

Лабораторная работа №5

Создание и процесс обработки программ на языке ассемблера NASM

Медникова Екатерина Михайловна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	9
4	Выполнение самостоятельной работы	12
5	Выводы	15

Список иллюстраций

3.1	Создание каталога и переход в него	9
3.2	Создание файла	9
3.3	Открытие файла	9
3.4	Вставка текста	10
3.5	Компиляция текста	10
3.6	Компиляция и проверка	11
3.7	Результат выполненных команд	11
4.1	Создание копии файла	12
4.2	Внесение изменений в текст	12
4.3	Компиляция и запуск	13
4.4	Копирование файлов в локальный репозиторий	13
4.5	Загрузка файлов на Github	14

Список таблиц

1 Цель работы

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

2 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.

Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — **машинные коды**. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — **Ассемблер**.

Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто

переводит мнемонические обозначения команд в последовательности бит (нулей и единиц).

Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров x86, ARM, SPARC, PowerPC, M68k). Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера.

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 который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис 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.о. 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). Макси-

мальная 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.

3 Выполнение лабораторной работы

1. Создала каталог для работы с программами на языке ассемблера NASM и перешла в него.

```
[emmednikova@fedora work]$ mkdir arch-pc  
[emmednikova@fedora work]$ cd arch-pc  
[emmednikova@fedora arch-pc]$ mkdir lab05  
[emmednikova@fedora arch-pc]$ cd lab05  
[emmednikova@fedora lab05]$
```

Рис. 3.1: Создание каталога и переход в него

2. Создала текстовый файл с именем hello.asm.

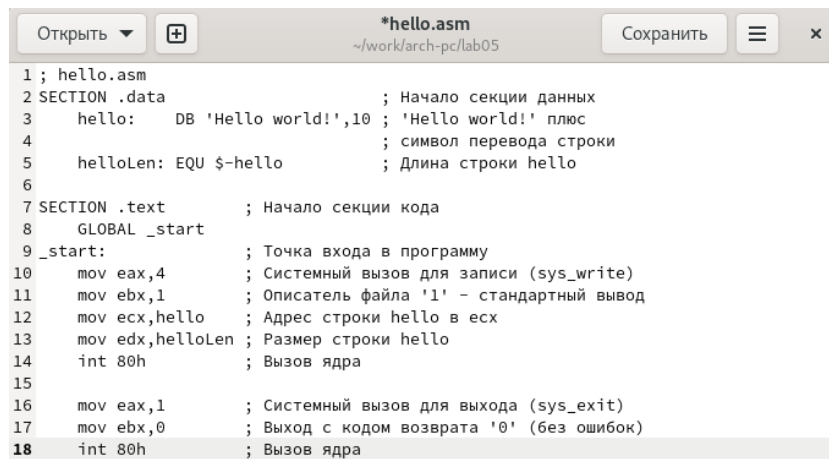
```
[emmednikova@fedora work]$ mkdir arch-pc  
[emmednikova@fedora work]$ cd arch-pc  
[emmednikova@fedora arch-pc]$ mkdir lab05  
[emmednikova@fedora arch-pc]$ cd lab05  
[emmednikova@fedora lab05]$ touch hello.asm  
[emmednikova@fedora lab05]$
```

Рис. 3.2: Создание файла

3. Открыла файл с помощью текстового редактора gedit и ввела в него текст.

```
[emmednikova@fedora work]$ mkdir arch-pc  
[emmednikova@fedora work]$ cd arch-pc  
[emmednikova@fedora arch-pc]$ mkdir lab05  
[emmednikova@fedora arch-pc]$ cd lab05  
[emmednikova@fedora lab05]$ touch hello.asm  
[emmednikova@fedora lab05]$ gedit hello.asm
```

