

Отчёта по лабораторной работе №4

Дисциплина: архитектура компьютера

Шибаета Алесандра Алексеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
4.1	Создание программы Hello world!	9
4.2	Работа с транслятором NASM	10
4.3	Работа с расширенным синтаксисом командной строки NASM . .	11
4.4	Работа с компоновщиком LD	11
4.5	Запуск исполняемого файла	12
4.6	Выполнение заданий для самостоятельной работы.	12
5	Выводы	16
6	Список литературы	17

Список иллюстраций

4.1	Перемещение между директориями	9
4.2	Создание пустого файла	9
4.3	Открытие файла в текстовом редакторе	9
4.4	Заполнение файла	10
4.5	Компиляция текста программы	10
4.6	Компиляция текста программы	11
4.7	Передача объектного файла на обработку компоновщику	11
4.8	Передача объектного файла на обработку компоновщику	12
4.9	Запуск исполняемого файла	12
4.10	Создание копии файла	12
4.11	Изменение программы	13
4.12	Компиляция текста программы	13
4.13	Передача объектного файла на обработку компоновщику	13
4.14	Запуск исполняемого файла	13
4.15	Создании копии файлов в новом каталоге	14
4.16	Удаление лишних файлов в текущем каталоге	14
4.17	Добавление файлов на GitHub	15
4.18	Отправка файлов	15

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства: - арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; - регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных

хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI — 64-битные - EAX, ECX, EDX, EBX, ESI, EDI — 32-битные - AX, CX, DX, BX, SI, DI — 16-битные - AH, AL, CH, CL, DH, DL, BH, BL — 8-битные

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды;

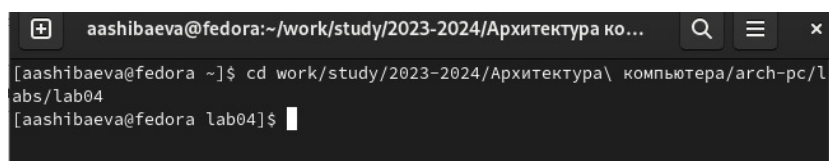
4. переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. 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.

4 Выполнение лабораторной работы

4.1 Создание программы Hello world!

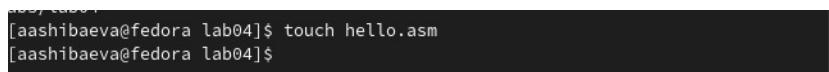
С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. [4.1]).



```
aashibaeva@fedora:~/work/study/2023-2024/Архитектура ко...
[aashibaeva@fedora ~]$ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab04
[aashibaeva@fedora lab04]$
```

Рис. 4.1: Перемещение между директориями

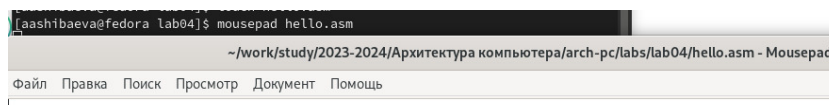
Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. [4.2]).



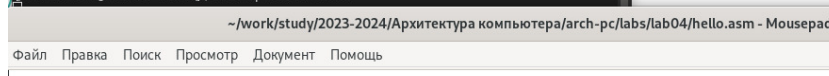
```
[aashibaeva@fedora lab04]$ touch hello.asm
[aashibaeva@fedora lab04]$
```

Рис. 4.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе `mousepad` (рис. [4.3]).



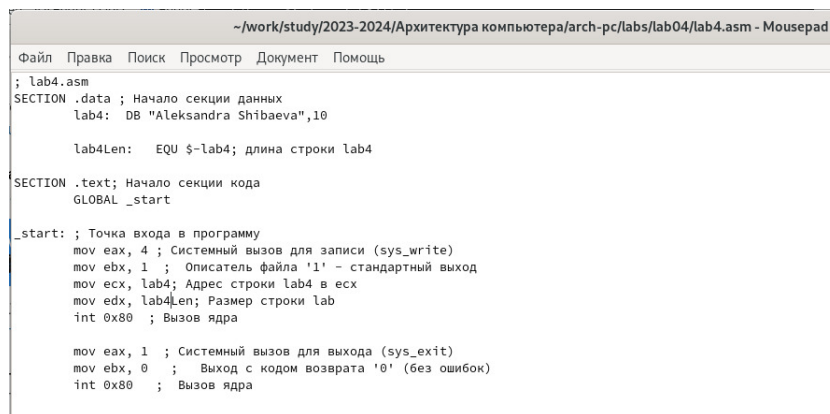
```
[aashibaeva@fedora lab04]$ mousepad hello.asm
```



~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/hello.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь

Рис. 4.3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. [4.4]).



```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/lab4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
; lab4.asm
SECTION .data ; Начало секции данных
    lab4: DB "Aleksandra Shibaeva",10

    lab4Len: EQU $-lab4; длина строки lab4

SECTION .text; Начало секции кода
GLOBAL _start

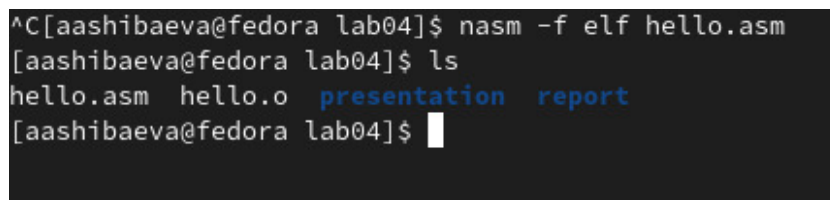
_start: ; Точка входа в программу
    mov eax, 4 ; Системный вызов для записи (sys_write)
    mov ebx, 1 ; Описатель файла '1' - стандартный выход
    mov ecx, lab4; Адрес строки lab4 в ecx
    mov edx, lab4Len; Размер строки lab
    int 0x80 ; Вызов ядра

    mov eax, 1 ; Системный вызов для выхода (sys_exit)
    mov ebx, 0 ; Выход с кодом возврата '0' (без ошибок)
    int 0x80 ; Вызов ядра
```

