1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68

COMPUTER ORGANIZATION AND ARCHITECTURE

Architecture —
1. Introduction
2. Floating Point Representation
3. Basics of Computer System & Microoperations
4. Instruction and Addressing Modes
5. CPU, Data Path & Control Unit.

Organization —
6. IO Organization
7. Memory & cache
8. Pipelining.

ARCHITECTURE —
Conceptual design & fundamental operational structure.

CPU design, Instructions, Addressing modes, Data Format.

Stored Program Architecture.

The program currently executing should be stored in memory (RAM

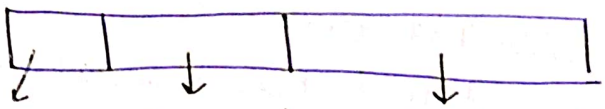
2 architectures —

- Von-neumann architecture
- Harward Architecture

} Not required for GATE

Floating Point Representation

provides larger range of numbers with limited number of bits.



Sign (1 bit)

Exponent (stored biased form. Only unsigned)

Mantissa (fraction) Signed fraction number

Bias value is added in exponents.

Bias value = $2^n - 1$ where n = no. of bits in exponent.

also known as excess notation

Mantissa is normalized.

101.11 →
- Explicit normalization: After binary point, first bit should be 1 and before point should be 0 → 0.1011×2^3 (1011 → Mantissa)
- Implicit normalization: 1 should be before binary point → 1.011×2^2

Explicit normalization is default (if not given)

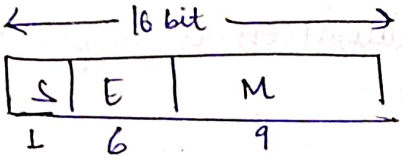
- Binary point is not stored
- Digits before binary point are not stored.
- Base of ~~binary~~ number is not stored

Value in implicit normalization = $(-1)^S * 0.1.M * 2^{E-bias}$

Value in extend normalization = $(-1)^S * 0.M * 2^{E-bias}$

Question:- Consider a 16 bit register used to store floating point number. The mantissa is normalized signed fractional number. Exponent is represented in excess 32 form. What is the 16 bit value for $+(19.25)_{10}$ in this register?

32 16 8 4 2 1
1 0 0 1 1
0 0 1 0 1
½ ¼



bias = +32
∴ No. of bits in E = 6
$2^{k-1} = 32 \Rightarrow k-1 = 5$
$k = 6$

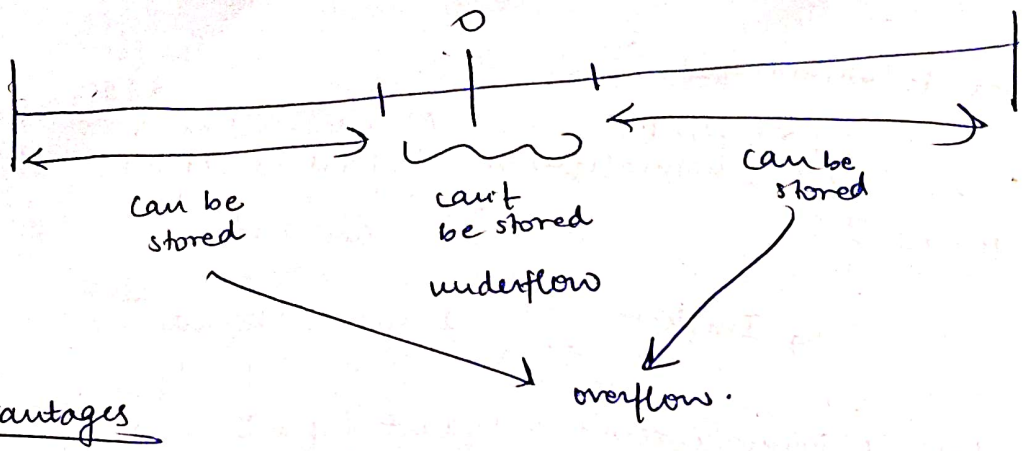
19.25 = 10011.01

not mentioned ∴ explicit normalization

∴ $19.25 = 0.1001101 \times 2^5$

E = 5+32 = 37 = 100101

0 100101 0010 1100



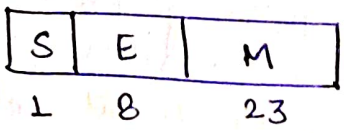
Disadvantages

- can't store 0
- can't store values very close to 0

IEEE - 754 Floating point Representation

⊙ Standard.

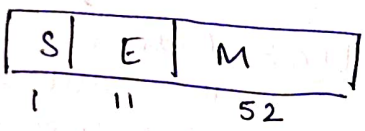
Single Precision (32 bits)



Bias = 127

E = all 0's OR E = all 1's — used to store Special numbers

Double Precision (64 bits)



Bias = 1023

For IEEE, default normalization is implicit.

Special numbers

S	E	M	Number
0	00.....0	0.....0	+0
1	00.....0	0.....0	-0
0	11.....1	0.....0	+∞
1	11.....1	0.....0	-∞
0/1	11.....1	M≠0...0	Not a number (NAN)
0/1	00.....0	M≠0...0	Denormalized number (fraction no.)
0/1	E≠00..0 and E≠00...0	M=xx.....x	Normal no. (Implicit normalized)

Value (implicit) $= (-1)^S * 1.M * 2^{E-bias}$

Value (denormalized) $= (-1)^S * 0.M * 2^{-126 \text{ or } -1022}$

↓

very very small number that cannot be normalized

Smallest possible value of $E = (000\ldots1)_2 = 1$

hence, $e = 1 - 127 = -126$

If number = 0.00000 ..... 011 (1 — 128 times) ⇒ Implicit normal

$M = 1.1$

$e = -129$

$E = -129 + 127 = -2$

∴ not possible to normalize.

∴ the number is stored in denormalized form.

$0.0000\ldots011 = 0.0011 * 2^{-126}$ (128 times)

$M = 0011$

$E = 000000\ldots$

for denormalized no., $e = -126$ always

$M = 1.1$

$E = -129$

$1.1 * 2^{-129}$

$1.1 * 2^{-126} * 2^{-3}$

$0.0011 * 2^{-126}$

$M = 0011000\ldots$

$E = 0000\ldots$

COMPONENTS OF COMPUTER

- CPU
  - Control unit (gives signal to different components)
  - ALU (computations)
- Memory
  - Primary (RAM, ROM)
  - Secondary (HDD)
- I/O Devices
  - Input
  - Output.

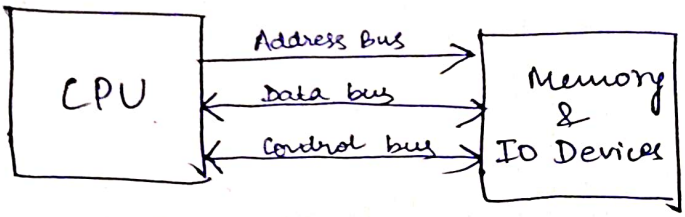
Other components

- System Bus
- Registers

System Bus

- Collection of communication lines between components of computer
- 3 Types
  - Address Bus
  - Data Bus
  - Control Bus

  } based on ~~component~~ content carried.



Operations that can be performed on memory —

- Read [content goes from memory to CPU]
- Write [data goes from CPU to memory]

If a memory has 2^n no. of addresses,

Address length (No. of bits in the address bus) = n bits.

Memory Access : Read

- CPU sends address to memory through Address bus.
- CPU sends control signal to memory.
- Memory performs read operation and sends data through data bus.

Memory Access : Write

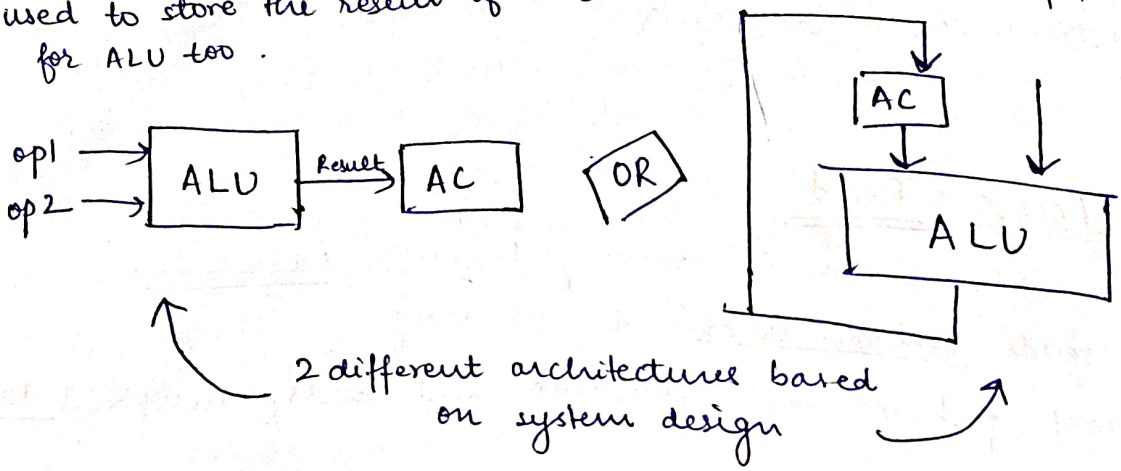
- CPU sends address to memory through addres bus
- CPU sends data to memory through data bus
- CPU sends control signal to memory via control bus
- Memory performs write operation.

CPU Registers

- small storage locations within CPU.
- 2 types
  - General Purpose Registers (GPR) R0, R1, R2 ...
  - ~~General~~ Special Purpose Registers (SPR)
    1. Accumulator
    2. Program Counter
    3. Instruction Register
    4. Stack Pointer
    5. Flag Register / Program Status Word (PSW)
    6. Address / Memory address Register (MAR)
    7. Data / Memory Data Register (MDR/MBR)

① Accumulator

used to store the result of ALU and sometimes one of the operands for ALU too.



2 different architecture based on system design

② Program Counter

used to store address of next instruction to be executed.

③ Instruction Register

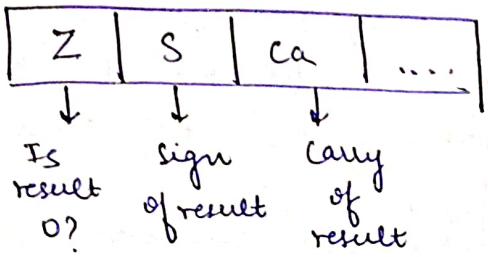
used to store currently executing instruction.

④ Stack Pointer

used to store address of the top of stack

⑤ Flag Register | Status Register | Program status word

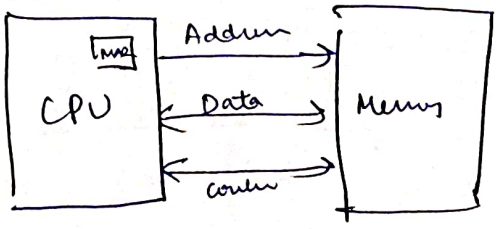
used to store status of ALU & result



(condition check. also known as condition code)

⑥ Memory Address Register

used to send address to memory



⑦ Memory Data Register

used to send data from CPU to memory or memory to CPU

Micro-Operations

The operations executed on values stored in registers

Symbolic Notation to describe microoperations = Register Transfer Language

Register Transfer $R1 \leftarrow R2$

2 mutually exclusive operations can be performed simultaneously

$R1 \leftarrow R2$, $PC \leftarrow PC+1$

Memory Transfer

- Memory Read: $R \leftarrow M[Add]$
- Memory write: $M[Add] \leftarrow R2$

Q:- If the content of register R1 is 5 and content of ~~register R2~~ memory address 1000 is 20. Then, the content of regoter R2 after following code execution is

$R1 \leftarrow R1+1$ $\quad R1 = 6$

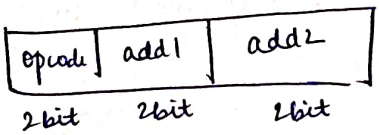
$R2 \leftarrow R1 + M[1000]$ $\quad R2 = 6+20 = 26$

Instruction Length

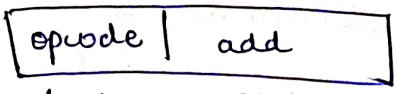
- Fixed length instruction: ~~fixed~~ variable length opcode.
- Variable Length instruction: fixed length opcode.

Q:- Consider a system which supports 2 address instructions and 1 address instructions both. The system has 6 bit instructions and 2 bits addresses. If there are three 2-address instruction in the system, then, the maximum and minimum how many 1-address instructions the system can support?

2 add instruction



1 address instruction



If CPU receives an instruction, it needs to decide if it is 2 adress instruction or 1 address instruction.

No. of bits in opcode of 2 add ins = 2 [∵ 2 bits are checked]

Only 3 2 address instructions are possible

∴ unused combination of bits = 1

∴ First 2 bits of opcode of 1 address instruction are fixed.

∴ Max no. of 1 address instruction = $2^2 = 4$

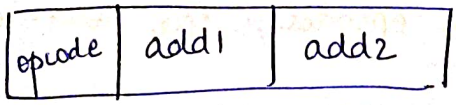
Min no. of 1 address instruction = 1 (minimum is always 1)

No. of 2 address instructions supported.	Unused opcode	No. of 1 address instruction supported.
3	1	$1 \times 2^2 = 4$
2	2	$2 \times 2^2 = 8$
1	3	$3 \times 2^2 = 12$

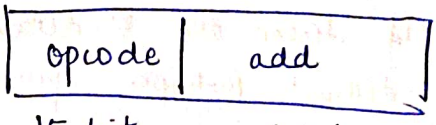
2 bcoz ▲ (bits opcode)$_{1add}$ − (bits opcode)$_{2add}$ = 2

Q:- Consider a system with 24 bit instruction and 9 bit addresses. If there are 60 2-address instructions, then, maximum how many 1 address instructions can be formulated in the system?

2 address instruction



1 address instruction



No. of 2 address instructions = 60

No. of 2 address instructions possible = $2^6 = 64$

No. of unused combinations = 4

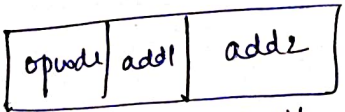
First 6 bits in opcode of 1 address instruction can be any of the four combinations.

∴ Max No. of 1 address instructions that can be formulated = $4 \times 2^9 = 2^2 \times 2^9 = 2^{11}$

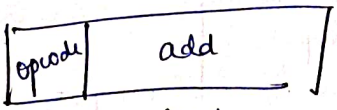
= 2048

Q:- Consider a system with 32 bit instructions and 12 bit addresses. If there are 254 2-address instructions and 8000 1-address instructions then, maximum how many 0-address instructions can be formulated?

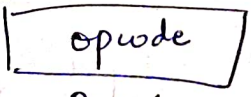
2 address ins



1 address ins



0 address



$(2^8 - 254) \times 2^{12}$ = Max. no. of 1 address instructions

$2 \times 2^{12} = 2^{13}$

Max no. of 0 address instruction = $(2^{13} - 8000) \times 2^{12}$

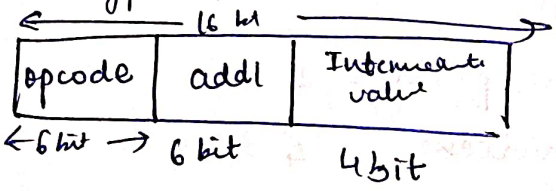
$= (8 \times 1024 - 8000) \times 2^{12}$

$= (8192 - 8000) \times 2^{12} = 192 \times 2^{12}$

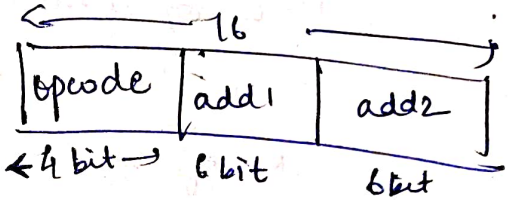
⑧ Gate 2020

A processor has 64 registers and uses 16 bit instruction format. It has 2 types of instructions i-type & R-type. Each I-type instruction contains an opcode, a register name & a 4 bit intermediate value. Each R-type instruction contains an opcode and 2 register names. If there are 8 distinct I-type opcodes, the max. no. of distinct R-type opcodes is

I-type instruction



R-type instruction



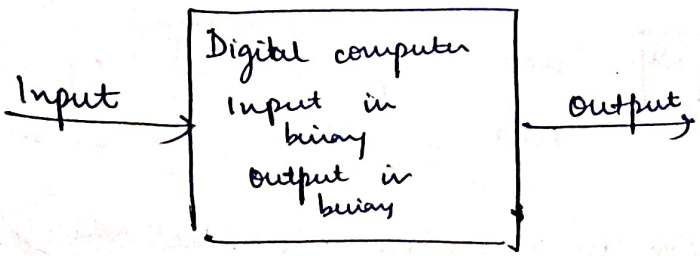
Let max. no. of R-type instructions = x

$$(2^4 - x) \times 2^2 = 8$$

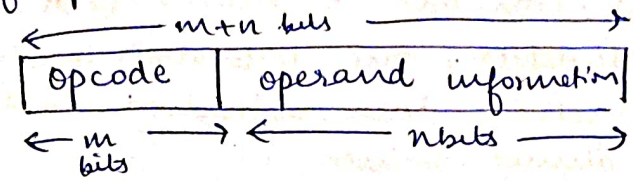
$$2^4 - x = 2 \quad , \quad x = 2^4 - 2 = 16 - 2 = \underline{14}$$

INSTRUCTION

① The commands to CPU for performing operations
② Group of bits which tell computer system to perform an operation



The type of operation to be performed is determined by opcode



2^m types of instructions supported.

