

Архитектура компьютера

Отчёт по лабораторной работе №4

Арбатова Варвара Петровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выполнение заданий для самостоятельной работы	13
6	Выводы	15
	Список литературы	16

Список таблиц

Список иллюстраций

4.1	Создание папки	10
4.2	Переход в созданную папку	10
4.3	Создание файла	10
4.4	Заполнение файла	11
4.5	Скачивание	11
4.6	Преобразование файла в объектный код	11
4.7	Преобразование файла	12
4.8	Передача файла на обработку	12
4.9	1	12
4.10	2	12
5.1	Файл в папо	13
5.2	Транслирую файл	13
5.3	Компановка и исполнение	14
5.4	Копирование файлов	14
5.5	Выгружаю изменения	14
5.6	Копирование файла с отчётом	14

1 Цель работы

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

2 Задание

- 1) Создать программу Hello world
- 2) Работа с транслятором NASM
- 3) Работа с расширенным синтаксисом командой строки NASM
- 4) Работа с компоновщиком LD
- 5) Запуск исполняемого файла
- 6) Выполнение заданий для самостоятельной работы

3 Теоретическое введение

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 объема, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. В процессе создания ассемблерной программы можно выделить четыре шага: • Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип `asm`. • Трансляция — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — `o`, файла листинга — `lst`. • Компоновка или линковка — этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение

мар. • Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага. 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. В состав ЭВМ также входят периферийные устройства, которые можно разделить на: • устройства внешней памяти, которые предназначены для длительного хранения больших объемов 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. Набор машинных команд определяется устройством конкретного процессора. 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

вание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде.

4 Выполнение лабораторной работы

- 1) Создаю рекурсивно вложенные в папку work папки arch-pc и lab04, проверяю их создание

```
vparbatova@Varvarishe:~$ mkdir -p ~/work/arch-pc/lab04
vparbatova@Varvarishe:~$ ls work
arch-pc  study
vparbatova@Varvarishe:~$ ls arch-pc
```

Рис. 4.1: Создание папки

- 2) Перехожу в созданную папку

```
vparbatova@Varvarishe:~$ cd ~/work/arch-pc/lab04
vparbatova@Varvarishe:~/work/arch-pc/lab04$
```

Рис. 4.2: Переход в созданную папку

- 3) Создаю файл hello с разрешением asm и проверяю его создание

```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ touch hello.asm
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ls
hello.asm
vparbatova@Varvarishe:~/work/arch-pc/lab04$
```

Рис. 4.3: Создание файла

- 4) Открываю этот файл в папо и копирую туда код из задания лабораторной работы

```

GNU nano 6.2                                hello.asm *
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Hello world!',10 ; 'Hello world!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра

```

Рис. 4.4: Заполнение файла

5) Скачиваю nasm

```

vparbatova@Varvarishe:~/work/arch-pc/lab04$ sudo apt install nasm
[sudo] password for vparbatova:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nasm
0 upgraded, 1 newly installed, 0 to remove and 51 not upgraded.
Need to get 375 kB of archives.
After this operation, 3345 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 nasm amd64 2.15.05-1 [375 kB]
Fetched 375 kB in 2s (241 kB/s)
Selecting previously unselected package nasm.
(Reading database ... 269099 files and directories currently installed.)
Preparing to unpack .../nasm_2.15.05-1_amd64.deb ...
Unpacking nasm (2.15.05-1) ...
Setting up nasm (2.15.05-1) ...
Processing triggers for man-db (2.10.2-1) ...
vparbatova@Varvarishe:~/work/arch-pc/lab04$

```

Рис. 4.5: Скачивание

6) Преобразовываю файл hello.asm в объектный код, записанный в файл hello.o. Проверяю, был ли создан файл

```

vparbatova@Varvarishe:~/work/arch-pc/lab04$ nasm -f elf hello.asm
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
vparbatova@Varvarishe:~/work/arch-pc/lab04$

```

Рис. 4.6: Преобразование файла в объектный код

7) Преобразую файл hello.asm в obj.o с помощью опции -o, которая позволяет задать имя объекта. Из-за elf -g формат выходного файла будет elf, и в него

будут включены символы для отладки, а так же будет создан файл листинга list.lst, благодаря -l. Проверяю созданные файлы

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

Рис. 4.7: Преобразование файла

- 8) Передаю файл компоновщику с помощью ld. Проверяю, создан ли исполняемый файл

