I/O

? ? ?

Besides the cache/memory managemen unit, what sorts of things does the processor need to talk to?

Abstraction of I/O

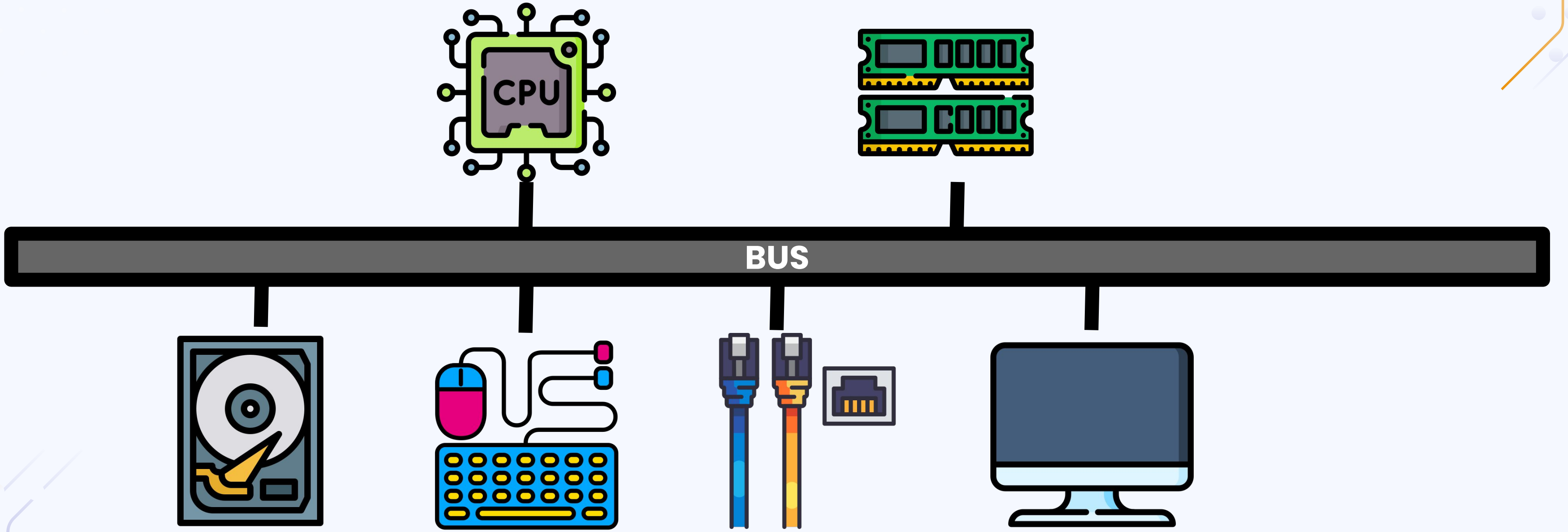
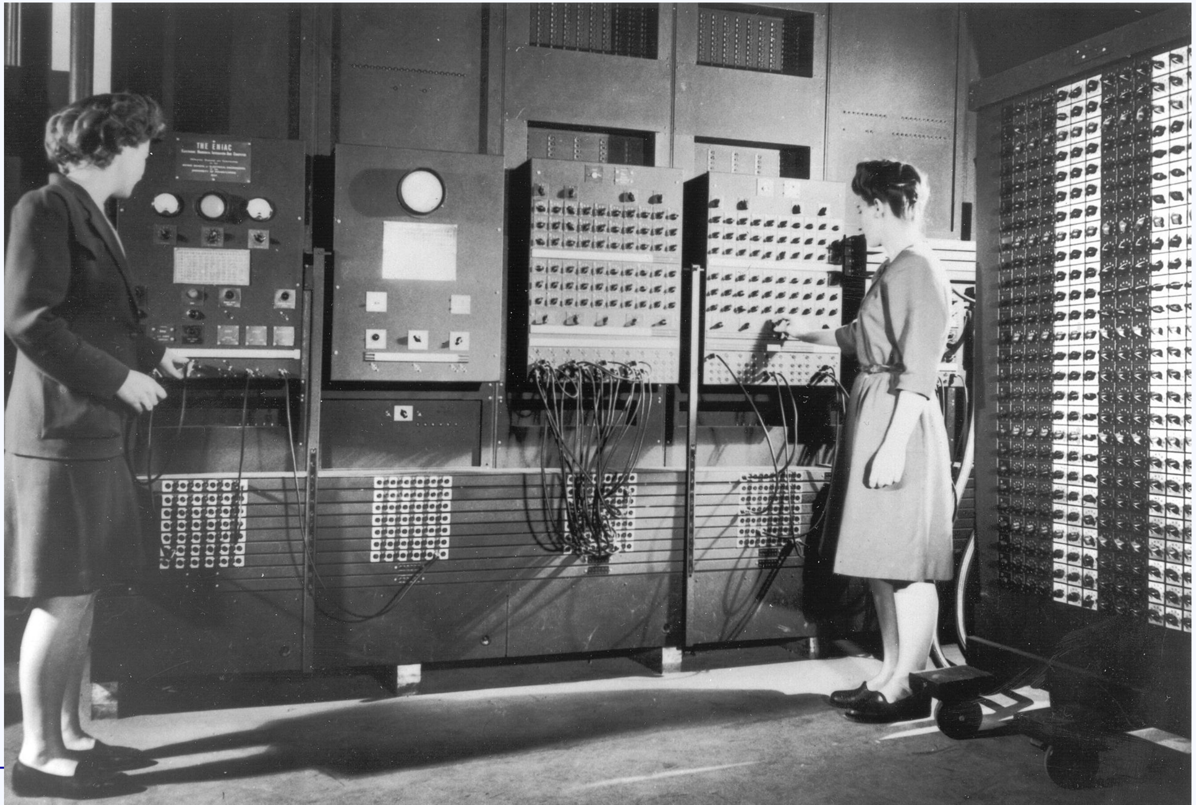


