**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Организация систем и ЭВМ»**

Тема: Освоение трансляции, выполнения и отладки программ на языке Ассемблера процессора Intel X86

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Еникеев А.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

## Цель работы

Ознакомиться с тем, как происходит трансляция, отладка и выполнение программ на языке Ассемблера.

## Задание

Часть 1.

1. Подготовить среду для запуска исполняемых файлов DOS. Загрузить файл hello1.asm в каталог \MASM.
2. Просмотреть программу в режиме редактирования, разобраться в структуре и реализации каждого сегмента программы. Непонятные фрагменты прояснить у преподавателя. Строку-приветствие преобразовать в соответствии со своими личными данными.
3. Протранслировать программу с помощью строки

> masm имя\_файла.asm

c созданием объектного файла имя\_файла.obj и файла диагностических сообщений (файла листинга) имя\_файла.lst. Объяснить и исправить синтаксические ошибки, если они будут обнаружены транслятором и представлены в файле листинга. Повторить трансляцию программы до получения объектного файла (модуля).

1. Скомпоновать загрузочный модуль (имя\_файла.exe) с помощью строки > link имя\_файла.obj

c созданием загрузочного модуля (имя\_файла.exe) и файла карты памяти (имя\_файла.map). По карте памяти оценить размещение и длину сегментов программы.

1. Выполнить программу в автоматическом режиме путём набора строки > имя\_файла.exe

и убедиться в её работоспособности (результат выполнения просмотреть в режиме отображения экрана пользователя, получаемого набором клавиш ctrl^O);

1. Выполнить программу hello1 в пошаговом режиме под управлением отладчика:

> afd имя\_файла.exe

с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды. Записать содержимое всех регистров процессора, включая сегментные, перед выполнением 1-ой команды. Дальнейшие результаты прогона программы под управлением отладчика должны быть представлены в виде, показанном на примере одной команды в табл.1.

Табл. 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| до вып. команды | после вып. команды |
| 0003 | mov ds,ax | 8E D8 | (ax)= 2D87  (ds)= 2D75  (ip)= 0003 | (ax)= 2D87  \*(ds)= 2D87  \*(ip)= 0005 |

Часть 2.

1. Просмотреть программу hello2.asm в режиме редактирования, разобраться в структуре и реализации каждого сегмента программы. Обратить внимание на использование полного описания сегментов и на использование процедуры для вывода строки. Непонятные фрагменты прояснить у преподавателя. Строки-приветствия преобразовать в соответствии со своими пожеланиями и личными данными.
2. Протранслировать программу hello2.asm с помощью транслятора masm c созданием объектного файла и файла диагностических сообщений (файла листинга). Объяснить и исправить синтаксические ошибки, если они будут обнаружены транслятором. Повторить трансляцию программы до получения объектного модуля.
3. Скомпоновать загрузочный модуль hello2.exe с помощью компоновщика Link.
4. Выполнить программу в автоматическом режиме и убедиться в её работоспособности.
5. Выполнить программу hello2 в пошаговом режиме под управлением отладчика afd с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команд. Аналогично пункту 6 части 1.

## Основные теоретические положения

Всего в процессоре Intel 8086 имеется 14 16-разрядных регистров.

**8 регистров общего назначения** (AX, BX, CX, DX):

* Регистр AX является основным сумматором и применяется для всех операций ввода-вывода, некоторых операций над строками и некоторых арифметических операций. Например, команды умножения, деления и сдвига предполагают использование регистра AX. Некоторые команды генерируют более эффективный код, если они имеют ссылки на регистр AX.
* Регистр BX является базовым регистром. Это единственный регистр общего назначения, который может использоваться в качестве "индекса" для расширенной адресации. Другое общее применение его - вычисления.
* Регистр CX является счетчиком. Он необходим для управления числом повторений циклов и для операций сдвига влево или вправо. Регистр CX используется также для вычислений.
* Регистр DX является регистром данных. Он применяется для некоторых операций ввода/вывода и тех операций умножения и деления над большими числами, которые используют регистровую пару DX и AX.

**4 сегментных регистра** (CS, SS, DS, ES):

* Регистр сегмента кода CS содержит начальный адрес сегмента кода. Этот адрес плюс величина смещения в командном указателе (IP) определяет адрес команды, которая должна быть выбрана для выполнения. Для обычных программ нет необходимости делать ссылки на регистр CS.
* Регистр сегмента данных DS содержит начальный адрес сегмента данных. Этот адрес плюс величина смещения, определенная в команде, указывают на конкретную ячейку в сегменте данных.
* Регистр сегмента стека SS содержит начальный адрес в сегменте стека.
* Некоторые операции над строками используют дополнительный сегментный регистр для управления адресацией памяти. В данном контексте регистр ES связан с индексным регистром DI. Если необходимо использовать регистр ES, ассемблерная программа должна его инициализировать.

**2 индексных регистра** (SI, DI):

* SI является индексом источника и применяется для некоторых операций над строками. В данном контексте регистр SI связан с регистром DS.
* DI является индексом назначения и применяется также для строковых операций. В данном контексте регистр DI связан с регистром ES.

**2 указательных** (BP, SP):

* SP указатель стека обеспечивает использование стека в памяти, позволяет временно хранить адреса и иногда данные. Этот регистр связан с регистром SS для адресации стека.
* Указатель базы BP облегчает доступ к параметрам: данным и адресам переданным через стек.

**Регистр IP** содержит смещение на команду, которая должна быть

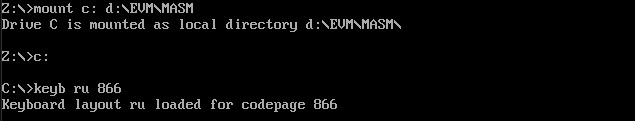
выполнена. Обычно этот регистр в программе не используется, но он может изменять свое значение при использовании отладчика DOS DEBUG для тестирования программы.

**Регистр флагов** (FLAGS, включает в себя 9 флагов): девять из 16 битов флагового регистра являются активными и определяют текущее состояние машины и результатов выполнения. Многие арифметические команды и команды сравнения изменяют состояние флагов.

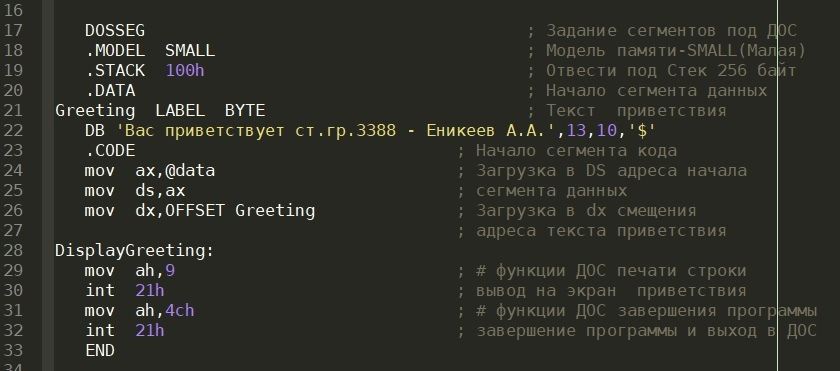
## Выполнение работы

## Часть 1.

1. Запускаем DOSBox, монтируем директорию MASM как виртуальный диск C в DOSBox, переключаемся на ранее смонтированный диск C, настраиваем раскладку клавиатуры на русскоую (RU) с кодировкой 866. (см. рис. 1)
2. Открываем программу HELLO1 с помощью редактора Geany в режиме

Рисунок 1

редактирования, строку-приветствие преобразовываем. (см. рис. 2)

Рисунок 2

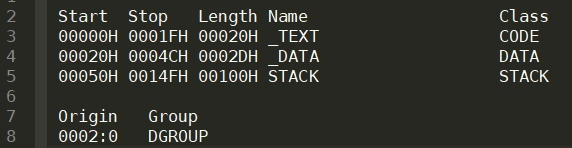
1. Транслируем программу с помощью команды masm HELLO1.ASM c созданием объектного файла HELLO1.OBJ и файла диагностических сообщений (файла листинга) HELLO1.LST. (см. рис. 3). Ошибок транслятором не обнаружено.
2. Компануем загрузочный модуль (HELLO1.EXE) с помощью команды link HELLO1.OBJ с созданием загрузочного модуля (HELLO1.EXE) и файла карты памяти (HELLO1.MAP). (см. рис. 4). Оценим по карте памяти (см. рис. 5) размещение и длину сегментов программы:

Рисунок 3

* Сегмент CODE: длина 20H
* Сегмент DATA: длина 2DH
* Сегмент STACK: длина 100H

1. Выполним программу в автоматическом режиме командой HELLO1.EXE (см. рис. 6)

Рисунок 4

Рисунок 5

1. Выполним программу HELLO1 в пошаговом режиме под управлением отладчика. Значения регистров до выполнения программы см. на рисунке 7. Фиксация используемых регистров на каждом шаге приведена в табл. 2.

Рисунок 6

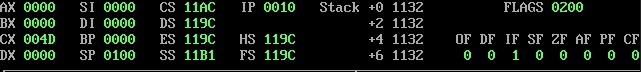
Рисунок 7

Табл. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| до вып. команды | после вып. команды |
| 0010 | MOV AX,11AE | B8AE11 | (AX) =0000  (IP) = 0010 | (AX) = 11AE  (IP) = 0013 |
| 0013 | MOV DS,AX | 8ED8 | (DS) = 119C  (IP) = 0013 | (DS) = 11AE  (IP) = 0015 |
| 0015 | MOV DX,0000 | BA0000 | (DX) = 0000  (IP) = 0015 | (DX) = 0000  (IP) = 0018 |
| 0018 | MOV AH,09 | B409 | (AX) = 11AE  (IP) = 0018 | (AX) = 09AE  (IP) = 001A |
| 001А | INT 21 | CD21 | (IP) = 001A | (IP) = 001C |
| 001C | MOV AH,4C | B44C | (AX) = 09AE  (IP) = 001C | (AX) = 4CAE (IP) = 001E |
| 001E | INT 21 | CD21 | (AX) = 4CAE (DS) = 1AE  (IP) = 001E  (CX) = 004F | (AX) = 0000 (DS) = 119C  (IP) = 0010  (CX) = 0000 |

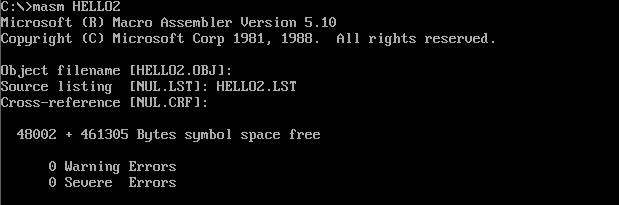
## Часть 2.

## 1. Открываем программу HELLO2 с помощью редактора Geany в режиме редактирования, строку-приветствие преобразовываем. (см. рис.8). Полный код программы см. в прил. А.

2. Транслируем программу с помощью команды masm HELLO2.ASM c созданием объектного файла HELLO2.OBJ и файла диагностических сообщений (файла листинга) HELLO2.LST. (см. рис. 9). Ошибок транслятором не обнаружено.

Рисунок 8

3. Компануем загрузочный модуль (HELLO2.EXE) с помощью команды link HELLO2.OBJ. (см. рис. 10).

Рисунок 9

4. Запускаем программу в автоматическом режиме командой HELLO2.EXE (см. рис. 11).

Рисунок 10

Рисунок 11

5. Выполним программу HELLO2 в пошаговом режиме под управлением отладчика. Значения регистров до выполнения программы см. на рисунке 12. Фиксация используемых регистров на каждом шаге приведена в табл. 3.

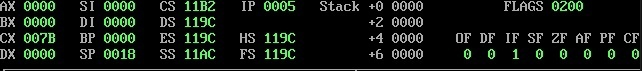
Рисунок 12

Табл. 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| до вып. команды | после вып. команды |
| 0005 | PUSH DS | 1E | Stack(+0) = 0000  (IP) = 0005  (SP) = 0018 | Stack(+0) = 119C (IP) = 0006  (SP) = 0016 |
| 0006 | SUB AX,AX | 2BC0 | (IP) = 0006  (AX) = 0000 | (IP) = 0008  (AX) = 0000 |
| 0008 | PUSH AX | 50 | (IP) = 0008  (SP) = 0016 Stack(+0) = 119C Stack(+2) = 0000 | (IP) = 0009  (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C |
| 0009 | MOV AX,11AE | B8AE11 | (AX) = 0000  (IP) = 0009 | (AX) = 11AE  (IP) = 000C |
| 000C | MOV DS,AX | 8ED8 | (DS) = 119C  (IP) = 000C | (DS) = 11AE  (IP) = 000E |
| 000E | MOV DX,0000 | BA0000 | (IP) = 000E  (DX) = 0000 | (IP) = 0011  (DX) = 0000 |
| 0011 | CALL 0000 | E8ECFF | (IP) = 0011  (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C Stack(+4) = 0000 | (IP) = 0000  (SP) = 0012  Stack(+0) = 0014  Stack(+2) = 0000 Stack(+4) = 119C |
| 0000 | MOV AH, 09 | B409 | (AX)=11AE  (IP) = 0000 | (AX)=09AE  (IP) = 0002 |
| 0002 | INT 21H | CD21 | (IP) = 0002 | (IP) = 0004 |
| 0004 | RET | C3 | (IP) = 0004  (SP) = 0012  Stack(+0) = 0014  Stack(+2) = 0000 Stack(+4) = 119C | (IP) = 0014  (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C Stack(+4) = 0000 |
| 0014 | MOV DX,0010 | BA1000 | (IP) = 0014  (DX) = 0000 | (IP) = 0017  (DX) = 0010 |
| 0017 | CALL 0000 | E8E6FF | (IP) = 0017  (SP) = 0014  Stack(+0) =0000  Stack(+2) = 119C Stack(+4) = 0000 | (IP) = 0000  (SP) = 0012 Stack(+0) =001A |
| 0000 | MOV AH, 9 | B409 | (IP) = 0000  (AX) = 09AE | (IP) = 0002  (AX) = 09AE |
| 0002 | INT 21 | CD21 | (IP) = 0002 | (IP) = 0004 |
| 0004 | RET | C3 | (IP) = 0004  (SP) = 0012 Stack(+0) = 001A Stack(+2) = 0000 Stack(+4) = 119C | (IP) = 001A (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C Stack(+4) = 0000 |
| 001A | RET Far | CB | (IP) = 001A  (CS) = 11B2  (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C | (IP) = 0000  (CS) = 119C  (SP) = 0018  Stack(+0) = 0000  Stack(+2) = 0000 |
| 0000 | INT 20 | CD20 | (AX) = 09AE (CS)= 119C  (DS) =11AE  (IP) = 0000  (CX) = 007B | (AX) = 0000 (CS)=11B2  (DS) =119C  (IP) = 0005  (CX) = 0000 |

## Выводы

В ходе выполнения лабораторной работы была освоена трансляция, изучено, как происходит выполнение и отладка программ на языке Ассемблер, а также разобраны структуры приведенных в работе программ.

# Приложение A

**Файл HELLO1.ASM:**

; HELLO1.ASM - упрощенная версия учебной программы лаб.раб. N1

; по дисциплине "Архитектура компьютера"

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Назначение: Программа формирует и выводит на экран приветствие

; пользователя с помощью функции ДОС "Вывод строки"

; (номер 09 прерывание 21h), которая:

; - обеспечивает вывод на экран строки символов,

; заканчивающейся знаком "$";

; - требует задания в регистре ah номера функции=09h,

; а в регистре dx - смещения адреса выводимой

; строки;

; - использует регистр ax и не сохраняет его

; содержимое.

