Архитектура вычислительных систем

Лекция 9.

Яревский Е.А.

Кафедра вычислительной физики

Появление архитектуры x86 (IA-32)

1978:
Разработка 16-битового МП 8086/8088.
Фактически, не является архитектурой с регистрами общего назначения.
(первые IBM PC-XT).

1980:
Сопроцессор 8087 для работы с числами с плавающей точкой
(60 инструкций, **стековая архитектура**).

1985:
Представлен 32-разрядный 80386 (32-разр. регистры и память).
Добавлены новые режимы адресации, страничная адресация памяти.
Есть обратная совместимость.

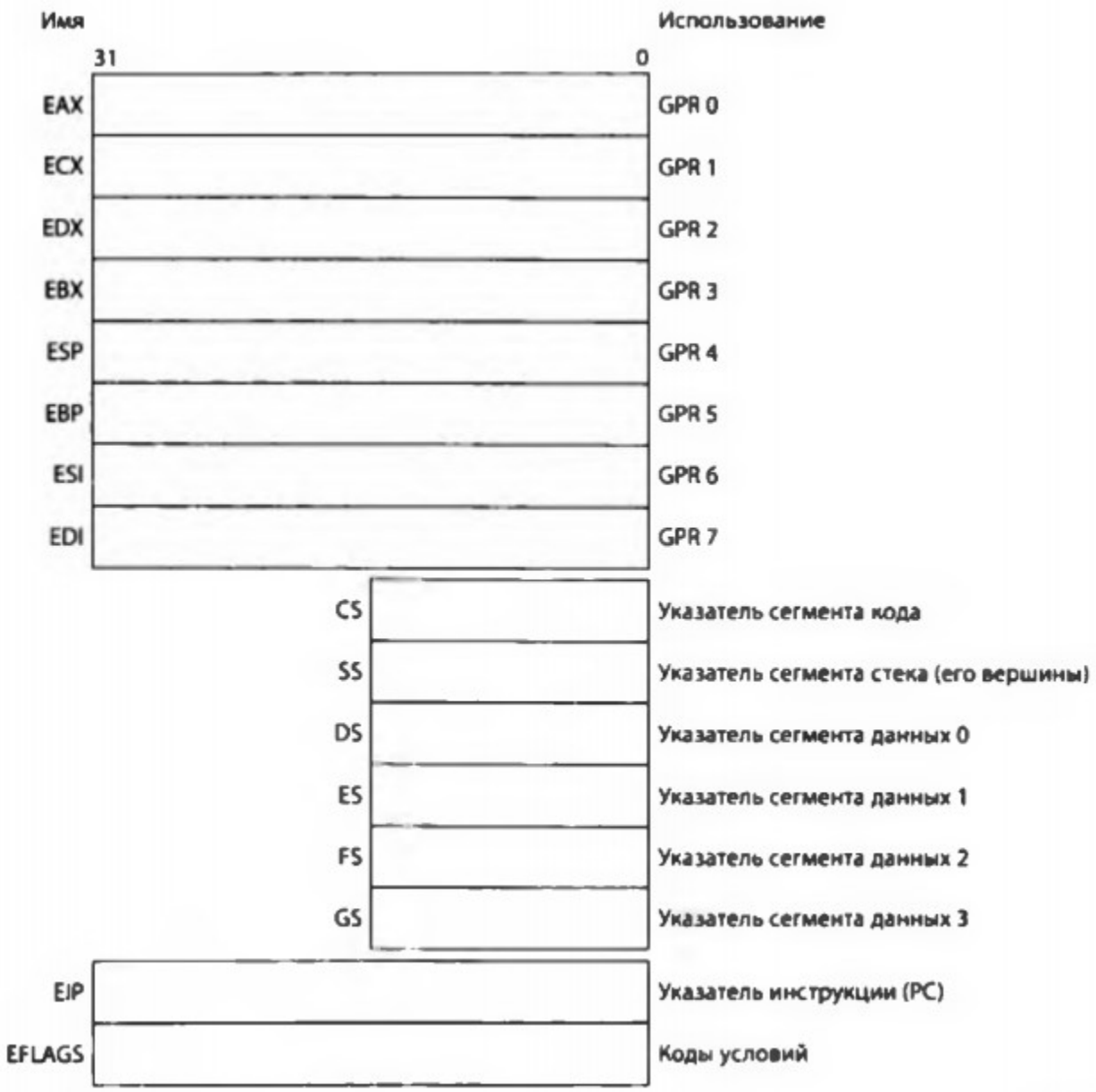
Процессорные архитектуры, совместимые с 80386 – **x86 архитектуры**.
Производители: **Intel**, **AMD** (VIA, Cyrix, …).

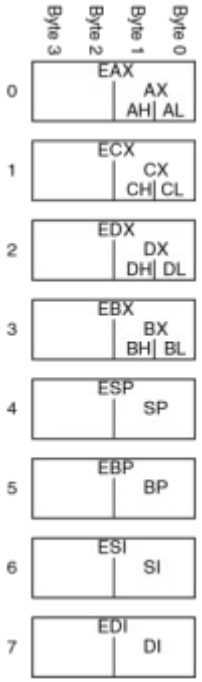
Основные отличия между MIPS и x86

Характеристики	MIPS	x86
Количество регистров	32, общего назначения	8, некоторые ограничения по использованию
Количество операндов	3 (2 источника, 1 назначение)	2 (1 источник, 1 источник/назначение)
Расположение операндов	Регистры или непосредственные операнды	Регистры, непосредственные операнды или память
Размер операнда	32 бита	8, 16 или 32 бита
Коды условий	Нет	Да
Типы команд	Простые	Простые и сложные
Размер команд	Фиксированный, 4 байта	Переменный, 1–15 байтов

MIPS – RISC-архитектура
X86 – CISC-архитектура

Регистры x86





Операнды x86

Табл. 6.10 Расположение операндов

Источник/ Назначение	Источник	Пример	Выполняемая функция
Регистр	Регистр	`add EAX, EBX`	`EAX <- EAX + EBX`
Регистр	Непосредственный операнд	`add EAX, 42`	`EAX <- EAX + 42`
Регистр	Память	`add EAX, [20]`	`EAX <- EAX + Mem[20]`
Память	Регистр	`add [20], EAX`	`Mem[20] <- Mem[20] + EAX`
Память	Непосредственный операнд	`add [20], 42`	`Mem[20] <- Mem[20] + 42`

Табл. 6.11 Режимы адресации памяти

Пример	Назначение	Комментарий
`add EAX, [20]`	`EAX <- EAX + Mem[20]`	Смещение (displacement)
`add EAX, [ESP]`	`EAX <- EAX + Mem[ESP]`	Базовая адресация
`add EAX, [EDX+40]`	`EAX <- EAX + Mem[EDX+40]`	Базовая адресация + смещение
`add EAX, [60+EDI*4]`	`EAX <- EAX + Mem[60+EDI*4]`	Смещение + масштабируемый индекс
`add EAX, [EDX+80+EDI*2]`	`EAX <- EAX + Mem[EDX+80+EDI*2]`	Базовая адресация + смещение + масштабируемый индекс

Архитектура x86 имеет 32-битное пространство памяти с побайтовой адресацией. x86 поддерживает много различных режимов адресации памяти. Расположение ячейки памяти задается при помощи комбинации регистра базового адреса, смещения и регистра масштабируемого индекса.

Смещение может иметь 8-, 16- или 32-битное значение.
Регистр масштабируемого индекса может быть умножен на 1, 2, 4 или 8. Масштабируемый индекс обеспечивает простой способ доступа к массивам и структурам с 2-, 4- или 8-байтовыми элементами без необходимости использовать команды для явного расчета адресов.

Команды x86 могут использовать 8-, 16- или 32-битные данные.
(MIPS всегда оперирует с 32-битными словами данных).

Пример	Назначение	Размер операндов
`add AH, BL`	`AH <- AH + BL`	8 бит
`add AX, -1`	`AX <- AX + 0xFFFF`	16 бит
`add EAX, EDX`	`EAX <- EAX + EDX`	32 бита

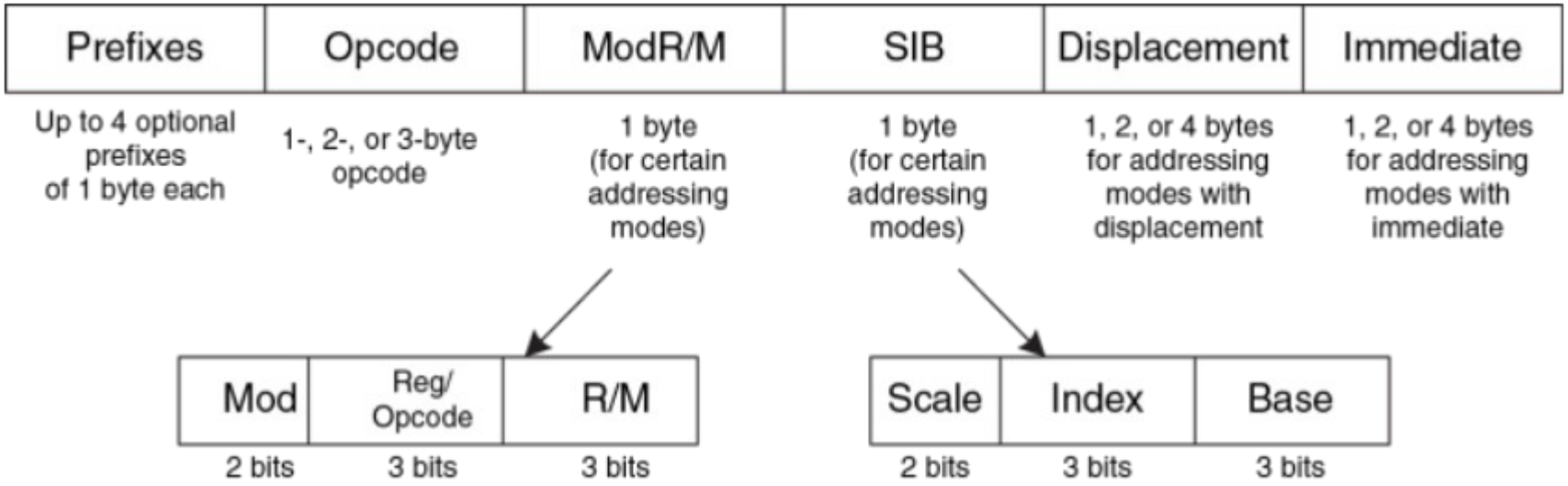
Флаги состояния

x86 использует флаги состояния (=коды условий) для ветвлений, отслеживания переносов и арифметических переполнений.
Используется 32-битный регистр EFLAGS, в котором хранятся флаги состояния.

CF (Carry Flag, флаг переноса) – результат вышел за пределы разрядной сетки.
ZF (Zero Flag, флаг нуля) – результат равен 0.
SF (Sign Flag, флаг знака) – результат отрицательный.
OF (Overflow Flag, флаг переполнения) – переполнение в дополнительном коде.

Остальные флаги используются ОС.

Длина команды x86 может составлять от 1 до 15 байтов.



Код операции (*opcode*): 1, 2 или 3 байта.
Поле *ModR/M* определяет режим адресации.
Поле *SIB* определяет коэффициент масштабирования (*scale*), индексный (*index*) и базовый (*base*) регистры в некоторых режимах адресации.
Поле *Displacement* содержит 1-, 2- или 4-байтовое смещение, используемое в соответствующих режимах адресации.
Поле *Immediate* содержит 1-, 2- или 4-байтовую константу для команд, использующих непосредственный операнд.

Поле ModR/M использует 2-битное поле режима Mod и 3-битное поле R/M для задания режима адресации одного из операндов.
Операнд – в одном из восьми регистров, или адресуется при помощи одного из 24 режимов адресации памяти.
В поле Reg – номер регистра, используемого в качестве второго операнда.

reg	w = 0	W =	= 1	r/m	mod =0		mod = l		mod = 2		mod = 3
		16b	32 b		16b	32b	16b	32b	16b	32b	
0	AL	AX	EAX	0	addr=BX+SI	=EAX	Тот же адрес, что и в mod=0 + dispS	Тот же адрес, что и в mod=0 + disp8	Тот же адрес, что и в mod=0 + displ6	Тот же адрес, что и в mod=0 + disp32	То же, что и в поле reg
1	CL	cx	ECX	1	addr=BX+DI	=ECX					
2	DL	DX	EDX	2	addr=BP+SI	=EDX					
3	BL	BX	EBX	3	addr=BP+SI	=EBX					
4	AH	SP	ESP	4	addr=SI	=(sb)	SI+disp8	(S/OHdispS	SI+disp8	(s/ty+disp32	"
5	CH	BP	EBP	5	addr=DI	=disp32	DI+disp8	EBP+disp8	DI+displ6	EBP+disp32	"
6	DH	SI	ESI	6	addr=displ6	=ESI	BP+d!sp8	ESI+disp8	BP+displ6	ESI+disp32	"
7	BH	DI	EDI	7	addr=BX	=EDI	BX+disp8	EDI+disp8	BX+displ6	EDI+disp32	"

