I/O and exceptions

? ? ?

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

Abstraction of I/O

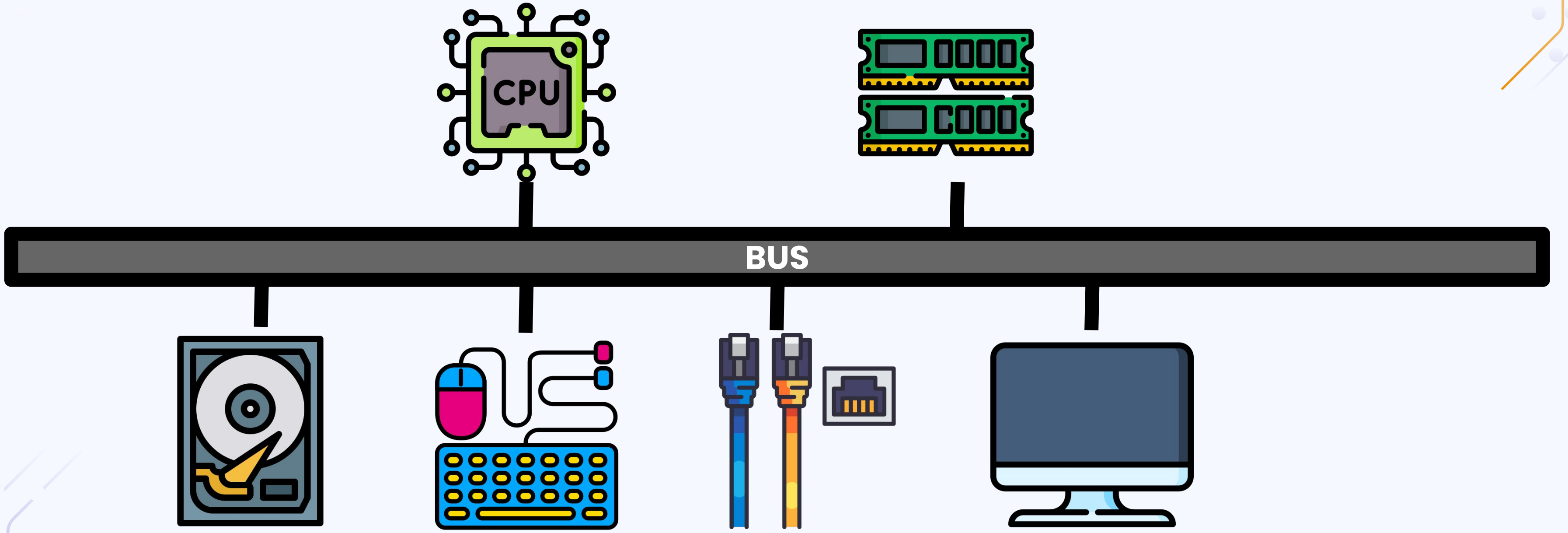
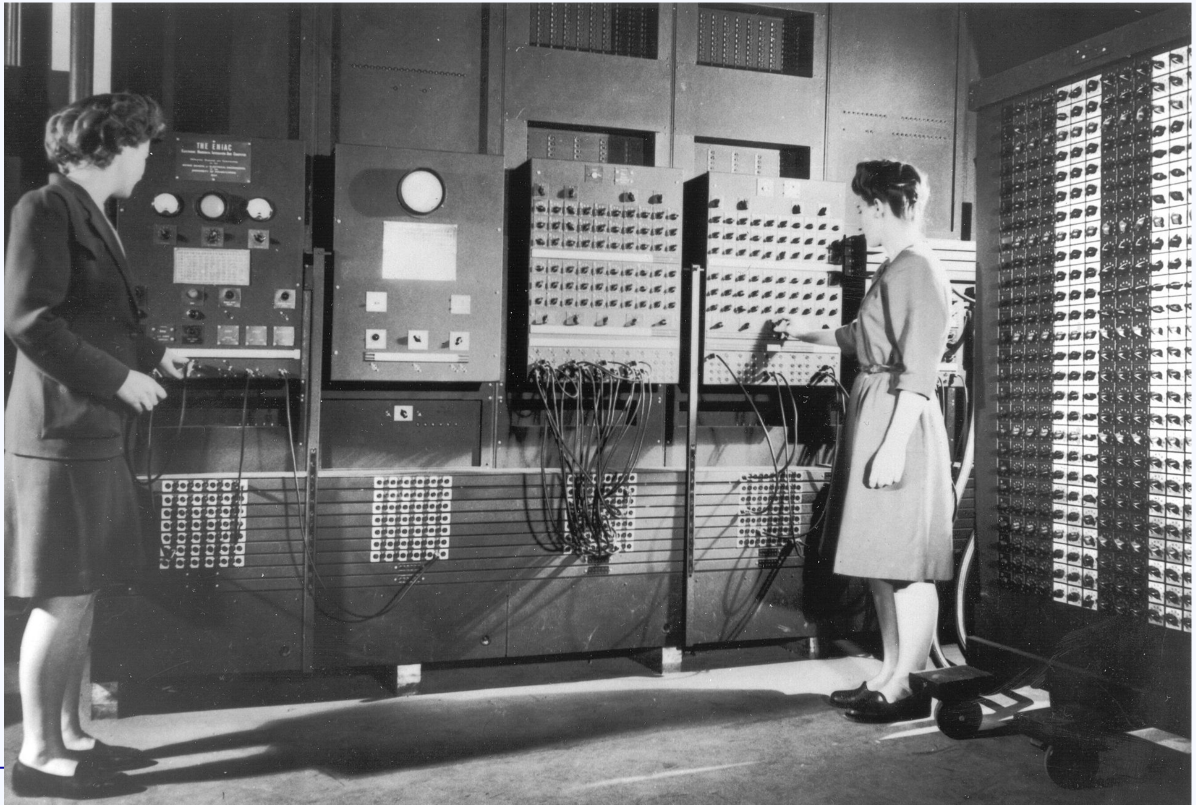


