Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Гаязов Рузаль Ильшатович

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

Создаю католог .Копирую в созданный файл программу из листинга. (рис. -@fig:001).

```
 GNU nano 7.2              /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/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:
; ----- 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:002).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-1 lab8-1.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-1
Введите N: 9
9
8
7
6
5
4
3
2
1
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08#
```

Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра ecx (рис. -@fig:003).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# nasm -f elf lab8-1.asm
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-1 lab8-1.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-1
Введите N: 8
7
5
3
1
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08#
```

Изменение программы

Из-за того, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -@fig:004).

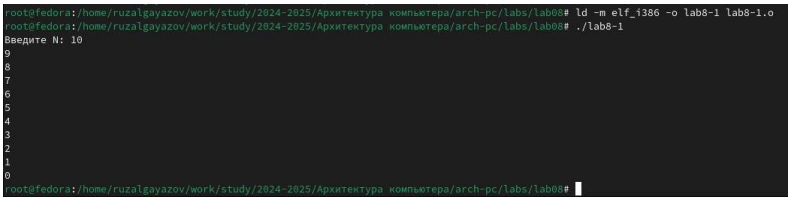
```
 GNU nano 7.2              /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/lab8-1.asm
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call 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 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

Запуск измененной программы

Добавляю команды push и pop в программу (рис. -@fig:005).

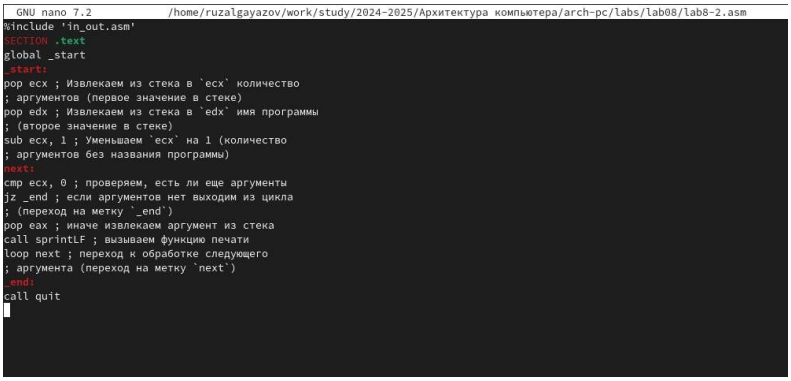
Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. -@fig:006).



Запуск измененной программы

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -@fig:007).



Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обратоно то же количество аргументов, что и было введено (рис. -@fig:008).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# nasm -f elf lab8-2.asm
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-2 lab8-2.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-2
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08#
```

Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -@fig:009).

```
 GNU nano 7.2              /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/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, 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 третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. -@fig:010).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# nasm -f elf lab8-3.asm
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-3 lab8-3.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-3
Результат: 0
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-3 10 20 30 40
Результат: 100
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08#
```

Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -@fig:011).

```
 GNU nano 7.2              /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/lab8-3.asm
%include 'in_out.asm'

SECTION .data
msg db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 1

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

loop next

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

Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. -@fig:012).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# nasm -f elf lab8-3.asm
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-3 lab8-3.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-3 1 111 6
Результат: 666
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# S
```

Запуск измененной третьей программы

Задание для самостоятельной работы

Пишу программму, которая будет находить сумма значений для функции f(x) = 4x+3, которая совпадает с моим девытым варинтом (рис. -@fig:013).

```
 GNU nano 7.2              /home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08/lab8-4.asm
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=4x+3", 10, 0
msg db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprint

    pop ecx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    lea eax, [eax*4+3]
    add esi, eax
    dec ecx
    jmp next

_end:
    mov eax, msg
    call sprint
    mov eax, esi

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo    M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo    M-6 Copy
```

Написание программы для самостоятельной работы

Код программы: %include ‘in_out.asm’

SECTION .data msg_func db “Функция: f(x)=4x+3”, 10, 0 msg db “Результат:”, 0

SECTION .text GLOBAL _start

_start: mov eax, msg_func call sprint

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

next: cmp ecx, 0 jz _end pop eax call atoi lea eax, [eax*4+3] add esi, eax dec ecx jmp next

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

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. -@fig:014).

```
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# nasm -f elf lab8-4.asm
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ld -m elf_i386 -o lab8-4 lab8-4.o
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08# ./lab8-4 1 2 3 4
Функция: f(x)=4x+3
Результат: 36
root@fedora:/home/ruzalgayazov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab08#
```

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
3. Программирование на языке ассемблера NASM Столяров А. В.