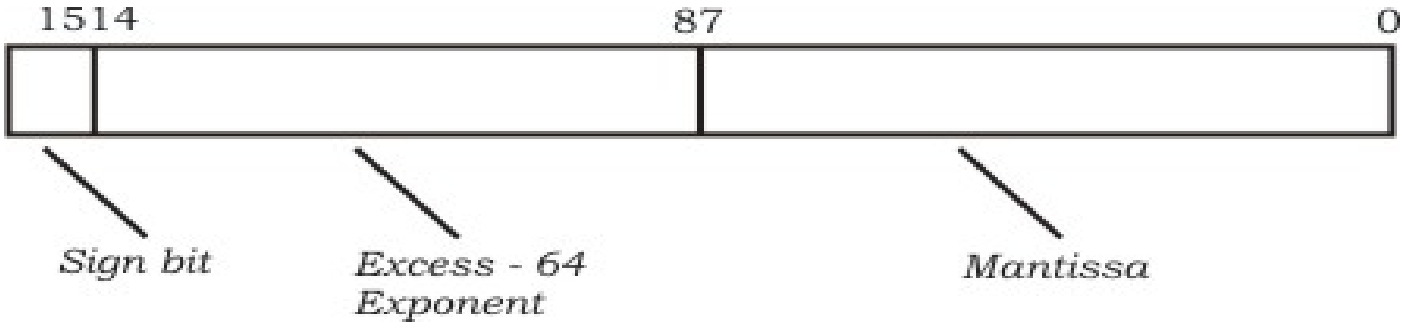
Floating point representation

- Floating point number is stored in mantissa/exponent form i.e. $m*r^e$
- Mantissa is a signed magnitude fraction for most of the cases.
- The exponent is stored in biased form.
- The biased exponent is an unsigned number representing signed true exponent.
- If the biased exponent field contains K bits, the biased = 2^{k-1}
- True value expression is $V = (-1)^S(.M)_2*2^{E-Bias}$, note it is explicit representation
  True value expression is $V = (-1)S(1.M)_2*2^{E-Bias}$, note it is implicit representation, it has more precision than explicit normalization.
- Biased exponent(E) = True exponent(e) + Bias
- Range of true exponent: from -2^{k-1} to $+2^{k-1}-1$, where k is number of bits assigned for exponent
- After adding bias 2^{k-1}, new range go from 0 to 2^k-1
- How to convert a signed number into floating point representation
- The floating-point normalized number distribution is not uniform. Representable number are dense towards zero and spared towards max value
- This uneven distribution result negligible effect of rounding towards zero and dominant towards max value.

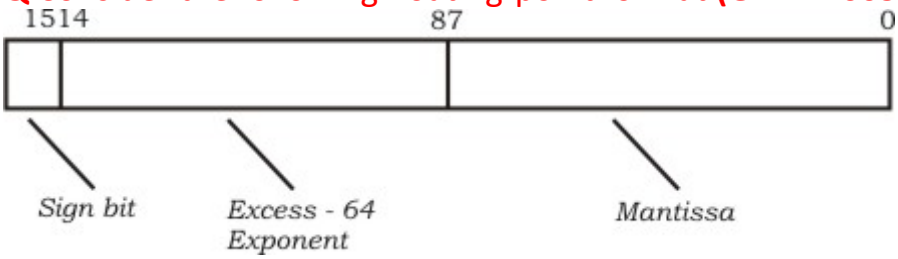
Q Represent -21.75 (s=1, k=7, m=8)

Q Consider the following floating-point format



Mantissa is a pure fraction in sign-magnitude form. The decimal number 0.239×2^{13} has floating point representation_____?

Q Consider the following floating-point format **(GATE-2005) (2 Marks)**

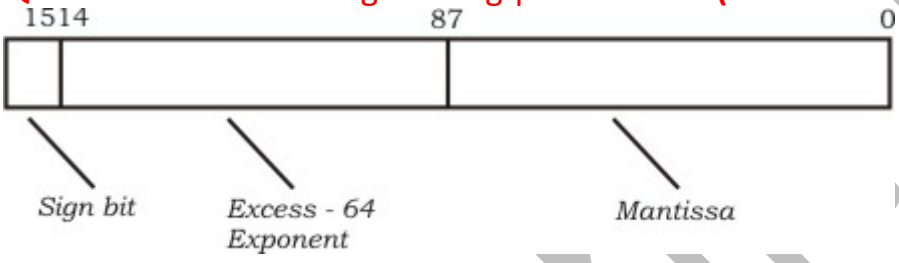


Mantissa is a pure fraction in sign-magnitude form. The decimal number 0.239×2^{13} has the following hexadecimal representation (without normalization and rounding off:

(A) 0D 24 **(B)** 0D 4D **(C)** 4D 0D **(D)** 4D 3D

Answer: (D)

Q Consider the following floating-point format **(GATE-2005) (2 Marks)**



Mantissa is a pure fraction in sign-magnitude form. The normalized representation for the above format is specified as follows. The mantissa has an implicit 1 preceding the binary (radix) point. Assume that only 0's is padded in while shifting a field. The normalized representation of the above number (0.239×2^{13}) is:

(A) 0A 20 **(B)** 11 34 **(C)** 4D D0 **(D)** 4A E8

Answer: (D)

Q The range of representable normalized numbers in the floating-point binary fractional representation in a 32-bit word with 1-bit sign, 8-bit excess 128 biased exponent and 23-bit mantissa is **(NET-DEC-2014)**

(A) 2^{-128} to $(1 - 2^{-23}) \times 2^{127}$

(B) $(1 - 2^{-23}) \times 2^{-127}$ to 2^{128}

(C) $(1 - 2^{-23}) \times 2^{-127}$ to 2^{23}

(D) 2^{-129} to $(1 - 2^{-23}) \times 2^{127}$

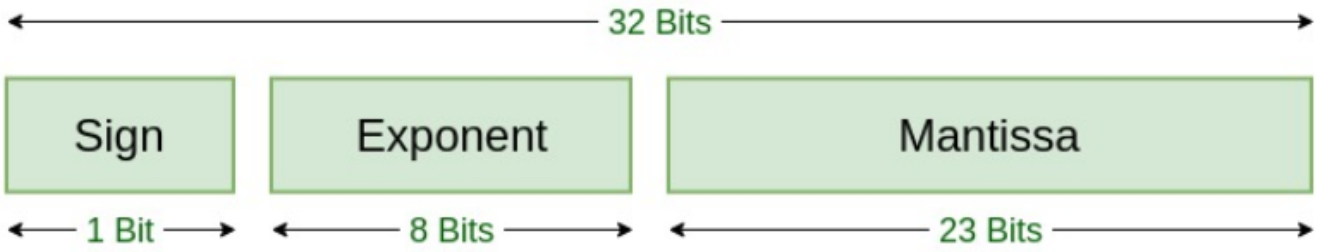
Ans: a

IEEE 754 floating point standard

- IEEE 754 standard represent floating point number standards
- It gives a provision for +-0 & +-∞ (by reserving certain pattern for Mantissa/Exponent pattern)
- there are number of modes for storage from half precision (16 bits) to very lengthy notations
- based of the system is 2
- the floating-point number can be stored either in implicit normalized form or in fractional form
- If the biased exponent field contains K bits, the biased = 2^{k-1} -1
- Certain Mantissa/Exponent pattern does not denote any number (NAN i.e. not a number)

Name	Common name	Mantissa Bits	Exponent bits	Exponent bias	E min	E max
binary16	Half precision	10	5	$2^{5-1}-1 = 15$	−14	+15
binary32	Single precision	23	8	$2^{8-1}-1 = 127$	−126	+127
binary64	Double precision	52	11	$2^{11-1}-1 = 1023$	−1022	+1023
binary128	Quadruple precision	112	15	$2^{15-1}-1 = 16383$	−16382	+16383
binary256	Octuple precision	236	19	$2^{19-1}-1 = 262143$	−262142	+262143

Single pression



Sign bit (1)	Exponent (8)	Mantissa (23)	Value
0	00…….0(E=0)	00…….0(M=0)	+0
1	00…….0(E=0)	00…….0(M=0)	-0
0	11……1(E=255)	00…….0(M=0)	+∞
1	11……1(E=255)	00…….0(M=0)	-∞
0/1	1<=E<=254	XX…….X (M! =0)	Implicit normalised number
0/1	00…….0(E=0)	XX…….X (M! =0)	Fraction
0/1	11……1(E=255)	XX…….X (M! =0)	NAN (Not a Number)

Q Given the following binary number in 32-bit (single precision) IEEE-754 format:
00111110011011010000000000000000
The decimal value closest to this floating-point number is (GATE-2017) (2 Marks)

(A) 1.45×10^{1} **(B)** 1.45×10^{-1}

(C) 2.27×10^{-1} **(D)** 2.27×10^{1}

ANSWER- C

Q (NET-DEC-2018)

The decimal floating point number –40.1 represented using IEEE–754 32-bit representation and written in hexadecimal form is

Options :

91394342261. 0xC2206666

91394342262. 0xC2206000

91394342263. 0xC2006666

91394342264. 0xC2006000

Q The decimal value 0.5 in IEEE single precision floating point representation has **(GATE-2012) (2 Marks)**

a) fraction bits of 000…000 and exponent value of 0
b) fraction bits of 000…000 and exponent value of –1
c) fraction bits of 100…000 and exponent value of 0
d) no exact representation
ANSWER b

Q The following bit pattern represents a floating-point number in IEEE 754 single precision format **(GATE-2008) (1 Marks)**
1 10000011 10100000000000000000000
The value of the number in decimal form is
(A) -10 **(B)** -13 **(C)** -26 **(D)** None of these
Answer: (C)

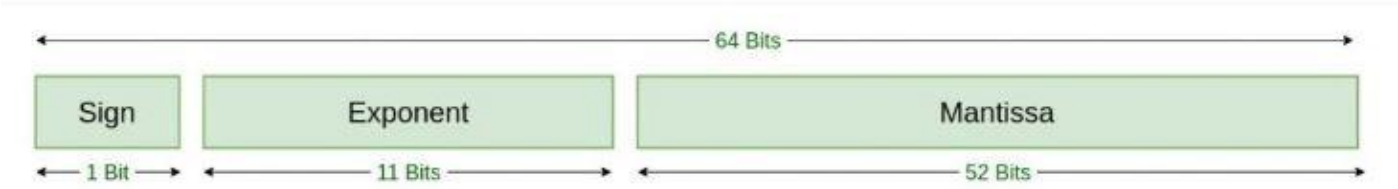
Q In the IEEE floating point representation, the hexadecimal value 0×00000000 corresponds to **(GATE-2008) (2 Marks)**

a) The normalized value 2^{-127}
b) The normalized value 2^{-126}
c) The normalized value +0
c) The special value +0
ANSWER D

Q How the number 85.125 will be represented in the IEEE floating point representation in single precision in hexadecimal notation?

Q In the IEEE floating point representation in single precision in hexadecimal notation is 40400000, then the floating-point value will be?

Double Precision



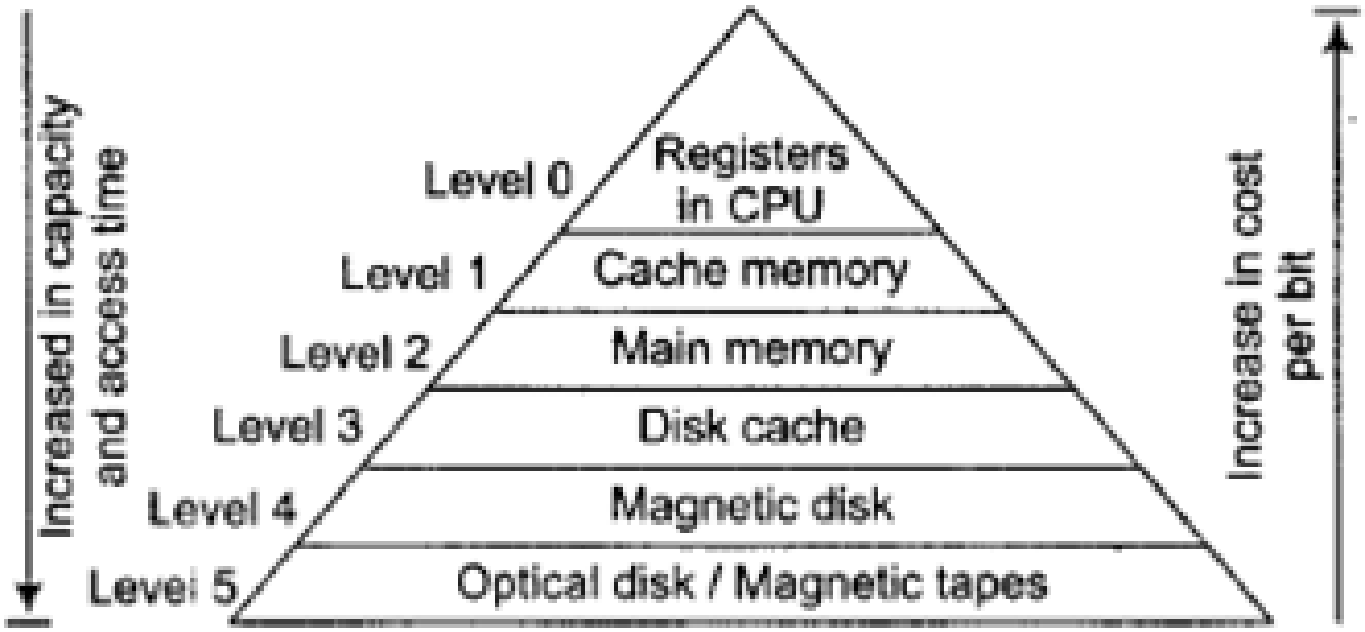
Q The IEEE-754 double-precision format to represent floating point numbers, has a length of _____ bits. **(NET-JULY-2016)**

(a) 16 **(b)** 32 **(c)** 48 **(d)** 64

Ans: d

Memory Hierarchy

- The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory accessible to the high-speed processing logic



- At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files.
- Next are the magnetic disks used as backup storage.
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
- When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory.
- Programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.
- A special very-high speed memory called a Cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.
- The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic.

Cache Memory

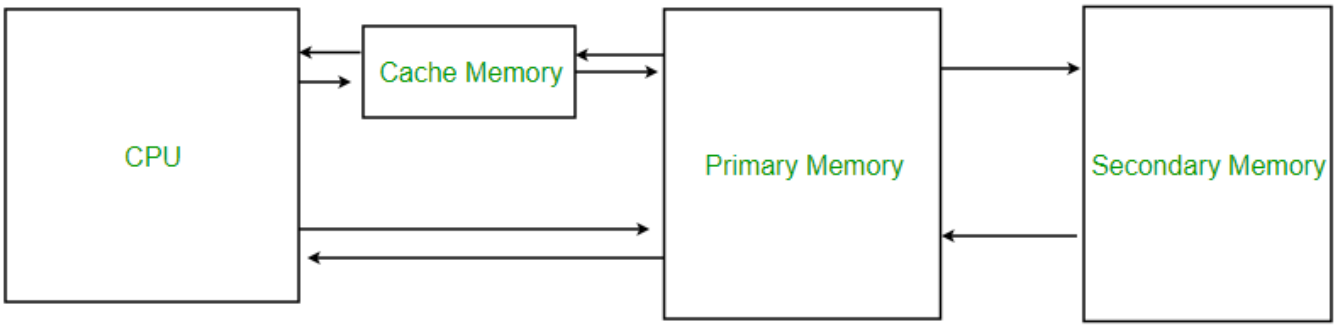
- ***Locality of Reference:*** the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference. There are two types of locality of reference:
    - **Spatial Locality:** Spatial locality refers to the use of data elements in the nearby locations.
    - **Temporal Locality:** Temporal locality refers to the reuse of specific data, and/or resources, within a relatively small-time duration. Or, the most frequently used items will be needed soon. (LRU and LFU is used for temporal locality)

Q The principle of locality justifies the use of: **(Gate-1995) (1 Marks)**
a) Interrupts **b)** DMA **c)** Polling **d)** Cache Memory
Answer: (D)

Q More than one word are put in one cache block to **(Gate-2001) (1 Marks)**
(A) exploit the temporal locality of reference in a program
(B) exploit the spatial locality of reference in a program
(C) reduce the miss penalty
(D) none of the above
Answer: (B)

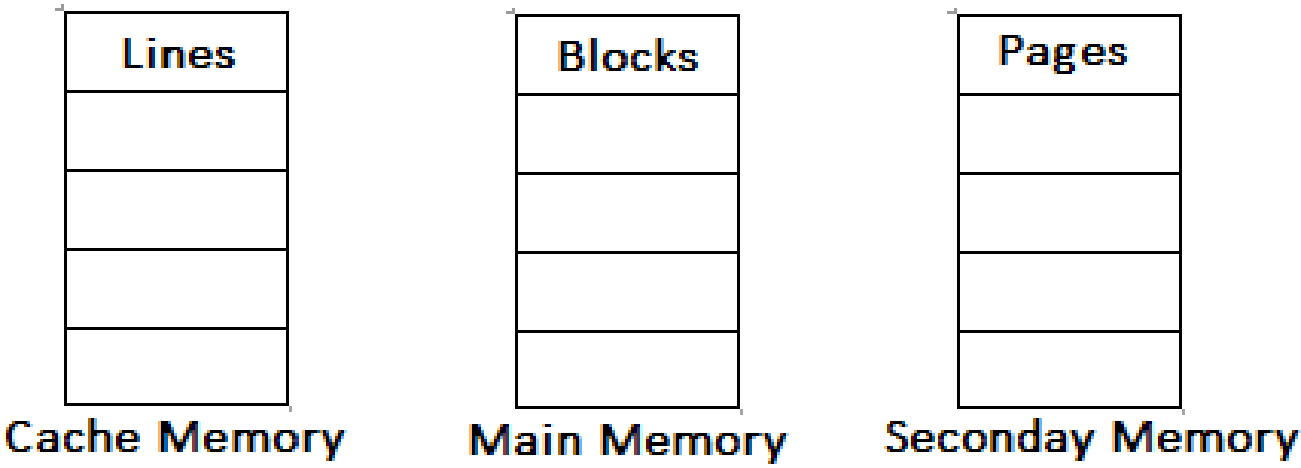
- The average memory access time of a computer system can be improved considerably by use of a cache.
- **Cache Hit:** The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word in cache, it is said to produce a hit.
    - **Hit Latency:** The time required to match the data into cache memory is known as hit latency.
- **Cache Miss:** If the word is not found in cache, it is in main memory and it counts as a miss.
    - **Miss Latency:** When cache miss occurs, the time required to get the needed data into cache memory from main memory. **Or,** The time required to recover from a cache miss.
- If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of the fast cache memory.
- **Page Hit:** when the required data is present in the main memory it is known as page hit.
    - **Page miss / Page Fault:** When the required data is not present in the main memory is known as page hit/fault.

- **Page fault service time:** The time required to recover from the page fault is known as page fault service time. Page fault is recovered by bringing a page from secondary memory to main memory.



Cache Mapping

- Main memory is divided into frames in OS, but here will call it block, so 1 frame of main memory in os = 1 block of main memory in
- Cache is also divided into same size blocks called lines. 1 block of main memory = 1 line in cache
- Blocks are further divided into words.
- We use the notion 1 word = 1 byte.

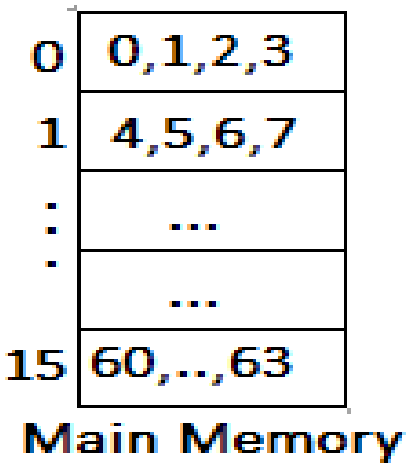


- K = m%n
- M is main memory block no, no of cache lines are n, then m^{th} block will be present in kth line of cache

Q Consider that the main memory has 64 words and block size is 4 words. Let us assume that the cache is of 16 words?

Ans: Then, total number of blocks the main memory will have: 64/4 = 16 blocks. Such that, it has blocks from 0 to 15 and in each block, we have 4 words.

Q In designing a computer's cache system, the cache block (or cache line) size is an important parameter. Which one of the following statements is correct in this context?
(Gate-2014) (1 Marks)
(A) A smaller block size implies better spatial locality
(B) A smaller block size implies a smaller cache tag and hence lower cache tag overhead
(C) A smaller block size implies a larger cache tag and hence lower cache hit time
(D) A smaller block size incurs a lower cache miss penalty
Answer: (D)

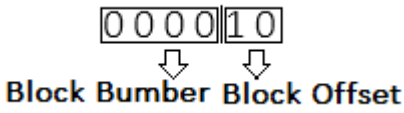


The number of lines in cache: total words in cache / block size = 16/4 = 4 lines.

Now, for 64 words we will need 6 bits to represent them as $2^6 = 64$.

Assume that the CPU generates a 6-bit address as: 000010, it means that the CPU wants to access the word number 3.

We divide the CPU address: 0 0 0 0 1 0



The 2 LSB are denoting the block offset and 4 MSB are denoting the Block number.

That means that: Go to block number 0 and look at offset 3 which is word number 3.

- The transformation of data from main memory to cache memory is referred to as a mapping process.

- Three types of mapping procedures:
  - ***Direct mapping***
  - ***Associative mapping***
  - ***Set-associative mapping***

Q Which of the following mapping is not used for mapping process in cache memory? **(NET-JULY-2018)**

(a) Associative mapping
(b) Direct mapping
(c) Set-Associative mapping
(d) Segmented - page mapping

Q A processor can support a maximum memory of 4 GB, where the memory is word-addressable (a word consists of two bytes). The size of the address bus of the processor is at ____ least bits. **(Gate-2016) (1 Marks)**

Answer: (31)

Direct Mapping

- In direct mapping scheme the main memory blocks are directly mapped onto a particular cache memory line. It is also known as many to one mapping. **Example:** In the above given example; the 15 blocks of the main memory will be mapped into cache in such a way that:

Main Memory	
B-0	W-0
	W-1
	W-2
	W-3
B-1	W-4
	W-5
	W-6
	W-7
B-2	W-8
	W-9
	W-10
	W-11
B-3	W-12
	W-13
	W-14
	W-15
B-4	W-16
	W-17
	W-18
	W-19
B-5	W-20
	W-21
	W-22
	W-23
B-6	W-24
	W-25
	W-26
	W-27
B-7	W-28
	W-29
	W-30
	W-31
B-8	W-32
	W-33
	W-34
	W-35
B-9	W-36
	W-37
	W-38
	W-39
B-10	W-40
	W-41
	W-42
	W-43
B-11	W-44
	W-45
	W-46
	W-47
B-12	W-48
	W-49
	W-50
	W-51
B-13	W-52
	W-53
	W-54
	W-55
B-14	W-56
	W-57
	W-58
	W-59
	W-60
	W-61

B-15	W-62
	W-63

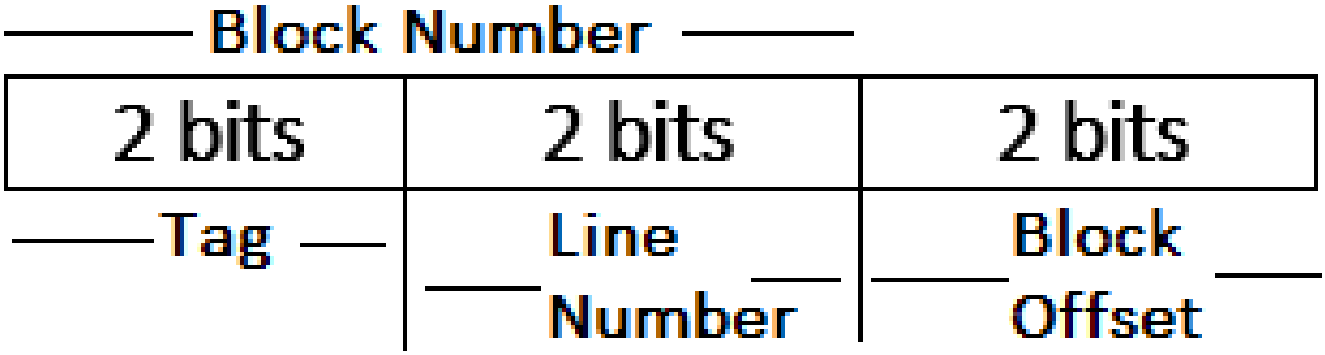
Main Memory				
B-0	W-0	W-1	W-2	W-3
B-1	W-4	W-5	W-6	W-7
B-2	W-8	W-9	W-10	W-11
B-3	W-12	W-13	W-14	W-15
B-4	W-16	W-17	W-18	W-19
B-5	W-20	W-21	W-22	W-23
B-6	W-24	W-25	W-26	W-27
B-7	W-28	W-29	W-30	W-31
B-8	W-32	W-33	W-34	W-35
B-9	W-36	W-37	W-38	W-39
B-10	W-40	W-41	W-42	W-43
B-11	W-44	W-45	W-46	W-47
B-12	W-48	W-49	W-50	W-51
B-13	W-52	W-53	W-54	W-55
B-14	W-56	W-57	W-58	W-59
B-15	W-60	W-61	W-62	W-63

Cache								
CL-0	B-0	W-0	B-4	W-16	B-8	W-32	B-12	W-48
		W-1		W-17		W-33		W-49
		W-2		W-18		W-34		W-50
		W-3		W-19		W-35		W-51
CL-1	B-1	W-4	B-5	W-20	B-9	W-36	B-13	W-52
		W-5		W-21		W-37		W-53
		W-6		W-22		W-38		W-54
		W-7		W-23		W-39		W-55
CL-2	B-2	W-8	B-6	W-24	B-10	W-40	B-14	W-56
		W-9		W-25		W-41		W-57
		W-10		W-26		W-42		W-58
		W-11		W-27		W-43		W-59
CL-3	B-3	W-12	B-7	W-28	B-11	W-44	B-15	W-60
		W-13		W-29		W-45		W-61
		W-14		W-30		W-46		W-62
		W-15		W-31		W-47		W-63

Cache Memory	
CL-0	B-0 / B-4 / B-8 / B-12
CL-1	B-1 / B-5 / B-9 / B-13
CL-2	B-2 / B-6 / B-10 / B-14
CL-3	B-3 / B-7 / B-11 / B-15

CL-0	TAG = Block no – Cache line	W-
		W-
		W-
		W-
CL-1	TAG = Block no – Cache line	W-
		W
		W-
		W-
CL-2	TAG = Block no – Cache line	W-
		W-
		W-
		W-
CL-3	TAG = Block no – Cache line	W-
		W-
		W-
		W-

Now, further analysing this structure we will find out that the 6-bit address generated by CPU is now divided into:

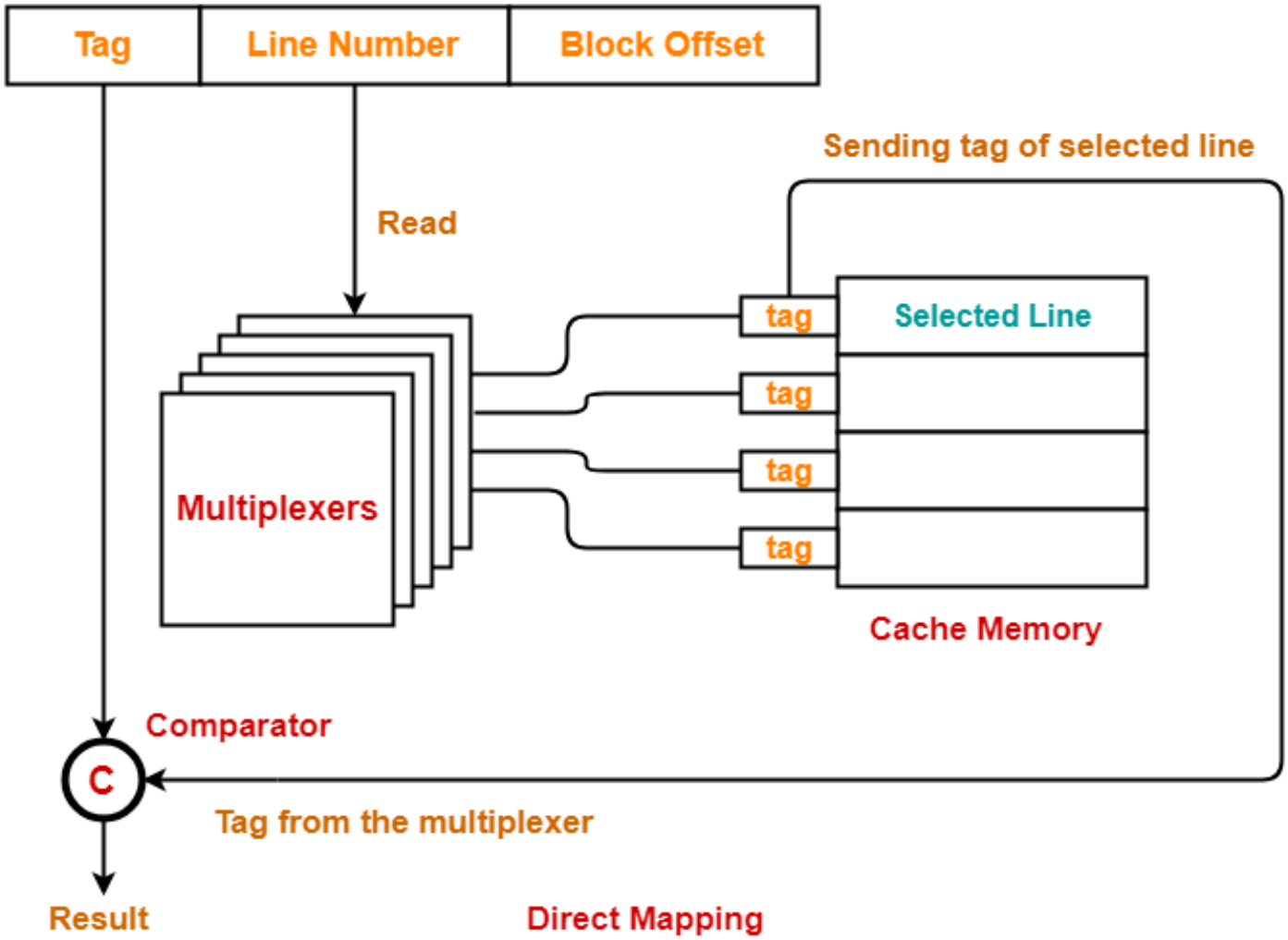


- The block number is further divided into two parts: **Tag** and **Line Number.**
- The line number will specify at which line in the cache is the data present.
- The tag will specify the location in cache that needs to be searched.

<table>
<tr><td colspan="3">Main Memory</td></tr>
<tr><td colspan="2">Block Number</td><td>Block Offset</td></tr>
<tr><td>Tag</td><td>Line Number</td><td></td></tr>
<tr><td></td><td colspan="2">Cache</td></tr>
</table>

Architecture of Direct Mapping

- If we have ‘n’ lines in cache, then n × 1 multipliers are required.
- If tag size is ‘n’ bits then ‘n’ number of comparators are required.
- If tag size is ‘n’ bits then ‘n’ numbers of MUX’s are required.



- **Hit latency = Multiplexer latency + Comparator latency**

Advantages

- Direct addressing is faster as time required to access is less.
- Hardware cost of direct addressing is less.

Disadvantages

- The main disadvantage of direct mapping is conflict miss.

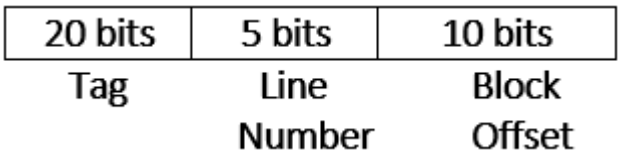
Q Let the Main memory size: 32 GB, cache size: 32 KB, Block size: 1KB. Find the tag bits and tag directory size?

Since main memory size: 32 GB, so Physical address: $2^{30} * 2^5 = 2^{35}$ = 35 bits

Block Size: 1 KB so, **block offset: 2^{10} = 10 bits**

Line Number = Cache size / Block Size = 32 KB / 1 KB = 2^5 = 5 bits.

Since out of 35 bits, we have 10 bits for offset and 5 bits for line number, then remaining 20 bits are for Tag.



- Now the **Tag directory: Number of lines in cache * tag size** = 20 bits for tag * 2^5 (each line is going to take 20 bits for tag)

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size
128 KB	16 KB	256 B		
32 GB	32 KB	1 KB		
	512 KB	1 KB	7	
16 GB		4 KB	10	
64MB			10	
	512 KB		7	

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size
128 KB	16 KB	256 B	3	$3 * 2^6$
32 GB	32 KB	1 KB	20	$20 * 2^5$
2^{26}B	512 KB	1 KB	7	$7*2^9$
16 GB	2^{24} B	4 KB	10	$10*2^{12}$
64 MB	2^{16} B	?	10	?
2^{26} B	512 KB	?	7	?