image source: flaticon.com

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



History of I/O: UNIVAC 1 (1951)

image source

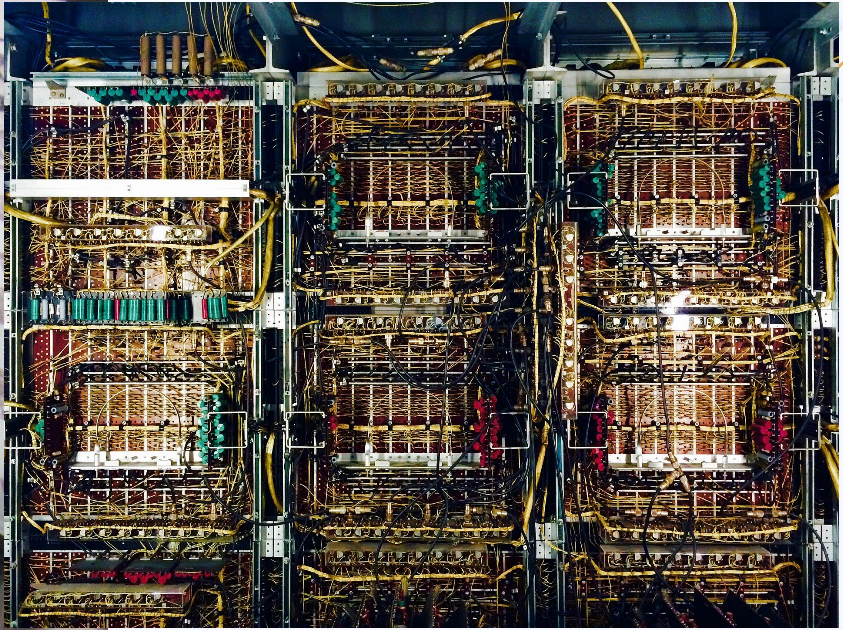


image source

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

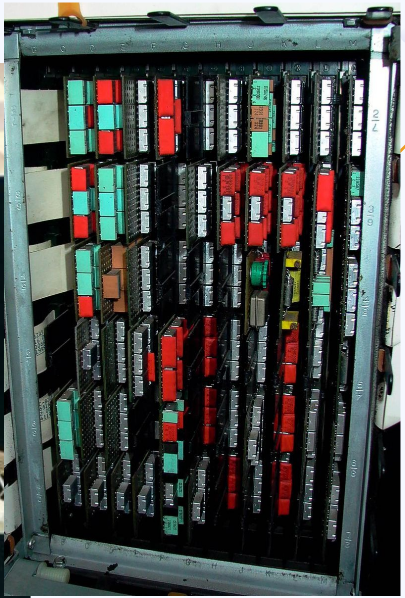


image source

image source

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

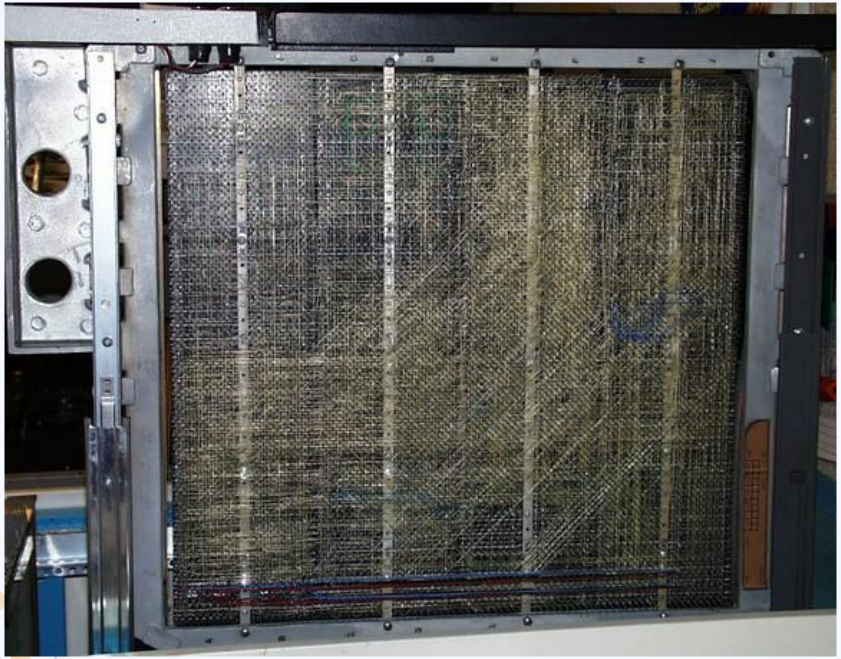
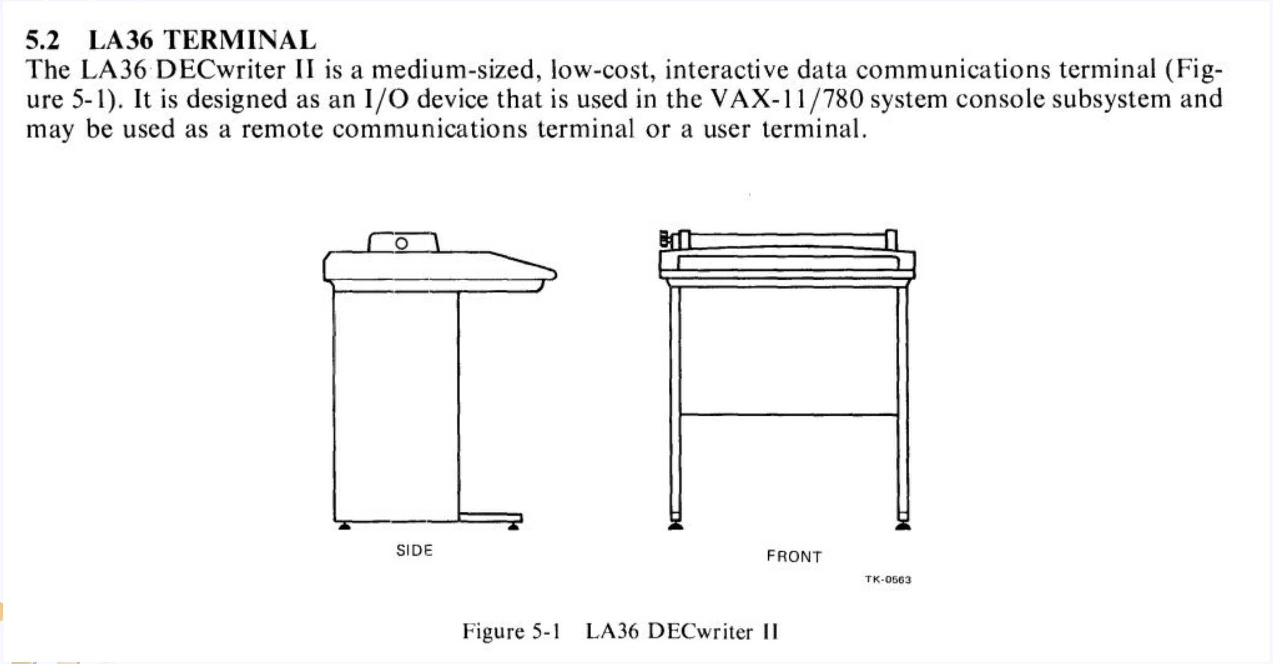


image source

image source

VAX11: Successor to the PDP (1977)



5.2 LA36 TERMINAL

The LA36 DECwriter II is a medium-sized, low-cost, interactive data communications terminal (Figure 5-1). It is designed as an I/O device that is used in the VAX-11/780 system console subsystem and may be used as a remote communications terminal or a user terminal.

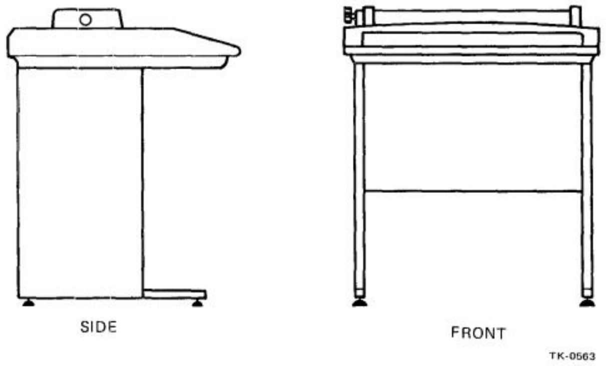


Figure 5-1 LA36 DECwriter II

source (VAX11 hardware user's guide)

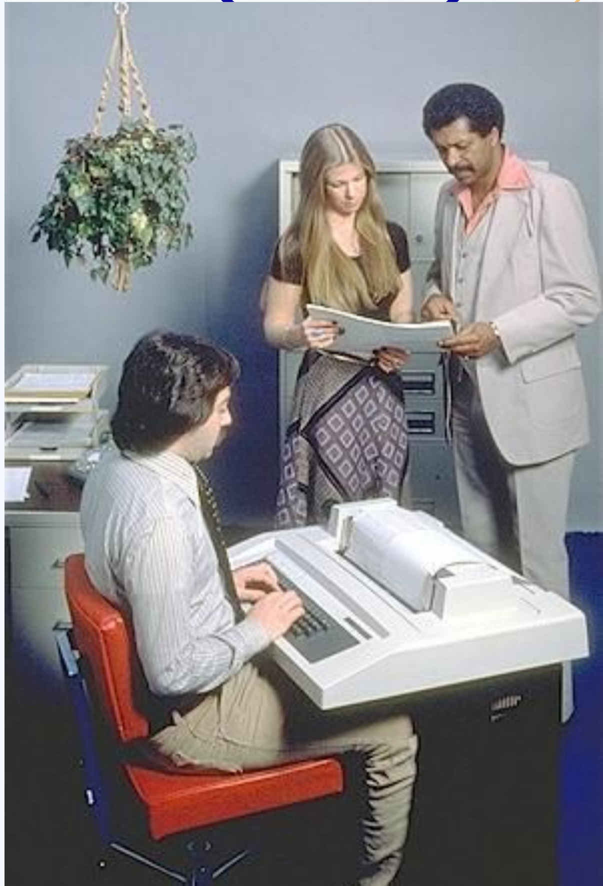
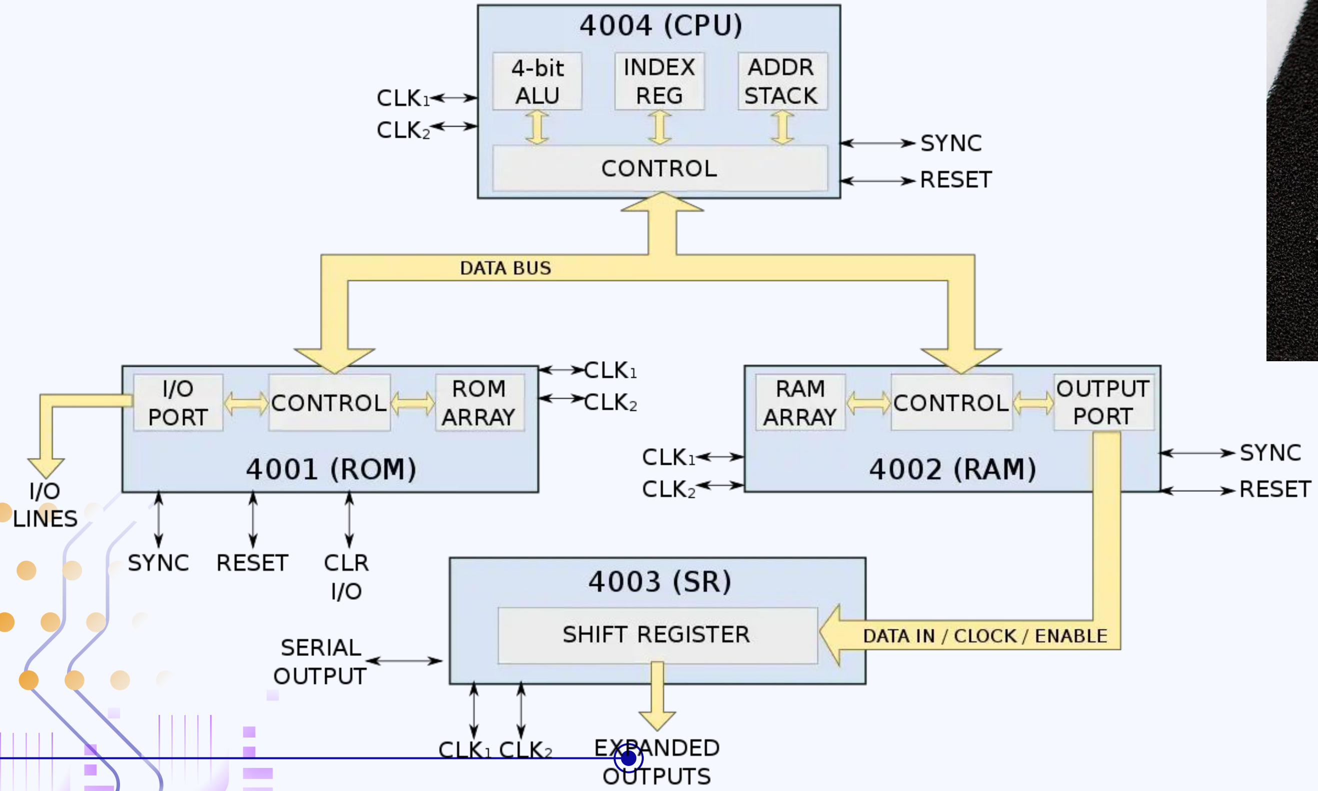