Рис. 2.19. Кодирование первого спецификатора адреса x86: mod, reg, r/m. Первые четыре столбца показывают кодирование 3-битового поля reg, которое зависит от значения бита w кода операции (opcode) и от режима, в котором находится машина: 16-разрядного режима (8086) или 32-разрядного режима (80386). Остальные столбцы раскрывают значения полей mod и r/m. Значение 3-битового поля r/m зависит от значения 2-битового поля mod и размера адреса. В основном регистры, используемые в вычислениях адреса, перечислены в шестом и седьмом столбцах под общим заголовком mod = 0; в режиме mod = 1 к ним добавляется 8-разрядное смещение (displacement, сокращенно disp), а в режиме mod = 2 — 16-разрядное или 32-разрядное смещение, в зависимости от режима адресации. Исключения следующие: 1) при r/m = 6, когда mod = 1 или mod = 2, в 16-разрядном режиме выбирается BP плюс смещение; 2) при r/m = 5, когда mod = 1 или mod = 2, в 32-разрядном режиме выбирается EBP плюс смещение; и 3) при r/m = 4, в 32-разрядном режиме, когда значение mod ≠ 3, (sib) означает использование режима масштабируемого индекса, показанного в табл. 2.15. Когда mod = 3, поле r/m показывает регистр, используя то же самое кодирование, которое используется в поле reg в сочетании с битом w

В режимах адресации с регистром масштабируемого индекса, байт SIB определяет индексный регистр и коэффициент масштабирования (1, 2, 4 или 8). Если при адресации используются базовый адрес и индекс, то SIB также определяет регистр базового адреса.

x86 использует разное количество бит для определения разных команд. Часто используемые команды имеют меньший размер, что уменьшает среднюю длину команд в программе. Некоторые команды имеют несколько кодов операций.

