Лабораторная работа №8

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки

2 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 ре- гистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указа- тель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №8 и создаю файл lab8-1.asm (рис. 3.1).

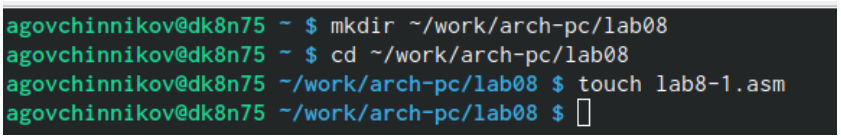

Рис. 3.1: Создание каталога

Ввожу в файл текст из листинга 8.1 (рис. 3.2).

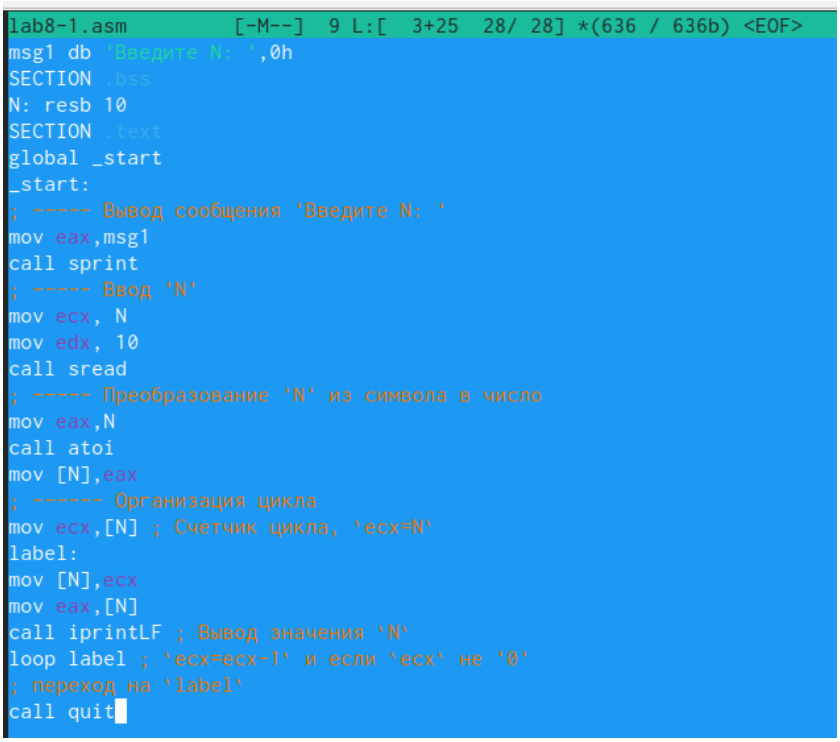

Рис. 3.2: Текст файла

Создаю исполняемый файл и проверяю его работу (рис. 3.3). В данном случае число проходов цикла не соответствует значению N введенному с клавиатуры.

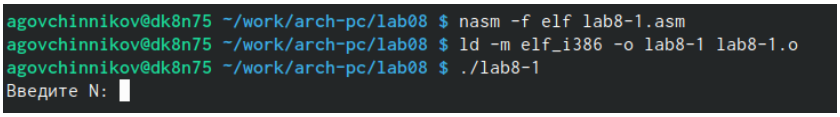

Рис. 3.3: Вывод программы

Меняю текст листинга файла и проверяю рузльтат программы (рис. 3.4). В данном случае число проходов цикла соответствует значению N введенному с клавиатуры.

```
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 3
2
1
0
```

Рис. 3.4: Результат измененого файла

Создаю файл lab8-2.asm и ввожу туда текст из листинга 8.2 (рис. 3.5).

```
lab8-2.asm          [----]  9 L:[  1+19  20/ 20] *(943 / 943b) <EOF>
%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. 3.5: Текст файла

Создаю исполняемый файл и проверяю его работу (рис. 3.6).

```
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-2
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 3.6: Вывод программы

Создаю файл lab8-3.asm и ввожу в него текст программы из листинга 8.3 (рис. 3.7)

