Отчет по лабораторной работе №8

Архитектура компьютера

Дмитрий Константинович Кобзев

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

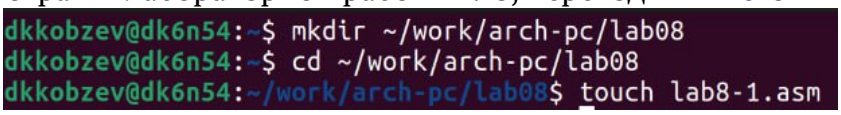
2 Задание

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.

3 Выполнение лабораторной работы |

[1–6]

Создаем каталог для программ лабораторной работы №8, переходим в него и создаем файл lab8-1.asm:.



Вводим в файл lab8-1.asm текст программы из листинга 8.1. Создаем исполняе-

```
%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 симв
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `e
; переход на `label`
call quit
```

мый файл и проверяем его работу (рис. 1.2), (рис. 1.3).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Измените текст программы добавив изменение значение регистра ecx в цикле

```
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
```

(рис. 1.4).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-1
```

Создаем исполняемый файл и запускаем его (рис. 1.5).

Вносим изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop

```
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
```

(рис. 1.6).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
```

Создаем исполняемый файл и запускаем его (рис. 1.7).

Создаем файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.2. (рис. 1.8).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ touch lab8-2.asm
```

```
%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
```

Создаем исполняемый файл и запускаем его, указав аргументы (рис. 1.10).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
dkkobzev@dk6n54:~/work/arch-pc/lab08$  ld -m elf_i386 -o lab8-2 lab8-2.o
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-2 1 2 3
1
2
3
```

Создаем файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и вводим в него текст программы из листинга 8.3. (рис. 1.11). `dkkobzev@dk6n54:~/work/arch-pc/lab08$ touch lab8-3.asm`

```
%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
```

Создаем исполняемый файл и запускаем его, указав аргументы (рис. 1.13).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Изменяем текст программы из листинга 8.3 для вычисления произведения аргу-

```
%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, 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
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
```

ментов командной строки. (рис. 1.11).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
```

4 Самостоятельная работа

Задание 1. Пишем 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. Создаем исполняемый файл и проверяем его работу на нескольких наборах x = x1,

```
%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
sub esi,1
imul esi,10
_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
```

x2, ..., xn. (рис. 2.1).

```
dkkobzev@dk6n54:~/work/arch-pc/lab08$ nasm -f elf sr.asm
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ld -m elf_i386 -o sr sr.o
dkkobzev@dk6n54:~/work/arch-pc/lab08$ ./sr 1 2 3
Результат: 50
```

5 Выводы

В ходе выполнения лабораторной работы мною были приобретены навыки написания программ с использованием циклов и обработки аргументов командной строки.

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 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
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.