

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Хоюгбан Ганчыыр Анатольевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	19
4	Выводы	21

Список иллюстраций

2.1	Создание файла lab8-1.asm	6
2.2	Текст программы файла lab8-1.asm	7
2.3	Исполнение файла lab8-1.asm	8
2.4	Создание файла lab8-2.asm	8
2.5	Текст программы lab8-2.asm	9
2.6	Исполнение программы файла lab8-2.asm	10
2.7	Создание файла lab8-3.asm	10
2.8	Текст программы файла lab8-3.asm	11
2.9	Исполнение файла lab8-3.asm	12
2.10	Создание файла lab8-4.asm	12
2.11	Текст программы lab8-4.asm	13
2.12	Исполнение программы файла lab8-4.asm	14
2.13	Создание программы файла lab8-5.asm	14
2.14	Текст программы файла lab8-5.asm	15
2.15	Исполнение программы файла lab8-5.asm	15
2.16	Создание файла lab8-6.asm	16
2.17	Текст программы lab8-6.asm	17
2.18	Исполнение файла lab8-6.asm	18
3.1	Создание файла lab8-7.asm	19
3.2	Текст программы файла lab8-7.asm	20

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки

2 Выполнение лабораторной работы

Для начала я перешел в терминал, перешел на каталог arch-pc. Создал каталог lab08 для выполнения лабораторной работы, в котором создал файл lab8-1.asm(рис. 2.1)

```
gakhoyugban@dk4n65 ~ $ cd work
gakhoyugban@dk4n65 ~/work $ ls
arch-pc  study
gakhoyugban@dk4n65 ~/work $ cd arch-pc
gakhoyugban@dk4n65 ~/work/arch-pc $ ls
lab05  lab06  lab07
gakhoyugban@dk4n65 ~/work/arch-pc $ mkdir lab08
gakhoyugban@dk4n65 ~/work/arch-pc $ ls
lab05  lab06  lab07  lab08
gakhoyugban@dk4n65 ~/work/arch-pc $ cd lab08
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ touch lab8-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $
```

Рис. 2.1: Создание файла lab8-1.asm

Я написал текст программы для вывода значений регистра есх, что демонстрирую вам на рисунке(рис. 2.2)

```

;-----
; Программа вывода значений регистра 'ecx'
;-----
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ---- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ---- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ---- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit

```

Рис. 2.2: Текст программы файла lab8-1.asm

Перевел файл lab8-1.asm в объектный, сделал компоновку и отправил на исполнение. В итоге вместо n я ввел число 10, на что мне программа выдала последовательность чисел от 10 до 1 включительно. Чтобы окончательно проверить программу я ввел вместо n 5, что дало мне убедиться, что программа работает верно (рис. 2.3)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1

```

Рис. 2.3: Исполнение файла lab8-1.asm

Затем по ходу лабораторной работы от меня требуется изменить текст программы файла lab8-1.asm. Для чего я и создаю файл lab8-2.asm, что я вам показываю на рисунке(рис. 2.4)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ touch lab8-2.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ls
lab8-1.asm  lab8-2.asm

```

Рис. 2.4: Создание файла lab8-2.asm

Я изменил текст программы, использовав регистр esx в теле цикла loop. А сам текст программы на рисунке(рис. 2.5)


```

;-----
; Программа вывода значений регистра 'ecx'
;-----
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
sub ecx,1 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit

```

Рис. 2.5: Текст программы lab8-2.asm

Перевел файл lab8-2.asm в объектный, сделал компоновку и отправил на исполнение. Сначала я ввел число 5, что выдало мне что-то непонятное. А затем я ввел число 10 и программа вывела последовательно все числа от 10 до 1, которые являются нечетными (рис. 2.6)

```

4294955984
4294955982
4294955980
4294955978
4294955976
4294955974
4294955972
4294955970
4294955968
4294955966
4294955964
4294955962
4294955960
4294955958
4294955956
4294955954
4294955952
4294955950
4294955948
4294955946
4294955944
4294955942
4294955940
4294955938
4294955936
4294955934
4294955932
4294955930
4294955928
4294955926
4294955924
4294955922
4294955920^Z
[2]+  Остановлен      ./lab8-2
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ loop labelloop labelloop labelloop label
bash: loop: команда не найдена
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-2
Введите N: 10
9
7
5
3
1
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $

```

Рис. 2.6: Исполнение программы файла lab8-2.asm

Затем по ходу лабораторной работы от меня требуется изменить снова текст программы файла lab8-2.asm. Для чего я и создаю файл lab8-3.asm, что я вам показываю гна рисунке(рис. 2.7)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ touch lab8-3.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $

```

Рис. 2.7: Создание файла lab8-3.asm

Я изменил текст прграммы, добавив команды push и pop для сохранения значения счетчика цикла loop. А сам текст программы на рисунке(рис. 2.8)