image source

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



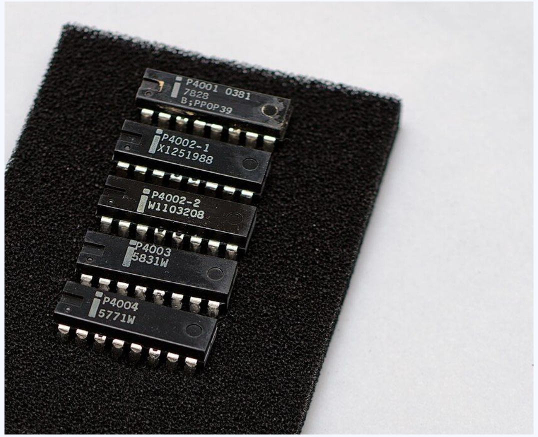
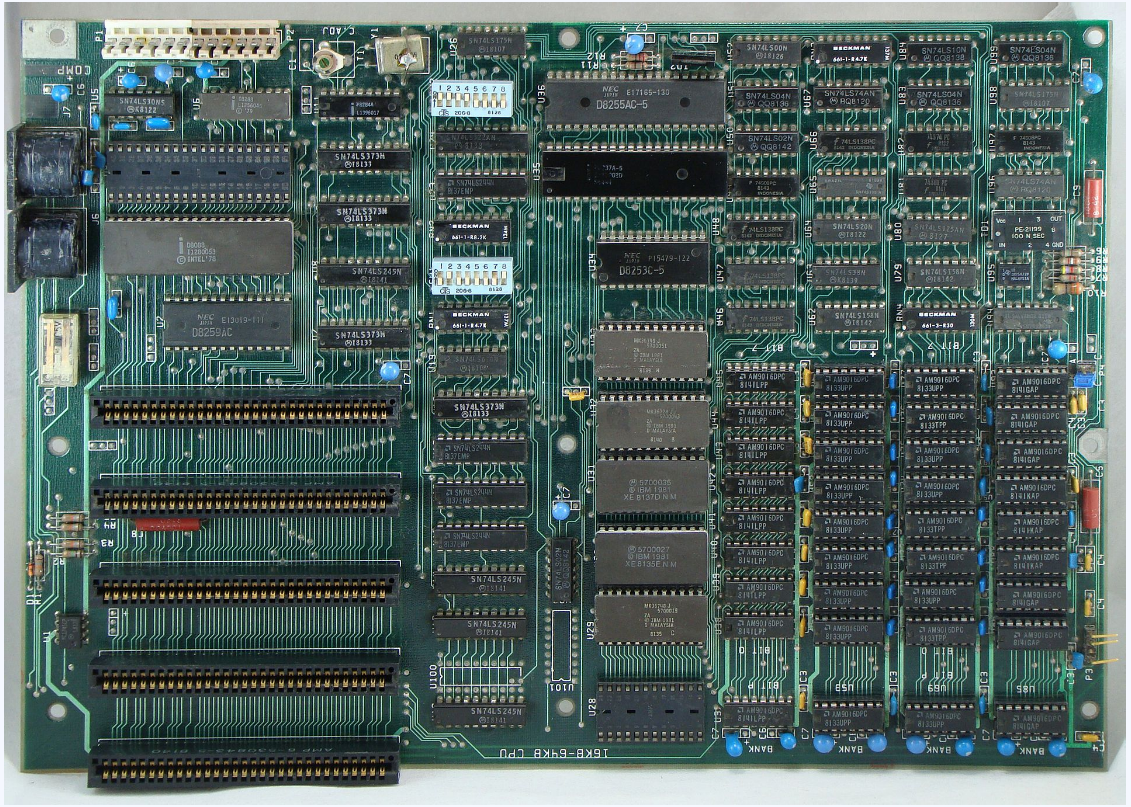


image source

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

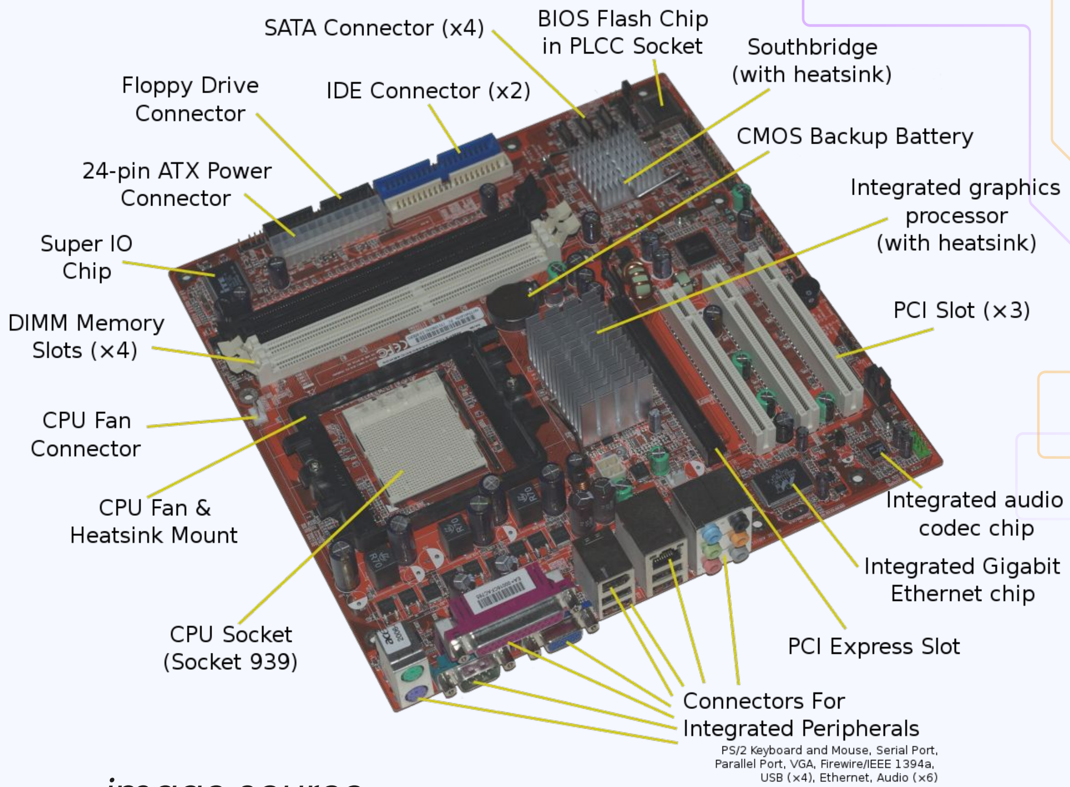

SATA Connector (x4)
BIOS Flash Chip in PLCC Socket
Southbridge (with heatsink)
Floppy Drive Connector
IDE Connector (x2)
CMOS Backup Battery
24-pin ATX Power Connector
Integrated graphics processor (with heatsink)
Super IO Chip
DIMM Memory Slots (×4)
PCI Slot (×3)
CPU Fan Connector
CPU Fan & Heatsink 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

Chipsets (1990s-2000s)

