РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

по лабораторной работе № 8

дисциплина: Архитектура компьютера

Студент: Сачковская София Александровна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

Список иллюстраций:

1.Цель работы

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

2. Теоретическое введение

Организация стека

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. 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).

Добавление элемента в стек.

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.

Примеры:

push-10; Поместить-10 в стек

push ebx; Поместить значение регистра ebx в стек

push [buf];Поместить значение переменной buf в стек

push word [ax] ; Поместить в стек слово по адресу в ax

Существует ещё две команды для добавления значений в стек. Это команда push a,которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: ax, cx, dx, bx, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов.Обе эти команды не имеют операндов.

Извлечение элемента из стека.

Команда pop 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. Нужно помнить,что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек. Примеры:

pop eax ; Поместить значение из стека в регистр eax

pop [buf] ; Поместить значение из стека в buf pop

word[si] ; Поместить значение из стека в слово по адресу в si

Аналогично команде записи в стек существует команда pop a,которая восстанавливает из стека все регистры общего назначения,и команда popf для перемещения значений из вершины стека в регистр флагов.

Инструкции организации циклов

Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx.Наиболее простой является инструкция loop.Она позволяет организовать безусловный цикл,типичная структура которого имеет следующий вид:

```
mov  ecx, 100 ; Количество проходов

NextStep:

 ...

... ; тело цикла

...

loop NextStep ; Повторить `ecx` раз от метки NextStep
```

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.

3.Выполнение лабораторной работы

Создадим каталог для программ лабораторной работы №8,перейдем в него и создадим файл lab8-1.asm:

```
sachkovskayasofiya@sachkovskayasofiya:~$ mkdir ~/work/arch-pc/lab08
sachkovskayasofiya@sachkovskayasofiya:~$ cd ~/work/arch-pc/lab08
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$
```

Рис.3.1

Введем в файл lab8-1.asm текст программы из листинга 8.1.Создадим исполняемый файл и проверим его работу.

```
/home/sachkovskayasofiya/work/arch-pc/lab08/lab8-1.asm   [-M--] 24 L:
;-------------------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------------------
%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
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
```

```
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ nasm -f elf lab8-1.a
sm
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ld -m elf_i386 -o la
b8-1 lab8-1.o
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$
```

Рис 3.2,3.3

Изменим текст программы добавив изменение значение регистра ecx в цикле:

```
/home/sachkovskayasofiya/work/arch-pc/lab08/lab8-1.asm   [----]  9 L:
;-------------------------------------------------------------------
; Программа вывода значений регистра 'ecx'
;-------------------------------------------------------------------
%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
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
```

Рис.3.4

Создадим исполняемый файл и проверим его работу.Какие значения принимает регистр ecx в цикле?Соответствует ли число проходов цикла значению N введенному с клавиатуры?

Ответ:Если N=10, регистр ecx будет принимать следующие значения: **9, 7, 5, 3, 1.** Число проходов цикла не соответствует значению N введенному с клавиатуры

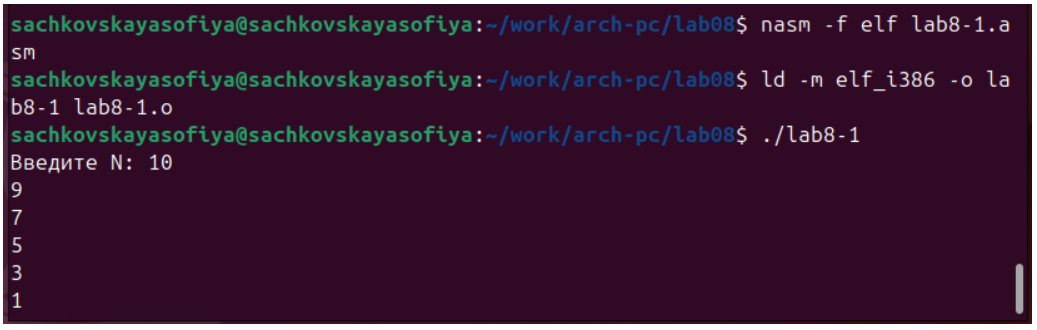

Рис 3.5

Внесем изменения в текст программы добавив команды push и pop(добавления в стек и извлечения из стека)для сохранения значения счетчика цикла loop:Создадим исполняемый файл и проверим его работу.Соответствует ли в данном случае число проходов цикла значению N введенному с клавиатуры?