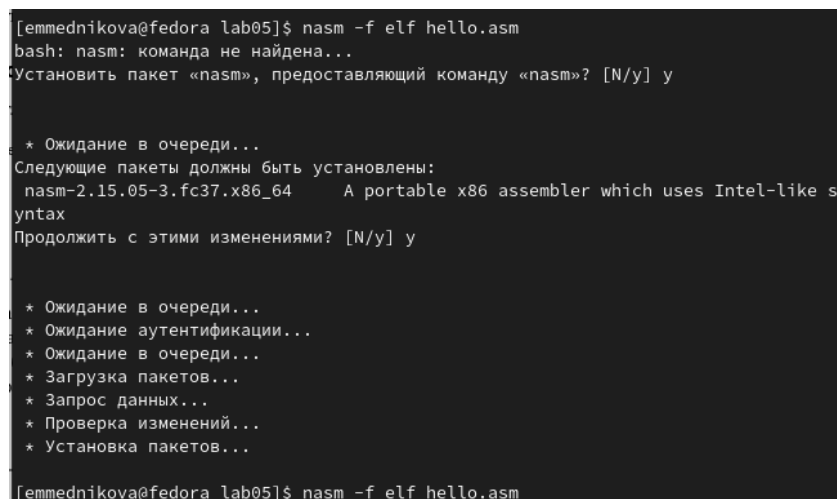
Рис. 3.3: Открытие файла



```
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 _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,hello       ; Адрес строки hello в ecx
13    mov edx,helloLen    ; 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. 3.4: Вставка текста

4. Провела компиляцию текста программы с помощью команды `nasm -f elf hello.asm`.



```
[emmednikova@fedora lab05]$ nasm -f elf hello.asm
bash: nasm: команда не найдена...
Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
Следующие пакеты должны быть установлены:
 nasm-2.15.05-3.fc37.x86_64      A portable x86 assembler which uses Intel-like s
yntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...
[emmednikova@fedora lab05]$ nasm -f elf hello.asm
```

Рис. 3.5: Компиляция текста

5. Скомпилировала файл с помощью команды `nasm -o obj.o -f elf -g -l list.lst hello.asm`. С помощью команды `ls` проверила, что файлы были созданы.

```
[emmednikova@fedora lab05]$ nasm -f elf hello.asm
[emmednikova@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[emmednikova@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o
[emmednikova@fedora lab05]$
```

Рис. 3.6: Компиляция и проверка

6. Передала файл на обработку компоновщику с помощью команды `ld -m elf_i386 hello.o -o hello`. Далее использовала команду `ld -m elf_i386 obj.o -o main`. Запустила на выполнение исполняемый файл с помощью команды `./hello`.

```
[emmednikova@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[emmednikova@fedora lab05]$ ld -m elf_i386 obj.o -o main
[emmednikova@fedora lab05]$ ./hello
Hello world!
[emmednikova@fedora lab05]$
```

Рис. 3.7: Результат выполненных команд

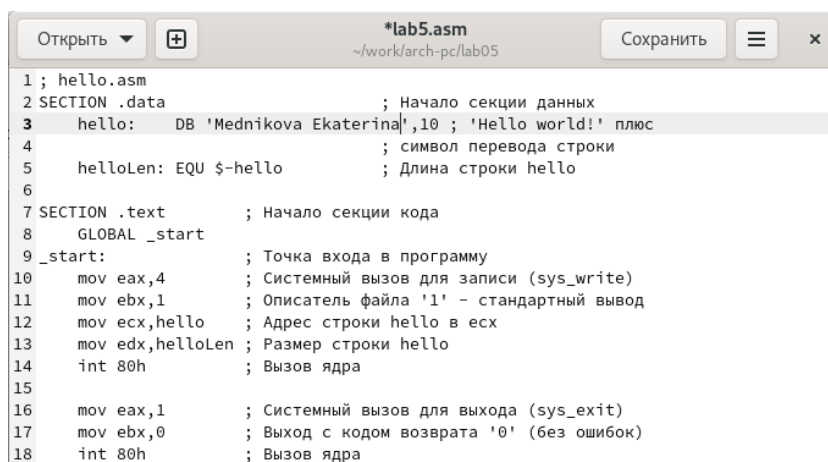
4 Выполнение самостоятельной работы

1. В каталоге ~/work/arch-pc/lab05 с помощью команды `cp` создала копию файла `hello.asm` с именем `lab5.asm`.

