

Отчёт лабораторной работы №8

Дисциплина: архитектура коипьютера

Худдыева Дженнет

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Реализация циклов в NASM	9
4.2	Обработка аргументов командной строки	12
4.3	Задание для самостоятельной работы	14
5	Выводы	17

Список иллюстраций

4.1	Создание файлов	9
4.2	Ввод теста	10
4.3	Запуск исполняемого файла	10
4.4	Изменение текста программы	11
4.5	Запуск обновленной программы	11
4.6	Изменение текста программы	11
4.7	Запуск исполняемого файла	12
4.8	Ввод текста программы	12
4.9	Запуск исполняемого файла	13
4.10	Ввод текста программы	13
4.11	Запуск исполняемого файла	14
4.12	Изменение текста программы	14
4.13	Запуск исполняемого файла	14
4.14	Текст программы	15
4.15	Запуск исполняемого файла	16

Список таблиц

1 Цель работы

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

2 Задание

1.Реалтзация циклов 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 “последний 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. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистр 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.

Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд - значение, которое необходимо поместить в стек.

команда pop извлекает значение из стека, т.е. извлекает значения из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. у этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

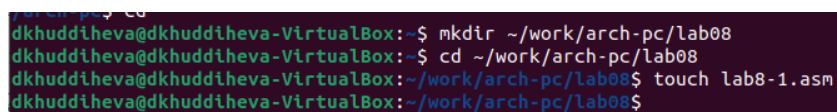
Для организации циклов существуют специальные инструкции. Для всех ин-

струкций максимальное количество проходов задаётся в регистре esx. Наиболее простой является инструкция loor. Она позволяет организовать безусловный цикл.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm (рис. [4.1]).



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

Рис. 4.1: Создание файлов

Ввожу в файл lab8-1.asm текст программы из листинга 8.1 (рис. [4.2]).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5     N: resb 10
6 SECTION .text
7 global _start
8 _start:
9
10 mov eax,msg1
11 call sprint
12 mov ecx, N
13 mov edx, 10
14 call sread
15 mov eax,N
16 call atoi
17 mov [N],eax
18 mov ecx,[N]
19 label:
20 mov [N],ecx
21 mov eax,[N]
22 call iprintLF ; Вывод значения 'N'
23 loop label    ; 'ecx=ecx-1' и 'ecx' не '0'
24 call quit     ; переход на 'label'

```

Рис. 4.2: Ввод теста

Создаю исполняемый файл и проверяю его работу. (рис. [4.3]).

```

dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ gedit lab8-1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
6
5
4
3
2
1
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 4.3: Запуск исполняемого файла

Данная программа выводит числа от N до 1 включительно.

Изменяю текст программы, добавив изменение значения регистра ecx в цикле.
(рис. [4.4]).

```

23 label:
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF ; Вывод значения `N`
28 loop label ; `ecx=ecx-1` и если `ecx` не `0`
29 ; переход на `label`
30 call quit

```

Рис. 4.4: Изменение текста программы

Создаю исполняемый файл и проверяю его работу.(рис. [4.5]).

```

dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ gedit lab8-1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1
1 lab8-1.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 6
5
3
1
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 4.5: Запуск обновленной программы

В данном случае число проходов цикла не соответствует введенному с клавиатуры значению.

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop. (рис. [4.6]).

```

23 label:
24 push ecx
25 sub ecx,1
26 mov [N],ecx
27 mov eax,[N]
28 call iprintLF ; Вывод значения `N`
29 pop ecx
30
31 loop label ; `ecx=ecx-1` и если `ecx` не `0`
32 ; переход на `label`
33 call quit

```

Рис. 4.6: Изменение текста программы

Создаю исполняемый файл и проверяю его работу. (рис. [4.7]).

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

Рис. 4.7: Запуск исполняемого файла

В данном случае число проходов цикла соответствует введенному с клавиатуры значению и выводит числа от N-1 и 0 включительно.

4.2 Обработка аргументов командной строки

Создаю файл lab8-2.asm и ввожу в него текст программы из листинга 8.2. (рис. [4.8]).

```
1 %include 'in_out.asm'
2 SECTION .text
3 global _start
4 _start:
5 pop ecx    ; Извлекаем из стека в `ecx` количество
6            ; аргументов (первое значение в стеке)
7 pop edx    ; Извлекаем из стека в `edx` имя программы
8            ; (второе значение в стеке)
9 sub ecx, 1  ; Уменьшаем `ecx` на 1 (количество
10           ; аргументов без названия программы)
11 next:
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы
13 jz _end    ; если аргументов нет выходим из цикла
14           ; (переход на метку `_end`)
15 pop eax    ; иначе извлекаем аргумент из стека
16 call sprintf ; вызываем функцию печати
17 loop next  ; переход к обработке следующего
18           ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

Рис. 4.8: Ввод текста программы

Создаю исполняемый файл и запускаю его,указав нужные аргументы. (рис. [4.9]).

```
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ gedit lab8-2.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск исполняемого файла

Рассмотрим пример программы,которая выводит сумму чисел, котрые передаются в программу как аргументы. Создаю файл lab8-3.asm и ввожу в него текст программы из листинга 8.3. (рис. [4.10]).

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

Рис. 4.10: Ввод текста программы

Создаю исполняемый файл и запускаю его, указав аргументы. (рис. [4.11]).

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

Рис. 4.11: Запуск исполняемого файла

Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис. [4.12]).

```
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 1 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
```

Рис. 4.12: Изменение текста программы

Создаю исполняемый файл и запускаю его, указав аргументы. (рис. [4.13]).

```
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3
Результат: 1
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 8 8 10
Результат: 640
```

Рис. 4.13: Запуск исполняемого файла

4.3 Задание для самостоятельной работы

Пишу текст программы, которая находит сумму значений функции $f(x)=10(x-1)$ в соответствии с моим номером варианта (17) для $x=x_1, x_2, \dots, x_n$. (рис. [4.14]).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8 pop edx
9 sub ecx,1
10 mov esi, 0
11 mov edi,10
12 next:
13 cmp ecx,0h
14 jz _end
15 pop eax
16 call atoi
17 add eax,-1
18 mul edi
19 add esi,eax
20 loop next
21 _end:
22 mov eax, msg
23 call sprint
24 mov eax, esi
25 call iprintLF
26 call quit
```

Рис. 4.14: Текст программы

Создаю исполняемый файл и проверяю его работу на нескольких наборах

$x=x_1, x_2, \dots, x_n$. (рис. [4.15]).

```
Результат: 120
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf task1.asm
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o task1
task1.o
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./task1 1 2 3
Результат: 30
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$ ./task1 4 5 6
Результат: 120
dkhuddiheva@dkhuddiheva-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск исполняемого файла

5 Выводы

Благодаря данной лабораторной работе я приобрела навыки написания программ использованием циклов командной строки, что поможет мне при выполнении последующих лабораторных работ.