x86

From Wikipedia, the free encyclopedia

x86	
Designer	Intel, later updated by AMD
Bits	16-bit, 32-bit, and/or 64-bit
Introduced	1978
Design	CISC
Type	Register-Memory
Encoding	Variable (1 to 15 bytes)
Branching	Status register
Endianness	Little
Page size	**8086–i286**: None **i386, i486**: 4 KB pages **P5 Pentium**: added 4 MB pages (Legacy **PAE**: 4 KB→2 MB) **x86-64**: added 1 GB pages.
Extensions	x87, IA-32, P6, MMX, SSE, SSE2, x86-64, SSE3, SSSE3, SSE4, SSE5, AVX
Open	Uncertain. For some advanced features, x86 may require license from Intel; x86-64 may require an additional license from AMD. The 80486 processor has been on the market for over 20 years [1] and so cannot be subject to patent claims. This subset of the x86 architecture is therefore fully open.
	Registers
General purpose	**16-bit**: 6 semi-dedicated registers + BP and SP; **32-bit**: 6 GPRs + EBP and ESP; **64-bit**: 14 GPRs + RBP and RSP.
Floating point	**16-bit**: Optional separate x87 FPU. **32-bit**: Optional

The term **x86** refers to a family of instruction set architectures[2] based on the Intel 8086. The 8086 was launched in 1978 as a fully 16-bit extension of Intel's early 8-bit based microprocessors and also introduced segmentation to overcome the 16-bit addressing barrier of earlier chips. The term x86 derived from the fact that early successors to the 8086 also had names ending in "86". Many additions and extensions have been added to the x86 instruction set over the years, almost consistently with full backward compatibility.[3] The architecture has been implemented in processors from Intel, Cyrix, AMD, VIA, and many others.

The term is not synonymous with IBM PC compatibility as this implies a multitude of other hardware; embedded systems as well as computers used x86 chips before the PC-compatible market started,[4] some of them before the IBM PC itself.

As the term became common *after* the introduction of the 80386, it usually implies binary compatibility with the 32-bit instruction set of the 80386. This may sometimes be emphasized as x86-32 to distinguish it either from the original 16-bit "x86-16" or from the 64-bit x86-64.[5] Although most x86 processors used in *new* personal computers and servers have 64-bit capabilities, to avoid compatibility problems with older computers or systems, the term *x86-64* (or *x64*) is often used to denote 64-bit software, with the term *x86* implying only 32-bit.[6][7]

Although the 8086 was primarily developed for embedded systems and small single-user computers, largely as a response to the successful 8080-compatible Zilog Z80[8], the x86 line soon grew in features and processing power. Today, x86 is ubiquitous in both stationary and portable personal computers and has replaced midrange computers and RISC-based processors in a majority of servers and workstations as well. A large amount of software, including OSs such as MS-DOS, Windows, Linux, BSD, Solaris, and Mac OS X supports x86-based hardware.

Modern x86 is relatively uncommon in embedded systems however, and small low power applications (using tiny batteries) and low-cost microprocessor markets, such as home appliances and toys, lack any significant x86 presence.[9] Simpler 16-bit x86 chips are more common here, although VIA C7, VIA Nano, AMD's Geode and Intel Atom are examples of 32- and 64-bit designs used in parts of these segments.

There have been several attempts, also within Intel itself, to break the market dominance of the "inelegant" x86 architecture that descended directly from the first simple 8-bit microprocessors. Examples of this are the iAPX 432 (alias Intel 8800), the Intel 960, Intel 860 and Intel and Hewlett Packard Itanium architecture. However, the continuous refinement of x86 microarchitectures, circuitry, and semiconductor manufacturing would prove it hard to replace x86 in many segments. AMD's 64 bit extension of x86 (which Intel eventually responded to with a compatible design)[10] and the scalability of x86 chips such as the eight-core Intel Xeon and 12-core AMD Opteron is underlining x86 as an example of how continuous refinement of established industry standards can resist the competition from completely new architectures.[11]

separate or integrated x87 FPU, integrated SSE2 units in later processors.
64-bit: Integrated x87 and SSE2 units.

The Intel 8086.

Intel Core 2 Duo - an example of an x86-compatible, 64-bit multicore processor.

AMD Athlon (early version) - another technically different, but fully compatible, x86 implementation.

Contents

- 1 Chronology
- 2 History
  - 2.1 Background
    - 2.1.1 iAPX 432 and the 80286
  - 2.2 Other manufacturers
  - 2.3 Extensions of word size
- 3 Overview
  - 3.1 Basic properties of the architecture
    - 3.1.1 Floating point and SIMD
  - 3.2 Current implementations

- 4 Segmentation
- 5 Addressing modes
- 6 x86 registers
  - 6.1 16-bit
  - 6.2 32-bit
  - 6.3 64-bit
  - 6.4 Miscellaneous/special purpose
  - 6.5 Purpose
  - 6.6 Structure
- 7 Operating modes
  - 7.1 Real mode
  - 7.2 Protected mode
    - 7.2.1 Virtual 8086 mode
  - 7.3 Long mode
- 8 Extensions
  - 8.1 Floating point unit
  - 8.2 MMX
  - 8.3 3DNow!
  - 8.4 SSE
  - 8.5 Physical Address Extension (PAE)
  - 8.6 x64
  - 8.7 Virtualization
- 9 See also
- 10 Notes and references
- 11 External links

Chronology

The table below lists brands of common[12] consumer targeted processors implementing the x86 instruction set, grouped by generations that highlight important points in x86 history. Note: CPU generations are not strict: each generation is roughly marked by significantly improved or commercially successful processor microarchitecture designs.

Generation	First introduced	Prominent Consumer CPU brands	linear / physical address space	Notable (new) features
1	1978	Intel 8086, Intel 8088	**16-bit** / 20-bit (*segmented*)	first x86 microprocessors
2	1982	Intel 80186, Intel 80188, NEC V20/V30		hardware for fast address calculations, fast mul/div etc.
		Intel 80286	**16-bit** (30-bit *virtual*) / 24-bit (*segmented*)	MMU, for protected mode and a larger address space
3 (IA-32)	1985	Intel386, AMD Am386	**32-bit** (46-bit *virtual*) / 32-bit	32-bit instruction set, MMU with paging
4	1989	Intel486, AMD Am486, Cyrix III-Samuel, VIA C3-Samuel2 / VIA C3-Ezra (2001)		risc-like pipelining, integrated FPU, on-chip cache
5	1993	Pentium, Pentium MMX		superscalar, 64-bit databus, faster FPU, MMX
5/6	1996	Cyrix 6x86, Cyrix MII, Cyrix III-Joshua (2000)		register renaming, speculative execution
6	1995	Pentium Pro, AMD K5, Nx586 (1994), Rise mP6	*as above* / **36**-bit *physical* (PAE)	µ-op translation, PAE (Pentium Pro), integrated L2 cache (Pentium Pro), conditional move instructions
	1997	AMD K6/-2/3, Pentium II/III, IDT/Centaur-C6		L3-cache support, 3DNow, SSE
7	1999	Athlon, Athlon XP		superscalar FPU, wide design (*up to three x86 instr./clock*)

	2000	Pentium 4		deeply pipelined, high frequency, SSE2, hyper-threading
6-M/7-M	2003	Pentium M, VIA C7 (2005), Intel Core (2006)		optimized for low power
8 (x86-64)		Athlon 64, Opteron	**64-bit** / *40-bit physical in first AMD implementation.*	x86-64 instruction set, on-die memory controller, hypertransport
	2004	Pentium 4 Prescott		very deeply pipelined, very high frequency, SSE3, 64-bit capability is available only in LGA 775 sockets
9	2006	Intel Core 2		low power, multi-core, lower clock frequency, SSE4 (Penryn)
10	2007	AMD Phenom	*as above / 48-bit physical for AMD Phenom*	monolithic quad-core, 128 bit FPUs, SSE4a, HyperTransport 3 or QuickPath, native memory controller, on-die L3 cache, modular design
	2008	Intel Core i3, Intel Core i5, Intel Core i7		
		Intel Atom		In-order but highly pipelined, very-low-power
		VIA Nano		Out-of-order, superscalar, hardware-based encryption, very low power, adaptive power management
11	2010	Intel Sandy Bridge, AMD Bulldozer		SSE5/AVX, highly modular design

History

Background

The x86 architecture first appeared as the Intel 8086 CPU released in 1978, a fully 16-bit design based on the earlier 8-bit based 8008, 8080 and 8085. Although not binary compatible, it was designed to allow assembly language programs written for these processors to be mechanically translated into the equivalent 8086 assembly. This made the new processor a tempting software migration path, but not without significant hardware redesign, largely due to the 16-bit external databus. To address this, Intel introduced the almost identical 8088, with an 8-bit external databus, which permitted simpler printed circuit boards and demanded fewer (1-bit wide) DRAM chips. The 8088 could also be interfaced to already established (i.e. low-cost) 8-bit system and peripheral chips more easily. Among other, non-technical factors, this contributed to the fact that IBM built their IBM PC around the 8088, despite a presence of 16-bit microprocessors from Motorola, Zilog, and National Semiconductor. The IBM PC subsequently took over from Z80-based CP/M systems, Apple IIs, and other popular computers, and became a dominant de-facto standard for personal computers, thus enabling the 8088 and its successors to dominate this large branch of the microprocessor market.

iAPX 432 and the 80286

Another factor was that the advanced but non-compatible 32-bit Intel 8800 (alias iAPX 432) failed in the marketplace around the time the original IBM-PC was launched; the new and fast 80286 actually contributed to the disappointment in the performance of the semi-contemporary 8800 in early 1982. (The 80186, launched in parallel with the 80286, was intended for embedded systems, and would therefore have had a large market anyway.) The market failure of the 32-bit 8800 was a significant impetus for Intel to continue to develop more advanced 8086-compatible processors instead, such as the 80386 (a 32-bit extension of the well performing 80286).

Other manufacturers

Further information: List of former IA32 compatible processor manufacturers

At various times, companies such as IBM, NEC[13], AMD, TI, STM, Fujitsu, OKI, Siemens, Cyrix, Intersil, C&T, NexGen, and UMC started to design and/or manufacture x86 processors intended for personal computers as well as embedded systems. Such x86 implementations are seldom plain copies but often employ different internal microarchitectures as well as different solutions at the electronic and physical levels. Quite naturally, early compatible chips were 16-bit, while 32-bit designs appeared much later. For the personal computer market, real quantities started to appear around 1990 with i386 and i486 compatible processors, often named similarly to Intel's original chips. Other companies, which designed or manufactured x86 or x87 processors, include ITT Corporation, National Semiconductor, ULSI System Technology, and Weitek.

Following the fully pipelined i486, Intel introduced the Pentium brand name (which, unlike numbers, could be trademarked) for their new line of superscalar x86 designs. With the x86 naming scheme now legally cleared, IBM partnered with Cyrix to produce the 5x86 and then the very efficient 6x86 (M1) and 6x86MX (MII) lines of Cyrix designs, which were the first x86 chips implementing register renaming to enable speculative execution. AMD meanwhile designed and manufactured the advanced but delayed 5k86 (K5), which, *internally*, was heavily based on AMD's earlier 29K RISC design; similar to NexGen's Nx586, it used a strategy where dedicated pipeline stages decode x86 instructions into uniform and easily handled micro-operations, a method that has remained the basis for most x86 designs to this day.

Some early versions of these chips had heat dissipation problems. The 6x86 was also affected by a few minor compatibility issues, the Nx586 lacked an FPU and (the then crucial) pin-compatibility, while the K5 had somewhat disappointing performance when it was (eventually) launched. A low customer awareness of alternatives to the Pentium line further contributed to these designs being comparatively unsuccessful, despite the fact that the K5 had very good Pentium compatibility and the 6x86 was significantly faster than the Pentium on integer code.[14] AMD later managed to establish itself as a serious contender with the K6 line of processors, which gave way to the highly successful Athlon and Opteron. There were also other contenders, such as Centaur Technology (formerly IDT), Rise Technology, and Transmeta. VIA Technologies' energy efficient C3 and C7 processors, which were designed by Centaur, have been sold for many years. Centaur's newest design, the VIA Nano, is their first processor with superscalar and speculative execution. It was, perhaps interestingly, introduced at about the same time as Intel's first "in-order" processor since the P5 Pentium, the Intel Atom.

Extensions of word size

The instruction set architecture has twice been extended to a larger word size. In 1985, Intel released the 32-bit 80386 (or i386) which gradually replaced the earlier 16-bit chips in computers (although typically not in embedded systems) during the following years; this extended programming model was originally referred to as *the i386 architecture* (like its first implementation) but Intel later dubbed it IA-32 when introducing its (unrelated) IA-64 architecture. In 1999-2003, AMD extended this 32-bit architecture to 64 bits and referred to it as x86-64 in early documents and later as AMD64. Intel soon adopted AMD's architectural extensions under the name IA-32e which was later renamed EM64T and finally Intel 64. Among these five names, the original x86-64 is probably the most commonly used, although Microsoft and Sun Microsystems also use the term x64.

Overview

Basic properties of the architecture

The x86 architecture is a variable instruction length, primarily two-address "CISC" design with emphasis on backward compatibility. The instruction set is not typical CISC however, but basically an extended and orthogonalized version of the simple eight-bit 8008, 8080, and 8085 architectures. Byte-addressing is supported and words are stored in memory with little-endian byte order. Memory access to unaligned addresses is allowed for all supported word sizes. The largest native size for integer *arithmetics* and memory addresses (or offsets) is 16, 32, or 64 bits depending on architecture generation (newer processors include direct support for smaller integers as well). Multiple scalar values can be handled simultaneously via the SIMD unit present in later generations, as described below.[15] Immediate addressing offsets and immediate data may be expressed as 8-bit quantities for the frequently occurring cases or contexts where a -128..127 range is enough. Typical instructions are therefore 2 or 3 bytes in length (although some are much longer, and some are single-byte).

To further conserve encoding space, most registers are expressed in opcodes using three bits, and at most one operand to an instruction can be a memory location (some highly orthogonal "CISC" designs, such as the PDP-11, may use two). However, this memory operand may also be the *destination* (or a combined *source* **and** *destination*), while the other operand, the *source*, can be either *register* or *immediate*. Among other factors, this contributes to a code footprint that rivals eight-bit machines and enables efficient use of instruction cache memory. The relatively small number of general registers (also inherited from its 8-bit ancestors) has made register-relative addressing (using small immediate offsets) an important method of accessing operands, especially on the stack. Much work has therefore been invested in making such accesses as fast as register accesses, i.e. a one cycle instruction throughput, in most circumstances where the accessed data is available in the top-level cache.

Floating point and SIMD

A dedicated floating point processor with 80-bit internal registers, the 8087, was developed for the original 8086. This chip subsequently developed into the extended 80387, and later processors incorporated a backward compatible version of this functionality on the same chip as the main processor. In addition to this, modern x86 designs also contain a SIMD-unit (see SSE below) where instructions can work in parallel on (one or two) 128-bit words, each containing 2 or 4 floating point numbers (each 64 or 32 bits wide), or, alternatively, 2,4,8 or 16 integers (each 64,32,16 or 8 bits wide). The wide SIMD registers means that existing x86 processors can load or store up to 128 bits of memory data in a single instruction and also perform bitwise operations (although not integer arithmetics[16]) on full 128-bits quantities in parallel. Planned x86 processors will have 256-bit SIMD operations (including 256-bit memory load and store).

Current implementations

During execution, current x86 processors employ a few extra decoding steps to split most instructions into smaller pieces (micro-operations). These are then handed to a control unit that buffers and schedules them in compliance with x86-semantics so that they can be executed, partly in parallel, by one of several (more or less specialized) execution units. These modern x86 designs are thus superscalar, and also capable of out of order and speculative execution (via register renaming), which means they may execute multiple (partial or complete) x86 instructions simultaneously, and not necessarily in the same order as given in the instruction stream.

When introduced, this approach was sometimes referred to as a "RISC core" or as "RISC translation", partly for marketing reasons, but also because these micro-operations share some properties with certain types of RISC instructions. However, *traditional* microcode (used since the 1950s) also inherently shares many of the same properties; the new approach differs mainly in that the translation to micro-operations now occurs asynchronously. Not having to synchronize the execution units with the decode steps opens up possibilities for more analysis of the (buffered) code stream, and therefore permits detection of operations that can be performed in parallel, simultaneously feeding more than one execution unit.

The latest processors also do the opposite when appropriate; they combine certain x86 sequences (such as a compare followed by a conditional jump) into a more complex micro-op which fits the execution model better and thus can be executed faster or with less machine resources involved.

Another way to try to improve performance is to cache the decoded micro-operations, so the processor can directly access the decoded micro-operations from a special cache, instead of decoding them again. The Execution Trace Cache found in Intel's NetBurst Microarchitecture (Pentium 4) is so far the only widespread example of this technique.

Transmeta use a completely different method in their x86 compatible CPUs. They use just-in-time translation to convert x86 instructions to the CPU's native instructions. Transmeta argues that their approach allows for more power efficient designs since the CPU can forgo the complicated decode step of more traditional x86 implementations.

Segmentation

Further information: x86 memory segmentation

Minicomputers during the late 1970s were running up against the 16-bit 64-KB address limit, as memory had become cheaper. Some minicomputers like the PDP-11 used complex bank-switching schemes, or, in the case of Digital's VAX, redesigned much more expensive processors which could directly handle 32-bit addressing and data. The original 8086, developed from the simple 8085 microprocessor and primarily aiming at very small and inexpensive computers and other specialized devices, instead adopted simple segment registers which increased the memory address width by only 4 bits. By multiplying a 64-KB address by 16, the 20-bit address could address a total of one megabyte (1,048,576 bytes) which was quite a large amount for a small computer at the time. The concept of segment registers was not new to many mainframes which used segment register to quickly swap to different tasks. In practice, on the x86 it was (is) a much-criticized implementation which greatly complicated many common programming tasks and compilers. But as it also simplified hardware design and cost, it would be cost-competitive in its dominant market segments. With the emergence of standards such as the IBM-PC, programming development costs could be spread over the sales of many copies of software, and the architecture would eventually evolve to full 32 and even 64-bit memory addressing by the 21st century.

Data and/or code could be managed within "near" 16-bit segments within this 1 MB address space, or a compiler could operate in a "far" mode using 32-bit `segment:offset` pairs reaching (only) 1 MB. While that would also prove to be quite limiting by the mid-1980s, it was working for the emerging PC market, and made it very simple to translate software from the older 8080, 8085, and Z80 to the newer processor. In 1985, the 16-bit segment addressing model was effectively factored out by the introduction of 32-bit offset registers, in the 386 design.

In real mode, segmentation is achieved by shifting the segment address left by 4 bits and adding an offset in order to receive a final 20-bit address. For example, if DS is A000h and SI is 5677h, DS:SI will point at the absolute address DS × 10h + SI = A5677h. Thus the total address space in real mode is 2^{20} bytes, or 1 MB, quite an impressive figure for 1978. All memory addresses consist of both a segment and offset; every type of access (code, data, or stack) has a default segment register associated with it (for data the register is usually DS, for code it is CS, and for stack it is SS). For data accesses, the segment register can be explicitly specified (using a segment override prefix) to use any of the four segment registers.

In this scheme, two different segment/offset pairs can point at a single absolute location. Thus, if DS is A111h and SI is 4567h, DS:SI will point at the same A5677h as above. This scheme makes it impossible to use more than four segments at once. CS and SS are vital for the correct functioning of the program, so that only DS and ES can be used to point to data segments outside the program (or, more precisely, outside the currently-executing segment of the program) or the stack.

In protected mode, a segment register no longer contains the physical address of the beginning of a segment, but contain a "selector" that points to a system-level structure called a *segment descriptor*. A segment descriptor contains the physical address of the beginning of the segment, the length of the segment, and access permissions to that segment. The offset is

checked against the length of the segment, with offsets referring to locations outside the segment causing an exception. Offsets referring to locations inside the segment are combined with the physical address of the beginning of the segment to get the physical address corresponding to that offset.

The segmented nature can make programming and compiler design difficult because the use of near and far pointers affects performance.

Addressing modes

Addressing modes for 16-bit x86 processors can be summarized by this formula:

$$\begin{Bmatrix} CS: \\ DS: \\ SS: \\ ES: \end{Bmatrix} \left[\begin{Bmatrix} BX \\ BP \end{Bmatrix} \right] + \left[\begin{Bmatrix} SI \\ DI \end{Bmatrix} \right] + [\text{displacement}]$$

Addressing modes for 32-bit address size on 32-bit or 64-bit x86 processors can be summarized by this formula:

$$\begin{Bmatrix} CS: \\ DS: \\ SS: \\ ES: \\ FS: \\ GS: \end{Bmatrix} \left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ ESP \\ EBP \\ ESI \\ EDI \end{Bmatrix} \right] + \left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ EBP \\ ESI \\ EDI \end{Bmatrix} * \begin{Bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{Bmatrix} \right] + [\text{displacement}]$$

Addressing modes for 64-bit code on 64-bit x86 processors can be summarized by these formulas:

$$\begin{Bmatrix} : \\ FS: \\ GS: \end{Bmatrix} [\text{general register}] + \left[\text{general register} * \begin{Bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{Bmatrix} \right] + [\text{displacement}]$$

and

$$RIP + [\text{displacement}]$$

The 8086 had 64 KB of 8-bit (or alternatively 32 K-word of 16-bit) I/O space, and a 64 KB (one segment) stack in memory supported by hardware. Only words (2 bytes) can be pushed to the stack. The stack grows downwards (toward numerically lower addresses), its bottom being pointed by SS:SP. There are 256 interrupts, which can be invoked by both hardware and software. The interrupts can cascade, using the stack to store the return address.

x86 registers

For a description of the general notion of a CPU register, see Processor register.

16-bit

The original Intel 8086 and 8088 have fourteen 16-bit registers. Four of them (AX, BX, CX, DX) are general-purpose registers (GPRs; although each may have an additional purpose: for example only CX can be used as a counter with the *loop* instruction). Each can be accessed as two separate bytes (thus BX's high byte can be accessed as BH and low byte as BL). There are two pointer registers. SP points to the top of the stack and BP (base pointer) which is used to point at some other

place in the stack, typically above the local variables. Two registers (SI and DI) are for array indexing.

Four segment registers (CS, DS, SS and ES) are used to form a memory address. The FLAGS register contains flags such as carry flag, overflow flag and zero flag. Finally, the instruction pointer (IP) points to the next instruction that will be fetched from memory and then executed.

In the Intel 80286, three special registers hold descriptor table addresses (GDTR, LDTR, IDTR), and a fourth task register (TR).

32-bit

With the advent of the 32-bit 80386 processor, the 16-bit general-purpose registers, base registers, index registers, instruction pointer, and FLAGS register, but not the segment registers, were expanded to 32 bits. This is represented by prefixing an "**E**" (for **Extended**) to the register names in x86 assembly language. Thus, the AX register corresponds to the lowest 16 bits of the new 32-bit EAX register, SI corresponds to the lowest 16 bits of ESI, and so on. The general-purpose registers, base registers, and index registers can all be used as the base in addressing modes, and all of those registers except for the stack pointer can be used as the index in addressing modes.

Two new segment registers (FS and GS) were added. With a greater number of registers, instructions and operands, the machine code format was expanded. To provide backward compatibility, segments with executable code can be marked as containing either 16-bit or 32-bit instructions. Special prefixes allow inclusion of 32-bit instructions in a 16-bit segment or vice versa.

With the 80486 a floating-point processing unit (FPU) was added, with eight 80-bit wide registers.[17]

With the Pentium II, eight 64-bit MMX integer registers were added (MMX0..MMX7, which share lower bits with the 80-bit-wide FPU stack (st(0)..st(7))).[17] With the Pentium III, a 32-bit Streaming SIMD Extension (SSE) control/status register (MXCSR) and eight 128-bit SSE floating point registers (XMM0..XMM7) were added.[17]

64-bit

Starting with the AMD Opteron processor, the x86 architecture extended the 32-bit registers into 64-bit registers in a way similar to how the 16 to 32-bit protected mode extension was done (RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP, RFLAGS, RIP), and eight additional 64-bit general registers (R8, R9..R15) were also introduced in the creation of x86-64. However, these extensions are only usable in 64-bit mode, which is one of the two modes only available in long mode. The addressing modes were not dramatically changed from 32-bit mode, except that addressing was extended to 64 bits, physical addressing was now sign extended (so memory always added equally to the top and bottom of memory; note that this does not affect linear or virtual addressing), and other selector details were dramatically reduced.

Miscellaneous/special purpose

x86 processors (starting with the 80386) also include various special/miscellaneous registers such as control registers (CR0 through 4, CR8 for 64-bit only), debug registers (DR0 through 3, plus 6 and 7), test registers (TR3 through 7; 80486 only), descriptor registers (GDTR, LDTR, IDTR), a task register (TR), and model-specific registers (MSRs, appearing with the Pentium).

Purpose

Although the main registers (with the exception of the instruction pointer) are "general-purpose" and can be used for anything, it was envisioned that they be used for the following purposes:

- AX/EAX/RAX: accumulator
- BX/EBX/RBX: base index (ex: arrays)
- CX/ECX/RCX: counter
- DX/EDX/RDX: data/general
- SI/ESI/RSI: "source index" for string operations.
- DI/EDI/RDI: "destination index" for string operations.
- SP/ESP/RSP: stack pointer for top address of the stack.
- BP/EBP/RBP: stack base pointer for holding the address of the current stack frame.
- IP/EIP/RIP: instruction pointer. Holds the program counter, the current instruction address.

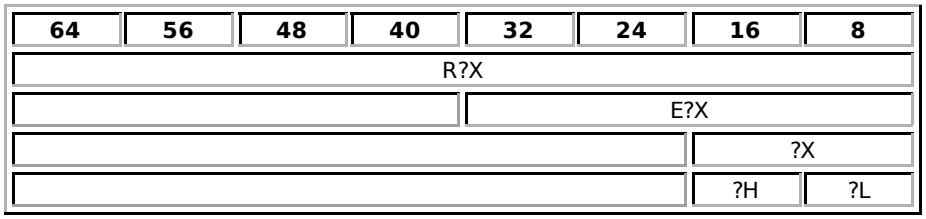
No particular purposes were envisioned for the other 8 registers available only in 64-bit mode.

Some instructions compiled and executed more efficiently when using these registers for their designed purpose. For example, using AL as an accumulator and adding an immediate byte value to it produces the efficient *add to AL* opcode of 04h, whilst using the BL register produces the generic and longer *add to register* opcode of 80C3h. Another example is double precision division and multiplication that works specifically with the AX and DX registers.

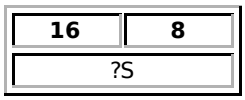
Modern compilers benefited from the introduction of the *sib* byte ("scaled index byte") that allows registers to be treated uniformly (minicomputer-like). Some special instructions lost priority in the hardware design and became slower than equivalent small code sequences. A notable example is the LODSW instruction.

Structure

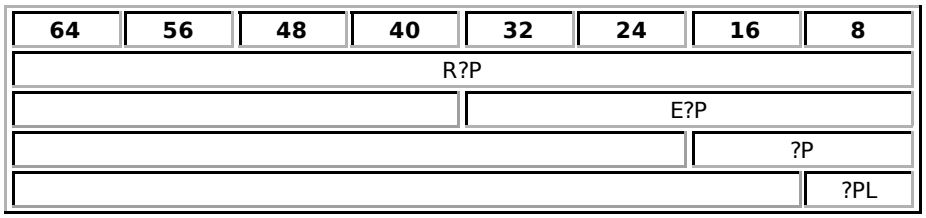
General Purpose Registers (A, B, C and D)



Segment Registers (C, D, S, E, F, and G)

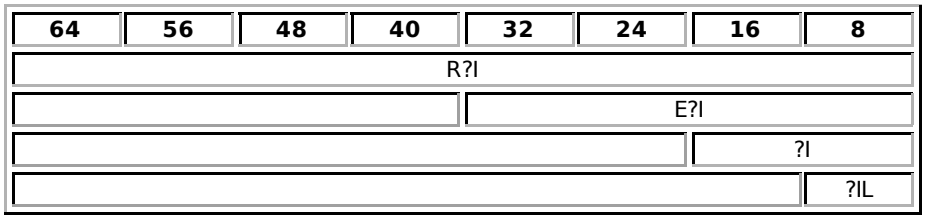


Pointer Registers (S and B)



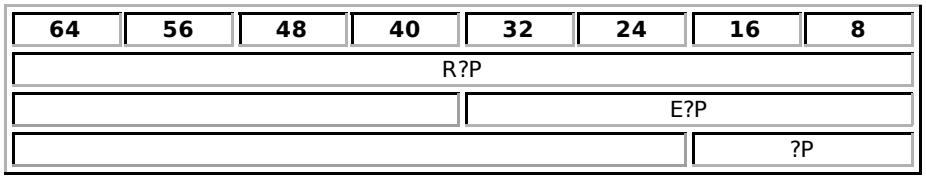
Note: The ?PL registers are only available in 64-bit mode.

