Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Абуков Ислам Ренатович

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

```
irabukov1@islamAbukov:~$ mkdir ~/work/arch-pc/lab08
irabukov1@islamAbukov:~$ cd ~/work/arch-pc/lab08
irabukov1@islamAbukov:~/work/arch-pc/lab08$ touch lab8-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Копирую в созданный файл программу из листинга.

```
 GNU nano 7.2    /home/irabukov1/work/arch-pc/lab08/lab8-1.asm *
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

Запускаю программу, она показывает работу циклов в NASM

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx

```
 GNU nano 7.2     /home/irabukov1/work/arch-pc/lab08/lab8-1.asm *
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]

label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
```

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

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
Ошибка сегментирования (образ памяти сброшен на диск)
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Добавляю команды push и pop в программу

```
 GNU nano 7.2    /home/irabukov1/work/arch-pc/lab08/lab8-1.asm
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx, N
mov edx, 10
call sread

mov eax,N
call atoi
mov [N],eax

mov ecx,[N]

label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
                    [ Прочитана 31 строка ]
^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить
```

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

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o

irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга

```
 GNU nano 7.2     /home/irabukov1/work/arch-pc/lab08/lab8-2.asm *
%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 sprintLF
loop next
_end:
call quit
```

Компилирую программу и запускаю, указав аргументы. Программой было обратоно то же количество аргументов, что и было введено

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-2
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg2 'arg3'
arg1
arg2
arg3
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Создаю новый файл для программы и копирую в него код из третьего листинга

```
 GNU nano 7.2    /home/irabukov1/work/arch-pc/lab08/lab8-3.asm       I
%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
add esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit


                [ Автоотступы включено ]
^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить
^X Выход      ^R ЧитФайл    ^\ Замена     ^U Вставить   ^J Выровнять
```

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции f(x) = 6x + 13, которая совпадает с моим пятнадцатым вариантом

```
 GNU nano 7.2    /home/irabukov1/work/arch-pc/lab08/lab8-4.asm
%include "in_out.asm"

SECTION .data
msg_func   db "Функция: f(x) = 6x + 13", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    ; печать описания функции
    mov eax, msg_func
    call sprintLF

    ; ecx = количество аргументов (без имени программы)
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0          ; сумма значений f(x)

next:
    cmp ecx, 0
    jz _end

    pop eax             ; взять аргумент
    call atoi           ; перевести строку в число, результат в eax

    mov ebx, 6
    mul ebx             ; eax = 6 * x
    add eax, 13         ; eax = 6 * x + 13
                  [ Прочитана 41 строка ]
^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить
^X Выход      ^R ЧитФайл    ^\ Замена     ^U Вставить   ^J Выровнять
```

Проверяю работу программы, указав в качестве аргумента несколько чисел

```
irabukov1@islamAbukov:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция: f(x) = 6x + 13
Результат: 75
irabukov1@islamAbukov:~/work/arch-pc/lab08$
```

Выводы

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