

1 Форматы команд и способы адресации

1.1 Цель работы

- Изучение среды программирования Microsoft Visual studio.
- Изучение базовых команд МП с архитектурой i386.
- Изучение способов адресации данных.

1.2 Порядок выполнения работы

- Изучение методических указаний к работе.
- Выполнение общих заданий.
- Защита работы с выполнением контрольных заданий преподавателя.

1.3 Теоретические сведения

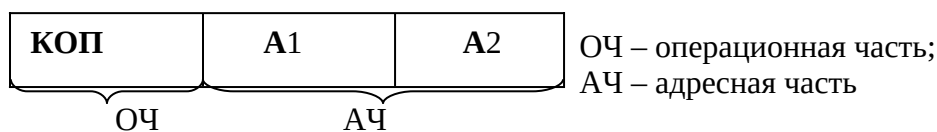
1.3.1 Основные понятия и определения

Информация, поступающая на вход ЭВМ, – числа, логические переменные, текст, графика и др., в том числе и команды - вводится в компьютер в виде последовательности 0 и 1 и далее представляется и хранится в виде *двоичных слов фиксированной длины*: 8, 16, 32, 64 бита, при этом 8-битное слово принято называть байтом, 16-битное – словом, 32-битное – двойным словом.

Команда в ЭВМ или машинная команда - это упорядоченная последовательность бит (двоичное слово), с помощью которой указывают:

- наименование операции, инициируемой командой (код операции КОП);
- адреса A1, A2 операндов, участвующих в операции.

Обобщенный формат команды (представление) изображен на рисунке (Рисунок 1).



ОЧ – операционная часть;
АЧ – адресная часть

Рисунок 1 - Формат команды

Формат команды задает длину команды в байтах или словах, количество полей для указания адреса, способ адресации, расположение полей в команде и т.д.

Для более наглядного представления машинных команд используются их мнемоники. Например, для команды пересылки данных из регистра ВХ в регистр АХ используется мнемоника **MOV AX, BX**. Программы, написанные в мнемоническом коде, обрабатываются ассемблерами.

В настоящее время ассемблеры как самостоятельные среды программирования практически не используются. Большинство аппаратно ориентированных программ (например, драйверы) создаются на языке C++, а критически важные участки оформляются в виде подпрограмм на языке ассемблера.

Среда программирования Microsoft Visual Studio позволяет встраивать в программу на языке C++ фрагменты кода на языке ассемблера (ассемблерные вставки). Эта возможность

среды позволяет в простой и наглядной форме изучить мнемоники команд процессора и регистровую модель процессора.

Лабораторная работа предназначена для изучения системы команд МП с архитектурой i386. Архитектура i386 является 32-разрядной, и поддерживается моделями процессоров i386, i486, P3, P4, Core2, i7, AMD Athlon и др. Среда программирования Microsoft Visual Studio позволяет разрабатывать приложения как для 32-разрядных, так и для 64-разрядных процессоров.

1.3.2 Программная модель процессора i386

Процессор с архитектурой i386 содержит в своем составе набор 32-разрядных регистров общего назначения (EAX, EBX, ECX, EDX), а также регистры указатели, регистры сегментов, указатель команд и др. (Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5). В настоящей работе будут изучаться только регистры общего назначения (Рисунок 2).

