

Лабораторная работа №4

Дисциплина: Архитектура компьютера

Серебрякова Дарья Ильинична

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Программа Hello world!	10
4.2	Транслятор NASM	11
4.3	Расширенный синтаксис командной строки NASM	12
4.4	Компоновщик LD	12
4.5	Запуск исполняемого файла	13
5	Выполнение заданий для самостоятельной работы	14
6	Выводы	17
	Список литературы	18

Список иллюстраций

4.1	Начало заполнения отчета	10
4.2	Создание текстового файла	10
4.3	Создание текстового файла	11
4.4	Превращение текста в объектный код	11
4.5	Компиляция файлов	12
4.6	Получение исполняемого файла	12
4.7	Получение исполняемого файла с новым именем	12
4.8	Запуск исполняемого файла	13
5.1	Создание копии файла с новым именем	14
5.2	Редактирование файла lab4.asm	14
5.3	Компиляция текста в объектный файл	15
5.4	Запуск исполняемого файла	15
5.5	Создание нужной папки	15
5.6	Перенос файлов	16
5.7	Удаление файлов	16

Список таблиц

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:

формирование адреса в памяти очередной команды; считывание кода коман-

ды из памяти и её дешифрация; выполнение команды; 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

Захожу в каталог Лабораторной работы 4, проверяю его содержимое и убеждаюсь, что шаблон для заполнения отчета присутствует. Копирую его с новым именем – Л04_Серебрякова_отчет и приступаю к его заполнению по ходу выполнения лабораторной работы (рис. 4.1).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04/report$ ls
bib image Makefile pandoc report.md
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04/report$ cp report.md Л04_Серебрякова_отчет.md
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04/report$ rm report.md
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04/report$ ls
bib image Makefile pandoc Л04_Серебрякова_отчет.md
```

Рис. 4.1: Начало заполнения отчета

4.1 Программа Hello world!

Командой touch создаю текстовый файл с именем hello.asm (рис. 4.2).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ touch hello.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello.asm presentation report
```

Рис. 4.2: Создание текстового файла

Открываю файл с помощью текстового редактора mouserad и ввожу в него предложенный текст (рис. 4.3).

```

*~/work/study/study_2024-2025_arh-pc/labs/lab04/hello.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
; hello.asm
SECTION .data                ; Начало секции данных
    hello: DB 'Hello world!',10 ; 'Hello world!' плюс
                                ; символ перевода строки
    helloLen: EQU $-hello    ; Длина строки hello

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 вывод
    mov ecx,hello            ; Адрес строки hello в ecx
    mov edx,helloLen         ; Размер строки hello
    int 80h                  ; Вызов ядра
    mov eax,1                ; Системный вызов для выхода (sys_exit)
    mov ebx,0                ; Выход с кодом возврата '0' (без ошибок)
    int 80h                  ; Вызов ядра

```

Рис. 4.3: Создание текстового файла

4.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» пишу команду `nasm -f elf hello.asm`. Для выполнения этого шага потребовалось установить пакет, предоставляющий команду `nasm`. После установки, прописываю команду повторно, затем использую команду `ls` и вижу, что создан файл `hello.o` (рис. 4.4).

```

diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ nasm -f elf hello.asm
bash: nasm: команда не найдена...
Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
nasm-2.16.01-7.fc40.x86_64    A portable x86 assembler which uses Intel-like syntax
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ nasm -f elf hello.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report

```

Рис. 4.4: Превращение текста в объектный код

4.3 Расширенный синтаксис командной строки NASM

Ввожу команду `nasm -o obj.o -f elf -g -l list.lst hello.asm`. Данная команда должна скомпилировать исходный файл `hello.asm` в `obj.o` (опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`), при этом формат выходного файла должен быть `elf`, и в него должны быть включены символы для отладки (опция `-g`). Кроме того, должен быть создан файл листинга `list.lst` (опция `-l`). С помощью команды `ls` проверяю, что файлы были созданы (рис. 4.5).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list
t.lst hello.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$
```

Рис. 4.5: Компиляция файлов

4.4 Компоновщик LD

Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello`. Командой `ls` проверяю, что файл создан (рис. 4.6).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
```

Рис. 4.6: Получение исполняемого файла

Ключ `-o` с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Для наглядности ввожу следующую команду: `ld -m elf_i386 obj.o -o main` (рис. 4.7).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$
```

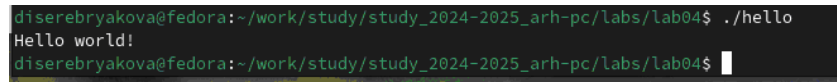
Рис. 4.7: Получение исполняемого файла с новым именем

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

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке ./hello (рис. 4.8).

A screenshot of a terminal window with a dark background. The prompt is 'diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04\$'. The user enters './hello' and the output is 'Hello world!'. The prompt is then shown again with a cursor at the end.

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ./hello
Hello world!
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$
```

