

Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютера

Абрамова Ульяна Михайловна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	13
6	Список литературы	14

Список иллюстраций

4.1	Перемещение в домашний каталог	9
4.2	Создание и заполнения файла для вывода “Hello world!”	9
4.3	Превращение текста программы для вывода “Hello world!” в объект- ный код	10
4.4	Ввод команды для скомпилирования файла hello.asm	10
4.5	Передача файла hello.o на обработку компоновщику LD	10
4.6	Аналогичная операция	11
4.7	Запуск	11
4.8	Копирование файла	11
4.9	Редактирование файла	11
4.10	Компилирование и передача объектного файла на обработку ком- пановщику	12
4.11	Запуск	12
4.12	Отправка файлов	12

List of Tables

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства:

- * арифметико-логическое устройство (АЛУ) – выполняет логические и арифметические действия
- * устройство управления (УУ) – обеспечивает управление и контроль всех устройств компьютера
- * регистры – сверхбыстрая оперативная память небольшого объема, входящая в состав процессора
 - RAX, RCX, RDX, RBX, RSI, RDI – 64-битные
 - EAX, ECX, EDX, EBX, ESI, EDI – 32-битные
 - AX, CX, DX, BX, SI, DI – 16-битные
 - AH, AL, CH, CL, DH, DL, BH, BL – 8-битные

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ – это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек

памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ:

- * устройства внешней памяти, которые предназначены для долговременного хранения больших объемов информации;
- * устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем:

- формирование адреса в памяти очередной команды;
- считывание кода команды из памяти и её дешифрация;
- выполнение команды;
- переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

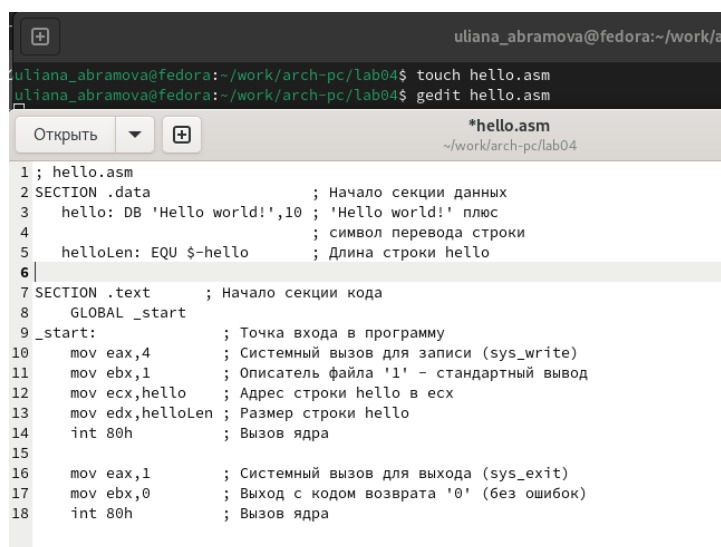
4 Выполнение лабораторной работы

1. Создание программы Hello world! С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 4.1)

```
uliana_abramova@fedora:~$ mkdir -p ~/work/arch-pc/lab04
uliana_abramova@fedora:~$ cd ~/work/arch-pc/lab04
```

Рис. 4.1: Перемещение в домашний каталог

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch`, открываю созданный файл в текстовом редакторе `gedit` и заполняю в него программу для вывода “Hello world!” (рис. 4.2)



The screenshot shows a terminal window at the top with the following commands and output:

```
uliana_abramova@fedora:~/work/arch-pc/lab04$ touch hello.asm
uliana_abramova@fedora:~/work/arch-pc/lab04$ gedit hello.asm
```

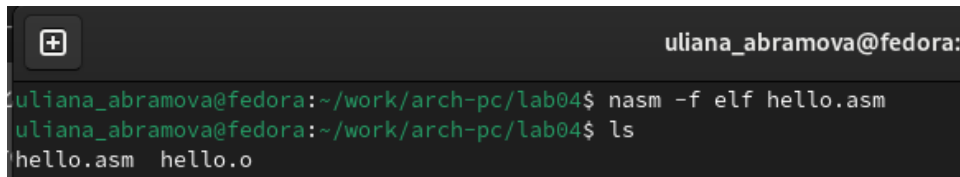
Below the terminal is a window of the `gedit` text editor. The title bar shows the filename `*hello.asm` and the path `~/work/arch-pc/lab04`. The editor contains the following assembly code:

```
1 ; hello.asm
2 SECTION .data          ; Начало секции данных
3     hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4                               ; символ перевода строки
5     helloLen: EQU $-hello ; Длина строки hello
6
7 SECTION .text          ; Начало секции кода
8     GLOBAL _start
9 _start:                ; Точка входа в программу
10    mov eax,4           ; Системный вызов для записи (sys_write)
11    mov ebx,1           ; Описатель файла '1' - стандартный вывод
12    mov ecx,hello       ; Адрес строки hello в ecx
13    mov edx,helloLen    ; Размер строки hello
14    int 80h            ; Вызов ядра
15
16    mov eax,1           ; Системный вызов для выхода (sys_exit)
17    mov ebx,0           ; Выход с кодом возврата '0' (без ошибок)
18    int 80h            ; Вызов ядра
```

