Отчёт по лабораторной работе №4

Язык ассемблера NASM

Мадалиев А.А

Содержание

Цель работы

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

Задание

1. ПрограммаHelloworld! 1.1 создать каталог для работы с программами на языке ассемблера NASM 1.2 перейти в созданный каталог 1.3 создать текстовый файл с именем hello.asm 1.4 открыть этот файл 1.5 ввести в него указанный текст
2. ТрансляторNasm 2.1 выполнить комппиляцию в объектный код
3. Расширенныйсинтаксис 3.1 выполнить компиляцию исходного файла
4. КомпановщикLD 4.1 передать объектный файл на обработку компановщику
5. Запуститьисполняемыйфайл
6. Заданиядлясамостоятельнойработы 6.1 создать копию файла hello.asm с именем lab4.asm 6.2 изменить скопированный файл, чтобы выводилась строка с именем и фамилией 6.3 оттранслировать полученный текст программы lab4.asm в объектный файл 6.4 скопировать файлы hello.asm и lab4.asm в локальный репозиторий

Теоретическое введение

В процессе создания ассемблерной программы можно выделить четыре шага: • Набор текста программы в текстовом редакторе и сохранение её в

отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm. • Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объ- ектного файла — o, файла листинга — lst. • Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняе- мый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map. • Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе програм- мы, поэтому может присутствовать этап отладки программы при помощи специ- альной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

Выполнение лабораторной работы

1. Переходим в каталог lab04 и создаем текстовый файл hello.asm

```
aamadaliev@dk3n55 ~ $ cd ~/work/arch-pc/lab04
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ touch hello.asm
```

Создание файла hello.asm (рис 1) 2. Открываем этот файл в gedit и вводим текст.

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Открытие файла (рис 2) 3. Компилируем написанный текст с помощью следующей команды.

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
```

Компиляция текста (рис 3) 4. Компилируем файл hello.asm в obj.o и проверяем с помощью команды ls

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hel
lo.asm
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
```

Компиляция файла (рис 4) 5. Передаем объектный файл на обработку компоновщику для получения исполняемой программы.

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
```

Обработка файла (рис 5) 6. Вносим изменения в тексте программы в файле lab4.asm

```
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'madaliev akmaljon',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
```

Обаботка файла (рис 6) 7. Транслируем полученный текст программы lab4.asm в объектный файл. Выполняем компоновку данного файла и запускаем получившийся файл.

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ nasm -o madaliev.o -f elf -g -l list.lst lab4.asm
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 madaliev.o -o madaliev
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ./madaliev
madaliev akmaljon
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $
```

Команда main (рис 7) 8. Копируем файлы hello.asm, lab4.asm в локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/labs04 с помощью утилиты cp и проверяем наличие файлов с помощью утилиты ls.

```
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/arch-pc/lab04/report
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/arch-pc/lab04/report
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ cd ~/work/arch-pc/lab04
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  lab4.asm  lab4.o  list.lst  madaliev  madaliev.o  main  obj.o  report
aamadaliev@dk3n55 ~/work/arch-pc/lab04 $
```

Запуск файла

Выводы

В ходе выполнения работы я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы