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: Нурмухаметова Каролина Марселевна

Группа: НКАбд-05-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Реализация циклов в NASM

1. Создала каталог для программам лабораторной работы № 8, перешла в него и создала файл lab8-1.asm.

```
kmnurmukhametov@dk6n01 ~ $ mkdir ~/work/arch-pc/lab08
kmnurmukhametov@dk6n01 ~ $ cd ~/work/arch-pc/lab08
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-1.asm
```

Рис. 2.1.1 — Создание каталога

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счётчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрели программу, которая выводит значение регистра ecx. Внимательно изучила текст программы Листинга 8.1 и ввела его в файл lab8-1.asm (Рис. 2.1.2). Создала исполняемый файл и проверила его работу (Рис. 2.1.3).

Открыть | lab8-... ~/work... | Сохранить

```
;-------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------

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

Рис. 2.1.2 — Текст программы lab8-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 15
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $
```

Рис. 2.1.3 — Вывод программы lab8-1.asm

Создала исполняемый файл и проверила его работу. Число проходов цикла значени ю соответствует введённому с клавиатуры (Рис. 2.1.3).

Изменила текст программы добавив изменение значение регистра ecx в цикле.

Создала исполняемый файл lab8-1-1.asm и проверила его работу.

```
 1 ;-------------------------------------------------
 2 ; Программа вывода значений регистра 'ecx'
 3 ;-------------------------------------------------
 4
 5 %include 'in_out.asm'
 6
 7 SECTION .data
 8    msg1 db 'Введите N: ',0h
 9
10 SECTION .bss
11    N: resb 10
12
13 SECTION .text
14    global _start
15 _start:
16
17 ; ----- Вывод сообщения 'Введите N: '
18    mov eax,msg1
19    call sprint
20
21 ; ----- Ввод 'N'
22    mov ecx, N
23    mov edx, 10
24    call sread
25
26 ; ----- Преобразование 'N' из символа в число
27    mov eax,N
28    call atoi
29    mov [N],eax
30
31 ; ------ Организация цикла
32    mov ecx,[N] ; Счетчик цикла, `ecx=N`
33 label:
34    sub ecx,1 ; `ecx=ecx-1`
35    mov [N],ecx
36    mov eax,[N]
37    call iprintLF
38    loop label
```

Рис. 2.1.4 — Текст программы lab8-1-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-1-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1-1 lab8-1-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-1-1
Введите N: 4
3
1
Ошибка сегментирования (образ памяти сброшен на диск)
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $
```

Рис. 2.1.5 — Вывод программы lab8-1-1.asm

Число проходов цикла не соответствует значению, введённому с клавиатуры.

Внесла изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счётчика цикла loop (Рис. 2.1.6)

```
;-------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------

%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, `ecx=N`
label:
   push ecx ; добавление значения ecx в стек
   sub ecx,1
   mov [N],ecx
   mov eax,[N]
   call iprintLF
   pop ecx ; извлечение значения ecx из стека
   loop label
```

Рис. 2.1.6 — Текст программы lab8-1-2.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-1-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1-2 lab8-1-2.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-1-2
Введите N: 4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $
```

Рис. 2.1.7 — Вывод программы lab8-1-2.asm

2.2 Обработка аргументов командной строки

2. При разработке программ иногда встаёт необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрели программу, которая выводит на экран аргументы командной строки. Внимательно изучила текст программы Листинга 8.2 (Рис. 2.2.1)

```
;-------------------------------------------------------------
; Обработка аргументов командной строки
;-------------------------------------------------------------

%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. 2.2.1 — Текст листинга 8.2

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2. Создала исполняемый файл и запустила его, указав аргументы (Рис. 4.2.2).

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-2 5 4 '3'
5
4
3
```

Рис. 2.2.2 — Вывод программы lab8-2.asm

Количество обработанных аргументов равно исходному количеству элементов в стеке минус 2.

Рассмотрели ещё один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.3 (Рис. 2.2.3).

```
%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. 2.2.3 — Текст листинга 8.3

Создала исполняемый файл и запустила его, указав аргументы. Пример результата работы программы показан на Рис. 2.2.4.

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-3.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 2.2.4 — Вывод программы lab8-3.asm

Изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (Рис. 2.2.5). Результат работы программы представлен на Рис. 2.2.6.

```
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

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
   sub eax,1
   mov ebx,10
   imul eax,ebx
   add esi,eax
   loop next

_end:
   mov eax, msg
   call sprint
   mov eax, esi
   call iprintLF
   call quit
```

Рис. 2.2.5 — Текст программы lab8-3-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3-1 lab8-3-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-3-1 7 10 5
Результат: 350
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $
```

Рис. 2.2.6 — Вывод программы lab8-3-1.asm

3 Задание для самостоятельной работы

1. Написала программу, которая находит сумму значений функции f(x) для x = x1, x2, ..., xn, т.е. программа должна выводить значение f(x1) + f(x2) + ... + f(xn). Значения передаются как аргументы. Вид функции: 10(x−1). Создала исполняемый файл и проверила его работу на нескольких наборах = x1, x2, ..., xn. (Рис. 2.3.1, Рис. 2.3.2).

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg db "Результат для функции f(x)=10(x-1): ",0
5
6 SECTION .text
7 global _start
8
9 _start:
10    pop ecx
11    pop edx
12    sub ecx,1
13    mov esi, 0
14 next:
15    cmp ecx,0h
16    jz _end
17    pop eax
18    call atoi
19    sub eax,1
20    mov ebx,10
21    imul eax,ebx
22    add esi,eax
23    loop next
24
25 _end:
26    mov eax, msg
27    call sprint
28    mov eax, esi
29    call iprintLF
30    call quit
```

Рис. 2.3.1 — Текст самостоятельного задания №1

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ touch lab8-4.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
Результат для функции f(x)=10(x-1): 60
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab08 $
```

Рис. 2.3.2 — Результат выполнения самостоятельного задания №1

4 Выводы

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

5 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/Learning-bash-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. — БХВ-Петербург, 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).