image source: flaticon.com

History of I/O: Plugboard computers (ENIAC, <1946)



History of I/O: UNIVAC 1 (1951)

image source

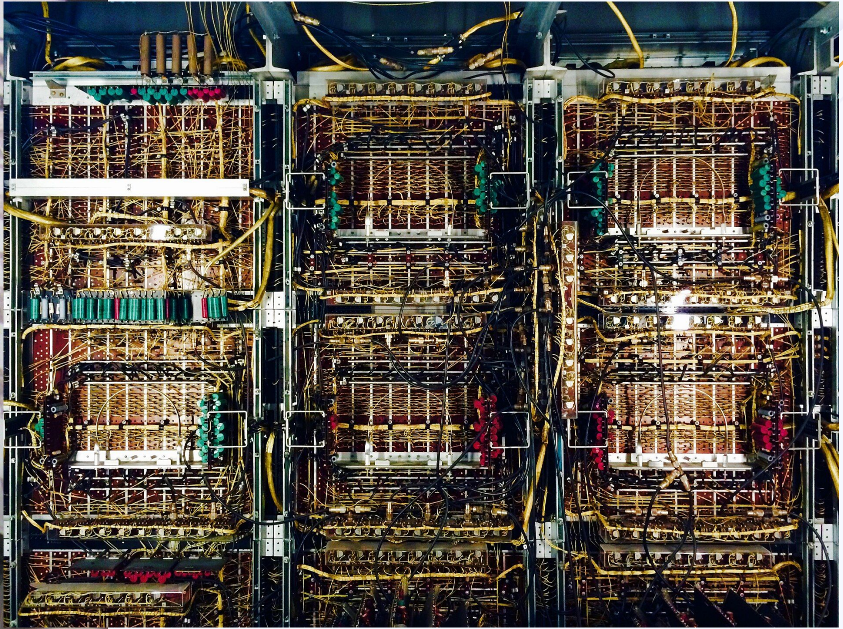


image source

History of I/O: IBM System/360 (>=1964)

image source

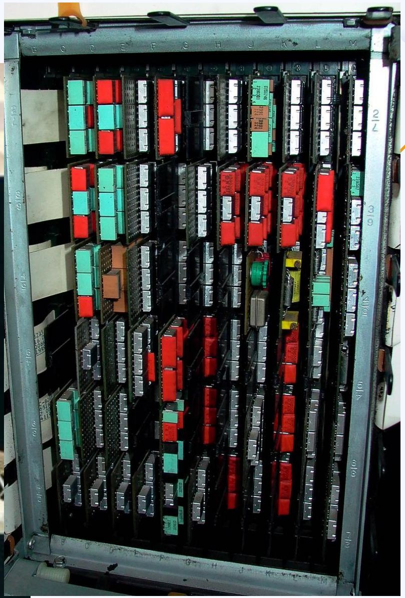


image source

History of I/O: PDP-8 (>=1965)

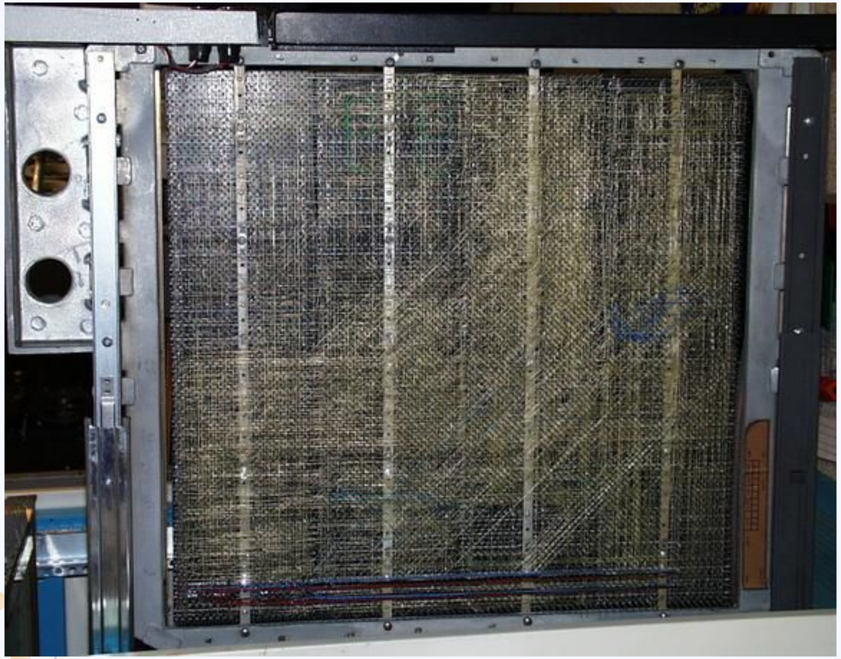
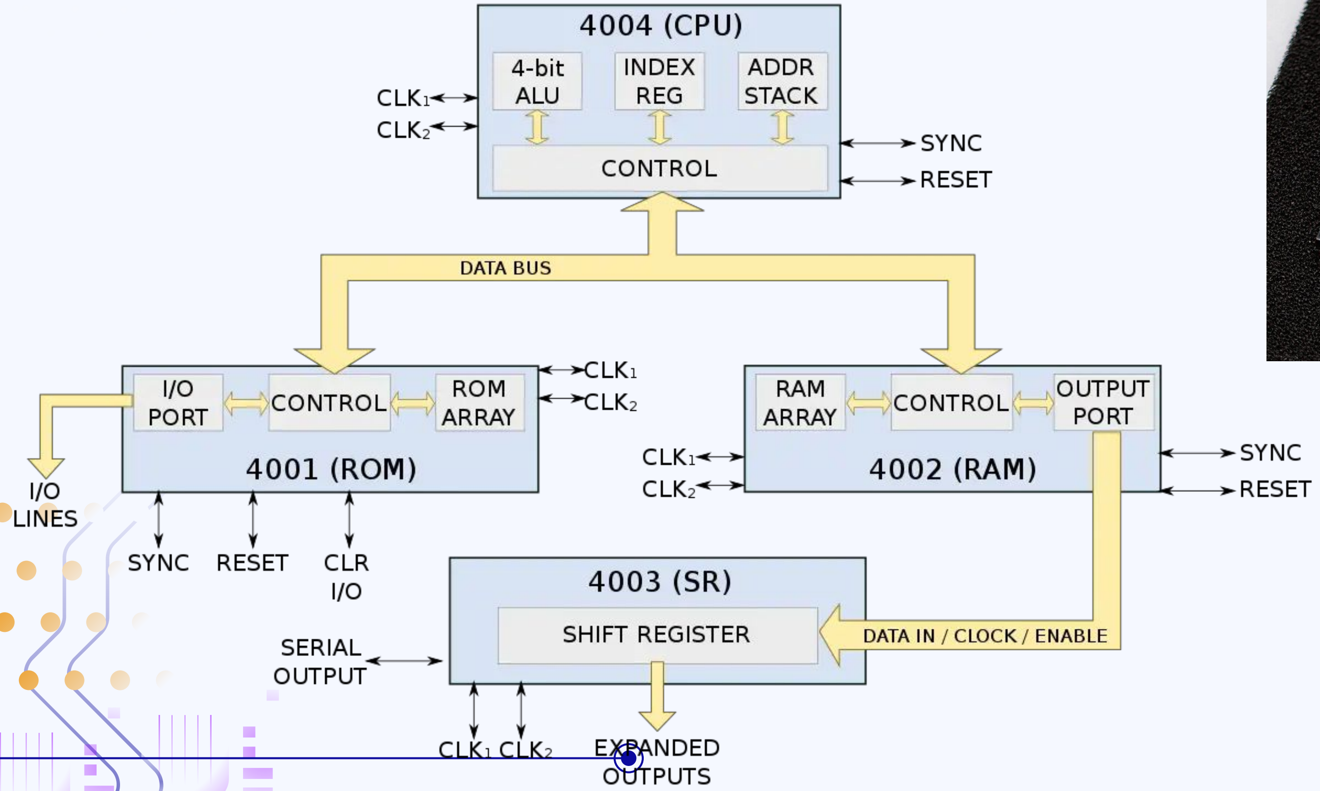


image source

image source

History of I/O: Intel 4004/MCS-4 (1971)



