Организация связи программ, реализованных на разных ЯП

Основы языка Assembler

С точки зрения программиста архитектура представляет собой набор програмнодоступных регистров систему команд и систему адресации. По сути архитектура с программной точки зрения представляет собой базовую программную модель процессора.

Для Intel:

* 8 регистров общего назначения, представляющих для хранения данных и указателя
* Регистра сегментов.
* Регистр управления и контроля сигналов
* Регистр счетчик команд

В состав 4-х регистров общего назначения входят 8 разрядные общие регистры.

EAX является регистром – аккумулятором, который используется для ряда арифметических операций.

EBX указывает на данные, находящиеся в сегменте данных, на которые указывает сегментный регистр DS. BX – регистр общего назначения, может свободно использоваться для указания операндов

ECX – регистр счетчик. Используется в качестве счетчика циклов и элементов при выполнений циклических и строковых операций. В качестве операнда может быть использован.

EDX – регистр общего назначения, используется специфически для указания на порты ввода – вывода.

EDI – регистр общего назначения, используется в качестве указателя на данные адресуемые через сегментный регистр ES.

ESI – Указывает на данные, адресуемые через сегментный регистр ES. Используется для организации индексной адресации.

EBP – базовый регистр, содержит указатель на данные хранящиеся в стеке, адресуемом через сегментный регистр FS. Используется для базовой адресации и при организации переходов между подпрограммами.

ESP – Указатель на доступную ячейку в стеке.

Сегментные регистры

SS – сегмент стека

DS – данные

CS – код

ES – расширенный

FS – дальний сегмент

GS – глобальный

EIP – програмно недоступен регистр, доступен только через JMP, в который указывается адрес в который следует перейти.

Регистр флагов. Эквивалентен регистру состояний процессора и хранит информацию о текущем его состоянии. Регистр флагов включает в себя 6 флагов состояний и 3 бита управления состояниями.

CF – флаг переноса. Инициирует перенос при выполнении арифметических операций, служит индикатором ошибки при обращении к системным функциям.

PF – Устанавливается в 1, если младшие 8 бит результата операций содержит четное число двоичных единиц.

AF – флаг дополнительного переноса. Используется в операции с упакованными данными, представление в виде 2-10 чисел. Инициирует перенос в старшую тетраду или заем.

ZF – Флаг нуля. Устанавливается 1 если результат равен 0.

SF – Флаг знака. Показывает знак результата операции.

OF – флаг переполнения. Фиксирует переполнение, выход результата за пределы допустимого значения.

DF – управляющий флаг направления. Используется особой группой команд для обработки строк.

TF – используется в отладчиках для организации пошагового выполнения программы. 1 – пошаговый режим. 0 – без остановки.

IF – разрешает или запрещает процессору реагировать на прерывания от внешних устройств.

При выполнении любой команды процессор обращается к памяти, в которой хранятся операнды и команды, способ формирования адреса операнда или метки перехода называют способом адресации. Инструкции Assembler включает наборы команд с операндами и без. Некоторые требуют явного указания, другие используют операнды по умолчанию. Местоположение операнда называют его эффективным адресом. Самым общим случаем вычисления эффективного адреса делается так EA = база + (индекс\*множитель) + смещение. Она называется базово-индексной.

Способы адресации:

* Прямая. В случае прямой адресации адрес операнда или перехода указывается в явном виде.
* Непрямая. требуется выполнить предварительные вычисления адреса.

**Прямая адресация**

Непосредственная адресация предполагает, что в качестве операнда в программе используется явно заданное значение, либо явный указатель на ячейку памяти.

Mov позволяет передать данные из операнда источника в операнд приемника. Mov ax, 14h

Прямая регистровая адресация предполагает, что операнды хранятся в программно доступных регистрах. Mov ax, bx

Неявная адресация. Операнд определяется самой мнемоникой команды.

LOOP – цикл.

Абсолютная адресация. В качестве операнда указывается адрес конкретной ячейки памяти. JMP 0x12345678

**Непрямая адресация**

Предполагает, что в качестве операнда указывается регистр, в котором хранится адрес используемого значения. Mov ax, [bx]

Косвенная, в качестве операнда указан адрес, в котором хранится адрес требуемого значения. MOV ax, [1CD5h]

Базовая адресация (относительная). Предполагает, что для вычисления адреса операнда необходимо выполнить смещение, относительно базового регистра. Mov ax, [bp]+5 или mov ax, [bp][5]

Индексная, используется для адресации массивов. При организации индексной адресации используются регистры, хранящие указатель на индексируемый массив.

