

Отчёта по лабораторной работе #8

НПИ-03-23

Махмудов Суннатило Баходир угли

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Задание для самостоятельной работы	15
6	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Созданный файл lab8-1.asm	8
4.2	Текст из листинга 8.1	9
4.3	Результат работы	9
4.4	Измененный текст	10
4.5	Проверял его работу	10
4.6	Измененный текст	11
4.7	Проверял его работу	11
4.8	Файл lab8-2.asm	12
4.9	Текст из листинга 8.2	12
4.10	Аргументы полученные	12
4.11	файл lab8-3.asm	12
4.12	Текст из листинга 8.3	13
4.13	Результат	13
4.14	Измененный текст	13
4.15	Результат работы	14
5.1	файл lab8-4.asm	15
5.2	Текст	15
5.3	Результат работы	16

Список таблиц

1 Цель работы

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

2 Задание

Реализация циклов в NASM

Обработка аргументов командной строки

Задание для самостоятельной работы

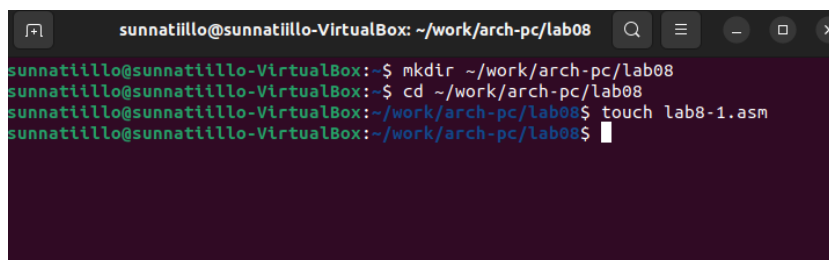
3 Теоретическое введение

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. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

4 Выполнение лабораторной работы

Реализация циклов в NASM

Создал каталог для программ лабораторной работы № 8, перешел в него и создал файл lab8-1.asm: (рис. 5.3).



```
sunnatillo@sunnatillo-VirtualBox: ~/work/arch-pc/lab08
sunnatillo@sunnatillo-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
sunnatillo@sunnatillo-VirtualBox:~$ cd ~/work/arch-pc/lab08
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.1: Созданный файл lab8-1.asm

Ввел в файл lab8-1.asm текст программы из листинга 8.1.(рис. 5.3).


```

GNU nano 6.2 /home/sunnatillo/work/arch-pc/lab08/lab8-1.asm *
;-----
; 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

```

Рис. 4.2: Текст из листинга 8.1

Исполняемый файлы проверял его работу.(рис. 5.3).

```

sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ nasm -f elf lab8-1.asm
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ ld -n elf_i386 -o lab8-1 lab8-1.o
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
4
3
2
1
0

```

Рис. 4.3: Результат работу

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

Изменил текст программы добавил изменение значение регистра ecx в цик-
ле(рис. 5.3).

```
GNU nano 6.2 /home/sunnatillo/work/arch-pc/lab08/lab8-1.asm *
;-----
; 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:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
```

Рис. 4.4: Измененный текст

```
4289821352
4289821350
4289821348
4289821346
4289821344
4289821342
4289821340
4289821338
4289821336
4289821334
4289821332
4289821330
4289821328
4289821326
4289821324
4289821322
4289821320
4289821318
4289821316
4289821314
4289821312
4289821310
4289821308
4289821306
4289821304
4289821302
4289821300
4289821298
4289821296
4289821294
4289821292
4289821290
4289821288
4289821286
4289821284
4289821282
4289821280
4289821278
4289821276
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.5: Проверка его работы

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. В изменения в текст программы добавив команды `push` и `pop` для сохранения значения счетчика цикла `loop`:

Опять изменил текст программы (рис. 5.3).

```

GNU nano 6.2 /home/sunnatillo/work/arch-pc/lab08/lab8-1.asm
; 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 lprintfLF
pop ecx ; извлечение значения ecx из стека
loop label

```

Рис. 4.6: Измененный текст

```

sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 50
50
49
48
47
46
45
44
43
42
41
40

```

Рис. 4.7: Проверка его работы

Обработка аргументов командной строки

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.

Создал файл lab8-2.asm (рис. 5.3).

```

sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ touch lab8-2.asm
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$

```

Рис. 4.8: Файл lab8-2.asm

Ввел в него текст программы из листинга 8.2.(рис. 5.3).

```

GNU nano 6.2 /home/sunnatillo/work/arch-pc/lab08/lab8-2.asm *
;-----
; Обработка аргументов командной строки
;-----
%include 'ln_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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
            ; аргумента (переход на метку 'next')
_end:
    call quit

```

Рис. 4.9: Текст из листинга 8.2

На исполняемый файл запустил указав аргументы:(рис. 5.3).

```

sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ nasm -f elf lab8-2.asm
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$

```

Рис. 4.10: Аргументы полученные

Рассмотрим еще один пример программы которая выводит сумму чисел, кото-
рые пере- даются в программу как аргументы.