Instruction Set Architecture (ISI)

Collection of all instructions supported by CPU

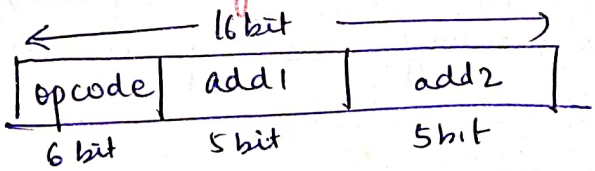
Types of instruction based on operands.

- 3 Address Instruction (max 3 addresses used)
- 2 Address Instruction (max 2 addresses used)
- 1 Address Instruction (max 1 address used)
- 0 Address Instruction (max 0 address used) → 2nd operand is accumulator & result is stored in accumulator

↙ supported in accumulator based architecture

stack based architecture.

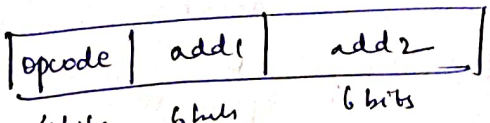
Q:- Consider a digital computer which supports only 2 address instructions each with 16 bits. If address length is 5 bits then maximum & minimum how many instructions the system can support?



Max. no. of instruction $= 2^6 = 64$

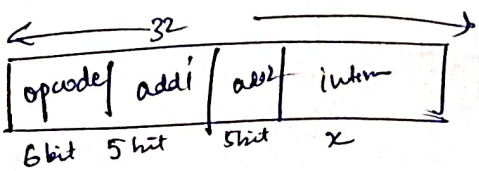
Min $= 1$

Q:- Consider a digital computer which supports 16 2 address instructions If address length is 6 bits then, the length of the instruction is ——



Size of instruction = 16 bits.

Q:- A processor has 40 distinct instructions and 24 general purpose registers. A 32 bit instruction word has an opcode, 2 register operands and an intermediate operand. The no. of bits available for intermediate operand is ——



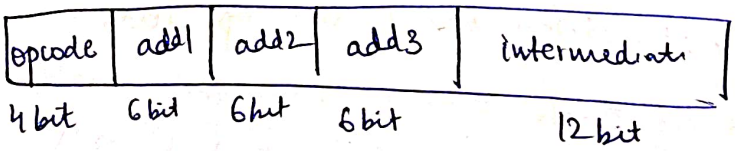
$x = 32 - (6+5+5)$

$= 32 - 16 = 16$ bits

Gate 2016

Consider a processor with 64 registers and an instruction set of size 12. Each instruction has 5 distinct fields — opcode, 2 source register identifier, one destination register identifier and a 12 bit immediate value. Each instruction must be stored in memory in byte aligned fashion.

If 100 instructions in program, the amount of memory (in bytes) consumed by the program text is ——?



length of instruction = 4 + 6 × 3 + 12
= 34

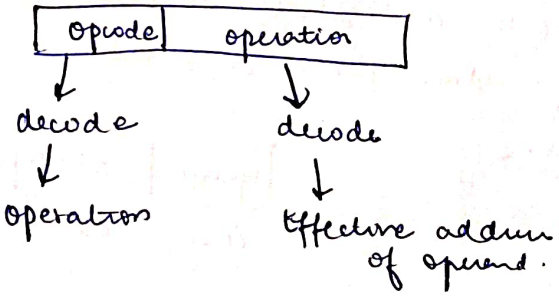
Since memory is byte aligned,
∴ 5 bytes are required to store 1 ins.

Memory req. to store 100 instructions = 100 × 5 = 500 bytes

Effective Address: address in memory where operand is stored.

How instruction is executed?

① Instruction Fetch
② Instruction Decode
③ Effective Addrn Calculation
④ Operand Fetch
⑤ Execute
⑥ Write Back Result



Instruction cycle.

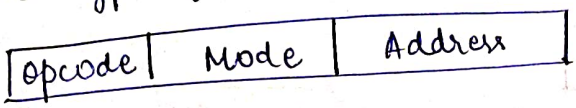
All phases are not required for all types of instructions.

Fetch, Decode, Execute } Mandatory in all instructions

Effective Addrn calculation, Operand fetch, Write Back Result } Optional in some instructions

Addressing modes

Tells the type of address in the instruction.



It specifies how and from where the operands are obtained ~~from~~ for the instruction.

Non computable modes ie. no computation required.

1. Implied mode
2. Immediate mode
3. Direct Addressing mode
4. Indirect Addressing mode
5. Register mode.
6. Register Indirect mode
7. Autoincrement & Autodecrement mode.
8. Index Register mode
9. PC Relative mode.
10. Base Register mode.

1. Implied mode

The opcode definition itself defines the operand.

Example → INCA (Increment accumulator)

0 address instruction.

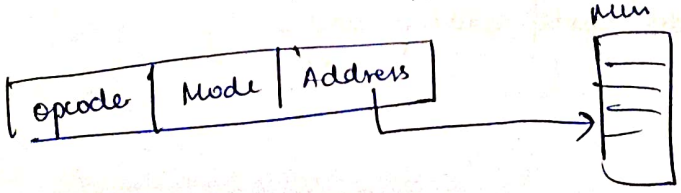
2. Immediate mode

Address field of instruction gives operand value.

↳ used to initialize registers with constant values

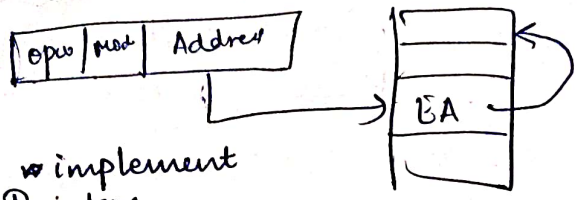
3. Direct mode

Address field specifies the effective address (address of memory)



4. Indirect mode

Address field of instruction specifies the address of effective addr

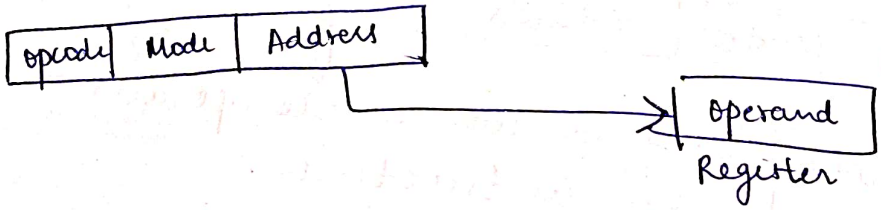


No. of memory accesses to get operand = 2.

to implement
Ex → Pointers

5. Register mode

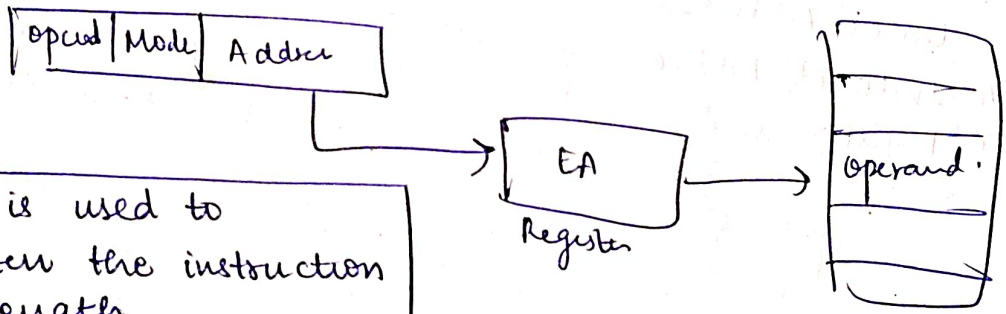
Address field specifies the register which holds the operand.



6. Register Indirect mode

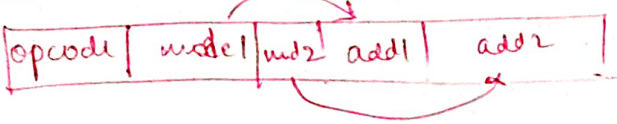
Address field specifies the address of register which holds effective address

1 register + memory access required

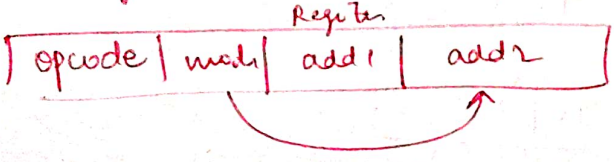


Mode is used to shorten the instruction length

2 address instruction format



Register memory architecture → one address is fixed as register.



(7) Auto increment / Auto decrement mode

- Variant of register indirect mode.
- Content of register is automatically increment / decremented access the sequential data.

Auto increment → Post Increment → perform operation then increment value of register.

Autodecrement → Pre decrement → first decrement then use the value of register

(8) Indexed / Index Register mode.

- used to access a single element of array.
- Effective Address of operand = Value in address field + Value in index register.

(9) PC - Related mode / Position independent mode.

- used for branch instruction [intra segment branching]

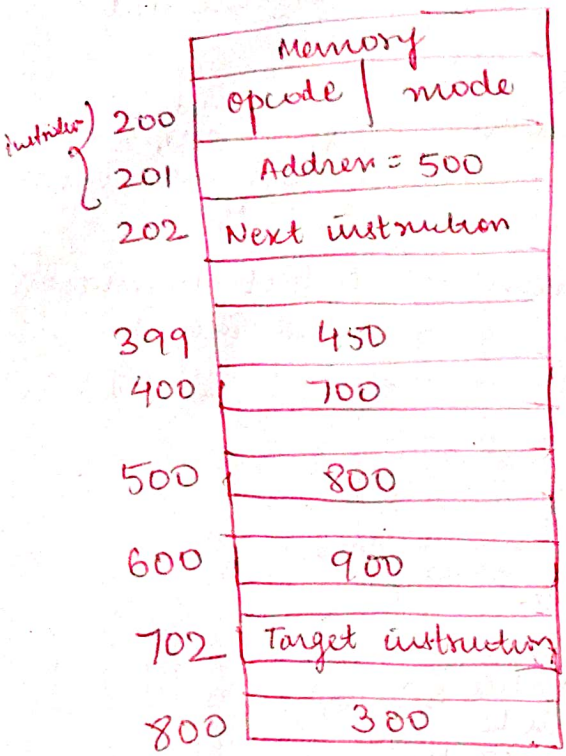
Effective address = Value in address field + Value in program counter

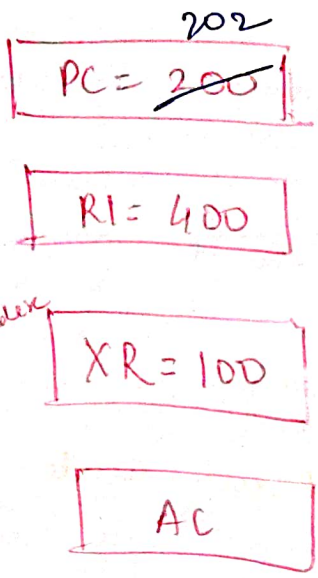
(10) Base Register mode

- used for intersegment branching.
- Base Register value is added to address field of instruction to get effective address.

Effective address
~~fetch~~ address
~~fetch~~ next
instruction ~~fetch~~

Example





After instruction fetch, PC = 202

Mode	Effective address	Operand
Immediate mode	201	500
Direct mode	500	800
Indirect mode	800	300
Register mode	←	400
Register Indirect mode	400	700
Autodecrement mode	399	450
Indexed mode	600	900
PC-Relative mode	702	~~400~~ Target

Q:- An instruction is stored at location 300 and its address field at location 301. The address field has the value 250. A processor register R1 contains number 200. Evaluate effective addrn. if addrng mode is

1) Direct — EA = 250
2) Relative (PC relative) — EA = 302 + 250 = 552
3) Register indirect — EA = 200

Q:- A relative branch mode type instruction is stored in memory at address 300. The branch is made to an address 450.

1. What should be the value of relative addrn field of the instruction?
2. Determine the value of PC before instruction fetch, after the fetch and after execution phase.

1. 149

2. Before instruction fetch = 300
   After fetch = 301
   After execute = 450

Q:- Consider a RISC machine where each instruction is exactly 4 bytes long. Conditional and unconditional branch instruction use PC - relative addressing mode with the offset specified in bytes to the target location of the branch instruction. Consider the following instruction sequence

Ins No.	Ins
i	add R2 R3 R4
i+1	sub R5 R6 R7
i+2	cmp R1 R9 R10
i+3	beq R1, offset

Branch ins.
PC value = i+4

$Target \overset{*4}{=} PC \overset{*4}{+} offset$

$offset = (Target - PC)*4$

$= (i - (i+4))*4$

$= -16$

Central Processing Unit (CPU)

It is the brain of a computer, containing all the circuitry needed to process input, store data and give outputs results.

CPU cycle time: Time CPU takes to perform a micro operation.

$$\text{CPU cycle time} = \frac{1}{\text{clock rate}}$$ → given in GHz.

CPI = ~~cycle~~ cycles Per Instruction

$$\text{Time taken to execute instruction} = \text{CPI}_{\text{avg}} * \text{CPU cycle time} = \frac{\text{CPI}}{\text{clock rate}}$$

Q: Consider computing the overall CPI for a machine A for which the following performing measures were recorded when executing a set of benchmark programs. Assume that the clock rate = 200MHz.

Instruction Category	% of occurance	No. of cycles per ins.
ALU	48	1
Load & store	10	3
Branch	39	4
Other	3	5

$$\text{CPI}_{avg} = 1 \times 0.48 + 3 \times 0.1 + 4 \times 0.39 + 5 \times 0.03$$

$$= 0.48 + 0.3 + 1.56 + 0.15$$

$$= 2.49$$

0.48
0.30
1.56
0.15
2.49

MIPS (Million Instructions per second)

- used to give CPU's performance

Gate 2014.

Consider two processors P1 and P2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P2 takes 25% less time but incurs 20% more CPI as compared to the program running on P1. If the clock frequency of P1 is 1GHz then the clock frequency of P2 (in GHz) is _____ ?

No. of instructions in P_1 and P_2 are same.

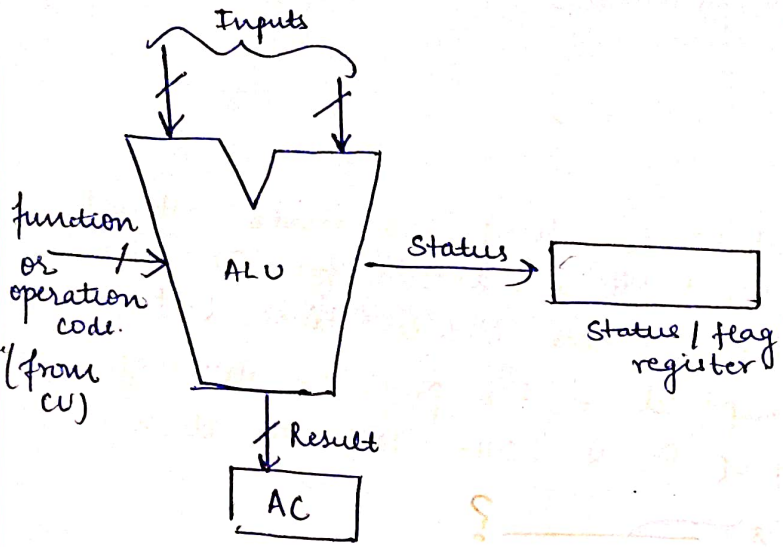
	P_1	P_2
Execution time	t_1	$0.75\,t_1$
CPI	c_1	$1.2\,c_1$
Freq.	1GHz	?

$$\text{No. of instructions executed, } n = \frac{\text{Execution time} * \text{clock rate}}{\text{CPI}}$$

$$n_1 = n_2$$

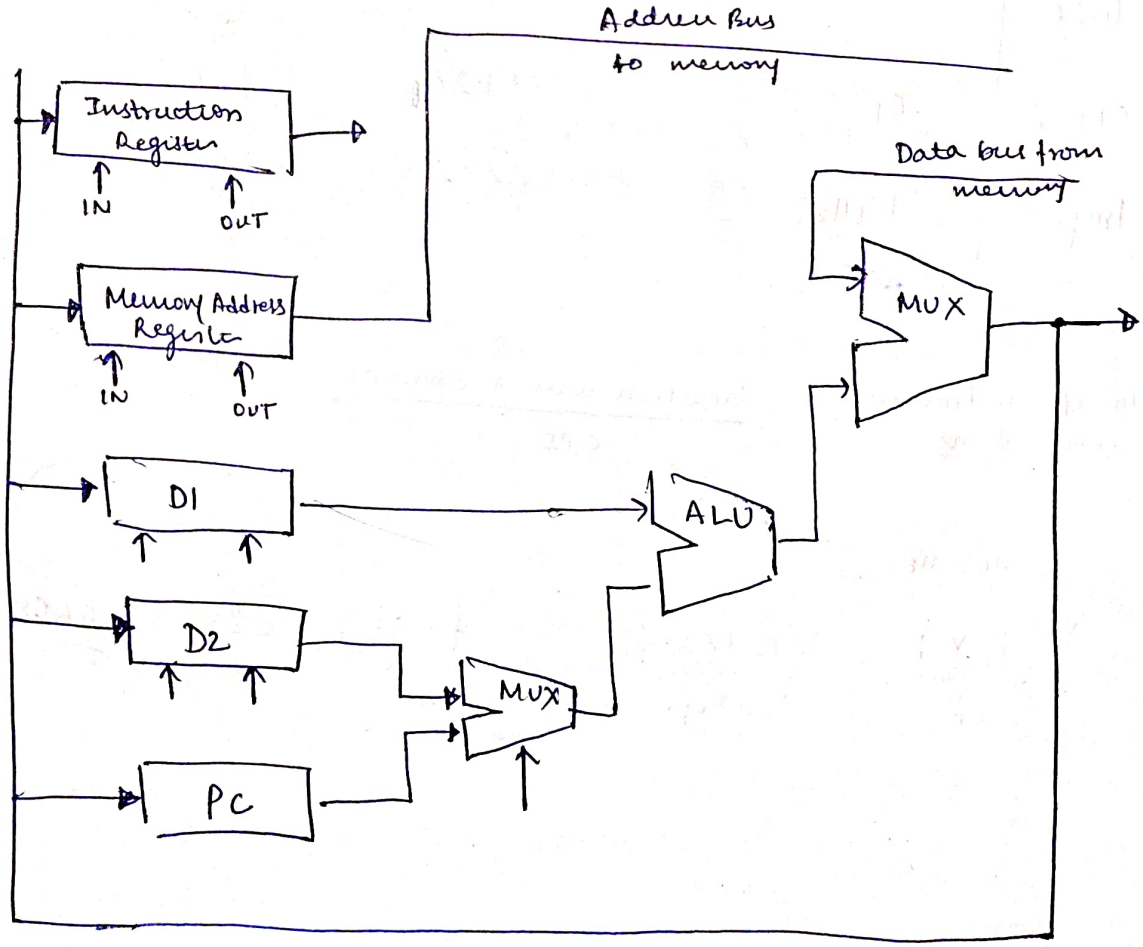
$$\Rightarrow \frac{t_1 \times 1}{c_1} = \frac{0.75\,t_1 \times f}{1.2\,c_1} \qquad \Rightarrow f = \frac{1.2}{0.75} = \frac{1.2 \times 4}{3} = 1.6\,\text{GHz}$$

ALU



Datapath (CPU Design)

- Collection of functional units such as ALU and multiplexers.
- Perform data processing operations.



