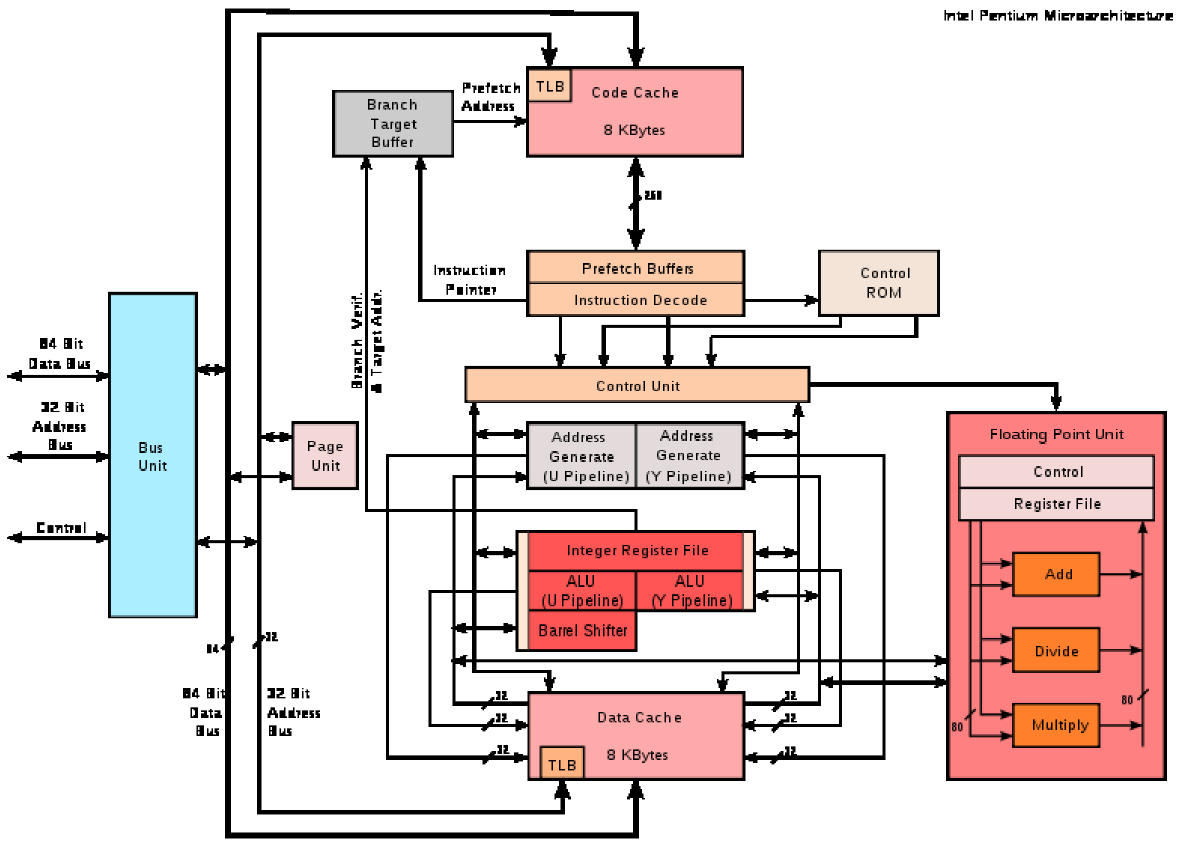
Contents

Pentium I

The original **Pentium** microprocessor was introduced in 1993. Its microarchitecture, deemed **P5**, was Intel's fifth-generation and first superscalar x86 architecture. As a direct extension of the 80486 architecture, it included ***dual integer pipelines, a faster FPU, wider data bus, separate code and data caches*** and features for further reduced address calculation latency. Some of the major improvements are:

1. **Superscalar architecture** — The Pentium has two pipelines that allow it to complete two instructions per clock cycle. The main pipe (U) can handle any instruction, while the other (V) can handle the most common simple instructions.
2. **64-bit external data bus** doubles the amount of information possible to read or write on each memory access and allows the Pentium to load its code cache faster than the 80486; it also allows faster access and storage of 64-bit and 80-bit x87 FPU data.
3. Separate code and data caches lessen the fetch and operand read/write conflicts compared to the 486.
4. Much **faster floating point unit.**
5. Four-input address-adders enables the Pentium to further reduce the address calculation latency compared to the 80486. The Pentium can calculate full addressing modes with *segment-base + base-register + scaled register + immediate offset* in a single cycle.
6. A faster fully hardware-based multiplier makes instructions such as MUL and IMUL several times as fast (and more predictable) than in the 80486.
7. Virtualized interrupt to speed up virtual 8086 mode. It also introduced enhanced self-test and debug features.
8. Pentium introduced the *System Management Mode* to manage power and security.



Pentium MMX

The P55C (or 80503) was sold as **Pentium with MMX Technology** (called **Pentium MMX**). Based on the P5 core, it featured a new set of ***57 "MMX" instructions*** intended to improve performance on multimedia tasks, such as encoding and decoding digital media data. The Penti um MMX line was introduced in October 1996.

MMX (unofficially *Multi Media eXtension*) is **Single Instruction Multiple Data Stream, *SIMD*** instruction set. After its introduction, MMX is sold as a processor supplementary capability on the IA-32 processors of Intel.

MMX defined eight registers, known as **MM0 through MM7** (referred to as *MMn*). To avoid compatibility problems with the context switch mechanisms in existing operating systems, these registers were aliases for the existing x87 FPU stack registers. Hence, anything that was done to the floating point stack would also affect the MMX registers and vice versa. However, unlike the FP U stack, the MMn registers are directly addressable.

Each of the MMn registers holds 64 bits (the mantissa-part of a full 80-bit FPU register). The main usage of the MMX instruction set is based on the concept of packed data types, which means that instead of using the whole register for a single 64-bit integer, two 32-bit integers, four 16-bit integers, or eight 8-bit integers may be processed concurrently.

Because the FPU stack registers are 80 bits wide, the upper 16 bits of the stack registers go unused in MMX, and these bits are set to all ones, which makes them infinities in the floating point representation. This can be used to decide whether a particular register's content is intended as floating point or SIMD data.

MMX provides only integer operations. When originally developed, for the Intel i860, the use of integer math made sense (both ***2D and 3D calculations*** required it), but as graphics cards that did much of this became common, integer SIMD in the CPU became somewhat redundant for graphical applications. On the other hand, the saturation arithmetic operations in MMX could significantly speed up some digital signal processing applications.

Intel addressed the shortcomings of the MMX technology through ***SSE (Streaming SIMD Extensions)***.
SSE is a greatly expanded set of SIMD instructions with 32-bit floating point support and an additional set of 128-bit vector registers that made it easy to perform SIMD and FPU operations at the same time.
SSE was in turn expanded with SSE2, which also extended MMX instructions so they can operate on 128-bit XMM and recently with SSE4.2, introduced in the Core microarchitecture.

Pentium Pro

The **Pentium Pro** is a sixth-generation x86 microprocessor. It introduced the P6 architecture and was originally intended to replace the original Pentium in a full range of applications. Later, it was reduced to a narrower role as a server and high-end desktop processor. The Pentium Pro was capable of both dual- and quad-processor configurations.

