

Отчёта по лабораторной работе №8

Архитектура компьютера

Еремина Оксана Андреевна НКАбд-02-23

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Вывод	16
	Список литературы	17

Список таблиц

Список иллюстраций

4.1	Создание папки и файла	8
4.2	Содержание файла lab8-1.asm	9
4.3	Выподнение программы	9
4.4	Изменение программы	10
4.5	Выподнение программы	10
4.6	Изменение программы	10
4.7	Выподнение программы	11
4.8	Содержание файла lab8-2.asm	11
4.9	Выподнение программы	11
4.10	Содержание файла lab8-3.asm	12
4.11	Выподнение программы	12
4.12	Изменение программы	13
4.13	Выподнение программы	13
4.14	Код прогрраммы	14
4.15	Выподнение программы	14

1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация циклов в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий 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.

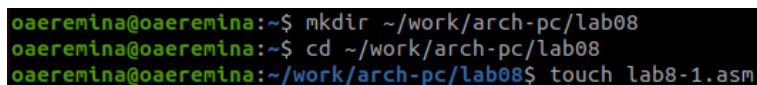
Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop.

4 Выполнение лабораторной работы

1. Реализация циклов в NASM

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



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

Рис. 4.1: Создание папки и файла

Ввожу в файл lab8-1.asm текст программы. (рис.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 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения `N`
26 loop label ; `ecx=ecx-1` и если `ecx` не `0`
27 ; переход на `label`
28 call quit

```

Рис. 4.2: Содержание файла lab8-1.asm

Создаю исполняемый файл и проверяю работу программы. (рис.3)

```

oaaeremina@oaaeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
oaaeremina@oaaeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
oaaeremina@oaaeremina:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
12
11
10
9
8
7
6
5
4
3
2
1

```

Рис. 4.3: Выподнение программы

Изменяю код программы, добавив изменение значение регистра ecx в цикле.
(рис.4)

```

22 label:
23 sub ecx,1 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF ; Вывод значения `N`
27 loop label ; `ecx=ecx-1` и если `ecx` не '0'

```

Рис. 4.4: Изменение программы

Создаю исполняемый файл и проверяю работу программы. (рис.5) Можно заметить, что в первом случае программа уменьшала значения на 1, начиная с вводимого числа (в моем случае 12), во втором же случае на 2 (пропуская вводимое число).

```

oaoeremina@oaoeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
oaoeremina@oaoeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
oaoeremina@oaoeremina:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
9
7
5
3
1

```

Рис. 4.5: Выподнение программы

Изменяю код программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис.6)

```

22 label:
23 push ecx
24 sub ecx,1 ; `ecx=ecx-1`
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF ; Вывод значения `N`
28 pop ecx
29 loop label ; `ecx=ecx-1` и если `ecx` не '0'

```

Рис. 4.6: Изменение программы

Создаю исполняемый файл и проверяю работу программы. (рис.7) Программа уменьшает значения на 1, пропуская вводимое с клавиатуры число.

```

oaeremina@oaeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
oaeremina@oaeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
oaeremina@oaeremina:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
10
9
8
7
6
5
4
3
2
1
0

```

Рис. 4.7: Выподнение программы

2. Обработка аргументов командной строки

Создаю файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Ввожу код программы в созданный файл. (рис.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: Содержание файла lab8-2.asm

Создаю исполняемый файл и провяю его работу для разных аргументов. (рис.9). Программа обработала все аргументы.

```

oaeremina@oaeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
oaeremina@oaeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
oaeremina@oaeremina:~/work/arch-pc/lab08$ ./lab8-2 12 23 45
12
23
45

```

Рис. 4.9: Выподнение программы

Создаю файл lab8-3.asm в каталоге ~/work/arch-pc/lab08. Ввожу код программы в созданный файл. (рис.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`
```

Рис. 4.10: Содержание файла lab8-3.asm

Создаю исполняемый файл и провяю его работу. (рис.11). Программа обработала и сложила все аргументы.

```
oaaeremina@oaaeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
oaaeremina@oaaeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
oaaeremina@oaaeremina:~/work/arch-pc/lab08$ ./lab8-3 10 13 17
Результат: 40
```

Рис. 4.11: Выподнение программы

Изменяю программу так, чтобы она умножала, а не складывала аргументы. (рис.12)

```

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, 1
14 next:
15 cmp ecx,0h ; проверяем, есть ли еще аргументы
16 jz _end ; если аргументов нет выходим из цикла
17 ; (переход на метку `_end`)
18 pop eax ; иначе извлекаем следующий аргумент из стека
19 call atoi ; преобразуем символ в число
20 mul esi
21 mov esi,eax ;
22 loop next ; переход к обработке следующего аргумента
23 _end:
24 mov eax,msg ; вывод сообщения "Результат: "
25 call sprint
26 mov eax, esi ; записываем сумму в регистр `eax`
27 call iprintf ; печать результата
28 call quit ; завершение программы

```

Рис. 4.12: Изменение программы

Создаю исполняемый файл и провяю его работу. (рис.13). Программа обработала и умножила все аргументы.

```

oae@oae:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
oae@oae:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
oae@oae:~/work/arch-pc/lab08$ ./lab8-3 10 13 17
Результат: 2210

```

Рис. 4.13: Выподнение программы

3. Выполнение заданий для самостоятельной работы

Пишу программу которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения переменных выбираю из табл. 8.1. Мой вариант под номером 17, поэтому мой вариант: $10(x - 1)$. (рис.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 sub 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: Код программы

Создаю исполняемый файл и проверяю работу программы, вводя разные значения (рис.15)

```

oаeremina@oаeremina:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
oаeremina@oаeremina:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
oаeremina@oаeremina:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Результат: 30
oаeremina@oаeremina:~/work/arch-pc/lab08$ ./lab8-4 2 4 6
Результат: 90
oаeremina@oаeremina:~/work/arch-pc/lab08$ ./lab8-4 11 21 31
Результат: 600

```

Рис. 4.15: Выподнение программы

Код программы:

```
%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
mov edi,10
next:
cmp ecx,0h
jz _end
pop eax
call atoi
sub eax,1
mul edi
add esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

5 Вывод

При выполнении данной лабораторной работы я приобрела практические навыки написания программ использованием циклов и обработкой аргументов командной строки.

Список литературы

Архитектура ЭВМ