**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

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

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

|  |  |
| --- | --- |
| Студент гр. 3382 | Копасова К. А. |
| Преподаватель | Фирсов М. А. |

Санкт-Петербург

2024

**Цель работы:** освоить трансляцию, выполнение и отладку программ на языке Ассемблера процессора Intel X86.

**Задание:**

Часть 1.

0. Подготовить среду для запуска исполняемых файлов DOS. Загрузить файл hello1.asm в каталог \MASM.

1. 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.

2. 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).

3. 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.

4. 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);

5. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 5 части 1.

**Теоретические сведения:**

**1. Регистры процессора (**Регистры – маленькие ячейки памяти на процессоре.)

В процессорах x86 есть несколько типов регистров, которые играют важную роль в ассемблерных программах:

* **Регистры общего назначения**:
  + AX, BX, CX, DX – используются для различных арифметических и логических операций.
  + Младшие и старшие байты этих регистров: AL, AH, BL, BH, CL, CH, DL, DH.
* **Сегментные регистры**:
  + CS (Code Segment) – указывает на сегмент кода.
  + DS (Data Segment) – указывает на сегмент данных.
  + SS (Stack Segment) – указывает на сегмент стека.
  + ES (Extra Segment) – дополнительный сегмент для операций с данными.
  + SP ((Stack Pointer) – указывает на вершину стека.), ID (Destination Index)- адресация данных) и т. п. – дополнительные регистры, отвечающие за определенные аспекты.

**2. Директивы ASSUME и DOSSEG**

* **ASSUME** – сообщает компилятору, какие сегменты использовать для работы с регистрами. Например:

*ASSUME CS:CODE, DS:DATA, SS:STACK*

Это указывает, что регистр CS содержит сегмент кода, DS – сегмент данных, а SS – сегмент стека. Без этой директивы компилятор не сможет правильно интерпретировать, к каким сегментам относятся инструкции и данные.

* **DOSSEG** – эта директива используется для установки сегментов в порядке, который ожидает DOS: код, затем данные, затем стек. Она важна при создании .exe программ.

**3. DGROUP**

* **DGROUP** – это специальная группа сегментов, которая объединяет несколько сегментов данных в одно целое. Обычно объединяются сегменты .DATA и .STACK. Это делается для упрощения доступа к переменным и стеку через один сегментный регистр (DS).

**4. Расположение сегментов в памяти**

В программе, работающей под DOS, обычно выделяются три сегмента:

* **Сегмент кода (CODE)** – хранит машинный код программы.
* **Сегмент данных (DATA)** – хранит глобальные переменные и константы.
* **Сегмент стека (STACK)** – используется для временного хранения данных, таких как адреса возврата и локальные переменные.

Обычно сегменты располагаются в памяти в таком порядке: код, данные, стек. Однако директивы и настройки компилятора могут менять этот порядок.

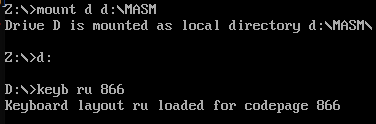
**5. Директивы сегментов**

* **.MODEL** – задаёт модель памяти, которая определяет, как программа будет работать с сегментами:
  + **TINY** – для программ с одним сегментом (используется для .COM файлов).
  + **SMALL** – программа имеет один сегмент кода и один сегмент данных (наиболее распространённая модель для .EXE программ).
  + **MEDIUM** – один сегмент данных и несколько сегментов кода.
  + **COMPACT** – несколько сегментов данных и один сегмент кода.
  + **LARGE** и **HUGE** – несколько сегментов кода и данных.
* **.STACK** – задаёт размер сегмента стека.
* **.DATA** – определяет сегмент данных, где хранятся переменные.
* **.CODE** – определяет сегмент кода, где размещены инструкции программы.

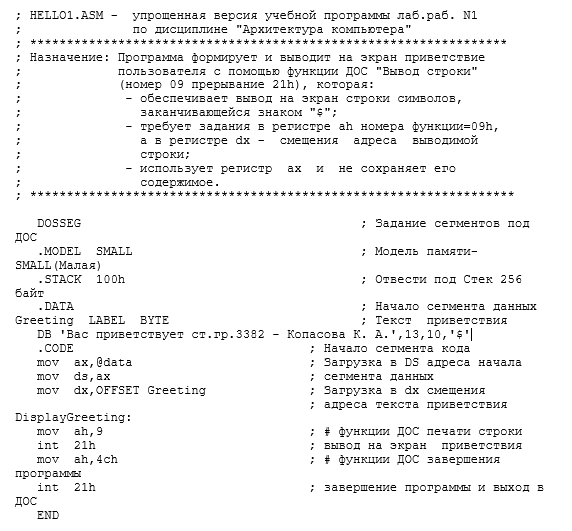
**Выполнение работы**

**Часть 1.**

1. Подготавливаем среду для запуска исполняемых файлов DOS. Загружаем файл HELLO1.ASM в каталог \MASM. Монтируем диск в DosBox и переключаем кодировку на CP866 для работы с кириллицей.

