


Intel 8086

Intel 8086

	
Produced	From 1978 to 1990s
Common manufacturer(s)	<ul style="list-style-type: none">Intel, AMD, NEC, Fujitsu, Harris (Intersil), OKI, Siemens AG, Texas Instruments, Mitsubishi.
Max. CPU clock rate	5 MHz to 10 MHz
Min. feature size	3μm
Instruction set	x86-16
Predecessor	(8080)
Successor	80186
Package(s)	<ul style="list-style-type: none">40 pin DIP
Variant	8088

The **8086**^[1] (also called **iAPX86**) is a 16-bit microprocessor chip designed by Intel between early 1976 and mid-1978, when it was released. The 8086 gave rise to the x86 architecture of Intel's future processors. The Intel 8088, released in 1979, was a slightly modified chip with an external 8-bit data bus (allowing the use of cheaper and fewer supporting logic chips^[2]), and is notable as the processor used in the original IBM PC.

History

Background

In 1972, Intel launched the 8008, the first 8-bit microprocessor.^[3] It implemented an instruction set designed by Datapoint corporation with programmable CRT terminals in mind, that also proved to be fairly general purpose. The device needed several additional ICs to produce a functional computer, in part due to its small 18-pin "memory-package", which ruled out the use of a separate address bus (Intel was primarily a DRAM manufacturer at the time).

Two years later, in 1974, Intel launched the 8080,^[4] employing the new 40-pin DIL packages originally developed for calculator ICs to enable a separate address bus. It had an extended instruction set that was source- (not binary-) compatible with the 8008 and also included some 16-bit instructions to make programming easier. The 8080 device, often described as the first truly useful microprocessor, was eventually replaced by the depletion-load based 8085 (1977) which could cope with a single 5V power supply instead of the three different operating voltages of earlier chips.^[5] Other well known 8-bit microprocessors that emerged during these years were Motorola 6800 (1974), General Instrument PIC16X (1975), MOS Technology 6502 (1975), Zilog Z80 (1976), and Motorola 6809 (1978).

The first x86 design

The 8086 project started in May 1976 and was originally intended as a temporary substitute for the ambitious and delayed iAPX 432 project. It was an attempt to draw attention from the less-delayed 16 and 32-bit processors of other manufacturers (such as Motorola, Zilog, and National Semiconductor) and at the same time to counter the threat from the Zilog Z80 (designed by former Intel employees), which became very successful. Both the architecture and the physical chip were therefore developed rather quickly by a small group of people, and using the same basic microarchitecture elements and physical implementation techniques as employed for the slightly older 8085 (and for which the 8086 also would function as a continuation).

Marketed as source compatible, the 8086 was designed so that assembly language for the 8008, 8080, or 8085 could be automatically converted into equivalent (sub-optimal) 8086 source code, with little or no hand-editing. The programming model and instruction set was (loosely) based on the 8080 in order to make this possible. However, the 8086 design was expanded to support full 16-bit processing, instead of the fairly basic 16-bit capabilities of the 8080/8085.

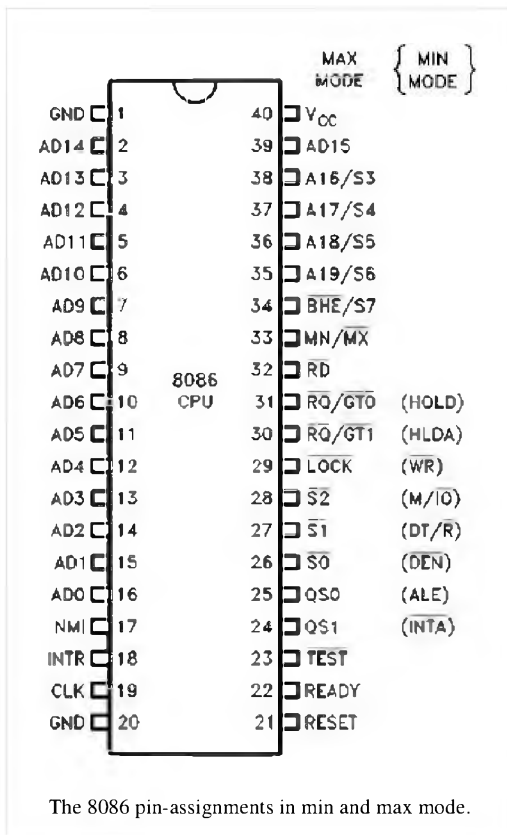
New kinds of instructions were added as well; full support for signed integers, base+offset addressing, and self-repeating operations were akin to the Z80 design^[6] but were all made slightly more general in the 8086. Instructions directly supporting nested ALGOL-family languages such as Pascal and PL/M were also added. According to principal architect Stephen P. Morse, this was a result of a more software centric approach than in the design of earlier Intel processors (the designers had experience working with compiler implementations). Other enhancements included microcoded multiply and divide instructions and a bus-structure better adapted to future co-processors (such as 8087 and 8089) and multiprocessor systems.

