

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

**МАШИННО-ЗАВИСИМЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ.
ЛАБОРАТОРНАЯ РАБОТА №1**

ОТЧЁТ

студентки 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета компьютерных наук и информационных технологий
Потапкиной Маргариты Андреевны

Проверено:

старший преподаватель

Е. М. Черноусова

СОДЕРЖАНИЕ

1	Текст задания	3
2	Тексты трёх программ на языке ассемблера с комментариями	4
2.1	Первая программа	4
2.2	Вторая программа	4
2.3	Третья программа	4
3	Скриншоты запуска трёх программ	5
4	Тексты 2-х командных файлов (для exe-программ и для com-программы).	7
4.1	Командный файл для exe-программы	7
4.2	Командный файл для com-программы	7
5	Таблицы трассировки программ	8
6	Ответы на контрольные вопросы	9

1 Текст задания

Измените программы из примеров 1, 2 и 3 так, чтобы они выводили на экран ваши фамилию, имя и номер группы. Используя командные файлы (с расширением bat), подготовьте к выполнению и запустите 3 программы. Убедитесь, что они выводят на экран нужный текст и успешно завершаются. Заполните таблицы трассировки для 3-х программ.

2 Тексты трёх программ на языке ассемблера с комментариями

2.1 Первая программа

```
stack segment stack 'stack'      ;Начало сегмента стека
db 256 dup (?)                  ;Резервируем 256 байт для стека
stack ends                      ;Конец сегмента стека
data segment 'data'             ;Начало сегмента данных
Info db 'Potapkina Margarita, 251$' ;Строка для вывода
data ends                      ;Конец сегмента данных
code segment 'code'             ;Начало сегмента кода
assume CS:code,DS:data,SS:stack ;Сегментный регистр CS будет указывать
    ↪ на сегмент команд,
                                ;регистр DS - на сегмент данных, SS -
                                ↪ на стек
start:                          ;Точка входа в программу start
;Обязательная инициализация регистра DS в начале программы
mov AX,data                    ;Адрес сегмента данных сначала
    ↪ загрузим в AX,
mov DS,AX                      ;а затем перенесем из AX в DS
mov AH,09h                    ;Функция DOS 9h вывода на экран
mov DX,offset Info            ;Адрес начала строки с фамилией
    ↪ записывается в регистр DX
int 21h                        ;Вызов функции DOS
mov AX,4C00h                  ;Функция 4Ch завершения программы с
    ↪ кодом возврата 0
int 21h                        ;Вызов функции DOS
code ends                     ;Конец сегмента кода
end start                     ;Конец текста программы с точкой входа
```

2.2 Вторая программа

```
.model small                    ;Модель памяти SMALL использует сегменты
                                ;размером не более 64Кб
.stack 100h                    ;Сегмент стека размером 100h (256 байт)
.data                          ;Начало сегмента данных
Info db 'Potapkina Margarita, 251$' ;строка для вывода
.code                          ;Начало сегмента кода
start:                         ;Точка входа в программу start
                                ;Предопределенная метка @data обозначает
                                ;адрес сегмента данных в момент запуска
                                ↪ программы,
mov AX, @data                  ;который сначала загрузим в AX,
mov DS,AX                     ;а затем перенесем из AX в DS
```

```

mov AH,09h
mov DX,offset Info
int 21h
mov AX,4C00h
int 21h
end start