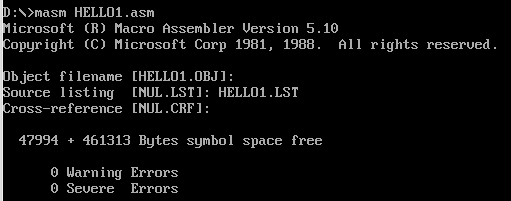


1. Просматриваем программу в режиме редактирования, разбираемся в структуре каждого сегмента программы. Изменяем строку-приветствие в соответствии со своими личными данными.



2. Транслируем программу HELLO1.asm с помощью строки:

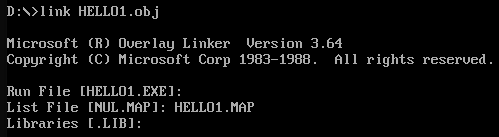
*>masm HELLO1.asm*



3. Скомпонуем загрузочный модуль с помощью строки

*>link HELLO1.obj*

С созданием загрузочного модуля (HELLO1.EXE) и файла карты памяти (HELLO1.MAP).



4. Выполним программу в автоматическом режиме путем набора строки

*>HELLO1.EXE*

Программа выполняется верно.

5. Выполним программу HELLO1 в пошаговом режиме под управлением отладчика с помощью команды

*>afd HELLO1.EXE*

Содержимое сегментных регистров до выполнения программы:

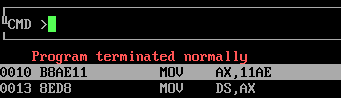


Выполним в пошаговом режиме с фиксацией используемых регистров и ячеек памяти до и после выполнения каждой команды, используя отладчик.

Таблица 2.

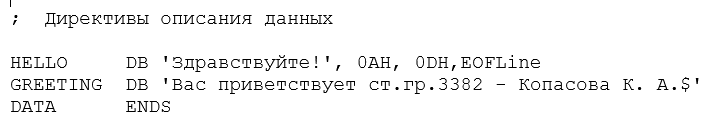
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  команды | Символический  код команды | 16-ричный  код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0010 | MOV AX, 11AE | B8AE11 | (AX) = 0000  (IP) = 0010 | (AX) = 11AE  (IP) = 0013 |
| 0013 | MOV DS, AX | 8ED8 | (DS) = 119C  (IP) = 0013 | (DS) = 11AE  (IP) = 0015 |
| 0015 | MOV DX, 0000 | BA0000 | (IP) = 0015 | (IP) = 0018 |
| 0018 | MOV AH, 09 | B409 | (AX) = 11AE  (IP) = 0018 | (AX) = 09AE  (IP) = 001A |
| 001A | 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) = 11AE  (IP) = 001E | (AX) = 0000  (DS) = 119C  (IP) = 0010 |

Сообщение после выполнения программы:

****

**Часть 2**

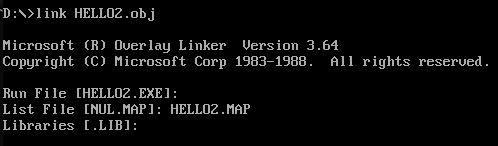
1. Просматриваем файл hello2.asm в режиме редактирования, разбираемся в структуре и реализации каждого сегмента программы. Преобразовываем сроку приветствия в соответствии со своими личными данными.



2. Выполняем трансляцию программы HELLO2.ASM с помощью транслятора *masm* (см. Часть 1) с созданием объектного файла (HELLO2\_test.obj) и файла диагностических сообщений (HELLO2\_test.lst). Ошибок в ходе создания файлов обнаружено не было.



3. Скомпонуем загрузочный модуль HELLO2.exe с помощью компоновщика *link* (см. Часть 1)



4. Выполним программу с помощью команды

*>HELLO2\_T.EXE*

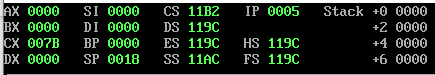
И убедимся, что она выполняется верно.



