Operating Systems in Practice

Delivery

- Two Lectures per week
- Two Hour Weekly Lab
- All material available on Blackboard
  (lecture notes, labs, past exams, etc.)

Operating Systems in Practice

Module Descriptor:

Available on Blackboard

Text Book:

Operating Systems – Internals and Design Principles by

William Stallings (pub. Pearsons)

How Linux Works – What every superuser should know

by Brian Ward (pub. No Starch Press)

Operating Systems in Practice

Lab Environment

Linux Ubuntu –

- can also install on your own computer
- install on a virtual machine
- install on a USB stick

Operating Systems in Practice

Assessment

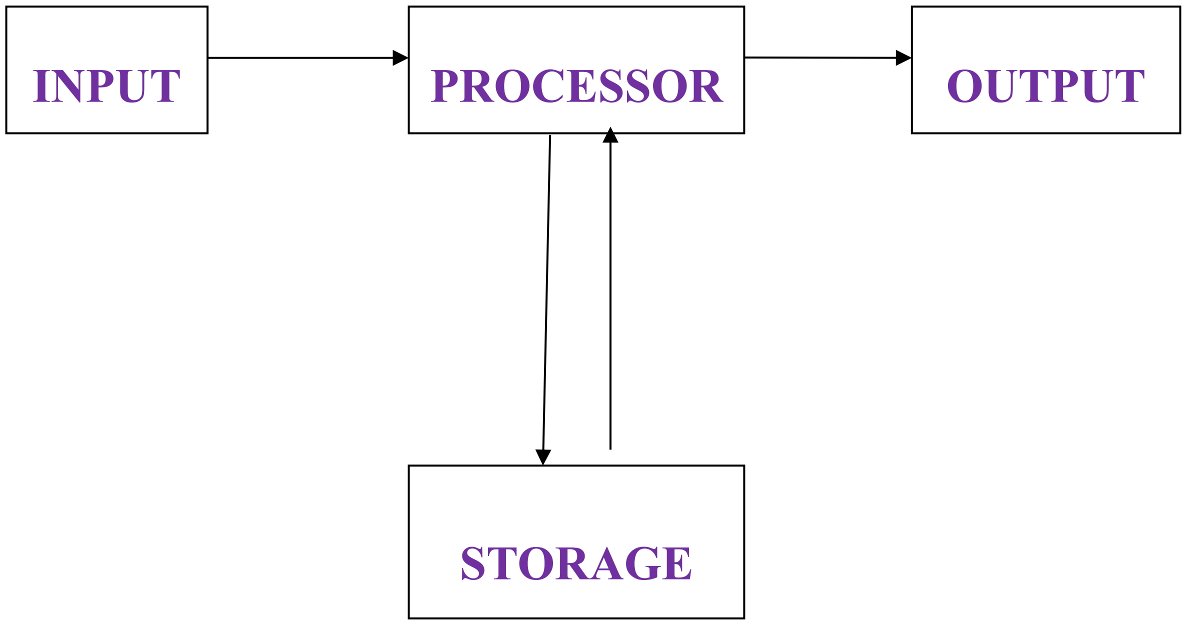
Continuous Assessment	50%
Final Exam	50%

Operating Systems in Practice

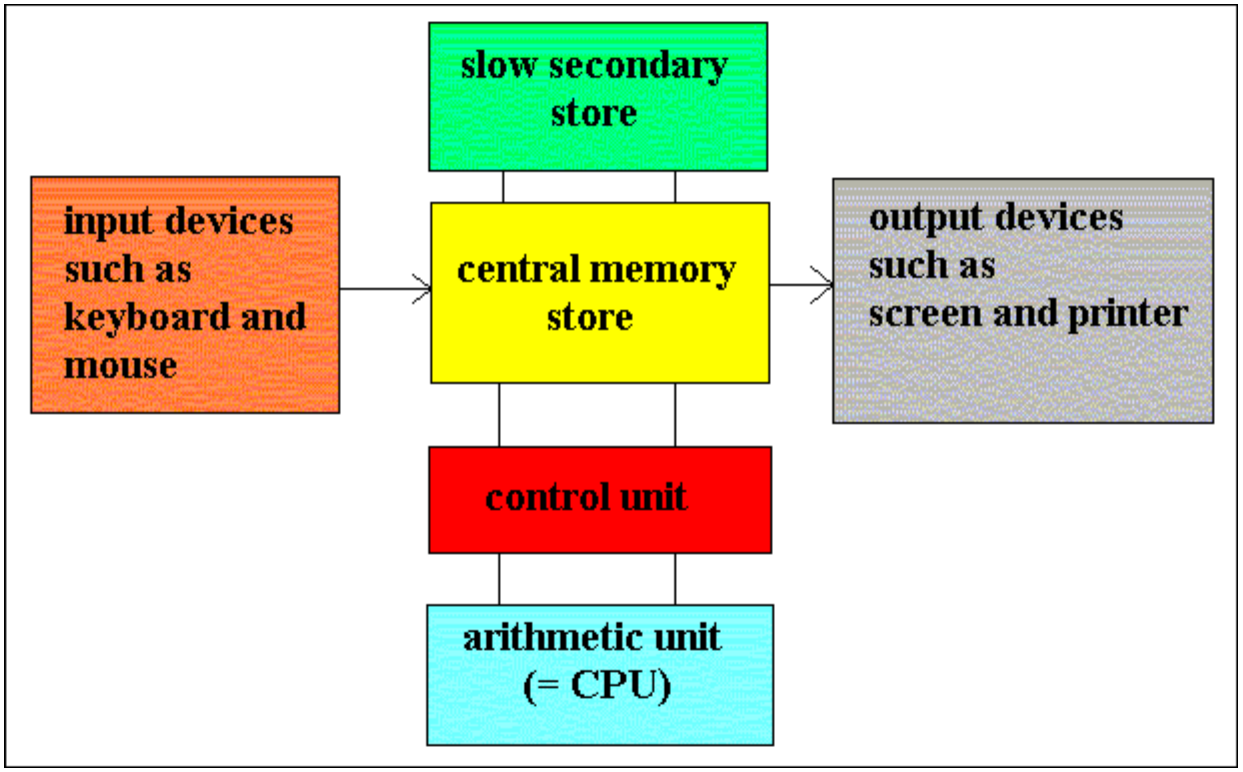
Main Areas of Study

- Overview of Computer System
- Operating System Overview
- Introduction to Linux
- Process Management
- Memory Management
- Storage Management

