

Лабораторная работа №4

Дисциплина: Архитектура компьютера

Малюга Валерия Васильевна

Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Основные принципы работы компьютера	7
Ассемблер и язык ассемблера	8
Процесс создания и обработки программы на языке ассемблера	8
Выполнение лабораторной работы	9
Создание программы Hello world!	9
Работа с транслятором NASM	9
Работа с расширенным синтаксисом командной строки NASM	10
Работа с компоновщиком LD	10
Запуск исполняемого файла	11
Выполнение заданий для самостоятельной работы	11
Выводы	14

Список таблиц

Список иллюстраций

1	Создание программы Hello world	9
2	Компиляция программы hello	10
3	Компиляция текста программы	10
4	Передача объектного файла на обработку компоновщику	10
5	Запуск программы Hello world	11
6	Изменение программы lab4.asm	12
7	Трансляция в объектный файл и его компоновка. Запуск исполняемого файла	12
8	Копирование файлов в локальный репозиторий и отправка на Github	13

Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных

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.

Теоретическое введение

Основные принципы работы компьютера

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются:

Основной задачей процессора является обработка информации, а также организация координации работы устройств.

- **арифметико-логическое устройство (АЛУ)** — выполняет логические и арифметические действия;
- **устройство управления (УУ)** — обеспечивает управление и контроль всех устройств компьютера;
- **регистры** — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора.

Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют.

Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр имеет своё имя.

Другим важным узлом ЭВМ является **оперативное запоминающее устройство (ОЗУ)**. ОЗУ — это устройство, которое хранит информацию, необходимую для работы процессора.

В состав ЭВМ также входят **периферийные устройства**, которые можно разделить на:

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

В основе вычислительного процесса ЭВМ лежит **принцип программного управления**. Это означает, что все действия выполняются в строго определённом порядке.

Набор машинных команд определяется устройством конкретного процессора. В коде машинной команды содержится информация о том, какое действие должен выполнить процессор.

При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий:

1. формирование адреса в памяти очередной команды;
2. считывание кода команды из памяти и её дешифрация;
3. выполнение команды;
4. переход к следующей команде.

Ассемблер и язык ассемблера

****Язык ассемблера**** (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближён к аппаратной структуре процессора. Процессор понимает не команды ассемблера, а последовательности из нулей и единиц — ****машинный код****. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы. В нём поддерживаются инструкции x86-64.

Типичный формат записи команд NASM имеет вид:

[метка:] мнемокод [операнд {, операнд}] [; комментарий]

Здесь ****мнемокод**** — непосредственно мнемоника инструкции процессору, которая является

обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.е. метка перед командой связана с адресом данной команды. Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

Процесс создания и обработки программы на языке ассемблера

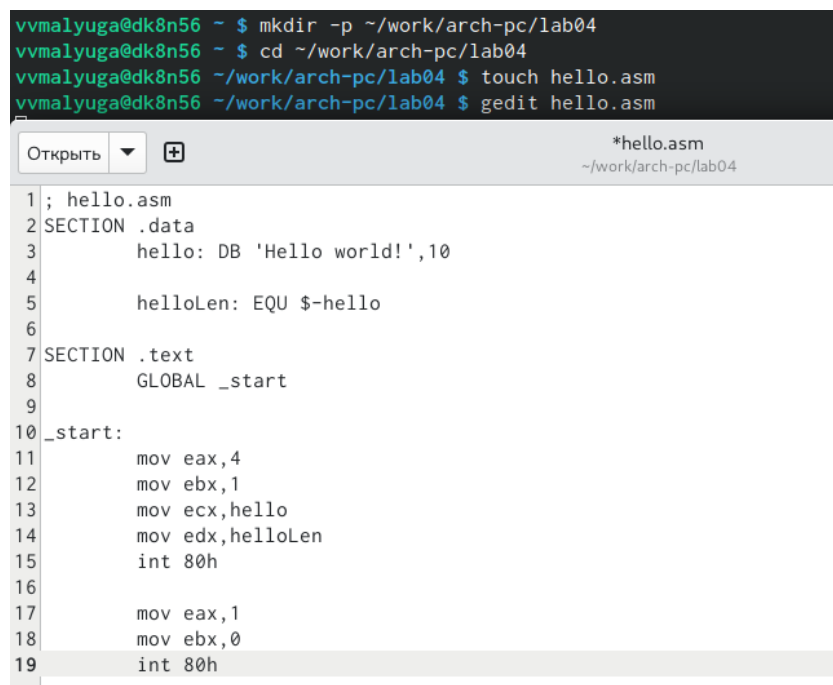
В процессе создания ассемблерной программы можно выделить четыре шага:

- ****Набор текста**** программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет расширение .asm.
- ****Трансляция**** — преобразование с помощью транслятора, например nasm, текста программы в машинный код.
- ****Компоновка или линковка**** — этап обработки объектного кода компоновщиком (ld), который производит загрузку в память.
- ****Запуск программы****. Конечной целью является работоспособный исполняемый файл. Ошибки в программе могут возникнуть на любом из этапов.

Выполнение лабораторной работы

Создание программы Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM. Перешла в созданный каталог



```
vvmalyuga@dk8n56 ~ $ mkdir -p ~/work/arch-pc/lab04
vvmalyuga@dk8n56 ~ $ cd ~/work/arch-pc/lab04
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ touch hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ gedit hello.asm
```

The screenshot shows a terminal window with the following commands and their outputs:

```
vvmalyuga@dk8n56 ~ $ mkdir -p ~/work/arch-pc/lab04
vvmalyuga@dk8n56 ~ $ cd ~/work/arch-pc/lab04
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ touch hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Below the terminal, a text editor window titled `*hello.asm` is open, showing the following assembly code:

```
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. 1: Создание программы Hello world

Работа с транслятором NASM

Для компиляции программы "Hello world" написала в терминале команду `nasm -f elf hello.asm`. Впоследствии проверила выполнение этой команды с помощью `ls`. Действительно, тран

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 2: Компиляция программы hello

Работа с расширенным синтаксисом командной строки

NASM

Ввела команду, которая скомпилировала файл `hello.asm` в файл `obj.o`, при этом в файл были включены (с помощью ключа `-g`), также с помощью ключа `-l` был создан файл листинга `list.lst`. Далее проверила с помощью утилиты `ls` (рис. 3).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 3: Компиляция текста программы

Работа с компоновщиком LD

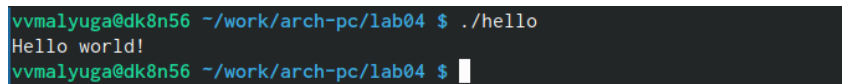
Передала объектный файл на обработку компоновщику, чтобы получить исполняемую программу. В командной строке указала `-m elf_i386` (рис. 4). Исполняемый файл будет иметь имя `main`, т.к. после ключа `-o` было задано значение `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o` (рис. 5).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4: Передача объектного файла на обработку компоновщику

Запуск исполняемого файла

Запустила созданный исполняемый файл, набрав в терминале команду (рис. [-@fig:005]).

A screenshot of a terminal window with a dark background. The prompt is 'vvmalyuga@dk8n56 ~/work/arch-pc/lab04 \$'. The user has entered './hello' and the output is 'Hello world!'. The prompt is now 'vvmalyuga@dk8n56 ~/work/arch-pc/lab04 \$' followed by a cursor.

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ./hello
Hello world!
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 5: Запуск программы Hello world

Выполнение заданий для самостоятельной работы

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создала копию файла `hello.asm` с именем `lab4.asm` (рис. [-@fig:006]).
2. С помощью текстового редактора `gedit` внесла изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моими фамилией и именем (рис. [-@fig:006]).

```
hello hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ gedit lab4.asm

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

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

Рис. 6: Изменение программы lab4.asm

3. Оттранслировала полученный текст программы lab4.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл (рис. [-@fig:007]).

```
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ ./lab4
Малюга Валерия
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $
```

Рис. 7: Трансляция в объектный файл и его компоновка. Запуск исполняемого файла

4. Скопировала файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузила файлы на Github (рис. [-@fig:008]).

```

vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/hello.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/lab4.asm
vvmalyuga@dk8n56 ~/work/arch-pc/lab04 $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm lab4.asm presentation report
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -am 'feat(main): add files lab-4'
[master 6e0d07a] feat(main): add files lab-4
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 796 байтов | 796.00 КиБ/с, готово.
Всего 6 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:vvmalyuga/study_2023-2024_arhpc.git
  64c53a2..6e0d07a master -> master
vvmalyuga@dk8n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $

```

Рис. 8: Копирование файлов в локальный репозиторий и отправка на Github

Выводы

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