

Отчет по лабораторной работе №4

Дисциплина: Архитектура компьютера

Краснова Камилла Геннадьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание каталога	10
4.2	Создание пустого каталога	10
4.3	Заполнение файла	11
4.4	Компиляция текста программы	11
4.5	Компиляция текста программы	11
4.6	Передача объектного файла на обработку компоновщику	12
4.7	Передача объектного файла на обработку компоновщику	12
4.8	Запуск исполняемого файла	12
4.9	Создание копии файла	12
4.10	Изменение программы	13
4.11	Компиляция текста программы	13
4.12	Передача объектного файла на обработку компоновщику	13
4.13	Запуск исполняемого файла	13
4.14	Копирование файлов	14
4.15	Отправка файлов	14

Список таблиц

1 Цель работы

Цель данной лабораторной работы – освоить 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 электронно-вычислительной машины (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 (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) входят следующие устройства: • арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; • устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; • регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью,

преобразование (арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ):

- 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 (половинки 16-битных регистров).

Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. 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. В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты);
- 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. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. 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. В самом общем виде он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде. Данный алгоритм позволяет выполнить хранящуюся в ОЗУ программу. Кроме того, в зависимости от команды при её выполнении могут проходить не все этапы.

4 Выполнение лабораторной работы

4.1 Создание программы Hello world! Создаю каталог для работы с программами, перехожу в него с помощью утилиты `cd` (рис. 4.1).

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

Рис. 4.1: Создание каталога

Создаю в текущем каталоге текстовый файл `hello.asm`, используя `touch`, и открываю его с помощью текстового редактора `gedit` (рис. 4.2).

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

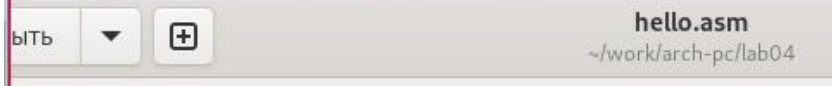


Рис. 4.2: Создание пустого каталога

Заполняю файл, вставляя в него программу для вывода “Hello world!” (рис. 4.3).

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

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 SECTION .text ; Начало секции кода
7     GLOBAL _start
8 _start: ; Точка входа в программу
9     mov eax,4 ; Системный вызов для записи (sys_write)
10    mov ebx,1 ; Описатель файла '1' - стандартный вывод
11    mov ecx,hello ; Адрес строки hello в ecx
12    mov edx,helloLen ; Размер строки hello
13    int 80h ; Вызов ядра
14
15    mov eax,1 ; Системный вызов для выхода (sys_exit)
16    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
17    int 80h ; Вызов ядра
```

Рис. 4.3: Заполнение файла

4.2 Работа с транслятором NASM Компилирую текст программы “Hello world!”, используя команду `nasm -f elf hello.asm`. Проверяю, что создан объектный файл `hello.o` (рис. 4.4).

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

Рис. 4.4: Компиляция текста программы

4.3 Работа с расширенным синтаксисом командной строки NASM Компилирую файл `hello.asm` в `obj.o`. Далее проверяю, что файлы были созданы (рис. 4.5).

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

Рис. 4.5: Компиляция текста программы

4.4 Работа с компоновщиком LD Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello`. Далее проверяю правильность выполнения команды (рис. 4.6).

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

Рис. 4.6: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. 4.7). 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.

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

Рис. 4.7: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла Запускаю на выполнение созданный исполняемый файл hello (рис. 4.8).

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

Рис. 4.8: Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы С помощью команды cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 4.9).

```
kamilla@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
kamilla@fedora:~/work/arch-pc/lab04$
```

Рис. 4.9: Создание копии файла

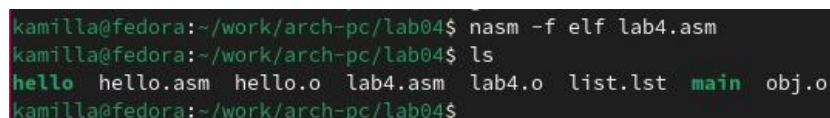
С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию (рис. 4.10).



```
kamilla@fedora:~/work/arch-pc/lab04$ gedit lab4.asm
*lab4.asm
~/work/arch-pc/lab04
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Krasnova Kamilla',10 ;
4     ; символ перевода строки
```

Рис. 4.10: Изменение программы

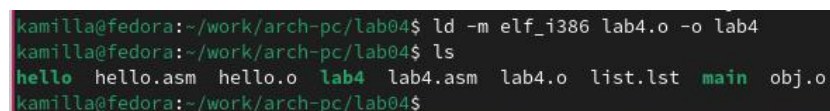
Компилирую текст программы в объектный файл. Проверяю, что файл lab4.o создан (рис. 4.11).



```
kamilla@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
kamilla@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
kamilla@fedora:~/work/arch-pc/lab04$
```

Рис. 4.11: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 4.12).



```
kamilla@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
kamilla@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
kamilla@fedora:~/work/arch-pc/lab04$
```

Рис. 4.12: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, вижу, что на экран действительно выводятся мои имя и фамилия (рис. 4.13).



```
kamilla@fedora:~/work/arch-pc/lab04$ ./lab4
Krasnova Kamilla
kamilla@fedora:~/work/arch-pc/lab04$
```

Рис. 4.13: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий, проверяю с помощью ls, что файлы скопировались (рис. 4.14).

```
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.14: Копирование файлов

С помощью команд `git add .` (рис. ??) и `git commit` (рис. ??) добавляю файлы на GitHub.

```
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add .
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

```
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add files"
2 files changed, 34 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
```

Отправляю файлы на сервер с помощью команды `git push` (рис. 4.15).

```
kamilla@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 976 байтов | 976.00 КиБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:kakras/study_2024-2025_arh-pc.git
```

Рис. 4.15: Отправка файлов

5 Выводы

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

Список литературы

1. Архитектура ЭВМ