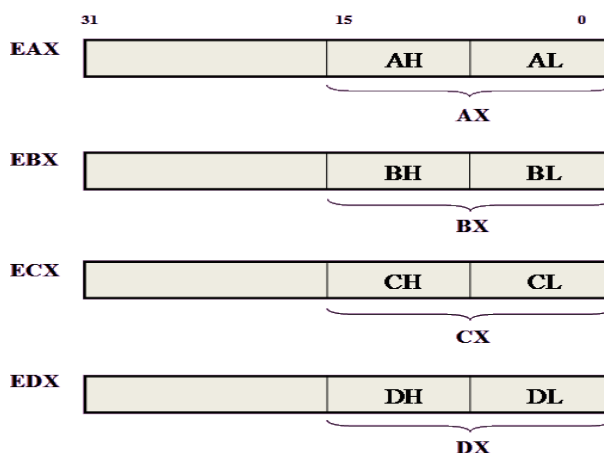


Рисунок 2 – Регистры общего назначения

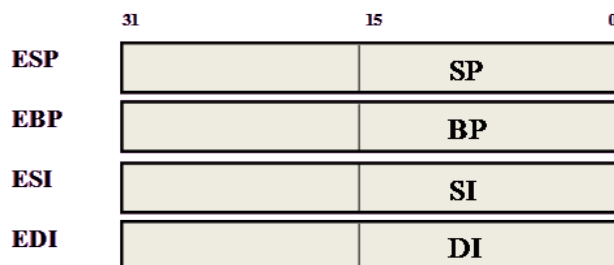


Рисунок 3 - Регистры указатели адреса

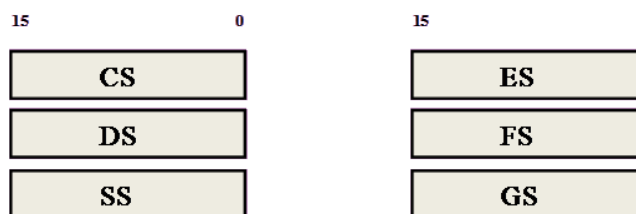


Рисунок 4 - Регистры указатели сегментов памяти



Рисунок 5 - Регистр признаков (флагов)

Регистры общего назначения это 32-разрядные ячейки памяти, размещенные непосредственно в процессоре и используемые для выполнения операций над данными. В языке ассемблера регистры могут адресоваться как 32-разрядные ячейки (мнемоника EAX), как 16-разрядные (мнемоника AX) и как отдельные байты (мнемоники AH и AL). Самый первый регистр EAX имеет специальное название – аккумулятор.

В командах ассемблера соответствующие мнемоники регистров могут указываться для загрузки байта, слова (16 бит) и двойного слова (32 бит) данных, например:

```
MOV EAX, 100000; //Загрузка числа 100000 в 32-разрядный регистр EAX
MOV AH, 11;      //Загрузка числа 11 во второй байт регистра EAX (8-разрядный
регистр AH)
MOV AX, 100;     //Загрузка числа 100 в 16-разрядный регистр AX (младшее слово
32-разрядного регистра EAX)
```

Аналогичные мнемоники имеют и остальные регистры общего назначения:

- EBX – BX, BH, BL;
- ECX – CX, CH, CL;
- EDX – DX, DH, DL.

Например:

```
MOV BL, 10; //Загрузка числа 10 в младший байт регистра EBX
```

1.3.3 Хранение данных в ЭВМ

Исходные данные для решения задачи могут храниться либо в регистрах общего назначения (рег) МП, либо в оперативной памяти (ОП) (ячейки М). Ячейки (рег и М), хранящие исходную информацию, назовем источником, а регистры и ячейки памяти принимающие информацию от источника — приемником.

В МП i386 приемник расположен слева, источник - справа. Информация передается от источника к приемнику.

Например, в команде передачи данных

MOV AX, BX

слева указан приемник (регистр AX), а справа регистр BX. На рисунке (Рисунок 6) приведены все возможные способы передачи данных.

reg	→	М	Из регистра в память
М	→	reg	Из памяти в регистр
reg	→	reg	Из регистра в регистр

Рисунок 6 – способы передачи данных

1.3.4 Способы адресации

Данные (элементы информации), участвующие в операции, определяются с помощью адресов, указанных в командах. Существует большое число способов представления адресов

в командах и их определения (вычисления) с целью доступа к операндам (данным) на основе информации, указанной в команде.

Правило определения (вычисления) адреса операнда на основе информации, указанной в команде, называют способом адресации. Т.е. способ адресации определяет порядок выполнения действий над адресной частью команды и содержимым одного или нескольких регистров ЦП для вычисления исполнительного адреса, по которому хранится операнд в памяти.

