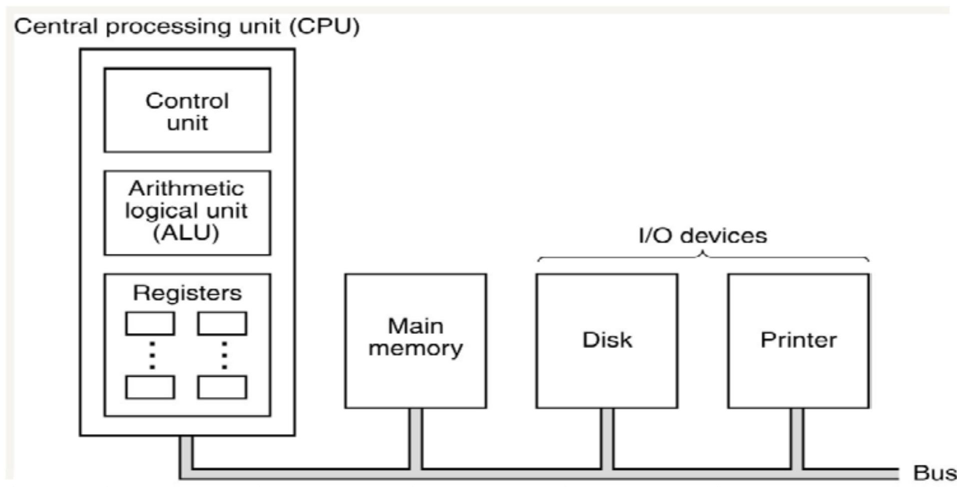
#

PS01CMCA55 - Computer Fundamentals

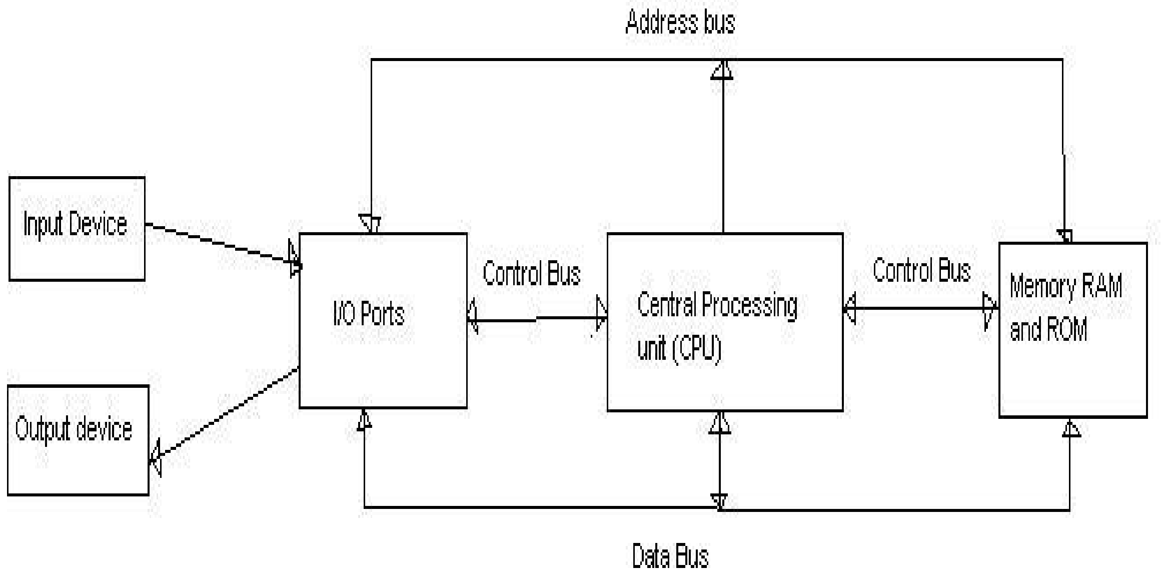
The organization of a simple computer



#

#

Block Diagram of a Simple Computer



Some basic terms

- Program : a sequence of instructions describing how to perform a certain task.
- Computer hardware : Physical components of a computer, such as a motherboard, monitor, keyboard, microprocessor chip, mouse, cables, etc.
- Software : computer programs, algorithms
- Firmware : Software embedded in hardware.
- Bit : A binary digit. (either 0 or 1).
- Byte : A group of 8 bits.
- Nibble : A group of 4 bits.
- 1 Kilo Bytes : 1 KB : 1024 Bytes
- 1 Mega Bytes : 1024 KB
- 1 Giga Bytes : 1 GB : 1024 MB
- 1 TB : 1024 GB

Representation of Information

Each computer has a set of characters that it uses.

Character Code : Mapping of characters onto integers is called a character code.

Each character is represented by some numeric code.

Examples :

American Standard Code for Information Interchange (ASCII) uses 7 bits to represent each character of the computer's character set. ASCII supports 128 different characters.

Extended Binary Coded Decimal Interchange Code (EBCDIC) uses 8 bits to represent each character.

UNICODE represents each character by 16-bit code. With 16-bit symbols, UNICODE has 65,536 code points.

Representation of Numeric Information in Memory

- **Integers**

* Signed Magnitude method : The Most Significant Bit (MSB) is used to represent sign of a number. If the sign bit is 0 , it indicates a positive number. If the sign bit is 1, it is interpreted as a negative number. Remaining n-1 bits out of n bits represent the magnitude. There are two representations of 0 in this method.

* Using 2's complement to represent negative numbers.

- **Floating-Point Numbers**

Floating-point numbers are represented using the following formats :

* IEEE-proposed Single-Precision Format

* IEEE-proposed Double-Precision Format

- **Rpresentation of Floating-Point Numbers**

Floating-point numbers are represented using the following formats :

* IEEE-proposed Single-Precision Format

* IEEE-proposed Double-Precision Format

IEEE-proposed Single-Precision Format

Uses 32 bits to represent floating-point numbers.

---S---	---------- E -----------	-------- F -----------
1 bit	8 bits	23 bits
0 indicates +ve	Exponent	Fraction part
1 indicates –ve	(Excess 127)	

$$V = (-1)^S \times 1.F \times 2^{E-127}$$

IEEE-proposed Double-Precision Format

Uses 64 bits to represent floating-point numbers.

---S---	---------- E -----------	-------- F -----------
1 bit	11 bits	52 bits
0 indicates +ve	Exponent	Fraction part
1 indicates –ve	(Excess 1023)	

$V = (-1)^S \times 1.F \times 2^{E-1023}$

Example : IEEE Single-Precision Format

Represent the floating-point number -0.0011101 in IEEE single-precision format.

First, we represent the given floating-point number in normalized form as follows :

-1.1101×2^{-3}

Biased exponent = True exponent + 127

Hence, Biased exponent = -3 + 127 = 124

The binary representation of 124 is 1111100. If we represent it in 8 bits it is 01111100.

The Fraction part is F = 11010000000000000000000.

Hence, the final answer is : 1 01111100 11010000000000000000000

Hamming Code

Hamming code can detect and correct a single error.

An n-bit codeword has m data bits and r parity bits.

$n = m + r$

where n = Number of bits in a codeword, m = Number of data bits,
and r = Number of parity bits
Parity bit positions are powers of 2.
Positions 1, 2, 4, 8, 16, 32,… are reserved for parity bits.

We add 4 parity bits to 7 data bits to construct a Hamming code consisting of totally 11 bits.

- Parity bit 1 checks bit positions 1, 3, 5, 7, 9, 11,…
- Parity bit 2 checks bit positions 2, 3, 6, 7, 10, 11,…
- Parity bit 4 checks bit positions 4, 5, 6, 7, …
- Parity bit 8 checks bit positions 8, 9, 10, 11,…

Example :

Let us construct a Hamming code for the ASCII character ‘A’ (ASCII value 65) considering even parity.

Binary equivalent of 65 is 1000001.

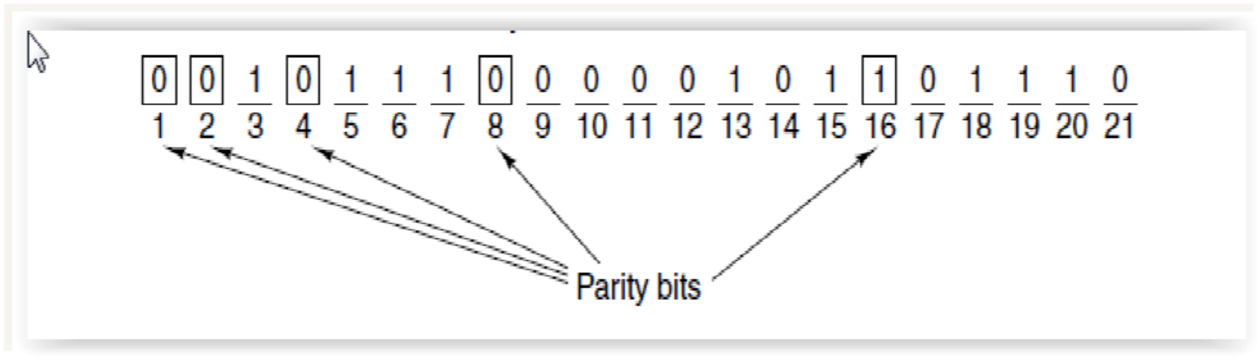
The 7 data bits are : 1 0 0 0 0 0 1

$\underline{0}$	$\underline{0}$	1	$\underline{0}$	0	0	0	$\underline{1}$	0	0	1
1	2	3	4	5	6	7	8	9	10	11

- Parity bit 1 checks bit positions 1, 3, 5, 7, 9, 11,…
- Parity bit 2 checks bit positions 2, 3, 6, 7, 10, 11,…
- Parity bit 4 checks bit positions 4, 5, 6, 7, …
- Parity bit 8 checks bit positions 8, 9, 10, 11,…

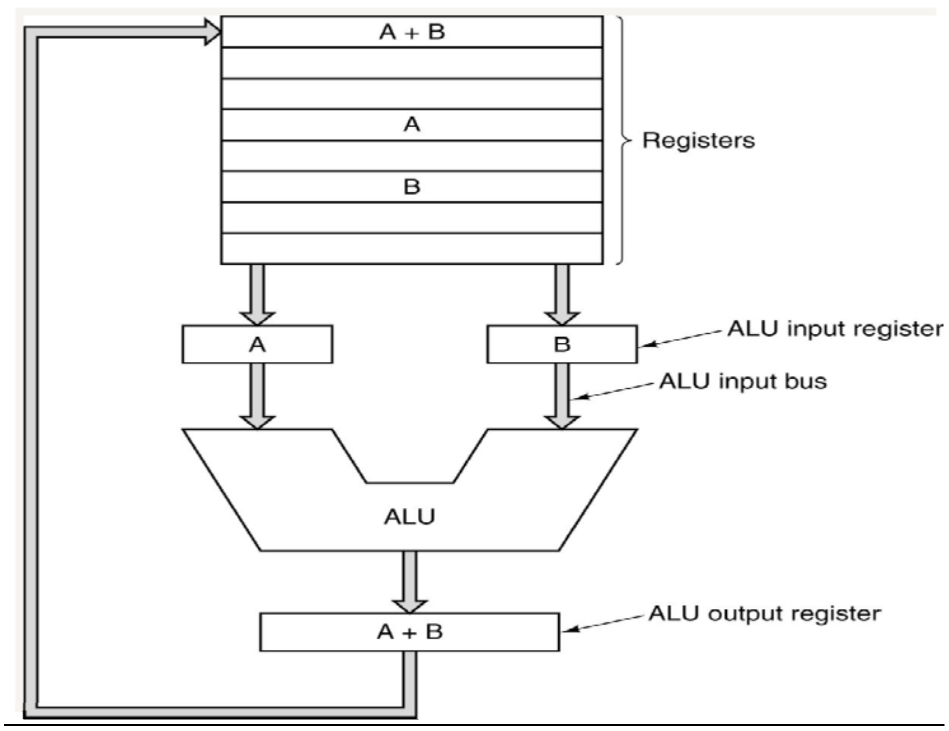
Example :

- Construction of a Hamming code for the memory word (data bits) 1111000010101110 by adding 5 parity bits (check bits) to the 16 data bits, considering even parity.



CPU Organization

The data path of a typical Von Neumann machine



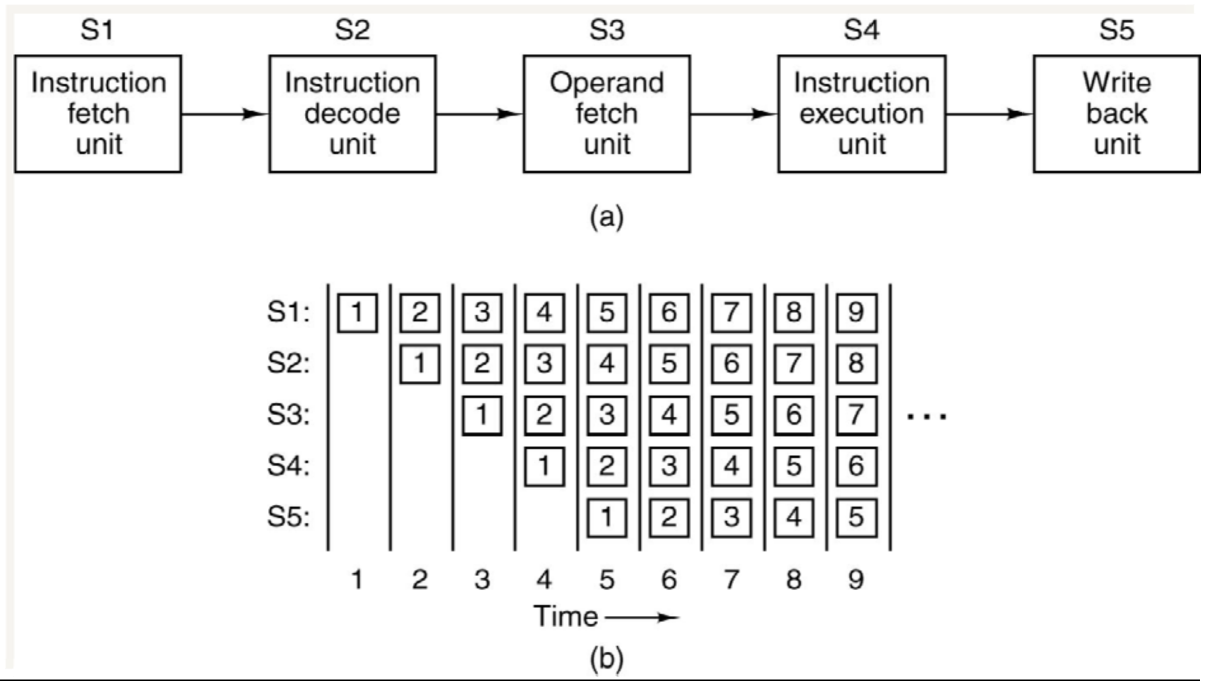
Instruction execution cycle

CPU executes each instruction in a series of small steps :

1. Fetch next instruction from memory into the Instruction Register.
2. Change the Program Counter register to point to the next instruction.
3. Determine the type of instruction just fetched. i.e., Decode the instruction.
4. If the instruction uses data in memory, determine where they are.
5. Fetch data, if necessary, from the main memory into the internal CPU registers.
6. Execute the instruction.
7. Store the result at an appropriate place.
8. Go to step 1 to begin executing the following instruction.

Instruction-Level Parallelism

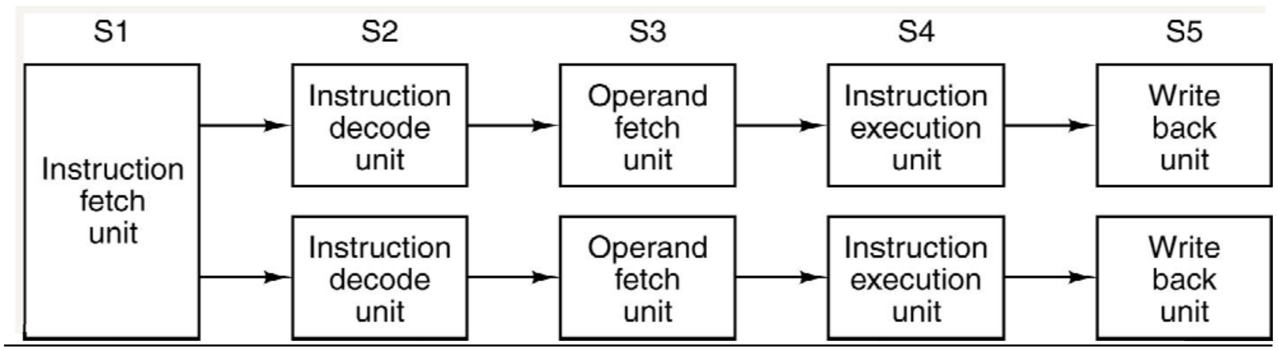
Pipeline Machines



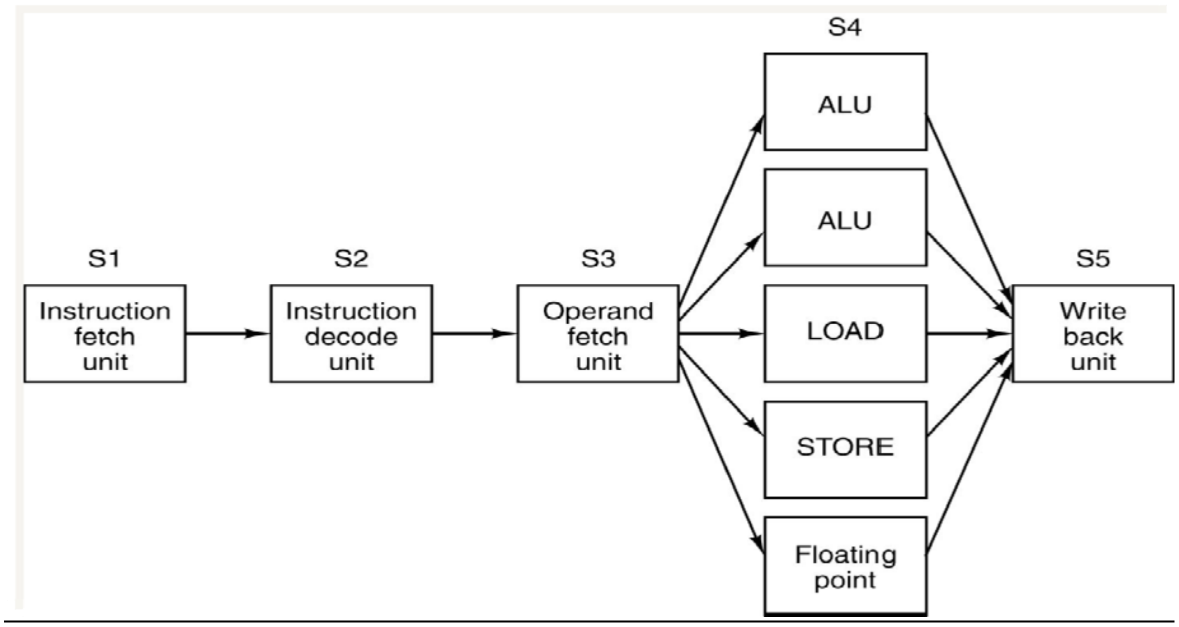
- Figure shows a pipeline machine with five units, also called stages.
- Stage 1 fetches an instruction from memory and places it in a buffer until it is needed. Stage 2 decodes the instruction, determining its type and what operands it needs. Stage 3 locates and fetches the operands, either from registers or from memory. Stage 4 actually does the work of carrying out the instruction, typically by running the instruction through the data path. Finally, stage 5 writes the result back to the proper register.
- Figure (b) shows how the pipeline operates as a function of time.
- During clock cycle 1, stage S1 works on instruction 1, fetching it from memory.
- During clock cycle 2, stage S2 decodes instruction 1, while stage S1 fetches instruction 2.

- During clock cycle 3, stage S3 fetches the operands for instruction 1, stage S2 decodes instruction 2, and stage S1 fetches the third instruction.
- The multiplicative increase in the execution speed of the pipeline machine is N, if there are N units in the pipeline.

Superscalar Architecture :
Dual five-stage pipelines
with a common instruction fetch unit



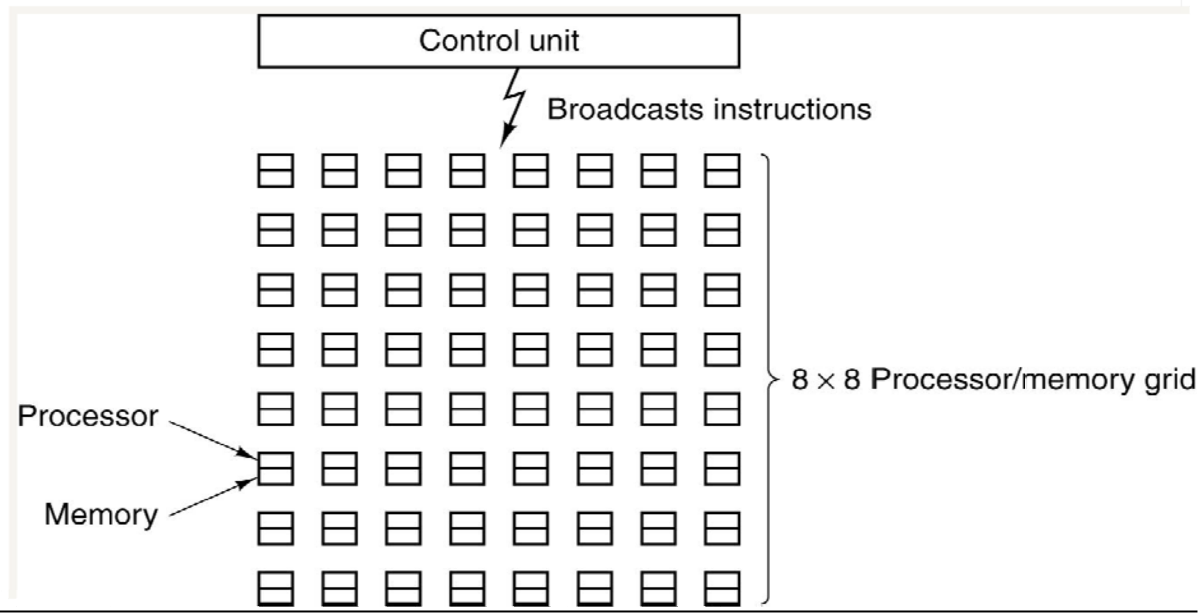
Superscalar Architecture :
A superscalar processor with five functional units



- Stage S3 issues instructions considerably faster than the stage S4 is able to execute them.
- Most of the functional units in stage S4 take appreciably longer time than 1 clock cycle to execute. The functional units that access memory or do floating-point arithmetic take a long time.
- It is possible to have multiple ALUs in stage S4.

Processor-Level Parallelism

An Array Processor of the ILLIAC IV type

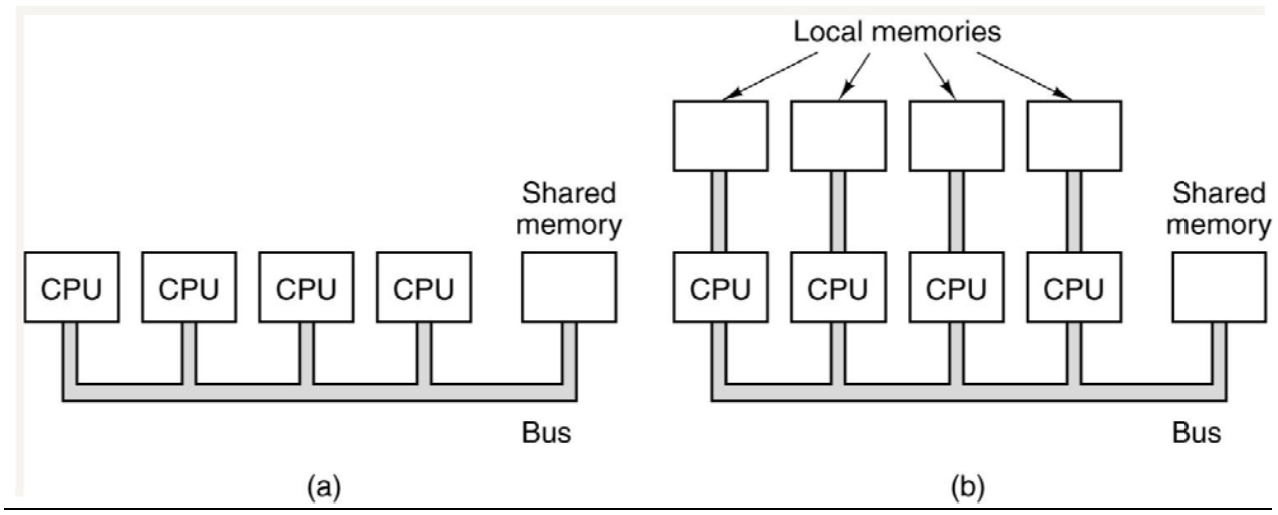


- An array processor consists of a large number of identical processors that perform the same sequence of instructions on different sets of data.
- The world's first array processor was built by the University of Illinois - the ILLIAC IV computer. The original plan was to build a machine consisting of

4 quadrants, each quadrant having an 8x8 square grid of processor / memory elements.

- A single control unit broadcasts instructions which are carried out in lockstep by all the processors, each one using its own data from its own memory.
- SIMD (Single Instruction-stream Multiple Data-stream).

<u>Processor-Level Parallelism :</u>
<u>A single-bus multiprocessor and</u>
<u>a multiprocessor with local memories</u>



(a) A single-bus multiprocessor (b) A multiprocessor with local memories

Bus traffic is a major issue in a single-bus multiprocessor, which can be solved by a multiprocessor with local memories.

<u>RISC versus CISC</u>

- RISC : Reduced Instruction Set Computer
- CISC : Complex Instruction Set Computer

- RISC uses a small number of simple instructions that execute in 1 cycle of the data path (fetching operands into two registers, performing the indicated operation, and storing the result back in a register).
- RISC instructions are fast as they are not interpreted.
- Usage of an instruction set that maximizes the total system performance.
- How long an instruction actually took mattered less than how many could be started per second.

Case study –Intel processors :

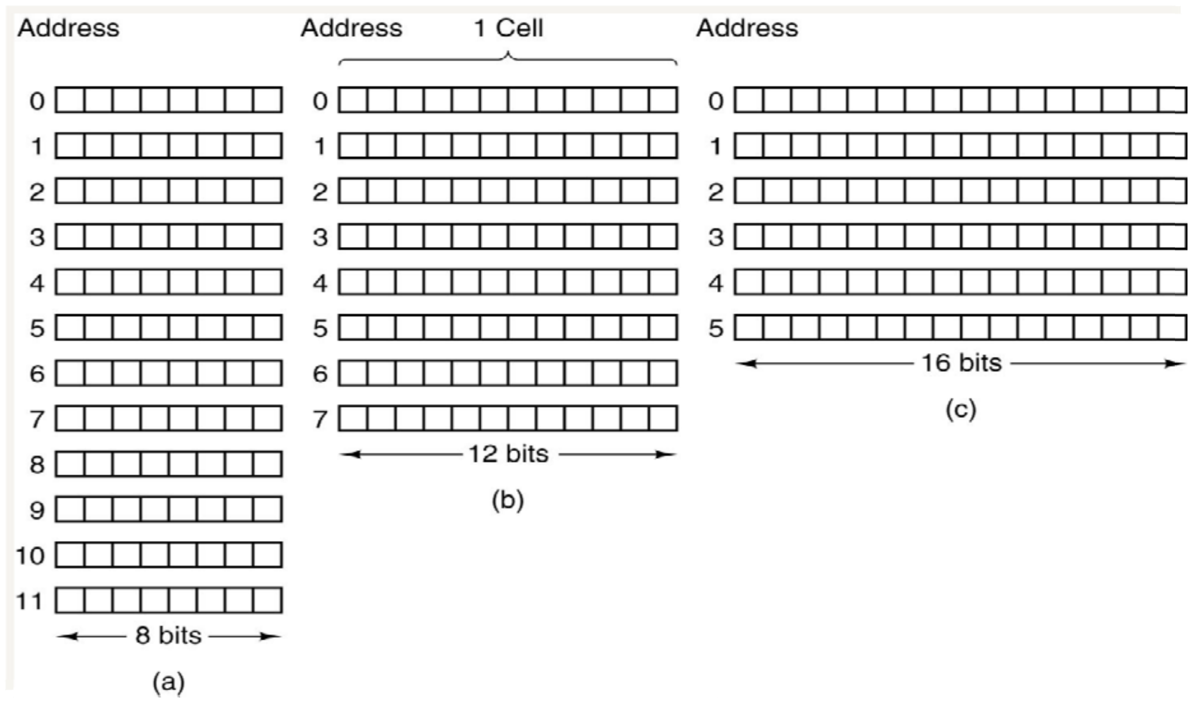
- Hybrid approach : Not as fast as a pure RISC design.
- Intel CPUs contain a RISC core that executes the simplest (and typically most common) instructions in a single data path cycle, while interpreting the more complicated instructions in the usual CISC way. The net result is that common instructions are fast and less common instructions are slow.
- Intel processors : backward compatibility; huge investment by various companies in software for the Intel line.
- Allows old software to run unmodified.
- Competitive overall performance.