Рис. 4.4: Заполнение файла

4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. [4.5]). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”.



```
^C[aashibaeva@fedora lab04]$ nasm -f elf hello.asm
[aashibaeva@fedora lab04]$ ls
hello.asm  hello.o  presentation  report
[aashibaeva@fedora lab04]$
```

Рис. 4.5: Компиляция текста программы

4.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. [4.6]). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

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

Рис. 4.6: Компиляция текста программы

4.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `hello` (рис. [4.7]). Ключ `-o` задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

```
[aashibaeva@fedora lab04]$ ld -m elf_i386 hello.o -o hello
[aashibaeva@fedora lab04]$ ls
hello hello.asm hello.o list.lst obj.o presentation report
[aashibaeva@fedora lab04]$
```

Рис. 4.7: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. [4.8]). Исполняемый файл будет иметь имя `main`, т.к. после ключа `-o` было задано значение `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o`

```

[aashibaeva@fedora lab04]$ ld -m elf_i386 hello.o -o main
[aashibaeva@fedora lab04]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
[aashibaeva@fedora lab04]$

```

Рис. 4.8: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [4.9]).

```

[aashibaeva@fedora lab04]$ ./hello
Hello, world!
[aashibaeva@fedora lab04]$

```

Рис. 4.9: Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы.

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. [4.10]).

```

Hello, world!
[aashibaeva@fedora lab04]$ cp hello.asm lab4.asm
[aashibaeva@fedora lab04]$

```

Рис. 4.10: Создание копии файла

С помощью текстового редактора mousepad открываю файл lab5.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. [4.11]).

```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/lab4.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь

; lab4.asm
SECTION .data ; Начало секции данных
    lab4: DB "Aleksandra Shibaeva",10

    lab4Len: EQU $-lab4; длина строки lab4

SECTION .text; Начало секции кода
GLOBAL _start

_start: ; Точка входа в программу
    mov eax, 4 ; Системный вызов для записи (sys_write)
    mov ebx, 1 ; Описатель файла '1' - стандартный выход
    mov ecx, lab4; Адрес строки lab4 в ecx
    mov edx, lab4Len; Размер строки lab
    int 0x80 ; Вызов ядра

    mov eax, 1 ; Системный вызов для выхода (sys_exit)
    mov ebx, 0 ; Выход с кодом возврата '0' (без ошибок)
    int 0x80 ; Вызов ядра
```

Рис. 4.11: Изменение программы

Компилирую текст программы в объектный файл (рис. [4.12]). Проверяю с помощью утилиты ls, что файл lab5.o создан.

```
^C[aashibaeva@fedora lab04]$ nasm -f elf lab4.asm
[aashibaeva@fedora lab04]$ ls
hello.o lab4.o main presentation
hello.asm lab4.asm list.lst obj.o report
[aashibaeva@fedora lab04]$
```

Рис. 4.12: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. [4.13]).

```
[aashibaeva@fedora lab04]$ ld -m elf_i386 lab4.o -o lab4
[aashibaeva@fedora lab04]$ ls
hello.o lab4.asm list.lst obj.o report
hello.asm lab4 lab4.o main presentation
[aashibaeva@fedora lab04]$
```

Рис. 4.13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. [4.14]).

```
hello.asm lab4 lab4.o main presentation
[aashibaeva@fedora lab04]$ ./lab4
Aleksandra Shibaeva
[aashibaeva@fedora lab04]$
```

Рис. 4.14: Запуск исполняемого файла

К сожалению, я начала работу не в том каталоге, поэтому создаю другую директорию lab05 с помощью mkdir, прописывая полный путь к каталогу, в котором хочу создать эту директорию. Далее копирую из текущего каталога файлы, созданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ *, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. [4.15]). Проверяю с помощью утилиты ls правильность выполнения команды.

```
[aashibaeva@fedora lab04]$ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
[aashibaeva@fedora lab04]$ cp ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
cp: после '/home/aashibaeva/work/study/2023-2024/Архитектура компьютера/arch-pc/lab04' пропущен операнд, задающий целевой файл
По команде «cp --help» можно получить дополнительную информацию.
[aashibaeva@fedora lab04]$ cp * ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
[aashibaeva@fedora lab04]$ ls
hello      hello.o  lab4.asm  list.lst  obj.o      report
hello.asm  lab4     lab4.o    main      presentation
```

Рис. 4.15: Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории (рис. [4.16]).

```
[aashibaeva@fedora lab04]$ ls
hello.asm  lab4.asm  presentation  report
[aashibaeva@fedora lab04]$
```

Рис. 4.16: Удаление лишних файлов в текущем каталоге

С помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №5 (рис. [4.17]).

```
[aashibaeva@fedora lab04]$ git add .
[aashibaeva@fedora lab04]$ git commit -m "Add fales for lab04"
[master b1b7e04] Add fales for lab04
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
[aashibaeva@fedora lab04]$
```

Рис. 4.17: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push` (рис. [4.18]).

```
[aashibaeva@fedora arch-pc]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.08 КиБ | 552.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:aashibaeva/study_2023-2024_arh-pc.git
 989a4d2..b1b7e04 master -> master
[aashibaeva@fedora arch-pc]$
```

Рис. 4.18: Отправка файлов

5 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы

1. Архитектура ЭВМ