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:

Мартемьянов Максим

Группа:

НКАбд-04-25

МОСКВА

2025 г.

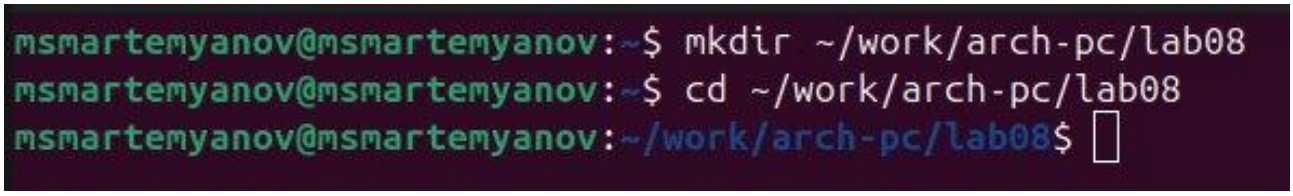
1. Цель работы:

Приобретение навыков написания программ при помощи использования циклов и обработкой аргументов командной строки.

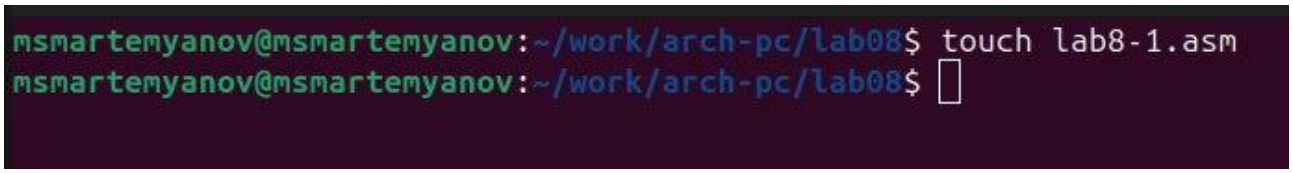
Выполнение лабораторной работы

Реализация циклов в NASM.

Создал каталог для программам лабораторной работы № 8 и перешел в него.



Создал файл lab8-1.asm.



Ввел в файл lab8-1.asm текст программы из листинга 8.1.

```
%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
    call sprint
    ; Ввод 'N'
    mov ecx, N
    mov edx, 10
    call sread
    ; Преобразование 'N' из символа в число
    mov eax, N
    call atoi
    mov [N], eax
    ; Организация цикла
    mov ecx, [N]    ; Счетчик цикла
label:
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    loop label
    call quit
```

Создал исполняемый файл и проверил его работу.

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

Изменил текст программы добавив изменение значение регистра **ecx** в цикле.

```
%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
    call sprint
    ; Ввод 'N'
    mov ecx, N
    mov edx, 10
    call sread
    ; Преобразование 'N' из символа в число
    mov eax, N
    call atoi
    mov [N], eax
    ; Организация цикла с изменением ecx
    mov ecx, [N]    ; Счетчик цикла
label:
    sub ecx, 1      ; изменяем ecx
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    loop label
    call quit
```

Создал исполняемый файл и проверил его работу.

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

Из-за того,что теперь регистр **ecx** на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое.

Внес изменения в текст программы добавив команды push и pop.

```
%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
    call sprit
    ; Ввод 'N'
    mov ecx, N
    mov edx, 10
    call sread
    ; Преобразование 'N' из символа в число
    mov eax, N
    call atoi
    mov [N], eax
    ; Организация цикла с сохранением в стек
    mov ecx, [N]    ; Счетчик цикла
label:
    push ecx        ; добавление значения ec
    sub ecx, 1
    mov [N], ecx
    mov eax, [N]
    call iprintLF
    pop ecx         ; извлечение значения ec
```

Создал исполняемый файл и проверил его работу.

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

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

Обработка аргументов командной строки.