Таким образом, под исполнительным адресом **Аисп** понимается адрес операнда в памяти (М) или в регистре (АХ, ВХ, СХ, ДХ и др.). **Аисп** - это целое двоичное число без знака, определяющее номер ячейки в памяти.

Прямая адресация.

Второе название данного способа адресации – регистровая. В качестве операнда указывается регистр, например:

MOV EAX, EBX;

Команда пересылает данные в регистр EAX (32-разрядный регистр) из регистра EBX.

Абсолютная адресация.

Адрес данных задается в самой команде, например:

MOV EAX, [0xFF00];

Примечание: обычно адрес указывается в 16-ричной системе счисления. Признаком 16-ричной системы счисления являются символы 0х, например: 0xFF – означает число FF в 16-ричной системе и 255 в десятичной.

Непосредственная адресация.

Операнд-константа задается непосредственно в самой команде, например:

MOV EAX, 10; //Загрузка числа 10 в 32-разрядный регистр EAX.

Косвенная адресация.

Адрес операнда находится в регистре, например:

MOV EAX, [EBX];

Загрузка данных в 32-разрядный регистр EAX из памяти по адресу, хранящемуся в регистре EBX. В данном случае квадратные скобки указывают на косвенную адресацию.

В данной лабораторной работе мнемонические команды процессора будут оформляться в виде ассемблерных вставок в программе на языке с++. В языке С++ используются расширенные мнемонические команды ассемблера, которые позволяют связать программу на С++ с ассемблерными вставками. Например, если объявлена переменная в программе на С++, то данные этой переменной можно загрузить в регистр процессора:

```
int nVar;           //Объявлена переменная целого типа  
  
nVar = 10;          //Присваиваем переменной значение 10  
  
asm MOV EAX, nVar;  //Пересылаем значение переменной (10) в регистр EAX.
```

В данном примере ключевое слово **asm** обозначает ассемблерную вставку.

Существуют также другие, более сложные способы адресации данных, которые не рассматриваются в данной лабораторной работе.

Технические подробности: Переменная на с++ физически представляет собой ячейку оперативной памяти компьютера. В исполняемом модуле мнемоника переменной (ее название) заменяется на физический адрес памяти, который используется в командах процессора. Для получения адреса переменной в ассемблерных вставках на с++ используется операция "&".

1.4 Практическая часть

Практическая работа состоит из двух частей:

- изучение среды программирования Microsoft Visual Studio;
- ввод и отладка простейшей ассемблерной программы;
- выполнение индивидуального задания и ответы на вопросы преподавателя.

1.4.1 Создание проекта консольного приложения

Необходимо произвести следующие действия:

- Запустить среду Microsoft Visual Studio.
- Выбрать пункты меню File-New-Project.
- В появившемся диалоговом окне выбрать тип проекта «Project types – Visual C++ - Win32» (Рисунок 7).

