

Отчет по лабораторной работе №4

Дисциплина: Архитектура Компьютера

Иванов Сергей Владимирович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	9

Список иллюстраций

2.1	Файл hello.asm	5
2.2	Редактируем hello.asm	5
2.3	Преобразование в объектный файл	6
2.4	Команда компиляции	6
2.5	Исполняемая программа	6
2.6	Компиляция main	6
2.7	Запуск исполняемого файла	7
2.8	Копируем файл	7
2.9	Редактируем lab04	7
2.10	Компиляция и запуск lab04	8
2.11	Копируем и загружаем на Github	8

1 Цель работы

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

2 Выполнение лабораторной работы

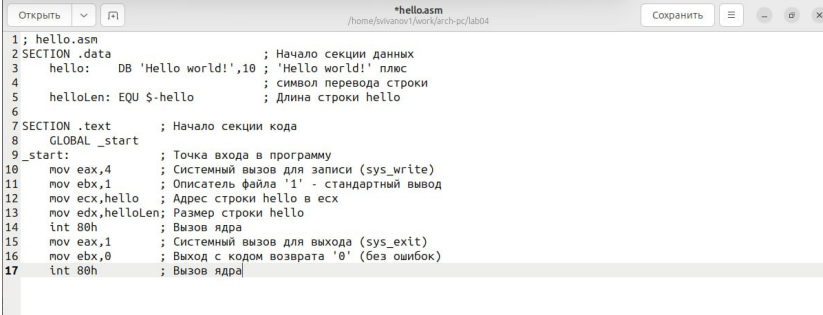
Создадим каталог для работы с программами на языке ассемблера NASM. Перейдем в созданный каталог, создадим текстовый файл с именем hello.asm и откроем его с помощью gedit. (Рис. 2.1)

```
svivanov1@svivanov1:~$ mkdir -p ~/work/arch-pc/lab04
svivanov1@svivanov1:~$ cd ~/work/arch-pc/lab04
svivanov1@svivanov1:~/work/arch-pc/lab04$ touch hello.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ gedit hello.asm
Gtk-Message: 13:07:29.623: Not loading module "atk-bridge": The functionality is
provided by GTK natively. Please try to not load it.

(gedit:2928): GLib-GIO-WARNING **: 13:07:29.921: Error creating IO channel for /
proc/self/mountinfo: Permission denied (g-file-error-quark, 2)
```

Рис. 2.1: Файл hello.asm

Введем в него следующий текст и сохраним. (Рис. 2.2)



```
1; hello.asm
2SECTION .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
7SECTION .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   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. 2.2: Редактируем hello.asm

Преобразуем текст программы в объектный код и проверим командой ls. (Рис. 2.3)

```
svivanov1@svivanov1:~/work/arch-pc/lab04$ nasm -f elf hello.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ ls
hello.asm hello.o
svivanov1@svivanov1:~/work/arch-pc/lab04$
```

Рис. 2.3: Преобразование в объектный файл

Выполним команду, которая скомпилирует исходный файл hello.asm в obj.o, при этом формат файла будет elf, и в него будут включены символы для отладки а также будет создан файл листинга. Проверим правильность командой ls. (Рис. 2.4)

```
svivanov1@svivanov1:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
svivanov1@svivanov1:~/work/arch-pc/lab04$
```

Рис. 2.4: Команда компиляции

Получим исполняемую программу, передав объектный файл на обработку компоновщику. Проверим командой ls. (Рис. 2.5)

```
svivanov1@svivanov1:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
svivanov1@svivanov1:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
svivanov1@svivanov1:~/work/arch-pc/lab04$
```

Рис. 2.5: Исполняемая программа

Выполним следующую команду. Исполняемый файл имеет имя main. Обыкновенный файл имеет имя obj.o. (Рис. 2.6)

```
svivanov1@svivanov1:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
svivanov1@svivanov1:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o
svivanov1@svivanov1:~/work/arch-pc/lab04$
```

Рис. 2.6: Компиляция main

Запустим исполняемый файл командой ./hello (Рис. 2.7)