Structure of a Computer System



Structure of a Computer System


slow secondary store
input devices such as keyboard and mouse
central memory store
output devices such as screen and printer
control unit
arithmetic unit (= CPU)

Basic Elements

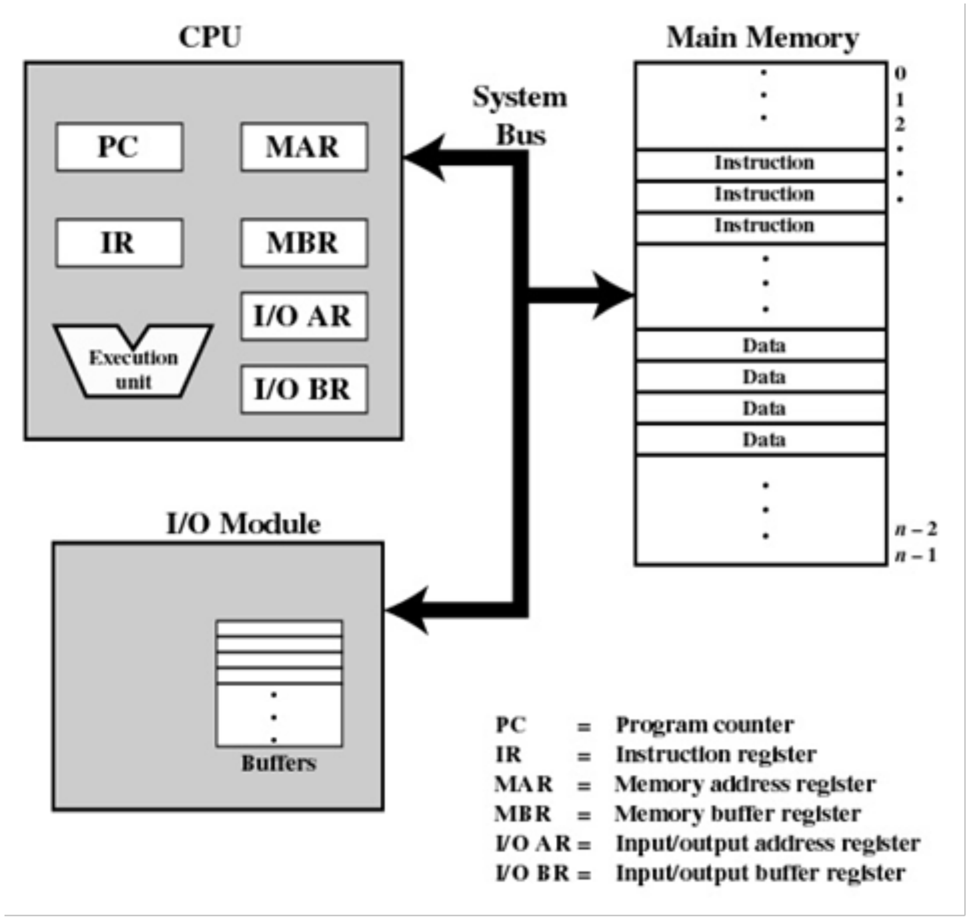
The Processor – often referred to as the central processing unit (CPU)

Main Memory – used to store data and programs. It loses its contents when switched off i.e. it’s volatile.

I/O Modules – enables data to be inputted and outputted.

System bus – Moves data to / from different components.

System Overview



PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

The Processor

- The Central Processor Unit (CPU) describes the circuitry that enables computer programs to be executed. A computer program is a sequence of instructions stored in memory designed to complete an operation.

- The processor can be divided into two major components the **arithmetic logic unit** (A.L.U.) and the **control unit** (C.U.).

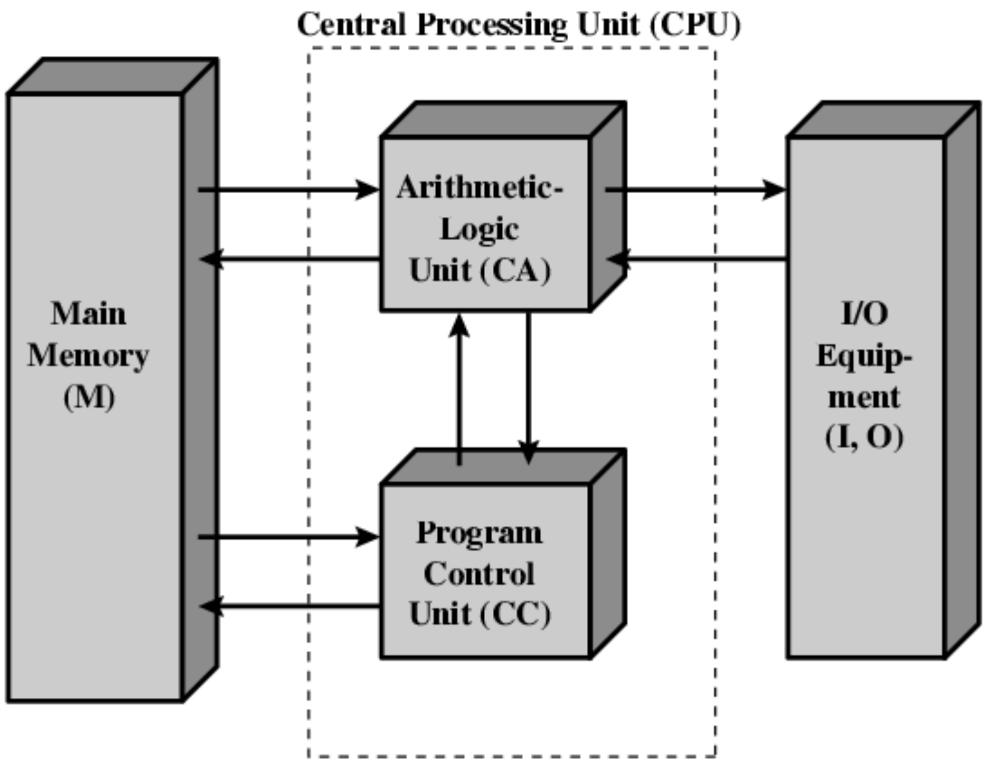
The ALU & CU

- ALU – Arithmetic Logic Unit which performs operations (add, sub, etc.)

- CU – Control Unit which co-ordinates operation of the various CPU components

The Processor

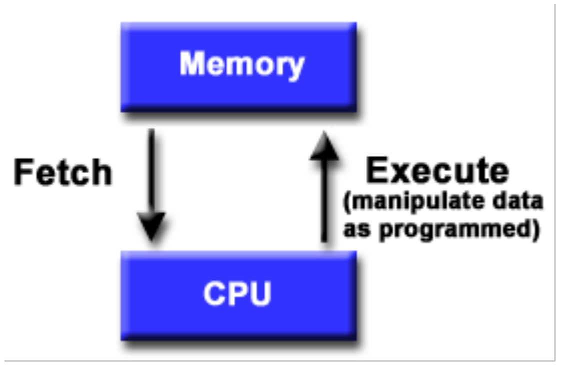
- The CPU is divided into an Arithmetic Logic Unit and a Control Unit.


Central Processing Unit (CPU)
Main Memory (M)
Arithmetic-Logic Unit (CA)
Program Control Unit (CC)
I/O Equip-ment (I, O)

The Processor

- For over 40 years, virtually all computers have followed a common machine model known as the **von Neumann** computer. This was named after the Hungarian mathematician John von Neumann (1903 – 1957).

- A von Neumann computer uses the stored program concept. The CPU fetches and executes instructions from memory**.**



The Processor

- Memory is used to store both the program (i.e. a set of instructions) and the data
- Program instructions are coded data which tell the computer to do something
- Data is simply information to be used by the program
- The central processing unit (CPU) gets instructions and/or data from memory, decodes the instructions and then ***sequentially*** performs them.

Registers

- Registers are memory cores that keep data temporarily while the ALU is processing it. These cores are small but fast. Data is shuttled into the registers when the ALU needs it and out again when the ALU is finished with it.

- Data in the registers is swapped to **RAM (Random Access Memory)** when it is no longer needed. Other forms of memory are slower and so the registers act as a type of buffer between the quick ALU and the slower RAM.

- This means that the ALU can be processing something while data is being swapped. Therefore the ALU does not need to wait as often for the slower components to catch up.

Processor Registers

Registers which store information to control operations, these include:

- **IR** – Instruction Register which contains the instruction last fetched
- **PC** – Program Counter which contains the memory address of the next instruction to be fetched
- **MAR** – Memory Address Register which contains the address for the next read/write
- **MBR** – Memory Buffer Register which contains the data to be written to memory OR contains the data to be read from memory
- **I/OAR** – I/O Address Register which specifies an i/o device
- **I/OBR** – I/O Buffer Register which is used for exchange of data to/from the i/o module

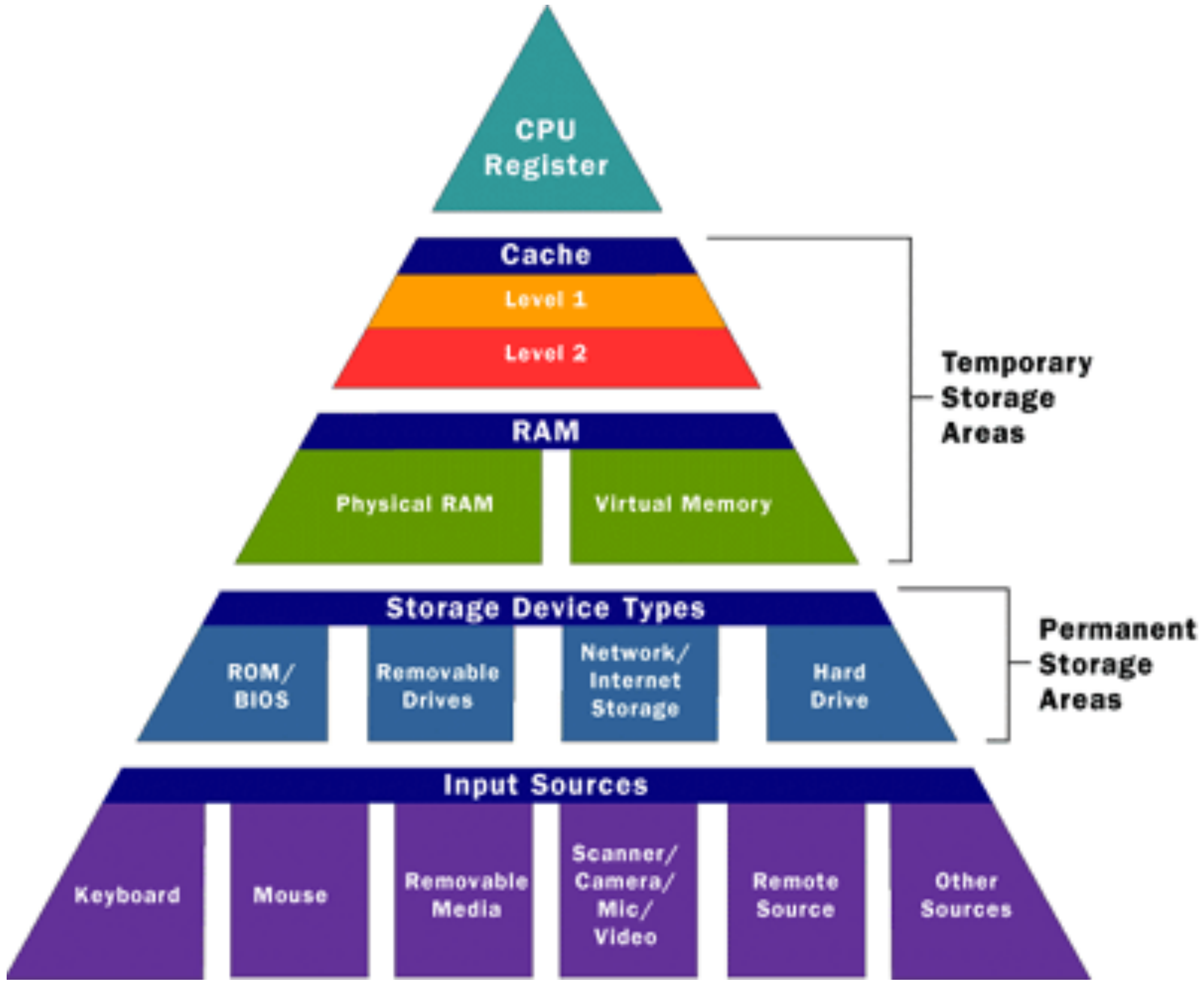
Another component is ***A Clock*** (not a register) that synchronizes the CPU and the entire system