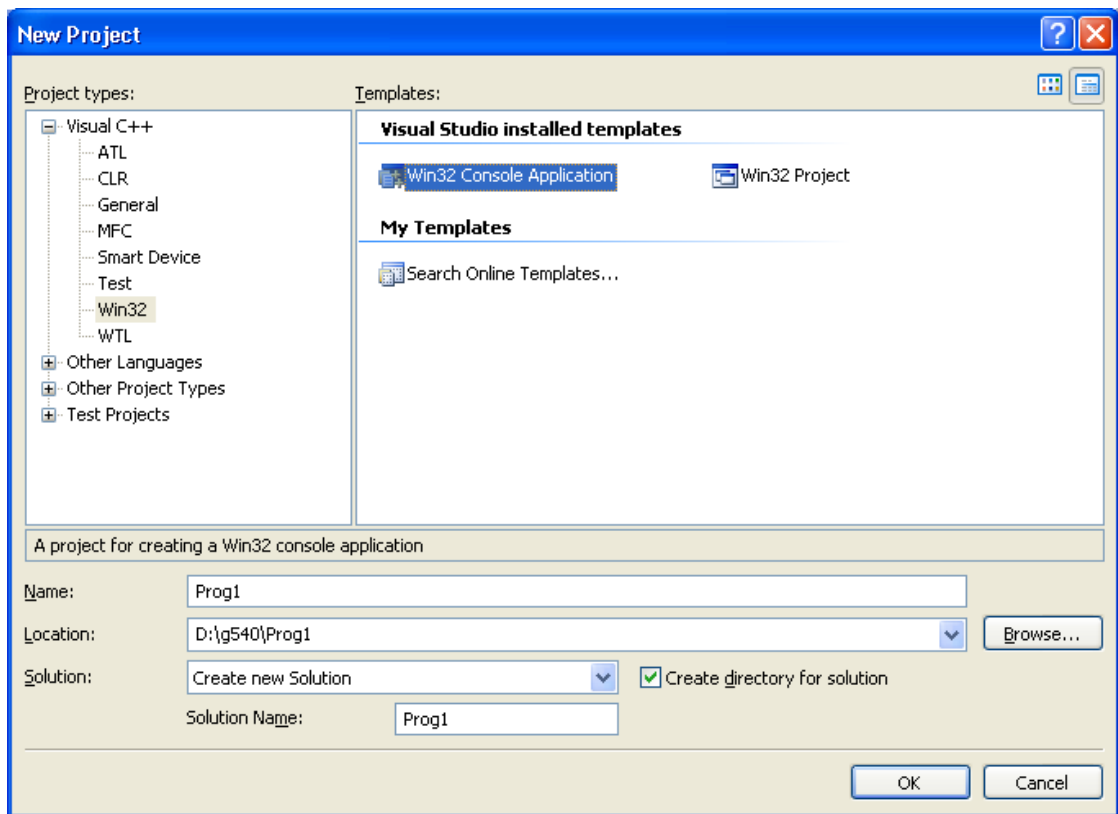


Рисунок 7 – Создание проекта консольного приложения

- Выбрать тип приложения «Win32 Console Application». Указать имя проекта (поле Name), каталог для создания нового проекта (название каталога уточните у преподавателя) и нажать кнопку Ок.

Если все сделано правильно, то на экране отобразится окно, содержащее шаблон текста программы на C++ (Рисунок 8).

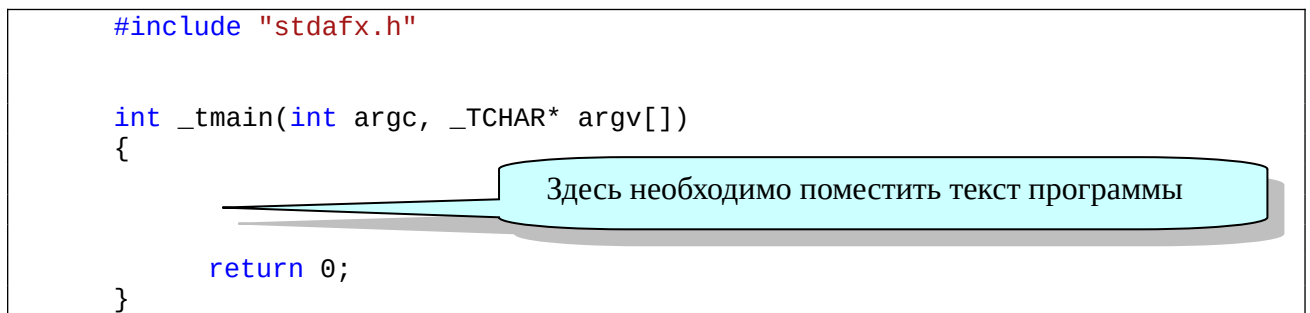


Рисунок 8 – Текст шаблона программы на C++

Текст экспериментальной программы необходимо поместить внутри функции «_tmain», которая ограничена фигурными скобками.