Q Consider a machine with byte addressable memory of 2^{32} bytes divided into blocks of size 32 bytes. Assume a direct mapped cache having 512 cache lines is used with this machine. The size of tag field in bits is _____ **(Gate-2017) (2 Marks)**

(A) 12 **(B)** 16 **(C)** 18 **(D)** 24

Answer: (C)

Q Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32-bit addresses. The number of bits needed for cache indexing and the number of tag bits are respectively **(Gate-2005) (2 Marks)**

(A) 10, 17 **(B)** 10, 22 **(C)** 15, 17 **(D)** 5, 17
Answer: (A)

Q 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 two consecutive bytes in main memory be $(E201F)_{16}$ and $(E2020)_{16}$. What are the tag and cache line address (in hex) for main memory address $(E201F)_{16}$? **(Gate-2015) (1 Marks)**
(A) E, 201 **(B)** F, 201 **(C)** E, E20 **(D)** 2, 01F
Answer: (A)

Q An 8KB direct-mapped write-back cache is organized as multiple blocks, each of size 32-bytes. The processor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following.
1 Valid bit
1 Modified bit
As many bits as the 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-data (tags) for the cache? **(Gate-2011) (2 Marks)**
(A) 4864 bits **(B)** 6144 bits **(C)** 6656 bits **(D)** 5376 bits
Answer: (D)

Associative Mapping

- To overcome the problem of conflict-miss in direct mapping we have Associative Mapping.
- A block of main memory can be mapped to any freely available cache line. This makes fully associative mapping more flexible than direct mapping.
- A replacement algorithm is needed to replace a block if the cache is full.
- It is also known as many to many mappings.
- In fully associative mapping we only have two fields: Tag/Block Number field and a Block offset field.
- Here the number of bits in tag = number of bits to require to represent block number

Tag / Block Number	Block Offset

Example:

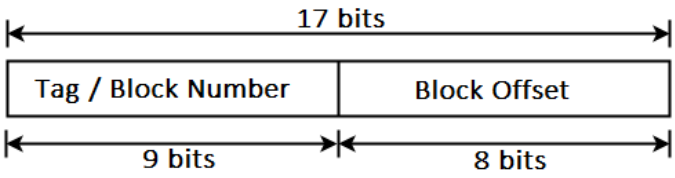
Q Consider a fully associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find out the: Number of bits in tag and Tag directory size?

Number of bits in physical address: Main memory = 128 KB = 2^{17} bytes**,** Number of bits in physical address = 17 bits

Number of Bits in Block Offset: Block size = 256 bytes = 2^8 bytes,

Number of bits in block offset = 8 bits

Number of bits in tag = Number of bits in physical address – Number of bits in block offset = 17 bits – 8 bits = 9 bits



Tag directory size = Number of tags x Tag size = Number of lines in cache x Number of bits in tag

Number of lines in cache = Cache size / Block Size = 16KB / 256 B = 2^6 line

Tag directory size = 2^6 * 9 bits = 576 bits = 72 bytes

Main Memory	
Block Number	Block Offset
Tag = block Number	
Cache	

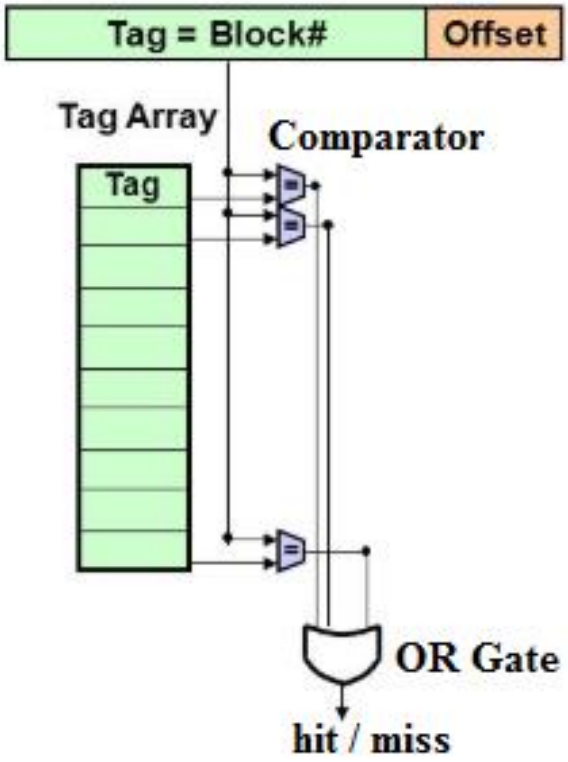
Main Memory				
B-0	W-0	W-1	W-2	W-3
B-1	W-4	W-5	W-6	W-7
B-2	W-8	W-9	W-10	W-11
B-3	W-12	W-13	W-14	W-15
B-4	W-16	W-17	W-18	W-19
B-5	W-20	W-21	W-22	W-23
B-6	W-24	W-25	W-26	W-27
B-7	W-28	W-29	W-30	W-31
B-8	W-32	W-33	W-34	W-35
B-9	W-36	W-37	W-38	W-39
B-10	W-40	W-41	W-42	W-43
B-11	W-44	W-45	W-46	W-47
B-12	W-48	W-49	W-50	W-51
B-13	W-52	W-53	W-54	W-55
B-14	W-56	W-57	W-58	W-59
B-15	W-60	W-61	W-62	W-63

CL-0	TAG = Block no	W-
		W-
		W-
		W-
CL-1	TAG = Block no	W-
		W
		W-
		W-
CL-2	TAG = Block no	W-
		W-
		W-
		W-
CL-3	TAG = Block no	W-
		W-
		W-
		W-

Main Memory		
Block Number		Block Offset
Tag	Set number	
	Cache	

Hardware Architecture

- If we have 'n' lines in cache then 'n' number of comparators are required.
- Size of comparator = Size of Tag
- If we have 'n' bit tag then we require 'n' bit comparator.



Hit latency = Time taken by one comparator (Comparators are in Parallel) + time taken by OR Gate

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Comp
128 KB	16 KB	256 B			
32 GB	32 KB	1 KB			
	512 KB	1 KB	17		
16 GB		4 KB	22		
64MB			10		
	512 KB		7		

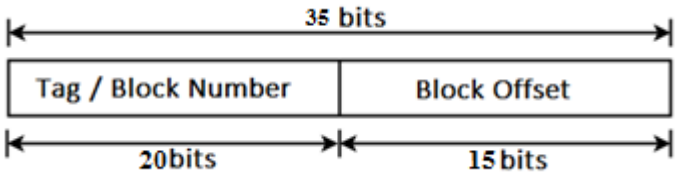
MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Comp
128 KB	16 KB	256 B	9	9 * 64	64
32 GB	32 KB	1 KB	25	25 * 32	32
128 MB	512 KB	1 KB	17	512 *17	512
16 GB	?	4 KB	22	?	?
64MB	?	64 KB	10	?	?
?	512 KB		7	?	?

Example: Let us assume that main memory size is 32GB, block size is 32KB and the propagation delay for the comparators and OR gate is 10 K ns and 10 ns respectively. Calculate the tag size and hit latency?

Main Memory = 32 GB = $2^{30} * 2^5 = 2^{35}$ = 35 bits Physical Address.

Block Size = 32 KB = 2^{15} = 15 bits block size.

So, the block number = 35 – 15 = 20 bits



Propagation Delay = 10 * 20 = 200 ns.

Hit latency: 200 ns + 10 ns (Propagation Delay of OR gate) = 210 ns.

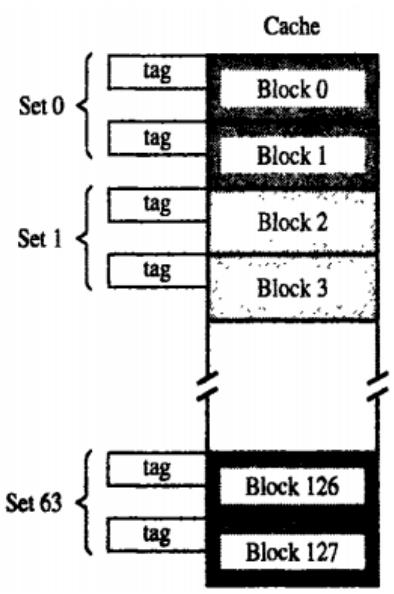
Disadvantage

- Hardware cost is high as compared to direct mapping.
- Tag directory size is more as compared to direct mapping.

Q If the associativity of a processor cache is doubled while keeping the capacity and block size unchanged, which one of the following is guaranteed to be NOT affected? **(Gate-2014) (2 Marks)**
(A) Width of tag comparator
(B) Width of set index decoder
(C) Width of way selection multiplexor
(D) Width of processor to main memory data bus
Answer: (D)

Set Associative Mapping

- In k-way set associative mapping, Blocks of cache / cache lines are grouped into sets where each set contains “k” number of lines.
- A particular block of main memory can map to only one particular set of the cache.
- However, within that set, the memory block can map to any freely available cache line.



Physical Address for the K-way set associative memory is divided as:

TAG	SET NUMBER	BLOCK OFFSET

Formulas:

- **Number of lines in cache = Cache size / Block Size.**
- **Number of Sets = Line Number / Set Size in terms of number of cache lines**

Main Memory				
B-0	W-0	W-1	W-2	W-3
B-1	W-4	W-5	W-6	W-7
B-2	W-8	W-9	W-10	W-11
B-3	W-12	W-13	W-14	W-15
B-4	W-16	W-17	W-18	W-19
B-5	W-20	W-21	W-22	W-23
B-6	W-24	W-25	W-26	W-27
B-7	W-28	W-29	W-30	W-31
B-8	W-32	W-33	W-34	W-35
B-9	W-36	W-37	W-38	W-39
B-10	W-40	W-41	W-42	W-43
B-11	W-44	W-45	W-46	W-47
B-12	W-48	W-49	W-50	W-51
B-13	W-52	W-53	W-54	W-55
B-14	W-56	W-57	W-58	W-59
B-15	W-60	W-61	W-62	W-63

Set Number-0	CL-0	TAG = Block no – CL	W-
			W-
			W-
			W-
	CL-1	TAG = Block no – CL	W-
			W
			W-
			W-
Set Number-1	CL-2	TAG = Block no – CL	W-
			W-
			W-
			W-
	CL-3	TAG = Block no - CL	W-
			W-
			W-

			W-

	Cache									
Set Number-0	CL-0	Tag	B-0	W-0	B-2	W-8	B-4	W-16	B-6	W-24
				W-1		W-9		W-17		W-25
				W-2		W-10		W-18		W-26
				W-3		W-11		W-19		W-27
	CL-1	Tag	B-8	W-32	B-10	W-40	B-12	W-48	B-14	W-56
				W-33		W-41		W-49		W-57
				W-34		W-42		W-50		W-58
				W-35		W-43		W-51		W-59
Set Number-1	CL-2	Tag	B-1	W-4	B-3	W-12	B-5	W-20	B-7	W-28
				W-5		W-13		W-21		W-29
				W-6		W-14		W-22		W-30
				W-7		W-15		W-23		W-31
	CL-3	Tag	B-9	W-36	B-11	W-44	B-13	W-52	B-15	W-60
				W-37		W-45		W-53		W-61
				W-38		W-46		W-54		W-62
				W-39		W-47		W-55		W-63

Example

Q Consider the main memory size is of 128 KB, the cache size is of 16 KB, the block size is of 256 B, the set size is 2. Find Tag.

Main Memory = 128 KB = $2^{10} * 2^7$ = 17 bits (Physical Address).

Cache Size = 16 KB

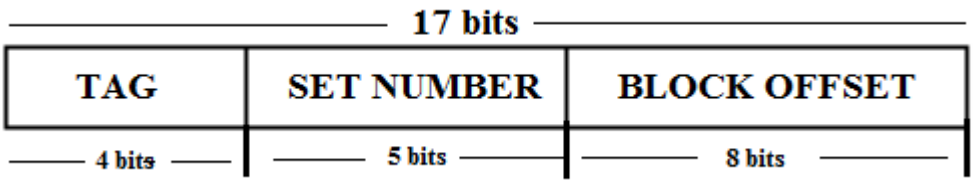
Block Size = 256 B = 2^8 = 8 Bits (Block Offset).

Number of lines in cache = Cache size / Block Size = 16 KB / 256 B = 2^6 lines.

Number of sets = $2^6 / 2 = 2^5$ = 5 bits (Set Number).

So, tag size = 17 – (8 +5) = 4 bits

So, the physical address division:



Tag Directory Size = Cache Lines * Tag = $2^6 * 4$

Number of Comparators = 2 (as it is two way set associative)

Example:

Consider: Main Memory = 32 GB, Cache Size = 32 KB, Block Size = 1 KB and it is a 4-way set associative cache?

Main Memory = 32 GB = $2^5 * 2^{30}$ = 35 bits (Physical Address)

Cache Size = 32 KB.

Block Size = 1 KB = 2^{10} = 10 bits (Block Offset)

Number of lines in cache = Cache Size / Block Size = 32 KB / 1 KB = 2^5 lines.

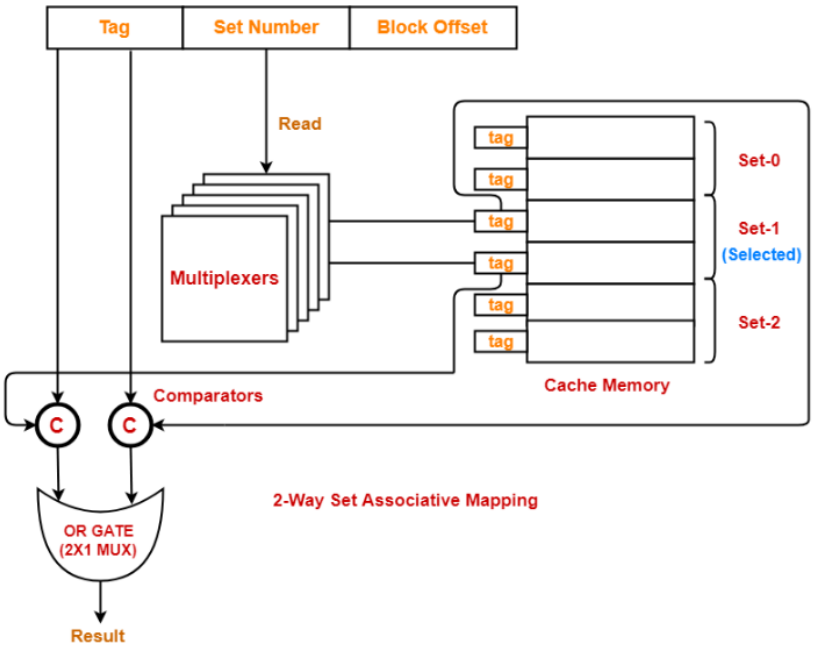
Set Number = Total lines in Cache / 4 (as 4-way set associative) = $2^5 / 2^2 = 2^3$ = 3 bits.

Tag = 35 – (10 + 3) = 22 bits

Tag directory = $2^5 * 22$

Number of comparators = 4 (as 4 way set associative)

Hardware Architecture



- The number of comparators depends upon the number of lines in a set.

Q consider a system where main memory is of size 4 GB, Cache memory is of size 4 MB and block size is 1 KB, find mapping for?

a) Direct mapping

b) Associative mapping

c) Set Associative Mapping (4-way set associative)

	TAG	Comparator	TAG Dir	Cache Line
Direct	10	1	$10 * 2^{12}$	2^{12}

Associative	22	2^{12}	$22 * 2^{12}$	2^{12}
4- Way	12	4	$12 * 2^{12}$	2^{12}

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Set Associative
128 KB	16 KB	256 B			2-way
32 GB	32 KB	1 KB			4-way
	512 KB	1 KB	7		8-way
16 GB		4 KB	10		4-way
64MB			10		4-way
	512 KB		7		8-way

MM Size	Cache Size	Block Size	No of bits in Tag	Tag Directory Size	Set Associative
128 KB	16 KB	256 B	4	$4 * 2^6$	2-way
32 GB	32 KB	1 KB	22	$22 * 32$	4-way
2^{23} B	512 KB	1 KB	7	$7 * 2^9$	8-way
16 GB	2^{26} B	4 KB	10	$10 * 2^{14}$	4-way
64MB	?	?	10	?	4-way
?	512 KB	?	7	?	8-way

Q The size of the physical address space of a processor is 2^P bytes. The word length is 2^W bytes. The capacity of cache memory is 2^N bytes. The size of each cache block is 2^M words. For a K-way set-associative cache memory, the length (in number of bits) of the tag field is **(Gate-2018) (2 Marks)**

(A) $P - N - \log_2 K$
(B) $P - N + \log_2 K$
(C) $P - N - M - W - \log_2 K$
(D) $P - N - M - W + \log_2 K$

Answer: (B)

Q A cache memory unit with capacity of N words and block size of B words is to be designed. If it is designed as direct mapped cache, the length of the TAG field is 10 bits. If the cache unit is now designed as a 16-way set-associative cache, the length of the TAG field is _____ bits. **(Gate-2017) (1 Marks)**

Answer: (14)

Q The width of the physical address on a machine is 40 bits. The width of the tag field in a 512 KB 8-way set associative cache is _____ bits **(Gate-2016) (2 Marks)**
Answer: (24)

Q A 4-way set-associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____ **(Gate-2014) (1 Marks)**
Answer: (20)

Q In a k-way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in sequence one after another. The lines in set s are sequenced before the lines in set (s+1). The main memory blocks are numbered 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from. **(Gate-2013) (1 Marks)**
(A) (j mod v) * k to (j mod v) * k + (k-1)
(B) (j mod v) to (j mod v) + (k-1)
(C) (j mod k) to (j mod k) + (v-1)
(D) (j mod k) * v to (j mod k) * v + (v-1)
Answer: (A)

Q A computer has a 256 KByte, 4-way set associative, write back data cache with block size of 32 Bytes. The processor sends 32-bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 1 replacement bit. The number of bits in the tag field of an address is **(Gate-2012) (2 Marks)**
(A) 11 **(B)** 14 **(C)** 16 **(D)** 27
Answer: (C)

Q The size of the cache tag directory is **(Gate-2012) (2 Marks)**
a) 160 Kbits **b)** 136 Kbits **c)** 40 Kbits **d)** 32 Kbits
Ans: a

Q Consider a computer with a 4-ways set-associative mapped cache of the following characteristics: a total of 1 MB of main memory, a word size of 1 byte, a block size of 128 words and a cache size of 8 KB. The number of bits in the TAG, SET and WORD fields, respectively are: **(Gate-2008) (2 Marks)**
(A) 7, 6, 7 **(B)** 8, 5, 7 **(C)** 8, 6, 6 **(D)** 9, 4, 7
Answer: (D)

Q Consider a computer with a 4-ways set-associative mapped cache of the following characteristics: a total of 1 MB of main memory, a word size of 1 byte, a block size of 128 words and a cache size of 8 KB. While accessing the memory location 0C795H by the CPU, the contents of the TAG field of the corresponding cache line is: **(Gate-2008) (2 Marks)**

a) 000011000 **b)** 110001111 **c)** 00011000 **d)** 110010101

Ans: a

Q Consider a 4-way set associative cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of a word in main memory. The number of bits in the TAG, LINE and WORD fields are respectively: **(Gate-2007) (1 Marks)**

(A) 9,6,5 **(B)** 7, 7, 6 **(C)** 7, 5, 8 **(D)** 9, 5, 6

Answer: (D)

Q A computer system has a level-1 instruction cache (1-cache), a level-1 data cache (D-cache) and a level-2 cache (L2-cache) with the following specifications:

	Capacity	Mapping Method	Block size
I-cache	4K words	Direct mapping	4 Words
D-cache	4K words	2-way set associative mapping	4 Words
L2-cache	64K words	4-way set associative mapping	16 Words

The length of the physical address of a word in the main memory is 30 bits. The capacity of the tag memory in the I-cache, D-cache and L2-cache is, respectively, **(Gate-2006) (2 Marks)**

(A) 1 K x 18-bit, 1 K x 19-bit, 4 K x 16-bit

(B) 1 K x 16-bit, 1 K x 19-bit, 4 K x 18-bit

(C) 1 K x 16-bit, 512 x 18-bit, 1 K x 16-bit

(D) 1 K x 18-bit, 512 x 18-bit, 1 K x 18-bit

Answer: (A)

Q The main memory of a computer has 2 cm blocks while the cache has 2c blocks. If the cache uses the set associative mapping scheme with 2 blocks per set, then the block k of main memory maps to the set: **(Gate-1999) (1 Marks)**

(A) (k mod m)of the cache
(B) (k mod c)of the cache
(C) (k mod 2c)of the cache
(D) (k mod 2cm)of the cache

Answer: (B)

Q A block-set associative cache memory consists of 128 blocks divided into four block sets. The main memory consists of 16,384 blocks and each block contains 256 eight-bit words. **(Gate-1990) (2 Marks)**

i) How many bits are required for addressing the main memory?
ii) How many bits are needed to represent the TAG, SET and WORD fields

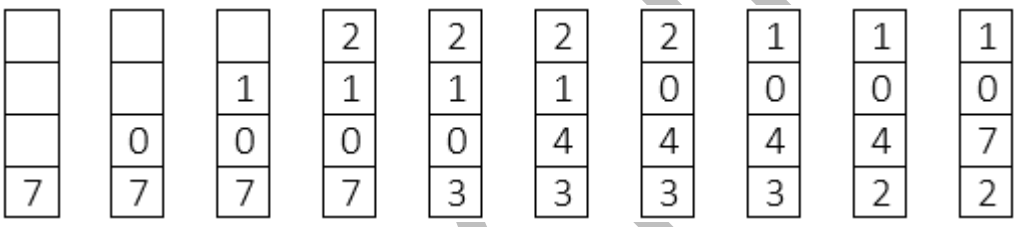
Cache Replacement Policies

- In direct mapped cache, the position of each block is predetermined hence no replacement policy exists.
- In fully associative and set associative caches there exists policies.
- When a new block is brought into the cache and all the positions that it may occupy are full, then the controller needs to decide which of the old blocks it can overwrite.

FIFO Policy

- The page which have entered first in the memory will be replaced first.
- This can lead to a problem known as "**Belady's Anomaly**", it states that if we increase the number of lines in cache memory the cache miss will increase.

Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.



Total hits = 10

Hit Rate = (10/20) * 100 = 50%

Total Miss = 10

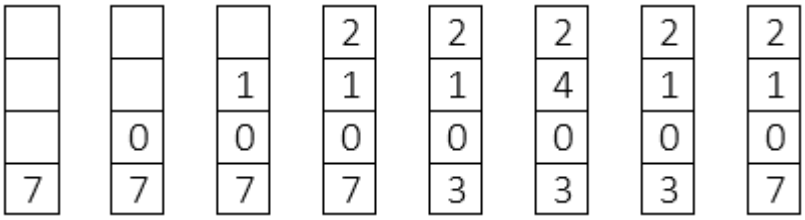
Number of cache replacement = total miss – number of lines available

= 10 – 4 = 6

LRU (Least Recently Used)

- The page which was not used for the longest period of time in the past will get replaced first.

Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.



Number of Misses = 8

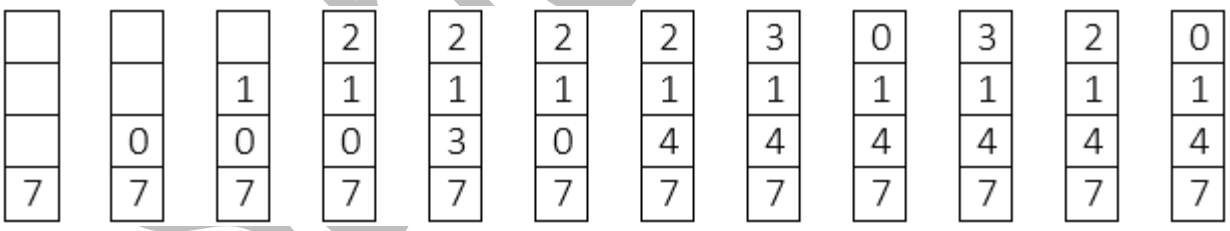
Hit Rate = 12 = 12/20 * 100 = 60%

Number of Cache Replacement = 8 – 4 = 4

Most Recently Used (MRU)

- The page which was used recently will be replaced first.

Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.



Number of Misses = 12

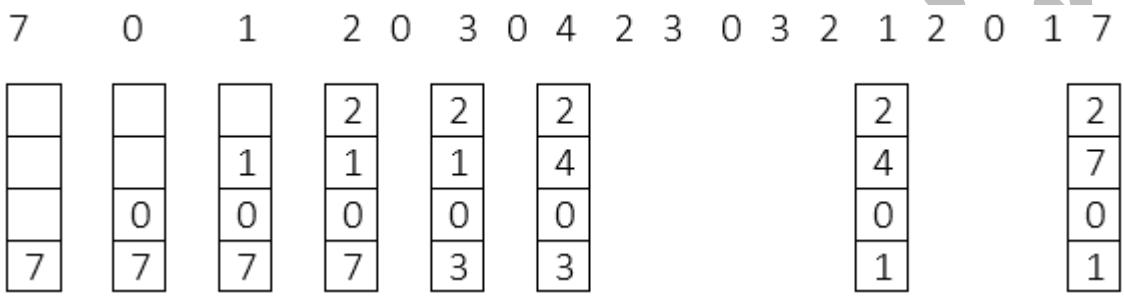
Number of Hit = 8, Hit rate = 8/20 * 100 = 40%

Number of cache replacement = 8

Optimal Algorithm

- The page which will not be used for the longest period of time in future references will be replaced first.
- The optimal algorithm will provide the best performance but it is difficult to implement as it requires the future knowledge of pages which is not possible.
- It is used as a benchmark for cache replacement algorithms.

Example: Let the blocks be in the sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 and the cache memory has 4 lines.



So, the total misses = 8

Total Hits = 12

Hit Ratio = 12/20 * 100 = 60%.

Number of cache replacements = 8 – 4 = 4

Types of Miss

- Compulsory Miss
- Capacity Miss
- Conflict Miss

Compulsory Miss

- When CPU demands for any block for the first time then definitely a miss is going to occur as the block needs to be brought into the cache, it is known as Compulsory miss.

Capacity Miss

- Occur because blocks are being discarded from cache because cache cannot contain all blocks needed for program execution.

Conflict Miss

- In the case of set associative or direct mapped block placement strategies, conflict misses occur when several blocks are mapped to the same set or block frame; also called collision misses or interference misses.

Q Consider a 2-way set associative cache with 256 blocks and uses LRU replacement, Initially the cache is empty. Conflict misses are those misses which occur due the contention of multiple blocks for the same cache set. Compulsory misses occur due to first time access to the block. The following sequence of accesses to memory blocks (0, 128, 256, 128, 0, 128, 256, 128, 1, 129, 257, 129, 1, 129, 257, 129) is repeated 10 times. The number of conflict misses experienced by the cache is _____. **(Gate-2017) (2 Marks)**
Ans: 76

Q Consider a 4-way set associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order: 0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155. Which one of the following memory blocks will NOT be in cache if LRU replacement policy is used? **(Gate-2009) (2 Marks)**
(A) 3 **(B)** 8 **(C)** 129 **(D)** 216
Answer: (D)

Q Consider a Direct Mapped Cache with 8 cache blocks (numbered 0-7). If the memory block requests are in the following order 3, 5, 2, 8, 0, 63, 9,16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82,17, 24. Which of the following memory blocks will not be in the cache at the end of the sequence? **(GATE-2007) (2 Marks)**

(A) 3 **(B)** 18 **(C)** 20 **(D)** 30

Answer: (B)

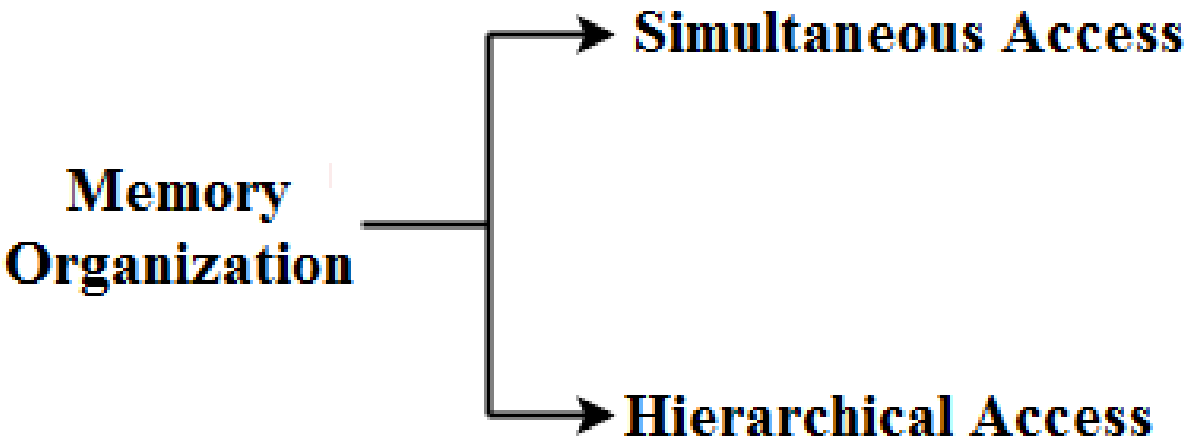
Q Consider a small two-way set-associative cache memory, consisting of four blocks. For choosing the block to be replaced, use the least recently used (LRU) scheme. The number of cache misses for the following sequence of block addresses is 8, 12, 0, 12, 8 **(Gate-2004) (2 Marks)**

(A) 2 **(B)** 3 **(C)** 4 **(D)** 5

Answer: (C)

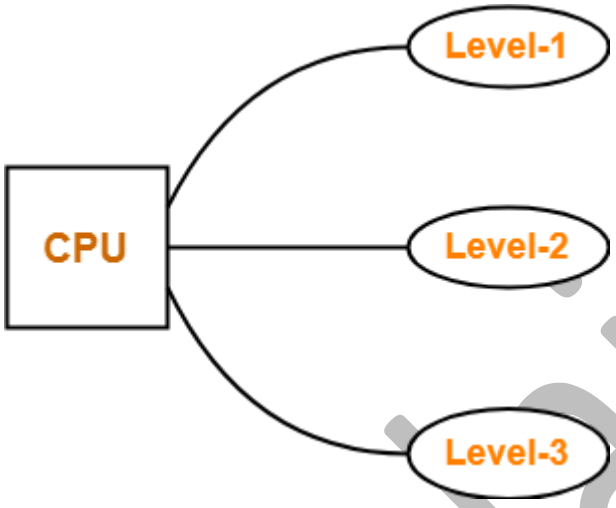
Memory Organization

- Memory is organized at different levels.
- CPU may try to access different levels of memory in different ways.
- On this basis, the memory organization is broadly divided into two types-



Simultaneous Access Memory Organization

- In simultaneous access all the levels of memory are directly connected to the CPU, whenever CPU requires any word, it starts searching for it in all the levels simultaneously.



• T_1 = Access time of level L_1 • S_1 = Size of level L_1 • C_1 = Cost per byte of level L_1 • H_1 = Hit rate of level L_1	• T_2 = Access time of level L_2 • S_2 = Size of level L_2 • C_2 = Cost per byte of level L_2 • H_2 = Hit rate of level L_2	• T_3 = Access time of level L_3 • S_3 = Size of level L_3 • C_3 = Cost per byte of level L_3 • H_3 = Hit rate of level L_3

Effective Memory Access Time

Average time required to access memory per operation =

$H_1 * T_1 + (1 - H_1) * H_2 * (T_1 + T_2) + (1 - H_1) (1 – H_2) * H_3 * (T_1 + T_2 + T_3)$

$H_1 * T_1 + (1 - H_1) * H_2 * (T_1 + T_2) + (1 - H_1) (1 – H_2) * (T_1 + T_2 + T_3)$

Average Cost per Byte

- Average cost per byte of the memory =

$$\frac{C_1S_1 + C_2S_2 + C_3S_3}{S_1 + S_2 + S_3}$$

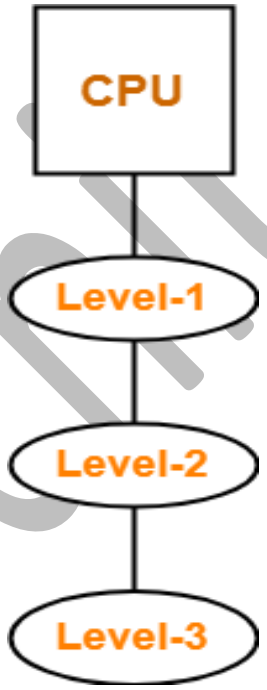
Hierarchical Access Memory Organization

In this memory organization, memory levels are organized as

- Level-1 is directly connected to the CPU.
- Level-2 is directly connected to level-1.
- Level-3 is directly connected to level-2 and so on.

Whenever CPU requires any word,

- It first searches for the word in level-1.
- If the required word is not found in level-1, it searches for the word in level-2.
- If the required word is not found in level-2, it searches for the word in level-3 and so on.



• T_1 = Access time of level L_1 • S_1 = Size of level L_1 • C_1 = Cost per byte of level L_1 • H_1 = Hit rate of level L_1	• T_2 = Access time of level L_2 • S_2 = Size of level L_2 • C_2 = Cost per byte of level L_2 • H_2 = Hit rate of level L_2	• T_3 = Access time of level L_3 • S_3 = Size of level L_3 • C_3 = Cost per byte of level L_3 • H_3 = Hit rate of level L_3

Effective Memory Access Time

Average time required to access memory per operation =

$H_1 * T_1 + (1 - H_1) * H_2 * (T_1 + T_2) + (1 - H_1) (1 – H_2) * H_3 * (T_1 + T_2 + T_3)$

$H_1 * T_1 + (1 - H_1) * H_2 * (T_1 + T_2) + (1 - H_1) (1 – H_2) * (T_1 + T_2 + T_3)$

Average Cost per Byte

- Average cost per byte of the memory =

$$\frac{C_1S_1 + C_2S_2 + C_3S_3}{S_1 + S_2 + S_3}$$

Example: Calculate the EMAT for a machine with a cache hit rate of 80% where cache access time is 5ns and main memory access time is 100ns, both for simultaneous and hierarchical access.

Q Assume that for a certain processor, a read request takes 50 nanoseconds on a cache miss and 5 nanoseconds on a cache hit. Suppose while running a program, it was observed that 80% of the processor's read requests result in a cache hit. The average read access time in nanoseconds is_____. **(GATE-2015) (2 Marks)**
Answer: 14

Q Consider a system with 2 level cache. Access times of Level 1 cache, Level 2 cache and main memory are 0.5 ns, 5 ns and 100 ns respectively. The hit rates of Level 1 and Level 2 caches are 0.7 and 0.8 respectively. What is the average access time of the system ignoring the search time within the cache? **(NET-DEC-2018)**
a) 35.20 ns **b)** 7.55 ns **c)** 20.75 ns **d)** 24.35 ns
Ans: b

Q For inclusion to hold between two cache levels L_1 and L_2 in a multi-level cache hierarchy, which of the following are necessary? **(Gate-2008) (2 Marks)**

I. L_1 must be a write-through cache

II. L_2 must be a write-through cache

III. The associativity of L_2 must be greater than that of L1

IV. The L_2 cache must be at least as large as the L1 cache

(A) IV only
(B) I and IV only
(C) I, III and IV only
(D) I, II, III and IV
Answer: (A)

Cache Coherence Problem

- If multiple copy of same data is maintained at different level of memories then inconsistency may occur, this problem is known as cache coherence problem.

Methods to resolve cache coherence problem

Cache coherence problem can be resolved using the following four techniques:

- Write Through
- Write Back

1. **Write Through**
    - Write through is used to maintain the consistency between the cache and main memory.
    - According to it if the cache copy is updated, at the same time main memory is also updated.

Advantages

- It provides the highest level of consistency.

Disadvantages

- It requires more number of memory access.

2. **Write Back**
    - Write back is also used to maintain the consistency between the cache and main memory.
    - According to it all the changes performed on cache are reflected back to the main memory in the end.

Advantage

- Less number of memory accesses and less write operations.

Disadvantage

- Inconsistency may occur.

Q In _____ method, the word is written to the block in both the cache and main memory, in parallel. **(NET-JULY-2016)**

(a) Write through **(b)** Write back

(c) Write protected **(d)** Direct mapping
Ans: a

Q A hierarchical memory system that uses cache memory has cache access time of 50 nano seconds, main memory access time of 300 nano seconds, 75% of memory requests are for read, hit ratio of 0.8 for read access and the write-through scheme is used. What will be the average access time of the system both for read and write requests? **(NET-DEC-2014)**
(A) 157.5 n.sec **(B)** 110 n.sec **(C)** 75 n.sec **(D)** 82.5 n.sec
Ans: a

Q In a two-level cache system, the access times of L_1 and L_2 is 1 and 8 clock cycles, respectively. The miss penalty from the L_2 cache to main memory is 18 clock cycles. The miss rate of L_1 cache is twice that of L_2. The average memory access time (AMAT) of this cache system is 2 cycles. The miss rates of L_1 and L_2 respectively are: **(Gate-2017) (2 Marks)**
(A) 0.111 and 0.056 **(B)** 0.056 and 0.111
(C) 0.0892 and 0.1784 **(D)** 0.1784 and 0.0892
Answer: (A)

Q The read access times and the hit ratios for different caches in a memory hierarchy are as given below:

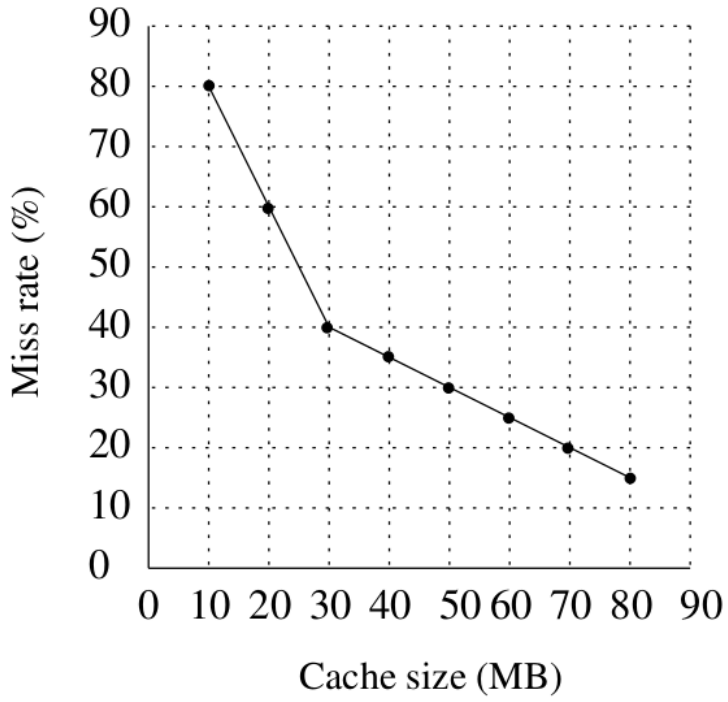
Cache	Read access time (in nanoseconds)	Hit ratio
I-cache	2	0.8
D-cache	2	0.9
L2-cache	8	0.9

The read access time of main memory in 90 nanoseconds. Assume that the caches use the referred-word-first read policy and the writeback policy. Assume that all the caches are direct mapped caches. Assume that the dirty bit is always 0 for all the blocks in the caches. In execution of a program, 60% od memory reads are for instruction fetch and 40% are for memory operand fetch. The average read access time in nanoseconds (up to 2 decimal places) is _____ **(Gate-2017) (2 Marks)**
Ans: 4.72

Q Consider a two-level cache hierarchy L1 and L2 caches. An application incurs 1.4 memory accesses per instruction on average. For this application, the miss rate of L1 cache 0.1, the

L2 cache experience on average. 7 misses per 1000 instructions. The miss rate of L2 expressed correct to two decimal places is _____. **(Gate-2017) (1 Marks)**
Answer: (0.05)

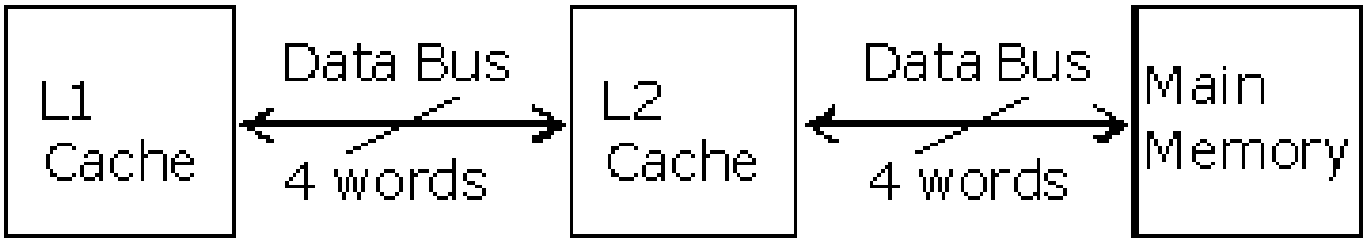
Q A file system uses an in-memory cache to cache disk blocks. The miss rate of the cache is shown in the figure. The latency to read a block from the cache is 1 ms and to read a block from the disk is 10 ms. Assume that the cost of checking whether a block exists in the cache is negligible. Available cache sizes are in multiples of 10 MB.



The smallest cache size required to ensure an average read latency of less than 6 ms is _____ MB. **(Gate-2016) (2 Marks)**
Answer: (30)

Q The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations, 60 memory operand read operations and 40 memory operands write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is _____. **(Gate-2014) (2 Marks)**
Answer: (1.68)

Q A computer system has an L1 cache, an L2 cache, and a main memory unit connected as shown below. The block size in L1 cache is 4 words. The block size in L2 cache is 16 words. The memory access times are 2 nanoseconds. 20 nanoseconds and 200 nanoseconds for L1 cache, L2 cache and main memory unit respectively.



When there is a miss in L1 cache and a hit in L2 cache, a block is transferred from L2 cache to L1 cache. What is the time taken for this transfer? **(Gate-2010) (2 Marks)**

(A) 2 nanoseconds
(B) 20 nanoseconds
(C) 22 nanoseconds
(D) 88 nanoseconds

Answer: (C)

Q When there is a miss in both L_1 cache and L_2 cache, first a block is transferred from main memory to L_2 cache, and then a block is transferred from L_2 cache to L_1 cache. What is the total time taken for these transfers? **(Gate-2010) (2 Marks)**

(A) 222 nanoseconds
(B) 888 nanoseconds
(C) 902 nanoseconds
(D) 968 nanoseconds

Answer: (C)

Q Consider a system with 2 level caches. Access times of Level 1 cache, Level 2 cache and main memory are 1 ns, 10ns, and 500 ns, respectively. The hit rates of Level 1 and Level 2 caches are 0.8 and 0.9, respectively. What is the average access time of the system ignoring the search time within the cache? **(Gate-2004) (1 Marks)**

(A) 13.0 ns **(B)** 12.8 ns **(C)** 12.6 ns **(D)** 12.4 ns

Answer: (C)

Q A dynamic RAM has a memory cycle time of 64 nsec. It has to be refreshed 100 times per msec and each refresh takes 100 nsec. What percentage of the memory cycle time is used for refreshing? **(Gate-2005) (1 Marks)**

(A) 10 **(B)** 6.4 **(C)** 1 **(D)** .64

Answer: (C)

Q A cache line is 64 bytes. The main memory has latency 32ns and bandwidth 1G.Bytes/s. The time required to fetch the entire cache line from the main memory is **(Gate-2006) (2 Marks)**

(A) 32 ns **(B)** 64 ns **(C)** 96 ns **(D)** 128 ns

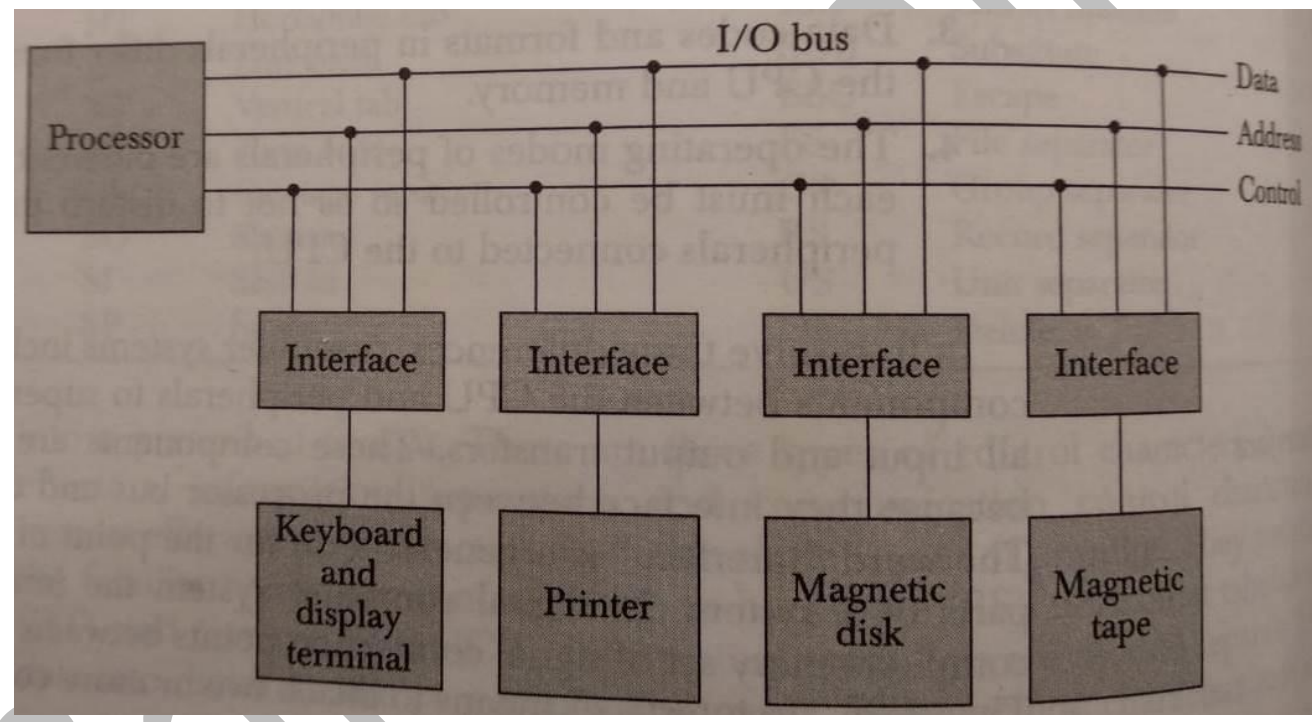
Answer: (C)

Input / Output management

- In general, when we say a computer, we understand a CPU + Memory (cache, main memory)
- But a computer does not serve any purpose if it cannot receive data from the outside world or cannot transmit the data to outside world.
- I/o or peripheral devices are those independent devices which serve this purpose.
- So, while designing i/o for a computer we must know the number of i/o device and the capacity of each device.

Interface

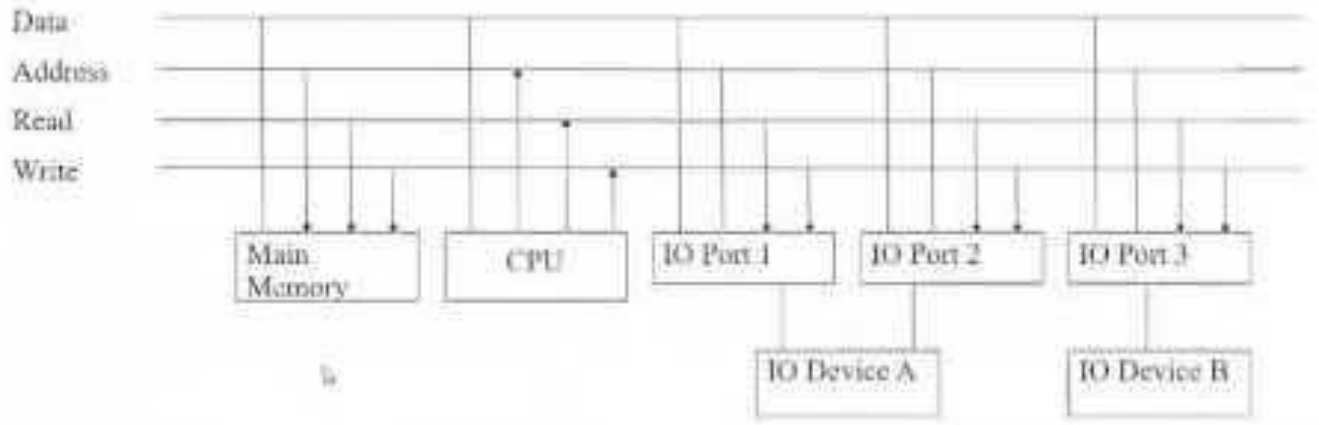
- But we cannot directly connect a i/o device to computer because of the following reasons
    - **Speed: -** The speed of CPU and i/o device will usually different.
    - **Format: -** The data code and format of CPU and peripherals may be different. E.g. ASCII, Unicode etc.
    - **Physical orientation: -** Different device have organizations like optical, magnetic, electrochemical and different controlling functions.
    - **Signal conversion: -** peripherals are electromagnetic and electrochemical device and their manner of operation is different from the operations of CPU and memory, which are electronic device, signal conversion is required



- **Address bus: -** Is used to identify the correct i/o devices among the number of i/o device , so CPU put an address of a specific i/o device on the address line, all devices keep monitoring this address bus and decode it, and if it is a match then it activates control and data lines.
- **Control bus: -** After selecting a specific i/o device CPU sends a functional code on the control line. The selected device(interface) reads that functional code and execute it. E.g. i/o command, control command, status command etc.
- **Data bus: -** In the final step depending on the operation either CPU will put data on the data line and device will store it or device will put data on the data line and CPU will store it.

How a computer (CPU) deals with Memory and I/O devices

- **Memory Mapped i/o**



- Here there is no separate i/o instructions. The CPU can manipulate i/o data residing in the interface registers with the same instructions that are used to manipulate memory words.
- Here computer can use memory type instructions for i/o data.
- **Advantage: -** In typical computer there are more memory reference instruction than i/o instruction, but in memory mapped i/o all instructions that refer to memory are also available for i/o.
- **Disadvantage: -**Total address get divided, some range is occupied by i/o while some memory.

- **Isolated i/o: -**

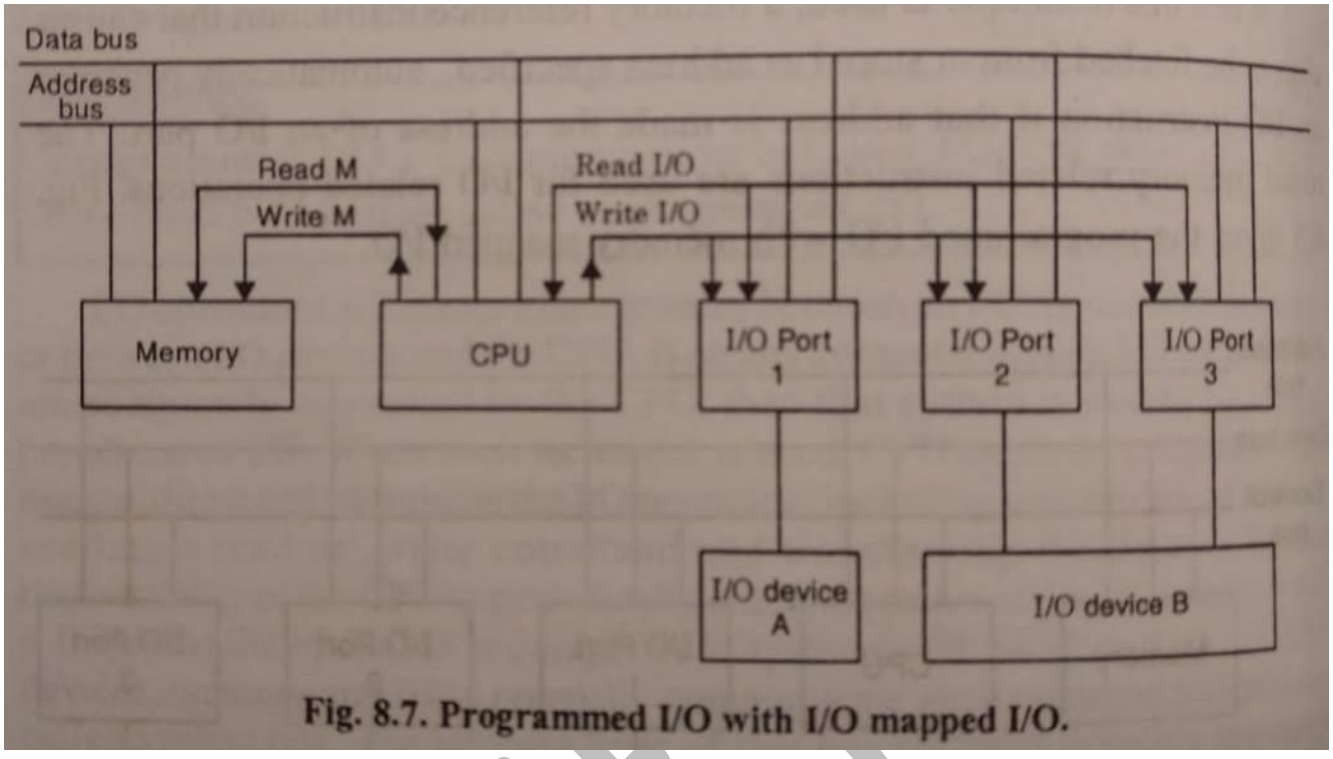
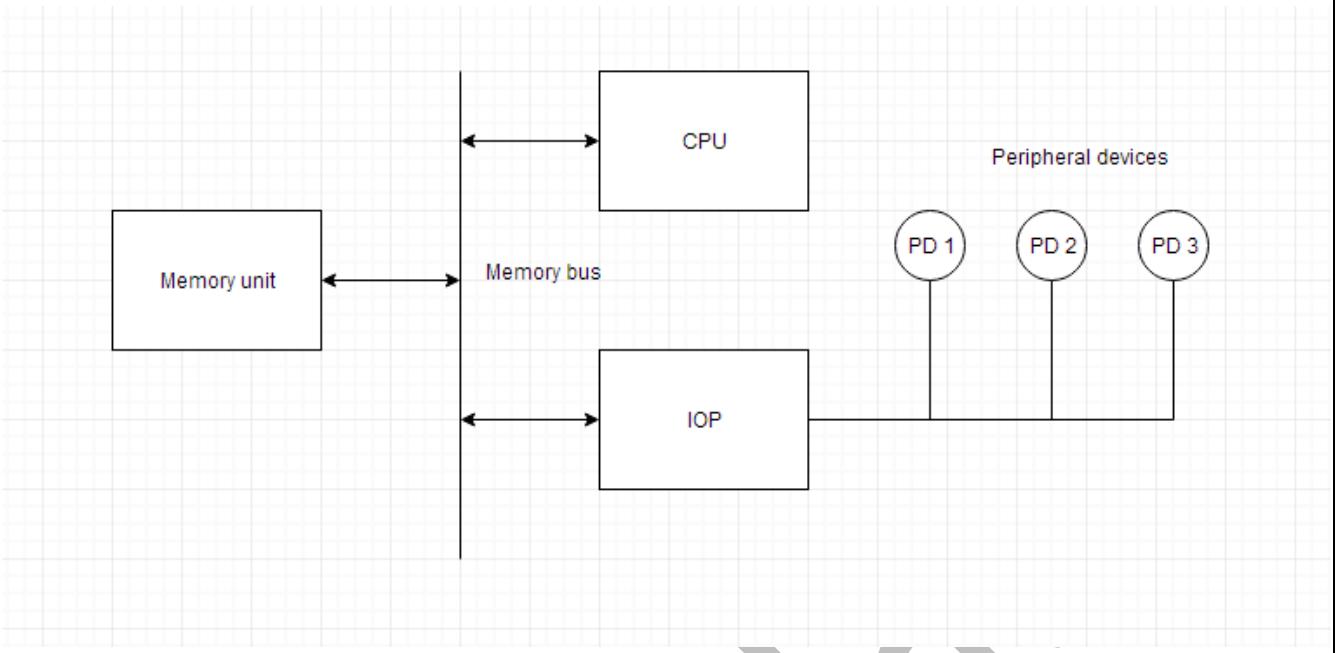


Fig. 8.7. Programmed I/O with I/O mapped I/O.

- Here common bus to transfer data between memory or i/o and CPU. The distinction between a memory and i/o transfer is made through separate read and write line.
- i/o read and i/o write control lines are enabled during an i/o transfer.
- Memory read and memory write control lines are enabled during a memory transfer.
- **Advantage: -** Here memory is used efficiently as the same address can be used two times.
- **Disadvantage: -** Need different control lines one for memory and other for i/o devices.

- **i/o Processor**



- Computer has independent sets of data, address and control buses, one for accessing memory and the other for i/o. this is done in computers that provide a separate i/o processor other than CPU.
- Memory communicate with both the CPU and iop through a memory bus.
- Iop communicates also with the input and output devices through a separate i/o bus with its own address, data and control lines. The purpose of iop is to provide an independent pathway for the transfer of information between external devices and internal memory.

Synchronous Vs Asynchronous data transfer

- Synchronization is achieved by a device called master generator, which generate a periodic train of clock pulse.
- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.
- When communication happens between devices which are under a same control unit or same clock then it is called synchronous communication. e.g. communication between CPU and its registers.

Asynchronous communication

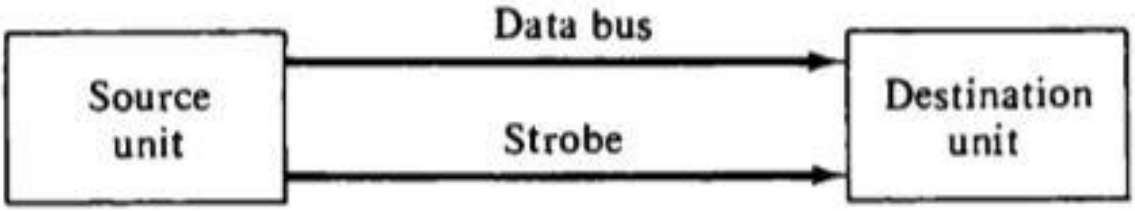
- When the timing units of two devices are independent that is they are under different control then it is called asynchronous communication.

Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by means of a *strobe* pulse supplied by one of the units to indicate to the other unit when the transfer has to occur. Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as *handshaking*.

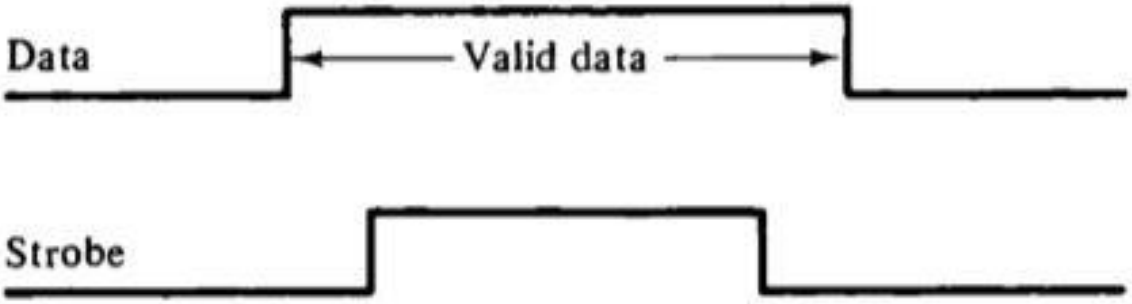
The strobe pulse method and the handshaking method of asynchronous data transfer are not restricted to I/O transfers. In fact, they are used extensively on numerous occasions requiring the transfer of data between two independent units. In the general case we consider the transmitting unit as the source and the receiving unit as the destination. For example, the CPU is the source unit during an output or a write transfer and it is the destination unit during an input or a read transfer. It is customary to specify the asynchronous transfer between two independent units by means of a timing diagram that shows the timing relationship that must exist between the control signals and the data in the buses. The sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination unit.

Strobe signal

- The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.



(a) Block diagram

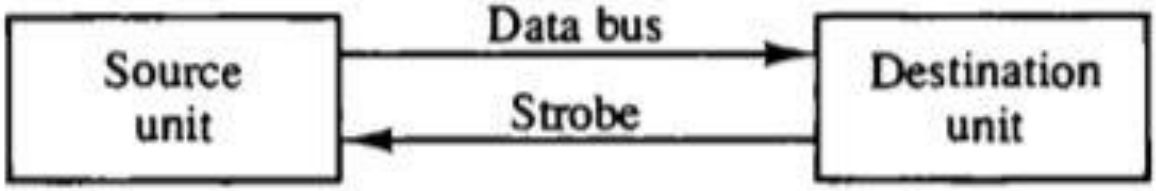


(b) Timing diagram

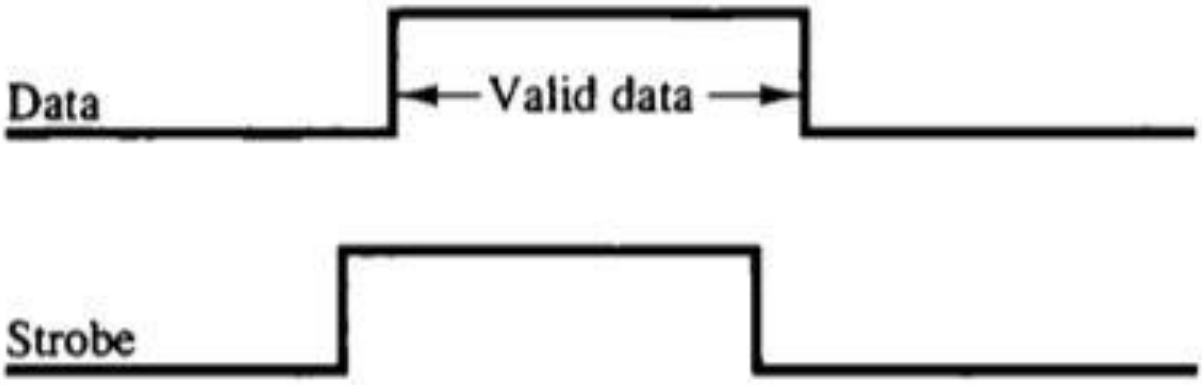
The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

As shown in the timing diagram of Fig. 11-3(b), the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse. The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data. Often, the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers. The source removes the data from the bus a brief period after it disables its strobe pulse. Actually, the source does not have to change the information in the data bus. The fact that the strobe signal is disabled indicates that the data bus does not contain valid data. New valid data will be available only after the strobe is enabled again.

Figure 11-4 shows a data transfer initiated by the destination unit. In this case the destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it. The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe. The source removes the data from the bus after a predetermined time interval.



(a) Block diagram

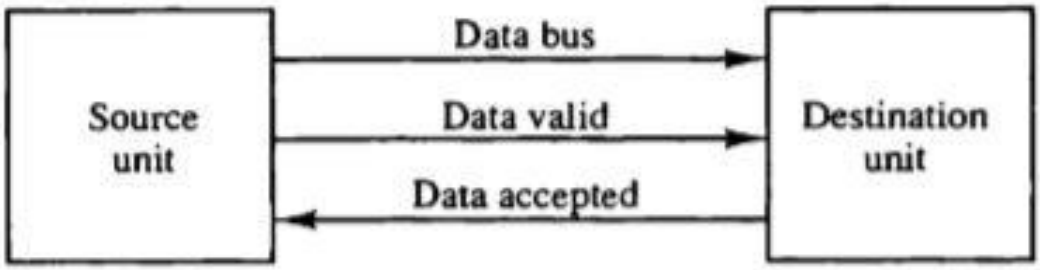


(b) Timing diagram

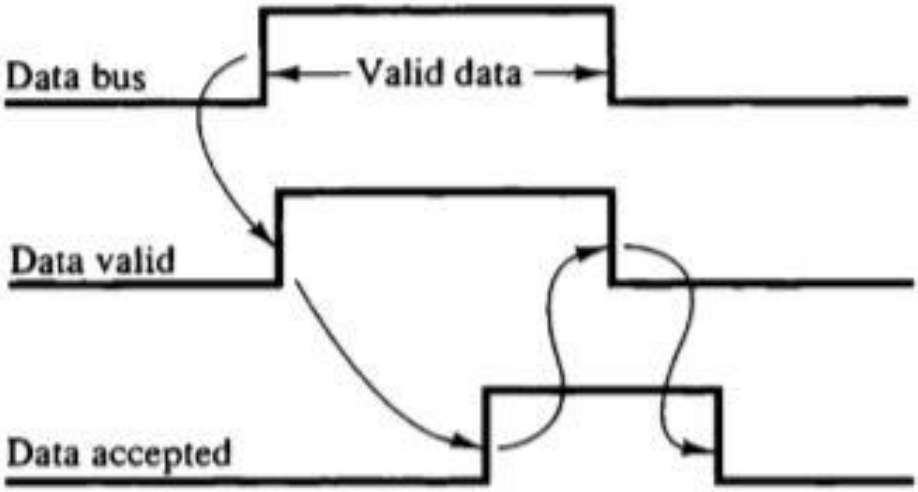
Handshaking

The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus. The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer. The basic principle of the two-wire handshaking method of data transfer is as follows. One control line is in the same direction as the data flow in the bus from the source to the destination. It is used by the source unit to inform the destination unit whether there are valid data in the bus. The other control line is in the other direction from the destination to the source. It is used by the destination unit to inform the source whether it can accept data. The sequence of control during the transfer depends on the unit that initiates the transfer.

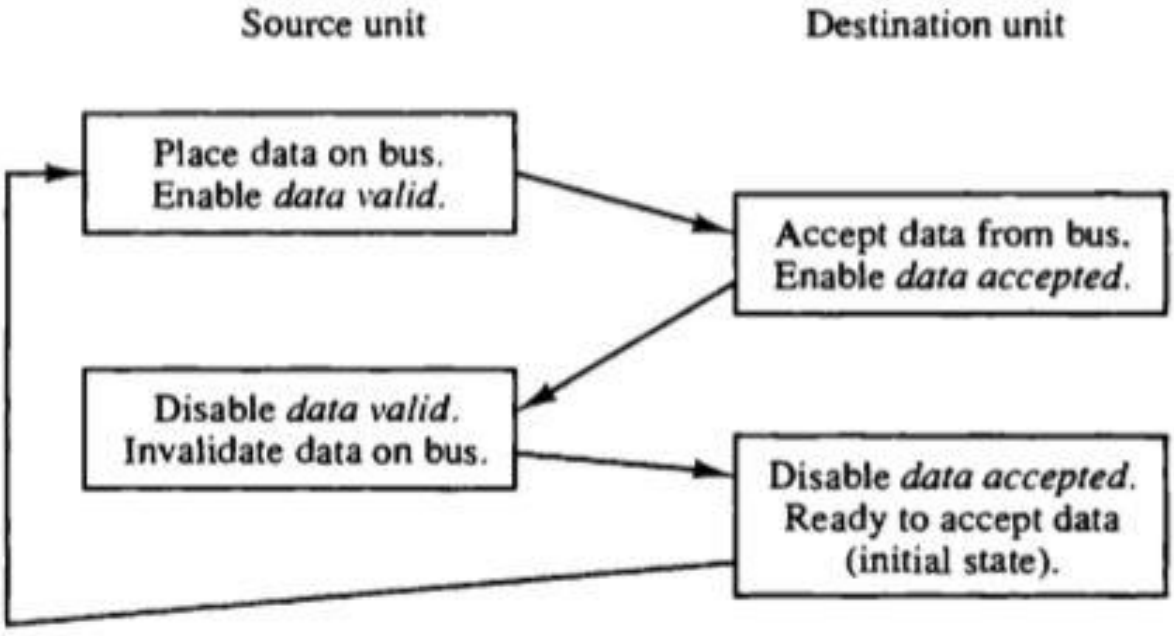
Source Initiated Transfer



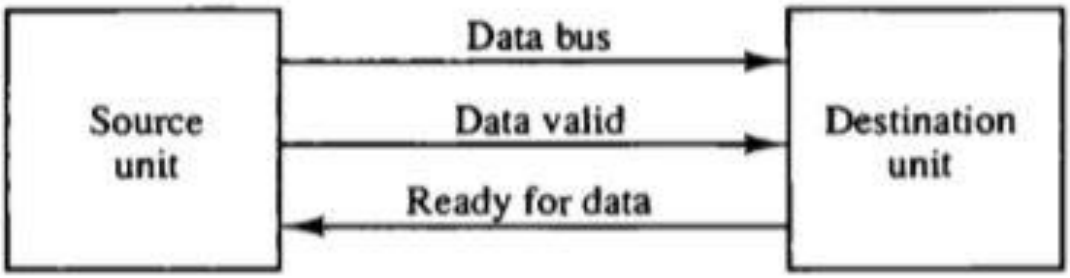
(a) Block diagram



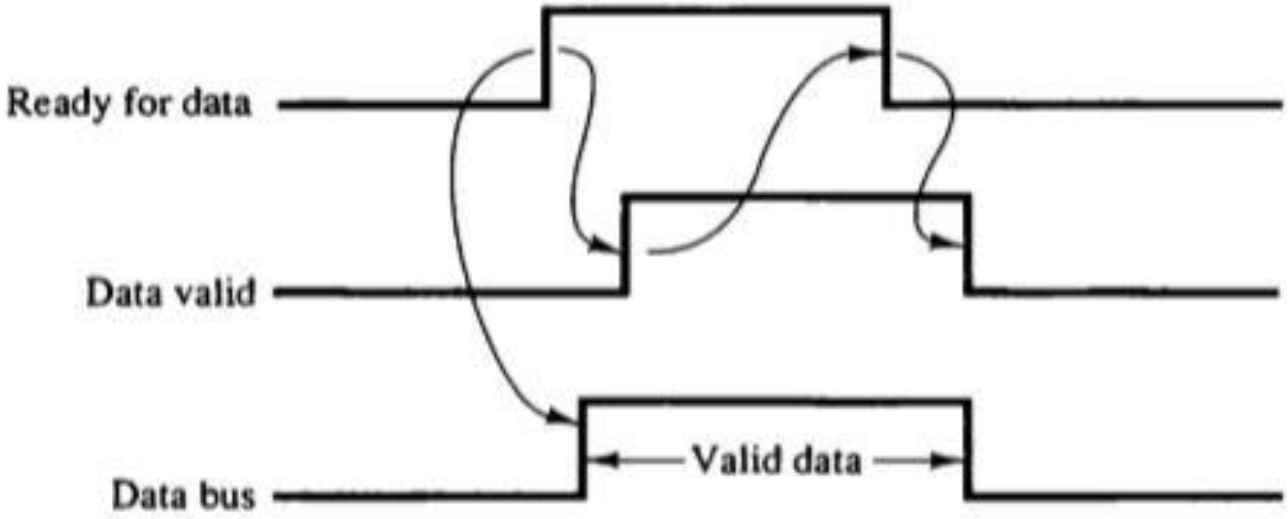
(b) Timing diagram



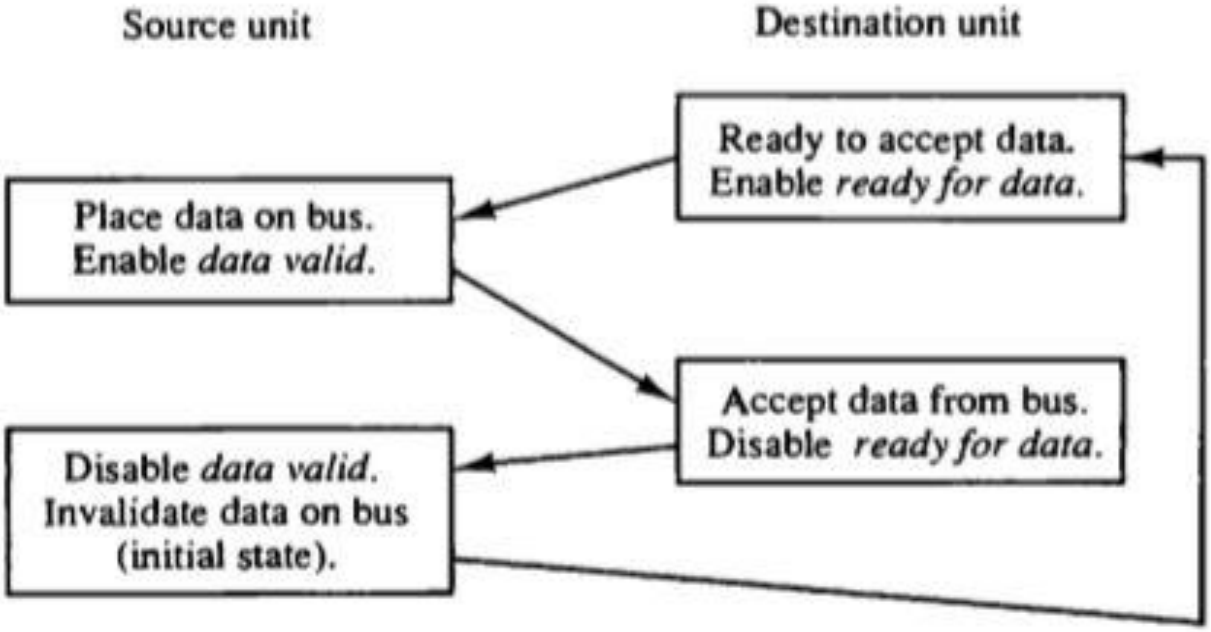
Destination initiated transfer



(a) Block diagram



(b) Timing diagram



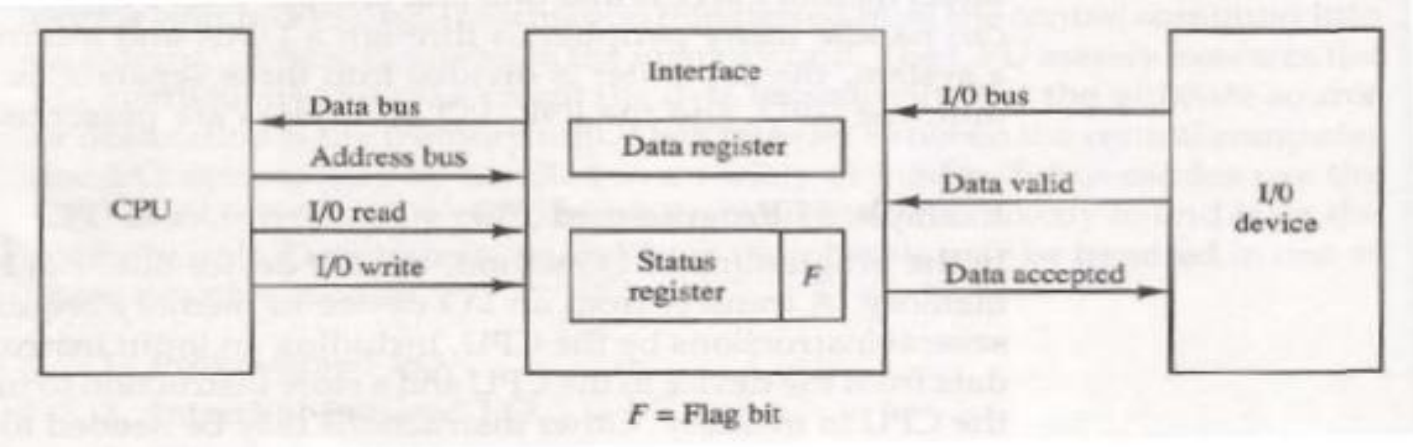
The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units. If one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a *timeout* mechanism, which produces an alarm if the data transfer is not completed within a predetermined time. The timeout is implemented by means of an internal clock that starts counting time when the unit enables one of its handshaking control signals. If the return handshake signal does not respond within a given time period, the unit assumes that an error has occurred. The timeout signal can be used to interrupt the processor and hence execute a service routine that takes appropriate error recovery action.

Modes of Data Transfer

- Here we deal with the problem that how data communication will take place CPU and i/o device.
- There are popularly three methods of data transfer: -
    - **Programmed i/o**
    - **Interrupt initiated i/o**
    - **Direct Memory Transfer**

Programmed I/O

- In this i/o device cannot access the memory directly. To complete data transfer number of instructions will be executed out of which input instruction are those which transfer data from device to CPU and store instructions from CPU to memory.



- **Step 1: -** i/o device will put data on the data bus and will enable valid data signal
- **Step 2: -** Interface is continuously sensing for data valid signal and when it receives the signal it will copy data from the data bus into it's data register and set its flag bit to 1 and enable data accepted line.
- **Step 3: -** CPU is continuously monitoring the status register in the programmed mode and as soon as it sees flag bit as 1, it immediately copies data from data register on the data bus and clear flag bit to zero.
- **Step 4: -** Now interface will disable data accepted line to tell i/o device, I am ready for new transitions.
- **Conclusion: -** CPU works in programmed mode or in busy wait mode so no of clock cycles are wasted. It will be difficult to handle multiple i/o device at the same time. It is not appropriate with the high-speed i/o devices.

Q The CPU of a system having 1 MIPS execution rate needs 4 machine cycles on an average for executing an instruction. The fifty percent of the cycles use memory bus. A memory read/write employs one machine cycle. For execution of the programs, the system utilizes 90% of the CPU time. For block data transfer, an I/O device is attached to the system while CPU executes the background programs continuously. What is the maximum I/O data transfer rate if programmed I/O data transfer technique is used? **(NET-JUNE-2015)**

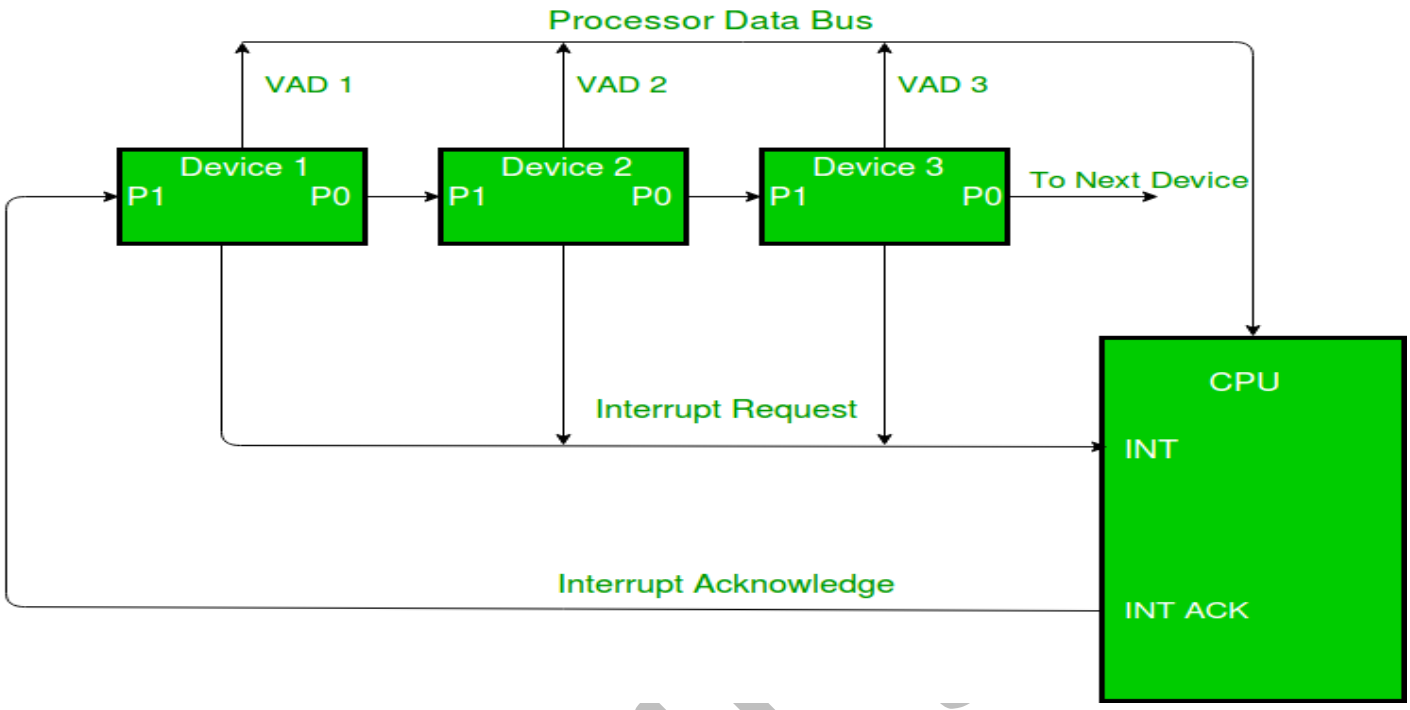
a) 500 Kbytes/sec
b) 2.2 Mbytes/sec
c) 125 Kbytes/sec
d) 250 Kbytes/sec

Ans: d

Interrupt initiated i/o

- **In** this method I/o device interrupt CPU when it is ready for data transfer.
- CPU keep executing instructions and after executing one instruction and before starting another instructions CPU wait and see if there is interrupt or not and if there is interrupt then takes a decision whether CPU should entertain this interrupt or continue with the execution.
- **Note: -** instruction is always absolute in nature and there is nothing like partial execution of instruction.
- The method of handling interrupts of different i/o devices are different, therefor every device has a program or routine called ISR (interrupt service routine) which tell CPU how the interrupt should be managed (and it saves CPU time).
- Interrupt can be of two types: -
    - **Non-vectored interrupt: -** Here there is a mutual understanding between CPU and device that where this routine is stored in memory (high priority device)
    - **Vectored interrupt: -** Some interrupts may be vectored where the interrupting device will also tell the address that where this routine is stored in the memory.
- It is possible that different i/o devices interrupt at the same time. Now CPU must have a priority decision that which interrupt should be service first and which should be service later.

- **H/w solution: -** It can be serial or parallel, serial solution is known as Daisy Channing is a h/w solution which is used to decide priority among different i/o devices (VAD- vector address device)



- Out of all possible devices 1 or more device may send an interrupt with a common line.
- When CPU completes 1 instruction and check interrupt in line and found an interrupt, then CPU will enable interrupt acknowledgment line to 1.
- The i/o device placed first in the arrangement will always get acknowledgement first. If it wants to perform i/o then it will put 0 on priority outline and will put the address of its interrupt service routine (ISR) on the vector address line.
- If device do not want to perform i/o then will set priority output as 1 and will give chance to the second device, and the processor continue.
- **Advantage: -** very simple, easy to use, easy to understand, relatively fast.
- **Disadvantage: -** here priority fixed and even in case of requirement we cannot change it.

- s/w solution: - Polling, in this method one common branch address for all interrupts. The program which takes care of interrupts begins at the branch address and polls the interrupt source in sequence.
- The order in which they are tested determines the priority of each interrupt. The highest priority source is tested first, and if its interrupt signal is no, control branches to a service routine for this source.
- Otherwise, the next lower priority source is tested and so on. Thus, the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines.
- The particular service routine reached belongs to the highest priority device among all devices that interrupted the computer.
- Disadvantage: - it is slow.

Q The following are some events that occur after a device controller issues an interrupt while process L is under execution. **(GATE-2018) (2 Marks)**
(P) The processor pushes the process status of L onto the control stack.
(Q) The processor finishes the execution of the current instruction.
(R) The processor executes the interrupt service routine.
(S) The processor pops the process status of L from the control stack.
(T) The processor loads the new PC value based on the interrupt.
(A) QPTRS **(B)** PTRSQ **(C)** TRPQS **(D)**QTPRS
(ANSWER- A)

Q A CPU handles interrupt by executing interrupt service subroutine _____. **(NET-DEC-2015)**
(a) By checking interrupt register after execution of each instruction
(b) by checking interrupt register at the end of the fetch cycle
(c) whenever an interrupt is registered
(d) by checking interrupt register at regular time interval
Ans: a

Q A CPU generally handles an interrupt by executing an interrupt service routine: **(GATE-2009) (1 Marks)**
a) As soon as an interrupt is raised.
b) By checking the interrupt register at the end of fetch cycle.
c) By checking the interrupt register after finishing the execution of the current instruction.
d) By checking the interrupt register at fixed time intervals.
Ans c

Q For the daisy chain scheme of connecting I/O devices, which of the following statements is true? **(GATE-1996) (1 Marks)**

a) It gives non-uniform priority to various devices

b) It gives uniform priority to all devices

c) It is only useful for connecting slow devices to a processor device

d) It requires a separate interrupt pin on the processor for each device

Ans: a

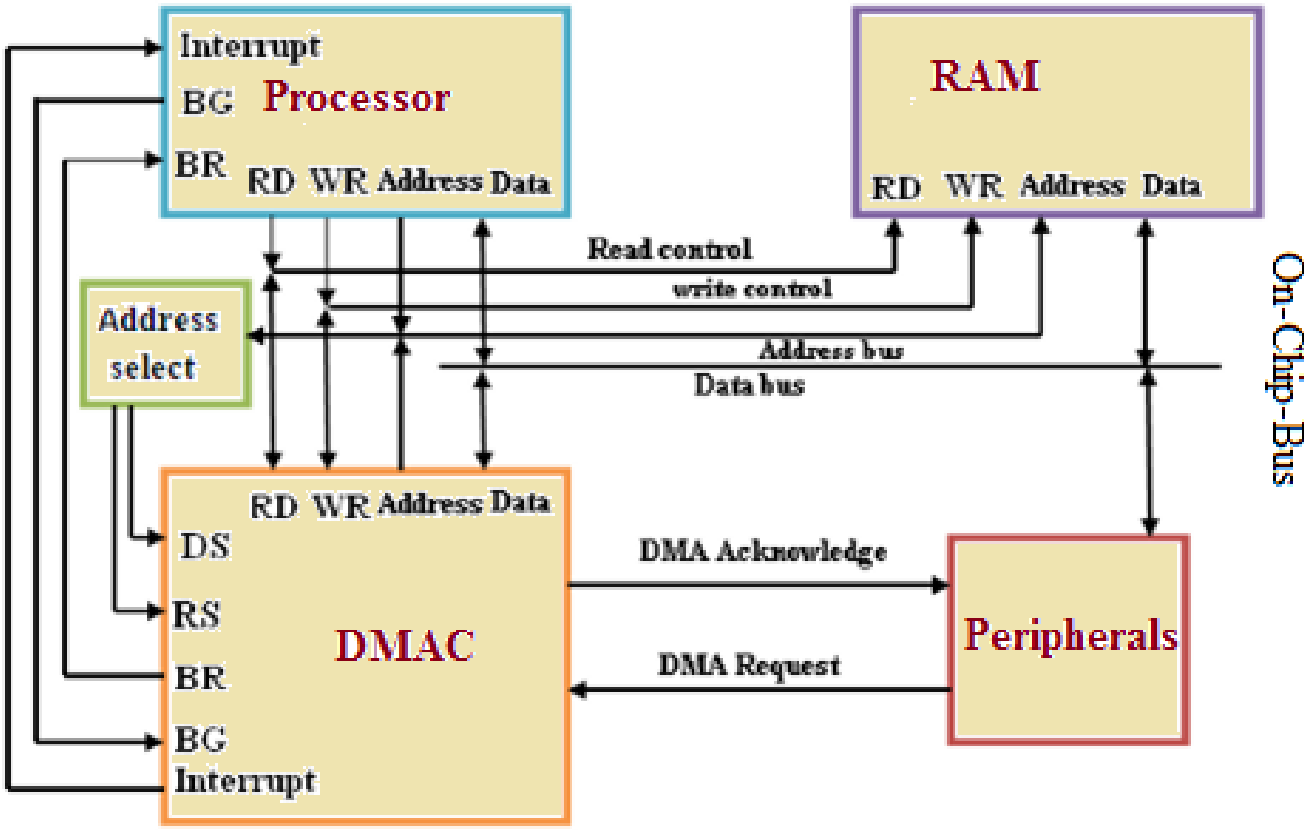
Q A device with data transfer rate 10 KB/sec is connected to a CPU. Data is transferred byte-wise. Let the interrupt overhead be 4 microsec. The byte transfer time between the device interface register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program-controlled mode? **(GATE-2005) (2 Marks)**

(A) 15 **(B)** 25 **(C)** 35 **(D)** 45

Answer: (B)

Direct Memory Access (DMA)

- When we want to perform i/o operations then the actual source or destination is either i/o device or memory, but CPU is placed in between just to manage and control the transfer.
- **DMA is** the idea where we use a new device is call DMA controller using which CPU allow DMA controller to take control of system buses and perform direct data transfer either from device to memory or from memory to device.



- **Sequence of DMA transfer: -**
  - **Step 1: -** I/O device will send a DMA request to DMA controller to perform an i/o operation.
  - **Step 2: -** DMA controller will set interrupt and bus request line to 1.
  - **Step 3: -** CPU using address bus will select device and registers and then will initiate i/o address register(location), counter (no of words)
  - **Step 4: -** CPU will put on the bus grant line to tell DMA controller, now you are the master of system buses.
  - **Step 5: -** now DMA controller will put 1 in DMA acknowledgement and using read/write control lines and address line will perform i/o directly between memory and device.

- **Mode of transfer: -**
  - **Burst mode: -** when entire i/o transfer is completed and then control comes back to CPU then it is called burst mode transfer. i.e. with high speed device like magnetic disk.
  - **Cycle stealing mode: -** When CPU executes an instruction then normally their could following phases.
    - **IF –** Instruction Fetch
    - **ID –** Instruction Decode
    - **OF –** Operand fetch
    - **IX –** Instruction execute
    - **WB –** write back or store result
- Normally in ID and IE phases CPU don't require system buses and if only in that time control is giving to DMA controller then it is called cycle stealing.

Q The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory is _____**(GATE-2016) (2 Marks)**

Answer: (456)

Q Which of the following DMA transfer modes and interrupt handling mechanisms will enable the highest I/O band-width? **GATE2006-IT-8 (GATE-2006) (1 Marks)**

a) Transparent DMA and Polling interrupts

b) Cycle-stealing and Vectored interrupts

c) Block transfer and Vectored interrupts

d) Block transfer and Polling interrupts

Ans c

Q A hard disk with a transfer rate of 10 Mbytes/ second is constantly transferring data to memory using DMA. The processor runs at 600 MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer respectively. If the size of the transfer is 20 Kbytes, what is the percentage of processor time consumed for the transfer operation? **(GATE-2004) (2 Marks)**

(A) 5.0% **(B)** 1.0% **(C)** 0.5% **(D)** 0.1%

Answer: (D)

Q A DMA controller transfers 32-bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 4800 characters per second. The CPU is fetching and executing instructions at an average rate of one million instructions per second. By how much will the CPU be slowed down because of the DMA transfer? **(NET-DEC-2015)**

(1) 0.6% **(2)** 0.12% **(3)** 1.2% **(4)** 2.5%

Ans: b

Q Consider a 32-bit microprocessor, with a 16-bit external data bus, driven by an 8 MHz input clock. Assume that this microprocessor has a bus cycle whose minimum duration equals four input clock cycles. What is the maximum data transfer rate for this microprocessor? **(NET-JUNE-2015)**

a) 8×10^6 bytes/sec **b)** 4×10^6 bytes/sec

c) 16×10^6 bytes/sec **d)** 4×10^9 bytes/sec

Ans: b

Q On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is given to transfer 500 bytes from an I/O device to memory.

```
Initialize the address register
Initialize the count to 500

LOOP: Load a byte from device
        Store in memory at address given by address register
        Increment the address register
        Decrement the count
        If count != 0 go to LOOP
```

Assume that each statement in this program is equivalent to machine instruction which takes one clock cycle to execute if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute. The designer of the system also has an alternate approach of using DMA controller to implement the same transfer. The DMA controller requires 20 clock cycles for initialization and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory. What is the approximate speedup when the DMA controller-based design is used in place of the interrupt driven program-based input-output? **(GATE-2011) (2 Marks)**

(A) 3.4 **(B)** 4.4 **(C)** 5.1 **(D)** 6.7

Answer: (A)

Q Consider a disk drive with the following specifications:
16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk is operated in cycle stealing mode whereby whenever one 4-byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4-byte word from the memory in each DMA cycle. Memory cycle time is 40nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is **(Gate-2005) (2 Marks)**

a) 10 **B)** 25 **C)** 40 **D)** 50

Ans: b

Q The storage area of a disk has the innermost diameter of 10 cm and outermost diameter of 20 cm. The maximum storage density of the disk is 1400 bits/cm. The disk rotates at a speed of 4200 RPM. The main memory of a computer has 64-bit word length and 1µs cycle time. If cycle stealing is used for data transfer from the disk, the percentage of memory cycles stolen for transferring one word is **(Gate-2004) (2 Marks)**

a) 0.5% **b)** 1% **c)** 5% **d)** 10%

Ans: c

Disk

Q Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively: **(GATE-2007) (1 Marks)**

(A) 256 Mbyte, 19 bits
(B) 256 Mbyte, 28 bits
(C) 512 Mbyte, 20 bits
(D) 64 Gbyte, 28 bit

Answer: (A)

Q For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to **(GATE-2008) (2 Marks)**

(A) non-uniform distribution of requests
(B) arm starting and stopping inertia
(C) higher capacity of tracks on the periphery of the platter
(D) use of unfair arm scheduling policies

Answer: (B)

Q If the disk is rotating at 360 rpm, determine the effective data transfer rate which is defined as the number of bytes transferred per second between disk and memory. (track size = 512bytes) **(GATE-1995) (2 Marks)**

Q Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50×10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512 byte sector of the disk is _____ **(GATE-2015) (2 Marks)**

Answer: (6.1)

Q Consider a disk pack with a seek time of 4 milliseconds and rotational speed of 10000 rotations per minute (RPM). It has 600 sectors per track and each sector can store 512 bytes of data. Consider a file stored in the disk. The file contains 2000 sectors. Assume that every sector access necessitates a seek, and the average rotational latency for accessing each sector is half of the time for one complete rotation. The total time (in milliseconds) needed to read the entire file is _____. **(GATE-2015) (1 Marks)**

Answer: (14020)

Q A certain moving arm disk storage, with one head, has the following specifications:
Number of tracks/recording surface = 200
Disk rotation speed = 2400 rpm
Track storage capacity = 62,500 bits
The average latency of this device is P ms and the data transfer rate is Q bits/sec. Write the values of P and Q. **(GATE-1993) (2 Marks)**
Ans: 2.5 mbps

Q A hard disk system has the following parameters: **(GATE-2007) (2 Marks)**

- Number of tracks = 500
- Number of sectors / track = 100
- Number of bytes / sector = 500

Time taken by the head to move from one track to adjacent track = 1 ms, Rotation speed = 600 rpm. What is the average time taken for transferring 250 bytes from the disk?
(A) 300.5 ms **(B)** 255.5 ms **(C)** 255.0 ms **(D)** 300.0 ms
Answer: (D)

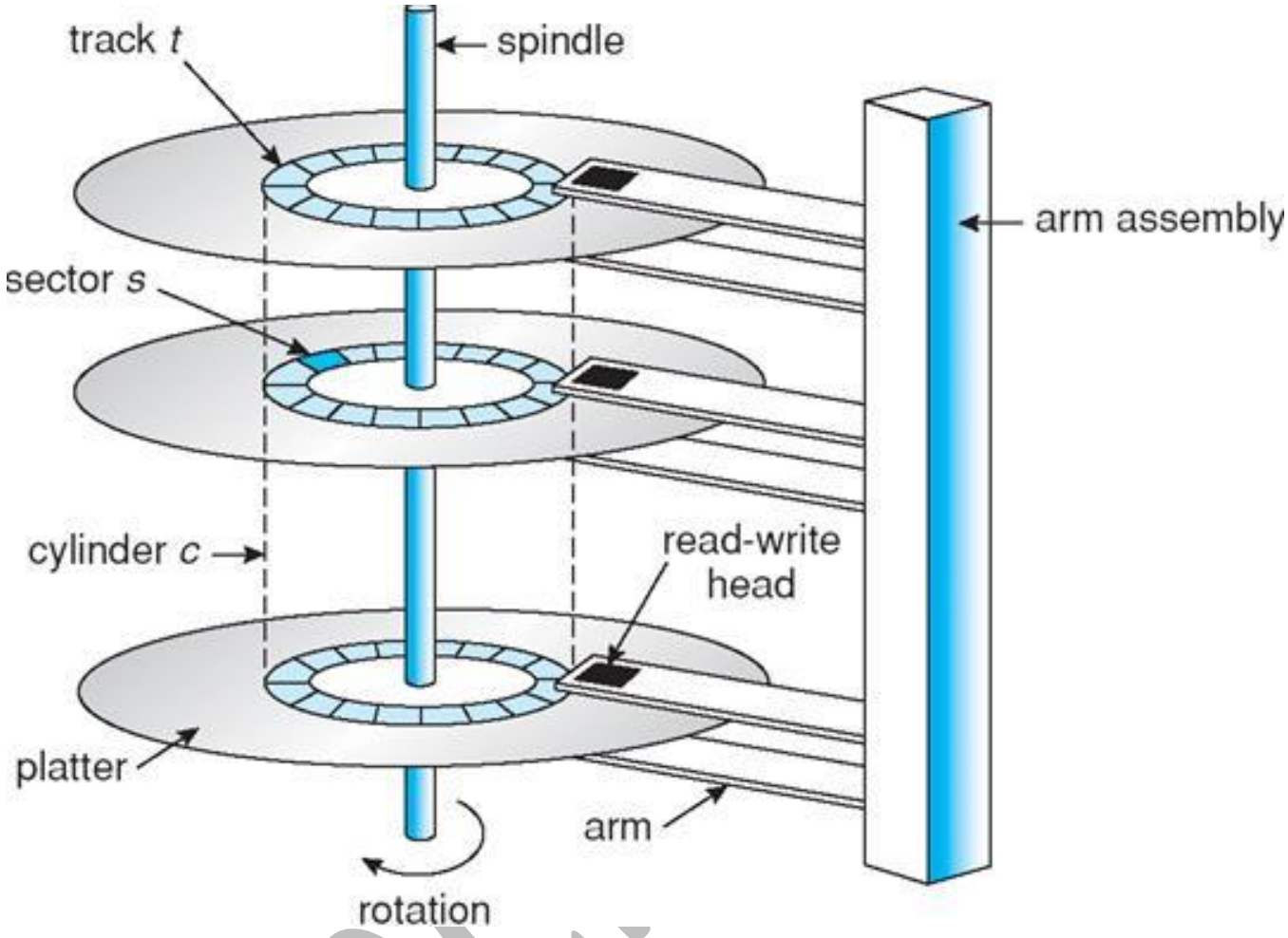
Q A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. The address of a sector is given as a triple (c, h, s), where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as (0, 0, 0), the 1st sector as (0, 0, 1), and so on The address <400,16,29> corresponds to sector number: **(GATE-2009) (2 Marks)**
(A) 505035 **(B)** 505036 **(C)** 505037 **(D)** 505038
Answer: (C)

Q The address of 1039^{th} <400,16,29> sector is **(GATE-2009) (2 Marks)**
a) <0,15,31> **b)** <0,16,30> **c)** <0,16,31> **d)** <0,17,31>
Ans: c

Q Consider a hard disk with 16 recording surfaces (0-15) having 16384 cylinders (0-16383) and each cylinder contains 64 sectors (0-63). Data storage capacity in each sector is 512 bytes. Data are organized cylinder-wise and the addressing format is <cylinder no., surface no., sector no.>. A file of size 42797 KB is stored in the disk and the starting disk location of the file is <1200, 9, 40>. What is the cylinder number of the last sector of the file, if it is stored in a contiguous manner? **(GATE-2013) (1 Marks)**

(A) 1281 **(B)** 1282 **(C)** 1283 **(D)** 1284

Answer: (D)



Q What is the swap space in the disk used for? **(GATE-2005) (1 Marks)**

a) Saving temporary html pages **b)** Saving process data

c) Storing the super-block **d)** Storing device drivers

Ans: b

Pipeline

- If the system has only one processor then at most one instruction can be executed at a time.
- And if we really want to execute multiple instruction together or concurrently then we must have multiple processors.
- Pipelining is a phenomena or method using which we will able to run more than one instruction at the same time, on a single processor.
- When we actually execute an instruction, it is divided into number of phases, like
    - Instruction Fetch
    - Instruction Decode
    - Operand fetch
    - Instruction executes
    - Instruction Store
- Where to execute each phase, we may require different number of clock cycles.
- Hardware architecture of non-pipelined and pipelined processor are different.

Q Consider a system where clock is triggering at a speed of 1MHz (1 clock = 1 µs). In a pipelined processor there are 4 stages and each stage take only 1 clock, if a program has 10 instruction then it will take what time?

i) On a non-pipelined processor
ii) On a pipelined processor

Ans

	1	2	3	4	5	6	7	8	9	10	11	12	13
IF	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	X	X	X
ID	X	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	X	X
EX	X	X	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	X
WB	X	X	X	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}

If all instructions are identical

Time without pipeline (T_{wp}) = (sum of clocks for each phase of one instruction) *(no of instruction)

If each phase requires same clock usually one (as we set the frequency in such a way)

= (no of phases * no of instruction) * time for each phase

= 4*10*1 = 40 µs

If all instructions are identical

Time with pipeline (T_p) = ((no of phase) + (no of instruction - 1)) * time of the slowest phase

If each phase requires same clock usually one (as we set the frequency in such a way)

= ((no of phase) + (no of instruction - 1)) * 1

= (4 + (10-1)) *1 = 13 µs

Speed up = (Time without pipeline (T_{wp}))/ (Time with pipeline (T_p)) = 40/13 = 3.07

Max Speed up = no of stages = 4

Efficiency = (speed up/max speed up) * 100 = 3.07/4 *100 = 76.75%

Q Consider a system where clock is triggering at a speed of 1KHz (1 clock = 1 ms). in a pipelined processor there are 5 stages and each stage take only 1 clock, if a program has 100 instruction then it will take what time?

i) On a non-pipelined processor
ii) On a pipelined processor

Ans

If all instructions are identical

Time without pipeline (T_{wp}) = (sum of clocks for each phase of one instruction) *(no of instruction)

If each phase requires same clock usually one (as we set the frequency in such a way)

= (no of phases * no of instruction) * time for each phase

= 5*100*1 = 500 ms

If all instructions are identical

Time with pipeline (T_p) = ((no of phase) + (no of instruction - 1)) * time of the slowest phase

If each phase requires same clock usually one (as we set the frequency in such a way)

= ((no of phase) + (no of instruction - 1)) * 1

= (5 + (100-1)) *1 = 104 µs

Speed up = (Time without pipeline (T_{wp}))/ (Time with pipeline (T_p)) = 500/104 = 4.8

Max Speed up = no of stages = 5

Efficiency = (speed up/max speed up) * 100 = (4.8/5) *100 = 96%

Q Consider a system where we have 'm' stages and program contains 'n' instruction if every stage take exactly 1 clock, then find the time required to execute the program without pipelining?

Ans

If all instructions are identical, have require 1 clock per instruction

Time without pipeline (T_{wp}) = m*n*1 = mn

Time with pipeline (T_p) = (m + (n-1)) *1 = m+(n-1)

Speed up(s) = (mn)/(m+(n-1)) = n/(1+ ((n-1)/(m)))

With an assumption that CPI (Clock Per Instruction) = 1

We know that n>>m therefore, T_p also approach to n. which means that speed up will approach to m and the final clock per instruction will approach to 1.

Pipelining is not only a s/w solution but it also requires h/w support: -

- where the processor must be divided into m number of stages (generally 4 or 5)
- buffers(registers) are placed b/w (some time also before and after them) so that when the first instruction complete stage 1 and switches to stage 2, then instead of waiting for first instruction to complete execution by going through all the stages, we will load 2 instruction, In the 1 stage and this idea is called pipelining, where before completing one instruction we start working another instruction.

Q consider 5 instruction with following clock requirement?

	F	D	E	WB
I_1	1	2	1	1
I_2	1	2	2	1
I_3	2	1	3	2
I_4	1	3	2	1
I_5	1	2	1	2

ANS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I_1	F	D	D	E	W										
I_2		F	X	D	D	E	E	W							
I_3			F	F	X	D	X	E	E	E	W	W			
I_4				F	F	X	D	D	D	E	E	W			
I_5					X	F	X	X	X	D	D	X	E	E	W

Ans

Time without pipeline (T_{wp}) = (sum of clocks for each phase of every instruction one by one

Time without pipeline (T_{wp}) = 5+6+8+7+6 = 31

Time with pipeline (T_p) = no formula finds it individually according to question = 15

Speed up = (Time without pipeline (T_{wp}))/ (Time with pipeline (T_p)) = 31/15 = 2.06

Note that because of stall, Efficiency = (speed up/max speed up) * 100 = 2.06/4 *100 = 51.66%

Theoretically if each stage takes only one clock and there are no stalls then speed up will be approaching to m, s=m

But in this case because the timing of stages was different and there were many stalls, we could get only 2.06 speed up.

Q Consider a 4-stage pipeline processor. The number of cycles needed by the four instructions I_1, I_2, I_3, I_4 in stages S_1, S_2, S_3, S_4 is shown below: **(Gate-2009) (2 Marks)**

	F	D	E	WB
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	2

For (I = 1; I <= 2; i++)

{

I_1

I_2

I_3

I_4

}

a) 16 **b)** 23 **c)** 28 **d)** 30

Ans: b

- We understand that different stages in a pipeline may have different delays, it also depends on the type of instruction that how much time a particular stage will take for a specific instruction.

Q Consider the time T_1 is taken for a single instruction on a pipelined processor and time T_2 is taken for a single instruction on a non-pipeline processor?

a) $T_1 = T_2$

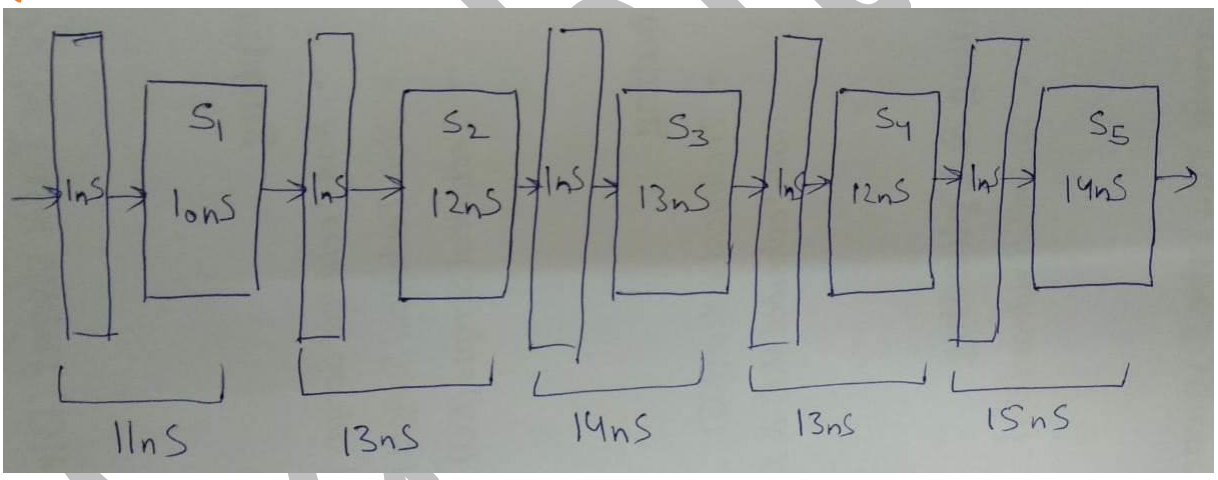
b) $T_1 <= T_2$

c) $T_1 >= T_2$

d) NONE

Ans) c, because of buffer delay in pipeline processor there are buffer between every stage so total time will contain time of stages as well as of the buffer. But for single instruction buffers are not required hence $T_1 >= T_2$ or non-pipelining processor works better.

Q

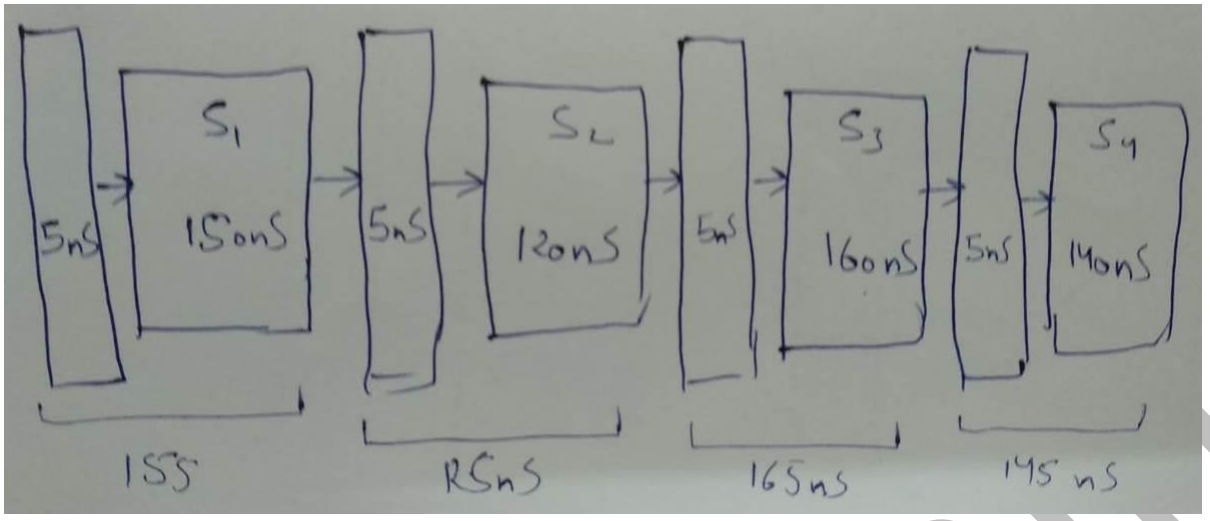


After considering this diagram what must be the frequency of the processor to ensure that work of every stage will complete in 1 clock stage wise.

Ans

What is the time taken by the longest stage along with buffer = 15 ns

F = 1/T = 1/15 GHz = .66 GHz

Q 

i) What should be the constant frequency of the processor to perform?
ii) What is the time taken by this processor without pipelining for each instruction?
iii) What is the time taken for executing a instruction with pipelining (avg)

Ans

i) What is the time taken by the longest stage along with buffer = 165 ns

F = 1/T = 1/165 GHz = 6.06 MHz

ii) Time without pipeline (T_{wp}) = 150 + 120 + 160 + 140 = 570 nS

iii) With pipelining if we run only one instruction then efficiency will be poor, if we run 10 instruction the efficiency will be better, if we run 100 instruction then efficiency will be better

a. For 1 inst = Time with pipeline (T_p) = 590 ns
b. For 10 inst, Time with pipeline (T_p) = (m + (n-1)) *time of longest stage = (4 + 9)* 165 = 2145/10 = 214.5 ns
c. For 100 inst. Time with pipeline (T_p) = (m + (n-1)) * time of longest stage = (4 + 9)* 165 = 16925/10 = 169.25 ns
d. For infinite inst, Time with pipeline (T_p) = (m + (n-1)) * time of longest stage = it will approach to 165

Q Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency. **(Gate-2014) (2 Marks)**

P_1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
P_2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
P_3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
P_4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

Which processor has the highest peak clock frequency?

(A) P_1 **(B)** P_2 **(C)** P_3 **(D)** P_4

Answer: (C)

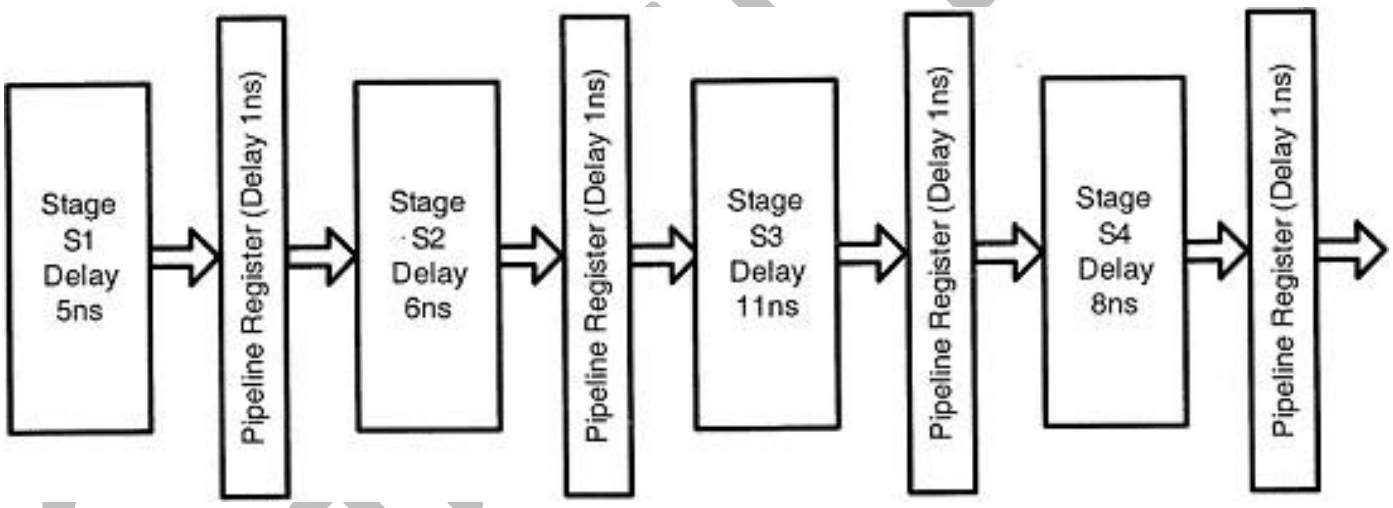
Q A 4-stage pipeline has the stage delays as 150, 120, 160 and 140 nanoseconds respectively. Registers that are used between the stages have a delay of 5 nanoseconds each. Assuming constant clocking rate, the total time taken to process 1000 data items on this pipeline will be (Gate-2004) (2 Marks)

(A) 120.4 microseconds **(B)** 160.5 microseconds

(C) 165.5 microseconds **(D)** 590.0 microseconds

Answer: (C)

Q Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure:



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation? (Gate-2011) (2 Marks)

(A) 4.0 **(B)** 2.5 **(C)** 1.1 **(D)** 3.0

Answer: (B)

Q A non-pipelined single cycle processor operating at 100 MHz is converted into a synchronous pipelined processor with five stages requiring 2.5 nsec, 1.5 nsec, 2 nsec, 1.5 nsec and 2.5 nsec, respectively. The delay of the latches is 0.5 nsec. The speedup of the pipeline processor for a large number of instructions is (Gate-2008) (2 Marks)

(A) 4.5 **(B)** 4.0 **(C)** 3.33 **(D)** 3.0

Answer: (C)

Q We have two designs D1 and D2 for a synchronous pipeline processor. D1 has 5 pipeline stages with execution times of 3 nsec, 2 nsec, 4 nsec, 2 nsec and 3 nsec while the design D2 has 8 pipeline stages each with 2 nsec execution time How much time can be saved using design D2 over design D1 for executing 100 instructions? **(Gate-2005) (2 Marks)**

(A) 214 nsec **(B)** 202 nsec **(C)** 86 nsec **(D)** – 200 nsec

Answer: (B)

Q Instruction execution in a processor is divided into 5 stages. Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Execute (EX), and Write Back (WB), These stages take 5,4,20, 10 and 3 nanoseconds (ns) respectively.

A pipelined implementation of the processor requires buffering between each pair of consecutive stages with a delay of 2 ns. Two pipelined implementations of the processor are contemplated:

(i) a naïve pipeline implementation (NP) with 5 stages and

(ii) an efficient pipeline (EP) where the OF stage id divided into stages OF1 and OF2 with execution times of 12 ns and 8 ns respectively.

The speedup (correct to two decimals places) achieved by EP over NP in executing 20 independent instructions with no hazards is_____. **(Gate-2017) (2-marks)**

Ans: 1.508

Q The stage delays in a 4-stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionally equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is_____percent. **(Gate-2016) (2 Marks)**

Ans:33.33%

Q Consider a non-pipelined processor with a clock rate of 2.5 gigahertz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume that there are no stalls in the pipeline. The speed up achieved in this pipelined processor is_____.**(Gate-2015)(2-marks)**

Ans: 3.2

Q Consider a 3 GHz (gigahertz) processor with a three-stage pipeline and stage latencies τ_1, τ_2 and τ_3 such that $\tau 1 = (3\ \tau_2)\ /4 = 2\tau_3$. If the longest pipeline stage is split into two pipeline stages of equal latency, the new frequency is_____GHz, ignoring delays in the pipeline registers. **(Gate-2016) (2 Marks)**

Ans: 4

Q Suppose the functions F and G can be computed in 5 and 3 nanoseconds by functional units UF and UG, respectively. Given two instances of UF and two instances of UG, it is required to implement the computation F(G(Xi)) for 1 <= i <= 10. ignoring all other delays, the minimum time required to complete this computation is nanoseconds (Gate-2016) (2 marks)

Ans: 28ns

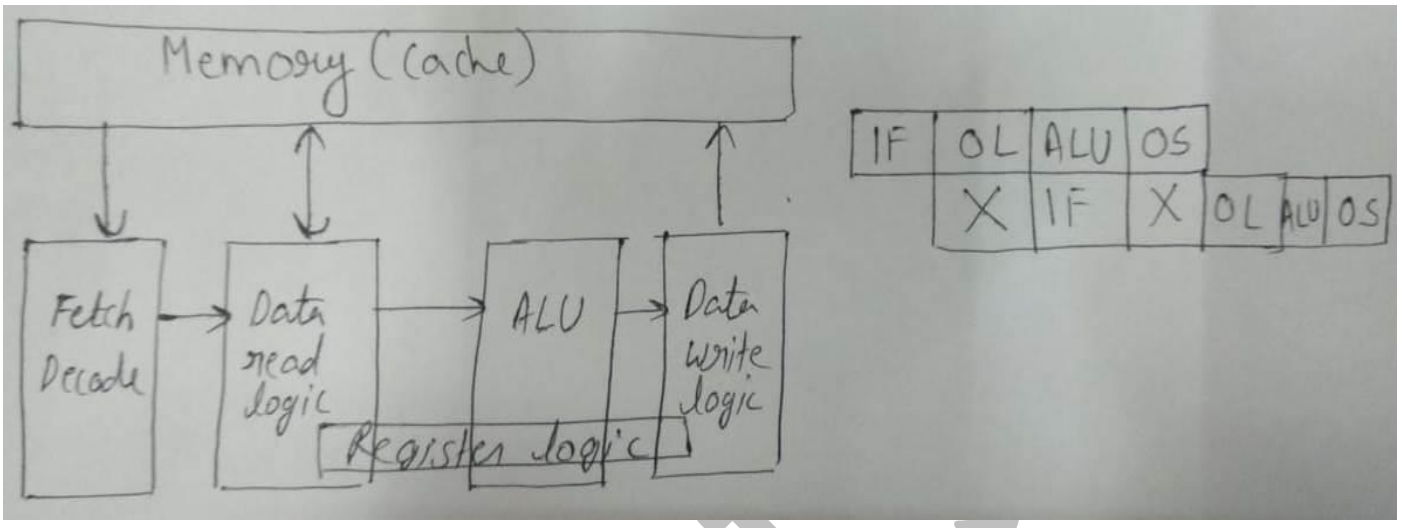
Hazards/ dependency

Sometimes we cannot execute instructions with full efficiency in a pipeline because of certain dependency or hazards.

- Structural hazards
- Control hazards
- Data hazards

Structural Hazards

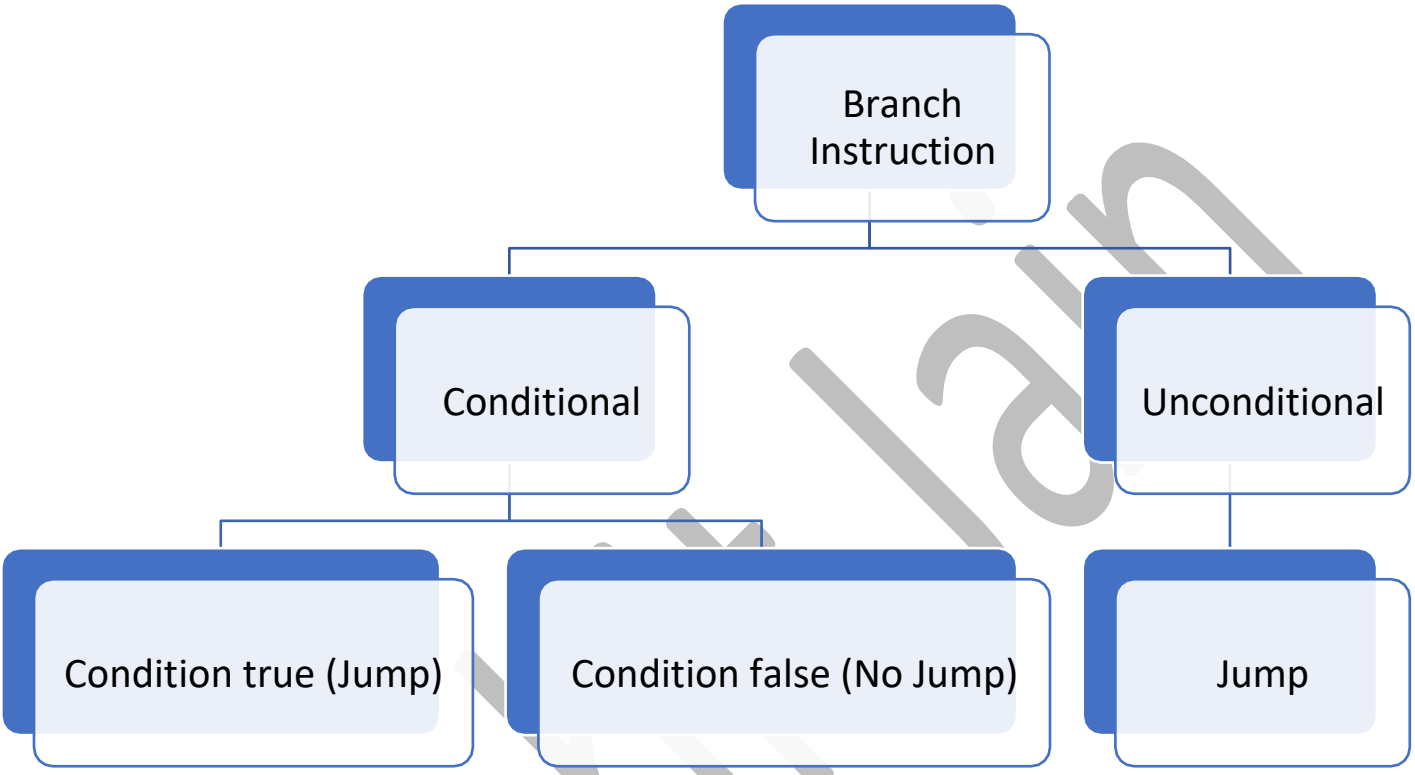
- Even by having ‘m’ stages in a pipeline processor and assuming that the combinational circuit of every stage is different many times there will be dependence because of external h/w a part from CPU like system bus, memory etc. because of this we have to use stall or stall cycles and the efficiency or speed up decreases.



- A structural hazard cannot be removed using efficient program. Therefore, one solution could be resource duplication but the cost of implementation will be very high.

Control Hazards

- If some instruction I_1 is branch instruction and after executing the entire instruction we understand that there is jump and next instruction to be executed is I_{10}. This time we have already partially executed I_1, I_2, I_3 which is the problem.



Q Consider a system where 20% instruction are branch instruction and because of them we suffer following stall. Find the CPI or Clock Per Instruction because of these control dependencies.

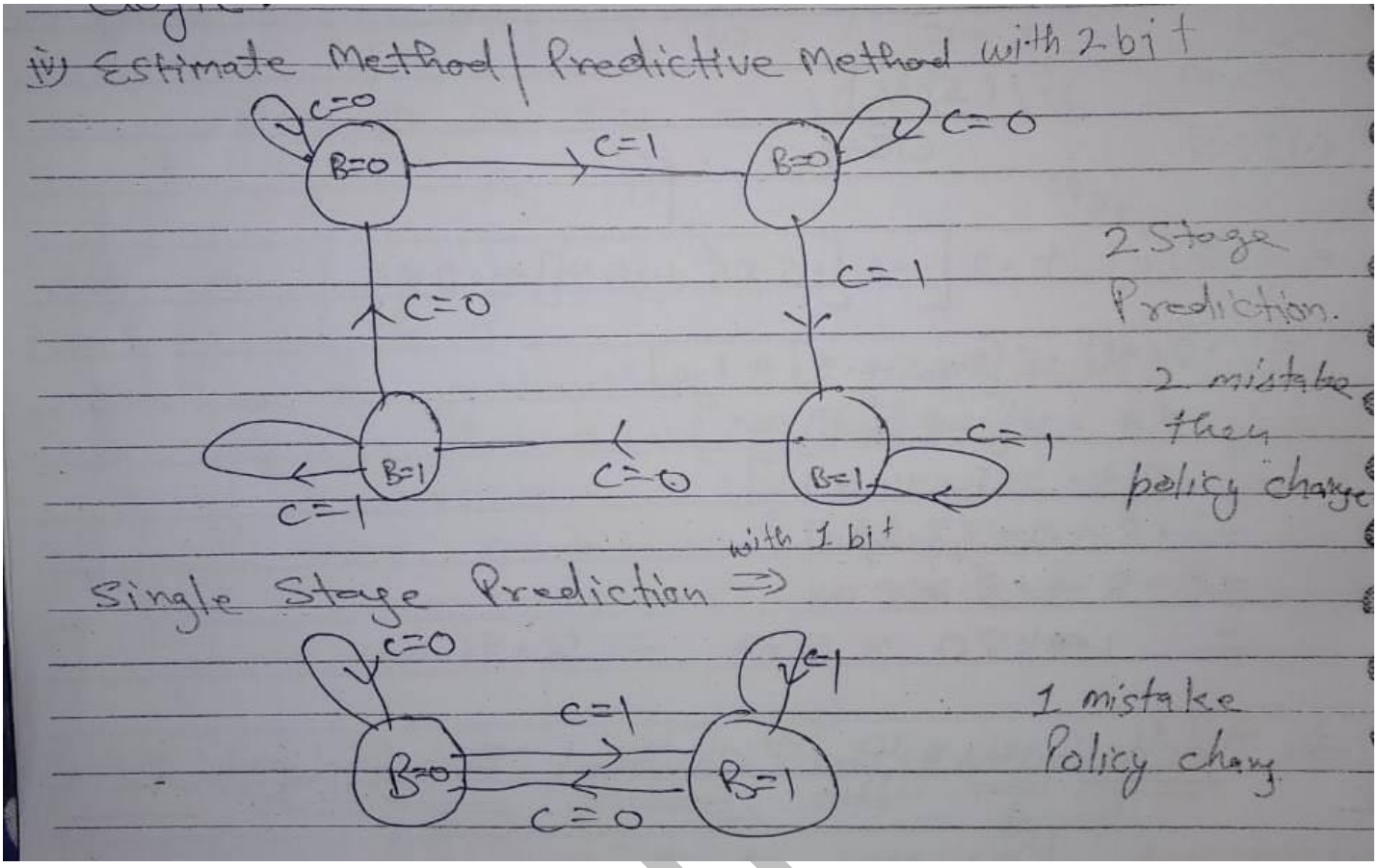
1) Branch Instruction = 1 Stall
   ((80/100)*1) + ((20/100)*2) = 1.2

2) Branch Instruction = 2 Stall
   ((80/100)*1) + ((20/100)*3) = 1.4

3) Branch Instruction = 3 Stall
   ((80/100)*1) + ((20/100)*4) = 1.6

Solution of control hazards: -

- **Flushing/Stalling: -**It is the worst-case scenario either we remove the computed data or we don't work at all.
- **NOP (no-operation): -** Which means compiler don't execute any instruction here.
- **Code rearrangement or delayed load: -** Here with smart compilers we can first execute some instruction which are independent from the current logic.
- **Estimation method/ predictive method: -** here we use either 2 stage prediction, then policy change or 1 stage prediction then policy change.



Q Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle-time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined execution, if 25% of the instructions incur 2 pipeline stall cycles is **(Gate-2014, 2marks)**

Ans: 4

Q An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register writeback (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nanoseconds).

To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns.

Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed. All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times

of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is_____.**(Gate-2014, 2marks)** ?

Ans: 1.54

Q Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 ns. A program consisting of 12 instructions I1, I2, I3, …, I12 is executed in this pipelined processor. Instruction I4 is the only branch instruction and its branch target is I9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is (Gate-2013) (2 Marks)

(A) 132 **(B)** 165 **(C)** 176 **(D)** 328

Ans: b

Q A CPU has a five-stage pipeline and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions out of which 20% are conditional branches. If each instruction takes one cycle to complete on average, the total execution time of the program is: **(Gate-2006) (2 Marks)**

(A) 1.0 second **(B)** 1.2 seconds **(C)** 1.4 seconds **(D)** 1.6 seconds

Answer: (C)

Data Dependency (Data Hazard)

- Let us consider an ADD instruction S, such that **S: ADD R1, R2, R3**
  - **Addresses read by S = I(S) = {R2, R3}**
  - **Addresses written by S = O(S) = {R1}**

- **We say that instruction S2 depends in instruction S1, when:** This condition is called **Bernstein condition.**

$$[I(S1) \cap O(S2)] \cup [O(S1) \cap I(S2)] \cup [O(S1) \cap O(S2)] \neq \phi$$

- Three cases exist:
  - **Flow (data) dependence: O(S1) ∩ I(S2), S1 → S2 and S1 writes after something read by S2**
  - **Anti-dependence: I(S1) ∩ O(S2), S1 → S2 and S1 reads something before S2 overwrites it**

- **Output dependence: O(S1) ∩ O(S2), S1 → S2 and both write the same memory location.**

Example: Let there be two instructions I1 and I2 such that

I1 : ADD R1, R2, R3

I2 : SUB R4, R1, R2

- When the above instructions are executed in a pipelined processor, then data dependency condition will occur, which means that I_2 tries to read the data before I_1 writes it, therefore, I_2 incorrectly gets the old value from I_1.
- To minimize data dependency stalls in the pipeline, **operand forwarding** is used.

Data Hazards

- Data hazards occur when instructions that exhibit data dependence, modify data in different stages of a pipeline.
- Hazard cause delays in the pipeline. There are mainly three types of data hazards:
    1. RAW (Read after Write) [Flow/True data dependency]
    2. WAR (Write after Read) [Anti-Data dependency]
    3. WAW (Write after Write) [Output data dependency]

- Let there be two instructions I and J, such that J follow I. Then,
- RAW hazard occurs when instruction J tries to read data before instruction, I write it.

    Example: I: R2 <- R1 + R3
    J: R4 <- R2 + R3

    Here, J is trying to read R2 before I have written it.

- WAR hazard occurs when instruction J tries to write data before instruction, I read it.

    Example: I: R2 <- R1 + R3
    J: R3 <- R4 + R5

    Here, J is trying to write R3 before I have read it.

- WAW hazard occurs when instruction J tries to write output before instruction, I write it.

    Example: I: R2 <- R1 + R3
    J: R2 <- R4 + R5

    Here J is trying to write R2 before I.

WAR and WAW hazards occur during the out-of-order execution of the instructions.

- In this example the content of the register is required by instruction, which will be available only if instruction will fetch and decode. I will have to wait and will suffer a single stall and then only can continue with execution.

IF	ID	EX	WB	
	IF	ID	EX	WB

IF	ID	EX	WB		
	IF	ID	X	EX	WB

Solution of Data dependency: -

i) We can use code movement or code relocation and can execute the dependent instruction after some time.

ii) Here we can use operator forwarding using which we can directly access the result after execution instead of waiting that it gets store in memory.

Q The instruction pipeline of a RISC processor has the following stages: Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Writeback (WB), The IF, ID, OF and WB stages take 1 clock cycle each for every instruction. Consider a sequence of 100 instructions. In the PO stage, 40 instructions take 3 clock cycles each, 35 instructions take 2 clock cycles each, and the remaining 25 instructions take 1 clock cycle each. Assume that there are no data hazards and no control hazards. The number of clock cycles required for completion of execution of the sequence of instruction is_____.
(GATE-2018) (2Marks)
Ans: 219

Q A 5-stage pipelined processor has Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO)and Write Operand (WO)stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD and SUB instructions,3 clock cycles for MUL instruction, and 6 clock cycles for DIV instruction respectively. Operand forwarding is used in the pipeline. What is the number of clock cycles needed to execute the following sequence of instructions? **(Gate-2010) (2 Marks)**

	Instruction	Meaning of instruction
iii)	I_0: MUL R2, R0, R1	R2 ¬ R0 *R1
iv)	I_1: DIV R5, R3, R4	R5 ¬ R3/R4
v)	I_2: ADD R2, R5, R2	R2 ¬ R5+R2
vi)	I_3: SUB R5, R2, R6	R5 ¬ R2-R6

(A) 13 **(B)** 15 **(C)** 17 **(D)** 19

Answer: (B)

Q Consider the sequence of machine instructions given below: **(Gate-2015) (2 Marks)**

```
MUL R5, R0, R1
DIV R6, R2, R3
ADD R7, R5, R6
SUB R8, R7, R4
```

In the above sequence, R0 to R8 are general purpose registers. In the instructions shown, the first register stores the result of the operation performed on the second and the third registers. This sequence of instructions is to be executed in a pipelined instruction processor with the following 4 stages:
(1) Instruction Fetch and Decode (IF),
(2) Operand Fetch (OF),
(3) Perform Operation (PO) and
(4) Write back the Result (WB).
The IF, OF and WB stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD or SUB instruction, 3 clock cycles for MUL instruction and 5 clock cycles for DIV instruction.
The pipelined processor uses operand forwarding from the PO stage to the OF stage. The number of clock cycles taken for the execution of the above sequence of instructions is _____

Answer: (13)

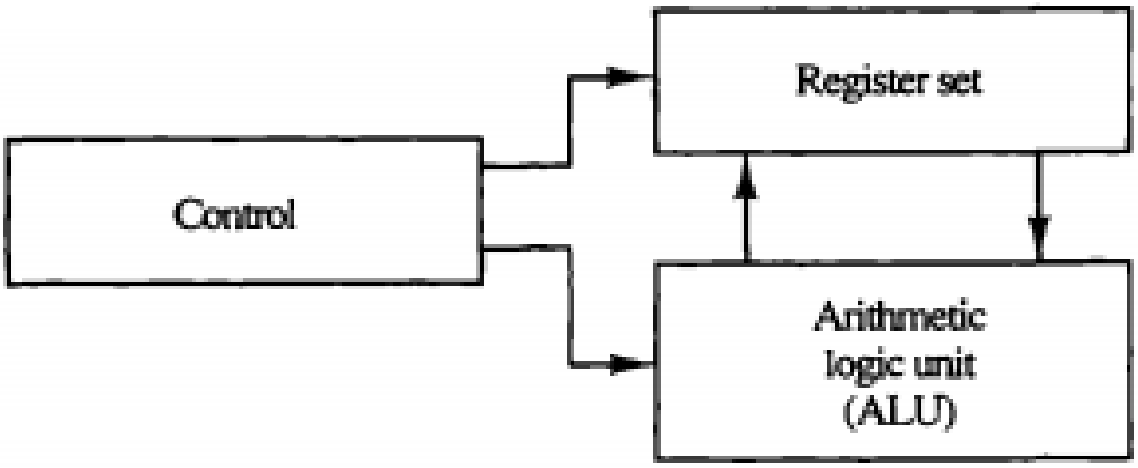
Instruction format and addressing mode

- Computer can be of two types
    - **General purpose computer: -** Which can perform anything which can be theoretically done by a Turing machine. These computer works on stored program concept where a programmer, writes a program store it in the memory and then computer executes it, based on the requirement program can be changed and so does the functionality.
    - **Dedicated device or embedded system: -** They are designed to perform a specific task their functionality is written in terms of program which is permanently fused in a chipset. E.g. washing machine, microwave etc

- **Machine / CPU architecture: -** In general a CPU contain 3 important components
    - **Control unit: -** Which generates control signal (master generator) to control every other part of the CPU.
    - **Registers: -** few important registers are in every processor like program counter which contains address of the next instruction then instruction register which contain address of the current register and base register which contain base address of program.
    - **ALU: -** ALU is a complex combination circuit which can perform arithmetic and logical operation.

    - The part of the computer that performs the bulk of data-processing operations is called the ***central processing unit or CPU.***

- The CPU is made up of three major parts: Control, Register Set and ALU.



- The register set stores intermediate data used during the execution of the instructions.
- The arithmetic logic unit (ALU) performs the required microoperations for executing the instructions.
- The control unit supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.

- how in general operation are performed?
    - ***memory -> register -> ALU -> perform operation -> register -> memory.***

Instruction format

- In general, based on the number of operands or reference made in the instructions. Instructions can be classified into the following types: -

• 4 Address Instruction

Mode/Opcode	Opcode1	Opcode2	Result	Next Instruction

- In early computer we have used these four-address instructions where the first part represent mode and opcode and then we have four address opcode1, opcode2, Result and the address of the next instruction.
- Advantage
  - Very simple, easy to use easy to understand
- Disadvantage
  - Very lengthy and occupied a lot of space.

Instruction Formats

- Computers may have instructions of several different lengths containing varying number of addresses.
- The number of address fields in the instruction format of a computer depends on the internal organization of its registers.
- Most computers fall into one of three types of CPU organizations:

  1. Single accumulator organization.

  2. General register organization.

  3. Stack organization.

- **In single accumulator organization**, all operations are performed with an implied accumulator register. The instruction format in this type of computer uses one address field.

  ADD X, where X is the address of the operand.

- The ADD instruction in this case results in the operation **AC □- AC + M [X].**
- AC is the accumulator register and M[X] symbolizes the memory word located at address X.

- The instruction format in **general register type organization** of computer needs three register address fields.

  ADD R1, R2, R3

  It denote the operation **R1 ← R2 + R3**.

- The number of address fields in the instruction can be reduced from three to two if the destination register is the same as one of the source registers. Thus the instruction

  ADD R1, R2

  It denote the operation **R1 ← R1 + R2.**

- Computers with multiple processor registers use the move instruction with a mnemonic MOV to symbolize a transfer instruction. Thus the instruction

  MOV R1, R2

  It denotes the transfer **R1 ← R2**

- **Computers with stack organization** would have PUSH and POP instructions which require an address field. Thus the instruction

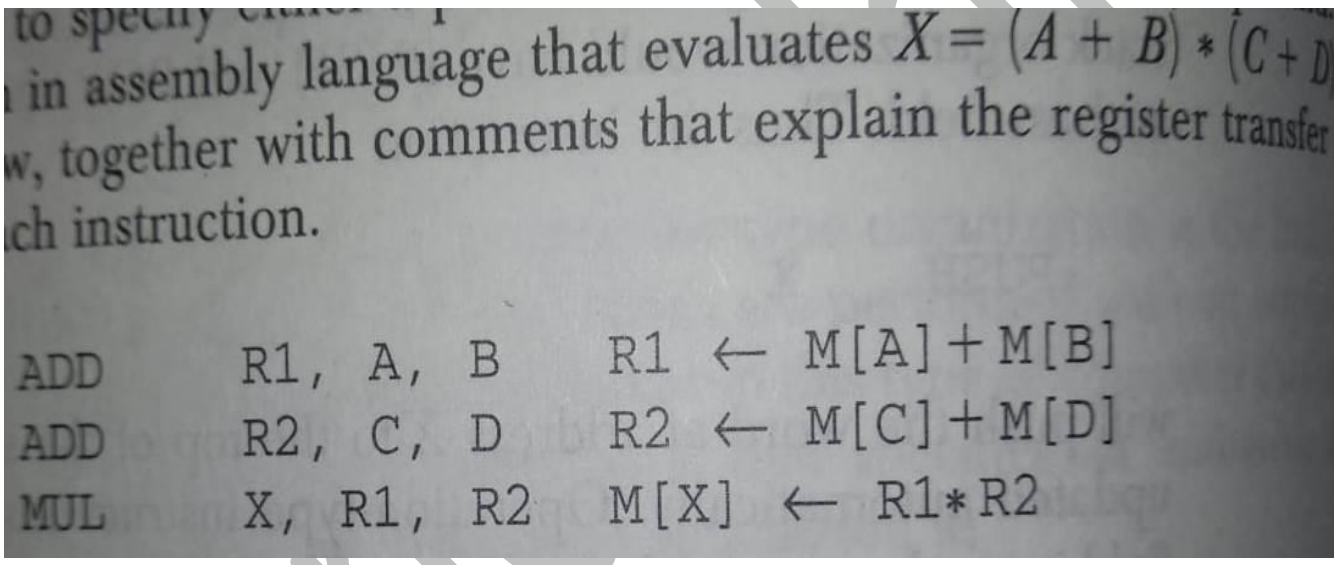
  PUSH X

  It will push the word at address X to the top of the stack.

- **3 Address Instruction**

Opcode	Opcode1	Opcode2	Result/Destination

- In three address instruction we don't use address of the next instruction because these systems have a special purpose register called program counter which do instruction sequencing or address of next instruction.
- In operand1, operand2 and data field to specify a register or a memory location.
- It is relatively better than 4 address instruction.
- As most of the operations are of binary nature because of this it is very convenient with human understanding and that is why very popular.
- Disadvantage is still it is very much lengthy, which uses a lot of space.


to specify
in assembly language that evaluates $X = (A + B) * (C + D$
w, together with comments that explain the register transfer
ch instruction.

```
ADD     R1, A, B    R1 ← M[A] + M[B]
ADD     R2, C, D    R2 ← M[C] + M[D]
MUL     X, R1, R2   M[X] ← R1 * R2
```

Three-Address Instructions

- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

Example: ***X = (A + B) * (C + D)***

ADD R1, A, B **R1 ← M [A] + M[B]**
ADD R2, C, D **R2 ← M [C] + M[D]**
MUL X, R1, R2 **M [X] ← R1 * R2**

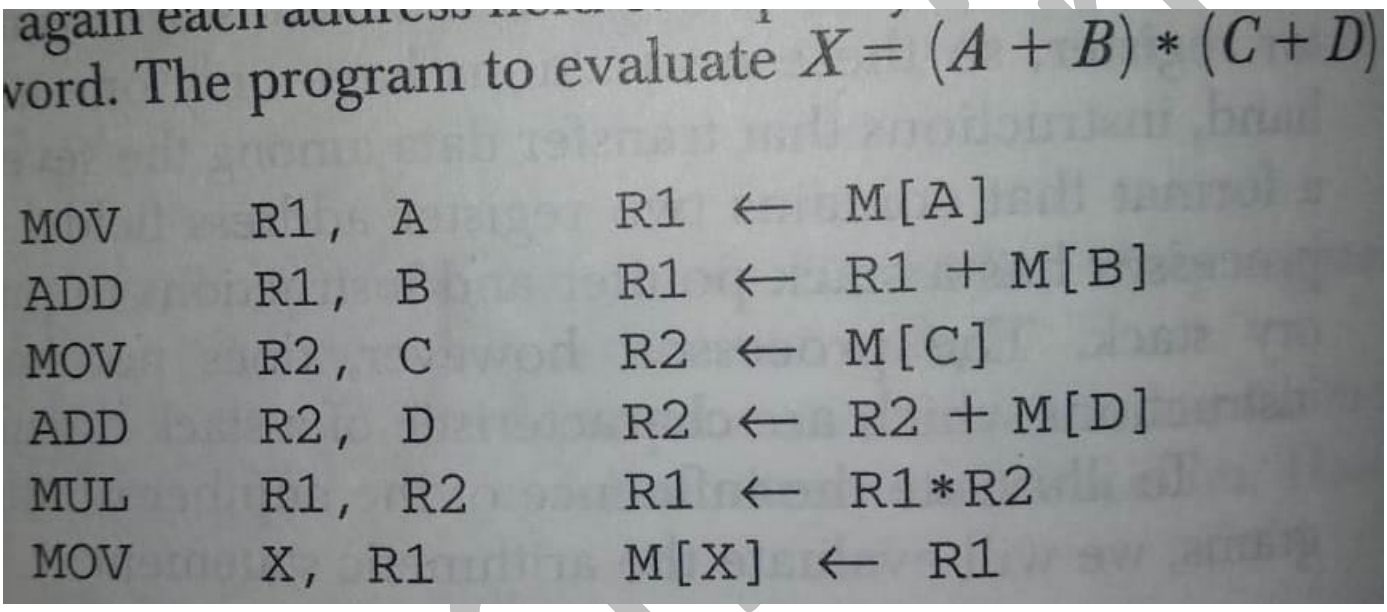
- The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions.

- The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

- **2 Address Instruction**

Opcode	Opcode1	Opcode2

- In effort to reduce the size of instruction here we remove the reference for result and opcode1 is used for storing the result.
- Less length compares to 3 address, more efficient, no of register required are less
- Here code optimization is relatively difficult.


again each address field ... word. The program to evaluate $X = (A + B) * (C + D)$

```
MOV     R1, A       R1 ← M[A]
ADD     R1, B       R1 ← R1 + M[B]
MOV     R2, C       R2 ← M[C]
ADD     R2, D       R2 ← R2 + M[D]
MUL     R1, R2      R1 ← R1*R2
MOV     X, R1       M[X] ← R1
```

Two-Address Instructions

- Each address field can specify either a processor register or a memory word.

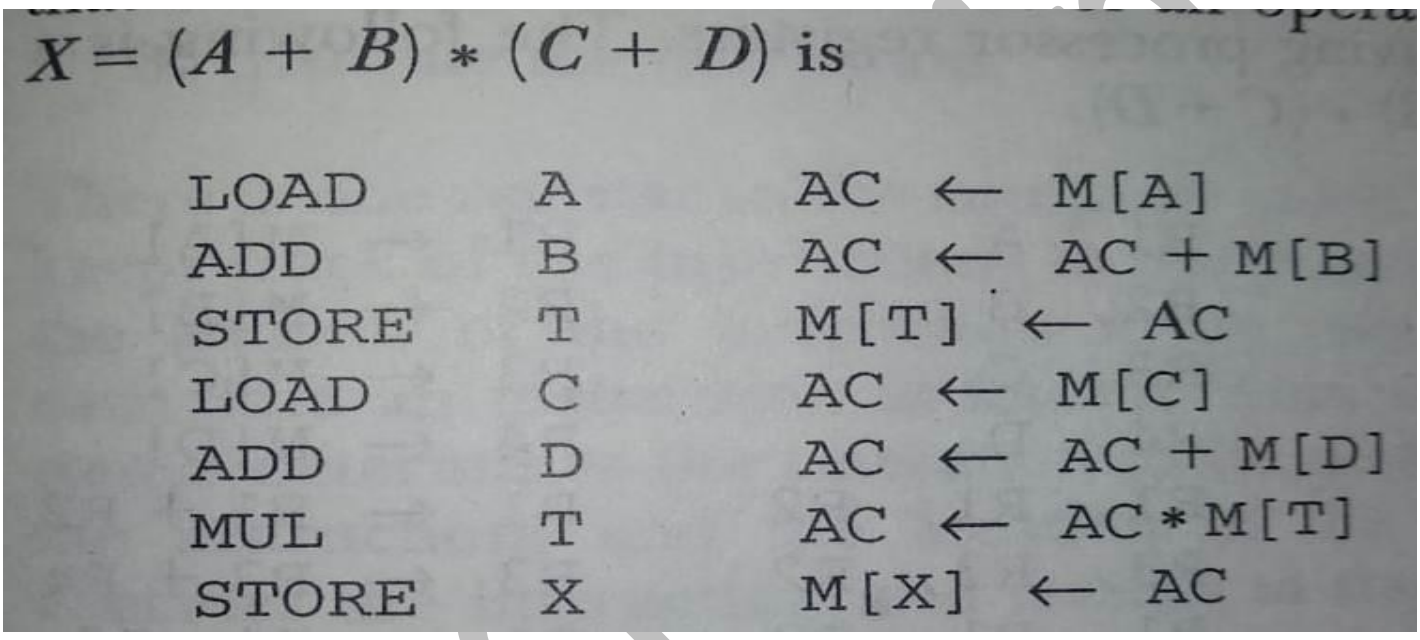
Example: X = (A + B) * (C + D)

```
MOV     R1, A       R1 ? M [A]
ADD     R1, B       R1 ? R1 + M [B]
MOV     R2, C       R2 ? M [C]
ADD     R2, D       R2 ? R2 + M [D]
MUL     R1, R2          R1 ? R1 * R2
MOV     X, R1       M [X] ? R1
```

- **1 Address Instruction**

Opcode	Opcode1

- One address instruction uses an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, we neglect the second register and assume that the AC contains the result of all operations.



One-Address Instructions

- One-address instructions use an implied accumulator (AC) register for all data manipulation.
- For multiplication and division there is a need for a second register. However, here we will neglect the second register and assume that the AC contains the result of all operations.

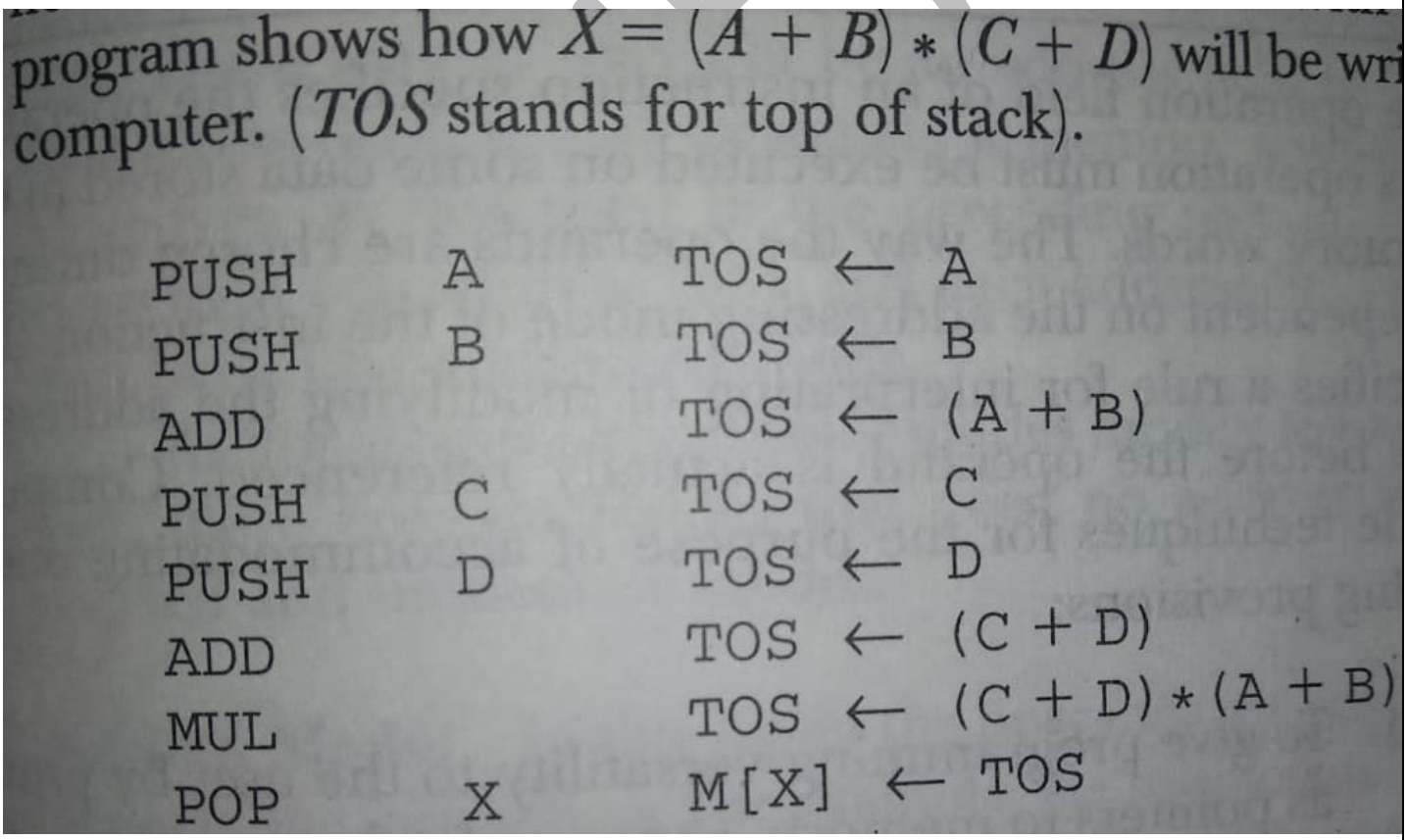
Example: X = (A + B) * (C + D)

LOAD	**A**	**AC M [A]**
ADD	**B**	**AC A C + M [B]**
STORE	**T**	**M [T] AC**
LOAD	**C**	**AC M [C]**
ADD	**D**	**AC AC + M [D]**
MUL	**T**	**AC AC * M [T]**
STORE	**X**	**M [X] A C**

- **0 Address Instruction**

Opcode

- Early computers which do not have complex circuit like ALU use few registers which are used as organised stack. While working on these stacks we do not require location of result as it is implied at the top of the stack.
- A stack organized computer does not use an address field for the instructions ADD and MUL. The push and pop instruction, however, need an address field to specify the operand that communicates with stack.
- To evaluate arithmetic expression in a stack computer, it is necessary to Covert the expression into reverse polish notation. The name zero-address is given to this type of computer because of the absences of an address field in the com
- Here mode/opcode is sufficient is sufficient to perform any operation.
- It is very simple and very short, but complex to design ad speed is very poor.


program shows how $X = (A + B) * (C + D)$ will be wri
computer. (*TOS* stands for top of stack).

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A + B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C + D)
MUL		TOS ← (C + D) * (A + B)
POP	X	M[X] ← TOS

Zero-Address Instructions

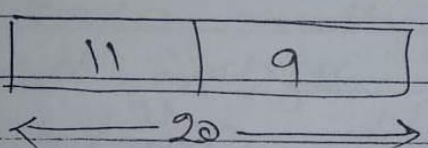
- A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack.

Example: X = (A + B) * (C + D) will be written for a stack organized computer:

PUSH	**A**	**TOS ← A**
PUSH	**B**	**TOS ← B**
ADD		**TOS ← (A + B)**
PUSH	**C**	**TOS ← C**
PUSH	**D**	**TOS ← D**
ADD		**TOS ← (C + D)**
MUL		**TOS ← (C + D) * (A + B)**
POP	**X**	**M [X] ← T O S**

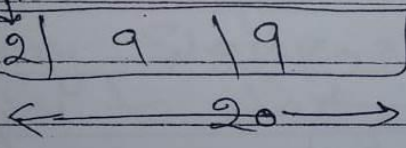
Consider a System where Memory Contains 512 words if we want 2048, 1-address instruction and 4, 2address instruction find the length of instruction.

Sol^{n} I_1 0 | 11 | 9 |



opcode (bcoz 4bit)

I_2 1 | 2 | 9 | 9 |



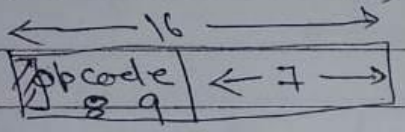
Total = 21 instruction

bcoz 1 show 1 address or 2 address register

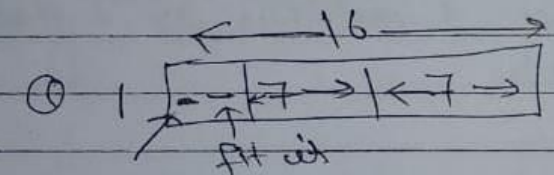
Q⇒ Consider a hypothetical machine which support both 1 address and 2 address instruction if each instruction of 16 bit and memory contain only 128 words.
if it know that there are P-two address instruction in the system how many one address instruction can be supported.

a) 128 b) 256 c) 384 d) 512

Soln ⇒ P



$= 2^8 = 256$



one bit for identy
0 → 1-address
1 → 2-address

(24)

PQR

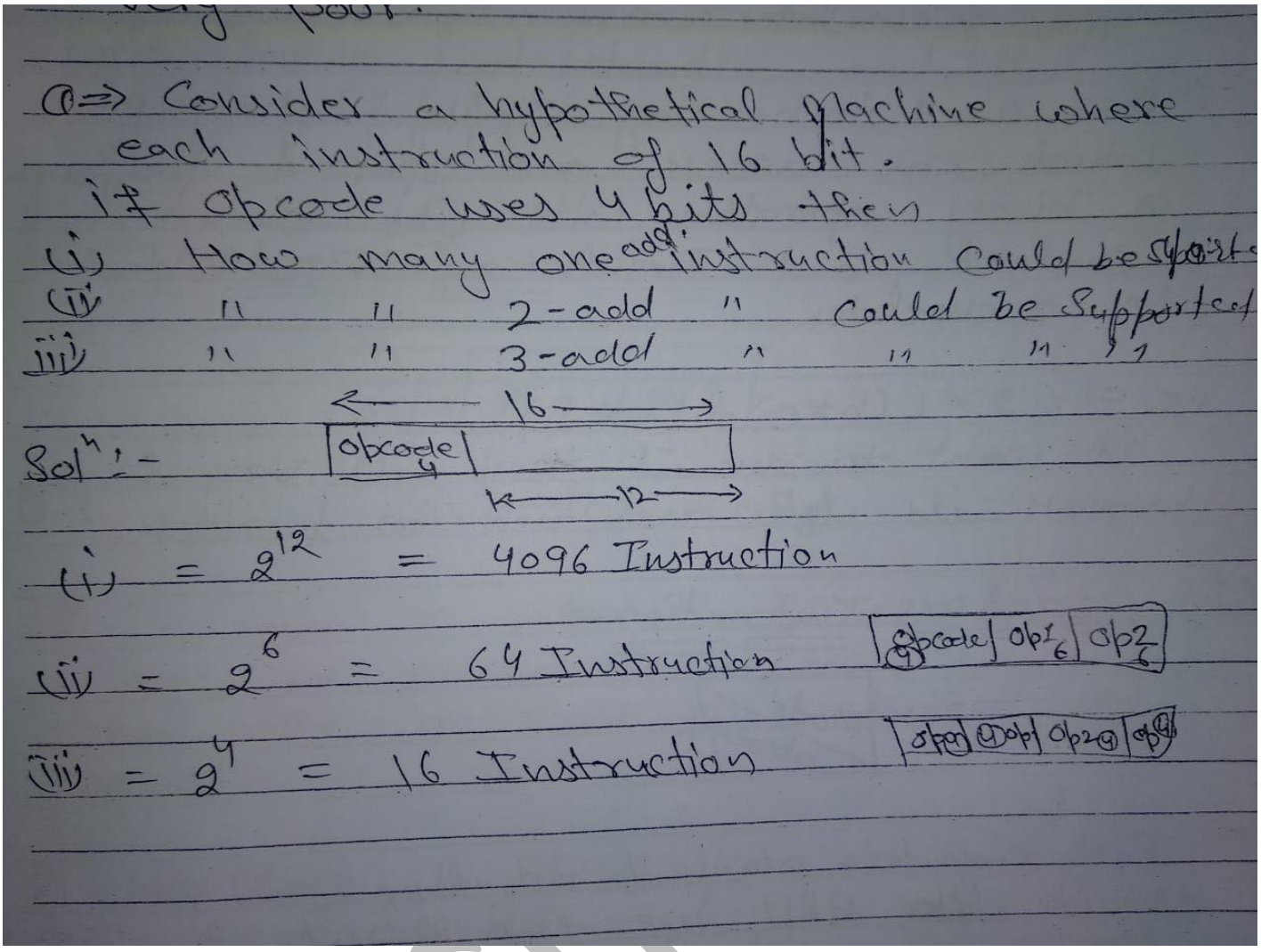
$\rightarrow (P * 2^7) + (Q * 2^7 * 2^7) = 2^{16}$

$P * 2^7 + Q \times 2^{14} + R \times 2^{21} = 2^{16}$

$P * 2^7 = 2^{16} - 2^{15}$

$P * 2^7 = 2^{15}$

$P = 2^8 = 256$



Q Consider a processor with 64 registers and an instruction set of size twelve. Each instruction has five distinct fields, namely, opcode, two source register identifiers, one destination register identifier, and a twelve-bit immediate value. Each instruction must be stored in memory in a byte-aligned fashion. If a program has 100 instructions, the amount of memory (in bytes) consumed by the program text is _____ **(Gate-2016) (2 Marks)**

Answer: (500)

Q A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)? **(Gate-2006) (1 Marks)**

(A) 400 **(B)** 500 **(C)** 600 **(D)** 700

Answer: (C)

Q A processor has 16 integer registers (R_0, R_1, …, R_{15}) and 64 floating point registers (F_0, F_1, …, F_{63}). It uses a 2-byte instruction format. There are four categories of instructions: Type-1, Type-2, Type-3, and Type 4. Type-1 category consists of four instructions, each with 3 integers register operands (3Rs). Type-2 category consists of eight instructions, each with 2 floating point register operands (2Fs). Type-3 category consists of fourteen instructions, each with one integer register operand and one floating point register operand (1R+1F). Type-4 category consists of N instructions, each with a floating-point register operand (1F). The maximum value of N is _____. **(Gate-2018) (2 Marks)**
Answer: (32)

Q A Computer uses a memory unit with 256K word of 32 bits each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code and a register code part to specify one of 64 registers and an address part. How many bits are there in operation code, the register code part and the address part? **(NET-DEC- 2018)**
a) 7 7 18 **b)** 18 7 7 **c)** 7 6 18 **d)** 6 7 18
ans: c

Q Computers can have instruction formats with **(NET-JUNE-2013)**
(A) only two address and three address instructions
(B) only one address and two address instructions
(C) only one address, two address and three address instructions
(D) zero address, one address, two address and three address instructions
Ans: d

Q Which of the following is a design criterion for instruction formats? **(NET-DEC-2013)**
(A) The size of instructions
(B) The number of bits in the address fields
(C) The sufficient space in the instruction format to express all the operations desired.
(D) All of these
Ans: d

Addressing Mode

- It specifies the different ways possible in which reference to the operand can be made.
- Effective address: - It is the final address of the location where the operand is stored.
- Calculation of the effective address can be done in two ways
    - Non-computable addressing
    - Computable addressing (which involve arithmetic's)
- Criteria for different addressing mode
    - It should be fast
    - The length of the instruction must be small
    - They should support pointers
    - They should support looping constructs, indexing of data structure
    - Program relocation

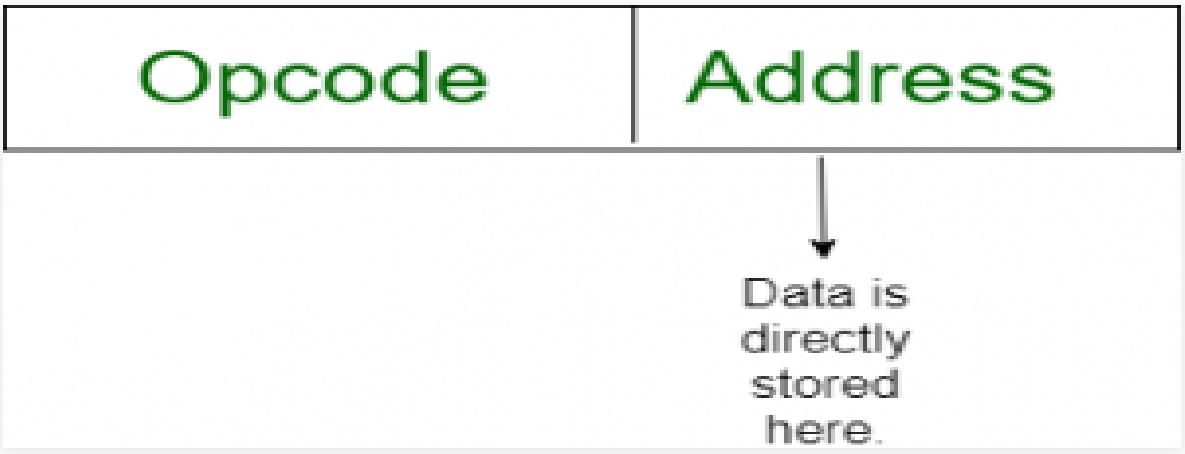
Addressing Modes

- It refers to the way in which the operand of an instruction is specified.

- The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

Immediate mode addressing

- It means the operand is itself part of the instruction.
- Advantage
  - Can be used for constants, where values are already known.
  - Extremely fast, no memory reference is required.
- Disadvantage
  - Cannot be used with variables whose values are unknown at the time of program writing.
  - Cannot be used for large constant whose values cannot be stored in the small part of instruction.
- Application
  - Mostly used when required data is directly moved to required register or memory
  - **Immediate addressing mode:** In this mode data is present in address field of instruction.
  - Designed like one address instruction format.
  - Limitation in the immediate mode is that the range of constants are restricted by size of address field.
  - Immediate- mode instructions are useful for initializing registers to a constant value.

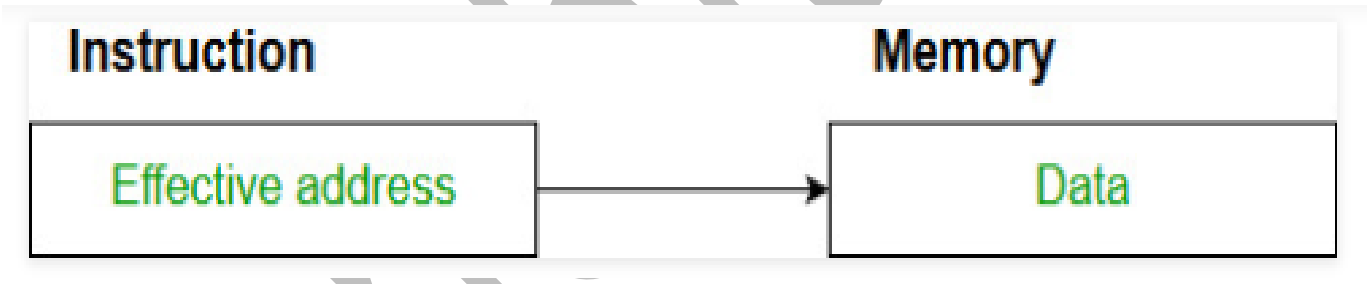


Q A processor has 40 distinct instructions and 24 general purpose registers. A 32-bit instruction word has an opcode, two register operands and an immediate operand. The number of bits available for the immediate operand field is _____. **(Gate-2016) (1 Marks)**

ANSWER 16 BITS

Direct mode addressing (absolute address mode)

- It means instruction contain address of the memory where variable or data is present (effective address).
- Advantage
  - With variable whose values are unknown direct addressing is the simplest one.
  - No restriction on the range of data values and largest value which can system hold can be used in this mode.
  - Can be used to access global variables whose address is known at compile time.
- Disadvantage
  - Relatively slow compare to immediate mode
  - No of variable used are limited
  - Its large calculation it will fail
- **Direct addressing / absolute addressing Mode:** The operand's offset is given in the instruction as an 8 bit or 16 bit displacement element.
- In this addressing mode the 16 bit effective address of the data is the part of the instruction.
- *Here only one memory reference operation is required to access the data.*



Q In the absolute addressing mode **(Gate-2002) (2 Marks)**
(A) the operand is inside the instruction
(B) the address of the operand is inside the instruction
(C) the register containing address of the operand is specified inside the instruction
(D) the location of the operand is implicit
Answer: (B)

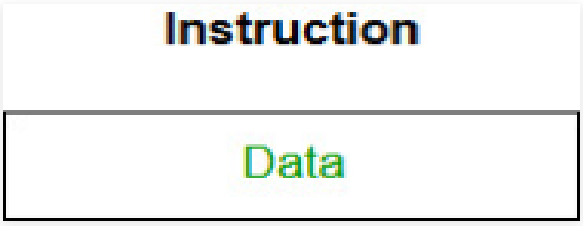
Indirect mode addressing

- Here in the instruction we store the address where the (address of the variable) effective address is stored using which we can access the actual data.
- Advantage
    - No limitation on no of variable or size of variables.
    - Implementation of pointer are feasible and relatively more secure.
- Disadvantage
    - Relatively slow as memory must be referred more than one time.
- **Indirect addressing Mode:** In this mode address field of instruction contains the address of effective address.
- Here two references are required.
- 1st reference to get effective address and 2nd reference to access the data.

Q In the indirect addressing scheme, the second part of an instruction contains: **(NET-DEC-2008)**
(A) the operand in decimal form
(B) the address of the location where the value of the operand is stored
(C) the address of the location where the address of the operand is stored
(D) the operand in an encoded form

Implied mode addressing

- This mode is mostly used with 0-address instruction which means we only tell the operation and system know the location already e.g. Accumulator
- Means the address is already provided and now it is not necessary to provide explicitly
- E.g. increment accumulator.
- ***Implied mode:*** In implied addressing the operand is specified in the instruction itself.
- Zero address instruction are designed with implied addressing mode.



- The instruction "complement accumulator" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction.

Register mode addressing

- It means variable are stored in register of the CPU instead of memory, in the instruction we will give the register no
- Advantage
  - Will be extremely fast as register access time will be less then cache access time.
  - Because no of registers is less, so bits required to specify a register is also less.
- Disadvantage
  - Because no of registers is very less so only with few variables this method can be used
- **Register mode:** In register addressing the operand is placed in one of 8 bit or 16 bit general purpose registers.
- In this mode the operands are in registers that reside within the CPU
- ***Here one register reference is required to access the data.***

Q A machine has a 32-bit architecture, with 1-word long instructions. It has 64 registers, each of which is 32 bits long. It needs to support 45 instructions, which have an immediate operand in addition to two register operands. Assuming that the immediate operand is an unsigned integer, the maximum value of the immediate operand is _____. **(Gate-2014) (1 Marks)**

Answer: (16383)

Register indirect mode addressing

- In this mode in the instruction we will specify the address of the register where inside the register actual memory address of the variable is stored effective address.
- Advantage
  - When same address is required again and again, this approach is very useful
  - Here the same register can be used to provide different data by charging the value of register, i.e. it can reference memory without paying the price of having a full memory in the instruction.
  - In pointer arithmetic, it provides more flexibility compare to register mode.
  - Register indirect mode can further be improved by auto increment and auto decrement command where we read or work on some continuous data structure like array or matrix.

- **Register Indirect mode:** In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.
- The effective address of the data is in the base register or an index register that is specified by the instruction.

- ***Here two register reference is required to access the data.***

Q In _____ addressing mode, the operands are stored in the memory. The address of the corresponding memory location is given in a register which is specified in the instruction. **(NET-Aug-2016)**

(A) Register direct
(B) Register indirect
(C) Base indexed
(D) Displacement

Answer: (B)

Q The advantage of _____ is that it can reference memory without paying the price of having a full memory address in the instruction. **(NET-JUNE-2014)**

(A) Direct addressing
(B) Indexed addressing
(C) Register addressing
(D) Register Indirect addressing

Ans: d

Q Which of the following is/are true of the auto-increment addressing mode? **(Gate-2008) (2 Marks)** **(NET-DEC-2018)**

i) It is useful in creating self-relocating code

ii) If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation

iii) The amount of increment depends on the size of the data item accessed

a) I only **b)** II only **c)** III only **d)** II and III only

Ans: c

Base register (off set) mode

- In multiprogramming environment, the location of the process in the memory keeps on changing from one place to another.
- But if we are using direct address in the process then it will create a problem.
- So, to solve this problem we try to save the starting of the program address in a register called base register. It is a very popular approach in most of the computer.
- Then instead to giving the direct branch address we give off set in the instruction and effective address = base address + off set (instruction)
- Now the advantage is even if we try to shift process in the memory, we only need to change the content of the base register.
- Final address will be the sum of what is given in instruction and given in base register.

- **Auto indexed (Auto increment / decrement mode)**: Effective address of the operand is the contents of a register specified in the instruction.

- Before accessing the operand, the contents of this register are automatically incremented / decremented to point to the next / previous consecutive memory location

- ***Here one register reference, one memory reference and one ALU operation is required to access the data.***

Based Indexed Addressing: In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

Q Consider a hypothetical processor with an instruction of type LW R1, 20(R2), which during execution reads a 32-bit word from memory and stores it in a 32-bit register R1. The effective address of the memory location is obtained by the addition of a constant 20 and the contents of register R2. Which of the following best reflects the addressing mode implemented by this instruction for operand in memory? **(Gate-2011) (1 Marks)**

(A) Immediate Addressing
(B) Register Addressing
(C) Register Indirect Scaled Addressing
(D) Base Indexed Addressing

Answer: (D)

Index addressing mode

- This mode is generally used when CPU has a number of registers, and out of which one can be used an index register
- This mode is especially useful, when we try to access large sized array elements
- Idea is, give the base address of the array in the instruction and the index which we want to access will be there in the register.
- So by changing the index in the index register we can use the same instruction to access the different element in the array.
- Base in present inside the instruction and index is present inside the register

- **Indexed addressing mode**: In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value.
- The address field of the instruction defines the beginning address of a data array in memory.
- Each operand in the array is stored in memory relative to the beginning address.
- The distance between the beginning address and the address of the operand is the index value stored in the index register.

Q Which of the following addressing mode is best suited to access elements of an array of contiguous memory locations? **(NET-NOV-2017)**

(A) Indexed addressing mode
(B) Base Register addressing mode
(C) Relative address mode
(D) Displacement mode

Answer: (A)

Q In which addressing mode, the effective address of the operand is generated by adding a constant value to the contents of register? **(NET-Dec-2012)**

(A) absolute mode
(B) immediate mode
(C) indirect mode
(D) index mode

Ans: d

Q Which type of addressing mode, less number of memory references are required? **(NET-JULY-2019)**

a) Immediate **b)** Implied **c)** Register **d)** Indexed

Ans: b

Q Which of the following addressing modes are suitable for program relocation at run time? **(Gate-2004) (1 Marks)**

(i) Absolute addressing **(ii)** Based addressing

(iii) Relative addressing **(iv)** Indirect addressing

(A) (i) and (iv) **(B)** (i) and (ii) **(C)** (ii) and (iii) **(D)** (i), (ii) and (iv)

Answer: (C)

Q For computers based on three-address instruction formats, each address field can be used to specify which of the following: **(Gate-2015) (1 Marks)**

S_1: A memory operand

S_2: A processor register

S_3: An implied accumulator register

(A) Either S_1 or S_2 **(B)** Either S_2 or S_3

(C) Only S_2 and S_3 **(D)** All of S_1, S_2 and S_3

Answer: (A)

Q Which is the most appropriate match for the items in the first column with the items in the second column: **(Gate-2001) (2 Marks)**

X. Indirect Addressing	**I.** Array implementation
Y. Indexed Addressing	**II.** Writing relocatable code
Z. Base Register Addressing	**III.** Passing array as parameter

a) (X, III), (Y, I), (Z, II) **b)** (X, II), (Y, III), (Z, I)

c) (X, III), (Y, II), (Z, I) **d)** (X, I), (Y, III), (Z, II)

Ans: a

Q The most appropriate matching for the following pairs **(Gate-2000) (1 Marks)**

X: Indirect addressing	**1:** Loops
Y: Immediate addressing	**2:** Pointers
Z: Auto decrement addressing	**3:** Constants

(A) X-3, Y-2, Z-1 **(B)** X-I, Y-3, Z-2 **(C)** X-2, Y-3, Z-1 **(D)** X-3, Y-I, Z-2

Answer: (C)

Q. Match the following: **(NET-JAN-2017)**

Addressing Mode	Location of operand
a. Implied	**i.** Registers which are in CPU
b. Immediate	**ii.** Register specifies the address of the operand.
c. Register	**iii.** Specified in the register
d. Register Indirect	**iv.** Specified implicitly in the definition of instruction

	a	b	c	d
1)	IV	III	I	II
2)	IV	I	III	II
3)	IV	II	I	III
4)	IV	III	II	I

Ans: b

Q Which one of the following is not an addressing mode? **(NET-JUNE-2013)**

(A) Register indirect **(B)** Auto increment

(C) Relative indexed **(D)** Immediate operand

Ans: c

Q Consider the C struct defines below:

```
struct data
{
    int marks [100];
    char grade;
    int cnumber;
};
struct data student;
```

The base address of student is available in register R1. The field student. grade can be accessed efficiently using

(A) Post-increment addressing mode. $(R_1)+$

(B) Pre-decrement addressing mode, $-(R_1)$

(C) Register direct addressing mode, R_1
(D) Index addressing mode, X(R1), where X is an offset represented in 2's complement 16-bit representation.
Answer: (D)

Q Match the following:

List – I	List – II
a. Indexed Addressing	**i.** is not used when an operand is moved from memory into a register or from a register to memory.
b. Direct Addressing	**ii.** Memory address is computed by adding up two registers plus an (optional) offset.
c. Register Addressing	**iii.** Addressing memory by giving a register plus a content offset.
d. Base access Indexed Addressing	**iv.** can only be used to global variables whose address is known at compiler time

Codes:

	a	b	c	d
a)	ii	i	iv	iii
b)	ii	iv	i	iii
c)	iii	iv	i	ii
d)	iii	i	iv	ii

Ans: c

Relative Address Mode: In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

Advantages of Addressing Modes

1. To give programmers to facilities such as Pointers, counters for loop controls, indexing of data and program relocation.
2. To reduce the number of bits in the addressing field of the Instruction.

CISC/RISC

Reduced Instruction Set Computer (RISC)

- Computers that use fewer instructions with simple constructs so they can be executed much faster within the CPU without having to use memory as often are classified as RISC.
- Relatively few instructions
- Relatively few addressing modes
- Memory access limited to load and store instructions
- All operations done within the registers of the CPU
- Fixed-length, easily decoded instruction format
- Single-cycle instruction execution
- Hardwired rather than microprogrammed control
- The small set of instructions of a typical RISC processor consists mostly of register-to-register operations, with only simple load and store operations for memory access.
- Thus, each operand is brought into a processor register with a load instruction.
- All computations are done among the data stored in processor registers.
- Results are transferred to memory by means of store instructions

Complex Instruction Set Computer (CISC)

- A computer with a large number of instructions is classified as a complex instruction set computer, abbreviated CISC.
- A large number of instructions-typically from 100 to 250 instructions
- Some instructions that perform specialized tasks and are used infrequently
- A large variety of addressing modes-typically from 5 to 20 different modes
- Variable-length instruction formats
- Instructions that manipulate operands in memory

Complex instruction set format (CISC)	Reduced instruction set format (RISC)
Large number of instructions (around 1000) Application point of view they are useful but for system designer it is a headache	Relatively Few numbers of instruction, only those instructions which are strictly required, and in case of implementation of a complex function a set of simple instruction will perform together

some instruction is there which perform specific task and are used infrequently for e.g. instructions for testing	few instructions will be there which will be preforming more frequent work
large number of addressing mode, leading to lengthen instruction, but compilation and translation will be faster	number of instruction mode will be less, hardly 3 to 4
variable length instruction format	fixed length easy to decode instruction format
Instruction that manipulate operand in memory	do not perform operation directly in memory, first load them in register and then perform, so all operations will be done with in the registers of CPU
Powerful but costly	Relatively less powerful but cheap
Microprogrammed control unit, relatively slow	Hardwired control unit, so fast

Control Unit Design

- We can classify computers into two class General purpose computers and Embedded systems
- Embedded system are kind of simple computers designed to do Specific task for e.g. microwave, washing machine etc., they have small microprocessors inside them. These microprocessors are hardwired and can not do everything, except specific task.
- A general-purpose computer is a programable computer, where we can change the program and by changing the program, we can accomplish different things using the same machine. here we have a memory where we store a program and just by changing the program, we can change functionality (stored program architecture).
- Now the question is what are the main elements we need
    - Memory (store a program)
    - ALU (circuit which perform operations)
    - Register (fast memory, sequence of flip-flop) (load, clear, increment pin)
    - Timing circuit (sequence counter) (to order certain operations, like fetch, decode, execute), will generate timing signals
    - Control unit will generate control signals-to select registers, select other circuit, to give inputs to registers, So we need a special unit called control unit, which will give signals to all the components
    - Flags – one-bit information
    - Bus – using which we will connect different component together, and perform data transfer using multiplexer

- Registers - The computer needs processor registers for manipulating data and a register for holding a memory address.

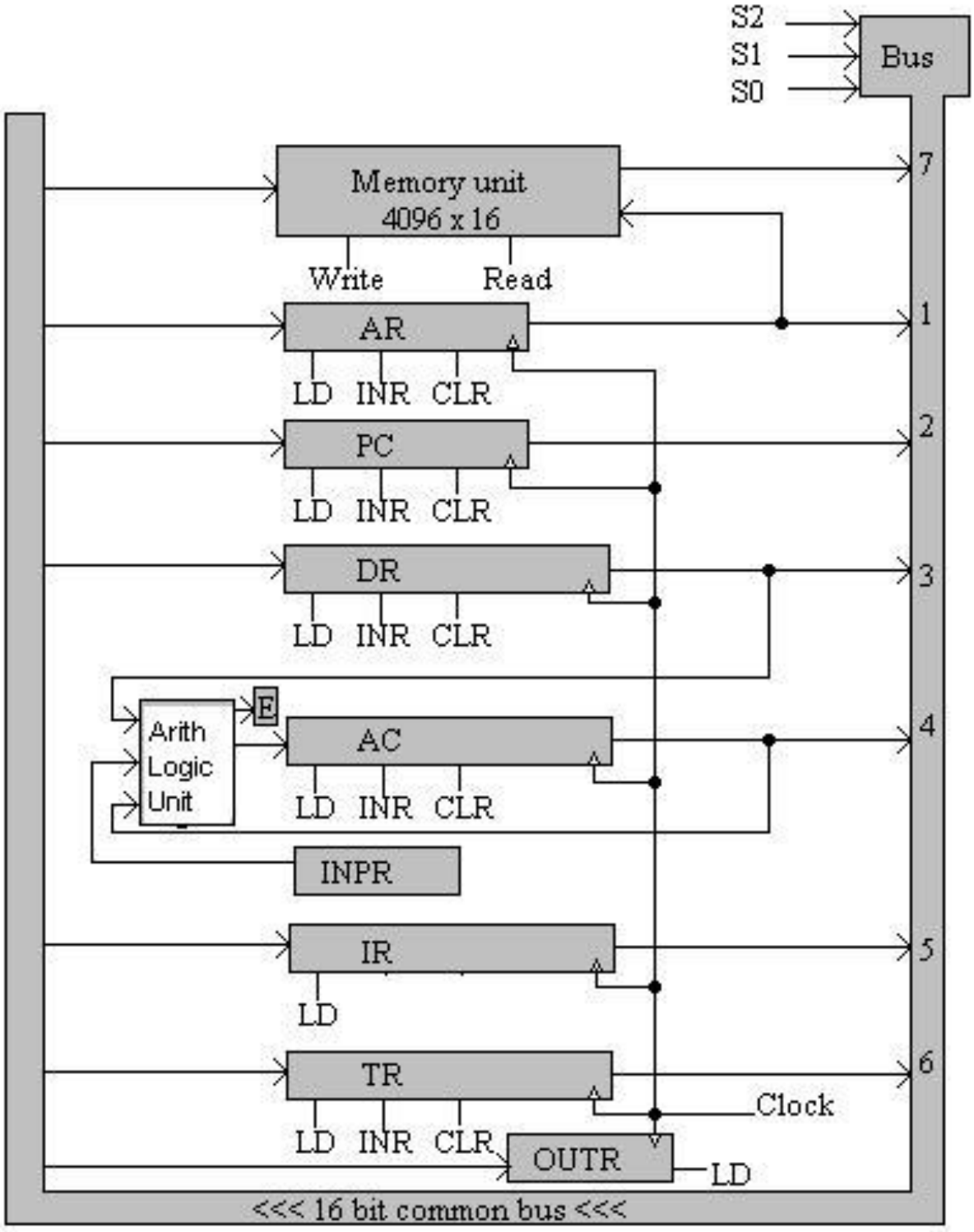
Register symbol	Number of bits	Register name	Function
DR	16	Data register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register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

- The data register (DR) is a register which is used to store operands, which are fetched from memory
- The accumulator (AC) register is a general-purpose processing register. will hold the output of the operation performed in the ALU, as ALU is not directly connected to the memory.
- Instruction register- is used to store instruction which we fetched from memory, so that we analyses the instruction using a decoder and can understand what instruction want to perform.
- The temporary register (TR) is used for holding temporary data during the processing.
- The memory address register (AR) has 12 bits since this is the width of a memory address (always used least significant bits of bus)
- The program counter (PC) also has 12 bits and it holds the address of the next instruction to be read from memory after the current instruction is executed.
- The PC goes through a counting sequence and causes the computer to read sequential instructions previously stored in memory.
- Two registers are used for input and output. The input register (INPR) receives an 8-bit character from an input device, and pass it to ALU then accumulator and then to memory.
- The output register (OUTR) holds an 8-bit character for an output device, screen, printer etc.

Common Bus System

- Since the number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers.
- An efficient scheme for transferring information in a system with many registers is to use a common bus.
- Figure to show a common bus system:
-
- The outputs of seven registers and memory are connected to the common bus.
- The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S2, S1, and S0
- The number along each output shows the decimal equivalent of the required binary selection.
- Example: the number along the output of DR is 3. The 16-bit outputs of DR are placed on the bus lines when $S_2S_1S_0 = 011$.
- The particular register whose LD (load) input is enabled receives the data from the bus during the next clock pulse transition.
- The memory receives the contents of the bus when its write input is activated.
- The memory places its 16-bit output onto the bus when the read input is activated and $S_2S_1S_0 = 111$.
- The input register INPR and the output register OUTR have 8 bits each and communicate with the eight least significant bits in the bus.
- INPR is connected to provide information to the bus but OUTR can only receive information from the bus. There is no transfer from OUTR to any of the other registers.
- Some additional points to notice
- The 16 lines of the common bus receive information from six registers and the memory unit.
- The bus lines are connected to the inputs of six registers and the memory.

Five registers have three control inputs: LD (load), INR (increment), and CLR (clear)



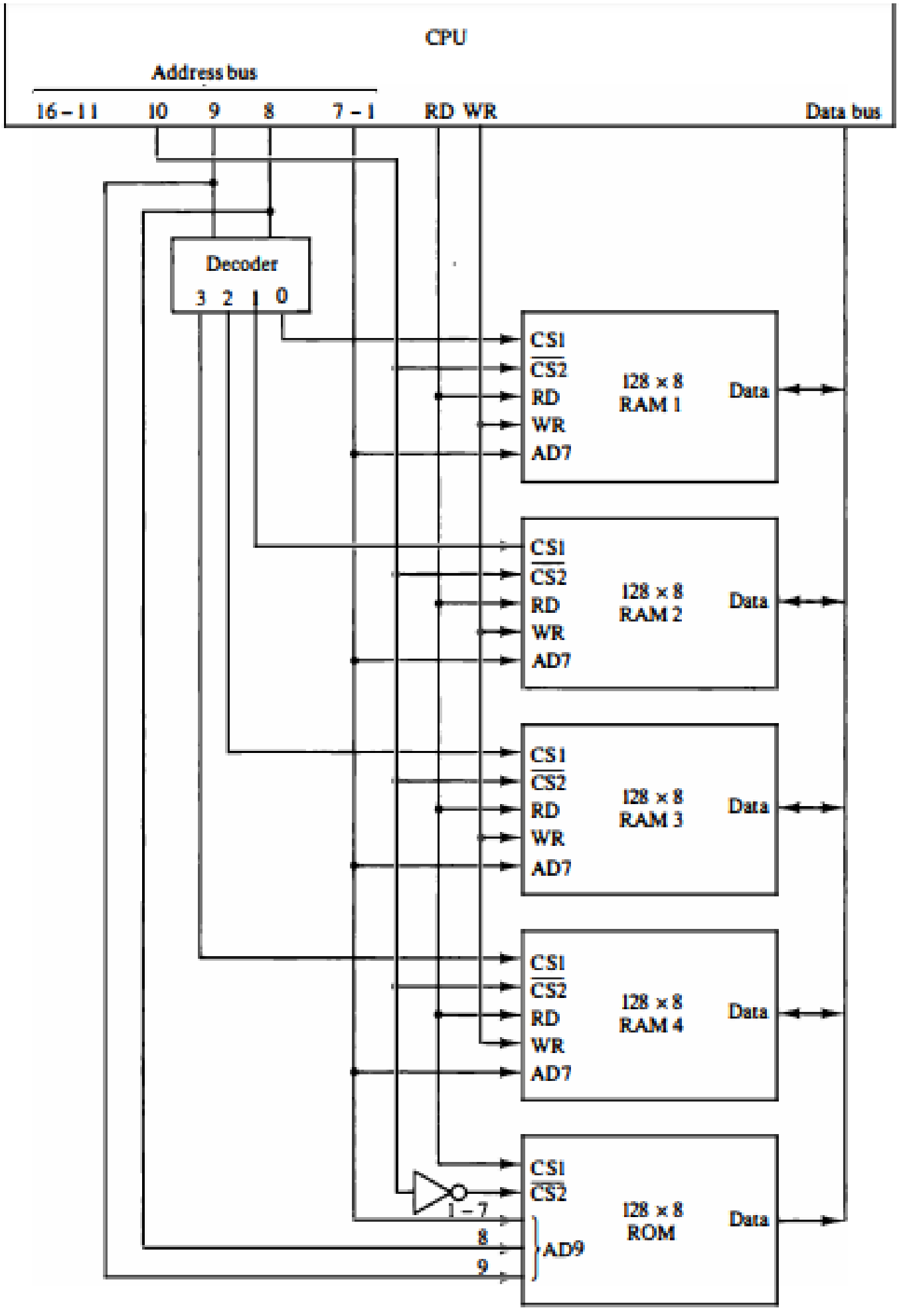
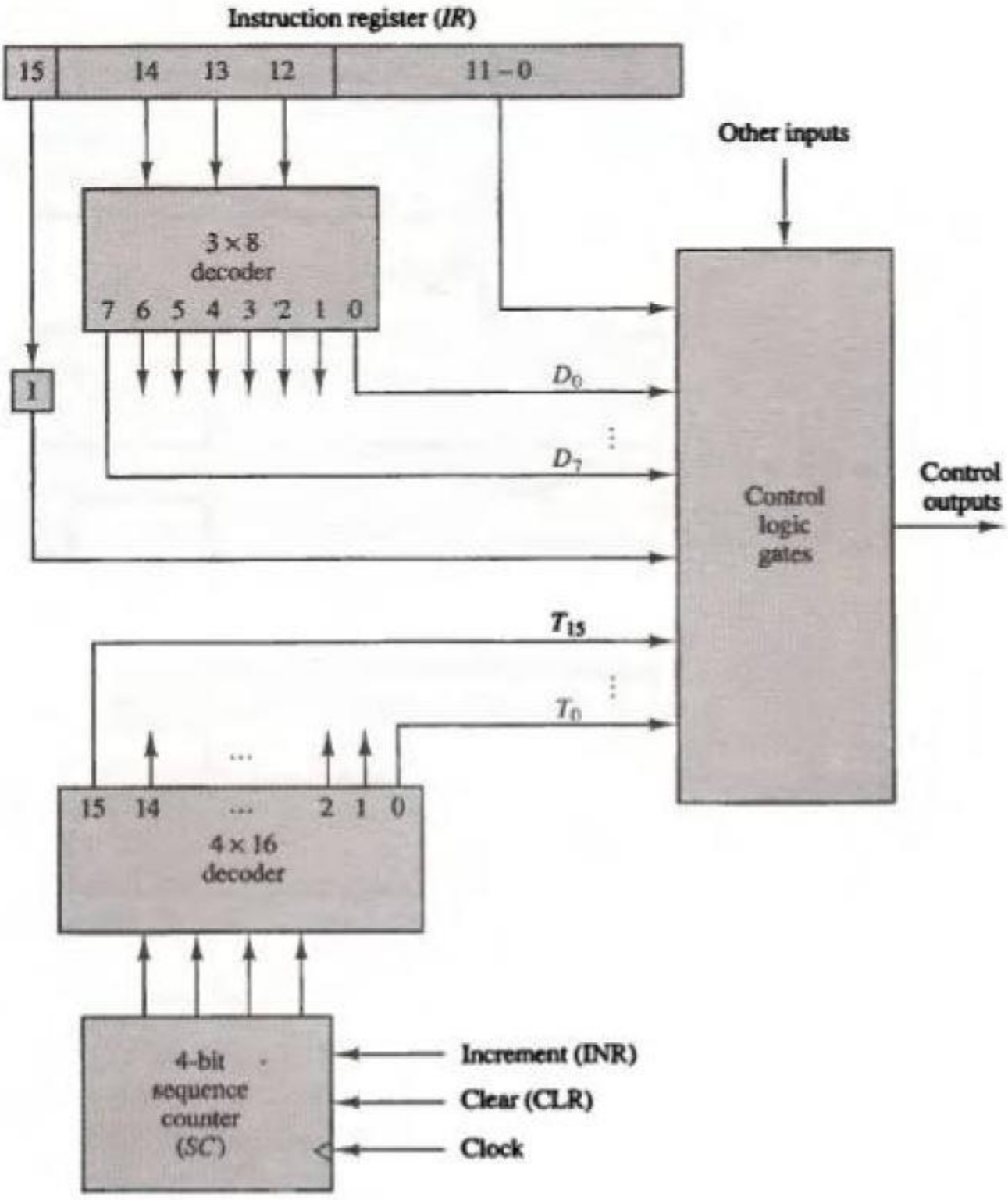


Figure 4 Memory connection to the CPU.

Timing Circuit

- In order to perform a instruction we need to perform a number of micro-operators, and these micro-operations must be timed
    - AR ← PC
    - IR ← M[AR], PC ← PC + 1
- we should distinguish one clock pulse from another during the execution of instruction
- Sequence counter followed by decoder is used to generate time signals


Instruction register (IR)
15
14
13
12
11 – 0
Other inputs
3 × 8
decoder
7 6 5 4 3 2 1 0
1
D_0
D_7
Control
logic
gates
Control
outputs
T_{15}
T_0
15 14 ... 2 1 0
4 × 16
decoder
4-bit
sequence
counter
(SC)
Increment (INR)
Clear (CLR)
Clock

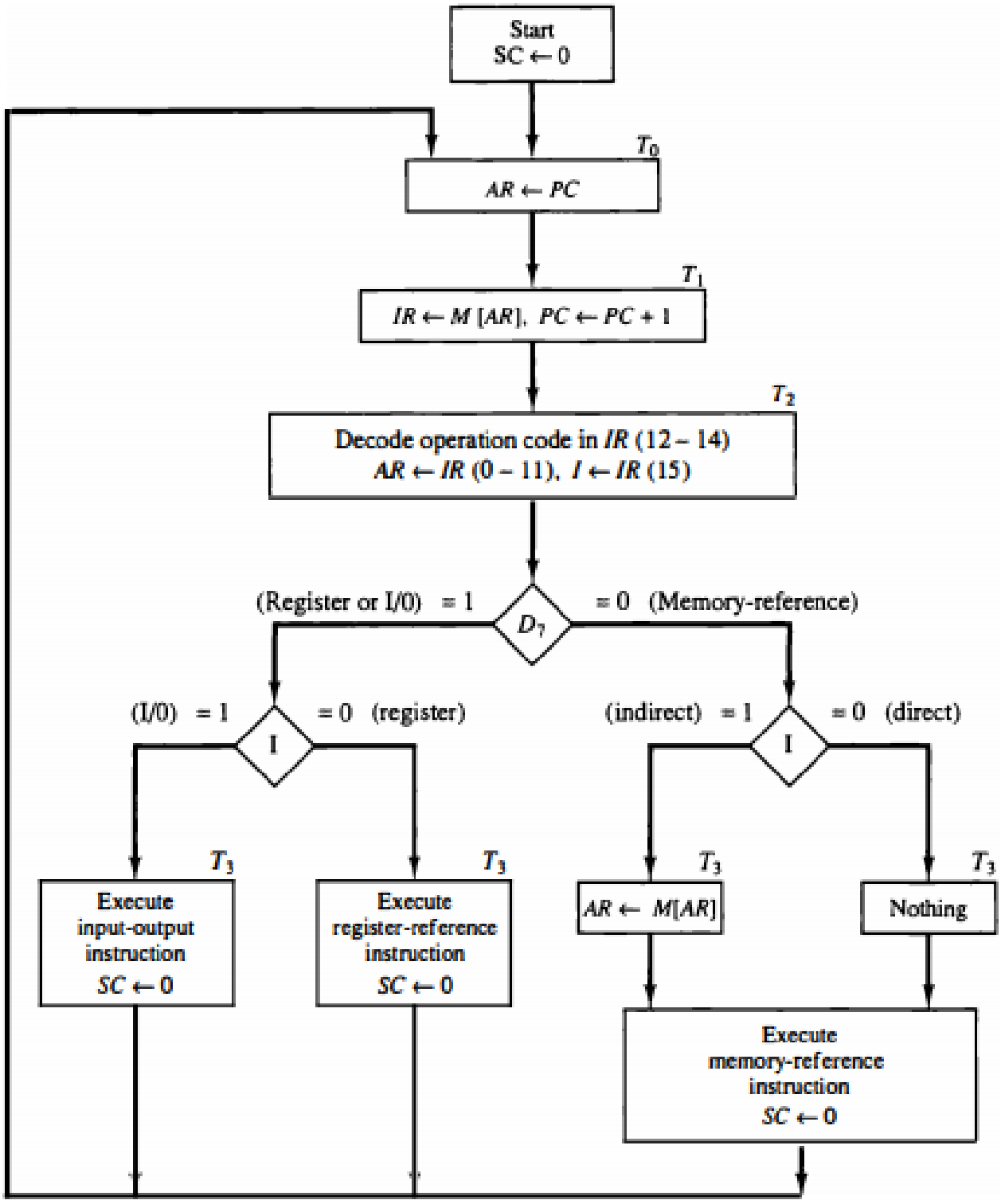
Instruction Cycle - A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases. from fetching of instruction to the completion of execution of instruction whatever happens is called instruction cycle. The reason we called this cycle because it will happen for every instruction.