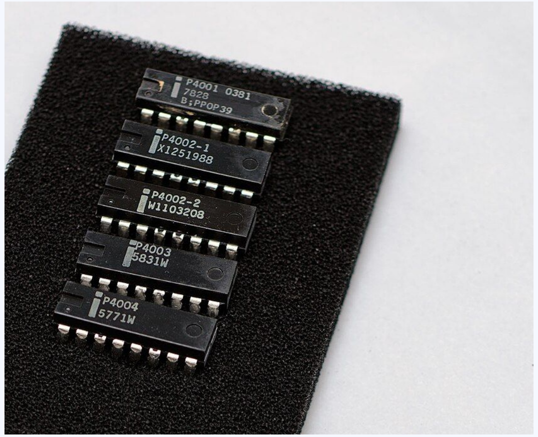
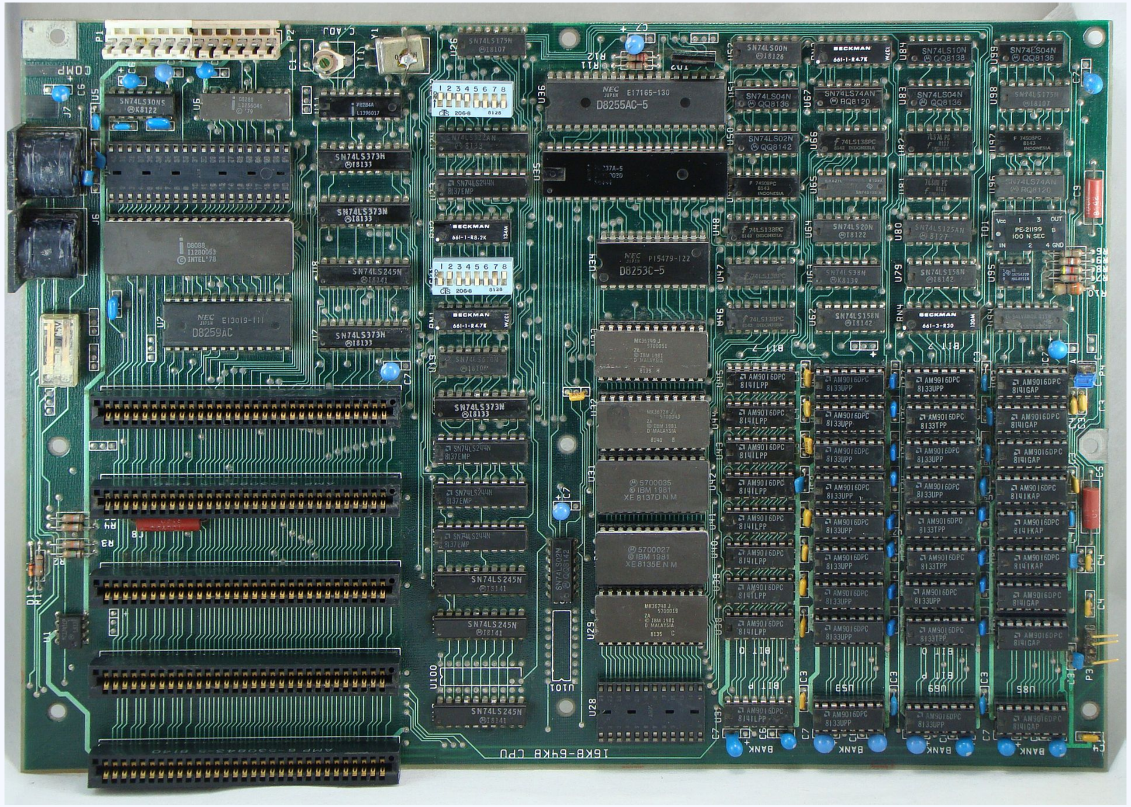


image source

History of I/O: IBM PC (1981)



Motherboards

Motherboard: printed circuit board (PCB) that holds computer components

Chipset: (usually on mobo) circuit that manages connection of CPU w/ memory and peripherals

