Шаблон отчёта по лабораторной работе 08

Простейший вариант

Абдуллахи шугофа

Содержание

1 Цель работы

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

2 Задание

3 Порядок выполнения лабораторной работы

1. Реализация переходов в NASM

1.1 Создайте каталог для программам лабораторной работы No 8, перейдите в него и создайте файл lab8-1.asm:

```
shogofa@shogofa-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
shogofa@shogofa-VirtualBox:~$ cd ~/work/arch-pc/lab08
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$
```

Figure 1:

1.2 Листинг 8.1. Программа вывода значений регистра ecx

```
;-------------------------------------------------------------------
; 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
```

Figure 2:

1.3 Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.

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

Figure 3:

1.4 Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра ecx в цикле:

```
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 ; `ecx=ecx-1`
 mov [N],ecx
 mov eax,[N]
 call iprintLF
 loop label
                                    ; переход на `label`
call   quit
```

Figure 4:

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

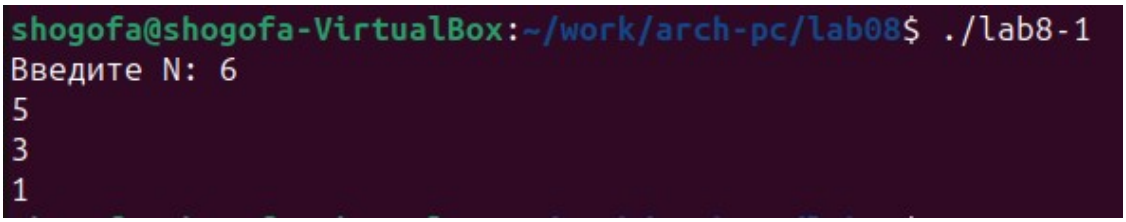

Figure 5:

1.6 Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

```
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                       ; 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
                                          ; переход на `label`
```

Figure 6:

1.7 Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению *N* введенному с клавиатуры?

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

Figure 7:

2. Листинг 8.2. Программа выводящая на экран аргументы командной строки

2.1 Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст про- граммы из листинга 8.2.

```
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ touch lab8-2.asm
;-------------------------------------------------------------------
; 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
```

2.2 Создайте исполняемый файл и запустите его, указав аргументы:

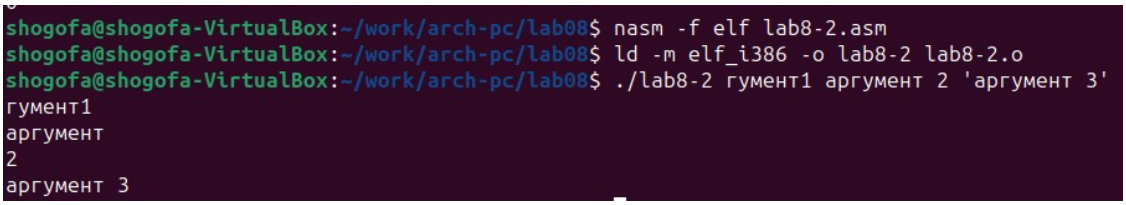

Figure 8:

3. Листинг 8.3. Программа вычисления суммы аргументов командной строки

3.1 Рассмотрим еще один пример программы которая выводит сумму чисел, которые пере- даются в программу как аргументы. Создайте файл lab8-3.asm в каталоге ~/work/arch- pc/lab08 и введите в него текст программы из листинга 8.3.

Figure 9:

```
%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
```

Figure 10:

3.2 Создайте исполняемый файл и запустите его, указав аргументы. Пример результата работы программы: user@dk4n31:~$ ./main 12 13 7 10 5 Результат: 47 user@dk4n31:~$

```
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Figure 11:

4 Задание для самостоятельной работы

1. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, ..., x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + ... + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы No 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, ..., x_n$. Пример работы программы для функции $f(x) = x + 2$ и набора $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$: user@dk4n31:~$ ./main 1 2 3 4 Функция: f(x)=x+2 Результат: 18 user@dk4n31:~$ Вариант 6:

```
shogofa@shogofa-VirtualBox:~/work/arch-pc/lab08$ touch lab8-hw.asm
```

Figure 12:

```
   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
   mov ebx,4
   mul ebx
   sub eax,3     ; добавляем к промежуточной сумме
   add  esi,eax
                 ; 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
```

Figure 13:

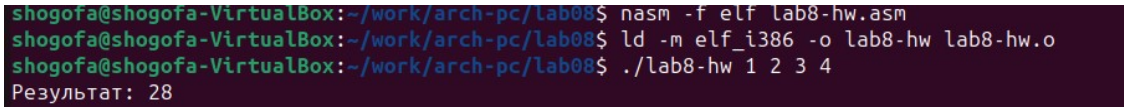

Figure 14:

5 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 1 приведено краткое описание стандартных каталогов Unix.

Table 1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

6 Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. ??).

Название рисунка

7 Выводы

Здесь кратко описываются итоги проделанной работы.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: https://www.gnu.org/software/bash/manual/.

2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 c.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 c.

4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 c.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.