Instruction Fetch –

IR ← M[PC]

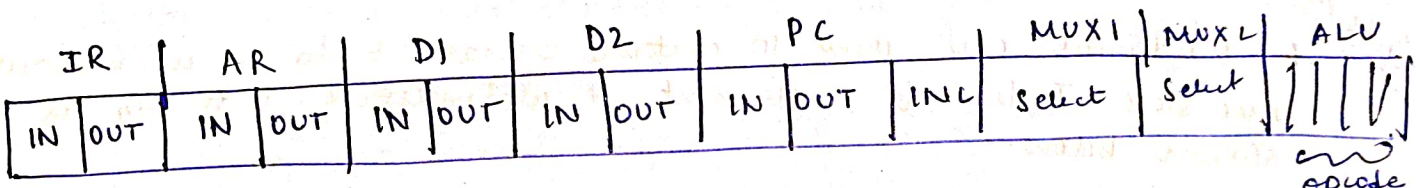
1. AR ← PC
2. IR ← M[AR] } mutually exclusive ∴ in same step.
3. PC ← PC + 1

Control unit

generates control signals and sends to various components of computer. All components operate according to these control signals.

Control ~~word~~ variable ~~signal~~ – name of control signal

Control word – collection of signals generated by control unit at once.



Control word ~~1. AR ← PC~~ ~~PCout~~

① Control unit generates control word.

② Control word signals are sent to different components.

③ Components perform micro operation based on control signal

Hardwired Control unit

Control logic is implemented with gates, flip flops, decoders & other digital circuits.

Advantage

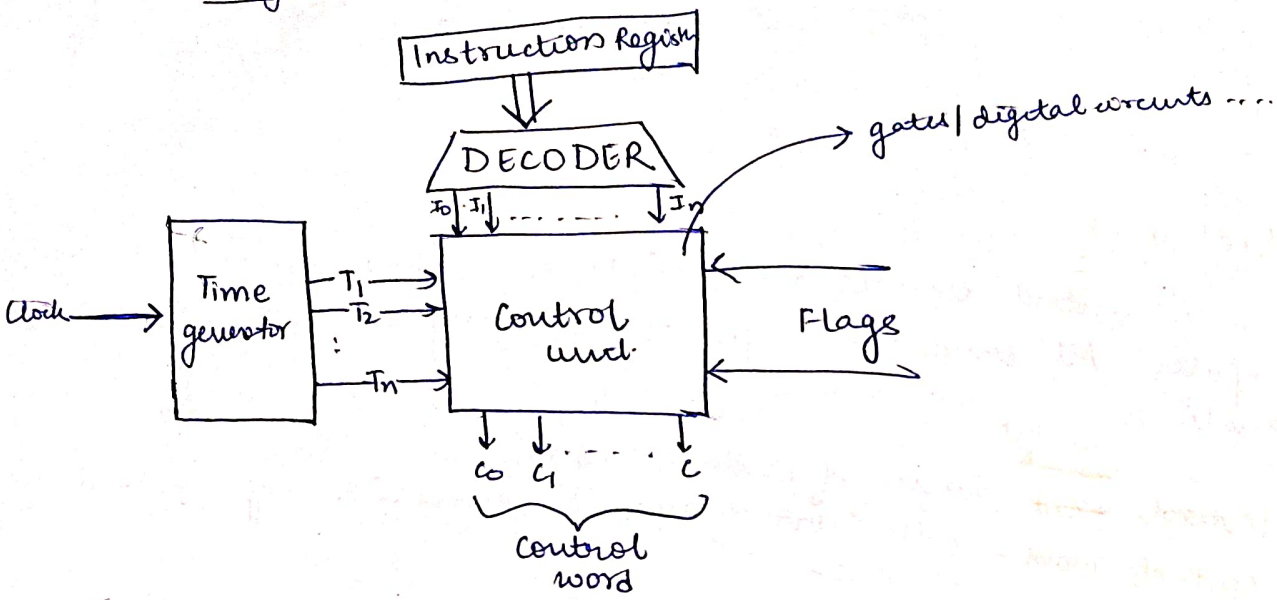
can be optimized to produce a faster mode of operation.

Disadvantage

Rearranging the wires among various components is difficult.

Updations in control logic is difficult.

Design of Hardwired Control unit



Gate 2005

A hardwired CPU uses 10 control signals S_1 to S_{10} in various time steps T_1 to T_5 to implement 4 instructions I_1 to I_4 as shown below—

	T_1	T_2	T_3	T_4	T_5
I_1	$S_1 S_3 S_5$	$S_2 S_4 S_6$	$S_1 S_7$	S_{10}	$S_3 S_8$
I_2	$S_1 S_3 S_5$	$S_8 S_9 S_{10}$	$S_6 S_8 S_7$	S_6	S_{10}
I_3	$S_1 S_3 S_5$	$S_7 S_8 S_{10}$	$S_2 S_6 S_9$	S_{10}	$S_1 S_3$
I_4	$S_1 S_3 S_5$	$S_2 S_6 S_7$	$S_5 S_{10}$	$S_6 S_9$	S_{10}

Which of the following pairs of expressions represent the circuit for generating control signals S_5 and S_{10} respectively?

A. $S_5 = T_1 + I_2 \cdot T_3$ and
$S_{10} = (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

B. $S_5 = T_1 + (I_2 + I_4) \cdot T_3$ and
$S_{10} = (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

C. $S_5 = T_1 + (I_2 + I_4) \cdot T_3$ and
$S_{10} = (I_2 + I_3 + I_4) \cdot T_2 + (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

D ✓ $S_5 = T_1 + (I_2 + I_4) \cdot T_3$ and
$S_{10} = (I_2 + I_3) \cdot T_2 + I_4 \cdot T_3 + (I_1 + I_3) \cdot T_4 + (I_2 + I_4) \cdot T_5$

$S_5 = T_1 + T_3 I_2 + T_3 \cdot I_4 = T_1 + (I_2 + I_4) T_3$

$S_{10} = (I_2 + I_3) \cdot T_2 + I_4 \cdot I_3 + (I_1 + I_3) T_4 + (I_2 + I_4) T_5$

Microprogrammed Control unit

① Control logic is implemented with micro-programs.
② All possible control words are stored in memory & based on requirement word is sent to different components.

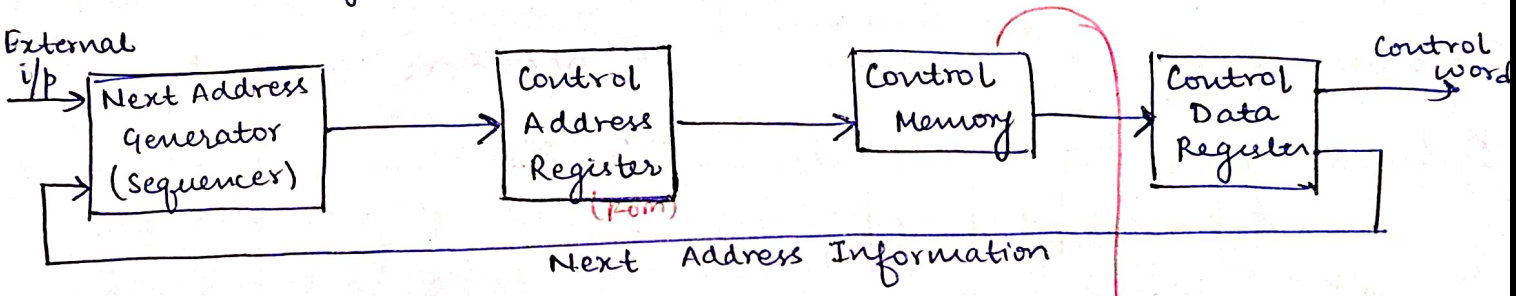
Advantage

updating the control logic is easy

Disadvantage

slower mechanism (because memory is involved)

control memory — control words are stored.



Each row stores the control word + next word information.

Control word	Next address

Micro-instruction

Standard micro instruction format

Signals (Control word)	MUX select	Next address

Control word sequencing.

Q:- Consider a CPU
1. 120 control signal bits
2. Microprogrammed CU micro-instruction

Signals	MUX select	Next add

3. 16 inputs MUX
4. 16 distinct MI's supported by CPU
64 microoperations per instruction

No. of bit in signal = 120
No. of bits in MUX sel = 4
Total no of micro-instructions in control mem = 16 × 64
$= 2^4 \times 2^6 = 2^{10}$
∴ Next add → 10 bits
∴ Size of microinst = 120+4+10 = 134
∴ Size of memory = 134×2^{10} bits
= 134 kB

Types of micro programmed

Horizontal

- One bit for each control signal
- Larger control word.
- No decoder
- Faster

Vertical

- Signals are divided into multiple groups such that at a time only 1 signal can be active.
  Each group's info is stored in an encoded form
- Smaller control word.
- Decoders are used.
- Slower.

Gate 2002

Horizontal microprogramming:

(A) Does not require use of signal decoders.
(B) Results in larger sized micro instructions than vertical
(C) uses one bit for each control signal
✓ (D) all of the above

Gate 1999

During ~~Decreasing~~ order of operating speeds —

Vertical < Horizontal < Hardwired.

Ques

A control unit generates 120 control signals, which are divided into 6 groups of mutually exclusive signals as below:

Group 1: 30	5X32 Decoder	No. of bits req. in vertical = 5+4+4+2+5+6 = 26 bits
Group 2: 13	4X16 Decoder	
Group 3: 12	4X16 Decoder	No. of bits req. in horizontal = 120 bits
Group 4: 3	2X4 Decoder	
Group 5: 27	5X32 Decoder	∴ No. of bits saved = 120 - 26 = 94 bits
Group 6: 35	6X64 Decoder	

How many bits are saved by using vertical microprogrammed control unit as compared to horizontal?

Types of CPU Design

RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
Less number of instructions.	More number of instructions.
Fixed length instructions (∵ variable length opcode)	Variable length instructions (∵ fixed length opcode)
Simple instructions	Complex instructions
Limited addressing modes.	More & complex addressing modes.
Hardwired CU	Microprogrammed CU
Only Register – Register arithmatic operations	Register – memory & memory to operations
More no. of registers	Less no. of registers.

<Preferable for piplined CPU>

Gate 2018

Consider the following processor design characteristics —

I. Register to register arithmetic operations only.
II. Fixed length instruction format.
III. Hardwired control unit.

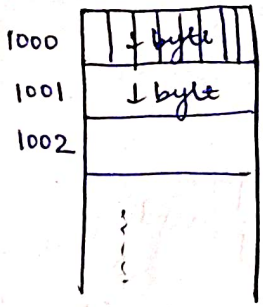
Which of the ~~following~~ characteristics above are used in design of RISC processor.

A) I & II only
(B) II and III only
(C) I and III only
(D) I, II and III

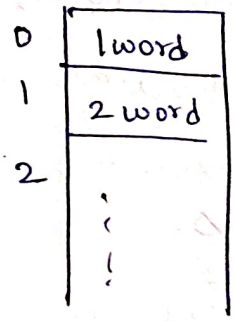
Byte v/s word addressable memory

Addressable meaning –

किस unit को एक unique address दिया गया है।
क्या वो byte है? या वो word है?
जानने के लिए बने रहिए हमारे साथ



byte addressable



word addressable
where word size = 32 bit

Byte addressable

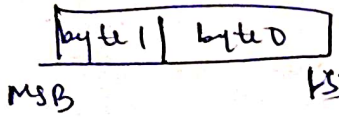
Address of first word = 0
Address of second word = 4
Address of third word = 8
⋮

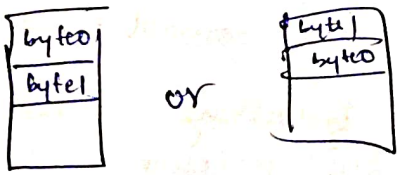
Word addressable

Address of first word = 0
Address of 2nd word = 1
Address of 3rd word = 2
⋮

Byte Ordering

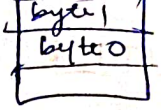
How multibyte data is stored in memory?
from LSB to MSB or from MSB to LSB





This ordering is decided by byte ordering.

Big Endian — MSB bytes are stored first



Little Endian — LSB bytes are stored first

Q:- 16 bit word — Assume 1 word = 2 bytes, memory is Byte addressable

11 00 1101	0001 0101

Big Endian

1100 1101
0001 0101

Little Endian

0001 0101
1100 1101

Q:- Assume 1 word = 4 bytes

word = 4ACD539D

Big Endian

0100 1010
1101 1110
0101 0011
1001 1101

little Endian

1001 1101
0101 0011
1101 1110
0100 1010

ARCHITECTURE खत्म !!

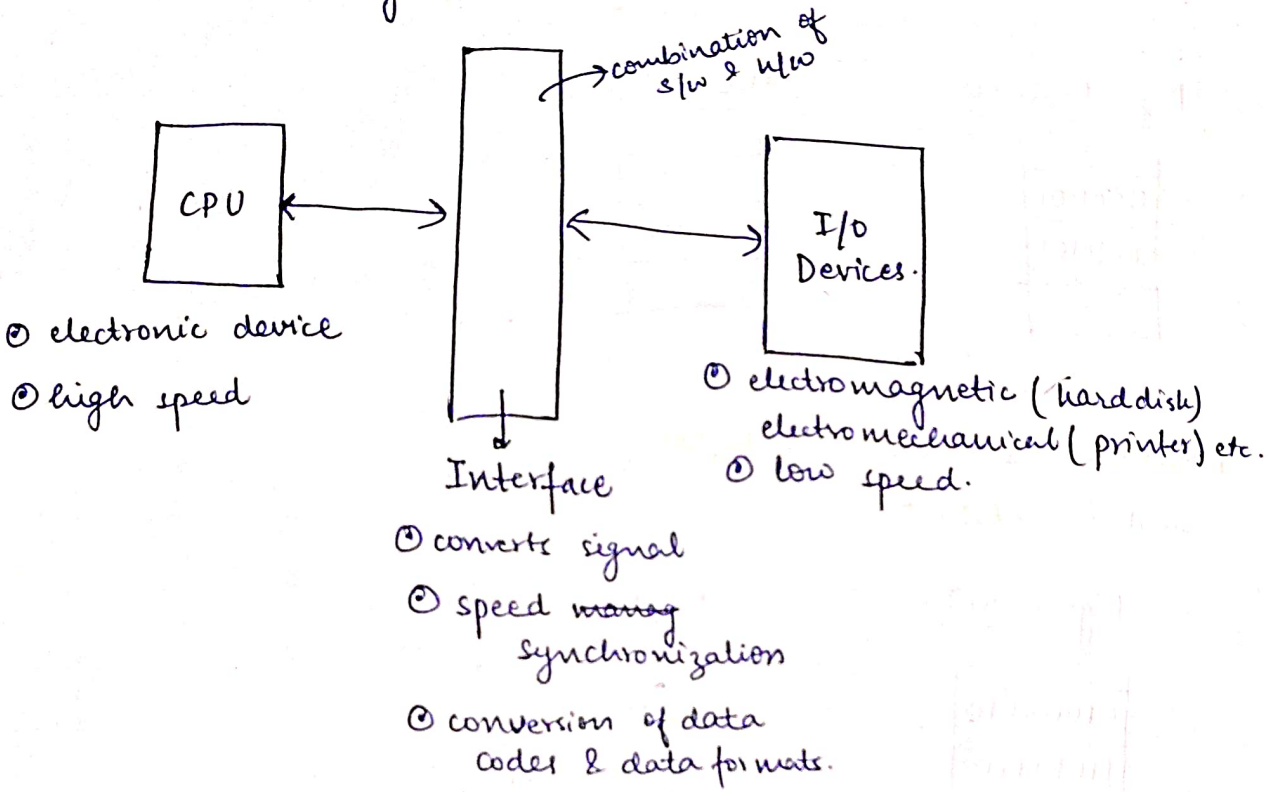
COMPUTER ORGANIZATION

IO Organization

Peripheral device:- Devices connected to processor externally.

