Computer Organization and Architecture
B. E. 4th SEM BIT Durg
Session 2022-23

Usha Kiran

April 13, 2023

Unit-III The Memory System

1 Overview

- Various Technologies Used in Memory Design
- Memory Hierarchy
- Cache Memory
  - Cache Mapping Functions: Direct mapping, associative mapping, k-way set associative mapping
  - Replacement Algorithms
  - Write Policy
  - Cache Coherency
- Virtual Memory
- Main Memory
- Auxiliary Memory
- Associative Memory
- Multi-module Memories and Interleaving

2 Memory Technologies

The term Memory can be defined as a collection of data in a specific format. It is used to store instructions and process data. The memory comprises a large array or group of words or bytes, each with its own location. The primary motive of a computer system is to execute programs. These programs, along with the information they access, should be in the main memory during execution. The CPU fetches instructions from memory according to the value of the program counter.

To achieve a degree of multiprogramming and proper utilization of memory, memory management is important. Many memory management methods exist, reflecting various approaches, and the effectiveness of each algorithm depends on the situation.

2.1 Performance Measures

Memory latency is traditionally quoted using two measures—access time and cycle time. Access time is the time between when a read is requested and when the desired word arrives, cycle time is the minimum time between requests to memory. One reason that cycle time is greater than access time is that the memory needs the address lines to be stable between accesses.

3 Technologies Used in Memory Design

- RAM
  - DRAM
  - SRAM
- ROM
  - MROM
  - PROM
  - EPROM
  - EEPROM
  - Flash Memory

4 RAM

4.1 Introduction

RAM (Random Access Memory) is the hardware in a computing device where the operating system (OS), application programs and data in current use are kept so they can be quickly reached by the device's processor. RAM is the main memory in a computer. It is much faster to read from and write to than other kinds of storage, such as a hard disk drive (HDD), solid-state drive (SSD) or optical drive.

Random Access Memory is volatile. That means data is retained in RAM as long as the computer is on, but it is lost when the computer is turned off. When the computer is rebooted, the OS and other files are reloaded into RAM, usually from an HDD or SSD.

4.2 Function of RAM

Because of its volatility, RAM can't store permanent data. RAM can be compared to a person's short-term memory, and a hard disk drive to a person's long-term memory. Short-term memory is focused on immediate work, but it can only keep a limited number of facts in view at any one time. When a person's short-term memory fills up, it can be refreshed with facts stored in the brain's long-term memory.

A computer also works this way. If RAM fills up, the computer's processor must repeatedly go to the hard disk to overlay the old data in RAM with new data. This process slows the computer's operation.

Types of RAM

- Static Random Access Memory (SRAM)
- Dynamic Random Access Memory (DRAM)

4.3 SRAM

- Static random-access memory (static RAM or SRAM) is a type of random-access memory (RAM) that uses latching circuitry (flip-flop) to store each bit.
- SRAM is volatile memory; data is lost when power is removed.
- SRAM will hold its data permanently in the presence of power, while data in DRAM decays in seconds and thus must be periodically refreshed.
- SRAM is faster than DRAM but it is more expensive in terms of silicon area and cost; it is typically used for the cache and internal registers of a CPU while DRAM is used for a computer's main memory.
- Since SRAM requires more transistors per bit to implement, it is less dense and more expensive than DRAM and also has a higher power consumption during read or write access.
- Used in personal computers, workstations, routers and peripheral equipment: CPU register files, internal CPU caches, internal GPU caches and external burst mode SRAM caches, hard disk buffers, router buffers, etc.

4.4 DRAM Technology

- The main memory of virtually every desktop or server computer sold since 1975 is composed of semiconductor DRAMs,.
- Access memory address as accessing Matrix.
- Emphasis is on cost per bit and capacity.
- The dynamic nature of the circuits in DRAM require data to be written back after being read, hence the difference between the access time and the cycle time as well as the need to refresh.
- Dynamic random-access memory (dynamic RAM or DRAM) is a type of random-access semiconductor memory that stores each bit of data in a memory cell, usually consisting of a tiny capacitor and a transistor.
- While most DRAM memory cell designs use a capacitor and transistor, some only use two transistors.
- In the designs where a capacitor is used, the capacitor can either be charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1.
- The electric charge on the capacitors gradually leaks away; without intervention the data on the capacitor would soon be lost.
- To prevent this, DRAM requires an external memory refresh circuit which periodically rewrites the data in the capacitors, restoring them to their original charge.
- This refresh process is the defining characteristic of dynamic random-access memory, in contrast to static random-access memory (SRAM) which does not require data to be refreshed.
- Unlike flash memory, DRAM is volatile memory (vs. non-volatile memory), since it loses its data quickly when power is removed.
- DRAM typically takes the form of an integrated circuit chip, which can consist of dozens to billions of DRAM memory cells.
- DRAM chips are widely used in digital electronics where low-cost and high-capacity computer memory is required.
- One of the largest applications for DRAM is the main memory (colloquially called the "RAM" in modern computers and graphics cards (where the "main memory" is called the graphics memory).

5 ROM

ROM, which stands for read only memory, is a memory device or storage medium that stores information permanently. It is also the primary memory unit of a computer along with the random access memory (RAM). It is called read only memory as we can only read the programs and data stored on it but cannot write on it. It is restricted to reading words that are permanently stored within the unit.

The manufacturer of ROM fills the programs into the ROM at the time of manufacturing the ROM. After this, the content of the ROM can't be altered, which means you can't reprogram, rewrite, or erase its content later. However, there are some types of ROM where you can modify the data.

ROM contains special internal electronic fuses that can be programmed for a specific interconnection pattern (information). The binary information stored in the chip is specified by the designer and then embedded in the unit at the time of manufacturing to form the required interconnection pattern (information). Once the pattern (information) is established, it stays within the unit even when the power is turned off. So, it is a non-volatile memory as it holds the information even when the power is turned off, or you shut down your computer.

The information is added to a RAM in the form of bits by a process known as programming the ROM as bits are stored in the hardware configuration of the device. So, ROM is a Programmable Logic Device (PLD).

ROM is an abbreviation for Read Only Memory, and is also referred to as firmware. It is an integrated circuit programmed with certain data during the time of it's manufacturing. Read only memories are used not only in computer systems but also in many other electronic devices such as IoT devices like digital assistants, smart gadgets such as smartwatches, etc as well.

1. MROM
2. PROM

3. EPROM
4. EEPROM
5. Flash Memory

5.1 Types of ROM:

- Masked Read Only Memory (MROM): It is the oldest type of read only memory (ROM). It has become obsolete so it is not used anywhere in today's world. It is a hardware memory device in which programs and instructions are stored at the time of manufacturing by the manufacturer. So it is programmed during the manufacturing process and can't be modified, reprogrammed, or erased later.

- The MROM chips are made of integrated circuits. Chips send a current through a particular input-output pathway determined by the location of fuses among the rows and columns on the chip. The current has to pass along a fuse-enabled path, so it can return only via the output the manufacturer chooses. This is the reason the rewriting and any other modification is not impossible in this memory.

- Programmable Read Only Memory (PROM): PROM is a blank version of ROM. It is manufactured as blank memory and programmed after manufacturing. We can say that it is kept blank at the time of manufacturing. You can purchase and then program it once using a special tool called a programmer.

  In the chip, the current travels through all possible pathways. The programmer can choose one particular path for the current by burning unwanted fuses by sending a high voltage through them. The user has the opportunity to program it or to add data and instructions as per his requirement. Due to this reason, it is also known as the user-programmed ROM as a user can program it.

  To write data onto a PROM chip; a device called PROM programmer or PROM burner is used. The process or programming a PROM is known as burning the PROM. Once it is programmed, the data cannot be modified later, so it is also called as one-time programmable device. Uses: It is used in cell phones, video game consoles, medical devices, RFID tags, and more.

- 3) Erasable and Programmable Read Only Memory (EPROM): EPROM is a type of ROM that can be reprograme and erased many times. The method to erase the data is very different; it comes with a quartz window through which a specific frequency of ultraviolet light is passed for around 40 minutes to erase the data. So, it retains its content until it is exposed to the ultraviolet light. You need a special device called a PROM programmer or PROM burner to reprogram the EPROM.

  Uses: It is used in some micro-controllers to store program, e.g., some versions of Intel 8048 and the Freescale 68HC11.

- Electrically Erasable and Programmable Read Only Memory (EEPROM): ROM is a type of read only memory that can be erased and reprogrammed repeatedly, up to 10000 times. It is also known as Flash EEPROM as it is similar to flash memory. It is erased and reprogrammed electrically without using ultraviolet light. Access time is between 45 and 200 nanoseconds. The data in this memory is written or erased one byte at a time; byte per byte, whereas, in flash memory data is written and erased in blocks. So, it is faster than EEPROM. It is used for storing a small amount of data in computer and electronic systems and devices such as circuit boards.

  Uses: The BIOS of a computer is stored in this memory.

- FLASH ROM: It is an advanced version of EEPROM. It stores information in an arrangement or array of memory cells made from floating-gate transistors. The advantage of using this memory is that you can delete or write blocks of data around 512 bytes at a particular time. Whereas, in EEPROM, you can delete or write only 1 byte of data at a time. So, this memory is faster than EEPROM.

  It can be reprogrammed without removing it from the computer. Its access time is very high, around 45 to 90 nanoseconds. It is also highly durable as it can bear high temperature and intense pressure.

  Uses: It is used for storage and transferring data between a personal computer and digital devices. It is used in USB flash drives, MP3 players, digital cameras, modems and solid-state drives (SSDs). The BIOS of many modern computers are stored on a flash memory chip, called flash BIOS.

6 Computer Memory Hierarchy

Memory Hierarchy Design is divided into 2 main types:

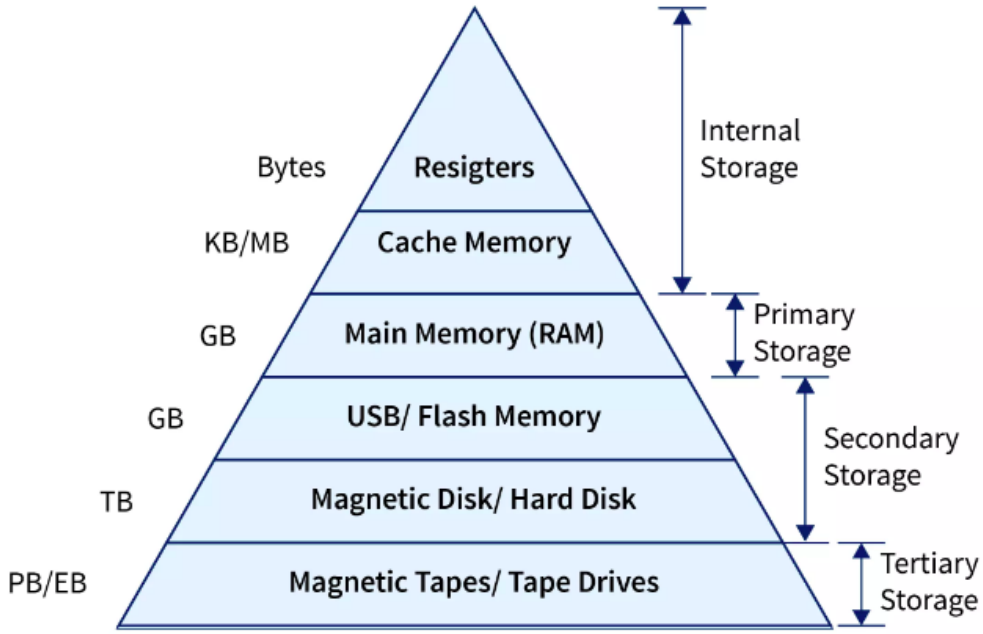


Figure 1: Computer Memory Hierarchy

- External Memory or Secondary Memory – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
- Internal Memory or Primary Memory – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

There are typically four levels of memory in a memory hierarchy:

- Registers: Registers are small, high-speed memory units located in the CPU. The register is usually an SRAM or static RAM in the computer processor that is used to hold the data word that is typically 64 bits or 128 bits. A majority of the processors make use of a status word register and an accumulator. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity.
- Cache Memory: Cache memory is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory, hence, minimize the time to access data. We can also find cache memory in the processor. In case the processor has a single-core, it will rarely have multiple cache levels. The present multi-core processors would have three 2-levels for every individual core, and one of the levels is shared.
- Main Memory: Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. Main memory is used to store data and instructions that are currently in use by the CPU.
- USB/ Flash Memory: A USB flash drive (also called a thumb drive in the US, or a memory stick in the UK)[1][note 1] is a data storage device that includes flash memory with an integrated USB interface. It is typically removable, re-writable and much smaller than an optical disc.
- Magnetic Disks: It is a secondary storage device, with high storage capacity, low cost and low speed. In a computer, the magnetic disks are circular plates that's fabricated with plastic or metal with a magnetised material. Two faces of a disk are frequently used, and many disks can be stacked on a single spindle by read/write heads that are obtainable on every plane. The disks in a computer jointly turn at high speed.
- Magnetic Tape: It is also known as tertiary storage. Magnetic tape refers to a normal magnetic recording designed with a slender magnetizable overlay that covers an extended, thin strip of plastic film. It is used mainly to back up huge chunks of data. When a computer needs to access a strip, it will first mount it to access the information. Once the information is allowed, it will then be unmounted. The actual access time of a computer memory would be slower within a magnetic strip, and it will take a few minutes for us to access a strip.

6.1 Characteristics of Memory

We can infer the following characteristics of Memory Hierarchy Design from Figure 1:

- Capacity: It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
- Access Time: It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
- Performance: Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
- Cost per bit: As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

Level	1	2	3	4
Name	Register	Cache	Main Memory	Secondary Memory
Size	¡1 KB	less than 16 MB	¡16GB	¿100 GB
Implementation	Multi-ports	On-chip/SRAM	DRAM (capacitor memory)	Magnetic
Access Time	0.25ns to 0.5ns	0.5 to 25ns	80ns to 250ns	50 lakh ns
Bandwidth	20000 to 1 lakh MBytes	5000 to 15000	1000 to 5000	20 to 150

Table 1: Caption

6.2 Processor Registers

- Quickly accessible to a computer's processor.
- Top of the memory hierarchy.
- Small capacity and fast accessing.
- May have specific hardware functions, and may be read-only or write-only mode.
- Used for arithmetic operations and is manipulated or tested by machine instructions.
- When a computer program accesses the same data repeatedly, this is called "locality of reference".
- Registers are normally measured by the number of bits they can hold, for example, an "8-bit register", "32-bit register", "64-bit register", or even more.

A processor often contains several kinds of registers.
Data Register

- User-accessible registers can be read or written by machine instructions. Types: Data Register, Address Register.
- Data registers can hold numeric data values such as integer and, in some architectures, floating-point values, as well as characters, small bit arrays and other data.

Address Register Address registers hold addresses and are used by instructions that indirectly access primary memory.

6.3 location of register in CPU

6.4 Cache Memory

- Cache Memory is a special very high-speed memory. It is used to speed up and synchronize with high-speed CPU. Cache memory is costlier than main memory or disk memory but more economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed. Cache memory is used to reduce the average time to access data from the Main memory. The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.

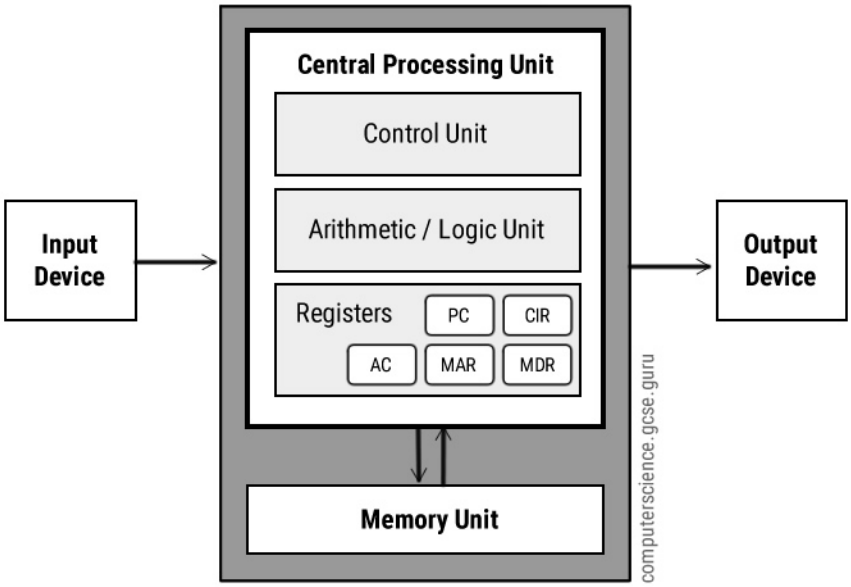


Figure 2: Register Position in CPU

6.5 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 occurred and data is read from the cache. If the processor does not find the memory location in the cache, a cache miss has occurred. For a cache miss, the cache allocates a new entry and copies in data from main memory, then the request is fulfilled from the contents of the cache.

The performance of cache memory is frequently measured in terms of a quantity called Hit ratio.

Hit ratio(H) = hit / (hit + miss) = no. of hits/total accesses Miss ratio = miss / (hit + miss) = no. of miss/total accesses = 1 - hit ratio(H)

We can improve Cache performance using higher cache block size, and higher associativity, reduce miss rate, reduce miss penalty, and reduce the time to hit in the cache.

6.6 Application of Cache Memory:

- Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.
- The correspondence between the main memory blocks and those in the cache is specified by a mapping function.
- Primary Cache – A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.
- Secondary Cache – Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.
- Spatial Locality of reference This says that there is a chance that the element will be present in close proximity to the reference point and next time if again searched then more close proximity to the point of reference.
- Temporal Locality of reference In this Least recently used algorithm will be used. Whenever there is page fault occurs within a word will not only load word in main memory but complete page fault will be loaded because the spatial locality of reference rule says that if you are referring to any word next word will be referred to in its register that's why we load complete page table so the complete block will be loaded.

7 Main Memory

The main memory is the fundamental storage unit in a computer system. It is associatively large and quick memory and saves programs and information during computer operations. The technology that makes the main

memory work is based on semiconductor integrated circuits.

RAM is the main memory. Integrated circuit Random Access Memory (RAM) chips are applicable in two possible operating modes are as follows.

Static: It consists of internal flip-flops, which store the binary information. The stored data remains solid considering power is provided to the unit. The static RAM is simple to use and has smaller read and write cycles. **Dynamic:** It saves the binary data in the structure of electric charges that are used to capacitors. The capacitors are made available inside the chip by Metal Oxide Semiconductor (MOS) transistors. The stored value on the capacitors contributes to discharge with time and thus, the capacitors should be regularly recharged through stimulating the dynamic memory.

8 Cache Mapping

Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. Further, a means is needed for determining which main memory block currently occupies a cache line. The choice of the mapping function dictates how the cache is organized.

There are three types of cache mapping:

- Direct Mapping
- Associative/Fully-associative Mapping
- Set-associative Mapping

8.1 Direct Mapping

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. The mapping is expressed as

$$i = j \ module \ m$$

Where,

$i = Cache\ line\ number$
$j = Main\ memory\ block\ number$
$m = Number\ of\ lines\ in\ the\ cache$

In direct mapping cache, instead of storing total address information with data in cache only part of address bits is stored along with data.

The new data has to be stored only in a specified cache location as per the mapping rule for direct mapping. So it doesn't need replacement algorithm.

Advantages of direct mapping

- The direct mapping technique is simple and inexpensive to implement.
- Here only tag field is required to match while searching word that is why it fastest cache.
- Direct mapping cache is less expensive compared to associative cache mapping.

Disadvantages of direct mapping

- Its main disadvantage is that there is a fixed cache location for any given block. Thus, if a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low (a phenomenon known as thrashing).
- It has higher miss ratio which is also known as conflict miss. The reason behind this is, even if cache line is empty we cannot allocate insert block into it, because blocks can be inserted into fixed line number in cache.
- The performance of direct mapping cache is not good as requires replacement for data-tag value.

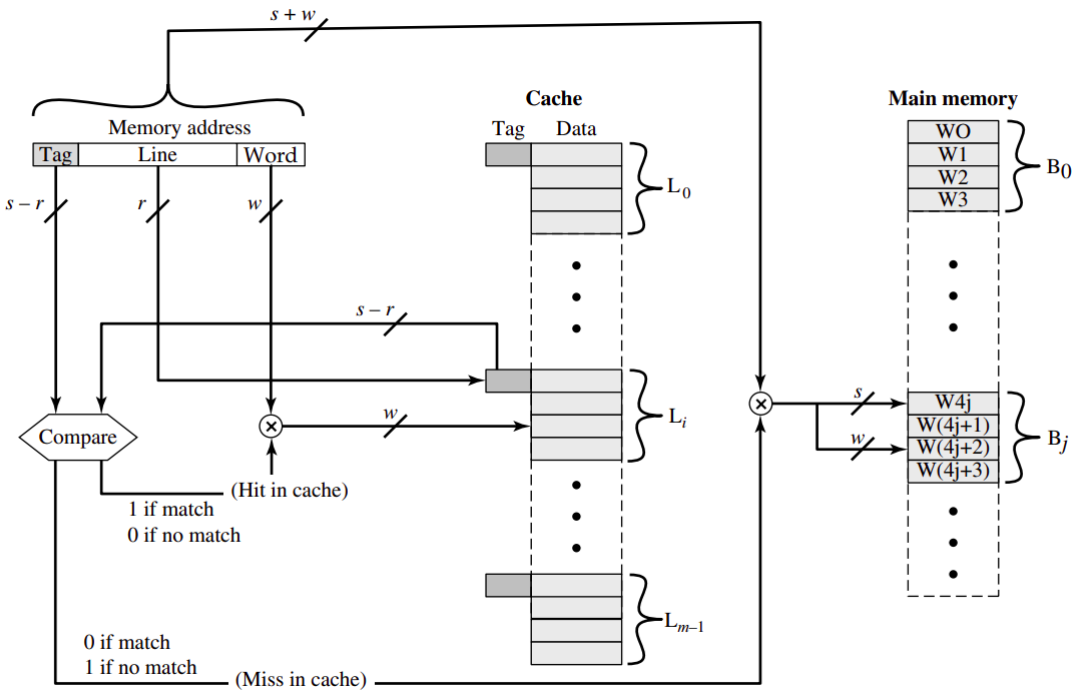


Figure 3: Direct Mapping

One approach to lower the miss penalty is to remember what was discarded in case it is needed again. Since the discarded data has already been fetched, it can be used again at a small cost. Such recycling is possible using a victim cache. Victim cache was originally proposed as an approach to reduce the conflict misses of direct mapped caches without affecting its fast access time. Victim cache is a fully associative cache, whose size is typically 4 to 16 cache lines, residing between a direct mapped L1 cache and the next level of memory.

8.2 Associative/Fully-associative Mapping

Associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache. In this case, the cache control logic interprets a memory address simply as a Tag and a Word field. The Tag field uniquely identifies a block of main memory. To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's tag for a match.

$Address\ length = (s + w) bits$

$Number\ of\ addressable\ units = 2^{s+w}\ words\ or\ bytes$

$Block\ size = line\ size = 2^{w}\ words\ or\ bytes$

$Number\ of\ blocks\ in\ main\ memory = w = 2^{s+w}/2^{w} = 2^{s}$

$Number\ of\ lines\ in\ cache = undetermined$ $\qquad$ $Size\ of\ tage = s\ bits$

Advantages of associative mapping

The principal disadvantage of asso- ciative mapping is the complex circuitry required to examine the tags of all cache lines in parallel.

- With associative mapping, there is flexibility as to which block to replace when a new block is read into the cache. Replacement algorithms are designed to maximize the hit ratio.
- Associative mapping is fast.
- Associative mapping is easy to implement.

Disadvantages of associative mapping

- The principal disadvantage of associative mapping is the complex circuitry required to examine the tags of all cache lines in parallel.

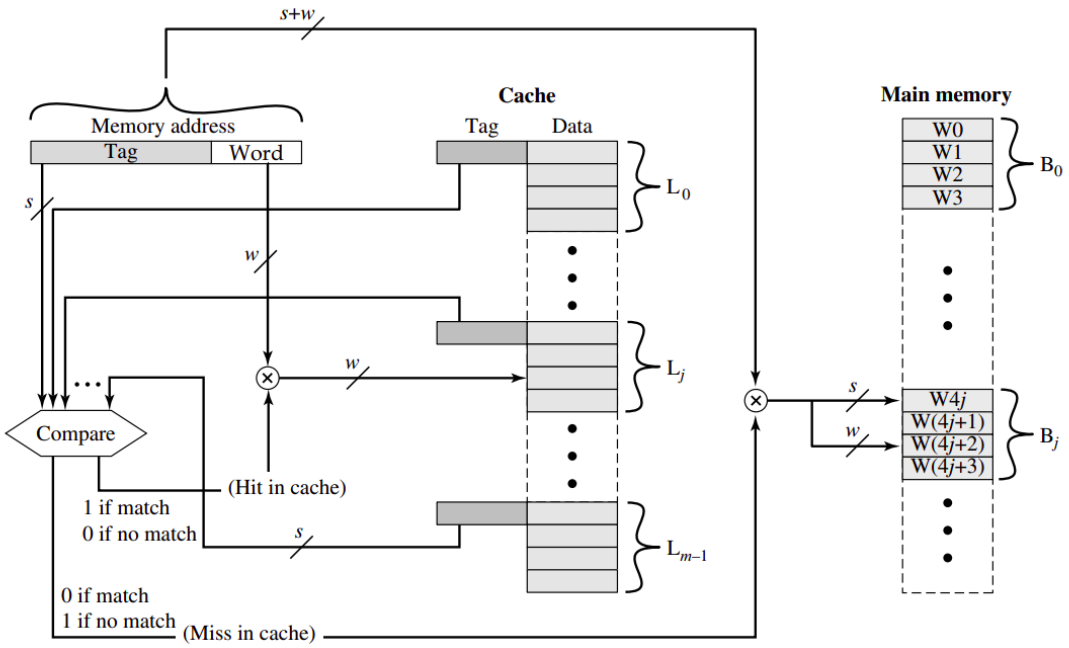


Figure 4: Associative Mapping

- Number of comparison increases.
- Number of bits in tag increases.

8.3 Set-Associative Mapping

In Set-Associative cache memory two or more words can be stored under the same index address.

Here every data word is stored along with its tag. The number of tag-data words under an index is said to form a text.

Advantages of Set-Associative mapping Set-associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages. In this case, the cache consists of a number sets, each of which consists of a number of lines. The relationships are

$$m = v * k$$

$$i = j \ modulo \ v$$

Where,

$i = cache \ set \ number$
$j = main \ memory \ block \ number$
$m = number \ of \ lines \ in \ the \ cache$
$n = number \ of \ sets$
$k = number \ of \ lines \ in \ each \ set$

Advantages

Set-Associative cache memory has highest hit-ratio compared two previous two cache memory discussed above. Thus its performance is considerably better.

Disadvantages of Set-Associative mapping

Set-Associative cache memory is very expensive. As the set size increases the cost increases.

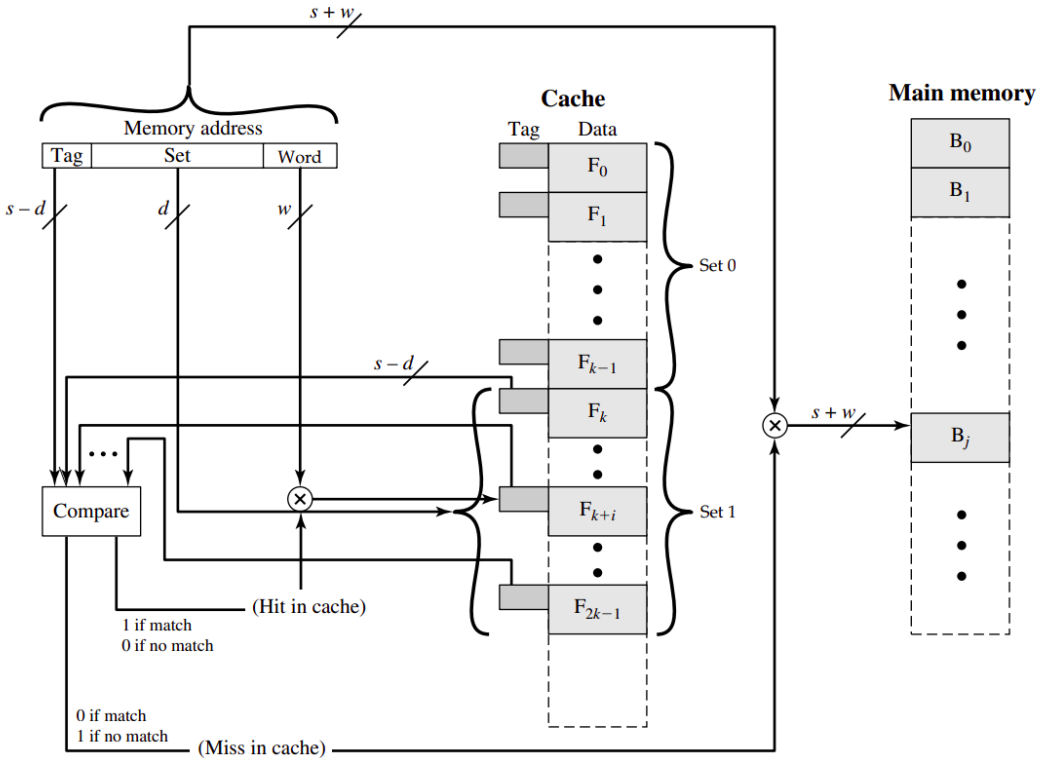


Figure 5: Set-associative Mapping

9 Replacement Algorithms

Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced. For direct mapping, there is only one possible line for any particular block, and no choice is possible. For the associative and set-associative techniques, a replacement algorithm is needed. To achieve high speed, such an algorithm must be implemented in hardware.

Following are some of the common and simpler types of cache replacement policies:

- First-in-first-out (FIFO) policy: The earliest inserted item in the cache will be evicted when a new item needs to be inserted.
- Last-in-first-out (LIFO) policy: The last item inserted in the cache will be evicted first.
- Least-recently used (LRU) policy: The item which is least recently used will be evicted first. This is one of the most simple and common cache replacement policies. It assumes that the more recently an item is accessed or used, the more likely it is to be used or accessed again (e.g., switching between tabs of a browser)
- Most-recently used (MRU) policy: The item which is most recently used will be evicted first. This policy is useful, when the chances of repeating the same request in the near future is unlikely (like scrolling through a social media feed or flipping through a photo album).
- Least-frequently-used(LFU) policy: The cache algorithm maintains a counter on the number of times an item in the cache is accessed. It will evict the least frequently accessed item in order to add a new item.
- Random replacement (RR) policy: The cache algorithm randomly selects an item to evict.

10 Write Policy

When a block that is resident in the cache is to be replaced, there are two cases to consider.

If the old block in the cache has not been altered, then it may be overwritten with a new block without first writing out the old block. If at least one write operation has been performed on a word in that line of the cache, then main memory must be updated by writing the line of cache out to the block of memory before bringing in the new block. A variety of write policies, with performance and economic trade-offs, is possible.

There are two problems to contend with. First, more than one device may have access to main memory. For example, an I/O module may be able to read-write directly to memory. If a word has been altered only in the cache, then the corresponding memory word is invalid. Further, if the I/O device has altered main memory, then the cache word is invalid. A more complex problem occurs when multiple processors are attached to the same bus and each processor has its own local cache. Then, if a word is altered in one cache, it could conceivably invalidate a word in other caches.

- **Write Through -** The simplest technique is called **write through.** Using this technique, all write operations are made to main memory as well as to the cache, ensuring that main memory is always valid. Any other processor–cache module can monitor traffic to main memory to maintain consistency within its own cache. The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck.

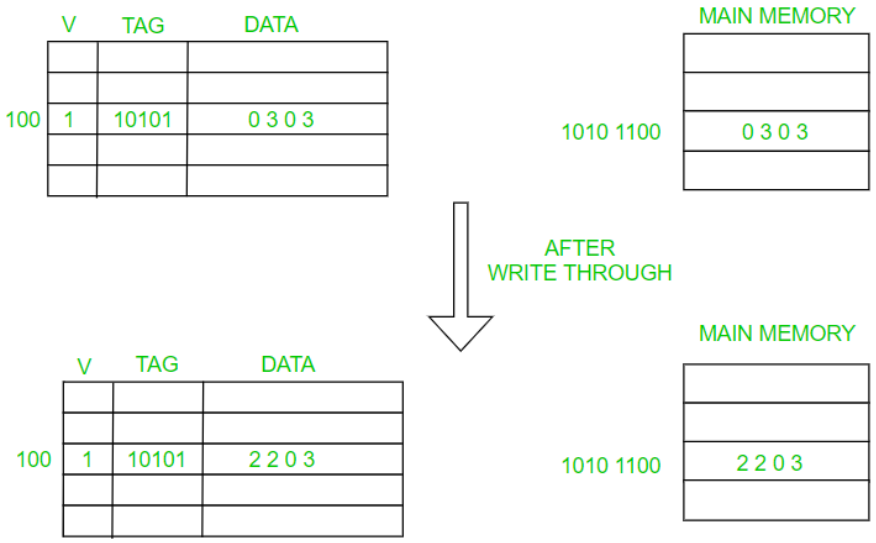


Figure 6: Write Through Policy

- **Write Back -** An alternative technique, known as **write back**, minimizes memory writes. With write back, updates are made only in the cache. When an update occurs, a dirty bit, or use bit, associated with the line is set. Then, when a block is replaced, it is written back to main memory if and only if the dirty bit is set. The problem with write back is that portions of main memory are invalid, and hence accesses by I/O modules can be allowed only through the cache. This makes for complex circuitry and a potential bottleneck.

Write Through Method	Write Back Method
In this method main memory is updated with every memory write operation as well as cache memory is updated in parallel if it contains the word at the specified address.	In this method only cache location is updated during write operation.
Main memory always contains same data as cache.	Main memory and cache memory may have different data.
Number of memory write operation in a typical program is more.	Number of memory write operation in a typical program is less.
When I/O device communicated through DMA would receive most recent data.	When I/O device communicated through DMA would not receive most recent data.
It is a process of writing cache and main memory simultaneously.	It is a process of writing cache and data is removed from cache, first copied to main memory.

Table 2: Difference Between Write Back and Write Through Policies

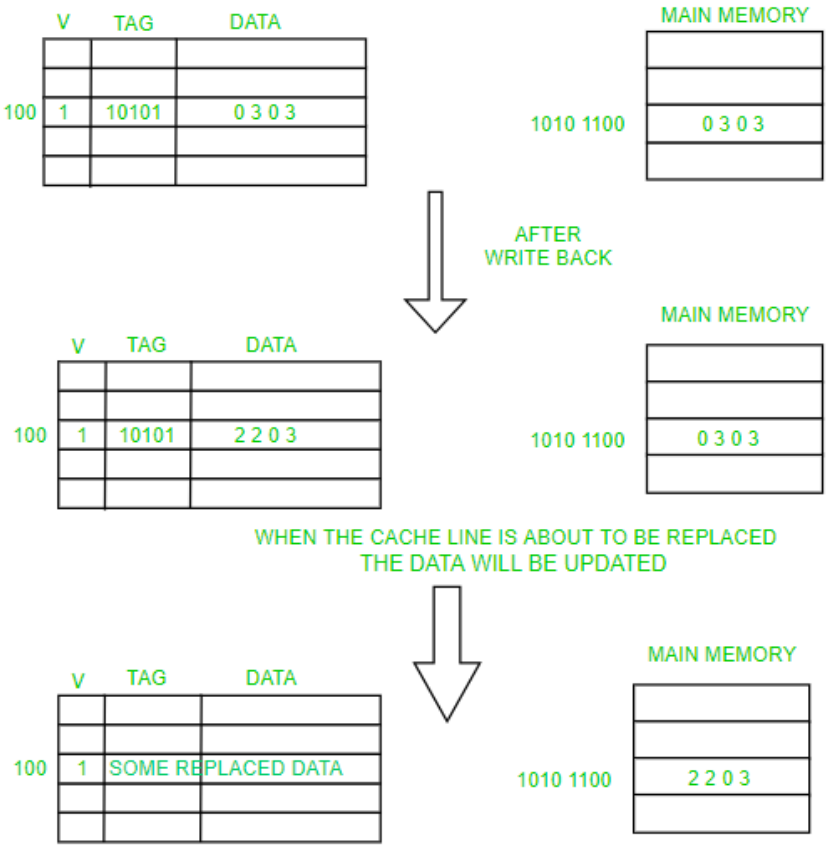


Figure 7: Write Back Policy

11 Cache Coherency

In a bus organization in which more than one device (typically a processor) has a cache and main memory is shared, a new problem is introduced. If data in one cache are altered, this invalidates not only the corresponding word in main memory, but also that same word in other caches (if any other cache happens to have that same word). Even if a write-through policy is used, the other caches may contain in-valid data. A system that prevents this problem is said to maintain cache coherency.

Possible approaches to cache coherency include the following:

- Bus watching with write through: Each cache controller monitors the address lines to detect write operations to memory by other bus masters. If another master writes to a location in shared memory that also resides in the cache memory, the cache controller invalidates that cache entry. This strategy depends on the use of a write-through policy by all cache controllers.
- Hardware transparency: Additional hardware is used to ensure that all updates to main memory via cache are reflected in all caches. Thus, if one processor modifies a word in its cache, this update is written to main memory. In addition, any matching words in other caches are similarly updated.
- Noncacheable memory: Only a portion of main memory is shared by more than one processor, and this is designated as noncacheable. In such a system, all accesses to shared memory are cache misses, because the shared memory is never copied into the cache. The noncacheable memory can be identified using chip-select logic or high-address bits.

12 Virtual Memory

Virtual memory is the partition of logical memory from physical memory. This partition supports large virtual memory for programmers when only limited physical memory is available.

Virtual memory can give programmers the deception that they have a very high memory although the computer has a small main memory. It creates the function of programming easier because the programmer no longer requires to worry about the multiple physical memory available.

Virtual memory works similarly, but at one level up in the memory hierarchy. A memory management unit (MMU) transfers data between physical memory and some gradual storage device, generally a disk. This storage area can be defined as a swap disk or swap file, based on its execution. Retrieving data from physical memory is much faster than accessing data from the swap disk.

There are two primary methods for implementing virtual memory are as follows

Paging

Paging is a technique of memory management where small fixed-length pages are allocated instead of a single large variable-length contiguous block in the case of the dynamic allocation technique. In a paged system, each process is divided into several fixed-size 'chunks' called pages, typically 4k bytes in length. The memory space is also divided into blocks of the equal size known as frames.

Advantages of Paging

There are the following advantages of Paging are:

In Paging, there is no requirement for external fragmentation.

In Paging, the swapping among equal-size pages and page frames is clear.

Paging is a simple approach that it can use for memory management.

Disadvantage of Paging

There are the following disadvantages of Paging are:

In Paging, there can be a chance of Internal Fragmentation.

In Paging, the page table employs more memory.

Because of Multi-level Paging, there can be a chance of memory reference overhead.

Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	Write once
Mapping Function	**Line Size**
Direct	**Number of caches**
Associative	Single or two level
Set Associative	Unified or split
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

Figure 8: Cache Design Elements

13 Virtual Memory

Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses.

The size of virtual storage is limited by the addressing scheme of the computer system and the amount of secondary memory is available not by the actual number of the main storage locations.

Dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of the main memory such that it occupies different places in the main memory at different times during the course of execution. A process may be broken into a number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time address translation and use of page or segment table permits this. If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded into memory whenever required. Virtual memory is implemented using Demand Paging or Demand Segmentation.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.

14 Memory module and Interleaving

Memory Interleaving is less or More an Abstraction technique. Though it's a bit different from Abstraction. It is a Technique that divides memory into a number of modules such that Successive words in the address space are placed in the Different modules.

14.1 Consecutive Word in a Module

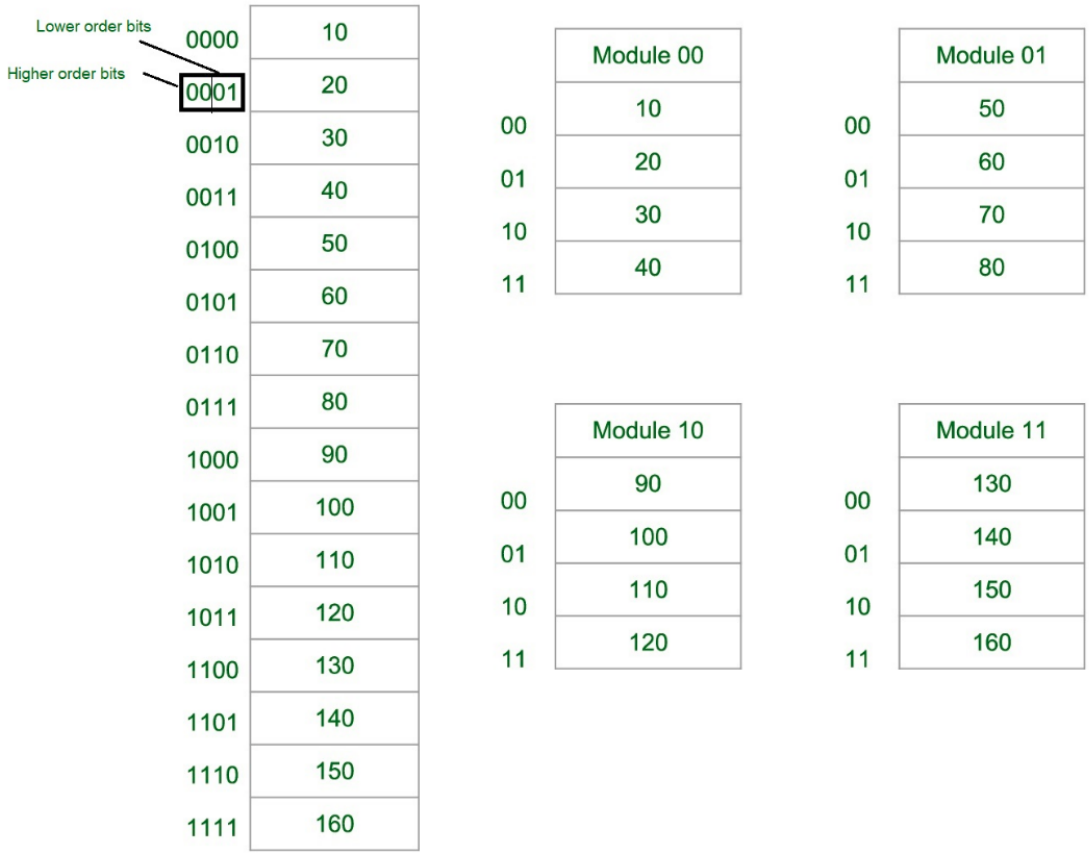


Figure 9: Consecutive word in a module

Let us assume 16 Data's to be Transferred to the Four Module. Where Module 00 be Module 1, Module 01 be Module 2, Module 10 be Module 3 & Module 11 be Module 4. Also, 10, 20, 30. . . .130 are the data to be transferred.

From the Figure 9 in Module 1, 10 [Data] is transferred then 20, 30 & finally, 40 which are the Data. That means the data are added consecutively in the Module till its max capacity.

Most significant bit (MSB) provides the Address of the Module & the least significant bit (LSB) provides the address of the data in the module.

For Example, to get 90 (Data) 1000 will be provided by the processor. This 10 will indicate that the data is in module 10 (module 3) & 00 is the address of 90 in Module 10 (module 3). So,

Module 1 Contains Data : 10, 20, 30, 40
Module 2 Contains Data : 50, 60, 70, 80
Module 3 Contains Data : 90, 100, 110, 120
Module 4 Contains Data : 130, 140, 150, 160

14.2 Consecutive Word in Consecutive Module:

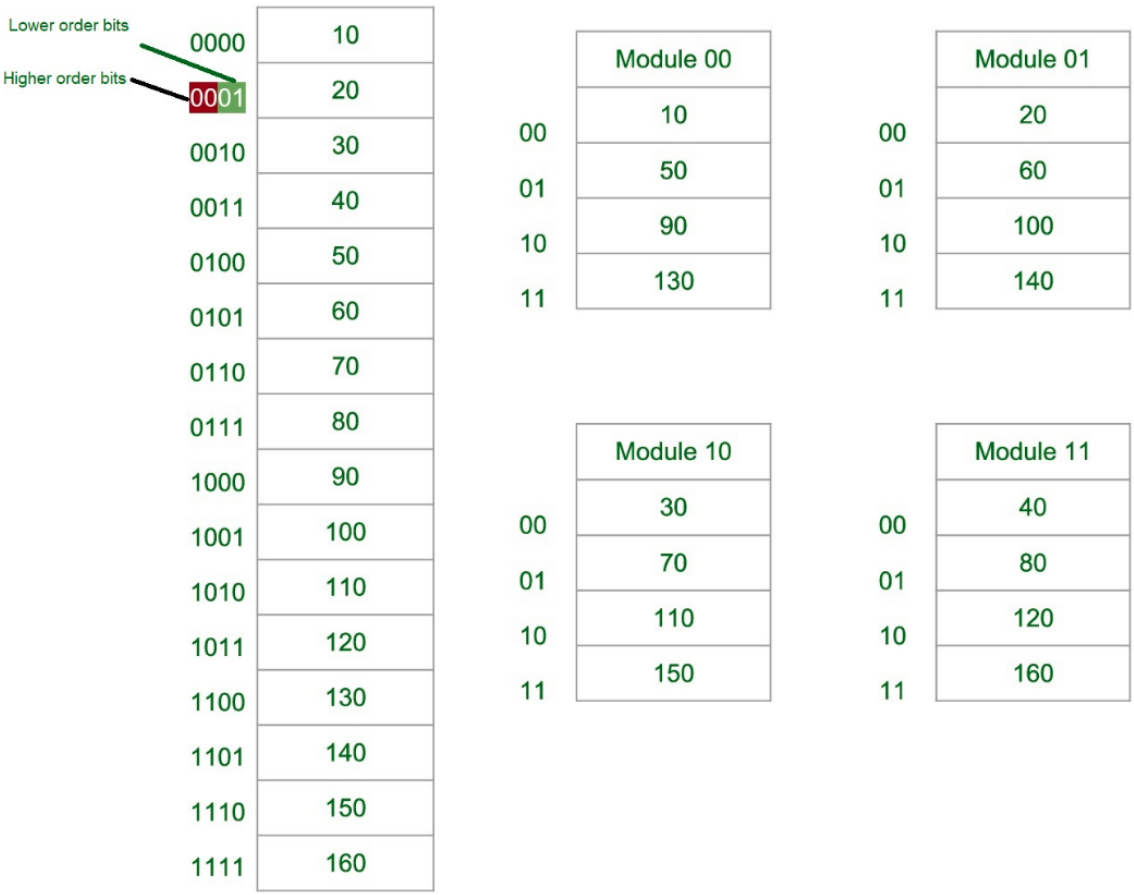


Figure 10: Consecutive word in consecutive module

Now again we assume 16 Data's to be transferred to the Four Module. But Now the consecutive Data are added in Consecutive Module. That is, 10 [Data] is added in Module 1, 20 [Data] in Module 2 and So on.

Least Significant Bit (LSB) provides the Address of the Module & Most significant bit (MSB) provides the address of the data in the module.

For Example, to get 90 (Data) 1000 will be provided by the processor. This 00 will indicate that the data is in module 00 (module 1) & 10 is the address of 90 in Module 00 (module 1). That is,

Module 1 Contains Data : 10, 50, 90, 130
Module 2 Contains Data : 20, 60, 100, 140
Module 3 Contains Data : 30, 70, 110, 150
Module 4 Contains Data : 40, 80, 120, 160

Why do we use Memory Interleaving? [Advantages]: Whenever Processor requests Data from the main memory. A block (chunk) of Data is Transferred to the cache and then to Processor. So whenever a cache miss occurs the Data is to be fetched from the main memory. But main memory is relatively slower than the cache. So to improve the access time of the main memory interleaving is used.

We can access all four Modules at the same time thus achieving Parallelism. From Figure 10 the data can be acquired from the Module using the Higher bits. This method Uses memory effectively.

15 Associative Memory

An associative memory can be treated as a memory unit whose saved information can be recognized for approach by the content of the information itself instead of by an address or memory location. Associative memory is also known as Content Addressable Memory (CAM).

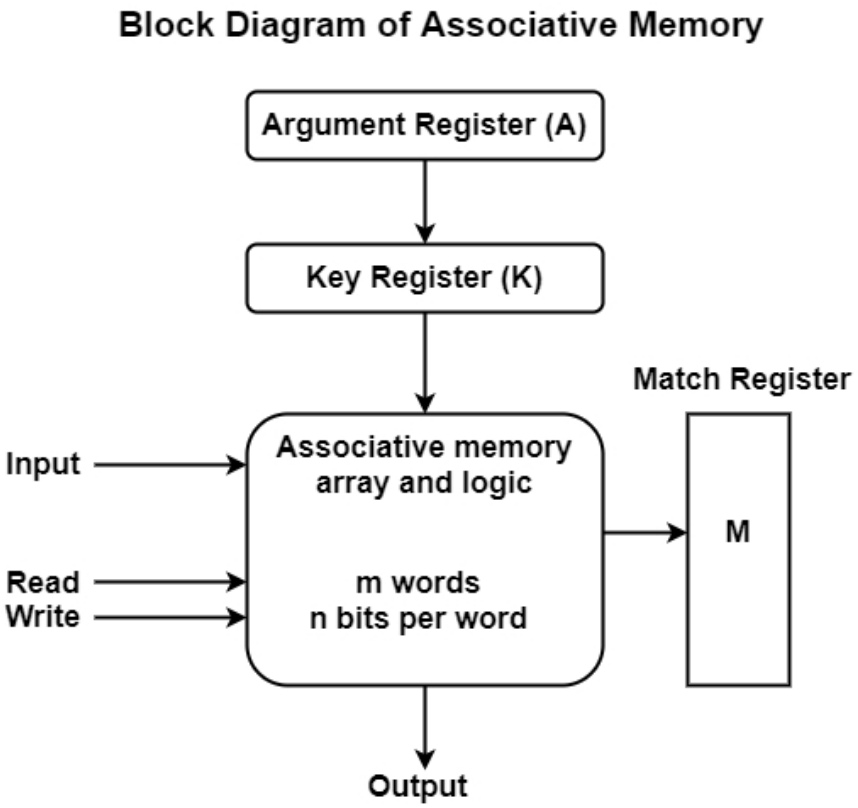


Figure 11: Associative Memory

The block diagram of associative memory is shown in the Figure 11. It includes a memory array and logic for m words with n bits per word. The argument register A and key register K each have n bits, one for each bit of a word.

The match register M has m bits, one for each memory word. Each word in memory is related in parallel with the content of the argument register.

The words that connect the bits of the argument register set an equivalent bit in the match register. After the matching process, those bits in the match register that have been set denote the fact that their equivalent words have been connected.

Reading is proficient through sequential access to memory for those words whose equivalent bits in the match register have been set.

The key register supports a mask for selecting a specific field or key in the argument word. The whole argument is distinguished with each memory word if the key register includes all 1's.

Hence, there are only those bits in the argument that have 1's in their equivalent position of the key register are compared. Therefore, the key gives a mask or recognizing a piece of data that determines how the reference to memory is created.

The following Figure 12 can define the relation between the memory array and the external registers in associative memory.

The cells in the array are considered by the letter C with two subscripts. The first subscript provides the word number and the second determines the bit position in the word. Therefore cell C_{ij} is the cell for bit j in word i.

A bit in the argument register is compared with all the bits in column j of the array supported that $K_j = 1$. This is completed for all columns j = 1, 2 . . . , n.

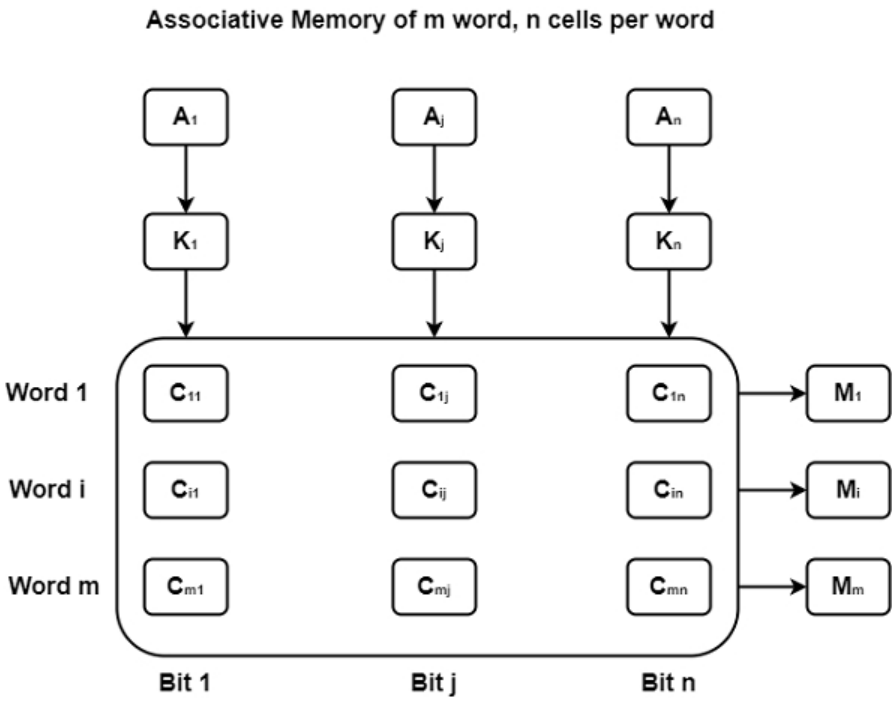


Figure 12: Memory Array

If a match appears between all the unmasked bits of the argument and the bits in word i, the equivalent bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

Associative memory of conventional semiconductor memory (usually RAM) with added comparison circuity that enables a search operation to complete in a single clock cycle. It is a hardware search engine, a special type of computer memory used in certain very high searching applications.

Applications of Associative memory :-

It can be only used in memory allocation format. It is widely used in the database management systems, etc.

Advantages of Associative memory :-

It is used where search time needs to be less or short. It is suitable for parallel searches. It is often used to speedup databases. It is used in page tables used by the virtual memory and used in neural networks.

Disadvantages of Associative memory :-

It is more expensive than RAM. Each cell must have storage capability and logical circuits for matching its content with external argument.

16 Auxiliary Memory

An Auxiliary memory is referred to as the lowest-cost, highest-space, and slowest-approach storage in a computer system. It is where programs and information are preserved for long-term storage or when not in direct use. The most typical auxiliary memory devices used in computer systems are magnetic disks and tapes.

Magnetic Disks

A magnetic disk is a round plate generated of metal or plastic coated with magnetized material. There are both sides of the disk are used and multiple disks can be stacked on one spindle with read/write heads accessible on each surface.

All disks revolve together at high speed and are not stopped or initiated for access purposes. Bits are saved in the magnetized surface in marks along concentric circles known as tracks. The tracks are frequently divided into areas known as sectors.

In this system, the lowest quantity of data that can be sent is a sector. The subdivision of one disk surface into tracks and sectors is displayed in the Figure 13.

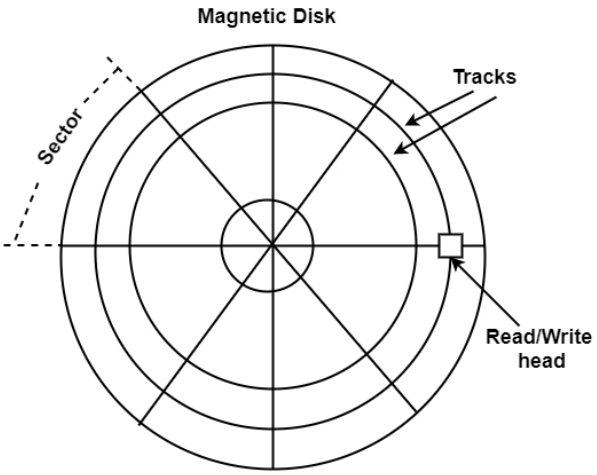


Figure 13: Magnetic Disk

17 Magnetic Tape

Magnetic tape transport includes the robotic, mechanical, and electronic components to support the methods and control structure for a magnetic tape unit. The tape is a layer of plastic coated with a magnetic documentation medium.

Bits are listed as a magnetic stain on the tape along various tracks. There are seven or nine bits are recorded together to form a character together with a parity bit. Read/write heads are mounted one in each track therefore that information can be recorded and read as a series of characters.

Magnetic tape units can be stopped, initiated to move forward, or in the opposite, or it can be reversed. However, they cannot be initiated or stopped fast enough between single characters. For this reason, data is recorded in blocks defined as records. Gaps of unrecorded tape are added between records where the tape can be stopped.

The tape begins affecting while in a gap and achieves its permanent speed by the time it arrives at the next record. Each record on tape has a recognition bit design at the starting and end. By reading the bit design at the starting, the tape control recognizes the data number.

17.1 Memory allocation strategies

First Fit Allocation
These are Contiguous memory allocation techniques. First-Fit Memory Allocation: This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, first job claims the first available memory with space more than or equal to it's size. The operating system doesn't search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

Advantages of First-Fit Allocation in Operating Systems:

- Simple and efficient search algorithm
- Minimizes memory fragmentation
- Fast allocation of memory

Disadvantages of First-Fit Allocation in Operating Systems:

- Poor performance in highly fragmented memory
- May lead to poor memory utilization
- May allocate larger blocks of memory than required.

Best-Fit Allocation in Operating System
Best-Fit Memory Allocation: This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

Advantages of Best-Fit Allocation :

- Memory Efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient.
- To save memory from getting wasted, it is the best method.
- Improved memory utilization
- Reduced memory fragmentation
- Minimizes external fragmentation

Disadvantages of Best-Fit Allocation :

- It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.
- Increased computational overhead
- May lead to increased internal fragmentation
- Can result in slow memory allocation times.

Worst-Fit Allocation

In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

Advantages of Worst-Fit Allocation :

Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.

Disadvantages of Worst-Fit Allocation :

It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

Unit-4 Input/Output Organizations

18 Peripheral Devices

Peripheral devices are those devices that are linked either internally or externally to a computer. These devices are commonly used to transfer data. The most common processes that are carried out in a computer are entering data and displaying processed data. Several devices can be used to receive data and display processed data. The devices used to perform these functions are called peripherals or I/O devices.

Peripherals read information from or write in the memory unit on receiving a command from the CPU. They are considered to be a part of the total computer system. As they require a conversion of signal values, these devices can be referred to as electromechanical and electromagnetic devices. The most common peripherals are a printer, scanner, keyboard, mouse, tape device, microphone, and external modem that are externally connected to the computer.

The following are some of the commonly used peripherals.

- **Keyboard:** The keyboard is the most commonly used input device. It is used to provide commands to the computer. The commands are usually in the form of text. The keyboard consists of many keys such as function keys, numeric keypad, character keys, and various symbols.
- **Monitor:** The most commonly used output device is the monitor. A cable connects the monitor to the video adapters in the computer's motherboard. These video adapters convert the electrical signals to the text and images that are displayed. The images on the monitor are made of thousands of pixels. The cursor is the characteristic feature of display devices. It marks the position on the screen where the next character will be inserted.
- **Printer:** Printers provide a permanent record of computer data or text on paper. We can classify printers as impact and non-impact printers. Impact printers print characters due to the physical contact of the print head with the paper. In non-impact printers, there is no physical contact.
- **Magnetic Tape:** Magnetic tapes are used in most companies to store data files. Magnetic tapes use a read-write mechanism. The read-write mechanism refers to writing data on or reading data from a magnetic tape. The tapes sequentially store the data manner. In this sequential processing, the computer must begin searching at the beginning and check each record until the desired data is available.
  Magnetic tape is the cheapest medium for storage because it can store a large number of binary digits, bytes, or frames on every inch of the tape. The advantages of using magnetic tape include unlimited storage, low cost, high data density, rapid transfer rate, portability, and ease of use.
- **Magnetic Disk:** There is another medium for storing data is magnetic disks. Magnetic disks have high-speed rotational surfaces coated with magnetic material. A read-write mechanism is used to achieve access to write on or read from the magnetic disk. Magnetic disks are generally used for the volume storage of programs and information.

  There are some peripheral devices found in computer systems including digital incremental plotters, optical and magnetic readers, analog to digital converters, and several data acquisition equipment.

19 Input - Output Interface

Input-Output Interface is used as an method which helps in transferring of information between the internal storage devices i.e. memory and the external peripheral device . A peripheral device is that which provide input and output for the computer, it is also called Input-Output devices. For Example: A keyboard and mouse provide Input to the computer are called input devices while a monitor and printer that provide output to the computer are called output devices. Just like the external hard-drives, there is also availability of some peripheral devices which are able to provide both input and output.

In micro-computer base system, the only purpose of peripheral devices is just to provide special communication links for the interfacing them with the CPU. To resolve the differences between peripheral devices and CPU, there is a special need for communication links.

The major differences are as follows:

- The nature of peripheral devices is electromagnetic and electro-mechanical. The nature of the CPU is electronic. There is a lot of difference in the mode of operation of both peripheral devices and CPU.
- There is also a synchronization mechanism because the data transfer rate of peripheral devices are slow than CPU.

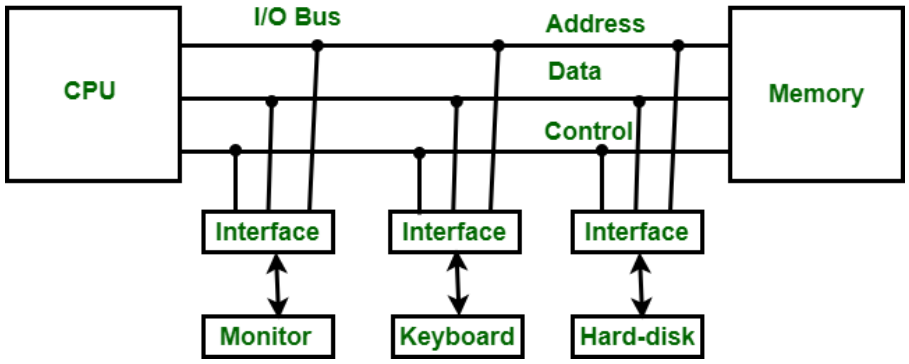


Figure 14: I/O Interface

- In peripheral devices, data code and formats are differ from the format in the CPU and memory.
- The operating mode of peripheral devices are different and each may be controlled so as not to disturb the operation of other peripheral devices connected to CPU.

There is a special need of the additional hardware to resolve the differences between CPU and peripheral devices to supervise and synchronize all input and output devices.

Functions of Input-Output Interface:

1. It is used to synchronize the operating speed of CPU with respect to input-output devices.
2. It selects the input-output device which is appropriate for the interpretation of the input-output device.
3. It is capable of providing signals like control and timing signals.
4. In this data buffering can be possible through data bus.
5. There are various error detectors.
6. It converts serial data into parallel data and vice-versa.
7. It also convert digital data into analog signal and vice-versa.

Types of Interfacing An interfacing circuit is designed for an I/O device to facilitate data transfer. Input and output devices are used to enter data as well as to take out data from microprocessor to peripheral. This interfacing is referred to as the mapping of I/O either by using peripheral devices, i.e. I/O or by memory.

When the processor, main memory and I/O share a common bus, two modes of addressing are possible:

1. Memory-Mapped I/O Interfacing : In this kind of interfacing, we assign a memory address that can be used in the same manner as we use a normal memory location.
2. I/O Mapped I/O Interfacing : A kind of interfacing in which we assign an 8-bit address value to the input/output devices which can be accessed using IN and OUT instruction is called I/O Mapped I/O Interfacing.

19.1 Memory mapped IO Interfacing

Memory mapped I/O is an interfacing technique in which memory related instructions are used for data transfer and the device is identified by a 16-bit address. In this type, the I/O devices are treated as memory locations. The control signals used are MEMR and MEMW. The interfacing between I/O and microprocessor will be same as single memory location. For data transfer between I/O device and microprocessor, microprocessor will send address, generate control signals MEMR and MEMW.

In MEMR, it accepts data from I/O device while in MEMW it transfers data to I/O device.

Memory mapped I/O scheme is shown in Figure 15.In Figure 15, the left diagram is for the Read control signal and right diagram is for the Write control signal.

The address is of 16 bits A0 to A15. Control signals RD and WR are used to control the buffer latch. Combination of address lines A0-A15 can be done by using NAND gates and then it is combined with RD or WR signal. For buffer when A0 to A15 are at logic 1, the output of NAND gate will be low.

With memory-mapped I/O, there is a single address space for memory locations and I/O devices. The processor treats the status and data registers of I/O modules as memory locations and uses the same machine instructions to access both memory and I/O devices.

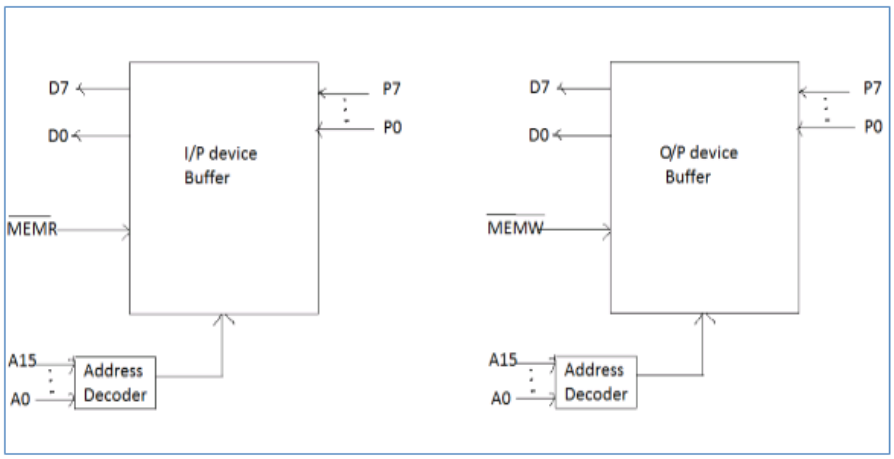


Figure 1

Figure 15: Memory Mapped IO

For example, with 10 address lines, a combined total of 210 1024 memory locations and I/O addresses can be supported, in any combination. With memory-mapped I/O, a single read line and a single write line are needed on the bus.

Advantage of memory-mapped I/O is that this large stock of instructions can be used, allowing more efficient programming.

A disadvantage is that valuable memory address space is used up.

In memory mapped I/O,

- I/O and memory, both in combination are treated as memory.
- Device address is of 16-bits.
- MEMR and MEMW control signals are used and Arithmetic and logical operations are performed.
- Memory related instructions are available such as LDA, STA.
- Data transfer takes place between any register and I/O.It consists of a total of 64K devices.
- Decoding 16-bits of address is required, therefore more hardware required.

Difference between Memory-Mapped I/O Interfacing and I/O Mapped I/O Interfacing :

Features	Memory mapped I/O	I/O mapped I/O
Addressing	IO devices are accessed like any other memory location	They cannot be accessed like any other memory location.
Address Size	They are assigned with 16-bit address values.	They are assigned with 8-bit address values.
Instructions Used	The instruction used are LDA and STA, etc.	The instruction used are IN and OUT.
Cycles	Cycles involved during operation are Memory Read, Memory Write.	Cycles involved during operation are IO read and IO writes in the case of IO Mapped IO.
Registers Communicating	Any register can communicate with the IO device in case of Memory Mapped IO.	Only Accumulator can communicate with IO devices in case of IO Mapped IO.
Control Signal	No separate control signal required since we have unified memory space in the case of Memory Mapped IO.	Special control signals are used in the case of IO Mapped IO.
Arithmetic and Logical operations	Arithmetic and logical operations are performed directly on the data in the case of Memory Mapped IO.	Arithmetic and logical operations cannot be performed directly on the data in the case of IO Mapped IO.

Table 3: Difference Between I/O mapped I/O and Memory mapped I/O

20 Modes of Transfer

The CPU executes the I/O instructions accepts the data temporarily but finally, the source/ destination will be any memory unit. There are various modes in which data transfer could take place between CPU I/O devices. Data transfer to from peripherals can be handled in one of the 3 given modes as follows.

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

20.1 Programmed I/O:

In program-controlled I/O, the processor program controls the complete data transfer. So only when an I/O transfer instruction is executed, the transfer could take place. It is required to check that device is ready/not for the data transfer in most cases. Usually, the transfer is to from a CPU register & peripheral. Here, CPU constantly monitors the peripheral. Here, until the I/O unit indicates that it is ready for transfer, the CPU wait & stays in a loop. It is time-consuming as it keeps the CPU busy needlessly.

20.2 Interrupt—Driven I/O:

To overcome the disadvantage of Programmed I/O,i.e., keeping CPU busy needlessly, Interrupt — Driven I/O is used. In this approach, when a peripheral sends an interrupt signal to the CPU whenever it is ready to transfer data. This indicates that the I/O data transfer is initiated by the external I/O device. The processor stops the execution of the current program & transfers the control to interrupt the service routine when interrupted. The interrupt service routine then performs the data transfer. After the completion of data transfer, it returns control to the main program to the point it was interrupted.

20.3 DMA — Direct Memory Access:

DMA transfer is used for large data transfers. Here, a memory bus is used by the interface to transfer data in out of a memory unit. The CPU provides starting address number of bytes to be transferred to the interface to initiate the transfer, after that it proceeds to execute other tasks. DMA requests a memory cycle through the memory bus when the transfer is made. DMA transfers the data directly into the memory when the request is granted by the memory controller. To allow direct memory transfer(I/O), the CPU delays its memory access operation. So, DMA allows I/O devices to directly access memory with less intervention of the CPU.

21 Interrupts

The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the Interrupt Service Routine (ISR).

When a device raises an interrupt at let's say process i, the processor first completes the execution of instruction i. Then it loads the Program Counter (PC) with the address of the first instruction of the ISR. Before loading the Program Counter with the address, the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt the processor can continue with process i+1.

While the processor is handling the interrupts, it must inform the device that its request has been recognized so that it stops sending the interrupt request signal. Also, saving the registers so that the interrupted process can be restored in the future, increases the delay between the time an interrupt is received and the start of the execution of the ISR. This is called Interrupt Latency.

Types of Interrupts: Although interrupts have highest priority than other signals, there are many type of interrupts but basic type of interrupts are

21.1 Hardware Interrupts:

If the signal for the processor is from external device or hardware is called hardware interrupts. In a hardware interrupt, all the devices are connected to the Interrupt Request Line. A single request line is used for all the n devices. To request an interrupt, a device closes its associated switch. When a device requests an interrupt, the value of INTR is the logical OR of the requests from individual devices.

Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are.

1. Maskable Interrupt: The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor.
2. Non Maskable Interrupt: The hardware which cannot be delayed and should process by the processor immediately.

21.2 Software Interrupts:

A sort of interrupt called a software interrupt is one that is produced by software or a system as opposed to hardware. Traps and exceptions are other names for software interruptions. They serve as a signal for the operating system or a system service to carry out a certain function or respond to an error condition.

A particular instruction known as a "interrupt instruction" is used to create software interrupts. When the interrupt instruction is used, the processor stops what it is doing and switches over to a particular interrupt handler code. The interrupt handler routine completes the required work or handles any errors before handing back control to the interrupted application.

Software interrupt can also divided in to two types. They are.

1. Normal Interrupts: the interrupts which are caused by the software instructions are called software instructions.
2. Exception: unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception.

Classification of Interrupts According to the Temporal Relationship with System Clock:

1. Synchronous Interrupt: The source of interrupt is in phase to the system clock is called synchronous interrupt. In other words interrupts which are dependent on the system clock. Example: timer service that uses the system clock.
2. Asynchronous Interrupts: If the interrupts are independent or not in phase to the system clock is called asynchronous interrupt.

The sequence of events involved in handling an IRQ:

- Devices raise an IRQ.
- The processor interrupts the program currently being executed.
- The device is informed that its request has been recognized and the device deactivates the request signal.
- The requested action is performed.
- An interrupt is enabled and the interrupted program is resumed.

Handling Multiple Devices: When more than one device raises an interrupt request signal, then additional information is needed to decide which device to be considered first. The following methods are used to decide which device to select: Polling, Vectored Interrupts, and Interrupt Nesting. These are explained as following below.

1. Polling: In polling, the first device encountered with the IRQ bit set is the device that is to be serviced first. Appropriate ISR is called to service the same. It is easy to implement but a lot of time is wasted by interrogating the IRQ bit of all devices.
2. Vectored Interrupts: In vectored interrupts, a device requesting an interrupt identifies itself directly by sending a special code to the processor over the bus. This enables the processor to identify the device that generated the interrupt. The special code can be the starting address of the ISR or where the ISR is located in memory and is called the interrupt vector.
3. Interrupt Nesting: In this method, the I/O device is organized in a priority structure for example Daisy chaining. Therefore, an interrupt request from a higher priority device is recognized whereas a request from a lower priority device is not. The processor accepts interrupts only from devices/processes having priority.

Processors' priority is encoded in a few bits of PS (Process Status register). It can be changed by program instructions that write into the PS. The processor is in supervised mode only while executing OS routines. It switches to user mode before executing application programs.

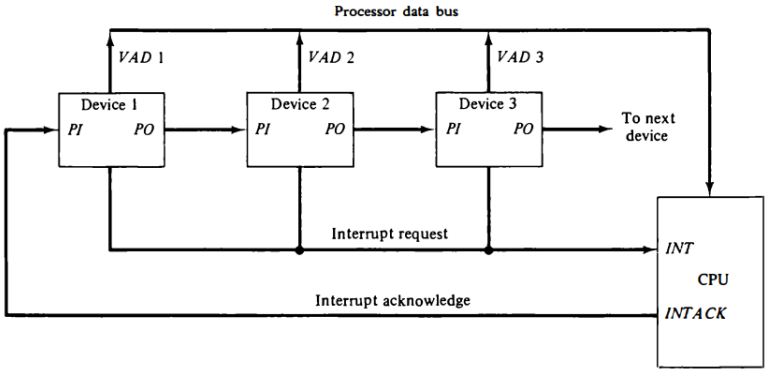


Figure 12 Daisy-chain priority interrupt.

Figure 16: Daisy Chaining Priority

21.3 Daisy-Chaining Priority

The daisy-chaining method of creating priority includes a serial connection of all devices that request an interrupt. The device with the highest priority is located in the first position, followed by lower-priority devices up to the device with the lowest priority, which is situated last in the chain. This technique of connection between three devices and the CPU.

The interrupt request line is average to all devices and design a wired logic connection. If some device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU. When no interrupts are pending, the interrupt line continues in the high-level state and no interrupts are identified by the CPU. This is similar to a negative logic OR operation.

The CPU responds to an interrupt request by enabling the interrupt to acknowledge the line. This signal is acknowledged by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.

If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by locating a 0 in the PO output. It then proceeds to insert its interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledged signal has been blocked. A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output.

If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output. Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus.

The daisy chain arrangement provides the highest priority to the device that receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position, the lower is its priority.

It displays the internal logic that should be included within each device when linked in the daisy-chaining scheme. The device sets its RF flip-flop when it needs to interrupt the CPU. The output of the RF flip-flop goes through an open-collector inverter, a circuit that supports the wired logic for the common interrupt line.

21.4 Difference between Synchronous and Asynchronous Transmission

Synchronous Transmission: In Synchronous Transmission, data is sent in form of blocks or frames. This transmission is the full-duplex type. Between sender and receiver, synchronization is compulsory. In Synchronous transmission, There is no gap present between data. It is more efficient and more reliable than asynchronous transmission to transfer a large amount of data.

Example: Chat Rooms, Telephonic Conversations, Video Conferencing

Asynchronous Transmission: In Asynchronous Transmission, data is sent in form of byte or character. This transmission is the half-duplex type transmission. In this transmission start bits and stop bits are added with data. It does not require synchronization. Example: Email, Forums, Letters.

Now, let's see the difference between Synchronous Transmission and Asynchronous Transmission:

22 Direct Access Media (DMA) Controller in Computer Architecture

Direct Memory Access (DMA) :

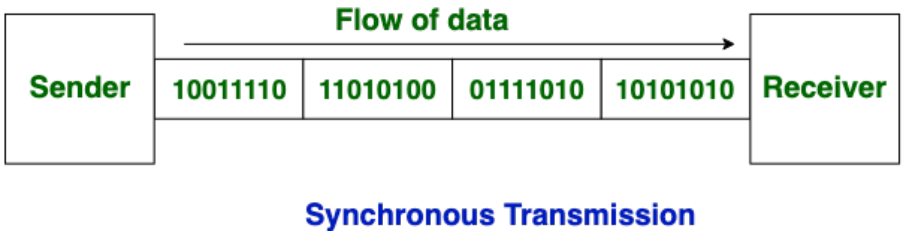


Figure 17: Synchronous Data Transfer

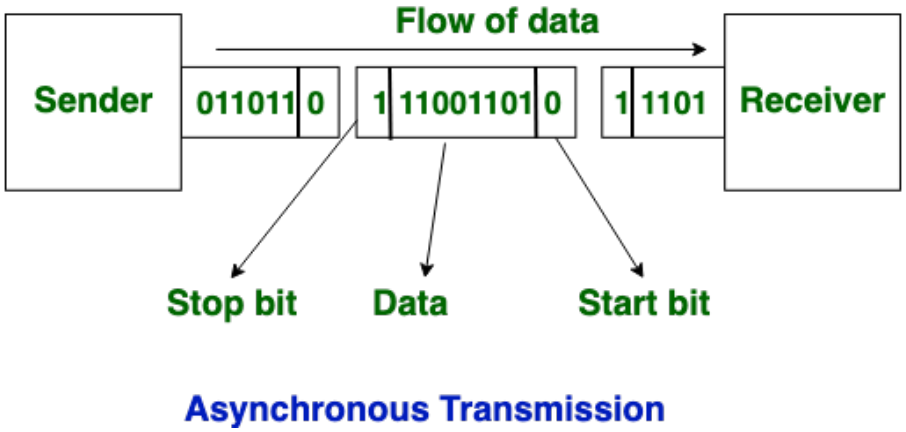


Figure 18: Asynchronous Data Transfer

S. No.	Synchronous Transmission	Asynchronous Transmission
1.	In Synchronous transmission, data is sent in form of blocks or frames.	In Asynchronous transmission, data is sent in form of bytes or characters.
2.	Synchronous transmission is fast.	Asynchronous transmission is slow.
3.	Synchronous transmission is costly.	Asynchronous transmission is economical.
4.	In Synchronous transmission, the time interval of transmission is constant.	In Asynchronous transmission, the time interval of transmission is not constant, it is random.
5.	In this transmission, users have to wait till the transmission is complete before getting a response back from the server.	Here, users do not have to wait for the completion of transmission in order to get a response from the server.
6.	In Synchronous transmission, there is no gap present between data.	In Asynchronous transmission, there is a gap present between data.
7.	Efficient use of transmission lines is done in synchronous transmission.	While in Asynchronous transmission, the transmission line remains empty during a gap in character transmission.
8.	The start and stop bits are not used in transmitting data.	The start and stop bits are used in transmitting data that imposes extra overhead.
9.	Synchronous transmission needs precisely synchronized clocks for the information of new bytes.	Asynchronous transmission does not need synchronized clocks as parity bit is used in this transmission for information of new bytes.

Table 4: Difference Between Synchronous and Asynchronous Data Transfer

DMA Controller is a hardware device that allows I/O devices to directly access memory with less participation of the processor. DMA controller needs the same old circuits of an interface to communicate with the CPU and Input/Output devices.

Figure 19 below shows the block diagram of the DMA controller. The unit communicates with the CPU through data bus and control lines. Through the use of the address bus and allowing the DMA and RS register to select inputs, the register within the DMA is chosen by the CPU. RD and WR are two-way inputs. When

BG (bus grant) input is 0, the CPU can communicate with DMA registers. When BG (bus grant) input is 1, the CPU has relinquished the buses and DMA can communicate directly with the memory.

DMA controller registers : The DMA controller has three registers as follows.

- Address register – It contains the address to specify the desired location in memory.
- Word count register – It contains the number of words to be transferred.
- Control register – It specifies the transfer mode.

Note – All registers in the DMA appear to the CPU as I/O interface registers. Therefore, the CPU can both read and write into the DMA registers under program control via the data bus.

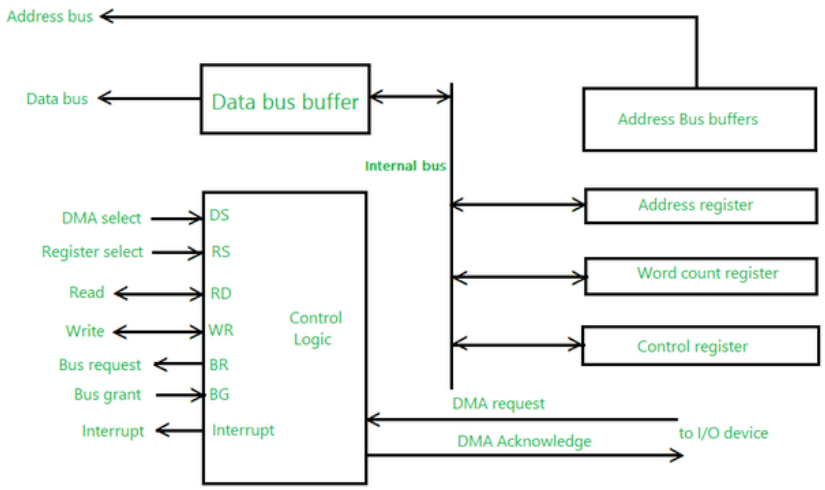


Figure 19: Block Diagram

Explanation :
The CPU initializes the DMA by sending the given information through the data bus.

- The starting address of the memory block where the data is available (to read) or where data are to be stored (to write).
- It also sends word count which is the number of words in the memory block to be read or write.
- Control to define the mode of transfer such as read or write.
- A control to begin the DMA transfer.

Unit-5: Pipeline and Vector Processing

23 Parallel Processing

For the purpose of increasing the computational speed of computer system, the term 'parallel processing' employed to give simultaneous data-processing operations is used to represent a large class. In addition, a parallel processing system is capable of concurrent data processing to achieve faster execution times.

As an example, the next instruction can be read from memory, while an instruction is being executed in ALU. The system can have two or more ALUs and be able to execute two or more instructions at the same time. In addition, two or more processing is also used to speed up computer processing capacity and increases with parallel processing, and with it, the cost of the system increases. But, technological development has reduced hardware costs to the point where parallel processing methods are economically possible.

Parallel processing derives from multiple levels of complexity. It is distinguished between parallel and serial operations by the type of registers used at the lowest level. Shift registers work one bit at a time in a serial fashion, while parallel registers work simultaneously with all bits of simultaneously with all bits of the word. At high levels of complexity, parallel processing derives from having a plurality of functional units that perform separate or similar operations simultaneously. By distributing data among several functional units, parallel processing is installed.

As an example, arithmetic, shift and logic operations can be divided into three units and operations are transformed into a teach unit under the supervision of a control unit.

One possible method of dividing the execution unit into eight functional units operating in parallel is shown in Figure 20. Depending on the operation specified by the instruction, operands in the registers are transferred to one of the units, associated with the operands. In each functional unit, the operation performed is denoted in each block of the diagram. The arithmetic operations with integer numbers are performed by the adder and integer multiplier.

Floating-point operations can be divided into three circuits operating in parallel. Logic, shift, and increment operations are performed concurrently on different data. All units are independent of each other, therefore one number is shifted while another number is being incremented. Generally, a multi-functional organization is associated with a complex control unit to coordinate all the activities between the several components.

The main advantage of parallel processing is that it provides better utilization of system resources by increasing resource multiplicity which overall system throughput.

24 Pipelining

Pipelining is a technique of breaking a sequential process into small fragments or sub-operations. The execution of each of these sub-procedure takes place in a certain dedicated segment that functions together with all other segments. The pipeline has a collection of processing segments which helps the flow of binary information.

The internal working in a pipeline is such that the outcome of one segment is conveyed to the next segment in the pipeline until the desired result is obtained. The outcome is acquired after the information is developed through all segments.

The term "pipeline" indicates that the flow of information takes place in parallel. Pipelining defines the temporal overlapping of processing. The overlapping of processing is done by relating a register with every segment in the pipeline. The registers help in providing isolation between each segment so that every segment can work on distinct data simultaneously.

A segment consists of an input register and a combinational circuit. The register stores the information and the combinational circuit operates in the specific segment. The output of the combinational circuit of a segment is sent to the input register of the next segment. To perform the activity in each segment, a clock is set for each register.

When we consider a sequential execution of three instructions, each having three stages of execution, the sequence got is the following instruction cycle.

If each stage requires one unit time and a separate unit for each action, then the total time taken would be nine units. In the case of pipelined execution of the same instruction set, the sequence would require only five units.

The Figure 21 shows the comparison between parallel processing and sequential processing.

As observed in the Figure 21, we can save approximately 50% of the execution time by using pipelining.

The pipeline organization will be indicated using a simple example. Suppose that we need to execute the combined multiply and add operations with a cascade of numbers.

$A_i * B_i + C_i$ for i = 1,2,3 ... 7

Each sub-operation is executed in a segment inside a pipeline. The separation between each segment is provided by registers for each segment to work on different data simultaneously.

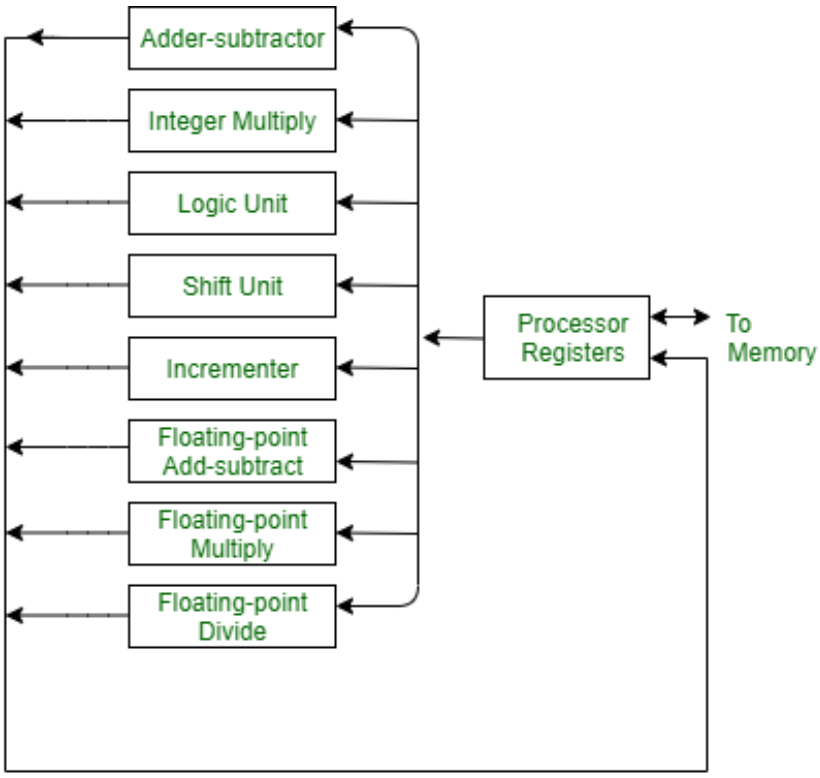


Figure - Processor with Multiple Functional Units

Figure 20: Processor with Multiple Functional Unit

Each segment uses one or three registers with combinational circuits. R1 through R7 are registers that get new information with each clock pulse. The combinational circuits are multiplier and adder.

R1 $\leftarrow$ A_i, R1 $\leftarrow$ B_i Input A_i and B_i

R3 $\leftarrow$ R1 * R2, R4 $\leftarrow$ C Multiply and Input C_i

R5 $\leftarrow$ R3 + R4 Add C_i to product

25 Arithmetic Pipeline and Instruction Pipeline

25.1 Arithmetic Pipeline :

An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations. The process or flowchart arithmetic pipeline for floating point addition is shown in the Figure **??**.

Let's understand 'Arithmetic Pipeline' with an example of addition of two floating point numbers.

We know that two floating point numbers are represented in their normalized form using mantissa and exponents. Mantissa represents the precision of the number and exponent represents the range.

Let the f1 and f2 be two floating point numbers whose sum is to be calculated.

$$f1 = x1 + 2^{y^1}$$

$$f2 = x2 + 2^{y^2}$$

Here x1 and x2 are mantissa of f1 and f2 respectively and y^1 and y^2 are exponents of f1 and f2 respectively. In addition of two floating point numbers, following steps are used:

- First we compare the exponents of the two numbers by taking their difference to choose the larger exponent as the exponent of the result.
- Using the difference of the exponent, the mantissa of the smaller one is shifted.

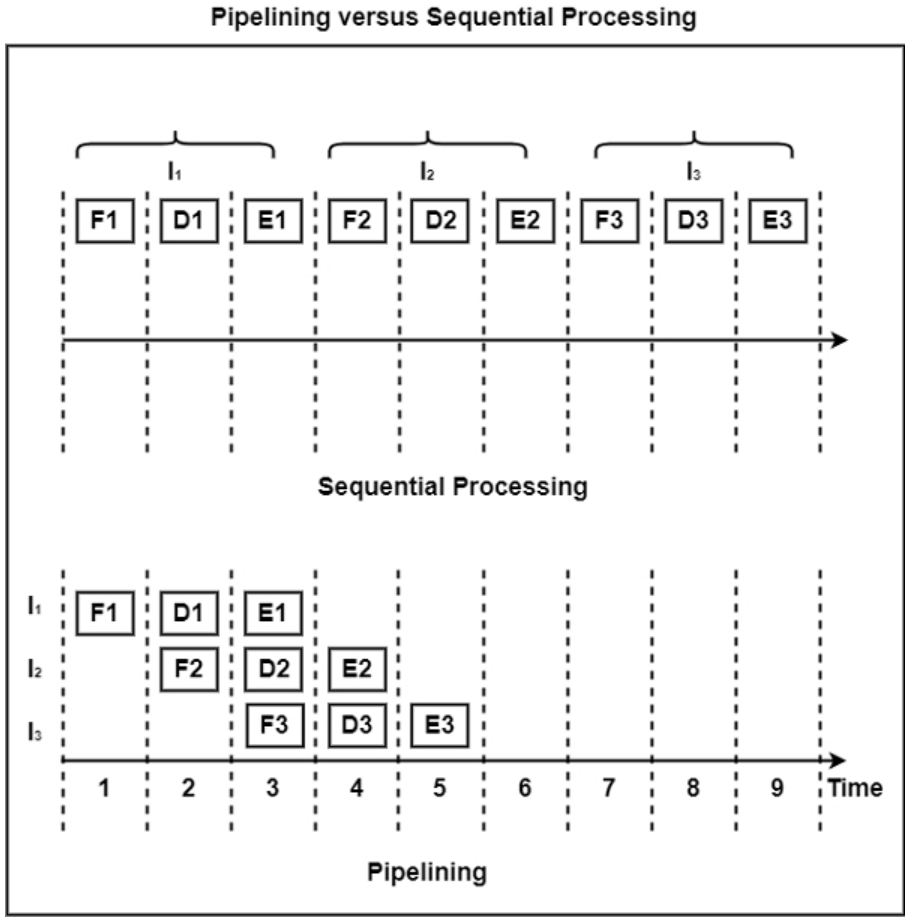


Figure 21: Sequential Processing Vs Pipelining

- After aligning the mantissa, the mantissas of the two numbers are added.
- Finally we normalize the result obtained in the previous step.
- In arithmetic pipeline, these four steps are performed in four different segments to improve the speed and throughput of the system.

The Figure 23 below shows the steps performed in four different segments in an arithmetic pipeline for the addition of two floating point numbers:

Example: Let us consider two numbers,

$$X = 0.3214 * 10^3 \; and \; Y = 0.4500 * 10^2$$

Explanation: First of all the two exponents are subtracted to give $3 - 2 = 1$. Thus 3 becomes the exponent of result and the smaller exponent is shifted 1 times to the right to give

$$Y = 0.0450 * 10^3$$

Finally the two numbers are added to produce

$$Z = 0.3664 * 10^3$$

As the result is already normalized the result remains the same.

25.2 Instruction Pipeline :

In this a stream of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle. This type of technique is used to increase the throughput of the computer system. An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration.

In the most general case computer needs to process each instruction in following sequence of steps:

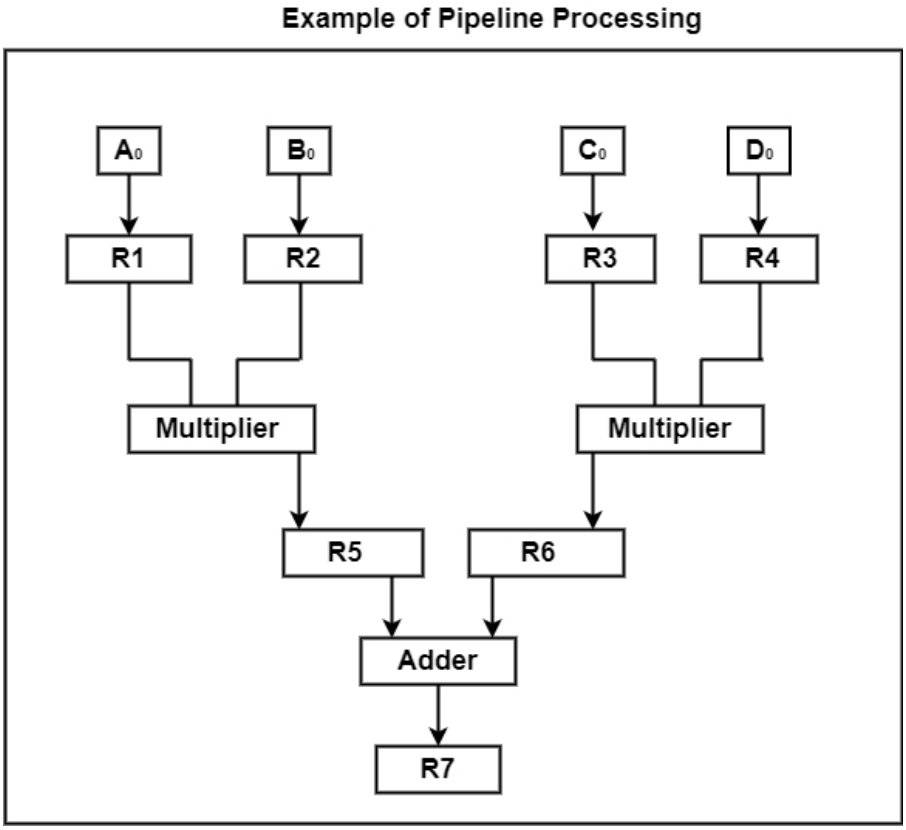


Figure 22: Pipelining

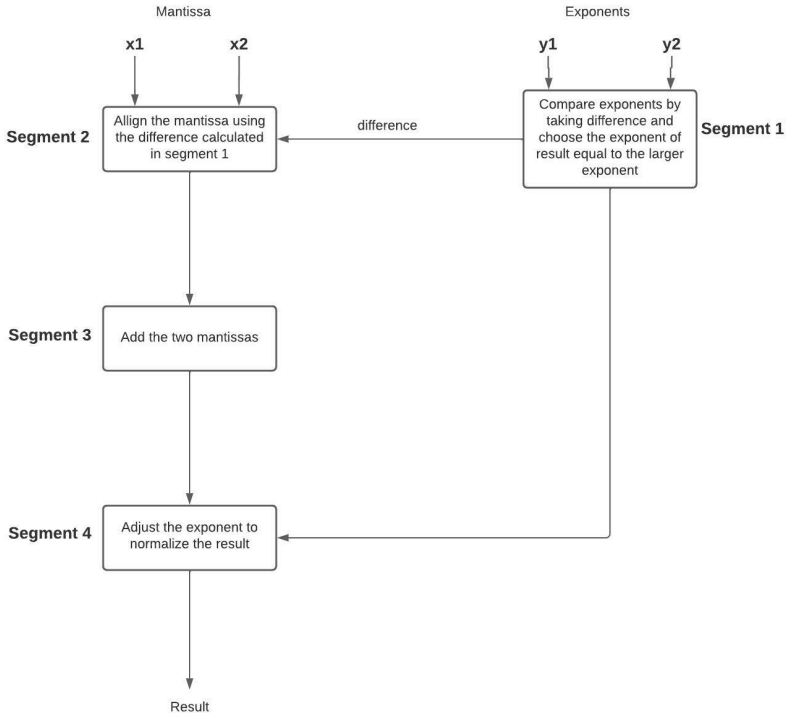


Figure 23: Floating Point Addition

1. Fetch the instruction from memory (FI)
2. Decode the instruction (DA)
3. Calculate the effective address

4. Fetch the operands from memory (FO)
5. Execute the instruction (EX)
6. Store the result in the proper place

The flowchart for instruction pipeline is shown below.

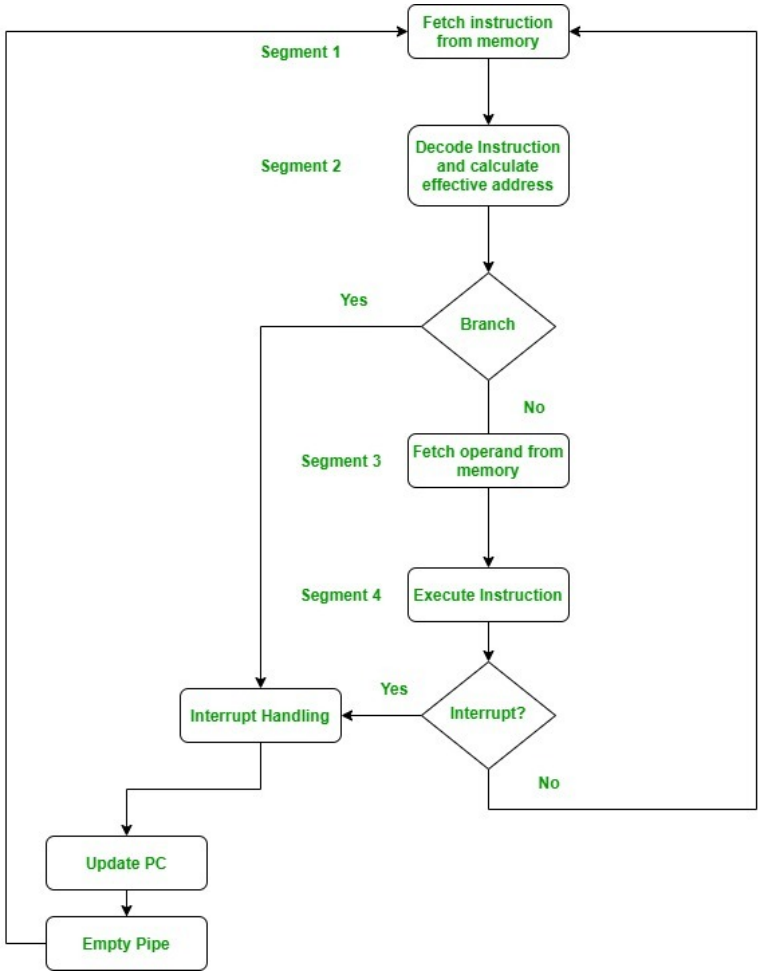


Figure 24: Instruction Pipelining

Let us see an example of instruction pipeline.

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
3			FI	DA	FO	EX							
4				FI	---	---	FI	DA	FO	EX			
5								FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Instruction
Branch

Figure 25: Example: Instruction Pipelining

Here the instruction is fetched on first clock cycle in segment 1. Now it is decoded in next clock cycle, then operands are fetched and finally the instruction is executed. We can see that here the fetch and decode phase overlap due to pipelining. By the time the first instruction is being decoded, next instruction is fetched by the pipeline.

In case of third instruction we see that it is a branched instruction. Here when it is being decoded 4th instruction is fetched simultaneously. But as it is a branched instruction it may point to some other instruction when it is decoded. Thus fourth instruction is kept on hold until the branched instruction is executed. When it gets executed then the fourth instruction is copied back and the other phases continue as usual.

Advantages of Pipelining

- Instruction throughput increases.
- Increase in the number of pipeline stages increases the number of instructions executed simultaneously.
- Faster ALU can be designed when pipelining is used.
- Pipelined CPU's works at higher clock frequencies than the RAM.
- Pipelining increases the overall performance of the CPU.

Disadvantages of Pipelining

- Designing of the pipelined processor is complex.
- Instruction latency increases in pipelined processors.
- The throughput of a pipelined processor is difficult to predict.
- The longer the pipeline, worse the problem of hazard for branch instructions.

25.3 Instruction Pipelining Hazards

The instruction pipelining hazards is a condition where the execution of the pipelined instructions is stopped or delayed, it is also called a pipeline bubble. There are three kinds of instruction pipeline hazards.

Resource Hazards

When two pipelined instructions or even more, want to access the same resource it results in resource hazards. It is also termed structural hazards. A solution to this hazard is that these instructions must be executed serially up to some portion of the pipeline.

Let us understand this with the help of an example. Consider the main memory you have has a single port that restricts the processor to perform instruction fetch, read data and write data one at a time. This means you cannot perform the operand read & write operation, from memory in parallel with instruction fetch.

Now consider that there are three instructions in the pipeline as shown in the Figure 26. So in the normal conditions, the operand fetch of instruction 1 must-have overlapped the instruction fetch of instruction 3. But there is a case that source operand of instruction 1 is present in main memory instead of register and the rest of all the operands are in register. So we would halt the instruction fetch of instruction 3 for one clock cycle. Because instruction 1 will fetch its source operand from memory so in the same clock cycle instruction 3 cannot perform instruction fetch in parallel to operand fetch of instruction 1.

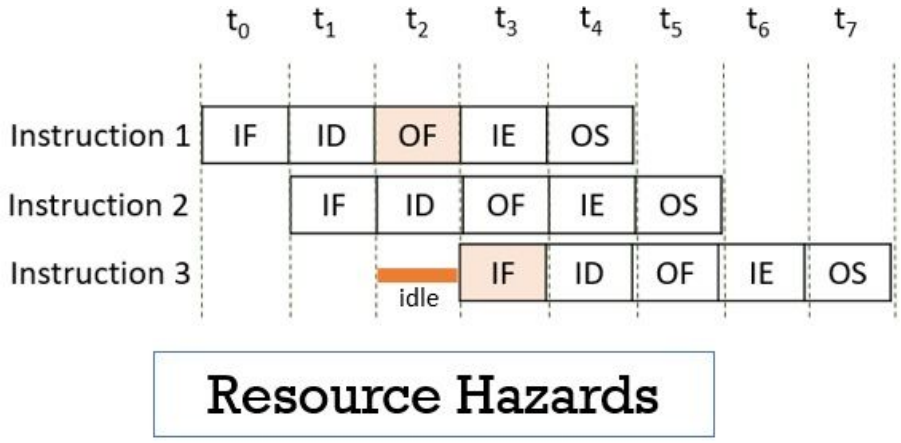


Figure 26: Caption

25.4 Data Hazards

The data hazard is a condition when accessing an operand location creates conflict. Consider that you have two instructions:

ADD R1, R2, R3

SUB A, R1

Observe the instructions above the result of add instruction is stored in register R1 after the execution. This R1 act as an operand for subtract instruction. Now in the Figure 28 below notice that the register R1 is

updated with the add result in clock cycle t4. But the subtract instruction need its operand R1 at the t3 clock cycle. But if subtract instruction fetches the operand R1 in the t3 clock cycle then it will generate an incorrect result.

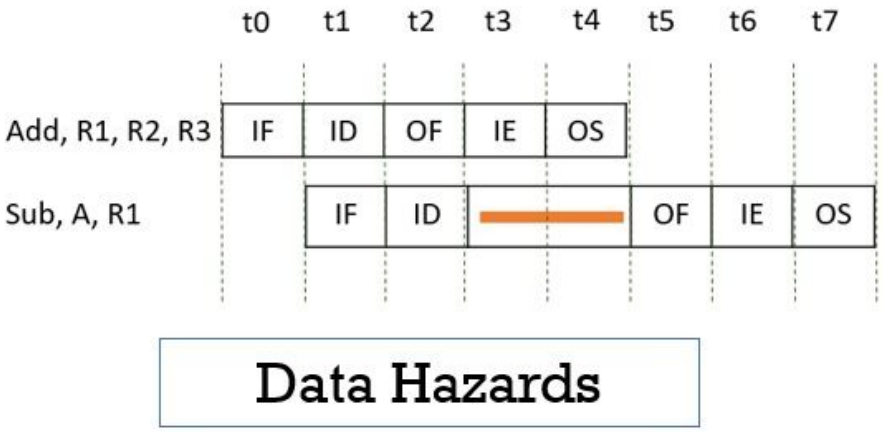


Figure 27: Data Hazards

So the subtract instruction must stall or halt for two clock cycles because for the correct result the subtract instruction is dependent on the result of add instruction. This dependency is also called data dependency.

25.5 Control Hazards

Control hazards occur when instruction pipelining fails in predicting branches in the instruction. Let us understand this with the help of an example. Consider the first instruction I1 in the pipeline is a branch instruction that targets instruction I6.

The instruction I1 is fetched in cycle t0, decoded in cycle t1, fetch operands in cycle t2 and perform execution in cycle t3 where the target address is computed. But till then three instructions I2, I3 and I4 are pipelined in cycle t1, t2 and t3 which must be discarded as instruction I1 is branch instruction which will compel the processor to execute the instruction I9 after instruction I1.

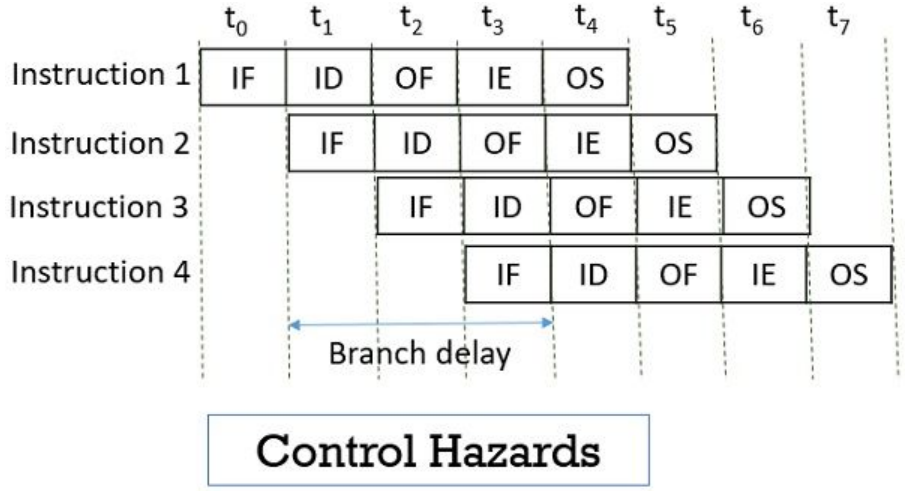


Figure 28: Control Hazards

This is how control hazards lead to delay of three cycles t1, t2 and t3 between I1 and I 6 which is also termed as branch delay. This branch delay could be minimized if the branches in the instruction could be predicted at the decoding stage only.

25.6 Problems on pipelining

Question - Consider a pipeline having 4 phases with duration 60, 50, 90 and 80 ns. Given latch delay is 10 ns. Calculate-

- Pipeline cycle time
- Non-pipeline execution time
- Speed up ratio
- Pipeline time for 1000 tasks
- Sequential time for 1000 tasks
- Throughput

Solution

Given information

Four stage pipeline is used

Delay of stages = 60, 50, 90 and 80 ns

Latch delay or delay due to each register = 10 ns

Part-01: Pipeline Cycle Time

Cycle time

= Maximum delay due to any stage + Delay due to its register

= Max 60, 50, 90, 80 + 10 ns

= 90 ns + 10 ns

= 100 ns

Part-02: Non-Pipeline Execution Time

Non-pipeline execution time for one instruction

= 60 ns + 50 ns + 90 ns + 80 ns

= 280 ns

Part-03: Speed Up Ratio:

Speed up

= Non-pipeline execution time / Pipeline execution time

= 280 ns / Cycle time

= 280 ns / 100 ns

= 2.8

Part-04: Pipeline Time For 1000 Tasks:

Pipeline time for 1000 tasks

= Time taken for 1st task + Time taken for remaining 999 tasks

= 1 x 4 clock cycles + 999 x 1 clock cycle

= 4 x cycle time + 999 x cycle time

= 4 x 100 ns + 999 x 100 ns

= 400 ns + 99900 ns

= 100300 ns

Part-05: Sequential Time For 1000 Tasks:

Non-pipeline time for 1000 tasks

= 1000 x Time taken for one task

= 1000 x 280 ns

= 280000 ns

Part-06: Throughput-

Throughput for pipelined execution

= Number of instructions executed per unit time

= 1000 tasks / 100300 ns

25.7 Difference between Synchronous and Asynchronous Pipeline

26 Non-linear Pipeline

A dynamic pipeline can be reconfigured to perform variable functions at different times. As shown in the Figure 29 it allows feed-forward and feedback connections in addition to the streamline connection.

Reservation Table: Displays the time space flow of data through the pipeline for one function evaluation.

Latency: The number of time units (clock cycles) between two initiations of a pipeline is the latency between them. Latency values must be non-negative integers.

Collision: When two or more initiations are done at same pipeline stage at the same time will cause a collision. A collision implies resource conflicts between two initiations in the pipeline, so it should be avoided.

Asynchronous Model	Synchronous Model
Data flow between adjacent stages is controlled by handshaking protocol	Clocked latches are used to interface between stages
Different amount of delay may be experienced in different stages	Approximately equal amount of delay is experienced in all stages
Ready and Acknowledgement signals are used for communication purpose	No such signals are used for communication purpose.
Transfer of data to various stages in not simultaneous.	All Latches Transfer of data to next stage simultaneous.
No Concept of Combinational Circuit is used in pipeline stages.	Pipeline stages are combinational logic circuits.

Table 5: Difference between Synchronous and Asynchronous Pipeline

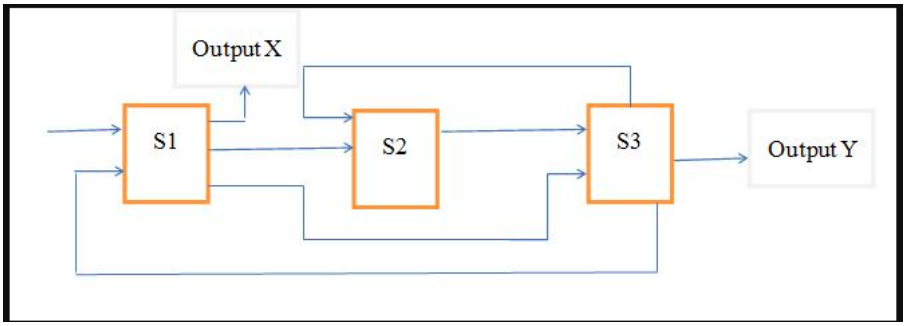


Figure 29: Non-linear Pipeline Stages

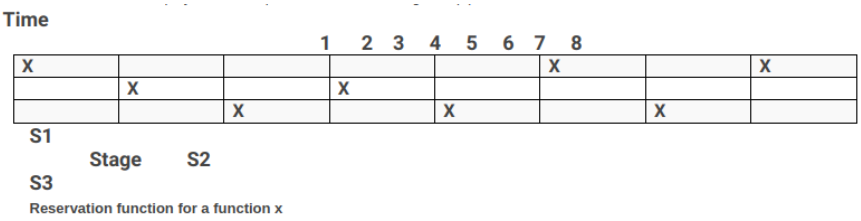


Figure 30: Reservation Table

Forbidden and Permissible Latency: Latencies that cause collisions are called forbidden latencies. (E.g. in above reservation table 2, 4, 5 and 7 are forbidden latencies).

Latencies that do not cause any collision are called **permissible latencies.** (E.g. in above reservation table 1, 3 and 6 are permissible latencies).

Latency Sequence and Latency Cycle: A Latency Sequence is a sequence of permissible non-forbidden latencies between successive task initiations.

A Latency cycle is a latency sequence which repeats the same subsequence (cycle) indefinitely.

The Average Latency of a latency cycle is obtained by dividing the sum of all latencies by the number of latencies along the cycle.

The latency cycle (1, 8) has an average latency of (1+8)/2=4.5.

A Constant Cycle is a latency cycle which contains only one latency value. (E.g. Cycles (3) and (6) both are constant cycle).

Collision Vector: The combined set of permissible and forbidden latencies can be easily displayed by a collision vector, which is an m-bit (m¡=n-1 in a n column reservation table) binary vector C=(CmCm-1....C2C1). The value of Ci=1 if latency I causes a collision and Ci=0 if latency i is permissible. (E.g. Cx= (1011010)).

State Diagram: Specifies the permissible state transitions among successive initiations based on the collision vector.

The minimum latency edges in the state diagram are marked with asterisk.

Simple Cycle, Greedy Cycle and MAL:

A Simple Cycle is a latency cycle in which each state appears only once. In above state diagram only (3), (6), (8), (1, 8), (3, 8), and (6, 8) are simple cycles. The cycle(1, 8, 6, 8) is not simple as it travels twice through state (1011010).

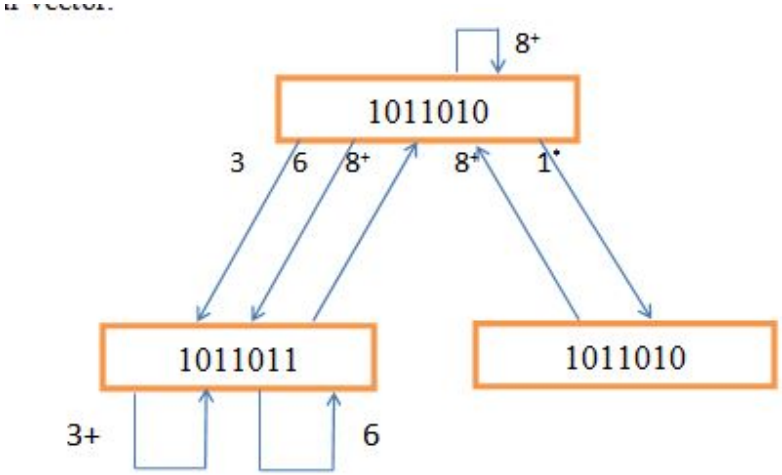


Figure 31: State-Diagram

A Greedy Cycle is a simple cycle whose edges are all made with minimum latencies from their respective starting states. The cycle (1, 8) and (3) are greedy cycles.

MAL (Minimum Average Latency) is the minimum average latency obtained from the greedy cycle. In greedy cycles (1, 8) and (3), the cycle (3) leads to MAL value 3.

26.1 Difference Between Linear and Non-Linear pipeline:

Linear Pipeline	Non-Linear Pipeline
Linear pipeline are static pipeline because they are used to perform fixed functions.	Non-Linear pipeline are dynamic pipeline because they can be reconfigured to perform variable functions at different times.
Linear pipeline allows only streamline connections.	Non-Linear pipeline allows feed-forward and feedback connections in addition to the streamline connection.
It is relatively easy to partition a given function into a sequence of linearly ordered sub functions.	Function partitioning is relatively difficult because the pipeline stages are interconnected with loops in addition to streamline connections.
The Output of the pipeline is produced from the last stage.	The Output of the pipeline is not necessarily produced from the last stage.
The reservation table is trivial in the sense that data flows in linear streamline.	The reservation table is non-trivial in the sense that there is no linear streamline for data flows.
Static pipelining is specified by single Reservation table.	Dynamic pipelining is specified by more than one Reservation table.
All initiations to a static pipeline use the same reservation table.	A dynamic pipeline may allow different initiations to follow a mix of reservation tables.

Table 6: Linear Vs Non-linear Pipeline

26.2 RISC Pipeline

RISC stands for Reduced Instruction Set Computers. It was introduced to execute as fast as one instruction per clock cycle. This RISC pipeline helps to simplify the computer architecture's design.

It relates to what is known as the Semantic Gap, that is, the difference between the operations provided in the high-level languages (HLLs) and those provided in computer architectures.

To avoid these consequences, the conventional response of the computer architects is to add layers of complexity to newer architectures. This also increases the number and complexity of instructions together with an increase in the number of addressing modes. The architecture which resulted from the adoption of this "add more complexity" are known as Complex Instruction Set Computers (CISC).

The main benefit of RISC to implement instructions at the cost of one per clock cycle is continually not applicable because each instruction cannot be fetched from memory and implemented in one clock cycle correctly under all circumstances.

The method to obtain the implementation of an instruction per clock cycle is to initiate each instruction with each clock cycle and to pipeline the processor to manage the objective of single-cycle instruction execution.

RISC compiler gives support to translate the high-level language program into a machine language program. There are various issues in managing complexity about data conflicts and branch penalties are taken care of by the RISC processors, which depends on the adaptability of the compiler to identify and reduce the delays encountered with these issues.

26.3 Principles of RISCs Pipeline

There are various principles of RISCs pipeline which are as follows.

- Keep the most frequently accessed operands in CPU registers.
- It can minimize the register-to-memory operations.
- It can use a high number of registers to enhance operand referencing and decrease the processor memory traffic.
- It can optimize the design of instruction pipelines such that minimum compiler code generation can be achieved.
- It can use a simplified instruction set and leave out those complex and unnecessary instructions.

Let us consider a three-segment instruction pipeline that shows how a compiler can optimize the machine language program to compensate for pipeline conflicts.

A frequent collection of instructions for a RISC processor is of three types are as follows.

- **Data Manipulation Instructions:** Manage the data in processor registers.
- **Data Transfer Instructions:** These are load and store instructions that use an effective address that is obtained by adding the contents of two registers or a register and a displacement constant provided in the instruction.
- **Program Control Instructions:** These instructions use register values and a constant to evaluate the branch address, which is transferred to a register or the program counter (PC).

26.4 Vector Processing

Vector processing is a central processing unit that can perform the complete vector input in individual instruction. It is a complete unit of hardware resources that implements a sequential set of similar data elements in the memory using individual instruction.

The scientific and research computations involve many computations which require extensive and high-power computers. These computations when run in a conventional computer may take days or weeks to complete. The science and engineering problems can be specified in methods of vectors and matrices using vector processing.

Features of Vector Processing There are various features of Vector Processing which are as follows.

- A vector is a structured set of elements. The elements in a vector are scalar quantities. A vector operand includes an ordered set of n elements, where n is known as the length of the vector.
- Each clock period processes two successive pairs of elements. During one single clock period, the dual vector pipes and the dual sets of vector functional units allow the processing of two pairs of elements.
- As the completion of each pair of operations takes place, the results are delivered to appropriate elements of the result register. The operation continues just before the various elements processed are similar to the count particularized by the vector length register.
- In parallel vector processing, more than two results are generated per clock cycle. The parallel vector operations are automatically started under the following two circumstances.
  - When successive vector instructions facilitate different functional units and multiple vector registers.
  - When successive vector instructions use the resulting flow from one vector register as the operand of another operation utilizing a different functional unit. This phase is known as chaining.

- A vector processor implements better with higher vectors because of the foundation delay in a pipeline.
- Vector processing decrease the overhead related to maintenance of the loop-control variables which creates it more efficient than scalar processing.

26.5 Array Processor

A processor which is used to perform different computations on a huge array of data is called an array processor. The other terms used for this processor are vector processors or multiprocessors. This processor performs only single instruction at a time on an array of data. These processors work with huge data sets to execute computations. So, they are mainly used for enhancing the performance of computers.

Array Processor performs computations on large array of data. These are two types of Array Processors: Attached Array Processor, and SIMD Array Processor. These are explained as following below.

Attached Array Processor : To improve the performance of the host computer in numerical computational tasks auxiliary processor is attached to it.

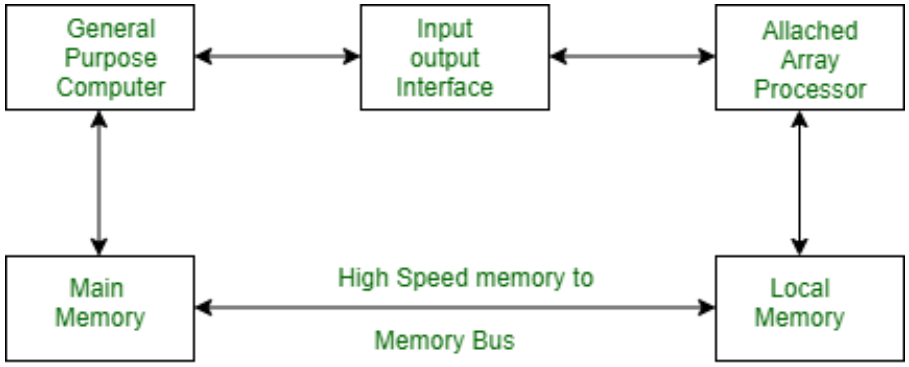


Figure - Interconnection of an attached array Processor to a host computer

Figure 32: Array Processor

Attached array processor has two interfaces:

1. Input output interface to a common processor.
2. Interface with a local memory.

Here local memory interconnects main memory. Host computer is general purpose computer. Attached processor is back end machine driven by the host computer.

The array processor is connected through an I/O controller to the computer the computer treats it as an external interface.

26.6 SIMD array processor :

This is computer with multiple process unit operating in parallel Both types of array processors, manipulate vectors but their internal organization is different.

SIMD is a computer with multiple processing units operating in parallel.

The processing units are synchronized to perform the same operation under the control of a common control unit. Thus providing a single instruction stream, multiple data stream (SIMD) organization. As shown in Figure 33, SIMD contains a set of identical processing elements (PES) each having a local memory M.

Each PE includes ALU, Floating point arithmetic unit, Working registers. Master control unit controls the operation in the PEs. The function of master control unit is to decode the instruction and determine how the instruction to be executed. If the instruction is scalar or program control instruction then it is directly executed within the master control unit.

Main memory is used for storage of the program while each PE uses operands stored in its local memory.

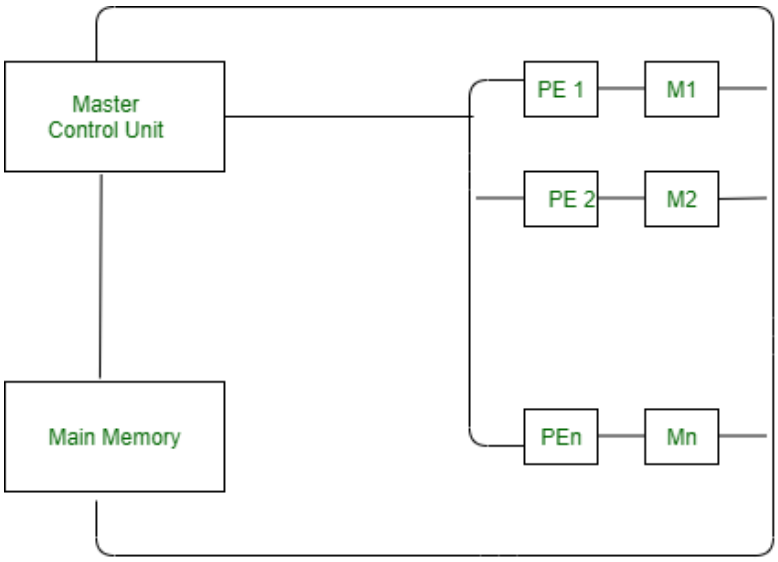


Figure - SIMD Array Processor Organization

Figure 33: SIMD Processor

Unit-II: Fixed Point and Floating Point Number Representations

Digital Computers use Binary number system to represent all types of information inside the computers. Alphanumeric characters are represented using binary bits (i.e., 0 and 1). Digital representations are easier to design, storage is easy, accuracy and precision are greater.

There are various types of number representation techniques for digital number representation, for example: Binary number system, octal number system, decimal number system, and hexadecimal number system etc. But Binary number system is most relevant and popular for representing numbers in digital computer system.

Storing Real Number These are structures as following below.

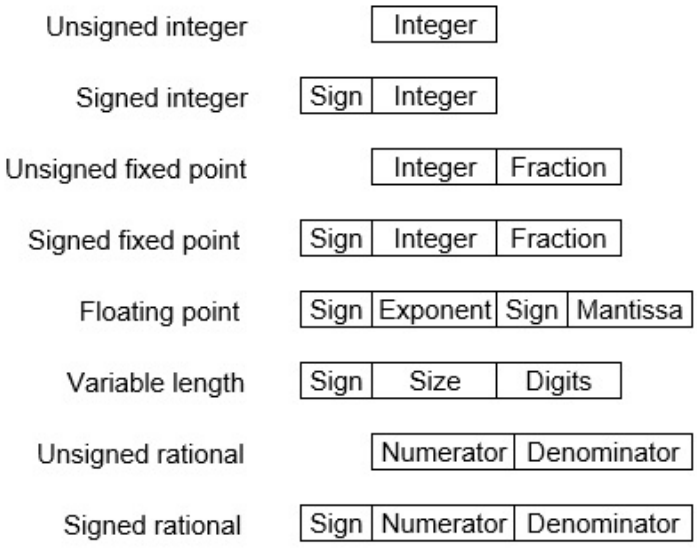


Figure 34: Data Types for storing real numbers

There are two major approaches to store real numbers (i.e., numbers with fractional component) in modern computing. These are (i) Fixed Point Notation and (ii) Floating Point Notation. In fixed point notation, there are a fixed number of digits after the decimal point, whereas floating point number allows for a varying number of digits after the decimal point.

27 Unit-II Fixed point and Floating Point Representations

Fixed-Point Representation:

This representation has fixed number of bits for integer part and for fractional part. For example, if given fixed-point representation is IIII.FFFF, then you can store minimum value is 0000.0001 and maximum value is 9999.9999. There are three parts of a fixed-point number representation: the sign field, integer field, and fractional field.

Figure 35: fixed-point numbers

We can represent these numbers using:

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.

2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

Example: Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:

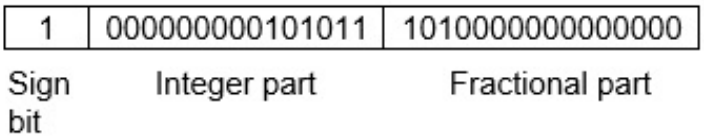


Figure 36: fixed-point representation

Where, 0 is used to represent + and 1 is used to represent. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.

The advantage of using a fixed-point representation is performance and disadvantage is relatively limited range of values that they can represent. So, it is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy. A number whose representation exceeds 32 bits would have to be stored inexactly.

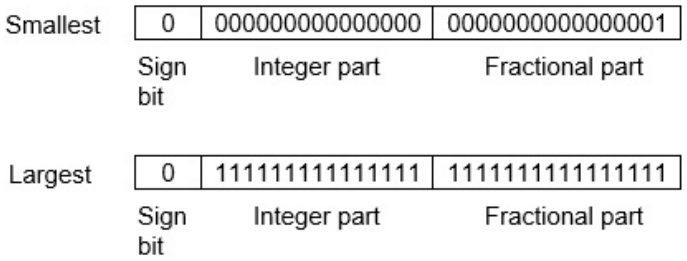


Figure 37: fixed-point representation

These are above smallest positive number and largest positive number which can be store in 32-bit representation as given above format. Therefore, the smallest positive number is $2^{-16} = 0.000015$ approximate and the largest positive number is $(2^{15}-1)+(1-2^{-16}) = 2^{15}(1-2^{-16}) = 32768$, and gap between these numbers is 2^{-16}.

We can move the radix point either left or right with the help of only integer field is 1.

27.1 Floating-Point Representation

This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).

The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: Mxr^e.

Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.

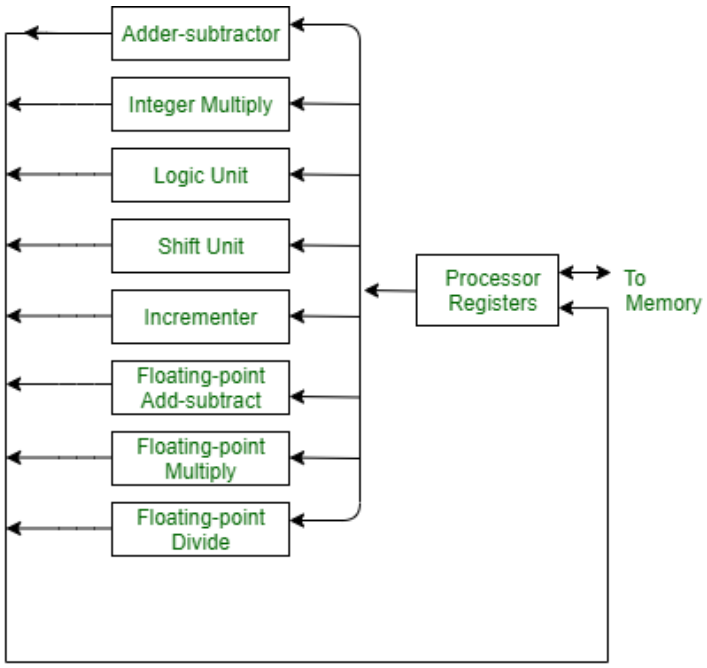


Figure - Processor with Multiple Functional Units

Figure 38: floating-point representation

So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where s is the sign bit, m is the mantissa, e is the exponent value, and Bias is the bias number.

Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.

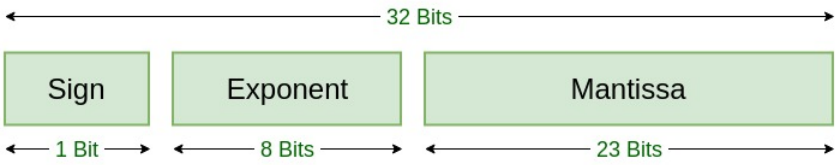
The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $(1.b_1b_2b_3...)_2 * 2^n$ This is normalized form of a number x.

27.2 IEEE Standard 754 Floating Point Numbers

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.

There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:

- The Sign of Mantissa – This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
- The Biased exponent – The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.



Single Precision
IEEE 754 Floating-Point Standard

Figure 39: IEEE Standard 754 Floating Point Numbers

- The Normalised Mantissa – The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

As shown in Figure 39 IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.

Explain the working of IEEE 754 Standard floating point Addition & Subtraction

- The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE).
- The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.
- There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:
  1. The Sign of Mantissa – This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
  2. The Biased exponent – The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
  3. The Normalised Mantissa – The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

IEEE 754 standard floating point Addition Algorithm

Brief overview of floating point addition algorithm have been explained below $X3 = X1+X2$ $X3 = (M1x2E1)+/-(M2x2E2)$

- X1 and X2 can only be added if the exponents are the same i.e E1=E2.
- We assume that X1 has the larger absolute value of the 2 numbers. Absolute value of X1 should be greater than absolute value of X2, else swap the values such that Abs(X1) is greater than Abs(X2). Abs(X1) ¿ Abs(X2).
- Initial value of the exponent should be the larger of the 2 numbers, since we know exponent of X1 will be bigger , hence Initial exponent result E3 = E1.
- Calculate the exponent's difference i.e. Exp_diff = (E1-E2).
- Left shift the decimal point of mantissa (M2) by the exponent difference. Now the exponents of both X1 and X2 are same.
- Compute the sum/difference of the mantissas depending on the sign bit S1 and S2.
  1. If signs of X1 and X2 are equal ($S1 == S2$) then add the mantissas.
  2. If signs of X1 and X2 are not equal ($S1! = S2$) then subtract the mantissas

- Normalize the resultant mantissa (M3) if needed. (1.m3 format) and the initial exponent result $E3 = E1$ needs to be adjusted according to the normalization of mantissa.
- If any of the operands is infinity or if ($E3 > Emax$), overflow has occurred ,the output should be set to infinity. If ($E3 < Emin$) then it's a underflow and the output should be set to zero.
- Nan's are not supported.

28 Non-Restoring Division Algorithm for Unsigned Integer

Instead of the quotient digit set 0, 1, the set -1, 1 is used by the non-restoring division. The non-restoring division algorithm is more complex as compared to the restoring division algorithm. But when we implement this algorithm in hardware, it has an advantage, i.e., it contains only one decision and addition/subtraction per quotient bit. After performing the subtraction operation, there will not be any restoring steps. Due to this, the numbers of operations basically cut down up to half. Because of the less operation, the execution of this algorithm will be fast. This algorithm basically performs simple operations such as addition, subtraction. In this method, we will use the sign bit of register A. 0 is the starting value/bit of register A.

Now we will learn steps of the non-restoring division algorithm, which are described as follows:

Step 1: In this step, the corresponding value will be initialized to the registers, i.e., register A will contain value 0, register M will contain Divisor, register Q will contain Dividend, and N is used to specify the number of bits in dividend.

Step 2: In this step, we will check the sign bit of A.

Step 3: If this bit of register A is 1, then shift the value of AQ through left, and perform A = A + M. If this bit is 0, then shift the value of AQ into left and perform A = A - M. That means in case of 0, the 2's complement of M is added into register A, and the result is stored into A.

Step 4: Now, we will check the sign bit of A again.

Step 5: If this bit of register A is 1, then Q[0] will become 0. If this bit is 0, then Q[0] will become 1. Here Q[0] indicates the least significant bit of Q.

Step 6: After that, the value of N will be decremented. Here N is used as a counter.

Step 7: If the value of N = 0, then we will go to the next step. Otherwise, we have to again go to step 2.

Step 8: We will perform A = A + M if the sign bit of register A is 1.

Step 9: This is the last step. In this step, register A contains the remainder, and register Q contains the quotient.

For example: Dividend (Q) = 11 Divisor (M) = 3 -M = 11101

N	M	A	Q	Action
4	00011	00000	1011	Begin
	00011	00001	011_	Shift left AQ
	00011	11110	011_	A = A - M
3	00011	11110	0110	Q[0] = 0
	00011	11100	110_	Shift left AQ
	00011	11111	110_	A = A + M
2	00011	11111	1100	Q[0] = 0
	00011	11111	100_	Shift left AQ
	00011	00010	100_	A = A +M
1	00011	00010	1001	Q[0] = 1
	00011	00101	001_	Shift left AQ
	00011	00010	001_	A = A - M
0	00011	00010	0011	Q[0] = 1

Table 7: Division Using Non-restoring Method

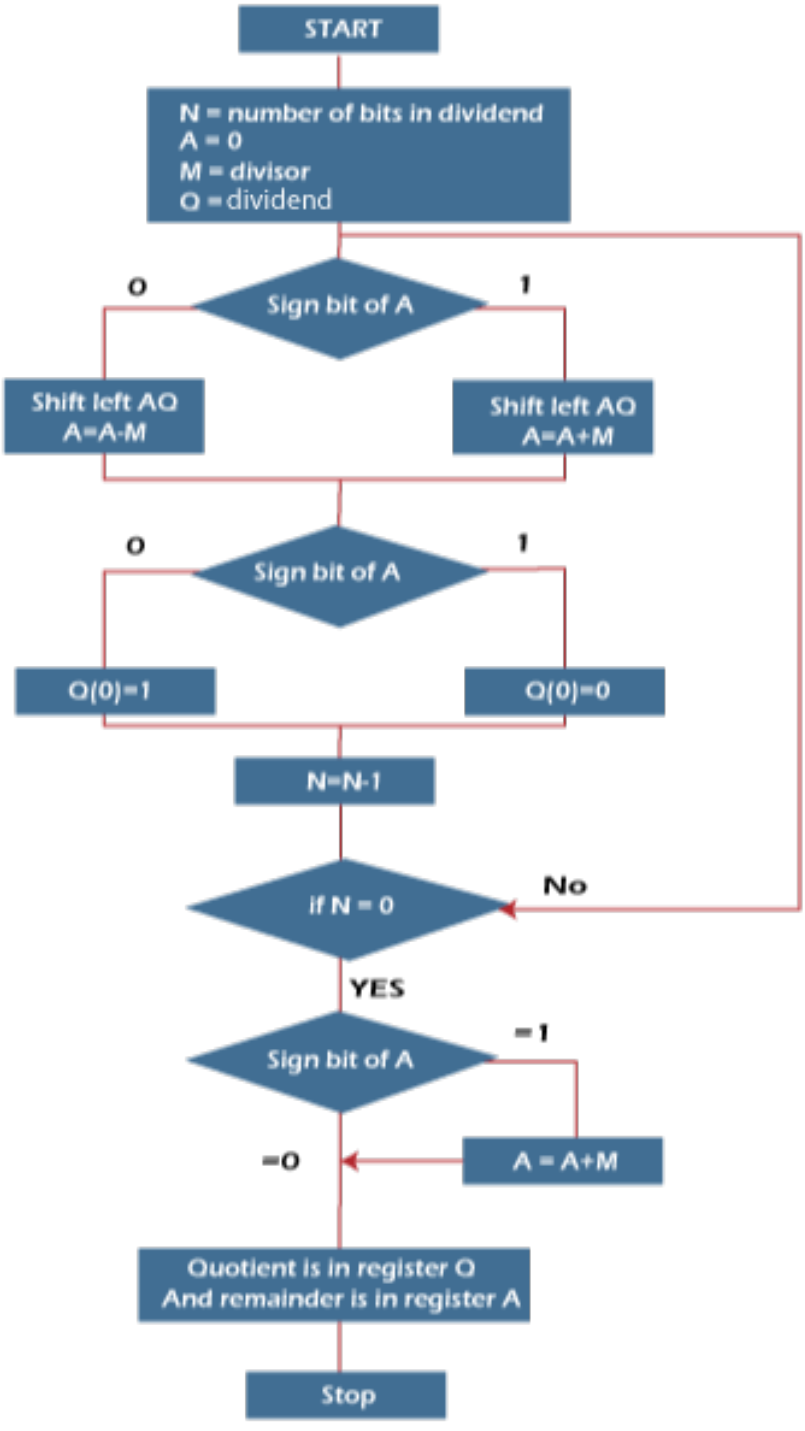


Figure 40: Division Using Non-restoring Method