Рис. 4.8: Запуск исполняемого файла

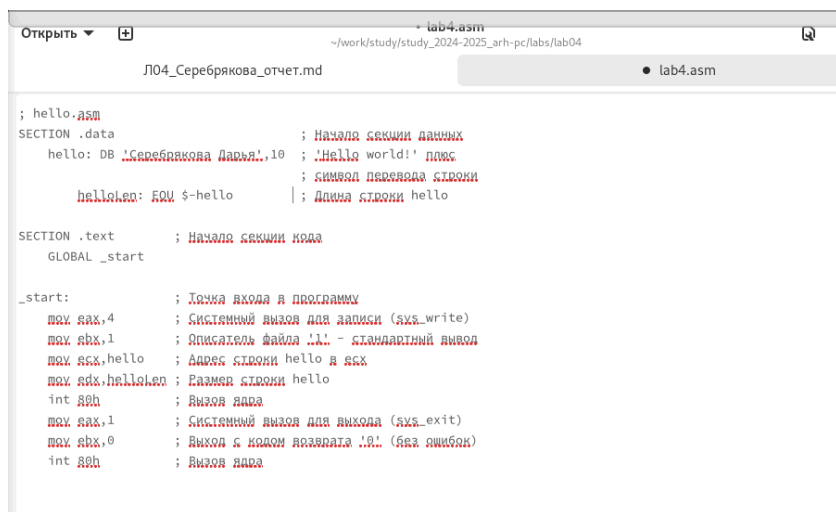
5 Выполнение заданий для самостоятельной работы

С помощью команды `cp` создаю копию файла `hello.asm` с именем `lab4.asm` (рис. 5.1).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ cp hello.asm lab4.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$
```

Рис. 5.1: Создание копии файла с новым именем

Открываю файл `lab4.asm` в текстовом редакторе `mousepad`, редактирую его так, чтобы вместо `Hello world!` на экран выводилась строка с моими фамилией и именем (рис. 5.2).



```
Открыть ▾ [+]  
~ /work/study/study_2024-2025_arh-pc/labs/lab04  
ЛО4_Церебрякова.отчет.md  
• lab4.asm  
; hello.asm  
SECTION .data  
    hello: DB "Церебрякова Дарья",10 ; 'Hello world!' логос  
    ; символ перевода строки  
    helloLen: EQU $-hello ; Длина строки hello  
  
SECTION .text  
    GLOBAL _start  
  
_start:  
    ; Точка входа в программу  
    mov eax,4 ; Системный вызов для записи (sys_write)  
    mov ebx,1 ; Описание файла '1' - стандартный вывод  
    mov ecx,hello ; Адрес строки hello в eax  
    mov edx,helloLen ; Размер строки hello  
    int 0x0 ; Вызов ядра  
  
    mov eax,1 ; Системный вызов для выхода (sys_exit)  
    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)  
    int 0x0 ; Выход ядра
```

Рис. 5.2: Редактирование файла `lab4.asm`

Компилирую текст программы в объектный файл. Проверяю командой `ls`, что

файл lab4.o создан (рис. 5.3).

```
hello hello.asm hello.o lab4.asm list.lst main obj.o presentation report
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ nasm -f elf lab4.asm
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
```

Рис. 5.3: Компиляция текста в объектный файл

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4. Запускаю исполняемый файл lab4, и вижу, что на экран действительно выводятся мои фамилия и имя (рис. 5.4).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ./lab4
Серебрякова Дарья
```

Рис. 5.4: Запуск исполняемого файла

На данном этапе выполнения заданий я заметила, что в начале выполнения лабораторной работы неправильно поняла один из шагов, а точнее, не создала каталог lab04, а начала работать в уже существующем каталоге lab4, находящемся в папке labs. Чтобы исправиться, возвращаюсь из папки labs и создаю необходимый каталог командой mkdir (рис. 5.5).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ cd ..
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs$ cd ..
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc$ mkdir lab04
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc$
```

Рис. 5.5: Создание нужной папки

Теперь необходимо перенести созданные файлы в новую папку. Копирую из текущего каталога файлы 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. Проверяю с помощью ls правильность выполнения команды (рис. 5.6).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ cp * ~/work/study/study_2024-2025_
arh-pc/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ cd ..
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs$ cd ..
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc$ ls
CHANGELOG.md  COURSE  labs      Makefile  presentation  README.git-flow.md  template
config        lab04   LICENSE  prepare  README.en.md  README.md
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc$ cd lab04
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab04$
```

Рис. 5.6: Перенос файлов

Возвращаюсь в папку, откуда копировала файлы и удаляю оттуда то, что должно храниться в другом месте (рис. 5.7).

```
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/lab04$ cd ..
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc$ cd labs/lab04
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ rm hello hello.o lab4 lab4.o list
.lst main obj.o
diserebryakova@fedora:~/work/study/study_2024-2025_arh-pc/labs/lab04$ ls
hello.asm  lab4.asm  presentation  report
```

Рис. 5.7: Удаление файлов

Завершаю заполнение Л04_Серебрякова_отчет.md и отправляю все необходимые данные на GitHub. С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, соответствующе комментируя действие

6 Выводы

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

Список литературы

1. https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%