- Input Devices
- Output Devices
- Storage Devices

} I/O Devices



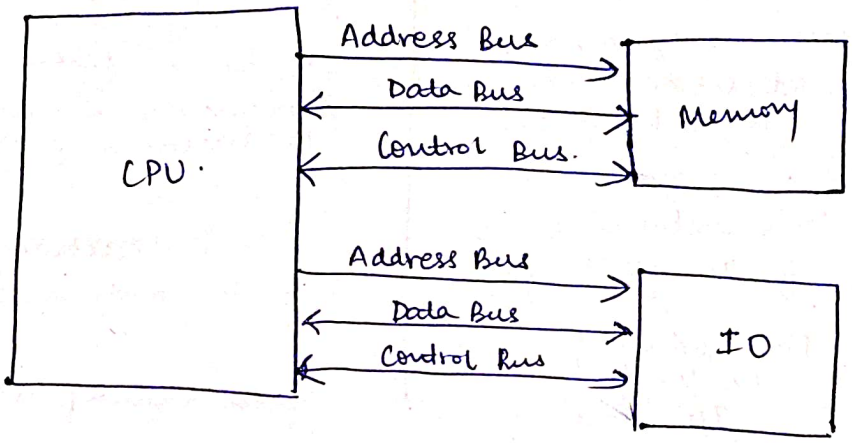
IO Interface

- DMA Controller
  - DMA + interface
- IO Processor
  - Interface + DMA + I/O instn executio } Graphic card.

IO v/s memory bus

⊙ 3 ways to connect memory and I/O with CPU through buses.

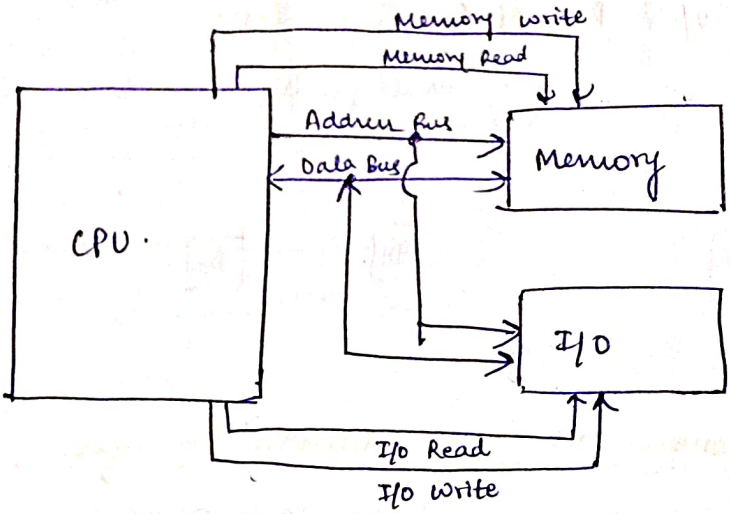
1) Separate Buses for Both



Disadvantage
↳ costly (bcoz more buses are required)

Advantage
↳ easy to design

2) Common Data, Address Bus



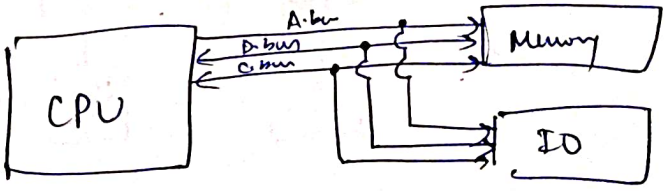
Advantage
↳ no memory wastage

CPU distinguishes b/w memory & I/O using control signal.

I/O Devices have their own addresses

∴ I/o mapped I/O
∴ Isolated IO

3) Common Address, Data & Control Bus



Disadvantage
↳ memory wastage

Advantage

✗ IO addresses do not have their own address known as Memory mapped IO

memory addresses are assigned to I/O device. Wherever CPU addresses 101, it goes directly to IO

correct

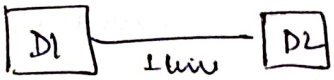
Some locations in memory are used to store address of I/O.

So, that CPU can access I/O devices from those assigned addresses.

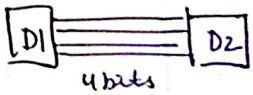
Memory mapped I/O	IO Mapped IO
Memory wastage	No memory wastage
No separate address space for IO	Separate address space for IO
All memory access instructions are used by I/O also	IO access instructions & memory access instructions are different.
More instructions for IO access	Less instruction
More addressing modes for I/O access	Less addressing modes

Serial v/s Parallel Transfer.

1 bit at a time
slow

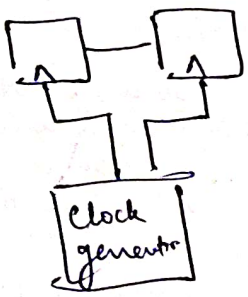


multiple bits at a time
fast
costly

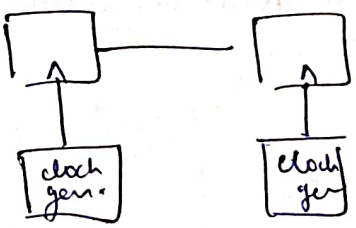


Synchronous v/s asynchronous transfer.

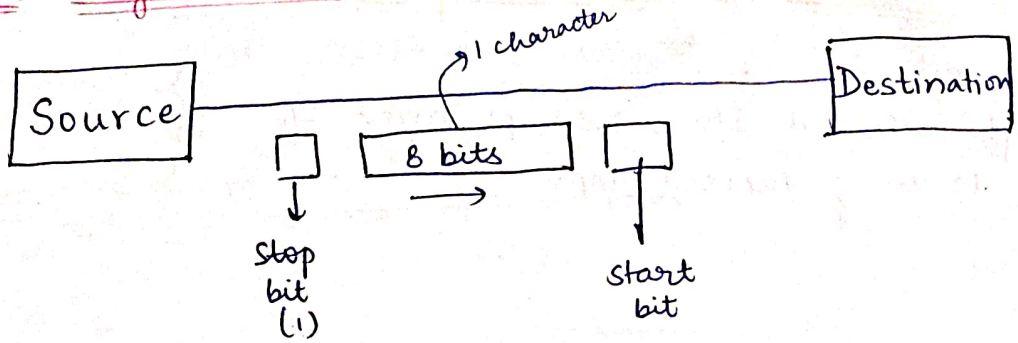
all devices operate on single clock



~~separ~~ each device has its own clock.



Serial Asynchronous Transfer



- No transfer means 1 state.
- 0 (start bit) detected means transmission started
- character bits follow start bit.
- After last bit of character '1' (stop bit) will be sent.

$$\text{Efficiency of transmission line} = \frac{8}{10} = 0.8 = 80\%$$

Q. How many 8 bit characters can be transmitted per second over 9600 bits/second serial communication link using a parity synchronous mode of transmission with 1 start bit, 8 data bits, & 2 stop bits and 1 parity bit

To send one 8 bit character, 1+8+2+1 = 12 bits are required.

9600 bits can be transferred in 1 sec

∴ No of characters = $\frac{9600}{12}$ = 800 characters/sec.

Q. An asynchronous serial communication is employing 8 character bits, 1 parity bit, 2 start bits & 1 stop bit. To maintain rate of 400 char/sec the minimum transfer rate required is _____?

1 char → 1+2+1+8 = 12 bits

400 char/sec

= 400 × 12 bits/sec

= 4800 bits/sec

Modes of data transfer

→ Programmed IO or Program controlled I/O
→ Interrupt initiated I/O or Interrupt Driven I/O
→ Direct Memory Access (DMA)

} Data transfer b/w CPU & I/O
} Data transfer b/w I/O & memory.

Programmed IO

- No provision through which IO can inform to CPU about data transfer.
- IO sets its own status bit and waits
- CPU runs a program periodically and checks the status of each device one by one.
- If any device has its status ~~set~~ set, then, CPU performs data transfer

 → time wastage of CPU

Time required in Programmed IO

Total time =	Time to read status of I/O device	+	Time taken by CPU to check if status flag is set or not	+	Data transfer time.
	↓ depends upon IO speed		↓ almost 0		

By default, status flag = 1 byte.

Ques Consider a device which operates with 20 MBPS operating speed. The device is operating on program controlled mode of IO & it has to transfer a data of 20 B from it. Data is transfered byte wise.
Size of status register = 2 bytes
Total time needed to perform data transfer in microseconds?

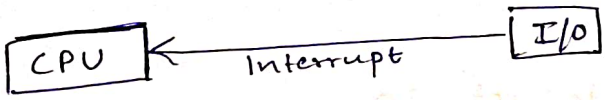
~~Wrong~~
To transfer 1 byte, 3 bytes are needed [bcoz byte wise transfer status = 2B]
To transfer 20 B, 60 bytes are needed

∴ Time req. = $\frac{60}{20 \times 10^6}$ s = 3 μs

Total time = 2+20 bytes = 22 bytes
Total time = $\frac{22}{20}$ μs = 1.1 μs

Interrupt IO

- IO device has a provision (Interrupt signal) to inform to CPU about communication.
- When CPU receives interrupt:
  - → Completes execution of current instruction.
  - → saves the status (PC, PSW etc.) of current process onto the stack.
  - → Branches to service the interrupt.
  - → Resumes the previous process by taking out the value from stack.



ISR = Interrupt Service Routine [CPU executes ISR when it receives interrupt]

Vector Address → address or location of ISR. (reference)

Interrupt

Vectored interrupt

Device sends interrupt and vector.

[Vector address is sent only after interrupt is acknowledged by CPU]

Data bus is used to send vector.

Interrupt → control signal.

Non-vectored interrupt (Scalar)

Device sends only interrupt

→ CPU executes default service routine (OR)

↓

CPU obtains location of actual ISR.

Maskable interrupt

CPU can accept or reject the interrupt.

Non Maskable interrupt

CPU always accepts the interrupt.

Interrupt

- External interrupt
  - I/O Device generates the interrupt.
- Internal interrupt
  - Interrupt due to unexpected error during instruction execution.
  - (Ex → page fault) System call

Time Required in Interrupt IO

= Interrupt overhead time + Service time.

Simultaneous Interrupt

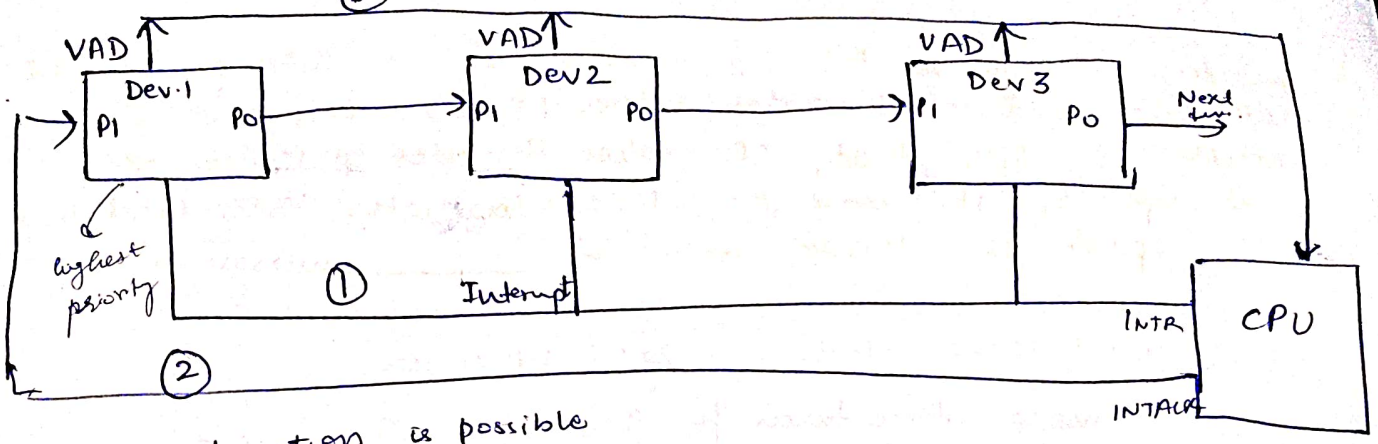
- 2 device generate interrupt simultaneously.
- The highest priority device's interrupt should be serviced first

Priority Based Interrupt Handling

- Software solution
  - ⇓
  - Run a program Resolve the priority
- Hardware solution
  - Serial
    - Daisy chaining
  - Parallel
    - X

VAD = Vector Address

Daisy chaining



Starvation is possible

Gate 2005

A device with data transfer rate 10 KB/sec is connected to a CPU. Data is transfered byte wise. Let the interrupt overhead be 4 microsecond. The byte transfer time b/w the device interface register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program controlled mode?

$$\text{Performance gain} = \frac{\text{performance of interrupt}}{\text{performance of programmed IO}} = \frac{\text{prog IO time}}{\text{Interrupt I/O time.}}$$

or speed up.

For programmed I/O —

Status register = 1 byte (default)

Time to read status register = $\frac{1}{10 \times 10^3} = 10^{-4}$ sec = 100 μs

Data transfer rate = 0

∴ Total programmed I/O time = 100 μs

For interrupt I/O —

Total time = overhead = 4 μs

∴ Performance gain = $\frac{100}{4} = 25$

Ques Consider a CPU which takes 2 microseconds as interrupt overhead time when a device generates interrupt for CPU, and CPU accepts it. After that, CPU takes 9 cycles to service the interrupt. If CPU runs on 2MHz clock rate, then, total time CPU spends for interrupt service is _____ microseconds.

Clock rate = 2MHz = 2×10^6 cycles/sec.

Service Time = time taken for 9 cycles = $\frac{9}{2 \times 10^6}$ = 4.5 μs

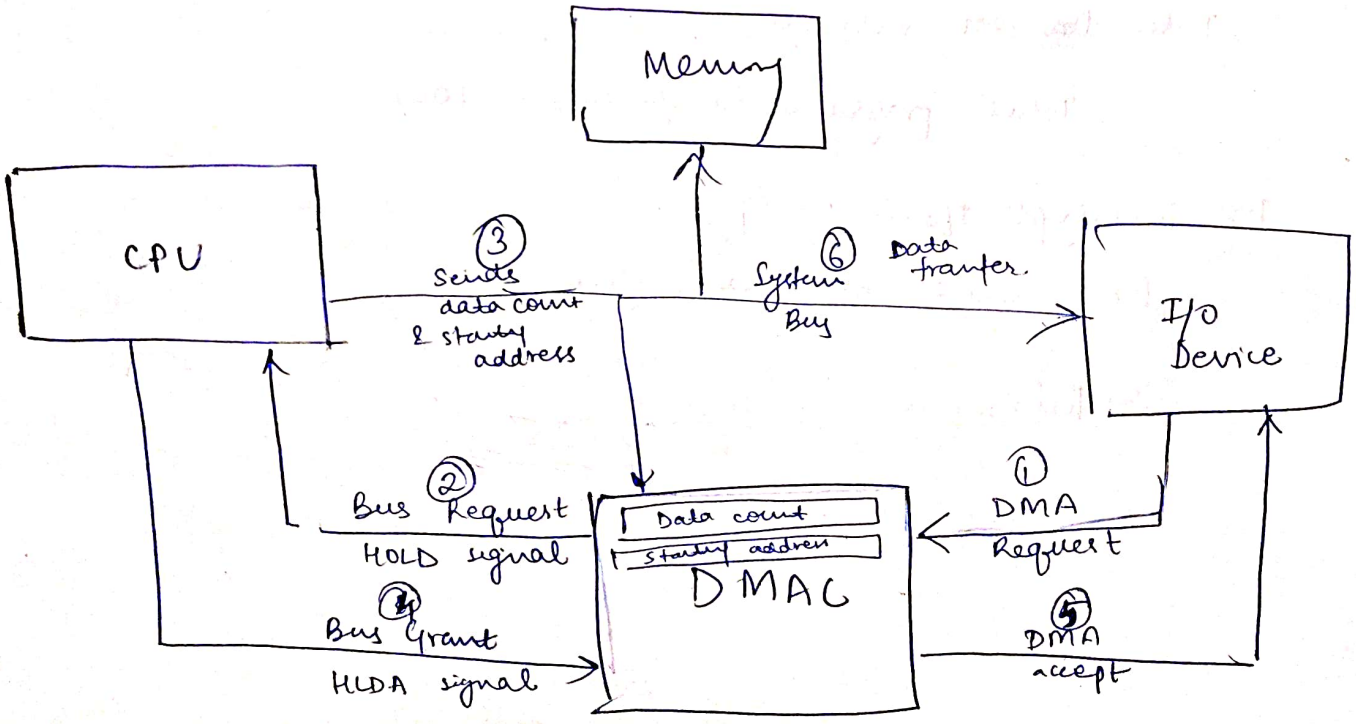
Interrupt overhead = 2 μs

∴ Total time = 2 + 4.5 = **6.5 μs**

DIRECT MEMORY ACCESS (DMA)

Enables data transfer operation b/w I/O device and memory without CPU intervention.

Need of hardware —— DMAC (DMA Controller)



Starting address = memory address starting from where the data transfer should be started.

Data Count = No. of bytes or words to be transferred.
↓
depends upon memory byte addressable / word addressable.

During DMA transfer, CPU can perform only those operations which do not require system bus
→ means CPU will be mostly blocked.

↓

time is saved bcoz there is no need to store present state of CPU in stack } it is a time consuming process.

means a lot!!

Gate 2016

The size of data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29154 kB from disk to main memory. Memory is byte addressable. The minimum number of times the DMA controller needs to get the control of system bus from the processor to transfer the file from the disk to main memory is _____.

Max. no of bytes that can be transfered if DMA controller has got the control of system bus = 2^{16} = [illegible] = [illegible].
= $(2^{16}-1)$ bytes
↳ -1 is necessary

Max. no. that can be represented by 16 bit = $2^{16}-1$

∴ No. of times control of system bus needed = $\frac{29154}{2^{16}-1}$ = 455.53 = 456

Modes of DMA Transfer

① Entire data is not transferred at once from I/O to memory.
② For sometime, DMAC has control of bus then CPU takes back the control again } to enhance user experience Screen freeze ho jaegi if CPU is blocked
gives it back to ② DMAC

- Burst Mode
- Cycle Stealing mode
- Interleaving mode.

BURST MODE

A burst of data (1kB - 2kB generally) is transferred in one time before CPU takes back the control of the buses.