```

;-----
; Программа вывода значений регистра 'ecx'
;-----
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ---- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ---- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ---- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
call quit

```

Рис. 2.8: Текст программы файла lab8-3.asm

Перевел файл lab8-3.asm в объектный, сделал компоновку и отправил на исполнение. Сначала, введя число 10 в n, я могу сделать вывод, что числа идут от 10 до 1 но не включительно 10. А затем, введя число 5 вместо n, я удостоверился в работе программы(рис. 2.9)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-3
Введите N: 10
9
8
7
6
5
4
3
2
1
0
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-3
Введите N: 5
4
3
2
1
0

```

Рис. 2.9: Исполнение файла lab8-3.asm

Затем мне требуется написать еще одну программу, для чего я создаю файл lab8-4.asm(рис. 2.10)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ touch lab8-4.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3 lab8-3.asm lab8-3.o lab8-4.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $

```

Рис. 2.10: Создание файла lab8-4.asm

По ходу лабораторной программы надо переписать текст программы выводящую на экран аргументной, а затем мне требуется написать еще одну программу, для чего я создаю файл lab8-4.asm командной строки, что я вам показываю на рисунке(рис. 2.11)

```

;-----
; Обработка аргументов командной строки
;-----
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в 'ecx' количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в 'edx' имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
call quit

```

Рис. 2.11: Текст программы lab8-4.asm

Перевел файл lab8-4.asm в объектный, сделал компоновку и отправил на исполнение. Я ввел три аргумента, как и требовалось по лабораторной работы, что мне выдало ровно 5 аргументов (рис. 2.12)


```

#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в 'ecx' количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в 'edx' имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем 'esi' для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент 'esi=esi+eax'
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр 'eax'
call iprintLF ; печать результата
call quit ; завершение программы

```

Рис. 2.14: Текст программы файла lab8-5.asm

Перевел файл lab8-5.asm в объектный, сделал компоновку и отправил на исполнение. Сначала, введя несколько аргументов, которые требовалось ввести по ходу лабораторной работы, я удостоверился, что программа работает верно. Затем, введя числа от 1 до 5, я уже окончательно понял, что программа работает верно, так как ответ программы был 15, а $1+2+3+4+5=15$ (рис. 2.15)

```

gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-5.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-5 lab8-5.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-5 12 13 7 10 5
Результат: 47
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-5 1 2 3 4 5
Результат: 15
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $

```

Рис. 2.15: Исполнение программы файла lab8-5.asm

Мне требовалось решить последний пункт лабораторной работы, для чего я и создал файл lab8-6.asm (рис. 2.16)

```

jakhoyuban@b4n4s: ~/work/arch-pc/lab8 $ touch lab8-6.asm
jakhoyuban@b4n4s: ~/work/arch-pc/lab8 $ ls
in_out.asm lab8-1 lab8-1.o lab8-2 lab8-2.o lab8-3 lab8-3.o lab8-4 lab8-4.o lab8-5 lab8-5.o lab8-6.asm
jakhoyuban@b4n4s: ~/work/arch-pc/lab8 $

```

Рис. 2.16: Создание файла lab8-6.asm

Мне требовалось изменить текст программы lab8-5.asm, чтобы программа не складывала все введенные аргументы, а перемножала. Я изменил текст программы для выполнения этой задачи, что демонстрирую на рисунке(рис. 2.17)


```

#include "in_out.asm"
SECTION .data
msg db 'результат: '
SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi,1

next:
cmp ecx,0
jz _end

pop eax
call atoi
mul esi
mov esi, eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

```

Рис. 2.17: Текст программы lab8-6.asm

Перевел файл lab8-6.asm в объектный, сделал компоновку и отправил на исполнение. Чтобы удостовериться в решении моей программы, я ввел числа от 1 до 5 включительно. В результате, я получил верное решение, так как $12345 = 120$ (рис. 2.18)

```
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ nasm -f elf lab8-6.asm
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-6 lab8-6.o
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ ./lab8-6 1 2 3 4 5
результат: 120
gakhoyugban@dk4n65 ~/work/arch-pc/lab08 $ mc
```

Рис. 2.18: Исполнение файла lab8-6.asm

3 Выполнение самостоятельной работы

Для решения самостоятельной работы я создам отдельный файл lab8-7.asm(рис. 3.1)

```
lab8@yugbank64n05: ~/work/arch-pr/lab8 $ touch lab8-7.asm
lab8@yugbank64n05: ~/work/arch-pr/lab8 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3 lab8-3.asm lab8-3.o lab8-4 lab8-4.asm lab8-4.o lab8-5 lab8-5.asm lab8-5.o lab8-6 lab8-6.asm lab8-6.o lab8-7.asm
lab8@yugbank64n05: ~/work/arch-pr/lab8 $
```

Рис. 3.1: Создание файла lab8-7.asm

Я напишу текст программы для решения, что я демонстрирую вам на рисунке(рис. 3.2)

```

%include 'in_out.asm'

SECTION .data
f_x db "функция: 10(x - 1)",0h
msg db 10,13,'результат: ',0h

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
dec eax
mov ebx, 10
mul ebx
add esi, eax

loop next

_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintLF

call quit

```

Рис. 3.2: Текст программы файла lab8-7.asm

4 Выводы

Были получены знания по организации циклов и работе стеков на языке NASM