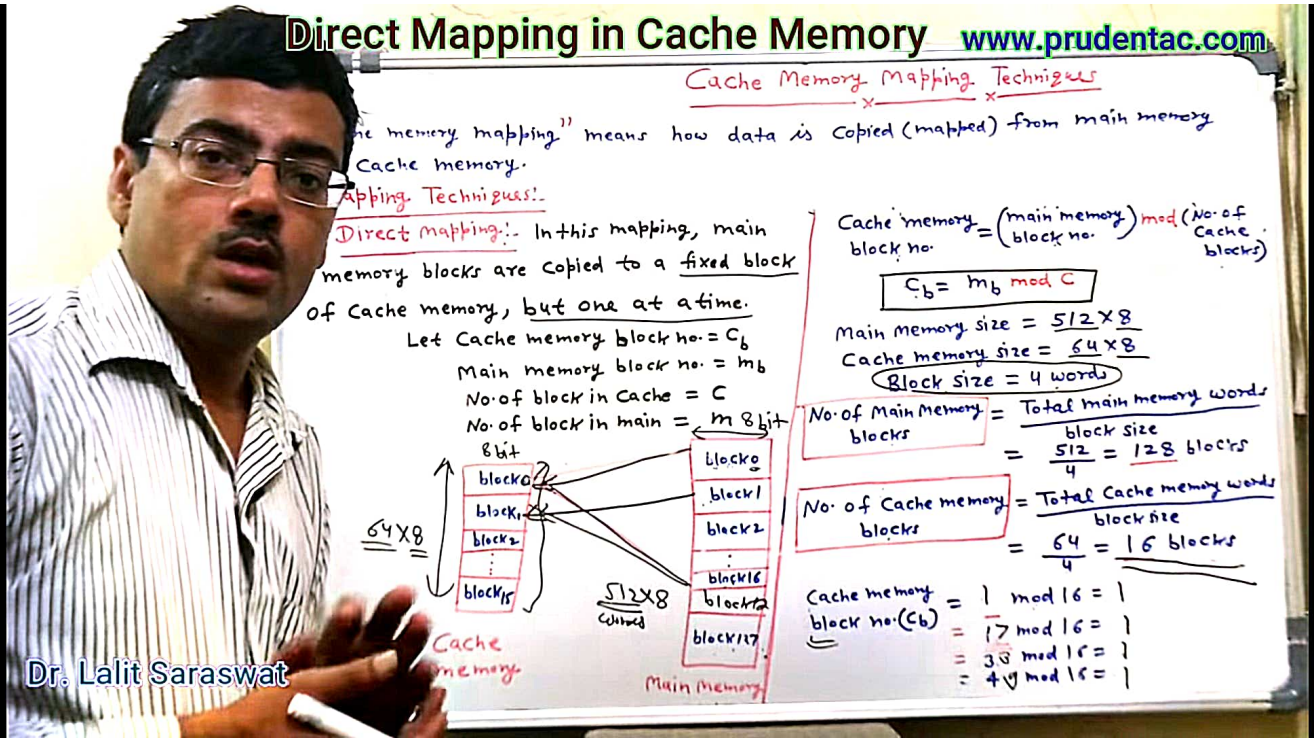
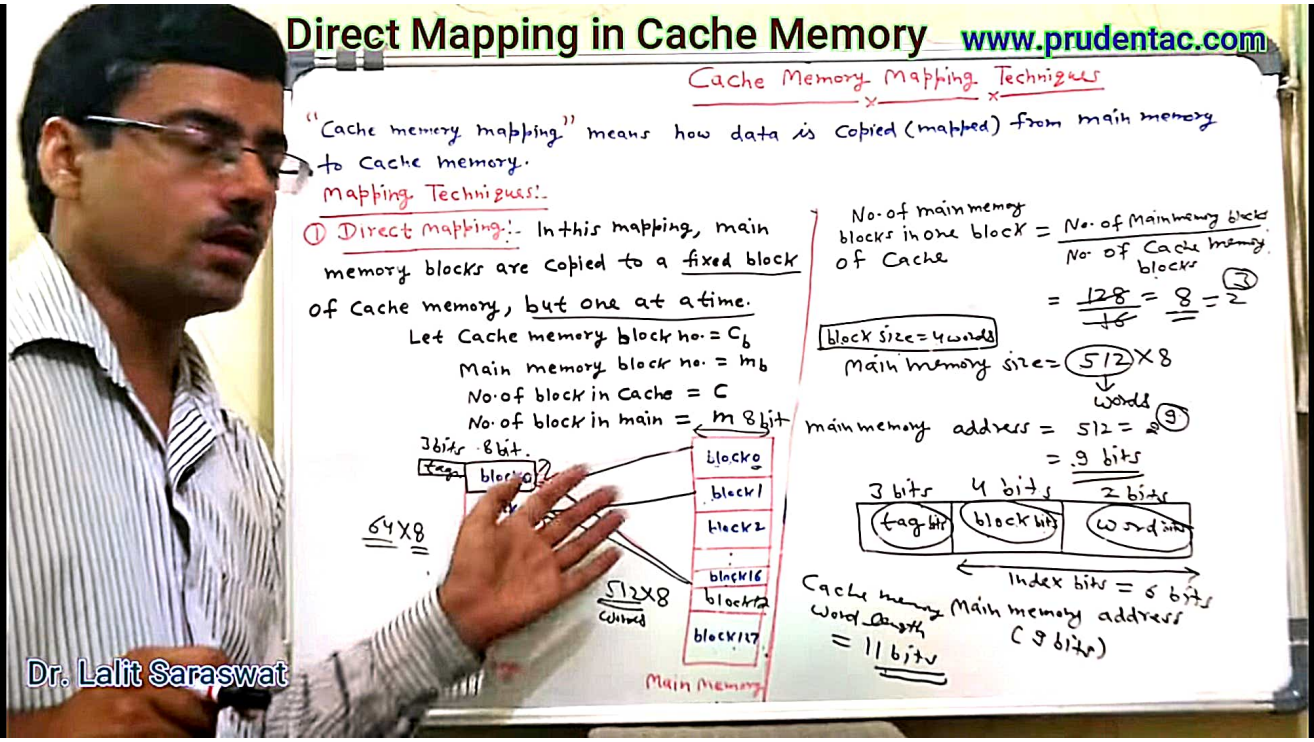
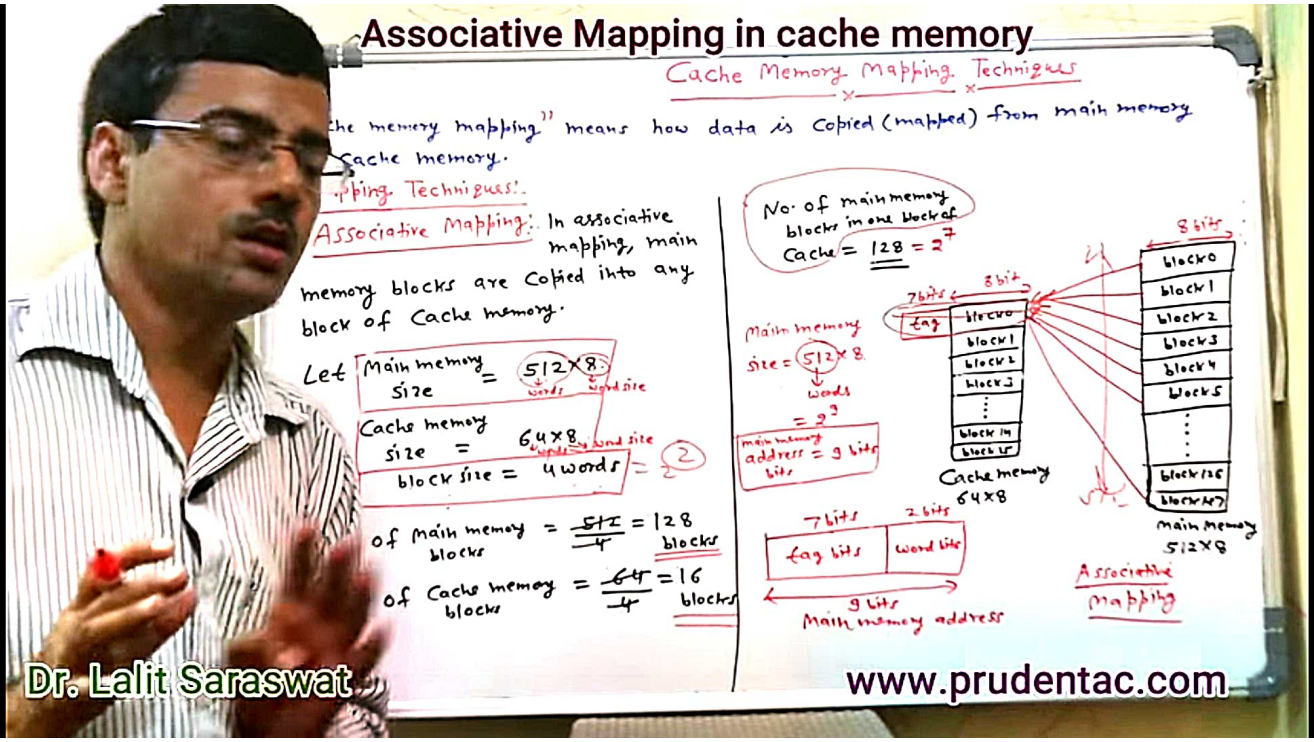

Direct Mapping in Cache Memory
www.prudentac.com
Cache Memory Mapping Techniques
"Cache memory mapping" means how data is copied (mapped) from main memory to Cache memory.
Mapping Techniques:-
Direct mapping:- In this mapping, main memory blocks are copied to a fixed block of Cache memory, but one at a time.
Let Cache memory block no. = C_b
Main memory block no. = m_b
No. of block in Cache = C
No. of block in main = m
8 bit
block0
block1
block2
block15
64X8
512X8
block0
block1
block2
block16
block17
Cache memory
Main memory
Cache memory block no. = (main memory block no.) mod (No. of Cache blocks)
$C_b = m_b \bmod C$
Main memory size = 512X8
Cache memory size = 64X8
Block size = 4 words
No. of Main Memory blocks = Total main memory words / block size = 512/4 = 128 blocks
No. of Cache memory blocks = Total Cache memory words / block size = 64/4 = 16 blocks
Cache memory block no.(C_b) = 1 mod 16 = 1
= 17 mod 16 = 1
= 33 mod 16 = 1
= 49 mod 16 = 1
Dr. Lalit Saraswat


Direct Mapping in Cache Memory
www.prudentac.com
Cache Memory Mapping Techniques
"Cache memory mapping" means how data is copied (mapped) from main memory to cache memory.
Mapping Techniques:-
① Direct mapping:- In this mapping, main memory blocks are copied to a fixed block of cache memory, but one at a time.
Let Cache memory block no. = C_b
Main memory block no. = m_b
No. of block in Cache = C
No. of block in main = m
3 bits 8 bit
tag block
64X8
512X8
block0
block1
block2
block16
block17
Main Memory
No. of main memory blocks in one block of Cache = No. of main memory blocks / No. of Cache memory blocks
= 128/16 = 8 = 2^3
Block size = 4 words
Main memory size = 512 X 8
words
main memory address = 512 = 2^9
= 9 bits
3 bits 4 bits 2 bits
tag bits block bits word bits
Index bits = 6 bits
Cache memory Word length = 11 bits
Main memory address (9 bits)
Dr. Lalit Saraswat

