4.4 SERIAL BUS ARCHITECTURES

The peripheral devices and external buffer that operate at relatively low frequencies communicate with the processor using serial bus. There are two popular serial buses: Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C).

Serial Peripheral Interface (SPI)

> ***Serial Peripheral Interface (SPI) is an interface bus designed by Motorola to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device.***

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave-devices are supported through selection with individual slave select (SS), sometimes called chip select (CS), lines.

Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface,[1] but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. The SSI protocol employs differential signaling and provides only a single simplex communication channel. SPI is one master and multi slave communication.

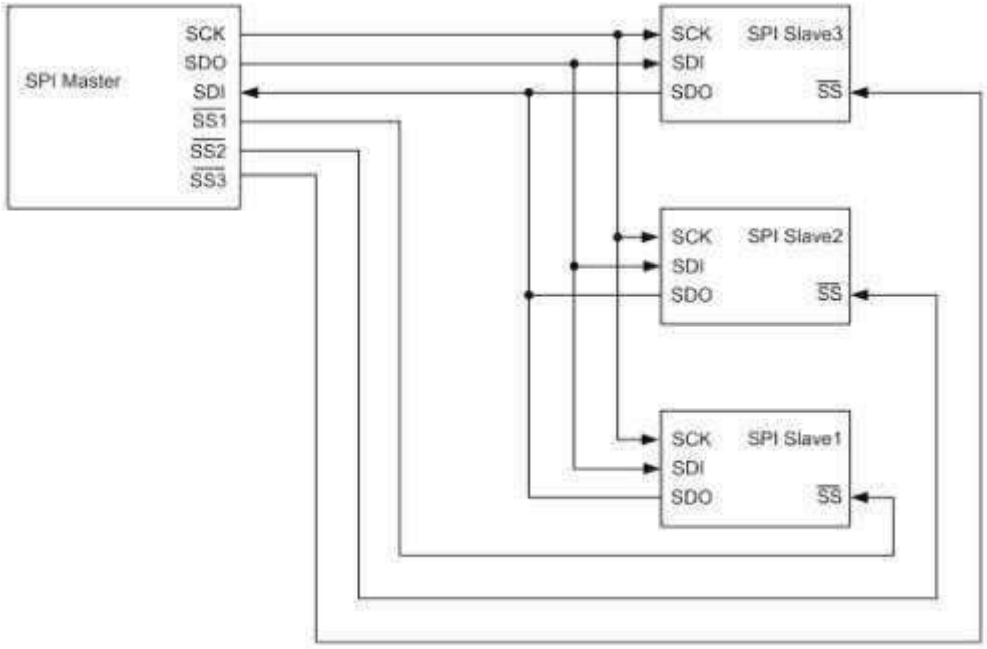


Fig 1: SPI master with three slaves

Source: Miles J. Murdocca and Vincent P. Heuring, —"Computer Architecture and Organization: An Integrated approach"

Modes in SPI

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

- In addition to the standard 4-wire configuration, the SPI interface has been extended to include a variety of IO standards including 3-wire for reduced pin count and dual or quad I/O for higher throughput.
- In 3-wire mode, MOSI and MISO lines are combined to a single bidirectional data line.
- Transactions are half-duplex to allow for bidirectional communication.

Reducing the number of data lines and operating in half-duplex mode also decreases maximum possible throughput; many 3-wire devices have low performance requirements and are instead designed with low pin count in mind.

- Multi I/O variants such as dual I/O and quad I/O add additional data lines to the standard for increased throughput.
- Components that utilize multi I/O modes can rival the read speed of parallel devices while still offering reduced pin counts. This performance increase enables random access and direct program execution from flash memory (execute-in-place).

Inter-Integrated Circuit (I2C)

An inter-integrated circuit (Inter-IC or I2C) is a multi-master serial bus that connects low-speed peripherals to a motherboard, mobile phone, embedded system or other electronic devices.

- Philips Semiconductor created I2C with an intention of communication between chips reside on the same Printed Circuit Board (PCB).
- It is a multi-master, multi-slave protocol.
- It is designed to lessen costs by streamlining massive wiring systems with an easier interface for connecting a central processing unit (CPU) to peripheral chips in a television.
- It had a battery-controlled interface but later utilized an internal bus system.
- It is built on two lines
- SDA (Serial Data) – The line for the master and slave to send and receive data
- SCL (Serial Clock) – The line that carries the clock signal.
- Devices on an I2C bus are always a master or a slave. Master is the device which always initiates a communication and drives the clock line (SCL). Usually a microcontroller or microprocessor acts a master which needs to read

data from or write data to slave peripherals.

- Slave devices are always responds to master and won't initiate any communication by itself. Devices like EEPROM, LCDs, RTCs acts as a slave device. Each slave device will have a unique address such that master can request data from or write data to it.
- The master device uses either a 7-bit or 10-bit address to specify the slave device as its partner of data communication and it supports bi-directional data transfer.

Working of I2C

- The I2C, data is transferred in messages, which are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted.
- The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame.
- The following are the bits in data frames:

1. Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
2. Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.
3. Address Frame: A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
4. Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
5. ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

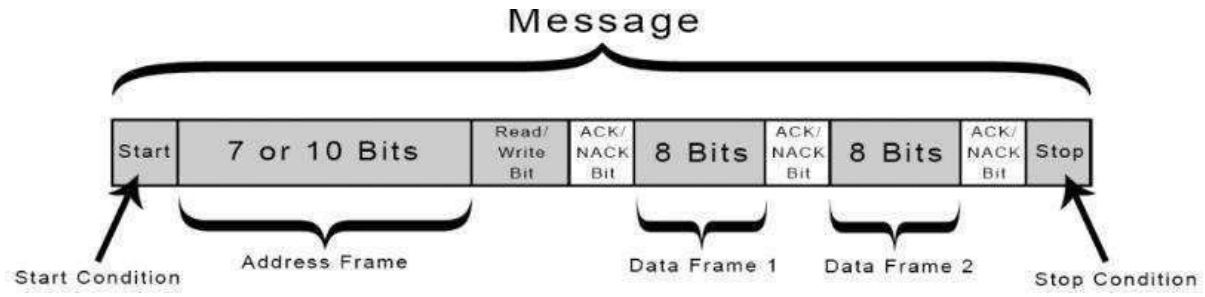


Fig 2: I_2C Message Format

Source: Miles J. Murdocca and Vincent P. Heuring, —"Computer Architecture and Organization: An Integrated approach"

Addressing:

I_2C doesn't have slave select lines like SP), so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by addressing. The address frame is always the first frame after the start bit in a new message

- The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address.
- If the address matches, it sends a low voltage ACK bit back to the master. If the address

Doesn't match, the slave does nothing and the SDA line remains high.

Read/Write Bit

- The address frame includes a single bit at the end that informs the slave whether the master wants to write data to it or receive data from it. If the master wants to send data to the slave, the read/write bit is a low voltage level. If the master is requesting data from the slave, the bit is a high voltage level.

Data Frame

- After the master detects the ACK bit from the slave, the first data frame is ready to be sent.

- The data frame is always 8 bits long, and sent with the most significant bit first.
- Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully.
- The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent.
- After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission.
- The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

Steps in Data transmission

1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level before switching the SCL line from high to low.
   The master sends each slave the 7 or 10 bit address of the slave it wants to communicate with, along with the read/write bit.
2. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the SDA line high.
3. The master sends or receives the data frame.
4. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.
5. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high.

Advantages

- It uses two wires.

- This supports multiple masters and multiple slaves.
- ACK/NACK bit gives confirmation that each frame is transferred successfully.
- Well known and widely used protocol

Disadvantages

- Slower data transfer rate than SPI.
- The size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI.

Mass storage refers to various techniques and devices for storing large amounts of data. Mass storage is distinct from memory, which refers to temporary storage areas within the computer. Unlike main memory, mass storage devices retain data even when the computer is turned off.

MASS STORAGE

The mass storage medium includes:

- solid-state drives (SSD)
- hard drives
- external hard drives
- optical drives
- tape drives
- RAID storage
- USB storage
- flash memory cards

Solid State Devices

- Solid-state devices are electronic devices in which electricity flows through solid semiconductor crystals like silicon, gallium arsenide, and germanium rather than through
  vacuum tubes.
- It do not involve any moving parts or magnetic materials.
- RAM is a solid state device that consists of microchips that store data on non-

moving components, providing for fast retrieval of that data.

- Transistors are the most important solid state devices. The transistors contain two p– n junctions, have three contacts or terminals.
- They require the action of perpendicular electrical fields, their behavior is more difficult to understand than that of diodes.
- The different types of transistors are: bipolar junction transistor (BJT) where the current is amplified, while in the field effect transistor (FET) a voltage controls a current.
- In a solid-state component, the current is confined to solid elements and compounds engineered specifically to switch and amplify it.
- Current flows in two forms: as negatively charged electrons, and as positively charged electron deficiencies called holes.
- In some semiconductors, the current consists mostly of electrons; in other semiconductors, it consists mostly of holes. Both the electron and the hole are called charge carriers.

Hard Drives

- A hard disk drive is a non-volatile memory hardware device that permanently stores and retrieves data on a computer.
- A hard drive is a secondary storage device that consists of one or more platters to which data is written using a magnetic head, all inside of an air-sealed casing.
- Internal hard disks reside in a drive bay, connect to the motherboard using an ATA, SCSI, or SATA cable, and are powered by a connection to the power supply unit.

External Hard Drives

- An external hard drive is a portable storage device that can be attached to a computer through a USB or FireWire connection, or wirelessly.
- External hard drives typically have high storage capacities and are often used

to back up computers or serve as a network drive.

Optical Drives

- An Optical Drive refers to a computer system that allows users to use DVDs, CDs and Blu- ray optical drives.
- The drive contains some lenses that project electromagnetic waves that are responsible for reading and writing data on optical discs.
- An optical disk drive uses a laser to read and write data. A laser in this context means an electromagnetic wave with a very specific wavelength within or near the visible light spectrum.
- An optical drive that works with all types of discs will have two separate lenses: one for CD/DVD and one for Blu-ray.
- An optical drive has a rotational mechanism to spin the disc. Optical drives were originally designed to work at a constant linear velocity (CLV) (i.e.) the disc spins at varying speeds depending on where the laser beam is reading, so the spiral groove of the disc passes by the laser at a constant speed.
- An optical drive also needs a loading mechanism: A tray-loading mechanism, where the disc is placed onto a motorized tray, which moves in and out of the computer case and slot- loading mechanism, where the disc is slid into a slot and motorized rollers are used to move the disc in and out.

Tape disks

- A tape drive is a device that stores computer data on magnetic tape, especially for backup and archiving purposes.
- Tape drives work either by using a traditional helical scan where the recording and playback heads touch the tape, or linear tape technology, where the heads never actually touch the tape.
- Drives can be rewinding, where the device issues a rewind command at the end of a session, or non-rewinding.

- Rewinding devices are most commonly used when a tape is to be unmounted at the end of a session after batch processing of large amounts of data.
- Non-rewinding devices are useful for incremental backups and other applications where new files are added to the end of the previous session's files.
- The different types of tapes are audio, video and data storage tape.

Redundant Array of Inexpensive Disks (RAID) Storage

- RAID is a way of storing the same data in different places on multiple hard disks to protect data in the case of a drive failure.
- RAID works by placing data on multiple disks and allowing input/output (I/O) operations to overlap in a balanced way, improving performance. Because the use of multiple disks increases the mean time between failures (MTBF), storing data redundantly also increases fault tolerance.
- A RAID controller can be used as a level of abstraction between the OS and the physical disks, presenting groups of disks as logical units. Using a RAID controller can improve performance and help protect data in case of a crash.

- Levels in RAID:

1. RAID 0 (Disk striping):

   RAID 0 splits data across any number of disks allowing higher data throughput. An individual file is read from multiple disks giving it access to the speed and capacity of all of them. This RAID level is often referred to as striping and has the benefit of increased performance.

2. RAID 1 (Disk Mirroring):

   RAID 1 writes and reads identical data to pairs of drives. This process is often called data mirroring and it┼ s a primary function is to provide redundancy. If any of the disks in the array fails, the system can still access data from the remaining disk(s).

3. RAID 5 (Striping with parity):

RAID 5 stripes data blocks across multiple disks like RAID 0, however, it also stores parity information (Small amount of data that can accurately describe larger amounts of data) which is used to recover the data in case of disk failure. This level offers both speed (data is accessed from multiple disks) and redundancy as parity data is stored across all of the disks.

4. RAID 6 (Striping with double parity):

   Raid 6 is similar to RAID 5, however, it provides increased reliability as it stores an extra parity block. That effectively means that it is possible for two drives to fail at once without breaking the array.

5. RAID 10 (Striping + Mirroring):

   RAID 10 combines the mirroring of RAID 1 with the striping of RAID 0. Or in other words, it combines the redundancy of RAID 1 with the increased performance of RAID 0. It is best suitable for environments where both high performance and security is required.

Universal Serial Bus (USB) Devices

- USB is a system for connecting a wide range of peripherals to a computer, including pointing devices, displays, and data storage and communications products.
- The Universal Serial Bus is a network of attachments connected to the host computer.
- These attachments come in two types known as Functions and Hubs.
- Functions are the peripherals such as mice, printers, etc.
- Hubs basically act like a double adapter does on a power-point, converting one socket, called a port, into multiple ports.
- Hubs and functions are collectively called devices.
- When a device is attached to the USB system, it gets assigned a number called its address.

  The address is uniquely used by that device while it is connected.

- Each device also contains a number of endpoints, which are a collection of sources and destinations for communications between the host and the device. The combination of the address, endpoint number and direction are what is used by the host and software to determine along which pipe data is travelling.

Flash Drives

- A flash drive stores data using flash memory. Flash memory uses an electrically erasable programmable read-only (EEPROM) format to store and retrieve data.
- Flash drives are non-volatile, which means they do not need a battery backup.
- Most computers come equipped with USB ports, which detect inserted flash drives and install the necessary drives to make the data retrievable.
- Computer users can store and retrieve data once the operating system has detected a connection to the USB port.
- Flash drives have a USB mass storage device classification, which means they do not require additional drivers.
- The computer┼s operating system recognizes a block-structured logical unit, which means it can use any file system or block addressing system to read the information on the flash drive.
- A flash drive enters emulation mode, or acts a hard drive, once it has connected to the USB port. This makes it easier to transfer data between the flash drive and the computer.
- Flash memory is known as a solid state storage device, meaning there are no moving parts — everything is electronic instead of mechanical.

4.2 MEMORY CHIP ORGANISATION

A memory consists of cells in the form of an array. The basic element of the semiconductor memory is the **cell**. Each cell is capable of storing one bit of information. Each row of the cells constitutes a memory words and all cells of a row are connected to a common line referred to as a **word line.** AW×b memory has w words, each word having b number of bits. The bellow Fig 1 shows the 16x8 memory.

The basic memory element called cell can be in two states (0 or 1). The data can be written into the cell and can be read from it.

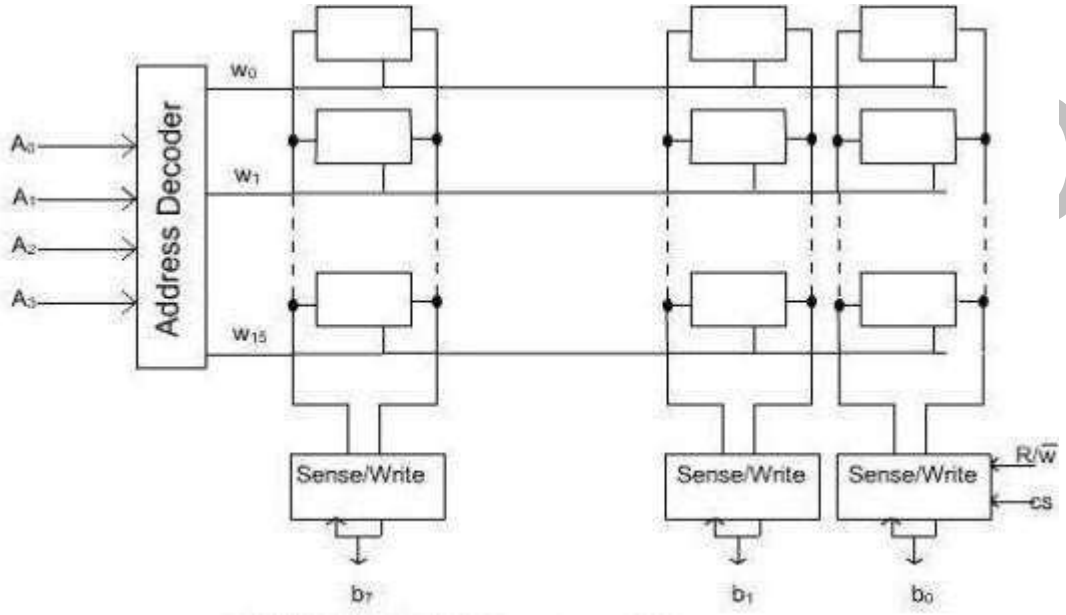


Fig 1: Organization of 16 x 8 memory

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

- In the above diagram there are 16 memory locations named as w_0, w_1, $w_3 \ldots w_{15}$. Each location can store at most 8 bits of data (b_0, b_1, $b_3 \ldots$ b_7). Each location (w_n)

is the word line. The word line of Fig 4.4 is 8.

- Each row of the cell is a memory word. The memory words are connected to a common line termed as word line. The word line is activated based on the address it receives from the address bus.
- An address decoder is used to activate a word line.
- The cells in the memory are connected by two **bit lines** (column wise). These are connected to data input and data output lines through sense/ write circuitry.
- **Read Operation:** During read operation the sense/ write circuit reads the information by selecting the cell through word line and bit lines. The data from this cell is transferred through the output data line.
- **Write Operation:** During write operation, the sense/ write circuitry gets the data and writes into the selected cell.
- The data input and output line of sense / write circuit is connected to a bidirectional data line.
- It is essential to have n bus lines to read 2^n words.

Organization of 1M x 1 memory chip:

The organization of 1024 x 1 memory chips, has 1024 memory words of size 1 bit only. The size of data bus is 1 bit and the size of address bus is 10 bits. A particular memory location is identified by the contents of memory address bus. A decoder is used to decode the memory address.

Organization of memory word as a row:

The whole memory address bus is used together to decode the address of the specified location.

The bellow Fig 2 represents Organization of memory word as row

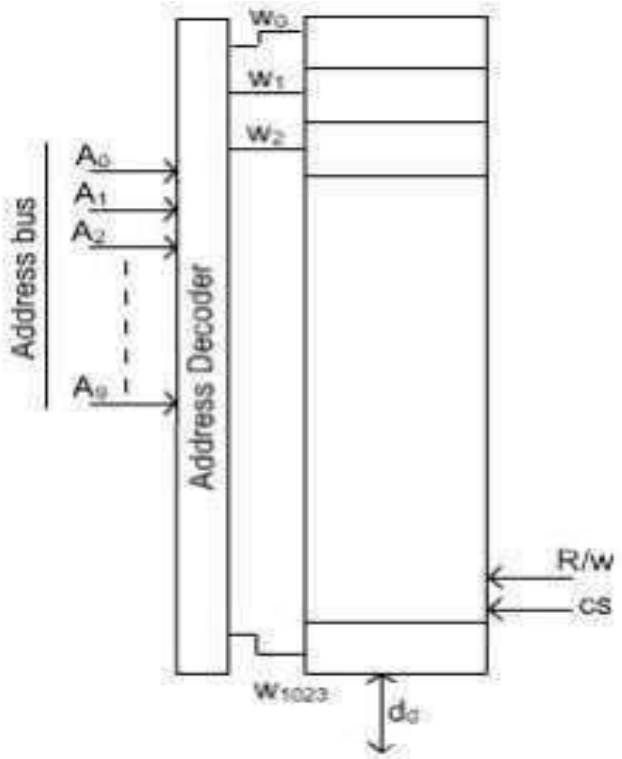


Fig 2: Organization of memory word as row

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Organization of several memory words in row:

The bellow Fig represents Organization of several memory word as row.

- One group is used to form the row address and the second group is used to form the column address.
- The 10-bit address is divided into two groups of 5 bits each to form the row and column address of the cell array.
- A row address selects a row of 32 cells, all of which could be accessed in parallel. Regardingthe column address, onlyone ofthese cells is connected to the externaldata line via the input output multiplexers

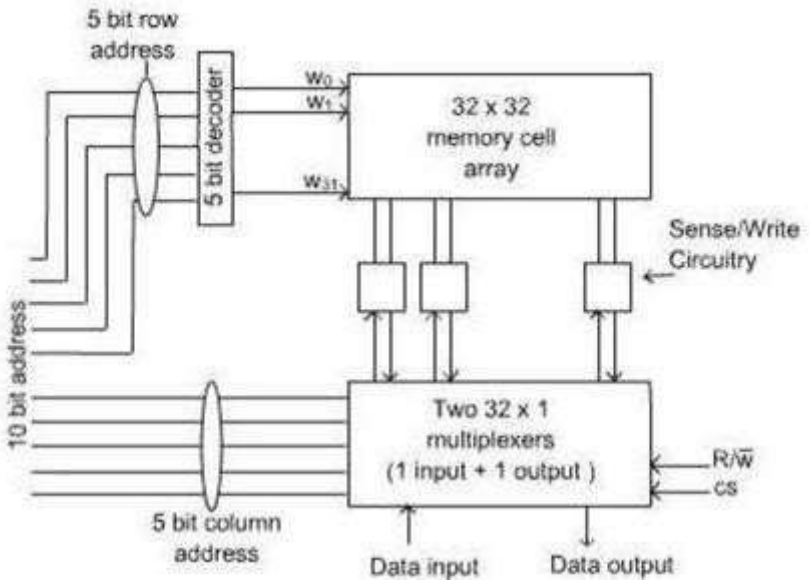


Fig 3: Organization of several memory words in row

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Signals used in memory chip:

- A memory unit of 1MB size is organized as 1M x 8 memory cells.
- It has got220 memory location and each memory location contains 8 bits of information.
- The size of address bus is20 and the size of data bus is 8.
- The number of pins of a memory chip depends on the data bus and address bus of the memory module.
- To reduce the number of pins required for the chip, the cells are organized in the form of a square array.
- The address bus is divided into two groups, one for column address and other one is for row address.
- In this case, high- and low-order 10 bits of 20-bitaddress constitute of row and column address of a given cell, respectively.
- In order to reduce the number of pin needed for external connections, the row and

column addresses are multiplexed on tenpins.

- During a Read or a Write operation, the row address is applied first. In response to a signal pulse on the Row Address Strobe (RAS) input of the chip, this part of the address is loaded into the row address latch.
- All cell of this particular row is selected. Shortly after the row address is latched, the column address is applied to the address pins. The bellow Fig 4 shows signals in accessing the memory.
- It is loaded into the column address latch with the help of Column Address Strobe (CAS) signal, similar toRAS.
- The information in this latch is decoded and the appropriate Sense/Write circuit is selected.

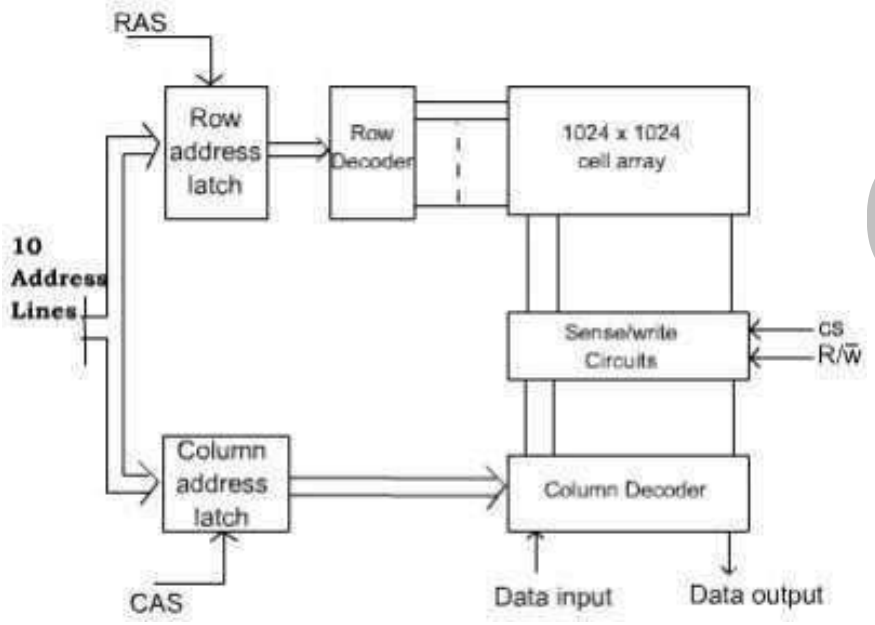


Fig 4: Signals in accessing the memory

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

- Each chip has a control input line called Chip Select (CS). A chip can be enabled to accept data input or to place the data on the output bus by setting its

Chip Select input to 1.

- The address bus for the 64K memory is 16 bits wide.
- The high order two bits of the address are decoded to obtain the four chip select control signals.
- The remaining 14 address bits are connected to the address lines of all the chips.
- They are used to access a specific location inside each chip of the selected row.
- The R/ W inputs of all chips are tied together to provide a common read / write control.

4.2.1 CACHE MEMORY

The cache memory exploits the locality of reference to enhance the speed of the processor.

Cache memory or CPU memory, is high-speed SRAM that a processor can access more quickly than a regular RAM. This memory is integrated directly into the CPU chip or placed on a separate chip that has a separate bus interconnect with the CPU.

The cache memory stores instructions and data that are more frequently used or data that is likely to be used next. The processor looks first in the cache memory for the data. If it finds the instructions or data then it does perform a more time-consuming reading of data from larger main memory or other data storage devices.

The processor do not need to know the exact location of the cache. It can simply issue read and write instructions. The cache control circuitry determines whether the requested data resides in the cache.

- **Cache and temporal reference:** When data is requested by the processor, the data should be loaded in the cache and should be retained till it is needed again.
- **Cache and spatial reference:** Instead of fetching single data, a contiguous block of data is loaded into the cache.

Terminologies in Cache

- **Split cache:** It has separate data cache and a separate instruction cache. The two caches work in parallel, one transferring data and the other transferring

instructions.

- **A dual or unified cache:** The data and the instructions are stored in the same cache. A combined cache with a total size equal to the sum of the two split caches will usually have a better hit rate.
- **Mapping Function:** The correspondence between the main memory blocks and those in the cache is specified by a mapping function.
- **Cache Replacement:** When the cache is full and a memory word that is not in the cache is referenced, the cache control hardware must decide which block should be removed to

  create space for the newblockthatcontains the referenced word. The collectionof rules for making this decision is the replacement algorithm.

Cache performance:

When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache. If the processor finds that the memory location is in the cache, a **cache hit** has said to be occurred. If the processor does not find the

$$Hit\,ratio = hit/(hit+miss) = \quad Number\,of\,hits/Total\,accesses\,to\,the\,cache$$

memory location in the cache, a **cache miss** has occurred. When a cache miss occurs, the cache replacement is made by allocating a new entry and copies in data from main memory. The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

Miss penalty or cache penalty is the sum of time to place a bock in the cache and time to deliver the block to CPU.

$$Miss\ Penalty = time for block\ replacement + time\ to\ deliver\ the\ block\ to\ CPU$$

Cache performance can be enhanced by using higher cache block size, higher associativity, reducing miss rate, reducing miss penalty, and reducing the time to hit in the cache. CPU execution Time of a given task is defined as the time spent by the system executing that task, including the time spent executing run-time or system services.

CPU execution time=(CPU clock cycles + memory stall cycles (if any)) x Clock cycle time

The **memory stall cycles** are a measure of count of the memory cycles during which the CPU is waiting for memory accesses. This is dependent on caches misses and cost per miss (cache penalty).

Memory stall cycles = number of cache misses x miss penalty

- Instruction Count x (misses/ instruction) x miss penalty
- Instruction Count (IC) x (memory access/ instruction) x miss penalty
- IC x Reads per instruction x Read miss rate X Read miss penalty + IC x Write per instruction x Write miss rate X Write miss penalty

Misses / instruction = (miss rate x memory access)/ instruction

Issues in Cache memory:

- **Cache placement:** where to place a block in the cache?
- **Cache identification:** how to identify that the requested information is available in the cache or not?
- **Cache replacement:** which block will be replaced in the cache, making way for an incoming block?

Cache Mapping Policies:

The given Fig 5 represents the Catch Mapping. These policies determine the way of loading the main memory to the cache block. Main memory is divided into equal size partitions called as **blocks or frames.** The **c**ache memory is divided into fixedsize partitions called as **lines.** During cache mapping, block of main memory is copied to the cache and further access is made fromthe cache not from the main memory.

Cache mapping is a techniqueby whichthe contentsofmain memory are brought into the cache memory.

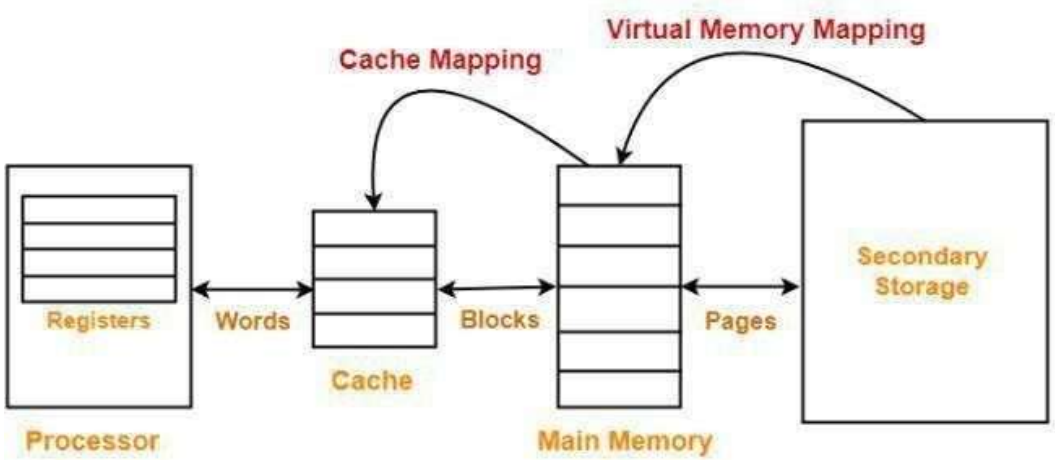


Fig 5: Cache mapping

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

There are three different cache mapping policies or mapping functions:

- Direct mapping
- Fully Associative mapping
- Set Associative mapping

Direct Mapping

- The simplest technique is direct mapping that maps each block of main memory into only one possible cache line. The following Fig 6 shows the Direct Memory Mapping.
- Here, each memory block is assigned to a specific line in the cache.
- If a line is previously taken up by a memory block and when a new block needs to be loaded, then the old block is replaced.

The direct mapping concept is if the i^{th} block of main memory has to be placed at the j^{th} block of cache memory $j = i \% (number\ of\ blocks\ in\ cache\ memory)$

- Direct mapping's performance is directly proportional to the It's ratio.
- Consider a 128 block cache memory. Whenever the main memory blocks 0,

128, 256 are loaded in the cache, they will be allotted cache block 0, since j= (0 or 128 or 256) % 128 is zero).

- Contention or collision is resolved by replacing the older contents with latest contents.
- The placement of the block from main memory to the cache is determined from the 16 bit memory address.
- The lower order four bits are used to select one of the 16 words in the block.
- The 7 bit block field indicates the cache position where the block has to be stored.
- The 5 bit tag field represents which block of main memory resides inside the cache. This method is easy to implement but is not flexible.
- **Drawback:** The problem was that every block of main memory was directly mapped to the cache memory. This resulted in high rate of conflict miss. Cache memory has to be very frequently replaced even when other blocks in the cache memory were present as empty.

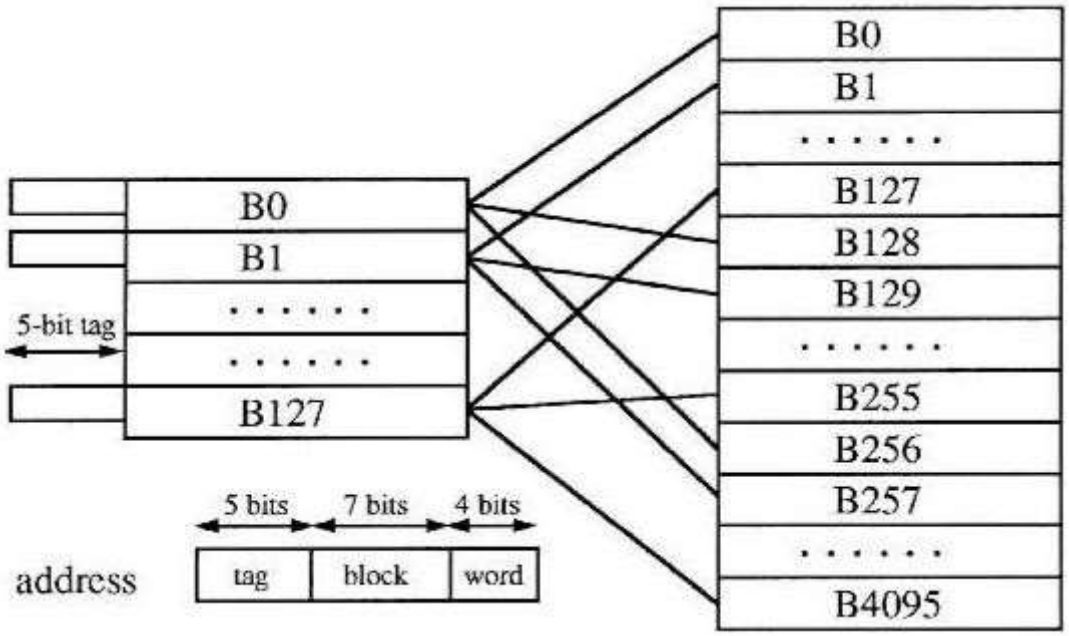


Fig 6: Direct memory mapping

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Associative Mapping:

- The associative memory is used to store content and addresses of the memory word. The bellow Fig 7 shows the Associative Memory Mapping.
- Any block can go into any line of the cache. The 4 word id bits are used to identify which word in the block is needed and the remaining 12 bits represents the tag bit that identifies the main memory block inside the cache.
- This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form.
- The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to check, if the desired block is present. Hence it is knownas Associative Mapping technique.
- Cost of an associated mapped cache is higher than the cost of direct-mapped because ofthe need to search all 128 tag patterns to determine whether a block is in cache.

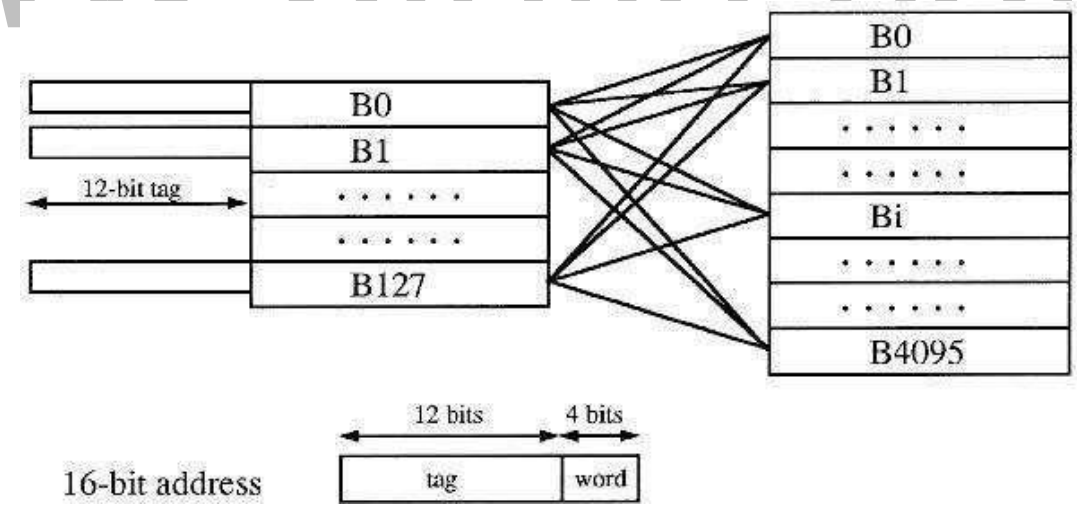


Fig 7: Associative Mapping

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Set associative mapping:

It is the combination of direct and associative mapping technique. The bellow Fig 8 shows Set associative mapping. Cache blocks are grouped into sets and mapping allow block of main memory to reside into any block of a specific set.

This reduces contention problem (issue in direct mapping) with low hardware cost (issue in associative mapping). It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creatinga set. Thena block in memorycan map to any one of the lines of a specific set. The 6 bit set field of the address determines which set of the cache might contain the desired block. The tag bits of address must be associatively compared to the tags of the two blocks of the set to check if desired block is present.

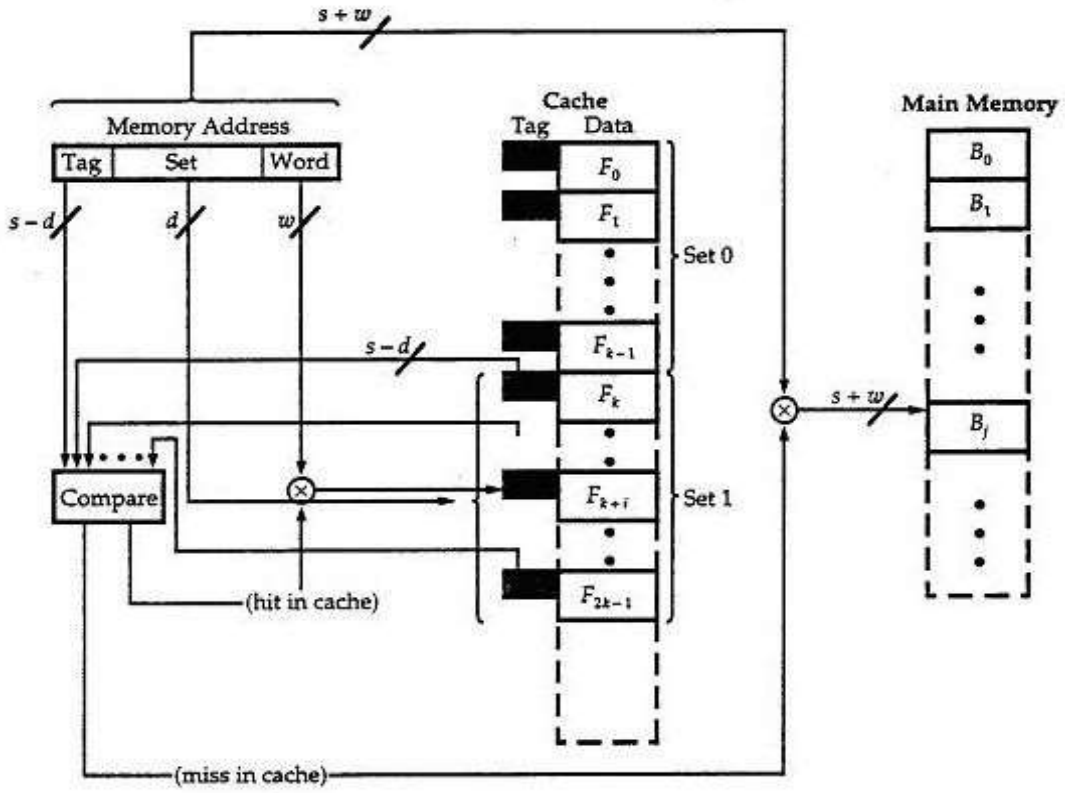


Fig 8: Set associative mapping

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Handling Cache misses:

When a program accesses a memory location that is not in the cache, it is called a cache miss. The performance impact of a cache miss depends on the latency of fetching the data from the next cache level or main memory. The cache miss handling is done with the processor control unit and with a separate controller that initiates the memory access andrefillsthecache. The following are the steps taken when a cache miss occurs:

- Send the original PC value (PC - 4) to the memory.
- Instruct main memory to perform a read and wait for the memory to complete its access.
- Write the cache entry, putting the data from memory in the data portion of the entry, writing the upper bits of the address (from the ALU) into the tag field, and turning the valid bit on.
- Restart the instruction execution at the first step, which will refetch the instruction, this time finding it in the cache.

Writing to a cache:

- Suppose on a store instruction, the data is written into only the data cache (without changing main memory); then, after the write into the cache, memory would have a different value from that in the cache. This leads to inconsistency. Fig 9 shows the Cache Memory.
- The simplest way to keep the main memory and the cache consistent is to always write the data into both the memory and the cache. This scheme is called write-through.

Writethroughisaschemeinwhichwritesalwaysupdateboththecacheandthememory, ensuring that data is always consistent between the two.

- With a write-through scheme, every write causes the data to be written to main memory.

  These writes will take a long time.
- A potential solution to this problem is deploying write buffer.

- A write buffer stores the data while it is waiting to be written to memory.
- After writing the data into the cache and into the write buffer, the processor can continue execution.
- When a write to main memory completes, the entry in the write buffer is freed.
- If the write buffer is full when the processor reaches a write, the processor must stall until there is an empty position in the write buffer.
- If the rate at which the memory can complete writes is less than the rate at which the processor is generating writes, no amount of buffering can help because writes are being generated faster than the memory system can accept them.

Write buffer is a queue that holds data while the data are waiting to be written to memory.

i) The rate at which writes are generated may also be less than the rate at which the memory can accept them, and yet stalls may still occur. To reduce the occurrence of such stalls, processors usually increase the depth of the write buffer beyond a single entry.

ii) Another alternative to a write-through scheme is a scheme called write-back. Whenawrite occurs, the new value is written only to the block in the cache.

iii) The modified block is written to the lower level of the hierarchy when it is replaced.

iv) Write-back schemes can improve performance, especially when processors can generate writes as fast or faster than the writes can be handled by main memory; a write-back scheme is, however, more complex to implement than write-through.

Write-back is a scheme that handles writes by updating values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.

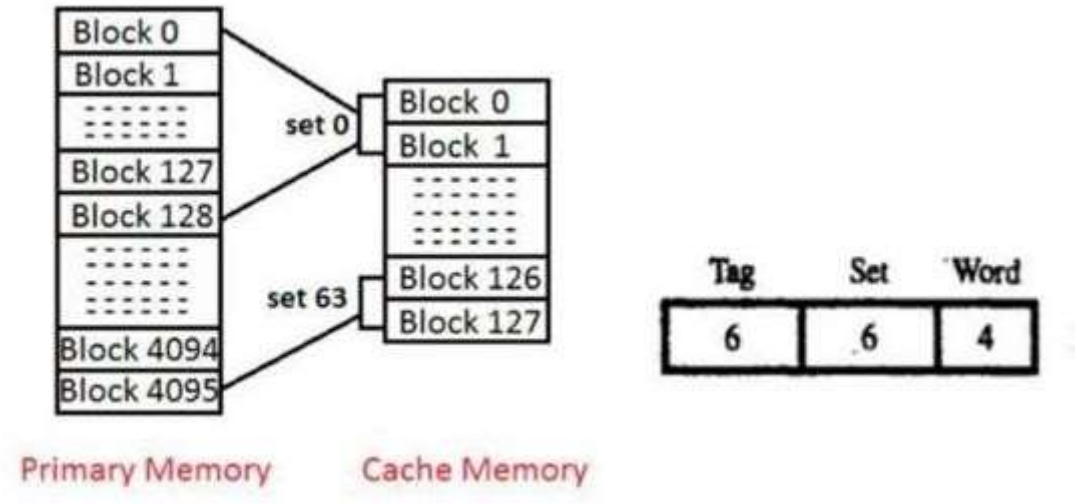


Fig 9 Cache Memory

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Cache Replacement Algorithms

When a main memory block needs to be brought into the cache while all the blocks are occupied, then one of them has to be replaced. This selection of the block to be replaced is using cache replacement algorithms. Replacement algorithms are only needed for associative and set associative techniques. The following are the common replacement techniques:

- **Least Recently Used (LRU):** This replaces the cache line that has been in the cache the longest with no references to it.
- **First-in First-out (FIFO):** This replaces the cache line that has been in the cache the longest.
- **Least Frequently Used (LFU):** This replaces the cache line that has experienced the fewest references.

Random: This picks a line at random from the candidate lines.

Example 4.1: Program P runs on computer A in 10 seconds. Designer says clock rate can be increased significantly, but total cycle count will also increase by 20%. What clock rate do we need on computer B for P to run in 6 seconds? (Clock rate on A is 100 MHz). The new machine is B. We want CPU Time_B = 6 seconds.

We know that Cycles count_B = 1.2 Cycles count_A. Calculate Cycles count_A. CPU Time_A = 10 sec. = ; Cycles count_A = 1000 x 106 cycles Calculate Clock rate_B:

CPU Time_B = 6 sec. = ; Clock rate_B = = 200 MHz

Machine B must run at twice the clock rate of A to achieve the target execution time.

Example 4.2: We have two machines with different implementations of the same ISA. Machine A has a clock cycle time of 10 ns and a CPI of 2.0 for program P; machine B has a clock cycle time of 20 ns and a CPI of 1.2 for the same program. Which machine is faster? Let IC be the number of instructions to be executed. Then Cycles count_A = 2.0 IC

Cycles count_B = 1.2 IC

calculate CPU Time for each machine: CPU Time_A = 2.0 IC x 10 ns = 20.0 IC ns

CPU Time_B = 1.2 IC x 20 ns = 24.0 IC ns

» Machine A is 20%faster.

Example 4.3: Consider an implementation of MIPS ISA with 500 MHz clock and

– each ALU instruction takes 3 clock cycles,

– each branch/jump instruction takes 2 clock cycles,

– each sw instruction takes 4 clock cycles,

– eachlw instruction takes 5 clock cycles.

Also, consider a program that during its execution executes:

– x=200 million ALU instructions

– y=55 million branch/jump instructions

– z=25 million sw instructions

– w=20 million lw instructions

Find CPU time. Assume sequentially executing CPU. Clock cycles for a program = (3x + 2y + 4z + 5w)

= 910 x 10^6 clock cycles CPU_time = Clock cycles for a program /

Clock rate = 910 x 10^6 / 500 x 10^6 = 1.82 sec

Example 4.4: Consider another implementation of MIPS ISA with 1 GHz clock and each ALU instruction takes 4 clock cycles,

– each branch/jump instruction takes 3 clock cycles,

– each sw instruction takes 5 clock cycles,

– eachlw instruction takes 6 clock cycles.

Also, consider the same program as in Example 1.

Find CPI and CPU time. Assume sequentially executing CPU. CPI = (4x + 3y + 5z + 6w) / (x + y + z + w)

= 4.03 clock cycles/ instruction

CPU time = Instruction count x CPI / Clock rate

= (x+y+z+w) x 4.03 / 1000 x 10^6

= 300 x10^6 x 4.03 /1000 x 10^6

= 1.21 sec

4.2.2 VIRTUAL MEMORY

Virtual memory is a memory management capability of an operating system that uses hardware and software to allow a computer to compensate for physical memory shortages by temporarily transferring data from RAM to disk storage.

The concept of virtual memory in computer organization is allocating memory from the hard disk and making that part of the hard disk as a temporary RAM. In other

words, it is a technique that uses main memory as a cache for secondary storage. The motivations for virtual memoryare:

- To allow efficient and safe sharing of memory among multiple programs
- To remove the programming burdens of a small, limited amount of main memory.

Virtual memory provides an illusion to the users that the PC has enough primary memory left to run the programs. Sometimes the size of programs to be executed may sometimes very bigger than the size of primary memory left, the user never feels that the system needs a bigger primary storage to run that program. When the RAM is full, the operating system occupies a portion of the hard disk and uses it as a RAM. In that part of the secondary storage, the part of the program which not currently being executed is stored and all the parts of the program that are executed are first brought into the main memory. This is the theory behindvirtual memory.

Terminologies:

- **Physical address** is an address in main memory.
- **Protection** is a set of mechanisms for ensuring that multiple processes sharing the processor, memory, or I/O devices cannot interfere, with one another by reading or writing each other's data.
- Virtual memory breaks programs into fixed-size blocks called **pages.**
- **Page fault** is an event that occurs when an accessed page is not present in main memory.
- **Virtual address** is an address that corresponds to a location in virtual space and is translated by address mapping to a physical address when memory is accessed.
- **Address translation or address mapping** is the process by which a virtual address is mapped to an address used to access memory.

Working mechanism

- In virtual memory, blocks of memory are mapped from one set of addresses

(virtual addresses) to another set (physical addresses).

- The processor generates virtual addresses while the memory is accessed using physical addresses.
- Both the virtual memory and the physical memory are broken into pages, so that a virtual page is really mapped to a physical page.
- It is also possible for a virtual page to be absent from main memory and not be mapped to a physical address, residing instead on disk.
- Physical pages can be shared by having two virtual addresses point to the same physical address. This capability is used to allow two different programs to share data or code. Virtual memory also simplifies loading the program for execution by providing relocation. The following Fig 10 represents Mapping of virtual and physical memory.
- **Relocation** maps the virtual addresses used by a program to different physical addresses before the addresses are used to access memory. This relocation allows us to load the program anywhere in main memory.

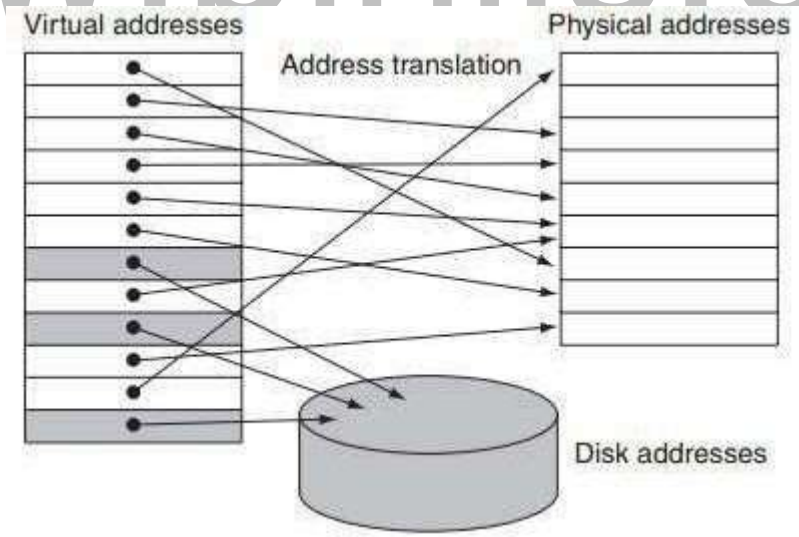


Fig 10: Mapping of virtual and physical memory

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Addressing in virtual memory

- A virtual address is considered as a pair (p,d) where lower order bits give an offset d within the page and high-order bits specify the page p.
- The job of the Memory Management Unit (MMU) is to translate the page number p to a frame number f. The physical address is then (f,d), and this is what goes on the memory bus. For every process, there is a page and page-number p is used as an index into this array for the translation. The following Fig 11 shows Conversion of logical address to physical address
- The following are the entries in page tables:
  1. Validity bit: Set to 0 if the corresponding page is not in memory
  2. Frame number: Number of bits required depends on size of physical memory
  3. Protection bits: Read, write, execute accesses
  4. Referenced bit is set to 1 by hardware when the page is accessed: used by page replacement policy
  5. Modified bit (dirty bit) set to 1 by hardware on write-access: used to avoid writing when swapped out.

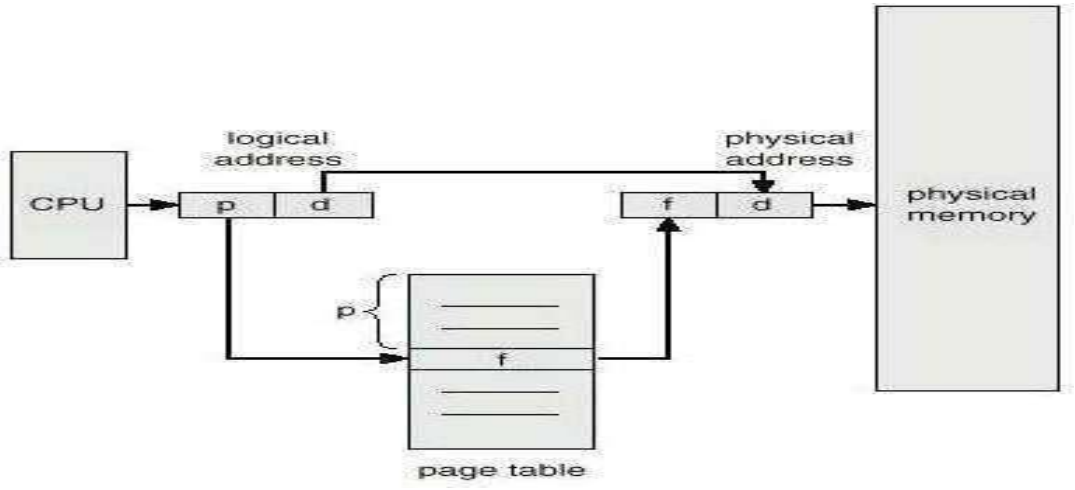


Fig 11: Conversion of logical address to physical address

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

Role of control bit in page table

The control bit (v) indicates whether the page is loaded in the main memory. It also indicates whether the page has been modified during its residency in the main memory.

This information is crucial to determine whether to write back the page to the disk before it is removed from the main memory during next page replacement. The bellow Fig 12 shows Page table Page faults and page replacement algorithms.

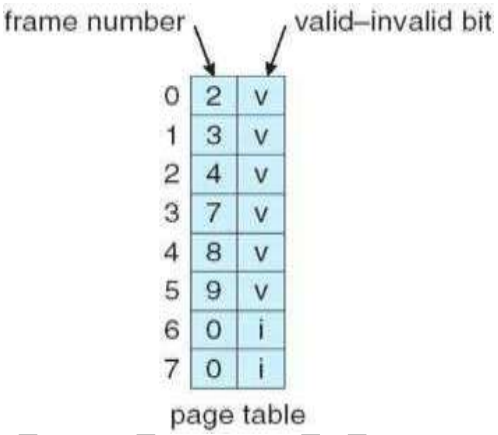


Fig 12: Page table Page faults and page replacement algorithms

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization

Page replacement is a process of swapping out an existing page from the frame of a main memory and replacing it with the required page.

An Integrated approach

A page fault occurs when a page referenced by the CPU is not found in the main memory. The required page has to be brought from the secondary memory into the main memory. A page that is currently residing in the main memory, has to be replaced if all the frames of main memory are already occupied.

Page replacement is done when all the frames of main memory are already occupied and a page has to be replaced to create a space for the newly referenced page.

A good replacement algorithm will have least number of page faults. The bellow Fig 13 shows the Occurrence of page fault.

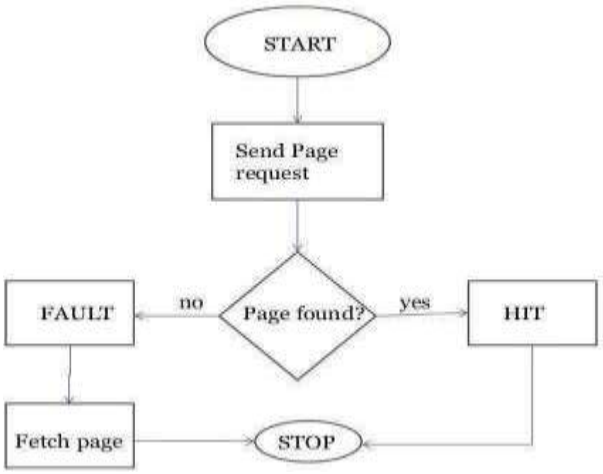


Fig 13: Occurrence of page fault

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

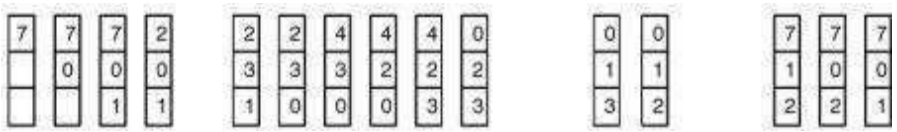
The following are the page replacement algorithms:

6. FIFOPage Replacement Algorithm
7. LIFOPage Replacement Algorithm
8. LRU Page Replacement Algorithm
9. Optimal Page Replacement Algorithm
10. Random Page Replacement Algorithm

1. **First In First Out (FIFO) page replacement algorithm**

   It replaces the oldestpage that has beenpresent in the mainmemoryforthe longest time. It is implemented by keeping track of all the pages in a queue.

   Example 4.5. Find the page faults when the following pages are requested to be



loaded ina page frame of size 3: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Page faults= 15

2. **Last In First Out (LIFO) page replacement algorithm**

   It replaces the newest page that arrived at last in the main memory. It is implemented by keeping track of all the pages in a stack.

3. **Least Recently Used (LRU) page replacement algorithm** The new page will be replaced with least recently used page.

   Example 4.6: Consider the following reference string. Calculate the number of page faults whenthe page framesize is 3 using LRUpolicy.7, 0, 1, 2, 0, 3, 4, 2, 3, 0, 3, 2,1,2,0,1,7,0,1

7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
F	F	F	F		F		F	F	F	F			F		F		F		

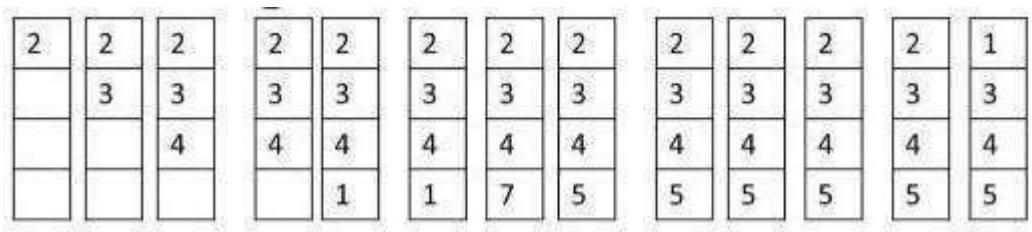
Page faults= 12 (F bit indicates the occurrence of page faults)

4. **Optimal page replacement algorithm**

   In this method, pages are replaced which would not be used for the longest durationoftime in the future.

   Example 4.7: Find the number of misses and hits while using optimal page replacement algorithm on the following reference string with page frame size as 4: 2, 3, 4, 2, 1, 3, 7, 5, 4,

   3, 2, 3, 1.



Page fault=13Number of page hit= 6Number of page misses=7

5. **Random page replacement algorithms**

Random replacement algorithm replaces a random page in memory. This eliminates the overhead cost of tracking page references.

Translation Look aside Buffer (TLB)

The page tables are stored in main memory and every memory access by a program to the page table takes longer time. This is because it does one memory access to obtain the physical address and a second access to get the data. The virtual to physical memory address translation occurs twice. But a TLB will exploit the locality of reference and can reduce the memory access time.

TLB hit is a condition where the desired entry is found in translation look aside buffer. If this happens then the CPU simply access the actual location in the main memory.

If the entry is not found in TLB (TLB miss) then CPU has to access page table in the main memory and then access the actual frame in the main memory. Therefore, in the case of TLB hit, the effective access time will be lesser as compare to the case of TLB miss.

If the probability of TLB hit is P% (TLB hit rate) then the probability of TLB miss (TLB miss rate) will be (1-P) %. The effective access time can be defined as

$$\textit{Effective access time} = P\,(t + m) + (1 - p)\,(t + k.m + m)$$

Where, p is the TLB hit rate, t is the time taken to access TLB, m is the time taken to access main memory. K indicates the single level paging has been implemented. The following Fig 14 shows Cache Access Levels.

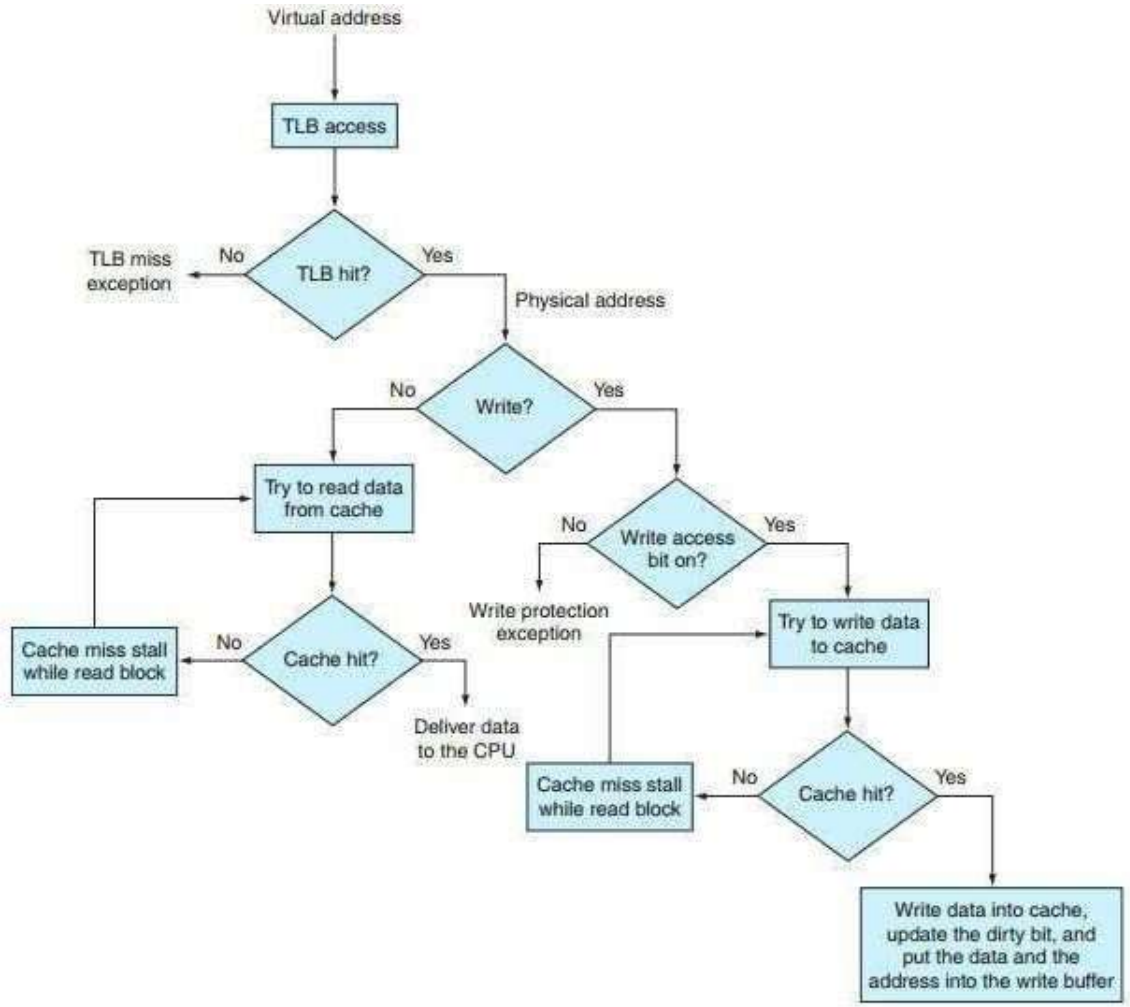


Fig 14: Cache access levels

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

4.3.5 Protection in Virtual memory

- Virtual memory allows sharing of main memory by multiple processes. So protection mechanisms, while providing memory protection.
- The protection mechanism must ensure one process cannot write into the address space of another user process or into the operating system.
- Memory protection can be done at two levels: hardware and software levels.

Hardware Level:

Memory protection at hardware level is done in three methods:

- The machine should support two modes: supervisor mode and user mode. This indicates whether the current running process is a user or supervisory process. The processes running in supervisor or kernel mode is an operating system process.
- Include user / supervisor bit in TLB to indicate whether the process is in user or supervisor mode. This is an access control mechanism imposed on the user process only to read from the TLB and not write to it.

- The processors can switch between user and supervisor mode. The switching from user to system mode is done through system calls that transfers control to a dedicated location in supervisor code space.

System call is a special instruction that transfers control from user mode to a dedicated location in supervisor code space, invoking the exception mechanism in the process.

4.1 INTRODUCTION

Memory unit enables us to store data inside the computer. The computer memory always had here's to principle of locality.

Principle of locality or locality of reference is the tendency of a processor to access the same set of memory locations repetitively over a short period of time.

Two different types of locality are:

- Temporal locality: The principle stating that if a data location is referenced then it will tend to be referenced again soon.
- Spatial locality: The locality principle stating that if a data location is referenced, data locations with nearby addresses will tend to be referenced soon.

Memory hierarchy is a structure that uses multiple levels of memories; as the distance from the CPU increases, the size of the memories and the access time both increase.

The locality of reference is useful in implementing the memory hierarchy.A memory hierarchy consists of multiple levels of memory with different speeds and sizes. The faster memories are more expensive per bit than the slower memories and thus smaller. The bellow Fig 1 shows the Memory hierarchy.

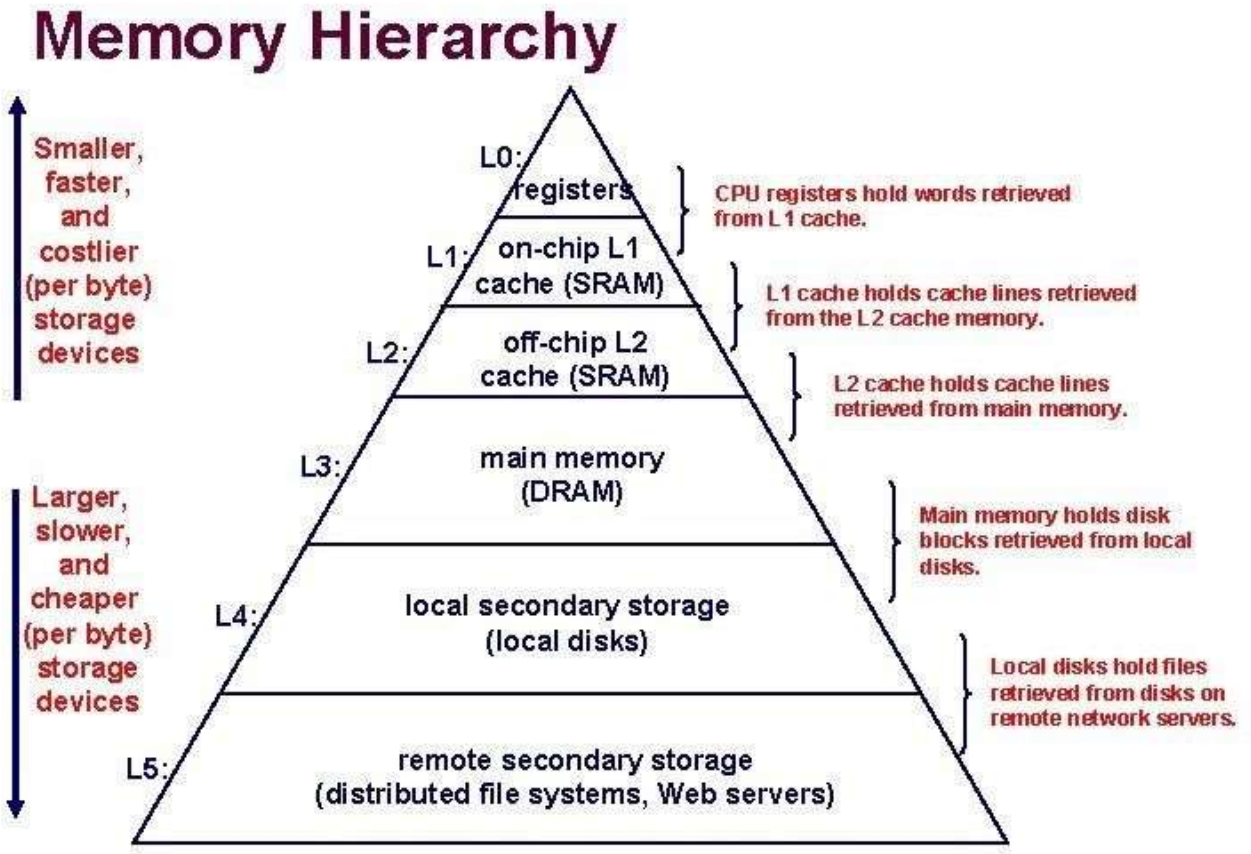


Fig 1: Memory Hierarchy

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

- Main memory is implemented from Dynamic Random Access Memory (DRAM).
- The levels closer to the processor (caches) use Static Random Access Memory (SRAM).
- DRAM is less costly per bit than SRAM, although it is substantially slower.
- For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at levelk+1.
- The computer programs tend to access the data at level k more often that at level k+1.
- The storage at level at k+1 can be slower

Cache memory (CPU memory) is high-speed SRAM that a computer Microprocessor can access more quickly than it can access regular RAM. This memory is typically integrated directly into the CPU chip or placed on a separate chip that has a separate bus interconnect with the CPU.

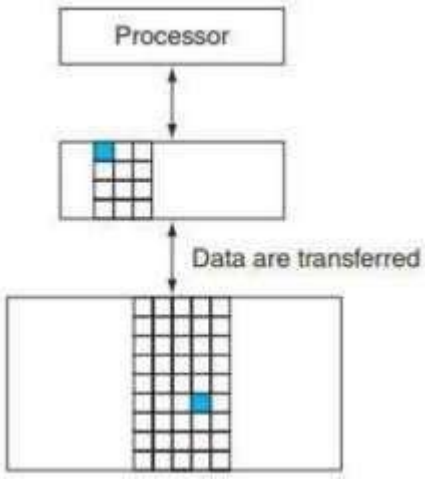


Fig 2: Data access by processor

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach, Second edition, Wiley India Pvt Ltd, 2015

The data transfer between various levels of memory is done through blocks. The minimum unit of information is called a **block.** If the data requested by the processor appears in some block in the upper level, this is called a **hit.** If the data is not found in the upper level, the request is called a **miss.** The lower level in the hierarchy is thenaccessed to retrieve the block containing the requested data.

Thefraction ofmemory accesses foundin acacheis termed as hit rateorhitratio.

Miss penalty is the time required to fetch a block into a level of the memory hierarchy from the lower level, including the time to access the block, transmit it from one level to the other, and insert it in the level that experienced the miss.

Miss rate is the fraction of memory accesses not found in a level of the memory hierarchy. Hit time is the time required to access a level of the memory hierarchy, includingthe time needed to determine whether the access is a hit or a miss.

Because the upper level is smaller and built using faster memory parts, the hit time will be much smaller than the time to access the next level in the hierarchy, which is the major component of the miss penalty.

4.1.1 MEMORY HIERARCHY

- At the top of the hierarchy are registers that are matched in speed to the CPU, but tend to be large and consume a significant amount of power. There are normally only a small number of registers in a processor, on the order of a few hundred or less.
- At the bottom of the hierarchy are secondary and off. Line storage memories such as hard magnetic disks and magnetic tapes, in which the cost per stored bit is small in terms of money and electrical power, but the access time is very long when compared with registers.
- Between the registers and secondary storage are a number of other forms of memory that bridge the gap between the two.
- As we move up through the hierarchy, greater performance is realized, at greater costs.

Table: Properties of memory hierarchy

Memory type	Access time	Cost/MB	Typical amount used	Typical cost
Registers	0.5ns	High	2kB	-
Cache	5-20ns	$80	2MB	$160
Main Memory	40-80ns	$0.40	512 MB	$205
Disk Memory	5ms	$0.005	40GB	$200

- Expensive memory tends to be closer to CPU, which shortens communication times.
- Typical cost arrived at by multiplying cost/MB x Typical amount used, is roughly the same for each member of the hierarchy.
- Access times vary by approximately factors of 10 except for disks, which have access times 10,000 times slower than main memory.

A memory unit is a collection of semi-conductor storage cells with circuits to access the data stored in them. The data storage in memory is done in words. The number of bits in a word depends on the architecture of the computer. Generally a word is always multiple of 8. Memory is accessed through unique system assigned address. The accessing of data from memory is based on principle of locality.

Principle of Locality

The locality of reference or the principle of locality is the term applied to situations where the same value or related storage locations are frequently accessed. There arethreebasic types of locality of reference:

- **Temporal locality:** Here a resource that is referenced at one point in time is referenced again soonafterwards.
- **Spatial locality:** Here the likelihood of referencing a storage location is greater if a storage location near it has been recently referenced.
- **Sequential locality:** Here storage is accessed sequentially, in descending or

ascending order. The locality or reference leads to memory hierarchy.

Need for memory hierarchy

Memory hierarchy is an approach for organizing memory and storage systems. It consist of multiple levels of memory with different speeds and sizes. The following are the reasons for such organization:

- Fast storage technologies cost more per byte and have less capacity
- Gap between CPU and main memory speed is widening.
- Well-written programs tend to exhibit good locality. The memory hierarchy is shown in Fig 1. The entire memory elements of the computer fall under the following three categories:

- **Processor Memory:**
  This is present inside the CPU for high-speed data access. This consists of small set of registers that act as temporary storage. This is the costliest memory component.
- **Primary memory:**
  This memory is directly accessed by the CPU. All the data must be brought inside main memory before accessing them. Semiconductor chips acts as main memory.
- **Secondary memory:**
  This is cheapest, large and relatively slow memory component. The data from the secondary memory is accessed by the CPU only after it is loaded to main memory.

There is a trade-off among the three key characteristics of memory namely-

- Cost
- Capacity
- Access time

Terminologies in memory access

- **Block or line:** The minimum unit of information that could be either present or totally absent.
- **Hit**: If the requested data is found in the upper levels of memory hierarchy it is called hit.

- **Miss:** If the requested data is not found in the upper levels of memory hierarchy it is called miss.
- **Hit rate or Hit ratio:** It is the fraction of memory access found in the upper level .It is a performance metric.

  Hit Ratio = Hit/ (Hit + Miss)
- **Miss rate:** It is the fraction of memory access not found in the upper level (1-hit rate).
- **Hit Time:** The time required for accessing a level of memory hierarchy, including the time needed for finding whether the memory access is a hit or miss.
- **Miss penalty:** The time required for fetching a block into a level of the memory hierarchy from the lower level, including the time to access, transmit, insert it to new level and pass the block to the requestor.
- **Bandwidth:** The data transfer rate by the memory.
- **Latency or access time:** Memory latency is the length of time between the memory's receipt of a read request and its release of data corresponding with the request.
- **Cycle time:** It is the minimum time between requests to memory. The bellow Fig 2 shows the memory level versus access time.

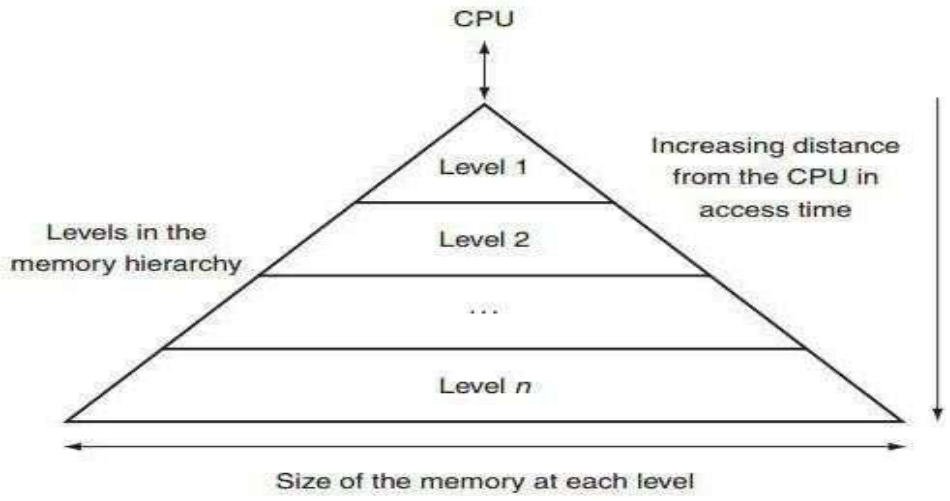


Fig 2: Memory level vs Access Time

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

- The memory access time increases as the level increases. Since the CPU registers are located in very close proximity to the CPU they can be accessed very quickly and they are the more costly. As the level increases, the memory access time also increases thereby decreasing the costs.

Levels in Memory Hierarchy

The following are the levels in memory hierarchy:

- **CPU Registers:**

  They are at the top most level of this hierarchy, they hold the most frequently used data. They are very limited in number and are the fastest. They are often used by the CPU and the ALU for performing arithmetic and logical operations, for temporary storage of data.

- **Static Random Access Memory (SRAM):**

  Static Random Access Memory (Static RAM or SRAM) is a type of RAM that holds data in a static form, that is, as long as the memory has power. SRAM stores a bit of data on four transistors using two cross-coupled inverters. The two stable states characterize 0 and 1. During read and write operations another two access transistors are used to manage the availability to a memory cell.

- **Main memory or Dynamic Random Access Memory (DRAM):**

  Dynamic random access memory (DRAM) is a type of memory that is typically used for data or program code that a computer processor needs to function. In other words it is said to be the main memory of the computer. Random access allows processor to access any part of the memory directly rather than having to proceed sequentially from a starting place. The main advantages of DRAM are its simple design, speed and low cost in comparison to alternative types of memory. The main disadvantages of DRAM are volatility and high power consumption relative to other options.

- **Local Disks (Local Secondary Storage):**

  A local drive is a computer disk drive that is installed directly within the host or the

local computer. It is a computer's native hard disk drive (HDD), which is directly accessed by the computer for storing and retrieving data. It is a cheaper memory with more memoryaccess time.

- **Remote Secondary Storage:**

This includes Distributed file system (DFS) and online storage like cloud. The storage area is vast with low cost but larger access time.

Distinction between Static RAM and Dynamic RAM

SRAM	DRAM
Stores data till the power is supplied. Uses nearly 6 transistors for each memory cell.	Stored data only for few milliseconds irrespective of the power supply. Uses single transistor and capacitor for each memory cell.
Do not refresh the memory cell.	Refreshing circuitry is needed.
Faster data access.	Slower access.
Consumes more power.	Low power consumption.
Cost pet bit is high.	Comparatively lower costs.
They are made of more number of components percells.	They are made of less number of components percells.

CLASSIFICATION O MEMORY

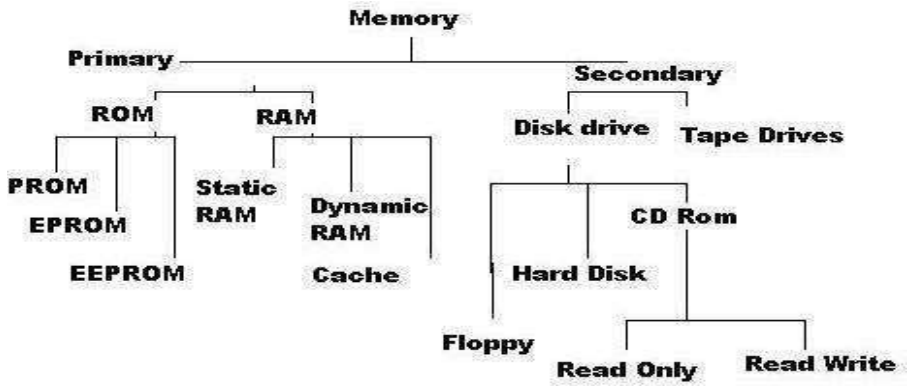


Fig 3: Classification of Memory

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach

The above Fig 3 shows the classification of memory. The instructions and data are stored in memory unit of the computer system are divided into following maingroups:

- Main or Primary memory
- Secondary memory.

Primary Memory:

Primary memory is the main area in a computer in which data is stored for quick access by the computer's processor. It is divided into two parts:

i) Random Access Memory (RAM):

RAM is a type of computer primary memory. It accessed any piece of data at any time. RAM stores data for as long as the computer is switched on or is in use. This type of memory is volatile. The two types of RAM are:

- **Static RAM:** This type of RAM is static in nature, as it does not have to be refreshed at regular intervals. Static RAM is made of large number of flip-flops on IC. It is being costlier and having packing density.
- **Dynamic RAM:** This type of RAM holds each bit of data in an individual capacitor in an integrated circuit. It is dynamic in the sense that the capacitor charge is repeatedly refreshed to ensure the data remains intact.

ii) Read Only Memory (ROM):

The ROM is nonvolatile memory. It retains stored data and information if the power is turned off.)n ROM, data are stored permanently and can't alter by the programmer. There are four types of ROM:

- **MROM (mask ROM):** MROM (mask ROM) is manufacturer-Programmed ROM in which data is burnt in by the manufacturer of the electronic equipment in which it is used and it is not possible for a user to modify programs or data stored inside the ROM chip.
- **PROM (programmable ROM):** PROM is one in which the user can load and store

"read- only" programs and data. In PROM the programs or data are stored only fast time and the stored data cannot modify the user.

- **EPROM (erasable programmable ROM):** EPROM is one in which is possible to erase information stored in an EPROM chip and the chip can be reprogrammed to store new information. When an EPROM is in use, information stored in it can only be read and the information remains in the chip until it is erased.
- **EEPROM (electronically erasable and programmable ROM):** EEPROM is one type of EPROM in which the stored information is erased by using high voltage electric pulse. It is easier to alter information stored in an EEPROM chip.

Secondary Memory:

Secondary memory is where programs and data are kept on a long time basis. It is cheaper from of memory and slower than main or primary memory. It is non-volatile and cannot access data directly by the computer processor. It is the external memory of the computer system.

Example: hard disk drive, floppy disk, optical disk/ CD-ROM.

4.3 PARALLEL BUS ARCHITECTURES

Single bus architectures connect multiple processors with their own cache memory using shared bus. This is a simple architecture but it suffers from latency and bandwidth issues. This naturally led to deploying parallel or multiple bus architectures. Multiple bus multiprocessor systems use several parallel buses to interconnect multiple processors with multiple memory modules. The following are the connection schemes in multi bus architectures:

1. **Multiple-bus with full bus–memory connection (MBFBMC)**

   This has all memory modules connected to all buses. The multiple-bus with single bus memory connection has each memory module connected to a specific bus. For N processors with M memory modules and B buses, the number of connections requires are: B(N+M) and the load on each bus will ne N+M.

2. **Multiple bus with partial bus–memory connection (MBPBMC)**

   The multiple-bus with partial bus–memory connection, has each memory module connected to a subset of buses.

3. **Multiple bus with class-based memory connection (MBCBMC)**

   The multiple-bus with class-based memory connection (MBCBMC), has memory modules grouped into classes whereby each class is connected to a specific subset of buses. A class is just an arbitrary collection of memory modules.

4. **Multiple bus with single bus memory connection (MBSBMC)**

   Here, only single bus will be connected to single memory, but the processor can access all the buses. The numbers of connections:

   $$BN + \sum_{j=1}^{k} M_j(j + B - k)$$

   And load on each bus is given by

   $$N + \sum_{j=\max(i+k-B,1)}^{k} M_j,\ 1 \le i \le B$$

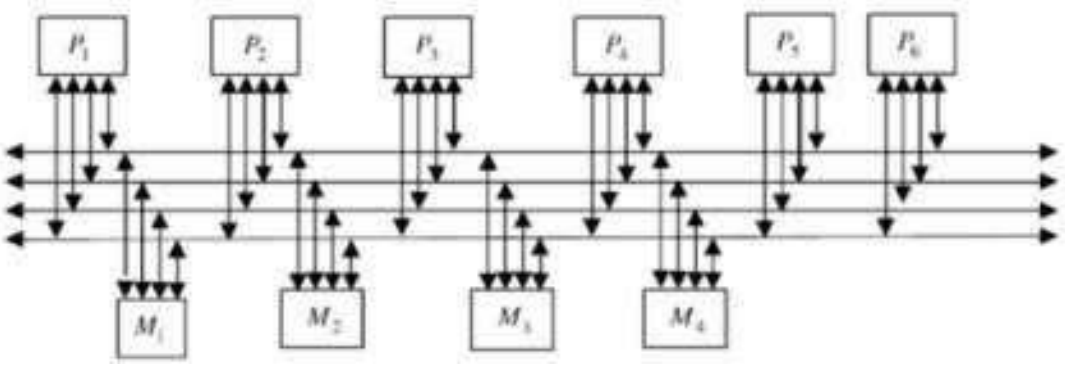


Fig 1: a) Multiple-bus with full bus–memory connection (MBFBMC)

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

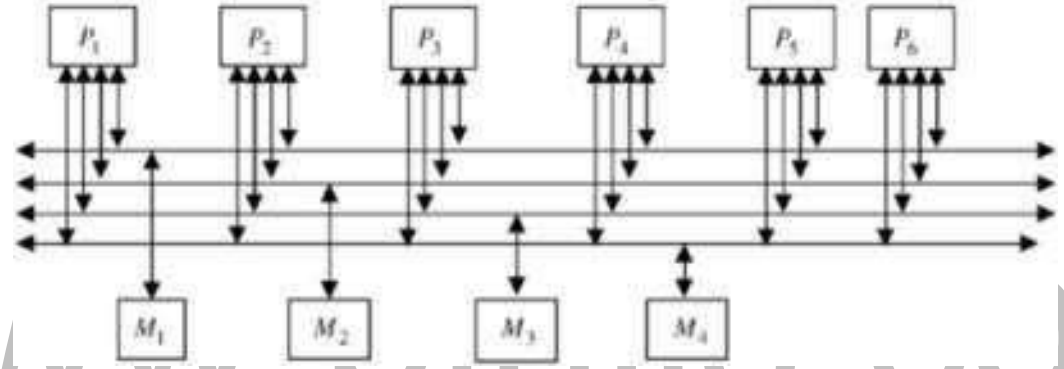


Fig 1b) Multiple bus with single bus memory connection (MBSBMC)

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

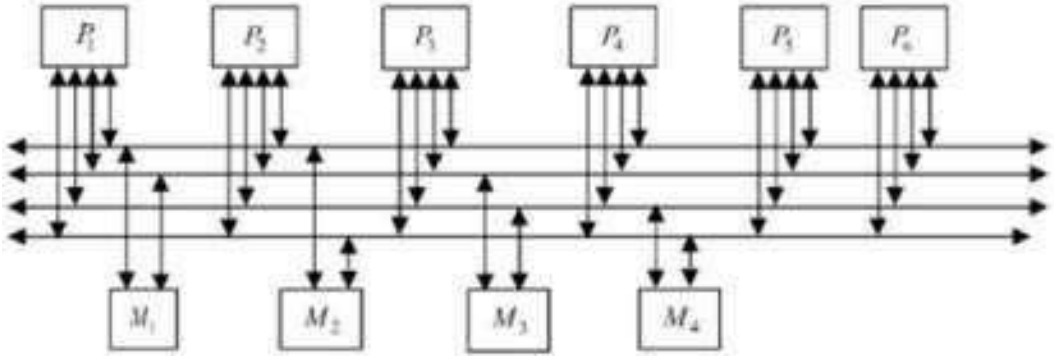


Fig 1 c) Multiple bus with partial bus–memory connection (MBPBMC)

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

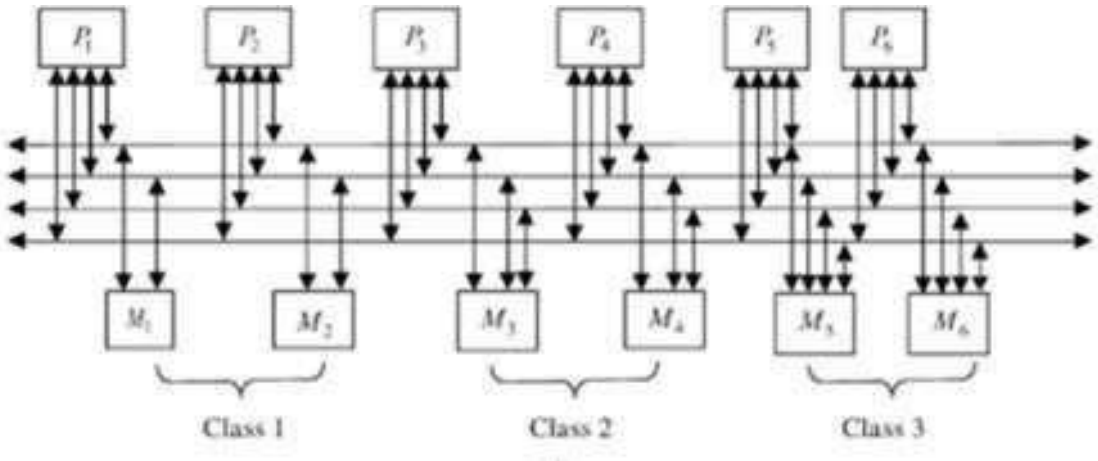


Fig 1d) Multiple bus with class-based memory connection (MBCBMC)

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

Bus Synchronization:

- In a single bus multiprocessor system, bus arbitration is required in order to resolve the bus contention that takes place when more thanone processor competes to accessthebus.
- A bus can be classified as synchronous or asynchronous. The time for any transaction over a synchronous bus is known in advance. Asynchronous bus depends on the availability of data and the readiness of devices to initiate bus transactions.
- The processors that want to use the bus submit their requests to bus arbitration logic. The latter decides, using a certain priority scheme, which processor will be granted access to the bus during a certain time interval (bus master).
- The process of passing bus mastership from one processor to another is called **handshaking,** which requires the use of two control signals: bus request and bus grant.
- Bus request indicates that a given processor is requesting mastership of the bus.
- Bus grant: indicates that bus mastership is granted.
- Bus busy: is usually used to indicate whether or not the bus is currently being used.
- In deciding which processor gains control of the bus, the bus arbitration logic uses a predefined priorityscheme.

- Among the priority schemes used are random priority, simple rotating priority, equal priority, and least recently used (LRU) priority.
- After each arbitration cycle, in simple rotating priority, all priority levels are reduced one place, with the lowest priority processor taking the highest priority. In equal priority, when two or more requests are made, there is equalchance of anyone request beingprocessed.

In the LRU algorithm, the highest priority is given to the processor that has not used the bus for the longest time.

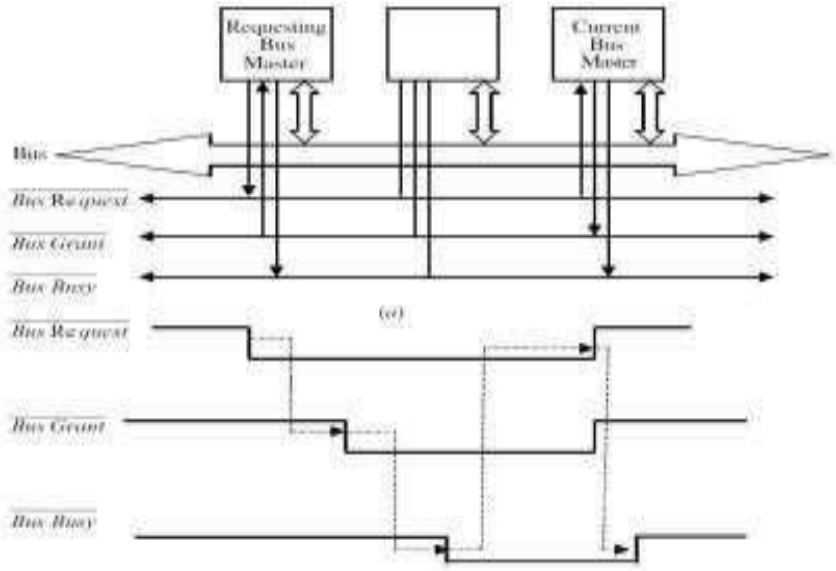


Fig 2: Bus synchronization

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

4.3.1 INTERNAL COMMUNICATION METHODOLOGIES

CPU of the computer system communicates with the memory and the I/O devices in order to transfer data between them. The method of communication of the CPU with memory and I/O devices is different. The CPU may communicate with the memory either directly or through the Cache memory. However, the communication between the CPU and I/O devices is usually implemented with the

help of interface. There are three types of internal communications:

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

Programmed I/O

- Programmed I/O is implicated to data transfers that are initiated by a CPU, under driver software control to access Registers or Memory on a device.
  With programmed I/O, data are exchanged between the processor and the I/O module.
- The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, this is wasteful of processor time. With interrupt-driven I/O, the processor issues I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.
- With both programmed and interrupt I/O, the processor is responsible for extracting data from main memory for output and storing data in main memory for input.
- The alternative is known as direct memory access. In this mode, the I/O module and main memory exchange data directly, without processor involvement.
- With programmed I/O, the I/O module will perform the requested action and then set the appropriate bits in the I/O status register.
- The I/O module takes no further action to alert the processor.
- When the processor is executing a program and encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate

I/O module. In particular, it does not interrupt the processor.

- It is the responsibility of the processor periodically to check the status of the I/O module. Then if the device is ready for the transfer (read/write).
- The processor transfers the data to or from the I/O device as required. As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either receptionortransmissionofdata.
- The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as **Polling.**

The level of the performance of the entire system is severely degraded.

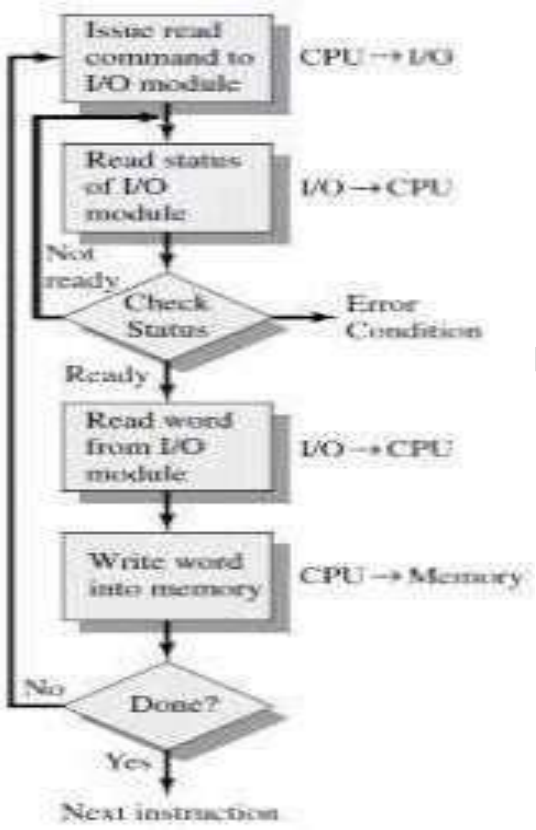


Fig 3: Workflow in programmed I/O

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

Interrupt Driven I/O

- The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work.
- For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.
- For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.
- Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory.

Below are the basic operations of Interrupt:

1. CPU issues read command
2. I/O module gets data from peripheral whilst CPU does other work
3. I/O module interrupts CPU
4. CPU requests data
5. I/O module transfers data

Direct Memory Access (DMA)

- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement.
- DMA module controls exchange of data between main memory and the I/O device.
- Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus.
- CPU is only involved at the beginning and end of the transfer and interrupted

only after entire block has been transferred.

Fig 4: CPU bus signals for DMA transfer

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.

- The CPU programs the DMA controller by setting its registers so it knows what to transfer where.
- It also issues a command to the disk controller telling it to read data from the disk into its internal buffer and verify the checksum.
- When valid data are in the disk controller┼s buffer, DMA can begin. The DMA controller initiates the transfer by issuing a read request over the bus to the disk controller.
- This read request looks like any other read request, and the disk controller does not know whether it came from the CPU or from a DMA controller.
- The memory address to write to is on the bus address lines, so when the disk controller fetches the next word from its internal buffer, it knows where to write it.
- The write to memory is another standard bus cycle.
- When the write is complete, the disk controller sends an acknowledgement

signal to the DMA controller, also over the bus.

- The DMA controller then increments the memory address to use and decrements the byte count. If the byte count is still greater than 0, steps 2 through 4 are repeated until the count reaches 0.
- At that time, the DMA controller interrupts the CPU to let it know that the transfer is now complete.
- When the operating system starts up, it does not have to copy the disk block to memory; it is alreadythere.

- The DMA controller requests the disk controller to transfer data from the disk controller's buffer to the main memory. In the first step, the CPU issues a command to the disk controller telling it to read data from the disk into its internal buffer.

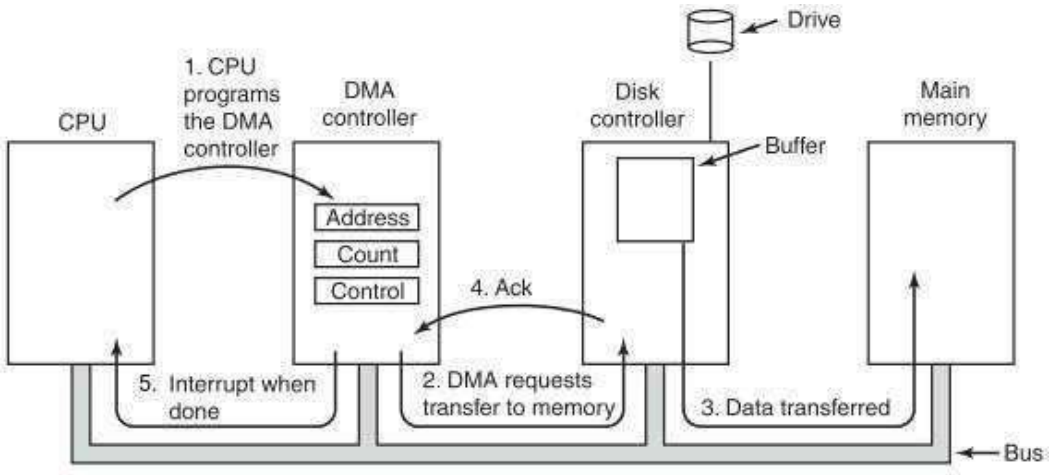


Fig 5: Operations in DMA

Source: Miles J. Murdocca and Vincent P. Heuring, —Computer Architecture and Organization: An Integrated approach.