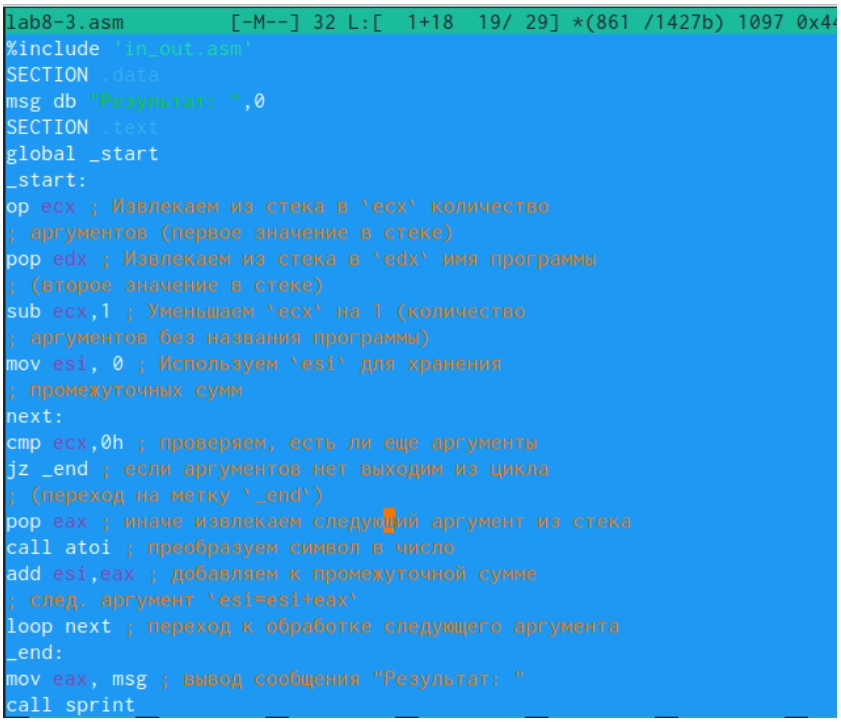

Рис. 3.7: Файл lab8-3.asm

Создаю исполняемый файл и запускаю его, при этом указывая аргументы (рис. 3.8)

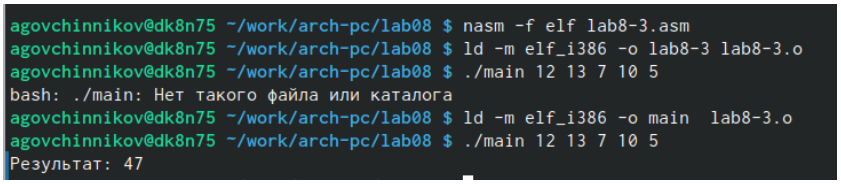

Рис. 3.8: Вывод программы

Изменяю текст программы из листинга для вычисления произведения аргументов командной строки (рис. 3.9)

Рис. 3.9: Изменненый текст

Создаю исполняемый файл и проверяю его работу (рис. 3.10).

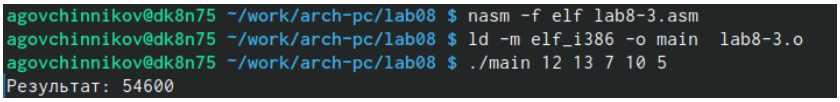

Рис. 3.10: Вывод программы

4 Задание для самостоятельной работы

Вариант 15

Пишу программу, которая находит сумму значений функции f(x) (рис. 4.1).

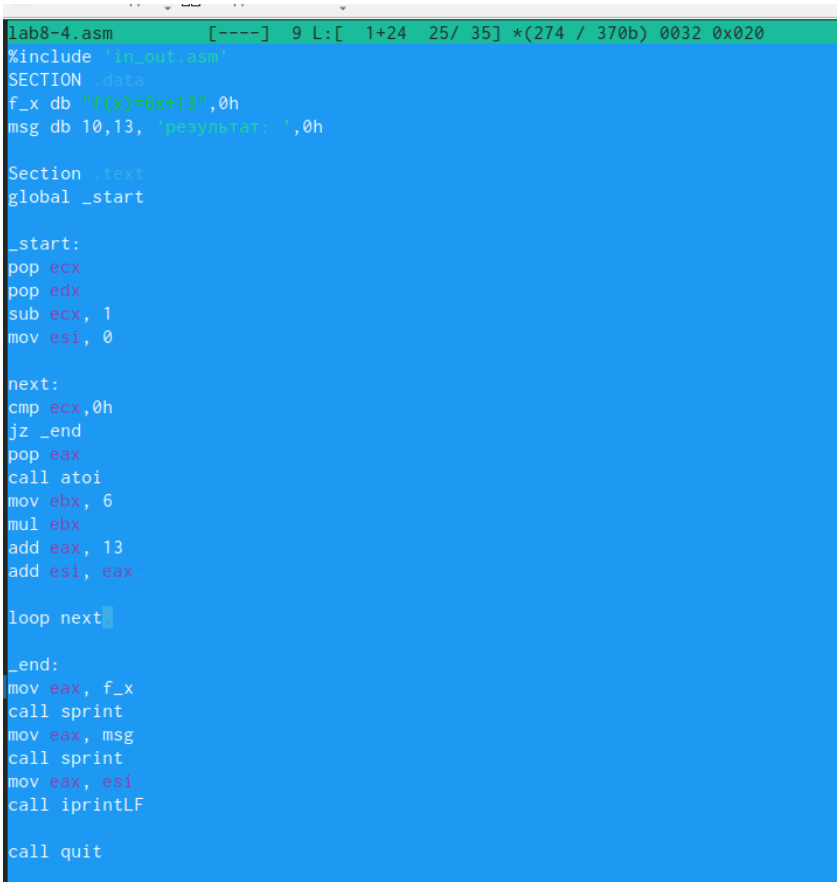

Рис. 4.1: Текст программы

Создаю исполняемый файл и проверяю его работу (рис. 4.2).

```
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-4.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./main 1 2 3 4
f(x)=6x+13
результат: 112
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $
```

Рис. 4.2: Вывод программы

5 Листинг

```
%include 'in_out.asm'
SECTION .data
f_x db "f(x)=6x+13",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
mov ebx, 6
mul ebx
add eax, 13
```

```
add esi, eax

loop next

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

call quit
```

6 Выводы

Я приобрел навыки написания программ с использованием циклов и обраткой аргументов командной строки