Отчет по лабораторной работе №4

Дисциплина: Архитектура компьютера

Карпова Анастасия Александровна

Содержание

Список иллюстраций

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Созание 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 электронно-вычислительной машины (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 (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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, преоб-

разование (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. 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. В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы. Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня,таких как C/C++, Perl, Python и пр.Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды

4 Выполнение лабораторной работы

Программа Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM и перехожу в созданный каталог (рис. 4.1).

```
akarpova@Justclown:~$ mkdir -p ~/work/arch-pc/lab04
akarpova@Justclown:~$ cd  ~/work/arch-pc/lab04
```

Рис. 4.1: Создание каталога и перемещение

Создаю текстовый файл с именем hello.asm и открываю его в gedit (рис. 4.2)

```
akarpova@Justclown:~/work/arch-pc/lab04$ touch hello.asm
akarpova@Justclown:~/work/arch-pc/lab04$ gedit hello.asm
```

Рис. 4.2: Создание текстового файла

Ввожу следующий текст (рис. 4.3)

```
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 _start:               ; Точка входа в программу
10    mov eax,4 ; Системный вызов для записи (sys_write)
11    mov ebx,1 ; Описатель файла '1' - стандартный вывод
12    mov ecx,hello ; Адрес строки hello в ecx
13    mov edx,helloLen ; Размер строки hello
14    int 80h ; Вызов ядра
15
16    mov eax,1 ; Системный вызов для выхода (sys_exit)
17    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18    int 80h ; Вызов ядра
19
20
```

Рис. 4.3: Ввод текста

Транслятор NASM

NASM превращает текст программы в объектный код. Для компиляции приведённого выше текста программы «Hello World» необходимо написать следующее. (рис. 4.4)

```
akarpova@Justclown:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

Рис. 4.4: Компиляция

Расширенный синтаксис командной строки NASM

Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o (рис. 4.5)

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

Рис. 4.5: Компиляция hello.asm в файл obj.o

Компоновщик LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить файд hello (рис. 4.6)

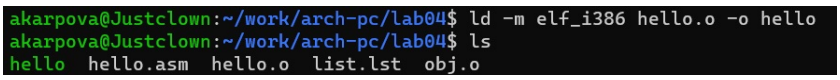

Рис. 4.6: Передача файла

Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Поэтому ввожу следующую команду (рис. 4.7)

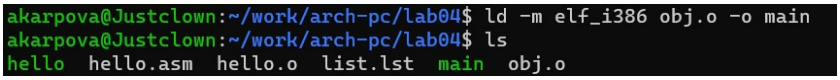

Рис. 4.7: Передача файла

Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл, находящийся в текущем каталоге набрав в командной строке следующее (рис. 4.8)

Рис. 4.8: Запуск исполняемого файла

Выполнение заданий для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm и открываю lab4.asm при помощи gedit, редактирую (рис. 4.9) (рис. 4.10)

Рис. 4.9: Копирование

```
; lab4.asm
SECTION .data ; Начало секции данных
    lab4: DB 'Karpova Anastasia!',10 ;
                                ; символ перевода строки
    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. 4.10: Запуск и редактирование

Компилирую текст программы в объектный файл (рис. 4.11)

```
akarpova@Justclown:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
```

Рис. 4.11: Компиляция

Передаю объектный файл lab4.o на обработку компоновщику LD (рис. 4.12)

```
akarpova@Justclown:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
```

Рис. 4.12: Передача файла

Запускаю исполняемый файл. (рис. 4.13)

```
akarpova@Justclown:~/work/arch-pc/lab04$ ./lab4
Karpova Anastasia!
```

Рис. 4.13: Запуск исполняемого файла

Отправляю все файлы на git (рис. 4.14)

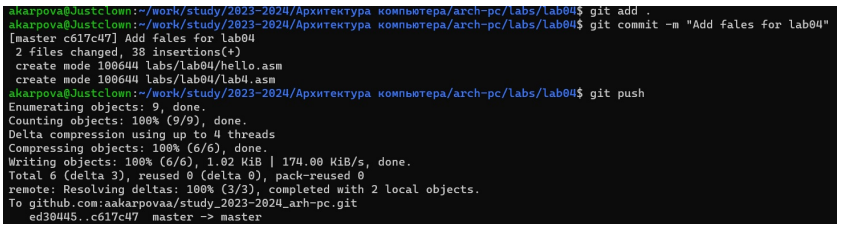

Рис. 4.14: Отправка файлов на github

5 Выводы

В ходе лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

1. Архитектура ЭВМ ::: {#refs} :::