COMPS260F Computer Architecture and Operating Systems

Chapter 6. Technologies of Computer Components

This chapter will discuss the technologies for real computer systems. The discussion on the LMC has concluded with four major components in computer operations.

- CPU or processor: executing instructions
- Bus: data movement for executing instructions
- Memory: data storage and retrieval
- IO: data input and output

This chapter will discuss the technologies developed for these four major components.

1. CPU Technologies and Manufacturing Process

CPU is the component in a computer system that executes instructions of a computer program. It typically consists of two core components: the arithmetic and logic unit (ALU) and the control unit (CU).

Depending on the current trend of CPU design, a CPU may also have other components such as:

- Memory management unit (MMU): for controlling data transfer in and out of the CPU.
- IO control unit: for controlling peripheral devices.
- Cache memory: an internal fast memory structure for mirroring data in the main memory system.

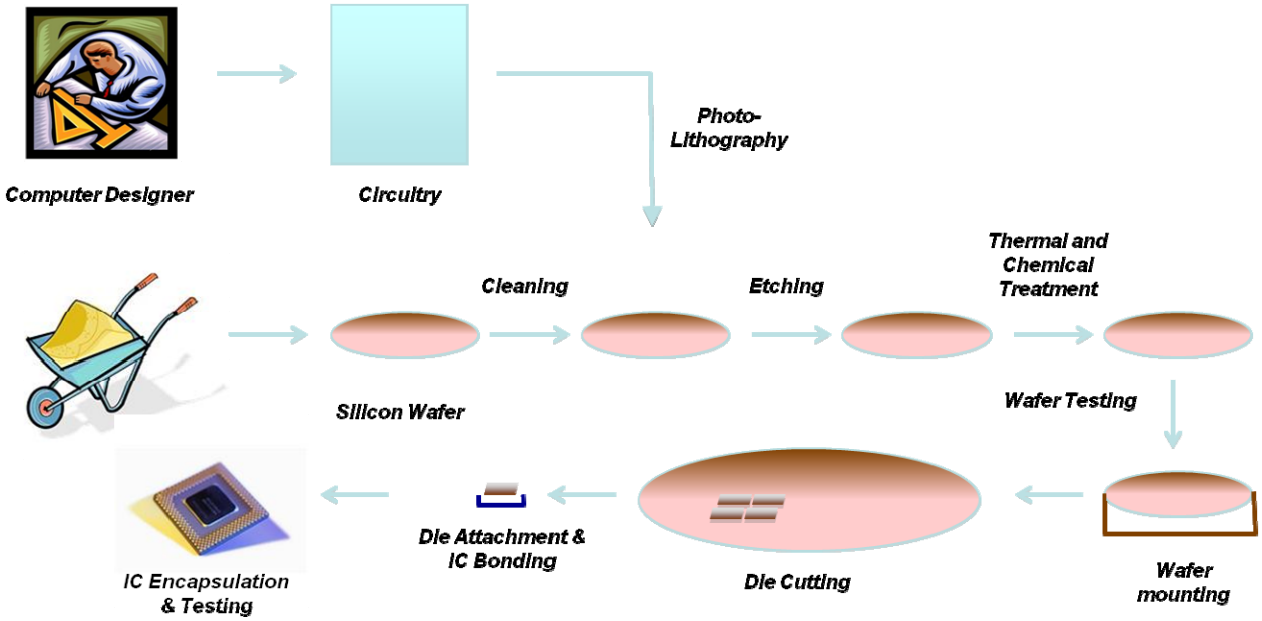
The CPU is a highly sophisticated electronic device based on complex circuitry. The physical appearance of a CPU is often a chip or a set of chips. A chip is a package containing integrated circuit, which means electronic circuit manufactured by depositing or diffusion of chemicals on a thin substrate of a semi-conductor material such as silicon.

The process **lithography** describes the process of "printing" a circuit on a semi-conductor substrate. CPU designers first design the complete circuit for the ALU and the CU on a computer. Then the circuit as an image is printed on a CMOS wafer, which is a piece of circular and chemically treated semi-conducting material. Usually, a wafer is large enough so that many units of CPU are printed and then cut out for packaging.

The current technology of integrated circuit manufacturing is based on **photolithography**, a process that is similar to photocopying an image of circuitry onto the wafer.

The entire manufacturing process takes place in highly controlled environment in special manufacturing plants. The duration of the process is typically around 2 months. Here are the common steps taken:

- Wafer preparation: crystallisation of pure silicon
- Wafer processing: printing circuitry onto silicon wafer
- Die cutting and attachment: units of CPU die are cut from wafer
- Chip packaging and testing



CPU cost is usually a significant part of the cost of a computer system. The cost of a CPU chip depends on a few factors, and the most important ones are:

- Maturity of the manufacturing process.
- Size of the CPU chip.
- Raw materials.
- Competition in the market.

The current manufacturing process in 2012 is called the 22-nanometre process (22 nm). The figure is roughly indicative of the (half) distance between features in the printed circuitry. The first 22 nm device was shipped in 2010 and it is still maturing.

One important feature of a matured process is the yield. A very new and immature manufacturing process produces more defects in the wafers and the dies. The overall cost is therefore elevated to cover the loss caused by the defects. The following formula gives an estimation of the cost of a die.

$$DieCost = \frac{WaferCost}{Dies\_per\_wafer \times Die\_yield}$$

In 2009, processing a 300mm wafer costed around US$2800 but a 150mm costed less than US$450 (Reference: GSA Wafer Fabrication Pricing Reports). The 300mm and 150mm are diameters of a circular wafer. The number of dies per wafer can be estimated using the following formula.

$$Dies\_per\_wafer = \frac{\pi \times (Wafer\_diameter/2)^2}{Die\_area} - \frac{\pi \times Wafer\_diameter}{\sqrt{2 \times Die\_area}}$$

Exercise: Dies per wafer

Question: The die size of an Intel core i7 is 263 mm square, calculate how many dies can be cut from a 300 mm wafer.

Answer:

$$Dies\_per\_wafer = \frac{\pi \times (300/2)^2}{263} - \frac{\pi \times 300}{\sqrt{2 \times 263}} = 268.7 - 41.1 = 227 \text{ dies}$$

The die yield is dependent on the wafer yield (how many wafers are defected) and the defects per unit area in the manufacturing process. Typical range is 0.3 to 0.6 for new processes. A large die makes it more likely that a defect occurs in the area occupied by the die.

Miniaturization and Performance

Using a smaller die size in CPU chip manufacturing can increase die yield. However, the size must be sufficient for printing the entire circuitry. The trend is to scale down the circuitry with methods such as optical means so that the resulting die is smaller. The technology challenge is to make the features on the circuitry as close together as possible, while the chip is still operating smoothly.

The semi-conductor manufacturing process has undergone relentless miniaturization since the 1970s. The following shows the major process stages.

Year	1971	1975	1982	1985	1989	1994	1995	1998	1999	2000	2002	2006	2008	2010	2012
Process (nm)	10000	3000	1500	1000	800	600	350	250	180	130	90	65	45	32	22

The miniaturization has a number of advantages:

- Improve die yield: smaller die size
- Reduce power consumption: fewer electrons can drive the circuitry
- Increased potential highest clock rate: less time to travel for signal to travel

The most direct way to improve CPU performance is to increase the clock rate. The execution of instructions depends on a certain number of micro-operations, and making the clock rate faster can reduce the time to complete the execution.

Clock rate cannot be increased indefinitely. The signals in the CPU require time to switch from one state to another and the time is dependent on the current electronic technologies and physical laws. In addition, heat is dissipated in state transition and so more heat is generated with higher clock rate. If cooling is not sufficient, electronic devices will be damaged.

2. Bus Technologies

A bus is a data channel for transferring data from one device to one or more other devices. The system bus, which connects the various registers in the CPU, is an example. There are many buses in a computer system.

A bus consists of a number of lines, each of which serves one of the following four purposes

- Data. A data line is binary encoded, and therefore it can carry one bit of data at an instance.
- Addressing. An address line is binary encoded, and therefore it can carry one bit of data at a time. The data on an address line represents an address.
- Control. A control line is also binary encoded. The data on a control line represents a signal. For example, the control unit (CU) sends a signal to the program counter (PC) on a control line (to invoke the increment function).
- Power. The power at a particular stable voltage supplied by the computer system.

Although the data sent on a bus is said to be binary encoded, there is usually a lower level encoding scheme to code the binary values 0 and 1 into other signal representation. For example, the USB encode binary data with the NRZI encoding scheme, which represent 0 and 1 with transition of two signal states.

Bus Throughput

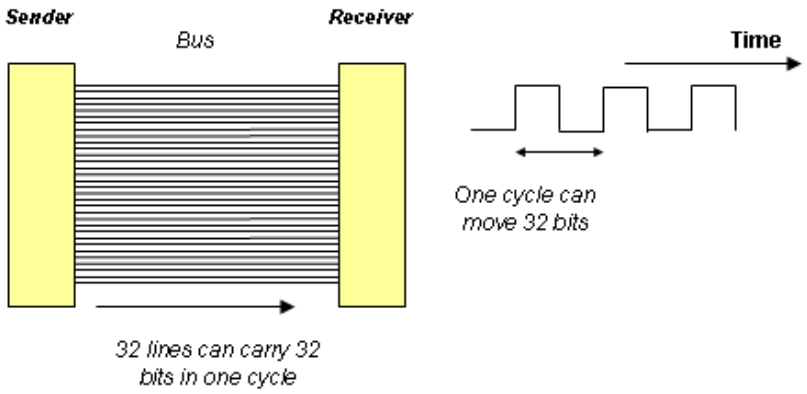
Bus throughput is the amount of data transfer on a bus per second. Bus throughput is often called data rate or bandwidth. For example, USB 2.0 data rate is 480 Mbit per second.

Some buses such as the front side bus (FSB) on a PC is rated in term of frequency. The frequency defines the period required to send 1 unit of data. The relation between frequency and period is given in the formula:

$$Frequency(Hz) = \frac{1}{Period(\text{s})}$$

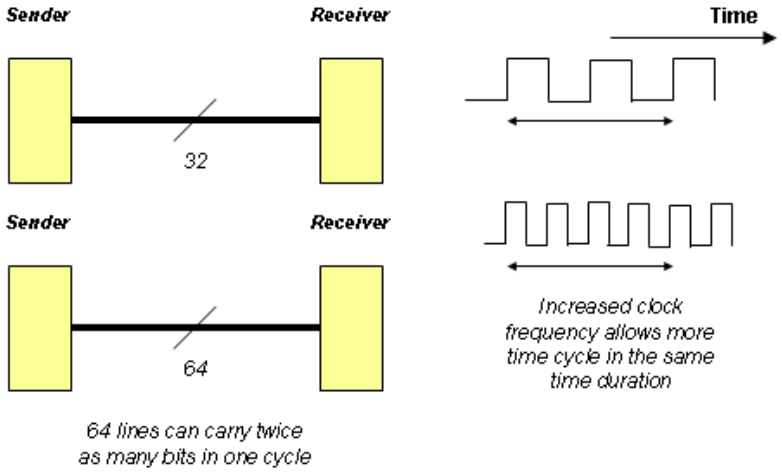
For example, a 500 MHz FSB means that the cycle period is 2 ns.

- A high throughput means moving more data in a particular time frame.
- If one data line in a bus can move a unit of data in a cycle, then theoretically a 32-line data bus can move 32 units in a cycle.
- Basically there are two ways to achieve high throughput: increasing the transfer rate and increasing the number of lines.



The above diagram assumes that one data line can transfer 1 bit of data per cycle.

The following diagram illustrates the benefits of higher data rate and a wider bus.



The following formula shows the theoretical throughput of a multi-line bus.

$$BusThroughput = NumberOfLanes \times DataRatePerLane$$

Exercise: Bus Throughput

Question: Given that each data line can complete the transfer of 1 bit in 200 ns. Calculate the throughput if the bus has a total of 32 lines (a 32-bit bus)

Answer:

Data Rate per Line = 1 bit / 200 ns = 5 M bit / second
Bus Throughput = 32 x 5 M bit / second = 160 M bit / second = 20 M byte / second

Some modern bus systems supports multiple data movement moments in one clock cycle. For example AGTL+ allows 4 transfers per cycle.

Exercise: Bus Throughput

Question: Given that an AGTL+ is running on a clock rate of 100 MHz, and the bus is 64 bit. Calculate the throughput.

Answer:

Data Rate per Line = 100 M bit / second x 4 transfers = 400 M bit / second
Bus Throughput = 64 x 400 M bit / second = 25.6 G bit / second = 3.2 G byte / second

In general, bus throughput is dependent on the following factors:

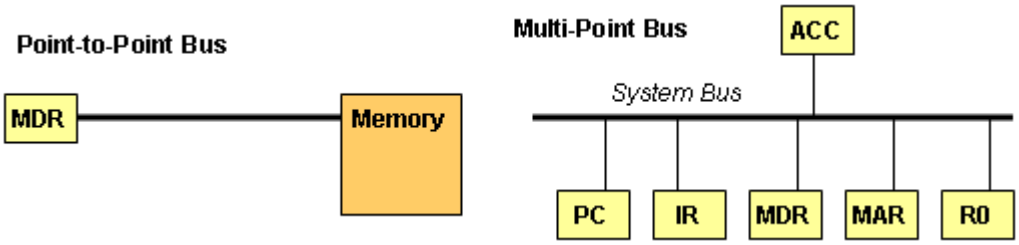
- Data transfer rate
- Number of lines (or bits)
- Overhead of protocols (used in encoding data)
- Distance between connected devices
- Addressing and control

Bus Connectivity

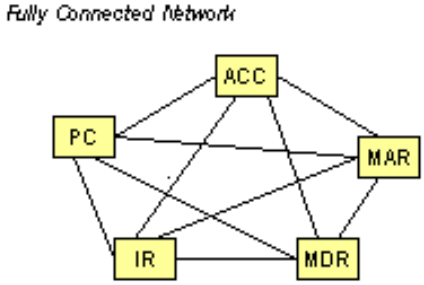
Generally there two types of buses: **point-to-point bus** and a **multi-point bus**.

- A point-to-point bus carries data from a specific source to a specific destination. For examples, the bus between the CPU and the Memory system, and the bus leading to a port for an external device are point-to-point buses.
- A multi-point bus broadcasts data to everything connected to it. The system bus is one example. The PCI bus is another example that connects many peripherals devices on PC.

Although a multi-point bus connects many components, there is a limitation to the communication modes. Only one component is allowed to send data out. One or more components can listen and receive data. Therefore, a single multi-point bus cannot allow two pairs of communication to happen together.



An expensive alternative is to fully connect all permutations of register pairs.



This facilitates data transfer to occur in parallel and reduces waiting time. However, each register can only handle one update (store operation) at one time.

Parallel Buses and Serial Buses

Parallel buses have more than one data line or data channels sending data at the same time. Parallel buses transfer more than 1 bit at a time, as opposite to serial buses. For the multiple data lines to be synchronized, a clock signal is usually sent on a separate control line.

It seems logical to assume that throughput of parallel buses should be greater. There are more data lines for data transfer simultaneously and so more bits can be sent at a time. However, parallel buses suffer from the following problems:

- Clock skew: the signals of different data lines arrive at slightly different times. It may be due to cable length difference and material difference. For faster clock rate, the smaller margin for error, and so a little difference can cause errors.
- Crosstalk: the signals between data lines may interfere with each other, causing errors in the signal. Shielding of cables can help but it will increase size and cost.

Parallel buses are particularly not suitable for connecting devices separated by long distance. The increased length would make the above problems more severe.

Serial buses have a single data line connecting two components. They allow the sending of 1 bit at a time. Serial buses can work over a longer distance.

Recently, serial buses have become the most common form of buses even for short distance communication. A serial bus running at a significantly faster clock rate can outperform a parallel bus and at a cheaper price as well.

3. Review of Bus Technologies for PC

The Personal Computer (PC) is a class of desktop computers available at an affordable price for people to use in home and offices. This section reviews the different types of buses found in generations of PC.

Peripheral Component Interconnect (PCI)

The PCI bus (Peripheral Component Interconnect) is a standard for connecting a computer to peripheral devices.

- PCI is a multi-point bus and a parallel bus between the IO controller hub (the south-bridge) and PCI devices.
- Configuration: clock rate from 33MHz to 66MHz, with data width 32-bit or 64-bit, giving throughput from 133 MB/s to 533 MB/s.
- PCI supports plug-and-play, and the device interrupt identifier is assigned by firmware rather than using jumpers.
- PCI has a variant called PCI-eXtended (PCI-X), which runs on clock rate of 133MHz, giving bandwidth of 1066 MB/s.

Peripheral Component Interconnect Express (PCI Express)

The PCI Express bus is a standard for connecting a computer to peripheral devices.

- PCI-Express is a point-to-point and a serial bus. Data and signals are transferred on lanes.
- However, link between 2 PCIe devices may operate on different number of lanes, depending on the need of throughput. High demand applications such as graphics can run on multiples of PCIe lanes.
- Data transmission on a multi-lane connection is interleaved, that successive bytes are transferred on different lanes.
- Data rate is around 250MB/s per lane, and a 16-lane connection is capable of around 4000MB/s.
- First-generation PCIe is constrained to a single signalling-rate of 2.5 G bits/s. The figure of 2.5 GB/s is a calculation from the physical signalling-rate (2500 M baud) divided by the encoding overhead (10bits/byte). This means a 16 lane (x16) PCIe card would then be theoretically capable of 250 * 16 = 4000 MB/s (3.7 GiB/s) in each direction.

Accelerated Graphics Port (AGP)

The AGP Port (Accelerated Graphics Port) is a bus for connecting video device to computer. The point of connection is often the primary controller hub to the main memory and the CPU.

- AGP is a point-to-point bus and a parallel bus. It is superseded by PCI-Express bus already.
- Originally designed as 8-bit bus (at 4.77MHz), and subsequently upgraded to 16-bit (at 8MHz).
- AGP has a variety of speed and size
  - AGP 2x: 32-bit, 66MHz, double-pumped (data transfer)
  - AGP 4x: 32-bit, 66MHz, quad-pumped
  - AGP 8x: 32-bit, 66MHz, eight-time per clock cycle
  - AGP 2x throughput: 4 bytes x 66MHz x 2 = 533MB/s

Industry Standard Architecture

The ISA bus (Industry Standard Architecture) is an old standard of computer bus on PC connecting peripheral devices.

- ISA is a parallel and multi-point bus.
- Originally designed as 8-bit bus (at 4.77MHz), and subsequently upgraded to 16-bit (at 8MHz).
- The EISA improvement extends the bus further to 32-bits (at 8.33MHz) and allowed more than one CPU to connect to the bus.
- ISA supports DMA and an early version of plug-and-play, which did not perform well.

Advanced Technology Attachment (ATA) and SATA

The ATA bus (Advanced Technology Attachment) is another conventional standard of computer bus on PC. It connected the IO controller hub (south-bridge) to a hard-disk. It is also called IDE, ATAPI, or UDMA.

- The IO controller of ATA bus is situated on the hard disk itself, rather than on the motherboard. Apart from harddisks, many other devices are connected to computer using ATA including CDROM, ZIPDisk, and tape drives.
- The speed of transfer was improved with DMA and Ultra DMA (UDMA) so that data can be written directly to memory without the intervention of the CPU. Speed of transfer depends on the generation of ATA. For example UDMA 100 runs at 100 MB/s.
- This standard is superseded by SATA (Serial ATA). SATA 150 runs at 1.5 GB/s.
- Serial ATA (SATA or S-ATA) is a computer bus technology primarily designed for transfer of data to and from a hard disk.
- It is the successor to the ATA. This older technology was retroactively renamed Parallel ATA (PATA) to distinguish it from Serial ATA.

- First-generation Serial ATA interfaces, also known as SATA/150, run at 1.5 gigahertz. Because Serial ATA uses 8B/10B encoding with an efficiency of 80% at the physical layer, this results in an actual data transfer rate of 1.2 gigabits per second (Gbit/s), or 120 megabytes per second.
- This transfer rate is only slightly higher than that provided by the fastest "Parallel ATA" mode, Ultra ATA at 133 MB/s (UDMA/133).
- With the release of the NVIDIA nForce4 chipset in 2004, the clock rate of SATA II was doubled to 3.0 GHz, for a maximum throughput of 300 MB/s or 2.4 Gbit/s.

External Advanced Technology Attachment (eATA)

External SATA or eSATA (131MB/s), the SATA devices can be plugged by shielded cable lengths up to two meters outside the PC.

- Up to six times faster than contemporary external storage solutions: USB 2.0 and Firewire (IEEE 1394).
- eSATA is not faster than USB 3.0 (400MB/s) and Firewire S3200 (3200Mbit/s).

USB Bus

USB bus has become the popular means of peripheral connection.

- 1-bit serial
- Hot-pluggable
- USB devices driven by host computer
- USB 2.0 (60MB/s) USB 3.0 (400MB/s), compared to Firewire 400 (400Mbit/s) and 800 (800Mbit/s).

Front-side Bus (FSB)

Front-side bus (FSB) connects the processor to the primary controller hub, which then connects to the main memory and the IO controller hub. In von Neumann architecture, FSB plays the key role of transferring both data and instructions from the main memory to the processor. The performance of front-side bus is most important.

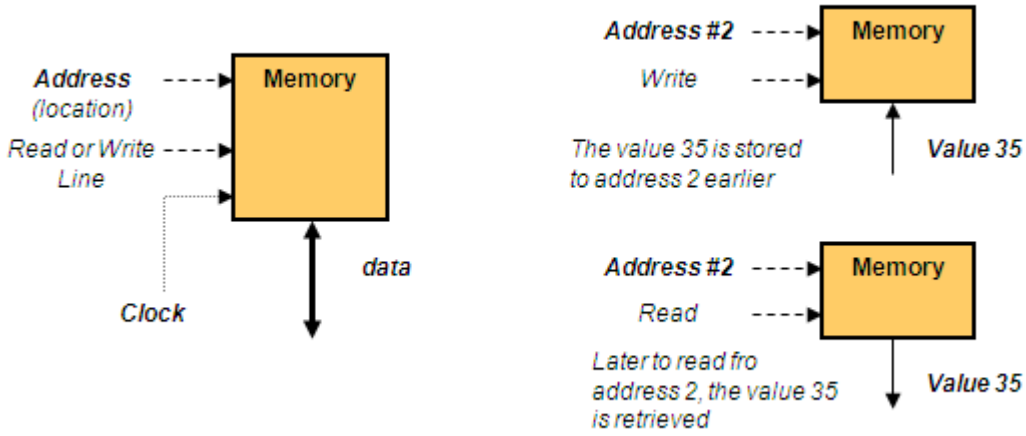
- It is used in Intel processors such as Pentium, Celeron, and Core 2.
- It is a parallel bus, of which the width is normally 64-bits.
- The later versions can perform 2 or 4 data transfer per clock cycle. For example, Pentium III FSB supports 1 data transfer but Pentium 4 supports 4 data transfer per clock cycle. Even if both are running on 100 MHz clock, the throughput increased from 800 MB/s to 3200 MB/s.
- FSB is generally regarded as the performance bottleneck of the computer in that generation. The CPU cannot execute instructions before the instruction and data can be read in through FSB.
- Intel QuickPath Interconnect (QPI) and AMD HyperTransport now offer superior technology. For example, QPI provides high-speed serial data transfer between multiple components similar to network communication. There are multiple data channels allows two components (i.e. CPU and memory) to separate input and output data flow.

4. Memory Technologies

The range of memory technologies for computer spans across many dimensions: speed, cost, and other characteristics. While a computer designer could ignore the cost issue and choose the best memory technology, the market would however favour using the memory type fit for the purpose.

Data stored in a memory system is structured based on a unit of data. The size of the basic unit of data varies from one type of memory to another. It could be 1-bit, 8-bit, 32-bit, 64-bit and so on. This is often referred as the **word size**.

Each data unit is uniquely identifiable with the **address** of the data unit. The address is an essential parameter for load/store operations of a memory system



Classes and Hierarchy of Memory

Computer memory has a variety of classes. The classes are summarized in the following table.

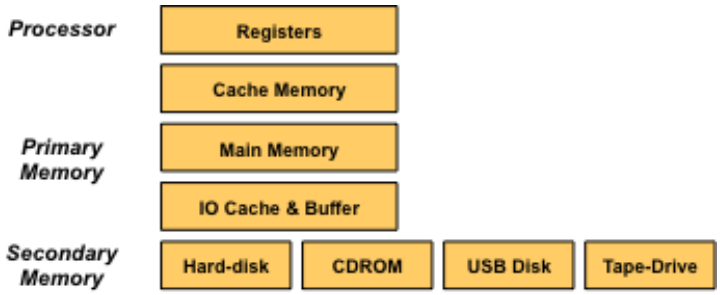
Volatility	**Volatile** Volatile memory requires electric power to keep the stored values	**Non-Volatile** Non-volatile memory can retain its value without electric power.
CPU Accessibility	**Primary** Directly accessible by the CPU	**Secondary** CPU access through indirect means of data transfer.
Mutability	**Mutable** Read and write allowed	**Immutable** Read only
Access Restriction	**Random Access** Any of the addressable unit can be directly accessed with a constant speed	**Sequential Access** Memory must be retrieved in an order

Memory technologies can also be described with the following attributes:

- Cost: the cost of manufacturing and maintenance. In the case where consumables are involved (such as tapes for a tape drive), the cost of consumables should be included. It may be measured in dollars per Mbytes.

- Compactness: the space occupied by memory can be an important consideration when it is integrated with other computer components. Usually the smaller the better.
- Throughput or data transfer rate: the time taken to transfer an amount of data. Usually measured in the same way as throughput in buses (Mbytes per second)
- Latency: the time taken for a memory system to begin data transfer. Some memory systems (such as CDROMs, hard disks or tape-drives) require a setup time after receiving instructions to perform read/write operations. Other memory systems purposely add latency in order to achieve a higher throughput (such as DDR RAM).

The following shows a typical memory hierarchy of a desktop computer. There are varies types of memory, each with different characteristics and purposes.



Memory in Processors

- Registers in processors are very fast, compact, and mutable memory. The operating speed must be fast enough to match the internal processor clock rate. It should be compact and therefore fit into the physical package of processors. Memory technology suitable for this purpose is costly and volatile.
- Cache memory is processors are used to mirror part of the main memory. If the required data and instruction is already in processor, then access to main memory through front-side bus can be avoided. Cache memory should also be fast, compact, and mutable. Therefore, the cost would prevent the amount of cache memory included.

Primary Memory

- Primary memory is the memory system that is directly addressable from the processor. In other words, data in the primary memory can be directly referred with instructions.
- Primary memory is often known as the Main Memory system. The Main Memory should be large size, less costly, and mutable. A large amount of Main Memory is critical for the execution of programs, especially in multi-programming systems. The Main Memory is often not physically contained within processor and compactness is not a major concern.
- IO Cache and Buffer is sometimes part of the Main Memory. They can make IO operations more efficient.

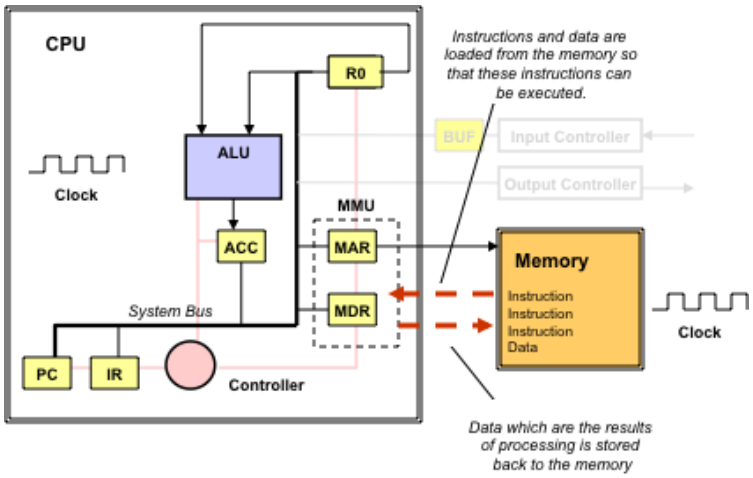
Secondary Memory

- Secondary memory provides long-term data and program storage.
- The demand for capacity is higher given the larger total amount of data handled by a computer system.
- It is often non-volatile and low-cost. The available technologies for secondary memory are slower and latency is often not a concern.
- The media for storage determines whether the secondary memory device is mutable.

The Main Memory

The Main Memory is the memory system that feeds the processor with instructions and data. It works closely with the processor in the fetch and execution cycle.

- In the fetch phase, processor needs to load the next instruction from the Main Memory.
- In the execution phase, processor may execute an instruction that involves data from the Main Memory.



The Memory Address Register (MAR) and the Memory Data Register (MDR) form the interface between the Main Memory and the CPU.

- The MAR specifies the address of the memory required.
- The MDR holds the data for the transaction.

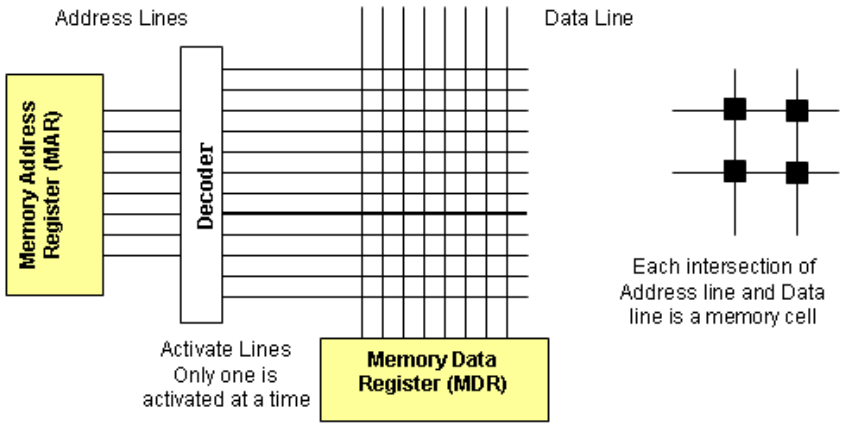
The MAR and the MDR are connected to the Main Memory in the following manner.

- The MAR holds the address in 8-bits or its multiple (depending on the addressable space). A decoder converts this 8-bit address into a set of activate lines, which only one is activated according to the value of MAR. The activated line connects to the memory cells of the address.
- The memory cells on the activated address line are connected to the MDR. Then MDR can either read values from the memory cells, or write new values to them.

The following lists the three main buses and signal lines involved in the operation of memory.

- There are usually 32 lines, 64 lines or 128 address lines corresponding to the address size of 32 bits, 64 bits and 128 bits. The number of address lines is exactly the size of MAR used in the CPU. One of the activate lines is activated according to the value represented by the address lines.
- There is usually a R/W line associated with the MAR/MDR to indicate whether this memory access is a read or a write operation.
- There are also multiple lines connecting the MDR to the cells of each memory address.

The following figure shows the design of a basic memory system:



The CPU and the MAR/MDR operates in the fetch phase of the fetch-execution cycle in following manner.

- The content of the Program Counter Register is copied to the MAR, which is the address storing the next instruction.
- The R/W line is set to read.
- The content of the given address is stored in the MDR (with previous value overwritten). The content (instruction) is copied to Instruction Register (IR).

The CPU then examines the instruction stored in the IR to determine the actions to follow. If the instruction is to store the value of an accumulator to a memory location (similar to the STO instruction in the LMC), then the following happens in the execution phase of the cycle.

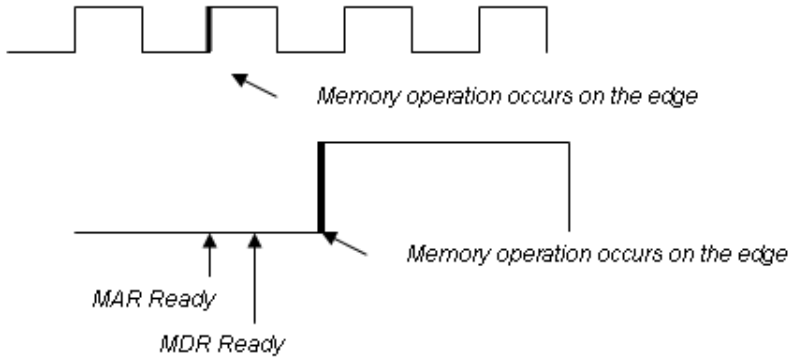
- The address part of the instruction is copied from the IR to the MAR, which is the address where data is to be stored.
- The R/W line is set to write.
- The data in the accumulator is copied to the MDR. The content of the MDR is stored to the memory cells activated by the MAR and the decoder.

In modern PC computer systems, the MAR and MDR are part of the Memory Management Unit (MMU) that also performs other memory related functions.

Operation of Main Memory Systems

Operation of the main memory system can occur when the MAR, MDR, and the address R/W line are loaded with data. The loading of data takes time. The operation is therefore synchronized with a memory clock so that the loading of data and the operation can take place at correct timings.

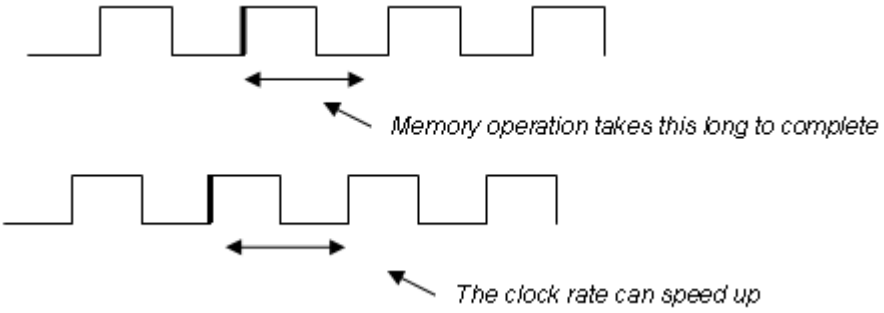
An electronic clock on a computer is a signal that goes between high and low repeatedly. Memory operation occurs according to the signal, and usually triggered by the edge (rising or falling edge) of the clock.



The rising edge or falling edge is useful because it represent a time instance that every data (or signal) involved are ready.

The clock rate (or frequency) has a bearing on the speed of the memory. A slow clock rate would means that the memory operation occurs less frequently, and therefore slowing the data movement.

However, we cannot simply increase the clock rate without making other considerations. Memory operations take time to complete. So the clock rate must allow the completion of one operation before triggering the next one.



Semiconductor Memory: Static RAM and Dynamic RAM

Current main memory system is based on semiconductor memory. A standard circuitry called flip-flop can store 1-bit of data. A memory system can be designed with millions of these circuitry integrated together.

- **Random access memory** (RAM) refers to such memory system in which the stored data to be accessed in any order.
- RAM based on flip-flops is called static RAM (SRAM). SRAM is fast and non-volatile as long as powered and volatile if there is no power.
- Each flip-flop is made up of 6 to 8 transistors, which can take up some space if larger memory size is to be packaged.

Packaged RAM chip is available in various standard shapes. The manufacturing process of RAM is similar to that of semiconductor microprocessors.

An alternative technology is called Dynamic RAM (DRAM).

- Dynamic RAM (DRAM) store data as charge on a capacitor, arranged in an array or table of cells. So the array of cells provides the storage of multiple bits of data.
- The capacitors used in DRAM tend to lose their charge quickly, and therefore require a periodic refresh cycle (in milliseconds) or data will be lost. Therefore a memory subsystem is required to support this refreshing.
- DRAM is less expensive, requiring more power, smaller in size, compared to SRAM.

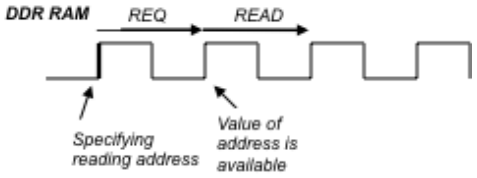
Improving the Throughput

DRAM is significantly slower than SRAM. A processor may have to wait for 4 to 6 cycles before DRAM can make the data available.

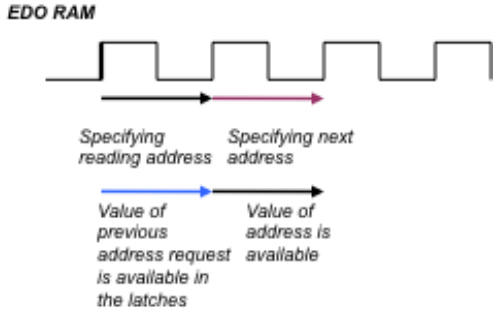
There are variants of DRAM that are designed to provide better data throughput through some clever designs.

Extended Data Out RAM (EDORAM)

EDORAM allows read and write operations to happen in a single cycle. This could double the throughput.



The above example shows that one memory operation requires two cycles (the real case should be more). The first cycle is REQ, and the second cycle is READ.



EDO RAM supports overlapped REQ/READ cycles.

- Each memory operation still requires two cycles, but each cycle two operations are happening: READ for the last operation and REQ for this operation.
- EDO RAM uses latches to cache the data of the previous operation. The data is available even after the computer starts specifying the address of the next location to be read. The data remains there for reading.
- Single cycle EDO RAM can carry out a memory access in 1 cycle, which other DRAM needs 2 to 3 cycles.

Burst EDO RAM further improves on EDO RAM as it allows 4 x REQ addresses at the same time. The latches are sufficiently large to cache all the data.

- Four consecutive addresses are requested. This exploits the common phenomenon of locality of references. Consecutive addresses in memory are often assessed one after another.

Video RAM (VRAM)

VRAM is designed for video adapters. VRAM systems are dual-port. It allows simultaneously read and write operations.

- RAM is normally a single port device. The CPU can perform reading or writing but not both at the same time.
- With VRAM, the PC can write into the memory to change what will be displayed, while the video adapter continuously reads the memory to refresh the monitor's display. The performance is greatly increased.

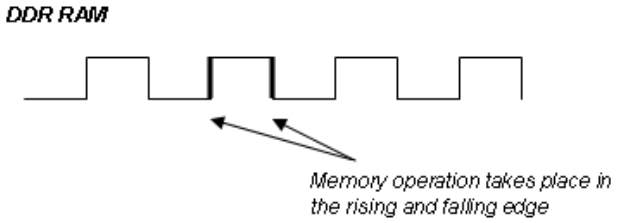
Early PCs supported 64K for video RAM, not 256 K. This might surprise you as the video card has 256K video RAM size. To fit this RAM into the 64K space, the RAM is paged. Programs can only access a small portion (or PAGE) of video RAM area at a time. Some newer cards map their entire memory directly into the PC's RAM space in high memory (above 1024K) hence creating a video aperture. Only Windows-based operating systems, not DOS, can support such cards.

This technology is superseded by DRAM technology.

Double Data Rate DRAM (DDR-RAM)

DDR RAM is the current mainstream memory technology.

- The DDR DRAM serves data in the beginning and ending phase of a memory cycle, therefore serving double amount of data.
- A bus frequency of 100 MHz can allow a single channel DDR RAM to serve 1.6 GB/s.
- PC-1600 64-bit DDR RAM using DDR-200 chips runs on 100MHz bus has a single channel output of 1.6GB/s.
    - Transfer rate = 100 MHz (memory clock rate) x 2 (for dual rate) x 64 (number of bits transferred) / 8 (number of bits/byte).



DDR RAM makes use of both rising and falling edge - double pumped.

- DDR-200 (PC-1600): 100MHz = 1.600 GB/s
- DDR-266 (PC-2100): 133MHz = 2.133 GB/s
- DDR-400 (PC-3200): 200MHz = 3.200 GB/s

DDR2 DRAM series allows internal memory clock to run faster than bus clock - fetch double more data on average in one cycle.

- DDR2-400 (PC2-3200): 100MHz (Memory) = 200MHz (IO) = 3.200 GB/s
- DDR2-800 (PC2-6400): 200MHz (Memory) = 400MHz (IO) = 6.400 GB/s
- Higher latency that makes DDR2-400 performing worse than original DDR

DDR3 DRAM series allows internal memory clock to run faster than memory clock - fetch quadruple more data in one cycle.

- DDR3-800 (PC3-6400): 100MHz (Memory) = 400MHz (IO) = 6.400 GB/s
- Even higher latency that results in long access time.

EEPROM and Flash Memory

EEPROM (Electrically Erasable Programmable Read-Only Memory) and flash memory are non-volatile memory allowing rewriting of data. It uses a technique called Fowler-Nordheim tunnelling that allows the trapping of electrons within insulator to record the state of a bit.

Flash memory is the latest form of EEPROM, allowing faster rewriting of data of a whole blocks.

5. Input and Output Devices

IO design is an often-overlooked issue. Many people are more concerned about the CPU speed. However, the performance of a computer system often rests on IO performance. IO devices are significantly slower than CPU and memory speed.

There are many attributes separating one IO device from another. The following lists the major characteristics:

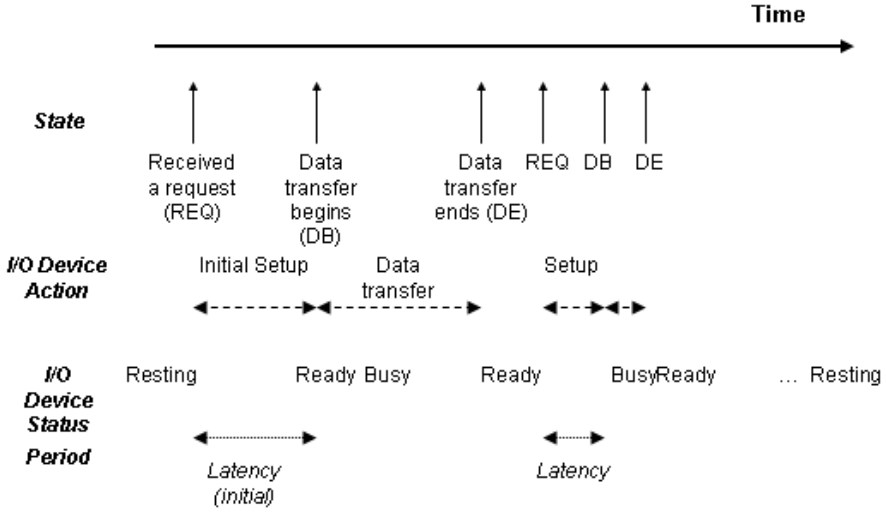
- Transfer data unit: character-stream or block transfer.
- Relation between IO operations and programs: synchronous or asynchronous data transfer.
- Data access order: sequential or random access.
- IO device exclusiveness: Sharable or dedicated device.
- Data mutability: read and write allowances.
- IO device latency: fast or slow setup time.
- IO operation speed: high or low data transfer rate.

For example, a keyboard is a character-stream, asynchronous, sequential, dedicated, read-only, fast setup, and low data transfer rate IO device.

On the other hand, an electro-mechanical hard disk is a block transfer, asynchronous, random access, sharable, read and write, slow setup time, and high data transfer rate IO device.

IO Operation and Latency

Latency is a major issue in IO operations. The following diagram explains the detail stages of IO operations.



Many IO devices are mechanical-electronic devices

- Mechanical-electronic devices will rest to save energy if there is no action required.
- From a device changing from a resting state to a ready state would take time (for example switching on the motors). This is called the **initial setup time**. The time required could be quite long.
- After the device is ready, then data transfer can occur and the **rate of transfer** depends on the device.
- The device will then complete the data transfer and wait for another request.
- Upon receiving another request, the device will still take some time to begin the data transfer. This is called the **latency time**.

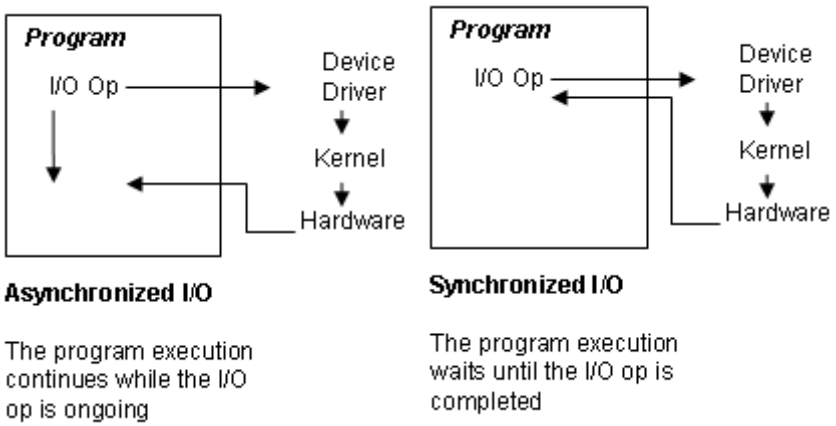
Synchronous and Asynchronous IO

Two possible methods of handling IO exist.

- Control does not return to user program until IO is completed.
- Control returns to user program after registering with the request. The user program is notified of the completion of the IO with interrupts.

The first method is known as **synchronous IO** and the second method **asynchronous IO**. We need different architectures and services to handle these types of IO.

- Advantage. Asynchronous IO allows the user program to do something else while the IO device is handling the request.
- Disadvantage. Asynchronous IO is more difficult to for the programmer to write program to manage exceptional situations, such as an error occurring while the program is doing something else.



Characteristics of synchronous IO include the following.

- Wait instruction idles the CPU until the next interrupt
- Wait loop (contention for memory access).
- At most one IO request is outstanding at a time, no simultaneous IO processing.

Characteristics of asynchronous IO include the following.

- System call, which is a request sent to the operating system to allow user to wait for IO completion.
- Device-status table contains entry for each IO device indicating its type, address, and state.
- Operating system indexes into IO device table to determine device status and to modify table entry to include interrupt.

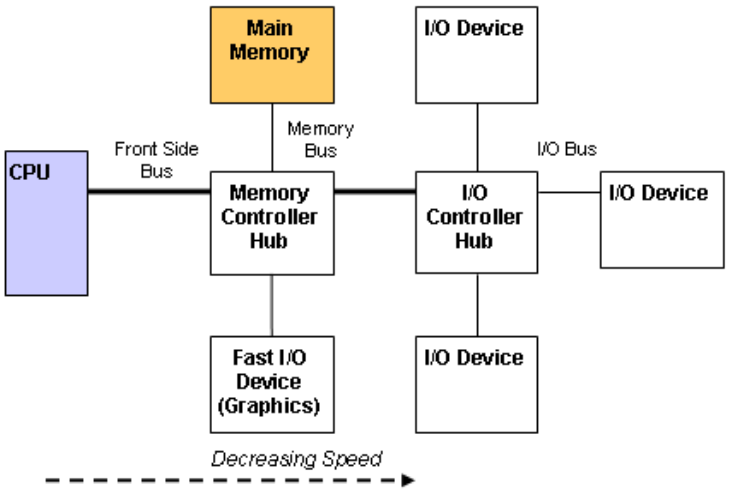
Asynchronous IO allows concurrent IO operations to more than one device. A **device-status table** is needed to book-keep the state. Each entry includes device type, address, and state. The OS also maintain a wait queue for each IO device.

IO Design for Computer Systems

IO devices are running at a significantly slower speed compared to the CPU and the Memory System.

The basic design strategy is to separate them into different worlds of speed, in the same way as the Memory System is separated from the CPU.

The following figure shows how IO devices are connected to IO controller hub. The hub is then connected to the memory controller hub, before reaching the CPU. The bus speed is decreasing as the bus moves further away from the CPU.

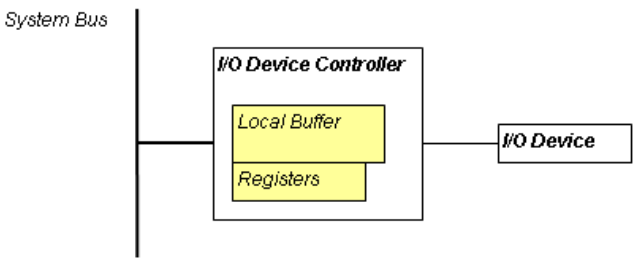


IO devices are connected to the bus leading out from the IO controller hub.

- Each **device controller** handles a specific type of device. Sometimes one controller can manage more than one device (such as SCSI).
- Each device controller has a local buffer, which is used to hold data when data is reading/writing between the computer system and a device.
- Device controllers operate independently from the CPU.

CPU instructs the device controller by loading the appropriate registers.

- Device controller examines the registers to see what instruction is there.
- The device controller notifies the CPU of the completion of the instruction by triggering an interrupt.
- CPU can move the data to/from local buffer to main memory.



Sending instructions to IO devices

The IO device is now separated from the CPU by at least two controllers in the current programmable computer design. The CPU cannot directly send data or signals to IO devices.

While the CPU can directly send data to the Memory System through the MAR/MDR registers, there is no such mechanism built in for IO device.

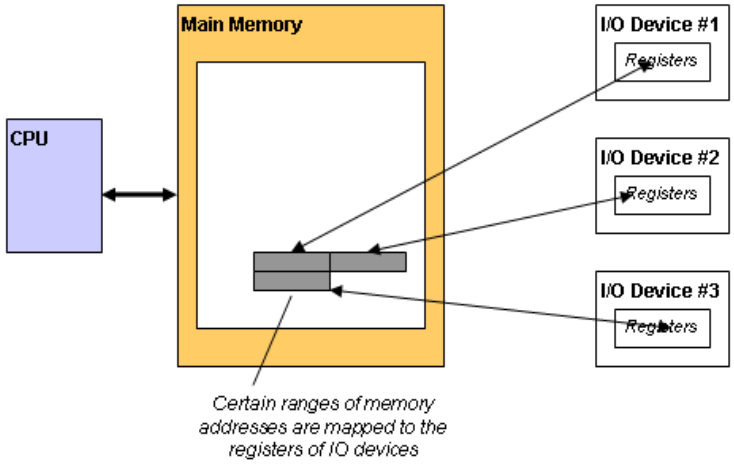
Port-mapped IO

Port-mapped IO uses dedicated instructions for IO operations. An example is the LMC instructions IN and OUT. These instructions are handled directly by the CPU and the CPU sends signals directly to the IO devices to carry out the operation. The IO devices have their own memory space for data movement between the CPU and the IO device.

Memory-mapped IO

Another solution to this problem is to provide **memory-mapped IO**. Memory-mapped IO unified the access to the Memory System and access to IO devices. Sending signals or data to an IO device is now done by writing data to the Memory System. Certain areas in the Memory System are declared special places. Any data written to one of the areas is read and handled by an IO device. On the other hand, an IO device sending data back by writing to the areas.

The following figure illustrates how memory-mapped IO operates.



CPU transfer data to an IO device by writing instruction to the mapped data registers and set the control register appropriately. The device controller monitors the control register, takes the data, and then clears the control registry for next data transfer.

Signalling from IO Devices

The CPU has one of the two following options after sending an instruction to an IO device:

- Synchronous IO mode. The CPU will wait and keep **polling** if the IO operation is still going on.
- Asynchronous IO mode. The CPU will forget about the IO operation for the time being and does something else.

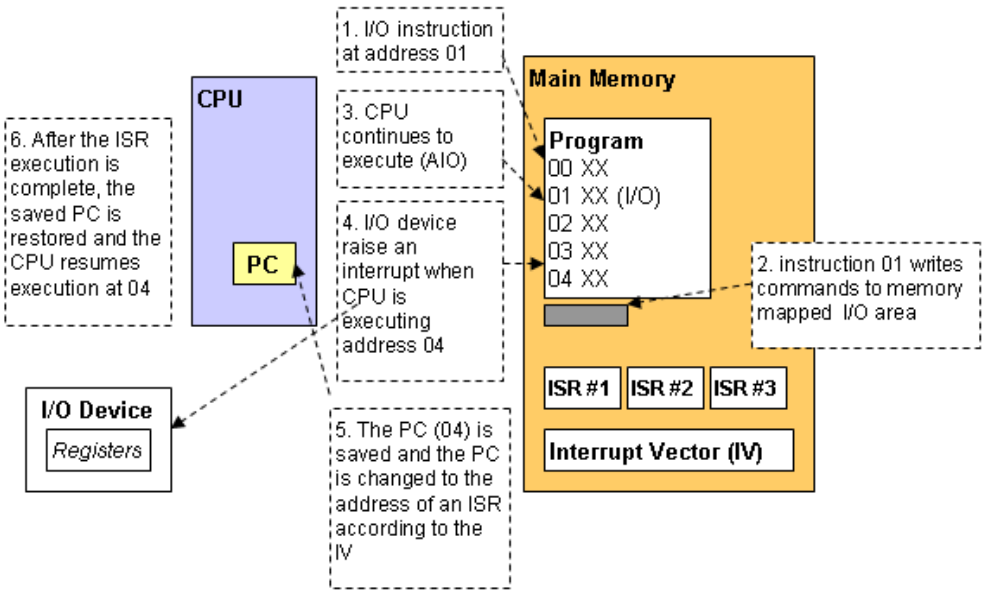
In asynchronous IO mode, the CPU will wait for a signal from the IO device when the IO operation is complete or an error has occurred. The signal is known as **interrupt**.

IO Interrupt is handled in the following steps.

- After the CPU receives an interrupt signal, the execution of the current program is suspended.
- The program counter (PC) is saved so that the execution will return to the suspended place later.
- The controller then refers to the type of the interrupt and the CPU is made to execute a segment of code according to an **interrupt vector**.
- Interrupt vector is a table of pointers usually stored in the lower part of the memory.
- The pointers are the starting addresses of **interrupt service routines (ISR)** that are designed to handle a particular type of interrupt.

After the interrupt is handled, the CPU is made to return to executing the address of the program when the interrupt occurred.

The following figure illustrates the steps involved in handling an IO interrupt.



6. Input and Output Device Case Study: Hard Disk

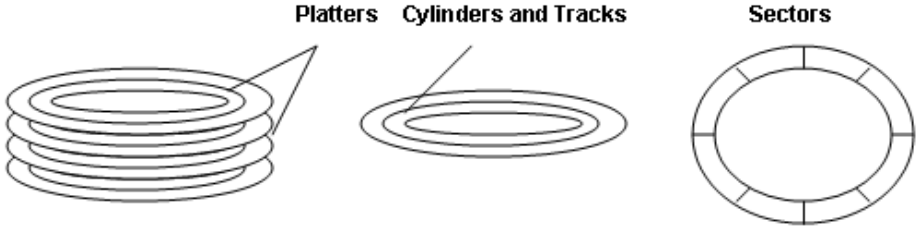
Hard disk is arguably the most important IO device of modern computer systems. Hard disks are currently based on magnetic disks technology, which has been serving us since 1960's despite facing a lot of challenges from other technologies.

Hard disks contribute to modern computer systems in two major ways.

- Provide non-volatile long-term storage for data and files.
- Provide secondary memory to supplement the main memory during the operation of computers. Data from the main memory can be moved temporarily to hard disk to spare some space for other data.

Structure of Magnetic Disks

Magnetic disks are physically composed of platters of solid disks stacked up on a rotational spindle.



Platters are usually made of metal or glass, deposited with magnetic materials on both sides. So one platter has two surfaces for data storage. Each platter is divided into a number of cylinders, and then tracks. Typically there are tens of thousands of tracks on a platter.

Each track is further divided into sectors. A sector is the smallest unit involved in read/write. Because the outer tracks are longer, usually more sectors are designated there. This scheme is called **constant bit density**.

To perform a read/write operation, a moving arm with a read/write head is moved over the track of the desired sector. This is called a seek operation and the time required to move the arm is called the **seek time**.

After the read/write head has moved to the desired track, it may not be over the desired sector. The time for the head to move over the desired sector is called **rotational delay** or **rotational latency**. It needs to wait until the rotation of the platter. If the platter is not already moving, further delay would be taken into consideration.

Read/Write Performance

The time taken to read/write data on magnetic hard disks must take the following overhead into consideration.

- Seek time. The time taken for the read/write arm to move over the desired track.
- Rotational Delay. The time taken for the desired sector to be rotated under the read/write head.
- Controller time. The time taken for the IO controller to process an IO request.
- Queuing time or queuing delay. A hard disk can serve one request each time. Other requests must wait and queue for the service for the hard disk.

A typical performance specification of a modern magnetic hard disk is shown in the following.

Fujitsu Hard Disk MHT2160BT

Model	***MHV2160BT***
Storage capacity(formatted)	160.0 GB
Bytes/Sector	512
Seek time	Track to track 1.5 ms typ.
Average	12 ms typ.(Read), 14 ms typ.(Write)
Maximum	22 ms typ.
Rotational speed	4,200 RPM
Data transfer to/from host	150 MB/s
Interface	SATA
Buffer size	8MB

Data obtained from
http://www.fujitsu.com/tw/services/computer/harddisk/MHV2160BT.html

Exercises: Read/Write Time

Question: Calculate the time taken in a read/write operation on the hard disk. Given that the size of one sector is 512 bytes. The controller time is 0.1ms. The disk is free of queue and it is available.

Answer:

The time taken to transfer 512 bytes can be worked out as the following.

Average disk access is the sum of the following: average seek time, average rotational latency, transfer time, and controller time.

Average seek time = 12 ms

Average rotational latency = 50% * (1 / 4200 RPM) = 7.1 ms

Transfer time = 512 bytes / 150 MB/s = 0.003 ms

Controller overhead = 0.1 ms

Overall average disk access = 12ms + 7.1ms + 0.003ms + 0.1ms = 19.2ms

Note that the data transfer (0.003ms) only contributes a small percentage of the average disk access. The various seek time (including the average seek time and rotational latency) factors are predominant

To speed up the disk access, we should first focus on the seek time which contributes 90 percent of the overall disk access time.

To minimize seek time, one can read more data than request, and hoping that the next requests happen to use the data **read-ahead**. The success of read-ahead lies on the observation that requests has a property of **spatial locality**. When data is stored on hard disks, the data is arranged in a sequence on neighbouring sectors and tracks.

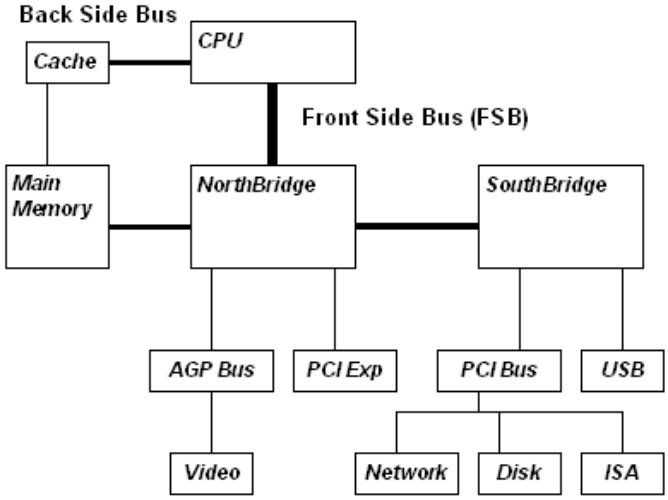
7. Integration: PC Motherboard Design

The motherboard is a printed circuit board where the major computer components are integrated together. In addition to the circuitry connecting the components, it also provides power, cables, connectors, and physical housing.

A motherboard is designed around the features of the main processor. Apart from the main processor, it contains a **chip set** for providing other functions such as memory and IO control.

Classic PC Motherboard Design

The following shows the classic PC Motherboard design for Pentium series processor.



There are two chips that work with the CPU: one is called **memory controller hub (north-bridge)** and the other **IO controller hub (south-bridge).**

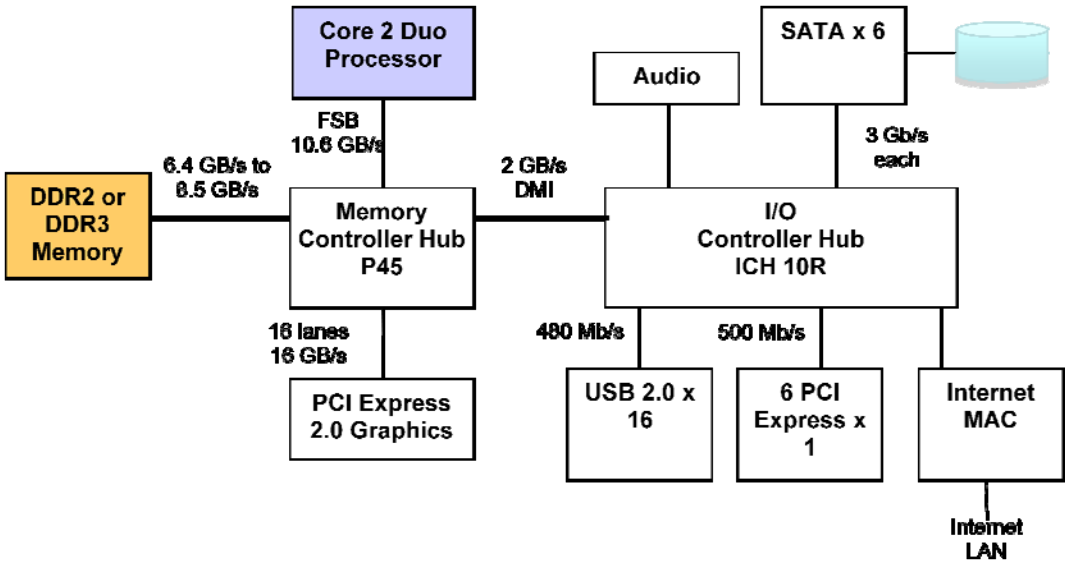
- The north-bridge connects the CPU and the following components: main memory, AGP bus (video), and the south-bridge. The north-bridge determines the performance of the data transfer between the memory and the CPU (often the deciding factor in system performance), and the type and amount of memory that can be used.
- The south-bridge is detached from the CPU, and it is responsible for handling slower communications. The separation of the two allows the critical high speed transfer between the CPU and memory to happen without the intervention of the slower communication between peripheral devices, which is the main role of the south-bridge.
- The south-bridge includes an interrupt controller that allows peripheral devices to alert the CPU.
- The south-bridge also includes a DMA controller that allows data transfer between IDE hard disks and the main memory.
- The CPU and the north-bridge are connected with a high-speed bus known as the front-side bus (FSB). The speed of CPU is determined from the speed of the front-side bus times a multiplier. Example Intel technology for FSB is known as GTL+ and AGTL+.

The CPU connects to the L2 Cache Memory through the backside bus.

The FSB is often regarded as the bottleneck of performance of this classic PC design. The all-important memory to CPU data transfer running on the FSB is shared with data write-back, and data of IO operations.

Motherboard for Core Duo Processor

The following diagram is adapted from Intel information, showing the specific components and the bus data rates of a motherboard design for Intel Core Duo processor.



Motherboard for Core i7 Processor

Core i7 is the current top of the range processor offered by Intel. The following diagram is adapted from Intel information, showing the FSB is now replaced with Quick-Path Inter-Connect (QPI).

The memory controller is within the Core i7 processor, allowing a direct access to DDR3 DRAM.