```
[emmednikova@fedora lab05]$ cp hello.asm lab5.asm
[emmednikova@fedora lab05]$ ls
hello.o  hello.asm  lab5.asm  list.lst  main  obj.o
[emmednikova@fedora lab05]$
```

Рис. 4.1: Создание копии файла

2. С помощью редактора `gedit` внесла изменения в текст программы в файле `lab5.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моей фамилией и моим именем.



```
*lab5.asm
~/work/arch-pc/lab05
Открыть + Сохранить ≡ ×

1 ; hello.asm
2 SECTION .data                ; Начало секции данных
3     hello: DB 'Mednikova Ekaterina',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 _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,hello            ; Адрес строки hello в ecx
13    mov edx,helloLen         ; 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. 4.2: Внесение изменений в текст

3. Оттранслировала полученный текст программы lab5.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл.

```
[emmednikova@fedora lab05]$ nasm -f elf lab5.asm
[emmednikova@fedora lab05]$ ls
hello hello.asm hello.o lab5.asm lab5.o list.lst main obj.o
[emmednikova@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst lab5.asm
[emmednikova@fedora lab05]$ ls
hello hello.asm hello.o lab5.asm lab5.o list.lst main obj.o
[emmednikova@fedora lab05]$ ld -m elf_i386 hello.o -o lab5
[emmednikova@fedora lab05]$ ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.lst main obj.o
[emmednikova@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
[emmednikova@fedora lab05]$ ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.lst main obj.o
[emmednikova@fedora lab05]$ ld -m elf_i386 lab5.o -o main
[emmednikova@fedora lab05]$ ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.lst main obj.o
[emmednikova@fedora lab05]$ ./lab5
Mednikova Ekaterina
```

Рис. 4.3: Компоновка и запуск

4. Скопировала файлы hello.asm и lab5.asm в свой локальный репозиторий в каталог ~/work/study/2022-2023/“Архитектура компьютера”/arch-pc/labs/lab05/. Загрузила файлы на Github.

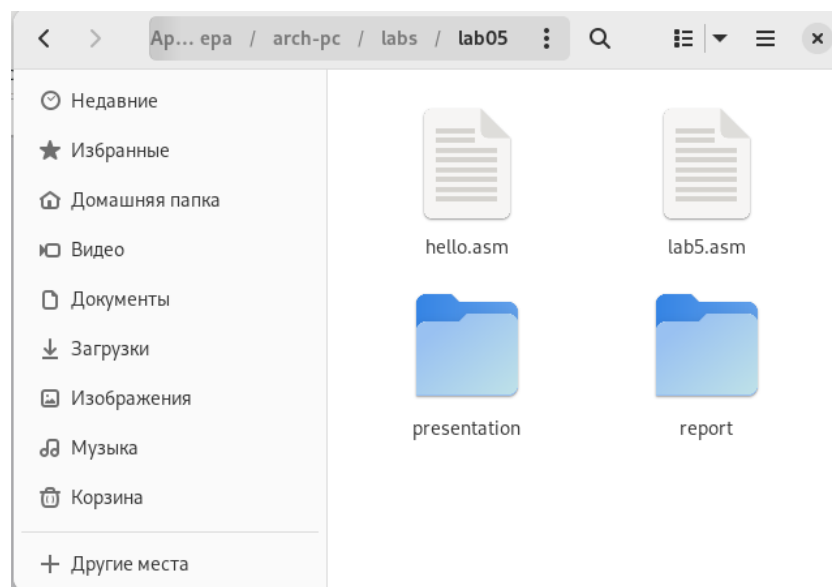


Рис. 4.4: Копирование файлов в локальный репозиторий

```
[emmednikova@fedora lab05]$ git add .
[emmednikova@fedora lab05]$ git commit -am "add files lab05"
[master 4a400a7] add files lab05
2 files changed, 36 insertions(+)
create mode 100644 labs/lab05/hello.asm
create mode 100644 labs/lab05/lab5.asm
[emmednikova@fedora lab05]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 964 байта | 74.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано п
акетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:emmednikova/study_2022-2023_arh-pc.git
be03f44..4a400a7 master -> master
[emmednikova@fedora lab05]$
```

Рис. 4.5: Загрузка файлов на Github

5 Выводы

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