Primary Memory

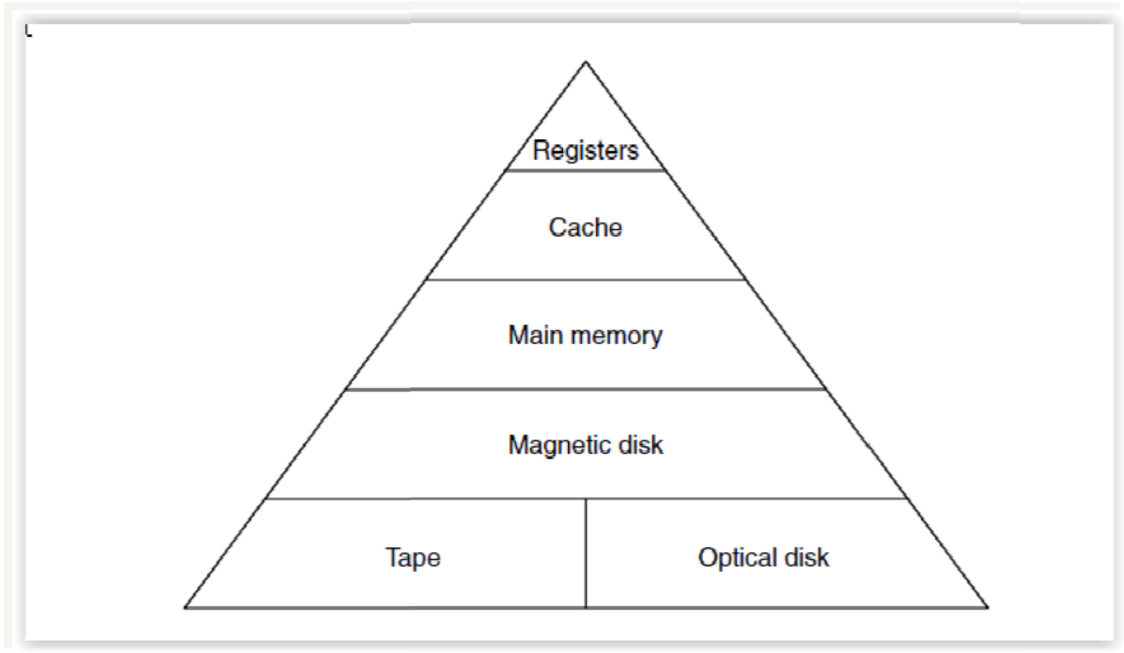
The memory is that part of a computer where programs and data are stored. The primary memory is composed of a large number of cells. Each cell is uniquely identified by a number called a cell address.

Memory Addresses

Three different ways of organizing a 96-bit memory



Memory Hierarchies

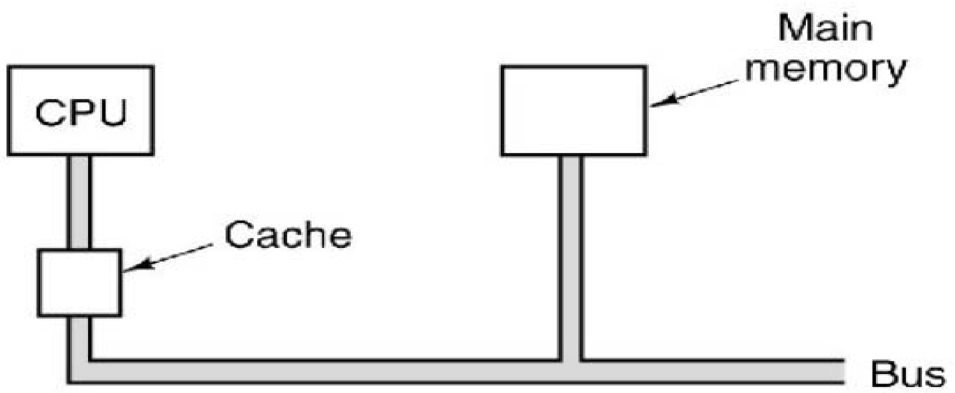


- As we move down the hierarchy, three key parameters change :

    Access time,
    Storage capacity,
    Cost.
- The access time increases as we go downward.
- The storage capacity increases as we move downward.
- The cost decreases as we go downward.

Cache Memory

Cache Memory



The cache is logically between the CPU and main memory. Physically, there are several possible places it could be located.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

- The most heavily used memory words are kept in the cache.
- When a CPU needs a word , it first looks in the cache. Only if the word is not there, does it go to main memory.
- If a substantial fraction of the words are in the cache, the average access time can be greatly reduced.
- Success or failure in finding data depends on what fraction of the words are in the cache.
- Principle of locality : Memory references made in any short time interval tend to use only a small fraction of the total memory.
- Programs do not access their memories completely at random.

- If a given memory reference is to address A, it is likely that the next memory reference will be in the general vicinity of A.
- Except for branches and procedure calls, instructions are fetched from consecutive locations in memory.
- General idea : When a word is referenced , it and some of its neighbors are brought from the large slow memory into the cache, so that the next time it is used, it can be accessed quickly.
- Hit ratio : Fraction of all the references that can be satisfied out of the cache.
- Miss ratio = 1-h.
- If a word is read or written k times in a short interval, the computer will need 1 reference to slow memory and k-1 references to fast memory. The larger k is, the better the overall performance. In this example, the hit ratio $h = (k-1)/k$.
- Mean access time = $c + (1-h)\, m$

  where

  c = cache access time

  m = main memory access time

  h = hit ratio
- As $h \rightarrow 1$, all references can be satisfied out of the cache, and the access time approaches c.
- On the other hand, as $h \rightarrow 0$, a memory reference is needed every time, so the access time approaches c+m, first a time c to check the cache (unsuccessfully), and then a time m to do the memory reference.

