# 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**»

Студентка гр. 3388 Беннер В.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.

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).

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 команд ы | Символическ ий 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.3Скомпоновать загрузочный модуль hello2.exe с помощью компоновщика Link.
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.

# Основные теоретические положения.

Все внутренние регистры процессора Intel 8086 являются 16-битными. Всего процессор содержит 12 программно-доступных регистров, а также регистр флагов (FLAGS) и указатель команд (IP).

**Регистры общего назначения (РОН)** AX, BX, CX и DX используются для хранения данных и выполнения различных арифметических и логических операций. Кроме того, каждый из этих регистров поделён на 2 части по 8-бит, с которыми можно работать как с 8-битными регистрами (AH, AL, BH, BL, CH, CL, DH, DL). Младшие части регистров имеют в названии букву L (от слова Low), а старшие H (от слова High). Некоторые команды неявно используют определённый регистр, например, CX может выполнять роль счетчика цикла.

**Индексные регистры** предназначены для хранения индексов при работе с массивами. SI (Source Index) содержит индекс источника, а DI (Destination Index) — индекс приёмника, хотя их можно использовать и как регистры общего назначения.

**Регистры-указатели BP и SP** используются для работы со стеком. BP (Base Pointer) позволяет работать с переменными в стеке. Его также можно использовать в других целях. SP (Stack Pointer) указывает на вершину стека. Он используется командами, которые работают со стеком. (Про стек я подробно расскажу в отдельной части учебного курса)

**Сегментные регистры.** Cегментом называется область, которая начинается на границе параграфа, т.е по любому адресу, который делится на 16 без остатка. Хотя сегмент может располагаться в любом месте памяти и иметь размер до 64 Кбайт, он требует столько памяти, сколько необходимо для выполнения программы. Имеется три главных сегмента:

1. Сегмент кодов. Сегмент кодов содержит машинные команды, которые будут выполняться. Обычно первая выполняемая команда находится в начале этого сегмента и операционная система передает управление по адресу данного сегмента для выполнения программы. Регистр сегмента кодов (CS) адресует данный сегмент.

2. Сегмент данных. Сегмент данных содержит определенные данные, константы и рабочие области, необходимые программе. Регистр сегмента данных (DS) адресует данный сегмент.

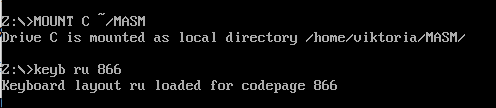
3. Сегмент стека. Стек содержит адреса возврата как для программы для возврата в операционную систему, так и для вызовов программ для возврата в главную программу. Регистр сегмента стека (SS) адресует данный сегмент.

COM-программа всегда находится в одном сегменте, который является одновременно сегментом кода, данных и стека. При запуске COM-программы сегментные регистры будут содержать одинаковые значения.

**Указатель команд IP** (Instruction Pointer) содержит адрес команды (в сегменте кода). Напрямую изменять его содержимое нельзя, но процессор делает это сам. При выполнении обычных команд значение IP увеличивается на размер выполненной команды. Существуют также команды передачи управления, которые изменяют значение IP для осуществления переходов внутри программы.

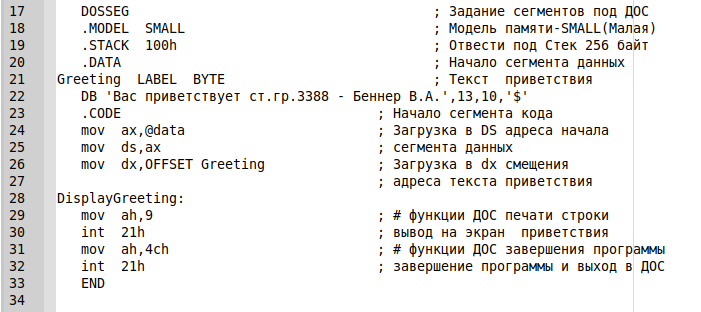
# Выполнение работы Часть 1.

0. В консоли DosBox при помощи команды mount директория MASM была смонтирована как диск С, также был подключен русский язык с помощью команды keyb ru 866. (см. рис. 1)

 Рисунок 1

5

1. Программа HELLO1.ASM была просмотрена в режиме редактирования с помощью текстового редактора geany, была изучена её структура и реализация каждого её сегмента. Строка- приветствие была изменена в соответствии с личными данными выполнявшего работу. (см. рис. 2)

Рисунок 2

1. Программа была протранслирована с помощью команды masm hello1 c созданием объектного файла HELLO1.OBJ и файла диагностических сообщений (файла листинга) HELLO1.LST. (см. рис. 3)



Рисунок 3

1. Был скомпонован загрузочный модуль (HELLO1.EXE) с помощью команды link hello1 с созданием загрузочного модуля (HELLO1.EXE) и фаsйла карты памяти (HELLO1.MAP). (см. рис. 4)

Рисунок 4

1. Программа была запущена в автоматическом режиме командой hello1 (см. рис. 5)

 Рисунок 5

1. Программа была выполнена в пошаговом режиме с использованием отладчика afd. Значения регистров до выполнения программы см. на рисунке 6. Фиксация используемых регистров на каждом шаге приведена в табл. 1.

