

Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Программа Hello world!	9
4.2	Транслятор NASM	10
4.3	Расширенный синтаксис командной строки NASM	11
4.4	Компоновщик LD	11
4.5	Запуск исполняемого файла	12
4.6	Задание для самостоятельной работы	12
5	Выводы	15
6	Список литературы	16

Список иллюстраций

4.1	Рис. 1. Создание каталога	9
4.2	Рис. 2. Создание текстового файла	9
4.3	Рис. 3. Открытие файла в текстовом редакторе	9
4.4	Рис. 4. Ввод текста	10
4.5	Рис. 5. Компиляция программы	10
4.6	Рис. 6. Проверка создания файла	11
4.7	Рис. 7. Выполнение команды	11
4.8	Рис. 8. Проверка создания файла	11
4.9	Рис. 9. Передача файла компоновщику	11
4.10	Рис. 10. Проверка создания файла	12
4.11	Рис. 11. Выполнение команды	12
4.12	Рис. 12. Запуск созданного файла на выполнение	12
4.13	Рис. 13. Создание копий файлов	12
4.14	Рис. 14. Изменений текста программы	13
4.15	Рис. 15. Трансляция текста в объектный файл	13
4.16	Рис. 16. Выполнение комановки	13
4.17	Рис. 17. Запуск получившегося файла	13
4.18	Рис. 18. Создание копии файла hello.asm	13
4.19	Рис. 19. Создание копии файла lab04.asm	14
4.20	Рис. 20. Проверка создания копий	14
4.21	Рис. 21. Загрузка файлов на Github	14

Список таблиц

1 Цель работы

Целью данной работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Создание программы Hello world!
2. Компиляция текста программы при помощи транслятора NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Передача объектного файла на обработку компоновщику LD
5. Запуска исполняемого файла
6. Задания для самостоятельной работы

3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык. Следует отметить, что процессор понимает не команды ассемблера, а последовательность программ, написанные на языке ассемблера, не уступают в качестве и скорости работы. Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры и т.д. Наиболее распространёнными ассемблерами для архитектуры x86 являются:

- для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM);
- для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис.

NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и 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. Допустимыми символами в метках являются буквы, цифры, а также следующие символы: `,` `$`, `#`, `@`, `~`, `.` и `?`. Начинаться метка или идентификатор могут с буквы, `.`, и `?`. Перед идентификаторами, которые пишутся как зарезервированные слова, нужно писать `$`,

чтобы компилятор трактовал его верно (так называемое экранирование). Максимальная длина идентификатора 4095 символов. 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.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM и перехожу в созданный каталог(рис. [4.1])

```
[vaosina@fedora ~]$ mkdir -p ~/work/arch-pc/lab04  
[vaosina@fedora ~]$ cd ~/work/arch-pc/lab04
```

Рис. 4.1: Рис. 1. Создание каталога

Создаю текстовый файл с именем hello.asm (рис. [4.2])

```
[vaosina@fedora lab04]$ touch hello.asm  
[vaosina@fedora lab04]$
```

Рис. 4.2: Рис. 2. Создание текстового файла

Открываю этот файл с помощью текстового редактора gedit (рис. [4.3])

```
[vaosina@fedora lab04]$ gedit hello.asm
```

Рис. 4.3: Рис. 3. Открытие файла в текстовом редакторе

и ввожу в него необходимый текст (рис. [4.4])

```

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
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 ехх
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. 4.4: Рис. 4. Ввод текста

4.2 Транслятор NASM

Для компиляции текста программы «Hello World» пишу следующую команду (рис. [4.5])

```

[vaosina@fedora lab04]$ nasm -f elf hello.asm
bash: nasm: команда не найдена...
Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
nasm-2.16.01-3.fc38.x86_64      A portable x86 assembler which uses Intel-like syntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

```

Рис. 4.5: Рис. 5. Компиляция программы

С помощью команды ls проверяю, что объектный файл был создан. Файл создается в формате ELF (рис. [4.6])

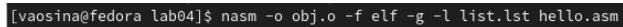


```
[vaosina@fedora lab04]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Рис. 4.6: Рис. 6. Проверка создания файла

4.3 Расширенный синтаксис командной строки NASM

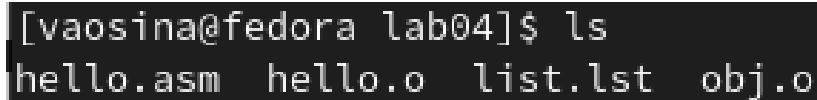
Выполняю следующую команду: (рис. [4.7])



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

Рис. 4.7: Рис. 7. Выполнение команды

С помощью команды ls проверяю, что файл был создан (рис. [4.8])

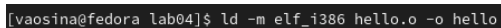


```
[vaosina@fedora lab04]$ ls  
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.8: Рис. 8. Проверка создания файла

4.4 Компоновщик LD

Чтобы получить исполняемую программу, передаю объектный файл на обработку компоновщику LD (рис. [4.9])