Стековая адресация.

mass db 250 dup (‘\*’)

lea – команда передачи адресов

Все машинные команды по функциональному назначению можно разделить на команды передачи данных, обработки данных, команды управления и доп команды.

* MOV <приемник> <источник>. При этом размеры источника и приемника должны совпадать. При команде mov только один из операндов может быть задан косвенно.
* In – читает из порта ввода – вывода и записывает в порт ввода вывода. Чтение производится в регистр dx или по непосредственному указанному адресу, результат чтения обычно помещается в регистр аккумулятор. 0х60 – последняя нажатая клавиша
* Out – работает наоборот

Арифметические команды:

* Add, Sub <приемник > <операнд>
* Inc, Dec
* MUL, DIV. Без знаковое целочисленное умножение и деление
* IMUl, IDIV. Знаковые операции

Логические команды:

* And
* OR
* XOR
* NOT

Команды передачи управления.

Позволяют организовать вычислительный процесс используя следующие команды jmp в качестве параметра указывается адрес, обычно это метка.

Условные переходы делятся на 2 типа:

* По равенству флага. Jz, jc, js, jo. Je – равенство, jne – неравно, ja – больше, jb - меньше
* По неравенству флага

Loop <метка> - организация цикла

Команды работы со стеком

Push <данные>

Pop <данные>

Call <adr> - передача управление функции

RET (RETURN) – возвращение обратно по адресу из стека

Команда загрузки эффективного адреса и записывает его в первый операнд.

LEA 01, [02]

SHR 01 {size} – логический сдвиг вправо

SHL 01 {size} – логический сдвиг влево

SAR 01 {size} – арифметический вправо

SAL 01 {size} – арифметический влево

RCR 01 {size} – циклический сдвиг вправо

RCL 01 {size} – циклический влево

ROR 01 {size} -

ROL 01 {size}

TEST – поразрядное сравнение, без сохранения результата, но с сохранением флага

**Каркас программы**

Программа на ассемблере состоит из набора секций, каждая секция определяется ассемблерной директивой.

***.MODEL FLAT, STDCALL***

***.DATA***

***<инициализированные данные>***

***.DATA?***

***<не инициализируемые данные>***

***.CONST***

***<константы>***

***.CODE***

***<метка> proc***

***<код>***

***<метка> endp***

***END***

**Интерфейс ассемблера с языками высокого уровня**

При организации взаимосвязи программ, написанных на различных ЯП используют специальное соглашение, которое принято называть соглашение об именах и связях. Данное соглашение определяет следующие особенности работы подпрограмм:

1. Расположение входных параметров подпрограмм и возвращаемых значений:
   1. Передача через регистр
   2. Через стек
   3. Через динамическую память
2. Порядок передачи параметров. Т. Е. определяется в каком порядке параметры указанные в заголовке процедуры будут располагаться в стеке при передаче управления. При использовании регистров данное соглашение определяет порядок сопоставления регистра и параметра.
   1. В прямом порядке. Параметры идут по порядку
   2. В обратном: от конца.
3. Кто возвращает указатель стека на исходную позицию?
   1. Если вызываемая подпрограмма. Сокращается количество команд необходимых для вызова подпрограммы.
   2. Если вызывающая, то сам вызов становится сложнее и возврат внутри вызывающей подпрограммы должен очищаться стек.
4. Содержимое каких регистров подпрограмма обязана освободить перед возвратом.

Соглашение вызова зависит от целевой машины на которой реализуется программа и используемого компилятора. Способ передачи параметров может быть изменен путем внесения соответствующих директив. Эти директивы определяют способы синхронизации области стека, при использовании его (стека) подпрограммой. Существует следующий набор директив:

1. Fastcall. Предполагает, что параметры передаются слева направо, за восстановление отвечает вызываемая подпрограмма. Для передачи параметров могут использоваться регистры eax, edx, ecx (pascal) и ecx, edx (VC++).
2. Pascal. Передача параметров слева направо, за восстановление отвечает вызываемая подпрограмма, параметры в регистры не дублируются.
3. Cdecl. Параметры передаются в обратном порядке. За восстановление отвечает основная программа (вызывающая). Параметры не дублируются
4. Stdcall. В обратном порядке. За восстановление отвечает вызываема подпрограмма.

Если возвращаемое функцией значение не превышает размеры 1 байта, то оно помещается в AL. 2 байта – AX, до 4-х байт – EAX, если больше, то EDX:EAX.