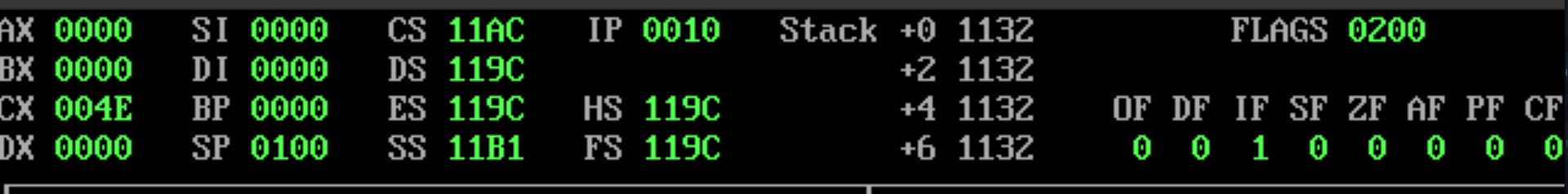


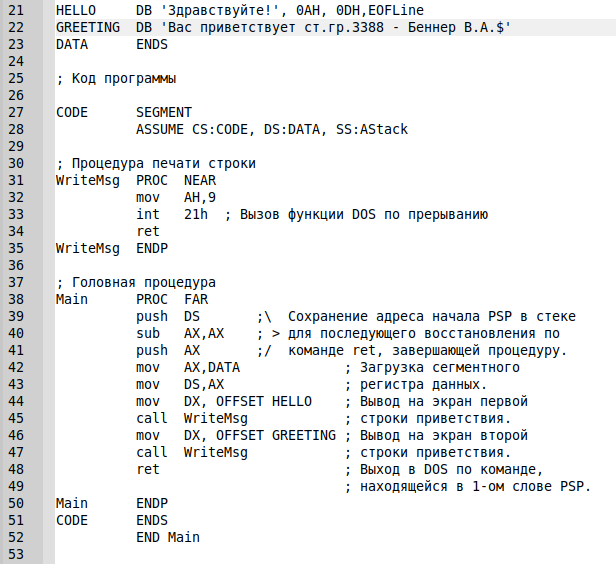
Рисунок 6

Таблица 1 – Протокол пошагового исполнения HELLO1.EXE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 1 6 - р и ч н ы й код команды | Содержимое регистров и ячеек  памяти | |
| До  выполнения | После  выполнения |
| 0010 | MOV AX,11AX | 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  (DX) = 0000 | (IP) = 0018  (DX) = 0000 |
| 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) = 11AE (IP) = 001E (CX) = 004D | (AX) = 0000 (DS) = 119C (IP) = 0010  (CX) = 0000 |

# Часть 2.

1. Программа HELLO2.ASM была просмотрена в режиме редактирования с помощью текстового редактора geany, была изучена её структура и реализация каждого её сегмента. Строка-приветствие была изменена в соответствии с личными данными выполнявшего работу. (см. рис.7)

Рисунок 7

1. Программа была протранслирована с помощью команды masm hello2 c созданием объектного файла HELLO2.OBJ и файла диагностических сообщений (файла листинга) HELLO2.LST. (см. рис.8)

Рисунок 8

1. Был скомпонован загрузочный модуль (HELLO2.EXE) с помощью команды link hello2 с созданием загрузочного модуля (HELLO2.EXE) и файла карты памяти (HELLO2.MAP). (см. рис.9)

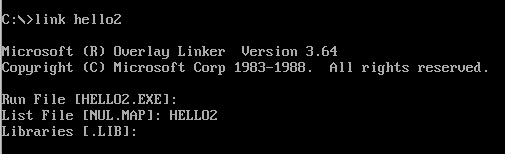


Рисунок 9

1. Программа была запущена в автоматическом режиме командой

hello2 (см. рис. 10)

Рисунок 10

1. Программа была выполнена в пошаговом режиме с использованием отладчика afd. Значения регистров до выполнения программы см. на рисунке 11. Фиксация используемых регистров на каждом шаге приведена в табл. 2.

