

Отчет по лабораторной работе №8

Архитектура компьютера

Исаханян Армен Артурович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение самостоятельной работы	16
5	Выводы	18
	Список литературы	19

Список иллюстраций

3.1	Создание каталога и переход в него	7
3.2	Создание файла	8
3.3	Ввод программы	8
3.4	Запуск файла	8
3.5	Изменение программы	9
3.6	Изменение программы	10
3.7	Запуск файла	11
3.8	Изменение программы	12
3.9	Запуск файла	13
3.10	Создание файла	13
3.11	Ввод программы	13
3.12	Запуск файла	14
3.13	Создание файла	14
3.14	Ввод программы	14
3.15	Запуск файла	15
4.1	Создание файла	16
4.2	Ввод программы	16
4.3	Проверка	17

Список таблиц

1 Цель работы

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

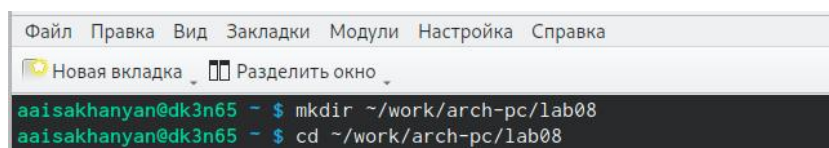
2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

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. На рис. 8.1 показана схема организации стека в процессоре. 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). # Выполнение лабораторной работы

Создал каталог work/arch-pc/lab08 и перешел в него (рис. 3.1).



```
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно
aaisakhanyan@dk3n65 ~ $ mkdir ~/work/arch-pc/lab08
aaisakhanyan@dk3n65 ~ $ cd ~/work/arch-pc/lab08
```

Рис. 3.1: Создание каталога и переход в него

Создал файл в каталоге (рис. 3.2).

```
aaisakhanyan@dk3n65 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 3.2: Создание файла

Ввел программу в файл lab8-1.asm (рис. 3.3).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения 'N'
26 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
27 ; переход на 'label'
28 call quit
```

Рис. 3.3: Ввод программы

Запустил исполняемый файл (рис. 3.4).

```
aaisakhanyan@dk3n65 ~/work/arch-pc/lab07 $ nasm -f elf lab8-1.asm
aaisakhanyan@dk3n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aaisakhanyan@dk3n65 ~/work/arch-pc/lab07 $ ./lab8-1
Введите N: 5
5
4
3
2
1
aaisakhanyan@dk3n65 ~/work/arch-pc/lab07 $
```

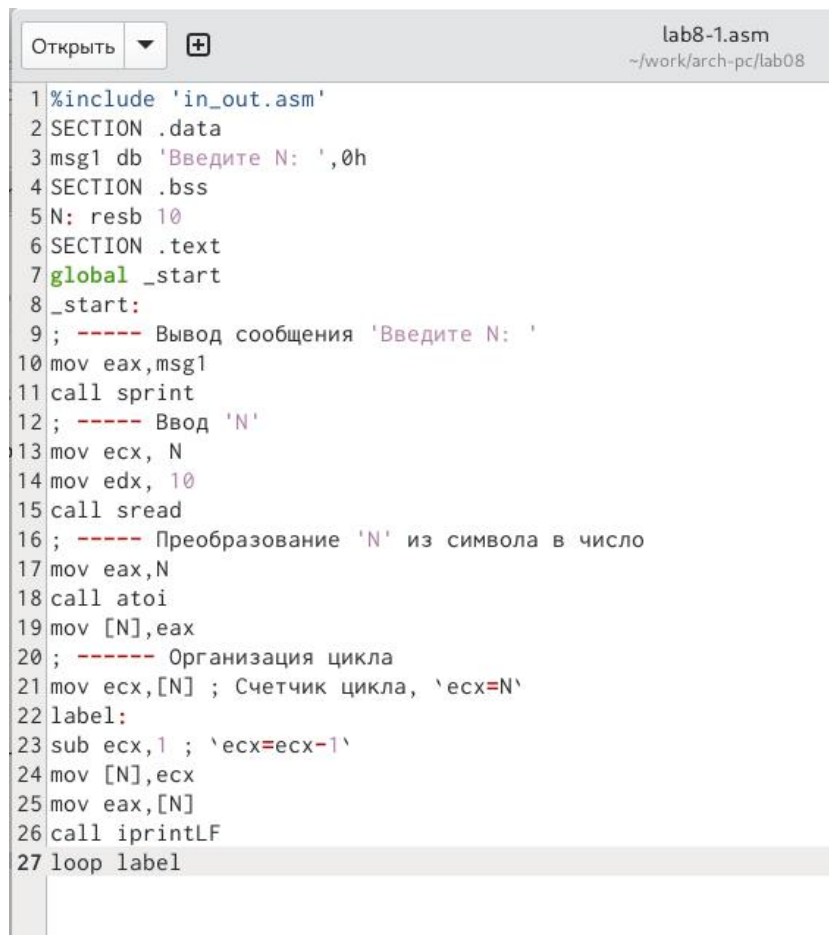
Рис. 3.4: Запуск файла

Изменил текст программы добавив изменение значение регистра ecx в цикле (рис. 3.5).

```
29 label:
30 sub ecx,1 ; 'ecx=ecx-1'
31 mov [N],ecx
32 mov eax,[N]
33 call iprintLF
34 loop label
```