Northbridge: connects CPU, RAM, GPU

Southbridge: slower, connects I/O

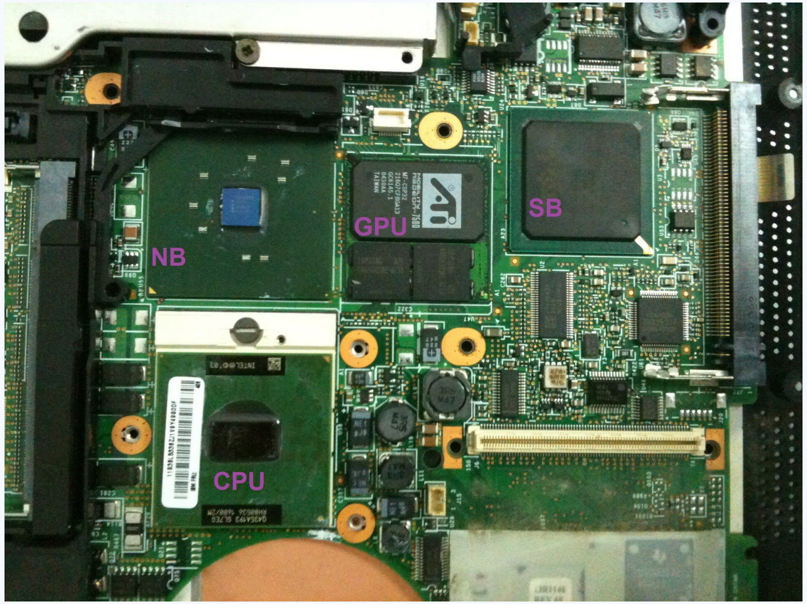
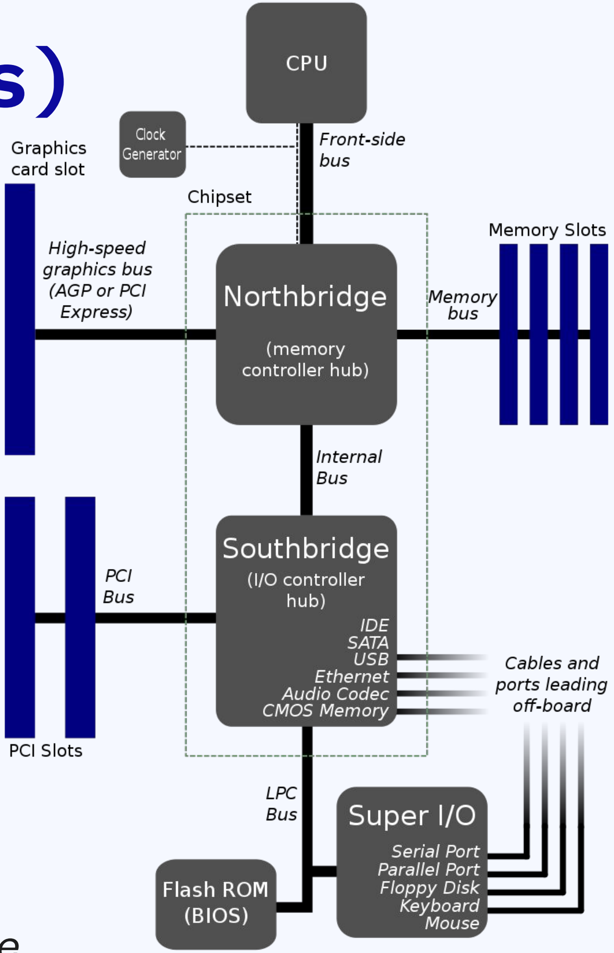
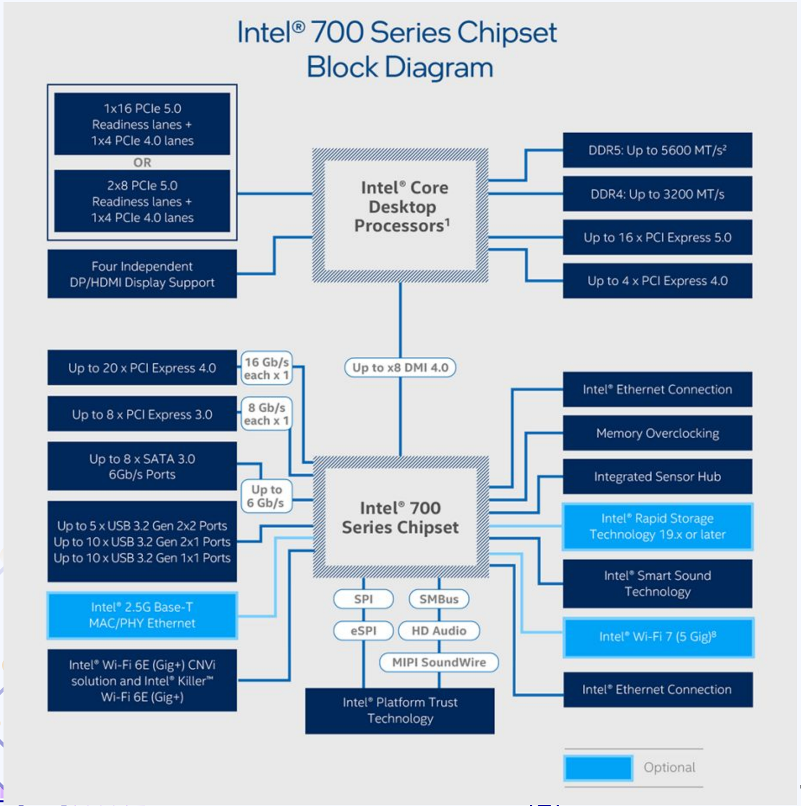
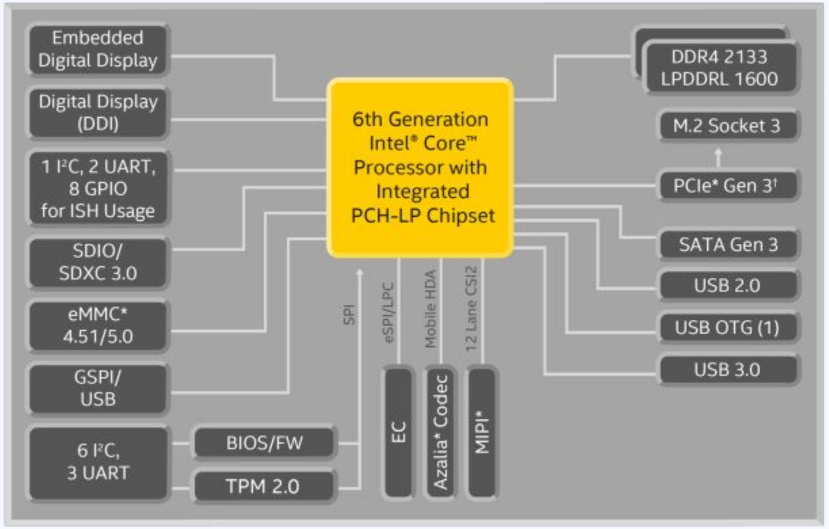


image source



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



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

image source

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 does a CPU access an I/O bus? There’s no lio or sio instruction in RISCV…

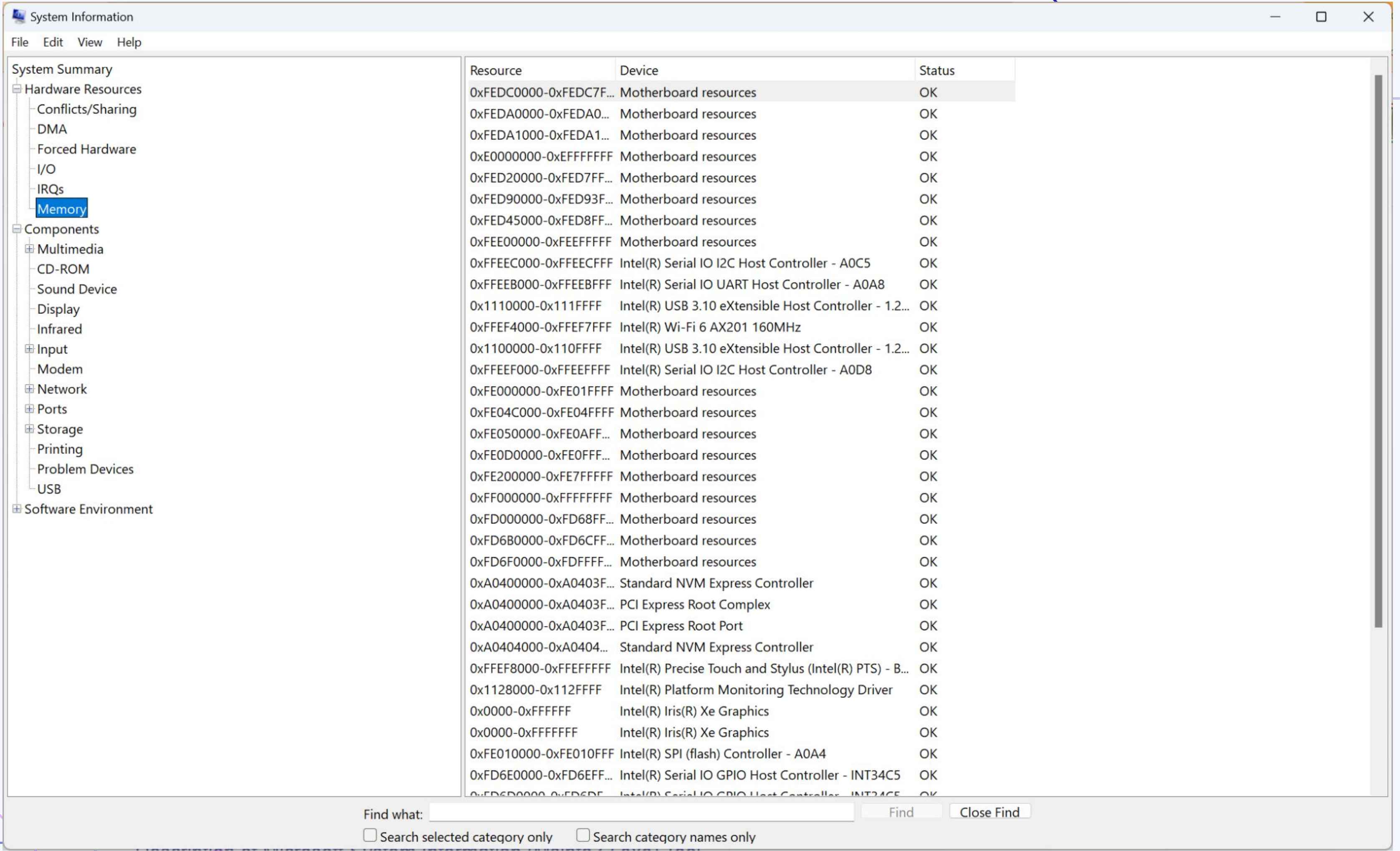
Memory-mapped I/O

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

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

Contrast with port-mapped I/O: special instructions to access I/O ports

- x86 `in` and `out` instructions (x86 supports both port- and memory-mapped I/O; see chapter 19 of the Intel 64 manual)



Resource	Device	Status
0xFEDC0000-0xFEDC7F...	Motherboard resources	OK
0xFEDA0000-0xFEDA0...	Motherboard resources	OK
0xFEDA1000-0xFEDA1...	Motherboard resources	OK
0xE0000000-0xEFFFFFFF	Motherboard resources	OK
0xFED20000-0xFED7FF...	Motherboard resources	OK
0xFED90000-0xFED93F...	Motherboard resources	OK
0xFED45000-0xFED8FF...	Motherboard resources	OK
0xFEE00000-0xFEEFFFFF	Motherboard resources	OK
0xFFEEC000-0xFFEECFFF	Intel(R) Serial IO I2C Host Controller - A0C5	OK
0xFFEEB000-0xFFEEBFFF	Intel(R) Serial IO UART Host Controller - A0A8	OK
0x1110000-0x111FFFF	Intel(R) USB 3.10 eXtensible Host Controller - 1.2...	OK
0xFFEF4000-0xFFEF7FFF	Intel(R) Wi-Fi 6 AX201 160MHz	OK
0x1100000-0x110FFFF	Intel(R) USB 3.10 eXtensible Host Controller - 1.2...	OK
0xFFEEF000-0xFFEEFFFF	Intel(R) Serial IO I2C Host Controller - A0D8	OK
0xFE000000-0xFE01FFFF	Motherboard resources	OK
0xFE04C000-0xFE04FFFF	Motherboard resources	OK
0xFE050000-0xFE0AFF...	Motherboard resources	OK
0xFE0D0000-0xFE0FFF...	Motherboard resources	OK
0xFE200000-0xFE7FFFFF	Motherboard resources	OK
0xFF000000-0xFFFFFFFF	Motherboard resources	OK
0xFD000000-0xFD68FF...	Motherboard resources	OK
0xFD6B0000-0xFD6CFF...	Motherboard resources	OK
0xFD6F0000-0xFDFFFF...	Motherboard resources	OK
0xA0400000-0xA0403F...	Standard NVM Express Controller	OK
0xA0400000-0xA0403F...	PCI Express Root Complex	OK
0xA0400000-0xA0403F...	PCI Express Root Port	OK
0xA0404000-0xA0404...	Standard NVM Express Controller	OK
0xFFEF8000-0xFFEFFFFF	Intel(R) Precise Touch and Stylus (Intel(R) PTS) - B...	OK
0x1128000-0x112FFFF	Intel(R) Platform Monitoring Technology Driver	OK
0x0000-0xFFFFFF	Intel(R) Iris(R) Xe Graphics	OK
0x0000-0xFFFFFFF	Intel(R) Iris(R) Xe Graphics	OK
0xFE010000-0xFE010FFF	Intel(R) SPI (flash) Controller - A0A4	OK
0xFD6E0000-0xFD6EFF...	Intel(R) Serial IO GPIO Host Controller - INT34C5	OK

