РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Архитектура компьютера

Студент: Валиева Карина Ринатовна

Группа: НБИбд-01-25

МОСКВА

2025 г.

Оглавление

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) — машинноориентированный 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 лабороторной работы

4.1 Создание программы Hello World!

Создаю каталог в котором буду работать и с помощью утилиты cd перемещаюсь в этот каталог. (Рис.1)

```
krvalieva@vbox:~$ mkdir -p ~/work/arch-pc/lab04
krvalieva@vbox:~$ cd ~/work/arch-pc/lab04
```

Рис. 1. Перемещаюсь между директориями

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

```
krvalieva@vbox:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 2.Создание пустого файла

Открываю созданный файл в текстовом редакторе gedit. (Рис.3)

```
krvalieva@vbox:~/work/arch-pc/lab04$ gedit hello.asm
```

Рис. 3. Открытие файла в текстовом редакторе

Заполняю файл и вставляю в него программу для ввода «Hello word!». (Рис.4)

Open | *hello.asm ~/work/arch-pc/lab04 | Save

```
; 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. 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. Далее проверяю правильность выполнения команды с помощью утилиты ls. (Рис.5)

```
krvalieva@vbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 5. 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. Далее проверяю с помощью утилиты ls правильность выполнения команды. (Рис.6)

```
krvalieva@vbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 6. Компиляция текста программы

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)

```
krvalieva@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

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)

```
krvalieva@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 8. Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello. (Рис.9)

```
krvalieva@vbox:~/work/arch-pc/lab04$ ./hello
Hello world!
```

Рис. 9. Запуск исполняемого файла

4.6 Выполнение заданий для самостоятельной работы.

1. С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm. (Рис.10)

```
krvalieva@vbox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 10. Создание копии файла

2. С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (Рис.11)

Open | lab4.asm ~/work/arch-pc/lab04 | Save

```
; lab4.asm
SECTION .data                       ; Начало секции данных
        lab4: DB 'Karina Valieva',10

        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 lab4
        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. 11. Изменение программы

3. Компилирую текст программы в объектный файл. Проверяю с помощью утилиты ls, что файл lab4.o создан. (Рис.12)

```
krvalieva@vbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
krvalieva@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
```

Рис. 12. Компиляция текста программы

Передаю объектный файл lab5.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4. (Рис.13)

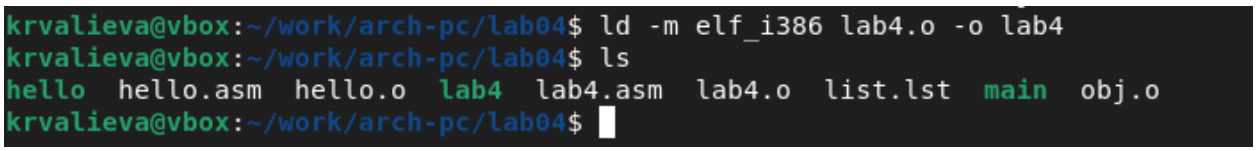

Рис. 13. Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия. (Рис.14)

Рис. 14. Запуск исполняемого файла

4. Копирую файлы hello.asm и lab4.asm в личный репозиторий. (Рис.15)

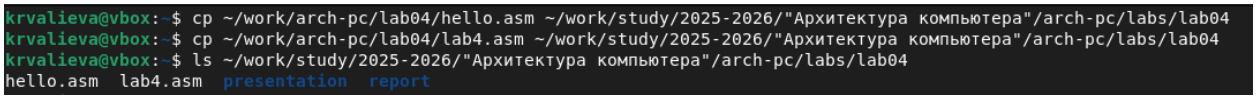

Рис. 15. Копирование файла

В каталог ~/work/study/2025-2026/“Архитектура компьютера”/archpc/labs/lab04/report загружаю файл отчета в формате .docx и .pdf. С помощью команды git add . и git commit “…” добавляю файлы на GitHub, комментируя изменения как добавление файлов. Затем 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.