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: Мурылев И.В.

Группа: НПИбд-03-25

МОСКВА

2025г.

Содержание

Цель работы

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

Теоретическое введение

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]
Здесь мнемокод — непосредственно мнемоника инструкции процессору,которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.о. метка перед командой связана с адресом данной команды.
Допустимыми символами в метках являются буквы, цифры, а также следующие символы:
_, $, #, @,~,. и ?.
Начинаться метка или идентификатор могут с буквы, ., _ и ?. Перед идентификаторами,
которые пишутся как зарезервированные слова, нужно писать $, чтобы компилятор трактовал его верно (так называемое экранирование).
Максимальная длина идентификатора 4095 символов.
Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора.
Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.
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 (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX

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 объектного файла — o, файла листинга — lst.

• Компоновка или линковка — этап обработки объектного кода компоновщиком (ld),который принимает на вход объектные файлы и собирает по ним исполняемый 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 программы,поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

Задание

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создайте копию файла hello.asm с именем lab4.asm
2. С помощью любого текстового редактора внесите изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем.
3. Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.
4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/. Загрузите файлы на Github.

Выполнение лабораторной работы

1. Выполняю перемещение в локальный репозиторий по адресу /home/ivmurihlev/Documents/work/study/2025-2026/Arch-pc/arch_pc/arch_evm/labs/lab04/report.

Затем копирование файла hello.asm в lab4.asm и проверка успешности копировании ls.

```
[ivmurihlev@PersonalMach report]$ ls
arch-pc--lab04--report.qmd  _assets  bib  hello.asm  image  Makefile  _quarto.yml  report.md  _resources
[ivmurihlev@PersonalMach report]$ cp hello.asm lab4.asm
[ivmurihlev@PersonalMach report]$ ls
arch-pc--lab04--report.qmd  _assets  bib  hello.asm  image  lab4.asm  Makefile  _quarto.yml  report.md  _resources
```

2. После измения проверяем командой cat.

```
[ivmurihlev@PersonalMach report]$ cat lab4.asm
SECTION .data
        hello:      db "Мурылев Иван",0xa
                helloLen:   equ $ - hello
SECTION .text
        global _start

_start:
        mov eax, 4
        mov ebx, 1
        mov ecx, hello
        mov edx, helloLen
        int 0x80

        mov eax, 1
        mov ebx, 0
        int 0x80
```

3. Транслируем файл.

```
[ivmurihlev@PersonalMach report]$ nasm -f elf lab4.asm
[ivmurihlev@PersonalMach report]$ ls
arch-pc--lab04--report.qmd  bib        image     lab4.o    _quarto.yml  _resources
_assets                     hello.asm  lab4.asm  Makefile  report.md
```

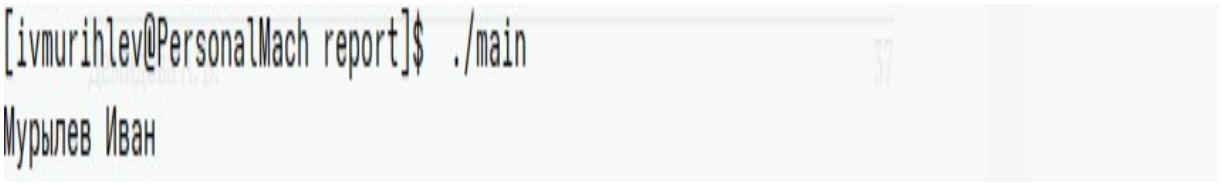
И как в примере создадим объектный файл.

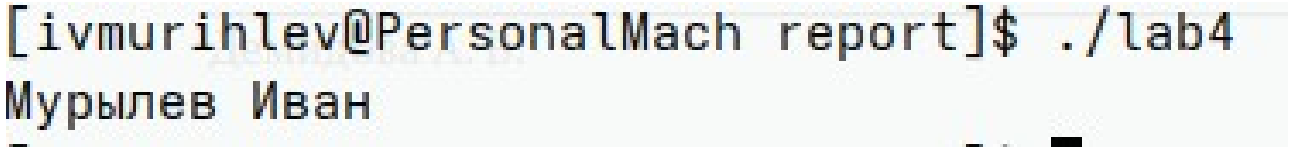
```
[ivmurihlev@PersonalMach report]$ nasm -o obiekt.o -f elf -g -l list.lst lab4.asm
[ivmurihlev@PersonalMach report]$ ls
arch-pc--lab04--report.qmd  bib        image     lab4.o    Makefile  _quarto.yml  _resources
_assets                     hello.asm  lab4.asm  list.lst  obiekt.o  report.md
```

Через него компануем и получаем исполняемую программу (Созданы 2 объектных файлов)

```
[ivmurihlev@PersonalMach report]$ ld -m elf_i386 lab4.o  -o lab4
[ivmurihlev@PersonalMach report]$ ls
arch-pc--lab04--report.qmd  bib        image  lab4.asm  list.lst  obiekt.o     report.md
_assets                     hello.asm  lab4   lab4.o    Makefile  _quarto.yml  _resources
```

4. Запуск исполняемых файлов





Выводы

Были освоенны базовые комманды Ассамблера и его различие от других языков программирования, которое заключается в более сложной компиляции и способу обращения к памяти.Также было изученно понятие регистра в Ассамблере.

Список литературы

1. https://esystem.rudn.ru/mod/resource/view.php?id=1030937