The first revision of the instruction set and high level architecture was ready after about three months,^[7] and as almost no CAD-tools were used, four engineers and 12 layout people were simultaneously working on the chip.^[8] The 8086 took a little more than two years from idea to working product, which was considered rather fast for a complex design in 1976–1978.

The 8086 was sequenced^[9] using a mixture of random logic^[10] and microcode and was implemented using depletion-load nMOS circuitry with approximately 20,000 active transistors (29,000 counting all ROM and PLA sites). It was soon moved to a new refined nMOS manufacturing process called HMOS (for High performance MOS) that Intel originally developed for manufacturing of fast static RAM products.^[11] This was followed by HMOS-II, HMOS-III versions, and, eventually, a fully static CMOS version for battery-powered devices, manufactured using Intel's CHMOS processes.^[12] The original chip measured 33 mm² and minimum feature size was 3.2 μm.

The architecture was defined by Stephen P. Morse with some help and assistance by Bruce Ravenel (the architect of the 8087) in refining the final revisions. Logic designer Jim McKeivitt and John Bayliss were the lead engineers of the hardware-level development team^[13] and William Pohlman the manager for the project. The legacy of the 8086 is enduring in the basic instruction set of today's personal computers and servers; the 8086 also lent its last two digits to later extended versions of the design, such as the Intel 286 and the Intel 386, all of which eventually became known as the x86 family. (Another reference is that the PCI Vendor ID for Intel devices is 8086_h.)

Details



Main registers																
AH	AL	AX (primary accumulator)														
BH	BL	BX (base, accumulator)														
CH	CL	CX (counter, accumulator)														
DH	DL	DX (accumulator, other functions)														
Index registers																
	SI	Source Index														
	DI	Destination Index														
	BP	Base Pointer														
	SP	Stack Pointer														
Status register																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(bit position)
-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C	Flags
Segment register																
	CS	Code Segment														
	DS	Data Segment														
	ES	ExtraSegment														
	SS	Stack Segment														
Instruction pointer																
	IP	Instruction Pointer														
<i>The 8086 registers</i>																

Buses and operation

All internal registers, as well as internal and external data buses, were 16 bits wide, firmly establishing the "16-bit microprocessor" identity of the 8086. A 20-bit external address bus gave a 1 MB physical address space ($2^{20} = 1,048,576$). This address space was addressed by means of internal 'segmentation'. The data bus was multiplexed with the address bus in order to fit a standard 40-pin dual in-line package. 16-bit I/O addresses meant 64 KB of separate I/O space ($2^{16} = 65,536$). The maximum **linear** address space was limited to 64 KB, simply because internal registers were only 16 bits wide. Programming over 64 KB boundaries involved adjusting segment registers (see below) and was therefore fairly awkward (and remained so until the 80386).

Some of the control pins, which carry essential signals for all external operations, had more than one function depending upon whether the device was operated in *min* or *max* mode. The former was intended for small single processor systems whilst the latter was for medium or large systems, using more than one processor.

Registers and instructions

The 8086 had eight (more or less general) 16-bit registers including the stack pointer, but excluding the instruction pointer, flag register and segment registers. Four of them, AX, BX, CX, DX, could also be accessed as twice as many 8-bit registers (see figure) while the other four, BP, SI, DI, SP, were 16-bit only.

