Отчет по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

ФИО: Маркеш Виейра Нанке Грасимилде

1 Цель работы

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

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. 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.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 и файл lab8-1.asm, куда буду копировать программу из листинга. (рис. -fig.4.1).

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

Создание каталога

Копирую в созданный файл программу из листинга. (рис. -fig.4.2).

```
;-------------------------------------------------------------------
; 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 програмлмы из листинга

Запускаю программу, она показывает работу циклов в NASM (рис. -fig.4.3).

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

Запуск програмлмы

Изменяю изначальную программу так, что в теле цикла я изменяю значение регистра ecx (рис. -fig.4.4).

```
%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 iprintLF ; 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.4.5).

```
gracimildevieira@fedora:~/work/arch-pc/lab08$ gedit lab8-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
gracimildevieira@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
gracimildevieira@fedora:~/work/arch-pc/lab08$
```

Запуск измененной программы

Добавляю команды push и pop в программу (рис. -fig.4.6).

Abrir

lab8-2.asm
~/work/arch-pc/lab08

```
%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:
cmpecx, 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
```

Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 ((рис. -fig.4.7).

```
gracimildevieira@fedora:~/work/arch-pc/lab08$ gedit lab8-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
gracimildevieira@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
gracimildevieira@fedora:~/work/arch-pc/lab08$
```

Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -fig.4.8).

```
%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
```

Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обратоно то же количество аргументов, что и было введено (рис. -fig.4.9).

```
gracimildevieira@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ gedit lab8-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
gracimildevieira@fedora:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg 2 'arg 3'
Результат: 2
gracimildevieira@fedora:~/work/arch-pc/lab08$
```

Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -fig.4.10).

Abrir

lab8-3.asm
~/work/arch-pc/lab08

```
 4 msg db "Результат: ",0
 5
 6 SECTION .text
 7 global _start
 8
 9 _start:
10 pop ecx ; Извлекаем из стека в `ecx` количество
11 ; аргументов (первое значение в стеке)
12
13 pop edx ; Извлекаем из стека в `edx` имя программы
14 ; (второе значение в стеке)
15
16 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
17 ; аргументов без названия программы)
18
19 mov esi, 0 ; Используем `esi` для хранения
20 ; промежуточных сумм
21
22 next:
23 cmp ecx,0h ; проверяем, есть ли еще аргументы
24
25 jz _end ; если аргументов нет выходим из цикла
26  ; (переход на метку `_end`)
27
28 pop eax ; иначе извлекаем следующий аргумент из стека
29
30 call atoi; преобразуем символ в число
31
32 add esi,eax
33
34 loop next ; переход к обработке следующего аргумента
35
36 _end:
37 mov eax, msg ; вывод сообщения "Результат: "
38 call sprint
39 mov eax, esi ; записываем сумму в регистр `eax`
40 call iprintLF; печать результата
41 call quit; завершение программы
```

Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. -fig.4.11).

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

Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -fig.4.12).

Abrir

lab8-4.asm
~/work/arch-pc/lab08

```
SECTION .data
msg_func db 'Функция: f(x) = 10(x - 1)', 0h
msg_res  db 'Результат: ', 0h

SECTION .text
global _start

_start:
    ; Показываем функцию
    mov eax, msg_func
    call sprintLF

    pop ecx          ; Извлекаем количество аргументов
    pop eax          ; Извлекаем имя программы
    sub ecx, 1       ; Убираем имя программы из количества

    mov ebx, 0       ; Сумма начинается с 0

process_loop:
    pop eax          ; Извлекаем x_i
    call atoi        ; Преобразуем в число

    ; === f(x) = 10(x - 1) ===
    sub eax, 1       ; x - 1
    mov edx, eax     ; Сохраняем (x-1) в EDX
    add eax, eax     ; 2(x-1)
    add eax, eax     ; 4(x-1)
    add eax, eax     ; 8(x-1)
    add eax, edx     ; 8(x-1) + (x-1) = 9(x-1)
    add eax, edx     ; 9(x-1) + (x-1) = 10(x-1)

    add ebx, eax     ; Добавляем f(x) к общей сумме
    loop process_loop

    ; Показываем результат
    mov eax, msg_res
    call sprint
```

Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. -fig.4.13).

```
gracimildevieira@fedora:~/work/arch-pc/lab08$ gedit lab8-3.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
gracimildevieira@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
gracimildevieira@fedora:~/work/arch-pc/lab08$
```

Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программму, которая будет находить сумма значений для функции f(x) = 10(x - 1), которая совпадает с моим вариантом - 17.

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db 'Функция: f(x) = 10(x - 1)', 0h
msg_res db 'Результат: ', 0h

SECTION .text
global _start

_start:
    ; Показываем функцию
    mov eax, msg_func
    call sprintLF

    pop ecx         ; Извлекаем количество аргументов
    pop eax         ; Извлекаем имя программы
    sub ecx, 1      ; Убираем имя программы из количества

    mov ebx, 0      ; Сумма начинается с 0

process_loop:
    pop eax         ; Извлекаем x_i
    call atoi       ; Преобразуем в число

    ; === f(x) = 10(x - 1) ===
    sub eax, 1      ; x - 1
    mov edx, eax    ; Сохраняем (x-1) в EDX
    add eax, eax    ; 2(x-1)
    add eax, eax    ; 4(x-1)
    add eax, eax    ; 8(x-1)
    add eax, edx    ; 8(x-1) + (x-1) = 9(x-1)
    add eax, edx    ; 9(x-1) + (x-1) = 10(x-1)

    add ebx, eax    ; Добавляем f(x) к общей сумме
    loop process_loop

    ; Показываем результат
    mov eax, msg_res
    call sprint
    mov eax, ebx
    call iprintLF
```

```
call quit
```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. -fig.4.14).

```
gracimildevieira@fedora:~/work/arch-pc/lab08$ gedit lab8-4.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
gracimildevieira@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
gracimildevieira@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: f(x) = 10(x - 1)
Результат: 60
gracimildevieira@fedora:~/work/arch-pc/lab08$
```

Запуск программы для самостоятельной работы

5 Выводы

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

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8