

Шаблон отчёта по лабораторной работе №4

Архитектура компьютера

Гафоров Нурмухаммад Вомикович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Создание програма Hello word	10
5.1	Работа с транслятором NASM	11
5.2	Работа с расширенным синтаксисом командной строки NASM . .	12
5.3	Работа с компоновщиком LD	12
5.4	Запуск исполняемого файла	13
5.5	Выполнение заданий для самостоятельной работы.	13
6	Выводы	16
	Список литературы	17

Список иллюстраций

5.1	Открыть Терминал	10
5.2	Создал каталог NASM	10
5.3	Приходил в созданный каталог	10
5.4	создали файл	10
5.5	Открыли файл	11
5.6	Созданный файл Hello asm	11
5.7	Написали текст	11
5.8	Компиляция текста программы транслятор NASM	12
5.9	Расширенный синтаксис командной строки NASM	12
5.10	Передача объектного файла на обработку компоновщику	12
5.11	Передача объектного файла на обработку компоновщику	13
5.12	Запуск исполняемого файла	13
5.13	Создал файл lab4	13
5.14	Изменение программы	14
5.15	создал файл lab4 и открыл	14
5.16	Загрузит файл на Github	15
5.17	Проверка файлы	15

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 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 объёма, 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 длительного хранения бол
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

5 Создание програма Hello word

Открыли терминал (рис.[5.1]).

```
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.1: Открыть Терминал

Создайте каталог для работы с программами на языке ассемблера NASM(рис. [5.2]).

```
nvgaforov@dk3n52 ~ $ mkdir -p ~/work/arch-pc/lab04
nvgaforov@dk3n52 ~ $
```

Рис. 5.2: Создал каталог NASM

Прейдите в созданный каталог (рис. [5.3])

```
nvgaforov@dk3n52 ~ $ cd ~/work/arch-pc/lab04
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.3: Приходил в созданный каталог

Создаю в текущем каталоге пустой текстовый файл Hello.asm с помощью touch (рис. [5.4])

```
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ touch hello.asm
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.4: создали файл

Откройте файл с помощью gedit(рис. [5.5])

```
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ gedit hello.asm
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.5: Открыли файл

Введите в него следующий текст(рис. [5.6])

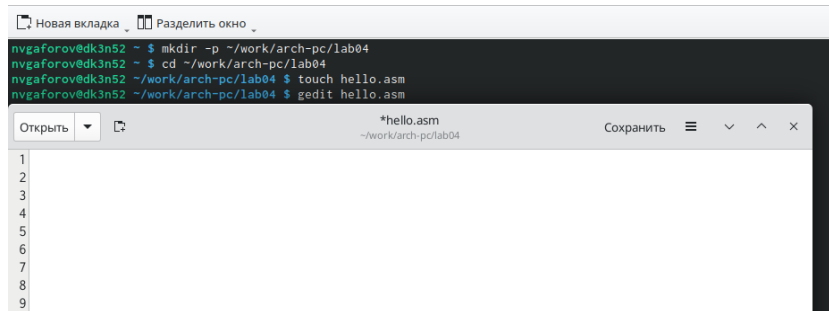


Рис. 5.6: Созданный файл Hello asm

Напишите текст (рис. [5.7])



Рис. 5.7: Написали текст

5.1 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл `hello.o`.(рис. [5.8])

```
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o
```

Рис. 5.8: Компиляция текста программы транслятор NASM

5.2 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`. Далее проверяю с помощью утилиты `ls` правильность выполнения команды.(рис. [5.9])

```
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o list.lst obj.o
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.9: Расширенный синтаксис командной строки NASM

5.3 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. [5.10])

```
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o list.lst obj.o
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.10: Передача объектного файла на обработку компоновщику

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. [5.11])

```
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.11: Передача объектного файла на обработку компоновщику

5.4 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [5.12])

```
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ./hello
Hello world!
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.12: Запуск исполняемого файла

5.5 Выполнение заданий для самостоятельной работы.

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm(рис. [5.13])

```
hello world!
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/hello.asm ~/work/arch-pc/lab04/lab4.asm
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
nvgafarov@dk3n52 ~/work/arch-pc/lab04 $
```

Рис. 5.13: Создаль файл lab4

С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем (рис. [5.14])

```

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello:      DB 'Гафоров НУРМУХАММАД!',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
10    _start:      ; Точка входа в программу
11        mov eax,4 ; Системный вызов для записи (sys_write)
12        mov ebx,1 ; Описатель файла '1' - стандартный вывод
13        mov ecx,hello ; Адрес строки hello в ecx
14        mov edx,helloLen ; Размер строки hello
15        int 80h ; Вызов ядра
16
17        mov eax,1 ; Системный вызов для выхода (sys_exit)
18        mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19        int 80h ; Вызов ядра

```

Рис. 5.14: Изменение программы

Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл на экран выводилась строка с вашими фамилией и именем(рис. [5.15])

```

nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ gedit lab4.asm
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.o lab4.asm lab4.o list.lst main obj.o
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst lab4.asm
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.o lab4.asm lab4.o list.lst main obj.o
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.o lab4 lab4.asm lab4.o list.lst main obj.o
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ls
hello.o lab4 lab4.asm lab4.o list.lst main obj.o
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $ ./lab4
Гафоров Нурмухаммад
nvgaforov@dk3n52 ~/work/arch-pc/lab04 $

```

Рис. 5.15: создал файл lab4 и открыл

Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github.(рис. [5.16])

```
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (7/7), 587.94 КиБ | 23.52 МБ/с, готово.
Всего 7 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:muhammad200204/study_2023-2024_arh-pc.git
   0002f87..b96b8e2  master -> master
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git add .
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -am""
error: switch 'm' requires a value
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -am "files"
[master e406316] files
 2 files changed, 19 insertions(+)
 create mode 100644 labs/lab04/report/hello.asm
 create mode 100755 labs/lab04/report/lab4
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (7/7), 1.44 КиБ | 1.44 МБ/с, готово.
Всего 7 (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:muhammad200204/study_2023-2024_arh-pc.git
   b96b8e2..e406316  master -> master
nvgaforov@dk3n52 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 5.16: Загрузит файл на Github

Проверяем файлы (рис. [5.17])

study_2023-2024_arh-pc / labs / lab04 / report /

muhammad200204 files e406316 - 1 minute ago History

Name	Last commit message	Last commit date
..		
bb	feat(main): make course structure	3 weeks ago
image	feat(main): make course structure	3 weeks ago
pandoc	feat(main): make course structure	3 weeks ago
Makefile	feat(main): make course structure	3 weeks ago
hello.asm	files	1 minute ago
lab4	files	1 minute ago
report.md	feat(main): make course structure	3 weeks ago

Рис. 5.17: Проверка файлы

6 Выводы

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

Список литературы

https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%D0%