Due to a compact encoding inspired by 8-bit processors, most instructions were one-address or two-address operations which means that the result were stored in one of the operands. At most one of the operands could be in memory, but this memory operand could also be the *destination*, while the other operand, the *source*, could be either *register* or *immediate*. A single memory location could also often be used as both *source* and *destination* which, among other factors, further contributed to a code density comparable to (and often better than) most eight bit machines.

Although the degree of generality of most registers were much greater than in the 8080 or 8085, it was still fairly low compared to the typical contemporary minicomputer, and registers were also sometimes used implicitly by instructions. While perfectly sensible for the assembly programmer, this complicated register allocation for compilers compared to more regular 16- and 32-bit processors such as the PDP-11, VAX, 68000, etc.; on the other hand, compared to semi-contemporary simple (but popular and ubiquitous) 8-bit microprocessors such as the 6502, 6809, or 8085, it was significantly easier to generate code for the 8086 design.

The 8086 also featured 64 KB of 8-bit (or alternatively 32 K-word of 16-bit) I/O space. A 64 KB (one segment) stack growing towards lower addresses is supported by computer hardware; 2-byte words are pushed to the stack and the stack top (bottom) is pointed out 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 addresses.

The processor had some new instructions (not present in the 8080 and 8085) to better support stack based high level programming languages such as Pascal and PL/M; some of the more useful ones were **push mem-op**, and **ret size**, supporting the "pascal calling convention" directly. (Several others, such as **push immed** and **enter**, would be added in the subsequent 80186, 80286, and 80386 designs.) It also had a stack-marker mechanism. There are three control flags IF(Intrrupt Flag)TF(Trap Flag)DF(Direction flag).

Flags

8086 has a 16-bit flag register. Out of these, 9 are active, and indicate the current state of the processor. These are — Carry flag, Parity flag, Auxiliary flag, Zero flag, Sign flag, Trap flag, Interrupt flag, Direction flag and Overflow flag.

Segmentation

There were also four 16-bit segment registers (see figure) that allowed the 8086 CPU to access one megabyte of memory in an unusual way. Rather than concatenating the segment register with the address register, as in most processors whose address space exceeded their register size, the 8086 shifted the 16-bit segment only four bits left before adding it to the 16-bit offset ($16 \times \text{segment} + \text{offset}$), therefore producing a 20-bit external (or effective or physical) address from the 32-bit segment:offset pair. As a result, each external address could be referred to by $2^{12} = 4096$ different segment:offset pairs. The 16-byte separation between segment bases (due to the 4-bit shift) was called a *paragraph*. Although considered complicated and cumbersome by many programmers, this scheme also had advantages; a small program (less than 64 KB) could be loaded starting at a fixed offset (such as 0) in its own segment, avoiding the need for relocation, with at most 15 bytes of alignment waste.

Compilers for the 8086-family commonly supported two types of pointer, *near* and *far*. Near pointers were 16-bit offsets implicitly associated with the program's code and/or data segment and so could be used only within parts of a program small enough to fit in one segment. Far pointers were 32-bit segment:offset pairs resolving to 20-bit external addresses. Some compilers also supported *huge* pointers, which were like far pointers except that pointer

arithmetic on a huge pointer treated it as a linear 20-bit pointer, while pointer arithmetic on a far pointer wrapped around within its 16-bit offset without touching the segment part of the address.

To avoid the need to specify *near* and *far* on numerous pointers, data structures, and functions, compilers also supported "memory models" which specified default pointer sizes. The *tiny* (max 64K), *small* (max 128K), *compact* (data > 64K), *medium* (code > 64K), *large* (code,data > 64K), and *huge* (individual arrays > 64K) models covered practical combinations of near, far, and huge pointers for code and data. The *tiny* model meant that code and data was shared in a single segment, just as in most 8-bit based processors, and could be used to build *.com*-files for instance. Precompiled libraries often came in several versions compiled for different memory models.