CYCLE STEALING MODE

→ Slow I/O device takes some time to prepare a byte or word.
→ During this time CPU keeps the control of the buses.
→ Whenever the word/byte is ready, then, CPU gives the control of the buses to DMAC for 1 memory cycle.
→ During this cycle, the word/byte is transferred to memory.

INTERLEAVING MODE

CPU will never be in waiting mode.

Whenever CPU performs internal operations and does not require the buses then DMAC will be given control of the buses.

% of time CPU is blocked

Time required to prepare the data in I/O = t_x
Time required to transfer the data to memory = t_y

% of time CPU is blocked due to DMA	$= \frac{t_y}{t_x + t_y}$	Burst mode
% of time CPU is blocked	$= \frac{t_y}{t_x}$	cycle stealing mode.
% of time CPU is blocked	$= 0$	Interleaving mode

t_x is dependent on the speed of I/o device. (given in ques)
t_y can be given in question or depends upon cycle time.

Ques Consider a device operating on 2MBPS speed & transferring the data to memory using cycle stealing mode of DMA. If it takes 2μs to transfer 16B of data to memory when it is ready/ prepared, then, % of time CPU is blocked due to DMA is ——?

Suppose x B of data is to be transferred.

$t_x = \frac{x}{16} \times 2\,\mu s$

$t_y = \frac{x}{2}\,\mu s$

$\therefore \% = \frac{t_y}{t_x} = \frac{x \times 16}{2 \times x \times 2} = 4\%$

16 bytes of data is to be transferred.

t_y = time taken to transfer = 2μs

$t_x = \frac{16}{2} \times 2\,\mu s = 8\,\mu s$

$\therefore$ % of time CPU is blocked $= \frac{t_y}{t_x} = \frac{2}{8} = \frac{1}{4} = 25\%$

Ques A device is constantly transferring the data with 40 MBps speed to memory using DMA. Assume that the transmission time to memory from IO once the data (4 bytes) is ready is 10 μs.

1. % of time CPU is blocked in burst mode.
2. % of time CPU is blocked in cycle stealing mode

Data size = 4 bytes

$t_y = \frac{4 \text{ bytes}}{10\mu s}$ $\quad t_x = \frac{4}{40 \times 10^3}\ s = \frac{1}{10^4}\ s = 100\mu s$

1) % of time CPU is blocked in burst mode $= \frac{t_y}{t_x + t_y} = \frac{10}{10+100} = \frac{10}{110} = \frac{1}{11}$

$= \frac{100}{11}\% = \underline{9.09\%}$

2) % of time CPU is blocked in cycle stealing mode $= \frac{t_y}{t_x} = \frac{10}{100} = \underline{10\%}$

Gate 2021

Consider a ~~DMA support~~ computer system with DMA support. The DMA module is transferring one 8 bit character in one CPU cycle from a device to memory through cycle stealing at regular intervals.

Consider a 2 MHz processor.

If 0.5% cycles are used for DMA, the data transfer rate of the device is _____ bits per second.

8 bit character in 1 CPU cycle

2 MHz processor $\Rightarrow$ 2×10^6 cycles per second

$\frac{0.5}{100} \times 2 \times 10^6$ cycles are used for DMA [50%.]

$\Rightarrow 10^4$ cycles are used for DMA per second

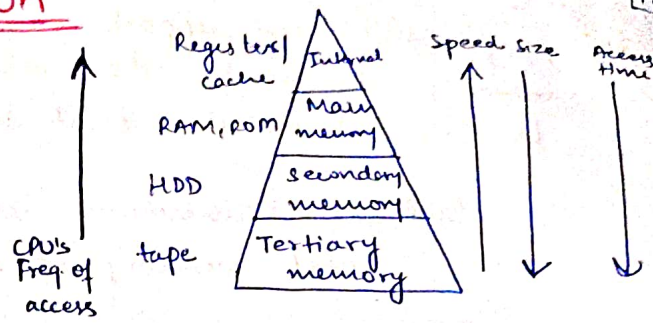
1 cycle $\longrightarrow$ 8 bit data

$\therefore 10^4$ cycles $\longrightarrow 8 \times 10^4$ bit data

$\therefore$ Data transfer rate = $\underline{80000 \text{ bits/sec}}$

Memory Organization

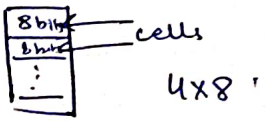
Memory hierarchy is used
- To maximize access speed.
- To minimize per bit storage cost.



Memory Representation

= No. of cells × Each cell capacity.

= No. of memory locations × Bits per location



Default unit of storage = Bits (if not given in question)

Ques Consider a CPU with 13 bits address bus and 16 bits in data bus. Maximum memory size which can be supported is ___16___ kbytes

$2^{13} \times 2^{16} = 2^{29}$ bits $= 2^{13} \times 16$ bits

$= 2^{14}$ bytes $=$ 16kB

Ques A 32 bit wide memory has 24 bits address to be accessed. Maximum memory capacity = _____

Size of each memory location = 32 bit

No. of memory locations $= 2^{24}$

∴ Maximum memory capacity $= 2^{24} \times 32 = 2^{29}$ bits

> Memory cycle time = time req. to perform read / write operation in 1 location.

Ques The memory cycle time of a memory is 200 nsec. The maximum rate with which the memory can accessed _____ bytes/sec.

Note: byte addressable

200 nsec ⟶ 1 byte data is accessed

∴ Rate $= \frac{1}{200 \times 10^{-9}}$ bytes/sec $\frac{10^9}{200}$ bytes/sec $= 5 \times 10^6$ bytes/sec

Gate 2016

A processor can support a maximum memory of 4GB. where memory is word addressable (2 byte word). The size of address bus of the processor is at least 31 bits.

Size of 1 memory location = 2 bytes

Total memory size = 4GB = 4×2^{30} bytes

$\therefore$ No. of locations = $\frac{4 \times 2^{30}}{2} = 2 \times 2^{30} = 2^{31}$

$\therefore$ No. of bits in address bus = $\log_2(2^{31})$ 31 bits

Main memory

Used to store currently running programs and their data.

RAM Random Access Memory (volatile)

ROM Read Only Memory. (non-volatile)

→ P.O.S.T (Power on self test) hardwares connected to CPU

Booting (loading of OS)

Bootstrap program = Program responsible for booting.

ROM is accessed only when power is turned off.

Once OS is loaded, ROM is not accessed again until next start.

Types of RAM

Static

- Mode = flip flops
- No refresh req.
- Faster
- Expensive
- used for cache
- High Power Consumption

Dynamic (DRAM)

- Storage in form of electric charge in Capacitors.
- Periodic refresh required.
- Slower
- less expensive.
- used for main memory.
- Low operational power consumption

Ques The amount of ROM needed to store the table of multiplication of two 4 bit unsigned integers is?

a) 64 bits
b) 128 bits
c) 1K bits
d) 2K bits. ✓

4 bit unsigned integers.

No. of bits in product = 4+4 = 8 bits.

No. of combinations of product = $2^4 * 2^4 = 2^8$

$\therefore$ Size of ROM = $2^8 \times 8 = 2^{11}$ bits

= 2K bits

Ques A ROM is used to store the table of multiplication of two 8 bit unsigned integers. The size of ROM required is

a) 256 x 16
b) 64K x 8
c) 4K x 16
d) 64K x 16 ✓

8 bit.

Size of each cell = No. of bits in product = 8+8 = 16 bits.

No. of combinations of product = $2^8 \times 2^8 = 2^{16}$

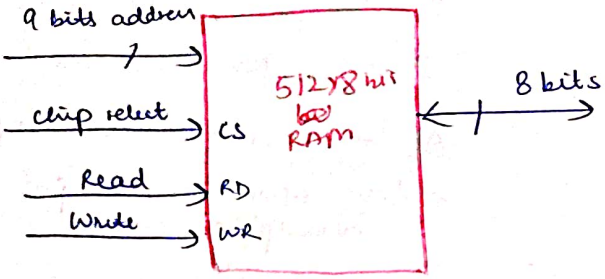
$\therefore$ Size of ROM = $2^{16} \times 16$

= 64K x 16

Decoder required to access memory of size 128K x 16 bits $= 17 \times 2^{17}$ decoder

$2^7 \times 2^{10} = 2^{17}$

RAM chip



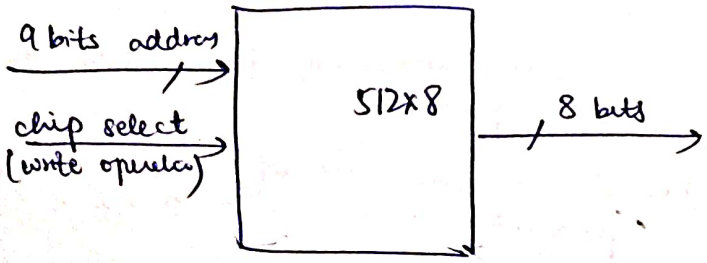
No. of pins required in RAM chip : $9+8+1+1+1$

: 20

CS	RD	WR	Operation
0	X	X	No operation
1	0	0	No operation
1	0	1	Write
1	1	X	Read

even if both RD & WR are 1, a read operation takes place.

ROM chip



No. of pins required = $8+9+1$

= 18

CS	Operation
0	No operation
1	~~write~~ read

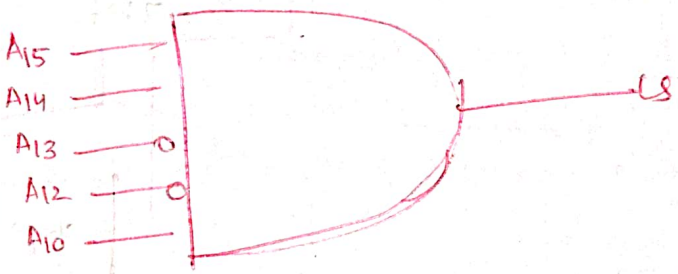
Ques A ROM chip of 2048×16 bits has four select inputs and operates from a 5 volt power supply. How many pins are needed for the IC package?

No. of pins for address line = 11
No. of pins for data line = 16
No. of pins for chip select = 4

2 connections are required →
→ power — 1 bit
→ ground — 1 bit

$\therefore$ No. of pins = $11+16+4+2 = 33$

Gate 2019 The chip select logic for a certain DRAM chip in a memory system design is shown below. Assume that the memory system has 16 address lines denoted by A_{15} to A_0. What is the range of addresses (in hexadecimal) of the memory system that get enabled by the chip select (CS) signal?



(A) C800 to CFFF
(B) CA00 to CAFF
(C) C800 to C8FF
(D) DA00 to DFFF

1100 1000 0000 0000 → Beginning
C 8 0 0

1100 1111 1111 1111 → Ending
C F F F

$\therefore$ C800 to CFFF

Multiple Chips in Single System

Total capacity of system = no of chips * capacity of 1 chip.

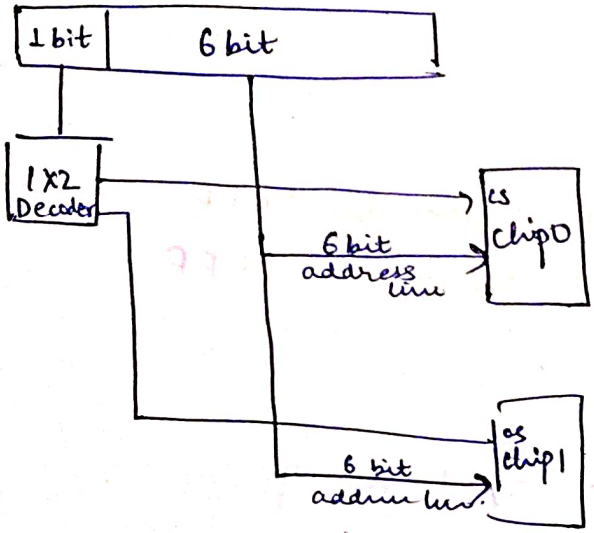
Vertical Arrangement

used when number of addresses required in memory system is more than individual chips.

Ex →
chip0 — RAM 64x8 bit
chip1 — RAM 64x8 bit

Total capacity = 128x8 bit.

7 bit address generated by CPU.

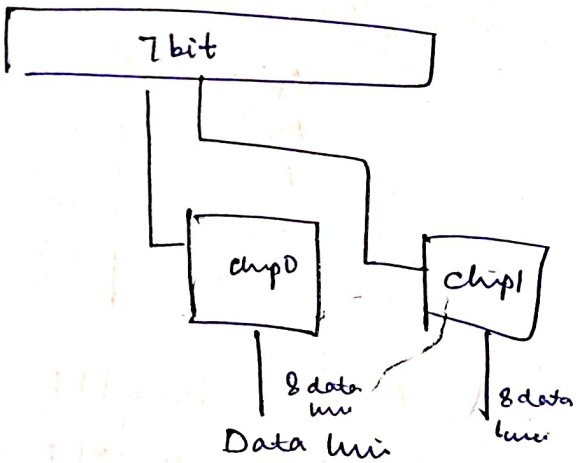


Horizontal Arrangement

No of data lines required are greater than number of data lines in each chip.

CPU generates 7 bit address.

RAM chip — 128 x 8 bit
Total capacity = 128 x 16 bit



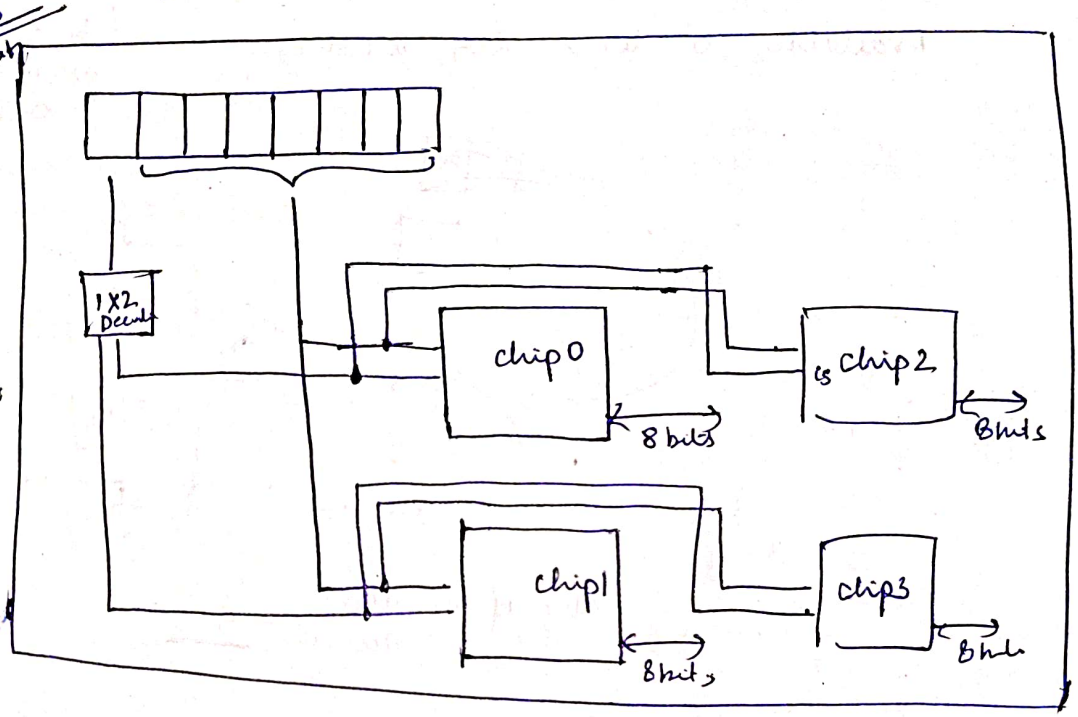
Used when data per location required is more than one chip location

Ques How many 128×8 bit RAM chips are needed to provide a memory capacity of 256×16 bits

No. of chips required $= \frac{256 \times 16}{128 \times 8} = 4$ RAM chips.

CPU generates 8 bit address.

256×16 bit



Hybrid Arrangement

Gate 2009 How many 32K×1 RAM chips are needed to provide a memory capacity of 256 KBytes

(A) 8 (B) 32

(C) 64 (D) 128

No. of RAM chips required $= \frac{256K \times 8 \text{ bits}}{32K \times 1 \text{ bits}}$

$= \frac{2^8 \times 2^3}{2^5} = 2^6$

$= 64$

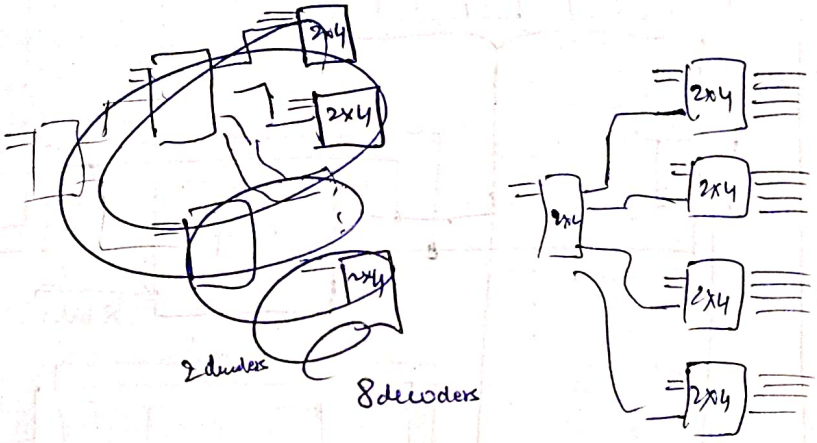
Gate 2013

A RAM chip has a capacity of 1024 words of 8 bits each (1K×8). The number of 2×4 decoders with enable line needed to construct a 16K × 16 RAM from 1K × 8 RAM is ——?

No. of RAM chips required = $\frac{16K \times 16}{1K \times 8}$ = 16 × 2 = 32 chips.