```

2.3 Третья программа

```

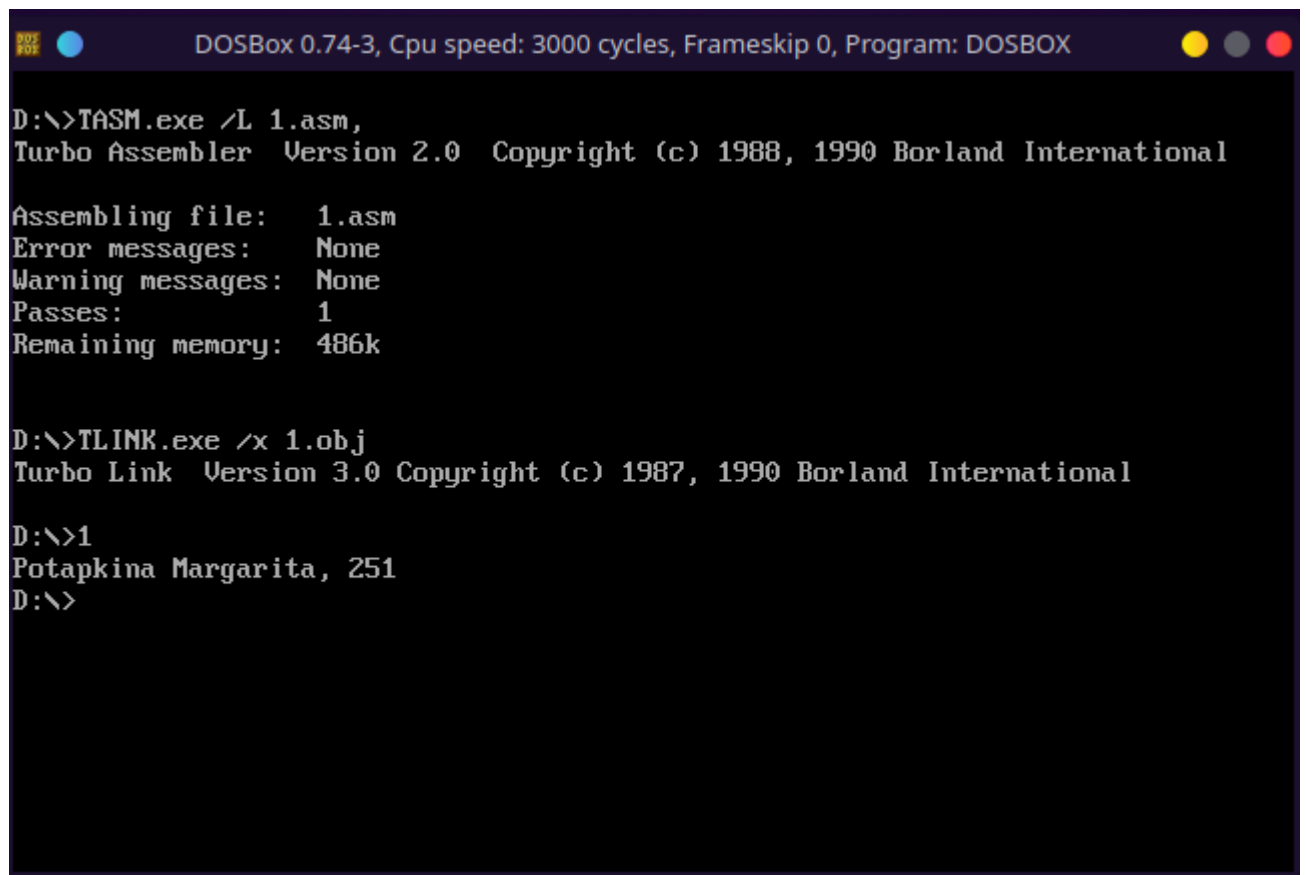
.model tiny                ;Модель памяти TINY, в которой код, данные и стек
                           ;размещаются в одном и том же сегменте размером до
                           ;→ 64Кб

.code                      ;Начало сегмента кода
org 100h                   ;Устанавливает значение программного счетчика в 100h