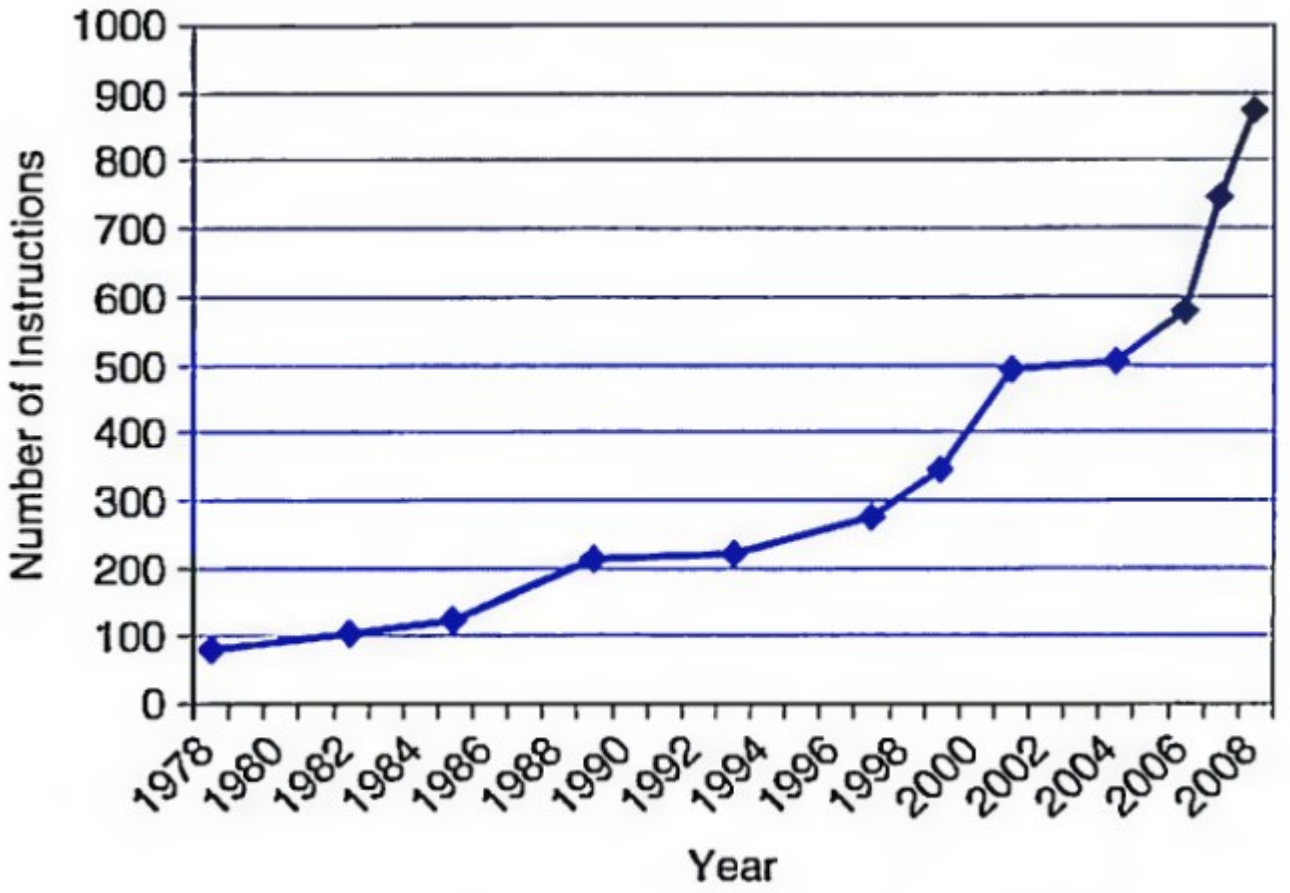
В **i8086** в коде операции указывалась разрядность операндов (8 или 16 бит). При добавлении в **i80386** 32-битных операндов свободных кодов уже не было. Команды, использующие 16- и 32-битные операнды, имеют одинаковые КОП. Чтобы различать их, используют дополнительный бит в дескрипторе сегмента кода, который устанавливается ОС и указывает процессору, какую команду он должен выполнить.
Для обратной совместимости с 8086, этот бит = 0, все операнды по умолчанию считаются 16-битными.
Когда этот бит = 1, используются 32-битные операнды.
При помощи префикса можно модифицировать отдельные команды:
если перед КОП добавить префикс 0x66, то будет использоваться альтернативный размер операндов (16 битов в 32-битном режиме и 32 бита в 16-битном режиме).

Начиная с **80286**, добавлен механизм сегментации для разделения памяти на сегменты размером до 64 Кбайт. Когда операционная система включала сегментацию, то все адреса вычислялись относительно начала сегмента. Процессор проверял адреса и при выходе за пределы сегмента формировал сигнал ошибки.
В современных ОС не используется.

x86 поддерживает команды, работающие с цепочками байтов или слов: копирование, сравнение и поиска определенного значения.
В современных процессорах такие команды, как правило, работают медленнее, чем последовательность простых команд :)

Префиксы, помимо 0x66, используются для захвата внешней шины (для обеспечения атомарного доступа к переменным в общей памяти в многопроцессорных системах), предсказания ветвлений, повторения команды при обработке цепочки байтов или слов.

Расширение набора инструкций



Рост количества инструкций x86 во времени.

В среднем за 30 лет добавлялась одна инструкция в месяц.

Расширение набора инструкций

1997: MMX (Multi Media Extensions). 57 SIMD команд, использующих стек для обработки коротких данных.
1999: SSE (Streaming SIMD Extensions). 70 команд, добавлены 8 128-бит регистров, 32-бит FP, управление кэшем.
2001: SSE2. 144 команды, в основном версии MMX и SSE для 64-бит FP.
2003: AMD64, общее расширение до регистров и памяти до 64 бит. Количество регистров увеличено до 16 (GPR и SSE). Режим старых команд и режим совместимости.
2004: Intel согласился с AMD64, Extended Memory 64 Technology (EM64T). Ввел SSE3 – добавлено 13 команд.
2006: SSE4, добавлено 54 команды.
2007: AMD, SSE5, добавлено 170 команд, в т.ч. трехадресные.
2008: Advanced Vector Extension, SSE регистры – 256 бит, переопределены 250 команд и добавлено 128.

…