Отчет по лабораторной работе № 4

Архитектура компьютера

Иванов Александр Олегович

Содержание

# 1 Цель работы

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

# 2 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. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с Архитектура ЭВМ языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер. Программы, написанные на языке ассемблера, не уступают в качестве и скорости програм- мам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения команд в последовательности бит (нулей и единиц).

# 3 Выполнение лабораторной работы

Создание каталога для работы с программами на языке ассемблера NASM (рис. [1](#fig:001)).

Figure 1: Использование команды ‘mkdir’ для создания каталога

Figure 1: Использование команды ‘mkdir’ для создания каталога

Переход в созанный каталог (рис. [2](#fig:002))

Figure 2: Использование команды ‘cd’ для пеерхпда в каталог

Figure 2: Использование команды ‘cd’ для пеерхпда в каталог

Создание текстового файла с именем hello.asm (рис. [4](#fig:003))

Figure 3: Использование команды ‘touch’ для создания файла

Figure 3: Использование команды ‘touch’ для создания файла

Превращение текста программы в объектный код (рис. ??)

Figure 4: Использование команды ‘nasm -f elf’ для создания объектного файла

Figure 4: Использование команды ‘nasm -f elf’ для создания объектного файла

Компилирование исходного файла hello.asm в obj.o (опция -o позволяет задать имя объектного файла, в данном случае obj.o), при этом формат выходного файла будет elf, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l) (рис. [5](#fig:005)).

Figure 5: Использование команды ‘nasm -o obj.o -f elf -g -l list.lst hello.asm’ для создания объектного файла

Отправка объектоного файла на обработку компановщику(рис. [6](#fig:006))

Figure 6: Использование команды ‘ld -m elf_i386 hello.o -o hello’

Figure 6: Использование команды ‘ld -m elf\_i386 hello.o -o hello’

Запуск на выполнение созданный файл(рис. [7](#fig:007))

Figure 7: Использование команды ‘./hello’ и выводимый результат

Figure 7: Использование команды ‘./hello’ и выводимый результат

# 4 Самостоятельная работа

Создание копии файла hello.asm с именем lab4.asm(рис. [8](#fig:008))

Figure 8: Использование команды ‘./hello’ и выводимый результат

Figure 8: Использование команды ‘./hello’ и выводимый результат

Транслирование полученного текста программы lab4.asm в объектный файл. Выполнение компоновки объектного файла и запуск получившегося исполняемого файла (рис. [9](#fig:009))

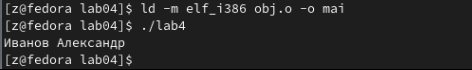


Figure 9: самостоятельная работа с командами NASM и вывод результатов

# 5 Выводы

В ходе лабораторной работы я освоил язык программирования низкого уровня асемблер, а имеено научился компилировать и собирать программы