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DOSSEG ; Задание сегментов под ДОС

.MODEL SMALL ; Модель памяти-SMALL(Малая)

.STACK 100h ; Отвести под Стек 256 байт

.DATA ; Начало сегмента данных

Greeting LABEL BYTE ; Текст приветствия

DB 'Вас приветствует ст.гр.3388 - Еникеев А.А.',13,10,'$'

.CODE ; Начало сегмента кода

mov ax,@data ; Загрузка в DS адреса начала

mov ds,ax ; сегмента данных

mov dx,OFFSET Greeting ; Загрузка в dx смещения

; адреса текста приветствия

DisplayGreeting:

mov ah,9 ; # функции ДОС печати строки

int 21h ; вывод на экран приветствия

mov ah,4ch ; # функции ДОС завершения программы

int 21h ; завершение программы и выход в ДОС

END

**Файл HELLO1.LST:**

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 16:49:42

Page 1-1

; Учебная программа лабораторной работы №2 по

дисциплине "Организация ЭВМ и С";

;

= 0024 EOL EQU '$'

= 0002 ind EQU 2

= 01F4 n1 EQU 500

=-0032 n2 EQU -50

; Стек программы

0000 AStack SEGMENT STACK

0000 000C[ DW 12 DUP(?)

????

]

0018 AStack ENDS

; Данные программы

0000 DATA SEGMENT

; Директивы описания данных

0000 0000 mem1 DW 0

0002 0000 mem2 DW 0

0004 0000 mem3 DW 0

0006 01 02 03 04 08 07 vec1 DB 1,2,3,4,8,7,6,5

06 05

000E F6 EC 0A 14 E2 D8 vec2 DB -10,-20,10,20,-30,-40,30,40

1E 28

0016 01 02 03 04 FC FD matr DB 1,2,3,4,-4,-3,-2,-1,5,6,7,8,-8,

-7,-6,-5

FE FF 05 06 07 08

F8 F9 FA FB

0026 DATA ENDS

; Код программы

0000 CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура

0000 Main PROC FAR

0000 1E push DS

0001 2B C0 sub AX,AX

0003 50 push AX

0004 B8 ---- R mov AX,DATA

0007 8E D8 mov DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИ

Й

; Регистровая адресация

0009 B8 01F4 mov ax,n1

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 16:49:42

Page 1-2

000C 8B C8 mov cx,ax

000E B3 24 mov bl,EOL

0010 B7 CE mov bh,n2

; Прямая адресация

0012 C7 06 0002 R FFCE mov mem2,n2

0018 BB 0006 R mov bx,OFFSET vec1

001B A3 0000 R mov mem1,ax

; Косвенная адресация

001E 8A 07 mov al,[bx]

mov mem3,[bx]

LR2\_comp.ASM(54): error A2052: Improper operand type

; Базированная адресация

0020 8A 47 03 mov al,[bx]+3

0023 8B 4F 03 mov cx,3[bx]

; Индексированная адресация

0026 BF 0002 mov di,ind

0029 8A 85 000E R mov al,vec2[di]

002D 8B 8D 000E R mov cx,vec2[di]

LR2\_comp.ASM(61): warning A4031: Operand types must match

; Адресация с базированием и индексированием

0031 BB 0003 mov bx,3

0034 8A 81 0016 R mov al,matr[bx][di]

0038 8B 89 0016 R mov cx,matr[bx][di]

LR2\_comp.ASM(65): warning A4031: Operand types must match

003C 8B 85 0022 R mov ax,matr[bx\*4][di]

LR2\_comp.ASM(66): error A2055: Illegal register value

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ вариант 1

0040 B8 ---- R mov ax, SEG vec2

0043 8E C0 mov es, ax

0045 26: 8B 07 mov ax, es:[bx]

0048 B8 0000 mov ax, 0

; ------ вариант 2

004B 8E C0 mov es, ax

004D 1E push ds

004E 07 pop es

004F 26: 8B 4F FF mov cx, es:[bx-1]

0053 91 xchg cx,ax

; ------ вариант 3

0054 BF 0002 mov di,ind

0057 26: 89 01 mov es:[bx+di],ax

; ------ вариант 4

005A 8B EC mov bp,sp

005C 3E: 8B 86 0016 R mov ax,matr[bp+bx]

LR2\_comp.ASM(86): error A2046: Multiple base registers

0061 3E: 8B 83 0016 R mov ax,matr[bp+di+si]

LR2\_comp.ASM(87): error A2047: Multiple index registers

; Использование сегмента стека

0066 FF 36 0000 R push mem1

006A FF 36 0002 R push mem2

006E 8B EC mov bp,sp

0070 8B 56 02 mov dx,[bp]+2

0073 CB ret

0074 Main ENDP

LR2\_comp.ASM(94): error A2006: Phase error between passes

0074 CODE ENDS

END Main

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 16:49:42

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0074 PARA NONE

DATA . . . . . . . . . . . . . . 0026 PARA NONE

Symbols:

N a m e Type Value Attr

EOL . . . . . . . . . . . . . . NUMBER 0024

IND . . . . . . . . . . . . . . NUMBER 0002

MAIN . . . . . . . . . . . . . . F PROC 0000 CODE Length = 0074

MATR . . . . . . . . . . . . . . L BYTE 0016 DATA

MEM1 . . . . . . . . . . . . . . L WORD 0000 DATA

MEM2 . . . . . . . . . . . . . . L WORD 0002 DATA

MEM3 . . . . . . . . . . . . . . L WORD 0004 DATA

N1 . . . . . . . . . . . . . . . NUMBER 01F4

N2 . . . . . . . . . . . . . . . NUMBER -0032

VEC1 . . . . . . . . . . . . . . L BYTE 0006 DATA

VEC2 . . . . . . . . . . . . . . L BYTE 000E DATA

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT LR2\_comp

@VERSION . . . . . . . . . . . . TEXT 510

96 Source Lines

96 Total Lines

19 Symbols

47800 + 459460 Bytes symbol space free

2 Warning Errors

5 Severe Errors

**Файл HELLO1.MAP:**

Start Stop Length Name Class

00000H 0001FH 00020H \_TEXT CODE

00020H 0004CH 0002DH \_DATA DATA

00050H 0014FH 00100H STACK STACK

Origin Group

0002:0 DGROUP

0001:0 DGROUP

**Файл HELLO2.ASM:**

; HELLO2 -Учебная программа N2 лаб.раб.#1 по дисциплине "Архитектура компьютера"

; Программа использует процедуру для печати строки

;

; ТЕКСТ ПРОГРАММЫ

EOFLine EQU '$' ; Определение символьной константы

; "Конец строки"

; Стек программы

AStack SEGMENT STACK

DW 12 DUP(?) ; Отводится 12 слов памяти

AStack ENDS

; Данные программы

DATA SEGMENT

; Директивы описания данных

HELLO DB 'Здравствуйте!', 0AH, 0DH,EOFLine

GREETING DB 'Вас приветствует ст.гр.3388 - Еникеев А.А.$'

DATA ENDS

; Код программы

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Процедура печати строки

WriteMsg PROC NEAR

mov AH,9

int 21h ; Вызов функции DOS по прерыванию

ret

WriteMsg ENDP

; Головная процедура

Main PROC FAR

push DS ;\ Сохранение адреса начала PSP в стеке

sub AX,AX ; > для последующего восстановления по

push AX ;/ команде ret, завершающей процедуру.

mov AX,DATA ; Загрузка сегментного

mov DS,AX ; регистра данных.

mov DX, OFFSET HELLO ; Вывод на экран первой

call WriteMsg ; строки приветствия.

mov DX, OFFSET GREETING ; Вывод на экран второй

call WriteMsg ; строки приветствия.

ret ; Выход в DOS по команде,

; находящейся в 1-ом слове PSP.

