

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Архитектура компьютера

Студент: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1	Цель работы	2
2	Задание.....	2
3	Теоретическое введение	2
4	Выполнение лабораторной работы.....	3
4.1	Реализация переходов в NASM	3
4.2	Изучение структуры файла листинга.....	5
4.3	Задания для самостоятельной работы	8
5	Выводы	14
	Список литературы	14

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 1).

```
aryusupova@aryusupova-VirtualBox:~$ mkdir ~/work/arch-pc/lab07
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/arch-pc/lab07
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 1: Создание каталога и файла lab7-1.asm в нём

Перехожу в файл lab7-1.asm и ввожу текст программы из листинга 7.1 (рис. 2).

Файл Правка Вид Поиск Сервис Документы Справка

*lab7-1.asm (~/work/arch-pc/lab07)

```
%include 'in.out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
    jmp _label2
_label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 1'
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'
_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 3'
_end:
    call quit ; вызов подпрограммы завершения
```

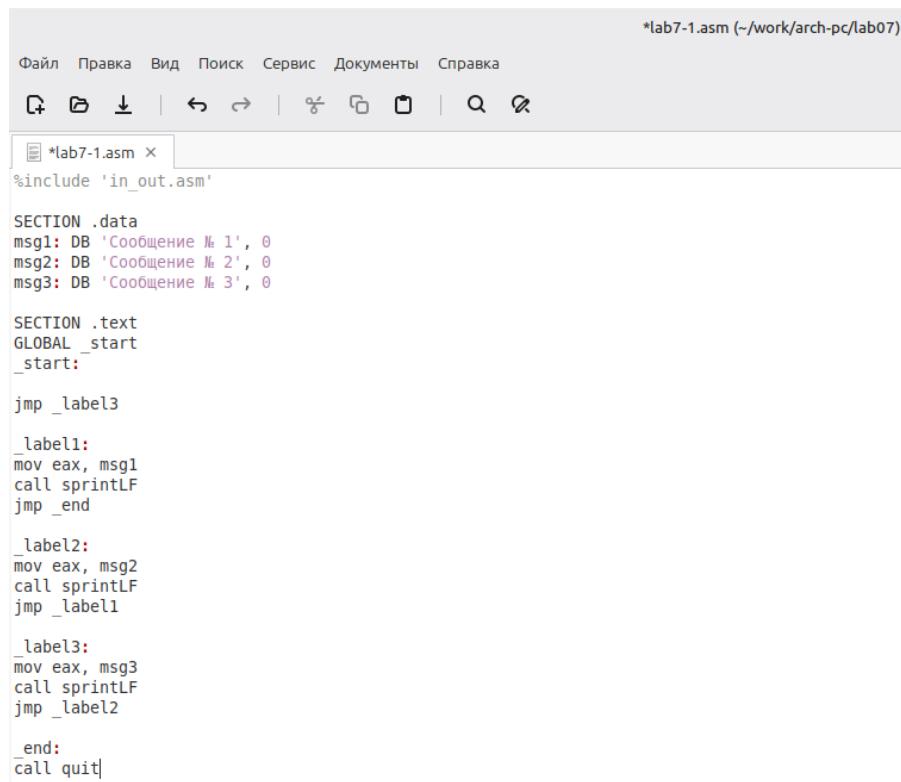
Рисунок 2: Ввод кода программы из листинга 7.1

Создала исполняемый файл и запустила его. Результат работы данной программы совпадает с указанным в задании. При запуске программы я убедилась в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рисунок 3: Запуск программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляю инструкцию jmp с меткой _label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой _end (т.е. переход к инструкции call quit). Изменяю текст программы в соответствии с листингом 7.2 (рис. 4).



The screenshot shows a Windows Notepad window with the title bar reading '*lab7-1.asm (~/work/arch-pc/lab07)'. The menu bar includes 'Файл', 'Правка', 'Вид', 'Поиск', 'Сервис', 'Документы', and 'Справка'. Below the menu is a toolbar with icons for file operations. The main text area contains the following assembly code:

```
%include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1', 0
msg2: DB 'Сообщение № 2', 0
msg3: DB 'Сообщение № 3', 0

SECTION .text
GLOBAL _start
_start:
    jmp _label3

    _label1:
    mov eax, msg1
    call sprintLF
    jmp _end

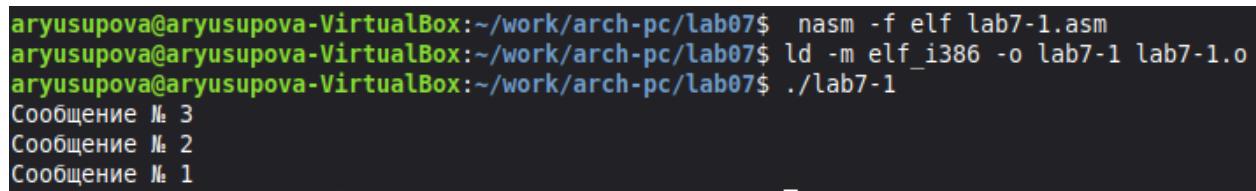
    _label2:
    mov eax, msg2
    call sprintLF
    jmp _label1

    _label3:
    mov eax, msg3
    call sprintLF
    jmp _label2

    _end:
    call quit
```

Рисунок 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны. Вывод программы соответствует (рис. 5).



```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рисунок 5: Запуск изменённой программы

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 с помощью команды touch (рис. 6). И пишу код из листинга 7.3 (рис. 7). Программа выводит значение переменной с максимальным значением, проверяю работу программы, вводя разные значения (рис. 8).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.asm  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 6: Создание файла lab7-2.asm

The screenshot shows a text editor window titled "lab7-2.asm (~/work/arch-pc/lab07)". The code is as follows:

```
%include 'in_out.asm'  
section .data  
msg1 db 'Введите B: ',0h  
msg2 db "Наибольшее число: ",0h  
A dd '20'  
C dd '50'  
section .bss  
max resb 10  
B resb 10  
section .text  
global _start  
.start:  
; ----- Вывод сообщения 'Введите B: '  
mov eax,msg1  
call sprint  
; ----- Ввод 'B'  
mov ecx,B  
mov edx,10  
call sread  
; ----- Преобразование 'B' из символа в число  
mov eax,B  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [B],eax ; запись преобразованного числа в 'B'  
; ----- Записываем 'A' в переменную 'max'  
mov ecx,[A] ; 'ecx = A'  
mov [max],ecx ; 'max = A'  
; ----- Сравниваем 'A' и 'C' (как символы)  
cmp ecx,[C] ; Сравниваем 'A' и 'C'  
jg check_B ; если 'A>C', то переход на метку 'check_B',  
mov ecx,[C] ; иначе 'ecx = C'  
mov [max],ecx ; 'max = C'  
; ----- Преобразование 'max(A,C)' из символа в число  
check_B:
```

Рисунок 7: Ввод кода программы из листинга 7.3

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2  
Введите B: 20  
Наибольшее число: 50  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2  
Введите B: 23  
Наибольшее число: 50  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2  
Введите B: 66  
Наибольшее число: 66
```

Рисунок 8: Проверка программы на нахождения максимума из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mcedit (рис. 9).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm  
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рисунок 9: Создание файла и открытие его

Проверяю файл листинга (рис.10).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07
/home/aryusupova/work/arch-pc/lab07/lab7-2.lst [---] 0 L[ 1+ 0 1/226 ] *(0 /14499b) 0032 0x020
1           %include 'in_out.asm'
1          ;----- slen -----
2          ; Функция вычисления длины сообщения
3          ;----- sten:
4 00000000 53
5 00000001 89C3
6
7          ;----- nextchar:
8 00000003 803800
9 00000006 7403
10 00000008 40
11 00000009 EBF8
12
13          ;----- finished:
14 0000000B 29D8
15 0000000D 5B
16 0000000E C3
17
18          ;----- sprint -----
19          ; Функция печати сообщения
20          ; входные данные: mov eax,<message>
21          ;----- sprint:
22
23 0000000F 52
24 00000010 51
25 00000011 53
26 00000012 50
27 00000013 E8E8FFFF
28
29 00000018 89C2
30 0000001A 58
31
32 0000001B 89C1
33 0000001D BB01000000
34 00000022 B804000000
35 00000027 CD80
36
37 00000029 5B
1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Копия 6 Переместить 7 Поиск 8 Удалить 9 ЦенюМС 10 Выход
```

Рисунок 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

The screenshot shows a text editor window with the title bar "*lab7-2.asm (~/work/arch-pc/lab07)". The menu bar includes "Файл", "Правка", "Вид", "Поиск", "Сервис", "Документы", and "Справка". Below the menu is a toolbar with icons for file operations like Open, Save, Print, and Search. The main text area contains the following assembly code:

```
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

check_B:
mov eax,|
call atoi
mov [max],eax ; запись преобразованного числа в `max`

mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin'
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx

fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рисунок 11: Удаление операнды в программе

Выполнила трансляцию с получением файла листинга (рис.12).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
```