~~Number of 2×4~~ Decoder required = 4×16 ~~5 × 32~~ Decoder (bcoz 16 chips arranged vertically)

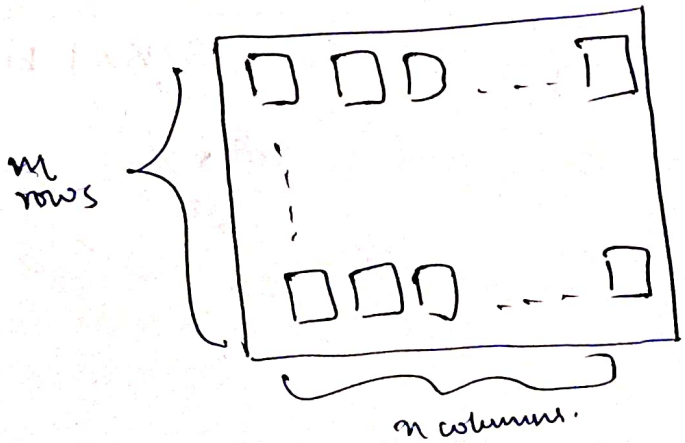
Available decoder = 2×4 decoder.



No. of decoders required = 5

DRAM Refresh

DRAM needs to be periodically refreshed because it is made up of capacitors which store data through electric charge.



In one refresh operation, entire row gets refreshed.

∴ Total time required for refresh 1 chip = No. of rows (m) × Time taken in 1 refresh operation

★ Time required to refresh n chips = 1 chip refresh-time

Ques A DRAM chip of 512K×8 bits has 1k rows of cells with 512 cells in each row. If DRAM takes 20ns for 1 refresh, then, total refresh time of the DRAM is _____ microseconds.

No. of rows in DRAM chip = 1K = 2^{10}

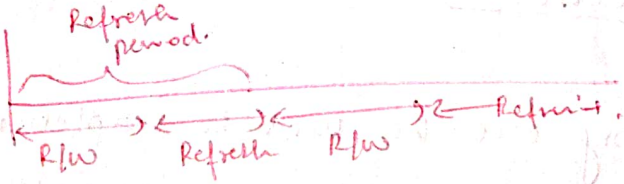
Time for 1 refresh = 20ns

Total refresh time = $\frac{1K}{2^{10}} \times 20\,ns$ ~~$= 2 \times 20 \times 10^{-9}\,s$~~

$= 20\,\mu s$ = 0

Refresh period

Time period in which the DRAM chip should be refreshed atleast once



Gate 2018

A 32 bit wide memory unit with a capacity of 1GB is built using 256M × 4 bit RAM chips. The number of rows of memory cells is 2^{14}. Time taken to perform 1 refresh operation is 50 ns. The refresh period is 2 milli seconds. The percentage of time available for performing memory read/write operations is _____?

No. of rows = 2^{14}

Total refresh time = $2^{14} \times 50\,ns = 0.8192\,ms$ = ~~8192 ms~~

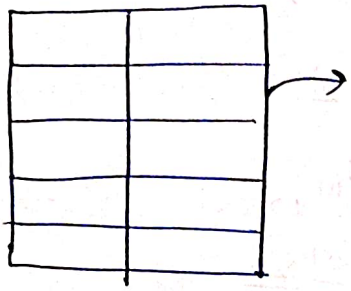
R/W in 1 refresh period = 2 ms − 0.8192 = 1.18 ms

% of time for r/w = $\frac{1.18}{2} \times 100 = 0.59 \times 100 = 59\%$

For solving question, $2^{10} = 1K \approx 1000$

Associative memory

- Known as content addressable memory.
- Cells do not addresses.



→ every cell is divided into 2 parts.
1st part is like address.
If CPU sends a content, it is compared against first part and if that's matches, 2nd part of the cell is sent.

- The searching takes place parallelly in all the cells.
  ∴ Associative memory is very fast. (faster than SRAM)
  ∴ expensive

- Associative memory can be used to implement TLB (Translation lookAhead Buffer.)

Ques The memory cycle time of a memory is 500nsec. The maximum rate with which memory can be accessed?
byte addressable

a) 500 Bytes/sec
b) 2000 bytes/sec
c) 2 Mbytes/sec
d) 2 GBytes/sec

Memory cycle time
= time to read/write operation in memory in one cell.

500 nsec → 1 byte can be addressed

∴ Rate = $\frac{1}{500 \times 10^{-9}} = \frac{10^9}{500}$

$= \frac{2}{1000} \times 10^6 =$ 2 MBytes/s

Ques Consider two 16 bits unsigned values A and B. What will be the maximum size of result for

1. Addition of A and B — $16+1 = 17$ bits
2. Multiplication of A and B — $16+16 = 32$ bits

Size of addition table = $2^{32} \times 17 = 2^{32} \times 17$ bits

Size of multiplication table = $2^{32} \times 32 = 2^{32} \times 32$ bits

Ques A memory M is defined as number of words multiplied by the number of bits per word. The number of address and data lines needed for the memory $M = 2K \times 32$ bits are

No. of address lines needed = $\log_2(2K) = \log_2 2^{11} =$ 11 lines

No. of data lines needed = 32

Ques Consider a memory of size $2K \times 8$ bits. What is the size of decoder needed to access the cells of the memory unit?

No. of address lines = $\log_2 2K = 11$

∴ Size of decoder needed = 11×2^{11}

Ques

Statement 1: Static RAM memory devices retain data for as long as power is supplied. ✓

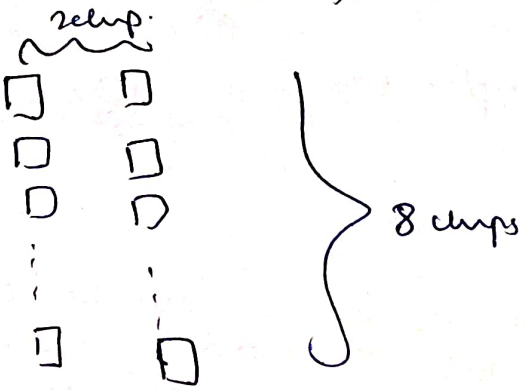
Statement 2: Static RAM is used when the size of read/write memory required is large. ✗

TOC DBMS OS
COA CD
DL CN

Ques How many 8K×8 bits RAM chips are needed to provide a memory capacity of 64K×16 bits? And how the chips are arranged?

No. of RAM chips req. = $\frac{64K \times 16}{8K \times 8} = 16$ RAM chips

8×2 hybrid



8 vertically
2 horizontally

Ques A DRAM chip of 512K×8 bits has 256 rows of cells with 2K cells in each row. If DRAM takes 20ns for 1 refresh cycle, total refresh time of the DRAM is _____ micro seconds.

Refresh time = time of refresh / 1 row = 20ns

Total Time to refresh DRAM = 20ns * 256

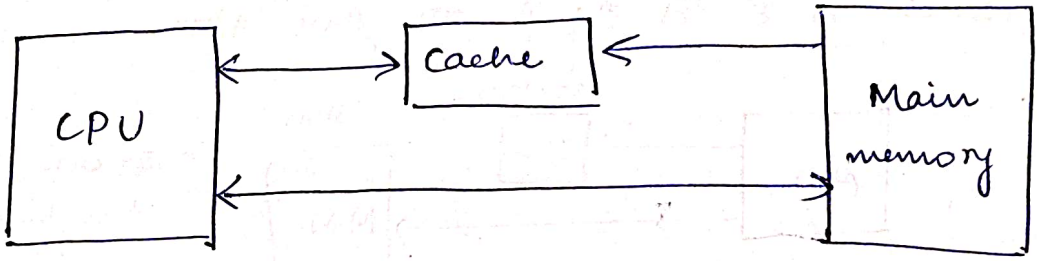
= 5120 ns = 5.12 μs

Locality of Reference

If CPU has requested one address for memory access, then that particular address or nearby addresses will be accessed soon.

Temporial locality of reference = CPU address same location again after sometime (based on time)

Spatial locality of reference = CPU addresses nearby location. (based on space)



Cache Hit

CPU's demanded content is present in cache

Cache Miss

CPU's demanded content is not present in cache.

Cache Hit Ratio (H) = No. of hits / Total memory references.

Miss Ratio = 1 − H

In case of miss, CPU goes to main memory accesses the required content and then brings the block (which contains demanded content) from main memory to cache memory.

Average Memory Access Time = H * Memory access time for each hit + (1 − H) Memory access time for each miss.

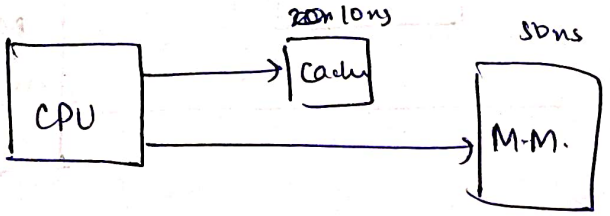
Types of Cache Accesses

1. Simultaneous access.

Signal is sent to both cache and main memory at the same time requesting for an address.

अगर data cache में मिल गया तो उसे use कर लिया जाएगा और memory में search करना बन्द कर दिया जाएगा

अगर cache miss होता है तो memory में पहले से ही search हो ही रहा है। तो कम time लगेगा



In case of miss, time taken = 50ns

$$T_{avg} = H * t_{CM} + (1-H)\, t_{M.M}$$

2. Hierarchial access

Memory में search तब ही start होगा जब cache miss हो जाए.

$$T_{avg} = H * t_{CM} + (1-H)(t_{CM} + t_{MM})$$

$$= H * t_{CM} + t_{CM} + t_{MM} - H t_{CM} - H t_{MM}$$

$$T_{avg} = t_{CM} + (1-H)\, t_{MM}.$$

T_{avg} = Avg memory access time
H = cache hit ratio.
t_{CM} = cache memory access time
t_{MM} = main memory access time

If t_{CM}, t_{MM} are not given explicitly,

T_{avg} = H * memory access time in case of hit + (1−H) * memory access time if miss

else

if hierarchial/level/serial word is mentioned in question

$T_{avg} = t_{CM} + (1-H)\, t_{MM}$

if simultaneous/parallel word is mentioned in question

$T_{avg} = H * t_{CM} + (1-H)\, t_{MM}$

Cache look up time is zero

default if not given

T_{avg} when Locality of Reference is not included.

Simultaneous → $H * t_{CM} + (1-H)\, t_{Block}$

Hierarchial → $t_{CM} + (1-H)(t_{Block} + t_{CM})$

t_{Block} = time required to access block = Block size * t_{MM}.

T_{avg} when Block Transfer is included

Simultaneous = $H * t_{CM} + (1-H)(t_{Block} + t_{CM})$

Hierarchial = $t_{CM} + (1-H)(t_{Block} + t_{CM} + t_{CM})$

Note:- In one access of cache, entire block can be accessed

↓

cache memory is designed in such a way.

Ques Assume that for a certain processor, a read request takes 50 ns on a cache miss and 10 ns on cache hit. Suppose while running a program, it is observed that 90% of the processor's read requests result in a cache hit. The avg. read access time in nanoseconds is _____?

$$T_{avg.} = H * T_{cache\ hit} + (1-H) * T_{cache\ miss}$$

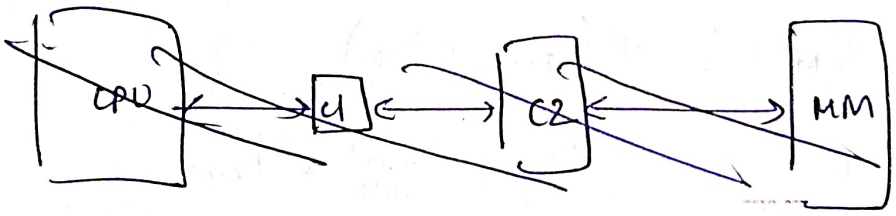
$$= 0.9 \times 10 + 0.1 \times 50$$

$$= 9 + 5 = \underline{14\ ns}$$

Ques In a 2 level hierarchy, if the top level has an access time of 30 ns and the bottom level has an access time of 150 ns, what is the hit rate on the top level required to give an average access time of 45 ns.

~~$T_{avg} = H * t_{CM} + (1-H) t_{MM}$~~.

Top level memory = closest to CPU



For ~~C2~~,

~~$T_{avg} = t_{C_2M} + (1-H) t_{MM}$~~

$\Rightarrow 150 =$

$t_{MM} = 150\ ns$

$t_{CM} = 30\ ns$

$T_{avg} = H * t_{CM} + (1-H) t_{MM}$

$\Rightarrow 45 = 30 + (1-H)\,150$

$\Rightarrow \frac{1}{10} = 1-H \Rightarrow H = \frac{9}{10} = \underline{0.9}$

Cache Write

Updating value in cache.

Cache Write problem :- If CPU updates the content in cache then the original content in main memory also should be updated.

Also known as write propagation

Write Through

Parallel update of content in both cache memory as well as main memory is done by the CPU.

Simultaneous write operation.

Adv —
↳ Data consistency [cache & m.m. will always have same value of a content]

Disadv —
↳ Time consuming [For write operation, mm is accessed regardless of hit or miss]

	Simultaneous	Hierarchial
$t_{Avg\ Read}$ =	$H * t_{CM} + (1-H)\, t_{MM}$	$H * t_{CM} + (1-H)(t_{CM} + t_{MM})$
$t_{Avg\ Write}$ =	$\max(t_{CM}, t_{MM}) = t_{MM}$	

$\therefore t_{Avg}$ = fraction of read * $t_{Avg\ Read}$ + fraction of write * $t_{Avg\ Write}$.

Ques A system has write through cache with access time 100ns and hit ratio of 90%. The main memory access time is 500ns. The 70% of memory oper references are read.

1. $t_{Avg.\ Read} = H * t_{CM} + (1-H) t_{MM} = 0.9 \times 100 + 0.1 \times 500 = 90 + 50 = \underline{140\ ns}$
2. $t_{Avg.\ Write} = \max(t_{CM}, t_{MM}) = 500ns$
3. $t_{Avg} = 0.7 \times 140 + 0.3 \times 500 = 98 + 150 = \underline{248\ ns}$
4. Effective hit ratio
   Let H_e be effective hit ratio
   $H * 100 + (1-H) * 500 = 248$
   $\Rightarrow 100H + 500 - 500H = 248 \Rightarrow 400H = 252 \Rightarrow H = \frac{252}{400} = \underline{63\%}$

CPU updates only cache. Whenever a block is replaced from cache, it is updated in memory.

Write Back

When a block is to be removed from cache memory, if updation is done in the block, it is transferred to main memory.

Adv
- time saving

Disadv
- Data inconsistency

$$t_{Avg\ Read\ or\ write} = H * t_{CM} + (1-H)(t_{MM} + t_{write\ back})$$

[simultaneous]

$$H * t_{CM} + (1-H)(t_{MM} + t_{CM} + t_{write\ back})$$

[Hierarchial]

Write Allocate & No Write Allocate

For read operation
- for a cache miss, a missed block is brought from main memory to cache memory.

For write operation
- Write allocate
  - For a write miss, missed block should be copied to cache memory from main memory.
- No write allocate
  - For write miss, updation is done in memory directly without bringing it to cache.

For better performance,

Write Through ⇒ used no allocate write allocate.

Write back ⇒ uses write allocate

Cache Mapping

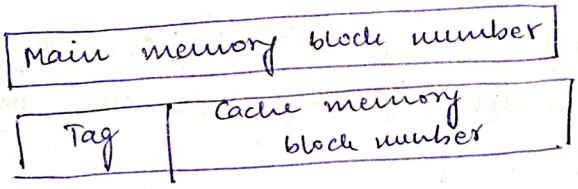
- CPU generates main memory address.
- Conversion of main memory address to the main memory address is called cache mapping.

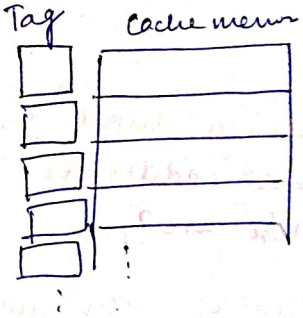
- Types —
  - → Direct Mapping
  - → Set Associative mapping
  - → Fully Associative mapping.

Direct Mapping —

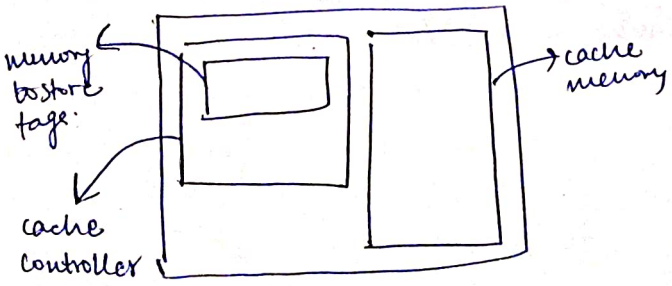
(a) Cache memory block number = Main memory block number % No. of blocks in cache memory.

(b) Tag identifies which block of main memory (out of all possibles) is present in cache at a given time.





(c) No. of tag fields required = No. of blocks in cache (line)



Main memory address = | Block number | Block offset |

Cache memory addrss = | Tag | Block number | Block offset |

Block offset is not used to tell if it is a hit or not.

deciding whether hit/miss

Ques

Block size = 16 bytes
Size of Cache memory = 64KB
Size of main memory addrss = 32 bits
Direct mapping

1) Bits in byte offset = $\log_2$[Block size] = $\log_2 16$ = 4

2) Bits in cache block number $\log_2\left(\frac{64\,KB}{16\,B}\right)$ = $\log_2 4K$ = 12 bits

3) Bits in tag = 32 − (4+12) = 16 bits

4) Tag Directory size = 16 × 4K = $2^4 \times 2^{12} = 2^{16}$ = 64K bits
/Size of metadata
= 8KB (kilobytes)

Ques

Consider a direct mapped cache of size 128KB. The CPU generates 32 bits addresses. The number of tag bits in main memory address are?

