

Лабораторная работа № 9

Дисциплина: Архитектура компьютера

Никулина Ксения Ильинична

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7

Список иллюстраций

4.1	Создание файла	7
4.2	Текст	8
4.3	Проверка файлов	8
4.4	Изменение текста	9
4.5	Работа файла	9
4.6	Работа файла	10
4.7	Создала файл	10
4.8	Текст из листига 9.2	11
4.9	Запуск файла	11
4.10	Создала файл	12
4.11	Текст из листига 9.3	12
4.12	Запуск файла	12
4.13	Текст	13
4.14	Проверка файлов	13
4.15	Текст	14
4.16	Запуск файла	14

1 Цель работы

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

2 Задание

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

3 Теоретическое введение

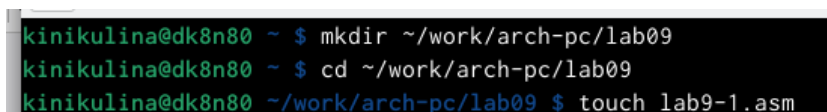
Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией 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.

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.

Иструкция `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

1. Создала каталог для программ лабораторной работы No 9, перешла в него и создайте файл lab9-1.asm: (рис. 4.1)



```
kinikulina@dk8n80 ~ $ mkdir ~/work/arch-pc/lab09
kinikulina@dk8n80 ~ $ cd ~/work/arch-pc/lab09
kinikulina@dk8n80 ~/work/arch-pc/lab09 $ touch lab9-1.asm
```

Рис. 4.1: Создание файла

2. Ввела в файл lab9-1.asm текст программы из листинга 9.1. Создала исполняемый файл и проверила его работу(рис. 4.2,рис. 4.3)

```
kinikulina@dk8n80:~/work/arch-pc/lab09  mc [kinikulina@dk8n80]:~/work/arch-pc/
/afs/.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/
%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
[ Прочитано 29 строк ]
```

Рис. 4.2: Текст

```
kinikulina@dk8n80 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
kinikulina@dk8n80 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
kinikulina@dk8n80 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 24
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
```

Рис. 4.3: Проверка файлов

3. Изменила текст программы добавив изменение значение регистра ecx в

цикле (рис. 4.4)

```
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
```

□

^G Справка ^O Записать ^W Поиск ^K Вырезать

Рис. 4.4: Изменение текста

4. Создала исполняемый файл и проверила его работ (рис. 4.5)

```
lab9-1.asm:1: error: label or instruction expected at start of line
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 24
23
21
19
17
15
13
11
9
7
5
3
1
```

kinikulina@dk5n53 ~/work/arch-pc/lab09 \$

Рис. 4.5: Работа файла

5. Внесла изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop`. Создала исполняемый файл и проверила его работу. (рис. 4.6)

```

kinikulina@dk5n53 ~/work/arch-pc/lab09 $ nasm -f elf lab9-1.asm
kinikulina@dk5n53 ~/work/arch-pc/lab09 $
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./lab9-1
Введите N: 24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

```

Рис. 4.6: Работа файла

6. Создайте файл lab9-2.asm в каталоге ~/work/arch-pc/lab09 и введите в него текст программы из листинга 9.2. (рис. 4.7,рис. 4.8)

```

0
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ touch lab9-2.asm
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ 

```

Рис. 4.7: Создала файл

```

...fu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab09/lab9-2
%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. 4.8: Текст из листига 9.2

7. Создала исполняемый файл и запустила его, указав аргументы:
 user@dk4n31:~\$./lab9-2 аргумент1 аргумент 2 'аргумент 3' (рис. 4.9)

```

kinikulina@dk5n53 ~/work/arch-pc/lab09 $ touch lab9-2.asm
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 4.9: Запуск файла

9. Создайте файл lab9-3.asm в каталоге ~/work/arch-pc/lab09 и введите в него текст программы из листинга 9.3 (рис. 4.10,рис. 4.11)

```
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ touch lab9-3.asm
```

Рис. 4.10: Создала файл

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в 'ecx' количество
; аргументов (первое значение в стеке)
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
```

Рис. 4.11: Текст из листига 9.3

10. Создала исполняемый файл и запустила его, указав аргументы (рис. 4.12)

```
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./lab9-3 12 13 7 10 5
Результат: 47
kinikulina@dk5n53 ~/work/arch-pc/lab09 $
```

Рис. 4.12: Запуск файла

11. Изменила текст программы из листинга 9.3 для вычисления произведения аргументов командной строки. Создала исполняемый файл и запустила его, указав аргументы. рис. 4.13,рис. 4.14)

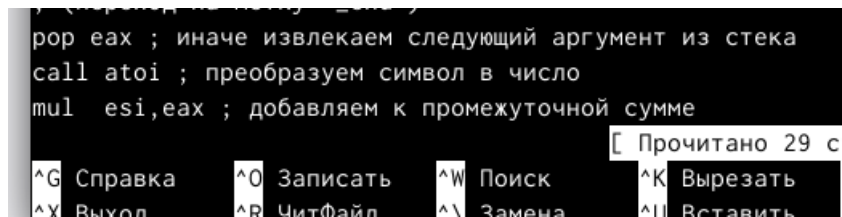


Рис. 4.13: Текст

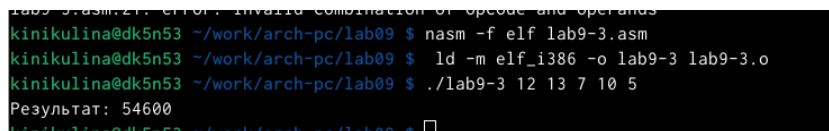


Рис. 4.14: Проверка файлов

#Самостоятельная работа

1. Написала программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$ ($f(x) = 7 + 2x$) (рис. 4.15)

```

GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/k/i/kinikulina/work/arch-pc/lab09/sam_
%include 'in_out.asm'

SECTION .data
prim DB 'f(x) = 7 + 2x',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end

mov ebx,2
pop eax
call atoi
mul ebx

add eax, 7

add esi, eax
loop next

_end:
mov eax,otv
call sprintf
mov eax,esi
call iprintLF

```

Рис. 4.15: Текст

2. Создала исполняемый файл и запустила его, указав аргументы (рис. 4.16)

```

kinikulina@dk5n53 ~/work/arch-pc/lab09 $ nasm -f elf sam_rabota.asm
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o sam_rabota sam_rabota.o
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./sam_rabota 1 2 3 4
f(x) = 7 + 2x
Результат: 48
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./sam_rabota 1 2 3 4 5 6
f(x) = 7 + 2x
Результат: 84
kinikulina@dk5n53 ~/work/arch-pc/lab09 $ ./sam_rabota 1 2 3 4 5 6 7 8
f(x) = 7 + 2x
Результат: 128

```

Рис. 4.16: Запуск файла

#Выводы Я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки