

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Краснова Камилла Геннадьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	11
4.3	Выполнение заданий для самостоятельной работы	13
5	Выводы	17
6	Список литературы	18

Список иллюстраций

4.1	Создание директории	8
4.2	Редактирование файла	9
4.3	Запуск исполняемого файла	9
4.4	Редактирование файла	10
4.5	Запуск исполняемого файла	10
4.6	Редактирование файла	11
4.7	Запуск исполняемого файла	11
4.8	Создание файла	12
4.9	Запуск исполняемого файла	12
4.10	Создание файла	12
4.11	Создание файла	13
4.12	Редактирование файла	13
4.13	Запуск исполняемого файла	13
4.14	Создание файла	14
4.15	Редактирование файла	14
4.16	Запуск исполняемого файла	14

Список таблиц

1 Цель работы

Цель данной лабораторной работы - приобретение навыков написания программ с использованием циклов обработки аргументов командной строки.

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий для самостоятельной работы

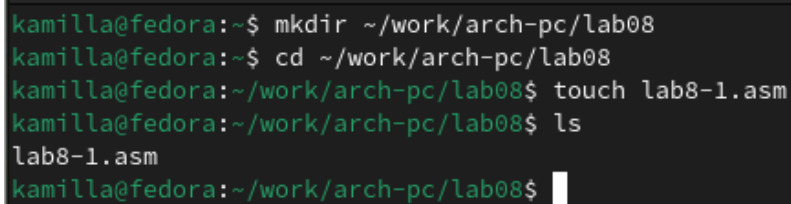
3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop). Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек. Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №8. Перехожу в созданный каталог с помощью утилиты `cd` и с помощью утилиты `touch` создаю файл `lab8-1.asm`. (рис. 4.1).



```
kamilla@fedora:~$ mkdir ~/work/arch-pc/lab08
kamilla@fedora:~$ cd ~/work/arch-pc/lab08
kamilla@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
kamilla@fedora:~/work/arch-pc/lab08$ ls
lab8-1.asm
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание директории

Открываю созданный файл `lab8-1.asm`, вставляю в него программу, которая выводит значение регистра `ecx`. (рис. 4.2).


```
kamilla@fedora:~/work/arch-pc/lab08$ gedit lab8-1.asm
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите N: ',0h
5
6 SECTION .bss
7 N: resb 10
8
9 SECTION .text
10 global _start
11 _start:
12
13 ; ----- Вывод сообщения 'Введите N: '
14 mov eax,msg1
15 call sprint
16
17 ; ----- Ввод 'N'
18 mov ecx, N
```

Рис. 4.2: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.3).

```
kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.3: Запуск исполняемого файла

Изменяю текст программы (рис. 4.4).

```

20 call sread
21
22 ; ----- Преобразование 'N' из символа в числс
23 mov eax,N
24 call atoi
25 mov [N],eax
26
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, `ecx=N`
29 label:
30 sub ecx,1 ; `ecx=ecx-1`
31 mov [N],ecx
32 mov eax,[N]
33 call iprintLF
34 loop label

```

Рис. 4.4: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.5). Количество итераций уменьшается вдвое, так как регистр ecx на каждой итерации уменьшается на 2 значения.

```

kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
kamilla@fedora:~/work/arch-pc/lab08$

```

Рис. 4.5: Запуск исполняемого файла

Изменяю текст программы, добавив команды push и pop (рис. 4.6).

```

25 mov [N],eax
26
27 ; ----- Организация цикла
28 mov ecx,[N] ; Счетчик цикла, `ecx=N`
29 label:
30 push ecx ; добавление значения ecx в стек
31 sub ecx,1
32 mov [N],ecx
33 mov eax,[N]
34 call iprintLF
35 pop ecx
36 loop label
37

```

Рис. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.7). Количество проходов цикла по значению N совпадает, но происходит смещение на -1.

```

kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
kamilla@fedora:~/work/arch-pc/lab08$

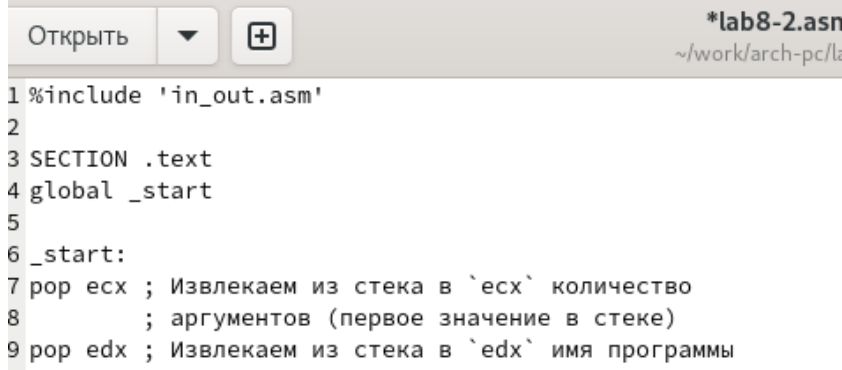
```

Рис. 4.7: Запуск исполняемого файла

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы (рис. 4.8).

```
kamilla@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
kamilla@fedora:~/work/arch-pc/lab08$ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2.asm
kamilla@fedora:~/work/arch-pc/lab08$ gedit lab8-2.asm
```



```
1 %include 'in_out.asm'
2
3 SECTION .text
4 global _start
5
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8         ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
```

Рис. 4.8: Создание файла


Создаю и запускаю новый исполняемый файл (рис. 4.9). Было обработано столько же аргументов, сколько было введено.

```
kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-2 5 9 11
5
9
11
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск исполняемого файла

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввожу в него текст программы вычисления суммы аргументов командной строки (рис. 4.10).

```
kamilla@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
kamilla@fedora:~/work/arch-pc/lab08$ ls
in_out.asm lab8-1.asm lab8-2 lab8-2.o
lab8-1 lab8-1.o lab8-2.asm lab8-3.asm
kamilla@fedora:~/work/arch-pc/lab08$ gedit lab8-3.asm
```



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
```

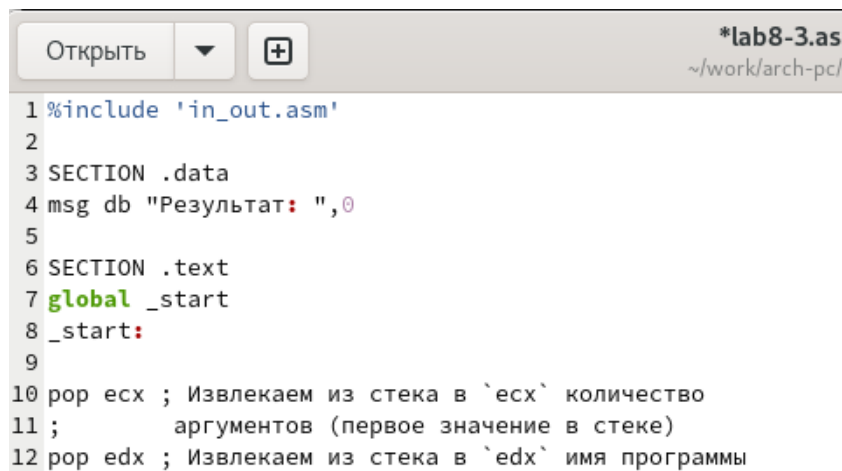
Рис. 4.10: Создание файла

Создаю и запускаю, указав аргументы, исполняемый файл (рис. 4.11).

```
kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-3 8 4 11 3 7
Результат: 33
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.11: Создание файла

Редактирую текст программы для вычисления произведения аргументов строки (рис. 4.12).



```
*lab8-3.as
~/work/arch-pc/

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат: ",0
5
6 SECTION .text
7 global _start
8 _start:
9
10 pop ecx ; Извлекаем из стека в `ecx` количество
11 ;      аргументов (первое значение в стеке)
12 pop edx ; Извлекаем из стека в `edx` имя программы
```

Рис. 4.12: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.13). Программа работает верно.

```
kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-3 1 2 3 4 5
Результат: 120
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.13: Запуск исполняемого файла

4.3 Выполнение заданий для самостоятельной работы

1. Создаю файл lab8-4.asm с помощью утилиты touch (рис. 4.14).

```
kamilla@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
kamilla@fedora:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o  lab8-3.asm  lab8-4.asm
lab8-1      lab8-1.o    lab8-2.asm  lab8-3    lab8-3.o
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.14: Создание файла

Ввожу в созданный файл программу для нахождения суммы значений функции $f(x)$ для $x = 1, 2, \dots, n$ (рис. 4.15). Мой вариант - 17.

```
Открыть  + *lab8-4.asm
~/work/arch-pc/lab08
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg_func db "Функция: f(x) = 10(x-1)", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax, msg_func
12 call sprintf
13
14 pop ecx
15 pop edx
16 sub ecx, 1
17 mov esi, 0
18
19 next:
20 cmp ecx, 0h
```

Рис. 4.15: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.16).

```
kamilla@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
kamilla@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
kamilla@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 10(x-1)
Результат: 60
kamilla@fedora:~/work/arch-pc/lab08$
```

Рис. 4.16: Запуск исполняемого файла

Листинг. Программа для нахождения суммы значений функции $f(x)$ для $x = 1, 2, \dots, n$.

```
%include 'in_out.asm'
```

SECTION .data

```
msg_func db "Функция:  $f(x) = 10(x-1)$ ", 0
```

```
msg_result db "Результат: ", 0
```

SECTION .text

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintf
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx, 0h
```

```
jz _end
```

```
pop eax
```

```
call atoi
```

```
sub eax, 1
```

```
mov ebx, 10
```

```
mul ebx
```

```
add esi, eax
```

```
loop next
```

```
_end:  
mov eax, msg_result  
call sprint  
mov eax, esi  
call iprintLF  
call quit
```


5 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

6 Список литературы

1. Лабораторная работа №8