Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Сокирка Анна Константиновна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Самостоятельное написание программы по материалам лабораторной 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.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создам каталог для программам лабораторной работы № 8, перейду в него и создам файл lab8-1.asm (рис. 4.1).

Рис. 4.1: Создание каталога

Введу в файл lab8-1.asm текст программы из листинга 8.1. Создам исполняемый файл и проверю его работу (рис. 4.2).

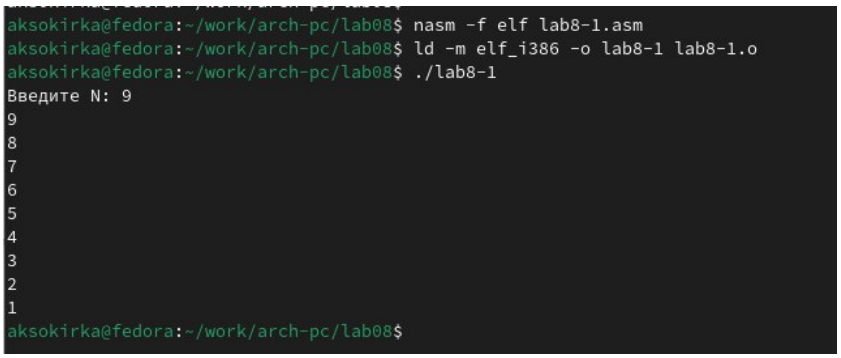

Рис. 4.2: Ввод программы из листинга

Изменю текст программы, добавив изменение значение регистра ecx в цикле. Создам исполняемый файл и проверю его работу (рис. 4.3).

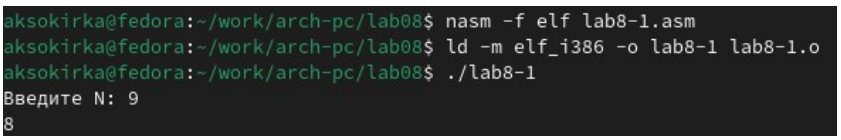

Рис. 4.3: Изменение текста программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое. Число проходов цикла не соответсвует значению введенному с клавиатуры.

Внесу изменения в текст программы, добавив команды push и pop. Создам исполняемый файл и проверю его работу (рис. 4.4).

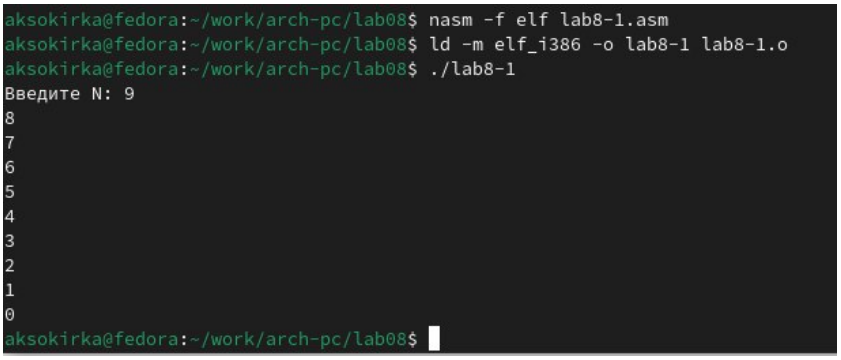

Рис. 4.4: Изменение текста программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1

4.2 Обработка аргументов командной строки

Создам файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введу в него текст программы из листинга 8.2. (рис. 4.5).

