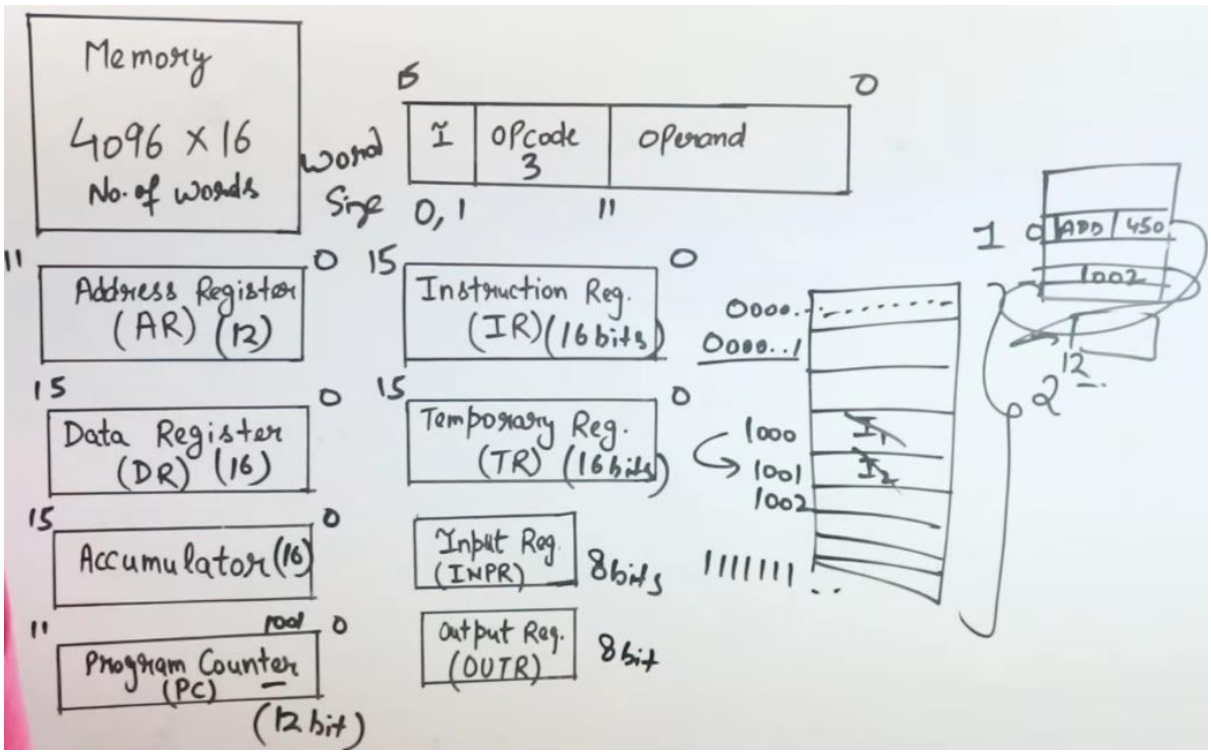
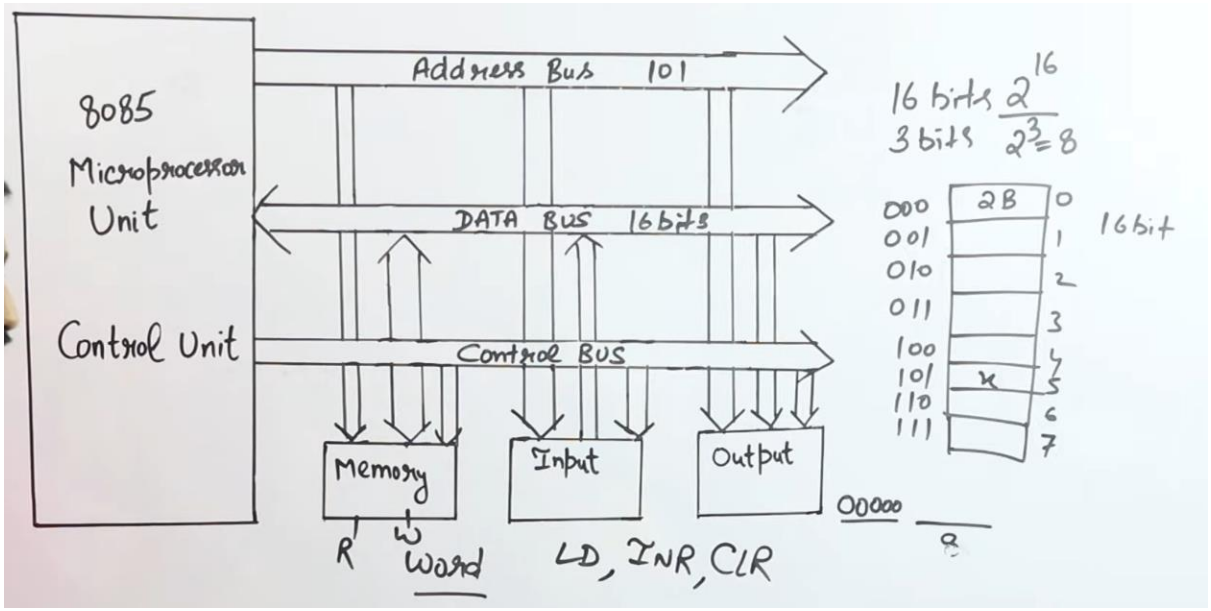
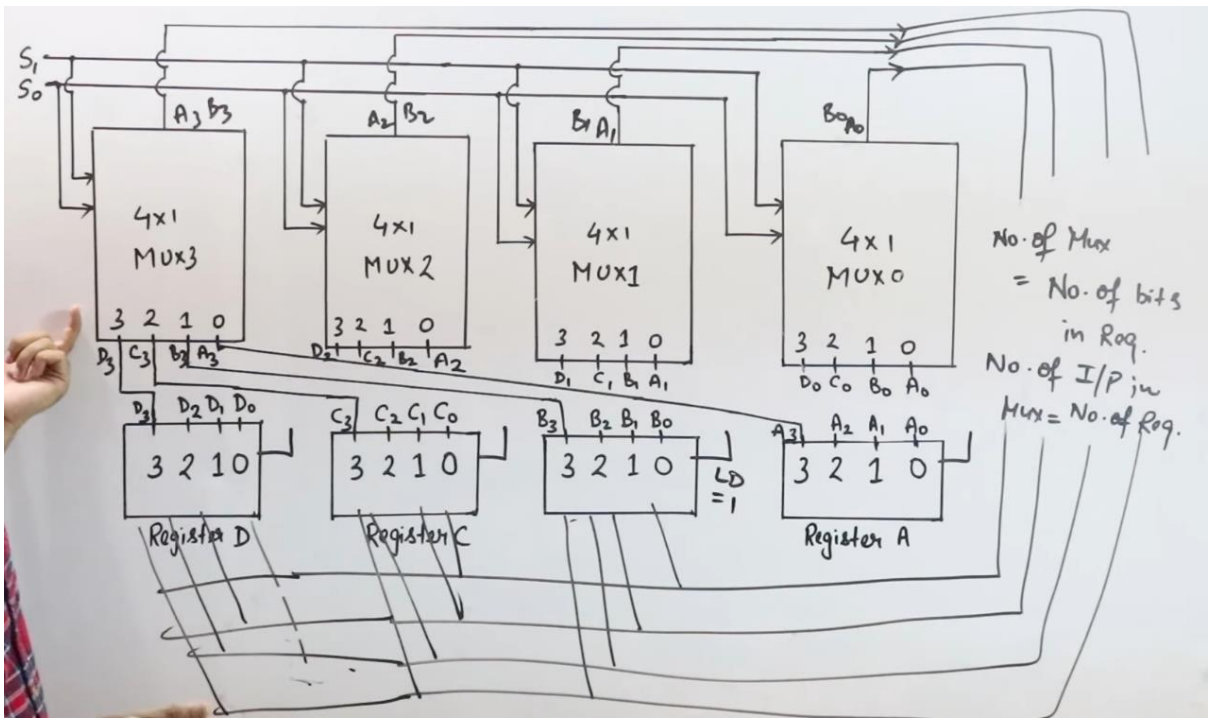
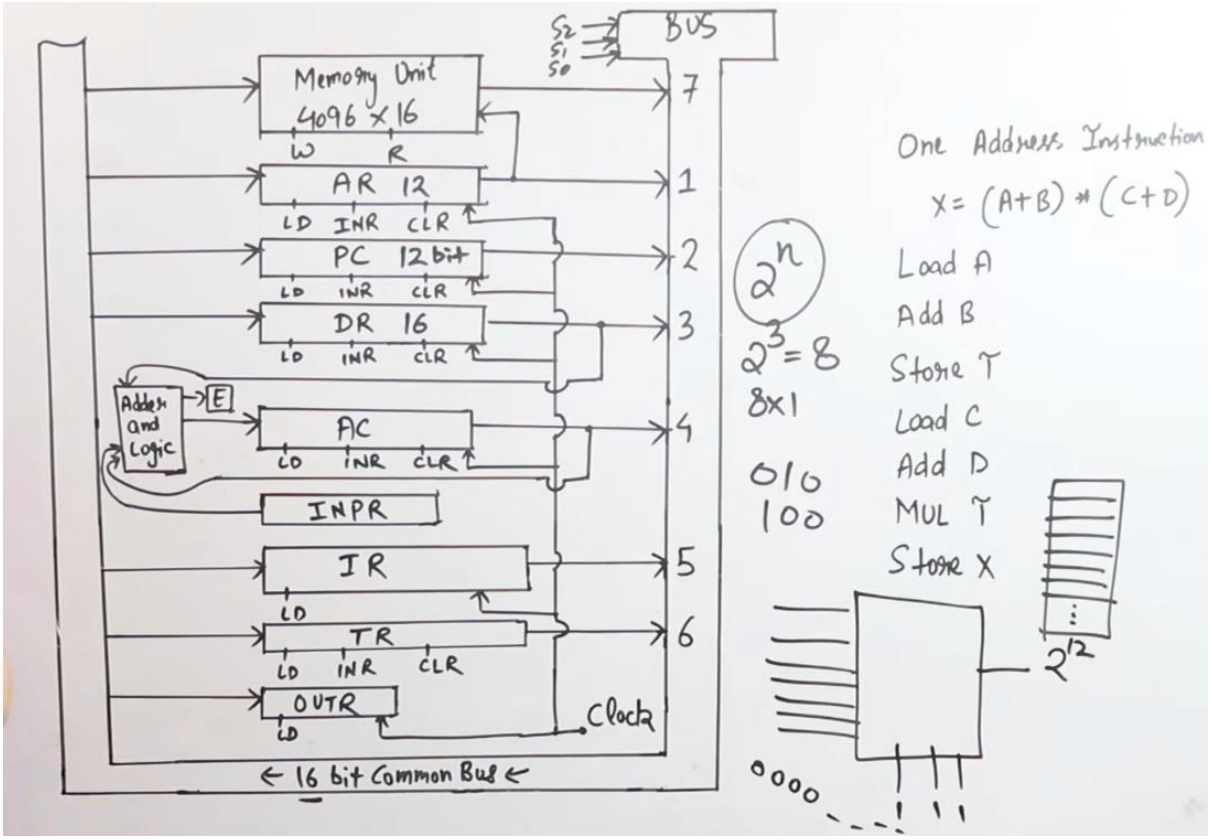
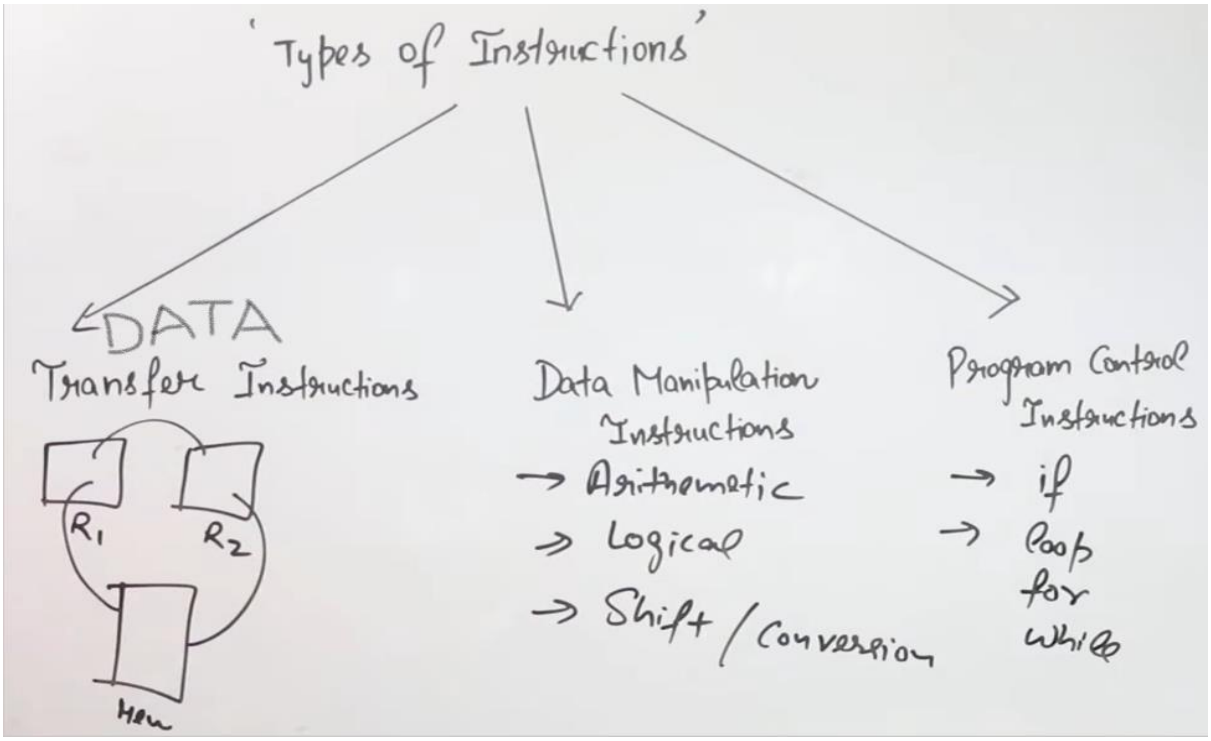

Memory
4096 x 16
No. of words
Word Size
15 0
I
OPcode
3
Operand
0, 1
11
Address Register (AR) (12)
Instruction Reg. (IR) (16 bits)
Data Register (DR) (16)
Temporary Reg. (TR) (16 bits)
Accumulator (16)
Input Reg. (INPR) 8bits
Output Reg. (OUTR) 8bit
Program Counter (PC)
(12 bit)
ADD 450
1002
12
2
0000..0
0000..1
1000
1001
1002
1111111


8085
Microprocessor Unit
Control Unit
Address Bus 101
DATA BUS 16 bits
Control BUS
Memory
Input
Output
R
W
Word
LD, INR, CLR
16 bits 2^16
3 bits 2^3 = 8
000
001
010
011
100
101
110
111
2B
16 bit
00000


S1
S0
4x1 MUX3
4x1 MUX2
4x1 MUX1
4x1 MUX0
3 2 1 0
Register D
Register C
Register B
Register A
LD = 1
No. of Mux = No. of bits in Reg.
No. of I/P in Mux = No. of Reg.

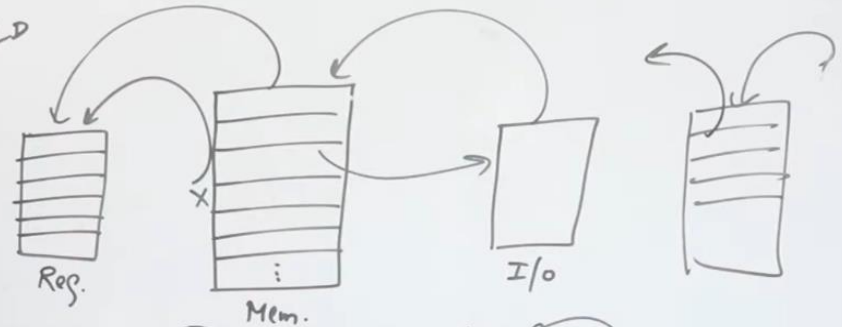

S2
S1
S0
BUS
Memory Unit
4096 x 16
W
R
AR 12
LD INR CLR
PC 12 bit
LD INR CLR
DR 16
LD INR CLR
Adder and Logic
E
AC
LD INR CLR
INPR
IR
LD
TR
LD INR CLR
OUTR
LD
Clock
16 bit Common Bus
7
1
2
3
4
5
6
One Address Instruction
X = (A+B) * (C+D)
Load A
Add B
Store T
Load C
Add D
MUL T
Store X
2^n
$2^3 = 8$
8x1
010
100
2^{12}
0000


'Types of Instructions'
DATA
Transfer Instructions
R1
R2
Mem
Data Manipulation
Instructions
→ Arithmetic
→ Logical
→ Shift / Conversion
Program Control
Instructions
→ if
→ Loop
for
while

Data Transfer Instructions

- Mov
- Load
- Store
- Exchange
- Input
- Output
- Push
- Pop

LD STA In XCHG



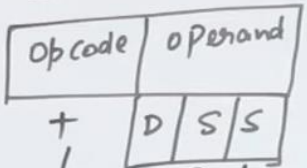
Mov R1, R2
Mov R1, 500
Mov R1, X

LD I 500
LD R1, R2
LD R1, X

Arithmetic Instructions

- Add
- Sub
- MUL
- DIV
- INC
- DEC
- Add with Carry
- Sub with borrow
- Negate

C = a+b



C = a+b
a*b
a-b
a/b

5
-5

10 20
30

2*3 = 6

2+2+2 = 6

1 1 0
1 1 0 1
0 1 0 1
1 0 0 0

Logical Instructions

- Complement (COM or NOT)
- Clear (CLR)
- Logical-And (AND)
- Logical-OR (OR)
- Ex-OR (XOR)
- Clear Carry (CLRC)
- Set Carry (STC)
- Complement carry (CMC)
- Enable Interrupt (EI)
- Disable Interrupt (DI)

0 1 0 0 1

$1^c = 0$ $0^c = 1$

0 1 0 1
1 1 1 1
1 0 1 0

1 0 1 0
0 0 0 1
0 0 0 0

1 0 1 1
0 0 0 1
0 0 0 1

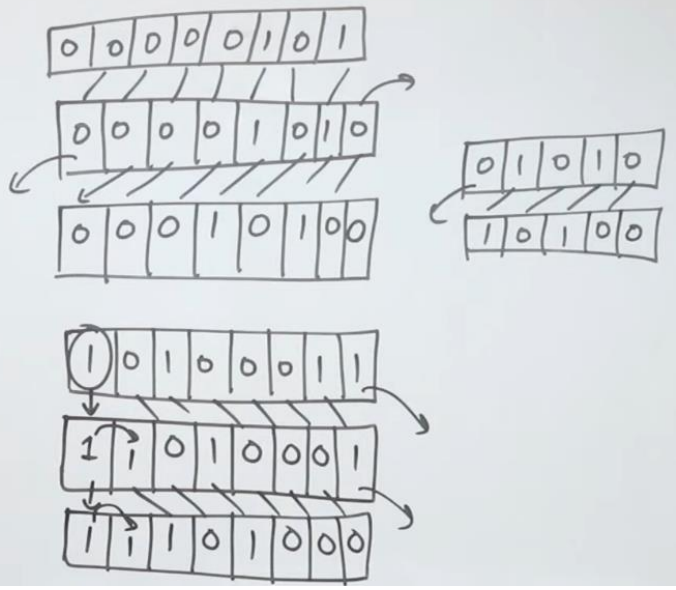
1 0 1 1
0 0
0 0

1 0 1 1
1 1 1 1
1 1 1 1

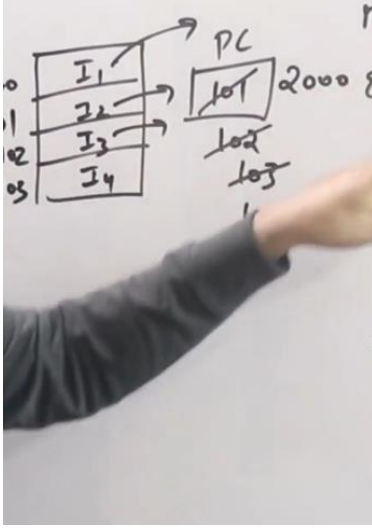
0 1 0 1
1 0 1 0

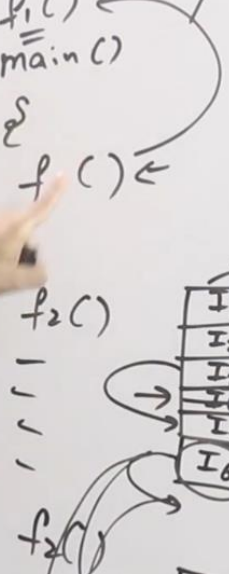
Shift Instructions

- Logical shift left
- Logical shift right
- → Arithmetic shift right
- → Arithmetic shift left
- → Rotate right
- → Rotate Left
- → Rotate right through carry
- → Rotate Left through Carry



Program Control Instructions





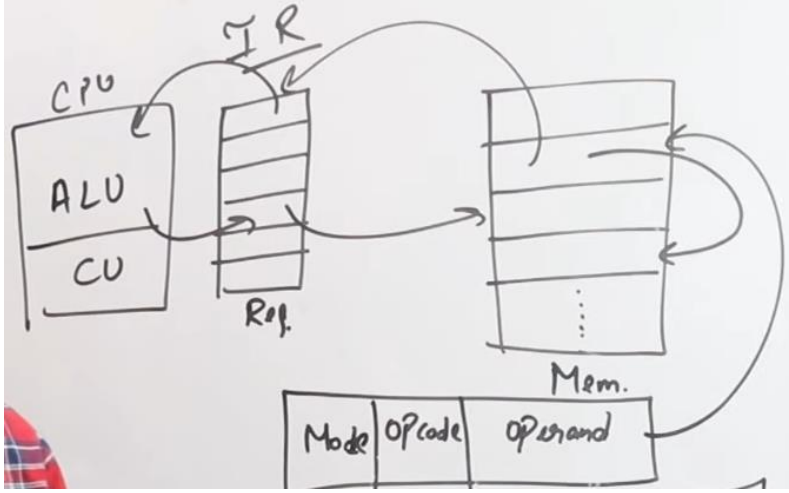
Transfer of Control Instructions

Branch. → Uncond
→ Cond

- Jmp 2000
- B 3000
- Skip Skp
- Call 3000
- Return

BE R1, R2 2000
R1 = R2
R1 ≠ R2
BNZ R1, 2000
R1 = 0

Instruction Format



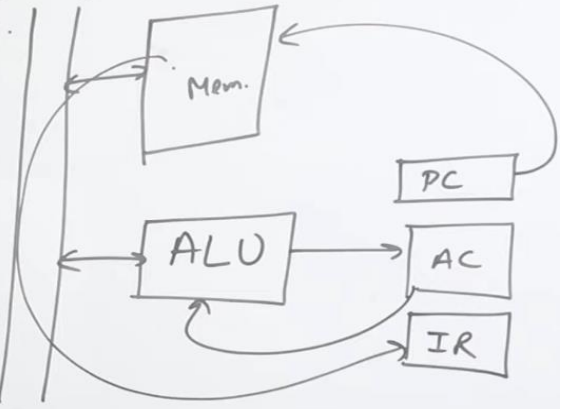

```
main()
{
int a=10;   Data
int b=20;
int c;
c=a+b;      instr.
Pf(c)       instr
}
```

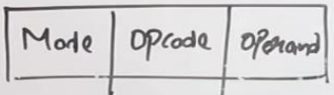
Types of CPU organisations

1) Single Accumulator org.
2) General Register org.
3) Stack organisation

$X = A + B$

Single Address inst.



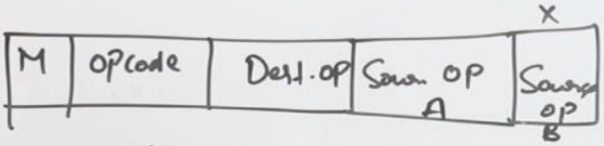


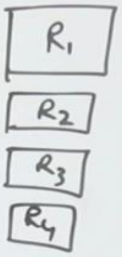
LD A (A)
ADD (B)
Store X ← AC

AC ← AC + B

Types of CPU organisations

1) Single Accumulator org. — 1 Add.
2) General Register org. — 2 add, 3 add
3) Stack organisation





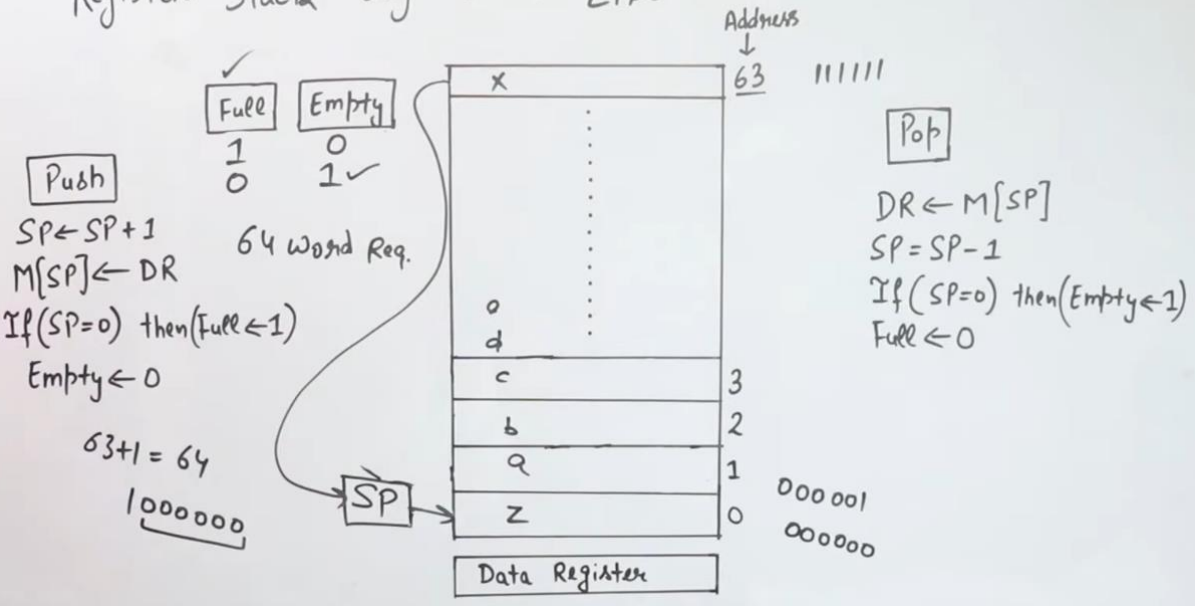
$X = (A+B) * (C+D)$

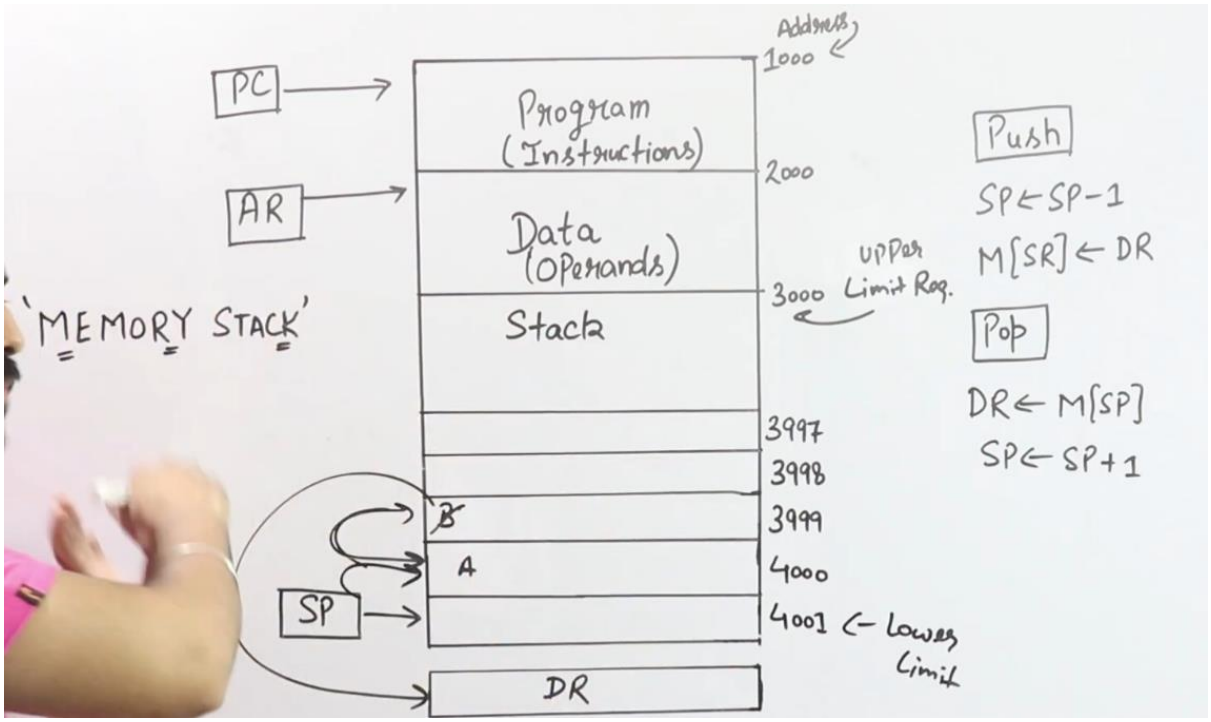
Mov R_1 A
Add R_1 B — $R_1 \leftarrow R_1 + B$
Mov R_2 C
Add R_2 D — $R_2 \leftarrow R_2 + D$
MUL R_1 R_2 — $R_1 \leftarrow R_1 * R_2$

Add R_1 A,B
Add R_2 , C,D
MUL X, R_1, R_2

$R_1 \leftarrow (A)+(B)$
$R_2 \leftarrow (C+D)$
$M[X] \leftarrow R_1 * R_2$

Register Stack organisation LIFO





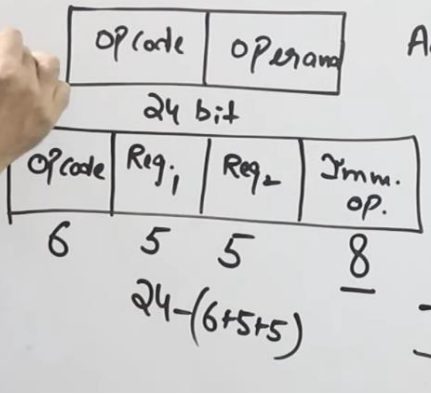
A machine has 24 bit instruction format. It has 32 registers and each of which is 32 bit long. It needs to support 49 instructions. Each instruction has two register operands and one immediate operand. If the immediate operand is signed integer the minimum value of immediate operand is ____.

A) -64

B) -128

C) -256

d) -32



Add R_1 R_2 50

$2^5 = 32$

$2^6 = 64$

0-255

256

-128 to 127

Implied Mode: Operand is specified implicitly in the definition of the instruction. It is generally used for zero and one address mode.

Immediate Mode: Operand is directly provided as constant in address part. Hence, no computation is required for calculating the Effective Address.

Register Mode: Register number is there in the address part of the instruction. (Register contains the actual data). Due to register, instruction size is very less, and it is fast as well.

Register Indirect Mode: Register number is there in the instruction, but the register contains the address of the actual data. Instruction size is less, execution is fast but computational work is more.

Auto Increment or Decrement Mode : Special case of the above mode, but the data in the memory is in form of sequence (array of data). Used when data is in sequential form.

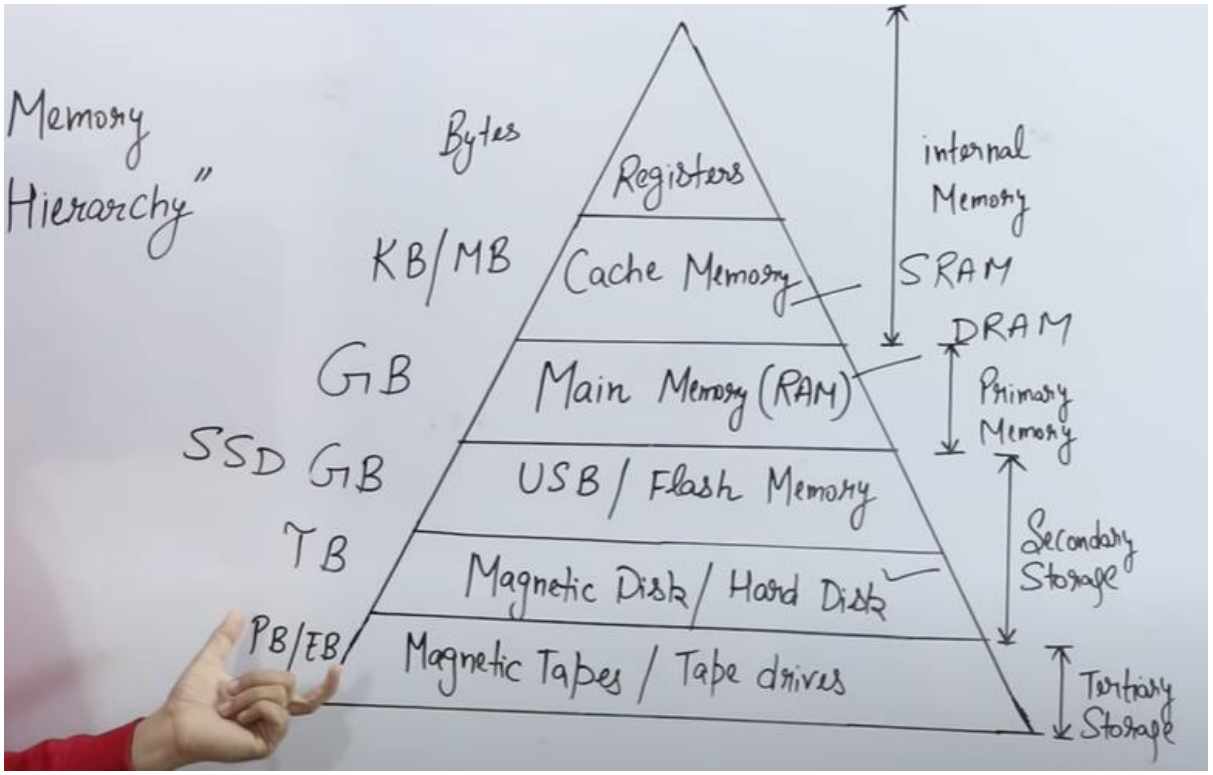
*Direct Addressing Mode/ Actual Address Mode : Actual address is given in the address part and helps in access of variables. Disadvantage - we cannot give large address due to limited size of instruction.

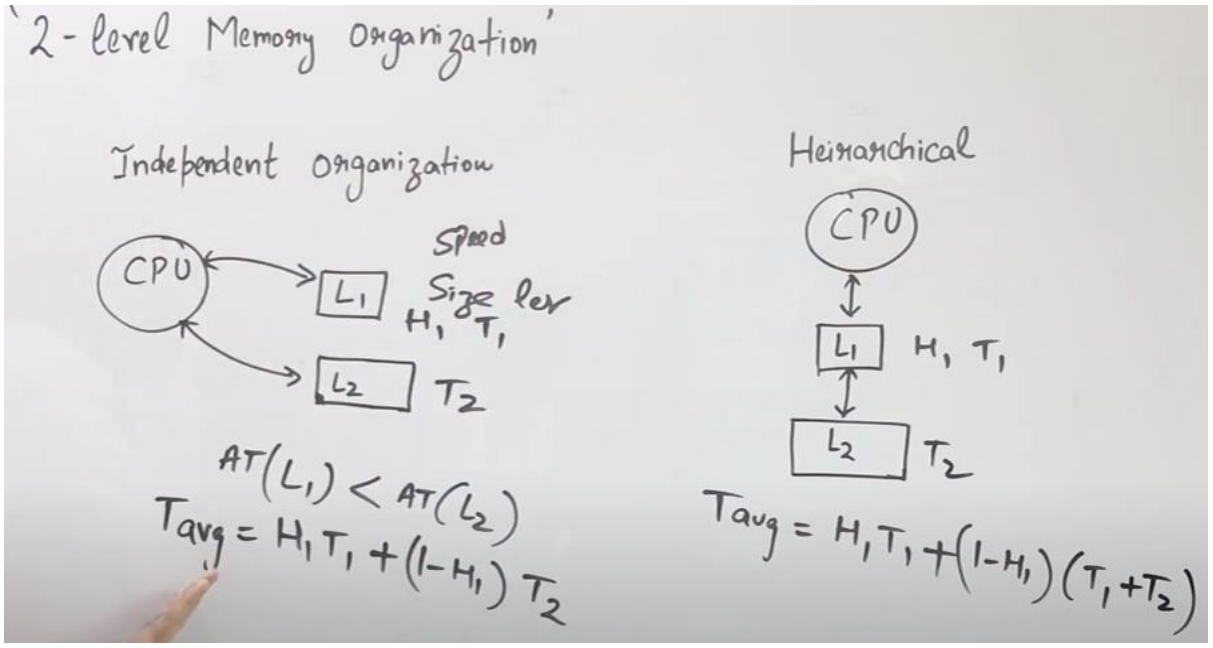
*Indirect Addressing Mode: Instruction contains the address which itself contains the actual address of the data. It is used to implement pointers and passing parameters. We need to access the memory twice.
Relative Addressing Mode: Instruction contains the address which has offsets to jump to achieve the real address. Hence, Effective Address = PC + Offsets. Hence, we do not need to give whole address.

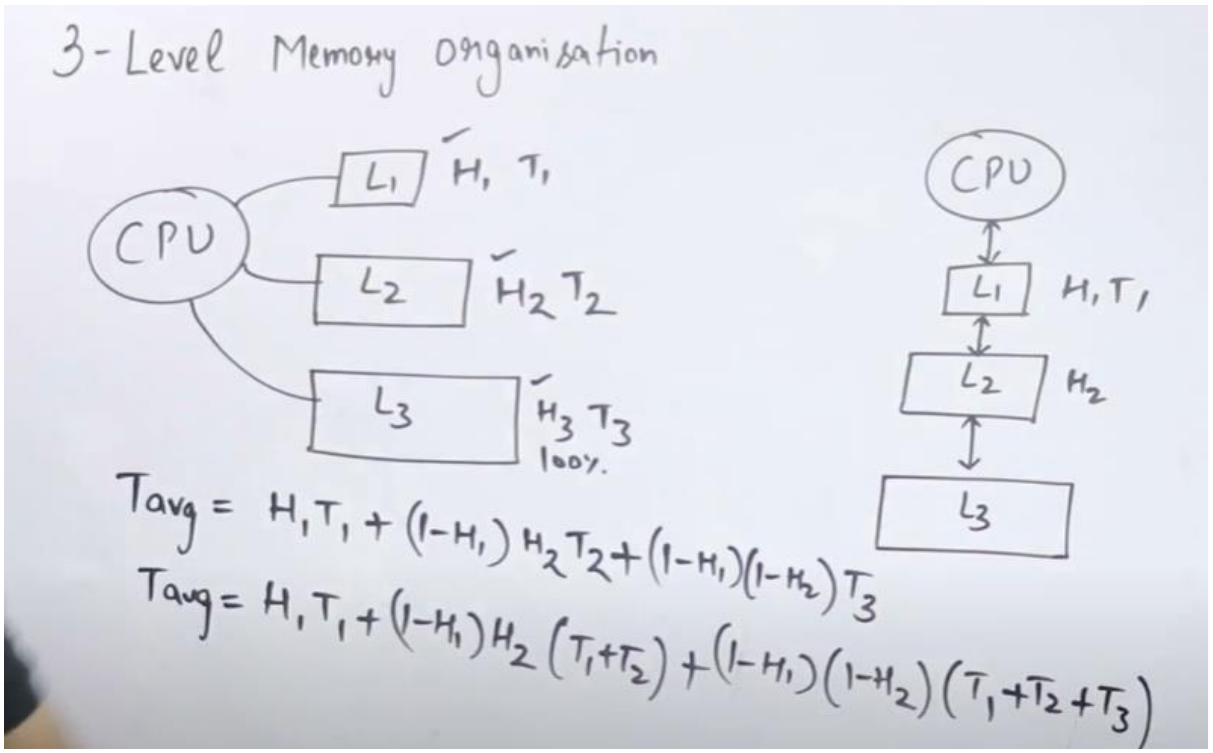
Base Register Addressing Mode: Used in program reallocation. Hence,

Effective Address = Base Address + Offsets (or Displacement).

