

Отчёт по лабораторной работе 4

дисциплина: Архитектура компьютера

Магомедов Султан Гасанович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Программа Hello world!	6
2.2	Трансляция кода с помощью NASM	7
2.3	Линковка с использованием LD	8
2.4	Выполнение заданий для самостоятельной работы	9
2.5	Выводы	11

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Программа hello.asm	7
2.3	Трансляция hello.asm	8
2.4	Трансляция hello.asm с дополнительными опциями	8
2.5	Линковка программы	8
2.6	Линковка программы	9
2.7	Запуск программ	9
2.8	Код программы в файле lab4.asm	10
2.9	Запуск программы lab4.asm	10

Список таблиц

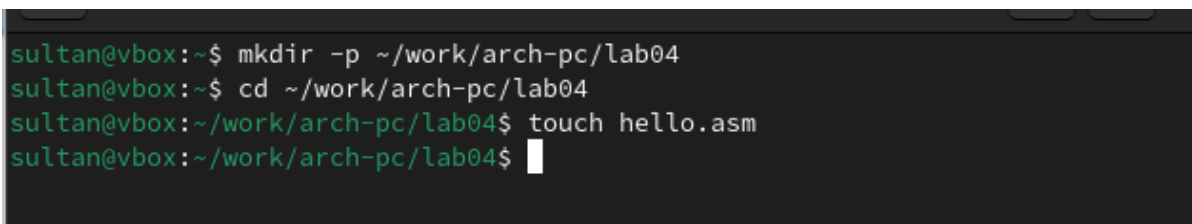
1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Выполнение лабораторной работы

2.1 Программа Hello world!


Для начала создаю новый каталог `lab04` с помощью команды `mkdir`, затем перехожу в него, используя команду `cd`. После этого создаю файл `hello.asm`. На рис. 2.1 показан процесс создания каталога и файла.



```
sultan@vbox:~$ mkdir -p ~/work/arch-pc/lab04
sultan@vbox:~$ cd ~/work/arch-pc/lab04
sultan@vbox:~/work/arch-pc/lab04$ touch hello.asm
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.1: Создание каталога и файла

Открываю файл `hello.asm` в текстовом редакторе и пишу код программы по заданию, как показано на рис. 2.2.

Открыть ▾ 

hello.asm
~/work/arch-pc/lab04

```
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Hello world!',10 ; 'Hello world!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
|
```

Рис. 2.2: Программа hello.asm

2.2 Трансляция кода с помощью NASM

Для того чтобы скомпилировать файл, использую транслятор NASM. С помощью команды `nasm` я создаю объектный файл `hello.o`, что показано на рис. 2.3.

```
sultan@vbox:~/work/arch-pc/lab04$  
sultan@vbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm  
sultan@vbox:~/work/arch-pc/lab04$ ls  
hello.asm hello.o  
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.3: Трансляция hello.asm

Для более детального анализа программы применяю команду `nasm` с дополнительными опциями, которые позволяют создать листинг (`list.lst`), объектный файл (`obj.o`), а также добавить отладочную информацию. Результат показан на рис. 2.4.

```
sultan@vbox:~/work/arch-pc/lab04$  
sultan@vbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm  
sultan@vbox:~/work/arch-pc/lab04$ ls  
hello.asm hello.o list.lst obj.o  
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.4: Трансляция hello.asm с дополнительными опциями

2.3 Линковка с использованием LD

После успешной трансляции выполняю линковку объектного файла `hello.o`, используя компоновщик `ld`. Это позволяет создать исполняемый файл, как показано на рис. 2.5.

```
sultan@vbox:~/work/arch-pc/lab04$  
sultan@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello  
sultan@vbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o list.lst obj.o  
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.5: Линковка программы