Index Registers (S and D)

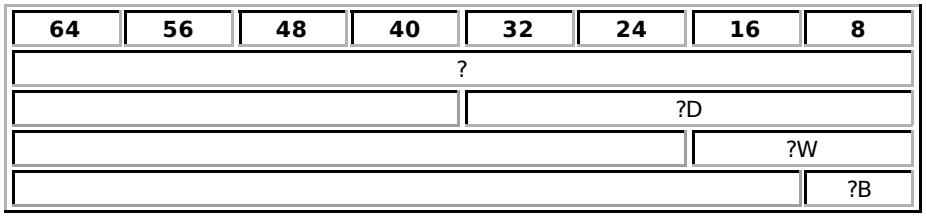


Note: The ?IL registers are only available in 64-bit mode.

Instruction Pointer Register (I)



64-bit mode-only General Purpose Registers (R8, R9, R10, R11, R12, R13, R14, R15)



Operating modes

Real mode

Main article: Real mode

Real mode is an operating mode of 8086 and later x86-compatible CPUs. Real mode is characterized by a 20 bit segmented memory address space (meaning that only 1 MB of memory can be addressed), direct software access to BIOS routines and peripheral hardware, and no concept of memory protection or multitasking at the hardware level. All x86 CPUs in the 80286 series and later start up in real mode at power-on; 80186 CPUs and earlier had only one operational mode, which is equivalent to real mode in later chips.

In order to use more than 64 KB of memory, the segment registers must be used. This created great complications for C compiler implementors who introduced odd pointer modes such as "near", "far" and "huge" to lever the implicit nature of segmented architecture to different degrees, with some pointers containing 16-bit offsets within implied segments and other pointers containing segment addresses and offsets within segments.

Protected mode

Main article: Protected mode

In addition to real mode, the Intel 80286 supports protected mode, expanding addressable physical memory to 16 MB and addressable virtual memory to 1 GB, and providing protected memory, which prevents programs from corrupting one another. This is done by using the segment registers only for storing an index to a segment table. There were two such tables, the Global Descriptor Table (GDT) and the Local Descriptor Table (LDT), each holding up to 8192 segment descriptors, each segment giving access to 64 KB of memory. The segment table provided a 24-bit base address, which can be added to the desired offset to create an absolute address. Each segment can be assigned one of four ring levels used for hardware-based computer security.

The Intel 80386 introduced support in protected mode for paging, a mechanism making it possible to use virtual memory.

Paging is used extensively by modern multitasking operating systems. Linux, 386BSD and Windows NT were developed for the 386 because it was the first Intel architecture CPU to support paging and 32-bit segment offsets. The 386 architecture became the basis of all further development in the x86 series.

x86 processors that support protected mode boot into real mode for backward compatibility with the older 8086 class of processors. Upon power-on (a.k.a. booting), the processor initializes in real mode, and then begins executing machine code stored in ROM. An operating system boot sequence may place the processor into the protected mode which enables paging and other features. The instruction set in protected mode is backward compatible with the one used in real mode.

Virtual 8086 mode

Further information: Virtual 8086 mode

There is also a sub-mode of operation in 32-bit protected mode, called *virtual 8086 mode*. This is basically a special hybrid operating mode that allows real mode programs and operating systems to run while under the control of a protected mode supervisor operating system. This allows for a great deal of flexibility in running both protected mode programs and real mode programs simultaneously. This mode is exclusively available for the 32-bit version of protected mode; virtual 8086 mode does not exist in the 16-bit version of protected mode, or in the 64-bit long mode.

Long mode

Main article: Long mode

By 2002, it was obvious that the 32-bit address space of the x86 architecture was limiting its performance in applications requiring large data sets. A 32-bit address space would allow the processor to directly address only 4 GB of data, a size surpassed by applications such as video processing and database engines, while using the 64-bit address, one can directly address 16777216 TB (or 16 billion GB) of data, although most 64-bit architectures don't support access to the full 64-bit address space (AMD64, for example, supports only 48 bits, split into 4 paging levels, from a 64-bit address).

AMD developed the extension of the 32-bit x86 architecture to 64-bit that is currently used in x86 processors, initially calling it *x86-64*, later renaming it *AMD64*. The Opteron, Athlon 64, Turion 64, and later Sempron families of processors use this architecture. The success of the AMD64 line of processors coupled with the lukewarm reception of the IA-64 architecture forced Intel to release its own implementation of the AMD64 instruction set. Intel had previously implemented support for AMD64[18] but opted not to enable it in hopes that AMD would not bring AMD64 to market before Itanium's new IA-64 instruction set was widely adopted. It branded its implementation of AMD64 as *EM64T*, and later re-branded it *Intel 64*.

In its literature and product version names, Microsoft and Sun refer to AMD64/Intel 64 collectively as *x64* in the Windows and Solaris operating systems respectively. Linux distributions refer to it either as "x86-64", its variant "x86_64", or "amd64". BSD systems use "amd64" while Mac OS X uses "x86_64".

Long mode is mostly an extension of the 32-bit instruction set, but unlike the 16–to–32-bit transition, many instructions were dropped in the 64 bit mode. This does not affect actual binary backward compatibility (which would execute legacy code in other modes that retain support for those instructions), but it changes the way assembler and compilers for new code have to work.

This was the first time that a *major* upgrade of the x86 architecture was initiated and originated by a manufacturer other than Intel. It was also the first time that Intel accepted technology of this nature from an outside source.

Extensions

Floating point unit

Main article: x87

Further information: Floating point unit

Early x86 processors could be extended with floating-point hardware in the form of a series of floating point numerical co-processors with names like 8087, 80287 and 80387. With very few exceptions, the 80486 and subsequent x86 processors then integrated this x87 functionality on chip which made the x87 instructions a de facto integral part of the x86 instruction set.

Each x87 register, known as ST(0) through ST(7), is 80 bits wide and stores numbers in the IEEE floating-point standard double extended precision format. These registers are organized as a stack with ST(0) as the top. This was done in order to conserve opcode space, and the registers are therefore randomly accessible only for either operand in a register-to-register arithmetic instruction; ST0 must always be one of the two operands, either the source or the destination, regardless of whether the other operand is ST(x) or a memory operand.

MMX

Main article: MMX (instruction set)

MMX is a SIMD instruction set designed by Intel, introduced in 1997 for Pentium MMX microprocessors. It developed out of a similar unit first used on the Intel i860. It first appeared in the Pentium MMX. It is supported on most subsequent IA-32 processors by Intel and other vendors. MMX is typically used for video processing (in "multimedia" applications for instance).

MMX added 8 new "registers" to the architecture, known as MM0 through MM7 (henceforth referred to as *MMn*). In reality, these new "registers" were just aliases for the existing x87 FPU stack registers. Hence, anything that was done to the floating point stack would also affect the MMX registers. Unlike the FP stack, these MMn registers were fixed, not relative, and therefore they were randomly accessible. The instruction set did not adopt the stack-like semantics so that existing operating systems could still correctly save and restore the register state when multitasking without modifications.

Each of the MMn registers are 64-bit integers. However, one of the main concepts of the MMX instruction set is the concept of *packed data types*, which means instead of using the whole register for a single 64-bit integer (quadword); two 32-bit integers (doubleword), four 16-bit integers (word) or eight 8-bit integers (byte) may be used. Also because the MMX's 64-bit MMn registers are aliased to the FPU stack, and each of the stack registers are 80-bit wide, the upper 16-bits of the stack registers go unused in MMX, and these bits are set to all ones, which makes it look like NaNs or infinities in the floating point view. This makes it easier to tell whether you are working on floating point data or MMX data.

3DNow!

Main article: 3DNow!

In 1997 AMD introduced 3DNow! The introduction of this technology coincided with the rise of 3D entertainment applications and was designed to improve the CPU's vector processing performance of graphic-intensive applications. 3D video game developers and 3D graphics hardware vendors use 3DNow! to enhance their performance on AMD's K6 and Athlon series of processors.

3DNow! was designed to be the natural evolution of MMX from integers to floating point. As such, it uses the exact same register naming convention as MMX, that is MM0 through MM7. The only difference is that instead of packing byte to quadword integers into these registers, one would pack single precision floating points into these registers. The advantage of aliasing registers with the FPU registers is that the same instruction and data structures used to save the state of the FPU registers can also be used to save 3DNow! register states. Thus no special modifications are required to be made to operating systems which would otherwise not know about them.

SSE

Main articles: Streaming SIMD Extensions, SSE2, SSE3, SSSE3, SSE4, and SSE5

In 1999, Intel introduced the Streaming SIMD Extensions (SSE) instruction set, following in 2000 with SSE2. The first addition allowed offloading of basic floating-point operations from the x87 stack and the second made MMX almost obsolete and allowed the instructions to be realistically targeted by conventional compilers. Introduced in 2004 along with the *Prescott* revision of the Pentium 4 processor, SSE3 added specific memory and thread-handling instructions to boost the performance of Intel's HyperThreading technology. AMD licensed the SSE3 instruction set and implemented most of the SSE3 instructions for its revision E and later Athlon 64 processors. The Athlon 64 does not support HyperThreading and lacks those SSE3 instructions used only for HyperThreading.

SSE discarded all legacy connections to the FPU stack. This also meant that this instruction set discarded all legacy connections to previous generations of SIMD instruction sets like MMX. But it freed the designers up, allowing them to use larger registers, not limited by the size of the FPU registers. The designers created eight 128-bit registers, named XMM0 through XMM7. (*Note*: in AMD64, the number of SSE XMM registers has been increased from 8 to 16.) However, the downside was that operating systems had to have an awareness of this new set of instructions in order to be able to save their register states. So Intel created a slightly modified version of Protected mode, called Enhanced mode which enables the usage of SSE instructions, whereas they stay disabled in regular Protected mode. An OS that is aware of SSE will activate Enhanced mode, whereas an unaware OS will only enter into traditional Protected mode.

SSE is a SIMD instruction set that works only on floating point values, like 3DNow!. However, unlike 3DNow! it severs all legacy connection to the FPU stack. Because it has larger registers than 3DNow!, SSE can pack twice the number of single precision floats into its registers. The original SSE was limited to only single-precision numbers, like 3DNow!. The SSE2 introduced the capability to pack double precision numbers too, which 3DNow! had no possibility of doing since a double precision number is 64-bit in size which would be the full size of a single 3DNow! MMn register. At 128-bit, the SSE XMMn registers could pack two double precision floats into one register. Thus SSE2 is much more suitable for scientific calculations than either SSE1 or 3DNow!, which were limited to only single precision. SSE3 does not introduce any additional registers.

Physical Address Extension (PAE)

Main article: Physical Address Extension

By default, physical addresses are 32-bit; but there exists a page extension mode called Physical Address Extension or PAE, first added in the Intel Pentium Pro, which allows an additional 4 bits of physical addressing. The size of memory in Protected mode is usually limited to 4 GB. Through tricks in the processor's page and segment memory management systems, x86 operating systems may be able to access more than 32-bits of address space, even without the switchover to the 64-bit paradigm. This mode does not change the length of segment offsets or linear addresses; those are still only 32 bits.

x64

Main article: x86-64

See also: Itanium

In April 2003, AMD released the first x86 processor with 64-bit physical memory address registers capable of addressing much more than 4 GB of memory using the new x86-64 extension (also known as x64). Intel introduced its first x86-64 processor on July 2004.

x86-64 had been preceded by another architecture employing 64-bit memory addressing: Intel introduced Itanium in 2001 for the high-performance computing market. However, Itanium was incompatible with x86 and is less widely used today. x86-64 also introduced the NX bit, which offers some protection against security bugs caused by buffer overruns.

Virtualization

Main article: x86 virtualization

Until recently, the x86 architecture did not meet the Popek and Goldberg requirements - a specification for virtualization created in 1974 by Gerald J. Popek and Robert P. Goldberg. Nevertheless, there are several commercial x86 virtualization products, such as VMware vSphere, Parallels, Microsoft Hyper-V Server, and Microsoft Virtual PC. Among the open source virtualization projects, most notable are QEMU/KQEMU, VirtualBox, and Xen.

Intel and AMD have introduced x86 processors with hardware-based virtualization extensions that overcome the classical virtualization limitations of the x86 architecture. These extensions are known as Intel VT (code named "Vanderpool",) and AMD-V (code named "Pacifica".) Although most modern x86 processors include these extensions, the technology is generally considered immature at this point with most software-based virtualization outperforming these extensions.[19] This is expected to change as the technology matures.

See also

- Itanium
- PowerPC
- Input/Output Base Address
- Interrupt request
- x86 assembly language
- x86 instruction listings
- List of AMD microprocessors
- List of Intel microprocessors
- List of VIA microprocessors
- List of x86 manufacturers
- Microarchitecture
- Rosenblum, Mendel; and Garfinkel, Tal (May, 2005). "Virtual machine monitors: current technology and future trends" (http://ieeexplore.ieee.org/iel5/2/30853/01430630.pdf?tp=&isnumber=&arnumber=1430630) . *IEEE Computer, volume 38, issue 5* (http://ieeexplore.ieee.org/iel5/2/30853/01430630.pdf?tp=&isnumber=&arnumber=1430630) . http://ieeexplore.ieee.org/iel5/2/30853/01430630.pdf?tp=&isnumber=&arnumber=1430630.

Notes and references

1. ^ 80486 32-bit CPU breaks new ground in chip density and operating performance. (Intel Corp.) (product announcement) EDN | May 11, 1989 | Pryce, Dave
2. ^ Unlike the microarchitecture (and specific electronic and physical implementation) used for a specific chip design.
3. ^ Intel abandoned its "x86" naming scheme with the *P5 Pentium* in 1993 (as *numbers* could not be trademarked). However, the term x86 was already firmly established among technicians, compiler writers etc.
4. ^ the GRID Compass laptop, for instance
5. ^ Intel uses IA-32 and Intel 64 (formerly EM64T or IA-32e) for x86 and x86-64 respectively. Likewise, AMD today prefers AMD64 over the x86-64 name it once introduced.
6. ^ "Linux* Kernel Compiling" (http://www.intel.com/cd/ids/developer/asmo-na/eng/182333.htm?page=4) . Intel. http://www.intel.com/cd/ids/developer/asmo-na/eng/182333.htm?page=4. Retrieved 2007-09-04.
7. ^ "Intel Web page search result for "x64"" (http://mysearch.intel.com/corporate/default.aspx?culture=en-US&q=x64&searchsubmit.x=21&searchsubmit.y=11) . http://mysearch.intel.com/corporate/default.aspx?culture=en-US&q=x64&searchsubmit.x=21&searchsubmit.y=11. Retrieved 2007-09-04.
8. ^ Birth of a Standard: The Intel 8086 Microprocessor (http://www.pcworld.com/article/146957/birth_of_a_standard_the_intel_8086_microprocessor.html)
9. ^ The embedded processor market is populated by more than 25 different architectures, which, due to the price sensitivity, low power and hardware simplicity requirements, outnumber the x86.
10. ^ "Time and again, processor architects have looked at the inelegant x86 architecture and declared it cannot be stretched to accommodate the latest innovations," said Nathan Brookwood, principal analyst, Insight 64. (http://bwrc.eecs.berkeley.edu/CIC/announce/1999/k8.annc.html)
11. ^ Microsoft to End Intel Itanium Support (http://www.eweek.com/c/a/IT-Infrastructure/Microsoft-to-End-Intel-Itanium-Support-370349/)
12. ^ "Microprocessor Hall of Fame" (http://www.intel.com/museum/online/hist%5Fmicro/hof/) . Intel. http://www.intel.com/museum/online/hist%5Fmicro/hof/. Retrieved 2007-08-11.
13. ^ *The NEC V20 and V30 also provided the older 8080 instruction set, allowing PC equipped with these chips to run CP/M applications at full speed (i.e. without the need to simulate a 8080 in software).*
14. ^ It had a slower Floating point unit however, which is slightly ironic as Cyrix started out as a designer of fast Floating point units for x86 processors.
15. ^ *16-bit and 32-bit chips were introduced in 1978 and 1985 respectively; plans for 64-bit was announced in 1999 and gradually introduced from 2003 and onwards.*
16. ^ *That is because integer arithmetics generates carry between subsequent bits (unlike simple bitwise operations).*
17. ^ ***a b c*** Intel 64 and IA-32 Architectures Software Developer's Manual, Vol. 1 (http://www.intel.com/Assets/PDF/manual/253665.pdf) , p. 2-33 (2009)
18. ^ Intel's Yamhill Technology: x86-64 compatible | Geek.com (http://www.geek.com/intels-yamhill-technology-x86-64-compatible/)
19. ^ Adams, Keith; and Agesen, Ole (2006-21-2006). "A Comparison of Software and Hardware Techniques for x86 Virtualization" (http://www.vmware.com/pdf/asplos235_adams.pdf) . *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, 2006* (http://www.vmware.com/pdf/asplos235_adams.pdf) . ACM 1-59593-451-0/06/0010. http://www.vmware.com/pdf/asplos235_adams.pdf. Retrieved 2006-12-22.

External links

- 25 Years of Intel Architecture (http://www.intel.com/museum/corporatetimeline/index.htm?iid=about+ln_history)
- x86 CPUs guide (http://www.x86-guide.com/)

Retrieved from "http://en.wikipedia.org/wiki/X86"
Categories: 1978 introductions | Intel products | Instruction set architectures | IBM PC compatibles | X86 architecture