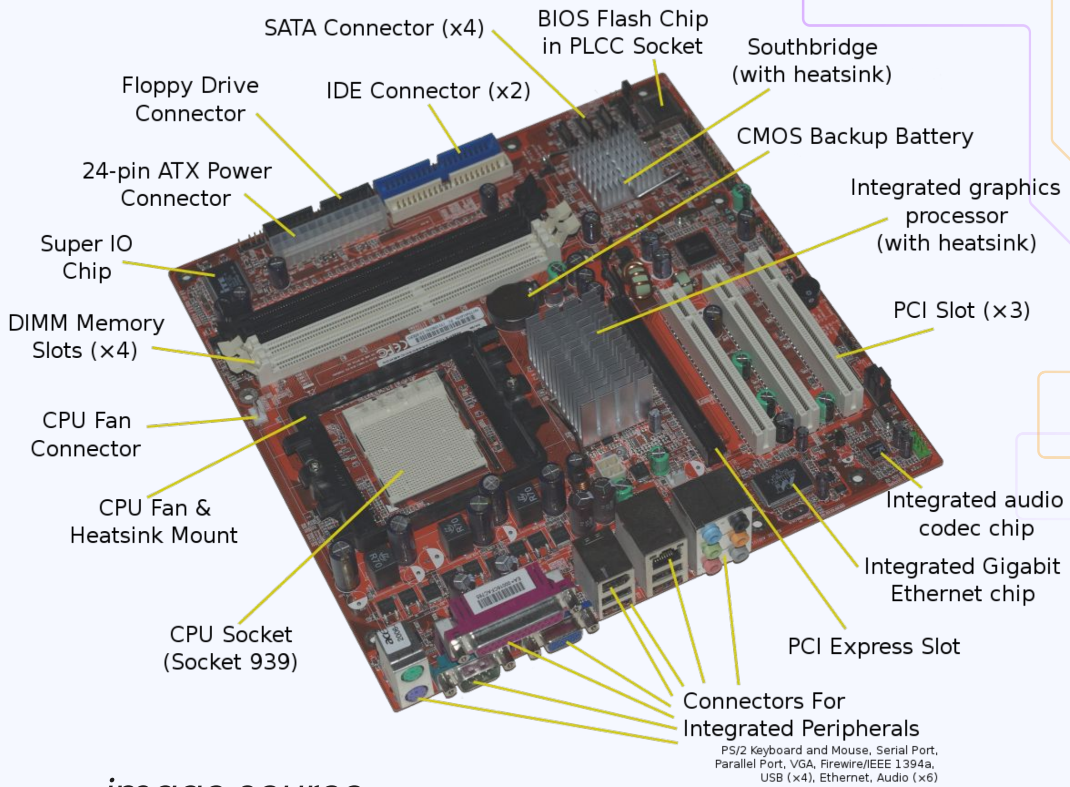

SATA Connector (x4)
BIOS Flash Chip in PLCC Socket
Southbridge (with heatsink)
Floppy Drive Connector
IDE Connector (x2)
CMOS Backup Battery
24-pin ATX Power Connector
Integrated graphics processor (with heatsink)
Super IO Chip
PCI Slot (×3)
DIMM Memory Slots (×4)
CPU Fan Connector
CPU Fan & Heatsink Mount
Integrated audio codec chip
Integrated Gigabit Ethernet chip
CPU Socket (Socket 939)
PCI Express Slot
Connectors For Integrated Peripherals
PS/2 Keyboard and Mouse, Serial Port, Parallel Port, VGA, Firewire/IEEE 1394a, USB (×4), Ethernet, Audio (×6)

image source

image source

Chipsets (1990s-2000s)

Northbridge: connects CPU, RAM, GPU

Southbridge: slower, connects I/O

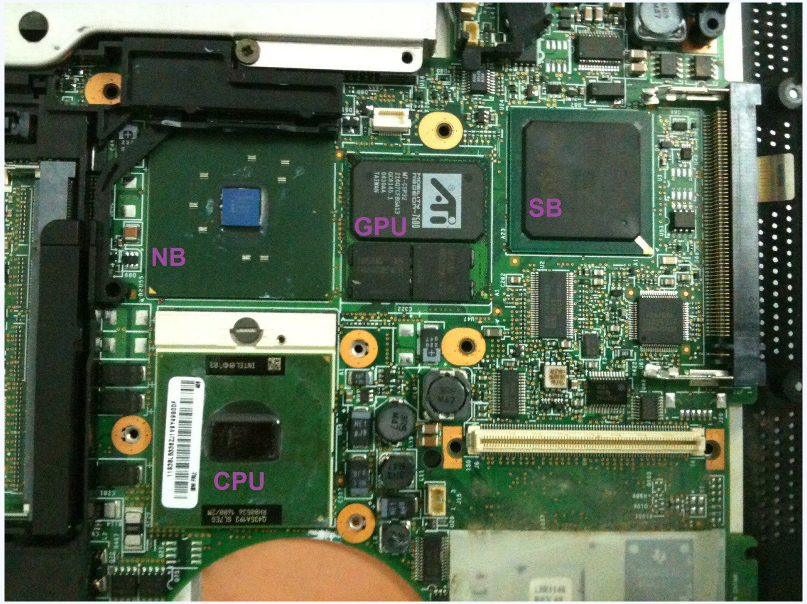
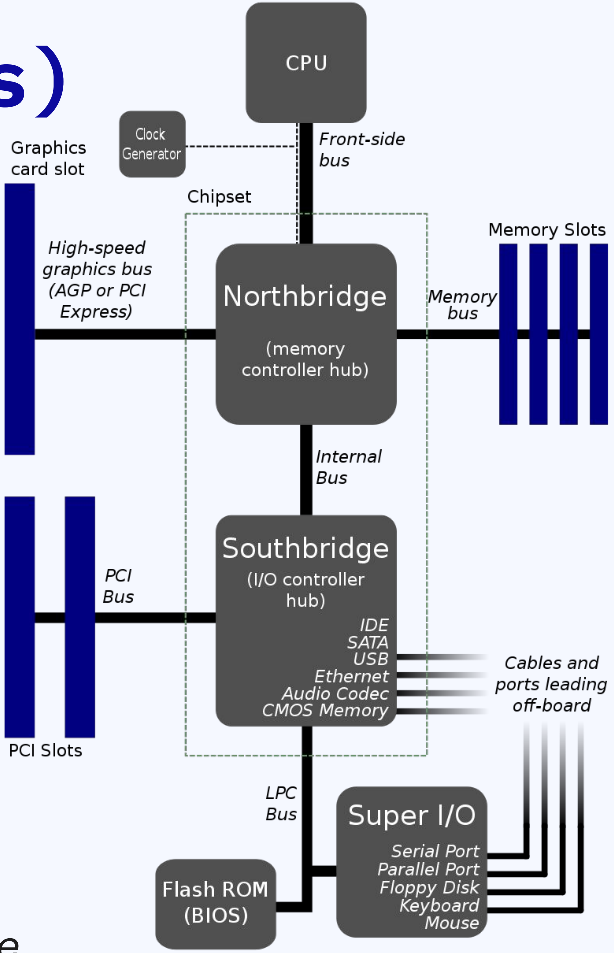


