

Отчёт по лабораторной работе №8

Выполнил студент НКАбд-01-24

Мориссала Донзо

Содержание

Цель работы

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

Задание

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

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.

Выполнение лабораторной работы

Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. -@fig:001).

```
morissaladonzo@box: ~/work/study/2024-2025/Архитектура компьютера/pc/lab08
morissaladonzo@box:~$ mkdir -p /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08
morissaladonzo@box:~$ cd /work/study/2024-2025/Архитектура компьютера/arch-pc/lab08
morissaladonzo@box:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$ touch lab8-1.asm
morissaladonzo@box:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08$
```

Создание каталога

Копирую в созданный файл программу из листинга. (рис. -@fig:002).

```

-----
; 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

```

Копирование программы из листинга

Запускаю программу, она показывает работу циклов в NASM (рис. -@fig:003).

```

nirissaladonzo@box:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab00$ nasm -f elf lab0-1.asm
nirissaladonzo@box:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab00$ ld -m elf_i386 -o lab0-1 lab0-1.o
nirissaladonzo@box:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab00$ ./lab0-1
Введите N: 10
9
8
7
6
5
4
3
2
1
nirissaladonzo@box:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab00$ 

```

Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. -@fig:004).

```
-----  
; 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 aread  
  
; ---- 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 %printlf ; Вывод значения 'N'  
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'  
  
; переход на 'label'  
call quit
```

Изменение программы

Из-за того, что теперь регистр `ecx` на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -@fig:005).

```
nerissa@adonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ nasm -f elf lab8-1.asm  
nerissa@adonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o  
nerissa@adonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ./lab8-1  
Введите N: 10  
9  
7  
5  
3  
1  
nerissa@adonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$
```

Запуск измененной программы

Добавляю команды `push` и `pop` в программу (рис. -@fig:006).

```

lab8-1.asm
/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8

; Программа вывода значений регистра 'еск'
;
#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]

label:
push ecx ; добавление значения еск в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintlf
pop ecx ; извлечение значения еск из стека
loop label

; переход на 'label'
call quit

```

Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. -@fig:007).

```

@
work@saladonozogbox:~$ cd /work/study/2024-2025/Архитектура компьютера/arch-pc/lab8
work@saladonozogbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ nasm -f elf lab8-1.asm
work@saladonozogbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-1 lab8-1.o
work@saladonozogbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0

```

Запуск измененной программы

Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -@fig:008).

```

Ouvrir ▾ ⓘ lab8-2.asm
~/work/study/2024-2025/Архитектура компьютера

%include 'in_out.asm'

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx, 1

next:
cmp ecx, 0
jz _end
pop eax
call sprintf
loop next

_end:
call quit

```

Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. -@fig:009).

```

mariissaladonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ nasm -f elf lab8-2.asm
mariissaladonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-2 lab8-2.o
mariissaladonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
mariissaladonze@ubuntu:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$

```

Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -@fig:010).

```

Oxide ▾ [x] • lab8-3.asm
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$

#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 iprintf ; печать результата

    call quit ; завершение программы

```

Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. -@fig:011).

```

mirtos@ladonzoeybox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ touch lab8-3.asm
mirtos@ladonzoeybox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ nasm -f elf lab8-3.asm
mirtos@ladonzoeybox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
mirtos@ladonzoeybox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ./lab8-3 12 13 7 10 5
Результат: 47

```

Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -@fig:012).

```

lab8-3.asm
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$

#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
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 ; добавляем к промежуточной сумме
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 iprintf ; печать результата
call quit ; завершение программы

```

Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. -@fig:013).

```


mari@saladonozovbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ nasm -f elf lab8-3.asm
mari@saladonozovbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-3 lab8-3.o
mari@saladonozovbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$ ./lab8-3 125 25 2
Результат: 6250
mari@saladonozovbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab8$

```

Запуск измененной третьей программы

Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции $f(x) = 10x - 5$, которая совпадает с моим девытым вариантом (рис. -@fig:014).

```
Ouvrir ▾  lab0-4.asm
~/Tableaux/asm04/

%include "in_out.asm"

SECTION .data
msg_func db "Функция: f(x) = 10x - 5", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 10
mul ebx
sub eax, 5

add esi, eax

loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintf
call quit
```

Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_func db "Функция: f(x) = 10x - 5", 0
```

```
msg_result db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintf
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

```
mov esi, 0
```

```
next:
```

```
cmp ecx, 0h
```

```
jz _end
```

```
pop eax
```

```
call atoi
```



```

mov ebx, 10
mul ebx
sub eax, 5

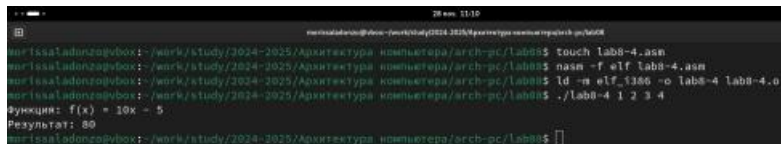
add esi, eax

loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. - @fig:015).



```

root@kali:~/work/study/2024-2025/Архитектура компьютера/arch-rc/lab8$ touch lab8-4.asm
root@kali:~/work/study/2024-2025/Архитектура компьютера/arch-rc/lab8$ nasm -f elf lab8-4.asm
root@kali:~/work/study/2024-2025/Архитектура компьютера/arch-rc/lab8$ ld -m elf_i386 -o lab8-4 lab8-4.o
root@kali:~/work/study/2024-2025/Архитектура компьютера/arch-rc/lab8$ ./lab8-4 1 2 3 4
Функция: f(x) = 10x - 5
Результат: 80
root@kali:~/work/study/2024-2025/Архитектура компьютера/arch-rc/lab8$

```

Запуск программы для самостоятельной работы

Выводы

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

Список литературы

1. [Курс на ТУИС](#)
2. [Лабораторная работа №8](#)