

Отчет по лабораторной работе №4

Дисциплина: Архитектура компьютера

Баазова Нина Эдгаровна

Содержание

1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

2 Задание

1. Выполнение порядка лабораторной работы №4
2. Задание для самостоятельной работы

3 Теоретическое введение

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.

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. Типичный формат записи команд NASM имеет вид: [метка:] мнемокод [операнд {, операнд}] [; комментарий]

4 Выполнение лабораторной работы

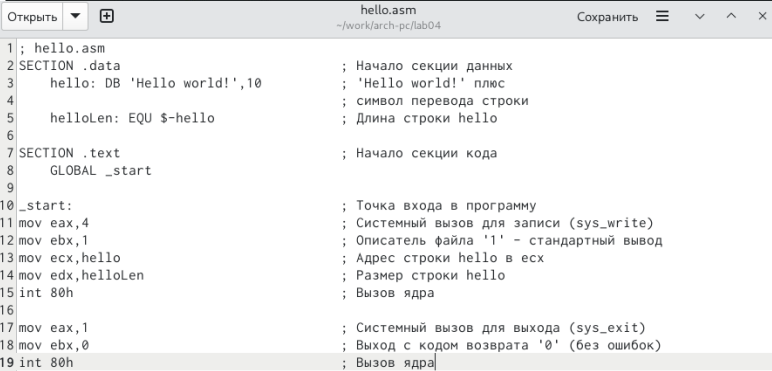
1. Порядок выполнения лабораторной работы:

1.1 Программа “Hello world!”

Создадим каталог для работы с программами на языке ассемблер NASM, используя команду `mkdir -p ~/work/arch-pc/lab04`. Затем переходим в него и создаём

текстовый файл с именем hello.asm, после открываем редактор gedit (рис 1) и вводим в него следующее (рис 2):

```
nebaazova@dk2n24 ~$ mkdir -p ~/work/arch-pc/lab04
nebaazova@dk2n24 ~$ cd ~/work/arch-pc/lab04
nebaazova@dk2n24 ~/work/arch-pc/lab04$ touch hello.asm
nebaazova@dk2n24 ~/work/arch-pc/lab04$ gedit hello.asm
```



```
1; hello.asm
2SECTION .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
7SECTION .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 ехх
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
```

1.2 Транслятор NASM.

Он представляет текст программы в объектный код. Для нашего текста программы введём команду `nasm -f elf hello.asm`. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат `elf64` позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто `elf`.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04$ nasm -f elf hello.asm
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ls
hello.asm  hello.o  presentation  report
```

1.3 Разширенный синтаксис командой строки NASM.

Выполним команду `nasm -o obj.o elf -g -l list.lst hello.asm`, скомпилируем исходный файл в `obj.o` и создаем файл листинга `list.lst`.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
```

рис 5

1.4 Компоновщик LD.

Чтобы получить исполняемую программу, нужно передать на обработку объектный файл компоновщику, используя команду (`ld -m elf_i386 hello.o -o hello`) и проверяя `ls`. Затем вводим команду (`ld -m elf_i386 obj.o -o main`), где объектный файл будет `obj.o`, а исполняемый `main`, и тоже проверяем.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
nebaazova@dk2n24 ~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
```

рис 6

1.5 Запуск исполняемого файла.

Чтобы запустить исполняемый файл, который находится в текущем каталоге, следует набрать в командой строке: `./hello`.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ./hello
Hello world!
nebaazova@dk2n24 ~/work/arch-pc/lab04 $
```

рис 7

ВЫВОД: Мы написали программу “Hello world!” на языке ассемблера NASM.