5. Выполним программу HELLO2 в пошаговом режиме под управлением отладчика с помощью команды

*>afd HELLO2\_T.EXE*

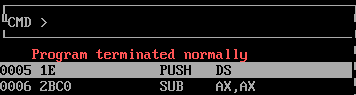
Содержимое сегментных регистров до выполнения программы:



Выполним в пошаговом режиме с фиксацией используемых регистров и ячеек памяти до и после выполнения каждой команды, используя отладчик.

Таблица 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  команды | Символический  код команды | 16-ричный  код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0005 | PUSH DS | 1E | (SP) = 0018  (IP) = 0005  STACK +0 0000  +2 0000  +4 0000  +6 0000 | (SP) = 0016  (IP) = 0006  STACK +0 119C  +2 0000  +4 0000  +6 0000 |
| 0006 | SUB AX, AX | 2BC0 | (IP) = 0006 | (IP) = 0008 |
| 0008 | PUSH AX | 50 | (SP) = 0016  (IP) = 0008  STACK +0 119C  +2 0000  +4 0000  +6 0000 | (SP) = 0014  (IP) = 0009  STACK +0 0000  +2 119C  +4 0000  +6 0000 |
| 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 | (IP) = 0011 |
| 0011 | CALL 0000 | E8ECFF | (SP) = 0014  (IP) = 0011  STACK +0 0000  +2 119C  +4 0000  +6 0000 | (SP) = 0012  (IP) = 0000  STACK +0 0014  +2 0000  +4 119C  +6 0000 |
| 0000 | MOV AH, 09 | B409 | (AX) = 11AE  (IP) = 0000 | (AX) = 09AE  (IP) = 0002 |
| 0002 | INT 21 | CD21 | (IP) = 0002 | (IP) = 0004 |
| 0004 | RET | C3 | (SP) = 0012  (IP) = 0004  STACK +0 0014  +2 0000  +4 119C  +6 0000 | (SP) = 0014  (IP) = 0014  STACK +0 0000  +2 119C  +4 0000  +6 0000 |
| 0014 | MOV DX, 0010 | BA1000 | (DX) = 0000  (IP) = 0014 | (DX) = 0010  (IP) = 0017 |
| 0017 | CALL 0000 | E8E6FF | (SP) = 0014  (IP) = 0017  STACK +0 0000  +2 119C  +4 0000  +6 0000 | (SP) = 0012  (IP) = 0000  STACK +0 001A  +2 0000  +4 119C  +6 0000 |
| 0000 | MOV AH, 09 | B409 | (IP) = 0000 | (IP) = 0002 |
| 0002 | INT 21 | CD21 | (IP) = 0002 | (IP) = 0004 |
| 0004 | RET | C3 | (SP) = 0012  (IP) = 0004  STACK +0 001A  +2 0000  +4 119C  +6 0000 | (SP) = 0014  (IP) = 001A  STACK +0 0000  +2 119C  +4 0000  +6 0000 |
| 001A | RET FAR | CB | (SP) = 0014  (CS) = 11B2  (IP) = 001A  STACK +0 0000  +2 119C  +4 0000  +6 0000 | (SP) = 0018  (CS) = 119C  (IP) = 0000  STACK +0 0000  +2 0000  +4 0000  +6 0000 |
| 0000 | INT 20 | CD20 | (IP) = 0000  (AX) = 09AE  (CX) = 007B  (CS) = 119C  (DS) = 11AE | (IP) = 0005  (AX) = 0000  (CX) = 0000  (CS) = 11B2  (DS) = 119C |

Сообщение отладчика после выполнения программы:

**Вывод**

В ходе данной лабораторной работы были выполнены анализ, трансляция, отладка и выполнение двух программ на языке Ассемблера. Изучены директивы .CODE, .DATA, .STACK, SEGMENT, ASSUME, DOSSEG для работы с памятью, ENDS, ENDF, PROC для работы с процессами и команды mov, int, sub, ret, call, push. Освоено использование регистров. Все файлы, созданные и использованные в процессе лабораторной работы находятся в Приложении 1.

**ПРИЛОЖЕНИЕ 1**

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.3382 - Копасова К. 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 9/9/24 10:36:15

Page 1-1

1

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

3 ;по дисциплине "Архитектура компьютера"

4 ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5 ; Назначение: Программа формирует и выводит на экран приветствие

6 ; пользователя с помощью функции ДОС "Вывод строки"

7 ; (номер 09 прерывание 21h), которая:

8 ; - обеспечивает вывод на экран строки символов,

9 ; заканчивающейся знаком "$";

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

11 ; а в регистре dx - смещения адреса выводимой

12 ; строки;

13 ; - использует регистр ax и не сохраняет его

14 ; содержимое.