                           ;Начало необходимое для COM-программы,
                           ;которая загружается в память с адреса PSP:100h

start:
mov AH,09h
mov DX,offset Info
int 21h
mov AX,4C00h
int 21h
;===== Data =====
Info db 'Potapkina Margarita, 251$' ;Строка для вывода
end start

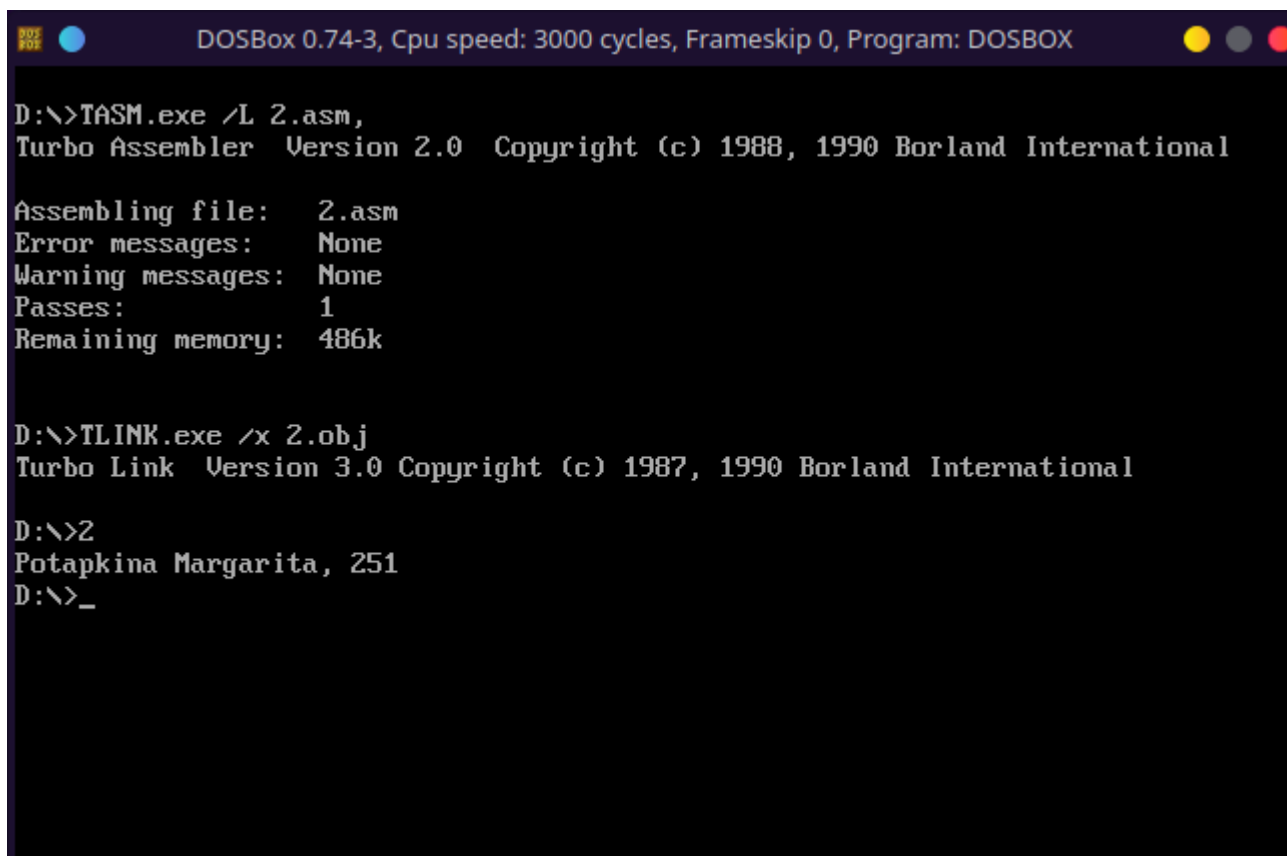
```