Создал файл lab8-3.asm (рис. 5.3).

```

sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$ touch lab8-3.asm
sunnatillo@sunnatillo-VirtualBox: /work/arch-pc/lab08$

```

Рис. 4.11: файл lab8-3.asm

Ввел на файл lab8-3.asm текст программы из листинга 8.3.(рис. 5.3).

```

GNU nano 6.2 /home/sunnatillo/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,1 ; 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
    imul 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 sprintf
    mov eax,esi ; записываем сумму в регистр 'eax'
    call iprintf ; печать результата
    call quit ; завершение программы

```

Рис. 4.12: Текст из листинга 8.3

Создал исполняемый файл и запустили его, указав аргументы.(рис. 5.3).

```

sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 3 4 12 5
Результат: 24
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 4.13: Результат

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.(рис. 5.3).

```

GNU nano 6.2 /home/sunnatillo/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,1 ; 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
    imul 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 sprintf
    mov eax,esi ; записываем сумму в регистр 'eax'
    call iprintf ; печать результата
    call quit ; завершение программы

```

Рис. 4.14: Измененный текст

Проверял его работу (рис. 5.3).

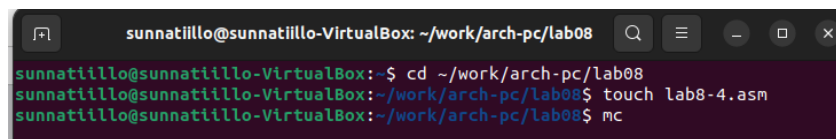
```
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab8$ nasm -f elf lab8-3.asm
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab8$ ./lab8-3 3 4 12 5
Результат: 720
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab8$
```

Рис. 4.15: Результат работы

5 Задание для самостоятельной работы

Вариант 14:

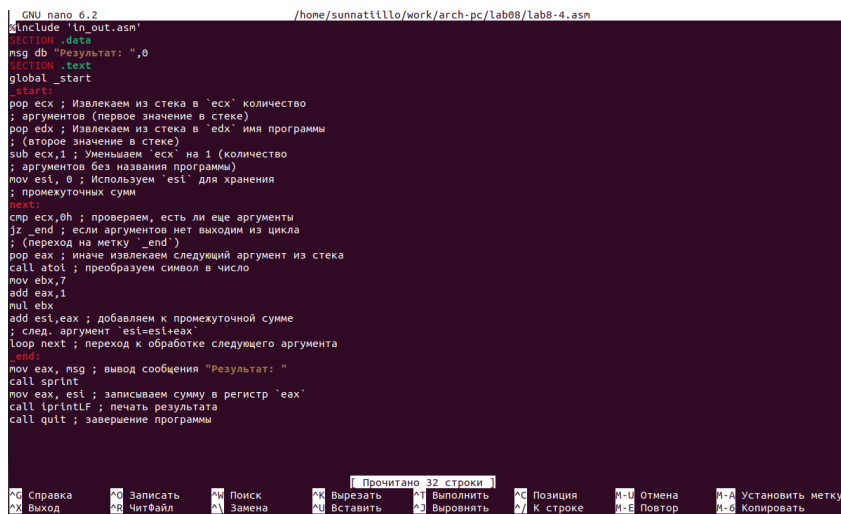
Я создал файл lab8-4.asm чтобы сделать свою вариант :(рис. 5.3).



```
sunnatillo@sunnatillo-VirtualBox: ~/work/arch-pc/lab08
sunnatillo@sunnatillo-VirtualBox:~$ cd ~/work/arch-pc/lab08
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ touch lab8-4.asm
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab08$ mc
```

Рис. 5.1: файл lab8-4.asm

Ввел на файл lab8-4.asm текст из листинга 8.3 и поменял (рис. 5.3).



```
GNU nano 6.2 /home/sunnatillo/work/arch-pc/lab08/lab8-4.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
    mov ebx,7
    add eax,1
    mul ebx
    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 sprintf
    mov eax,esi ; записываем сумму в регистр 'eax'
    call printf ; печать результата
    call quit ; завершение программы
```

Рис. 5.2: Текст

Проверял его работу (рис. 5.3).

```
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab0$ ./lab8-4
Результат: 0
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab0$ ./lab8-4 1 2 3 4
Результат: 98
sunnatillo@sunnatillo-VirtualBox:~/work/arch-pc/lab0$
```

Рис. 5.3: Результат работы

6 Выводы

Я научился писать программы с использованием циклов и 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/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,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).