Использования ассемблера для изучения архитектуры компьютера

Г. В. Гаркавенко, email: g.garkavenko@mail.ru[1]

Воронежский государственный педагогический университет

Аннотация. *В данной работе рассматривается то, как можно использовать написание программ на ассемблере при изучении работы регистров компьютера, а также проводить компиляцию и компоновку программ в командной строке, без использования среды программирования.*

Ключевые слова: *исследовательская работа, ассемблер, работа регистров, компиляция.*

Введение

В Воронежском государственном педагогическом университете на физико-математическом факультете учебные планы направления Педагогическое образование (с двумя профилями подготовки) профили «Математика», «Информатика» и Педагогическое образование профиль «Информатики и ИКТ» включают в себя дисциплину «Архитектура компьютера».

В рамках данного курса, для лучшего понимания структуры и работы регистров, изучаются основы языка ассемблера. В свою очередь, при изучении ассемблера необходимо уметь переводить десятичные числа в двоичную и шестнадцатеричную системы счисления. Переводы чисел между различными системами счисления изучаются в школьном курсе информатики и более подробно при изучении дисциплины «Теоретические основы информатики» [1].

Сейчас мало кто пишет программы на «чистом» ассемблере, чаще используются ассемблерные вставки в программах, написанных на языках высокого уровня, для увеличения быстроты выполнения программы [2]. Чтобы организовать программный ввод-вывод данных, в языках высокого уровня требуется всего пара строк, и за этими строками скрыто то, как на самом деле осуществляются эти действия. На ассемблере же для осуществления ввода и вывода данных потребуется написать более длинный код. При изучении ассемблера на занятиях по дисциплине «Архитектура компьютера» не ставится цель научится программировать на ассемблере, а он используется для понимания принципов работы с регистрами и памятью компьютера,

получения представления о том, что собой представляют машинные команды и способы адресации.

1. О выборе компилятора для языка ассемблер

Для того, чтобы писать программы и выполнять (запускать) их, вместе с языком программирования необходимо выбрать систему программирования. Система программирования включает в себя интегрированную среду разработки, текстовый редактор, компилятор или интерпретатор, библиотеки подпрограмм, компоновщик. Например, для языка Паскаль можно использовать систему программирования PascalABC.NET или FreePascal. Для C++ можно использовать CodeBlocks или Visual Studio. Причем в этих системах программирования используются разные компиляторы для одного и того же языка программирования.

Итак, для изучения архитектуры компьютера, посредством написания простейших программ на языке ассемблер необходимо выбрать компилятор. Существует множество разновидностей компиляторов для ассемблера: TASM, MASM, FASM, NASM и другие.

Для поддержки курса «Архитектура компьютера» используется компилятор TASM32, в состав пакета которого входит отладчик Turbo Debugger. Используя этот отладчик как раз и можно проследить за работой в регистрах. Ко времени изучения ассемблера студенты уже знакомы с программированием на одном из языков высокого уровня. И они привыкли использовать все те дополнительные возможности (подсветка синтаксиса, подсказки, удобную компиляцию, отладку, запуск программы и т.п.), которые предоставляет система программирования и не всегда понимают, что язык программирования и, например, редактор в котором набирается текст программы не одно и тоже. Поэтому, поскольку TASM не имеет собственной среды программирования, было решено не использовать среды сторонних разработчиков, а продемонстрировать студентам как производить компиляцию и сборку программы используя командную строку. Сам компилятор скачать на данный момент проблематично, но у нас он имеется достаточно давно.

Перед работой в корне рабочего диска (чтобы не пришлось писать длинные пути к файлам, и программы, предназначавшиеся ранее для dos, плохо работают с длинными именами папок и, файлами и папками, имеющими русские буквы в имени) создается папка asm и в неё помещаются все необходимые файлы. Для создания простых программ на ассемблере нам достаточно, чтобы в папке asm находились файлы:

tasm32.exe

tlink32.exe

td32.exe
rtm.exe
import32.lib

Для образца туда еще добавляется файл basa.asm, в котором записан простейший код на ассемблере.

