

Лабораторная работа №8

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Задание для самостоятельной работы	12
5	Листинг	14
6	Выводы	16

Список иллюстраций

3.1	Создание каталога	7
3.2	Текст файла	8
3.3	Вывод программы	8
3.4	Результат измененного файла	9
3.5	Текст файла	9
3.6	Вывод программы	9
3.7	Файл lab8-3.asm	10
3.8	Вывод программы	10
3.9	Измененный текст	11
3.10	Вывод программы	11
4.1	Текст программы	12
4.2	Вывод программы	13

Список таблиц

1 Цель работы

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

2 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. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №8 и создаю файл lab8-1.asm (рис. 3.1).

```
agovchinnikov@dk8n75 ~ $ mkdir ~/work/arch-pc/lab08
agovchinnikov@dk8n75 ~ $ cd ~/work/arch-pc/lab08
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ touch lab8-1.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Создание каталога

Ввожу в файл текст из листинга 8.1 (рис. 3.2).

```

lab8-1.asm      [-M--]  9 L:[ 3+25 28/ 28] *(636 / 636b) <EOF>
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

```

Рис. 3.2: Текст файла

Создаю исполняемый файл и проверяю его работу (рис. 3.3). В данном случае число проходов цикла не соответствует значению N введенному с клавиатуры.

```

agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 

```

Рис. 3.3: Вывод программы

Меняю текст листинга файла и проверяю результат программы (рис. 3.4). В данном случае число проходов цикла соответствует значению N введенному с клавиатуры.


```

agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 3
2
1
0

```

Рис. 3.4: Результат измененного файла

Создаю файл lab8-2.asm и ввожу туда текст из листинга 8.2 (рис. 3.5).

```

lab8-2.asm      [----]  9 L:[  1+19  20/ 20] *(943 / 943b) <EOF>
#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. 3.5: Текст файла

Создаю исполняемый файл и проверяю его работу (рис. 3.6).

```

agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-2
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рис. 3.6: Вывод программы

Создаю файл lab8-3.asm и ввожу в него текст программы из листинга 8.3 (рис. 3.7)

```

lab8-3.asm      [-M--] 32 L:[ 1+18 19/ 29] *(861 /1427b) 1097 0x4
#include "in_out.asm"
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
or 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

```

Рис. 3.7: Файл lab8-3.asm

Создаю исполняемый файл и запускаю его, при этом указывая аргументы (рис. 3.8)

```

agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
bash: ./main: Нет такого файла или каталога
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 47

```

Рис. 3.8: Вывод программы

Изменяю текст программы из листинга для вычисления произведения аргументов командной строки (рис. 3.9)

```

lab8-3.asm      [-M--] 12 L: [ 3+20 23/ 30] *(1089/1437b) 0010 0x00A
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 ; 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
mul esi ; добавляем к промежуточной сумме
; след. аргумент 'esi=esi+eax'
mov 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 iprintf ; печать результата

```

Рис. 3.9: Измененный текст

Создаю исполняемый файл и проверяю его работу (рис. 3.10).

```

agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-3.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./main 12 13 7 10 5
Результат: 54600

```

Рис. 3.10: Вывод программы

4 Задание для самостоятельной работы

Вариант 15

Пишу программу, которая находит сумму значений функции $f(x)$ (рис. 4.1).

```
lab8-4.asm          [----] 9 L:[ 1+24 25/ 35] *(274 / 370b) 0032 0x020
#include "io.h"
SECTION .data
f_x db "f(x)=6x+13",0h
msg db 10,13, "result: ",0h

Section .text
global _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx, 6
mul ebx
add eax, 13
add esi, eax

loop next

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

call quit
```

Рис. 4.1: Текст программы

Создаю исполняемый файл и проверяю его работу (рис. 4.2).

```
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o main lab8-4.o
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $ ./main 1 2 3 4
f(x)=6x+13
результат: 112
agovchinnikov@dk8n75 ~/work/arch-pc/lab08 $
```

Рис. 4.2: Вывод программы

5 Листинг

```
%include 'in_out.asm'
SECTION .data
f_x db "f(x)=6x+13",0h
msg db 10,13, 'результат: ',0h

Section .text
global _start

_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx, 6
mul ebx
add eax, 13
```

```
add esi, eax
```

```
loop next
```

```
_end:
```

```
mov eax, f_x
```

```
call sprint
```

```
mov eax, msg
```

```
call sprint
```

```
mov eax, esi
```

```
call iprintLF
```

```
call quit
```

6 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки