Addressing Modes:

It specifies a rule for interpreting or modifying the address field of the instruction.

Variety of addressing modes.

→ to give programming flexibility to the user.
→ to use the bits in the address field of the instruction efficiently.

Types of Adressing Modes:

1. Implied Mode:

Address of the operands are specified implicitly in the instruction. No need to specify in the instruction.

Ex: CMA (Complement the accumulator)

2. Immediate Mode:

Instead of specifying the address of the operand, the instruction contain the operand itself.

3) Direct Address Mode:

Instruction specifies the memory address which can be used directly to get the operand.

4. Indirect Addressing mode:

The address field of an

instruction specifies the address of a memory location that contains the address of the operand.

5 Register Mode:

Address specified in the instruction in the register address.

6. Register Indirect Mode:

Instruction specifies a register which contains the memory of the operand.

Auto - Increment:

Same as the register indirect but when the address in the register is used to access memory, the value in the register is incremented after the execution of the instruction.

Auto - Decrement:

Same as the register indirect but when the address in the register is used to access memory, the value in the register is decremented before the execution of instruction.

7. Relative Addressing Modes:

The Address field of an instruction specifies the part of the address which can be used along with a PC to calculate the address of operand.

EA = PC + IR (address)

8. Indexed Addressing Mode:

XR: Indexed Register

EA = XR + IR (address).

EA: Effective address

IR (address): Indexed Address

9. Base Register Addressing Mode.

BAR: Base Address Register

EA = BAR + IR (address).

Ques: 1

Address	Memory
200	load to AC
201	Address = 500
202	Next Instruction
399	450
400	700
500	800
600	900
702	325
800	300

PC = 200

R1 = 400

XR = 100

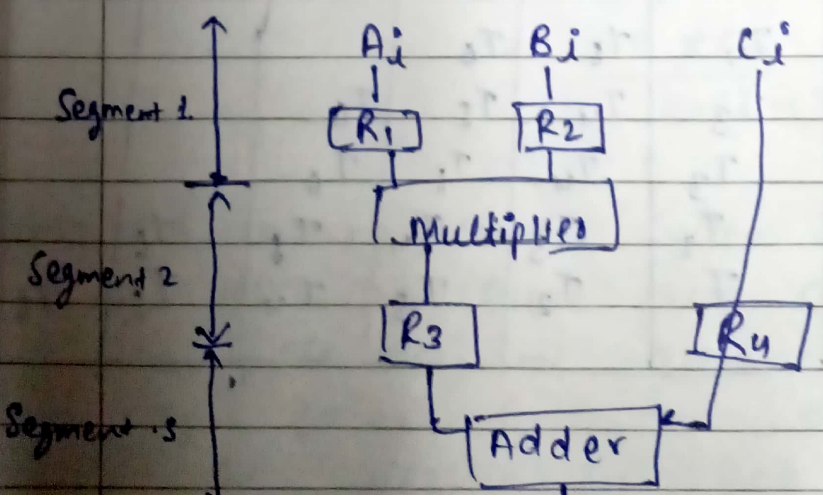
AC

Addressing modes	Effective Address	Content of AC
Direct Address	500	800
Immediate Operand	201	500
Indirect Address	800	300
Relative Address	702	325
Indexed Address	600	900
Register	-	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

Pipelining

Pipelining is a technique of decomposing a sequential process into sub-operation with each sub-process being executed in a segment that operates concurrently with all other segment.

Example: $A_i * B_i + C_i$ for $i = 1, 2, 3 \ldots 7$.



$R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$, $R_4 \leftarrow C_i$ Input A_i, B_i, C_i

$R_3 \leftarrow A_i * B_i$ Multiply

$R_5 \leftarrow R_3 + R_4$ Add.

Operations in each pipeline stage.

Clock Pulse No.	Segment 1 R_1	Segment 1 R_2	Segment 2 R_3	Segment 2 R_4	Segment 3 R_5
1	A_1	B_1	–	–	–
2	A_2	B_2	$A_1 * B_1$	C_1	–
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
	–	–	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
	–	–	–	–	$A_7 * B_7 + C_7$

Space - Time Diagram.

Segment	1	2	3	4	5	6	7	8	9
1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3		T_1	T_2	T_3	T_4	T_5	T_6		
4			T_1	T_2	T_3	T_4	T_5	T_6	
				T_1	T_2	T_3	T_4	T_5	T_6

Pipeline Speedup:

Let n: number of tasks to be performed
Pipelined machine (k segments)

k segment pipeline with a clock time of tp is used to execute n tasks.

The first task T_1 requires time = $k \times Tp$

The remaining (n-1) task emerge from the pipe at the rate of one task per cycle = $(n-1) \times Tp$

To complete n task with k-segment pipeline
= $k+(n-1)$ clock cycle.

Non-Pipelined (Conventional)
Time required to complete each task = T_n
Time required to complete n task = $n \times T_n$

Speed up:

$$S_k = \frac{\text{non-pipelined time}}{\text{Pipelined Time}}$$

$$= \frac{n \times T_n}{k+(n-1) \times Tp}$$

A non-pipelined system takes 50 ns to process a task. The same task can be processed in a six segment pipeline with a clock cycle of 10 ns. Determine the speed up ratio of the pipeline for 100 task.

Sol^{n}ⓐ. Given,

$t_n = 50ns$

$k = 6$

$t_p = 10ns$

$n = 100$.

we know that

$$S = \frac{n \times T_n}{k + (n-1) \times T_p}$$

$$= \frac{100 \times 50}{6 + (99) \times 10}$$

$$= \frac{5000}{1050} = 4.76 \text{ ns.}$$

$$S_{max} = \frac{T_n}{T_p} = \frac{50}{10} = 5ns.$$

Quer.2 Draw a space time diagram for a six segment pipeline showing the time it takes to process eight task.

Sol^{n}ⓑ. $k = 6$

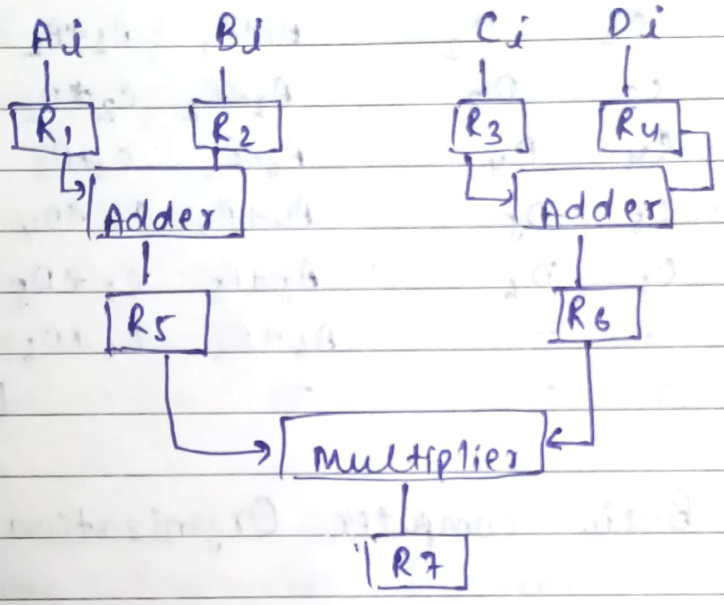
$n = 8$

$[k + (n-1)] T_p = 6 + 7 = 13$ cycle.

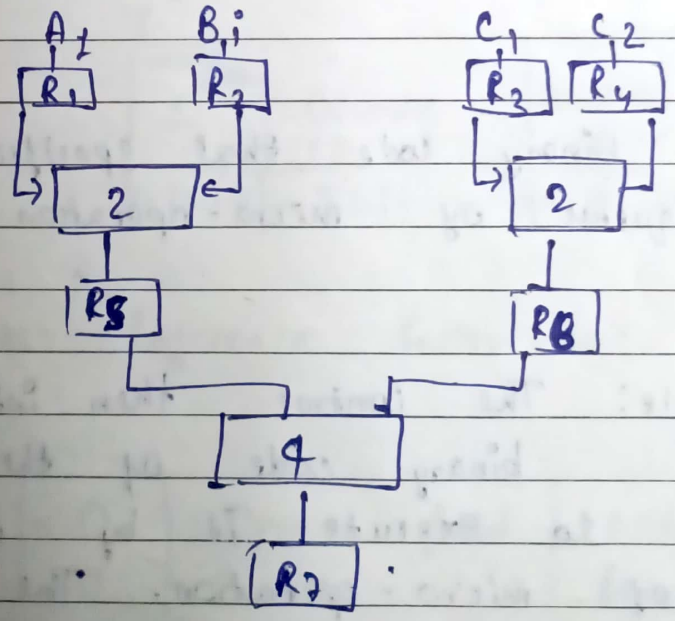
Clock → / Segment ↓	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8					
2		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8				
3			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8			
4				T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8		
5					T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	
6						T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8

Ques:3 In certain scientific calculation it is necessary to perform the arithmetic operation $(A_i + B_i) * (C_i + D_i$

a) Specify the pipeline configuration to carry out this task.



b) List the content of all register of all the register in the pipeline for i=1 to 2.



OR

no of cycle = $k+(n-1)$
= $3+(6-1)$
= 8.

Clock \ Seq.	R_1	R_2	R_3	R_4	R_5	R_6	R_7
1	A_1	B_1	C_1	D_1	-	-	-
2	A_2	B_2	C_2	D_2	A_1+B_1	A_2*B_2	-
3	A_3	B_3	C_3	D_3	A_2+B_2	C_2*D_2	$(A_1+B_1)*(C_1*D_1)$
4	A_4	B_4	C_4	D_4	A_3+B_3	C_3*D_3	$(A_2+B_2)*(C_2*D_2)$
5	A_5	B_5	C_5	D_5	A_4+B_4	C_4*D_4	$(A_3+B_3)*(C_3*D_3)$
6	A_6	B_6	C_6	D_6	A_5+B_5	C_5*D_5	$(A_4+B_4)*(C_4*D_4)$
7	-	-	-	-	A_6+B_6	C_6*D_6	$(A_5+B_5)*(C_5*D_5)$
8	-	-	-	-	-	-	$(A_6+B_6)*(C_6*D_6)$

Basic computer Organization & Design

Program: It is a set of instruction that specify the operations, operands and the sequence by which processing has to occur.

Instruction: A binary code that specifies a sequence of micro-operation for the computer.

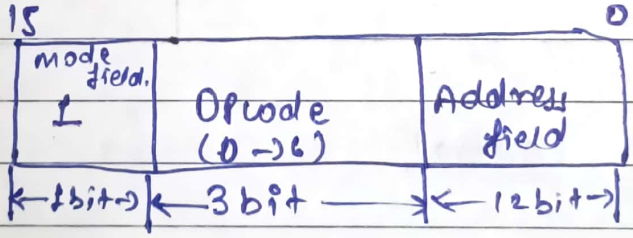
Instruction Cycle: The control then interpret the binary code of the instruction and proceeds to execute it by issuing a sequence of micro-operation. The time required to execute one instruction known as instruction cycle.

Instruction code: It is group of bits that instruct the computer to perform specific operation.

Operation code (Opcode): It is a group of bits that defines the operation. Number of bits required for opcode depends on number of operation available in computer.

Address (Operand): It specifies the location of operands (register or memory words). Memory words are specified by their address. Registers are specified by their K-bit binary code.

1. Memory reference instruction: In memory reference instruction mode field = 1, and opcode = 000 - - - 110 (0→6), i=0 (direct address), i=1 (indirect address)



2. Register Reference Instruction: Mode field (I) = 0, Opcode = 111 (7).

0	Opcode field	Addressing.
0	111	Register Operation

3. Input - Output Instruction:

Mode field (I) = 1

Opcode = 111 (7).

1	111	Inp/Oip Operation

List of Register for the Basic Computer

Register Symbol	Number of bits	Register name	Function.
DR	16	Data register	holds memory operand
AR	12	Address register	holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	holds instruction code
PC	12	Program counter	holds address of instruction
TR	16	Temporary register	holds temporary data.
INPR	8	Input register	holds input character
OUTR	8	Output register	holds output character.

Timing and control.

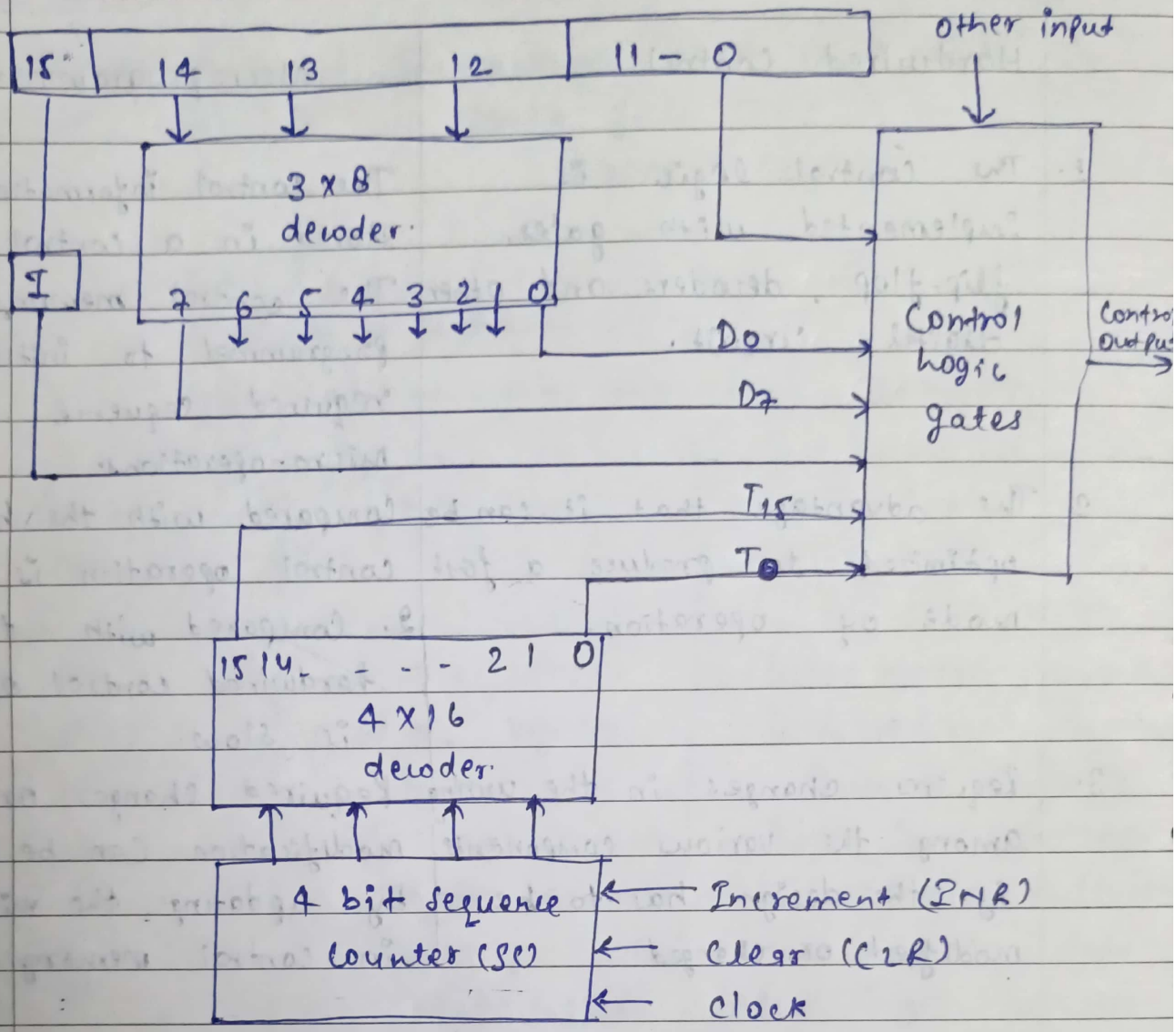
* The timing for all register in the basic computer is controlled by a master clock generator.

* The clock pulses are applied to all flip-flop and registers in the system in the system including the flip-flop and register in the control Unit.

Hardwired Control	Microprogrammed Control
1. The control logic is implemented with gates, flip-flop, decoders and other digital circuit.	The control information is stored in a control memory. The control memory is programmed to initiate the required sequence of micro-operations.
2. The advantage that it can be optimized to produce a fast mode of operation.	Compared with the hardwired control operation is slow.
3. Require changes in the wiring among the various components if the design has to be modified or changed	2. Compared with the hardwired control operation is slow. Required changes or modification can be done by updating the micro-program in control memory.

Block Diagram of Hardwired Control Unit.

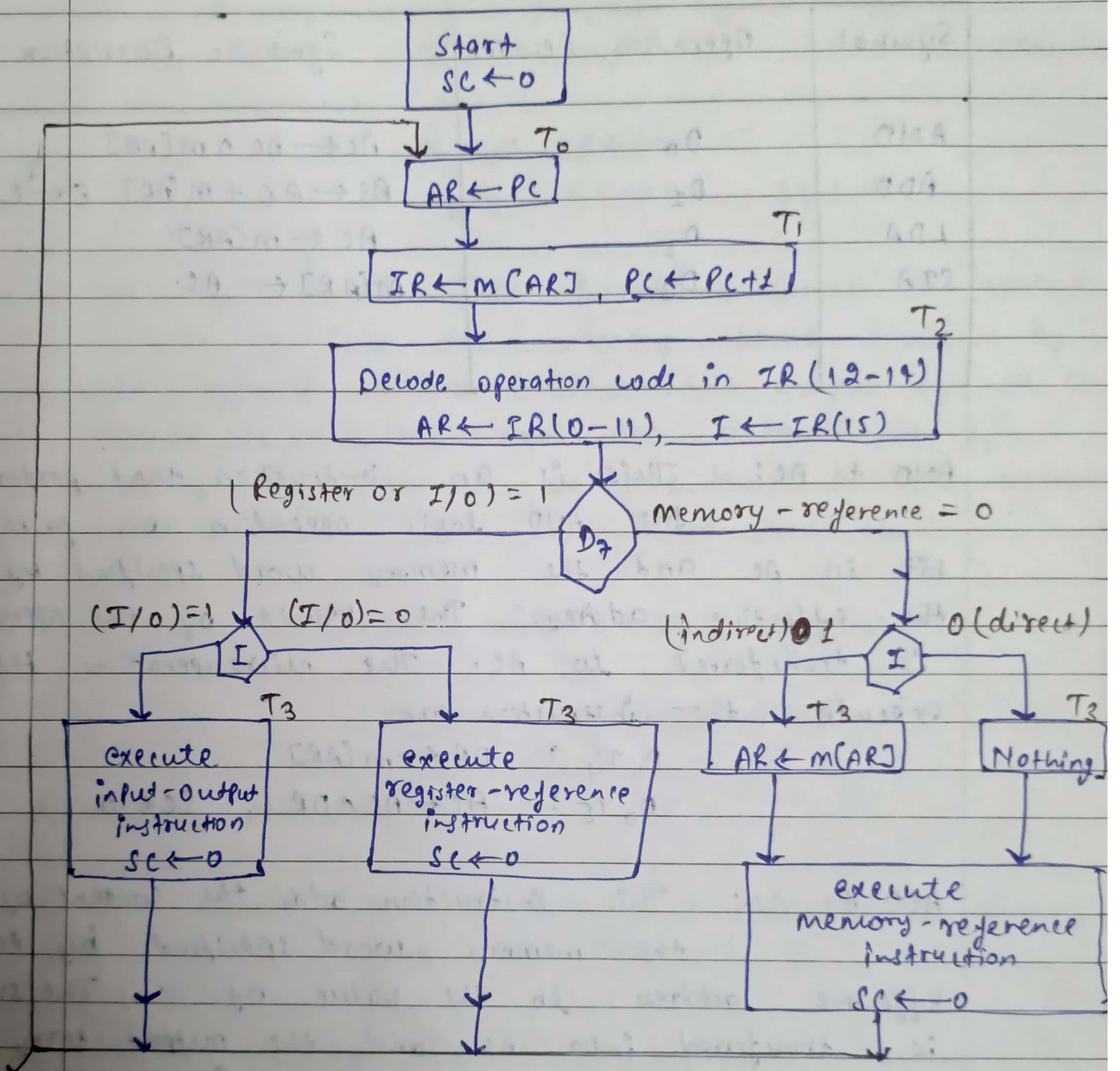
* It consist of 2 decoder, a sequence counter and a number of control logic gates.
* The operation code in bits 12 through 14 are decoded with a 3x8 decoder. The eight output of the decoder are designated by the symbol D_0 through D_7
* Bit 0 through 11 are applied to the control logic gates.



Instruction Cycle:

* A program residing in the memory unit of the computer consists of a sequence of instruction.
* The program is executed in the computer by going through a cycle for each instruction.
* Each instruction cycle in turn is sub-divided into a sequence of sub cycle or phases.

Flow chart for instruction cycle.



$D_7' I T_3$: AR ← M[AR]

$D_7' I' T_3$: Nothing

$D_7 I' T_3$: Execute a register-reference instruction

$D_7 I T_3$: Execute an input-output instruction.

Memory Reference Instruction.

Symbol	Operation Decoder	Symbolic Description.
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR]$, $E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$.

AND to AC: This is an instruction that perform the AND logic operation on pairs of bits in AC and the memory word specified by the effective address. The result of the operation is transferred to AC. The microoperation that execute the instruction are.

$D_0 T_4$: $DR \leftarrow M[AR]$

$D_0 T_5$: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$.

ADD to AC: This instruction adds the content of the memory word specified by the effective address to the value of AC. The sum is transferred into AC and the output carry C_{out} is transferred to the E (extended accumulator) flip flop. The micro operation needed to execute this instruction are.

$D_1 T_4$: $DR \leftarrow M[AR]$

$D_1 T_5$: $AC \leftarrow AC + M[AR]$, $E \leftarrow C_{out}$, $SC \leftarrow 0$

LDA: Load to AC. This instruction transfer the memory word specified by the effective address to AC. The microoperation needed to execute this instruction are.

D_2T_4: $DR \leftarrow M[AR]$

D_2T_5: $AC \leftarrow DR$, $SC \leftarrow 0$

STA: Store to AC. This instruction stores the content of AC into the memory word specified by the effective address. - Since the output of AC is applied to the bus and the data input of memory is connected to the bus, we can execute this instruction with one micro-operations.

D_3T_4: $M[AR] \leftarrow AC$, $SC \leftarrow 0$

Memory Organization

* Memory Hierarchy

1. Main Memory: Memory unit that communicates directly with the CPU (RAM)

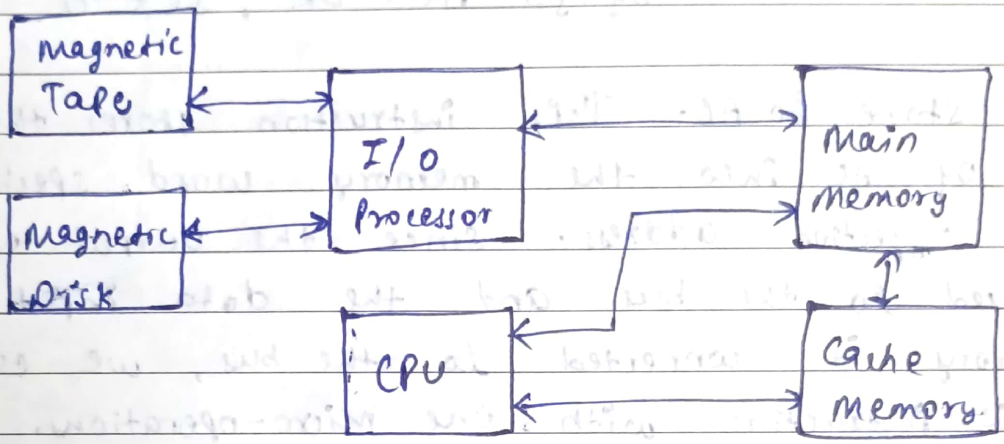
2. Auxiliary Memory: Device that provide backup storage (Disk Drives)

3. Cache Memory: Special very high speed memory to increase the processing speed (Cache RAM)

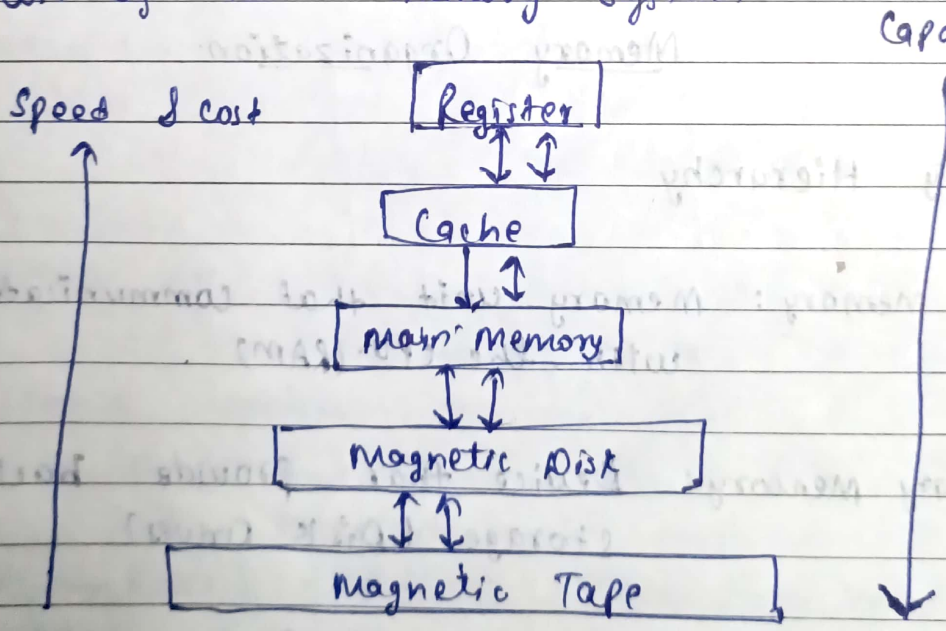
4. Multiprogramming: Enable the CPU to proces a numbe

of independent program concurrently.

Memory Management System: Supervise the flow of information between auxiliary memory and main memory.



Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system.

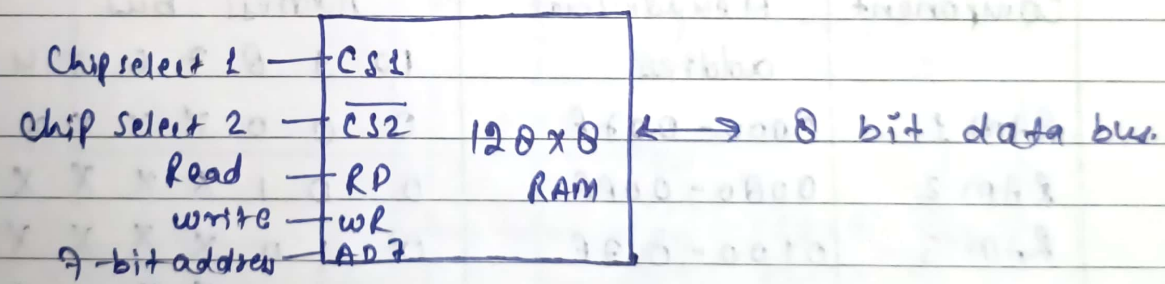


Difference between Static & Dynamic RAM.

	Static RAM	Dynamic RAM.
1.	It uses transistors to store a single bit of data.	It uses a separate capacitor to store each bit of data.
2.	It does not need periodic refreshment to maintain data.	It needs periodic refreshment to maintain data.
3.	Its structure is complex than dynamic RAM	Its structure is simplex than static RAM.
4.	They are expensive	It is less expensive.
5.	It is faster than dynamic RAM	It is slower than static RAM

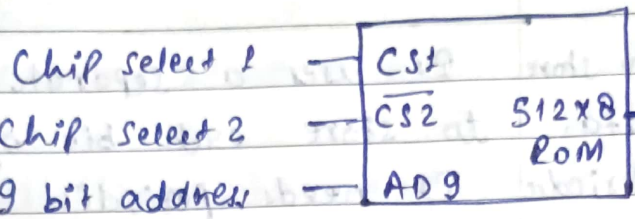
RAM and ROM Chips.

Typical RAM Chip.



CS1	CS2	RD	WR	Memory function	State of Data-bus
0	0	x	x	Inhibit	High Impedence
0	1	x	x	Inhibit	High Impedence
1	0	0	0	Inhibit	High Impedence
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High Impedence

Typical ROM Chips.



Address space assignment to each memory chip

Example: 512 bytes RAM and 512 bytes ROM.

No of RAM = $\frac{\text{Total capacity of RAM}}{\text{Single RAM capacity}}$

$= \frac{512}{128} = 4$ bytes.

No of ROM = $\frac{512}{512} = 1$ byte.

Component	Hexadecimal address	Address bus 10 9 8 7 6 5 4 3 2 1
RAM 1	0000-007F	0 0 0 x x x x x x x
RAM 2	0080-00FF	0 0 1 x x x x x x x
RAM 3	0100-017F	0 1 0 x x x x x x x
RAM 4	0180-01FF	0 1 1 x x x x x x x
ROM	0200-03FF	1 x x x x x x x x x

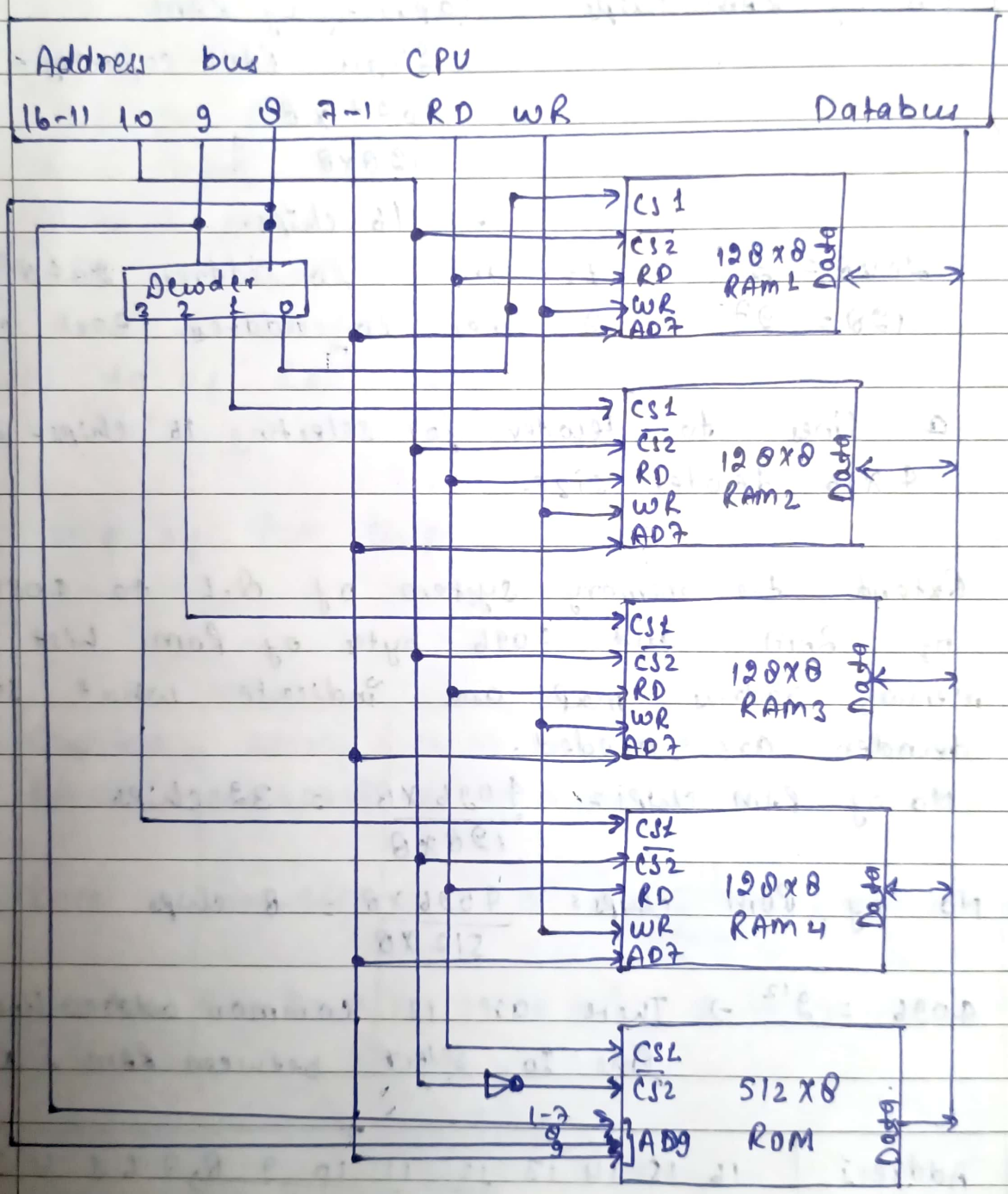
Memory Connection to CPU

→ RAM and ROM chips are connected to a CPU through the data and address buses.

→ The low-order line in the address bus select the byte within the chips and other lines in the

address bus select a particular chip through its chip select inputs.

Connection of Memory to CPU:



Quiz: 1a) How many 128x8 RAM chips are needed to provide a memory capacity of 2048 bytes.

b) How many lines of the address bus must be

used to access 2048 bytes of memory. How many of these lines will be common to all chips.

c) How many list must be decoded for chip select? Specify the size of decoder.

Sol^n a) No of RAM chips = $\frac{\text{Capacity of RAM}}{\text{Single RAM capacity}}$

$= \frac{2048 \times 8}{128 \times 8}$

$= 16$ chips.

b) $2048 = 2^{11}$, 11 lines to address 2048 byte.
$128 = 2^7$, 7 lines to address each chip

c) 4 lines to decoder for selecting 16 chips $= 2^4 = 16$.
4×16 decoder size.

Ques: 2 Extend the memory system of Q.1 to 4096 bytes of RAM and 4096 byte of Rom. List the memory address map and indicate what size decoder are needed.

Sol^n ⇒ No of RAM chips = $\frac{4096 \times 8}{128 \times 8}$ = 32 chips.

No of ROM chips = $\frac{4096 \times 8}{512 \times 8}$ = 8 chips.

$4096 = 2^{12}$ → There are 12 common address line + 1 line to select between RAM & ROM

Component	Address	16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
RAM	0000-0fff	0 0 0 0 ← 5×32 decoder → x x x x x x x
ROM	1000-1fff	0 0 0 1 ← 3×8 decoder → x x x x x x x x

Ques:3 A computer employs RAM chips of 256 x 8 and ROM chips of 1024 x 8. The computer system needs 2K bytes of RAM, 4K bytes of ROM, and four interface units each with four registers. A memory mapped I/O configuration used. The two highest order bits of the address bus are assigned 00 for RAM 01 for ROM, and 10 for interface registers.

a) How many RAM & ROM chips are needed.

b) Draw a memory address map for the system.

c) Give the address range in hexadecimal for RAM, ROM & Interface.

Sol:- a) No of RAM chips = $\frac{2 \times 1024 \times 8}{256 \times 8}$ = 8 chips

$= 2^3$

$= 3 \times 8$ decoder.

No of ROM chips = $\frac{4 \times 1024 \times 8}{1024 \times 8}$ = 4 chips

$= 2^2$

$= 2 \times 4$ decoder.

b)

Component	Hexadecimal Address	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM	0000-07FF	0	0	0	0	0	3x8 decoder			x	x	x	x	x	x	x	x
ROM	4000-4FFF	0	1	0	0	2x4 decoder				x	x	x	x	x	x	x	x
Interface	8000-800F	1	0	0	0	0	0	0	0	0	0	0	x	x	x	x	

Cache Memory.

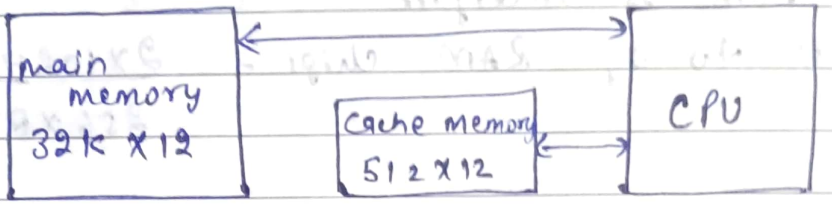
Locality of Reference: The references to memory tend to be confirmed within a few localized areas in the memory.

Cache Memory: A fast small memory : It keeps the most frequently accessed instruction and data in the fast cache memory.

Hit Ratio: The ratio of the number of hits divided by the total CPU reference (hit + misses) to memory:

hit: The CPU finds the words in the cache.

Miss: The word is not found in cache (CPU must read main memory).



Mapping: The transformation of data from main memory to cache memory.

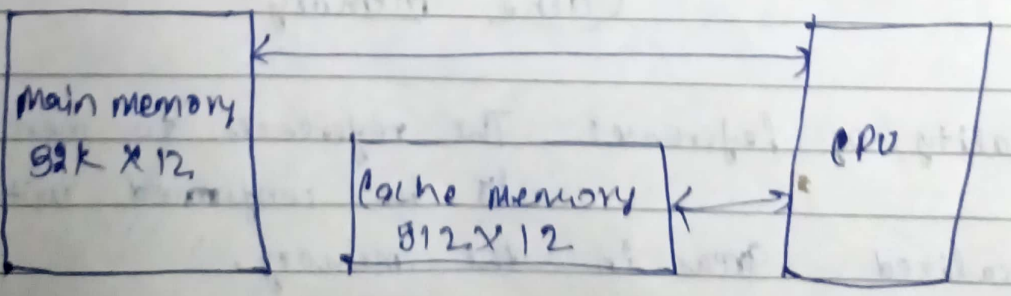
1) Associative Memory: Main memory : 32K x 12 bit word (15 bit address)

Cache Memory: 512 x 12 bit word.

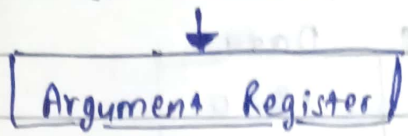
CPU sends a 15-bit address to cache.

Hit: CPU accept the 12 bit data from cache.

Miss: CPU read the data from main memory (then data is written to cache)



CPU address (15 bits)



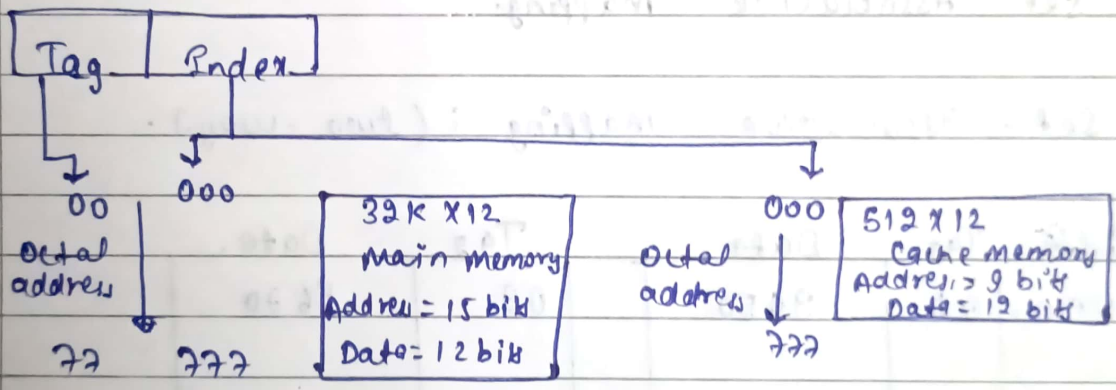
Address	Data
01000	3450
02777	6710
22345	1234

2) Direct Mapping: n bit memory address

Tag field (n-k) : Index field (k)

2^k words cache memory + 2^n words main memory

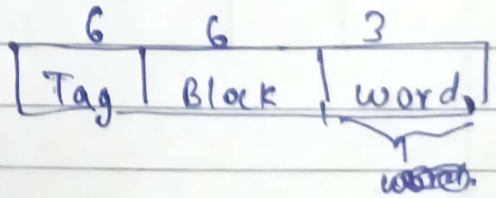
Tag = 6 bit (15-0) , Index = 9 bit



* Direct mapping Cache with block size of 8 words.

64 blocks X 8 word = 512 cache words size.

	Index	Tag	Data
Block 0	000	01	3450
	007	01	6578
Block 1	010		
	017		
Block 63	770	02	
	777	02	6710



Memory address	Memory data
00000	1220
00777	2345
01000	3450
01777	4560
02000	5670
02777	6710

a) Main memory

Index address	Tag	Data
000	00	1220
777	02	6710

b) Main memory.

3) Set - Associative Mapping.

Set - associative mapping : (two - way).

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340