Отчет по лабараторной работе №4

Дисциплина:архитектура компьютера

Никуленков Степан Сергеевич

Содержание

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, 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 каче- стве 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:

формирование адреса в памяти очередной команды; считывание кода коман-

ды из памяти и её дешифрация; выполнение команды; 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 лабараторной работы

4.1 Создание программы Hello world!

Создаю новый каталог с помощью команды mkdir.

```
root@vbox:/home/ssnikulenkov/work# mkdir -p ~/work/arch-pc/lab04
```

Рис. 4.1: Создание рабочего каталога

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch

```
root@vbox:~# cd /home/ssnikulenkov/work/arch-pc/lab04/
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# touch hello.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04#
```

Рис. 4.2: Создание файла hello.asm

Открываю файл в текстовом редакторе mousepad.

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

```
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ld -m elf_i386 hello.o -o hello
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ls
ello  hello.asm  hello.o  list.lst  obj.o
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ld -m elf_i386 obj.o -o main
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ls
ello  hello.asm  hello.o  list.lst  main  obj.o
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04#
```

Рис. 4.3: Проверка правильности выполненных команд

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. 4.4). Далее проверяю с помощью утилиты ls правильность выполнения команды.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# nasm -o obj.o -f elf -g -l list
.lst hello.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.4: Проверка

4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 4.5). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.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

```
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ld -m elf_i386 hello.o -o hello
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ls
ello  hello.asm  hello.o  list.lst  obj.o
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ld -m elf_i386 obj.o -o main
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ls
ello  hello.asm  hello.o  list.lst  main  obj.o
oot@vbox:/home/ssnikulenkov/work/arch-pc/lab04#
```

Рис. 4.5: Проверка правильности выполненныйх команд

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ./hello
Hello world!
```

Рис. 4.6: Запуск файла

5 Выполнение заданий для самостоятельной работы.

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

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# cp hello.asm lab4.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# pwd
```

Рис. 5.1: Создание копии lab4.asm

Изменяю файл.

```
; hello.asm
SECTION .data                          ; Начало секции данных
    hello:      DB 'Степан Никуленков',10 ; 'Степан Никуленков' 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. 5.2: Редактирование файла

Компилирую текст программы в объектный файл.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# nasm -f elf lab4.asm
```

Рис. 5.3: Создание объектного файла

Передаю объектный файл на обработку LD,чтобы получить исполняемый фыйл lab4.Открываю созданный файл.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ld -m elf_i386 lab4.o -o lab4
root@vbox:/home/ssnikulenkov/work/arch-pc/lab04# ./lab4
Степан Никуленков
```

Рис. 5.4: Проверка

Копирую файлы в нужную директорию.

```
root@vbox:/home/ssnikulenkov# cp /home/ssnikulenkov/work/arch-pc/lab04/lab4.asm
/home/ssnikulenkov/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/la
b04
```

Рис. 5.5: Копирую lab4.asm

```
root@vbox:~# cp /home/ssnikulenkov/work/arch-pc/lab04/hello.asm /home/ssnikulenk
ov/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04
```

Рис. 5.6: Копирую hello.asm

Отправляю файлы на GitHub

6 Выводы

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

7 Список литературы

Архитектура ЭВМ