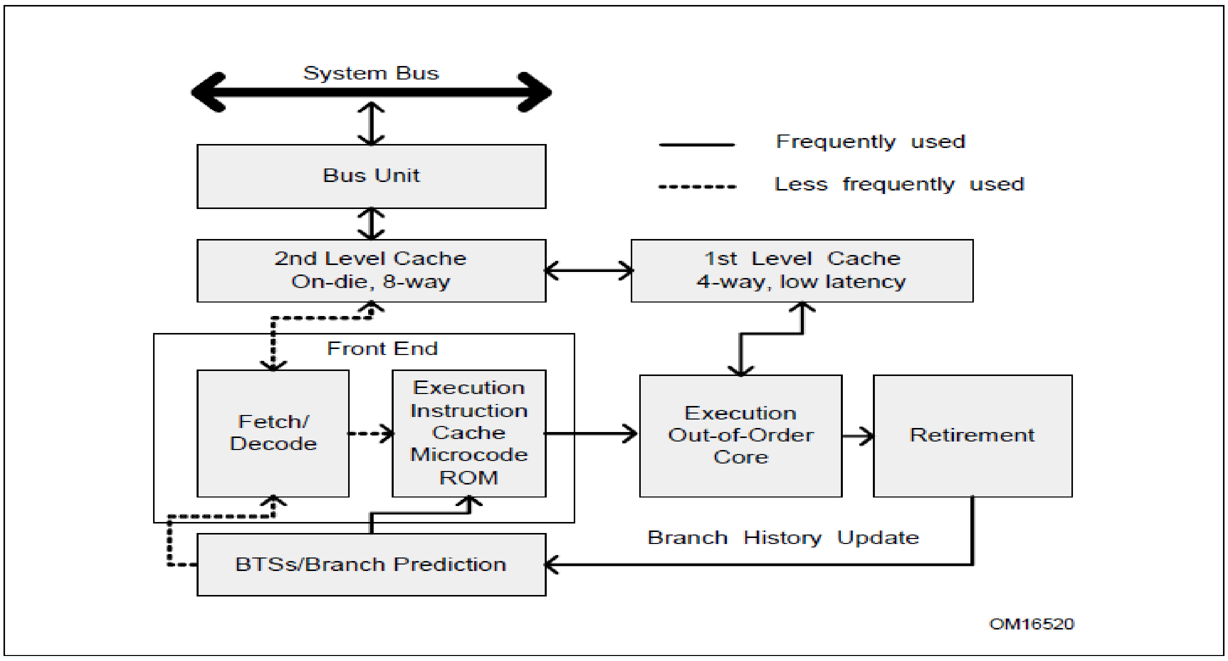


Figure 2-1. The P6 Processor Microarchitecture with Advanced Transfer Cache Enhancement

It has a **decoupled, 12 stage, super-pipelined architecture** which uses an instruction pool. The Pentium Pro pipeline had extra decode stages to dynamically translate IA-32 instructions into buffered micro-operation sequences which could then be analysed, reordered, and renamed in order to detect parallelizable operations that may be issued to more than one execution unit at once. The Pentium Pro thus featured ***out of order execution***, including ***speculative execution*** via register renaming. It also had a wider 36-bit address bus.

The Pentium Pro has **an 8 KB instruction cache**, from which up to 16 bytes are fetched on each cycle and sent to the instruction decoders. Pentium Pro's most noticeable addition was its ***on-package L2 cache, which ranged from 256 KB-1 MB***. Intel placed the L2 die(s) separately in the package which still allowed it to run at the same clock speed as the CPU core. Additionally, the Pentium Pro's cache had its own back-side bus (called ***dual independent bus*** by Intel). Because of this, the CPU could read main memory and cache concurrently. The cache was also "non-blocking", meaning that the processor could issue more than one cache request at a time (up to 4), reducing cache-miss penalties, thus, and **implemented *Memory Level Parallelism***.

The Pentium Pro **has two integer units and one floating-point unit (FPU).** One of the integer units shares the same ports as the FPU, and therefore the Pentium Pro can only dispatch two integer micro-ops and one floating-point micro-op per cycle. Of the two integer units, only one has the full complement of functions such as a barrel shifter, multiplier and divider the other is limited to simple operations such as add, subtract, and the calculation of branch target addresses.
Addition and multiplication are pipelined and have a latency of three and five cycles, respectively. Division and square-root are not pipelined and are executed in separate units that share the FPU's ports.

Pentium II

The **Pentium II** brand refers to Intel's sixth-generation microarchitecture ("P6") and x86-compatible microprocessors introduced in May 1997. Containing 7.5 million transistors, the Pentium II featured **an improved version of the first *P6*-generation core of the Pentium Pro**, which contained 5.5 million transistors. However, its L2 cache subsystem was a downgrade when compared to Pentium Pros.

The Pentium II microprocessor was largely based upon the microarchitecture of its predecessor, the Pentium Pro, but with some significant improvements. Unlike previous Pentium and Pentium Pro processors, the Pentium II CPU was packaged in a slot-based module rather than a CPU socket. A fixed or removable heat sink was carried on one side, sometimes using its own fan. This larger package was a compromise allowing Intel to separate the secondary cache from the processor while still keeping it on a closely coupled back-side bus. The L2 cache ran at half the processor's clock frequency. The smallest cache size was increased to 512 KB.

Intel notably improved 16-bit code execution performance on the Pentium II. The Pentium II was also ***the first P6-based CPU to implement the Intel MMX integer SIMD instruction set*** which had already been introduced on the Pentium MMX. The improved 16-bit performance and MMX support made it a better choice for consumer-level operating systems, such as Windows 9x, and multimedia applications. Combined with the larger L1 cache and improved 16-bit performance, the slower and cheaper L2 cache's performance impact was reduced.

Some of the other features of the processor and in general of the P6 family are:

1. Instruction cache is replaced by a front-end fetch/decode micro-coding unit, which converts instructions into micro-codes and stores them in a pool. ***Out of order instruction*** execution is supported.
2. Branch instructions are handled in the ***branch target speculation*** and ***branch prediction unit***. The branches are speculatively handled are committed only when validated. In case of a false prediction, recovery can be made easily.
3. Several different *Model Specific Registers* are added.
4. Eight additional registers, 128 bit each, were introduced to enhance the multi-media capabilities. They were designated ***XMM0-XMM7***. These implemented Intel's MMX technology which was later superseded by the SSE technology.
5. A ***Thermal Monitoring Scheme*** was added that monitored the processor chip's temperature and took suitable action by reducing the clocking of the processor.