Рис. 4.2: Создание и заполнения файла для вывода “Hello world!”

2. Работа с транслятором NASM Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя

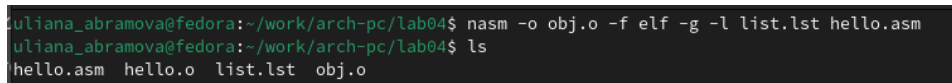
команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. ??). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл `hello.o`.



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

Рис. 4.3: Превращение текста программы для вывода “Hello world!” в объектный код

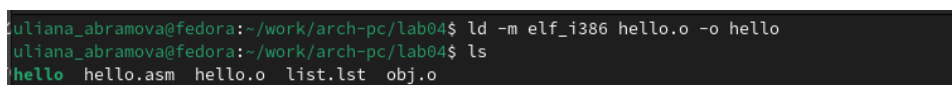
3. Работа с расширенным синтаксисом командной строки NASM Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. ??). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.



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

Рис. 4.4: Ввод команды для скомпилирования файла `hello.asm`

4. Работа с компоновщиком LD Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello` (рис. ??). Ключ `-o` задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты `ls` правильность выполнения команды.



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

Рис. 4.5: Передача файла `hello.o` на обработку компоновщику LD

Выполняю следующую команду (рис. ??). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

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

Рис. 4.6: Аналогичная операция

5. Запуск исполняемого файла Запускаю на выполнение созданный исполняемый файл hello (рис. ??).

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

Рис. 4.7: Запуск

6. Выполнение заданий для самостоятельной работы С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. ??).

```
uliana_abramova@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
uliana_abramova@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
```

Рис. 4.8: Копирование файла

С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. ??).

```
uliana_abramova@fedora:~/work/arch-pc/lab04$ gedit lab4.asm
lab4.asm
~/work/arch-pc/lab04
Сохранить

1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3 lab4: DB 'Ульяна Абрамова',10 ; 'Ульяна Абрамова' плюс
4 ; символ перевода строки
5 lab4Len: EQU $-lab4 ; Длина строки lab4
6
7 SECTION .text ; Начало секции кода
8 GLOBAL _start
9 _start: ; Точка входа в программу
10 mov eax,4 ; Системный вызов для записи (sys_write)
11 mov ebx,1 ; Описатель файла '1' - стандартный вывод
12 mov ecx,lab4 ; Адрес строки lab4 в ecx
13 mov edx,lab4Len ; Размер строки lab4
14 int 80h ; Вызов ядра
15
16 mov eax,1 ; Системный вызов для выхода (sys_exit)
17 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18 int 80h ; Вызов ядра
```

Рис. 4.9: Редактирование файла

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты `ls`, что файл `lab4.o` создан. Передаю объектный файл `lab5.o` на обработку компоновщику LD, чтобы получить исполняемый файл `lab4` (рис. ??).

```
uliana_abramova@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
uliana_abramova@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
uliana_abramova@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
uliana_abramova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
uliana_abramova@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
uliana_abramova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
uliana_abramova@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
```

Рис. 4.10: Компилирование и передача объектного файла на обработку компоновщику

Запускаю исполняемый файл `lab4`, на экран действительно выводятся мои имя и фамилия (рис. ??).

```
uliana_abramova@fedora:~/work/arch-pc/lab04$ ./lab4
Ульяна Абрамова
```

Рис. 4.11: Запуск

Далее добавляю файлы на GitHub и отправляю на сервер с помощью команды `git push` (рис. ??).

```
uliana_abramova@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 77, готово.
Подсчет объектов: 100% (71/71), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (63/63), готово.
Запись объектов: 100% (63/63), 5.99 МиБ | 958.00 КиБ/с, готово.
Total 63 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), completed with 4 local objects.
To github.com:UlianaAbrams/study_2023-2024_arh--pc.git
eb8ad89..df06171 master -> master
```

Рис. 4.12: Отправка файлов

5 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы

1. Архитектура ЭВМ