Size of cache memory = 128KB

No. of tags = $\frac{2^{32}}{128\,KB} = \frac{2^{32}}{2^7 \times 2^{10}} = \frac{2^{32}}{2^{17}} = 2^{15}$

No. of tag bits = 15.

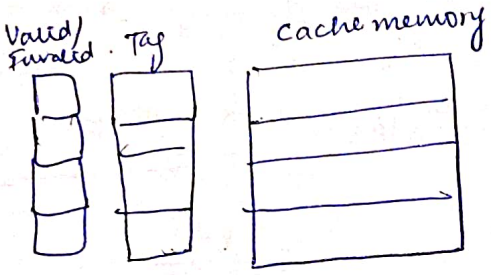
No. of tag bits = 32 − $\log_2$(Cache memory size)

Index bit = No. of bits in cache block number

Cache Initialization

When computer restarts, garbage value is present in cache.
Possible that CPU generated tag bit matches the ~~tag~~ garbage tag of block.
So, garbage value is accessed.

To solve this problem, an extra bit called valid / invalid bit is present with each block.



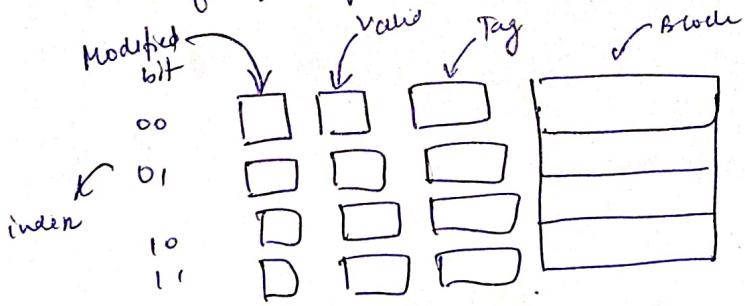
0 ⟶ invalid
1 ⟶ valid

When CPU starts, all the valid / invalid bits are set to 0.

Write Back Cache Performance.

Performance can be improved if we can keep track of each cache block which is updated or not.

So, one more bit is present for each block in cache called Modified / Dirty bit.



Dirty bit
→ 0 → only read
→ 1 → write

Tag Directory size = No. of blocks in cache mem * [Tag bits + Extra bits]

Ques A 16kB direct mapped write back cache is organized as multiple blocks, each of size 32 bytes. The processor generates 30 bit address. The cache controller maintains the tag information for each cache block comprising of the following —

1) Valid bit
2) Modified bit.

As many as minimum needed to identify the memory block mapped in the cache.

What is the total size of memory needed at the cache controller to store meta tags for the cache.

No. of blocks in cache $= \frac{16kB}{32B} = \frac{2^4 \times 2^{10}}{2^5} = 2^9$

Tag $= \log_2 \left[\frac{16kB}{2^{14}}\right] = \log_2 1 = 0$

Size of metadata $= 2^9 [16+1+1] = 2^9 \times 18 = 2^9 \times 2^3 = 2^{12}$ bit

$= 2^9 \times 2 \times 9$

$= 9$ kbits

$= 1$ KB

No. of blocks in main memory $= \frac{2^{30}}{2^5} = 2^{25}$

$\therefore$ Tag bits $= \log_2 \left\lceil \frac{2^{25}}{2^9} \right\rceil = \log_2 16$

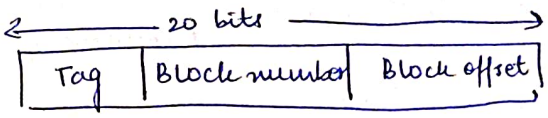
$= 16$ bits

Tag for write back cache —

→ Simultaneous

Consider a machine with a byte addressable main memory of 2^{20} bytes, block size of 16 bytes and a direct mapped cache having 2^{12} cache lines.
Let the addresses of 2 consecutive bytes in memory be $(E201F)_{16}$ and $(E2020)_{16}$. What are the cache and tag line address (in hex) for main memory address $(E201F)_{16}$

Block size = 16 bytes
Line size = 16 bytes
Block offset = 4 bits
No. of cache lines = 2^{12}
Main memory address =



E201F = 1110 0010 0000 0001 1111
Tag bits
Cache line

$\frac{2^{20}}{2^{12}} = 2^8$

E201F = 1110 0010 0000 0001 1111
Tag | Line number | offset
(E) (201) (F)

(A) E, 201
(B) F, 201
(C) E, E20
(D) 2, 01F

Ques Consider a 32kB cache with 32 bytes of blocks. The main memory is of 16MB. The main memory address is 0x67ABIE has tag, block and offset values as.

Size of cache = 32kB = $2^5 \times 2^{10}$ bytes = 2^{15} bytes
Size of 1 block = 32 bytes
∴ No. of blocks = $\frac{2^{15}}{2^5} = 2^{10}$ blocks

No. of bits in block offset = $\log_2 32 = 5$

$2^4 \times 2^{20}$
$\frac{2^{24}}{2^{15}} = 2^9$

No. of tag bits = 32 − (10+5) = 17

0x67ABIE = 0110 0111 1010 1011 0001 1110
Tag | block no. | offset
$(0CF)_{16}$ $(158)_{16}$ $(1E)_{16}$

Prob'em with Direct Mapping

Blocks in main memory have fixed position in cache Freq. access of blocks occupying same b'ine.

Set Associative mapping

① Multiple blocks at an index.

② n-way set associative mapping ⇒ n blocks at one index.

Associativity = n

No. of indices/ sets = $\frac{\text{No. of blocks in cache}}{\text{Associativity}}$

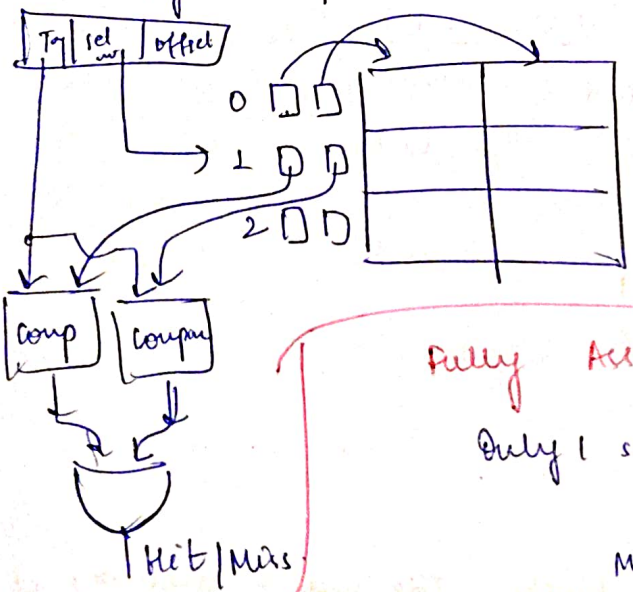
Cache memory set no. = Main memory block % No. of sets.

Main memory address.

Tag	C.M. set number	Byte offset

Tag Directory size = No. of blocks in cache * [Extra bits + Tag]

Tag bit is present with each block of a particular set.



Tag bits are compared with tags of each block in a set.

If matches, it is a hit, if not, then miss

Fully Associative set mapping

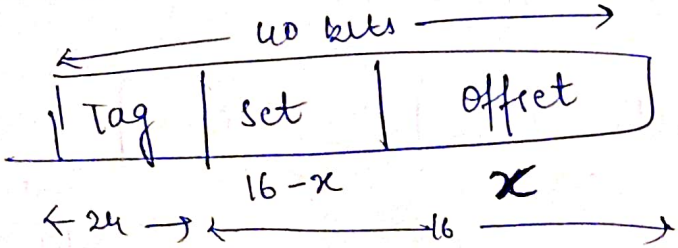
Only 1 set [similar to set associative mapping]

Main mem address

Tag	Block offset

Gate 2016

The width of the physical addrss on a machine is 40 bits. The width of the tag field in a 512 kB 8 way set associative cache is _____ bits



No. of bits in tag field = 24

let block size = 2^x

No of blocks = $\frac{2^{19}}{2^x} = 2^{19-x}$

No. of sets = $\frac{2^{19-x}}{8} = 2^{16-x}$

Hardware implementation of Direct mapping –

No. of comparators required = 1
Size of comparator = No. of tag bits.

No. of multiplexers required = No. of tag bits
Size of multiplexers = No. of blocks X 1.

Hit Latency Time
= 1 MUX delay for tag selection + 1 comparator delay for tag comparison

Hardware implementation of k-way set associative mapping

No. of comparators required = k
Size of comparator = No. of tag bits

No. of multiplexers req. = k * No. of tag bits
Size of multiplexers = No. of sets in cache * 1

k-input OR gate = 1

Hit Latency time
= 1 MUX delay for tag selection + 1 comparator delay + OR gate delay

Hardware implementation of fully associative set mapping

No. of comparators required = No. of blocks in cache memory.
Size of comparator = No. of tag bits.

No. of multiplexers = 0

1 OR gate — n input OR gate.

Hit Latency time
= 1 comparator delay + OR delay

Cache Miss Penalty

Time Required to bring a missed block from main memory to cache.

gate 2019
A certain processor deploys a single level cache. The cache block size is 8 words and word size is 4 bytes. The memory system uses a 60MHz clock. To service cache miss, the memory controller first takes 1 cycle to accept the starting address of the block, 3 cycles to fetch the requested block at the rate of 1 word/cycle and finally transmits the words of the for the memory system when the program running on the processor issues a series of read operations is _____ $\times 10^6$ bytes/sec.

Block size = 8X4 = 32 bytes
Clock rate = 60MHz
Time to transmit 1 word = 1 cycle
" " " 8 words = 8 cycles
Total cycle in case of miss = 1+8+3 = 12 cycles

~~32 bytes~~
To transfer 32 bytes, 12 cycles are req.
~~To transfer 1 byte~~
Bandwidth = $\frac{32}{12}$ bytes/cycle
$= \frac{32}{12} \times 60 \times 10^6$ bytes/s
$= 160 \times 10^6$ bytes/s

Types of cache Miss

1) Cold/Compulsory miss

First time access of a block will always cause miss.

Solution :- increase ~~no. of block~~ the block size of cache/memory.

2) Capacity miss

If it is not cold miss and miss occurs when the cache is full.

Solution to reduce – increase the size of cache.

3) Conflict miss

If its not a cold & capacity miss but due to the set is full and miss occurs.

Solution to reduce → Increase associativity.

Example:–

- No of blocks in cache = 4
- 2 way set associative cache
- Cache is initially empty
- LRU Replacement policy
- CPU Requests for main memory

} No of sets in cache = 2

blocks – 0, 4, 0, 8, 0, 4, 1, 3, 1, 5, 1, 3

0 → cold miss, 4 → cold, 0 → hit, 8 → cold, 0 → ~~capa~~ conflict, 4 → conflict, 1 → cold, 3 → cold, 1 → hit, 5 → cold, 1 → capacity, 3 → capacity

(conflict ← hit)

Set		
0	~~0~~ ~~8~~ 4	~~4~~ 0
1	~~1~~ ~~5~~ 3	~~3~~ 1

Concept ~~समझो~~ Answer तो खुद ही होगा !!

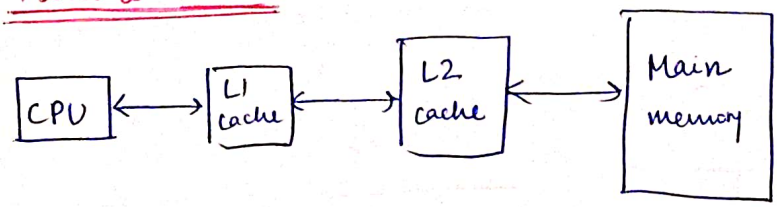
Goals of using cache memory

1. Minimize Access Time ⟶ use small cache
2. Maximize Hit Rate ⟶ use large cache

} multilevel cache

3. Minimize Miss Penalty.

Multilevel cache



Access time	t_1	t_2	t_{MM}.
Hit ratio	H_1	H_2.	

Simultaneous access –

$$T_{avg} = H_1 * t_1 + (1-H_1)\left[H_2 * t_2 + (1-H_2)\, t_{MM}\right]$$

Hierarchical access –

$$T_{avg} = H_1 * t_1 + (1-H_1)\left[H_2 * (t_1 + t_2) + (1-H_2)\left[t_1 + t_2 + t_{MM}\right]\right.$$

OR

$$T_{avg} = t_1 + (1-H_1)\left[t_2 + (1-H_2)\, t_{MM}\right]$$

Policies

- Inclusion: Blocks in L1 are also present in L2 [Block is copied from L2 to L1]
- Exclusion: Blocks in L1 need not be present in L2 [Block is moved from L2 to L1]
  - ↓
  - L2 is filled with only replaced (victim blocks) of L1
  - ∴ it is known as victim cache.

Ques

Consider a memory hierarchy with a write back cache. The cache has an access time of 25ns. 90% of all memory accesses are found in the cache itself. The main memory access time is 300ns. The 20% of the cache blocks are dirty. The cache block size is 4 bytes. Calculate the average memory access time

$t_{CM} = 25ns$ $\quad t_{MM} = 300ns$ $\quad t_{block} = 300*4 = 1200ns$

$t_{write\_back} =$ % of dirty blocks $* t_{block} = 0.2 * 1200 = 240ns$

$$T_{avg} = H * t_{CM} + (1-H)\left[t_{CM} + t_{MM_{block}} + t_{write\ back}\right]$$

$$= 0.9 * 25 + (1-0.9)(25 + 1200 + 240)$$

$$= 25 + 0.1(1200 + 240)$$

$$= 25 + 0.1 * 1440 = 25 + 144 = \underline{169\ ns}$$

Magnetic Disk

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

- **Seek Time**: Time required to position read/write head on a particular track (will be mentioned directly in ques)
- **Rotational Latency**: Time required to position read write head on a particular sector in track. $RL = \frac{1 \text{ rotation time}}{2}$
- **Transfer Time**: Time required to transfer data from 1 sector. $TT = \frac{1 \text{ rotation time}}{\text{no. of sectors per track}}$

Data transfer करो या ना करो, rotation speed of disk same ही रहती

* At any given time, only one read-write head can work

Ques Consider a disk with 16 platters, 2 surfaces per platter, 1K tracks per surface, 1K sectors per track and 2048 bytes per sector. Disk rotates with 3000 rpm. Seek time is 10ms.

1) Capacity of disk
2) No. of bits req. for addressing
3) Disk access time
4) Disk Transfer rate.

3000 rotations — 60 sec

1 rotation — $\frac{60}{3000}$ sec = $\frac{1}{50}$ sec = $\frac{1000}{50}$ ms = 20ms.

Rotational latency = $\frac{20}{2}$ = 10 ms

Capacity of disk = $16 \times 2 \times 2^{10} \times 2^{10} \times 2048 = 2^4 \times 2 \times 2^{20} \times 2^{11} = 2^{36}$

No. of bits in addr = 2^{36} No. of sectors = $16 \times 2 \times 1K \times 1K = 2^{25}$ ∴ 25 bits

Disk access time = $10\text{ms} + 10\text{ms} + \frac{20\text{ms}}{2^{10}} = 20 + 0.02$

= 20.02 ms

~~Data transfer rate = $\frac{20}{10}$ bytes/sec.~~ 0.02 ms → 1 sector data = 2048 bytes = 2KB

∴ 1s → $\frac{2048 \times 1000}{0.02}$ bytes = $2^{10} \times$ 100 MBPS.

Addressing is done
- track wise
- cylinder wise.

<cylinder no., surface no., sector no.>

No. of sectors per cylinder = No. of sectors in 1 track * No. of surface.

Address <c, h, s>

No. of sectors per track = n_t

No. of sectors per cylinder = n_c

$$\text{Sector no.} = c * n_c + h * n_t + s$$

$$c = \lfloor \text{sector no.} / n_c \rfloor$$

$$h = \lfloor (\text{sector no.} \,\%\, n_c) / n_t \rfloor$$

$$s = (\text{sector no.} \,\%\, n_c) \,\%\, n_t.$$

Ques Consider a disk with 2K sectors per track. Disk rotates with 6000 rpm. Seek time is 20ms. Calculate disk access time.

Seek time = 20ms

6000 rotations — 60s

1 rotation — $\frac{60}{6000}$ s = 0.01s = 10ms

Rotational latency = $\frac{10}{2}$ = 5ms.

Data transfer rate = $\frac{10}{200}$ = 0.5 × 0.01 ms = 0.005 ms.

Disk access time = 20ms + 5ms + 0.005ms = 25.005ms

Constant linear Velocity	Variable sector capacity
Constant angular velocity	Constant sector capacity

* Ques

Let $\frac{60}{6000} = \frac{1}{x}$

Flynn's classification of Computer

Pipeline system
- → SISD — Single Instruction stream Single Data stream.
- → SIMD — Single Instruction stream Multiple Data stream
- → MISD — Multiple Instruction stream Single Data stream (theoretical)

Multiple pipeline
- → MIMD — Multiple Instruction stream Multiple Data Stream.

PIPELINING

@ Pipelining is useful when same processing is applied over multiple inputs.

@ Pipelining is a technique of decomposing a sequential process into suboperations.
Suboperations are performed in segments.
All segments perform their respective suboperations parallely on different inputs

Pipeline Cycle Time

Minimum time in which all segments can perform their respective suboperation.

$$\text{Max}[t_{seg1}, t_{seg2}, t_{seg3} \ldots]$$

General Consideration

Consider a k segment pipeline.
With clock cycle time = t_p
to perform n tasks.
Time required to perform 1st task = $k * t_p$
Time required to perform remaining n-1 tasks = $(n-1)\,t_p$
∴ Total time required for all n tasks = $(k+n-1)\,t_p$

Consider a non pipeline system that takes t_n time to perform a task
Time required to perform n tasks = $n * t_n$.