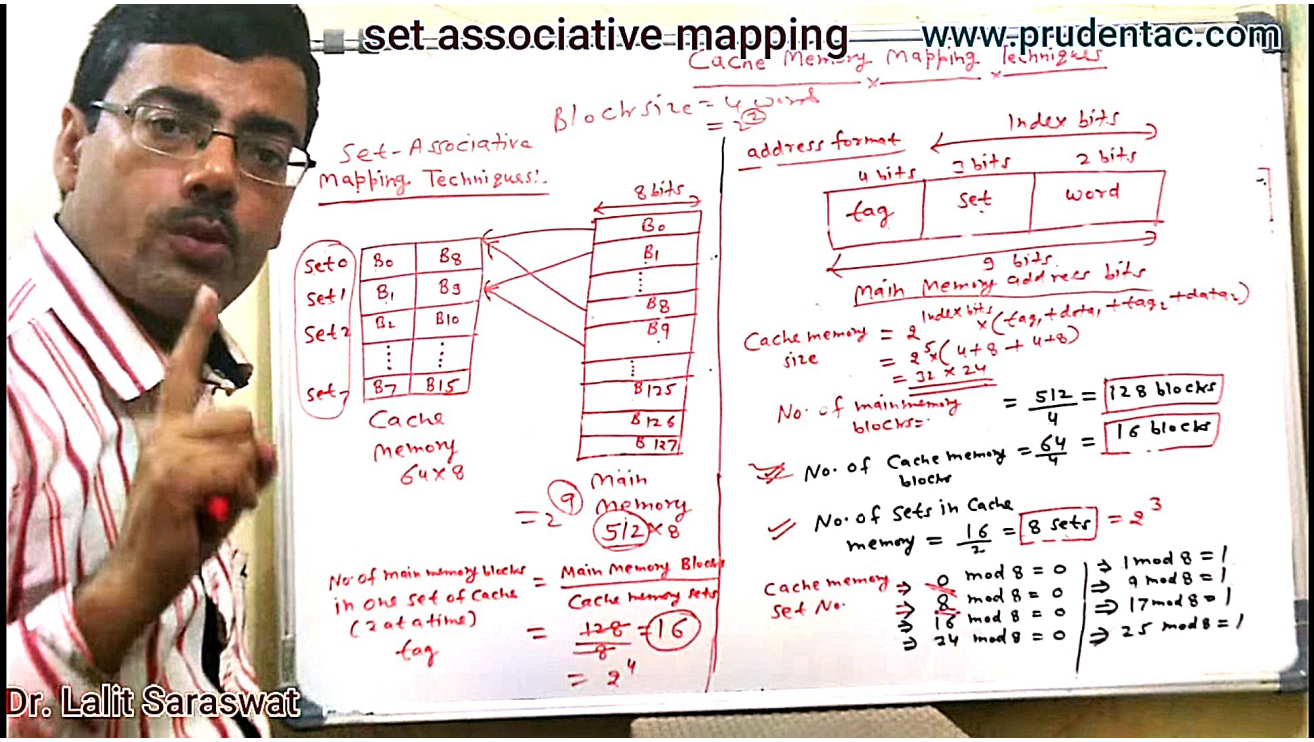
Associative Mapping in cache memory

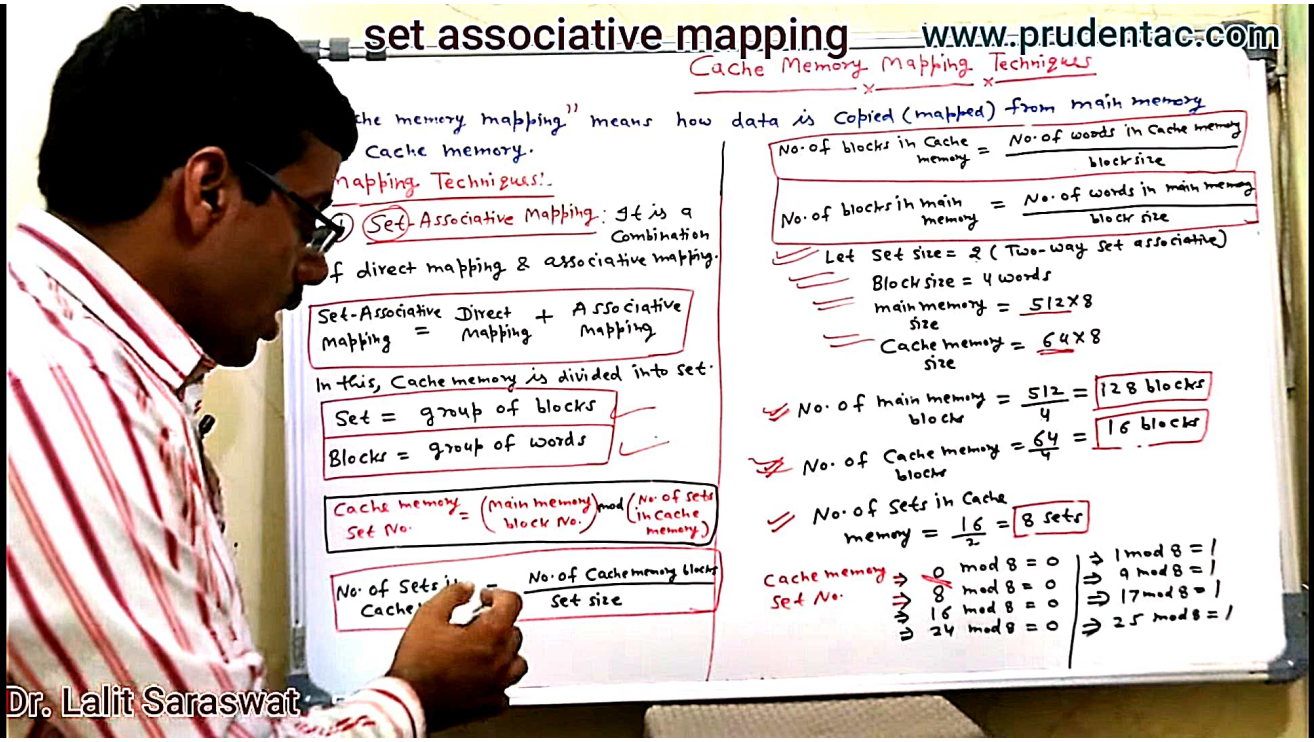


Least Recently used (Replace the least recently used page in Past)

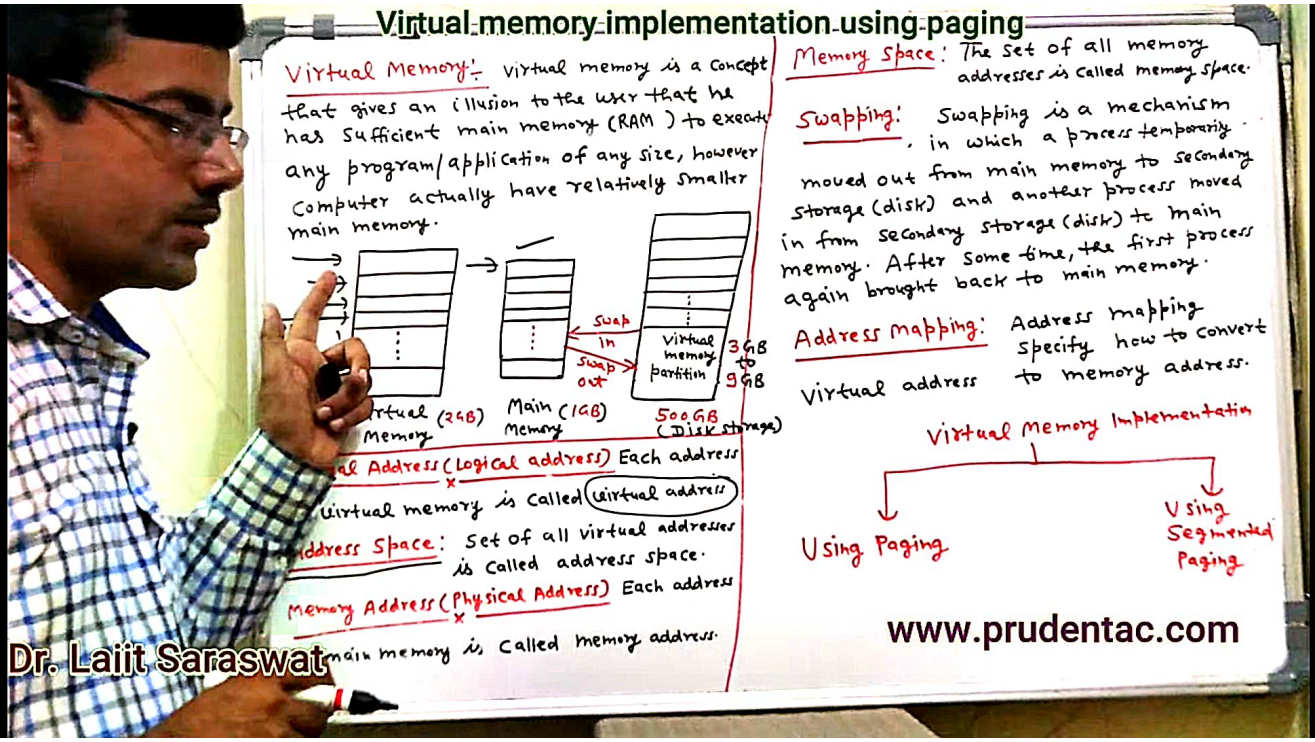
f_4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3			1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
f_2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hi

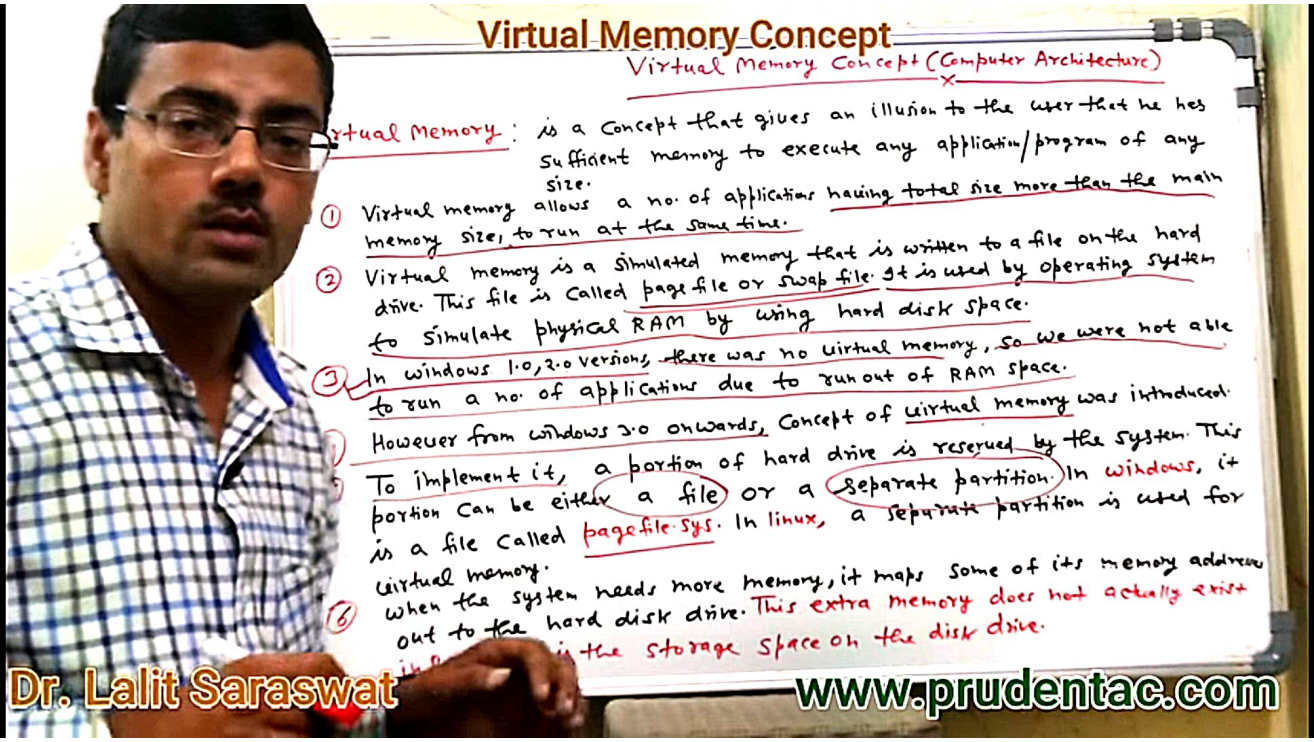
Ref → 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

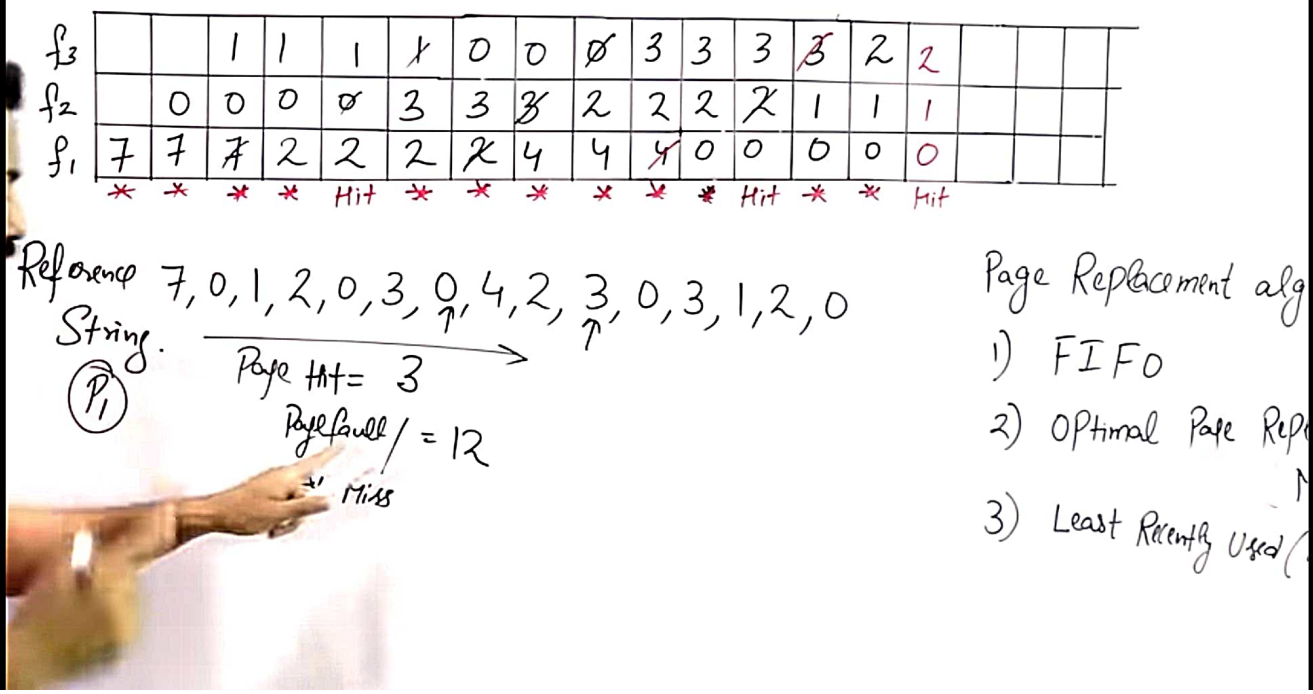

set associative mapping
www.prudentac.com
Cache Memory Mapping Techniques
Block size = 4 words
Set-Associative Mapping Techniques:
address format
Index bits
4 bits
3 bits
2 bits
tag
Set
word
9 bits
Main Memory address bits
8 bits
Set 0
Set 1
Set 2
Set 7
Cache memory 64 x 8
Main memory 512 x 8
Cache memory size = 2^index bits x (tag1 + data1 + tag2 + data2)
= 2^5 (4+8+4+8)
= 32 x 24
No. of main memory blocks = 512/4 = 128 blocks
No. of Cache memory blocks = 64/4 = 16 blocks
No. of Sets in Cache memory = 16/2 = 8 sets = 2^3
No. of main memory blocks in one set of Cache (2 at a time) tag = Main Memory Blocks / Cache memory sets = 128/8 = 16 = 2^4
Cache memory Set No.
0 mod 8 = 0
8 mod 8 = 0
16 mod 8 = 0
24 mod 8 = 0
1 mod 8 = 1
9 mod 8 = 1
17 mod 8 = 1
25 mod 8 = 1
Dr. Lalit Saraswat