image source



Evolution of chipsets (Intel)

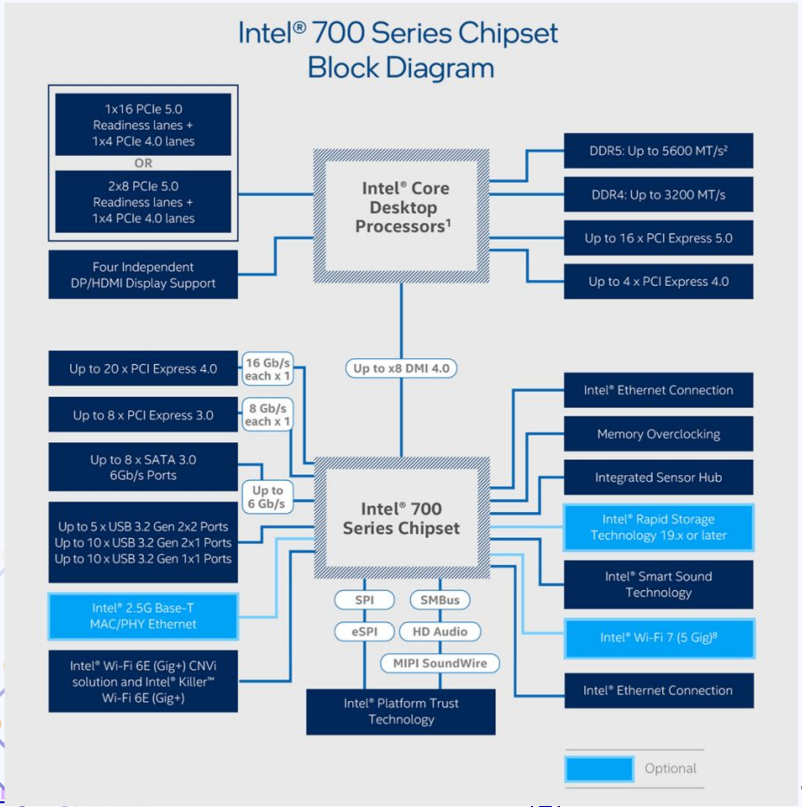
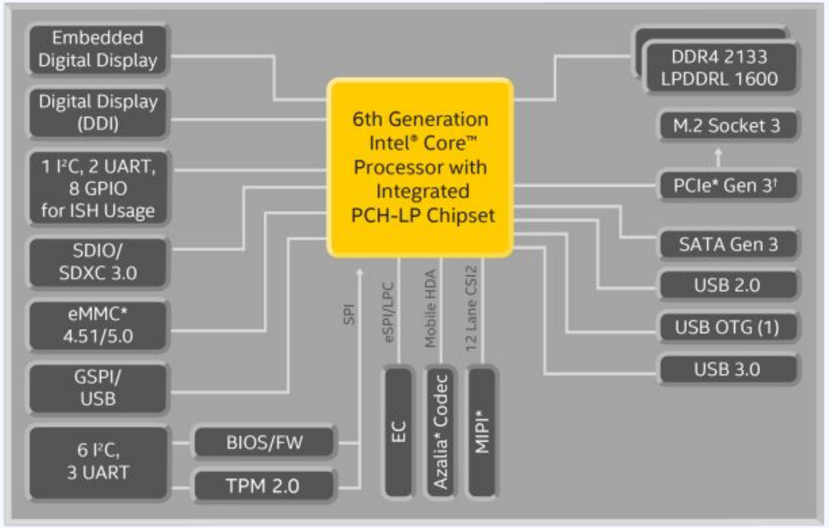

