МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 2

Разработка программы ввода-вывода и обработки   
последовательности кодов на ассемблере

Вариант №13

ОТЧЕТ

по лабораторной работе

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сапожников В.О.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

1. **Цель работы**

Приобретение навыков: разработки одно- и многокомпонентных программ на языке ассемблер, использования функции прерывания для организации ввода-вывода, управление трансляцией и компоновкой.

1. **Краткая теоретическая часть**

**Транслятор** обрабатывает текст программы, формируя локальный счетчик команд. Каждый сегмент программы транслируется отдельно (начиная с 0). Значение счетчика в текущей тоске программы возвращает **оператор $**. При обработке очередной команды (или данных) локальный счетчик увеличивается на число соответствующий байтов. Таким образом, значение счетчика определяет смещение команды или данных в соответствующем сегменте.

Компоновщик может создавать выполнимый файл с расширениями:

**.*com*** – содержит только код без какой-либо дополнительной информации (весь код в одном сегменте 64K, необходим ключ /t)

**.*exe*** – содержит заголовок и загрузочный модуль. В заголовке находится размер загрузочного модуля, относительные адреса: сегмента стека и sp, сегмента кода и ip.

Директивы определения сегментов:

имя ***SEGMENT*** [атрибуты]

тело сегмента

имя ***ENDS***

Атрибуты могут состоять из одного или нескольких следующий значений:

* **Выравнивание** – сообщает компоновщику о необходимости начать сегмент с указанной границы. По умолчанию – para
  + ***byte*** – начать сегмент с любого байта (выравнивание не выполняется)
  + ***word*** – с четного адреса (выровнять границу следующего слова)
  + ***dword*** – с адреса кратного 4 (выравнивание на границу следующего двойного слова)
  + ***para*** – с адреса кратного 16 (выравнивание на границу параграфа)
  + ***page*** – с адреса, кратного 256 (выравнивание на границу страницы)
* **Объединение** – указывает компоновщику возможность и способы объединения сегментов с одинаковым именем, находящихся в разных модулях. По умолчанию – ***PRIVATE***.
  + ***PUBLIC* –** конкатенация сегментов (сегмент будет объединятся с сегментами, имеющими такое же имя в один сегмент).
  + ***STACK* –** как PUBLIC, но для сегментов стека.
  + ***COMMON* –** все одноименные сегменты накладываются в памяти, начиная с одного адреса.
  + ***AT* *адреса* –** размещение с указанного сегментного адреса
  + ***PRIVATE* –** объединения не происходит
* **Класс** – определяется порядок наследования сегментов памяти. Задается в виде: ‘строка’. Строка – ключевое слово **(‘DATA’, ‘CODE’, ‘STACK’**) или произвольный идентификатор. Компоновщик группирует вместе все сегменты с одинаковым классом.
* **Размер** – определяет 16-ти или 32-битные сегменты (**USE16** (по умолчанию), **USE32**)

Директива **ASSUME** – устанавливает связь между сегментами и именами сегментов или групп, например ASSUME cs:cl, ds:dl, es:el, ss:sl

Упрощенные (точечные) директивы сегментации:

Ассемблер содержит ряд директив, учитывающих соглашения ЯВУ – стандартные модели памяти.

**MODEL** модель\_памяти [язык] [модификатор]

**Модели памяти:**

* **TINY:** код и данные в одном сегменте, объединены в группу *DGROUP (.com)*
* **SMALL:** код в одном сегменте, данные и стек в других данные объединены в группу *DGROUP*, адресация во всех сегментах – *near*
* **MEDIUM:** код в нескольких сегментах, по одному на каждый модуль, данные объединены в группу *DGROUP*. Адресация данных – *near*, вызовы процедур – *far*
* **COMPACT:** код в одном сегменте, данные в нескольких, для доступа к данным используется указатели дальнего типа
* **LARGE, HUGE:** и код, и данные в нескольких сегментах (HUGE позволяет использовать всю память для размещения программы)
* **FLAT** как **TINE,** но для 32-битовой адресации.

1. **Задание**

**Часть 1.** Написать программу на ассемблере, осуществляющую ввод последовательности символов с клавиатуры, обработку кодов символов в соответствии с заданием и вывод на экран результирующей последовательности

**Программа должна содержать:**

* ввод последовательности символов с клавиатуры, предваряемый соответствующим текстовым сообщением (в результате ввода формируется статический массив кодов символов, максимальное число вводимых символов выбирается самостоятельно);
* обработку кодов символов в соответствии с заданием, вариант задания, определяющий условия пересылки байтов или слов из входного массива в выходной, берется из лабораторной работы №1;
* вывод на экран сообщения о результате и результирующей последовательности символов.