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

Рис. 4.8: Передача файла на обработку

- 9) Передаю компоновщику файл obj.o и называю скомпонованный файл main. запуская сначала код для предыдущего файла(1), а затем для созданного сейчас(2)

```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ./hello
Hello world!
vparbatova@Varvarishe:~/work/arch-pc/lab04$
```

Рис. 4.9: 1

```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ./main
Hello world!
vparbatova@Varvarishe:~/work/arch-pc/lab04$
```

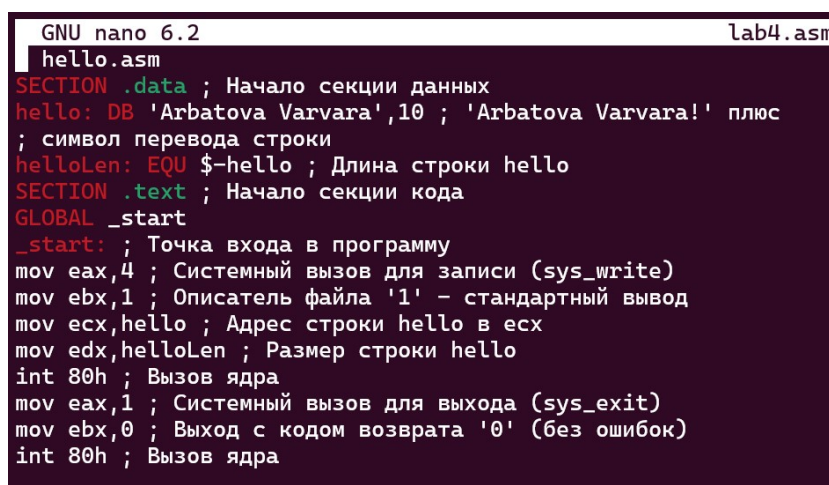
Рис. 4.10: 2

5 Выполнение заданий для самостоятельной работы

1) Копирую hello.asm с названием lab4.asm

!Копирование файла](image/10.jpg){#fig:001 width=70%}

2) С помощью nano изменяю текст кода так, чтобы он выводил моё имя и фамилию



```
GNU nano 6.2 lab4.asm
hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Arbatova Varvara',10 ; 'Arbatova Varvara!' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
```

Рис. 5.1: Файл в nano

3) Транслирую файл lab4.asm в объектный



```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
```

Рис. 5.2: Транслирую файл

4) Выполняю компоновку и запускаю исполняемый файл

```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
vparbatova@Varvarishe:~/work/arch-pc/lab04$ ./lab4
Arbatova Varvara
```

Рис. 5.3: Компоновка и исполнение

5) Копирую файлы в мой локальный репозиторий

```
vparbatova@Varvarishe:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/a
rch-pc/labs/lab04
vparbatova@Varvarishe:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/a
rch-pc/labs/lab04
-bash: rch-pc/labs/lab04: No such file or directory
vparbatova@Varvarishe:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/ar
ch-pc/labs/lab04
```

Рис. 5.4: Копирование файлов

6) Выгружаю изменения на GitHub

```
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git add .
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git commit -am "Ad
d file"
[master 35310eb] Add file
18 files changed, 34 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
create mode 100644 labs/lab04/report/image/1.bmp
create mode 100644 labs/lab04/report/image/10.bmp
create mode 100644 labs/lab04/report/image/11.bmp
create mode 100644 labs/lab04/report/image/12.bmp
create mode 100644 labs/lab04/report/image/13.bmp
create mode 100644 labs/lab04/report/image/14.bmp
create mode 100644 labs/lab04/report/image/2.bmp
create mode 100644 labs/lab04/report/image/3.bmp
create mode 100644 labs/lab04/report/image/4.bmp
create mode 100644 labs/lab04/report/image/5.bmp
create mode 100644 labs/lab04/report/image/6.bmp
create mode 100644 labs/lab04/report/image/7.bmp
create mode 100644 labs/lab04/report/image/8.bmp
create mode 100644 labs/lab04/report/image/9.bmp
create mode 100644 labs/lab04/report/report.docx
create mode 100644 labs/lab04/report/report.pdf
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git pull
Already up to date.
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git push
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 12 threads
Compressing objects: 100% (24/24), done.
```

Рис. 5.5: Выгружаю изменения

7) Копирую файл с отчётом и начинаю его заполнять

```
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ cd report
vparbatova@Varvarishe:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report$ cp report.m
d Л04_Арбатова_отчёт.md
```

Рис. 5.6: Копирование файла с отчётом

6 Выводы

Я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM

Список литературы