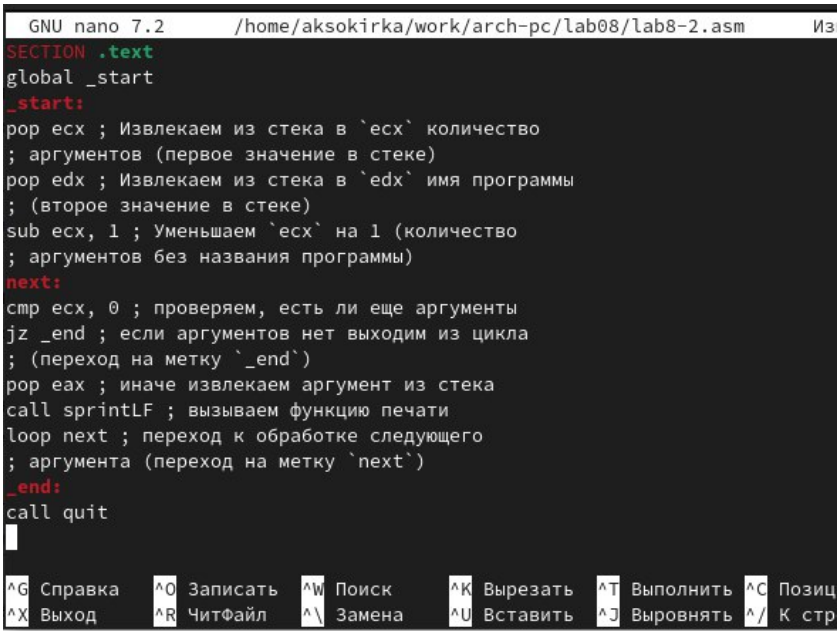

Рис. 4.5: Создание файла

Создам исполняемый файл и запущу его, указав аргументы (рис. 4.6).

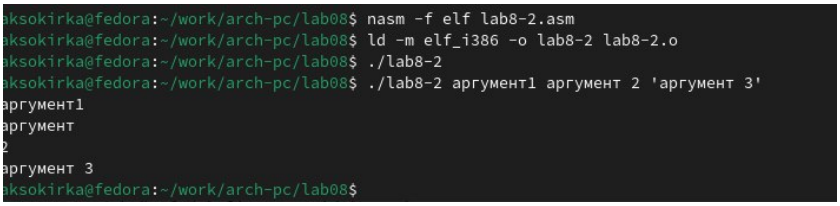

Рис. 4.6: Создание исполняемого файла

Программой было обратоно то же количество аргументов,которое было введено

Создам файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введу в него текст программы из листинга 8.3. (рис. 4.7).

```
 GNU nano 7.2                    /home/aksokirka/work/arch-pc/lab08/lab8-3.asm
%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. 4.7: Создание файла

Создайте исполняемый файл и запустите его, указав аргументы (рис. 4.8).

```
aksokirka@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aksokirka@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aksokirka@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
aksokirka@fedora:~/work/arch-pc/lab08$
```

Рис. 4.8: Создание исполняемого файла

Изменяю программу так, чтобы указанные аргументы умножались (рис. 4.9).

```
 GNU nano 7.2                  /home/aksokirka/work/arch-pc/lab08/lab8-3.asm
%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
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. 4.9: Изменение текста программы

Запускаю программу (рис. 4.10).

```
aksokirka@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aksokirka@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aksokirka@fedora:~/work/arch-pc/lab08$ ./lab8-3 1 99 100
Результат: 9900
aksokirka@fedora:~/work/arch-pc/lab08$
```

Рис. 4.10: Запуск программы

4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумму значений для функции f(x) = 8x-3, которая совпадает с моим девятнадцатым вариантом (рис. 4.11).

```
 GNU nano 7.2                    /home/aksokirka/work/arch-pc/lab08/lab8-4.asm
%include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 8x - 3", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 8
mul ebx
sub eax, 3
add esi, eax
loop next
_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 4.11: Программа для нахождения суммы значений функции

Запускаю программу (рис. 4.12).

```
aksokirka@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aksokirka@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aksokirka@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 8x - 3
Результат: 68
aksokirka@fedora:~/work/arch-pc/lab08$
```

Рис. 4.12: Запуск программы

5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов, а также научилась обрабатывать аргументы командной строки.

6 Список литературы

https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/Лабораторная%20работа%20№8.%20Программирование%20цикла.%20Обработка%20аргументов%20командной%20строки..pdf