Для удобства работы с командной строкой надо установить файловый менеджер Far, в нем также имеется текстовый редактор Edit, который будем использовать для набора текста программы.

2. Компиляция и сборка программы

Для начала на лекции рассказывается о названиях, назначении и структуре регистров, а также о типах данных ассемблера, которые непосредственно связаны с регистрами. Далее рассматривается структура программы и простейший пример, который находится в файле basa.asm.

Листинг 1

Простейшая программа на ассемблере для tasm32

```
includelib C:\asm\import32.Lib
extrn ExitProcess:near
.386
.model flat, stdcall
.data
 x db 4
 y db 3
 s db ?
.code
_start:
  mov eax,0
  mov al,x
  mov bl,y
  add al,bl
  mov s,al
  call ExitProcess,0
end _start
```

Данная программа выполняет всего лишь сложение двух чисел, но наша цель не освоить программирование на ассемблере как таковое, а изучить работу регистров, разобраться с тем, что программа компилируется и «линкуется».

Итак, прежде всего, программу надо скомпилировать. Запускаем менеджер файлов Far, переходим в папку asm и в командной строке набираем команду:

tasm32.exe /ml basa.asm

Можно воспользоваться сочетанием клавиш Ctrl+Enter, чтобы перенести в командную строку имя выбранного в списке панели Far

файла. Эти же действия можно осуществить с помощью командной строки Windows cmd, но придется дольше перемещаться по папкам и прописывать все пути и команды вручную рис.1.

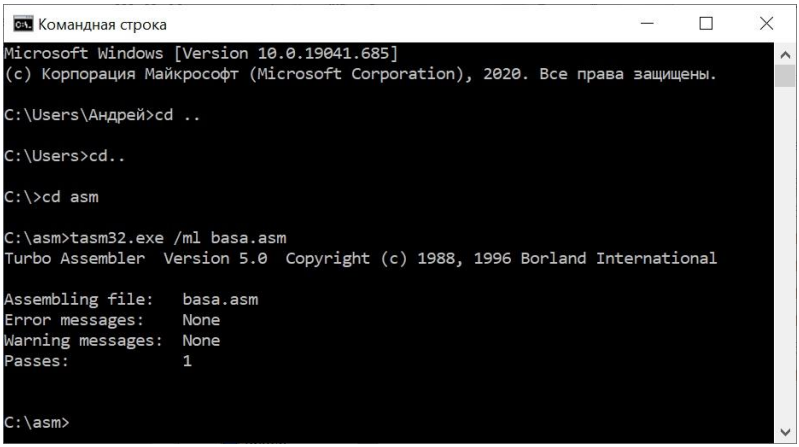

Рис. 1. Пример компиляции программы на ассемблере с использованием командной строки.

Видим, что компиляция прошла успешно. Ключ /ml указывает на то, что надо различать прописные и строчные буквы в идентификаторах. Результаты компиляции в Far можно посмотреть, потушив панели командой Ctrl+o, или увидев, что создан объектный файл basa.obj. При компиляции проверяется синтаксис и соответствие типов данных тем регистрам, в которые их загружают. На рис.2 приведен пример компиляции с ошибкой, когда строку *mov al,x* программы basa.asm заменили строкой *mov ax,x*.

Компилятор указывает номер строки, содержащую ошибку и ее тип. Если имеется в программе ошибка, то объектный файл не будет создан.

Следующий этап, это «линковка» (сборка) программы. Для этого вызываем компоновщик или, иначе, редактор связей:

tlink32.exe /ap basa.obj

Компоновщик включает в программу все необходимые внешние функции, находящиеся в подключаемых библиотеках. Если все прошло успешно, то будет создан исполняемый файл basa.exe. Но если в имена путей вкралась ошибка, то компоновщик, не найдя файла выдаст ошибку. Ключ /ap задает создание консольного приложения.

