Отчет по выполнению лабораторной работы №8

Дисциплина: архитектура компьютеров

Чернятьева Олеся Олеговна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm (рис. [3.1]).

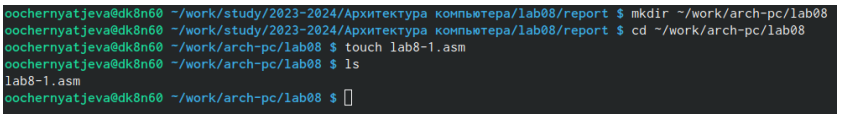

Рис. 3.1: Создание каталога и файла

Ввожу в файл lab8-1.asm текст программы из листинга 8.1.(рис. [3.2]).

```
;-------------------------------------------------------------------
; 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. 3.2: Ввод программы из листинга 8.1

Созаю исполняемый файл и проверяю его работу.(рис. [3.3]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.3: Проверка работы файла

Далее изменяю текст программы, добавив изменение значение регистра ecx в цикле.(рис. [3.4]).

```
24 mov ecx,[N] ; Счетчик цикла, `ecx=N`
25 label:
26 sub ecx,1 ; `ecx=ecx-1`
27 mov [N],ecx
28 mov eax,[N]
29 call iprintLF
30 loop label
31 call quit
```

Рис. 3.4: Изменение файла

Создаю исполняемый файл и проверяю его работу.(рис. [3.5]).

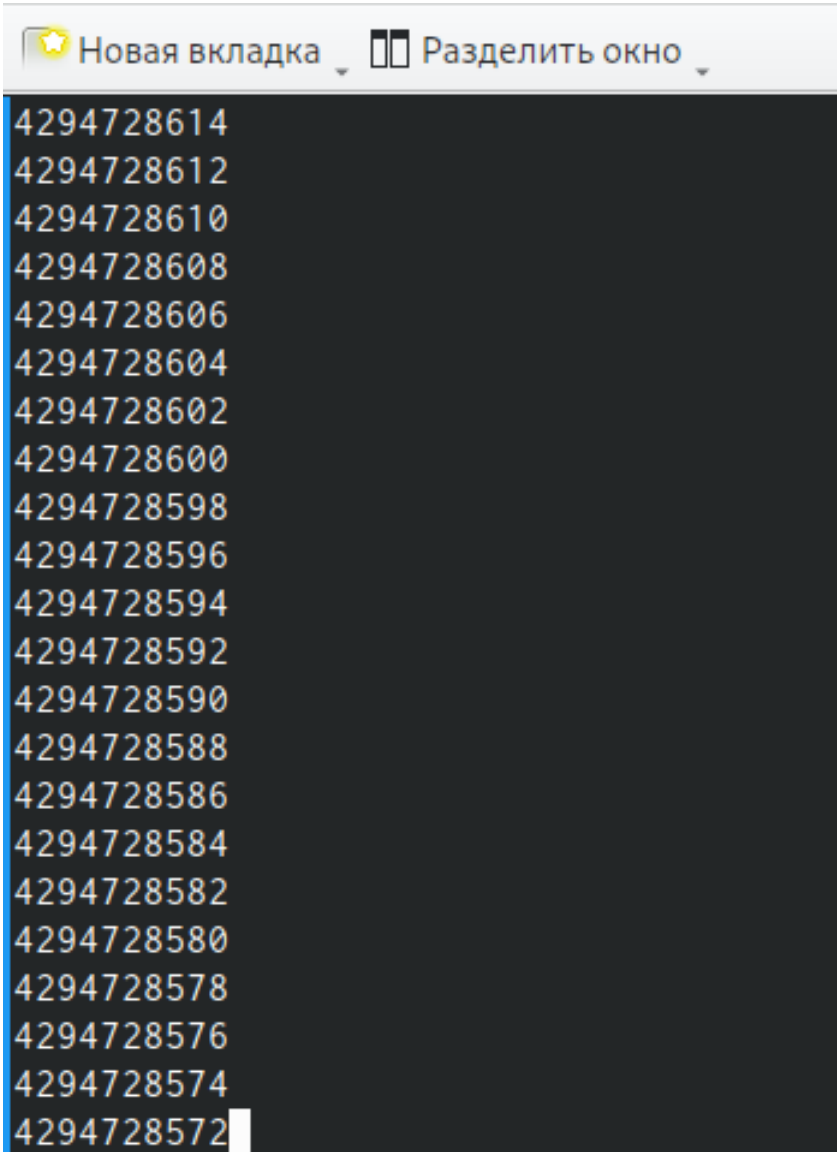

Рис. 3.5: Проверка работы файла

В этой программе число проходов цикла не соответствует значению N.

Вношу изменения в текст программы, добавив команды push и pop.(рис. [3.6]).