3 Скриншоты запуска трёх программ

A screenshot of a DOSBox window. The title bar at the top reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The main window area has a black background with white text. The text shows the execution of two programs: first, "D:\>TASM.exe /L 1.asm," followed by "Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International". Then, it shows assembly statistics: "Assembling file: 1.asm", "Error messages: None", "Warning messages: None", "Passes: 1", and "Remaining memory: 486k". Next, it shows the execution of "D:\>TLINK.exe /x 1.obj," followed by "Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International". Finally, it shows the execution of a batch file "D:\>1" which outputs "Potapkina Margarita, 251" and returns to the prompt "D:\>".

```
D:\>TASM.exe /L 1.asm,  
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International  
  
Assembling file: 1.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 486k  
  
D:\>TLINK.exe /x 1.obj  
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International  
  
D:\>1  
Potapkina Margarita, 251  
D:\>
```

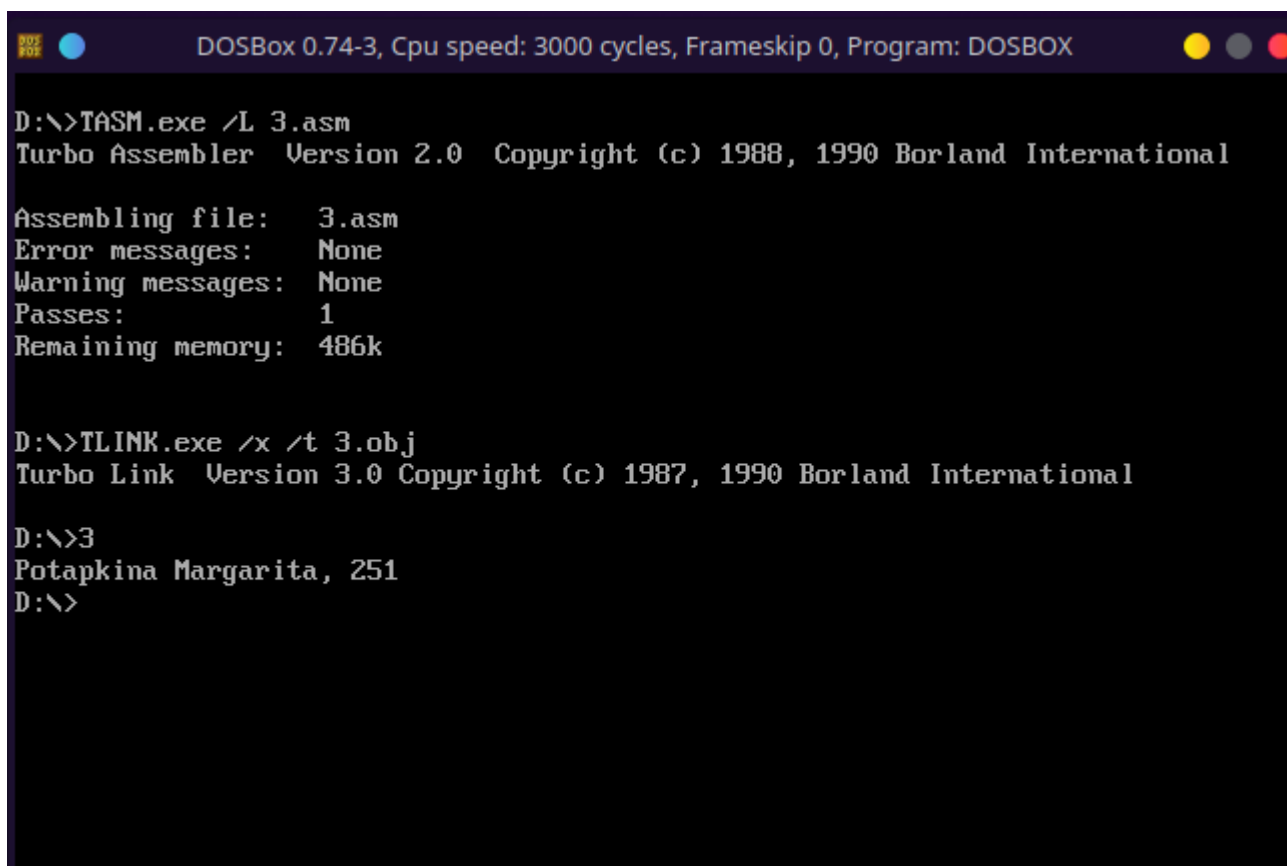
Рисунок 3.1 – Скриншот запуска первой программы



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
D:\>TASM.exe /L 2.asm,  
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International  
  
Assembling file: 2.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 486k  
  
D:\>TLINK.exe /x 2.obj  
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International  
  
D:\>2  
Potapkina Margarita, 251  
D:\>_
```

Рисунок 3.2 – Скриншот запуска второй программы



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
D:\>TASM.exe /L 3.asm  
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International  
  
Assembling file: 3.asm  
Error messages: None  
Warning messages: None  
Passes: 1  
Remaining memory: 486k  
  
D:\>TLINK.exe /x /t 3.obj  
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International  
  
D:\>3  
Potapkina Margarita, 251  
D:\>
```

Рисунок 3.3 – Скриншот запуска третьей программы

4 Тексты 2-х командных файлов (для exe-программ и для com-программы).

4.1 Командный файл для exe-программы

cls

TASM.exe /L %1.asm,

TLINK.exe /x %1.obj

%1

4.2 Командный файл для com-программы

cls

TASM.exe /L %1.asm

TLINK.exe /x /t %1.obj

%1

5 Таблицы трассировки программ

При составлении таблиц трассировки использовался Turbo Debugger.

Шаг	Машинный код	Команда	Регистры									Флаги CZSOPAIID
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
1	B8BD48	MOV AX,48BD	48BD	F560	0192	F664	0100	489D	48AD	48BF	0003	00000010
2	8ED8	MOV DS,AX	48BD	F560	0192	F664	0100	48BD	48AD	48BF	0005	00000010
3	B409	MOV AH,09	09BD	F560	0192	F664	0100	48BD	48AD	48BF	0007	00000010
4	BA0000	MOV DX,0000	09BD	F560	0192	0000	0100	48BD	48AD	48BF	000A	00000010
5	CD21	INT 21	09BD	F560	0192	0000	0100	48BD	48AD	48BF	000C	00000010
6	B8004C	MOV AX,4C00	4C00	F560	0192	0000	0100	48BD	48AD	48BF	000F	00000010
7	CD21	INT21	0192	F560	0192	F664	0106	2110	0192	0000	0000	10100011

Шаг	Машинный код	Команда	Регистры									Флаги CZSOPAIID
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
			0192	000B	F715	098D	0100	489D	48B1	48AD	0000	00000010
1	B8AF48	MOV AX,48AF	48AF	000B	F715	098D	0100	489D	48B1	48AD	0003	00000010
2	8ED8	MOV DS,AX	48AF	000B	F715	098D	0100	48AF	48B1	48AD	0005	00000010
3	B409	MOV AH,09	09AF	000B	F715	098D	0100	48AF	48B1	48AD	0007	00000010
4	BA0000	MOV DX,0000	09AF	000B	F715	0000	0100	48AF	48B1	48AD	000A	00000010
5	CD21	INT21	09AF	000B	F715	0000	0100	48AF	48B1	48AD	000C	00000010
6	B8004C	MOV AX,4C00	4C00	000B	F715	0000	0100	48AF	48B1	48AD	000F	00000010
7	CD21	INT21	0192	F560	0192	F664	0106	2110	0192	0000	0000	10100011

Шаг	Машинный код	Команда	Регистры									Флаги CZSOPAIID
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
			0192	F560	0192	F664	FFFE	489D	489D	489D	0100	00000010
1	B409	MOV AH,09	0992	F560	0192	F664	FFFE	489D	489D	489D	0102	00000010
2	BA0C01	MOV DX,010C	0992	F560	0192	010C	FFFE	489D	489D	489D	0105	00000010
3	CD21	INT21	0992	F560	0192	010C	FFFE	489D	489D	489D	0107	00000010
4	B8004C	MOV AX,4C00	4C00	F560	0192	010C	FFFE	489D	489D	489D	010A	00000010
5	CD21	INT21	0192	F560	0192	F664	0106	2110	0192	0000	0000	10100011

6 Ответы на контрольные вопросы

1. Что такое сегментный (базовый) адрес?

Сегментный (базовый) адрес — 16-битовое значение, содержащееся в сегментном регистре (CS, DS, SS, ES). При расширении базового адреса 4 нулевыми битами в младших разрядах адрес становится 20-битовым (полный адрес сегмента). Чтобы получить физический (абсолютный) адрес ячейки памяти, нужно к полному адресу прибавить смещение (исполнительный адрес).

2. Сделайте листинг для первой программы (файл с расширением lst), выпишите из него размеры сегментов. Из таблицы трассировки к этой программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации адресом сегмента данных). В каком порядке расположились сегменты программы в памяти? Расширя базовый адрес сегмента до физического адреса, прибавляя размер этого сегмента и округляя до кратного 16 значения, мы можем получить физический адрес следующего за ним сегмента. Сделайте это для первых 2-х сегментов. (Если данные не совпали, значит, неверно заполнена таблица трассировки.)

Данные из листинга:

CODE 0011

DATA 0019

STAK 0100

Данные из таблицы трассировки:

CS 48BF

DS 48BD

SS 48AD

Сегменты программы расположились в памяти в порядке: stack, data, code.

STACK: 48AD — базовый адрес; 48AD0 — физический адрес; $48AD0 + 0100 = 48BD0$ — физический адрес сегмента data, совпадает с полученным по таблице

DATA: 48BD — базовый адрес; 48BD0 — физический адрес; $48BD0 + 0019 = 48BE9$, после округления — 48BF0, совпадает с физическим адресом сегмента code из таблицы

3. Почему перед началом выполнения первой программы содержимое регистра DS в точности на 10h меньше содержимого регистра SS? (Сравниваются данные из первой строки таблицы трассировки)

Действительно, $DS = 489D$, $SS = 48AD$, $48AD - 489D = 10h$. До того, как в регистр DS была записана строка, он указывал на PSP (префикс программного сегмента), в соответствии с его размером и был посчитан адрес сегмента SS .

4. Из таблицы трассировки к первой программе выпишите машинные коды команд `mov AX,data` и `mov AH,09h`. Сколько места в памяти в байтах они занимают? Почему у них разный размер?

`MOV AX,DATA: B8BD48` — 3 байта

`MOV AH,09h: B409` — 2 байта

`MOV = B`, $AX = 8$, $AH = 4$, т.е. название команды и первый операнд на размер не влияют, в обоих случаях они занимают по 1 байту. Значит, на размер влияет только второй операнд: во втором случае это однобайтовое число, а в первом — двухбайтовый адрес строки в памяти. Поэтому первая команда занимает больше места.

5. Из таблицы трассировки ко второй программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации). При использовании модели `small` сегмент кода располагается в памяти первым. Убедитесь в этом. (Если это не так, значит, вы неверно заполнили таблицу трассировки.)

`CS 48AD`

`DS 48AF`

`SS 48B1`

Действительно, адрес сегмента кода наименьший, а значит, он находится в памяти первым.

6. Сравните содержимое регистра SP в таблицах трассировки для программах 2 и 3. Объясните, почему получены эти значения.

В программе 2 $SP = 0100$, а в программе 3 $SP = FFFE$. SP указывает на ячейку, расположенную под дном стека. Адрес SP в программе 2 может быть меньше, поскольку в модели `small` сегмент стека размещается первым, тогда как в модели `tiny` и код, и данные, и стек размещаются в одном сегменте, и стек мог быть размещён в конце, причём на него было выделено мало памяти.

7. Какие операторы называют директивами ассемблера? Приведите примеры директив.

В то время как операторы — это инструкции, управляющие работой

процессора, директивы указывают программе ассемблеру, каким образом следует объединять инструкции для создания модуля, который и станет работающей программой, т.е. управляют процессом ассемблирования. Директива может быть помечена символическим именем и содержать поле комментария. Примеры директив:

- `end` <метка, обозначающая точку входа в программу> — метка завершения модуля;
- `data/code/stack segment <имя сегмента>, <имя сегмента> ends` — директивы сегментации;
- `assume <сегментный регистр>:<имя сегмента>` — устанавливает соответствие между сегментными регистрами и программными сегментами;
- `.model` — задаёт модель памяти.

8. Зачем в последнем предложении `end` указывают метку, помечающую первую команду программы?

Метка первой команды программы называется точкой входа в программу и указывает на то, откуда начинать процесс ассемблирования. Директива `end`, в свою очередь, указывает на конец этого программного модуля, т.е. то, что этот блок кода закончился.

9. Как числа размером в слово хранятся в памяти и как они заносятся в двухбайтовые регистры?

В 16-разрядном машинном слове (Word) биты нумеруются от 0 до 15 справа налево, при этом байт с меньшим адресом считается младшим. Таким образом, в памяти число хранится в перевёрнутом виде: младшие 8 разрядов размещаются в первом байте, а старшие 8 — во втором. Напротив, в регистры число записывается в прямом порядке, например, при записи его в регистр `AX` старшие 8 битов запишутся в `AH` (старший регистр), а младшие 8 битов — в `AL` (младший регистр).

10. Как инициализируются в программе выводимые на экран текстовые строки? Строки для вывода помещаются в сегмент данных. Синтаксис инициализации строки выглядит следующим образом.

```
Message db 'Message$'
```

Здесь `db` — директива `define byte` (выделить память под данные), `Message` — имя строки, которое затем будет использоваться в основной программе при

её выводе, а 'Message\$' — сама строка, которая обязательно помещается в кавычки и заканчивается непечатаемым символом \$.

11. Что нужно сделать, чтобы обратиться к DOS для вывода строки на экран? Как DOS определит, где строка закончилась?

Для вывода строки на экран необходимо вызвать стандартную процедуру DOS, называемую прерыванием. Прерывание `int 21h` называется функцией DOS и определяется двумя параметрами: в регистре `AX` должно быть число 9, а в регистре `DX` — адрес строки символов. Символ \$, которым эта строка и должна заканчиваться, и показывает DOS, где конец строки.

12. Программы, которые должны исполняться как `.EXE` и `.COM`, имеют существенные различия по:

- размеру: `.COM` программы занимают меньше места на диске, т.к. `EXE` заголовок и сегмент стека отсутствуют в загрузочном модуле.
- сегментной структуре: `.EXE` программы содержат несколько программных сегментов (кода, данных и стека). Напротив, `.COM` программы содержат единственный сегмент (или, по крайней мере, не содержат явных ссылок на другие сегменты).
- механизму инициализации: `EXE`-файл загружается, начиная с адреса `PSP:0100h`. С помощью `EXE` заголовка в начале файла загрузчик выполняет настройку ссылок на сегменты в загруженном модуле, чтобы учесть тот факт, что программа была загружена в произвольно выбранный сегмент. После настройки ссылок управление передается загрузочному модулю к адресу `CS:IP`, извлеченному из заголовка `EXE`. Напротив, образ `COM`-файла считывается с диска и помещается в память, начиная с `PSP:0100h`, в связи с этим `COM`-программа должна содержать в начале сегмента кода директиву, позволяющую осуществить такую загрузку (`100h`).