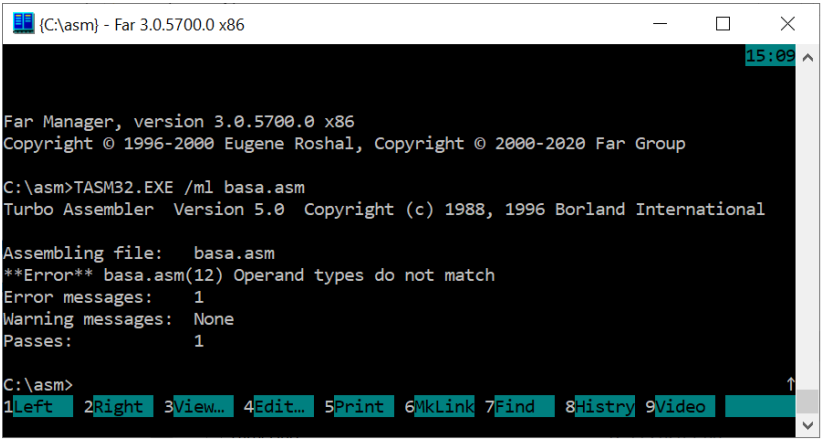

Рис. 2. Компиляция с ошибкой и просмотр результатов в Far.

3. Изучение работы 32-х битных регистров процессора

Наконец наше приложение готово и можно приступить к исследованию работы регистров процессора и основ архитектуры компьютера [3, 4]. Проверить работу программы в отладчике Turbo Debugger. И это как раз то, ради чего создавался exe-файл. Загружаем basa.exe в отладчик командой

td32.exe basa.exe

В окне Turbo Debugger (рис.3) можно увидеть загруженный исполняемый файл нашей программы, адреса памяти в которых расположен этот файл. Правее находятся 32-х битные регистры процессора, еще правее находится регистр флагов. Здесь студентам пригодится знание шестнадцатеричной системы счисления, именно в этой системе счисления хранятся данные в регистрах. А также можно увидеть связь между двоичной и шестнадцатеричной системами счисления, то есть системами с кратными основаниями. Известно, что при переводе числа из двоичной системы счисления в шестнадцатеричную, исходное число надо разбить на тетрады, и каждая такая тетрада из нулей и единиц соответствует цифре шестнадцатеричного числа. В отладчике можно увидеть, что в 32-х битных регистрах расположено 8 шестнадцатеричных цифр.

Чтобы внимательно рассмотреть, как изменяются данные в регистрах, программу надо выполнять пошагово, используя клавиши F7 или F8. Студенты должны увидеть, что регистры никогда просто так не бывают пустыми, значения в них не становятся просто так равными 0, через них проходят все команды процессора. Поэтому порой полезно

обнулить регистр перед записью в него данных, в программе basa.asm представлен один из способов обнуления 32-х битного регистра eax, а затем в его часть al, состоящую из 1 байта помещается значение 4, хранящееся в байте памяти по адресу [00402000]. В этот байт было помещено значение переменной x, заданной в сегменте данных программы рис.3.

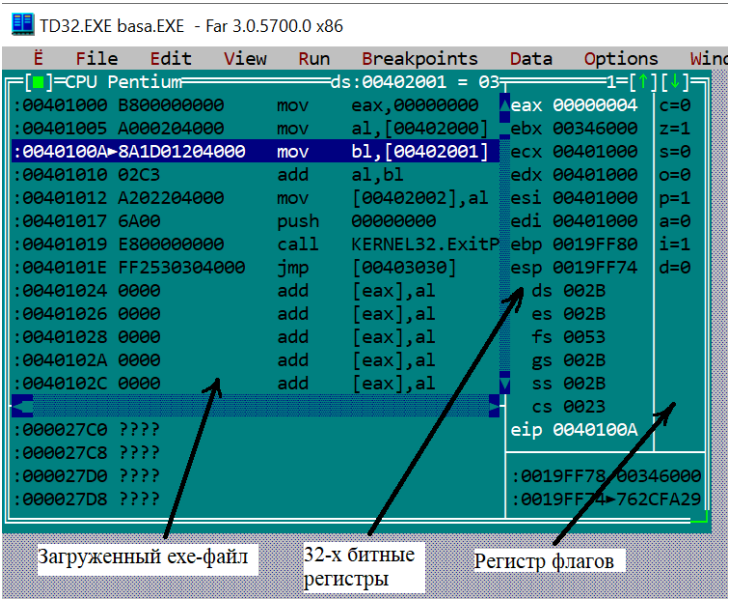

Рис. 3. Окно отладчика с загруженным исполняемым файлом.