Intel® 700 Series Chipset Block Diagram
1x16 PCIe 5.0 Readiness lanes + 1x4 PCIe 4.0 lanes
OR
2x8 PCIe 5.0 Readiness lanes + 1x4 PCIe 4.0 lanes
Four Independent DP/HDMI Display Support
Intel® Core Desktop Processors¹
DDR5: Up to 5600 MT/s²
DDR4: Up to 3200 MT/s
Up to 16 x PCI Express 5.0
Up to 4 x PCI Express 4.0
Up to x8 DMI 4.0
Up to 20 x PCI Express 4.0
16 Gb/s each x 1
Up to 8 x PCI Express 3.0
8 Gb/s each x 1
Up to 8 x SATA 3.0 6Gb/s Ports
Up to 6 Gb/s
Up to 5 x USB 3.2 Gen 2x2 Ports
Up to 10 x USB 3.2 Gen 2x1 Ports
Up to 10 x USB 3.2 Gen 1x1 Ports
Intel® 2.5G Base-T MAC/PHY Ethernet
Intel® Wi-Fi 6E (Gig+) CNVi solution and Intel® Killer™ Wi-Fi 6E (Gig+)
Intel® 700 Series Chipset
SPI
SMBus
eSPI
HD Audio
MIPI SoundWire
Intel® Platform Trust Technology
Intel® Ethernet Connection
Memory Overclocking
Integrated Sensor Hub
Intel® Rapid Storage Technology 19.x or later
Intel® Smart Sound Technology
Intel® Wi-Fi 7 (5 Gig)⁸
Intel® Ethernet Connection
Optional

image source



† Gen 3.0 available on premium PCH only. Gen 2.0 available on base PCH SKUs.

image source

Buses

There are many ways to transfer data

Different bus standards are used for different applications

SATA is used for storage (also **NVME**)

PCI/e is used for many things (GPU, sound, ethernet…)

SPI is used for serial embedded communication

We're not going to memorize these technologies – just know that there are different protocols for them

? ? ?

How should an I/O event (such as a keyboard key press) be detected and handled by the computer?

Polling

Process of checking the status of an I/O device to determine the need to service the device (P&H chapter 6)

Contrast with interrupts (in a few slides), which detect change in status and automatically disrupt execution

- Real-world example: checking app vs. push notification

? ? ?

How do we check the status/change the state of an I/O device?

Memory-mapped I/O and DMA

Memory-mapped I/O: translation of some memory addresses to I/O status and control registers

- CPU writes/reads that address as usual, gets info about device
- Kernel-space (not user-space) addresses

Cumbersome idea: use processor to transfer data from user space to memory-mapped I/O space

More efficient idea: **Direct Memory Access** (DMA), which transfers data to and from memory without going through the CPU, using a special controller

- Example: transfer page from disk

Interrupts

Exception: an unscheduled event that disrupts program execution (P&H chapter 3)

- Can come from SW, like the RISC-V ecall/ebreak instructions to invoke OS/debugger
- Can come from HW, like a divide by 0 or page fault

Interrupt: an exception that comes from outside the CPU (like a DMA I/O interrupt!)

Something needs to be responsible for saving processor state when an exception/interrupt happens!

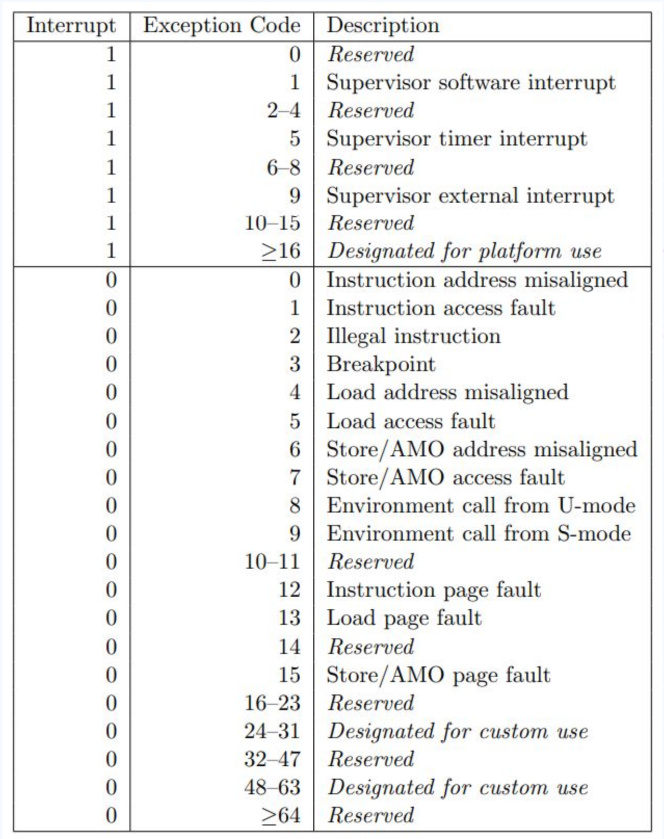
Warning: we are following P&H, but this terminology is not always consistent (if you took 1600, we said SW/HW interrupts and exceptions)

RISC-V approach: SEPC and SCAUSE

Need to put cause in SCAUSE register before transferring control to OS (by going to pre-determined PC)

For HW exceptions: put address of exception-causing instr in SEPC register, before immediately transferring control to OS

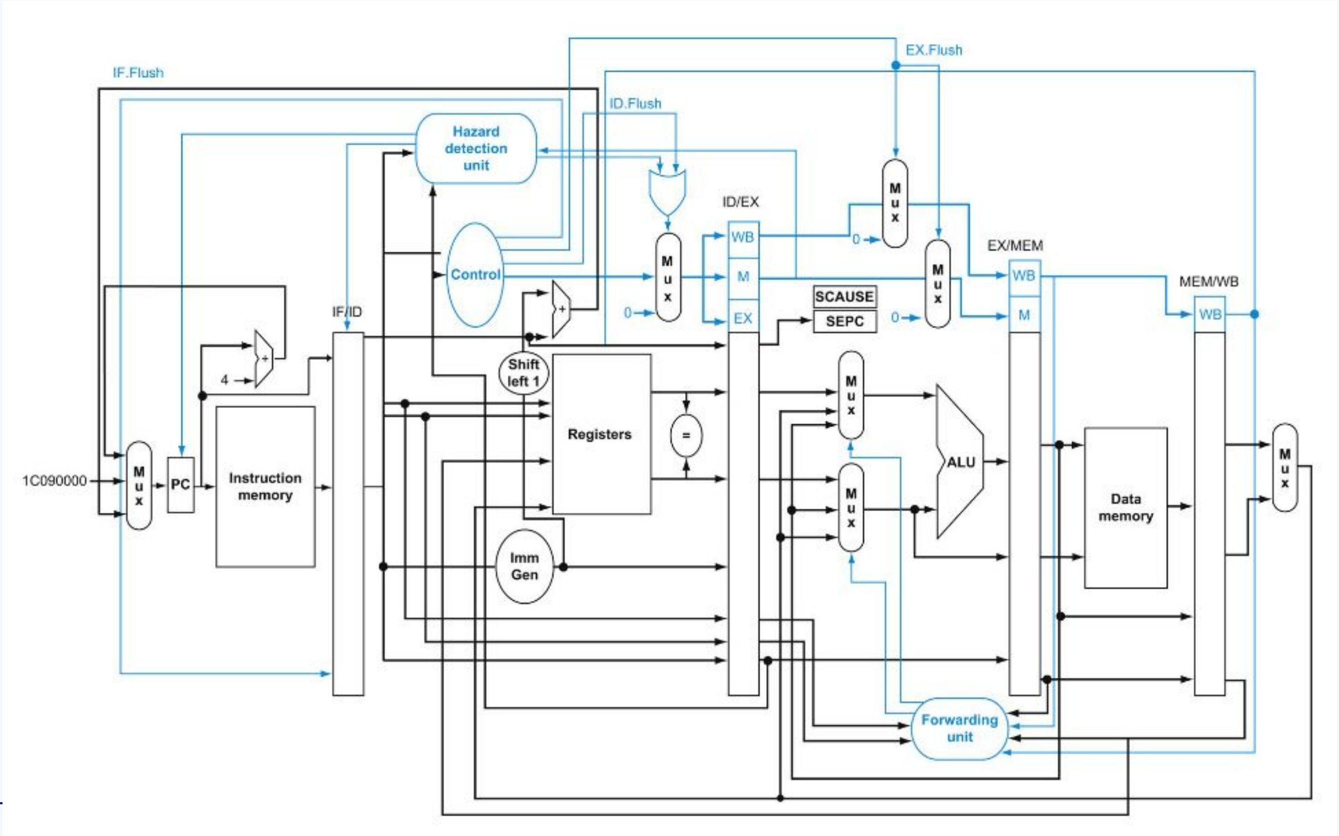
For I/O interrupts: handled asynchronously (not caused by instr), so current instr can finish and control unit checks for pending interrupt in next cycle



Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2–4	*Reserved*
1	5	Supervisor timer interrupt
1	6–8	*Reserved*
1	9	Supervisor external interrupt
1	10–15	*Reserved*
1	≥16	*Designated for platform use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10–11	*Reserved*
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–23	*Reserved*
0	24–31	*Designated for custom use*
0	32–47	*Reserved*
0	48–63	*Designated for custom use*
0	≥64	*Reserved*

RISC-V spec v2 Table 4.2

P&H Fig. 4.63



IF.Flush
ID.Flush
EX.Flush
Hazard detection unit
Control
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
SCAUSE
SEPC
Mux
0
4
1C090000
PC
Instruction memory
Shift left 1
Registers
=
Imm Gen
ALU
Data memory
Forwarding unit