15 ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

16

17 DOSSEG ;Задание сегментов под ДОС

18 .MODEL SMALL ; Модель памяти-SMALL(Малая)

19 .STACK 100h ; Отвести под Стек 256 байт

20 .DATA ; Начало сегмента данных

21 0000 Greeting LABEL BYTE ; Текст приветствия

22 0000 82 A0 E1 20 AF E0 DB 'Вас приветствует ст.гр.3382 - Копасова К. А.',13,10,'$'

23 A8 A2 A5 E2 E1 E2

24 A2 E3 A5 E2 20 E1

25 E2 2E A3 E0 2E 33

26 33 38 32 20 2D 20

27 8A AE AF A0 E1 AE

28 A2 A0 20 8A 2E 20

29 80 2E 0D 0A 24

30 .CODE ; Начало сегмента кода

31 0000 B8 ---- R mov ax,@data ; Загрузка в DS адреса начала

32 0003 8E D8 mov ds,ax ; сегмента данных

33 0005 BA 0000 R mov dx,OFFSET Greeting ; Загрузка в dx смещения

34 ; адреса текста приветствия

35 0008 DisplayGreeting:

36 0008 B4 09 mov ah,9 ; # функции ДОС печати строки

37 000A CD 21 int 21h ; вывод на экран приветствия

38 000C B4 4C mov ah,4ch ; # функции ДОС завершения программы

39 000E CD 21 int 21h ; завершение программы и выход в ДОС

40 END

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

DGROUP . . . . . . . . . . . . . GROUP

DATA . . . . . . . . . . . . 002F WORD PUBLIC 'DATA'

STACK . . . . . . . . . . . . 0100 PARA STACK 'STACK'

\_TEXT . . . . . . . . . . . . . 0010 WORD PUBLIC 'CODE'

Symbols:

N a m e Type Value Attr

DISPLAYGREETING . . . . . . . . L NEAR 0008 \_TEXT

GREETING . . . . . . . . . . . . L BYTE 0000 \_DATA

@CODE . . . . . . . . . . . . . TEXT \_TEXT

@CODESIZE . . . . . . . . . . . TEXT 0

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

@DATASIZE . . . . . . . . . . . TEXT 0

@FILENAME . . . . . . . . . . . TEXT HELLO1

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

33 Source Lines

33 Total Lines

19 Symbols

47442 + 461865 Bytes symbol space free

0 Warning Errors

0 Severe Errors

Файл HELLO1.MAP:

Start Stop Length Name Class

00000H 0001FH 00020H \_TEXT CODE

00020H 0004EH 0002FH \_DATA DATA

00050H 0014FH 00100H STACK STACK

Origin Group

0002:0 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 'Добро пожаловать!', 0AH, 0DH,EOFLine

GREETING DB 'Вас приветствует ст.гр. 3382 - Копасова К. 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 9/12/24 18:24:28

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

; Программа использует процедуру для печати строки

;

; ТЕКСТ ПРОГРАММЫ

= 0024

EOFLine EQU '$' ; Определение символьной константы

; "Конец строки"

; Стек программы

0000 AStack SEGMENT STACK

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

????

]

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.3382 - Коп

асова К. А.$'

A8 A2 A5 E2 E1 E2

A2 E3 A5 E2 20 E1

E2 2E A3 E0 2E 33

33 38 32 20 2D 20

8A AE AF A0 E1 AE

A2 A0 20 8A 2E 20

80 2E 24

003D 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 прерыванию

0004 C3 ret

0005 WriteMsg ENDP

; Головная процедура

0005 Main PROC FAR

0005 1E push DS ;\ Сохранение адреса начала PSP в стеке

0006 2B C0 sub AX,AX ; > для последующего восстановления по

0008 50 push AX ;/ команде ret, завершающей 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 приветствия.

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

0017 E8 0000 R call WriteMsg ; строки приветствия.

001A CB ret ; Выход в D по команде,

; находящейся в 1-ом слове PSP.

001B Main ENDP

001B CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10 9/12/24 18:24:28

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

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

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

DATA . . . . . . . . . . . . . . 003D 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 0005CH 0003DH DATA

00060H 0007AH 0001BH CODE

Program entry point at 0006:0005