COMPUTER PERIPHERAL

LECTURE 01: INTRODUCTION

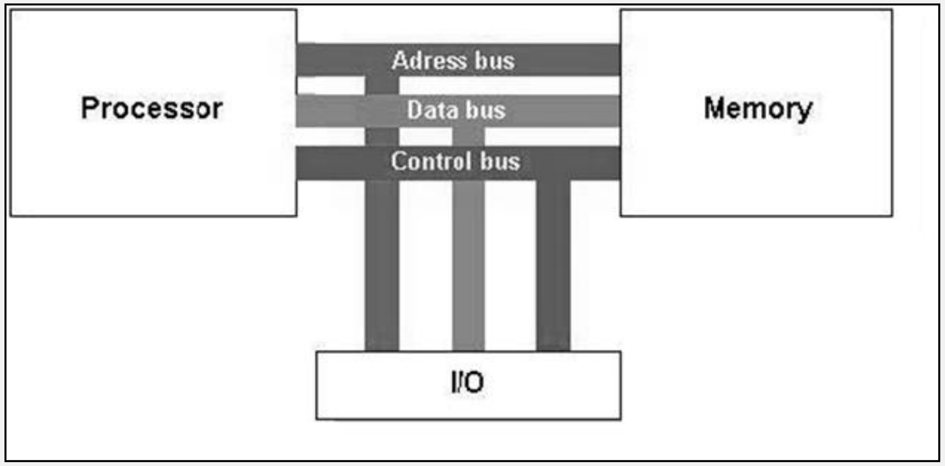
Dr. Reda M. Hussien

INTRODUCTION

FUNCTIONAL COMPONENTS OF A COMPUTER

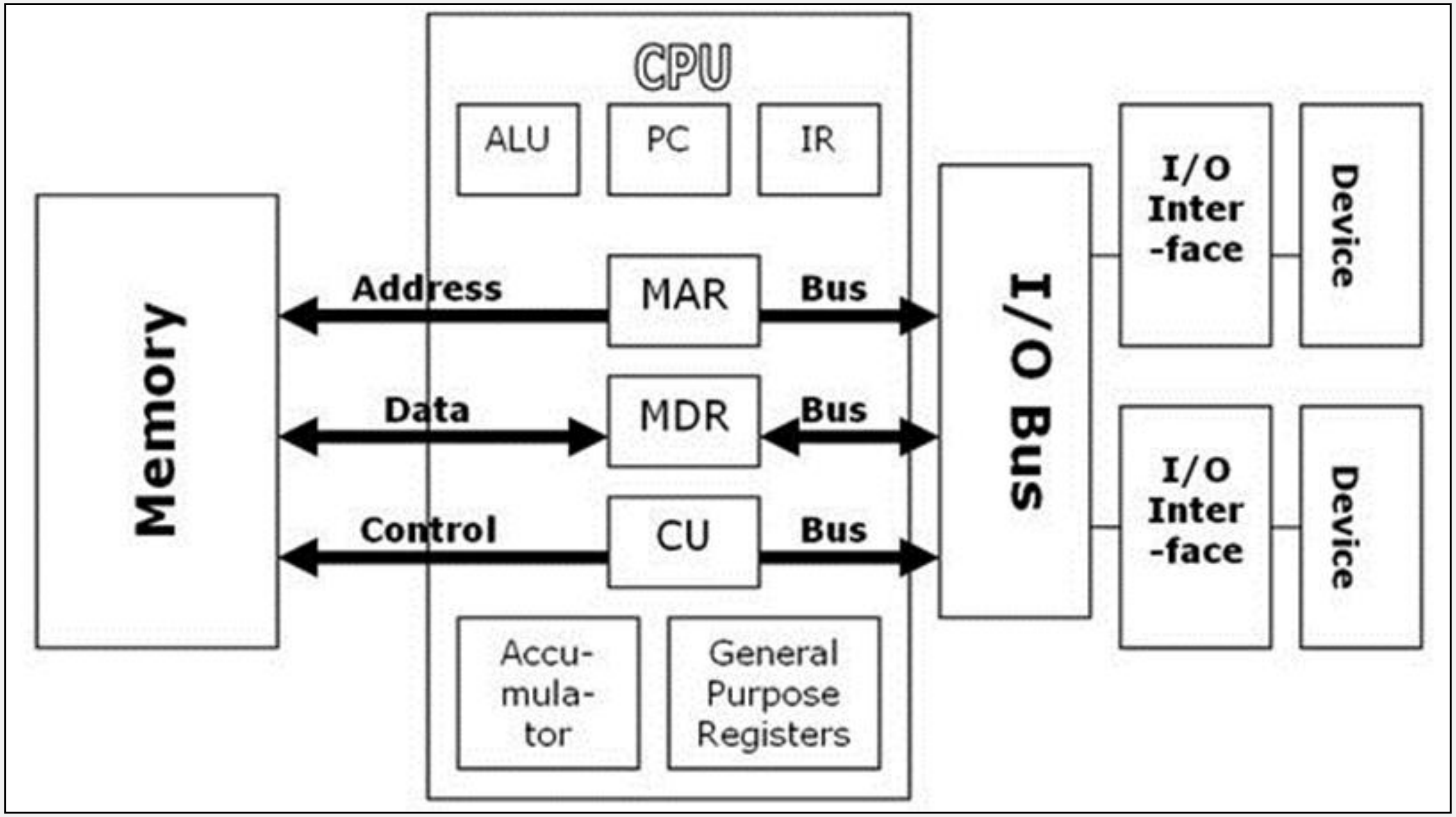
- A computer system contains many different objects such as a CPU, memory, disks, etc. These must all be connected for the system to function.

- The wires used to connect the components are called buses.



INTRODUCTION

FUNCTIONAL COMPONENTS OF A COMPUTER



INTRODUCTION

FUNCTIONAL COMPONENTS OF A COMPUTER

- Program Counter (PC) - an incrementing counter that keeps track of the memory address of which instruction is to be executed next...
- Memory Address Register (MAR) - holds the address of a memory block to be read from or written to
- Memory Data Register (MDR) - a two-way register that holds data fetched from memory (and ready for the CPU to process) or data waiting to be stored in memory

INTRODUCTION

FUNCTIONAL COMPONENTS OF A COMPUTER

- Instruction register (IR) - a temporary holding ground for the instruction that has just been fetched from memory

- Control Unit (CU) - decodes the program instruction in the IR, selecting machine resources such as a data source register and a particular arithmetic operation, and coordinates activation of those resources

- Arithmetic logic unit (ALU) - performs mathematical and logical operations The time period during which one instruction is fetched from memory and executed when a computer is given an instruction in machine language. There are typically four stages of an instruction cycle that the CPU carries out:

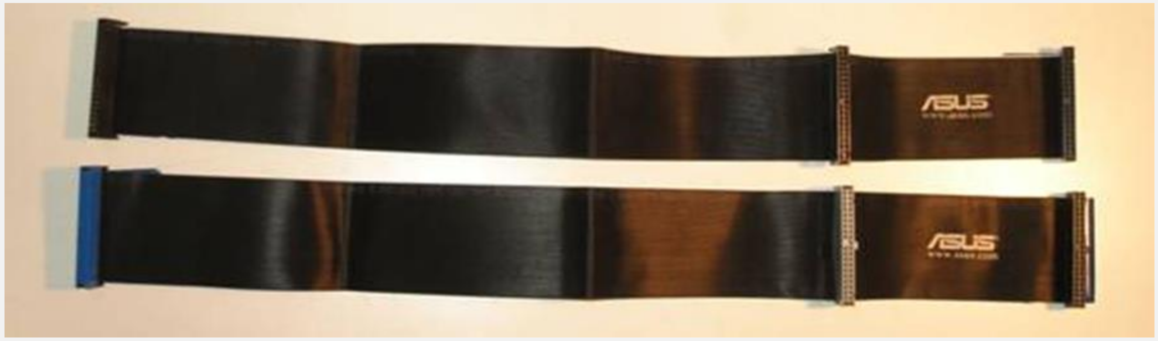
INTRODUCTION

BUSES

- The buses on a computer system are sets of wires that carry information to or from memory or I/O devices. They may be uni-directional (data travels one way) or bi-directional (data travels both ways).

Bus is A communication pathway connecting two or more devices

- Buses can be seen on the computer motherboard as parallel metal tracks. When the buses leave the motherboard to travel to a component such as a hard disk cables like these are used.



INTRODUCTION

BUSES

In general the number of wires on the bus corresponds to the *width* of the bus on the CPU.

Address Bus

- The address bus carries the address of the piece of memory or I/O device to be read from or written to.

- It is a unidirectional bus, which is to say that data travels only one way; from the CPU to memory.

- The number of lines on the bus determines the number of addressable memory elements. For example an 8 bit bus can represent 2 to the power of 8 unique addresses. This equates to 256 unique memory addresses. A 16 bit bus can address 65536 unique addresses and so on.

INTRODUCTION

BUSES

Data Bus

- The data bus carries the data that is to be written or has been read from memory.
- It is a bidirectional bus as it can carry data to or from memory.
- The width of the data bus is directly related to the largest number that the bus can carry. For example an 8 bit bus can represent 2 to the power of 8 unique values. This equates to the numbers 0 to 255. A 16 bit bus can carry the values 0 to 65535 and so on.

Control Bus

- The control bus carries signals that control the actions of the computer.

INTRODUCTION

I/O INTERFACING

- It is seen that each of the peripheral devices is connected to CPU through the I/O interface unit. The I/O interface contains the hardware necessary to allow communication with the I/O devices.

In order to function, I/O interfaces require at least the following elements:

- Transmit and receive data registers/buffer: registers and buffers (FIFO) are use to hold and transfer data to and from the peripheral device.

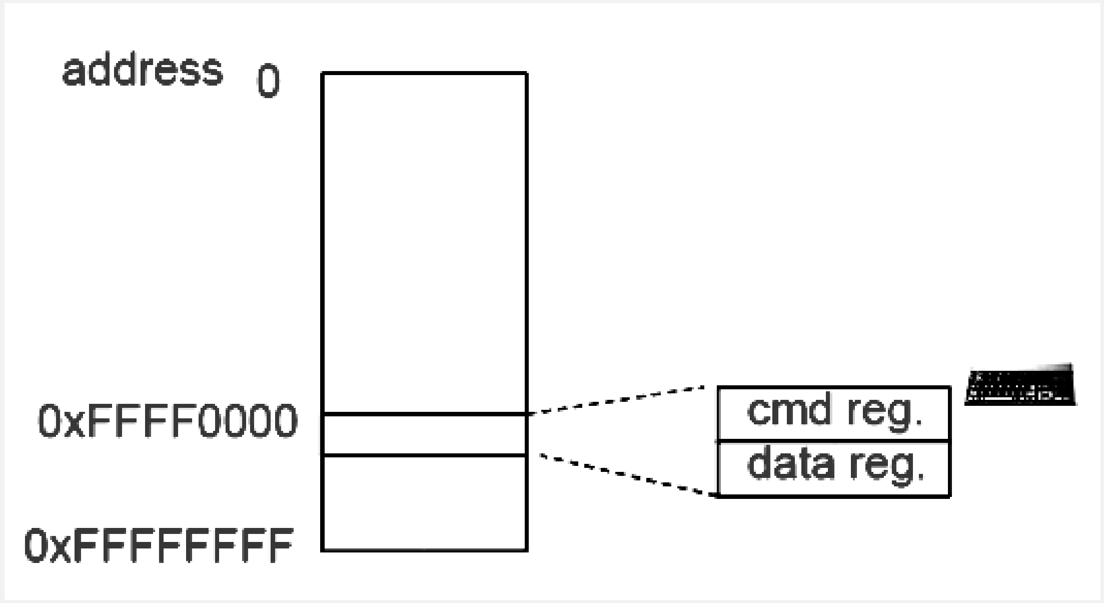
- Control registers: One or more control registers are used to capture and store the command received from the CPU.

- A status register: Each bit of the status register is used to indicate individual status conditions to the CPU.
-
- An address decoder: Irrespective of whether the device is interfaced using memorymapped or I/O-port techniques, the device will still have to decode the address information from the CPU to determine whether it should respond.
- Random logic: For simple devices, random logic circuits may be used to check the status registers, read and write the data registers, perform timing, handle interrupt signals and other functions.

INTRODUCTION

I/O ADDRESSING SCHEMES

- In order for the CPU to correctly identify the communicating device each device is assigned a unique address.
  - ✓ treat I/O devices as another type of memory. **memory mapped I/O**.
  - ✓ dedicated I/O commands to transfer data. When I/O instructions are issued the control bus will signal that the transfer to I/O and not memory. This is **dedicated I/O**.



INTRODUCTION

I/O INTERFACING TECHNIQUES

- The times at which data reaches a computer from the outside world can be quite unpredictable. The processor therefore needs some means of synchronising itself to external events, for scheduling I/O transfers.

- There are two main methods of achieving this synchronisation, namely
  - ✓ **polling** and
  - ✓ **interrupts**.

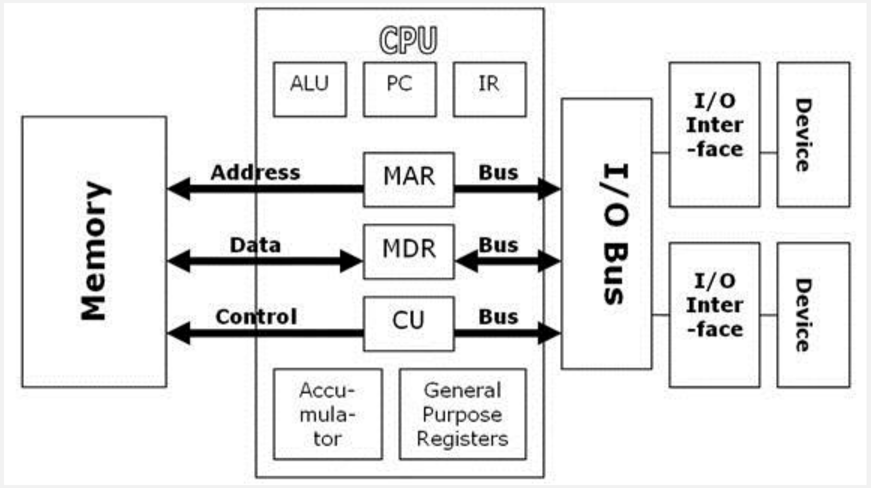
INTRODUCTION

I/O INTERFACING TECHNIQUES

- **Polling**
  - ✓ Polling is a software technique whereby the processor continually asks a peripheral device if it needs servicing.
- **Interrupts**
  - ✓ the I/O devices tell the processor when they have data ready. The processor can be carrying out its normal function, only responding to I/O transfers when there is data to respond to. On receipt of an interrupt, the CPU suspends its current operation (storing the contents of its program counter, SR and other registers), identifies the interrupting device, then jumps (vectors) to the appropriate interrupt service routine (interrupt handler).

Direct Memory Access (DMA)

- If we look at the computer system diagram we can see that when transferring data from a device such as a hard disk the CPU is heavily involved.

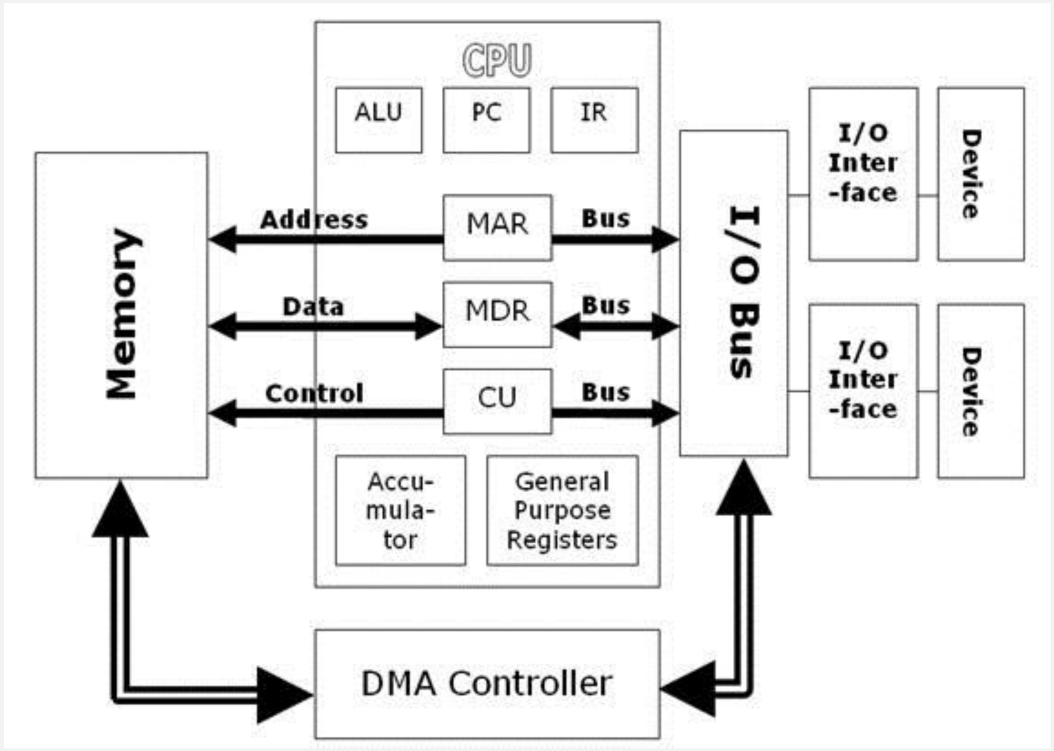


- Each word of data has to be transferred from the disk, through the CPU and individually placed into memory. This places a heavy burden on the CPU and stops it from performing any useful tasks.
- To address this problem we use a system of Direct Memory Access (DMA).

Direct Memory Access (DMA)

IMPLEMENTING DMA

- To implement the direct transfer of data between memory and I/O a hardware DMA controller is added to the system. This hardware works like a bypass for a town, allowing large blocks of data to bypass the CPU without snarling it up.



Direct Memory Access (DMA)

IMPLEMENTING DMA

- In this scenario large blocks of data are transferred using the DMA hardware. The CPU, on receiving an interrupt, initiates the DMA hardware with
  - ✓ A memory start address
  - ✓ The amount of data to be transferred
  - ✓ The device to be used (hard disk, CD, DVD, etc.)
  - ✓ The direction of transfer (input or output)
- The DMA controller then transfers the data. Upon completion the controller notifies the CPU that the transfer has been completed

Example of DMA

- Loading program from disk to memory
  - ✓ One option: CPU loads first byte copy it to memory (move or load/store operation)
  - ✓ Other one: DMA: does not include CPU into the loop. Initially an interrupt signal is sent to notify the DMA controller to initiate DMA operation. Source address, Destination address and bytes to transfer is written into control registers. Once the operation is done (memory bus is used as the communication channel) DMA controller sends signals to CPU