Создал файл lab8-2.asm в каталоге ~/work/arch-pc/lab08

```
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ touch lab8-2.asm
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$
```

Ввел в него текст программы из листинга 8.2.

```
%include 'in_out.asm'

SECTION .text
global _start

_start:
    pop ecx         ; Извлекаем из стека в е
    pop edx         ; Извлекаем из стека в е
    sub ecx, 1      ; Уменьшаем ecx на 1 (ко

next:
    cmp ecx, 0
    jz _end
    pop eax
    call sprintLF
    loop next
_end:
    call quit
```

Создал исполняемый файл и запустил его, указав аргументы.

```
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$
```

Программой было обработано то же количество аргументов, что и было введено.

```
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ touch lab8-3.asm
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$
```

```
%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 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
    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
```

Создал исполняемый файл и запустил его, указав аргументы.

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

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```
%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 программы)
    ; Проверка на наличие аргументов
    cmp ecx, 0
    jz _no_args
    pop eax         ; извлекаем первый аргумент
    call atoi       ; преобразуем в число
    mov esi, eax    ; сохраняем как начальное значение произведения
    dec ecx         ; уменьшаем счетчик, так как первый аргумент уже обработан
    cmp ecx, 0
    jz _end         ; если больше аргументов нет
next:
    pop eax         ; извлекаем следующий аргумент
    call atoi       ; преобразуем в число
    ; Умножение esi * eax (результат в eax)
    mov ebx, eax    ; сохраняем множитель
    mov eax, esi    ; переносим текущее произведение в eax
    mov edx, 0      ; обнуляем edx для умножения
    imul ebx        ; умножаем eax на ebx
    mov esi, eax    ; сохраняем результат обратно в esi
    loop next       ; переход к обработке следующего аргумента
```

Создал исполняемый файл и запустил его, указав аргументы.

```
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ./lab8-3 4 14 120
Результат: 6720
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$
```

Задание для самостоятельной работы.

Создал файл lab8-4.asm в каталоге ~/work/arch-pc/lab08, для выполнения заданий для самостоятельной работы.(Вариант номер 15)

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

Написал программу,которая подсчитает мою функцию(Вариант 15)

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
func_msg db "Функция: f(x)=6x+13",0
SECTION .text
global _start
_start:
    pop ecx         ; количество аргументов
    pop edx         ; имя программы
    sub ecx, 1      ; без имени программы
    mov esi, 0      ; для накопления суммы
    ; Вывод сообщения о функции
    mov eax, func_msg
    call sprintLF
    ; Проверка наличия аргументов
    cmp ecx, 0
    jz _end
next:
    pop eax         ; получаем аргумент
    call atoi       ; преобразуем в число
    ; Альтернативное вычисление f(x) = 6x + 1
    mov ebx, eax    ; сохраняем x в ebx
    add eax, eax    ; 2x
    mov edx, eax    ; сохраняем 2x в edx
    add eax, eax    ; 4x
    add eax, edx    ; 4x + 2x = 6x
    add eax, 13     ; 6x + 13
    ; Добавляем к общей сумме
```

Создал исполняемый файл и проверил его работу.

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

```
Результат: 112
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$ ./lab8-4 5 6 7
Функция: f(x)=6x+13
Результат: 147
msmartemyanov@msmartemyanov:~/work/arch-pc/lab08$
```

Вопросы для самопроверки.

1) Работает с регистром CX/ECX как счётчиком. Уменьшает его на 1 и переходит на метку, если значение не равно нулю.

2)Организация циклов без loop:

```
mov cx, N start:
; тело цикла
dec cx
jnz start
```

3)Стек – это структура данных с принципом LIFO (Last-In-First-Out), где элементы добавляются и извлекаются с одного конца (вершины).

4) Порядок выборки из стека:

Данные извлекаются в порядке, обратном их помещению. Последний сохранённый элемент извлекается первым.

Вывод. В результате данной лабораторной работы я приобрёл навыки написания программ с использованием циклов и научился обрабатывать аргументы командной строки.

Список литературы.

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.

9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).

Демидова А. В.