Базовая система команд микропроцессора

Базовую систему команд микропроцессора можно условно разделить на несколько групп по функциональному назначению:

- команды передачи данных
- команды установки единичного бита
- команды работы со стеком
- команды ввода-вывода
- арифметические команды
- логические команды
- сдвиговые команды
- команды коррекции двоично-десятичных чисел
- команды преобразования типов
- команды управления флагами
- команды прерываний
- команды передачи управления
- команды синхронизации работы процессора

- команды побитового сканирования
- строковые команды

Кроме базовой системы команд микропроцессора существуют также команды расширений:

- X87 – расширение, содержащее команды математического сопроцессора (работа с вещественными числами)
- MMX – расширение, содержащее команды для кодирования/декодирования потоковых аудио/видео данных;
- SSE – расширение включает в себя набор инструкций, который производит операции со скалярными и упакованными типами данных;
- SSE2 – модификация SSE, содержит инструкции для потоковой обработки целочисленных данных, что делает это расширение более предпочтительным для целочисленных вычислений, нежели использование набора инструкций MMX, появившегося гораздо раньше;
- SSE3, SSE4 – содержат дополнительные инструкции расширения SSE.

В таблице команд приняты следующие обозначения:
r – регистр
m – ячейка памяти
c – константа
8, 16, 32 – размер в битах
На все базовые команды процессора накладываются следующие ограничения:

- Нельзя в одной команде оперировать двумя областями памяти одновременно. Если такая необходимость возникает, то нужно использовать в качестве промежуточного буфера любой доступный в данный момент регистр общего назначения.

- Алгоритмы сортировки и поиска
- Задачи и их решение
- Программирование микроконтроллеро

Свежие записи

- Односвязный линейный список на базе классов ООП
  14.12.2018
- UML-диаграммы классов
  17.10.2017
- Защищено: Пользовательский сигнал
  28.06.2017
- Защищено: Цифро-аналоговый преобразователь
  19.06.2017
- Граф
  19.05.2017

Социальные сети

- Нельзя оперировать сегментным регистром и значением непосредственно из памяти. Поэтому для выполнения такой операции нужно использовать промежуточный объект. Это может быть регистр общего назначения или стек.
- Нельзя оперировать двумя сегментными регистрами. Это объясняется тем, что в системе команд нет соответствующего кода операции. Но необходимость в таком действии часто возникает. Выполнить такую пересылку можно, используя в качестве промежуточных регистры общего назначения. Например,

```
mov ax,ds
mov es,ax     ; es=ds
```

- Нельзя использовать сегментный регистр CS в качестве операнда приемника, поскольку в архитектуре микропроцессора пара CS:EIP всегда содержит адрес команды, которая должна выполняться следующей. Изменение содержимого регистра CS фактически означало бы операцию перехода, а не модификации, что недопустимо.
- Операнды команды, если это не оговаривается дополнительно в описании команды, должны быть одного размера.

Команды передачи данных

Основной командой передачи данных является команда MOV, осуществляющая операцию присваивания:

```
MOV приемник, источник
```

Команда MOV присваивает значению операнда приемника значение операнда источника. В качестве приемника могут выступать регистр общего назначения, сегментный регистр или ячейка памяти, в качестве источника могут выступать константа, регистр общего назначения, сегментный регистр

или ячейка памяти. Оба операнда должны быть одного размера.

Команды передачи данных представлены в таблице.

Команда	Операнды	Пояснение	Описание
MOV	r(m)8,r8 r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32 r(m)8,c8 r(m)16,c16 r(m)32,c32	r(m)8=r8 r(m)16=r16 r(m)32=r32 r8=r(m)8 r16=r(m)16 r32=r(m)32 r(m)8=c8 r(m)16=c16 r(m)32=c32	Пересылка операндов
XCHG	r(m)8, r8 r8, r(m)8 r(m)16,r16 r16, r(m)16 r(m)32, r32 r32, r(m)32	r(m)8 ↔r8 r8 ↔r(m)8 r(m)16↔r16 r16 ↔r(m)16 r(m)32↔r32 r32 ↔r(m)32	Обмен операндов
BSWAP	r32	TEMP ← r32 r32[7..0]←TEMP[31..24] r32[15..8]←TEMP[23..16] r32[23..16]←TEMP[15..8] r32[31..24]←TEMP[7..0]	Перестановка байтов из порядка «младший – старший» в порядок «старший – младший»
MOVSX	r16, r(m)8 r32, r(m)8 r32, r(m)16	r16,r(m)8 DW ← DB r32,r(m)8 DD ← DB r32,r(m)16 DD ← DW	Пересылка с расширением формата и дублированием знакового бита
MOVZX	r16,r(m)8 r32,r/m8 r32,r/m16	r16,r(m)8 DW ← DB r32,r(m)8 DD ← DB r32,r(m)16 DD ← DW	Пересылка с расширением формата и дублированием нулевого бита

Команда	Операнды	Пояснение	Описание
XLATXLATB	m8	AL=DS:[(E)BX+unsigned AL]	Загрузить в AL байт из таблицы в сегменте данных, на начало которой указывает EBX (BX); начальное значение AL играет роль смещения
LEA	r16, m r32, m	r16=offset m r32=offset m	Загрузка эффективного адреса
LDS	r16,m16 r32,m16	DS:r=offset m	Загрузить пару регистров из памяти
LSS		SS:r=offset m	
LES		ES:r=offset m	
LFS		FS:r=offset m	
LGS		GS:r=offset m	

Команды установки единичного бита

Проверяют условие состояния битов регистра EFLAGS и, если условие выполняется, то младший бит операнда устанавливается в 1, в противном случае в 0. Анализ битов производится аналогично условным переходам.

Команда	Операнды	Пояснение
SETA SETNBE	r(m)8	CF=0 и ZF=0
SETAE SETNB SETNC		CF=0
SETB SETC SETNAE		CF=1
SETBE SETNA		CF=1 или ZF=1
SETE SETZ		ZF=1

Команда	Операнды	Пояснение
SETG SETNLE		ZF=0 и SF=OF
SETGE SETNL		SF=OF
SETL SETNGE		SF!=OF
SETLE SETNG		SF!=OF или ZF=1
SETNE SETNZ		ZF=0
SETNO		OF=0
SETNP SETPO		PF=0
SETNS		SF=0
SETO		OF=1
SETP SETPE		PF=1
SETS		SF=1

Команды работы со стеком

Команда	Операнды	Пояснение	Описание
PUSH	r(m)32 r(m)16 c32	ESP=ESP-4; SS:ESP=r(m)32/c SP=SP-2; SS:SP=r(m)16	Поместить операнд в вершину стека

POP	r(m)32
	r(m)16
		r(m)32=SS:ESP;
		ESP=ESP+4
		r(m)16=SS:SP;
		SP=SP+2;
			Извлечь операнд из вершины стека

PUSHA PUSHAD	r(m)32 r(m)16	—	Поместить в стек регистры EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
POPA POPAD	—		Извлечь из стека содержимое и заполнить регистры EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
PUSHF	—		Поместить в вершину стека регистр EFLAGS

Команда	Операнды	Пояснение	Описание
POPF	—		Извлечь содержимое вершины стека и заполнить регистр EFLAGS

Команды ввода-вывода

Микропроцессор может передавать данные в порты ввода-вывода, которые поддерживаются аппаратно и используют соответствующие своим предназначениям линии ввода-вывода процессора. Аппаратное адресное пространство ввода-вывода процессора не является физическим адресным пространством памяти. Адресное пространство ввода-вывода состоит из 64Кбайт индивидуально адресуемых 8-битных портов ввода-вывода, имеющих адреса 0…FFFFh. Адреса 0F8h…0FFh являются резервными. Любые два последовательных 8-битных порта могут быть объединены в 16-битный порт, 4 последовательных 8-битных порта – в 32-битный порт.

Команда	Операнды	Пояснение	Описание
IN	AL,c8 AX,c8 EAX,c8 AL,DX AX,DX EAX,DX	AL= port byte AX= port word EAX= port dword AL= [DX-port] AX= [DX-port] EAX= [DX-port]	Ввод из порта
OUT	c8, AL c8, AX c8, EAX DX, AL DX, AX DX, EAX	port byte=AL port word=AX port dword=EAX [DX-port]=AL [DX-port]=AX [DX-port]=EAX	Вывод в порт

Команда	Операнды	Пояснение	Описание
INSB INSW INSD	—	ES:(E)DI = [DX-port]	Вводит данные из порта, адресуемого DX в ячейку памяти ES:[(E)DI]. После ввода 1, 2 или 4-байтного слова данных EDI/DI корректируется на 1,2,4. При наличии префикса REP процесс продолжается, пока ECX>0
OUTSB OUTSW OUTSD	—	[DX-port]=DS:(E)SI	Выводит данные из ячейки памяти, определяемой регистрами DS:[(E)SI], в порт, адрес которого находится в DX. После вывода данных производится коррекция указателя ESI/SI на 1,2 или 4

Команды целочисленной арифметики

Команда	Операнды	Пояснение	Описание
ADD	r(m)8,c8 r(m)16,c16 r(m)32,c32 r(m)8,r8 r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32	r(m)8=r(m)8+c8 r(m)16=r(m)16+c16 r(m)32=r(m)32+c32 r(m)8=r(m)8+r8 r(m)16=r(m)16+r16 r(m)32=r(m)32+r32 r8=r8+r(m)8 r16=r16+r(m)16 r32=r32+r(m)32	Сложение целых чисел
ADC			Сложение целых чисел с учетом флага переноса CF
INC	r(m)8 r(m)16 r(m)32	r/m8=r/m8±1 r(m)16=r(m)16±1 r(m)32=r(m)32±1	Увеличение на 1
DEC			Уменьшение на 1
SUB	r(m)8,c8 r(m)16,c16 r(m)32,c32 r(m)8,r8	r(m)8=r(m)8-c8 r(m)16=r(m)16-c16 r(m)32=r(m)32-c32 r(m)8=r(m)8-r8	Вычитание целых чисел

Команда	Операнды	Пояснение	Описание
SBB	r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32	r(m)16=r(m)16-r16 r(m)32=r(m)32-r32 r8=r8-r(m)8 r16=r16-r(m)16 r32=r32-r(m)32	Вычитание с учетом флага переноса CF
CMP	r(m)8,c8 r(m)16,c16 r(m)32,c32 r(m)8,r8 r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32	r(m)8-c8 r(m)16-c16 r(m)32-c32 r(m)8-r8 r(m)16-r16 r(m)32-r32 r8-r(m)8 r16-r(m)16 r32-r(m)32	Сравнение целых чисел По результату сравнения устанавливаются флаги CF PF AF ZF SF OF
NEG	r(m)8 r(m)16 r(m)32	r(m)8=-r(m)8 r(m)16=-r(m)16 r(m)32=-r(m)32	Изменение знака числа
MUL	r(m)8 r(m)16 r(m)32	AX=AL*r(m)8 DX:AX=AX*r(m)16 EDX:EAX=EAX*r(m)32	Умножение без знака
IMUL	r(m)8 r(m)16 r(m)32 r16,r(m)16 r32,r(m)32 r16,r(m)16,c r32,r(m)32,c r16,c r32,c	AX=AL*r(m)8 DX:AX=AX*r(m)16 EDX:EAX=EAX*r(m)32 r16=r16*r(m)16 r32=r32*r(m)32 r16=r(m)16*c16 r32=r(m)32*c32 r16=r16*c16 r32=r32*c32	Умножение со знаком
DIV	r(m)8 r(m)16 r(m)32	AL=AX/r(m)8, AH=mod AX=DX:AX/r(m)16, DX=mod EAX=EDX:EAX/r(m)32, EDX=mod	Деление без знака
IDIV			Деление со знаком

Особого внимания среди рассмотренных команд целочисленной арифметики заслуживает команда CMP, которая вычитает второй операнд из первого и не сохраняет результат, а устанавливает биты OF, SF, ZF, AF, PF, CF регистра признаков EFLAGS в соответствии с результатом. Команда CMP чаще всего предшествует командам знакового или беззнакового условных переходов.

Логические команды

Выполнение логических операций описано здесь

Команда	Операнды	Пояснение	Описание
AND	r(m)8,c8 r(m)16,c16 r(m)32,c32 r(m)8,r8 r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32	r(m)8=r(m)8 Ф c8 r(m)16=r(m)16 Ф c16 r(m)32=r(m)32 Ф c32 r(m)8=r(m)8 Ф r8 r(m)16=r(m)16 Ф r16 r(m)32=r(m)32 Ф r32 r8=r8 Ф r(m)8 r16=r16 Ф r(m)16 r32=r32 Ф r(m)32	Логическое умножение (И), конъюнкция
OR			Логическое сложение (ИЛИ), дизъюнкция
XOR			Исключающее ИЛИ
NOT	r(m)8 r(m)16 r(m)32	r(m)8=~r(m)8 r(m)16=~r(m)16 r(m)32=~r(m)32	Логическое отрицание (НЕ), инверсия
TEST	r(m)8,c8 r(m)16,c16 r(m)32,c32 r(m)8,r8 r(m)16,r16 r(m)32,r32 r8,r(m)8 r16,r(m)16 r32,r(m)32	r(m)8 & c8 r(m)16 & c16 r(m)32 & c32 r(m)8 & r8 r(m)16 & r16 r(m)32 & r32 r8 & r(m)8 r16 & r(m)16 r32 & r(m)32	Логическое умножение без сохранения результата. В соответствии с результатом устанавливаются флаги PF ZF SF

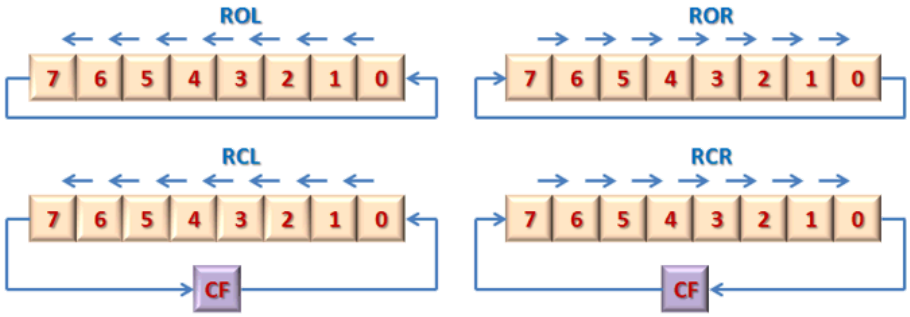
Сдвиговые команды

Выполнение сдвиговых операций в языке Си рассмотрено здесь.

Команда	Операнды	Пояснение	Описание
SHR	r(m)8 r(m)8,CL r(m)8,c r(m)16 r(m)16,CL r(m)16,c r(m)32 r(m)32,CL r(m)32,c	r(m)8 на 1 раздяд r(m)8 на CL разрядов r(m)8 на c разрядов r(m)16 на 1 разряд r(m)16 на CL разрядов r(m)16 на c разрядов r(m)32 на 1 разряд r(m)32 на CL разрядов r(m)32 на c разрядов	Логический сдвиг вправо
SAR			Арифметический сдвиг вправо (старшие разряды заполняются значением знакового)
SHL SAL			Логический (арифметический) сдвиг влево
ROR			Циклический сдвиг вправо
ROL			Циклический сдвиг влево
RCR			Циклический сдвиг вправо через перенос

Команда	Операнды	Пояснение	Описание
RCL			Циклический сдвиг влево через перенос

Команды циклического сдвига выполняются в соответствии со схемой



Команды коррекции двично-десятичных чисел

Команды коррекции двоично-десятичных чисел не имеют операндов и используют операнд по умолчанию, хранящийся в регистре AX (паре регистров AH:AL).

Команда	Пояснение	Описание

AAA	`if((AL&0Fh)>9 || AF) {`
	`  AH=AH+1;`
	`  AL=(AL+6) & 0Fh;`
	`  CF:=1; AF:=1;}`
		Коррекция AX после сложения двух неупакованных двоично-десятичных чисел

AAS	`if((AL&0Fh)>9)|| AF) {`
	`  AH=AH-1;`
	`  AL=(AL-6)& 0Fh;`
	`  CF=1; AF=1;}`
		Коррекция AX после вычитания двух неупакованных двоично-десятичных чисел

AAM	`AH=AL/10;` `AL=AL%10;`	Коррекция AX после умножения двух неупакованных двоично-десятичных чисел

Команда	Пояснение	Описание
AAD	AL=AH*10+AL; AH=0;	Коррекция AX перед делением двух неупакованных двоично-десятичных чисел

DAA	old_AL = AL;
	old_CF = CF;
	if(((AL & 0x0F)>9) || AF==1) {
	  AL = AL + 6;
	  CF = old_CF | CF;
	  AF = 1;}
	else
	  AF = 0;
	if((old_AL > 99h) || (CF==1)) {
	  AL = AL + 60h;
	  CF = 1;}
	else
	  CF = 0;
		Коррекция AL после сложения двух упакованных двоично-десятичных чисел

DAS	old_AL = AL;
	old_CF = CF;
	if(((AL & 0x0F)>9) || AF==1) {
	  AL = AL – 6;
	  CF = old_CF | CF;
	  AF = 1;}
	else
	  AF = 0;
	if((old_AL > 99h) || (CF==1)) {
	  AL = AL + 60h;
	  CF = 1;}
	else
	  CF = 0;
		Коррекция AL после вычитания двух упакованных двоично-десятичных чисел

Команды преобразования типов

Команды преобразования типов предназначены для корректного изменения размера операнда, заданного неявно в регистре-аккумуляторе (EAX, AX, AL). Непосредственно после аббревиатуры команды операнд не указывается.

Команда	Пояснение	Описание
CBW	AX=(DW)AL	2 байта ← 1 байт
CWDE	EAX=(DD)AX	4 байта ← 2 байта
CWD	DX:AX=(DD)AX	4 байта ← 2 байта
CDQ	EDX:EAX=(DQ)EAX	8 байт ← 4 байта

Команды управления флагами

Команды управления флагами предназначены для сброса или установки соответствующего бита регистра признаков EFLAGS. Команды управления флагами не имеют операндов.

Команда	Пояснение	Описание
CLC	CF = 0	Сброс бита переноса
CLD	DF=0	Сброс бита направления
CMC	CF=!CF	Инверсия бита переноса
STC	CF=1	Установка бита переноса
STD	DF=1	Установка бита направления
STI	IF=1	Установка бита прерывания

Команды прерываний

Команды прерываний предназначены для управления программными прерываниями.

Прерывание – это, как правило, асинхронная остановка работы процессора, вызванная началом работы устройства ввода-вывода. Исключением являются синхронные прерывания, возникающие при определении некоторых предопределенных условий в процессе выполнения команды.

Когда поступает сигнал о прерывании, процессор останавливает выполнение текущей программы и переключается на выполнение обработчика прерывания, заранее записанного для каждого прерывания.

Архитектура IA-32 поддерживает 17 векторов аппаратных прерываний и 224 пользовательских.

Команда INT вызывает обработчик указанного операндом прерывания (константой). Операнд определяет номер вектора системного прерывания BIOS от 0 до 255, представленный в виде беззнакового 8-битного целого числа. При вызове

обработчика прерывания в стеке сохраняются регистры EIP, CS и EFLAGS.
Прерывание по переполнению вызывается отдельной командой INTO и имеет вектор 04h.

Команда	Пояснение	Описание
INT c	EIP → стек CS → стек EFLAGS → стек переход к вектору c	Программное прерывание
INTO	OF=1	Прерывание по переполнению
IRET	EFLAGS ← стек CS ← стек EIP ← стек возврат	Возврат из обработчика прерывания

Команды передачи управления

Команда	Операнды	Пояснение	Описание
JMP	метка r(m)16 r(m)32	метка адрес в r(m)16 адрес в r(m)32	Безусловный переход на адрес, указанный операндом

Команды обращения к процедуре (функции)

Команда	Операнды	Пояснение	Описание
CALL	метка r(m)16 r(m)32	метка адрес в r(m)16 адрес в r(m)32	Вызов процедуры, указанной операндом
RET	— c16	— Удаляет из стека c16 байт	Возврат из процедуры

Команды поддержки языков высокого уровня

Команда	Операнды	Пояснение	Описание

Команда	Операнды	Пояснение	Описание
ENTER	c16, c8	PUSH EBP MOV EBP, ESP	Подготовка стека при входе в процедуру. Константа c16 указывает количество байт, резервируемых в стеке для локальных идентификаторов, константа c8 определяет вложенность процедуры
LEAVE	—	POP EBP	Приведение стека в исходное состояние
BOUND	r16, m16&16 r32, m32&32	m16<r16<m16&16 m32<r16<m32&32	Проверка индекса массива: сравнивает значение в регистре, заданном первым операндом с двумя значениями, расположенными последовательно в ячейке памяти, адресуемой вторым операндом.

Команды организации циклов — используют регистр ECX по умолчанию в качестве счетчика числа повторений цикла. Каждый раз при выполнении команды LOOPcc значение регистра ECX уменьшается на 1, а затем сравнивается с 0. Если ECX=0, выполнение цикла заканчивается, и продолжает выполняться код программы, записанный после команды LOOPcc. Если ECX содержит ненулевое значение, то осуществляется переход по адресу операнда команды LOOPcc.

Команда	Операнды	Пояснение	Описание
LOOP	метка	`ECX=ECX-1;` `if(CX>=0) EIP=метка;`	Переход если ECX>0

Команда	Операнды	Пояснение	Описание
LOOPE LOOPZ		ECX=ECX-1; if(ECX>0 && ZF==1) EIP=метка;	Переход если ECX>0 и ZF=1
LOOPNE LOOPNZ		ECX=ECX-1; if(ECX>0 && ZF==0) EIP=метка;	Переход если ECX>0 и ZF=0

Команды условных переходов — проверяют состояние одного или нескольких битов регистра признаков и при выполнении условия осуществляют передачу программного управления в другую точку кода, задаваемую операндом. Указанный класс команд не запоминает информацию для возврата. Операнд определяет адрес команды, которой должно быть передано управление.

Команда	Операнды	Пояснение	Описание
JCXZ	метка	if(ECX==0) EIP=метка;	Переход при ECX=0
JC		if(CF==1) EIP=метка;	Переход при переносе (CF=1)
JNC		if(CF==0) EIP=метка;	Переход при отсутствии переноса (CF=0)
JS		if(SF==1) EIP=метка;	Переход при отрицательном результате (SF=1)
JNS		if(SF==0) EIP=метка;	Переход при неотрицательном результате (SF=0)
JE JZ		if(ZF==1) EIP=метка;	Переход при нулевом результате (ZF=1)
JNE JNZ		if(ZF==0) EIP=метка;	Переход при ненулевом результате (ZF=0)
JP JPE		if(PF==1) EIP=метка;	Переход по четности (PF=1)

Команда	Операнды	Пояснение	Описание
JNP JPO		`if(PF==0)` `EIP=метка;`	Переход по нечетности (PF=0)
JO JNZ		`if(OF==1)` `EIP=метка;`	Переход при переполнении (OF=1)
JNO JNZ		`if(OF==0)` `EIP=метка;`	Переход при отсутствии переполнения (OF=0)

Беззнаковые переходы предназначены для сравнения беззнаковых величин и, как правило, используются непосредственно после команды сравнения CMP:

```
cmp m1, m2 ; сравнение m1 и m2, m1-m2
```

В аббревиатурах команд используются следующие обозначения:

- A (above) — выше;
- B (below) — ниже;
- E (equal) — равно.

Команда	Операнды	Пояснение	Описание

JB
JNAE
	метка	`cmp m1,m2`
		`if(CF==1)`
		`  EIP=метка;`
			Переход если ниже: m1<m2

JBE
JNA
		`cmp m1,m2`
		`if(CF==1 || ZF==1)`
		`  EIP=метка;`
			Переход если не выше: m1<=m2

JAE
JNB
		`cmp m1,m2`
		`if(CF==0)`
		`  EIP=метка;`
			Переход если не ниже: m1>=m2

JA
JNBE
		`cmp m1,m2`
		`if(CF==0 && ZF==0)`
		`  EIP=метка;`
			Переход если выше: m1>m2

Знаковые переходы предназначены для сравнения знаковых величин и, как правило, используются непосредственно после команды сравнения CMP:

```
cmp m1, m2 ; сравнение m1 и m2, m1-m2
```

В аббревиатурах команд используются следующие обозначения:

- L (less) — меньше;
- G (greater) — больше;
- E (equal) — равно.

Команда	Операнды	Пояснение	Описание

JL
JNGE
	метка	cmp m1,m2
		if(SF != OF)
		  EIP=метка;
			Переход если меньше: m1<m2

JLE
JNG
		cmp m1,m2
		if((SF != OF) || ZF==1)
		  EIP=метка;
			Переход если не больше: m1<=m2

JGE
JNL
		cmp m1,m2
		if(SF==OF)
		  EIP=метка;
			Переход если не меньше: m1>=m2

JG
JNLE
		cmp m1,m2
		if((SF==OF) && ZF==0)
		  EIP=метка;
			Переход если больше: m1>m2

Команды синхронизации работы процессора

Команда	Описание
HLT	Остановка процессора до внешнего прерывания
LOCK	Префикс блокировки шины. Заставляет процессор сформировать сигнал LOCK# на время выполнения находящейся за префиксом команды. Этот сигнал блокирует запросы шины другими процессорами в мультипроцессорной системе
WAIT	Ожидание завершения команды сопроцессора. Большинство команд сопроцессора автоматически вырабатывают эту команду
NOP	Пустая операция
CPUID	Получение информации о процессоре. Возвращаемое значение зависит от параметра в EAX

Команды побитового сканирования

Команда	Операнды	Описание
BSR	r16,r(m)16 r32,r(m)32	Ищет 1 в операнде 2, начиная со старшего бита. Если 1 найдена, ее индекс записывается в операнд 1

Команда	Операнды	Описание
BSF		Ищет 1 в операнде 2, начиная со младшего бита. Если 1 найдена, ее индекс записывается в операнд 1
BT	r(m)16,r16 r(m)32,r32 r(m)16,c8 r(m)32,c8	Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF.
BTC		Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF с инверсией.
BTR		Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF. Само значение бита сбрасывается в 0
BTS		Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF. Само значение бита устанавливается в 1

Строковые команды

Строковые команды предназначены для обработки цепочек данных. Операнды в строковых командах задаются по умолчанию. Как правило,

- операнд-источник адресуется регистром ESI внутри сегмента, на который указывает DS.
- операнд-источник адресуется регистром EDI внутри сегмента, на который указывает ES.

Команда	Операнды	Пояснение	Описание
MOVSB	1 байт	ES:EDI = DS:ESI	Копирование строки
MOVSW	2 байта		
MOVSD	4 байта		
LODSB	1 байт	AL = DS:ESI	Загрузка строки
LODSW	2 байта	AX = DS:ESI	
LODSD	4 байта	EAX = DS:ESI	
STOSB	1 байт	ES:EDI = AL	Сохранение строки
STOSW	2 байта	ES:EDI = AX	
STOSD	4 байта	ES:EDI = EAX	

Команда	Операнды	Пояснение	Описание
SCASB	1 байт	поиск AL в ES:EDI	Поиск данных в строке
SCASW	2 байта	поиск AX в ES:EDI	
SCASD	4 байта	поиск EAX в ES:EDI	
CMPSB	1 байт	поиск DS:ESI в ES:EDI	Поиск данных в строке
CMPSW	2 байта		
CMPSD	4 байта		

Перед выполнением строковых команд содержимое индексных регистров ESI, EDI должно быть проинициализировано. Сегментные регистры DS, ES должны указывать на соответствующие сегменты данных.

Для повторения выполнения строковых команд используются префиксы. Например, чтобы скопировать строку по байтам, используется команда

```
REP MOVSB
```

Здесь префикс REP сообщает процессору о том, что команда MOVSB должна повторяться. Максимальное количество повторений задается до вызова строковой команды в регистре ECX. Каждое выполнение строковой команды уменьшает содержимое регистра ECX на 1 и результат сравнивает с 0. В случае если ECX=0 выполнение повторяющейся строковой команды прекращается, и продолжается выполнение оставшейся части программы. Каждое повторение строковой команды также изменяет содержимое используемых индексных регистров (ESI, EDI) на размер операнда, заданный в команде (1, 2 или 4 байта). Направление модификации индексных регистров задается битом направления DF:

- DF=0: содержимое используемых индексных регистров увеличивается на размер операнда при каждом повторении.
- DF=1: содержимое используемых индексных регистров уменьшается на

размер операнда при каждом повторении.

Префиксы, используемые для повторения строковых команд, представлены в таблице

Префикс	Команды, с которыми используется	Описание
REP	MOVS* LODS* STOS*	Повторение
REPE REPZ	SCAS* CMPS*	Повторение пока операнды равны
REPNE REPNZ	SCAS* CMPS*	Повторение пока операнды не равны

Назад

Назад: *Язык ассемблера*

Добавить комментарий

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Комментарий

Имя *

E-mail *

Сайт

Я не робот

reCAPTCHA

Конфиденциальность - Условия использования

Отправить комментарий