Lecture 2 — Review of Computer Architecture

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

May 3, 2015

Computer Organization

A regular program like a word processor need not be concerned with the underlying hardware of the compute.

The OS must be aware of the details and manage them.

What is a program? Instructions and data.

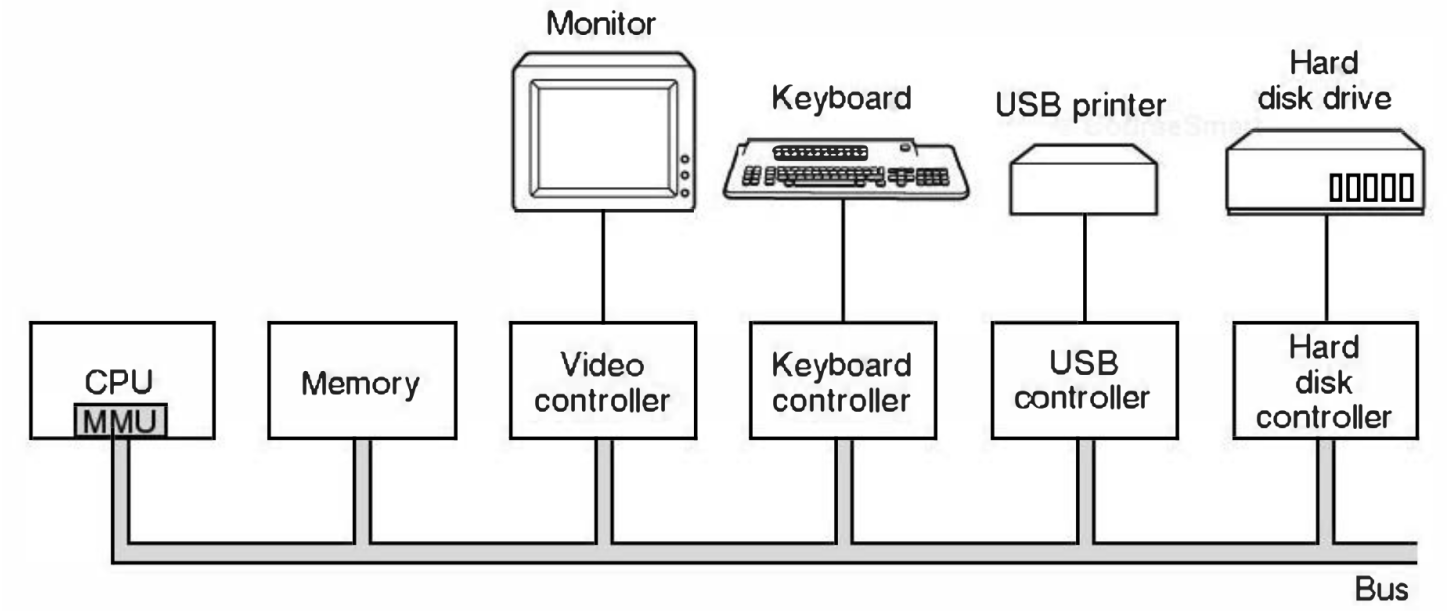
To Execute a Program

To execute a program we need:

1. **Main Memory**
2. **System Bus**
3. **Processor**

Of course, this is the minimal set.

Modern Personal Computer



A fourth element: **Input/Output** (I/O). Not strictly necessary, but a computer without it is not very useful...

Main Memory

Ideally, memory would be:

- Fast enough that the processor never has to wait;
- Large enough to hold all the data;
- Inexpensive.

The Iron Triangle: “fast, good, cheap; pick two.”

Good news: we can have different levels of memory.

Main Memory

Different levels of memory at different sizes, speed, and cost.
So what we end up with is a hierarchy of memory.

Let us compare the various levels I might have in my laptop from 2013:

Memory Level	**Access Time**	**Total Capacity**
Register	1 ns	< 1 KB
Cache	2 ns	6 MB
Main Memory (RAM)	10 ns	16 GB
Magnetic Hard Disk	10 ms	500 GB

Fast memory is expensive!

The difference in access time is often quite dramatic.

Memory Access Analogy

I am the CPU and a particular book is the piece of data needed.

If the data is in the cache: the book is on a bookshelf in my office.

If the data for the CPU on a magnetic hard disk, I have to get the book from Library and Archives Canada in Ottawa (550 km away).
And I would have to walk.

The CPU doesn't go get the data; instead it must wait for it to arrive.

What might I do in the meantime...?

System Buses

The bits on the bus go up and down, up and down, up and down...

System Buses

Every sort of communication using the same bus.
    Contention for this resource is a limiting factor.

The original IBM PC did work like that.
    A modern system has numerous buses.

Central Processing Unit (CPU)

The processor (CPU) is the brain of the computer.

Fetch instructions, decode them, execute them.

Fetch-decode-execute cycle repeated until the program finishes.

Different steps may be completed in parallel (pipeline).

Processors' largest unit is the word.
32-bit computer $\rightarrow$ 32-bit word. 64-bit computer $\rightarrow$ 64-bit word.

Central Processing Unit (CPU)

CPU instructions are specific to the processor.

Written assembly? You know the books.

Some operations only available in supervisor mode.
    Attempting to run it in user mode is an error.

Central Processing Unit (CPU)

CPUs have storage locations: registers.

They may store data or instructions.

Management of registers is partly the role of the OS.

Let us examine a few of the critical registers.

CPU Registers

A few of the registers in a typical CPU:

- **Program Counter** – Next instruction.
- **Status Register** – Array of bits to indicate flags.
- **Instruction Register** – Instruction most recently fetched.
- **Stack Pointer** – Top of the stack.
- **General Purpose Registers** – Store data, addresses, etc.

Program Execution

Program is a sequence of instructions. We can categorize them as:

1. **Processor-Memory**
2. **Processor-I/O**
3. **Data Processing**
4. **Control**

Interrupts

If I ordered a book from Ottawa, it takes a long time to arrive.

In the meantime, I should do something else.

Polling: check periodically if the book has arrived.

Interrupts: get a notification when the book is here.

If someone knocks on my door, I pause what I'm doing and get the book.

Sources of Interrupts

We can put interrupts into four categories, based on their origin:

1. **Program.**
2. **Timer.**
3. **Input/Output.**
4. **Hardware Failure.**

Interrupts

Interrupts are a way to improve processor utilization.

CPU time is valuable!

When an interrupt take place, the CPU might ignore it (rarely).

More commonly: we need to handle it in some way.

Analogy: professor in a lecture; student has a question.

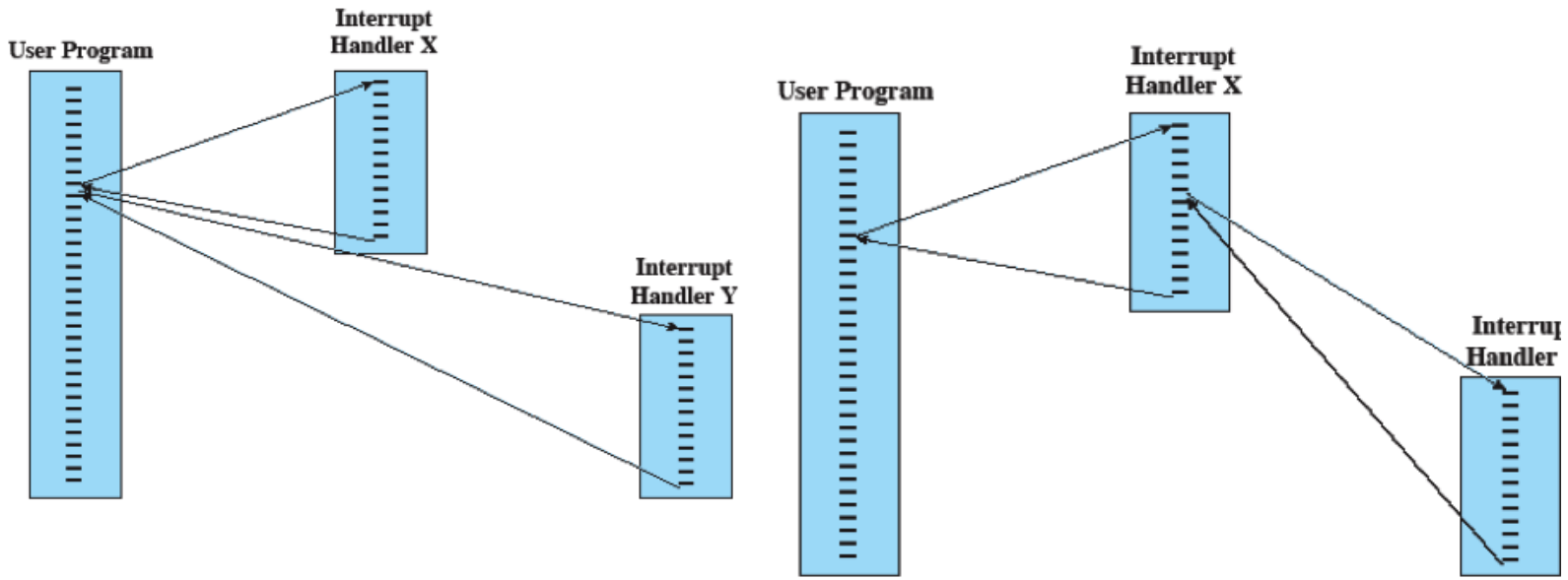
Interrupts

The OS: stores the state, handles the interrupt, and restores the state.

Sometimes the CPU is in the middle of something uninterruptible.
    Interrupts may be disabled (temporarily).

Interrupts can have different priorities.

We may also have multiple interrupts in a short period of time:



They may be sequential (left) or nested (right).

A combination may be used.

Storing and Restoring State

The OS must store the program state when an interrupt occurs.

The state must be stored.

State: values of registers.

Push them onto the stack.

Interrupt finished: restore the state (pop off the stack).

Then execution continues.

That is saving and restoring the same program.

Why not restore a different program?

We will come to this in scheduling.

I/O Communication

Three major strategies for communication:

1. **Programmed I/O.** – Polling.
2. **Interrupt Driven I/O.** – Interrupts.
3. **Direct Memory Access (DMA).** – CPU does setup.

Direct Memory Access

The CPU will do some set up, indicating:

1. The operation to perform (read or write)
2. The source
3. The destination
4. How much data is to be transferred

This data is sent to the DMA module (a delegate).

After that, the CPU can go on to do other work.

The I/O device will interact directly with memory.