```
;                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. 3.6: Внесение изменений в текст программы

Создаю исполняемый файл и запускаю его.(рис. [3.7]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.7: Запуск программы

В данном случае число проходов цикла соответствует значению N.

3.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm.(рис. [3.8]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ touch lab8-2.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.8: Создание файла lab8-2.asm

Ввожу в него программу из листинга 8.2.(рис. [3.9]).

```
%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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 3.9: Ввод программы из листинга 8.2

Создаю исполняемый файл и запускаю его, указав аргументы “12”, “39” и “1”.(рис. [3.10]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-2 12 39 1
12
39
1
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.10: Запуск программы

Программой было обработано 3 аргумента.

Создаю файл lab8-3.asm и ввожу в него программу из листинга 8.3.(рис. [3.11]).

```
%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. 3.11: Ввод программы из листинга 8.3

Созадю исполняемый файл и запускаю его, указав аргументы.(рис. [3.2]).

```
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-3
Результат: 0
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4
Результат: 10
oochernyatjeva@dk8n80 ~/work/arch-pc/lab08 $
```

Рис. 3.12: Проверка работы файла

Программа работает.

Теперь изменяю текст программы из листинга 8.3 так, чтобы он вычислял произведение агрументов каждой строки.(рис. [3.13]).

```
%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, 1 ; 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
mul esi
mov 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. 3.13: Изменение текста листинга 8.3

Создаю исполняемый файл и запускаю его, указав аргументы.(рис. [3.14]).

```
oochernyatjeva@dk4n62 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5
Результат: 120
oochernyatjeva@dk4n62 ~/work/arch-pc/lab08 $
```

Рис. 3.14: Проверка работы программы

#Задания для самостоятельной работы

Создаю файл lab8-4.asm и начинаю написание программы, которая находит сумму значений функии f(x) для своео варианта(вариант 16).(рис. [3.15]).

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: f(x)=30x-11"
SECTION .text
global _start
_start:
mov eax,msg2
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0
mov edi, 30
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul edi
sub eax,11
add esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 3.15: Написание программы для самостоятельной работы

Создаю исполняемый файл и запускаю его, указав агрументы.(рис. [3.16]).

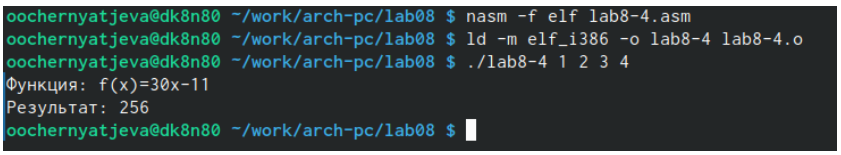

Рис. 3.16: Запуск программы

Программы работает корректно.

Текст прогораммы:

```
%include ‘in_out.asm’
SECTION .data
msg db “Результат:”,0
msg2 db “Функция: f(x)=30x-11”
SECTION .text
global _start
_start:
mov eax,msg2
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0
mov edi, 30
next:
cmp ecx,0h
jz _end
pop eax
call atoi
mul edi
sub eax,11
add esi,eax
```

```
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

4 Выводы

После выполнения данной лабораторной работы я научилась написанию программ с исползованием циклов и обработкой аргументов командной строки.