

Отчёт по лабораторной работе №4

Простейший вариант

Гомазкова Алина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	1. Создание программы Hello world!	10
4.2	2. Работа с транслятором NASM	11
4.3	3. Работа с расширенным синтаксисом командной строки NASM .	11
4.4	4. Работа с компоновщиком LD	12
4.5	5. Запуск исполняемого файла	12
4.6	6. Выполнение заданий для самостоятельной работы	13
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Рис. 1 Перемещение между директориями	10
4.2	Рис. 2 Создание пустого файла	10
4.3	Рис. 3 Открытие файла в текстовом редакторе	10
4.4	Рис. 4 Заполнение файла	11
4.5	Рис. 5 Компиляция текста программы	11
4.6	Рис. 6 Компиляция текста программы	12
4.7	Рис. 7 Передача объектного файла на обработку компоновщику .	12
4.8	Рис. 8 Передача объектного файла на обработку компоновщику .	12
4.9	Рис. 9	12
4.10	Рис. 10 Запуск исполняемого файла	13
4.11	Рис. 11 Создание копии файла	13
4.12	Рис. 12 Изменение программы	13
4.13	Рис. 13 Компиляция текста программы	14
4.14	Рис. 14 Передача объектного файла на обработку компоновщику .	14
4.15	Рис. 15 Запуск исполняемого файла	14
4.16	Рис. 16 Создании копии файлов в новом каталоге	14
4.17	Рис. 17 Удаление лишних файлов в текущем каталоге	15
4.18	Рис. 18 Добавление файлов на GitHub	15
4.19	Рис. 19 Отправка файлов	15

Список таблиц

1 Цель работы

Освоение 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 1. Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 1)

```
[alinagomazkova@fedora ~]$ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
[alinagomazkova@fedora lab04]$
```

Рис. 4.1: Рис. 1 Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. 2)

```
[alinagomazkova@fedora lab04]$ touch hello.asm
[alinagomazkova@fedora lab04]$ mousepad hello.asm
```

Рис. 4.2: Рис. 2 Создание пустого файла

Открываю созданный файл в текстовом редакторе `mousepad` (рис. 3)

```
[alinagomazkova@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: Рис. 3 Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4)

```
~/work/study/2023-2024/Архитектура компьютера/arch-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. 4.4: Рис. 4 Заполнение файла

4.2 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o” (рис. 5)

```
[alinagomazkova@fedora lab04]$ nasm -f elf64 hello.asm
[alinagomazkova@fedora lab04]$ ls
hello.asm  hello.o  presentation  report
```

Рис. 4.5: Рис. 5 Компиляция текста программы

4.3 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`. Далее проверяю с помощью утилиты `ls` правильность выполнения команды (рис. 6)

```
[alinagomazkova@fedora lab04]$ nasm -o obj.o -f elf64 -g -l list.lst hello.asm
[alinagomazkova@fedora lab04]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
```

Рис. 4.6: Рис. 6 Компиляция текста программы

4.4 4. Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды (рис. 7)

```
[alinagomazkova@fedora lab04]$ ld -m elf_x86_64 hello.o -o hello
```

Рис. 4.7: Рис. 7 Передача объектного файла на обработку компоновщику

Выполняю следующую команду. 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 (рис. 8) (рис. 9).

```
[alinagomazkova@fedora lab04]$ ld -m elf_x86_64 obj.o -o main
```

Рис. 4.8: Рис. 8 Передача объектного файла на обработку компоновщику

```
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
```

Рис. 4.9: Рис. 9

4.5 5. Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 10)

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

Рис. 4.10: Рис. 10 Запуск исполняемого файла

4.6 6. Выполнение заданий для самостоятельной работы

С помощью утилиты `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm` (рис. 11)

```
netto netto.asm netto.o test.tst main obj.o pre
[alinagomazkova@fedora lab04]$ cp hello.asm lab4.asm
[alinagomazkova@fedora lab04]$
```

Рис. 4.11: Рис. 11 Создание копии файла

С помощью текстового редактора `mousepad` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила мои имя и фамилию (рис. 12)

```
; lab4.asm
SECTION .data                ; Начало секции данных
lab4: DB 'Alina Gomazkova',10 ; 'Alina Gomazkova' плюс
                                ; символ перевода строки
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 вывод
mov ecx,lab4                 ; Адрес строки lab4 в ecx
mov edx,lab4Len              ; Размер строки lab
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. 4.12: Рис. 12 Изменение программы

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты `ls`, что файл `lab4.o` создан (рис. 13)

```
[alinagomazkova@fedora lab04]$ nasm -f elf64 lab4.asm
[alinagomazkova@fedora lab04]$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
```

Рис. 4.13: Рис. 13 Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 14)

```
[alinagomazkova@fedora lab04]$ ld -m elf_x86_64 lab4.o -o lab4
[alinagomazkova@fedora lab04]$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
```

Рис. 4.14: Рис. 14 Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 15)

```
[alinagomazkova@fedora lab04]$ ./lab4
Alina Gomazkova
```

Рис. 4.15: Рис. 15 Запуск исполняемого файла

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. Проверяю с помощью утилиты ls правильность выполнения команды (рис. 16)

```
[alinagomazkova@fedora lab04]$ cp * ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
cp: 'hello' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/hello' - один и тот же файл
cp: 'hello.asm' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/hello.asm' - один и тот же файл
cp: 'hello.o' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/hello.o' - один и тот же файл
cp: 'lab4' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/lab4' - один и тот же файл
cp: 'lab4.asm' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/lab4.asm' - один и тот же файл
cp: 'lab4.o' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/lab4.o' - один и тот же файл
cp: 'list.lst' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/list.lst' - один и тот же файл
cp: 'main' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/main' - один и тот же файл
cp: 'obj.o' и '/home/alinagomazkova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/obj.o' - один и тот же файл
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
[alinagomazkova@fedora lab04]$ ls ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
```

Рис. 4.16: Рис. 16 Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты `rm`, ведь копии файлов остались в другой директории (рис. 17)

```
[alinagomazkova@fedora lab04]$ rm hello hello.o lab4 lab4.o main obj.o
[alinagomazkova@fedora lab04]$ ls
hello.asm lab4.asm list.lst presentation report
[alinagomazkova@fedora lab04]$ rm list.lst
[alinagomazkova@fedora lab04]$ ls
hello.asm lab4.asm presentation report
```

Рис. 4.17: Рис. 17 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 №4 (рис. 18)

```
[alinagomazkova@fedora lab04]$ git add .
[alinagomazkova@fedora lab04]$ git commit -m "Add files for lab04"
[master b8bab53] Add files for lab04
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
```

Рис. 4.18: Рис. 18 Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git push` (рис. 19)

```
[alinagomazkova@fedora lab04]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.04 КиБ | 1.04 МБ/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:alinagomazkova/study_2023-2024_arh-pc.git
9da9d9c..b8bab53 master -> master
```

Рис. 4.19: Рис. 19 Отправка файлов

5 Выводы

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

Список литературы

1.Архитектура ЭВМ