Pentium III

The Pentium III introduced the ***Streaming SIMD Extensions (SSE)*** to the IA-32 architecture. SSE extensions expand the SIMD execution model introduced with the Intel MMX technology by providing a **new set of 128-bit registers and the ability to perform SIMD operations on packed single precision floating-point values**. Intel implemented the 128-bit SSE architecture by double-cycling the existing 64-bit data paths and by merging the SIMD-FP multiplier unit with the x87 scalar FPU multiplier into a single unit.

SSE extensions add the following features to the IA-32 architecture, while maintaining backward compatibility with all existing IA-32 processors, applications and operating systems.

• **Eight 128-bit data registers (called XMM registers)** in non-64-bit modes; sixteen XMM registers are available in 64-bit mode.

• **The 32-bit MXCSR register**, which provides control and status bits for operations performed on XMM registers.

• Instructions that perform **SIMD operations on single-precision floating-point values** and that extend SIMD operations that can be performed on integers:
— 128-bit Packed and scalar single-precision floating-point instructions that operate on data located in MMX registers
— 64-bit SIMD integer instructions that support additional operations on packed integer operands located in MMX registers

• Instructions that save and restore the state of the MXCSR register.

• Instructions that support explicit prefetching of data, control of the cache-ability of data, and control the ordering of store operations.

These features extend the IA-32 architecture's SIMD programming model in four important ways:

• The ability to perform SIMD operations on four packed single-precision floating point values enhances the performance of IA-32 processors for advanced media and communications applications that use computation-intensive algorithms to perform repetitive operations on large arrays of simple, native data elements.

• The ability to perform SIMD single-precision floating-point operations in XMM registers and SIMD integer operations in MMX registers provides greater flexibility and throughput for executing applications that operate on large arrays of floating-point and integer data.

• Cache control instructions provide the ability to stream data in and out of XMM registers without polluting the caches and the ability to pre-fetch data to selected cache levels before it is actually used. Applications that require regular access to large amounts of data benefit from these prefetching and streaming store capabilities.

• The ***SFENCE (store fence)*** instruction provides greater control over the ordering of store operations when using weakly-ordered memory types.

The Pentium III was the first x86 CPU to include a unique, retrievable, identification number, called ***PSN (Processor Serial Number).*** A Pentium III's PSN can be read by software through the CPUID instruction if this feature has not been disabled through the BIOS.

The Pentium III was superseded by the Pentium IV which introduced the ***NetBurst*** Architecture. The ***Tualatin core*** of Pentium III processors served as the base for the ***Intel Pentium M processors architecture*** which in turn formed the basis of Intel's energy-efficient Core micro-architecture used in the ***Core 2, Pentium Dual-Core, Celeron (Core), and Xeon.***

ISA (Industry Standard Architecture)

Industry Standard Architecture (abbrev. as **ISA**) was a computer bus standard for IBM compatible computers. The ISA bus was developed by a team lead by Mark Dean at IBM as part of the IBM PC project in 1981. It **originated as an 8-bit system** and was extended in 1983 for the XT system architecture. The newer **16-bit standard, the IBM AT bus**, was introduced in 1984. In 1988 PC manufacturers put forth the **32-bit EISA** standard in response to the MCA standard.

IBM designed the 8-bit version as a buffered interface to the external bus of the Intel 8088 (16/8 bit) CPU used in the original IBM PC and PC/XT, and the 16-bit version as an upgrade for the external bus of the Intel 80286 CPU used in the IBM AT. Designed to connect peripheral cards to the motherboard, ISA allows for bus mastering although only the first 16 MB of main memory are available for direct access. The 8-bit bus ran at 4.77 MHz while the 16-bit bus operated at 6 or 8 MHz. The features of the ISA bus are:

Bus width	8-bit/ 16-bit
Compatibility	IBM PC-XT/ IBM PC-AT
Vcc	-5 V, +5 V, -12 V, +12 V
Pins	62 pins/ 98 pins
Clock	4.77 MHz/ 6-8 MHz

The **ISA PC/XT-bus** uses **de-multiplexed, buffered versions of the 8 data and 20 address lines of the 8088 processor**. It directly supports PMOS and enhancement mode NMOS. ***The bus architecture uses a single Intel 8259 Programmable Interrupt Controller, giving 8 vector prioritized interrupts***. It has **4 DMA channels:**

