

Отчёт по лабораторной работе №8

Архитектура компьютера

Морозова Мария Вячеславовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выполнение самостоятельной работы	11
6	Выводы	12
7	Листинги	13

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Результат работы программы	8
4.3	Запуск программы	9
4.4	Запуск программы	9
4.5	Создание файла, запуск программы	10
4.6	Создание файла, запуск, результат работы программы	10
4.7	Создание файла, запуск, результат работы программы	10
5.1	Создание файла, результат работы программы	11

Список таблиц

1 Цель работы

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

2 Задание

Написать программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$.

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).

4 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы 8, перешла в него и создала файл lab8-1.asm: (рис. 4.1).

```
mvmorozova@dk8n64 ~ $ mkdir ~/work/arch-pc/lab08
mvmorozova@dk8n64 ~ $ cd ~/work/arch-pc/lab08
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 4.1: Создание каталога и файла

Создала файл и запустила программу с текстом листинга 8.1, проверила его работу. (рис. 4.2).

```
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 4.2: Результат работы программы

Изменила текст программы добавив изменение значения регистра есх в цикле, проверила работу программы. (рис. 4.3).


```

4291057958
4291057956
4291057954
4291057952
4291057950
4291057948
4291057946
4291057944
4291057942
42910^C
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ^C
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ mc

```

Рис. 4.3: Запуск программы

Создала исполняемый файл, проверила работу программы с добавлением команды push и pop. (рис. 4.4).

```

mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ █

```

Рис. 4.4: Запуск программы

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2. Создала исполняемый файл и запустила его, указав аргументы. (рис. 4.5).

```

mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-2.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ mc

mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-2
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 4.5: Создание файла, запуск программы

Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.3. Создала исполняемый файл и запустила его, указав аргументы.(рис. 4.6).

```

mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ touch lab8-3.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ mc

mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 47
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 4.6: Создание файла, запуск, результат работы программы

Запустила программу с изменениями текста из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. 4.7).

```

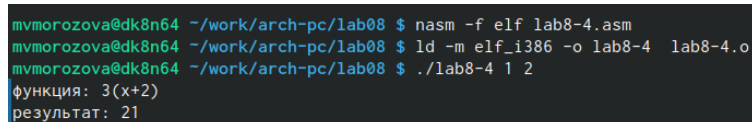
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 54600
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 4.7: Создание файла, запуск, результат работы программы

5 Выполнение самостоятельной работы

Создание файла и запуск программы для вычисления суммы значений функции. (рис. 5.1).



```
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
mvmorozova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-4 1 2
функция: 3(x+2)
результат: 21
```

Рис. 5.1: Создание файла, результат работы программы

6 Выводы

Были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.

7 Листинги

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

```
SECTION .data
```

```
f_x db "функция: 3(x+2)",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
```

```
add eax,2
mov ebx,3
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
```