

Отчет по лабораторной работе №4

Простейший вариант

Чигладзе Майя Владиславовна

Содержание

1	Цель работы	5
2	Порядок выполнения лабораторной работы	6
2.1	Программа Hello world!	6
2.2	Транслятор NASM	7
2.3	Расширенный синтаксис командной строки NASM	7
2.4	Компоновщик LD	8
2.5	Запуск исполняемого файла	8
3	Задание для самостоятельной работы	10
3.1	Задание 1	10
3.2	Задание 2	10
3.3	Задание 3	11
3.4	Задание 4	11
4	Выводы	12
	Список литературы	13

Список иллюстраций

2.1	Рисунок 1 - Создание нужных файлов	6
2.2	Рисунок 2 - Введенный текст	6
2.3	Рисунок 3 - Превращение	7
2.4	Рисунок 4 - Расширенный синтаксис	8
2.5	Рисунок 5 - Компоновщик ld	8
2.6	Рисунок 6 - Компоновщик ld	9
3.1	Рисунок 7 - Копируем	10
3.2	Рисунок 8 - Вносим изменения	10
3.3	Рисунок 9 - Процесс приводящий к запуску файла	11
3.4	Рисунок 10 - Копируем и добавляем	11
3.5	Рисунок 11 - Комитим и пушим	11

Список таблиц

1 Цель работы

Изучение процесса компиляции и сборки программ, написанных с использованием ассемблера NASM

2 Порядок выполнения лабораторной работы

2.1 Программа Hello world!

Создадим новый каталог и перейдем в него. В нем создадим текстовый файла формата asm и откроем его (Рисунок 1)

```
mvchigladze@dk8n57 ~ $ mkdir -p ~/work/arch-pc/lab04
mvchigladze@dk8n57 ~ $ cd ~/work/arch-pc/lab04
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ touch hello.asm
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 2.1: Рисунок 1 - Создание нужных файлов

Введем в него нужный текст, соблюдая строки и регистр (Рисунок 2).

*report.md	*hello.asm
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	mov eax,1 ; Системный вызов для выхода (sys_exit)
15	mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16	int 80h ; Вызов ядра

Рис. 2.2: Рисунок 2 - Введенный текст

2.2 Транслятор NASM

С помощью NASM превратим текст программы в объектный код и проверим созданный файл, командой `ls` (Рисунок 3). Новое имя объектного файла - `hello.o`. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF.

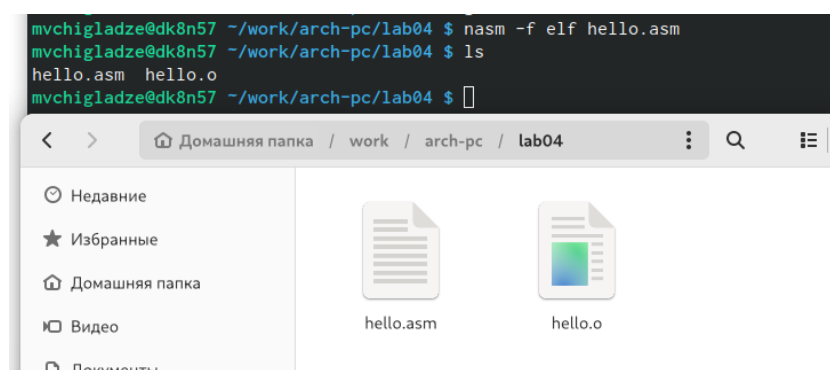


Рис. 2.3: Рисунок 3 - Превращение

2.3 Расширенный синтаксис командной строки NASM

Первой командой на Рисунке 4 мы скомпилировали исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла стал `elf`, и в него были включены символы для отладки (опция `-g`), кроме того, был создан файл листинга `list.lst` (опция `-l`) (Рисунок 4). С помощью `ls` проверили создание файлов, а также узнали подробную информацию о `nasm` и получили список форматов объектного файла (Рис. 4).

```

mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ man nasm
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ nasm -hf
Usage: nasm [-@ response_file] [options...] [--] filename
        nasm -v (or --v)

Options (values in brackets indicate defaults):

  -h          show this text and exit (also --help)
  -v (or --v) print the NASM version number and exit
  -@ file      response file; one command line option per line

  -o outfile   write output to outfile
  --keep-all  output files will not be removed even if an error happens

```

Рис. 2.4: Рисунок 4 - Расширенный синтаксис

2.4 Компоновщик LD

Передаем файл на обработку компоновщику и выполняем проверку созданных файлов (Рисунок 5). Исполняемый файл - main, объектный файл - obj.o. Также посмотрим на формат командной строки.