According to Morse et al., the designers actually contemplated using an 8-bit shift (instead of 4-bit), in order to create a 16 MB physical address space. However, as this would have forced segments to begin on 256-byte boundaries, and 1 MB was considered very large for a microprocessor around 1976, the idea was dismissed. Also, there were not enough pins available on a low-cost 40-pin package.^[14]

In principle, the address space of the x86 series *could* have been extended in later processors by increasing the shift value, as long as applications obtained their segments from the operating system and did not make assumptions about the equivalence of different segment:offset pairs.^[15] In practice the use of "huge" pointers and similar mechanisms was widespread and the flat 32-bit addressing made possible with the 32-bit offset registers in the 80386 eventually extended the limited addressing range in a more general way (see below).

Porting older software

Small programs could ignore the segmentation and just use plain 16-bit addressing. This allowed 8-bit software to be quite easily ported to the 8086. The authors of MS-DOS took advantage of this by providing an Application Programming Interface very similar to CP/M as well as including the simple *.com* executable file format, identical to CP/M. This was important when the 8086 and MS-DOS was new, because it allowed many existing CP/M (and other) applications to be quickly made available, greatly easing acceptance of the new platform.

Performance

Although partly shadowed by other design choices in this particular chip, the multiplexed bus limited performance slightly; transfers of 16-bit or 8-bit quantities were done in a four-clock memory access cycle (which was faster on 16-bit, although slower on 8-bit quantities, compared to typical contemporary "8-bit" CPUs). As instructions varied from one to six bytes, fetch and execution were made concurrent (as it remains in today's x86 processors): The *bus interface unit* fed the instruction stream to the *execution unit* through a 6-byte prefetch queue (a form of loosely coupled pipelining), speeding up operations on registers and immediates, while memory operations unfortunately became slower; four years later, this performance problem was fixed with the 80186 and 80286). However, the full (instead of partial) 16-bit architecture with a full width ALU meant that 16-bit arithmetic instructions could now be performed with a single ALU cycle (instead of two, via carry), speeding up such instructions considerably. Combined with orthogonalizations of operations versus operand-types and addressing modes, as well as other enhancements, this made the performance gain over the 8080 or 8085 fairly significant, despite cases where the older chips may be faster (see below).

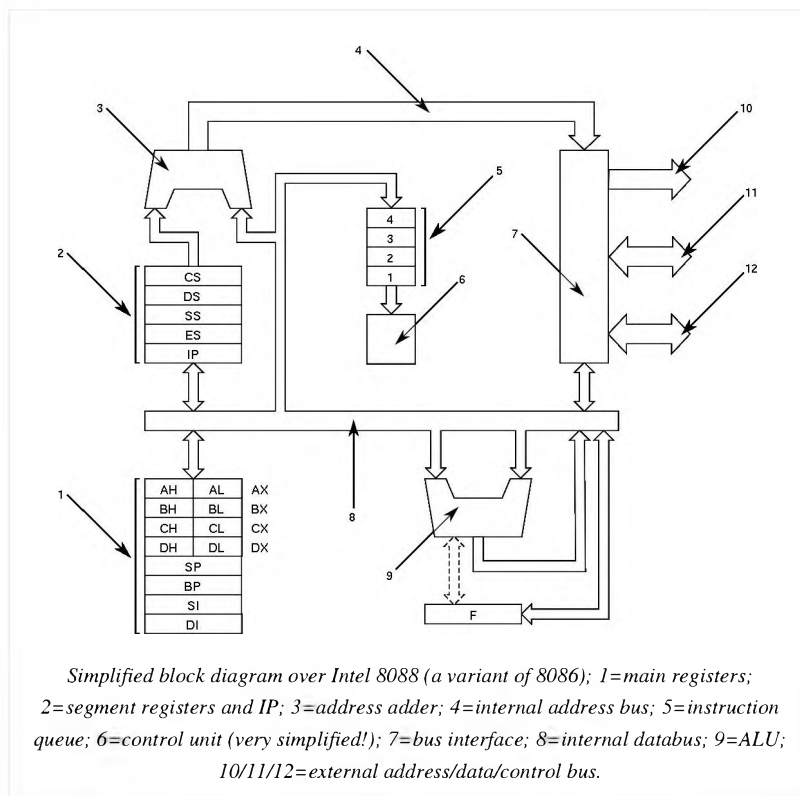
Execution times for typical instructions (in clock cycles)^[16]

instruction	register-register	register immediate	register-memory	memory-register	memory-immediate
mov	2	4	8+EA	9+EA	10+EA
ALU	3	4	9+EA.	16+EA.	17+EA
jump	<i>register => 11 ; label => 15 ; condition,label => 16</i>				
integer multiply	70~160 (depending on operand <i>data</i> as well as size) plus EA				
signed integer divide	80~190 (depending on operand <i>data</i> as well as size) plus EA				

- EA = time to compute effective address, ranging from 5 to 12 cycles.
- Timings are best case, depending on prefetch status, instruction alignment, and other factors.