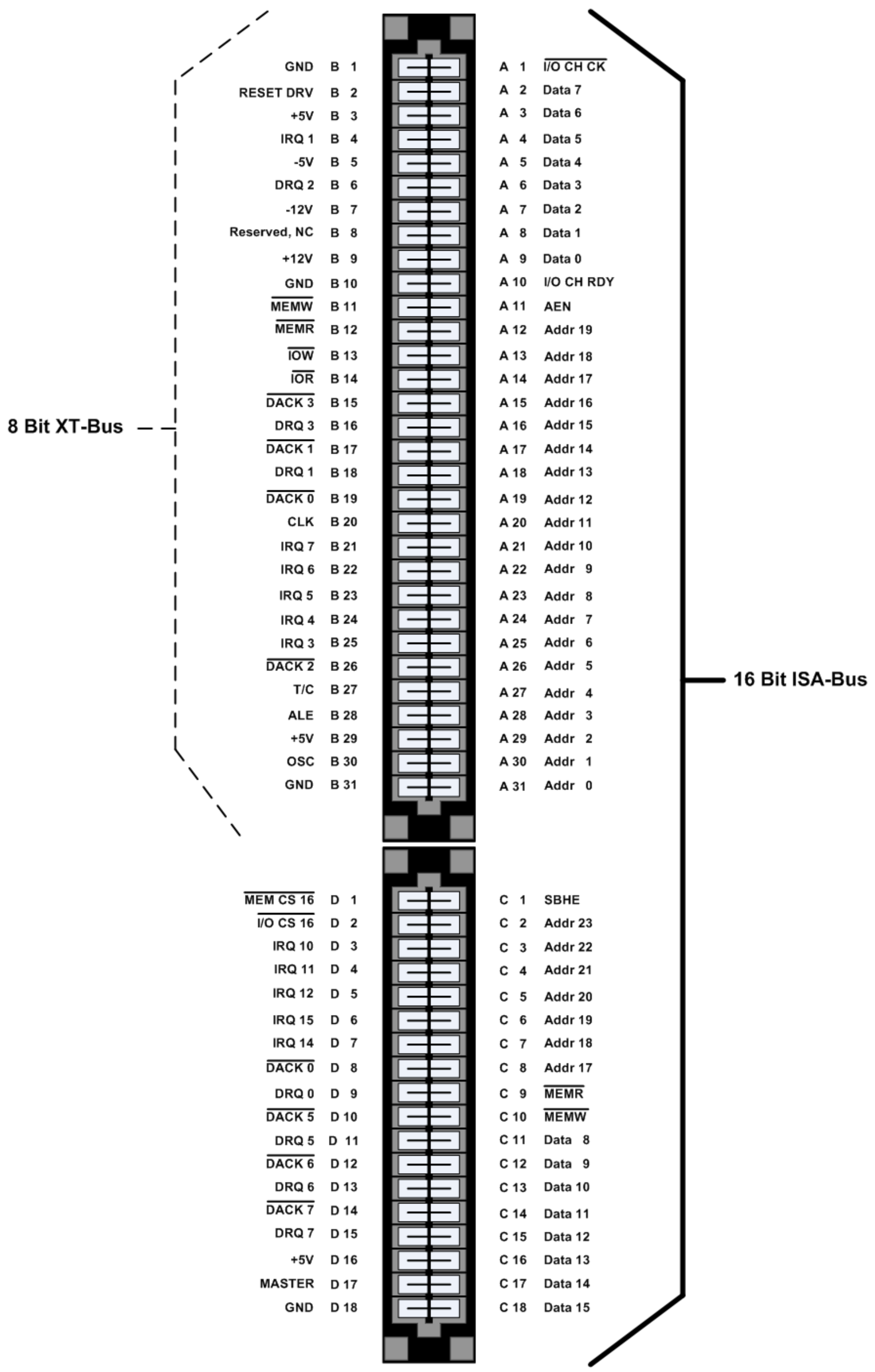
DMA channel	**Expansion**	**Standard function**
0	No	Dynamic RAM refresh
1	Yes	Add-on cards
2	Yes	Floppy disk controller
3	Yes	Hard disk controller

The **ISA PC/AT-bus** is a 16-bit version of the ISA PC/XT bus officially **termed *I/O Channel*** by IBM. **It extends the XT-bus by adding a second shorter edge connector in-line with the 8-bit XT-bus connector, which is unchanged, retaining compatibility with most 8-bit cards**. The second connector adds four additional address lines for a total of 24, and eight additional data lines for a total of 16. It also adds a second 8259 PIC (connected to one of the lines of the first) and four 16-bit DMA channels, as well as control lines to select 8 or 16 bit transfers.

Motherboard devices have dedicated IRQs. 16-bit devices can use either PC-bus or PC/AT-bus IRQs. It is therefore possible to connect up to 6 devices that use one 8-bit IRQ each, or up to 5 devices that use one 16-bit IRQ each. At the same time, up to four devices may use one 8-bit DMA channel each, while up to three devices can use one 16-bit DMA channel each.