Bypassing CPU for memory access

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

Supervisor transfers pages from disk using 1k lw + sw instructions

Direct Memory Access (DMA) transfers data to and from memory without going through the CPU, using the controller of the I/O device

PCIe uses an optimized version of DMA that involves bus arbitration

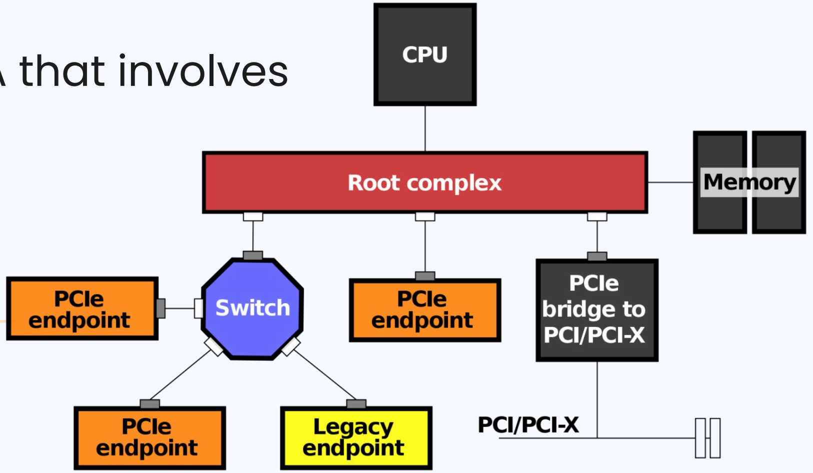


image source

? ? ?

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

Polling

Manually check the status of I/O device periodically

Pros:

Check message app over and over for your crush to text you

Interrupts

Disrupt CPU execution when an I/O operation happens

Pros:

Get push notification that your crush has texted you

Exceptions vs Interrupts

Exception: unscheduled event that disrupts execution (P&H chapter 4)

SW source: RISC-V ecall/ebreak instructions to invoke supervisor/debugger

HW source: divide by 0 or page fault

Interrupt: an exception that comes from outside the CPU

DMA I/O interrupt

Warning: terminology changes depending on textbook/context

What does the CPU hardware need to do when an exception happens?

Handling exceptions

1) Figure out if we need to stop immediately or finish current instruction

   Hardware exceptions: immediately (can’t proceed after dividing by 0)

   I/O interrupts: handle asynchronously (finish instruction)

2) Save processor state and provide exception info to supervisor
3) Switch control to supervisor
4) Go to PC of exception-handling routine
5) Handle exception
6) Switch back

RISC-V exception registers

SEPC holds address of exception-causing instr

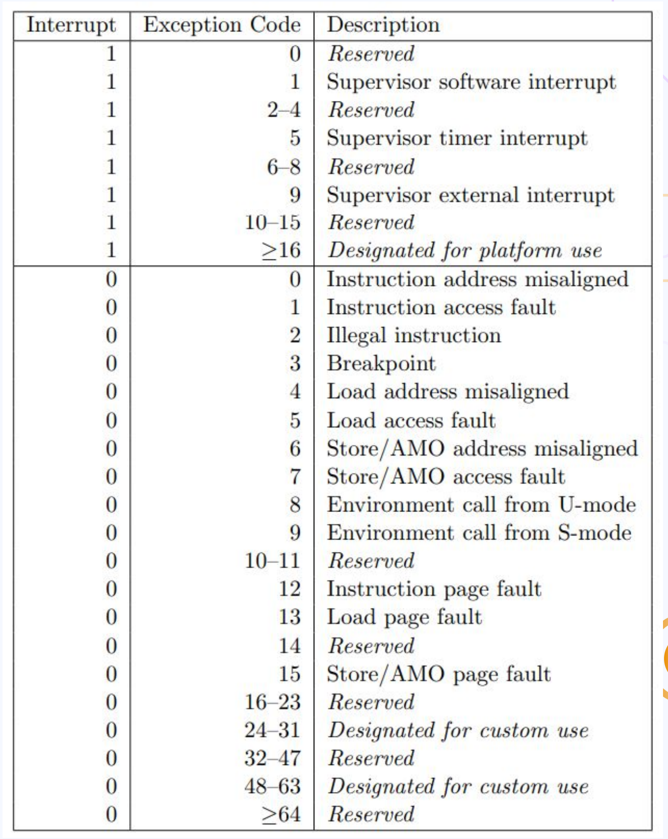
SCAUSE holds the cause

STVAL holds info about exception (e.g. fault-causing virtual address)

SIP holds pending status of potential interrupts

STVEC holds address of supervisor exception-handler

Alternative approach: vectored interrupts (arm)



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

Handling exceptions

1) Figure out if we need to stop immediately or finish current instruction

   Hardware exceptions: immediately (can’t proceed after dividing by 0)

   I/O interrupts: handle asynchronously (finish instruction) **SIP**

2) Save processor state and provide exception info to supervisor

   SCAUSE, STVAL, SEPC

3) Switch control to supervisor
4) Go to PC of exception-handling routine **STVEC**
5) Handle exception **(uses info stored in SCAUSE, STVAL)**
6) Switch back **SRET instruction (uses info stored in SEPC)**