Затем повторяю этот процесс для объектного файла `obj.o`, в результате чего получаю исполняемый файл с именем `main`. Результат показан на рис. 2.6.


```
sultan@vbox:~/work/arch-pc/lab04$  
sultan@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main  
sultan@vbox:~/work/arch-pc/lab04$ ls  
hello hello.asm hello.o list.lst main obj.o  
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.6: Линковка программы

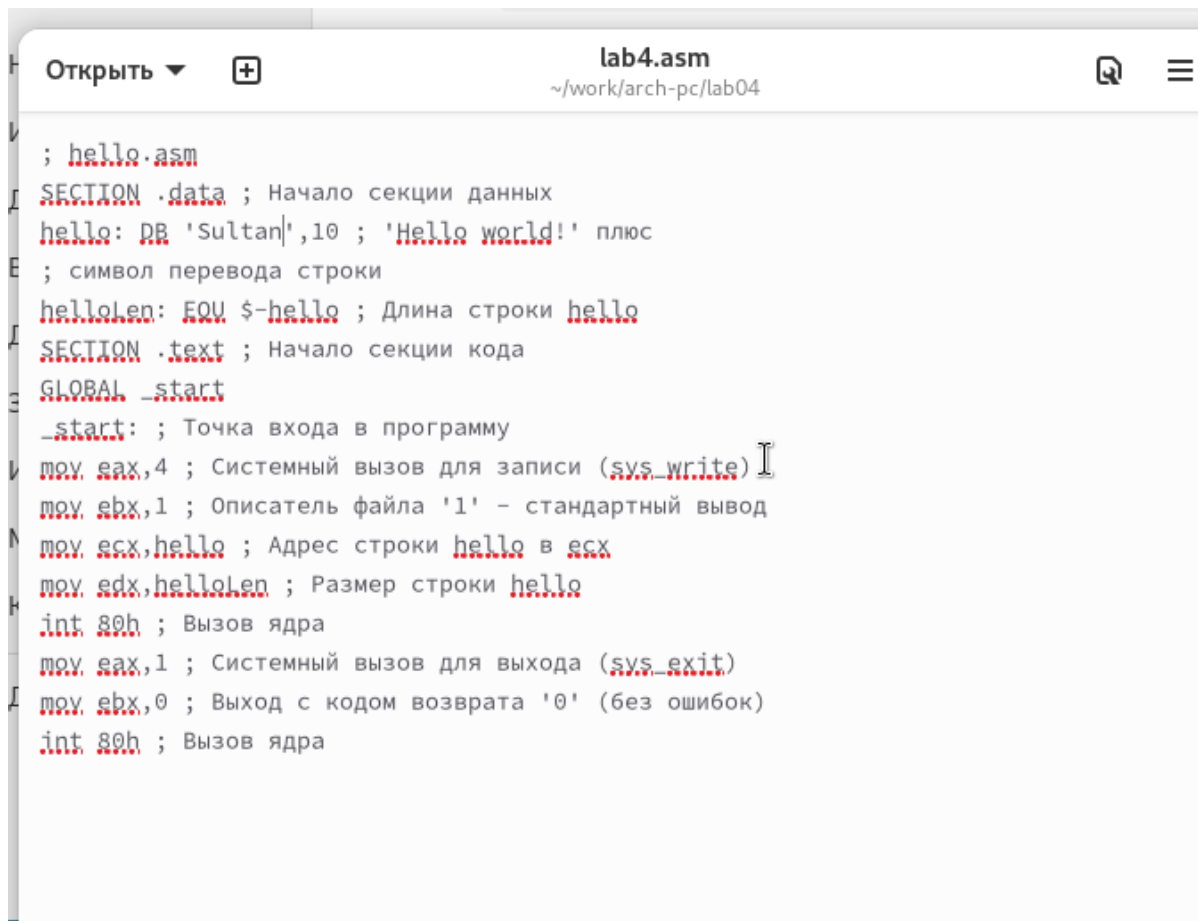
Запускаю оба полученных исполняемых файла, как видно на рис. 2.7.

```
sultan@vbox:~/work/arch-pc/lab04$  
sultan@vbox:~/work/arch-pc/lab04$ ./hello  
Hello world!  
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.7: Запуск программ

2.4 Выполнение заданий для самостоятельной работы

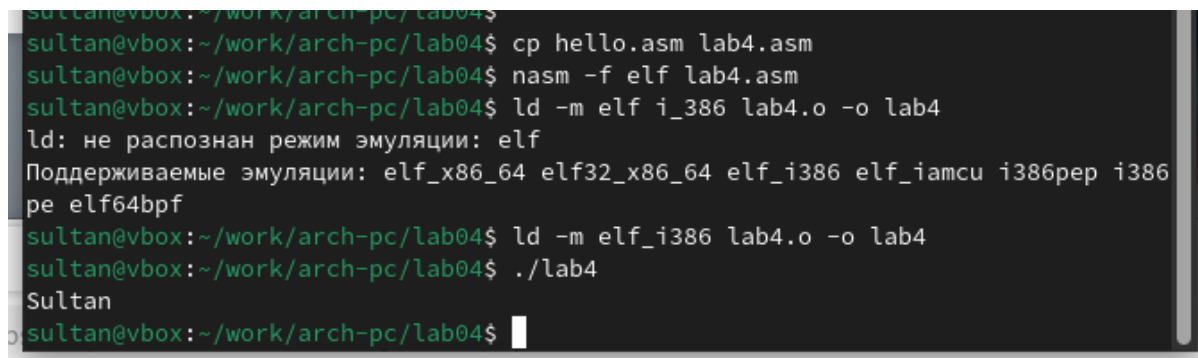
Для выполнения индивидуального задания копирую исходную программу в новый файл. Вношу изменения в код, заменяя сообщение “Hello world” на своё имя, что продемонстрировано на рис. 2.8. После этого запускаю изменённую программу (рис. 2.9).



```
Открыть ▾ + lab4.asm
~/work/arch-pc/lab04

; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Sultan|',10 ; 'Hello world!' плюс
; символ перевода строки
hellolen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,hellolen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
```

Рис. 2.8: Код программы в файле lab4.asm



```
sultan@vbox:~/work/arch-pc/lab04$
sultan@vbox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
sultan@vbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
sultan@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
ld: не распознан режим эмуляции: elf
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i386pep i386
pe elf64bpf
sultan@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
sultan@vbox:~/work/arch-pc/lab04$ ./lab4
Sultan
sultan@vbox:~/work/arch-pc/lab04$
```

Рис. 2.9: Запуск программы lab4.asm

2.5 Выводы

В ходе выполнения лабораторной работы ознакомились с основными этапами работы с программами на ассемблере с использованием NASM. Были освоены такие важные шаги, как создание объектных файлов, компиляция кода, использование компоновщика LD, а также работа с отладочной информацией и запуск готовых программ.