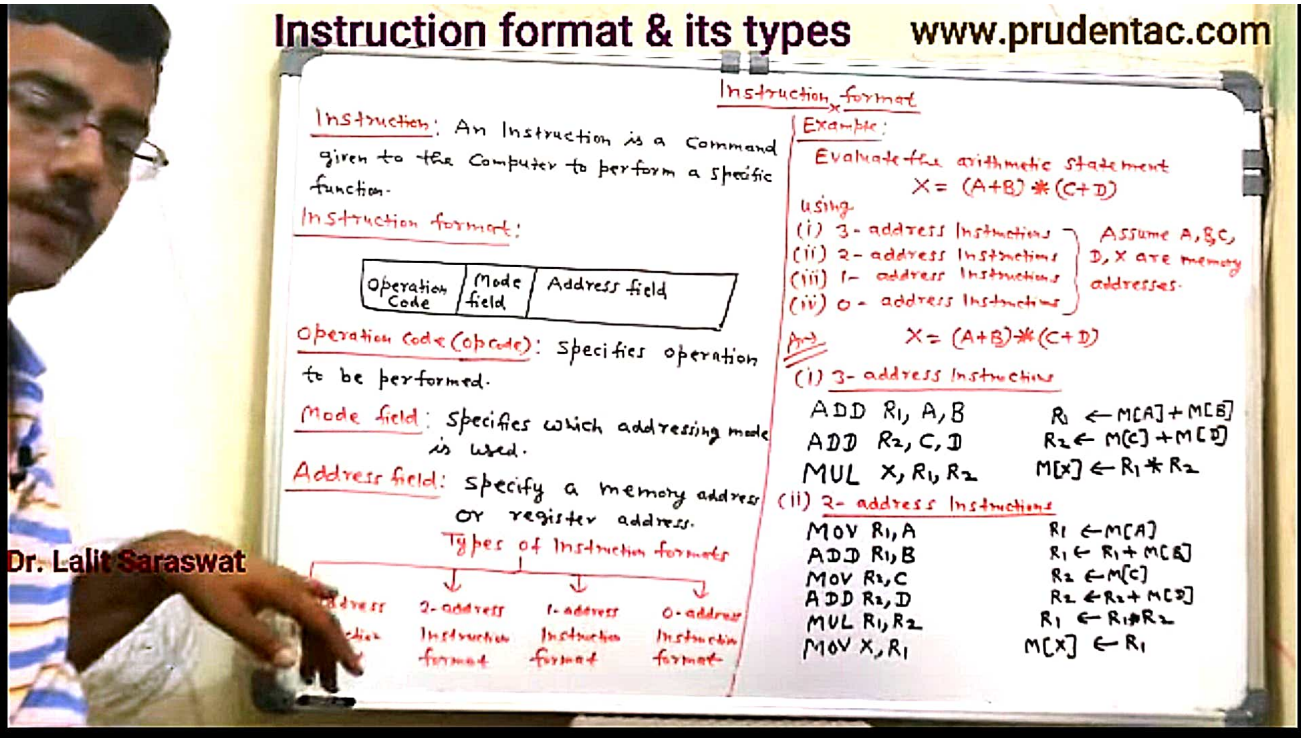

set associative mapping
www.prudentac.com
Cache Memory Mapping Techniques
Cache memory mapping" means how data is copied (mapped) from main memory
Cache memory.
Mapping Techniques:-
(3) Set Associative Mapping: It is a Combination
of direct mapping & associative mapping.
Set-Associative Mapping = Direct Mapping + Associative Mapping
In this, Cache memory is divided into set.
Set = group of blocks
Blocks = group of words
Cache memory Set No. = (main memory block No.) mod (No. of sets in cache memory)
No. of Sets in Cache = No. of Cache memory blocks / Set size
No. of blocks in Cache memory = No. of words in Cache memory / block size
No. of blocks in main memory = No. of words in main memory / block size
Let Set size = 2 (Two-way set associative)
Block size = 4 words
main memory size = 512 X 8
Cache memory size = 64 X 8
No. of main memory blocks = 512/4 = 128 blocks
No. of Cache memory blocks = 64/4 = 16 blocks
No. of Sets in Cache memory = 16/2 = 8 sets
Cache memory Set No.
0 mod 8 = 0
8 mod 8 = 0
16 mod 8 = 0
24 mod 8 = 0
1 mod 8 = 1
9 mod 8 = 1
17 mod 8 = 1
25 mod 8 = 1
Dr. Lalit Saraswat

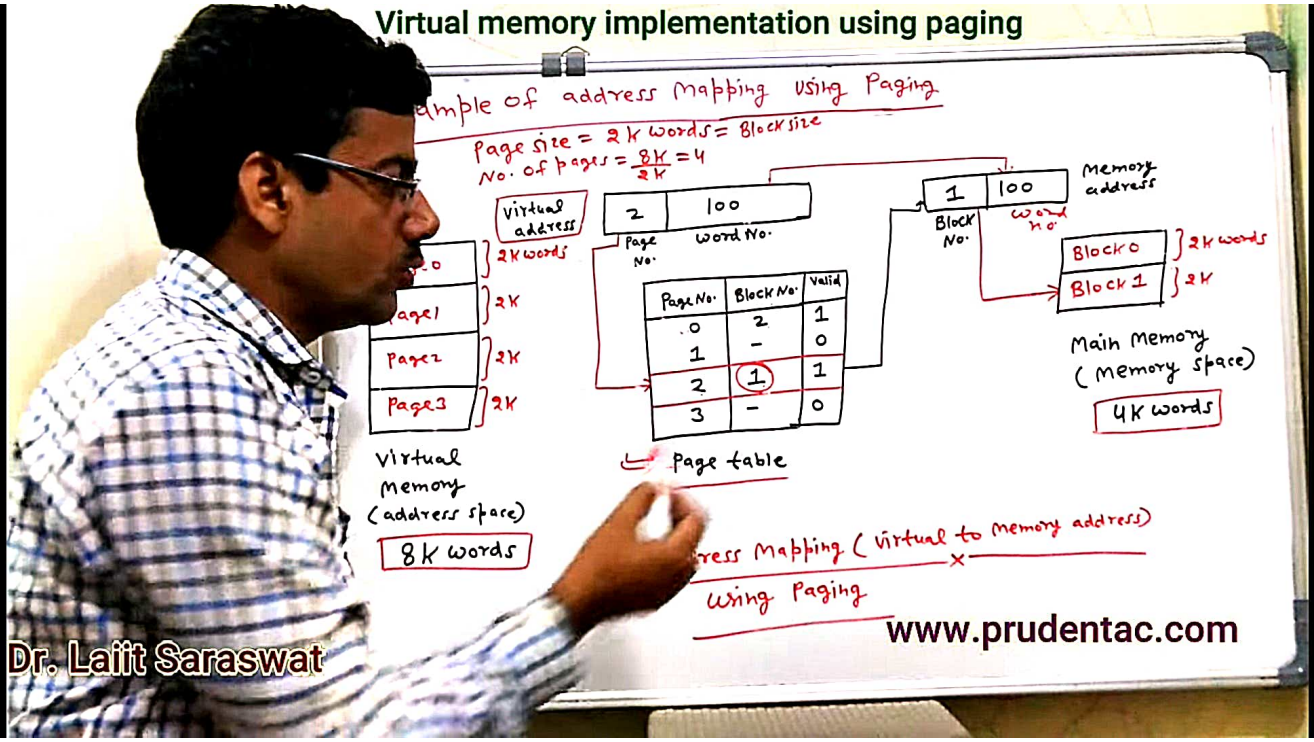
Virtual memory implementation using paging



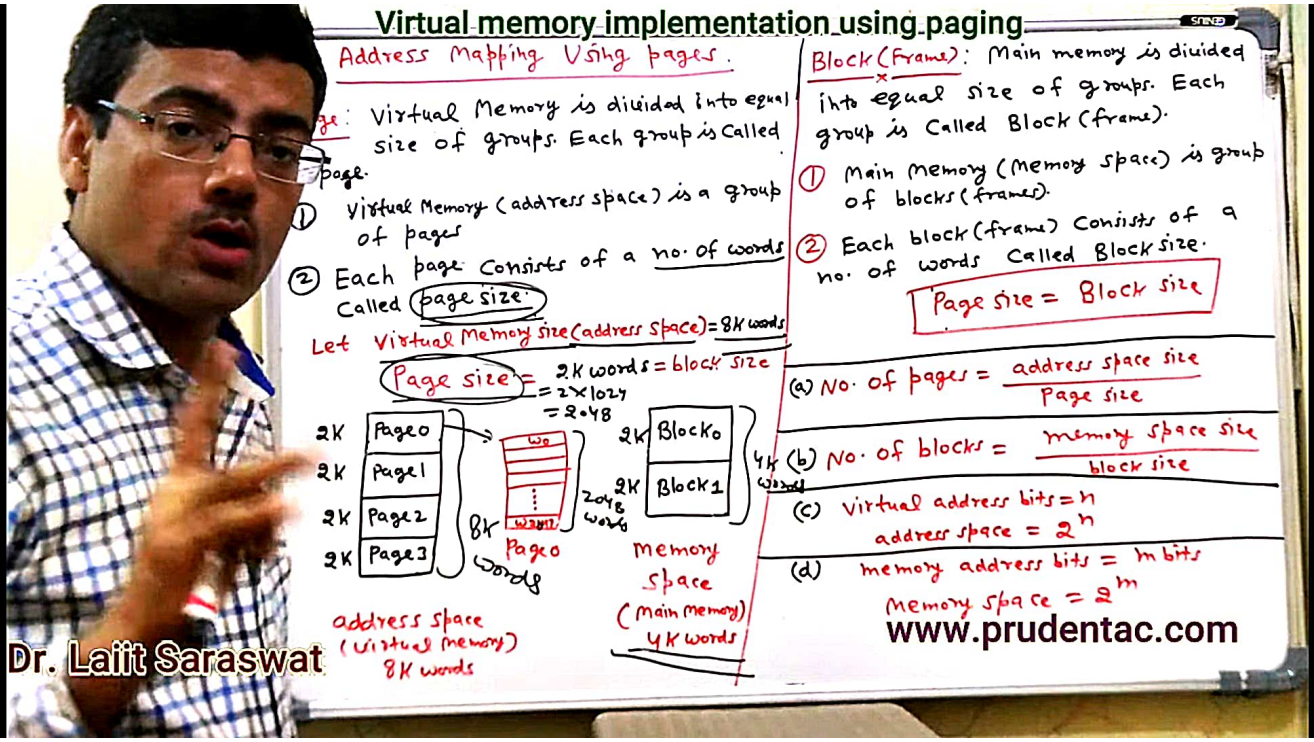

Virtual Memory Concept
Virtual Memory Concept (Computer Architecture)
rtual Memory : is a Concept that gives an illusion to the user that he has sufficient memory to execute any application/program of any size.
1 Virtual memory allows a no. of applications having total size more than the main memory size, to run at the same time.
2 Virtual memory is a simulated memory that is written to a file on the hard drive. This file is called pagefile or swap file. It is used by operating system to simulate physical RAM by using hard disk space.
3 In windows 1.0, 2.0 versions, there was no virtual memory, so we were not able to run a no. of applications due to run out of RAM space.
However from windows 3.0 onwards, Concept of virtual memory was introduced.
To implement it, a portion of hard drive is reserved by the system. This portion can be either a file or a separate partition. In windows, it is a file called pagefile.sys. In linux, a separate partition is used for virtual memory.
6 when the system needs more memory, it maps some of its memory address out to the hard disk drive. This extra memory does not actually exist
is the storage space on the disk drive.
Dr. Lalit Saraswat
www.prudentac.com


f3 | | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 2 | 2
f2 | | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 1
f1 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0
* * * * Hit * * * * * * Hit * * Hit
Reference String. 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0
Page Hit = 3
Page fault = 12
Miss
Page Replacement alg
1) FIFO
2) Optimal Page Repl
3) Least Recently Used

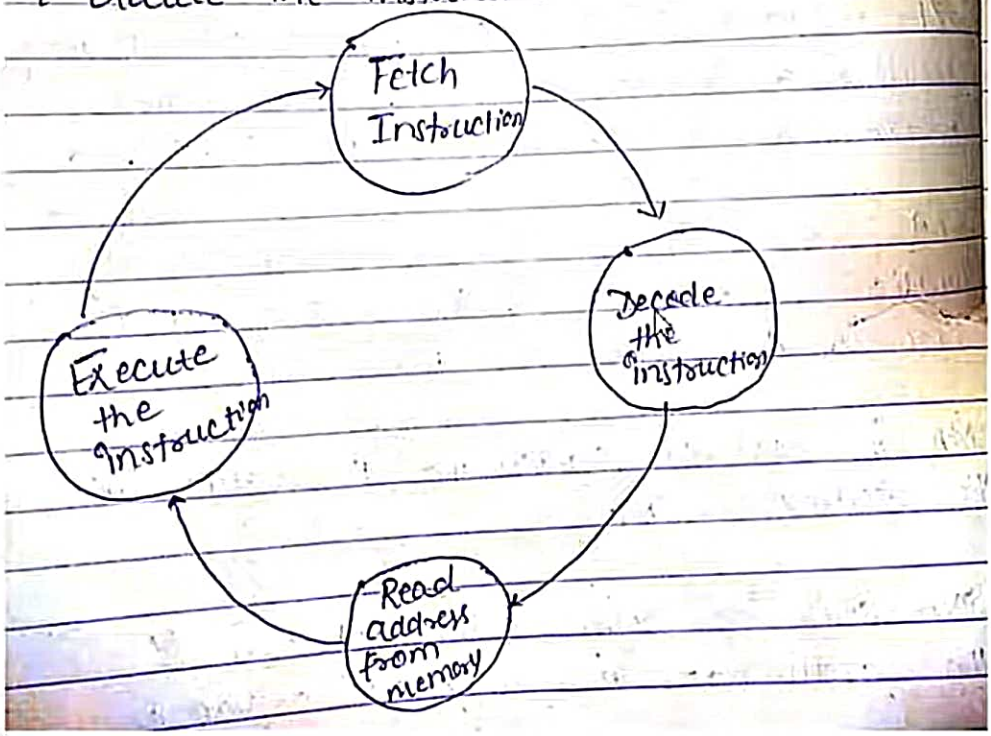

Instruction format & its types
www.prudentac.com
Dr. Lalit Saraswat
Instruction format
Instruction: An Instruction is a Command given to the Computer to perform a specific function.
Instruction format:
Operation Code | Mode field | Address field
Operation code (opcode): Specifies operation to be performed.
Mode field: Specifies which addressing mode is used.
Address field: Specify a memory address or register address.
Types of Instruction formats
2-address Instruction format
1-address Instruction format
0-address Instruction format
Example:
Evaluate the arithmetic statement
X = (A+B) * (C+D)
using
(i) 3-address Instructions
(ii) 2-address Instructions
(iii) 1-address Instructions
(iv) 0-address Instructions
Assume A, B, C, D, X are memory addresses.
Ans:
X = (A+B) * (C+D)
(i) 3-address Instructions
ADD R1, A, B
R1 ← M[A] + M[B]
ADD R2, C, D
R2 ← M[C] + M[D]
MUL X, R1, R2
M[X] ← R1 * R2
(ii) 2-address Instructions
MOV R1, A
R1 ← M[A]
ADD R1, B
R1 ← R1 + M[B]
MOV R2, C
R2 ← M[C]
ADD R2, D
R2 ← R2 + M[D]
MUL R1, R2
R1 ← R1 * R2
MOV X, R1
M[X] ← R1


Virtual memory implementation using paging
mple of address mapping using Paging
Page size = 2K words = Block size
No. of pages = 8K/2K = 4
Virtual address
2
100
Page No.
Word No.
2K words
Page1
Page2
Page3
Virtual memory (address space)
8K words
Page No. | Block No. | Valid
0 | 2 | 1
1 | - | 0
2 | 1 | 1
3 | - | 0
Page table
1
100
Block No.
Memory address
Block 0
Block 1
Main Memory (memory space)
4K words
ress Mapping (virtual to memory address)
using Paging
www.prudentac.com
Dr. Lalit Saraswat

Virtual memory implementation using paging



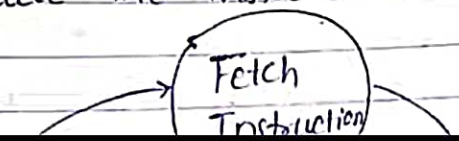
2. Decode the Instruction
3. Read the effective address from memory.
4. Execute the Instruction.

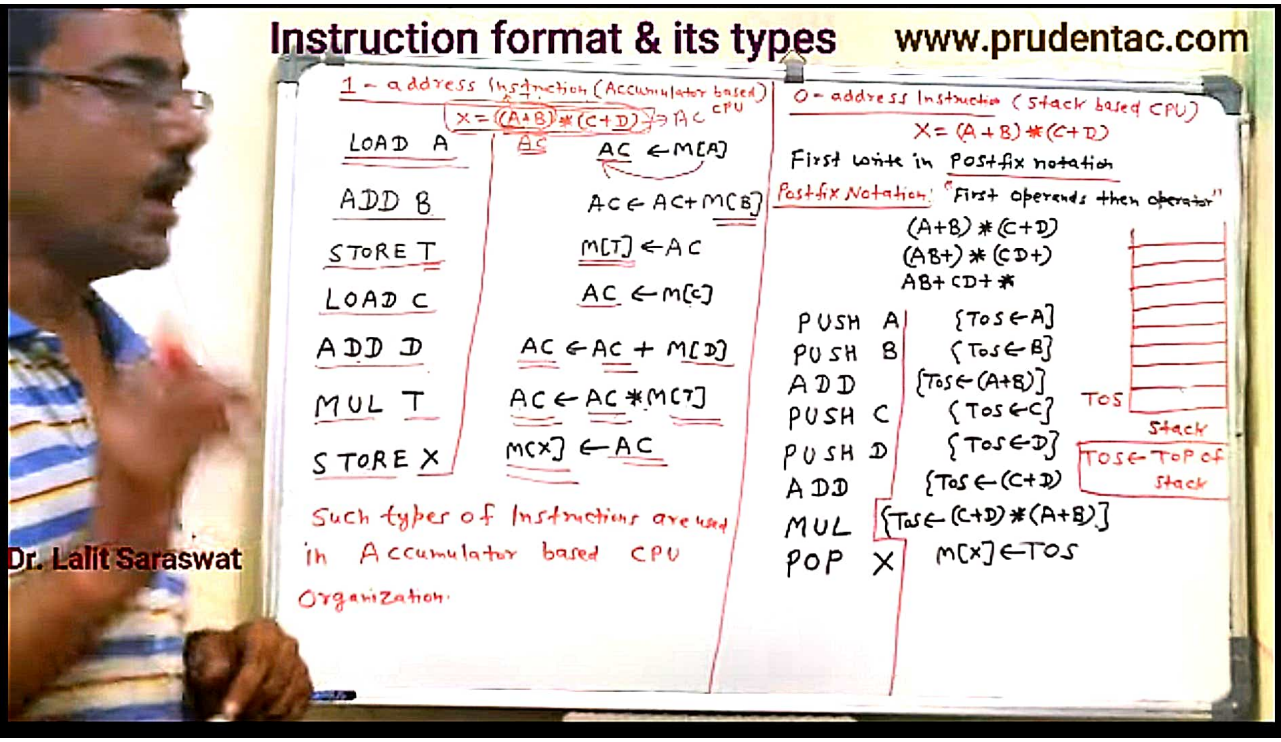


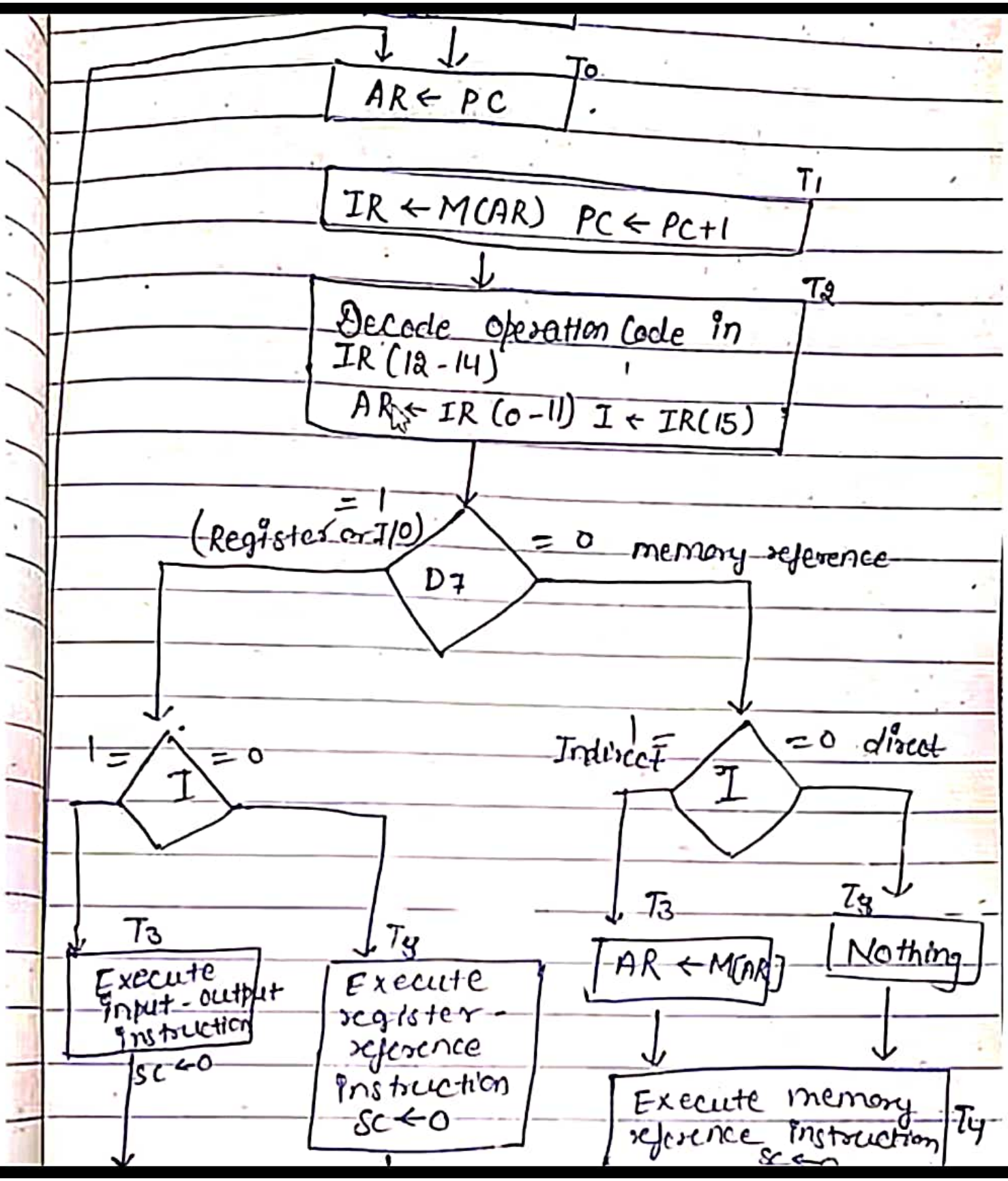
Instruction Cycle :→ A Program Consisting of the memory unit of the computer includes a series of instructions. The Program is implemented on the Computer by going through a cycle for each instruction.

In a basic computer, each instruction cycle consists of the following Phases :—

1. Fetch instruction from memory.
2. Decode the Instruction
3. Read the effective address from memory.
4. Execute the Instruction.




Instruction format & its types
www.prudentac.com
1 - address Instruction (Accumulator based CPU)
X = (A+B) * (C+D) → AC
LOAD A
AC
AC ← M[A]
ADD B
AC ← AC + M[B]
STORE T
M[T] ← AC
LOAD C
AC ← M[C]
ADD D
AC ← AC + M[D]
MUL T
AC ← AC * M[T]
STORE X
M[X] ← AC
Such types of Instructions are used in Accumulator based CPU Organization.
0 - address Instruction (Stack based CPU)
X = (A+B) * (C+D)
First write in Postfix notation
Postfix Notation: "First operands then operator"
(A+B) * (C+D)
(AB+) * (CD+)
AB+ CD+ *
PUSH A {TOS ← A}
PUSH B {TOS ← B}
ADD {TOS ← (A+B)}
PUSH C {TOS ← C}
PUSH D {TOS ← D}
ADD {TOS ← (C+D)}
MUL {TOS ← (C+D) * (A+B)}
POP X M[X] ← TOS
TOS
Stack
TOS - Top of stack
Dr. Lalit Saraswat


AR ← PC
T0
IR ← M(AR) PC ← PC+1
T1
Decode operation code in IR (12-14)
AR ← IR (0-11) I ← IR(15)
T2
D7
= 1 (Register or I/O)
= 0 memory reference
I
1 =
= 0
Indirect Ī
= 0 direct
T3
Execute input-output instruction
SC ← 0
Tg
Execute register-reference instruction
SC ← 0
AR ← M[AR]
Nothing
Execute memory reference instruction
SC ←
Ty

Step 2 → The address in MAR is Put on the address bus, now a Read Order is Provided by the Control unit on the Control bus and the result appears on the data bus is then copied in to the memory buffer register. Program Counter is incremented by one, to get ready for the next instruction. These two work can be carried out Simultaneously to save time.

step 3 → The content of the MBR is moved to the Instruction register (IR).

T1 : MAR ← PC

T2 : MB2 ← memory

PC ← PC + 1

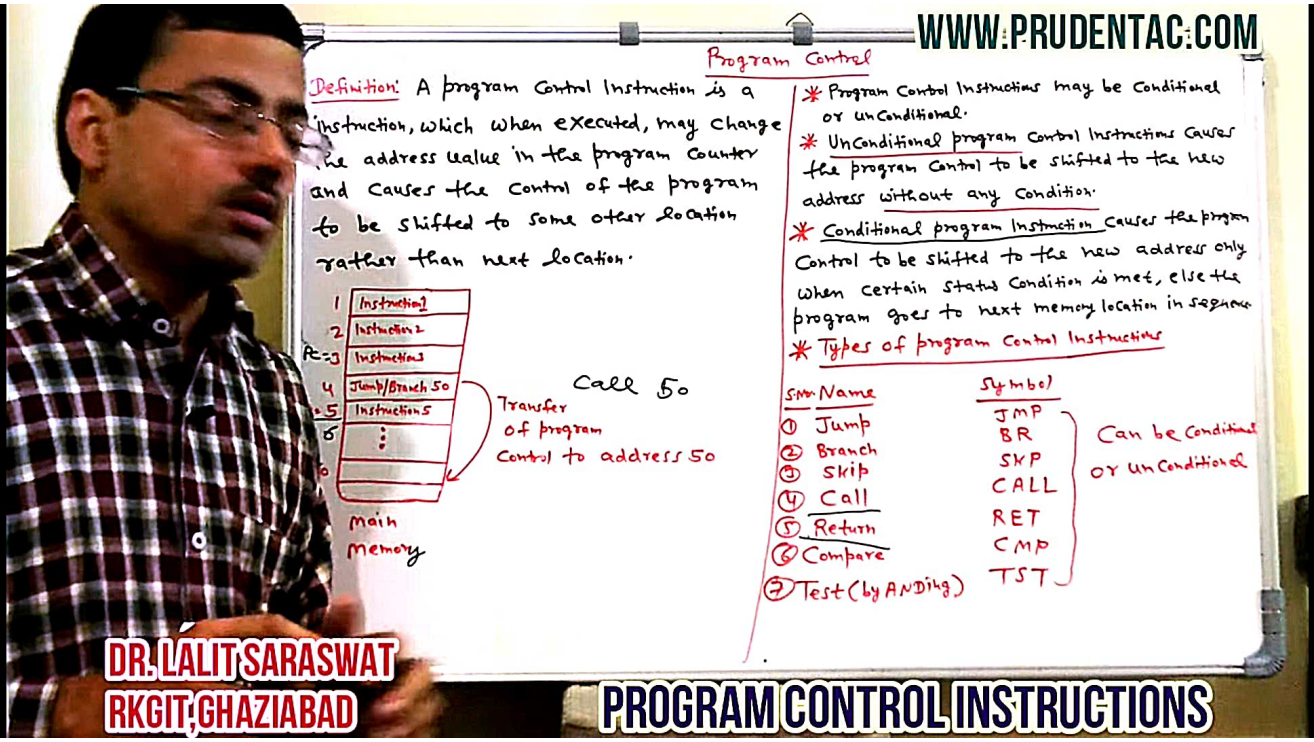
T3 : IR ← MBR.

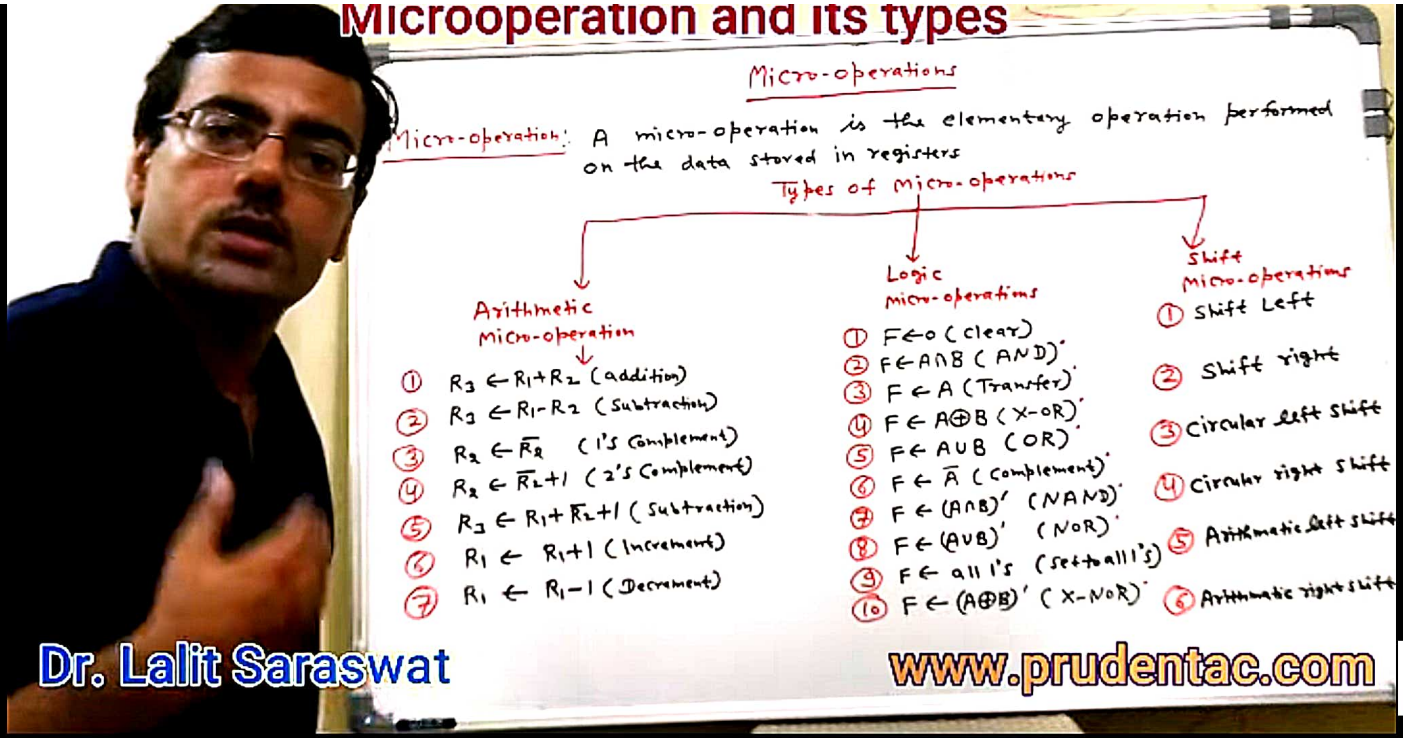
Fetch Cycle :→

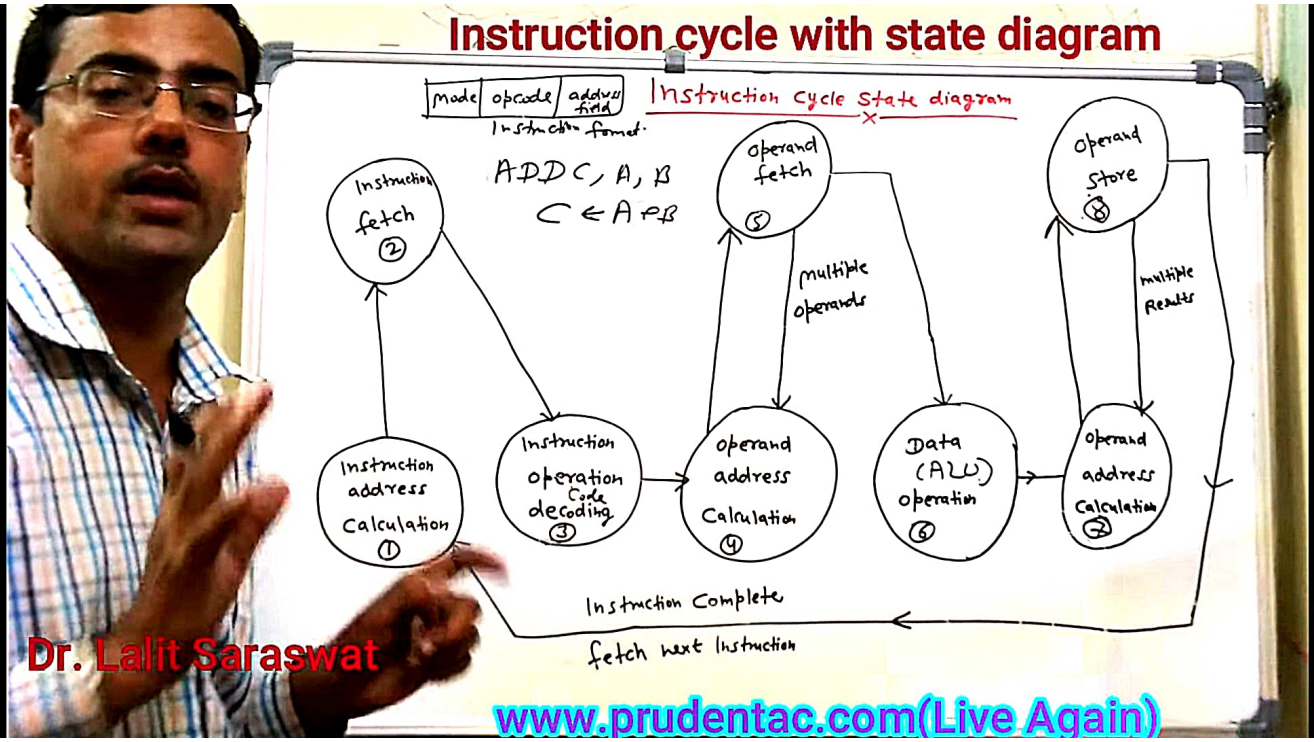
The address of the next instruction to execute is in the Program Counter (PC) at the beginning of the fetch cycle.

step 1 :- The address in the Program Counter is transferred to the Memory Address register (MAR).

Step 2 :→ The address in MAR is Put on the address bus, now a Read Order is Provided by the Control unit on the Control bus and the result appears on the data bus is then copied in to the memory buffer register. Program Counter is incremented by One, to get ready for the next instruction. These two work can be carried out Simultaneously to save time.


WWW.PRUDENTAC.COM
Program Control
Definition: A program control instruction is a instruction, which when executed, may change the address value in the program counter and causes the control of the program to be shifted to some other location rather than next location.
1 Instruction1
2 Instruction2
PC=3 Instruction3
4 Jump/Branch 50
5 Instruction5
6
Call 50
Transfer of program Control to address 50
main memory
* Program Control Instructions may be Conditional or unConditional.
* UnConditional program Control Instructions causes the program Control to be shifted to the new address without any Condition.
* Conditional program Instruction causes the program Control to be shifted to the new address only when certain status Condition is met, else the program goes to next memory location in sequence
* Types of program Control Instructions
Sno. Name
1 Jump
2 Branch
3 Skip
4 Call
5 Return
6 Compare
7 Test (by ANDing)
Symbol
JMP
BR
SKP
CALL
RET
CMP
TST
Can be Conditional or unConditional
DR. LALIT SARASWAT
RKGIT, GHAZIABAD
PROGRAM CONTROL INSTRUCTIONS


Microoperation and its types
Micro-operations
Micro-operation: A micro-operation is the elementary operation performed on the data stored in registers
Types of micro-operations
Arithmetic micro-operation
1. $R_3 \leftarrow R_1 + R_2$ (addition)
2. $R_3 \leftarrow R_1 - R_2$ (Subtraction)
3. $R_2 \leftarrow \overline{R_2}$ (1's Complement)
4. $R_2 \leftarrow \overline{R_2} + 1$ (2's Complement)
5. $R_3 \leftarrow R_1 + \overline{R_2} + 1$ (Subtraction)
6. $R_1 \leftarrow R_1 + 1$ (Increment)
7. $R_1 \leftarrow R_1 - 1$ (Decrement)
Logic micro-operations
1. $F \leftarrow 0$ (clear)
2. $F \leftarrow A \wedge B$ (AND)
3. $F \leftarrow A$ (Transfer)
4. $F \leftarrow A \oplus B$ (X-OR)
5. $F \leftarrow A \vee B$ (OR)
6. $F \leftarrow \overline{A}$ (Complement)
7. $F \leftarrow (A \wedge B)'$ (NAND)
8. $F \leftarrow (A \vee B)'$ (NOR)
9. $F \leftarrow$ all 1's (Set to all 1's)
10. $F \leftarrow (A \oplus B)'$ (X-NOR)
Shift micro-operations
1. Shift Left
2. Shift right
3. Circular left shift
4. Circular right shift
5. Arithmetic left shift
6. Arithmetic right shift
Dr. Lalit Saraswat
www.prudentac.com


Instruction cycle with state diagram
mode
opcode
address field
Instruction format
Instruction Cycle State diagram
ADD C, A, B
C ← A + B
Instruction fetch
2
Operand fetch
5
Operand Store
8
multiple operands
multiple Results
Instruction address Calculation
1
Instruction operation Code decoding
3
operand address Calculation
4
Data (ALU) operation
6
operand address Calculation
7
Instruction Complete
fetch next Instruction
Dr. Lalit Saraswat
www.prudentac.com(Live Again)

Design of Hardwired control unit

Design of microprogran

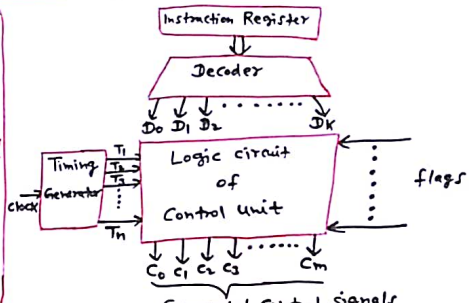
Hardwired Control Unit

Definition: In hardwired control unit, the control signals are generated using logic circuits (gates, flip-flops & decoder) in the hardware. Hardwired control unit is faster because control signals are generated using logic circuits.

Block diagram of Hardwired Control unit consists of:

1. **Instruction Register:** Used to store the instruction fetched from the memory.
2. **Decoder:** Used to decode (Interpret) the operation code of the instruction.
3. **Clock:** Used to generate a sequence of timing signals using timing generator.
4. **flags:** Condition codes / status flags are used to specify the status of previous ALU operation.

Generation of Control signals: An example of how the control signals are generated is as shown:



Consider C_5:

MBR ← memory

Cycle	Micro-operations	Control Signals
Fetch PQ=00	t_1: MAR ← PC	C_2
	t_2: MBR ← memory	C_5
Indirect PQ=01	t_2: MAR ← IR(address)	C_8
	t_3: MBR ← memory	C_5
Interrupt cycle PQ=10	t_1: MBR ← PC	C_4
	t_3: MBR ← memory	C_5

execute cycle PQ=11

$C_5 = P'Q't_2 + P'Qt_2 + PQ't_3$

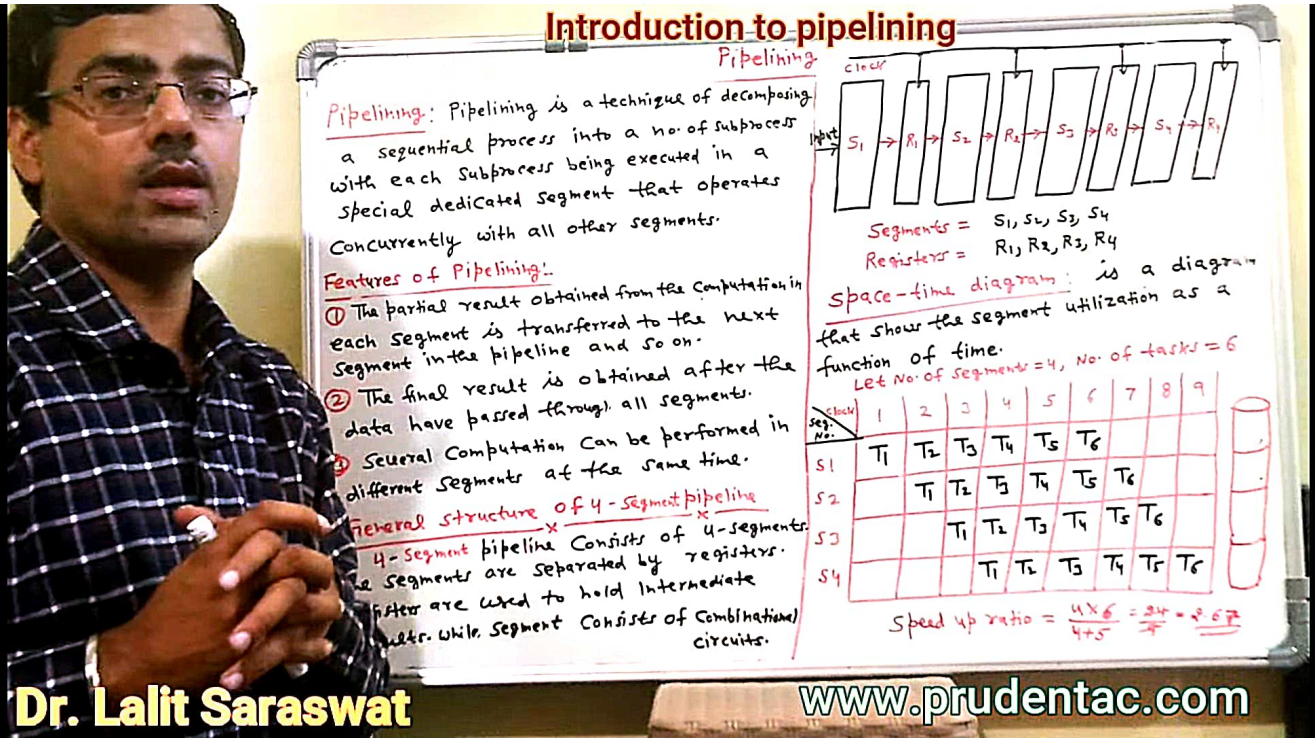
Let only LDA, ADD, AND instruction read from memory.

$$C_5 = P'Q't_2 + P'Qt_2 + PQ't_3 + PQ(LDA+ADD+AND)t_2$$

Dr. Lalit Saraswat

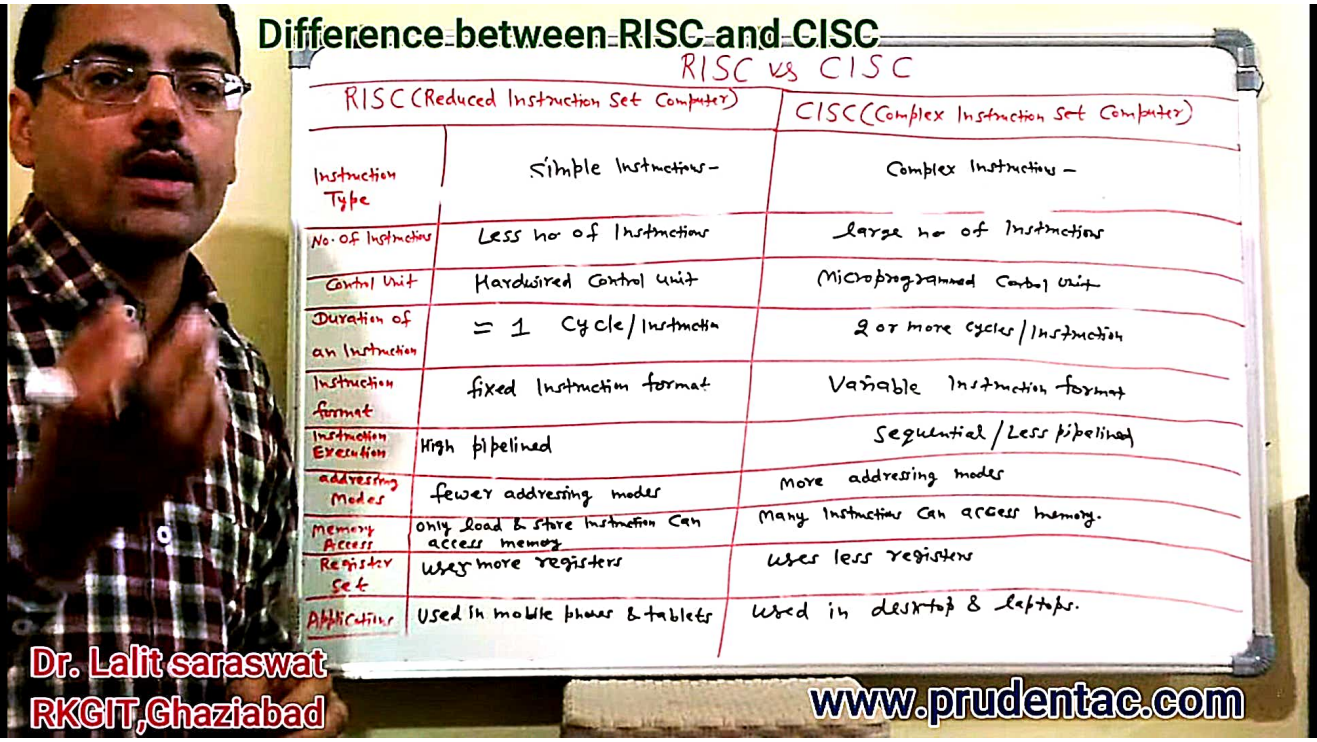
www.prudentac.com

Introduction to pipelining



Dr. Lalit Saraswat

www.prudentac.com

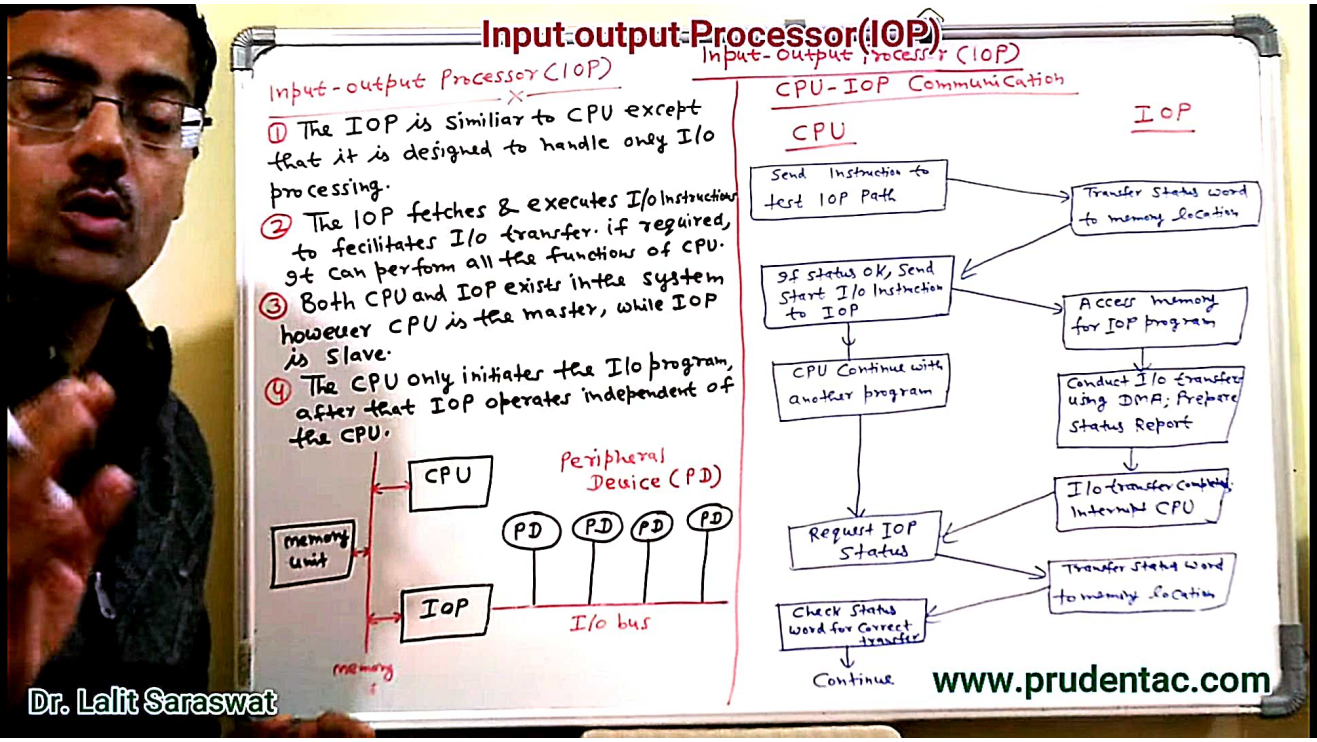

Difference between RISC and CISC
RISC vs CISC
RISC (Reduced Instruction Set Computer)
CISC (Complex Instruction Set Computer)
Instruction Type
Simple Instructions –
Complex Instructions –
No. of Instructions
Less no of Instructions
large no of Instructions
Control Unit
Hardwired Control Unit
Microprogrammed Control Unit
Duration of an Instruction
= 1 Cycle/Instruction
2 or more cycles/Instruction
Instruction format
fixed Instruction format
Variable Instruction format
Instruction Execution
High pipelined
Sequential/Less pipelined
addressing Modes
fewer addressing modes
more addressing modes
Memory Access
only load & store instruction can access memory
Many Instructions can access memory.
Register Set
uses more registers
uses less registers
Application
Used in mobile phones & tablets
Used in desktop & laptops.
Dr. Lalit saraswat
RKGIT, Ghaziabad
www.prudentac.com

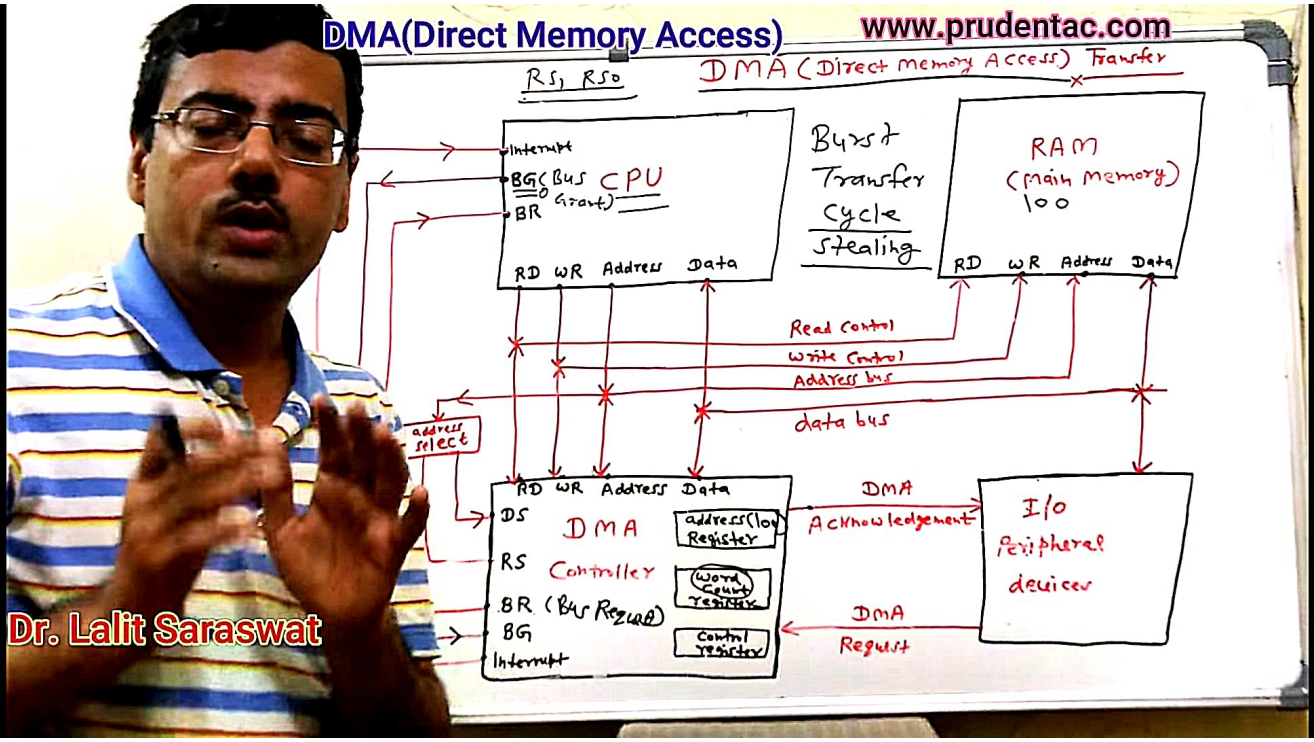
5.	It is difficult to handle Complex instructions.	It is easier to handle Complex instructions.
6.	Its design is complex	Its design is simple.
7.	It is difficult to perform instruction decoding.	Less difficult to perform instruction decoding.
8.	The architecture and instructions set are not specified.	The architecture and instructions set is specified.

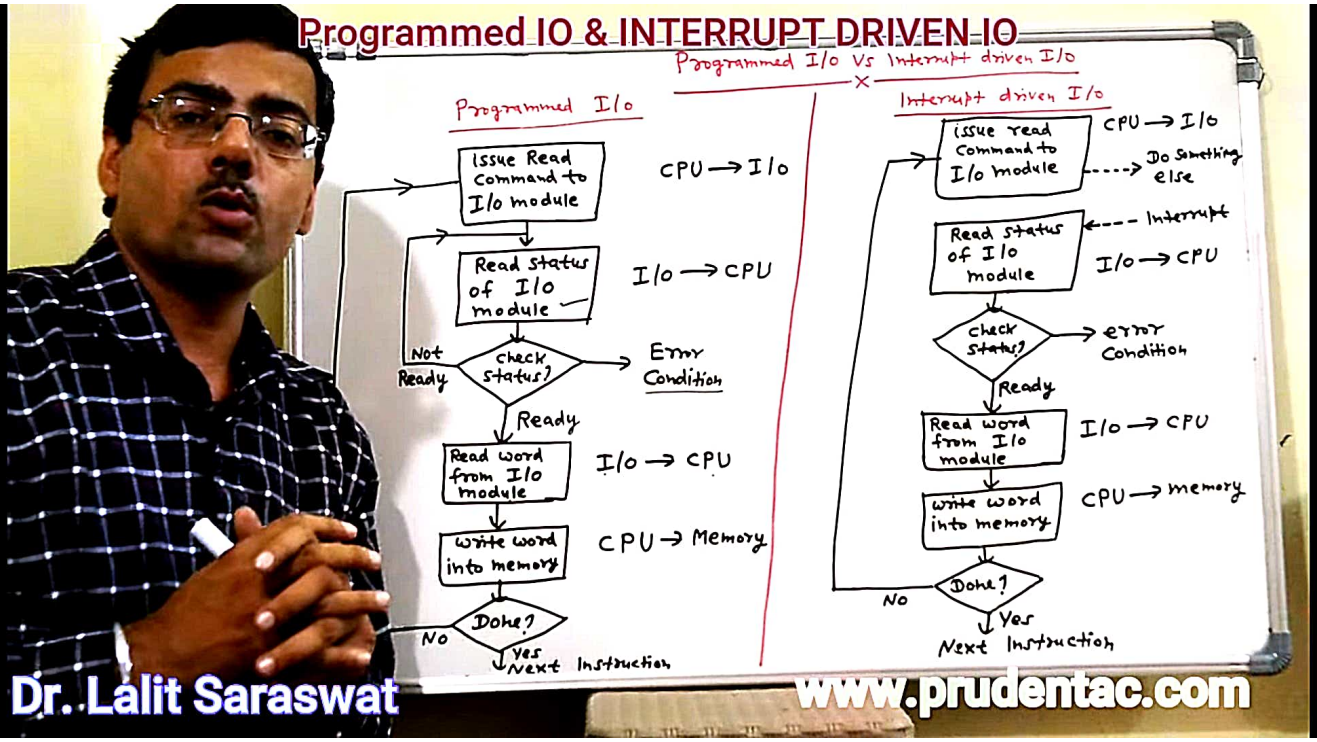
2.	Operations speed is fast.	Operations speed is slow.
3.	To do modifications, the entire unit should be redesigned.	Modifications can be implemented by changing the microinstructions in the control memory.
4.	More costly to implement.	Less costly to implement.
5.	It is difficult to handle complex instructions.	It is easier to handle complex instructions.
6.	Its design is complex	It design is simple

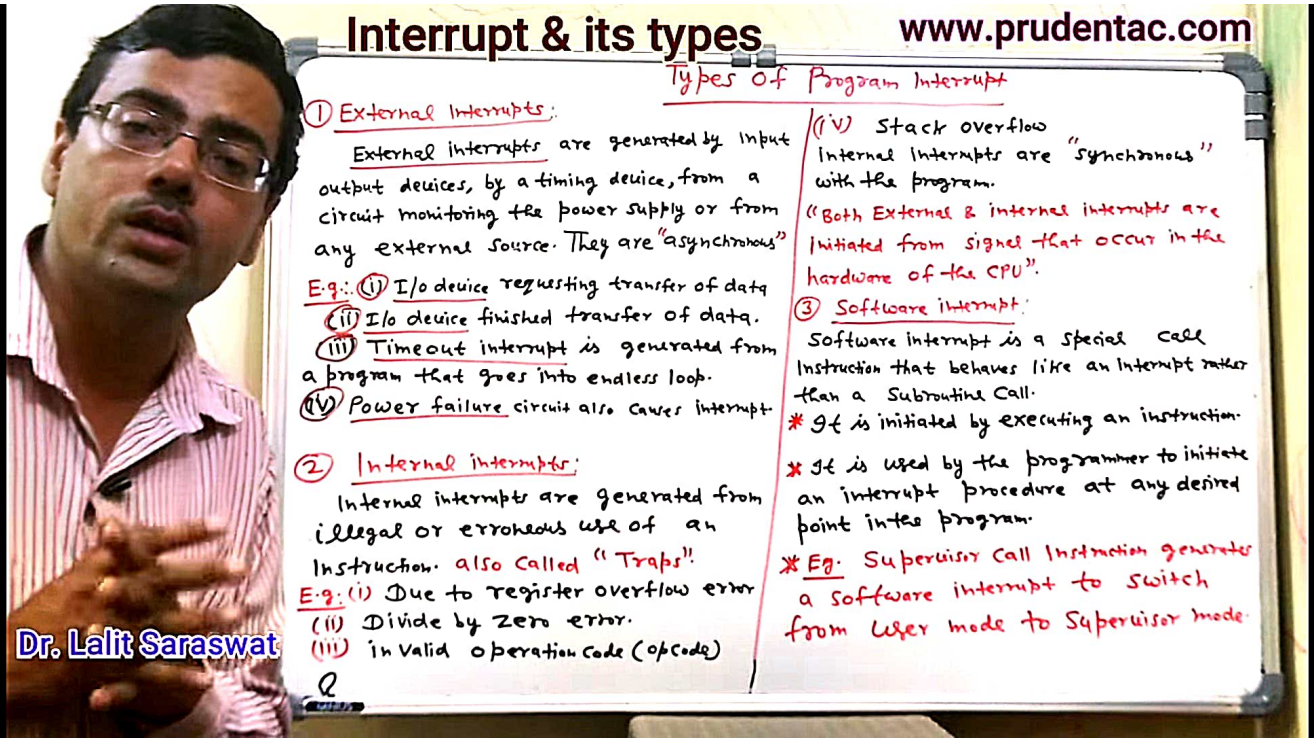
⇒	HARDWIRED CONTROL UNIT	MICROPROGRAMMED CU
1.	It generates the control signals needed for the processor using logic circuits.	It generates the control signals with the help of micro instructions stored in control memory.
2.	Operations speed is fast.	Operations speed is slow.
3.	To do modifications, the entire unit should be redesign	Modifications can be implemented by changing the micropro

Input-output-Processor(IOP)




DMA(Direct Memory Access)
www.prudentac.com
RS1 RS0
DMA (Direct Memory Access) Transfer
CPU
Interrupt
BG (Bus Grant)
BR
RD WR Address Data
Burst Transfer
Cycle Stealing
RAM
(main Memory)
100
RD WR Address Data
Read control
write control
Address bus
data bus
address select
RD WR Address Data
DS
RS
BR (Bus Request)
BG
Interrupt
DMA Controller
address/log Register
Word Count register
Control register
DMA Acknowledgement
DMA Request
I/O peripheral devices
Dr. Lalit Saraswat


Programmed IO & INTERRUPT DRIVEN IO
Programmed I/O Vs Interrupt driven I/O
Programmed I/O
Issue Read Command to I/o module
CPU → I/o
Read status of I/o module
I/o → CPU
Not Ready
check status?
Error Condition
Ready
Read word from I/o module
I/o → CPU
write word into memory
CPU → Memory
Done?
No
Yes
Next Instruction
Interrupt driven I/o
Issue read Command to I/o module
CPU → I/o
Do Something else
Interrupt
Read status of I/o module
I/o → CPU
check status?
error Condition
Ready
Read word from I/o module
I/o → CPU
write word into memory
CPU → memory
Done?
No
Yes
Next Instruction
Dr. Lalit Saraswat
www.prudentac.com


Interrupt & its types
www.prudentac.com
Types of Program Interrupt
① External Interrupts:
External interrupts are generated by input output devices, by a timing device, from a circuit monitoring the power supply or from any external source. They are "asynchronous"
E.g.: (i) I/o device requesting transfer of data
(ii) I/o device finished transfer of data.
(iii) Timeout interrupt is generated from a program that goes into endless loop.
(iv) Power failure circuit also causes interrupt.
② Internal interrupts:
Internal interrupts are generated from illegal or erroneous use of an instruction. also called "Traps".
E.g: (i) Due to register overflow error
(ii) Divide by zero error.
(iii) Invalid operation code (opcode)
(iv) Stack overflow
Internal interrupts are "synchronous" with the program.
"Both External & internal interrupts are initiated from signal that occur in the hardware of the CPU".
③ Software interrupt:
Software interrupt is a special call instruction that behaves like an interrupt rather than a Subroutine Call.
* It is initiated by executing an instruction.
* It is used by the programmer to initiate an interrupt procedure at any desired point into program.
* Eg. Supervisor call instruction generates a software interrupt to switch from user mode to Supervisor mode.
Dr. Lalit Saraswat

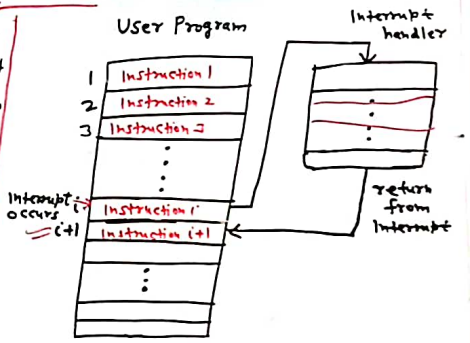
Interrupt & its types

www.prudentac.com

Program Interrupt

Definition: Program Interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request.

* Control returns to the original program after the service program is executed.
* When an interrupt is initiated, the state of CPU containing following information is saved.
  1. Content of Program Counter (PC)
  2. Content of all CPU registers
  3. Content of status bit conditions (PSW)
* CPU does not responds to an interrupt until the execution of currently running instruction ends.



Transfer of Program Control via Interrupts

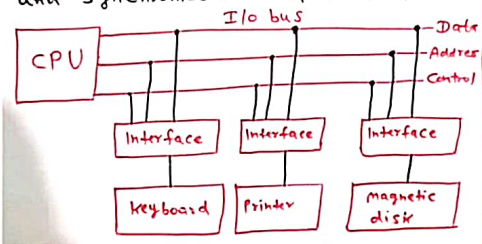
* Before going to next fetch cycle, it checks for an interrupt signal, if there is an interrupt
  1. Save CPU state (PC, CPU registers, PSW) in memory stack.
  2. PC ← Interrupt branch address
     PSW ← status bits of Interrupt service program
  3. Last instruction of service program is return from Interrupt.
  4. PSW ← old PSW, PC ← old PC, CPU register ← old CPU register

Dr. Lalit Saraswat

Input Output Interface

Input- output Interface

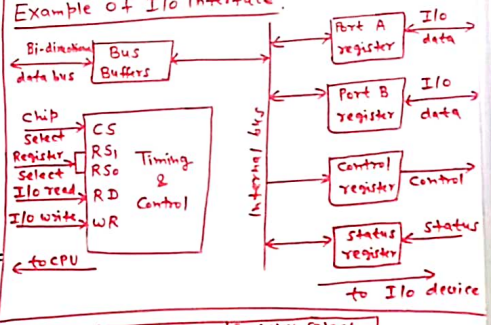
I/o Interface: It is a hardware circuit used between CPU and I/o devices to supervise and synchronize all input and output transfer.



Requirement (Need of I/o Interface)

1. I/o devices are electromechanical & electro-magnetic while CPU & memory are electronic devices. So they have different signals. So conversion of signals is required.
2. Data transfer rate of CPU is faster as compared to I/o devices. So synchronization is needed.
3. Data format of I/o devices are different from word format of CPU and memory.
4. Operating modes of I/o devices are different from each other and so each I/o device must be controlled so that it will not disturb the operation of other I/o devices to CPU.

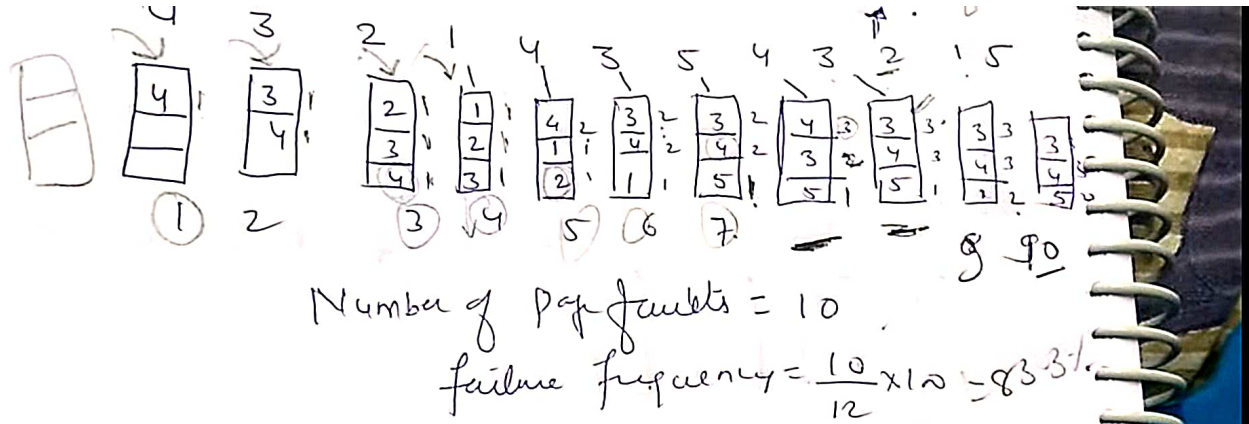
Example of I/o Interface:



CS	RS1	RS0	Register Select
0	x	x	None
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status Register

Dr. Lalit Saraswat

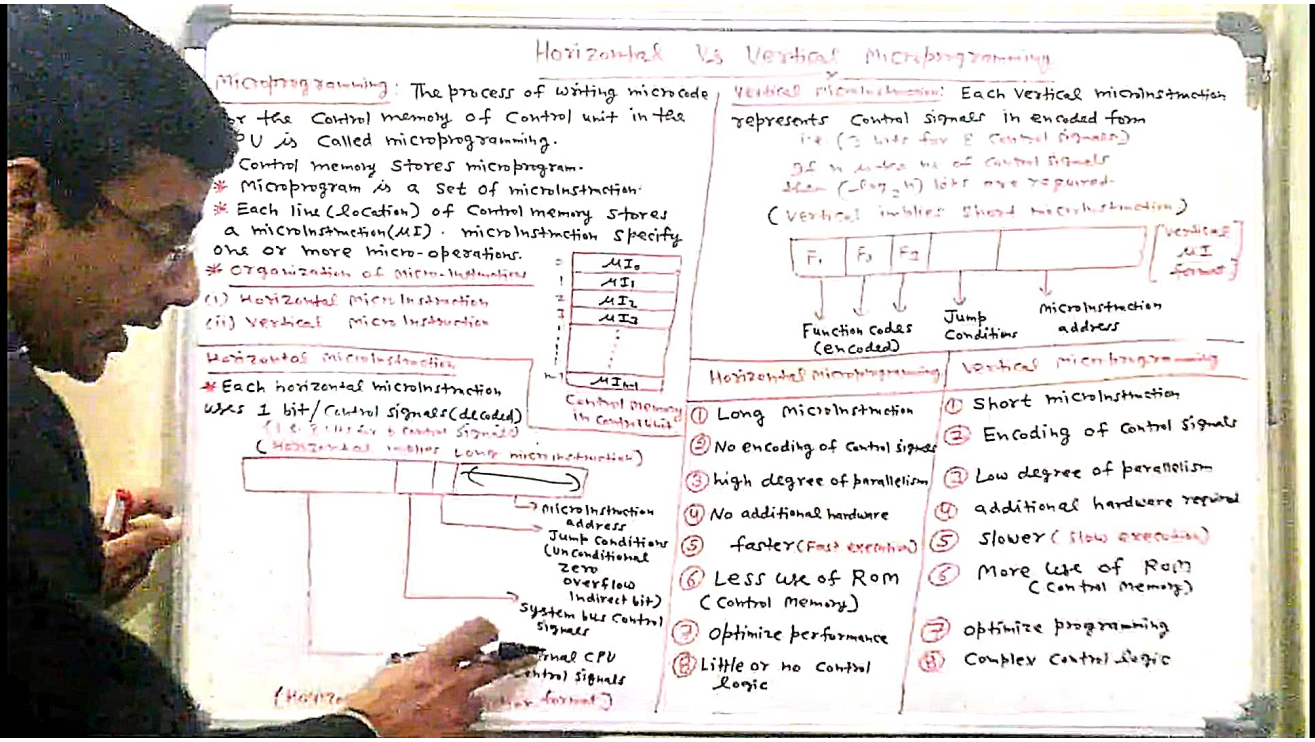
www.prudentac.com



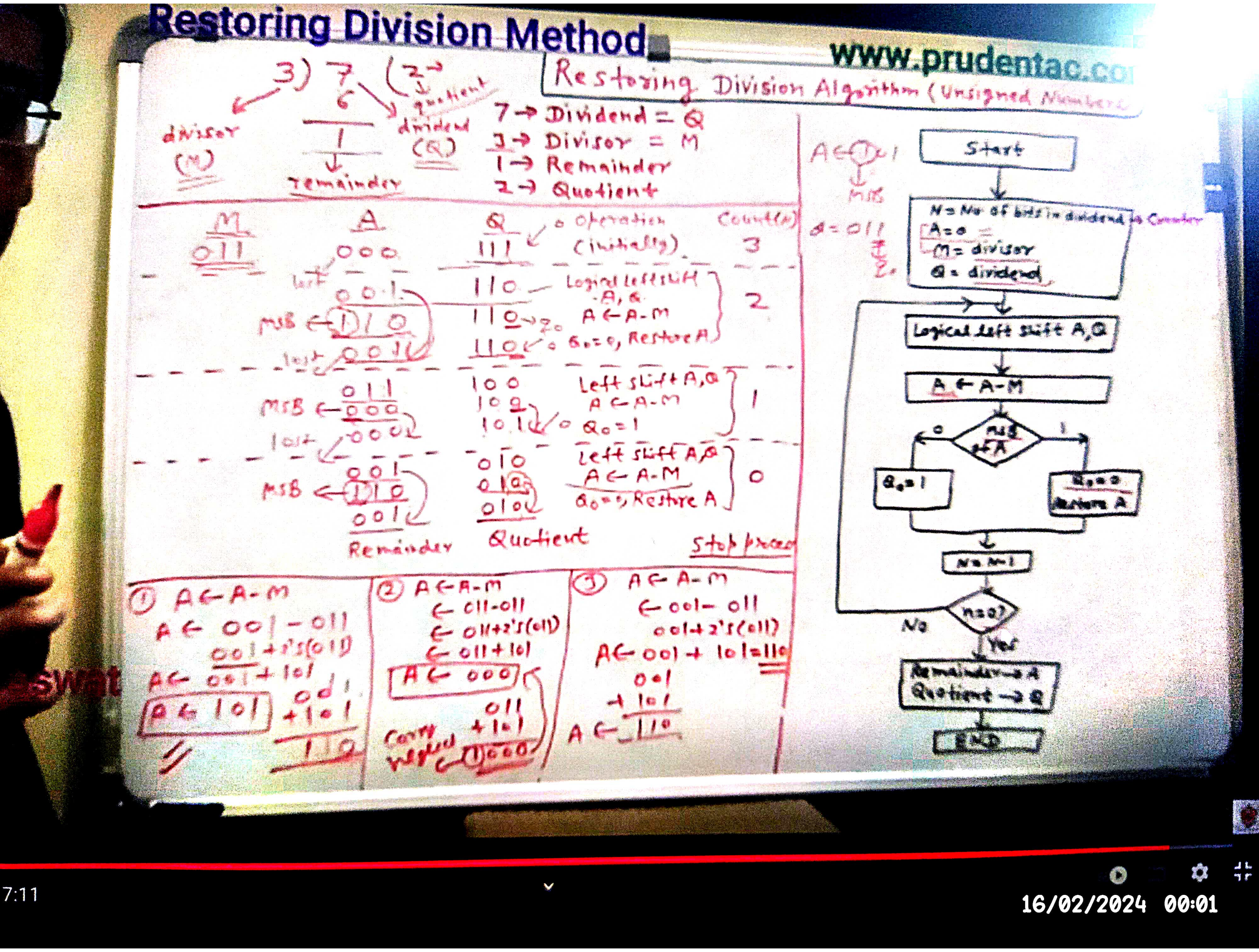
Number of page faults = 10

failure frequency = $\frac{10}{12} \times 100 = 83.3\%$

- In LFU we check old page as well as frequency of page


Horizontal Vs Vertical Microprogramming
Microprogramming: The process of writing microcode for the control memory of control unit in the CPU is called microprogramming.
Control memory stores microprogram.
Microprogram is a set of microInstruction.
Each line (location) of control memory stores a microInstruction(MI). microInstruction specify one or more micro-operations.
Organization of micro-Instructions
(i) Horizontal microInstruction
(ii) Vertical microInstruction
Control memory in control unit
Horizontal microInstruction
Each horizontal microInstruction uses 1 bit/control signals (decoded)
microInstruction address
Jump conditions (Unconditional zero overflow Indirect bit)
System bus control signals
Vertical microInstruction: Each vertical microInstruction represents control signals in encoded form
Function codes (encoded)
Jump Conditions
microInstruction address
Horizontal microprogramming
1 Long microInstruction
2 No encoding of control signals
3 high degree of parallelism
4 No additional hardware
5 faster (Fast execution)
6 Less use of Rom (Control memory)
7 optimize performance
8 Little or no control logic
Vertical microprogramming
1 Short microInstruction
2 Encoding of control signals
3 Low degree of parallelism
4 additional hardware required
5 slower (slow execution)
6 More use of Rom (Control memory)
7 optimize programming
8 Complex control logic

9.	There are no control memory usage.	Uses control memory.
10.	Used in processors that use a simple instruction set known as the Reduced instruction set computers (RISC).	Used in processors based on a complex instruction set known as complex instruction set computer (CISC)

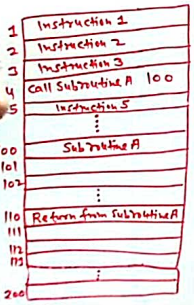

Restoring Division Method
www.prudentac.co
Restoring Division Algorithm (Unsigned Numbers
7 → Dividend = Q
3 → Divisor = M
1 → Remainder
2 → Quotient
divisor (M)
dividend (Q)
remainder
M
A
Q
Count(n)
011
000
111
3
Logical left shift A, Q
A ← A-M
Q₀=0, Restore A
2
Left shift A, Q
A ← A-M
Q₀=1
1
Left shift A, Q
A ← A-M
Q₀=0, Restore A
0
Remainder
Quotient
Start
N = No. of bits in dividend
A = 0
M = divisor
Q = dividend
Logical left shift A, Q
A ← A-M
MSB of A
Q₀=1
Q₀=0
Restore A
N = N-1
N=0?
No
Yes
Remainder → A
Quotient → Q
END
A ← A-M
A ← 001 - 011
A ← 101
A ← 000
A ← 110
7:11
16/02/2024 00:01

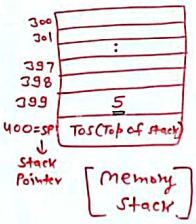
Subroutine call and Return

Subroutine Call & Return

Subroutine: A Subroutine (function) is a Subprogram in the main program that perform a specific task.

① During the execution of main program, a Subroutine may be called at various points in the main program.





② A Call Subroutine Instruction consists of an operation Code along with an address that specify the begining address of a Subroutine.

③ This Instruction is executed by performing two operations:

(i) The address of next Instruction (PC) is Stored in memory stack.

(ii) The PC (Program Counter) is loaded with the starting address of the Subroutine.

④ The last Instruction from Subroutine Causes a return to the address stored in stack.

Subroutine Call (List of micro-operations)

$SP \leftarrow SP-1$

$M[SP] \leftarrow PC$

$PC \leftarrow$ Starting address of Subroutine

Return from Subroutine (List of micro-operations)

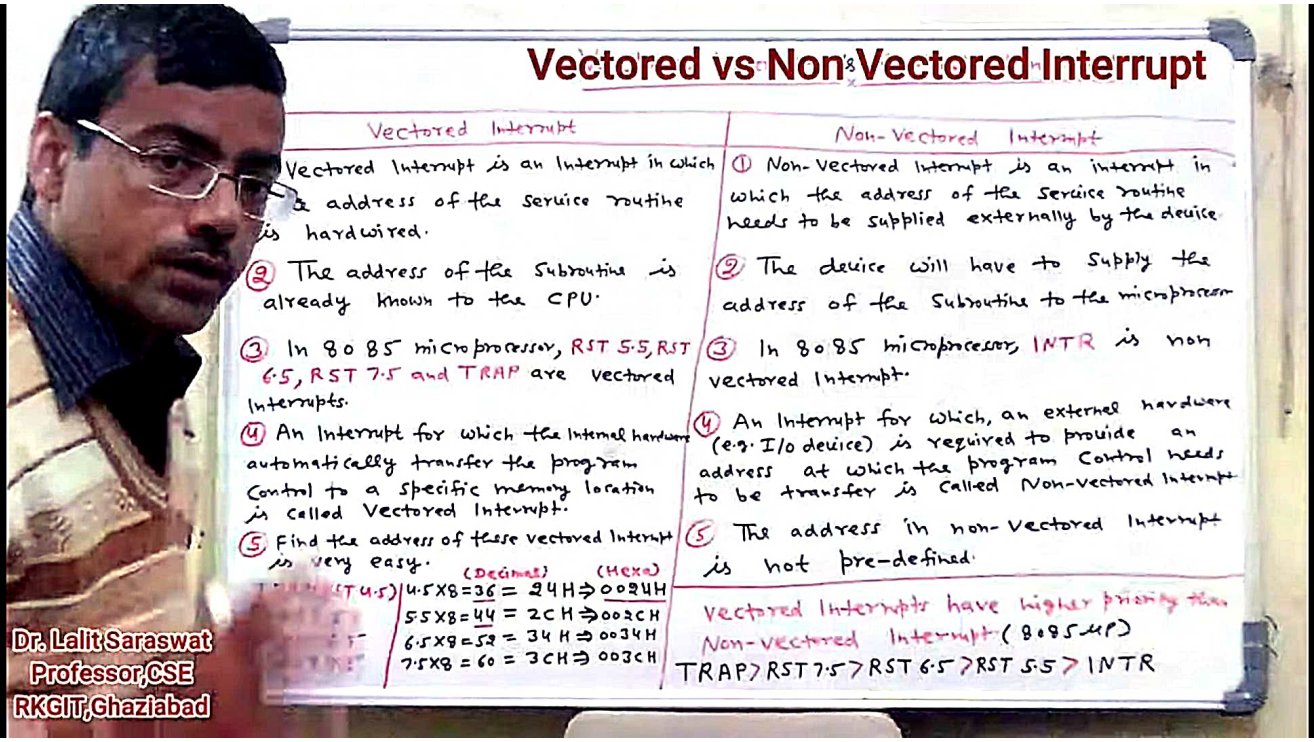
$PC \leftarrow M[SP]$

$SP \leftarrow SP+1$

Recursive Subroutine: A recursive Subroutine is a Subroutine that Calls itself again and again.

Dr. Lalit Saraswat

www.prudentac.com


Vectored vs Non Vectored Interrupt
Vectored Interrupt
Vectored Interrupt is an Interrupt in which address of the service routine is hardwired.
The address of the subroutine is already known to the CPU.
In 8085 microprocessor, RST 5.5, RST 6.5, RST 7.5 and TRAP are vectored Interrupts.
An Interrupt for which the internal hardware automatically transfer the program control to a specific memory location is called Vectored Interrupt.
Find the address of these vectored Interrupt is very easy.
(Decimal) (Hexa)
4.5X8=36 = 24H ⇒ 0024H
5.5X8=44 = 2CH ⇒ 002CH
6.5X8=52 = 34H ⇒ 0034H
7.5X8 = 60 = 3CH ⇒ 003CH
Non-Vectored Interrupt
Non-vectored Interrupt is an interrupt in which the address of the service routine needs to be supplied externally by the device.
The device will have to supply the address of the subroutine to the microprocessor
In 8085 microprocessor, INTR is non vectored Interrupt.
An Interrupt for which, an external hardware (e.g. I/O device) is required to provide an address at which the program control needs to be transfer is called Non-vectored Interrupt
The address in non-vectored Interrupt is not pre-defined.
Vectored Interrupts have higher priority than Non-vectored Interrupt (8085-μP)
TRAP > RST 7.5 > RST 6.5 > RST 5.5 > INTR
Dr. Lalit Saraswat
Professor, CSE
RKGIT, Ghaziabad