

Отчёта по лабораторной работе №4

Дисциплина: архитектура компьютера

Шабакowa Карина Баировна

Содержание

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.

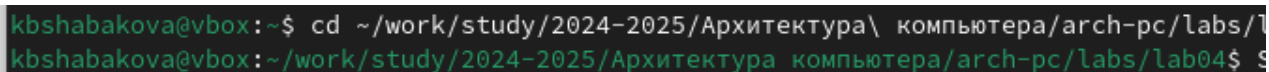
Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении 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. 1).



```
kbshabakova@vbox:~$ cd ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/labs/1
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch`. Открываю созданный файл в текстовом редакторе `mousepad` (рис. 2).

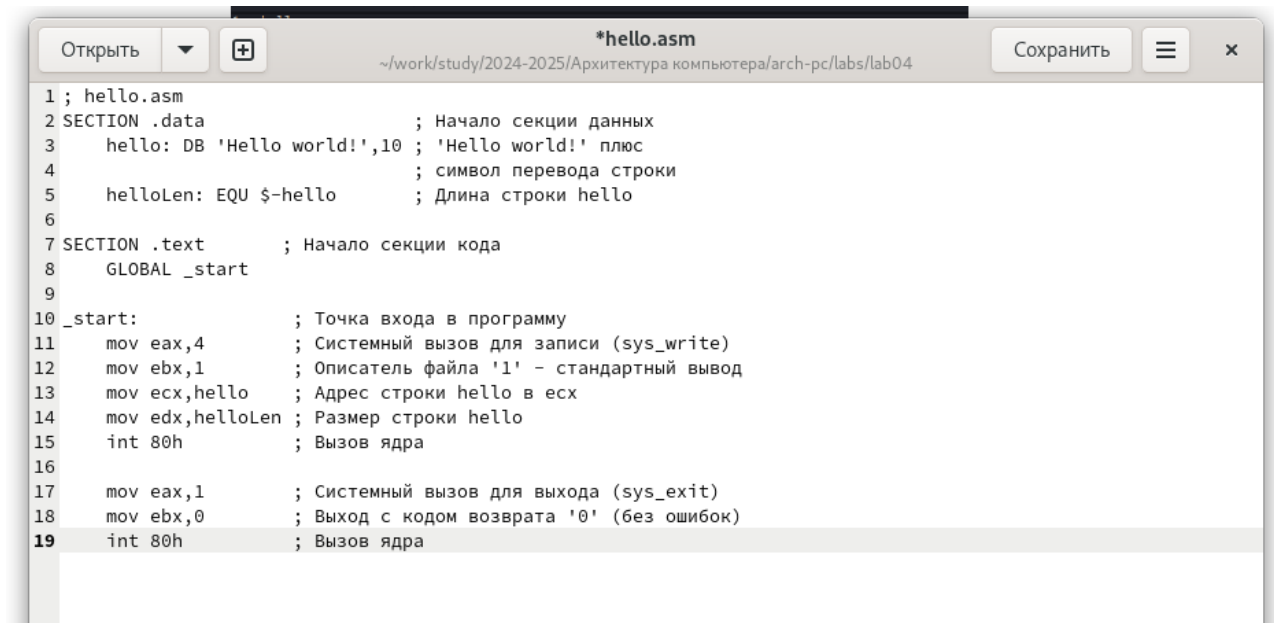
```

va@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ touch hello.asm
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ gedit hello.asm

```

Рис. 2: Создание пустого файла, открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 3).



```

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
7 SECTION .text          ; Начало секции кода
8     GLOBAL _start
9
10 _start:                ; Точка входа в программу
11     mov eax,4           ; Системный вызов для записи (sys_write)
12     mov ebx,1           ; Описатель файла '1' - стандартный вывод
13     mov ecx,hello       ; Адрес строки hello в ecx
14     mov edx,helloLen    ; Размер строки hello
15     int 80h            ; Вызов ядра
16
17     mov eax,1           ; Системный вызов для выхода (sys_exit)
18     mov ebx,0           ; Выход с кодом возврата '0' (без ошибок)
19     int 80h            ; Вызов ядра

```

Рис. 3: 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). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”.

```

kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf hello.asm
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm hello.o presentation report

```

Рис. 4: 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. 5). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

```

kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm hello.o list.lst obj.o presentation report

```

Рис. 5: Компиляция текста программы

4.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `hello` (рис. 6). Ключ `-o` задает имя создаваемого исполняемого файла.

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
```

Рис. 6: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. 7). 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`

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Рис. 7: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл `hello` (рис. 8).

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ./hello
Hello world!
```

Рис. 8: Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы.

С помощью утилиты `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab04.asm` (рис. 9).

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ cp hello.asm lab04.asm
```

Рис. 9: Создание копии файла

С помощью текстового редактора `mousepad` открываю файл `lab04.asm` и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. 10).



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

; lab04.asm
SECTION .data                                ; Начало секции данных
    lab04: DB 'Karina Shabakova',10          ; символ перевода строки
    lab04len: EQU $-lab04                    ; Длина строки lab04
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,lab04                           ; Адрес строки lab04 в ecx
    mov edx,lab04len                         ; Размер строки lab04
    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. 10: Изменение программы

Компилирую текст программы в объектный файл (рис. 11). Проверяю с помощью утилиты `ls`, что файл `lab04.o` создан.

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf lab04.asm
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o presentation report
```

Рис. 11: Компиляция текста программы

Передаю объектный файл lab04.o на обработку компоновщику LD, чтобы получить исполняемый файл lab04 (рис. 12).

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 lab04.o -o lab04
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o presentation report
```

Рис. 12: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab5, на экран действительно выводятся мои имя и фамилия (рис. 13).

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ./lab04
Karina Shabakova
```

Рис. 13: Запуск исполняемого файла

К сожалению, я начала работу не в том каталоге, поэтому создаю другую директорию lab04 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. 14). Проверяю с помощью утилиты ls правильность выполнения команды. Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории

```
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ mkdir -p ~/work/arch-pc/lab04
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ cd ~/work/arch-pc/lab04
kbshabakova@vbox:~/work/arch-pc/lab04$ cp * ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
kbshabakova@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o
kbshabakova@vbox:~/work/arch-pc/lab04$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o presentation report
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ rm hello hello.o lab04 lab04.o list.lst main obj.o
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab04.asm presentation report
kbshabakova@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add
```

Рис. 14: Создании копии файлов в новом каталоге и 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. 15). Отправляю файлы на сервер с помощью команды git push

```

kbshabakova@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add .
kbshabakova@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add fales for lab04"
[master 6aff058] Add fales for lab04
 2 files changed, 34 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab04.asm
kbshabakova@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.05 КиБ | 216.00 КиБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:dazz4q/study_2024-2025_arh-pc.git
 857623e..6aff058  master -> master

```

Рис. 15: Добавление файлов на GitHub и отправка файлов

5 Выводы

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

6 Список литературы

1.

https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20E2%84%964.%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.pdf