**Программа должна быть реализована в 2-х вариантах:**

а) односегментная (.*com*); б) многосегментная (.*exe*).

**13 вариант:** переслать слова, в которых число сброшенных битов больше половины.

**Алгоритм**

НАЧАЛО

Ввести данные (массив)

Цикл ДЛЯ каждого элемента массива:

Подсчитать кол-во установленных бит (кол-во единиц в двоичном коде)

ЕСЛИ кол-во единиц ≧ 8:

ТО пропустить данный элемент

ИНАЧЕ переслать данный элемент

Всё – цикл

КОНЕЦ

Особенности:

Так как, согласно заданию, мы работаем со словами (2 байта = 16 бит) необходимо это учитывать при увеличении счетчиков регистров источника и приемника данных.

Для подсчета установленных бит используется операция **adc** при сложении с 0 (данная реализация является лишь одним из вариантов решения данной задачи). Данная команда выполняет сложение с учетом флага переноса.

1. **Односегментная программа(.com)**

**lab2com.asm**

;Лабораторная работа №2

;Выполнил студент группы 19-В-1

;Сапожников В.О.

;Односегментная программа

.model tiny

.code

org 100h

\_main:

print macro string ;вывод строки

push ax

push dx

mov dx,offset string

mov ah,9

int 21h

pop dx

pop ax

endm

input macro string ;ввод строки

push ax

push dx

mov ah,0ah

mov dx,offset string

int 21h

pop dx

pop ax

endm

;Начало программы

print hello

print enter\_please

input str\_in

mov si, offset str\_in + 1

mov di, offset str\_out

mov cl, [si]

ror cl, 1 ;работаем со словами -> счетчик в 2 раза меньше

inc si

\_loop: ;Внешний цикл на кол-во слов

mov ax, [si]

push cx ;прячем текущий счетчик в стэк

mov cx, 16 ;заводим счетчик на 16 бит = 2 байта = слово

xor bl, bl

\_inner\_loop: ;внутренний цикл на 16 бит

ror ax,1

adc bl,0

loop \_inner\_loop

cmp bx, 8 ;если единиц 8 или больше

je \_next\_val ;то не записываем слово и переходим к следующему

mov [di], ax ;иначе записываем слово

inc di

inc di

\_next\_val: ;переход к следующему слову

pop cx

inc si

inc si

loop \_loop

print result

print str\_out ;вывод полученного значения

int 21h

mov ax,4c00h

int 21h

hello db 10,13, 'Laboratoty work #2', 10, 13, '$'

enter\_please db 10,13,'Input: ', 10, 13, '$'

result db 10,13,'Output: ', 10,13,'$'

str\_in dw 20, (?), 0Ah dup ("?")

str\_out dw 20 dup (' '), '$'

end \_main

**lab2com.lst**

Turbo Assembler Version 2.51 11/22/21 14:07:29 Page 1

lab2com.asm

1 ;Лабораторная работа №2

2 ;Односегментная программа

3 0000 .model tiny

4 0000 .code

5 org 100h

6

7 0100 \_main:

8 print macro string ;вывод строки

9 push ax

10 push dx

11 mov dx,offset string

12 mov ah,9

13 int 21h

14 pop dx

15 pop ax

16 endm

17

18 input macro string ;ввод строки

19 push ax

20 push dx

21 mov ah,0ah

22 mov dx,offset string

23 int 21h

24 pop dx

25 pop ax

26 endm

27

28 ;Начало программы

29 print hello

1 30 0100 50 push ax

1 31 0101 52 push dx

1 32 0102 BA 0166r mov dx,offset hello

1 33 0105 B4 09 mov ah,9

1 34 0107 CD 21 int 21h

1 35 0109 5A pop dx

1 36 010A 58 pop ax

37 print enter\_please

1 38 010B 50 push ax

1 39 010C 52 push dx

1 40 010D BA 017Dr mov dx,offset enter\_please

1 41 0110 B4 09 mov ah,9

1 42 0112 CD 21 int 21h

1 43 0114 5A pop dx

1 44 0115 58 pop ax

45

46 input str\_in

1 47 0116 50 push ax

1 48 0117 52 push dx

1 49 0118 B4 0A mov ah,0ah

1 50 011A BA 0196r mov dx,offset str\_in

1 51 011D CD 21 int 21h

1 52 011F 5A pop dx

1 53 0120 58 pop ax

54

55 0121 BE 0197r mov si, offset str\_in + 1

56 0124 BF 01AEr mov di, offset str\_out

57

Turbo Assembler Version 2.51 11/22/21 14:07:29 Page 2

lab2com.asm

58 0127 8A 0C mov cl, [si]

59 0129 D0 C9 ror cl, 1 ;работаем со словами -> счетчик в 2 раза +

60 меньше

61 012B 46 inc si

62

63 012C \_loop: ;Внешний цикл на кол-во слов

64 012C 8B 04 mov ax, [si]

65 012E 51 push cx

66 012F B9 0010 mov cx, 16

67 0132 32 DB xor bl, bl

68

69 0134 \_inner\_loop: ;внутренний цикл на 16 бит

70 0134 D1 C8 ror ax,1

71 0136 80 D3 00 adc bl,0

72 0139 E2 F9 loop \_inner\_loop

73

74 013B 83 FB 08 cmp bx, 8

75 013E 74 04 je \_next\_

76

77

78 0140 89 05 mov [di], ax

79 0142 47 inc di

80 0143 47 inc di

81

82 0144 \_next\_val:

83 0144 59 pop cx

84 0145 46 inc si

85 0146 46 inc si

86 0147 E2 E3 loop \_loop

87

88 print result

1 89 0149 50 push ax

1 90 014A 52 push dx

1 91 014B BA 0189r mov dx,offset result

1 92 014E B4 09 mov ah,9

1 93 0150 CD 21 int 21h

1 94 0152 5A pop dx

1 95 0153 58 pop ax

96 print str\_out

1 97 0154 50 push ax

1 98 0155 52 push dx

1 99 0156 BA 01AEr mov dx,offset str\_out

1 100 0159 B4 09 mov ah,9

1 101 015B CD 21 int 21h

1 102 015D 5A pop dx

1 103 015E 58 pop ax

104

105 015F CD 21 int 21h

106 0161 B8 4C00 mov ax,4c00h

107 0164 CD 21 int 21h

108

109

110

111 0166 0A 0D 4C 61 62 6F 72+ hello db 10,13, 'Laboratoty work #2', 10, 13, '$'

112 61 74 6F 74 79 20 77+

113 6F 72 6B 20 23 32 0A+

114 0D 24

Turbo Assembler Version 2.51 11/22/21 14:07:29 Page 3

lab2com.asm

115 017D 0A 0D 49 6E 70 75 74+ enter\_please db 10,13,'Input: ', 10, 13, '$'

116 3A 20 0A 0D 24

117 0189 0A 0D 4F 75 74 70 75+ result db 10,13,'Output: ', 10,13,'$'

118 74 3A 20 0A 0D 24

119

120 0196 0014 0000 0A\*(003F) str\_in dw 20, (?), 0Ah dup ("?")

121 01AE 14\*(0020) 0024 str\_out dw 20 dup (' '), '$'

122 end \_main

Turbo Assembler Version 2.51 11/22/21 14:07:29 Page 4

Symbol Table

Symbol Name Type Value

??DATE Text "11/22/21"

??FILENAME Text "lab2com "

??TIME Text "14:07:29"

??VERSION Number 0205

@CODE Text DGROUP

@CODESIZE Text 0

@CPU Text 0101H

@CURSEG Text \_TEXT

@DATA Text DGROUP

@DATASIZE Text 0

@FILENAME Text LAB2COM

@MODEL Text 1

@WORDSIZE Text 2

ENTER\_PLEASE Byte DGROUP:017D

HELLO Byte DGROUP:0166

RESULT Byte DGROUP:0189

STR\_IN Word DGROUP:0196

STR\_OUT Word DGROUP:01AE

\_INNER\_LOOP Near DGROUP:0134

\_LOOP Near DGROUP:012C

\_MAIN Near DGROUP:0100

\_NEXT\_VAL Near DGROUP:0144

Macro Name

INPUT

PRINT

Groups & Segments Bit Size Align Combine Class

DGROUP Group

\_DATA 16 0000 Word Public DATA

\_TEXT 16 01D8 Word Public CODE

**Результат работы**

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. **Многосегментная программа(.exe)**

**lab2.asm**

d1 SEGMENT para public 'DATA'

hello db 10,13, 'Laboratoty work #2', 10, 13, '$'

enter\_please db 10,13,'Input: ', 10, 13, '$'

result db 10,13,'Output: ', 10,13,'$'

str\_in dw 20, (?), 0Ah dup ("?")

str\_out dw 20 dup (' '), '$'

d1 ENDS

e1 SEGMENT para public 'DATA'

print macro string ;вывод строки

push ax

push dx

mov dx,offset string

mov ah,9

int 21h

pop dx

pop ax

endm

input macro string ;ввод строки

push ax

push dx

mov ah,0ah

mov dx,offset string

int 21h

pop dx

pop ax

endm

e1 ENDS

s1 SEGMENT para 'CODE'

assume cs:s1, ds:d1, es:e1, ss:stack1

\_main:

mov ax,d1

mov ds, ax

mov ax, e1

mov es, ax

;Начало программы

print hello

print enter\_please

input str\_in

mov si, offset str\_in + 1

mov di, offset str\_out

mov cl, [si]

ror cl, 1 ;работаем со словами -> счетчик в 2 раза меньше

inc si

\_loop: ;Внешний цикл на кол-во слов

mov ax, [si]

push cx ;прячем текущий счетчик в стэк

mov cx, 16 ;заводим счетчик на 16 бит = 2 байта = слово

xor bl, bl

\_inner\_loop: ;внутренний цикл на 16 бит

ror ax,1

adc bl,0

loop \_inner\_loop

cmp bx, 8 ;если единиц 8 или больше

je \_next\_val ;то не записываем слово и переходим к следущему

mov [di], ax ;иначе записываем слово

inc di

inc di

\_next\_val: ;переход к следущему слову

pop cx

inc si

inc si

loop \_loop

print result

print str\_out ;вывод полученного значения

int 21h

mov ax,4c00h

int 21h

s1 ENDS

stack1 SEGMENT para stack 'stack'

dw 100 dup (?)

stack1 ends

END \_main

**lab2.lst**

Turbo Assembler Version 2.51 11/22/21 14:38:15 Page 1

lab2.asm

1 0000 d1 SEGMENT para public 'DATA'

2 0000 0A 0D 4C 61 62 6F 72+ hello db 10,13, 'Laboratoty work #2', 10, 13, '$'

3 61 74 6F 74 79 20 77+

4 6F 72 6B 20 23 32 0A+

5 0D 24

6 0017 0A 0D 49 6E 70 75 74+ enter\_please db 10,13,'Input: ', 10, 13, '$'

7 3A 20 0A 0D 24

8 0023 0A 0D 4F 75 74 70 75+ result db 10,13,'Output: ', 10,13,'$'

9 74 3A 20 0A 0D 24

10

11 0030 0014 0000 0A\*(003F) str\_in dw 20, (?), 0Ah dup ("?")

12 0048 14\*(0020) 0024 str\_out dw 20 dup (' '), '$'

13 0072 d1 ENDS

14

15 0000 e1 SEGMENT para public 'DATA'

16 print macro string

17 push ax

18 push dx

19 mov dx,offset string

20 mov ah,9

21 int 21h

22 pop dx

23 pop ax

24 endm

25

26 input macro string ;ввод строки

27 push ax

28 push dx

29 mov ah,0ah

30 mov dx,offset string

31 int 21h

32 pop dx

33 pop ax

34 endm

35 0000 e1 ENDS

36

37 0000 s1 SEGMENT para 'CODE'

38 assume cs:s1, ds:d1, es:e1, ss:stack1

39 0000 \_main:

40 0000 B8 0000s mov ax,d1

41 0003 8E D8 mov ds, ax

42

43 0005 B8 0000s mov ax, e1

44 0008 8E C0 mov es, ax

45

46 ;Начало программы

47 print hello

1 48 000A 50 push ax

1 49 000B 52 push dx

1 50 000C BA 0000r mov dx,offset hello

1 51 000F B4 09 mov ah,9

1 52 0011 CD 21 int 21h

1 53 0013 5A pop dx

1 54 0014 58 pop ax

55 print enter\_please

1 56 0015 50 push ax

1 57 0016 52 push dx

Turbo Assembler Version 2.51 11/22/21 14:38:15 Page 2

lab2.asm

1 58 0017 BA 0017r mov dx,offset enter\_please

1 59 001A B4 09 mov ah,9

1 60 001C CD 21 int 21h

1 61 001E 5A pop dx

1 62 001F 58 pop ax

63

64 input str\_in

1 65 0020 50 push ax

1 66 0021 52 push dx

1 67 0022 B4 0A mov ah,0ah

1 68 0024 BA 0030r mov dx,offset str\_in

1 69 0027 CD 21 int 21h

1 70 0029 5A pop dx

1 71 002A 58 pop ax

72

73 002B BE 0031r mov si, offset str\_in + 1

74 002E BF 0048r mov di, offset str\_out

75

76 0031 8A 0C mov cl, [si]

77 0033 D0 C9 ror cl, 1

78

79 0035 46 inc si

80

81 0036 \_loop:

82 0036 8B 04 mov ax, [si]

83 0038 51 push cx

84 0039 B9 0010 mov cx, 16

85 003C 32 DB xor bl, bl

86

87 003E \_inner\_loop: ;внутренний цикл на 16 бит

88 003E D1 C8 ror ax,1

89 0040 80 D3 00 adc bl,0

90 0043 E2 F9 loop \_inner\_loop

91

92 0045 83 FB 08 cmp bx, 8

93 0048 74 04 je \_next\_val

94

95

96 004A 89 05 mov [di],

97 004C 47 inc di

98 004D 47 inc di

99

100 004E \_next\_val:

101 004E 59 pop cx

102 004F 46 inc si

103 0050 46 inc si

104 0051 E2 E3 loop \_loop

105

106 print result

1 107 0053 50 push ax

1 108 0054 52 push dx

1 109 0055 BA 0023r mov dx,offset result

1 110 0058 B4 09 mov ah,9

1 111 005A CD 21 int 21h

1 112 005C 5A pop dx

1 113 005D 58 pop ax

114 print str\_out

Turbo Assembler Version 2.51 11/22/21 14:38:15 Page 3

lab2.asm

1 115 005E 50 push ax

1 116 005F 52 push dx

1 117 0060 BA 0048r mov dx,offset str\_out

1 118 0063 B4 09 mov ah,9

1 119 0065 CD 21 int 21h

1 120 0067 5A pop dx

1 121 0068 58 pop ax

122

123 0069 CD 21 int 21h

124 006B B8 4C00 mov ax,4c00h

125 006E CD 21 int 21h

126 0070 s1 ENDS

127

128 0000 stack1 SEGMENT para stack 'stack'

129 0000 64\*(????) dw 100 dup (?)

130 00C8 stack1 ends

131

132 END \_main

|  |  |  |
| --- | --- | --- |
|  |  |  |

Turbo Assembler Version 2.51 11/22/21 14:38:15 Page 4

Symbol Table

Symbol Name Type Value

??DATE Text "11/22/21"

??FILENAME Text "lab2 "

??TIME Text "14:38:15"

??VERSION Number 0205

@CPU Text 0101H

@CURSEG Text STACK1

@FILENAME Text LAB2

@WORDSIZE Text 2

ENTER\_PLEASE Byte D1:0017

HELLO Byte D1:0000

RESULT Byte D1:0023

STR\_IN Word D1:0030

STR\_OUT Word D1:0048

\_INNER\_LOOP Near S1:003E

\_LOOP Near S1:0036

\_MAIN Near S1:0000

\_NEXT\_VAL Near S1:004E

Macro Name

INPUT

PRINT

Groups & Segments Bit Size Align Combine Class

D1 16 0072 Para Public DATA

E1 16 0000 Para Public DATA

S1 16 0070 Para none CODE

STACK1 16 00C8 Para Stack STACK

**Результат работы**

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. **Программа вывода даты создания BIOS**

**Часть 2.**  *Написать программу на ассемблере, осуществляющую вывод на экран даты создания* ***BIOS***(Aф=0*FFFF*5*h*) *прямой записью в видеопамять* (Аф=*B*8000*h*)*. Использовать точечные директивы (модель памяти, директивы сегментации).*

