

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Кириянова Екатерина Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Реализация циклов в NASM	7
4.2	Обработка аргументов командной строки	10
4.3	Задание для самостоятельной работы	12
5	Вывод	14
6	Список литературы	15

Список иллюстраций

4.1	Создание	7
4.2	Программа	7
4.3	Запуск	8
4.4	Редактирование	8
4.5	Запуск	9
4.6	Редактирование	9
4.7	Запуск	10
4.8	Создание	10
4.9	Редактирование	10
4.10	Запуск	11
4.11	Создание	11
4.12	Программа	11
4.13	Запуск	11
4.14	Произведение	12
4.15	Запуск	12
4.16	Создание	12
4.17	Программа	13
4.18	Запуск	13

1 Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

1. Реализация 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 «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. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре 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 ror извлекает значение из стека, т.е. извлекает значение 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. Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Инструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

4 Выполнение лабораторной работы

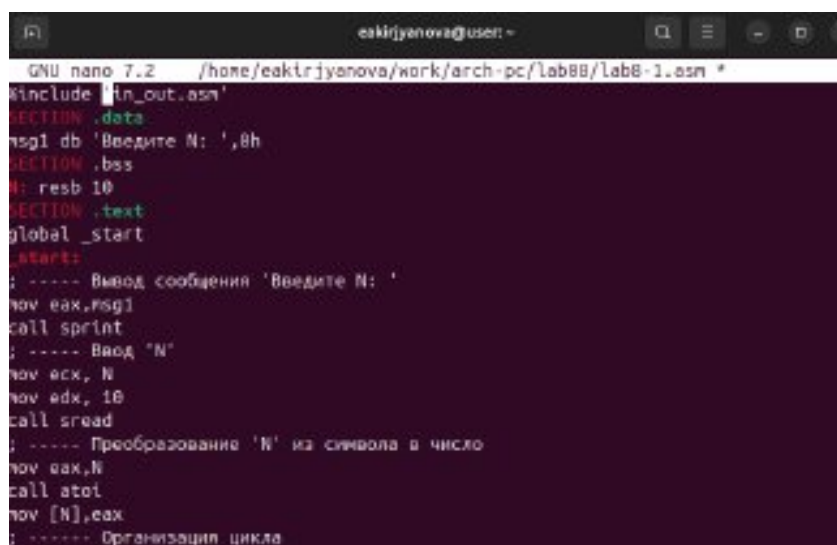
4.1 Реализация циклов в NASM

Создаю новый каталог и файл в нем (рис. 4.1).

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

Рис. 4.1: Создание

Ввожу текст программы из листинга 8.1 (рис. 4.2)



```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab08-1.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
```

Рис. 4.2: Программа

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. 4.3).

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

Рис. 4.3: Запуск

Меняю текст программы, добавив изменение значения регистра `ecx` в цикле (рис. 4.4).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-1.asm *
; ----- 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
```

Рис. 4.4: Редактирование

Создаю обновленный исполняемый файл и запускаю его. Цикл закольцевался и стал бесконечным (рис. 4.5).


```
eakirjyanova@user: ~/work/arch-pc/lab08
eakirjyanova@user:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
eakirjyanova@user:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
eakirjyanova@user:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
2
0
4294967294
4294967292
4294967290
4294967288
4294967286
4294967284
4294967282
4294967280
4294967278
4294967276
4294967274
4294967272
4294967270
4294967268
```

Рис. 4.5: Запуск

Вношу изменения в текст программы, добавив команды push и pop (рис. 4.6).

```
mc [eakirjyanova@user]:~/work/arch-pc/lab08
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-1.asm
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:
push ecx ; добавление значения ecx в стек
sub ecx, 1
mov [N], ecx
mov eax, [N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
loop label
```

Рис. 4.6: Редактирование

Транслирую текст программы в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл. Число проходов цикла стало соответствовать числу, введенному с клавиатуры (рис. 4.7).

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

Рис. 4.7: Запуск

4.2 Обработка аргументов командной строки

Создаю новый файл lab8-2.asm (рис. 4.8).

```
eakirjyanova@user:~/work/arch-pc/lab08$ touch lab8-2.asm
eakirjyanova@user:~/work/arch-pc/lab08$
```

Рис. 4.8: Создание

Ввожу текст программы из листинга 8.2 (рис. 4.9).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-2.asm *
%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:
    cmp ecx, 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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
           ; аргумента (переход на метку next)
_end:
call quit
```

Рис. 4.9: Редактирование

Создаю исполняемый файл и запускаю, указав аргументы. Программа выводит 3 аргумента в разных видах (рис. 4.10).

```
eakirjyanova@user:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
eakirjyanova@user:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
eakirjyanova@user:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 4.10: Запуск

Создаю файл lab8-3.asm (рис. 4.11).

```
eakirjyanova@user:~/work/arch-pc/lab08$ touch lab8-3.asm
eakirjyanova@user:~/work/arch-pc/lab08$
```

Рис. 4.11: Создание

Ввожу текст программы из листинга 8.3 (рис. 4.12).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-3.asm *
por 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)
por 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
```

Рис. 4.12: Программа

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

```
eakirjyanova@user:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
eakirjyanova@user:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
eakirjyanova@user:~/work/arch-pc/lab08$ ./lab8-3 5 27 3
Результат: 35
```

Рис. 4.13: Запуск

Изменяю текст программы для вычисления произведения (рис. 4.14).

```

eakirjyanova@user: ~/work/arch-pc/lab08
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-3.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, 1
    mov eax, 1
next:
    cmp ecx,0 ; 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
    mov ebx,eax
    mov eax,esi
    mul ebx
    mov esi,eax

```

Рис. 4.14: Произведение

Создаю исполняемый файл и запускаю его (рис. 4.15).

```

eakirjyanova@user:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
eakirjyanova@user:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
eakirjyanova@user:~/work/arch-pc/lab08$ ./lab8-3 5 27 3
Результат: 405

```

Рис. 4.15: Запуск

4.3 Задание для самостоятельной работы

Создаю новый файл lab8-4.asm (рис. 4.16).

```

eakirjyanova@user:~/work/arch-pc/lab08$ touch lab8-4.asm
eakirjyanova@user:~/work/arch-pc/lab08$

```

Рис. 4.16: Создание

Пишу программу, которая находит сумму значений функций. Вид функции выбираю согласно 2 варианту (рис. 4.17).

```

eakirjyanova@user: ~/work/arch-pc/lab08
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab08/lab8-4.asm
%include 'in_out.asm'

SECTION .data
prim DB 'f(x)=3x-1',0
otv DB 'Результат: ',0

SECTION .bss
sum resd 1

SECTION .text
GLOBAL _start

_start:
    mov eax, [esp]
    sub eax, 1
    mov ecx, eax

    mov dword [sum], 0

next:
    cmp ecx, 0
    jz _end

    mov eax, [esp + ecx * 4 + 4]
    call atoi

```

Рис. 4.17: Программа

Создаю исполняемый файл и запускаю. Программа работает корректно (рис. 4.18).

```

eakirjyanova@user:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
eakirjyanova@user:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
eakirjyanova@user:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Результат: 26

```

Рис. 4.18: Запуск

5 Вывод

В ходе выполнения данной лабораторной работы я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки.

6 Список литературы

1. Лабораторная работа №8