```
svivanov1@svivanov1:~/work/arch-pc/lab04$ ./hello
Hello world!
svivanov1@svivanov1:~/work/arch-pc/lab04$
```

Рис. 2.7: Запуск исполняемого файла

В этом же каталоге с помощью команды `cp` создадим копию файла `hello.asm` с именем `lab04.asm` (Рис. 2.8)

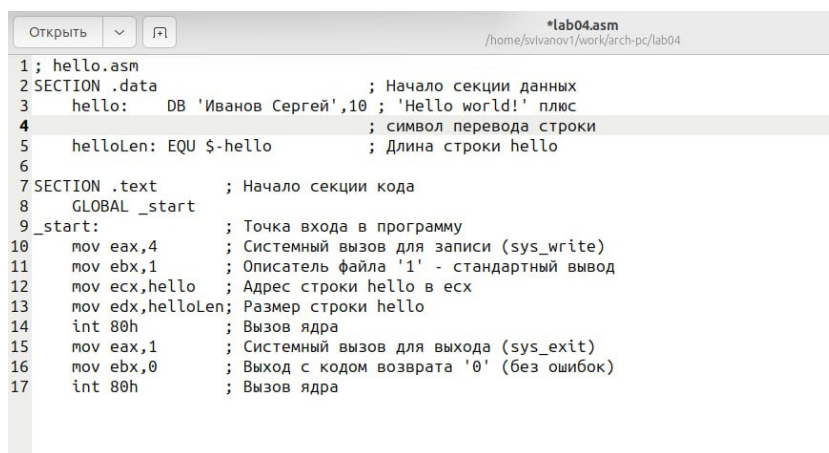
```
svivanov1@svivanov1:~/work/arch-pc/lab04$ cp hello.asm lab04.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ gedit lab04.asm
Gtk-Message: 15:59:38.635: Not loading module "atk-bridge": The functionality is
provided by GTK natively. Please try to not load it.

(gedit:1888): GLib-GIO-WARNING **: 15:59:38.922: Error creating IO channel for /
proc/self/mountinfo: Permission denied (g-file-error-quark, 2)

** (gedit:1888): WARNING **: 16:00:04.085: atk-bridge: get_device_events_reply:
unknown signature
```

Рис. 2.8: Копируем файл

С помощью `gedit` редактируем так, чтобы на экран выводилась строка с моими фамилией и именем вместо `Hello world!`. (Рис. 2.9)



```
*lab04.asm
/home/svivanov1/work/arch-pc/lab04

1; hello.asm
2 SECTION .data                ; Начало секции данных
3     hello:    DB 'Иванов Сергей',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    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. 2.9: Редактируем lab04

Оттранслируем полученный текст программы в объектный файл. Выполним компоновку объектного файла и запустим исполняемый файл. (Рис. 2.10)

```

svivanov1@svivanov1:~/work/arch-pc/lab04$ nasm -f elf lab04.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ nasm -o lab04.o -f elf -g lab04.asm
svivanov1@svivanov1:~/work/arch-pc/lab04$ ld -m elf_i386 lab04.o -o lab04
svivanov1@svivanov1:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o
svivanov1@svivanov1:~/work/arch-pc/lab04$ ./lab04
Иванов Сергей
svivanov1@svivanov1:~/work/arch-pc/lab04$

```

Рис. 2.10: Компиляция и запуск lab04

Скопируем файлы hello.asm и lab04.asm в локальный репозиторий и загрузим файлы на Github. (Рис. 2.11)

```

svivanov1@svivanov1:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
svivanov1@svivanov1:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): add files lab-4'
[master 9557e6e] feat(main): add files lab-4
2 files changed, 34 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab04.asm
svivanov1@svivanov1:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 983 байта | 983.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано 0 пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:SergeyIvanov21/study_2023-2024_arh-pc.git
2e28271..9557e6e master -> master
svivanov1@svivanov1:~/work/study/2023-2024/Архитектура компьютера/arch-pc$

```

Рис. 2.11: Копируем и загружаем на Github

3 Выводы

В ходе выполнения лабораторной работы мы освоили процедуры компиляции и сборки программ, написанных на ассемблере NASM.