Indexed Addressing Mode: Used to implement array efficiently and it requires several registers for the implementation. Due to array, we can access any element without changing the instruction.


Memory Hierarchy"
Bytes
KB/MB
GB
SSD GB
TB
PB/EB
Registers
Cache Memory
Main Memory (RAM)
USB / Flash Memory
Magnetic Disk / Hard Disk
Magnetic Tapes / Tape drives
internal Memory
SRAM
DRAM
Primary Memory
Secondary Storage
Tertiary Storage

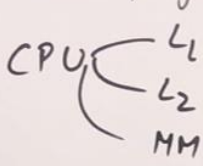

2-level Memory organization
Independent organization
CPU
L1
L2
Speed
Size
Cost per bit
H1 T1
T2
AT(L1) < AT(L2)
Tavg = H1T1 + (1-H1) T2
Heirarchical
CPU
L1
H1, T1
L2
T2
Tavg = H1T1 + (1-H1)(T1+T2)

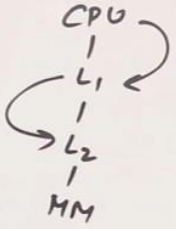

3-Level Memory organisation
CPU
L1
H1, T1
L2
H2 T2
L3
H3 T3
100%.
Tavg = H1T1 + (1-H1) H2T2 + (1-H1)(1-H2) T3
Tavg = H1T1 + (1-H1) H2 (T1+T2) + (1-H1)(1-H2)(T1+T2+T3)
CPU
L1
H1, T1
L2
H2
L3

Consider a system with 2-level caches. Access times of L_1 and L_2 and main memory are 1 ns, 10 ns and 500 ns. Hit ratio of L_1 and L_2 are .8 and .9 What is Avg. access time of system ignoring search time within Cache ?

A) 13.0 ns
B) 12.8 ns
~~C)~~ 12.6 ns
D) 12.4 ns

(GATE 2004)





$T_{avg} = H_1 L_1$

$= H_1 T_1 + (1-H_1) H_2 T_2 + (1-H_1)(1-H_2) T_3$

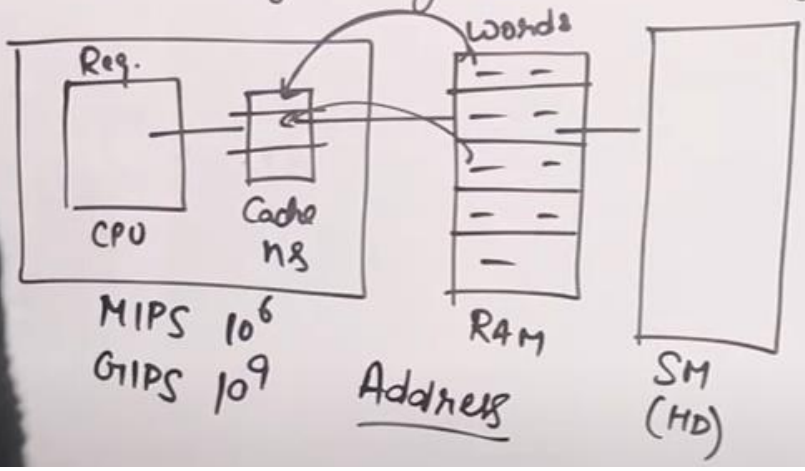
$= .8 \times 1 + .2 \times .9 \times 10 + .2 \times .1 \times 500$

$= .8 + 1.8 + 10$

$= 12.6$ ns.

Cache Mapping and its types

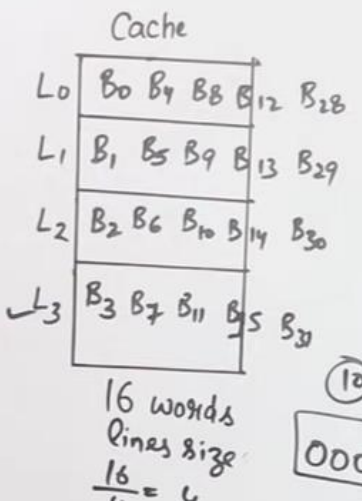
Direct Mapping Fully associative K-way Set associative