As can be seen from these tables, operations on registers and immediates were fast (between 2 and 4 cycles), while memory-operand instructions and jumps were quite slow; jumps took more cycles than on the simple 8080 and 8085, and the 8088 (used in the IBM PC) was additionally hampered by its narrower bus. The reasons why most memory related instructions were slow were threefold:

- Loosely-coupled fetch and execution units are efficient for instruction prefetch, but not for jumps and random data access (without special measures).
- No dedicated address calculation adder was afforded; the microcode routines had to use the main ALU for this (although there was a dedicated *segment + offset* adder).
- The address and data buses were multiplexed, forcing a slightly longer (33~50%) bus cycle than in typical contemporary 8-bit processors.



However, memory access performance was drastically enhanced with Intel's next generation chips. The 80186 and 80286 both had dedicated address calculation hardware, saving many cycles, and the 80286 also had separate (non-multiplexed) address and data buses.

Floating point

The 8086/8088 could be connected to a mathematical coprocessor to add hardware/microcode-based floating point performance. The Intel 8087 was the standard math coprocessor for the 8086 and 8088, operating on 80-bit numbers. Manufacturers like Cyrix (8087-compatible) and Weitek (*non* 8087-compatible) eventually came up with high performance floating point co-processors that competed with the 8087 as well as with the subsequent, higher performing Intel 80387.

Chip versions

The clock frequency was originally limited to 5 MHz (IBM PC used 4.77 MHz, 4/3 the standard NTSC color burst frequency), but the last versions in HMOS were specified for 10 MHz. HMOS-III and CMOS versions were manufactured for a long time (at least a while into the 1990s) for embedded systems, although its successor, the 80186/80188 (which includes some on-chip peripherals), has been more popular for embedded use.

Derivatives and clones

Compatible—and, in many cases, enhanced—versions were manufactured by Fujitsu, Harris/Intersil, OKI, Siemens AG, Texas Instruments, NEC, Mitsubishi, and AMD. For example, the NEC V20 and NEC V30 pair were hardware compatible with the 8088 and 8086, respectively, but incorporated the instruction set of the 80186 along with some (but not all) of the 80186 speed enhancements, providing a drop-in capability to upgrade both instruction set and processing speed without manufacturers having to modify their designs. Such relatively simple and low-power 8086-compatible processors in CMOS are still used in embedded systems.

The electronics industry of the Soviet Union was able to replicate the 8086 through both industrial espionage and reverse engineering. The resulting chip, K1810BM86, was binary and pin-compatible with the 8086, but was not mechanically compatible because it used metric measurements.

The 8088 and 8086 were the respective cores of the Soviet-made PC-compatible ES1840 and ES1841 desktops. However, these computers had significant hardware differences from their authentic prototypes, and the data/address bus circuitry was designed independently of Intel products. ES1841 was the first PC compatible computer with dynamic bus sizing (US Pat. No 4,831,514). Later some of the ES1841 principles were adopted in PS/2 (US Pat. No 5,548,786) and some other machines (UK Patent Application, Publication No. GB-A-2211325, Published June. 28, 1989).



Soviet clone KP1810BM86.



OKI M80C86A QFP-56.

