РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: Архитектура компьютера

Студент:Безлепкина Татьяна Игоревна

Группа:НКАбд-01-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

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

2 Порядок выполнения лабораторной работы

2.1.Реализация циклов в NASM

Создам каталог для программ лабораторной работы № 8, перейду в него и создам файл lab8-1.asm.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ mkdir ~/work/arch-pc/lab08
tibezlepkina1@localhost-live:~/work/arch-pc/lab07$ cd ~/work/arch-pc/lab08
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рисунок 2.1.1 - создание каталога,переход в него,создание файла lab8-1.asm

Найду файл «in_out.asm» в системе.Скопирую файл «in_out.asm» в текущую директорию.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ find / -name "in_out.asm" 2>/dev/null
/home/tibezlepkina1/work/arch-pc/lab07/in_out.asm
/home/tibezlepkina1/work/arch-pc/lab06/in_out.asm
/home/tibezlepkina1/work/arch-pc/lab05/in_out.asm
/run/overlayfs/home/tibezlepkina1/work/arch-pc/lab07/in_out.asm
/run/overlayfs/home/tibezlepkina1/work/arch-pc/lab06/in_out.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ cp /home/tibezlepkina1/work/arch-pc/lab07/in_out.asm
 ./
```

Рисунок 2.1.2 - поиск файла «in_out.asm» в системе,его копирование в текущую директорию

Перейду в текстовый редактор nano.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-1.asm
```

Рисунок 2.1.3 - переход в текстовый редактор nano

```
 GNU nano 8.3                                    lab8-1.asm
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
```

Рисунок 2.1.4 - ввод программы в текстовый редактор nano

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рисунок 2.1.5 - компиляция,линковка,запуск

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы.

Изменю текст программы добавив изменение значение регистра ecx в цикле:

Перейдем в текстовый редакор nano.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-1.asm
```

Рисунок 2.1.6 - переход в текстовый редактор nano

```
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения N
loop label ; ecx=ecx-1 и если ecx не '0'
; переход на label
```

Рисунок 2.1.7 - изменение текста программы в редакторе nano

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
```

Рисунок 2.1.8 - компиляция,линковка,запуск

Регистр ecx принимает значения: N, N-2, N-4, ..., 0, -2, -4, -6, ... и так до переполнения.Уменьшается на 2 за каждую итерацию из-за двойного уменьшения (sub ecx,1 + loop)

Число проходов цикла НЕ соответствует значению N,потому что в коде **цикл выполняется бесконечно** (до переполнения регистра), а не N раз.

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-1.asm
```

Рисунок 2.1.9 - переход в текстовый редактор nano

```
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения N
pop ecx
loop label ; ecx=ecx-1 и если ecx не '0'
; переход на label
```

Рисунок 2.1.10 - изменение текста программы в редакторе nano

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 7
6
5
4
3
2
1
0
```

Рисунок 2.1.11 -компиляция,линковка,запуск.

В этом исправленном коде число проходов цикла **соответствует** значению N.

Код теперь работает корректно благодаря использованию стека (push/pop) для сохранения значения счетчика цикла.

2.2 Обработка аргументов командной строки

При разработке программ часто возникает потребность передавать параметры непосредственно из командной строки в момент запуска программы. В NASM эти аргументы автоматически размещаются в стеке в обратной последовательности. Помимо пользовательских аргументов, система дополнительно помещает в стек два служебных параметра: имя запускаемой программы и общее количество переданных аргументов. Эти два элемента всегда занимают последние позиции в стеке для программ, скомпилированных через NASM.

Следовательно, для работы с аргументами командной строки программа должна последовательно извлекать их из стека. Обработку следует организовать в циклическом режиме: сначала из стека извлекается количество аргументов, а затем в цикле для каждого аргумента выполняется необходимая программная логика. В качестве иллюстрации данного подхода можно рассмотреть пример программы, которая выводит на экран все переданные через командную строку аргументы.

Создам файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введу в него текст программы.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ touch lab8-2.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-2.asm
```

Рисунок 2.2.1 - создание файла lab8-2.asm и переход в текстовый редактор

```
  GNU nano 8.3                                  lab8-2.asm
%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
```

Рисунок 2.2.2 - изменение текста программы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2  lab8-2.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рисунок 2.2.3 - компиляция,линковка,запуск

Количество выведенных строк = количеству обработанных аргументов = ecx - 1

Создам файл lab8-3.asm.Перейду в текстовый редактор.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ touch lab8-3.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nano lab8-3.asm
```

Рисунок 2.2.4 - создание файла,переход в текстовый редактор

```
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
```

Рисунок 2.2.5 - изменение текста программы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3  lab8-3.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рисунок 2.2.6 - компиляция,линковка,запуск

Изменю текст программы,чтобы она выводила произведение.

```
 GNU nano 8.3                                  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
pop edx ; Извлекаем из стека в edx имя программы
sub ecx,1 ; Уменьшаем ecx на 1 (количество аргументов без названия програ
mov esi, 1 ; Используем esi для хранения произведения (начальное значени
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
pop eax ; иначе извлекаем следующий аргумент из стека
```

Рисунок 2.2.7 - изменение текста программы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3  lab8-3.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
```

Рисунок 2.2.8 - компиляция,линковка,запуск

3 Задания для самостоятельной работы

Создам файл <lab8-4.asm>.

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ touch lab8-4.asm
```

Рисунок 3.1 - создание файла <lab8-4.asm>

Перейду в текстовый редактор nano и введу текст программы(дял 20 варианта).

```
 GNU nano 8.3                                    lab8-4.asm
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=3(10+x)",0
msg_result db "Результат: ",0

SECTION .text
global _start

_start:
    pop ecx             ; Количество аргументов (argc)
    pop edx             ; Имя программы (argv[0])
    sub ecx, 1          ; Убираем имя программы из счётчик
    mov esi, 0          ; Начальная сумма = 0

    ; Вывод информации о функции
    mov eax, msg_func
```

Рисунок 3.2 - ввод текста программы

```
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4  lab8-4.o
tibezlepkina1@localhost-live:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x)=3(10+x)
Результат: 150
```

Рисунок 3.3 - компиляция,линковка,запуск

Загружу данные отчета на github.

4 Вывод

В ходе выполнения лабораторной работы были приобретены практические навыки программирования на ассемблере NASM с использованием циклов и обработки аргументов командной строки. Была изучена организация стека и принципы работы с ним, включая операции push и pop. В процессе работы освоена инструкция loop для организации циклов, а также методы корректного использования регистра ecx в качестве счётчика. На практике была реализована программа, вычисляющая сумму значений функции f(x) = 3(10 + x) для аргументов, передаваемых через командную строку. Программа успешно обрабатывает произвольное количество входных данных и выводит корректный результат. Работа подтвердила важность правильного управления стеком и регистрами при обработке аргументов в циклических конструкциях. Полученные навыки позволяют создавать более сложные программы на ассемблере с эффективной обработкой входных параметров.