```

mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ld --help
Использование ld [параметры] файл...
Параметры:
  -а КЛЮЧЕВОЕ СЛОВО          Управление общей библиотекой для совместимости с HP/UX
  -A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА      Задать архитектуру
  -b ЦЕЛЬ, --format ЦЕЛЬ      Задать цель для следующих входных файлов
  -с ФАЙЛ, --mri-script ФАЙЛ  Прочитать сценарий компоновщика в формате MRI
  -d, -dc, -dp                Принудительно делать общие символы определёнными
  --dependency-file ФАЙЛ      Write dependency file

```

Рис. 2.5: Рисунок 5 - Компоновщик ld

2.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге (Рисунок 6).


```
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ./hello
Hello world!
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $
```

Рис. 2.6: Рисунок 6 - Компоновщик ld

3 Задание для самостоятельной работы

3.1 Задание 1

Скопируем файл `hello.asm` в эту же директорию, но с названием `lab4.asm` и проверим (Рисунок 7).

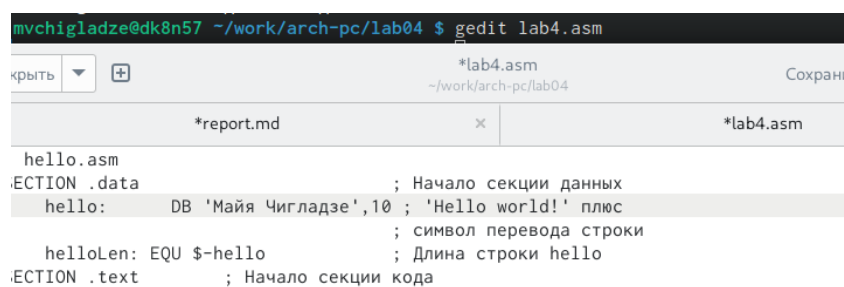
```
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/hello.asm ~/work/arch-pc/lab04/lab4.asm
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 3.1: Рисунок 7 - Копируем

3.2 Задание 2

С помощью текстового редактора внесем изменение в текст программы с моей фамилией и именем (Рисунок 8)

```
mvchigladze@dk8n57 ~/work/arch-pc/lab04 $ gedit lab4.asm
```



```
*lab4.asm
~/work/arch-pc/lab04
Сохранить

*report.md x *lab4.asm

hello.asm
SECTION .data
    hello: DB 'Майя Чигладзе',10 ; 'Hello world!' плюс
            ; символ перевода строки
    helloLen: EQU $-hello
            ; Длина строки hello
SECTION .text
            ; Начало секции кода
```

Рис. 3.2: Рисунок 8 - Вносим изменения

3.3 Задание 3

Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл (Рисунок 9).

```
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ gedit lab4.asm
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ ./lab4
Майя Чигладзе
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $
```

Рис. 3.3: Рисунок 9 - Процесс приводящий к запуску файла

3.4 Задание 4

Скопируем файлы hello.asm и lab4.asm в локальный репозиторий и загрузим файлы на Github (Рисунок 10 и Рисунок 11).

```
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/hello.asm ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/lab4.asm ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04
mvchigladze@dk8n57: ~/work/arch-pc/lab04 $ cd ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04
mvchigladze@dk8n57: ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04 $ git add .
```

Рис. 3.4: Рисунок 10 - Копируем и добавляем

```
mvchigladze@dk8n57: ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04 $ git commit -m "Лабораторная 4"
[master 5074449] Лабораторная 4
13 files changed, 77 insertions(+), 31 deletions(-)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
delete mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab04/report/image/Лаба4.1.png
create mode 100644 labs/lab04/report/image/Лаба4.2.png
create mode 100644 labs/lab04/report/image/Лаба4.3.png
create mode 100644 labs/lab04/report/image/Лаба4.4.png
create mode 100644 labs/lab04/report/image/Лаба4.5.png
create mode 100644 labs/lab04/report/image/Лаба4.6.png
create mode 100644 labs/lab04/report/image/Лаба4.7.png
create mode 100644 labs/lab04/report/image/Лаба4.8.png
create mode 100644 labs/lab04/report/image/Лаба4.9.png
mvchigladze@dk8n57: ~/work/study/2023-2024/Архитектура\ компьютера/study_2023-2024_arhpc/labs/lab04 $ git push
Перечисление объектов: 24, готово.
Подсчет объектов: 100% (24/24), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (18/18), готово.
```

Рис. 3.5: Рисунок 11 - Комитим и пушим

4 Выводы

В ходе лабораторной работы, я изучила процесс компиляции и сборки программ, написанных с использованием ассемблера NASM

Список литературы