Рис. 3.5: Изменение программы

Редактировал программу в файле lab8-1.asm (рис. 3.6).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 sub ecx,1 ; 'ecx=ecx-1'
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
```

Рис. 3.6: Изменение программы

Запустил исполняемый файл и получил такой результат(рис. 3.7).

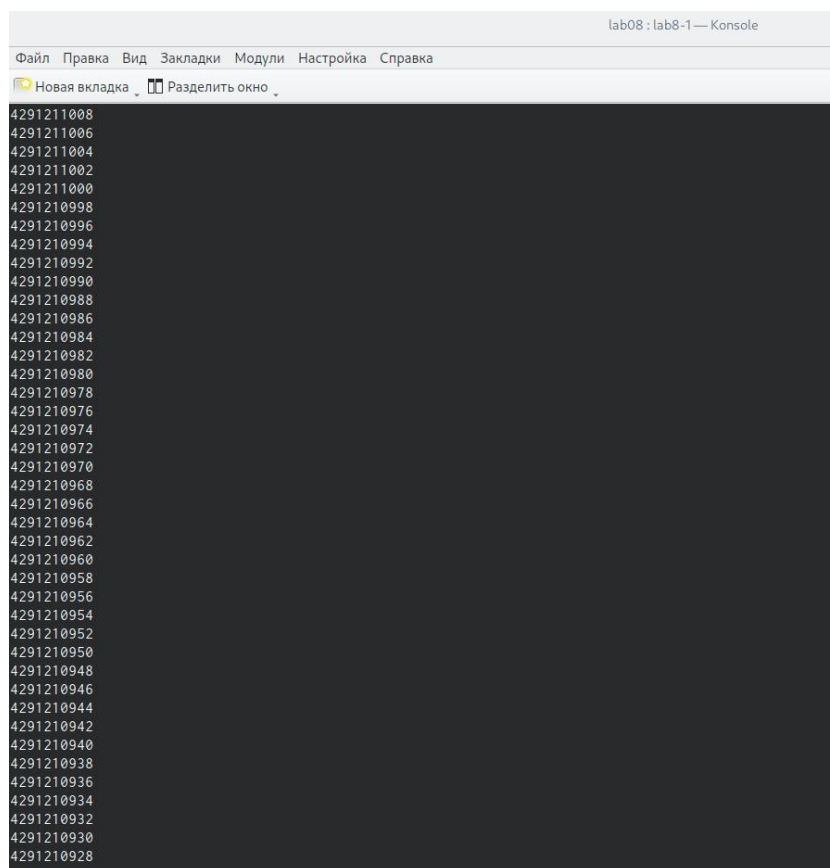
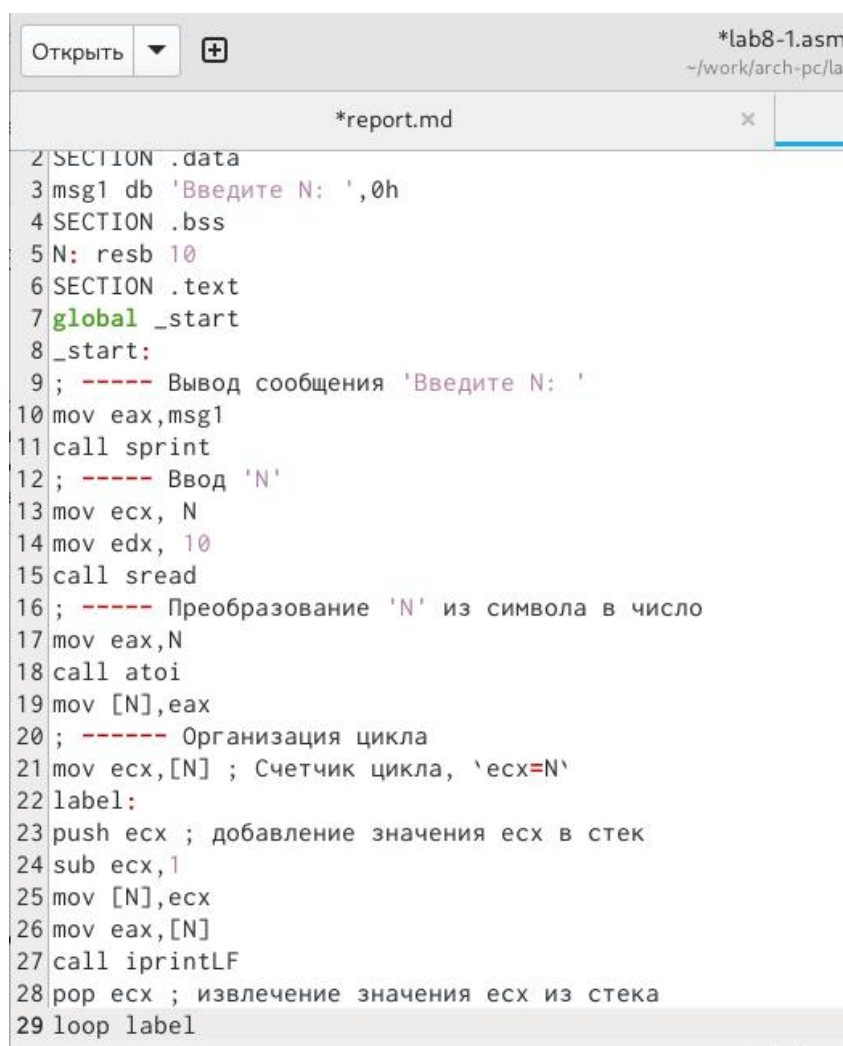


Рис. 3.7: Запуск файла

Внес изменения в текст программы добавив команды `push` и `pop` (рис. 3.8).

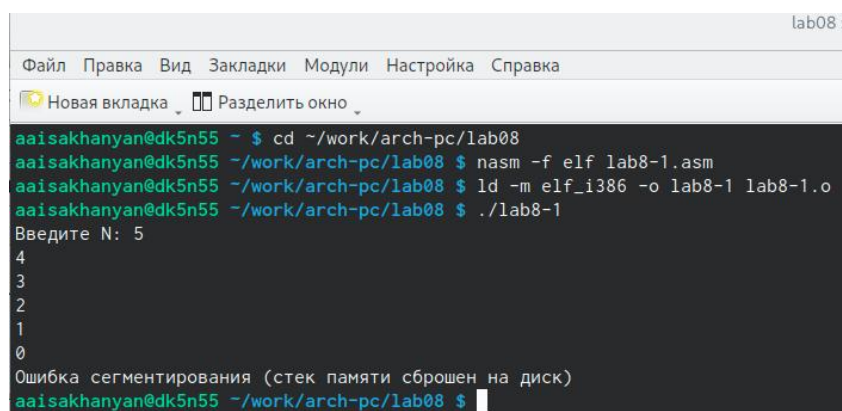


The screenshot shows a code editor window with a title bar containing "Открыть", a dropdown arrow, a plus icon, and the filename "*lab8-1.asm" with a path "~/work/arch-pc/la". Below the title bar is a tab labeled "*report.md" with a close button "x". The main area contains assembly code with line numbers 2 through 29. The code defines a data section with a message, a bss section with a variable N, and a text section with a loop that reads N and prints it.