Secondary Memory

- Magnetic Disks

  * Hard Disks

  * Floppy Disks

  * IDE Disks

  * SCSI Disks

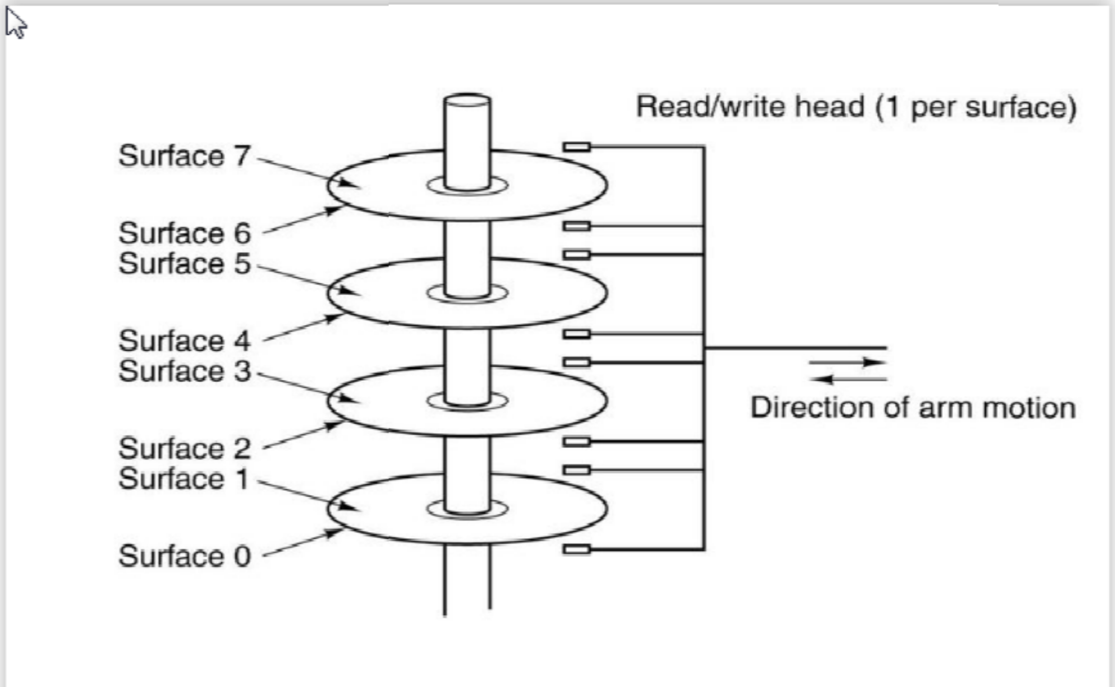
- Optical Storage Devices

  * CD-ROMs

  * CD – Rewritables

  * DVD

  * Blu-Ray



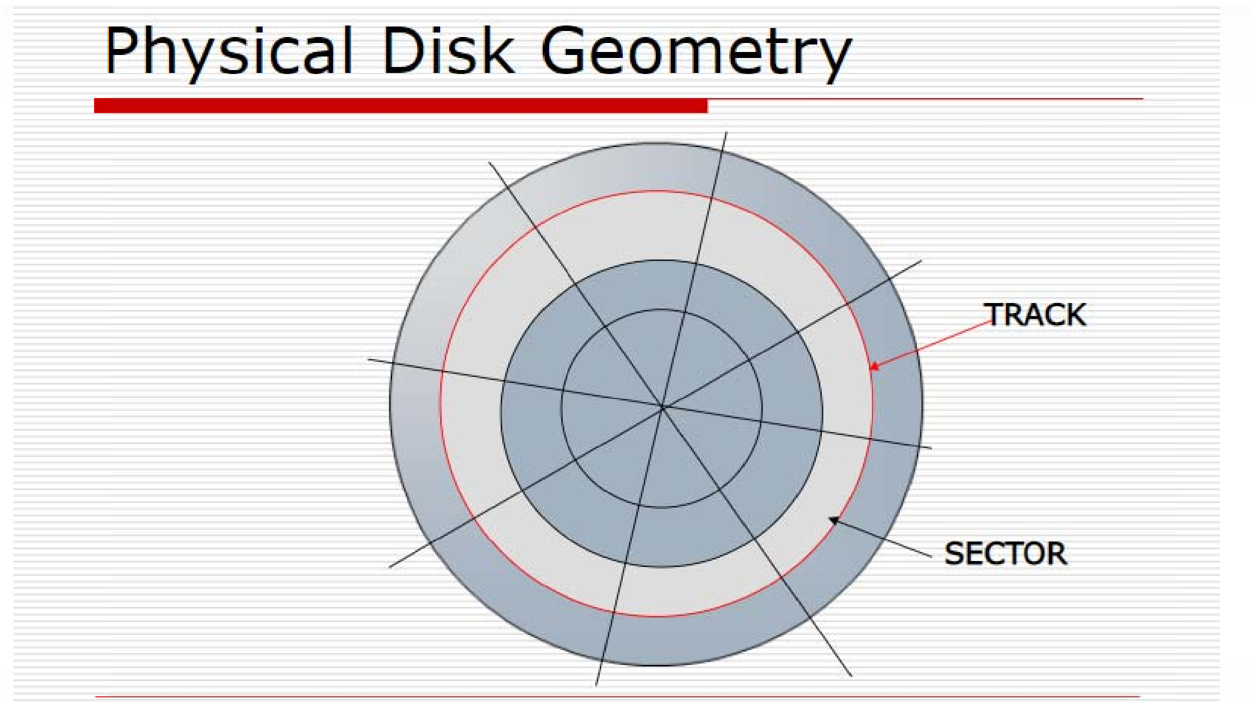
A disk with four platters.

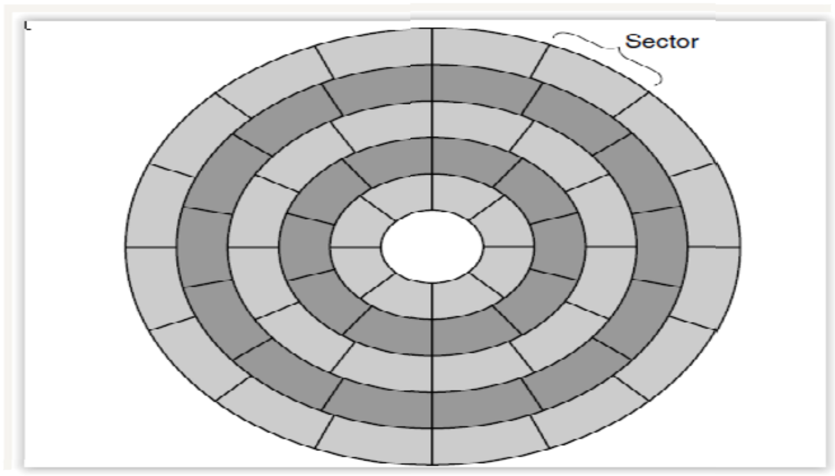
Hard Disks

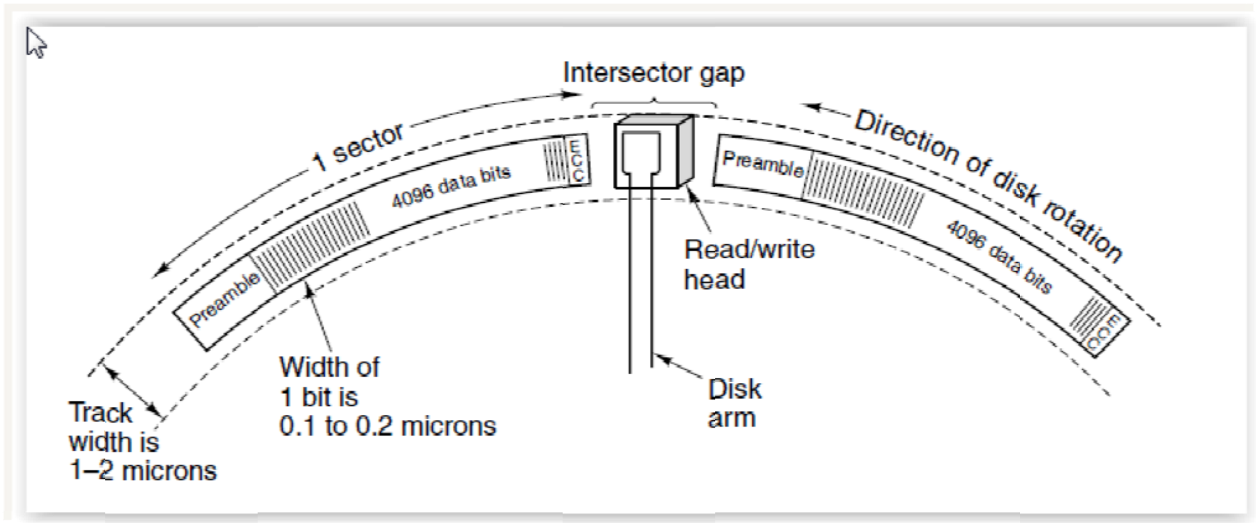
- A magnetic disk consists of one or more aluminum platters with a magnetizable coating.
- A disk head containing an induction coil floats just over the surface, resting on a cushion of air (except for floppy disks, where it touches the surface).
- When a positive or negative current passes through the head, it magnetizes the surface just beneath the head, aligning the magnetic particles facing left or facing right, depending on the polarity of the drive current.
- When a head passes over a magnetized area, a positive or negative current is induced in the head, making it possible to read back the previously stored bits.
- Information storage on a hard disk is organized on a set of concentric circular tracks on each surface of a platter.

- Each track is divided up into some number of fixed-length sectors, typically containing 512 data bytes, preceded by a preamble that allows the head to be synchronized before reading or writing. Following the data is an Error-Correcting Code (ECC), either a Hamming code, or more commonly, a code that can correct multiple errors, called a Reed-Solomon code.
- Between consecutive sectors, is a small intersector gap.
- All disks have movable arms that are capable of moving in and out to different radial distances from the spindle around which the platter rotates.

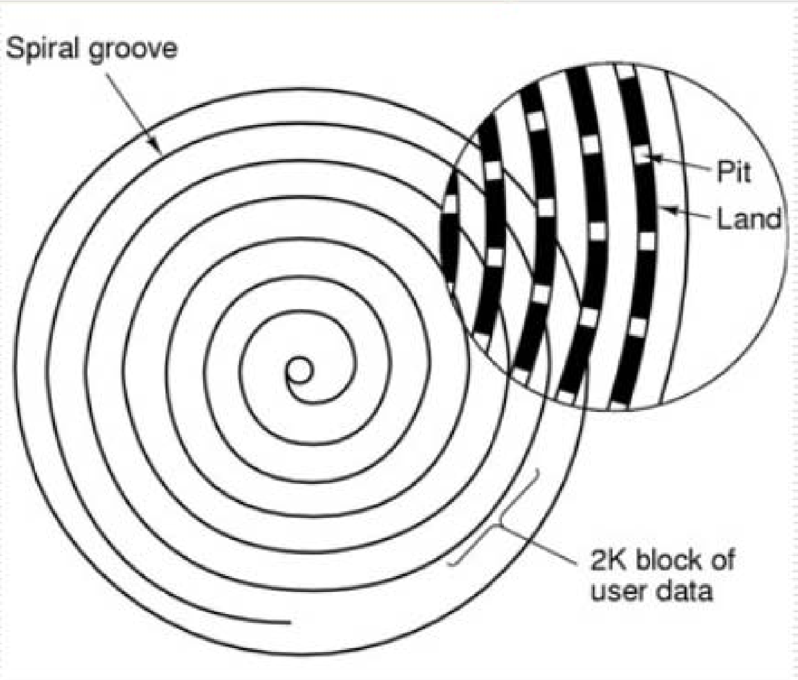
- <u>Seek time :</u> Amount of time required to move the R/W head in order to position it on desired circular track or cylinder.
- <u>Rotational latency :</u> Amount of time needed to rotate the disk in order to bring the desired sector under the R/W head.




Sector


Intersector gap
1 sector
Direction of disk rotation
Preamble
4096 data bits
ECC
Read/write head
Disk arm
Width of 1 bit is 0.1 to 0.2 microns
Track width is 1–2 microns

Compact Disks

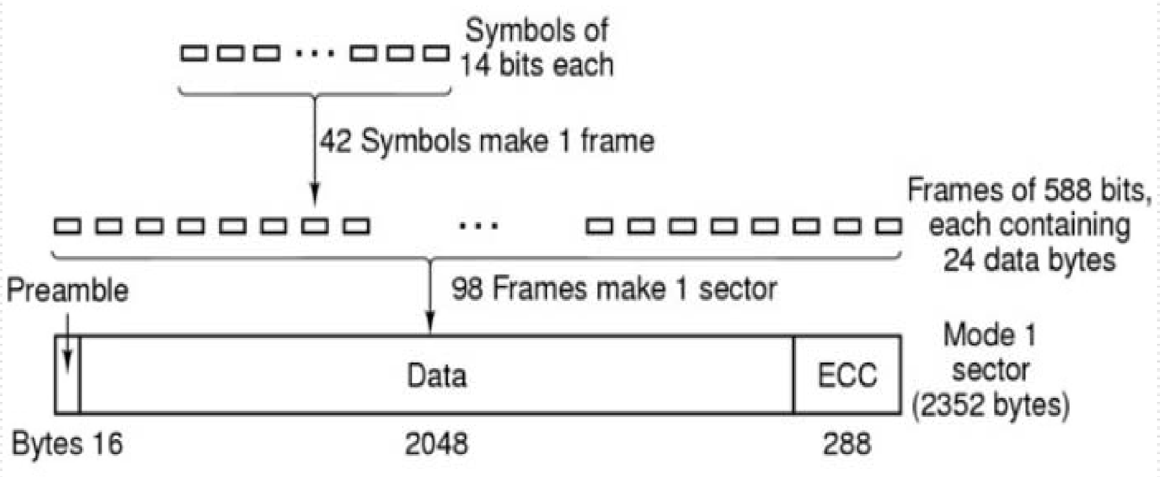


Source: Structured Computer Organization by Tanenbaum

- Information storage on a CD is organized as a single continuous spiral groove containing pits and lands.
- The burned areas due to high-power laser beam incident onto a CD are called **pits**.
- The unburned areas between the pits are called **lands**.
- A CD is prepared by using a high-power infrared laser to burn 0.8-micron diameter holes in a coated glass master disk.
- From this master, a mold is prepared, with bumps where the laser holes were.

- Into this mold, molten polycarbonate resin is injected to form a CD with the same pattern of holes as the glass master.
- Then a very thin layer of reflective aluminum is deposited on the polycarbonate, topped by a protective lacquer and finally a label.
- The depressions in the polycarbonate substrate are called **pits**.
- The unburned areas between the pits are called **lands**.
- When played back, a low-power laser diode shines infrared light with a wavelength of 0.78 micron on the pits and lands as they stream by.
- Because the pits have a height of one-quarter the wavelength of the laser light, light reflecting off a pit is half a wavelength out of phase with light reflecting off the surrounding surface.
- As a result, the two parts interfere destructively and return less light to the player's photodetector than light bouncing off a land. This I how the player tells a pit from a land.
- Although it might seem simplest to use a pit to record a 0 and a land to record a 1, it is more reliable to use a pit/land or land/pit transition for a 1 and its absence as a 0, so this scheme is used.
- The pits and lands are written in a single continuous spiral starting near the hole and working out a distance of 32 mm toward the edge.
- The spiral makes 22,188 revolutions around the disk (about 600 per mm). If unwound, it would be 5.6 km long.
- To make the music play at a uniform rate, it is necessary for the pits and lands to stream by at a constant linear velocity.

- Hence, the rotation rate of the CD must be continuously reduced as the reading head moves from the inside of the CD to the outside. At the iside, the rotation rate is 530 RPM to achieve the desired streaming rate of 120 cm/sec. At the outside, it has to drop to 200 RPM to give the same linear velocity at the head.
- The basic format of data layout on a CD-ROM consists of encoding every byte in a 14-bit symbol. 14-bits are enough to Hamming encode an 8-bit byte with 2 bits left over. The 14-to-8 mapping for reading is done in hardware by table lookup.
- A group of 42 consecutive symbols forms a 588-bit frame. (42x14 = 588). Each frame holds 192 data bits (24 bytes). The remaining 396 bits are used for error correction and control.
- 98 frames are grouped into a CD-ROM sector.
- Every CD-ROM sector begins with a 16-byte preamble, the first 12 of which are 00FFFFFFFFFFFFFFFFFFFF00 hex to allow the player to recognize the start of a CD-ROM sector. The next 3 bytes contain the sector number. To seek the sector, the software in the drive calculates approximately where to go, moves the head there, and then starts hunting around for a preamble to see how good its guess was. The last byte of the preamble is the mode.
- The Yellow Book defines two modes. Mode 1 uses the layout shown in the figure. Mode 2 combines the data and ECC fields.

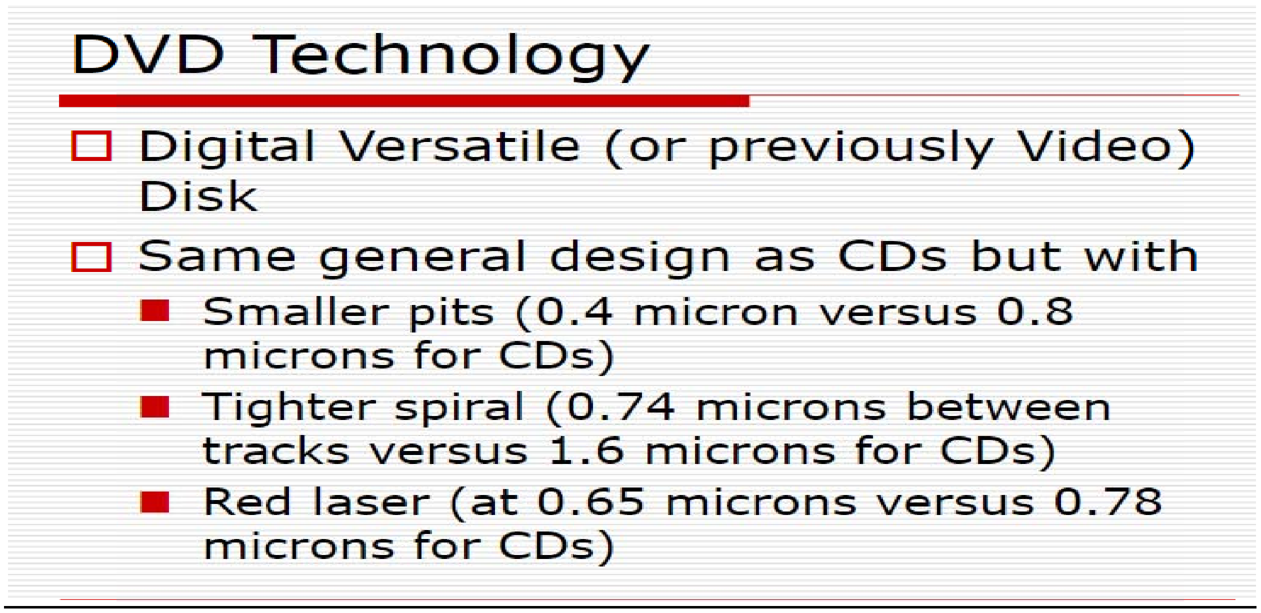


Source: Structured Computer Organization by Tanenbaum

•

CD-Rewritables

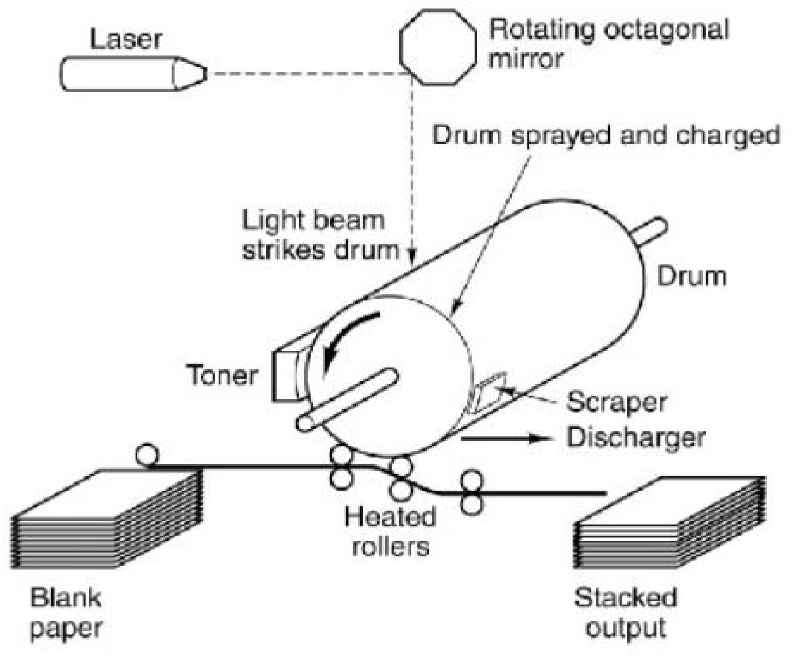
- Instead of a dye, an alloy of silver, indium, antimony and tellurium is used
- Depending on how energy is applied, it goes into an amorphous state or into a crystalline state
- When in an amorphous state, alloy does not reflect
- Again, simulates pits and lands



DVD Technology

- Digital Versatile (or previously Video) Disk
- Same general design as CDs but with
  - Smaller pits (0.4 micron versus 0.8 microns for CDs)
  - Tighter spiral (0.74 microns between tracks versus 1.6 microns for CDs)
  - Red laser (at 0.65 microns versus 0.78 microns for CDs)

<u>Laser Printers</u>

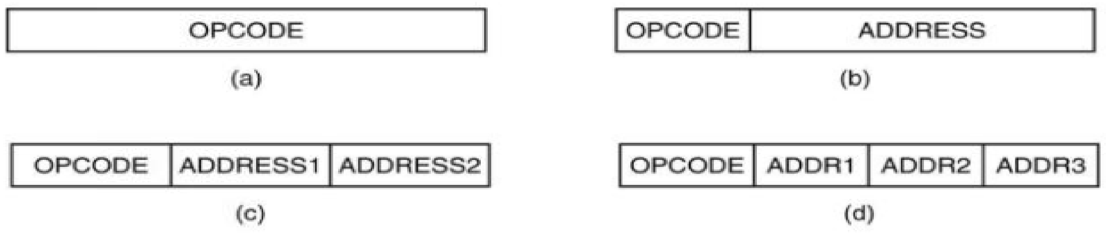


Operation of a laser printer.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

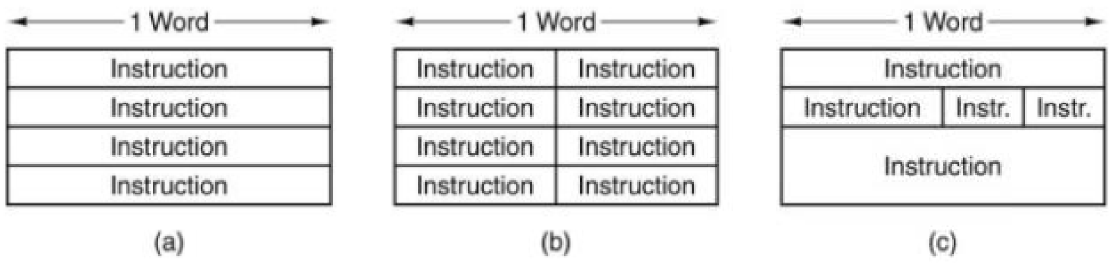
- Basic Technology :
- The heart of the printer is a rotating precision drum (or in some high-end systems, a belt).
- At the start of each page cycle, it is charged up to 1000 volts and coated with a photosensitive material.
- Light from a laser is scanned along the length of the drum much like the electron beam in a CRT.
- A rotating octagonal mirror is used to scan the length of the drum.
- The light beam is modulated to produce a pattern of light and dark spots.
- The spots where the beam hits lose their electrical charge.
- After a line of dots has been painted, the drum rotates a fraction of a degree to allow the next line to be painted.
- Eventually, the first line of dots reaches the toner, a reservoir of an electrostatically sensitive black powder.
- The toner is attracted to those dots that are still charged, thus forming a visual image of that line.
- A little later in the transport path, the toner-coated drum is pressed against the paper, transferring the black powder to the paper.
- The paper is then passed through the heated rollers to fuse the toner to the paper permanently, fixing the image.
- Later in its rotation, the drum is discharged and scraped clean of any residual toner, preparing it for being charged and coated again for the next page.

Instruction Formats



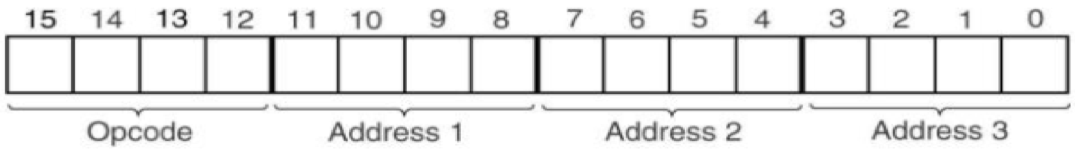
Four common instruction formats:
(a) Zero-address instruction. (b) One-address instruction
(c) Two-address instruction. (d) Three-address instruction.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



Some possible relationships between instruction and word length.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0



An instruction with a 4-bit opcode and three 4-bit address fields.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Design Criteria for Instruction Formats

(1) Short instructions are better than long ones.

Short instructions occupy small amount of memory and need less time to read.

(2) There should be sufficient room in the instruction format to express all the operations desired.

(3) Number of bits in an address field.

In order to gain a finer memory resolution one must pay the price of longer addresses and thus longer instructions.

(4) The instruction length versus character code length.

The instruction length should be integral multiple of the character code length or vice versa for achieving efficiency of algorithms for information retrieval.

Addressing Modes

- Immediate addressing
- Direct Addressing
- Register Addressing
- Register Indirect Addressing
- Indexed Addressing
- Based-Indexed Addressing
- Stack Addressing
- How the bits of an address field are interpreted to find an operand ?
- One possibility is that they contain the memory address of the operand. In addition to the large field needed to specify the full memory address, this method has the further constraint that the address must be determined at the compile time. Other possibilities exist, which provide both shorter specifications and the ability to determine addresses dynamically.

Immediate Addressing

- Address part of the instruction contains the operand itself rather than an address or other information describing where the operand is.
- Such an operand is called an immediate operand because it is automatically fetched from memory at the same time when the instruction itself is fetched. Hence, it is immediately available for use.
- It does not require an extra memory reference to fetch the operand.

- Disadvantage : Only a constant can be supplied this way. Also, the number of values is limited by the size of the field.
- Example : MOV R1, 4

Direct Addressing

- A method for specifying an operand in memory is just to give its full address.
- Restricted in its use : The instruction will always access exactly the same memory location.
- So while the value can change, the location cannot.
- Direct addressing can only be used to access global variables whose address is known at compile time.
- Example : MOV R1, PRICE

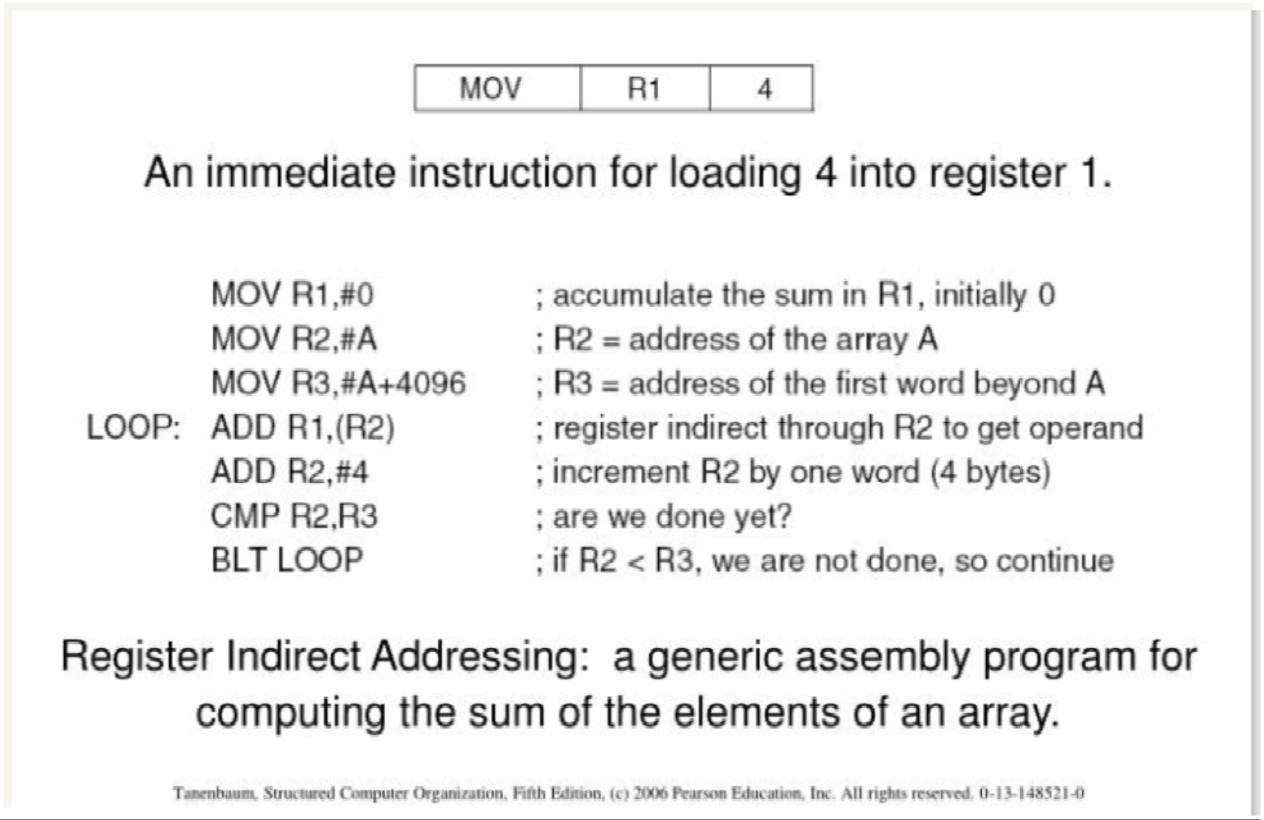
Register Addressing

- Register addressing is conceptually the same as direct addressing, but specifies a register instead of a memory location.
- Registers are very important : fast access, short addresses
- Determine which variables will be accessed most often (for example, a loop index) and put these variables in registers.
- Example : MOV AX, BX (Intel 8086 processor)

Register Indirect Addressing

- In this mode, the operand being specified comes from memory or goes to memory, but its address is not hardwired into the instruction, as in direct addressing. Instead, the address is contained in a register. When an address is use in this manner, it is called a pointer.

- It can reference memory without paying the price of having a full memory address in the instruction
- Example : A generic assembly program for computing the sum of the elements of an array.



MOV	R1	4

An immediate instruction for loading 4 into register 1.

```
       MOV R1,#0          ; accumulate the sum in R1, initially 0
       MOV R2,#A          ; R2 = address of the array A
       MOV R3,#A+4096     ; R3 = address of the first word beyond A
LOOP:  ADD R1,(R2)        ; register indirect through R2 to get operand
       ADD R2,#4          ; increment R2 by one word (4 bytes)
       CMP R2,R3          ; are we done yet?
       BLT LOOP           ; if R2 < R3, we are not done, so continue
```

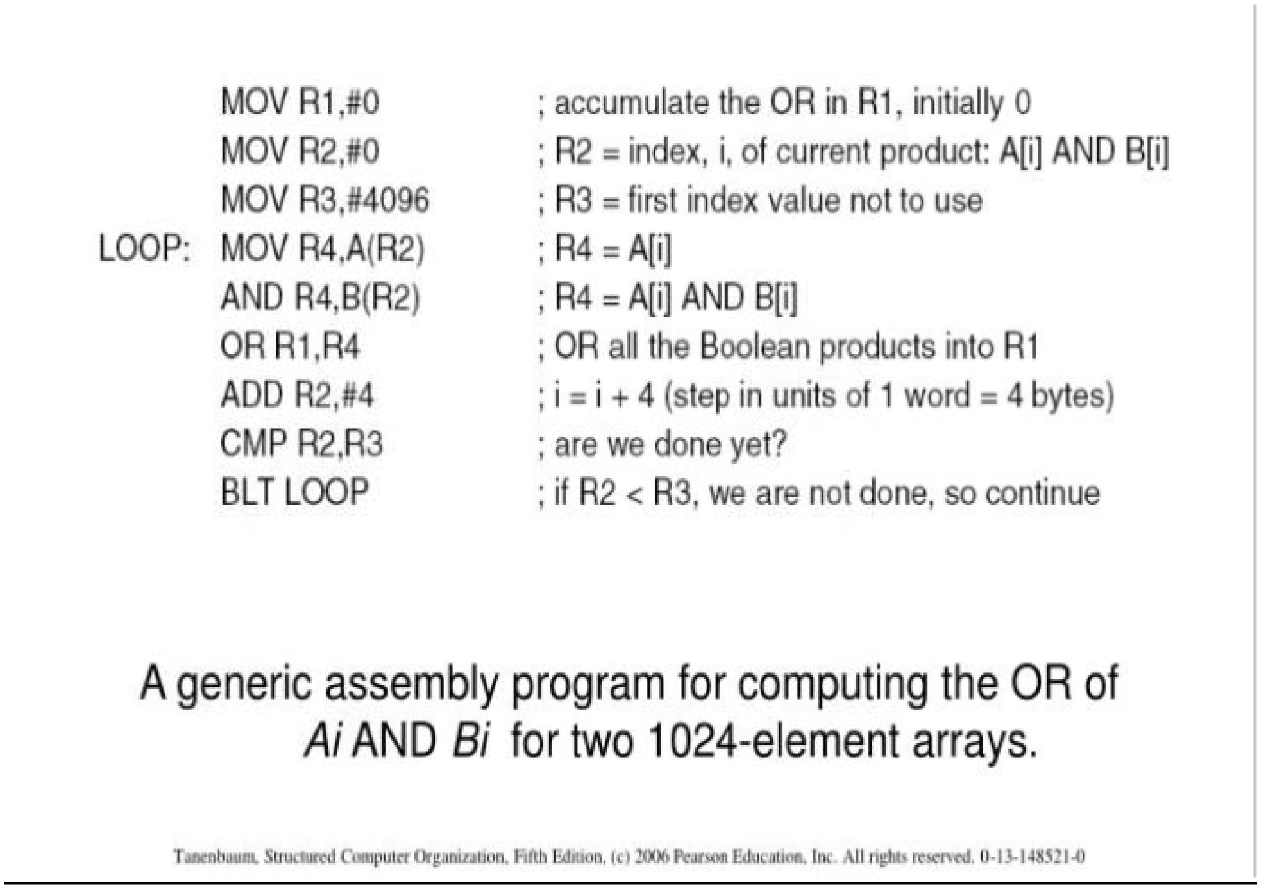
Register Indirect Addressing: a generic assembly program for computing the sum of the elements of an array.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Indexed Addressing

- It is useful to be able to reference memory words at a known offset from a register.
- Addressing memory by giving a register (explicit or implicit) plus a constant offset is called indexed addressing.

- Example : A generic assembly program for computing the OR of A_i and B_i for two 1024-element arrays.



```
        MOV R1,#0           ; accumulate the OR in R1, initially 0
        MOV R2,#0           ; R2 = index, i, of current product: A[i] AND B[i]
        MOV R3,#4096        ; R3 = first index value not to use
LOOP:   MOV R4,A(R2)        ; R4 = A[i]
        AND R4,B(R2)        ; R4 = A[i] AND B[i]
        OR R1,R4            ; OR all the Boolean products into R1
        ADD R2,#4           ; i = i + 4 (step in units of 1 word = 4 bytes)
        CMP R2,R3           ; are we done yet?
        BLT LOOP            ; if R2 < R3, we are not done, so continue
```

A generic assembly program for computing the OR of *Ai* AND *Bi* for two 1024-element arrays.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Traps

- A trap is a kind of automatic procedure call initiated by some condition caused by the program, usually an important but rarely occurring condition.
- When a trap occurs, the flow of control is transferred to some fixed memory location, instead of continuing in sequence. At that fixed location, is a branch to a procedure called a **trap handler.**
- A trap is initiated by some exceptional condition caused by a program itself and detected by the hardware or microprogram.

A few common conditions that can cause traps :

- Floating-point overflow,
- Floating-point underflow,
- Integer overflow,
- Protection violation,
- Undefined opcode,
- Stack overflow,
- Attempt to start non-existent I/O device,
- Division by zero
- Attempt to fetch a word from an odd-numbered address on some systems.

Interrupts

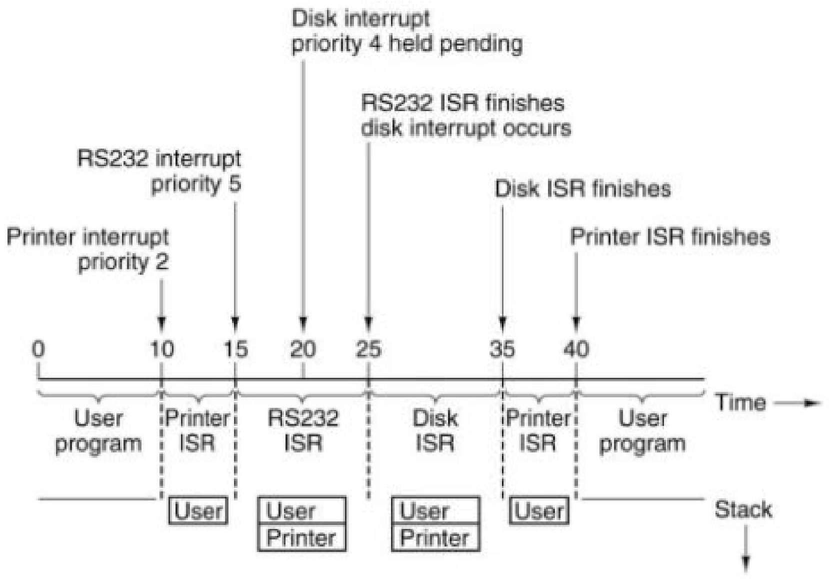
- Interrupts are changes in the flow of control caused not by the running program, but by something else, usually related to I/O.
- For example, a program may instruct he disk to start transferring information, and set the disk up to provide an interrupt as soon as the transfer is finished.
- Like the trap, the interrupt stops the running program and transfers the flow of control to an interrupt handler, which performs some appropriate action.
- When finished, the interrupt handler returns control to the interrupted program. It must restart the interrupted process in exactly the same state it was in when the interrupt occurred, which means restoring all the internal registers to their pre-interrupt state.

Traps and Interrupts

The essential difference between traps and interrupts :

- Traps are synchronous with the program and interrupts are asynchronous.
- If a program is rerun a million times with the same input, the traps will reoccur in the same place each time, but the interrupts may vary, depending on, for example, precisely when a person at a terminal hits carriage return.
- Reproducibility of traps versus irreproducibility of interrupts :
- Traps are caused directly by the program, and interrupts are, at best, indirectly caused by the program.

- Example : Having started an I/O, a CPU may be free to run another program or do something else.
- The key concept related to interrupt is transparency.
- When an interrupt occurs, some actions are taken and some code runs, but when everything is finished, the computer should be returned to exactly the same state it had before the interrupt. An interrupt routine with this property is said to be transparent.
- A large computer may have many I/O devices, and several may be running at the same time, possibly on behalf of different users.
- A non-zero probability exists that while an interrupt routine is running, a second I/O device wants to generate its interrupt.
- Assign each I/O device a priority.
- While a priority n interrupt routine is running, any attempt by a device with a lower priority to cause an interrupt is ignored until the interrupt routine is finished.



Time sequence of multiple interrupt example.

Excercise

1. Draw the block diagram of a simple computer . Explain the function performed by each block of the diagram.

2. Write the full forms of the following acronyms :

(i) ASCII (ii) EBCDIC (iii) CPU (iv) RAM (v) EPROM (vi) ALU

3. Construct a Hamming code for the ASCII character ‘B’ (ASCII : 66) considering odd parity.

4. Construct a Hamming code for the ASCII character ‘a’ (ASCII : 97) considering even parity.

5. Represent the binary number -0.0001011 in the IEEE single-precision format for floating-point numbers.

6. Represent the decimal number 53.75 in the IEEE single-precision format for floating-point numbers.

7. Find out the floating-point number that is represented by the following sequence of bits in the IEEE single-precision format :

1 01000011 11010000000000000000000

8. Write a short note on

(i) Pipeline machines (ii) Array processors

(iii) Multiprocessors

(iv) Superscalar computers with multiple functional units

9. Define a character code. Give any two examples.

10. Write the drawback of the signed magnitude method.

11. Which function is performed by a CPU ?

12. What are the distinct components of a CPU ?

13. Write a short not on a data path to explain the CPU organization.

14. Define : (i) computer hardware, (ii) software, (iii) firmware.

15. Specify examples of input devices and output devices.
16. Write the steps involved in instruction execution by a CPU.
17. Which register holds the address of the instruction to be executed next ?
18. Write a short note on (i) hard disks, (ii) CD-ROM, (iii) laser printers.
19. Define : (i) seek time, (ii) rotational latency.
20. Describe the design criteria for instruction formats.
21. What do you mean by an addressing mode? List various addressing modes.
22. What are the advantages and disadvantages of the immedite addressing mode ?
23. Distinguish between : direct and indirect addressing modes.
24. What do you mean by a trap ? Give examples of the common conditions that can cause a trap.
25. What is an interrupt ?
26. Distinguish between traps and interrupts.