Ответ:Да,соответствует

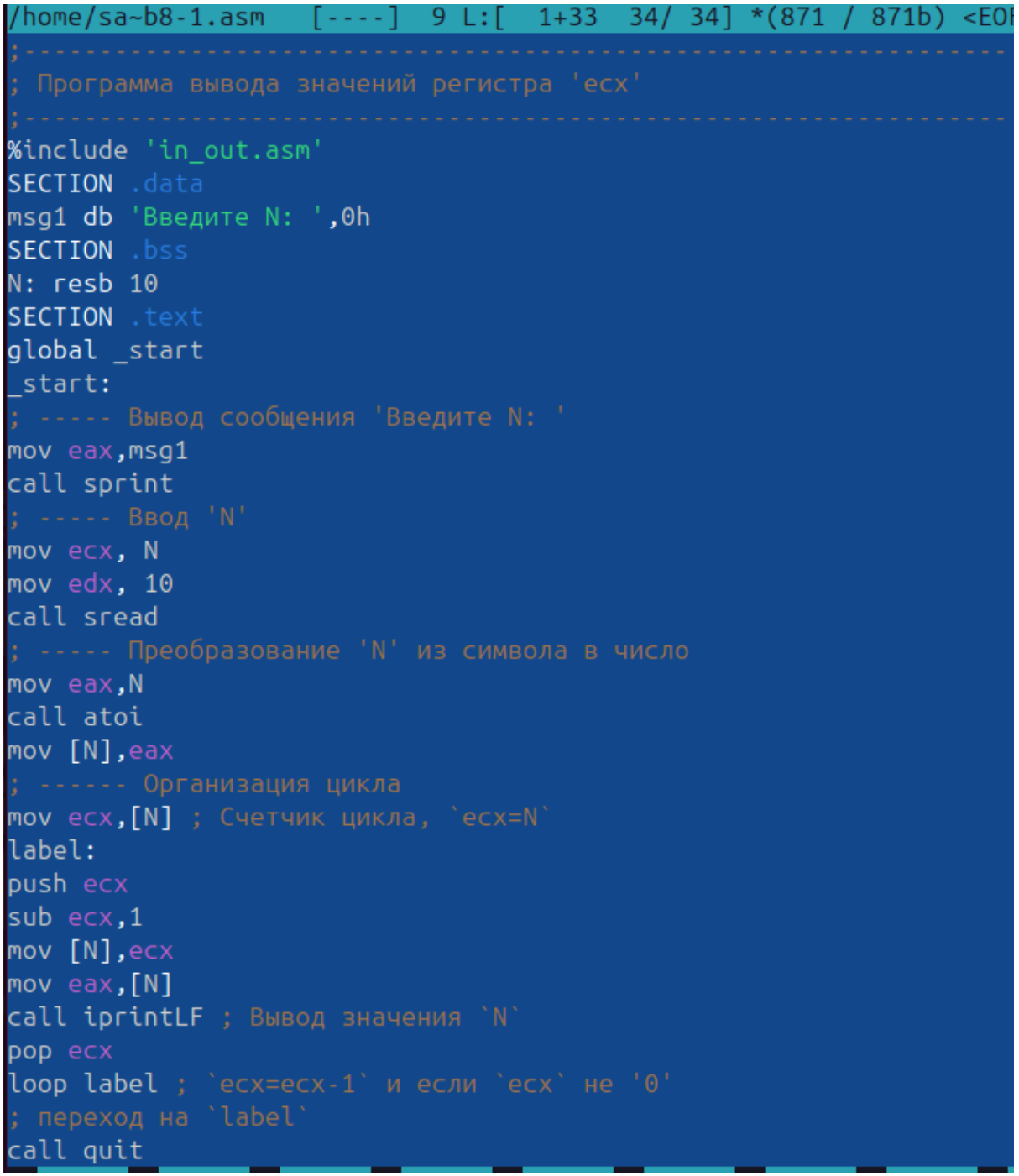
/home/sa~b8-1.asm [----] 9 L:[1+33 34/ 34] *(871 / 871b) <EOF
;---
; Программа вывода значений регистра 'ecx'
;---
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- 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
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit

Рис.3.6

```
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ nasm -f elf lab8-1.a
sm
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ld -m elf_i386 -o la
b8-1 lab8-1.o
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$
```

Рис.3.7

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы из листинга 8.2. Создадим исполняемый файл и запустим его,указав аргументы.Сколько аргументов было обработано программой?

```
/home/sa~b8-2.asm   [----]  9 L:[  1+22  23/ 23] *(1151/1151b) <EOF
;-------------------------------------------------------------------
; Обработка аргументов командной строки
;-------------------------------------------------------------------
%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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

```
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ nasm -f elf lab8-2.a
sm
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ld -m elf_i386 -o la
b8-2 lab8-2.o
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 а
ргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$
```

Ответ:4 аргумента

Рис.3.8,3.9

Создадим файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введем в него текст программы из листинга 8.3.

```
/home/sa~b8-3.asm   [-M--] 32 L:[  1+28  29/ 29] *(1428/
%include 'in_out.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, 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
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
```

Рис.3.10

Создадим исполняемый файл и запустим его,указав аргументы.

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


sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ nasm -f elf lab8-3.a
sm
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ld -m elf_i386 -o la
b8-3 lab8-3.o
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10
5
Результат: 47
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$
```

Рис.3.11

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```
/home/sa~b8-3.asm   [----] 32 L:[  1+28  29/ 29] *(1429
%include 'in_out.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, 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
imul 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
```

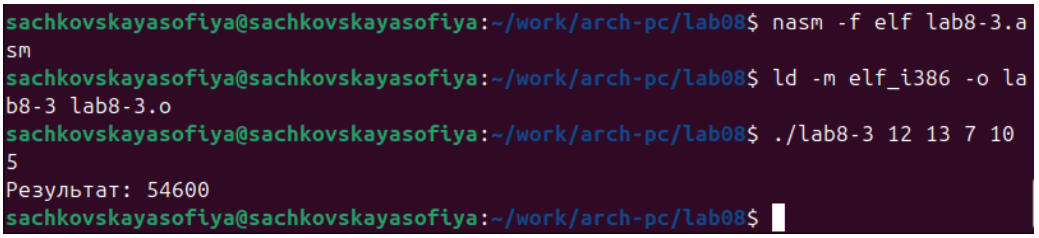

Рис.3.12,3.13

Вывод: выполнили задания лабораторной работы.

4.Выполнение самостоятельной работы

1. Напишите программу, которая находит сумму значений функции $f(x)$ для x =$x1,x2,...,xn$,т.е. программа должна выводить значение $f(x1)+f(x2)+...+f(xn)$. Значения xi передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = $x1,x2,...,xn$.

Вариант:11

Ответ:

```
/home/sa~main.asm   [----]
%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

next:
cmp ecx, 0
jz _end
pop eax
call atoi
mov ebx, 15
mul ebx
add eax, 2
add esi, eax
loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

```
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ nasm -f elf main.asm
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ld -m elf_i386 -o ma
in main.o
sachkovskayasofiya@sachkovskayasofiya:~/work/arch-pc/lab08$ ./main 1 2 3 4
Результат: 158
```

Рис.4.1,4.2

Вывод:Выполнили задания самостоятельной работы

5.Выводы

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