Direct Mapping

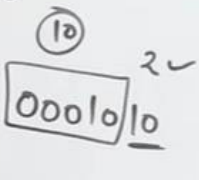
Kmod n
↓ Block No ↓ No. of lines

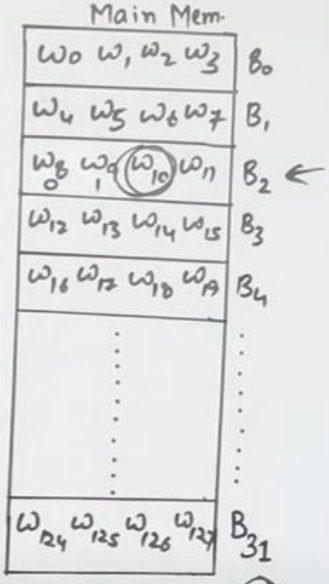
B_0
0 mod 4
1 mod 4
2 mod 4
4 mod 4



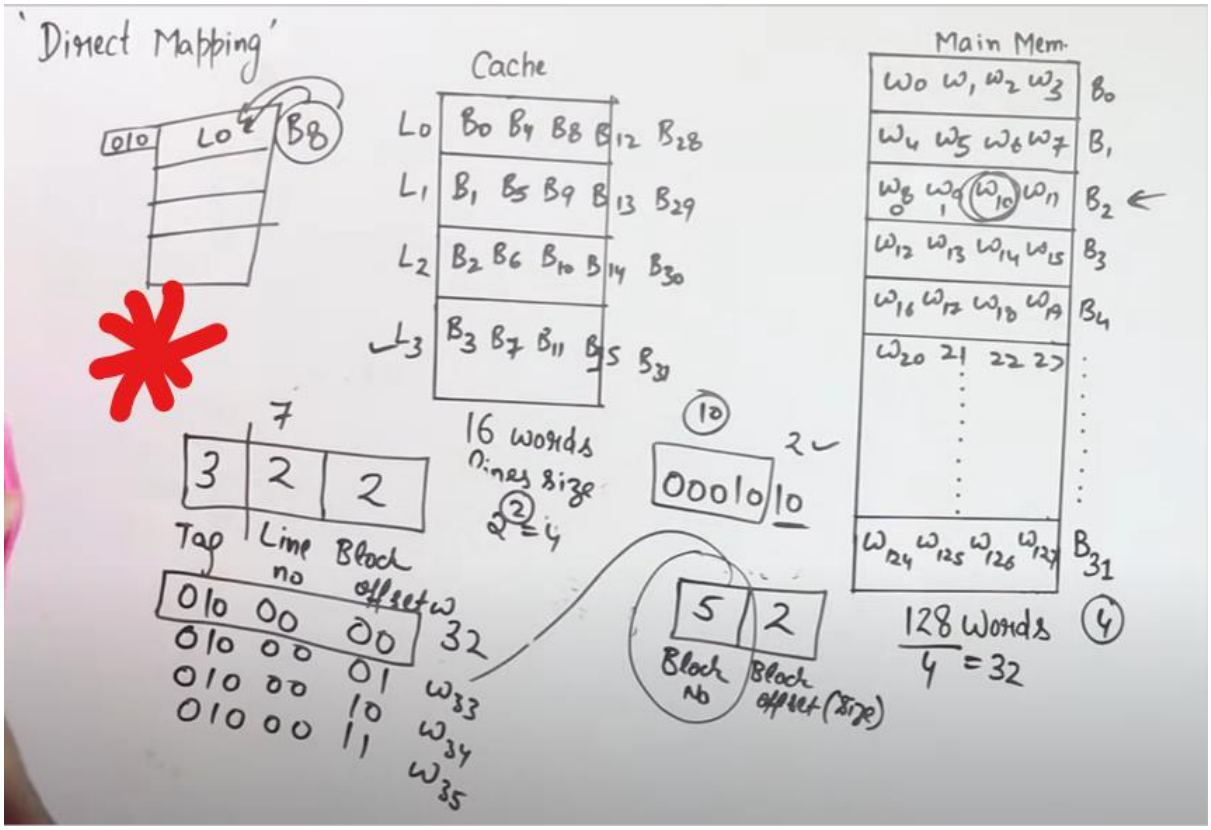
16 words
Lines size
$\frac{16}{4} = 4$

0-127 = 7 $2^7 = 128$
0-3 = $2^2 = 4$





128 Words
$\frac{128}{4} = 32$
4

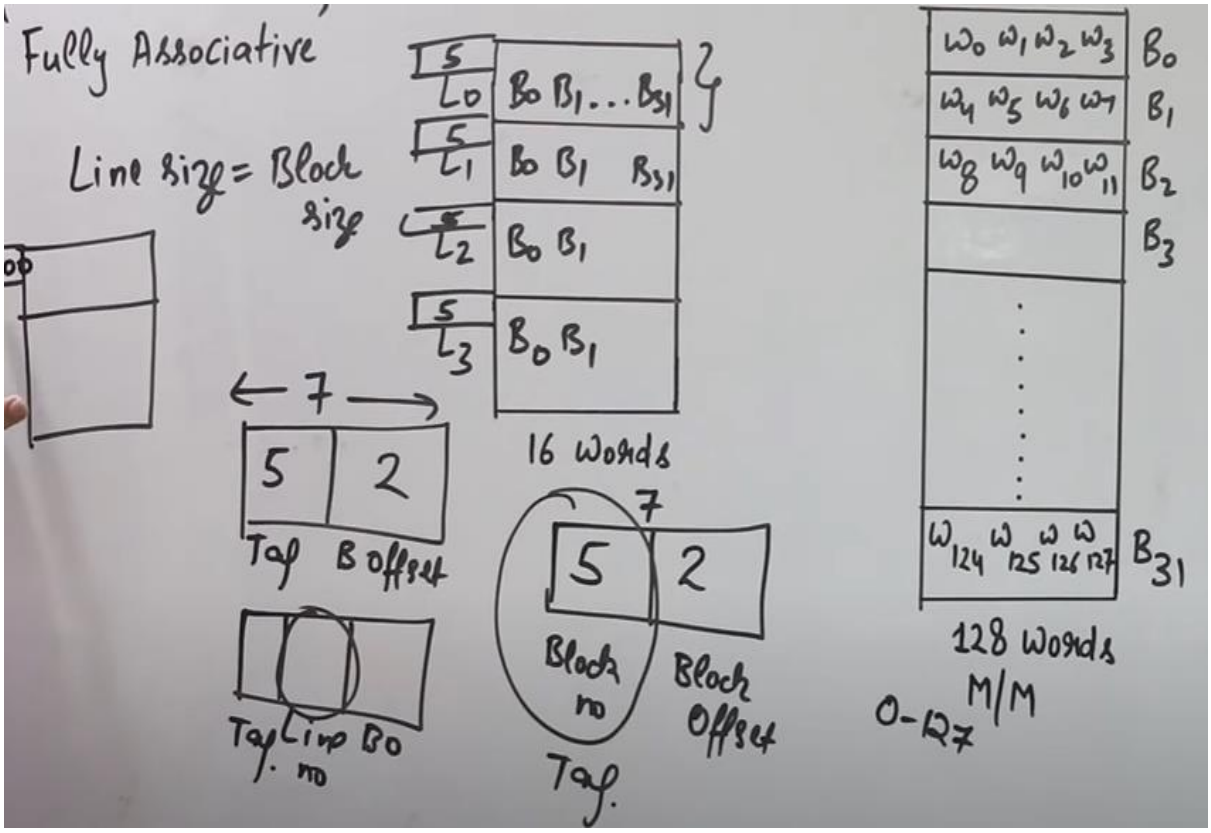


Advantages:

1. Line number is fixed hence we do not need to search whole lines.
2. Hits are easy and fast (Searching time is low).

Disadvantages:

1. Due to fixed place of block, even if we have space in other lines, we cannot place the block there (Conflict Miss).

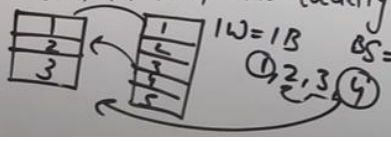


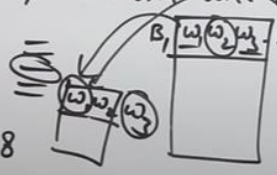
Locality of Reference

Space → Spatial locality
Time → Temporal Locality

CPU → 

- → if a word is accessed now then the word adjacent to it (close proximity) will be accessed next.
- → Keeping more words in a block affects spatial locality (block size)
- → if a word is referenced now then the same word will be referenced again in future
- → LRU is used in temporal locality.



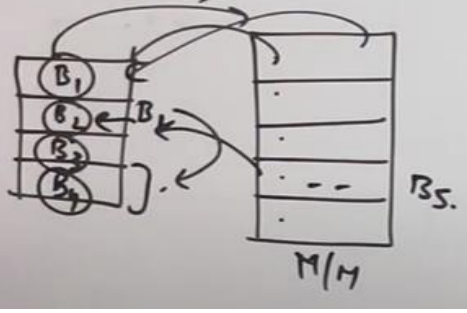


Cache Replacement Policies (Algo)

FIFO ✓ LRU ✓ MRU Random

(Least Recently used)

(Most Recently used)



Miss Penalty Reduce

Hit increase

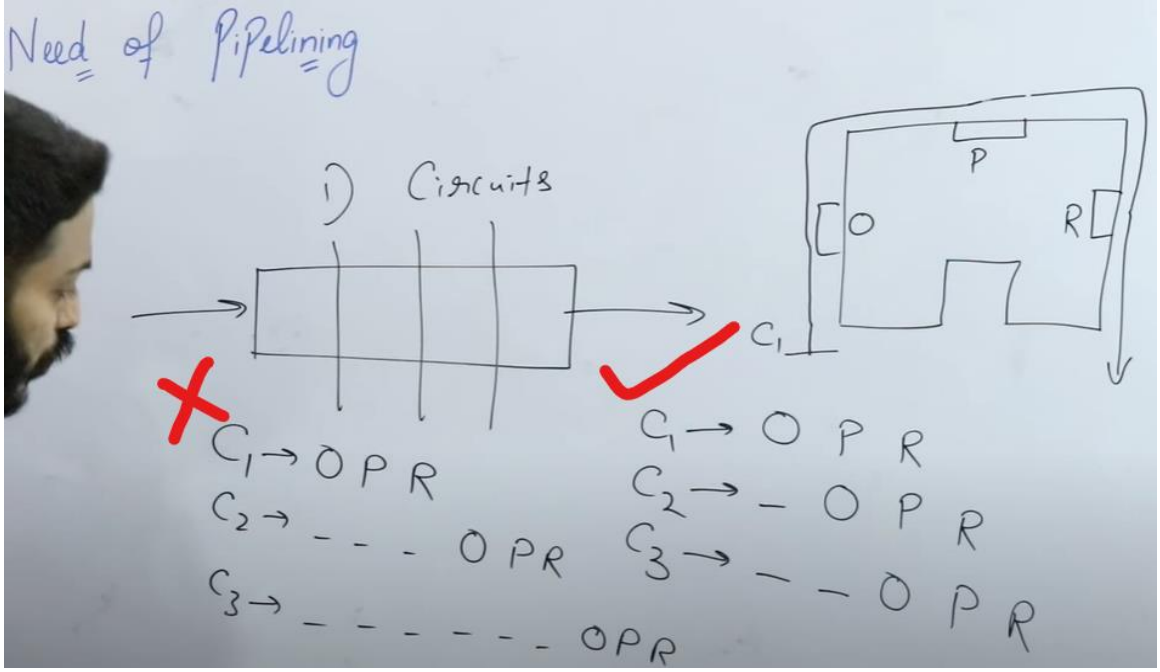
Consider a fully associative cache with 8 cache blocks (0-7) and the following sequence of memory block requests:

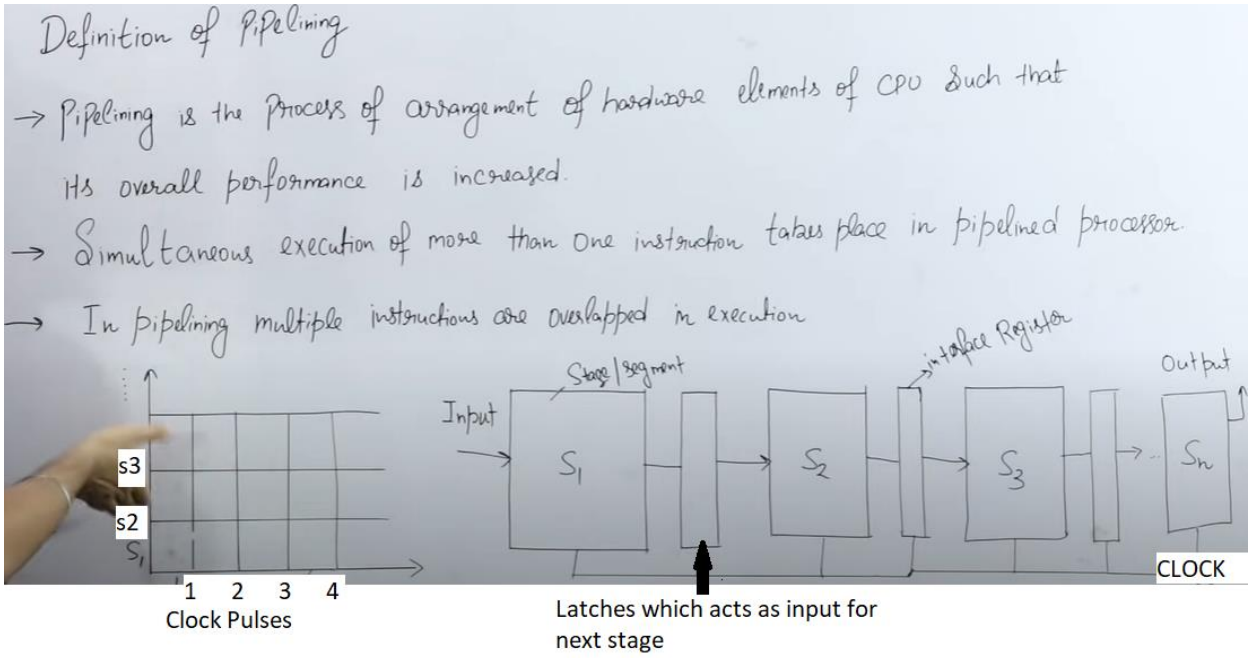
M H H H H H H H
4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7

if LRU replacement policy is used, which cache block will have memory block 7 (GIATE 2004)