Microcomputers using the 8086

- One of the most influential microcomputers of all, the IBM PC, used the Intel 8088, a version of the 8086 with an eight-bit data bus (as mentioned above).
- The first commercial microcomputer built on the basis of the 8086 was the Mycron 2000.
- The IBM Displaywriter word processing machine and the Wang Professional Computer, manufactured by Wang Laboratories, also used the 8086. Also, this chip could be found in the AT&T 6300 PC (built by Olivetti).
- The NEC PC-9801.
- The first Compaq Deskpro used an 8086 running at 7.14 MHz, but was capable of running add-in cards designed for the 4.77 MHz IBM PC XT.
- The IBM PS/2 models 25 and 30 were built with an 8 MHz 8086.
- The Tandy 1000 SL-series machines used 8086 CPUs.
- The Amstrad PC1512, PC1640, PC2086, PC3086 and PC5086 all used 8086 CPUs at 8 MHz.
- NASA used original 8086 CPUs on equipment for ground-based maintenance of the Space Shuttle Discovery until the end of the space shuttle program in 2011. This decision was made to prevent software regression that might result from upgrading or from switching to imperfect clones.^[17]

Notes and references

- [1] "Microprocessor Hall of Fame" (http://web.archive.org/web/20070706032836/http://www.intel.com/museum/online/hist_micro/hof/). Intel. Archived from the original (http://www.intel.com/museum/online/hist_micro/hof/) on 2007-07-06. . Retrieved 2007-08-11.
- [2] It also permitted cheap 8080-family chips to be used (such as the 8254 CTC, 8255 PIO, and 8259 PIC which were used in the IBM PC design). In addition, it made PCB layout simpler and boards cheaper, as well as demanding fewer (1- or 4-bit wide) DRAM chips.
- [3] using enhancement load PMOS logic (demanding 14V, achieving TTL-compatibility by having V_{CC} at +5V and V_{DD} at -9V)
- [4] using non-saturated enhancement load NMOS logic (demanding a higher gate voltage for the load transistor-gates)
- [5] made possible with depletion load nMOS logic (the 8085 was later made using HMOS processing, just like the 8086)
- [6] Birth of a Standard: The Intel 8086 Microprocessor. Thirty years ago, Intel released the 8086 processor, introducing the x86 architecture that underlies every PC-Windows, Mac, or Linux-produced today (http://www.pcworld.com/article/146957/birth_of_a_standard_the_intel_8086_microprocessor.html), PC World, June 17, 2008
- [7] Rev.0 of the instruction set and architecture was ready in about three months, according to Morse.
- [8] Using rubylith, light boards, rulers, electric erasers, and a digitizer (according to Jenny Hernandez, member of the 8086 design team, in a statement made on Intel's web-page for its 25th birthday).
- [9] 8086 used less microcode than many competitors designs, such as the MC68000 and others
- [10] Randall L. Geiger, Phillip E. Allen, Noel R. Strader *VLSI design techniques for analog and digital circuits*, McGraw-Hill Book Co., 1990, ISBN 0070232539, page 779 "Random Logic vs. Structured Logic Forms", illustration of use of "random" describing CPU control logic
- [11] Fast static RAMs in MOS technology (as fast as bipolar RAMs) was an important product for Intel during this period.
- [12] CHMOS is intels name for CMOS circuits manufactured using processing steps very similar to HMOS.
- [13] Other members of the design team were Peter A.Stoll and Jenny Hernandez.
- [14] Intel 8008 to 8086 by Stephen P. Morse et al.
- [15] Some 80186 clones did change the shift value, but were never commonly used in desktop computers.
- [16] *Microsoft Macro Assembler 5.0 Reference Manual*. Microsoft Corporation. 1987. "Timings and encodings in this manual are used with permission of Intel and come from the following publications: Intel Corporation. iAPX 86, 88, 186 and 188 User's Manual, Programmer's Reference, Santa Clara, Calif. 1986." (Similarly for iAPX 286, 80386, 80387.)
- [17] For Old Parts, NASA Boldly Goes ... on eBay (<http://www.nytimes.com/2002/05/12/technology/ebusiness/12NASA.html?pagewanted=2>), May 12, 2002

External links

- Intel datasheets (<http://datasheets.chipdb.org/Intel/x86/808x/datashts/8086>)
 - List of 8086 CPUs and their clones at CPUworld.com (<http://www.cpu-world.com/CPUs/8086/>)
 - 8086 Pinouts (<http://www.cpu-world.com/info/Pinouts/8086.html>)
-