Memory and Storage

- Memory and storage retain data to be used by the processor. The data stored in memory is volatile – it is lost when the computer is shut down. This type of memory is usually referred to as RAM (Random Access Memory).

- Data stored in storage devices such as disk drives is non-volatile – it can be retrieved after the machine is switched off. In addition to disk drives, other forms of storage are optical disks, memory sticks and (almost obsolete) floppy disks.

Memory Hierarchy

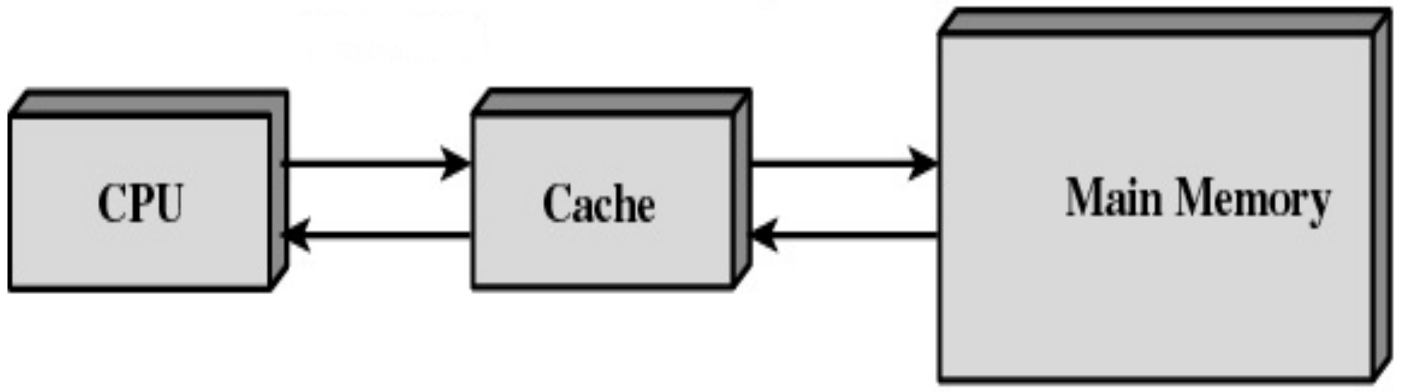


Memory and Storage (cont.)

- ***Main Memory*** stores the program being executed, the data used by the program and parts of the operating system. Main memory is volatile.
- ***Cache Memory*** is smaller and faster than main memory – here frequently accessed data can be stored for rapid access. It is possible to have multiple layers of cache.
- The processor first checks the cache - if not found in cache, the block of memory containing the needed data is moved to the cache.

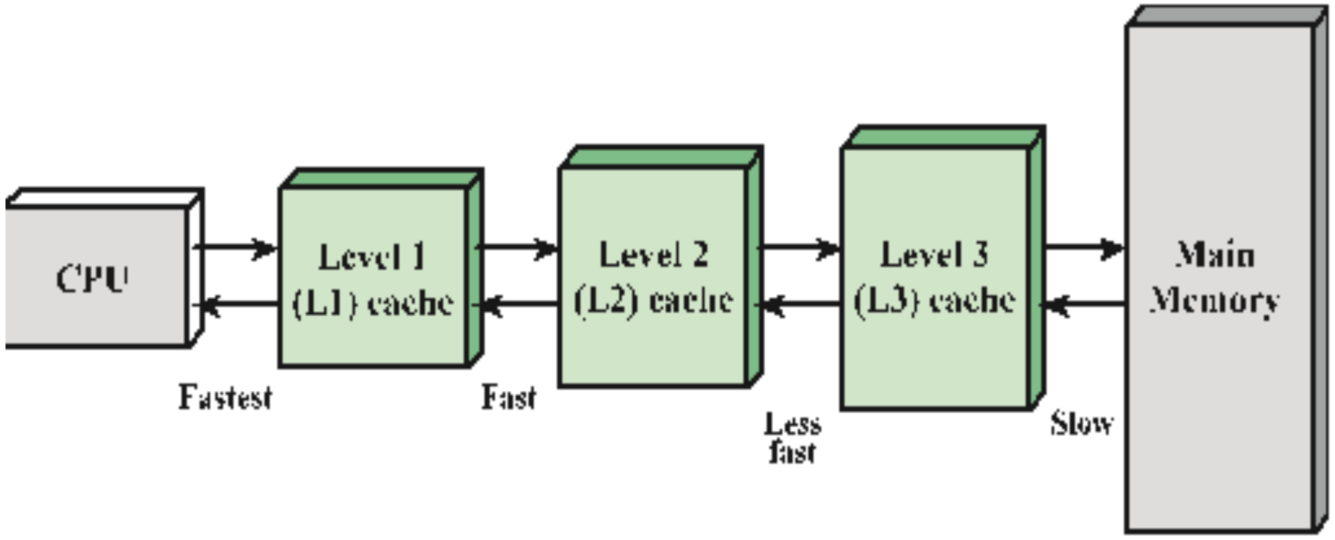
Cache Memory

Single Cache



Cache Memory

Three Level Cache



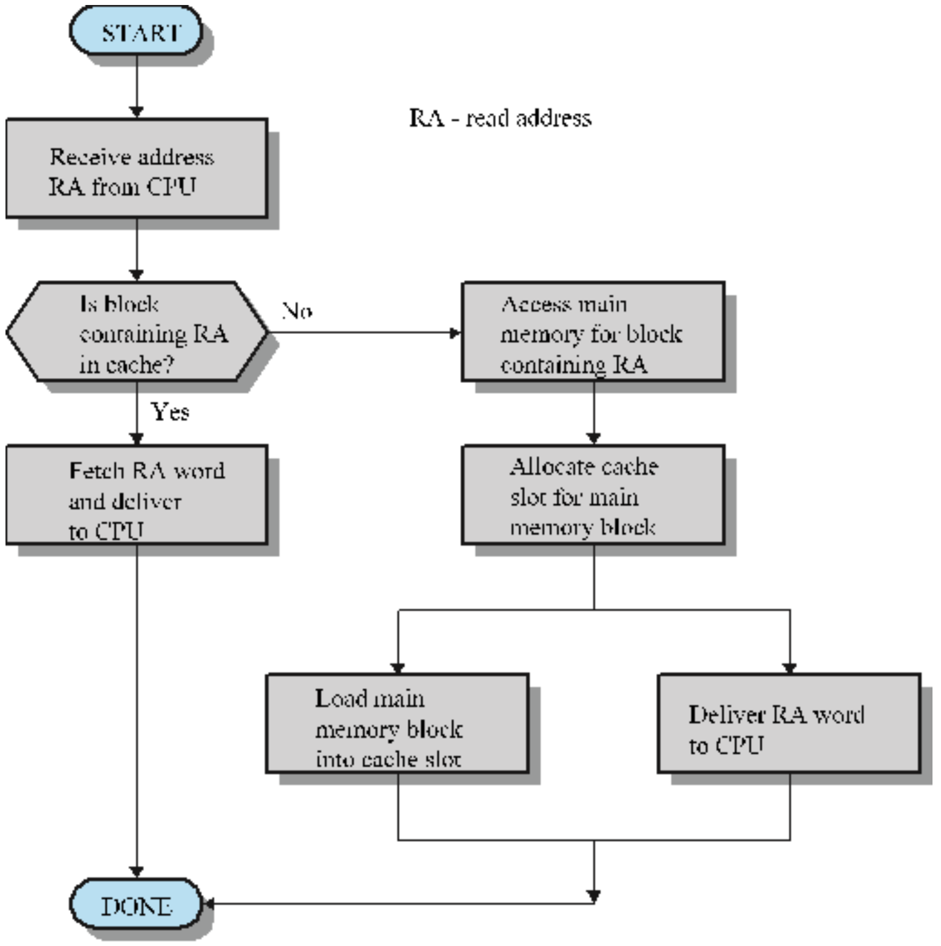
Cache (a read operation)

The following flowchart (next slide) describes how cache is used.

If data is contained in the cache, it is sent to the CPU.

Otherwise, it is loaded into cache (and then sent to the CPU)

Cache (a read operation)


START
Receive address RA from CPU
RA - read address
Is block containing RA in cache?
No
Yes
Access main memory for block containing RA
Fetch RA word and deliver to CPU
Allocate cache slot for main memory block
Load main memory block into cache slot
Deliver RA word to CPU
DONE

Virtual Memory

- Virtual memory is a technique to enable disk space to enhance main memory. This means that programs which require more memory than available can be run. It also means that more applications can be run concurrently.

- Using virtual memory is much slower as reading from a disk is significantly slower than reading from main memory.

Virtual Memory (cont.)

In Linux, the portion of the disk used for virtual memory is called swap space. Usually this is a separate partition to that which contains the main operating system. To see the size of the swap partition, enter the command fdisk –l (as root user)

In windows, the disk area is called a paging file. One can see the size of the page file via the control panel.

Managing Memory

Managing memory is a significant part of an operating systems' role. The operating system determines the amount of memory available for each activity and for how long that memory is available.

We will look at memory management requirements and management techniques.

Memory Management Requirements

Relocation: Processes are constantly being swapped out of memory and back again – often to a different location.

Protection: Each processes' memory space needs to be protected.

Sharing: It is sometimes efficient for multiple processes to share the same memory location.

Logical & Physical Organisation: Logical references need to be referenced to physical references.

