

Отчёта по лабораторной работе №4

Создание и процесс обработки программ на языке ассемблера NASM

Спелов Андрей Николаевич

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Программа Hello world!	6
3.2	Транслятор NASM	7
3.3	Расширенный синтаксис командной строки NASM	7
3.4	Компоновщик LD	8
3.5	Запуск исполняемого файла	9
3.6	Задание для самостоятельной работы	9
4	Выводы	11

Список иллюстраций

3.1	Создаем каталоги с помощью команды <code>mkdir</code>	6
3.2	Переходим в каталог с помощью команды <code>cd</code>	6
3.3	Создаем текстовый файл <code>hello.asm</code>	6
3.4	Открываем файл и заполняем его по примеру	7
3.5	Используем команду <code>nasm</code>	7
3.6	Проверяем работу команды	7
3.7	Преобразуем файл <code>hello.asm</code> в <code>obj.o</code>	7
3.8	Проверяем создание файла командой <code>ls</code>	8
3.9	Используем команду <code>ld</code>	8
3.10	Используем команду <code>ls</code>	8
3.11	Используем команду <code>ld</code> , создавая файл <code>main</code>	8
3.12	Используем команду <code>ls</code>	8
3.13	Используем команду <code>./hello</code>	9
3.14	Используем команду <code>cp</code>	9
3.15	Открываем файл в текстовом редакторе	9
3.16	Редактируем файл для своего имени и фамилии	9
3.17	Прописываем команды для работы файла и запускаем программу	10
3.18	Копируем файлы в каталог с ЛР4	10
3.19	Загружаем файлы	10

1 Цель работы

Освоить процедуры компиляции и сборки программ, познакомиться с языком ассемблера NASM.

2 Задание

Написать 2 программы(Hello world, lab4(Имя Фамилия))

3 Выполнение лабораторной работы

3.1 Программа Hello world!

Создаем каталог для работы с программами на языке ассемблера NASM (рис. 3.1).

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

Рис. 3.1: Создаем каталоги с помощью команды mkdir

Переходим в созданный каталог (рис. 3.2).

```
[spelovandrei@fedora ~]$ cd ~/work/arch-pc/lab04  
[spelovandrei@fedora lab04]$
```

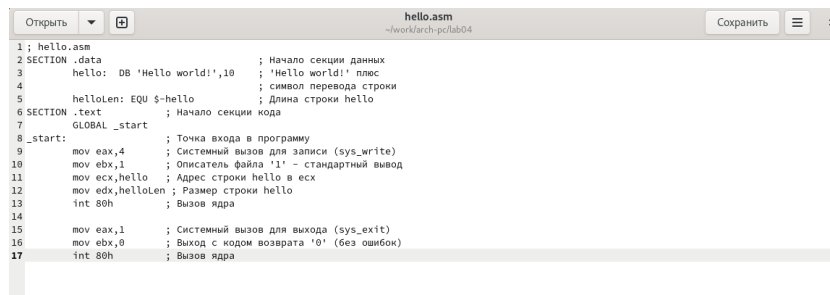
Рис. 3.2: Переходим в каталог с помощью команды cd

Создаем текстовый файл (рис. 3.3).

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

Рис. 3.3: Создаем текстовый файл hello.asm

Открываем данный файл в текстовом редакторе (рис. 3.4).

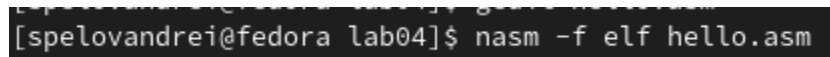


```
1: hello.asm
2 SECTION .data                ; Начало секции данных
3     hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4                                     ; символ перевода строки
5     hellolen: EQU $-hello      ; Длина строки hello
6 SECTION .text                ; Начало секции кода
7   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

Рис. 3.4: Открываем файл и заполняем его по примеру

3.2 Транслятор NASM

Преобразуем текст программы в объектный код (рис. 3.5).



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

Рис. 3.5: Используем команду nasm

Проверяем созданся ли объектный файл с помощью команды ls (рис. 3.6).

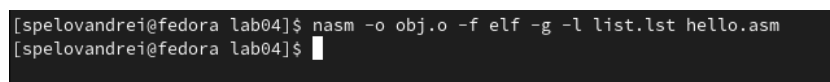