```
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 push ecx ; добавление значения ecx в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 pop ecx ; извлечение значения ecx из стека
29 loop label
```

Рис. 3.8: Изменение программы

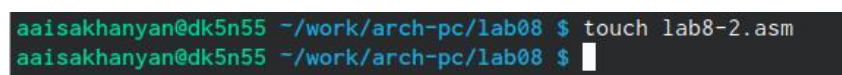
Запустил исполняемый файл (рис. 3.9).



```
lab08:
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно
aaisakhanyan@dk5n55 ~ $ cd ~/work/arch-pc/lab08
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
Ошибка сегментирования (стек памяти сброшен на диск)
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $
```

Рис. 3.9: Запуск файла

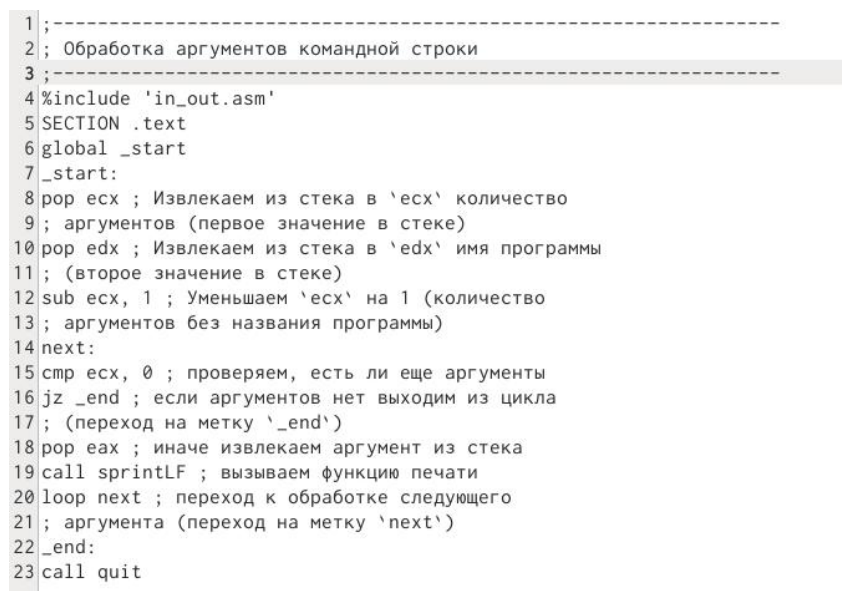
Создал файл lab8-2.asm в том же каталоге (рис. 3.10).



```
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ touch lab8-2.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $
```

Рис. 3.10: Создание файла

Ввел программу в файл (рис. 3.11).



```
1 ;-----
2 ; Обработка аргументов командной строки
3 ;-----
4 %include 'in_out.asm'
5 SECTION .text
6 global _start
7 _start:
8 pop ecx ; Извлекаем из стека в 'ecx' количество
9 ; аргументов (первое значение в стеке)
10 pop edx ; Извлекаем из стека в 'edx' имя программы
11 ; (второе значение в стеке)
12 sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
13 ; аргументов без названия программы)
14 next:
15 cmp ecx, 0 ; проверяем, есть ли еще аргументы
16 jz _end ; если аргументов нет выходим из цикла
17 ; (переход на метку '_end')
18 pop eax ; иначе извлекаем аргумент из стека
19 call sprintf ; вызываем функцию печати
20 loop next ; переход к обработке следующего
21 ; аргумента (переход на метку 'next')
22 _end:
23 call quit
```

Рис. 3.11: Ввод программы

Запустил исполняемый файл указав его аргументы (рис. 3.12).