1.4.2 Создание простейшей программы на C++ и ее отладка

Внутри функции «_tmain» введите текст программы, приведенный ниже (Рисунок 9).

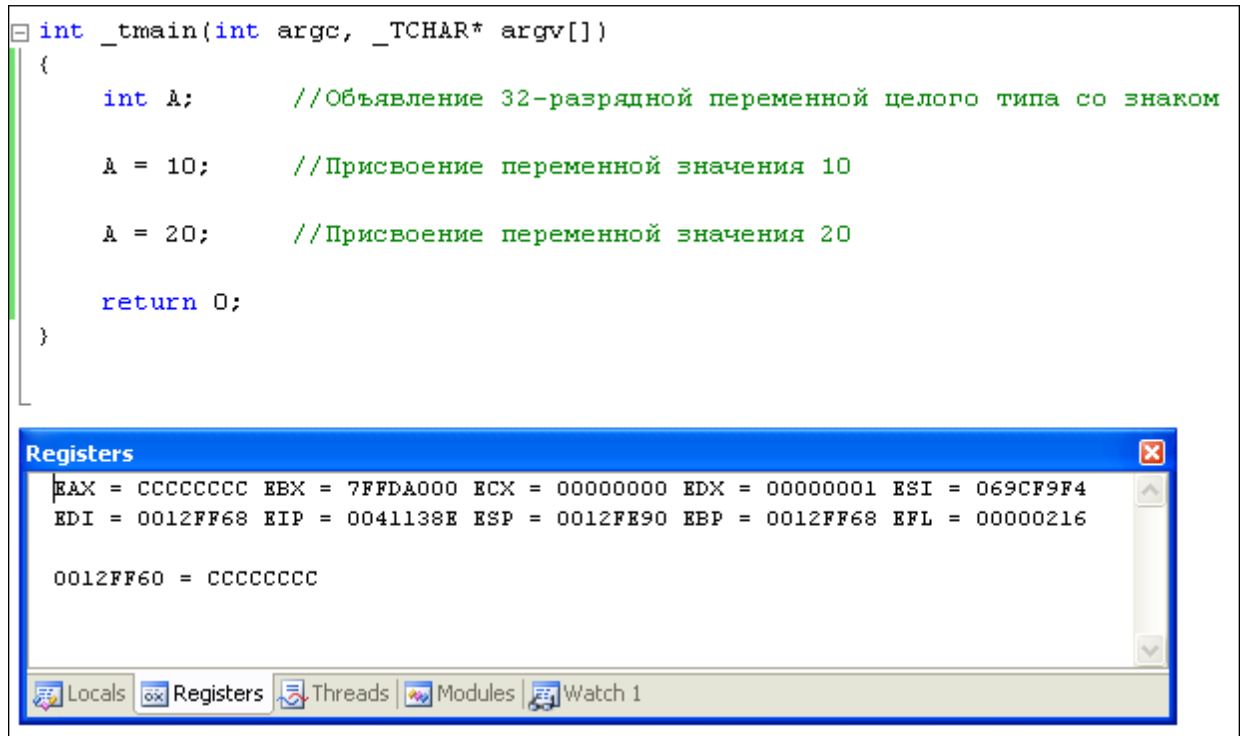



Рисунок 9 – Вид программы и окна регистров

Переведите курсор на строку, содержащую текст «A = 10;» и нажмите клавишу F9. При этом в левой части строки должна появиться точка останова (Рисунок 10). Запустите программу в режиме отладчика (Нажмите клавишу F5 или кнопку  - Start Debugging). При этом выполнение программы должно остановиться в указанной точке останова.