```
[spelovandrei@fedora lab04]$ ls
hello.asm  hello.o
[spelovandrei@fedora lab04]$
```

Рис. 3.6: Проверяем работу команды

3.3 Расширенный синтаксис командной строки NASM

Компилируем исходный файл (рис. 3.7).



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

Рис. 3.7: Преобразуем файл hello.asm в obj.o

Проверяем, как сработала команда (рис. 3.8).

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

Рис. 3.8: Проверяем создание файла командой ls

3.4 Компоновщик LD

Передаем объектный файл на обработку компоновщику (рис. 3.9).

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

Рис. 3.9: Используем команду ld

Проверяем создался ли исполняемый файл hello (рис. 3.10).

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

Рис. 3.10: Используем команду ls

Передаем объектный файл на обработку компоновщику (рис. 3.11).

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

Рис. 3.11: Используем команду ld, создавая файл main

Проверяем создался ли исполняемый файл hello (рис. 3.12).

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

Рис. 3.12: Используем команду ls

3.5 Запуск исполняемого файла

Запускаем на выполнение созданный исполняемый файл (рис. 3.13).

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

Рис. 3.13: Используем команду ./hello

3.6 Задание для самостоятельной работы

Создаем копию файла hello.asm (рис. 3.14).

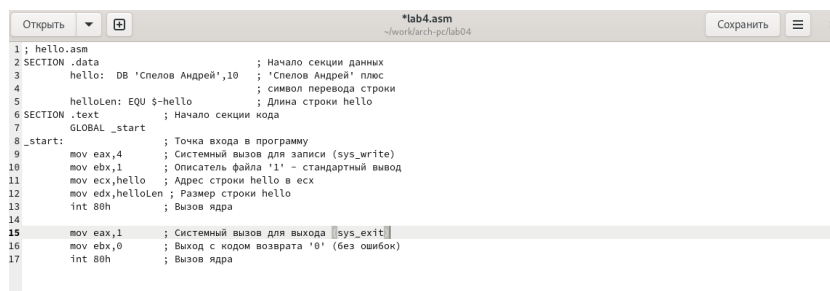
```
[spelovandrei@fedora lab04]$ cp hello.asm lab4.asm
[spelovandrei@fedora lab04]$
```

Рис. 3.14: Используем команду cp

Открываем файл и редактируем его (рис. 3.15).

```
[spelovandrei@fedora lab04]$ gedit lab4.asm
```

Рис. 3.15: Открываем файл в текстовом редакторе



```
1; hello.asm
2 SECTION .data
3     hello: DB 'Спелов Андрей',10 ; Начало секции данных
4           ; 'Спелов Андрей' плюс
5           ; символ перевода строки
6 SECTION .text
7     GLOBAL _start
8     _start:
9         mov eax,4 ; Точка входа в программу
10        mov ebx,1 ; Системный вызов для записи (sys_write)
11        mov ecx,hello ; Описание файла '1' - стандартный вывод
12        mov edx,helloLen ; Адрес строки hello в ecx
13        int 80h ; Размер строки hello
14        ; Вызов ядра
15        mov eax,1 ; Системный вызов для выхода [sys_exit]
16        mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
17        int 80h ; Вызов ядра
```

Рис. 3.16: Редактируем файл для своего имени и фамилии

Прописываем те же команды, что и с первой программой (рис. 3.17).

```
[spelovandrei@fedora lab04]$ nasm -f elf lab4.asm
[spelovandrei@fedora lab04]$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
[spelovandrei@fedora lab04]$ ld -m elf_i386 lab4.o -o hello
[spelovandrei@fedora lab04]$ ld -m elf_i386 obj.o -o main
[spelovandrei@fedora lab04]$ ./hello
Спелов Андрей
[spelovandrei@fedora lab04]$
```

Рис. 3.17: Прописываем команды для работы файла и запускаем программу

Копируем файлы в локальный репозиторий (рис. 3.18).

```
[spelovandrei@fedora lab04]$ cp hello.asm ~/work/study/2023-2024/"Архитектура ко
мпьютера"/arch-pc/labs/lab04/
[spelovandrei@fedora lab04]$ cp lab4.asm ~/work/study/2023-2024/"Архитектура ком
пьютера"/arch-pc/labs/lab04/
[spelovandrei@fedora lab04]$
```

Рис. 3.18: Копируем файлы в каталог с ЛР4

Переходим в каталог лабораторных работ и загружаем файлы на Github (рис. 3.19).

```
[spelovandrei@fedora lab04]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/
arch-pc
[spelovandrei@fedora arch-pc]$ git add .
[spelovandrei@fedora arch-pc]$ git commit -am 'feat(main): add files lab-4'
[master d70625c] feat(main): add files lab-4
2 files changed, 34 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
[spelovandrei@fedora arch-pc]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 979 байтов | 979.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использов
ано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:SpelovA/study_2023-2024_arh-pc.git
82f0b12..d70625c master -> master
[spelovandrei@fedora arch-pc]$
```

Рис. 3.19: Загружаем файлы

4 Выводы

Мы познакомились с языком ассемблера NASM и создали две работающих программы.