,30 mov ax,array[ebx+esi] |
| Д) mov ebx,0 mov esi,0 mov esi,6 add ebx,24 mov ax,array[ebx,esi] | Е) mov ebx,0 mov esi,0 mov esi,3 add ebx,3 mov ax,array[ebx][esi] |
| Ж) mov ebx,0 mov esi,0 mov esi,3 add ebx,3 mov ax,array[esi,ebx] | З) mov ebx,0 mov esi,0 mov esi,6 add ebx,24 mov ax,array[ebx][esi] |

Рис. 1.3.

- а) а);
- +б) б);
- +в) в);
- г) г);
- д) д);
- е) е);
- ж) ж);
- +з) з);

7. Двумерный массив array в программе описан следующим образом `array dw 16 dup (?)`, где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 4x4. При этом при обращении к элементу массива используются регистры `esi` – столбцы в матрице и `ebx` – строки в матрице. Чтобы записать в регистр `eax` второй элемент третьей строки матрицы необходимо выполнить команды (см. рис. 1.4):

| | |
|--|--|
| А) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,4</code> <code>add ebx,24</code> <code>mov eax,array[ebx][esi]</code> | Б) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,24</code> <code>add ebx,4</code> <code>mov ax,array[ebx][esi]</code> |
| В) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,0</code> <code>add ebx,28</code> <code>mov ax,array[ebx][esi]</code> | Г) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,0</code> <code>add ebx,28</code> <code>mov eax,array[ebx][esi]</code> |
| Д) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,0</code> <code>add ebx,28</code> <code>mov ax,array[ebx+esi]</code> | Е) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,4</code> <code>add ebx,24</code> <code>mov ax,array[ebx,esi]</code> |
| Ж) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,3</code> <code>add ebx,2</code> <code>mov ax,array[ebx][esi]</code> | З) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,4</code> <code>add ebx,24</code> <code>mov ax,array[ebx][esi]</code> |

Рис. 1.4.

- а) а);
- +б) б);
- +в) в);
- г) г);
- д) д);
- е) е);
- ж) ж);
- +з) з);

8. Двумерный массив array в программе описан следующим образом `array dw 6 dup (?)`, где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 2x3. При этом при обращении к элементу массива используются регистры `esi` – столбцы в матрице и `ebx` – строки в матрице. Чтобы записать в регистр `eax` первый элемент второй строки матрицы необходимо выполнить команды (см. рис. 1.5):

| | |
|--|---|
| A) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,10</code> <code>add ebx,0</code> <code>mov eax,array[ebx][esi]</code> | Б) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,1</code> <code>add ebx,2</code> <code>mov ax,array[ebx][esi]</code> |
| B) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,10</code> <code>add ebx,0</code> <code>mov ax,array[ebx][esi]</code> | Г) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,2</code> <code>add ebx,1</code> <code>mov eax,array[ebx][esi]</code> |
| Д) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,1</code> <code>add ebx,2</code> <code>mov ax,array[ebx+esi]</code> | Е) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,8</code> <code>add ebx,2</code> <code>mov ax,array[ebx,esi]</code> |
| Ж) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,8</code> <code>add ebx,2</code> <code>mov ax,array[ebx][esi]</code> | З) <code>mov ebx,0</code> <code>mov esi,0</code> <code>xor eax,eax</code> <code>mov esi,0</code> <code>add ebx,2</code> <code>mov ax,array[ebx][esi]</code> |

Рис. 1.5.

а) а);
б) б);
+в) в); +
г) г);
д) д);
е) е);
+ж) ж); +
з) з);

9. Двумерный массив `array` в программе описан следующим образом `array dw 18 dup (?)`, где элементы массива имеют размерность в 2 байта. Двумерный массив имеет размерность 6x3. При этом при обращении к элементу массива используются регистры `esi` – столбцы в матрице и `ebx` – строки в матрице. Чтобы записать в регистр `eax` четвёртый элемент второй строки матрицы необходимо выполнить команды (см. рис. 1.6):

| | |
|--|--|
| А) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,32 add ebx,0 mov eax,array[ebx][esi]</code> | Б) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,0 add ebx,32 mov eax,array[ebx][esi]</code> |
| В) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,4 add ebx,2 mov ax,array[ebx][esi]</code> | Г) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,0 add ebx,32 mov ax,array[ebx][esi]</code> |
| Д) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,2 add ebx,4 mov eax,array[ebx][esi]</code> | Е) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,4 add ebx,2 mov ax,array[ebx+esi]</code> |
| Ж) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,8 add ebx,24 mov ax,array[ebx][esi]</code> | З) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,32 add ebx,0 mov ax,array[ebx,esi]</code> |
| И) <code>mov ebx,0 mov esi,0 xor eax,eax mov esi,0 add ebx,8 mov ax,array[ebx][esi]</code> | |

Рис. 1.6.

