

Отчет по лабораторной работе №4

Дисциплина: архитектура компьютера

Рахматова Жылдыз Талатнбековна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
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 . . .	10
4.4	Работа с компоновщиком LD	11
4.5	Запуск исполняемого файла	11
4.6	Выполнение заданий для самостоятельной работы.	11
5	Выводы	14

Список иллюстраций

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,
устройство управления (УУ) – обеспечивает управление и контроль всех устройств компьютера
регистры – сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора
состояние операндов. Практически все команды представляют собой преобразование данных хранящихся
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 информации;
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.

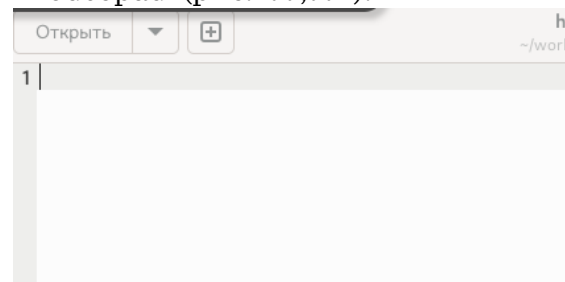
```
ztrakhmatova@dk8n60 ~ $ cd ~/work/arch-pc/lab04
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $
```

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью ути-

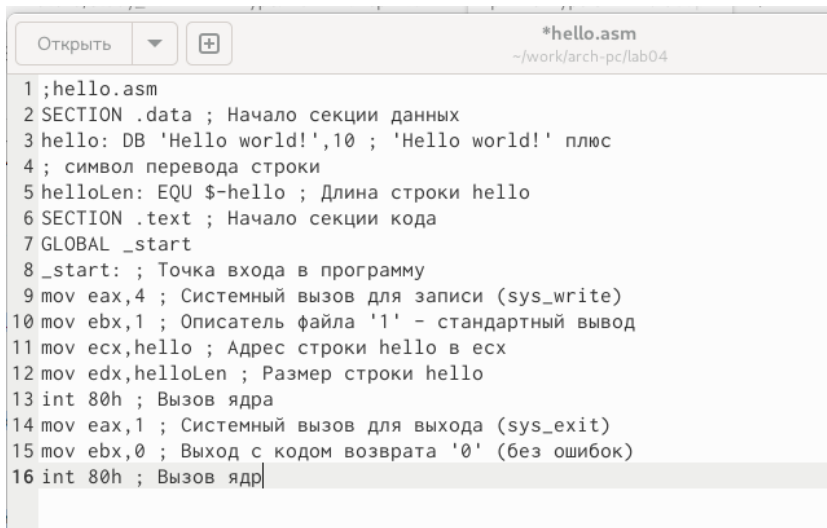
```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ touch hello.asm
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $
```

Открываю созданный файл в текстовом редакторе `mousetpad` (рис. ??,??).

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ gedit hello.asm
```



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



```
*hello.asm
~/work/arch-pc/lab04

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
```

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

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ █
```

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. ??). Далее проверяю с помощью утилиты

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.a
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o list.lst obj.o
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ █
```

`ls` правильность выполнения команды.

4.4 Работа с компоновщиком LD

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

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $
```

Выполняю следующую команду (рис. ??). 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

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $
```

4.5 Запуск исполняемого файла

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

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ./hello
Hello world!
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $
```

4.6 Выполнение заданий для самостоятельной работы.

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

```
ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
```

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

```

Открыть hello.asm
~/work/arch-pc/lab04
1 ;hello.asm
2 SECTION .data ; Начало секции данных
3 lab4: DB 'Rahmatova Zyldyz',10
4 lab4Len: EQU $-lab4 ; Длина строки lab4
5 SECTION .text ; Начало секции кода
6 GLOBAL _start
7 _start: ; Точка входа в программу
8 mov eax,4 ; Системный вызов для записи (sys_write)
9 mov ebx,1 ; Описатель файла '1' - стандартный вывод
10 mov ecx,hello ; Адрес строки lab4 в ecx
11 mov edx,lab4Len ; Размер строки lab
12 int 80h ; Вызов ядра
13 mov eax,1 ; Системный вызов для выхода (sys_exit)
14 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
15 int 80h ; Вызов ядра

```

Компилирую текст программы в объектный файл (рис. ??). Проверяю с помо-

щью утилиты ls, что файл lab4.o создан. `ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls`
`hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o`
`ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $`

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы полу-

чить исполняемый файл lab5 (рис. ??). `ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4`
`ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $ ls`
`hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o`
`ztrakhmatova@dk8n60 ~/work/arch-pc/lab04 $`

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия

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. ??). Проверяю с помощью утилиты ls правильность выполнения команды.

```

cp: цель 'lab4' не является файлом или каталогом
ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/study_2024-2025_arch-
pc/labs/lab04 $ cp * ~/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'

```

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь ко-

пии файлов остались в другой директории (рис. ??). `ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/`
`pc/labs/lab04 $ rm hello hello.o lab4 lab4.o list.lst main obj.o`
`ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/`
`pc/labs/lab04 $ ls`
`hello.asm lab4.asm presentation report`

С помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 (рис. ??).

```
ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/study_2024-2025_arch-  
pc/labs/lab04 $ git add .  
ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/study_2024-2025_arch-  
pc/labs/lab04 $ git commit -m "Add fales for lab04"  
[master 6a1093b] Add fales for lab04  
2 files changed, 34 insertions(+)  
create mode 100644 labs/lab04/hello.asm  
create mode 100644 labs/lab04/lab4.asm  
ztrakhmatova@dk8n60 ~/work/study/2024-2025/архитектура компьютера/study_2024-2025_arch-  
pc/labs/lab04 $
```

Отправляю файлы на сервер с помощью команды git push

5 Выводы

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