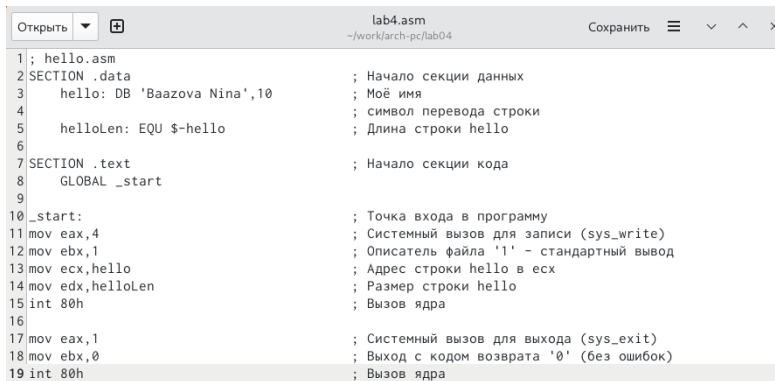
2. Задание для самостоятельной работы:

2.1 В текущем каталоге создаем копию файла `hello.asm` с именем `lab4.asm` с помощью команды `cp`, проверяя эфетивность команды командой `ls`. Затем открываем текстовой редактор `gedit`.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o presentation report
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ gedit lab4.asm
```

рис 8

2.2 Вносим туда свою фамилию и имя вместо “Hello world!”.



```
1; hello.asm
2SECTION .data
3    hello: DB 'Baazova Nina',10
4           ; Начало секции данных
5           ; Моё имя
6           ; символ перевода строки
7           ; Длина строки hello
8SECTION .text
9    GLOBAL _start
10   _start:
11       ; Точка входа в программу
12       mov eax,4
13       ; Системный вызов для записи (sys_write)
14       mov ebx,1
15       ; Описатель файла '1' - стандартный вывод
16       mov ecx,hello
17       ; Адрес строки hello в ecx
18       mov edx,helloLen
19       ; Размер строки hello
20       int 80h
21       ; Вызов ядра
22
23       mov eax,1
24       ; Системный вызов для выхода (sys_exit)
25       mov ebx,0
26       ; Выход с кодом возврата '0' (без ошибок)
27       int 80h
28       ; Вызов ядра
```

рис 9

2.3 Оттранслируем полученный текст программы `lab4.asm` в объектный файл, затем выполним его компоновку и запустим полученный исполняемый файл.

```
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ./lab4
Baazova Nina
nebaazova@dk2n24 ~/work/arch-pc/lab04 $
```

рис 10

2.4 Теперь скопируем файлы `hello.asm` и `lab4.asm` в наш локальный репозиторий в каталог `~/work/study/2023-2024/“Computer architecture”/arch-pc/labs/lab04/` и проверим `ls`. Затем удалим лишнии файльв в каталоге `~/work/arch-pc/lab04/`

```

nebaazova@dk2n24 ~/work/arch-pc/lab04 $ cp * ~/work/study/2023-2024/"Computer architecture"/arch-pc/labs/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ rm hello hello.o lab4 lab4.o list.lst obj.o
nebaazova@dk2n24 ~/work/arch-pc/lab04 $ ls
hello.asm lab4.asm main presentation report
nebaazova@dk2n24 ~/work/arch-pc/lab04 $

```

рис 11

Загрузим файл на github.

```

nebaazova@dk2n24 ~/work/study/2023-2024/Computer architecture/arch-pc $ git add .
nebaazova@dk2n24 ~/work/study/2023-2024/Computer architecture/arch-pc $ git commit -m "Add files for lab04"
[master 577d39c] Add files for lab04
23 files changed, 60 insertions(+)
nebaazova@dk2n24 ~/work/study/2023-2024/Computer architecture/arch-pc $ git push
Перечисление объектов: 39, готово.
Подсчет объектов: 100% (39/39), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (31/31), готово.
Запись объектов: 100% (31/31), 1.33 МБ | 2.20 МБ/с, готово.
Всего 31 (изменений 8), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (8/8), completed with 4 local objects.
To github.com:291103Nina/study_2023-2024_arh-pc.git
   b095701..577d39c  master -> master
nebaazova@dk2n24 ~/work/study/2023-2024/Computer architecture/arch-pc $

```

ВЫВОД: Мы выполнили задание для самостоятельной работы.

5 Вывод лабораторной работы

Мы освоили процедуры компиляции и сборки программ, написанных на ассемблере NASM.