	Cache
0	~~4~~ 45
1	~~3~~ 22
2	25
3	8
4	~~19~~ 3
5	~~6~~ 7
6	16
7	35

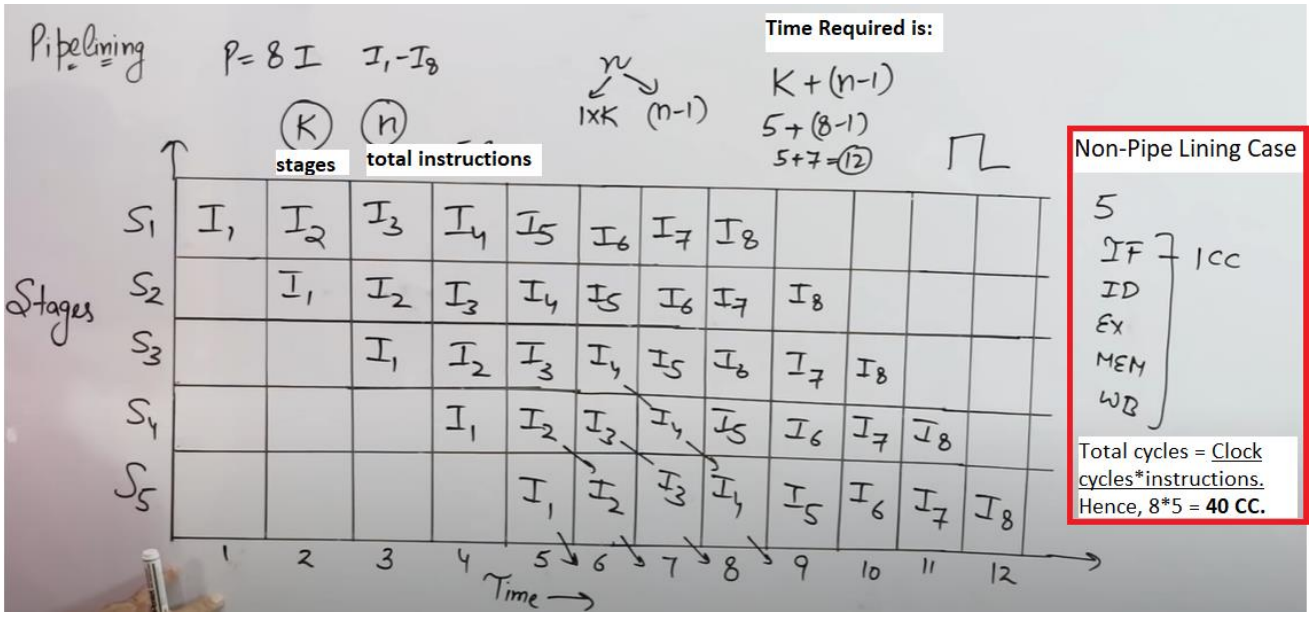
45
22





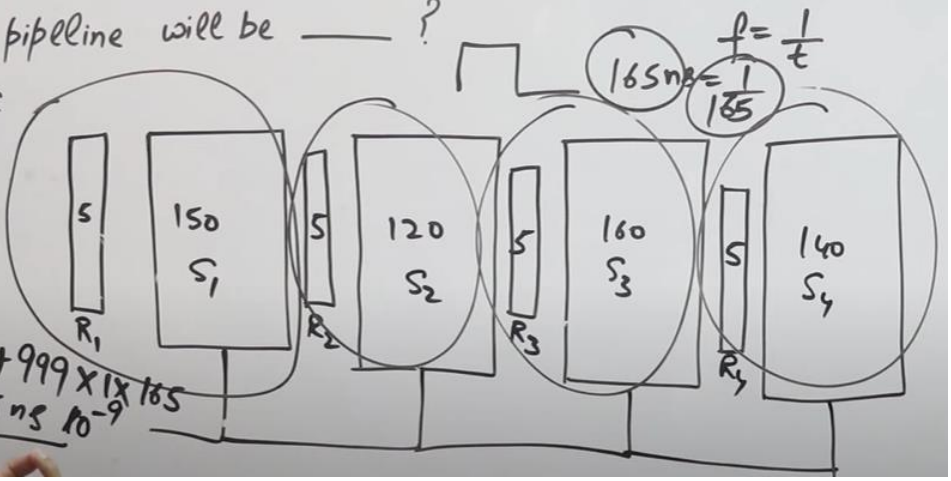
First Instruction's time = number of Stages (**K**). After that each instruction takes 1 unit.

Here, **CPI is nearly 1** (Clock Per Cycle) and Clock Cycle = **k + (n-1).**
Speed Up = non-Pipeline time /Pipeline time.



A 4-stage pipeline has stage delays as 150, 120, 160 and 140 ns. Registers are used between stages and have a delay of 5 ns each. Assuming Const. clock rate, the total time taken to process 1000 data items on this pipeline will be ___ ?

a) 120.4 μsec
b) 160.5 μsec
c) 165.5 μsec
d) 590 μsec.



$$= \frac{1 \times 4 \times 165 + 999 \times 1 \times 165}{165495 \text{ ns}} \; 10^{-9}$$

Consider a non pipelined processor with a clock rate of 2.5 gigahertz and avg. cycles / Instruction of four. The same processor is upgraded to a pipelined processor with five stages, but due to internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume that there is no stall in pipeline. The speedup achieved in pipeline processor is ___ ?

a) 3.2
b) 3.0
c) 2.2
d) 2.0

$$\text{Speedup} = \frac{T_{wp}}{T_p} \qquad f = 2.5 \text{ GHz} \qquad T = \frac{1}{f}$$

$$T_{wp} = 4 \times \frac{1}{2.5 \times 10^9} \text{ sec} = \frac{4 \times 1}{2.5 \times 10^9} \text{ sec}$$

$$T_p = \frac{1}{2 \times 10^9} \text{ sec}$$

$$\frac{\frac{4 \times 1}{2.5 \times 10^9} \text{ sec}}{\frac{1}{2 \times 10^9} \text{ sec}} = \frac{4 \times 2}{2.5} = \frac{8}{2.5}$$

Hazards in Pipelining

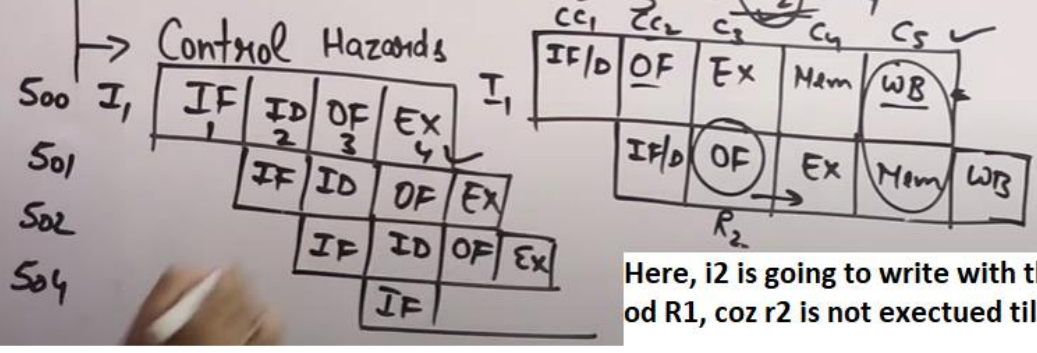
R = Read
W = Write

- Data Hazards
  - RAW (TD)
  - WAR (AD)
  - WAW (OD)
- Structural Hazards
- Control Hazards

CPI = 1

I_1 $R_2 \leftarrow R_2 + R_3$

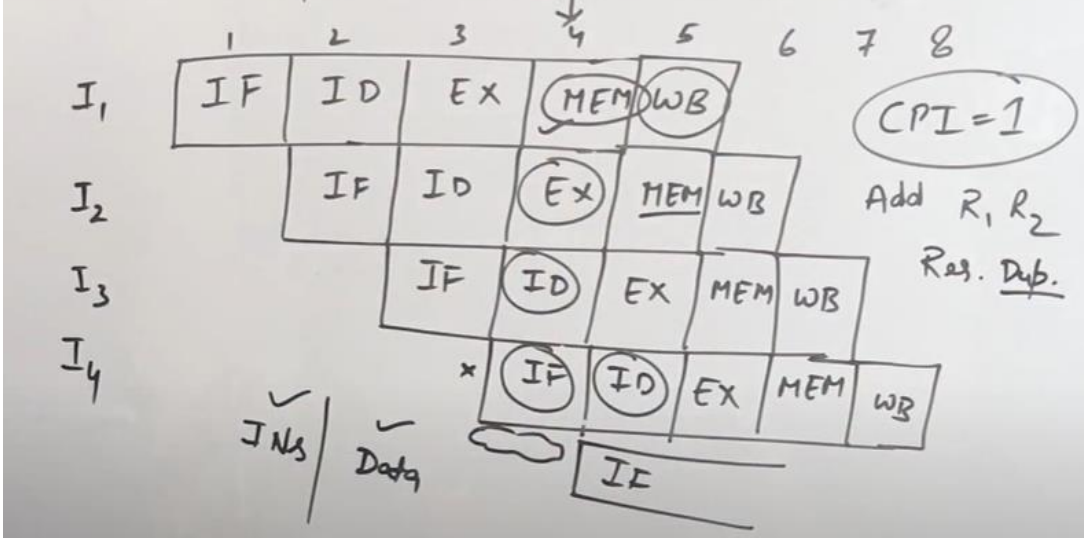
I_2 $R_5 \leftarrow R_2 + R_4$



Here, i2 is going to write with the old data od R1, coz r2 is not exectued till now.

