

Отчёта по лабораторной работе №4

Дисциплина: архитектура компьютера

Видмаер Егор

Содержание

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

Цель работы

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

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.

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 объема, 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 объемов 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.

Выполнение лабораторной работы

Создание программы Hello world!

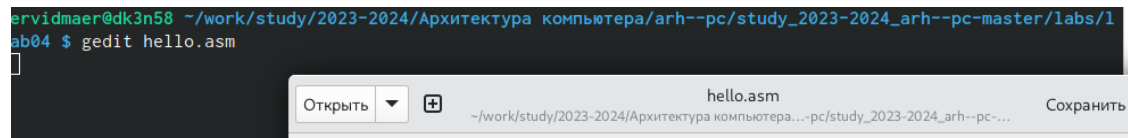
С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать. Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. 1).

```
ervidmaer@dk3n58 ~ $ cd ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/lab04
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/lab04 $ touch hello.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/lab04 $ ls
hello.asm presentation report
```

Создание пустого файла

Открываю созданный файл в текстовом редакторе `gedit` (рис. 2).

```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/lab04 $ gedit hello.asm
```



Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello world!”. Так, как ассемблер не является высокоуровневым языком, каждая команда размещается на отдельной строке, так же обращаю внимание на регистр, так как Assembly чувствителен к нему. (рис. 3).



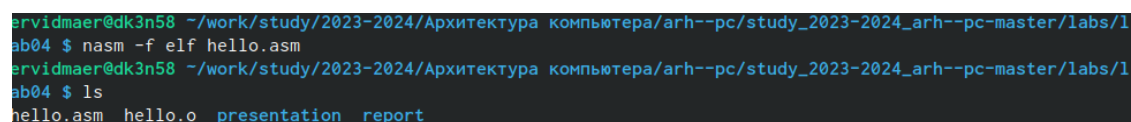
The screenshot shows a text editor window titled 'hello.asm'. The menu bar includes 'Открыть' (Open), a '+' icon, and 'Сохранить' (Save). The address bar shows the file path: '~/.work/study/2023-2024/Архитектура компьютера...-pc/study_2023-2024_arh--pc-...'. The code is as follows:

```
1 ;hello.asm
2 SECTION .data
3     hello:      DB 'Hello world!' , 10
4     helloLen:   EUQ $-hello
5
6 SECTION .text
7     GLOBAL _start
8
9 _start:
10     mov eax,4
11     mov ebx,1
12     mov ecx,hello
13     mov edx,helloLen
14     int 80h
15
16     mov eax,1
17     mov ebx,0
18     int 80h
19
```

Заполнение файла

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

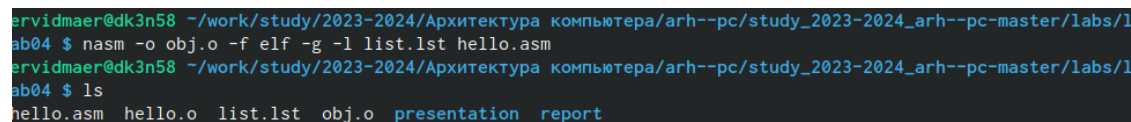


```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ nasm -f elf hello.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ls
hello.asm hello.o presentation report
```

Компиляция текста программы

Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, используя ключ `-o` который задает имя объектному файлу, так же в файл будут включены символы для отладки (ключ `-g`), с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. 5). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.



```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ls
hello.asm hello.o list.lst obj.o presentation report
```

Компиляция текста программы

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. Выполняю следующую команду (рис. 6). 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

```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ld -m elf_i386 hello.o -o hello
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ld -m elf_i386 obj.o -o main
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Передача объектного файла на обработку компоновщику

Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 7).

```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ./hello
Hello world!
```

Запуск исполняемого файла

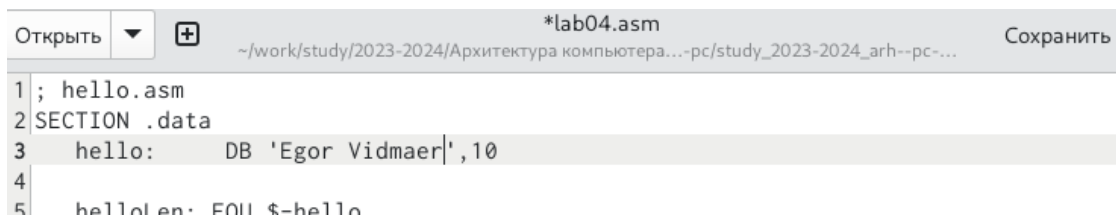
Выполнение заданий для самостоятельной работы.

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 9).

```
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ cp hello.asm lab04.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ls
hello.asm lab04.asm presentation report
```

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. 10).



```
Открыть + *lab04.asm Сохранить
~/work/study/2023-2024/Архитектура компьютера...-pc/study_2023-2024_arh--pc-...

1; hello.asm
2SECTION .data
3hello: DB 'Egor Vidmaer',10
4
5hello len EQU $-hello
```

Изменение программы

Компилирую текст программы в объектный файл. Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 11).

```

ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ nasm -f elf lab04.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ nasm -o obj.o -f elf -g -l list.lst lab04.asm
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ld -m elf_i386 lab04.o -o lab04
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ld -m elf_i386 obj.o -o main
ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ls
hello.asm lab04 lab04.asm lab04.o list.lst main obj.o presentation report

```

Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 12).

```

ervidmaer@dk3n58 ~/work/study/2023-2024/Архитектура компьютера/arh--pc/study_2023-2024_arh--pc-master/labs/1
ab04 $ ./lab04
Egor Vidmaer

```

Запуск исполняемого файла

Выводы

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