Рисунок 12: Трансляция вывода полученного файла

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 13).

```

aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab07
/home/aryusupova/work/arch-pc/lab07/lab7-2.lst [---] 0 L:[190+37 227/227] *(13696/13696b) <EOF>
 15 000000ED E81DFFFFFF      call sprint
 16 . . .
 17 000000F2 B9[0A000000]    mov ecx,B
 18 000000F7 BA0A000000     mov edx,10
 19 000000FC E842FFFFFF     call sread
 20 . . .
 21 00000101 B8[0A000000]    mov eax,B
 22 00000106 E891FFFFFF     call atoi.
 23 0000010B A3[0A000000]    mov [B],eax.
 24 . . .
 25 00000110 8B0D[35000000]  mov ecx,[A].
 26 00000116 890D[00000000]  mov [max],ecx.
 27 . . .
 28 0000011C 3B0D[39000000]  cmp ecx,[C].
 29 00000122 7F0C           jg check_B
 30 00000124 8B0D[39000000]  mov ecx,[C].
 31 0000012A 890D[00000000]  mov [max],ecx
 32 . . .
 33 . . .
 34 *****          error: invalid combination of opcode and operands
 35 00000130 E867FFFFFF     call atoi
 36 00000135 A3[00000000]    mov [max],eax ; запись преобразованного числа в `max`
 37 . . .
 38 0000013A 8B0D[00000000]  mov ecx,[max]
 39 00000140 3B0D[0A000000]  cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
 40 00000146 7F0C           jg fin ; если 'max(A,C)>B', то переход на 'fin',
 41 00000148 8B0D[0A000000]  mov ecx,[B] ; иначе 'ecx = B'
 42 0000014E 890D[00000000]  mov [max],ecx
 43 . . .
 44 . . .
 45 00000154 B8[13000000]    fin:
 46 00000159 E8B1FEFFFF     mov eax, msg2
 47 0000015E A1[00000000]    call sprint
 48 00000163 E81EFFFFFF     mov eax,[max]
 49 00000168 E86EFFFFFF     call iprintfLF ; Вывод 'max(A,B,C)'
 50 . . .

```

Рисунок 13: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

1 задание:

Создаю необходимый файл lab2-3.asm в каталоге ~/work/arch-pc/lab07 (рис. 14).

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-3.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ 

```

Рисунок 14: Создание файла lab7-3.asm

Открываю созданный файл и ввожу код для нахождения наименьшего значения из трёх переменных a,b,c
(рис. 15).

The screenshot shows a Windows Notepad window with the title bar "lab7-3.asm (~/work/arch-pc/lab07)". The menu bar includes "Файл", "Правка", "Вид", "Поиск", "Сервис", "Документы", and "Справка". Below the menu is a toolbar with icons for file operations. The main text area contains assembly code:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите В: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:
    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10
    call sread

    mov eax, B
    call atoi
    mov [B], eax

    mov ecx, [A]
    mov [min], ecx

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
```

Рисунок 15: Ввод кода программы файла lab7-3.asm

Код программы 1 задание:

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной а: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
```

```
mov eax, x
```

```
call atoi
```

```
mov edi, eax
```

```
mov eax, msg_a
```

```
call sprint
```

```
mov ecx, a
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, a
```

```
call atoi
```

```
mov esi, eax
```

```
cmp edi, esi
```

```
jle add_values
```

```
mov eax, esi
```

```
jmp print_result
```

```
add_values:
```

```
mov eax, edi
```

```
add eax, esi
```

```
print_result:
```

```
mov edi, eax
```

```
mov eax, res
```

```
call sprint
```

```
mov eax, edi
```

```
call iprintLF
```

```
call quit
```

Транслирую, компоную файл и проверяю его работу, вводя значения из таблицы вариант №10. Программа работает корректно: выводит наименьшее

значение (рис. 16).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите В: 41
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите В: 62
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите В: 35
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 16: Трансляция, компоновка файла lab7-3.asm и проверка его работы

2 задание:

Создаю необходимый файл lab7-4.asm в каталоге ~/work/arch-pc/lab07 (рис. 17).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ █
```

Рисунок 17: Создание файла lab7-4.asm

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту (вариант №9) для введенных с клавиатурых переменных а и х (рис. 18).

The screenshot shows a Windows Notepad window with the file name *lab7-4.asm (~/work/arch-pc/lab07) at the top. The menu bar includes Файл, Правка, Вид, Поиск, Сервис, Документы, and Справка. Below the menu is a toolbar with standard icons for opening, saving, and navigating files. The main text area contains the following assembly code:

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    mov edi, eax

    mov eax, msg_a
    call sprint
    mov ecx, a
    mov edx, 80
    call sread
    mov eax, a
    call atoi
    mov esi, eax

    cmp edi, esi
    jle add_values
    mov eax, esi
    jmp print_result
```

Рисунок 18: Ввод кода программы lab7-4.asm

Код программы 2 задание :

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
```

```
mov eax, msg_x  
call sprint  
mov ecx, x  
mov edx, 80  
call sread  
mov eax, x  
call atoi  
mov edi, eax
```

```
mov eax, msg_a  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax, a  
call atoi  
mov esi, eax
```

```
cmp edi, esi  
jle add_values  
mov eax, esi  
jmp print_result
```

```
add_values:  
mov eax, edi  
add eax, esi
```

```
print_result:  
mov edi, eax  
mov eax, res  
call sprint  
mov eax, edi
```

```
call iprintLF
```

```
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. 19). В моей случае (5,7) и (6,4). Программа работает корректно.

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
```

Рисунок 19: Трансляция, компоновка и запуск программы lab7-4

5 Выводы

В процессе выполнения лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с их использованием. Кроме того, я ознакомилась с назначением и структурой файлов листинга.

Список литературы

1. <https://esystem.rudn.ru/mod/resource/view.php?id=1030555>
2. <https://www.opennet.ru/docs/RUS/nasm/>
3. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.