

Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблера NASM

Самарханова Полина Тимуровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Создание программы Hello world!	8
4.2	Работа с транслятором NASM.	9
4.3	Работа с компоновщиком LD.	10
4.4	Запуск исполняемого файла.	10
4.5	Выполнение заданий для самостоятельной работы.	11
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Перемещение по директориям и создание файла	8
4.2	Файл hello.asm	8
4.3	Программа	9
4.4	Создание бинарного файла	9
4.5	Создание файла листинга	10
4.6	Получение исполняемого файла	10
4.7	Значение main	10
4.8	Проверка программы	11
4.9	Копирование файла hello.asm	11
4.10	Изменение программы	12
4.11	Создание объектного файла	12
4.12	Получение исполняемого файла	13
4.13	Проверка программы	13
4.14	Отправление работы на github	13

Список таблиц

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

Создание программы Hello world!

Работа с транслятором NASM

Работа с расширенным синтаксисом командной строки NASM

Работа с компоновщиком LD

Запуск исполняемого файла

Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). 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.

4 Выполнение лабораторной работы

4.1 Создание программы Hello world!

С помощью команды `cd` перехожу в каталог и создаю пустой текстовый файл `hello.asm`(рис. 4.1).

```
[spolina@fedora ~]$ cd work/study/2023-2024/"Архитектура компьютера"/arh-pc/labs/lab04  
[spolina@fedora lab04]$ touch hello.asm
```

Рис. 4.1: Перемещение по директориям и создание файла

Открываю файл в текстовом редакторе `gedit`(рис. 4.2).



Рис. 4.2: Файл `hello.asm`

Заполняю файл, заполняю программу для вывода Hello world (рис. ??).

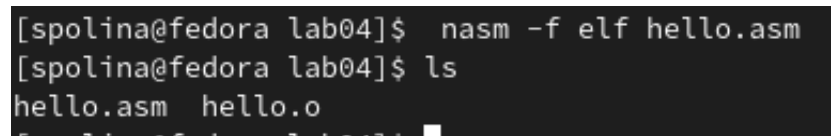


```
1 ; hello.asm
2 SECTION .data
3     hello:    DB 'Hello world!',10
4
5     helloLen: EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,1
13     mov ecx,hello
14     mov edx,helloLen
15     int 80h
16
17     mov eax,1
18     mov ebx,0
19     int 80h
```

Рис. 4.3: Программа

4.2 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 и с помощью утилиты `ls` проверяю создан ли файл `hello.o` (рис. 4.4).



```
[spolina@fedora lab04]$ nasm -f elf hello.asm
[spolina@fedora lab04]$ ls
hello.asm  hello.o
```

Рис. 4.4: Создание бинарного файла

##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` и правильность выполнения команды. (рис. 4.5).

```
[spolina@fedora lab04]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[spolina@fedora lab04]$ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.5: Создание файла листинга

4.3 Работа с компоновщиком LD.

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды. (рис. 4.6).

```
[spolina@fedora lab04]$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.6: Получение исполняемого файла

Выполняю следующую команду. 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. (рис. 4.7).

```
[spolina@fedora lab04]$ ld -m elf_i386 obj.o -o main
[spolina@fedora lab04]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4.7: Значение main

4.4 Запуск исполняемого файла.

Запускаю созданный файл, чтобы проверить программу. (рис. 4.8).

```
[spolina@fedora lab04]$ ./hello
Hello world!
```

Рис. 4.8: Проверка программы

4.5 Выполнение заданий для самостоятельной работы.

Создаю копию файла, называю его lab04.asm. (рис. 4.9)

```
[spolina@fedora lab04]$ cp hello.asm lab04.asm
```

Рис. 4.9: Копирование файла hello.asm

С помощью текстового редактора gedit открываю файл lab04.asm и меняю программу так, чтобы она выводила мои имя и фамилию.(рис. 4.10)

```
spolina@fedora lab04]$ gedit lab04.asm

Открыть ▼ + ~/work/study/2023-2024

1 ; lab04.asm
2 SECTION .data
3     hello:    DB 'Polina Samarkhanova',10
4
5     helloLen: EQU $-lab4
6
7 SECTION .text
8     GLOBAL _start
9
10 _start:
11     mov eax,4
12     mov ebx,1
13     mov ecx,hello
14     mov edx,helloLen
15     int 80h
16
17     mov eax,1
18     mov ebx,0
19     int 80h
```

Рис. 4.10: Изменение программы

Компилирую текст программы в объектный файл.Проверяю с помощью утилиты ls, что файл lab04.o создан.(рис. 4.11)

```
[spolina@fedora lab04]$ nasm -f elf lab04.asm
[spolina@fedora lab04]$ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o
```

Рис. 4.11: Создание объектного файла

Передаю объектный файл lab04.o на обработку компоновщику LD, чтобы получить исполняемый файл lab04.(рис. 4.12)

```
[spolina@fedora lab04]$ ld -m elf_i386 lab04.o -o lab04
[spolina@fedora lab04]$ ls
hello  hello.asm  hello.o  lab04  lab04.asm  lab04.o  list.lst  main  obj.o
```

Рис. 4.12: Получение исполняемого файла

Запускаю исполняемый файл lab04, на экран действительно выводятся мои имя и фамилия.(рис. 4.13)

```
[spolina@fedora lab04]$ ./lab04
Samarkhanova Polina
```

Рис. 4.13: Проверка программы

Отправление всех изменений на github. (рис. 4.14)

```
[spolina@fedora lab04]$ git add .
[spolina@fedora lab04]$ git commit -am 'feat(main): add files lab-4'
[master 380ea66] feat(main): add files lab-4
17 files changed, 120 insertions(+), 34 deletions(-)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab04.asm
create mode 100644 labs/lab04/report/image/1.png
create mode 100644 labs/lab04/report/image/10.png
create mode 100644 labs/lab04/report/image/11.png
create mode 100644 labs/lab04/report/image/12.png
create mode 100644 labs/lab04/report/image/13.png
create mode 100644 labs/lab04/report/image/2.png
create mode 100644 labs/lab04/report/image/3.png
create mode 100644 labs/lab04/report/image/4.png
create mode 100644 labs/lab04/report/image/5.png
create mode 100644 labs/lab04/report/image/6.png
create mode 100644 labs/lab04/report/image/7.png
create mode 100644 labs/lab04/report/image/8.png
create mode 100644 labs/lab04/report/image/9.png
delete mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
[spolina@fedora lab04]$ git push
Перечисление объектов: 28, готово.
Подсчет объектов: 100% (28/28), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (22/22), готово.
Запись объектов: 100% (22/22), 298.35 Киб | 1.74 Миб/с, готово.
Всего 22 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:PolinaSamarkhanova/study_2023-2024_arh-pc.git
 8bf9d33..380ea66 master -> master
```

Рис. 4.14: Отправление работы на github

5 Выводы

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

Список литературы

(1Архитектура ЭВМ)