Main ENDP

CODE ENDS

END Main

**Файл HELLO2.LST:**

#Microsoft (R) Macro Assembler Version 5.10 9/19/24 16:12:26

Page 1-1

; HELLO2 -Учебная программа N2 лаб.раб.#1 по ди

сциплине "Архитектура компьютера"

; Программа использует процедуру для п

ечати строки

;

; ТЕКСТ ПРОГРАММЫ

= 0024 EOFLine EQU '$' ; Определение символь

ной константы

; "Конец строки"

; Стек программы

0000 AStack SEGMENT STACK

0000 000C[ DW 12 DUP(?) ; Отводится 12 слов п

амяти

????

]

0018 AStack ENDS

; Данные программы

0000 DATA SEGMENT

; Директивы описания данных

0000 87 A4 E0 A0 A2 E1 HELLO DB 'Здравствуйте!', 0AH, 0DH,EOFLine

E2 A2 E3 A9 E2 A5

21 0A 0D 24

0010 82 A0 E1 20 AF E0 GREETING DB 'Вас приветствует ст.гр.3388 - Ени

кеев А.А.$'

A8 A2 A5 E2 E1 E2

A2 E3 A5 E2 20 E1

E2 2E A3 E0 2E 34

33 35 30 20 2D 20

85 AD A8 AA A5 A5

A2 20 80 2E 80 2E

24

003B DATA ENDS

; Код программы

0000 CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Процедура печати строки

0000 WriteMsg PROC NEAR

0000 B4 09 mov AH,9

0002 CD 21 int 21h ; Вызов функции DOS по пре

рыванию

0004 C3 ret

0005 WriteMsg ENDP

#Microsoft (R) Macro Assembler Version 5.10 9/19/24 16:12:26

Page 1-2

; Головная процедура

0005 Main PROC FAR

0005 1E push DS ;\ Сохранение адреса

начала PSP в стеке

0006 2B C0 sub AX,AX ; > для последующего в

осстановления по

0008 50 push AX ;/ команде ret, завер

шающей процедуру.

0009 B8 ---- R mov AX,DATA ; Загрузка

сегментного

000C 8E D8 mov DS,AX ; регистра

данных.

000E BA 0000 R mov DX, OFFSET HELLO ; Вывод на

экран первой

0011 E8 0000 R call WriteMsg ; строки пр

иветствия.

0014 BA 0010 R mov DX, OFFSET GREETING ; Вывод на

экран второй

0017 E8 0000 R call WriteMsg ; строки пр

иветствия.

001A CB ret ; Выход в D

OS по команде,

; находящей

ся в 1-ом слове PSP.

001B Main ENDP

001B CODE ENDS

END Main

#Microsoft (R) Macro Assembler Version 5.10 9/19/24 16:12:26

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 001B PARA NONE

DATA . . . . . . . . . . . . . . 003B PARA NONE

Symbols:

N a m e Type Value Attr

EOFLINE . . . . . . . . . . . . NUMBER 0024

GREETING . . . . . . . . . . . . L BYTE 0010 DATA

HELLO . . . . . . . . . . . . . L BYTE 0000 DATA

MAIN . . . . . . . . . . . . . . F PROC 0005 CODE Length = 0016

WRITEMSG . . . . . . . . . . . . N PROC 0000 CODE Length = 0005

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT HELLO2

@VERSION . . . . . . . . . . . . TEXT 510

52 Source Lines

52 Total Lines

13 Symbols

48002 + 461305 Bytes symbol space free

0 Warning Errors

0 Severe Errors

**Файл HELLO2.MAP:**

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 0005AH 0003BH DATA

00060H 0007AH 0001BH CODE

Program entry point at 0006:0005