Memory address decoding for the selection of 8 or 16-bit transfer mode was limited to 128 KB sections - A0000-BFFFF, C0000-DFFFF, E0000-FFFFF leading to problems when mixing 8 and 16-bit cards, as they could not co-exist in the same 128 KB area.

16 Bit ISA Bus – top view



GND B 1 — A 1 $\overline{\text{I/O CH CK}}$
RESET DRV B 2 — A 2 Data 7
+5V B 3 — A 3 Data 6
IRQ 1 B 4 — A 4 Data 5
-5V B 5 — A 5 Data 4
DRQ 2 B 6 — A 6 Data 3
-12V B 7 — A 7 Data 2
Reserved, NC B 8 — A 8 Data 1
+12V B 9 — A 9 Data 0
GND B 10 — A 10 I/O CH RDY
$\overline{\text{MEMW}}$ B 11 — A 11 AEN
$\overline{\text{MEMR}}$ B 12 — A 12 Addr 19
$\overline{\text{IOW}}$ B 13 — A 13 Addr 18
$\overline{\text{IOR}}$ B 14 — A 14 Addr 17
$\overline{\text{DACK 3}}$ B 15 — A 15 Addr 16
DRQ 3 B 16 — A 16 Addr 15
$\overline{\text{DACK 1}}$ B 17 — A 17 Addr 14
DRQ 1 B 18 — A 18 Addr 13
$\overline{\text{DACK 0}}$ B 19 — A 19 Addr 12
CLK B 20 — A 20 Addr 11
IRQ 7 B 21 — A 21 Addr 10
IRQ 6 B 22 — A 22 Addr 9
IRQ 5 B 23 — A 23 Addr 8
IRQ 4 B 24 — A 24 Addr 7
IRQ 3 B 25 — A 25 Addr 6
$\overline{\text{DACK 2}}$ B 26 — A 26 Addr 5
T/C B 27 — A 27 Addr 4
ALE B 28 — A 28 Addr 3
+5V B 29 — A 29 Addr 2
OSC B 30 — A 30 Addr 1
GND B 31 — A 31 Addr 0
$\overline{\text{MEM CS 16}}$ D 1 — C 1 SBHE
$\overline{\text{I/O CS 16}}$ D 2 — C 2 Addr 23
IRQ 10 D 3 — C 3 Addr 22
IRQ 11 D 4 — C 4 Addr 21
IRQ 12 D 5 — C 5 Addr 20
IRQ 15 D 6 — C 6 Addr 19
IRQ 14 D 7 — C 7 Addr 18
$\overline{\text{DACK 0}}$ D 8 — C 8 Addr 17
DRQ 0 D 9 — C 9 $\overline{\text{MEMR}}$
$\overline{\text{DACK 5}}$ D 10 — C 10 $\overline{\text{MEMW}}$
DRQ 5 D 11 — C 11 Data 8
$\overline{\text{DACK 6}}$ D 12 — C 12 Data 9
DRQ 6 D 13 — C 13 Data 10
$\overline{\text{DACK 7}}$ D 14 — C 14 Data 11
DRQ 7 D 15 — C 15 Data 12
+5V D 16 — C 16 Data 13
MASTER D 17 — C 17 Data 14
GND D 18 — C 18 Data 15
8 Bit XT-Bus
16 Bit ISA-Bus

EISA (Extended Industry Standard Architecture)

EISA extends the AT bus to 32 bits and allows more than one CPU to share the bus. The bus mastering support is also enhanced to provide access to 4 GB of memory. EISA can accept older XT and ISA boards — the lines and slots for EISA are a superset of ISA. IBM worried over losing market share created the MCA bus which had numerous enhancements over the ISA but patented it, and placed it under stringent licensing policies. EISA was developed by a group of PC manufacturers led by Compaq as an extension to the ISA AT bus to counter this.

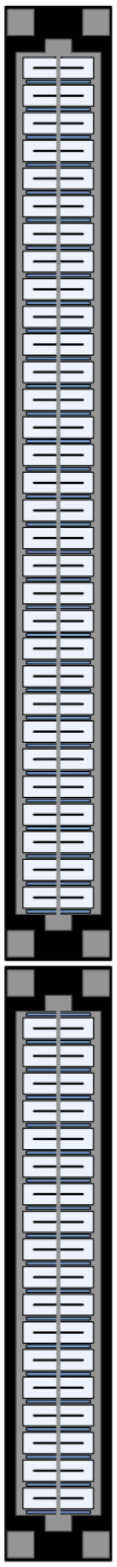
EISA bus had a slight performance disadvantage over the MCA, EISA contained almost all of the technological benefits that MCA boasted, including ***bus mastering, burst mode, software configurable resources, and 32-bit data/address buses***. The performance parameters for the bus are:

Bus width	32-bit
Compatibility	8-bit, 16-bit, 32-bit ISA
Vcc	-5 V, +5 V, -12 V, +12 V
Pins	98+100 inlay
Clock	8.33 MHz
Theoretical Data rate	About 33 MBps(8.33Mhz x 4bytes)
Usable Data rate	About 20 MBps

EISA replaced the tedious jumper configuration common with ISA cards with software-based configuration. Every EISA system shipped with an EISA configuration utility. The user would boot into this utility, either from floppy disk or on a dedicated hard drive partition. The utility software would detect all EISA cards in the system, and could configure any hardware resources (interrupts, memory ports etc) on any EISA card, or on the EISA system motherboard. The user could also enter information about ISA cards in the system, allowing the utility to automatically reconfigure EISA cards to avoid resource conflicts.

One of the benefits to come out of the EISA standard was a final codification of the standard to which ISA slots and cards should be held. Particularly, **clock speed was fixed at an industry standard of 8.33MHz**. Thus, even systems which didn't use the EISA bus gained the advantage of having the ISA standardized, which contributed to its longevity.

32 Bit EISA Bus – top view



GND	F 1	GND	B 1	A 1	$\overline{\text{I/O CH CK}}$	E 1	CMD
+5V	F 2	RESET DRV	B 2	A 2	Data 7	E 2	START
+5V	F 3	+5V	B 3	A 3	Data 6	E 3	EX RDY
X, NC	F 4	IRQ 1	B 4	A 4	Data 5	E 4	EX 32
X, NC	F 5	-5V	B 5	A 5	Data 4	E 5	GND
Kodierung, Key	F 6	DRQ 2	B 6	A 6	Data 3	E 6	Kodierung, Key
X, NC	F 7	-12V	B 7	A 7	Data 2	E 7	EX 16
X, NC	F 8	Reserved, NC	B 8	A 8	Data 1	E 8	SLBURST
+12V	F 9	+12V	B 9	A 9	Data 0	E 9	MSBURST
M-IO	F 10	GND	B 10	A 10	I/O CH RDY	E 10	W - R
LOCK	F 11	$\overline{\text{MEMW}}$	B 11	A 11	AEN	E 11	GND
Reserved	F 12	$\overline{\text{MEMR}}$	B 12	A 12	Addr 19	E 12	Reserved
GND	F 13	$\overline{\text{IOW}}$	B 13	A 13	Addr 18	E 13	Reserved
Reserved	F 14	$\overline{\text{IOR}}$	B 14	A 14	Addr 17	E 14	Reserved
BE 3	F 15	$\overline{\text{DACK 3}}$	B 15	A 15	Addr 16	E 15	GND
Kodierung, Key	F 16	DRQ 3	B 16	A 16	Addr 15	E 16	Kodierung, Key
BE 2	F 17	$\overline{\text{DACK 1}}$	B 17	A 17	Addr 14	E 17	BE 1
BE 0	F 18	DRQ 1	B 18	A 18	Addr 13	E 18	LA 31
GND	F 19	$\overline{\text{DACK 0}}$	B 19	A 19	Addr 12	E 19	GND
+5V	F 20	CLK	B 20	A 20	Addr 11	E 20	LA 30
LA 29	F 21	IRQ 7	B 21	A 21	Addr 10	E 21	LA 28
GND	F 22	IRQ 6	B 22	A 22	Addr 9	E 22	LA 27
LA 26	F 23	IRQ 5	B 23	A 23	Addr 8	E 23	LA 25
LA 24	F 24	IRQ 4	B 24	A 24	Addr 7	E 24	GND
Kodierung, Key	F 25	IRQ 3	B 25	A 25	Addr 6	E 25	Kodierung, Key
LA 16	F 26	$\overline{\text{DACK 2}}$	B 26	A 26	Addr 5	E 26	LA 15
LA 14	F 27	T/C	B 27	A 27	Addr 4	E 27	LA 13
+5V	F 28	ALE	B 28	A 28	Addr 3	E 28	LA 12
+5V	F 29	+5V	B 29	A 29	Addr 2	E 29	LA 11
GND	F 30	OSC	B 30	A 30	Addr 1	E 30	GND
LA 10	F 31	GND	B 31	A 31	Addr 0	E 31	LA 9