Performance of pipeline = Speedup (S) = $\frac{\text{Non pipeline time}}{\text{pipeline time}}$

$$S = \frac{n t_n}{(k+n-1)t_p}$$

When $n >>> k$,

$$S = \frac{n t_p}{n t_p} : \frac{t_n}{t_p}$$

max / ideal speedup

value of n is not given to

When time required to perform 1 task is equal in pipeline and non pipeline

$t_n = k t_p \Rightarrow S = \frac{t_n}{t_p} = k$

Speed up is always less than k

Pipeline Hazard.

Situations that prevent the next instruction from being executed dur its designated clock cycle.

- Structural Hazard / Resource Conflict
- Data Hazard / Data Dependency
- Control Hazard / Branch Difficulty

Structural Hazard

2 different segments try to use same resource at a same time.

Solution — 1) Incur stall cycles
2) Increase number of resources.

Data Hazard / Dependency

Result of a segment is input to the next segment.

i: R1 ← R2 + R3 } Dependency.
it: R4 ← R1 + R5

IF	DA	OF	EX	WB
	IF	DA	~~OF~~	

↓ Operand fetch cannot occur bcoz result is not available should be after WB.

A general instruction pipeline cannot detect data dependency.

The solution of data dependency is provided by compiler.

R1 ← R2 + R3
=
R5 ← R1 + R4

} independent (no operation instructions

Delayed Load.

— Software Solution.

① Hardware solutions are used when pipeline can detect and solve data dependency.

→ **Hardware interlock.**

stalls in case of data dependency.

i: IF DA OF EX WB

IF DA — — OF EX WB

stalls (hardware is locked).

→ **Operand Forwarding**

The result generated by ith instruction is transfered to EX phase of (i+1)th instruction from the accumulator.

IF DA OF EX WB

IF DA OF EX WB

R1

R1 ← R2+R3

P5 ← R1 +R4.

↓ wrong value of R1 is fetched R4 is also fetched.

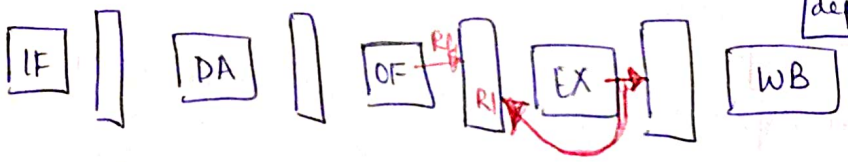
↓ value of R1 from accumulator is used.

no stalls

Operand forwarding can be used in this example because ALU operation in both the instructions.

No stalls for ALU to ALU depndy.

But for memory to ALU or ALU to memory dependency, stalls required.



ith → R1 ← M[Add]

(i+1) → R2 ← R1+R5

} operand forwarding cannot be used.

CONTROL HAZARD / BRANCH DEPENDENCY

For branch instruction, the CPU has to wait till branch outcome is available.

A general pipeline cannot detect branch dependency

⇓

Compiler provides solution — Delayed branch.

inserts 3-4 no operation instructions below branch instruction.

(after execute phase, target instruction is found out.)

Branch IF DA OF EX WB

IF DA OF EX WB

IF DA OF EX WB

IF DA OF EX WB

→ Target ins. IF DA OF EX WB.

Hardware solution:

→ **Branch Prediction**

Predicts whether the current branch instruction will take jump or not.

If prediction is correct — no problem.

If prediction is not correct — Roll back to ~~correct~~ branch instruction & execute target.

Classification of Data Hazard / Dependency

- True dependency
  - → RAW (Read After Write) — i: R1 ← R2+R3; j: R5 ← R1+R4 — [j reads before written by i. Hence j incorrectly gets old value]
  - → WAW (Write After Write) — i: R1 ← R2+R3; j: R1 ← R4*R5 — [j writes in destination before i writes it]
- Anti Dependency
  - → WAR (Write After Read) — i: R1 ← R2+R3; j: R2 ← R9+R7 — [if j modifies R2 before i uses its value, i incorrectly reads new value]
- false dependency.

Solution :- Register Renaming

i: R1 ← R2+R3
j: R6 ← R9+R7
(WAR)

i: R6 ← R2+R3
j: R4 ← R4*R5
(WAW)

CPI (Cycles Per Instructions)

No. of cycles needed to execute 1 instruction.

For pipelined system,

$$CPI = \frac{k+n-1}{n}$$

In ideal case,

$$CPI = 1$$

If because of hazards, total extra (stall) cycles = x

$$CPI = \frac{k+n-1+x}{n}$$

In ideal case,

$$CPI = \frac{n+x}{n} = 1 + \frac{x}{n}.$$

With hazard,

one instruction execution cycle = CPI$_{avg}$ * tp.

Pipelining Hazards –

① Structural Hazards [diff. segments same resource required] → incur stalls
→ increase no. of ~~stalls~~ resource.

② Data Hazards → RAW (Read after write) } True dependency
→ WAW
→ WAR → anti dependency. } false dependency

operand forwarding.
stalls.

③ Control Hazard → predict branch → static
→ dynamic (branch history table)
→ stalls

Amdahl's law

used to calculate speedup in multiprocessor system.

$$\text{Speedup} = \frac{1}{\text{Sequential part} + \frac{\text{Parallel Part}}{\text{Number of processors}}}$$

(ratio) (ratio)

Program Status word

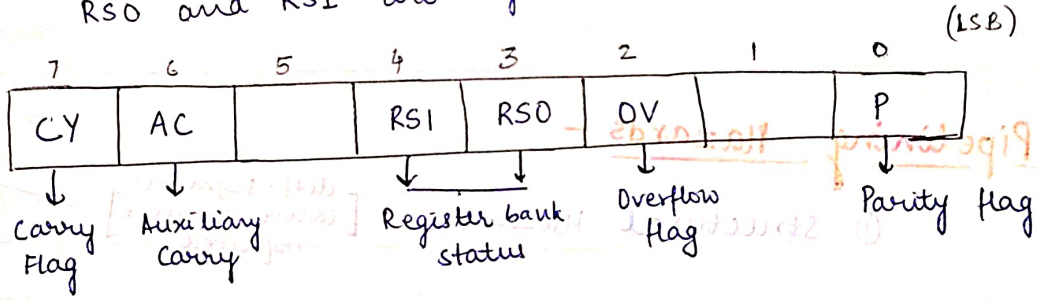
8 bit Register

Program Status Word (PSW) is a collection of data that operating system maintains to keep track of system's current state.

Four flags are called conditional flags —

- CY (Carry)
- AC (Auxiliary Carry)
- Parity (P)
- OV (Overflow)

RS0 and RS1 are register selection bit



In 16 bit PSW —

- 7 bits are unused
- 9 bits are used —
  - 3 control flags
  - 6 status flags (includes conditional fl

Control Flags —

- DF (Direction Flag) — 10th bit
- IF (Interrupt Flag) — 9th bit
- TF (Trap Flag) — 8th bit

Overflow — 11th bit

12-15, 1, 3, 5 - unused

CF → 0th

Parity → 2nd

AF → 4th

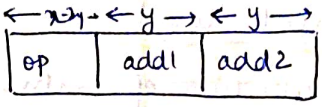
C — zero flag

7 — sign flag.

System supporting variable length opcode

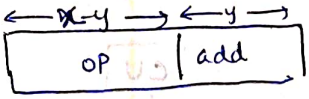
* length instruction

2 address instruction :



m 2 address instructions supported

1 address instructions :



n 1 address instructions supported

$$\text{Max. } n = \left(2^{x-2y} - m\right) \cdot 2^{(x-y)-(x-2y)}$$

Min n = 1

$$= \left(2^{x-2y} - m\right) \cdot 2^{y}$$

- $2^{x-2y} - m$: No. of unused combinations
- 2^{y} : opcode of 1 address ins − opcode length of 2 address instructions

IEEE-754 Floating Point Representation

Bias = 127 → Single Precision — 32 bits
- 1 sign bit (S)
- 8 exponent (E)
- 23 mantissa (M)

$$(-1)^S * 1.M * 2^{E-127}$$

Bias = 1023 → Double Precision — 64 bits
- 1 sign bit (S)
- 11 bit exponent (E)
- 52 bit Mantissa (M)

$$(-1)^S * 1.M * 2^{E-1023}$$

* For denormalized no., E = −126 always

S	E	M	Number
0	000....0	00...0	+0
1	000....0	00....0	−0
0	111....1	00....0	+∞
1	111....1	00....0	−∞
0/1	111....0	M ≠ 0	Not a number
0/1	000....0	M ≠ 0	Denormalized no. 2^{-126}
0/1	E ≠ 00...0 E ≠ 11...1	* can be anything	Implicit Normalization (Normal no.)

Control unit

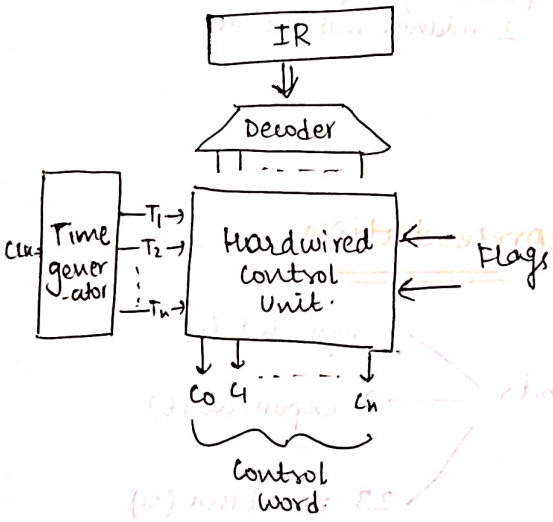
generates control word

↓

collection of control signals generated by CU at once

CU

- Hardwired Control Unit
  - ⊙ faster
  - ⊙ difficult to rearrange / change control logic
- Microprogrammed Control unit
  - ⊙ slower
  - ⊙ updation of control logic is easy.
    - Vertical Microprogramming
      - ⊙ signals divided into groups such that at a time, only 1 signal can be active
      - ⊙ Smaller control word
      - ⊙ Slower
      - ⊙ Decoders used
    - Horizontal Microprogramming
      - ⊙ one bit for every control signal.
      - ⊙ Larger control word
      - ⊙ Faster
      - ⊙ Decoders not used.



On the basis of operating speed –

Vertical Microprogrammed CU < Horizonal Microprogrammed CU < Hardwired CU

↳ slowest

↳ fastest.

RISC	CISC
⊙ Less no. of instructions	⊙ More no of instructions
⊙ Fixed length ins (variable opcode)	⊙ variable length ins (fixed opcode)
⊙ Less addressing modes	⊙ More addressing modes
⊙ Hardwired CU	⊙ μprogrammed CU
⊙ More no of registers	⊙ less no. of registers
⊙ only R-R operation	⊙ Reg. to Memory operation also prent.

↓ Preferable for pipelined CPU

Big endian -

little endian -

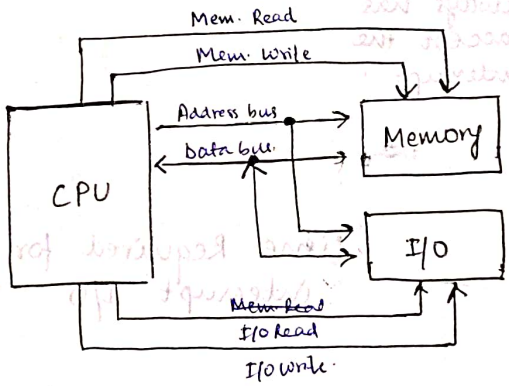
IO Mapped IO / Isolated IO

common data bus and address bus.

different control bus

CPU distinguishes b/w mem. & I/O using control signal

↳ no wastage of memory.

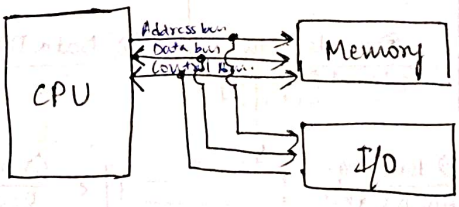


Memory Mapped IO

common address bus, data bus, control bus.

Some addresses in memory are reserved for I/O.

↳ memory wastage



<Data bus is 2 sided
Address bus is 1 sided>

Modes of IO Transfer

1. Programmed I/O

Total Time = Time to read Status of I/O device + Time to check if status flag is set or not + Data transfer time.

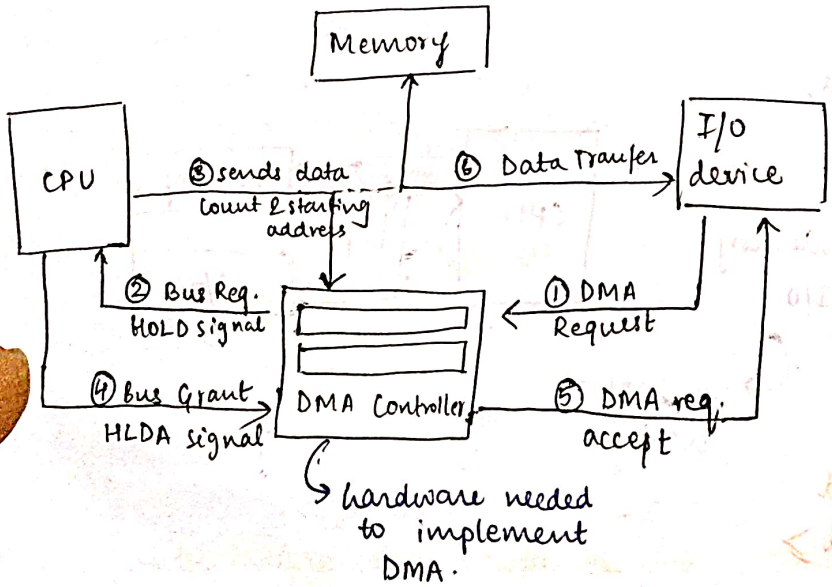
2. Interrupt I/O

Vector address = address / location of Interrupt Service Routine.

- Interrupt
  - Maskable
    - Interrupt can be accepted / rejected
  - Non Maskable
    - CPU always has to accept the interrupt
  - Vectored (Device sends interrupt and vector address) (Control bus → data bus after interrupt is acknowledged)
  - Non vectored interrupt (scaler) (Device sends only interrupt) (Default service routine is run by CPU to obtain vector address).

Time Required for interrupt I/O = Interrupt overhead time + Service time.

3. Direct Memory Access



Data count – No. of bytes/words to be transferred

Starting address – memory address starting from where data is to be transferred.

* Time is saved in this method as CPU does not have to store the present state of the system in stack.

Time taken to prepare the data in I/O = t_x → will be given in question

Time taken to transfer data to memory = t_y → depends upon clock freq./ cycle time

Imp.

GATE 2022

% of Time CPU is blocked —

1. Burst mode — $\frac{t_y}{t_x + t_y}$

2. Cycle stealing mode = $\frac{t_y}{t_x}$

3. Interleaving mode = 0.

Size of ROM required to store table —

n bit A × n bit B.

Size of multiplication table = $2^{2n} * 2n$

Size of addition table = $2^{2n} * (n+1)$.

RAM (Random Access Memory)

- Static
  - → flip flops
  - → no refresh
  - → faster
  - → expensive
  - * → high power consumption.
- Dynamic
  - ↦ data is stored in form of electric charge.
  - ↦ periodic refresh.
  - ↦ slower
  - ↦ less expensive.
  - * ↦ low power consumption.

⊙ In one refresh operation, entire row gets refreshed.

⊙ Time to refresh n chips = Time to refresh 1 chip.

Refresh period = Read / write time + Total refresh time.

⊙ Associative memory is used to implement TLB (Translation Look Aside Buffer)

↳ stores frequently used pages.

TLB Hit —

Page is retreived from TLB

Time = T_{TLB}.

TLB Miss —

Page is retreived from page table & then memory.

Time = $T_{TLB} + T_{MM} + T_{MM} = T_{TLB} + 2T_{MM}$

↳ coz page table is in main memory.

$\therefore T_{avg} = H * T_{TLB} + (1-H)(T_{TLB} + 2T_{MM})$

Cache write

↳ Write Through (simultaneous updation in M.M)

↳ Write Back. (cache block transferred to M.M. when replaced)

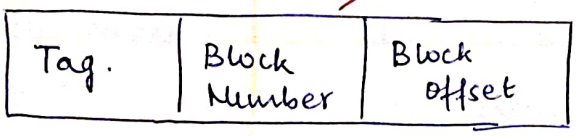
Write allocate — write miss → missed block should be copied ~~from~~ to ~~M.M to~~ cache memory ~~from~~ M.M.

No write allocate — write miss → updation is done in M.M. directly.

Cache Mapping

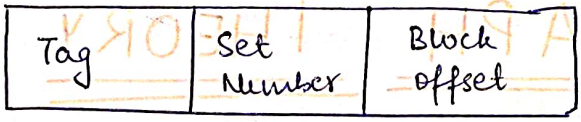
→ Direct Mapping.

decide if it is a cache hit or cache miss.



Cache memory block number = Main memory block number % No. of ~~blocks~~ lines in cache memory

→ Set Associative Mapping.

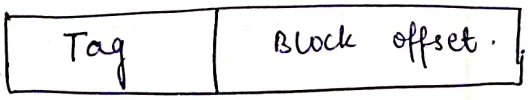


Cache memory set no. = Main memory block number % No. of sets in cache memory.

→ Fully Associative mapping

only 1 set.

similar to associative memory.



Size of metadata/ Tag directory size = [No. of Tag bits + Extra bits] * No. of lines in cache memory.

Types of cache miss.

1) Cold / Compulsory miss
   Soln → increase block size of cache memory.

2) Con Capacity miss
   Soln → increase size of cache.

3) Conflict miss
   soln → increase associativity of cache.