Memory Management Techniques

Fixed Partitioning

Dynamic Partitioning

Paging

Segmentation

The Bus

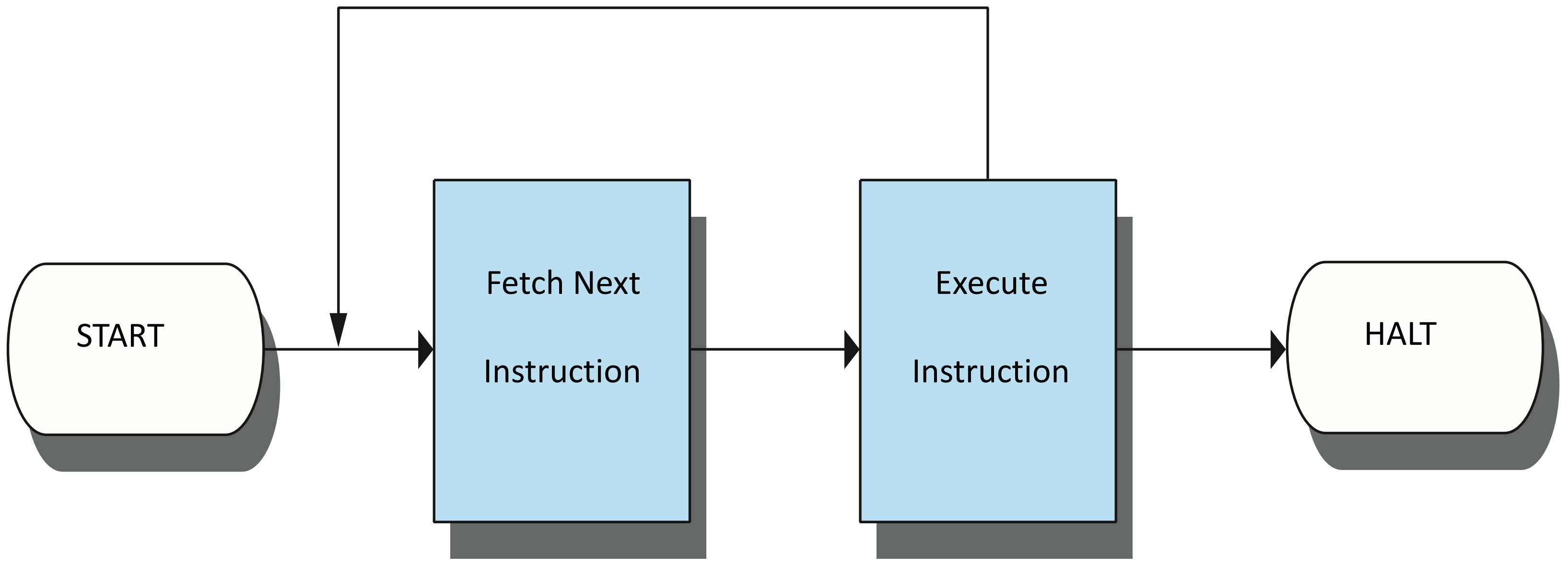
- The *Address, data and control buses* - are used for transfer of information between the components.
- The address bus carries addresses from a requesting component to the memory.
- The Data bus carries data to/from components and the Control bus specifies who’s in charge and which direction data is flowing (etc.).

Instruction Cycle (fetch execute)

A program consists of a set of instructions.

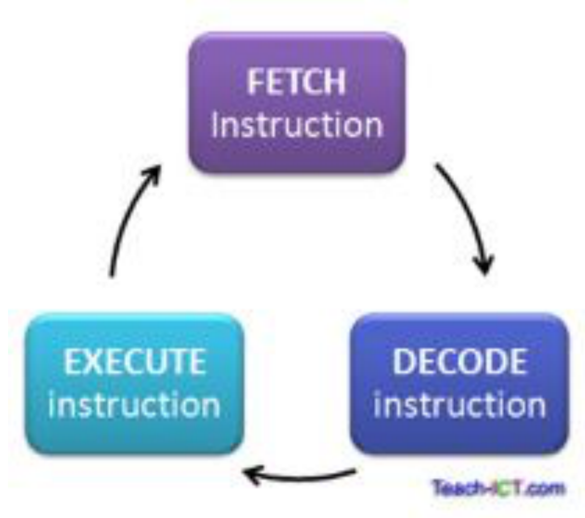
In its simplest form, the CPU fetches (reads) an instruction from memory one at a time and executes each instruction.

Instruction Cycle (fetch execute)



Fetch-Execute

The fetch–execute–decode (usually referred to as fetch-execute) outlines the main steps for processing to take place.



Fetch-Execute

Fetch

The CPU fetches data and instructions from memory.

This (data & instructions) is stored in temporary areas within the CPU.

These areas are known as registers.

Fetch-Execute

Decode

The CPU interprets the instructions.

The decoding is done by the CPU’s Control Unit.

Fetch-Execute

Execute

The instruction is carried out.

If this involves arithmetic or logic, the ALU is used.

Fetch-Execute

$$\begin{tikzcd}
\text{The program counter specifies a memory location} \arrow[d] \\
\text{An instruction is fetched from this location} \arrow[d] \\
\text{The program counter is incremented} \arrow[d] \\
\text{The instruction is executed.} \arrow[uuu, bend right=60]
\end{tikzcd}$$

Fetch-Execute Steps

PC contents are loaded to the MAR

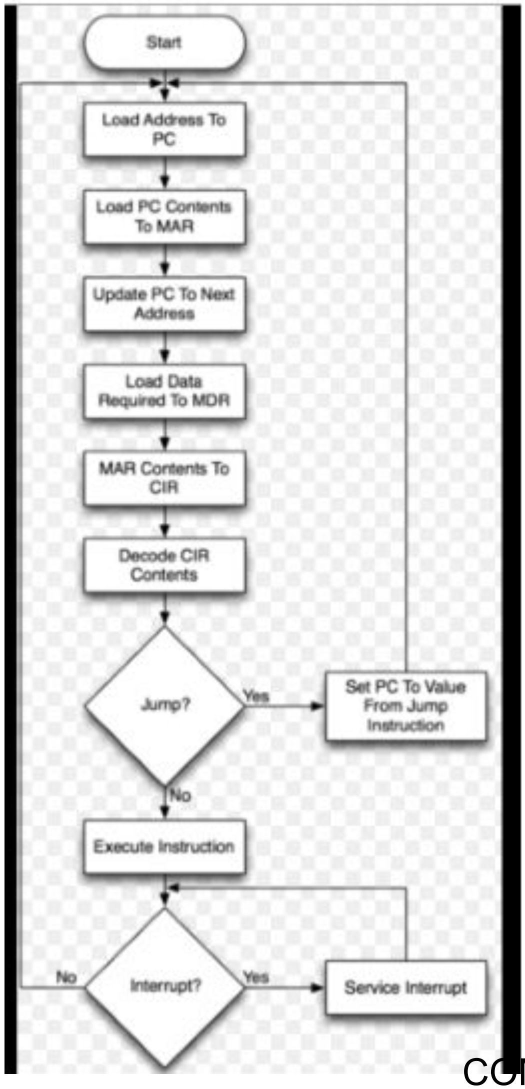
Data is read (from the MAR) to the MBR

Data is copied from MBR to IR

PC is incremented.