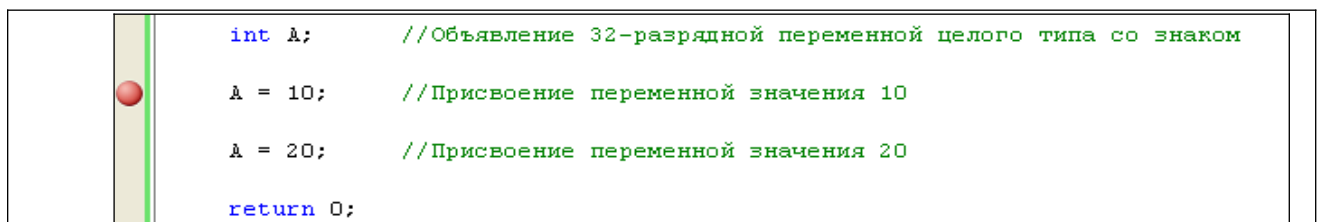


Рисунок 10 – Точка останова

Отобразите окно регистров процессора (Рисунок 9) путем выбора пунктов меню Debug-Windows-Registers или нажмите сочетание клавиш «Alt»+«5». Также вам понадобится окно «Locals» для просмотра значения переменных. Если оно не отображено, то выберете пункты меню Debug-Windows-Locals или нажмите сочетание клавиш «Alt»+«4».

Выполните программу по шагам нажимая клавишу F10. При этом обратите внимание на изменение значений регистров процессора. Значения переменных и регистров в пошаговом режиме можно посмотреть просто подведя курсор мыши к переменной (или к мнемонике регистра) в тексте программы.

*Примечание: в с++ для задания типа переменной других разрядностей используются ключевые слова **short** (16 бит) и **char** (8 бит) вместо **int**, например:*

short A2; //Объявлена переменная типа "слово со знаком"
char A3; //Объявлена переменная типа "байт со знаком"

1.4.3 Создание ассемблерных вставок

Добавьте в текст программы строки, приведенные ниже (ключевое слово `_asm` обозначает ассемблерную вставку):

```
A = 20;           //Присвоение переменной значения 20

_asm MOV EAX,10;   //Загрузка в EAX числа 10
_asm MOV EAX,A;     //Загрузка в EAX значение переменной A

void *pA = &A;     //Получение адреса переменной

_asm MOV EBX,pA;    //Загрузка адреса переменной в EBX
_asm MOV [EBX],100; //Запись значения 100 по адресу переменной A

return 0;
```

Рисунок 11 – Ассемблерная вставка

Запустите программу в отладочном режиме. Пройдите по шагам участок программы с ассемблерными вставками. Следите за значениями регистров процессора и значением переменной «А» (значение переменной можно узнать просто переведя курсор мыши на эту переменную).

1.4.4 Выполнение индивидуального задания

В таблице (Таблица 1) приведены варианты индивидуальных заданий по вариантам. Все задания должны выполняться с помощью ассемблерных команд.

Таблица 1 – Варианты заданий

Вариант №	Расшифровка задания
1	Объявить 3 переменные. С помощью команд ассемблера переслать данные из 1-й переменной во вторую, из 2-й в 3-ю.
2	Переслать данные в регистр АХ отдельными байтами (используя мнемоники AL и AH) и сохранить результат в 16-битной переменной
3	Переслать младшее слово регистра EAX в регистр EBX отдельными байтами
4	Переслать данные регистра АХ в две 8-битные переменные
5	Обменять значения 2х переменных
6	Обменять значения 2х регистров процессора с использованием оперативной памяти (переменных на с++)

Общее задание: переслать данные из одной переменной в другую используя абсолютную адресацию. Для выполнения этого задания необходимо вначале определить абсолютные адреса переменных в режиме отладки.

1.5 Контрольные вопросы

- Что такое регистры процессора?
- Какие типы адресации вы знаете?
- Как осуществляется обмен данными между процессором и ОП.
- Поясните понятие исполнительного адреса?
- Что такое прямая адресация?
- Что такое абсолютная адресация?
- Какие регистры общего назначения вы знаете?

- Что такое мнемоника команды?