```
[vaosina@fedora lab04]$ ld -m elf_i386 hello.o -o hello
```

Рис. 4.9: Рис. 9. Передача файла компоновщику

С помощью команды ls проверяю, что исполняемый файл hello был создан. (рис. [4.10])

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

Рис. 4.10: Рис. 10. Проверка создания файла

Выполняю следующую команду: (рис. [4.11])

```
[vaosina@fedora lab04]$ ld -m elf_i386 obj.o -o main
```

Рис. 4.11: Рис. 11. Выполнение команды

Исполняемый файл имеет имя main, объектный файл имеет имя obj.o

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл, находящийся в текущем каталоге, при помощи команды ./hello (рис. [4.12])

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

Рис. 4.12: Рис. 12. Запуск созданного файла на выполнение

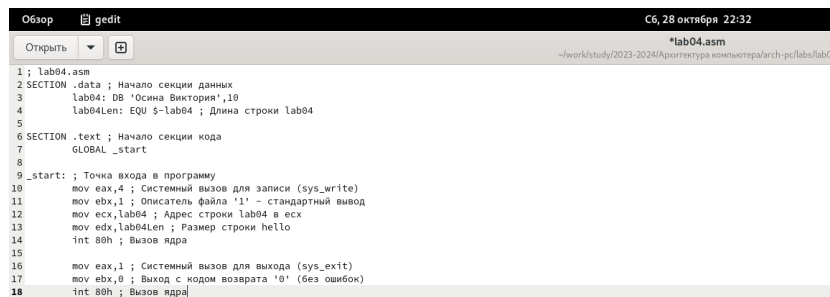
4.6 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm (рис. [4.13])

```
[vaosina@fedora lab04]$ cp hello.asm lab4.asm
```

Рис. 4.13: Рис. 13. Создание копий файлов

С помощью текстового редактора gedit вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. [4.14])



```
1; lab04.asm
2SECTION .data ; Начало секции данных
3    lab04: DB 'Осина Виктория',10
4    lab04Len: EQU $-lab04 ; Длина строки lab04
5
6SECTION .text ; Начало секции кода
7    GLOBAL _start
8
9_start: ; Точка входа в программу
10    mov eax,4 ; Системный вызов для записи (sys_write)
11    mov ebx,1 ; Описатель файла '1' - стандартный вывод
12    mov ecx,lab04 ; Адрес строки lab04 в ecx
13    mov edx,lab04Len ; Размер строки hello
14    int 80h ; Вызов ядра
15
16    mov eax,1 ; Системный вызов для выхода (sys_exit)
17    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18    int 80h ; Вызов ядра
```

Рис. 4.14: Рис. 14. Изменений текста программы

Транслирую полученный текст программы lab4.asm в объектный файл (рис. [4.15])



```
[vaosina@fedora lab04]$ nasm -f elf lab04.asm
```

Рис. 4.15: Рис. 15. Трансляция текста в объектный файл

Выполняю компоновку объектного файла (рис. [4.16])



```
[vaosina@fedora lab04]$ ld -m elf_i386 lab04.o -o lab04
```

Рис. 4.16: Рис. 16. Выполнение команд

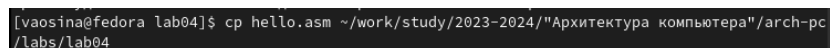
Запускаю получившийся исполняемый файл (рис. [4.17])



```
[vaosina@fedora lab04]$ ./lab04
Осина Виктория
[vaosina@fedora lab04]$
```

Рис. 4.17: Рис. 17. Запуск получившегося файла

Копирую файлы hello.asm и lab4.asm в мой локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. (рис. [4.18]), (рис. [4.19]) и (рис. [4.20])



```
[vaosina@fedora lab04]$ cp hello.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 4.18: Рис. 18. Создание копии файла hello.asm

```
[vaosina@fedora lab04]$ cp lab04.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 4.19: Рис. 19. Создание копии файла lab04.asm

```
[vaosina@fedora lab04]$ ls
hello.asm lab04.asm presentation report
[vaosina@fedora lab04]$
```

Рис. 4.20: Рис. 20. Проверка создания копий

Загружаю файлы на Github (рис. [4.21])

```
[vaosina@fedora lab04]$ git add .
[vaosina@fedora lab04]$ git commit -m "Added hello.asm and lab04.asm for lab04"
[master 42850c8] Added hello.asm and lab04.asm for lab04
4 files changed, 37 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab04.asm
create mode 100644 labs/lab04/report/report.docx
create mode 100644 labs/lab04/report/report.pdf
[vaosina@fedora lab04]$ git push
Перечисление объектов: 100% (13/13), готово.
Подсчет объектов: 100% (13/13), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (9/9), готово.
Запись объектов: 100% (9/9), 581.22 КиБ | 4.21 МБ/с, готово.
Всего 9 (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:urocean/study_2023-2024_arch-pc.git
d80d9c6..42850c8 master -> master
[vaosina@fedora lab04]$
```

Рис. 4.21: Рис. 21. Загрузка файлов на Github

5 Выводы

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

6 Список литературы

::: Архитектура ЭВМ - Лабораторная работа №4 :::