- а) а);
- б) б);
- в) в);
- +г) г);
- д) д);
- е) е);
- +ж) ж);
- з) з);
- и) и);

10) TASM поддерживает следующие сложные типы данных:

- +а) массивы;
- +б) структуры;
- +в) объединения;
- +г) записи;
- д) объекты;
- е) директивы;

11) Если описать одномерный массив следующим образом: `mas dd 5 dup (0)`. Какой объём памяти в данном случае будет выделен под массив?

- а) 5 байт;
- б) 4 байта;
- в) 10 байт;
- г) 9 байт;
- +д) 20 байт;
- е) 19 байт;

12) Задать двумерный массив в ассемблере можно следующим образом:

- а) mas array [0..m,0..n];
- б) mas array [1..m,1..n];
- в) mas db [ebx][esi];
- +г) mas db 23,4,5,67,5,6,7,99,67,8,9,23,87,9,0,8;
- +д) mas dw 5 dup (?);

13) Адрес элемента (i,j) в двумерном массиве можно определить следующим образом:

- +а) (база + количество_элементов_в_строке * размер_элемента * i+j*размер_элемента);
- б) (база+смещение + количество_элементов_в_строке * размер_элемента * i+j**размер_элемента);
- в) (смещение + количество_элементов_в_строке * размер_элемента * i+j);
- г) (база + количество_элементов_в_строке+смещение * размер_элемента * i+j*размер_элемента);
- д) (смещение + количество_элементов_в_строке +база* размер_элемента * i+j);

14) Для использования структур в программе необходимо обязательно выполнить:

- +а) задать шаблон структуры;
- +б) определить экземпляр структуры; +
- +в) организовать обращение к элементам структуры; +
- г) удалить экземпляр структуры после использования его в программе;
- д) удалить шаблон структуры;

15) С помощью оператора type можно:

- +а) определить объём памяти, выделяемый под один элемент структуры;
- б) определить объём памяти, выделяемый под массив структур;
- +в) организовать индексацию в массиве структур;
- г) обратиться к элементу структуры в виде (адресное_выражение) type (имя_поля_структуры);

16) Для того, чтобы объявить транслятору о том, что в программе имеется процедура, которая используется в другом модуле необходимо использовать директивы:

- +а) extrn и public;
- б) enter и leave;
- в) far и near;

17) Если необходимо собрать два модуля, например modul1.asm и modul2.asm в один исполняемый модуль, необходимо:

- +а) выполнить трансляцию модуля modul1.asm и получить объектный модуль modul1.obj;
- +б) выполнить трансляцию модуля modul2.asm и получить объектный модуль modul2.obj;
- +в) скомпоновать программу утилитой TLINK командной строкой вида tlink /v modul1.obj+ modul2.obj;
- г) скомпоновать программу утилитой TLINK командной строкой вида tlink /C modul1.obj+ modul2.obj;
- д) скомпоновать программу утилитой TLINK командной строкой вида tlink /C modul1.obj and modul2.obj;

18) Пролог в процедуре необходим для:

- +а) инициализации регистра bp;
- б) инициализации регистра sp;
- +в) для доступа к переданным в процедуру аргументам через стек;
- г) для того, чтобы правильно записать эпилог в конце процедуры;

19) Если в процедуру типа `near` были переданы через стек аргументы, то для доступа в процедуре к последнему переданному аргументу необходимо:

- +а) сместиться от содержимого `bp` на 4 байта; +
- б) сместиться от содержимого `bp` на 6 байта;
- в) сместиться от содержимого `bp` на 8 байта;
- г) сместиться от содержимого `bp` на 2 байта;

20) Команда `ret n` необходима для:

- +а) очистки стека от аргументов, которые передавались в процедуру;
- б) обращения к n-ому аргументу, который передавался через стек;
- в) передачи аргументов в процедуру по их адресу;

б) типовые практические задания:

Задание 1

Требуется написать код прерывания выполняющий чтение 20 байт из порта с адресом 80.

Критерии выполнения заданий 3

Задание считается выполненным, если обучающийся представил корректный код на языке ассемблера.

Задание 2

Требуется написать код прерывания выполняющий инверсию значений двумерного массива типа `byte`. Код должен быть оформлен как процедура с тремя параметрами (начальный адрес, ширина и высота массива данных).

Критерии выполнения заданий 3

Задание считается выполненным, если обучающийся представил корректный код на языке ассемблера.