```
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $
```

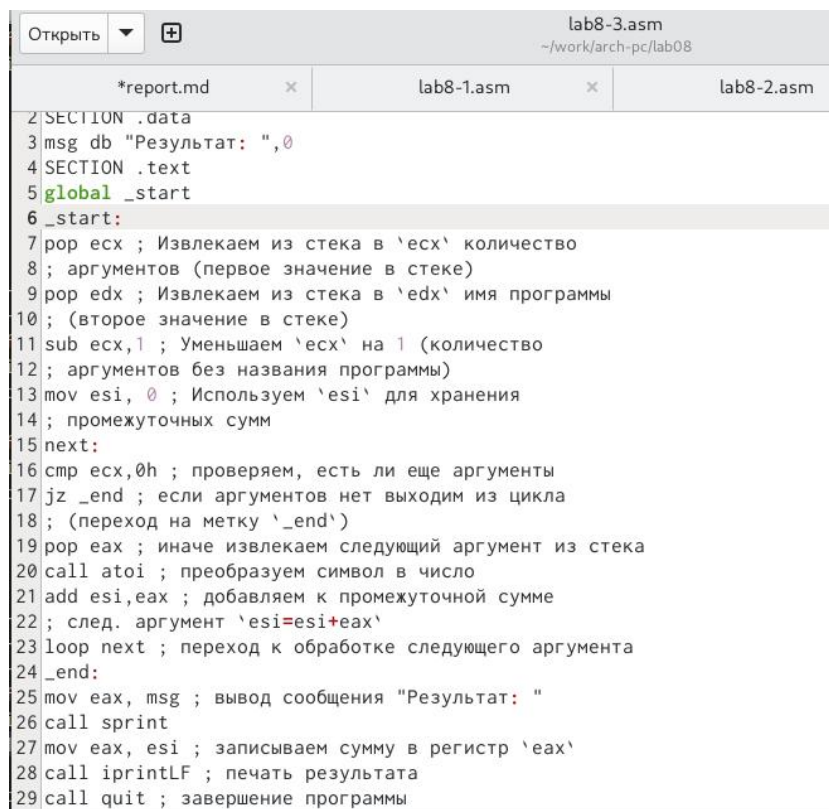
Рис. 3.12: Запуск файла

Создал файл lab8-3.asm в том же каталоге (рис. 3.13).

```
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ touch lab8-3.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $ gedit lab8-3.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab08 $
```

Рис. 3.13: Создание файла

Ввел программу в файл lab8-3.asm (рис. 3.14).



```
lab8-3.asm
~/work/arch-pc/lab08

*report.md x lab8-1.asm x lab8-2.asm

2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
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
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем 'esi' для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку '_end')
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент 'esi=esi+eax'
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр 'eax'
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Рис. 3.14: Ввод программы

Запустил исполняемый файл указав аргументы (рис. 3.15).

```
aaisakhanyan@dk5n55 ~/work/arch-pc/lab88 $ nasm -f elf lab8-3.asm
aaisakhanyan@dk5n55 ~/work/arch-pc/lab88 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aaisakhanyan@dk5n55 ~/work/arch-pc/lab88 $ ld -m elf_i386 -o main lab8-3.o
aaisakhanyan@dk5n55 ~/work/arch-pc/lab88 $ ./main 12 13 7 10 5
Результат: 47
aaisakhanyan@dk5n55 ~/work/arch-pc/lab88 $
```

Рис. 3.15: Запуск файла

4 Выполнение самостоятельной работы

Создал файл sam.asm для выполнения самостоятельной работы (рис. 4.1).

```
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $ touch sam.asm
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $
```

Рис. 4.1: Создание файла

Написал программу для нахождения суммы значений функции $10(x-1)$ (вариант 17) (рис. 4.2).

Открыть

*sam.asm

~/work/arch-pc/lab08

Сохранить

*report.md

*sam.asm

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 f_x db "функция: 10(x - 1)",0h
5 msg db 10,13,'результат: ',0h
6
7 SECTION .text
8 global _start
9
10 _start:
11 pop ecx
12 pop edx
13 sub ecx, 1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 dec eax
22 mov ebx, 10
23 mul ebx
24 add esi, eax
25
26 loop next
27
28 end;
```

Рис. 4.2: Ввод программы

Запускаю исполняемый файл с аргументами 1 и 2 (рис. 4.3).


```
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf sam.asm
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o sam sam.o
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $ ./sam 1 2
функция: 10(x - 1)
результат: 10
aaisakhanyan@dk8n75 ~/work/arch-pc/lab08 $
```

Рис. 4.3: Проверка

5 Выводы

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

Список литературы