LA 8	H 1	$\overline{\text{MEM CS 16}}$	D 1	C 1	SBHE	G 1	LA 7
LA 6	H 2	$\overline{\text{I/O CS 16}}$	D 2	C 2	Addr 23	G 2	GND
LA 5	H 3	IRQ 10	D 3	C 3	Addr 22	G 3	LA 4
+5V	H 4	IRQ 11	D 4	C 4	Addr 21	G 4	LA 3
LA 2	H 5	IRQ 12	D 5	C 5	Addr 20	G 5	GND
Kodierung, Key	H 6	IRQ 13	D 6	C 6	Addr 19	G 6	Kodierung, Key
Data 16	H 7	IRQ 14	D 7	C 7	Addr 18	G 7	Data 17
Data 18	H 8	$\overline{\text{DACK 0}}$	D 8	C 8	Addr 17	G 8	Data 19
GND	H 9	DRQ 0	D 9	C 9	$\overline{\text{MEMR}}$	G 9	Data 20
Data 21	H 10	$\overline{\text{DACK 5}}$	D 10	C 10	$\overline{\text{MEMW}}$	G 10	Data 22
Data 23	H 11	DRQ 5	D 11	C 11	Data 8	G 11	GND
Data 24	H 12	$\overline{\text{DACK 6}}$	D 12	C 12	Data 9	G 12	Data 25
GND	H 13	DRQ 6	D 13	C 13	Data 10	G 13	Data 26
Data 27	H 14	$\overline{\text{DACK 7}}$	D 14	C 14	Data 11	G 14	Data 28
Kodierung, Key	H 15	DRQ 7	D 15	C 15	Data 12	G 15	Kodierung, Key
Data 29	H 16	+5V	D 16	C 16	Data 13	G 16	GND
+5V	H 17	MASTER	D 17	C 17	Data 14	G 17	Data 30
+5V	H 18	GND	D 18	C 18	Data 15	G 18	Data 31
MAC KN	H 19					G 19	MRE QN

PCI (Peripheral Component Interconnect)

Conventional PCI is a bus standard for attaching hardware devices in a computer. These devices can take either the form of an IC fitted onto the motherboard itself or an expansion card that fits into a slot. The PCI Local Bus is common in modern PCs, where it has displaced ISA and EISA Local Bus as the standard expansion bus. The PCI specification covers the physical size of the bus (including the size and spacing of the circuit board edge electrical contacts), electrical characteristics, bus timing, and protocols. Typical PCI cards used in PCs include: network cards, sound cards, modems, extra ports such as USB, TV tuner cards and disk controllers.

The specifications for the PCI bus are:

Bus width	32-bits
Memory Address space	32-/64-bit memory
I/O port space	32-bit
Vcc	5 volt
Clock	33.33 MHz
Peak transfer rate	133 MBps

PCI bus also provides specification for 64-bit bus width, 3.3V signalling, and 66 MHz clock. The PCI bus arbiter performs bus arbitration among multiple masters on the PCI bus. **Any number of bus masters can reside on the PCI bus, as well as requests for the bus. One pair of request and grant signals is dedicated to each bus master.**

PCI provides **separate memory and I/O port address spaces** for the x86 processor family, **64 and 32 bits, respectively**. Addresses in these address spaces are assigned by software. A third address space, called **the PCI Configuration Space**, which uses a fixed addressing scheme, allows software to determine the amount of memory and I/O address space needed by each device. ***Each device can request up to six areas of memory space or I/O port space via its configuration space registers***.
In a typical system, the OS queries all PCI buses at start-up time to find out the devices present and system resources needed. The PCI configuration space also contains a small amount of device type information, which helps an OS choose device drivers for it.

The **PCI bus includes 4 interrupt lines**, all of which are available to each device. However, they are not wired in parallel as are the other PCI bus lines. The positions of the interrupt lines rotate between slots, so what appears to one device as the INTA# line is INTB# to the next and INTC# to the one after that. Single-function devices use their INTA# for interrupt signalling, so the device load is spread fairly evenly across the four available interrupt lines. This alleviates a common problem with sharing interrupts.
PCI bridges (between two PCI buses) map the four interrupt traces on each of their sides in varying ways. Some bridges use a fixed mapping, and in others it is configurable. In the general case, software cannot determine which interrupt line a device's INTA# pin is connected to across a bridge. The mapping of PCI interrupt lines onto system interrupt lines, through the PCI host bridge, is similarly implementation-dependent. The result is that it can be impossible to determine how a PCI device's interrupts will appear to software. Platform-specific BIOS code is meant to know this, and set a field in each device's configuration space indicating which IRQ it is connected to, but this process is not reliable. PCI interrupt lines are level-triggered.

Later revisions of the PCI specification add support for message-signalled interrupts. In this system a device signals its need for service by performing a memory write, rather than by asserting a dedicated line. This alleviates the problem of scarcity of interrupt lines. It also resolves the routing problem, because the memory write is not unpredictably modified between device and host. PCI Express does not have physical interrupt lines at all and uses message-signalled interrupts exclusively.