**bios.asm**

.model small

.stack 100h

.data

bios equ 0FFFFh

video equ 0B800h

color equ 10111010b

.code

\_main:

mov ah,0

mov al,3

int 10h

mov ax,bios

mov es,ax

mov ax, video

mov ds,ax

mov si, 05h

mov di, 00h

mov cx, 0008h

\_loop:

mov al, es:[si]

mov ah,color

mov ds:[di], ax

inc di

inc di

inc si

loop \_loop

mov ah, 4ch

mov al, 0

int 21h

end \_main

**bios.lst**

Turbo Assembler Version 2.51 11/22/21 20:59:32 Page 1

bios.asm

1 0000 .model small

2 0000 .stack 100h

3 0000 .data

4 = FFFF bios equ 0FFFFh

5 = B800 video equ 0B800h

6 = 00BA color equ 10111010b

7

8 0000 .code

9 0000 \_main:

10 0000 B4 00 mov ah,0

11 0002 B0 03 mov al,3

12 0004 CD 10 int 10h

13 0006 B8 FFFF mov ax,bios

14 0009 8E C0 mov es,ax

15

16 000B B8 B800 mov ax, video

17 000E 8E D8 mov ds,ax

18

19 0010 BE 0005 mov si, 05h

20 0013 BF 0000 mov di, 00h

21 0016 B9 0008 mov cx, 0008h

22

23 0019 \_loop:

24 0019 26: 8A 04 mov al, es:[si]

25 001C B4 BA mov ah,color

26 001E 89 05 mov ds:[di], ax

27 0020 47 inc di

28 0021 47 inc di

29 0022 46 inc si

30 0023 E2 F4 loop \_loop

31

32 0025 B4 4C mov ah, 4ch

33 0027 B0 00 mov al, 0

34 0029 CD 21 int 21h

35 end \_main

Turbo Assembler Version 2.51 11/22/21 20:59:32 Page 2

Symbol Table

Symbol Name Type Value

??DATE Text "11/22/21"

??FILENAME Text "bios"

??TIME Text "20:59:32"

??VERSION Number 0205

@CODE Text \_TEXT

@CODESIZE Text 0

@CPU Text 0101H

@CURSEG Text \_TEXT

@DATA Text DGROUP

@DATASIZE Text 0

@FILENAME Text BIOS

@MODEL Text 2

@WORDSIZE Text 2

BIOS Number FFFF

COLOR Number 00BA

VIDEO Number B800

\_LOOP Near \_TEXT:0019

\_MAIN Near \_TEXT:0000

Groups & Segments Bit Size Align Combine Class

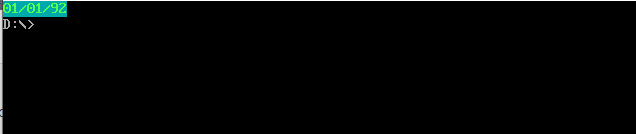
DGROUP Group

STACK 16 0100 Para Stack STACK

\_DATA 16 0000 Word Public DATA

\_TEXT 16 002B Word Public CODE

**Результат работы**



1. **Map файлы для различных параметров выравнивания**

**lab2byte.map**

Start Stop Length Name Class

00000H 00071H 00072H D1 DATA

00072H 00072H 00000H E1 DATA

00072H 000E1H 00070H S1 CODE

000E2H 001A9H 000C8H STACK1 STACK

Program entry point at 0007:0002

При выравнивании типа **byte** сегмент начинается с любого байта.

**lab2word.map**

Start Stop Length Name Class

00000H 00071H 00072H D1 DATA

00072H 00072H 00000H E1 DATA

00072H 000E1H 00070H S1 CODE

000E2H 001A9H 000C8H STACK1 STACK

Program entry point at 0007:0002

При выравнивании типа **word** сегмент начинается с четного байта.

**lab2para.map**

Start Stop Length Name Class

00000H 00071H 00072H D1 DATA

00080H 00080H 00000H E1 DATA

00080H 000EFH 00070H S1 CODE

000F0H 001B7H 000C8H STACK1 STACK

Program entry point at 0008:0000

При выравнивании типа **word** сегмент начинается с адреса краткого 16.

**lab2page.map**

Start Stop Length Name Class

00000H 00071H 00072H D1 DATA

00100H 00100H 00000H E1 DATA

00100H 0016FH 00070H S1 CODE

00200H 002C7H 000C8H STACK1 STACK

Program entry point at 0010:0000

При выравнивании типа **page** сегмент начинается с адреса краткого 256.

**Вывод:** ходе данной работы юли приобретены навыки разработки одно- и многосегметных программ на ЯП ассемблер, изучены функции прерывания для организации ввода-вывода и управление трансляцией и компоновкой.