Лабораторная работа №9. Программирование цикла. Обработка

Дисциплина: Архитектура ЭВМ

Плескачева Елизавета Андреевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработка аргументов коммандной строки.

2 Выполнение лабораторной работы

2.1 Реализация циклов

Создадим каталог для выполнения лабораторной, перейдем в него

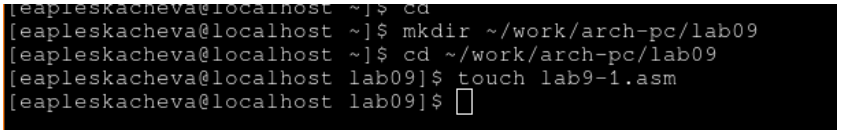

Рис. 2.1: Создание папки и файла

введем листинг 9.1 в файл

```
%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

```

Рис. 2.2: Листинг 9.1 в lab9-1.asm

Скомпилируем и запустим код. Проверим его.

```
[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1

Введите N: 10
10
9
8
7
6
5
4
3
2
1
[eapleskacheva@localhost lab09]$
```

Рис. 2.3: Компиляция и запуск lab9-1

Программа выводит числа от 10 до 1

Добавим строчку `sub ecx, 1` в программу

```
30
31    sub ecx, 1
32    mov [N],ecx
33    mov eax,[N]
```

Рис. 2.4: Добавление строчки

Скомпилируем и запустим.

```
[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1

Введите N: 10
9
7
5
3
1
[eapleskacheva@localhost lab09]$
```

Рис. 2.5: Измененный вывод программы

Теперь программа выводит числа 9 7 5 3 1

ЧТо бы программа работала как раньше, но начинала с 9, обернем код в push pop

```
29 label:
30
31   push ecx
32   sub ecx, 1
33   mov [N],ecx
34   mov eax,[N]
35   call iprintLF ; Вывод значения `N`
36   pop ecx
```

Рис. 2.6: Изменение кода lab9-1.asm

Скомпилируем и запустим

```
[eapleskacheva@localhost lab09]$ nasm -f elf ./lab9-1.asm
ld -m elf_i386 -o ./lab9-1 ./lab9-1.o
./lab9-1

Введите N: 10
9
8
7
6
5
4
3
2
1
0
[eapleskacheva@localhost lab09]$
```

Рис. 2.7: Вывод измененной програмы

Теперь программа выводит числа от 9 до 0

2.2 Обработка аргументов коммандной строки

Введем в lab9-2.asm листинг 9.2

```
%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 sprintLF ; вызываем функцию печати
  loop next ; переход к обработке следующего
            ; аргумента (переход на метку `next`)
_end:
  call quit

```

Рис. 2.8: Текст в lab9-2.asm

Скомпилируем и запустим програму указав аргументы

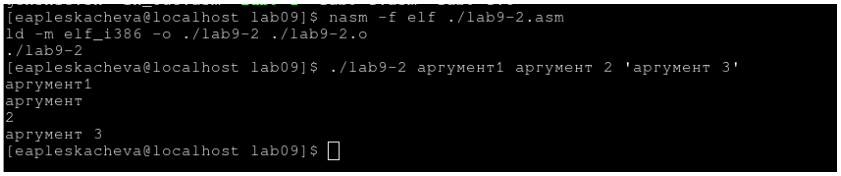

Рис. 2.9: Запуск lab9-2 с аргументами

Программа Выводит аргументы последовательно на экран. Программа выводит 4 аргумента

2.2.1 Выведение суммы аргументов

Введем код из листинга 9.3 в lab9-3, скомпилируем и запустим его

Введем аргументы 12 13 7 10 5

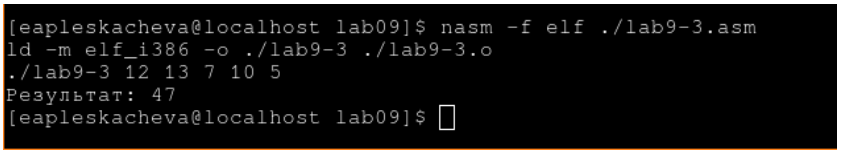

Рис. 2.10: Запуск кода lab9-3

Программа вывела сумму аргументов - 47

2.2.2 Выведение произведения аргументов

Изменим программу lab9-3.asm так, что бы она умножала аргументы. Сохраним измененную прогармму в lab9-3-mult.asm

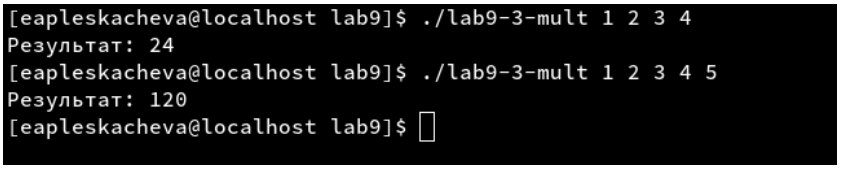

Рис. 2.11: Измененная часть программы

Скомпилируем и запустим программу, проверим ее на несокльких аргументах

```
21
22  push edx
23  push eax
24  mov edx, eax
25  mov eax, esi
26
27  mul edx ; добавляем к промежуточной сумме
28  mov esi, eax
29
30  pop eax
31  pop edx
32              ; след. аргумент `esi=esi+eax`
33  loop next ; переход к обработке следующего аргумента
34
```

Рис. 2.12: Запуск кода lab9-3-mult

Программа выводит правильный результат

3 Задания для самостоятельной работы

Мой вариант - 2, поэтому пишем программу, которая выводит сумму результатов функции `f(x) = 3x - 1`

```
%include 'in_out.asm'

SECTION .data
func db "f(x) = 3x - 1: ",0
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
  mov eax, func

  call sprintLF


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




  push ecx

  mov ecx,3
  mul ecx
  dec eax

  pop ecx







  add esi,eax ; добавляем к промежуточной сумме

  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. 3.1: Программа выводящая сумму результатов функций

Запустим программу и проверим ее на аргументах 1 2 3 4

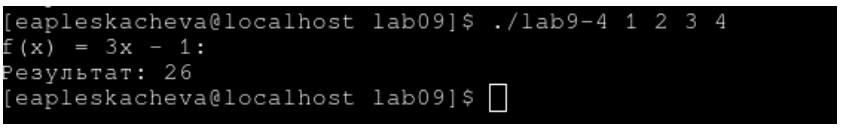

Рис. 3.2: Вывод программы lab9-4

Результат верный

4 Выводы

Мы приобрели навыки написания программ с использованием циклов и научились обрабатывать агументы коммандной строки.