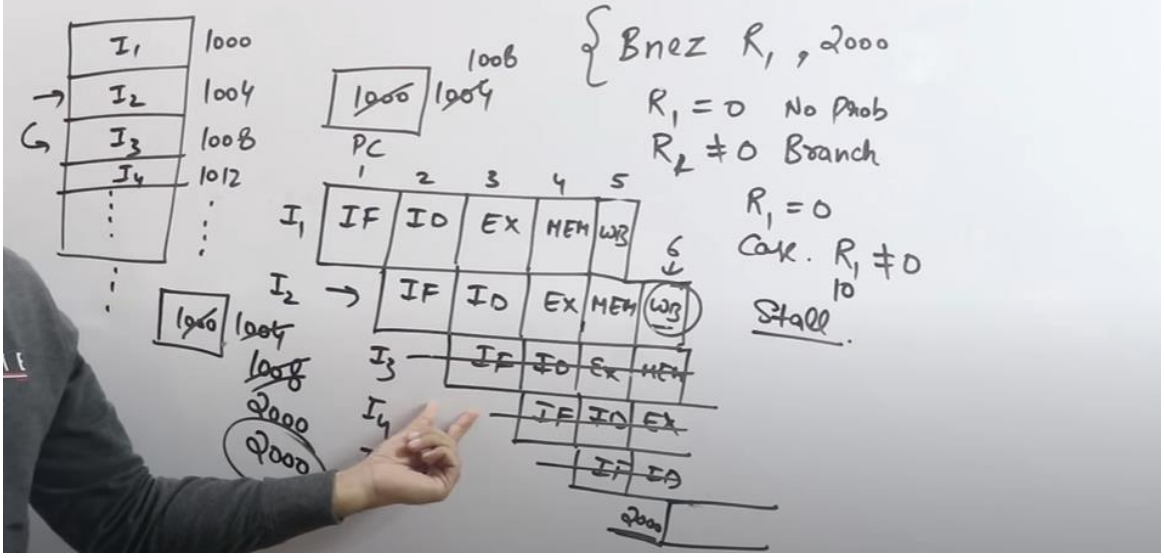
Structural Hazard

→ When multiple instructions needs same resource.



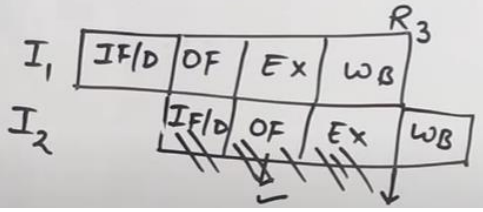
Control Hazard

→ All Instructions who change the program counter leads to control Hazards



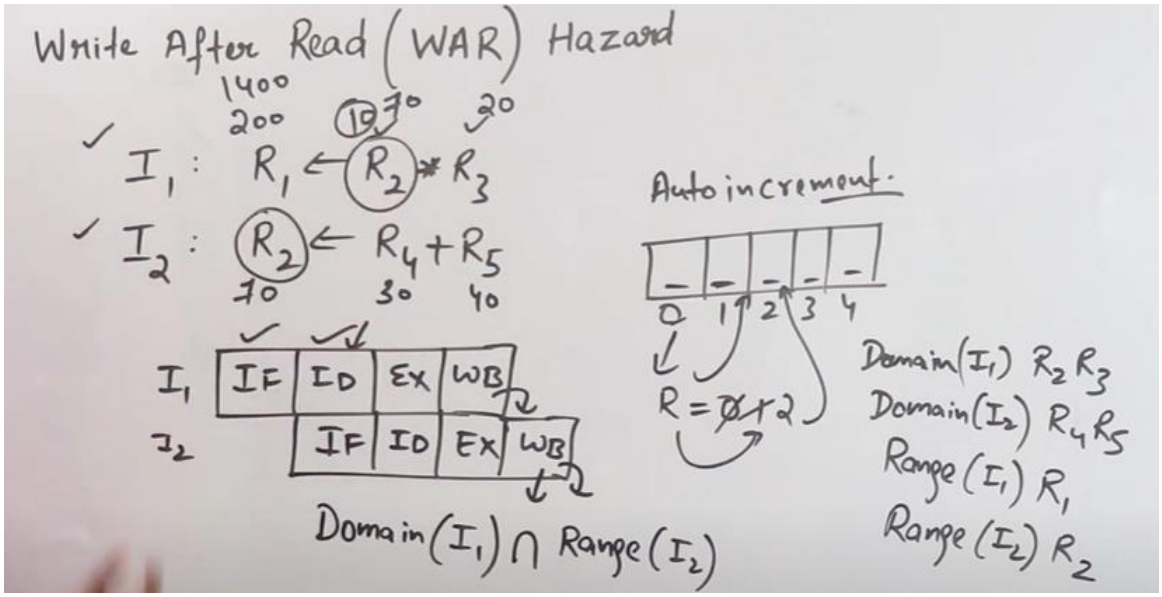
RAW (Read After Write) Hazard

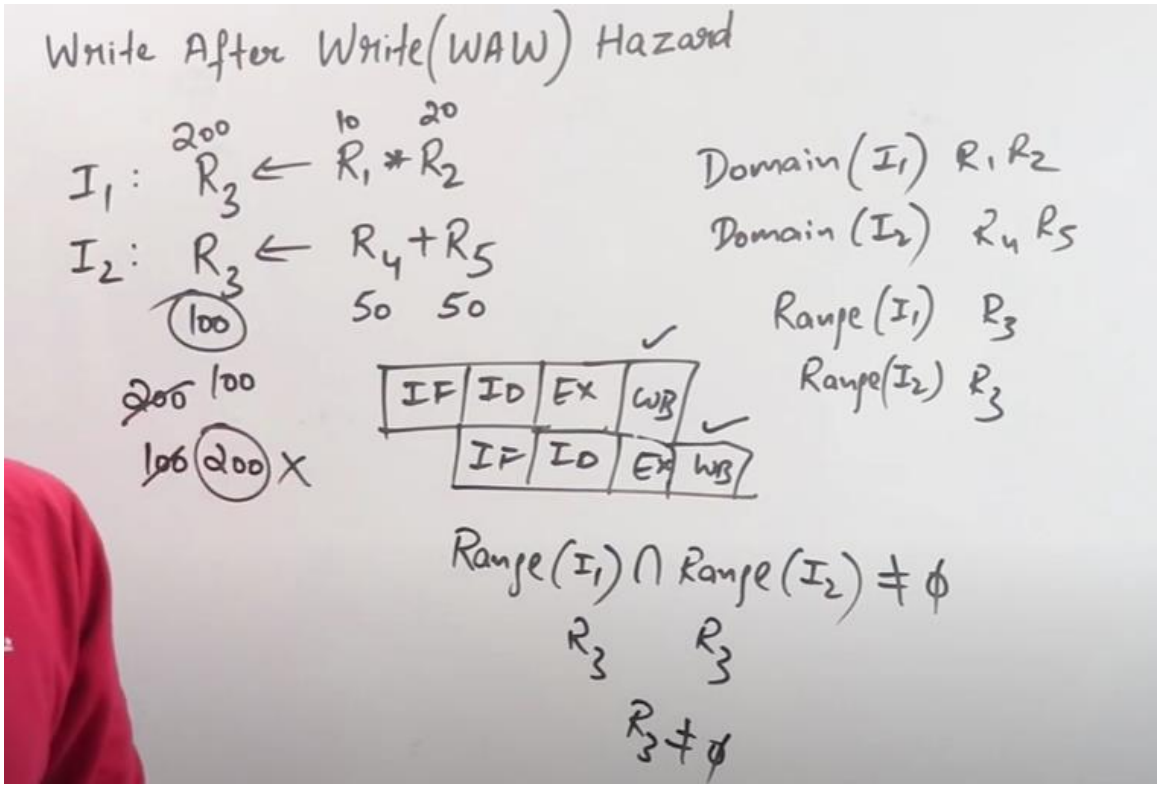
		Domain	Range
I_1	$R_3 \leftarrow R_1 + R_2$	R_1, R_2	R_3
I_2	$R_5 \leftarrow R_3 + R_4$	R_3, R_4	R_5



$I_1(\text{Range}) \cap I_2(\text{Domain}) \neq \phi$

RAW





Why we need interface?

1. To cope up with the speed of CPU and memory, coz we use I/O devices which is very slow comparatively.
2. For the conversion of signals b/w I/O and CPU.
3. For translator or interpreter b/w I/O and CPU.
4. For the conversion b/w data format (e.g. I/O is 8 bits and CPU is 64 bit).

Daisy Chaining Interrupt and Parallel Priority Interrupt:
https://upscfever.com/upsc-fever/en/gatecse/en-gatecse-chp165.html

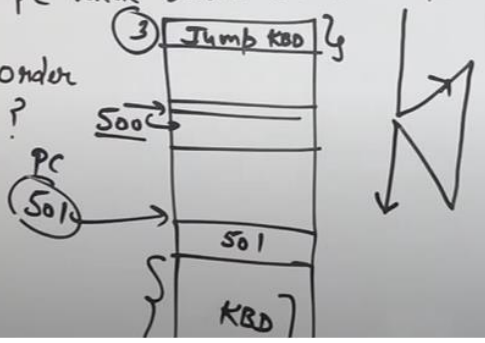
The following are some events that occurs after a device Controller issues an interrupt while process L is under execution.

P) The processor pushes the process status of L onto control stack.
Q) The processor finishes the execution of current instruction.
R) The processor executes the interrupt service routine.
S) The processor pops the process status of L from control stack.
T) The processor loads the new PC value based on interrupt.

Which of the following is correct order in which the above events occur ?

A) QPTRS
B) PTRSQ
C) TRPQS
D) QTPRS

Hence,
A) QPTRS
is the correct order of execution



"CISC"

1) Complex Instruction Set Computer
2) Large Number of Instructions
3) Variable length Instruction format
4) Large No. of addressing modes
5) Cost is High
6) More Powerful
7) Several Cycle Instructions
8) Manipulation directly in Memory
9) Microprogrammed Control Unit
10) Examples: Mainframes, Motorola 6800, Intel 8080
MULT 2:2, 3:3

"RISC"

1) Reduced Instruction Set Computer
2) Less No. of Instructions
3) Fixed length Instruction format
4) Few no. of AM
5) Less cost
6) Less Powerful
7) Single Cycle Instructions
8) Only in Registers
9) Hard wired Control Unit
10) MIPS, ARM, SPARC, Fugaku
LOAD A, 2:2
LOAD B, 3:3
PROD A, B