Регистр ebx, для примера не был обнулен, а в байт bl будет помещено значение переменной y, которое как видно из загруженного исполняемого файла находится по адресу [00402001], заметим, что значение адреса на единицу больше, чем адрес памяти, по которому расположена переменная x. Здесь студенты могут увидеть реализацию принципа адресности памяти фон Неймана. Также можно продемонстрировать студентам выполнение другого принципа Джона фон Неймана: принципа однородности памяти. В соответствии с этим принципом программы и данные хранятся в одной и той же памяти. Поэтому ЭВМ не различает, что хранится в данной ячейке памяти - число, текст или команда. В окне отладчика видно, что в регистре eip - указателе команд, находится адрес команды, которая будет выполнена следующей. Этот адрес можно увидеть левее самой команды.

Чтобы, исследовать и понять как работают регистры при выполнении различных арифметических действий, достаточно написать программу, которая, например, вычисляет среднее арифметическое пяти чисел, заданных формулой ai=3*(i+2), где i меняется от 1 до 5.

Листинг 2

Программа нахождения среднего арифметического пяти чисел

```
includelib C:\asm\import32.lib
extrn ExitProcess:proc
.386
.model flat, stdcall
.data
    n db 5
.code
start:
     xor eax,eax
     mov bl,3
     mov ecx,0
     mov cl,n
     mov dl,0
     mov bh,1
 m1: mov al,bh
     add al,2
     mul bl
     add dl,al
     inc bh
     loop m1
     mov al,dl
     div n
     nop
     call ExitProcess,0
end start
```

С помощью данной программы можно продемонстрировать то, что некоторые регистры процессора предназначены для выполнения определенных действий, и произвести эти действия, например умножение и деление, возможно только с их использованием. Также, стоит заметить, что количество регистров ограниченно, а команд, которые необходимо выполнить программе много и изучить, какие средства предоставляет компьютер для выполнения программы.

Заключение

Изучение основ языка ассемблера способствует лучшему пониманию архитектуры компьютера, дает представление о том, как работает аппаратная часть компьютера. Все это не могут помочь увидеть языки высокого уровня.

Для изучения основ архитектуры компьютера можно использовать учебную программу «ЛамПанель» с сайта Константина Полякова https://kpolyakov.spb.ru/prog/lamp.htm.

Список литературы

1. Башарина, С. О. Исследование особенностей работы с вещественными числами / С. О. Башарина, Г. В. Гаркавенко, Е. Р. Найденкина // Информатика: проблемы, методы, технологии. Материалы XX Международной научно-методической конференции. Под редакцией А.А. Зацаринного, Д.Н. Борисова. (Воронеж, 13-14 февраля 2020 г.) – Воронеж, 2020. – С. 1791-1797.

2. Гаркавенко, Г. В. О проблемах преподавания языков программирования в педагогическом вузе / Г. В. Гаркавенко, Е.А. Кубряков// Информатика: проблемы, методология, технологии. Информатика в образовании материалы XVIII Международной школы-конференции. – Воронеж, 2018. – С. 17-22.

3. Гаркавенко, Г. В. Об изучении оценки вычислительной сложности алгоритмов / Г. В. Гаркавенко, Е. О. Савенкова // Информационные технологии в образовательном процессе вуза и школы Материалы XII Региональной научно-практической конференции. Научный редактор В.В. Малев. (Воронеж, 28 марта 2018 г.) – Воронеж, 2018. – С. 46-50.

4. Гаркавенко, Г. В. Олимпиада по информатике в вгпу как средство популяризации предмета среди обучающихся / Г. В. Гаркавенко, Е.А. Кубряков// Актуальные проблемы прикладной математики, информатики и механики сборник трудов Международной научной конференции (Воронеж, 17-19 декабря 2018 г.) – Воронеж, 2019. – С. 1393-1398.