PC – program counter; MAR – memory address register
MBR – memory buffer register; IR – instruction register
See slide 9

Fetch-Execute Steps



CIR – Instruction Register

See http://www.ques10.com/p/10157/what-is-meant-by-fetch-cycle-instruction-cycle-m-1/

Interrupts

However, there are a number of exceptions (e.g. input waiting to be picked up) to this cycle known as interrupts.

Interrupts occur in the normal sequence of processing. Interrupts:

- Improve processing efficiency.
- Allow the processor to execute other instructions while an I/O operation is in progress.
- Cause a suspension of processes which can be resumed later.

Interrupts

Interrupts might happen for a number of reasons:

- Program – an error in a program e.g. division by zero

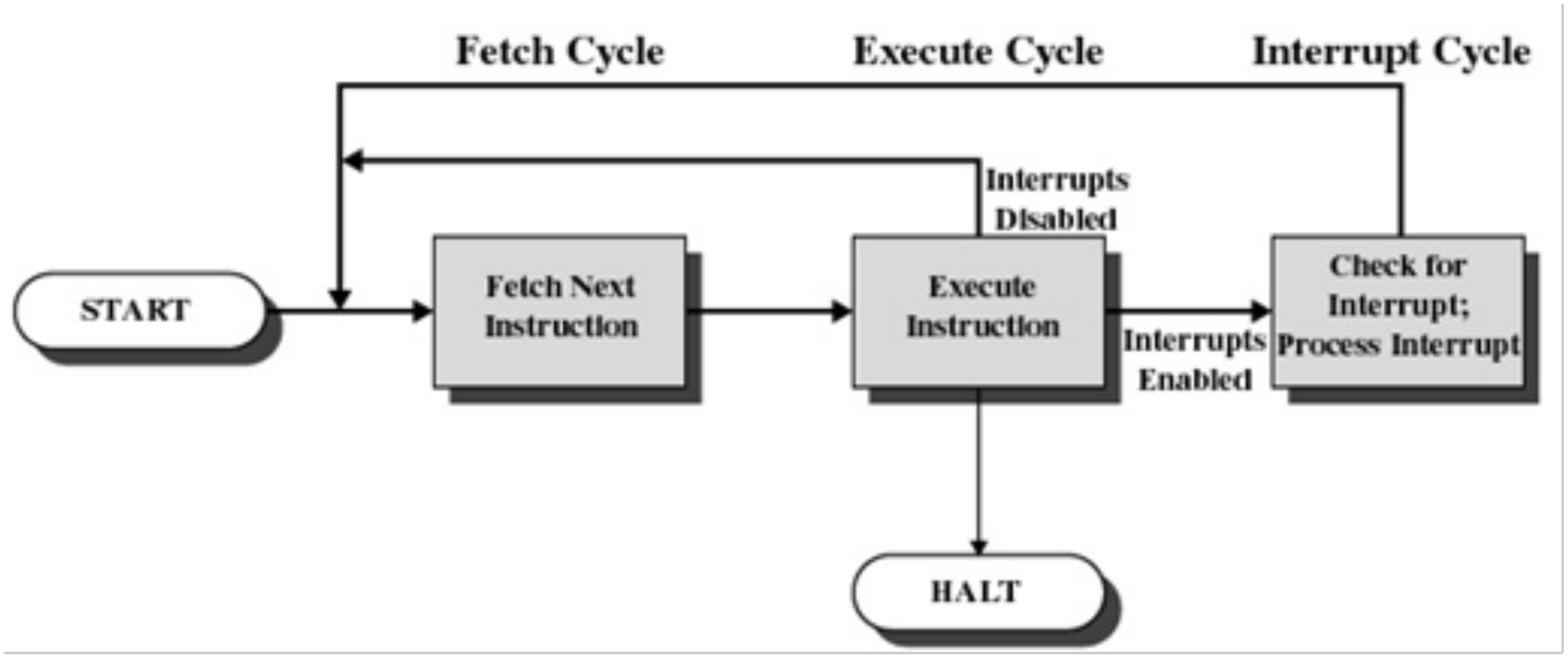
- Timer – A program has exceeded its processor time deadline

- Hardware failure

Interrupt Cycle

- Processor checks for interrupts
- If no interrupts fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt handler