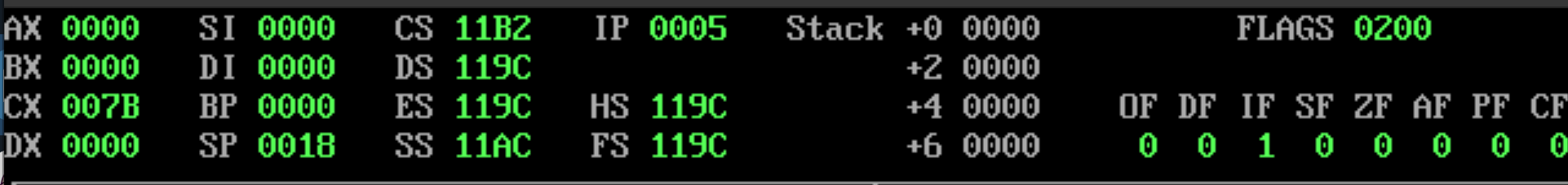


Рисунок 11

Таблица 2 - Протокол пошагового исполнения hello2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  команды | Символический  код команды | 1 6 - р и ч н ы й  код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 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 | (IP) = 0000  (SP) = 0012 Stack(+0) =001A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Stack(+2) = 119C Stack(+4) = 0000 | Stack(+2) = 0000 Stack(+4) = 119C |
| 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 |

# Вывод.

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 А**

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 - Беннер В.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

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 - Беннер В.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

HELLO1.LIST

#Microsoft (R) Macro Assembler Version 5.10 9/13/24 01:44:50

Page 1-1

; HELLO1.ASM - упрощенная версия учебной прогр

аммы лаб.раб. N1

; по дисциплине "Архитектура комп

ьютера"

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Назначение: Программа формирует и выводит на

экран приветствие

; пользователя с помощью функции ДО

С "Вывод строки"

; (номер 09 прерывание 21h), котора

я:

; - обеспечивает вывод на экран ст

роки символов,

; заканчивающейся знаком "$";

; - требует задания в регистре ah

номера функции=09h,

; а в регистре dx - смещения а

дреса 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

0000 Greeting LABEL BYTE

; Текст приветствия

0000 82 A0 E1 20 AF E0 DB 'Вас приветствует ст.гр.3388 - Беннер В.А

.',13,10,'$'

A8 A2 A5 E2 E1 E2

A2 E3 A5 E2 20 E1

E2 2E A3 E0 2E 33

33 38 38 20 2D 20

81 A5 AD AD A5 E0

20 82 2E 80 2E 0D

0A 24

.CODE ; Начал

о сегмента кода

0000 B8 ---- R mov ax,@data ; Загру

зка в DS адреса начала

0003 8E D8 mov ds,ax ; сегме

нта данных

0005 BA 0000 R mov dx,OFFSET Greeting ; Загру

зка в dx смещения

#Microsoft (R) Macro Assembler Version 5.10 9/13/24 01:44:50

Page 1-2

; адрес

а текста приветствия

0008 DisplayGreeting:

0008 B4 09 mov ah,9 ; # фун

кции ДОС печати строки

000A CD 21 int 21h ; вывод

на экран приветствия

000C B4 4C mov ah,4ch ; # фун

кции ДОС завершения программы

000E CD 21 int 21h ; завер

шение программы и выход в ДОС

END

#Microsoft (R) Macro Assembler Version 5.10 9/13/24 01:44:50

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

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

\_DATA . . . . . . . . . . . . 002C 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

47994 + 461313 Bytes symbol space free

0 Warning Errors

0 Severe Errors

HELLO2.LIST

#Microsoft (R) Macro Assembler Version 5.10 9/8/24 14:20:36

Page 1-1

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

дисциплине "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 - Бен

нер В.А.$'

A8 A2 A5 E2 E1 E2

A2 E3 A5 E2 20 E1

E2 2E A3 E0 2E 33

33 38 38 20 2D 20

81 A5 AD AD A5 E0

20 82 2E 80 2E 24

003A 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

; Головная процедура

#Microsoft (R) Macro Assembler Version 5.10 9/8/24 14:20:36

Page 1-2

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 9/8/24 14:20:36

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

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

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

DATA . . . . . . . . . . . . . . 003A 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

HELLO1.MAP

Start Stop Length Name Class

00000H 0001FH 00020H \_TEXT CODE

00020H 0004BH 0002CH \_DATA DATA

00050H 0014FH 00100H STACK STACK

Origin Group

0002:0 DGROUP

HELLO2.MAP

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 00059H 0003AH DATA

00060H 0007AH 0001BH CODE

Program entry point at 0006:0005