Interrupt Cycle



Interrupt Handler

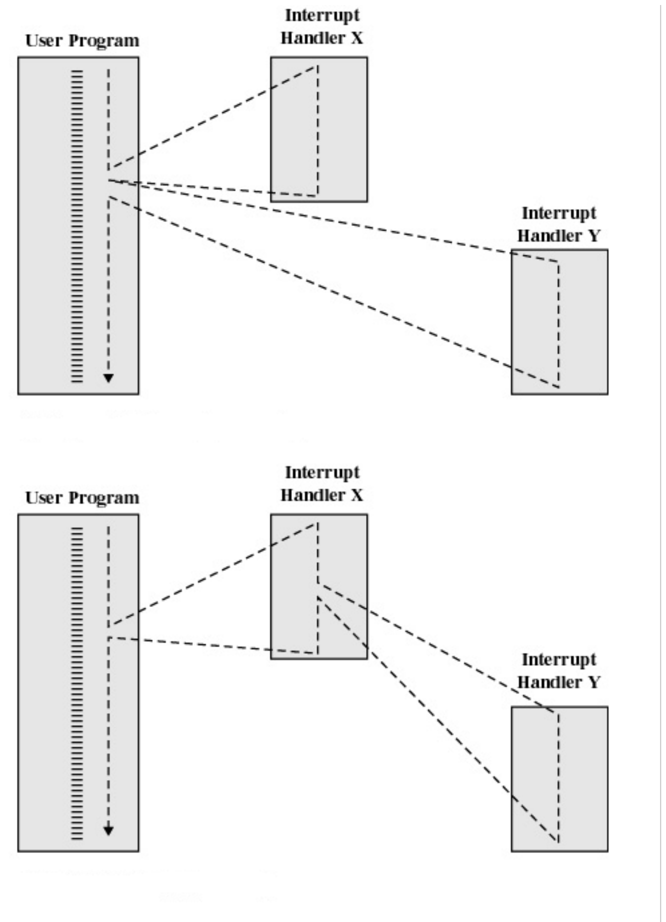
- This program determines nature of the interrupt and performs whatever actions are needed.
- Control is transferred to this program.
- Generally part of the operating system.

Multiple Interrupts

- Higher priority interrupts may cause lower-priority interrupts to wait

- Causes a lower-priority interrupt handler to be interrupted

Multiple Interrupts



Transfer of Control with Multiple Interrupts

Interrupts based on priorities

In the following, the user program begins at t = 0.

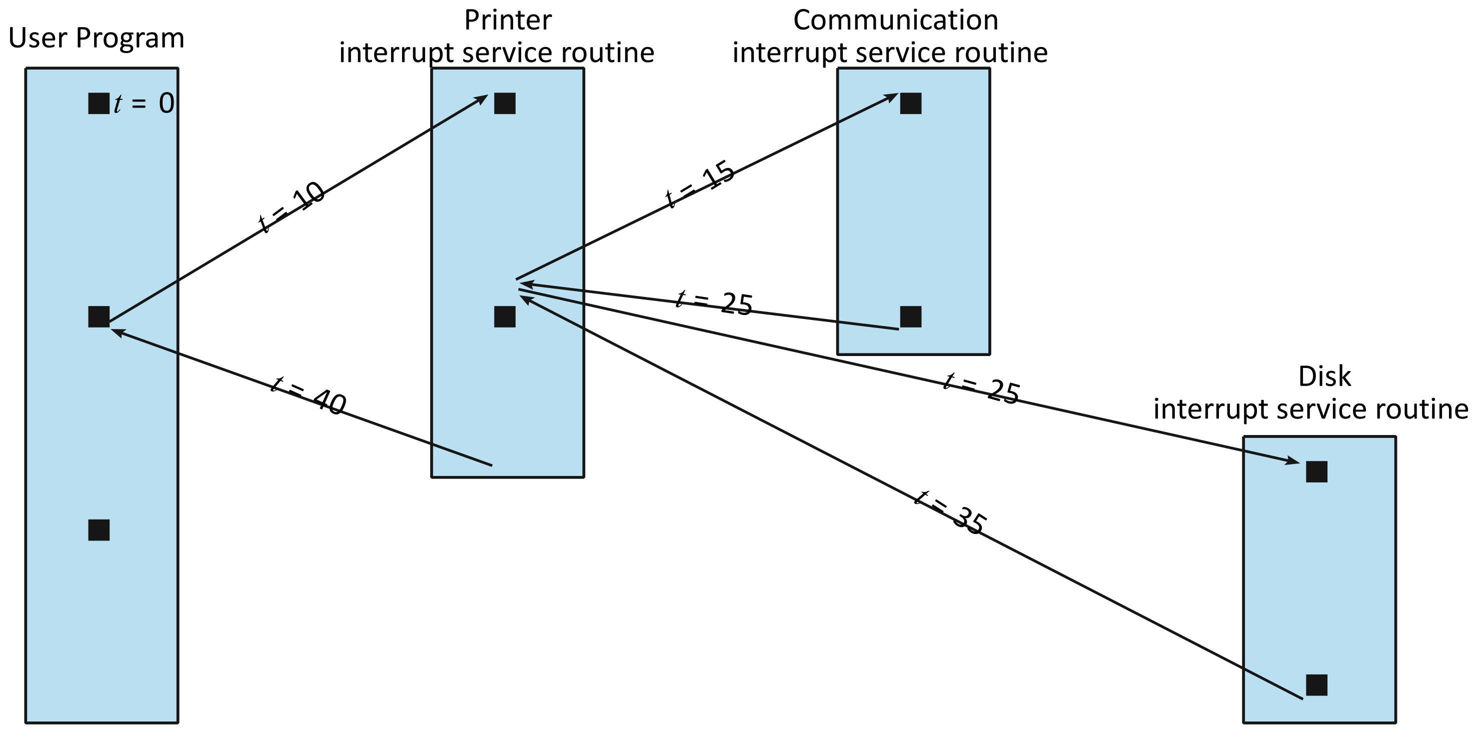
At t =10, there is a printer interrupt.

While it is still executing, it is interrupted by a communications interrupt (higher priority)

The next is a disk interrupt – lower priority than communications but higher than printer.

When done, the printer continues and back to the user program.

Sequence of Multiple Interrupts



Peripherals

- Peripherals are hardware devices added to a computer to enhance its features e.g. input and output devices.
- Examples of **input** devices: keyboard, mouse, joystick, barcode reader, etc. More specialist devices such as character readers and mark readers are used to automate activities. Early computers required data to be entered using punched cards.
- Examples of **output** devices: monitor, printer & speakers

I.O. Devices

Input / output devices can roughly be divided into three categories.

Human readable: used to communicate with humans e.g. keyboard, mouse, terminal & printer.

Machine readable: used to communicate with electronic equipment such as USB keys and sensors.

Communications: used to communicate with remote appliances e.g. network devices.