- Each instruction cycle consists of the following phases:
    - Fetch an instruction from memory.
    - Decode the instruction.
    - Read the effective address from memory if the instruction has an indirect address.
    - Execute the instruction.
    - Fetch and Decode

- Initially, the program counter PC is loaded with the address of the first instruction in the program.
- The sequence counter SC is cleared to 0, providing a decoded timing signal T0.
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statement

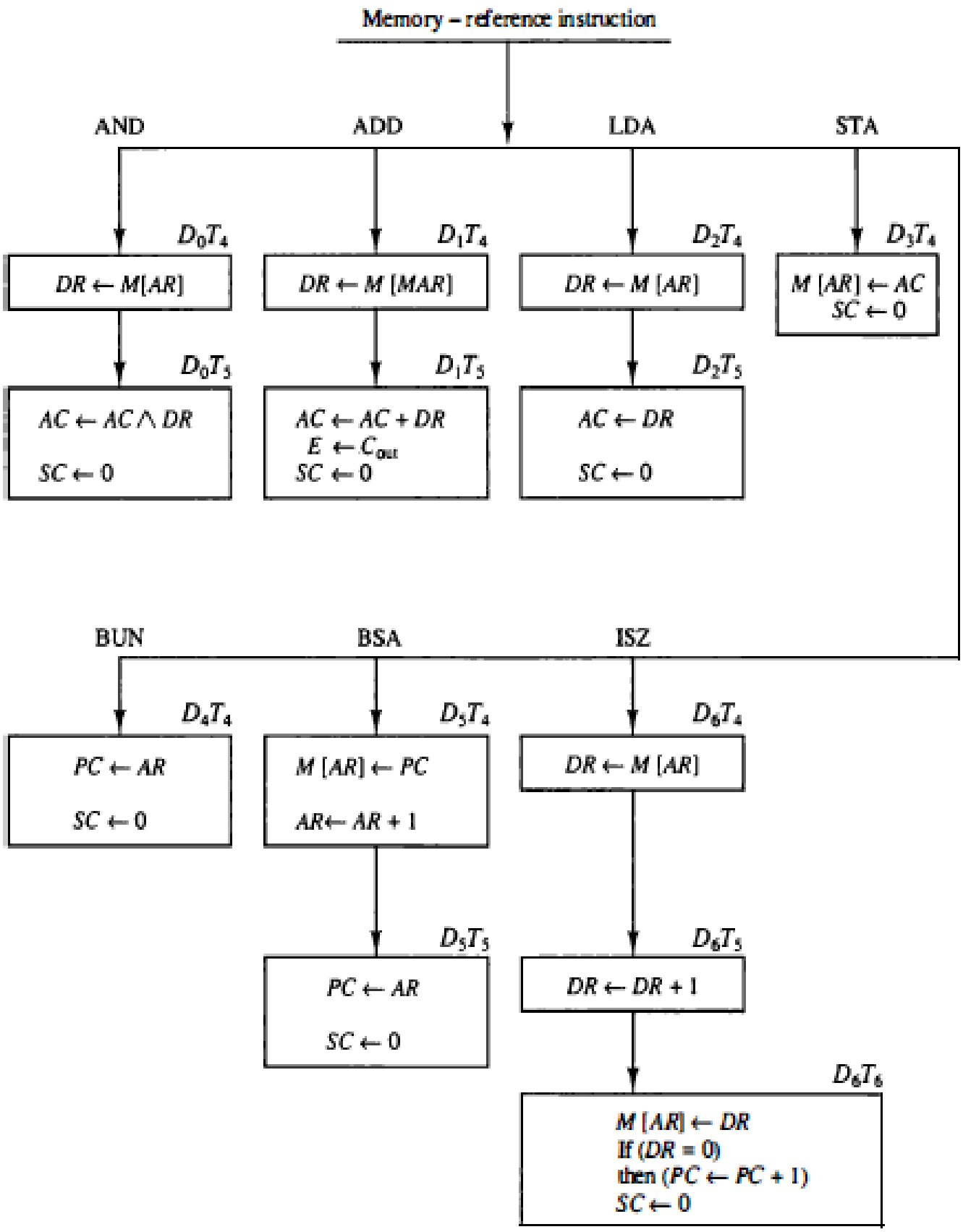
$$T_0: \quad AR \leftarrow PC$$
$$T_1: \quad IR \leftarrow M[AR], \quad PC \leftarrow PC + 1$$
$$T_2: \quad D_0, \ldots, D_7 \leftarrow \text{Decode } IR(12\text{–}14), \quad AR \leftarrow IR(0\text{–}11), \quad I \leftarrow IR(15)$$

- Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T0.
- The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1.
- At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.
- At time T_2, the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.
- SC is incremented after each clock pulse to produce the sequence T_0, T_1, and T_2
- Flow chart for the Instruction Cycle



- Determine the Type of Instruction
- The timing signal that is active after the decoding is T_3.
- During time T_3, the control unit determines the type of instruction that was just read from memory.
- Decoder output D, is equal to 1 if the operation code is equal to binary 111.
- If $D_7 = 1$, the instruction must be a register-reference or input-output type.

- If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.
- Control then inspects the value of the first bit of the instruction, which is now available in flip-flop I. If $D_7 = 0$ and I = 1, we have a memory reference instruction with an indirect address.
- In case of indirect address, it is necessary to read effective address from memory. The microoperation for the indirect address condition can be symbolized by the register transfer statement
- AR ← M [AR]
- When a memory-reference instruction with I = 0 is encountered, it is not necessary to do anything since the effective address is already in AR.
- The sequence counter SC is either incremented or cleared to 0 with every positive clock transition.


Memory – reference instruction
AND
ADD
LDA
STA
D_0T_4
$DR \leftarrow M[AR]$
D_1T_4
$DR \leftarrow M[MAR]$
D_2T_4
$DR \leftarrow M[AR]$
D_3T_4
$M[AR] \leftarrow AC$
$SC \leftarrow 0$
D_0T_5
$AC \leftarrow AC \wedge DR$
$SC \leftarrow 0$
D_1T_5
$AC \leftarrow AC + DR$
$E \leftarrow C_{out}$
$SC \leftarrow 0$
D_2T_5
$AC \leftarrow DR$
$SC \leftarrow 0$
BUN
BSA
ISZ
D_4T_4
$PC \leftarrow AR$
$SC \leftarrow 0$
D_5T_4
$M[AR] \leftarrow PC$
$AR \leftarrow AR + 1$
D_6T_4
$DR \leftarrow M[AR]$
D_5T_5
$PC \leftarrow AR$
$SC \leftarrow 0$
D_6T_5
$DR \leftarrow DR + 1$
D_6T_6
$M[AR] \leftarrow DR$
If $(DR = 0)$
then $(PC \leftarrow PC + 1)$
$SC \leftarrow 0$

- **Memory-Reference Instructions**

Symbol	Operation decoder	Symbolic description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1,$ If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

- The effective address of the instruction is in the address register AR and was placed there during timing signal T2 when I = 0, or during timing signal T3 when I = 1. The execution of the memory-reference instructions starts with timing signal T4.
- Operation of each Instruction

- AND to AC - This is an instruction that performs 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 microoperations that execute this instruction are:

D_0T_4: $DR \leftarrow M[AR]$
D_0T_5: $AC \leftarrow AC \wedge DR, \quad SC \leftarrow 0$

- 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 Cout is transferred to the E (extended accumulator) flip-flop. The microoperations needed to execute this instruction are:

$$D_1T_4: \quad DR \leftarrow M[AR]$$
$$D_1T_5: \quad AC \leftarrow AC + DR, \quad E \leftarrow C_{out}, \quad SC \leftarrow 0$$

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

$$D_2T_4: \quad DR \leftarrow M[AR]$$
$$D_2T_5: \quad AC \leftarrow DR, \quad SC \leftarrow 0$$

- STA: Store 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 microoperation:

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

- BUN: Branch Unconditionally - This instruction transfers the program to the instruction specified by the effective address
- The BUN instruction allows the programmer to specify an instruction out of sequence and we say that the program branches (or jumps) unconditionally. The instruction is executed with one microoperation:

$$D_4T_4: \quad PC \leftarrow AR, \quad SC \leftarrow 0$$

- BSA: Branch and Save Return Address - This instruction is useful for branching to a portion of the program called a subroutine or procedure

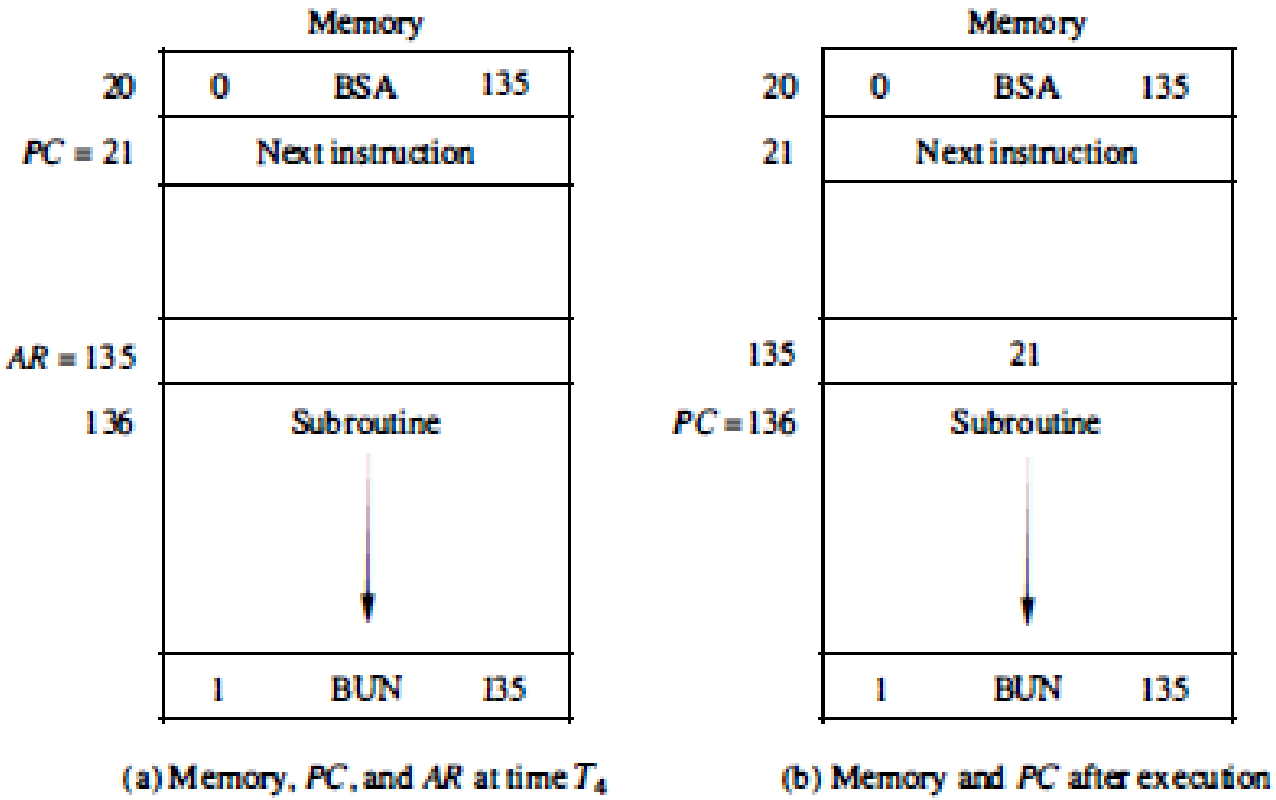
$$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$$

- Example: The BSA instruction is assumed to be in memory at address 20. The I bit is 0 and the address part of the instruction has the binary equivalent of 135. After the fetch

and decode phases, PC contains 21, which is the address of the next instruction in the program (referred to as the return address). AR holds the effective address 135.

$$M[135] \leftarrow 21, \quad PC \leftarrow 135 + 1 = 136$$

-



(a) Memory, PC, and AR at time T_4 (b) Memory and PC after execution

-

- ISZ: Increment and Skip if Zero - This instruction increments the word specified by the effective address, and if the incremented value is equal to 0, PC is incremented by 1.

$$D_6T_4: \quad DR \leftarrow M[AR]$$
$$D_6T_5: \quad DR \leftarrow DR + 1$$
$$D_6T_6: \quad M[AR] \leftarrow DR, \quad \text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), \quad SC \leftarrow 0$$

-
-

Register-Reference Instructions

- Register-reference instructions are recognized by the control when $D_7 = 1$ and I = 0.
- These instructions use bits 0 through 11 of the instruction code to specify one of 12 instructions. These 12 bits are available in IR(0-11).

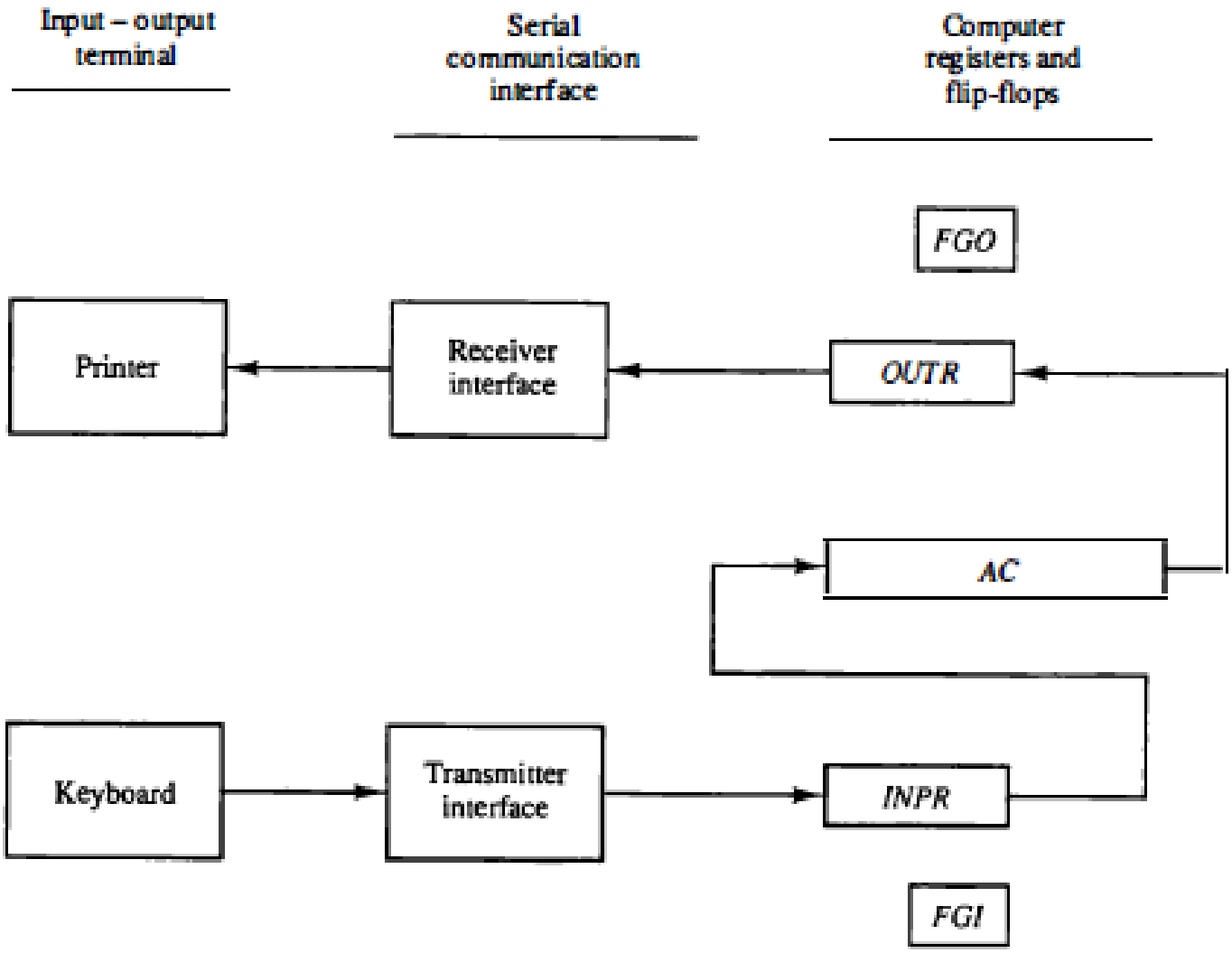
$D_7 I' T_3 = r$ (common to all register-reference instructions)
$IR(i) = B_i$ [bit in $IR(0–11)$ that specifies the operation]

	r:	$SC \leftarrow 0$	Clear SC
CLA	rB_{11}:	$AC \leftarrow 0$	Clear AC
CLE	rB_{10}:	$E \leftarrow 0$	Clear E
CMA	rB_9:	$AC \leftarrow \overline{AC}$	Complement AC
CME	rB_8:	$E \leftarrow \overline{E}$	Complement E
CIR	rB_7:	$AC \leftarrow$ shr AC, $AC(15) \leftarrow E$, $E \leftarrow AC(0)$	Circulate right
CIL	rB_6:	$AC \leftarrow$ shl AC, $AC(0) \leftarrow E$, $E \leftarrow AC(15)$	Circulate left
INC	rB_5:	$AC \leftarrow AC + 1$	Increment AC
SPA	rB_4:	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	Skip if positive
SNA	rB_3:	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	Skip if negative
SZA	rB_2:	If $(AC = 0)$ then $PC \leftarrow PC + 1$	Skip if AC zero
SZE	rB_1:	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	Skip if E zero
HLT	rB_0:	$S \leftarrow 0$ (S is a start–stop flip-flop)	Halt computer

- These instructions are executed with the clock transition associated with timing variable T_3.
- Each control function needs the Boolean relation $D_7 I' T_3$.

Input-Output Configuration

- The terminal sends and receives serial information. Each quantity of information has eight bits of an alphanumeric code.
- The serial information from the keyboard is shifted into the input register INPR.
- The serial information for the printer is stored in the output register OUTR.
- These two registers communicate with a communication interface serially and with the AC in parallel.



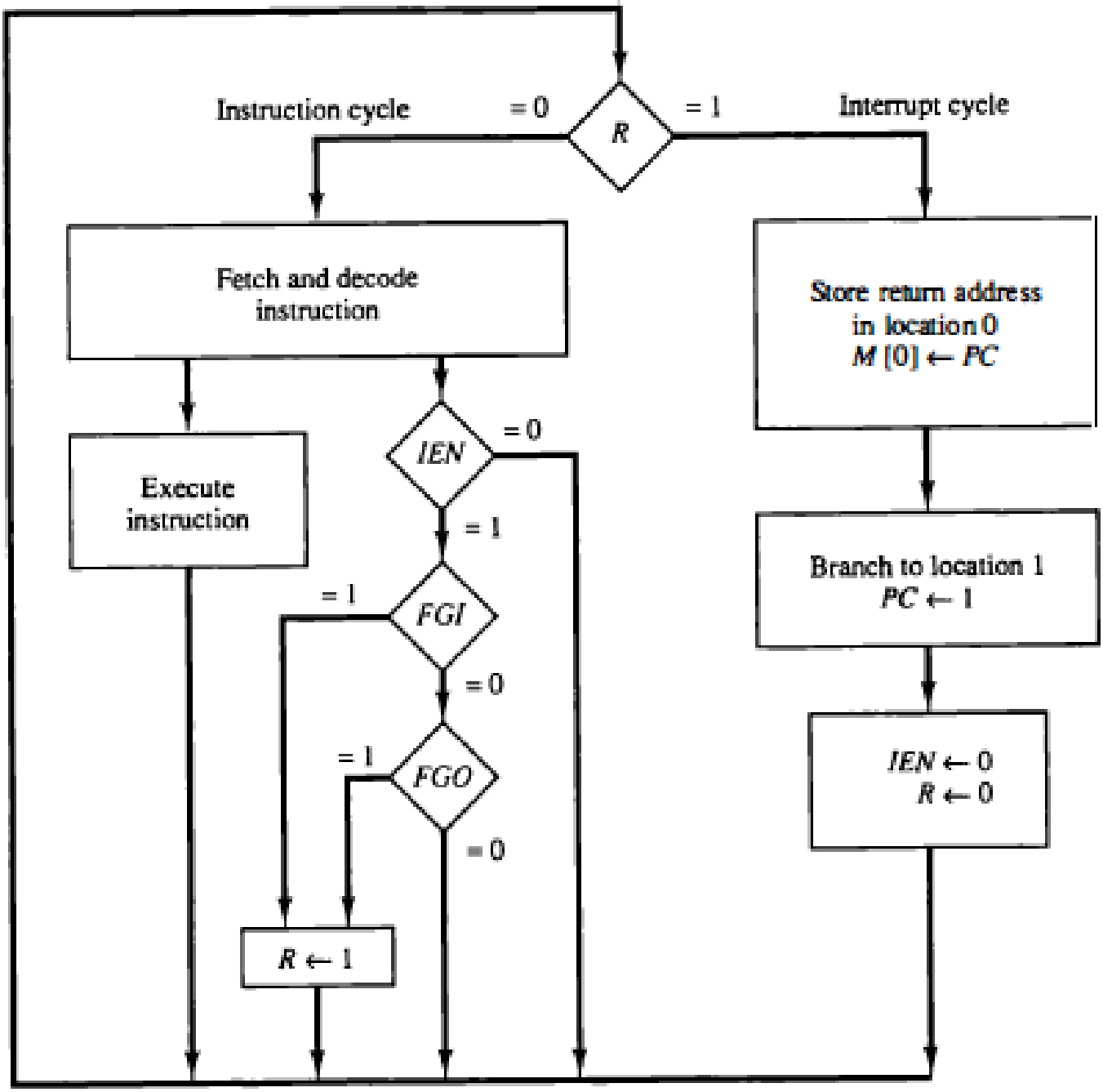
- The transmitter interface receives serial information from the keyboard and transmits it to INPR.
- The receiver interface receives information from OUTR and sends it to the printer serially.
- The 1-bit input flag FGI is a control flip-flop.
- The flag bit is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer.
- The flag is needed to synchronize the timing rate difference between the input device and the computer.

$D_7IT_3 = p$
$IR(i) = B_i$, i = 6, ..., 11

	p:	SC ← 0	Clear SC
INP	pB_{11}:	AC(0-7) ← INPR, FGI ← 0	Input char. to AC
OUT	pB_{10}:	OUTR ← AC(0-7), FGO ← 0	Output char. from AC
SKI	pB_9:	if(FGI = 1) then (PC ← PC + 1)	Skip on input flag
SKO	pB_8:	if(FGO = 1) then (PC ← PC + 1)	Skip on output flag
ION	pB_7:	IEN ← 1	Interrupt enable on
IOF	pB_6:	IEN ← 0	Interrupt enable off

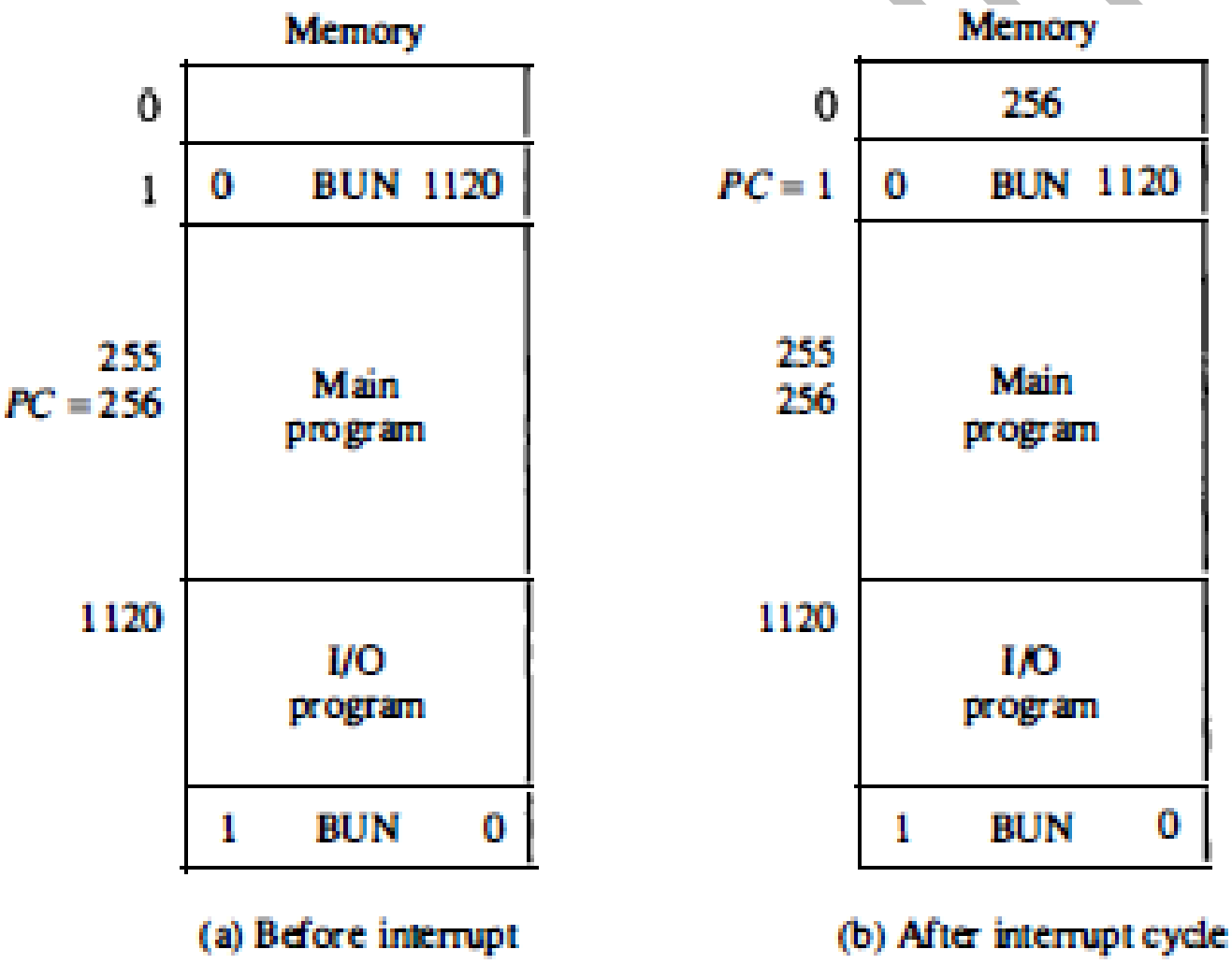
- Program Interrupts
- The computer keeps checking the flag bit, and when it finds it set, it initiates an information transfer.
- The difference of information flow rate between the computer and that of the input-output device makes this type of transfer inefficient.
- While the computer is running a program, it does not check the flags.
- However, when a flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that a flag has been set.
- The computer deviates momentarily from what it is doing to take care of the input or output transfer.

- Interrupt Cycle



- The interrupt enable flip-flop IEN can be set and cleared with two instructions.
- When IEN is cleared to 0 (with the IOF instruction), the flags cannot interrupt the computer.
- When IEN is set to 1 (with the ION instruction), the computer can be interrupted.
- When R = 0, the computer goes through an instruction cycle. During the execute phase of the instruction cycle IEN is checked by the control.
- If it is 0, it indicates that the programmer does not want to use the interrupt, so control continues with the next instruction cycle.

- If IEN is 1, control checks the flag bits. If both flags are 0, it indicates that neither the input nor the output registers are ready for transfer of information. In this case, control continues with the next instruction cycle.
- If either flag is set to 1 while IEN = 1, flip-flop R is set to 1. At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.
- Example: Suppose that an interrupt occurs and R is set to 1 while the control is executing the instruction at address 255. At this time, the return address 256 is in PC. The programmer has previously placed an input-output service program in memory starting from address 1120 and a BUN 1120 instruction at address 1.



- When control reaches timing signal T0 and finds that R = 1, it proceeds with the interrupt cycle.
- The content of PC (256) is stored in memory location 0, PC is set to 1, and R is cleared to 0.
- At the beginning of the next instruction cycle, the instruction that is read from memory is in address 1 since this is the content of PC.

- The branch instruction at address 1 causes the program to transfer to the input-output service program at address 1120.
- This program checks the flags, determines which flag is set, and then transfers the required input or output information.
- Once this is done, the instruction ION is executed to set IEN to 1 (to enable further interrupts), and the program returns to the location where it was interrupted.
- So main task of the computer designer is to connect different part of the computer… and then control them using control signals… so how to read instruction and then generate signals
- But before the actual design designer and programmer decide instruction set format. (machine instruction)
- So first we will discuss addressing mode …and then concentrate on design
- Instructions are divides into mode opcode and operand
- How operands will be read and decoded will be decided by mode
- Like zero address or three address etc. or weather the address is directly of the operand in the memory or it specify address of the register or directly the memory.

Designing of Control Unit

- The Control signal can be generated either by hardware (hardware Control unit design) or by memory + h/w (micro-program control unit design)
- In the hardware control unit design, each control signal is expressed as SOP expression and realized by digital hardware.
- The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as “hardwired”.
- Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
- It is the fastest control unit design and it is suitable for real time application. (The RISC machine employ hardware control unit), Hardwired control is faster than micro-programmed control.
- A controller that uses this approach can operate at high speed.
- RISC architecture is based on hardwired control unit
- Limitation: any modification to the design require re-design & re-connection of H/W component. it is not flexible; hence hardware control unit is not suitable for design & de-bugging environment.
- This problem is resolved using Micro-programmed control unit design.

Q Consider a hypothetical control unit implemented by hardware it uses three, 8-bits register A, B, C. It supports the two instruction I_1 and I_2, the following table gives control signals required for each micro-operation for both instructions?

Micro-operation	Control Signal	
T_1	A_{in}, B_{out}	A_{in}, A_{out}
T_2	B_{in}, C_{out}	C_{in}, A_{out}
T_3	B_{in}, B_{out}	C_{in}, C_{out}
T_4	A_{in}, A_{out}	A_{in}, B_{out}

Micro-Programmed Control Unit

- The idea of microprogramming was introduced by Maurice Wilkes in 1951 as an intermediate level to execute computer program instructions.
- Microprograms were organized as a sequence of *microinstructions* and stored in special control memory.
- The main advantage of the microprogram control unit is the simplicity of its structure. Outputs of the controller are organized in microinstructions and they can be easily replaced.
- In this, the binary patterns of control signals are stored in control memory. After accessing word from the control memory, hardware is used to generate the control signals.
- Any changes can affects only the control word but not the hardware
- The design is flexible & it is used in CISC system.
- Based on the no of words in the control word, the micro-programming may be
    - Horizontal micro-Program
    - Vertical Micro-program
- The horizontal micro-programmed provides higher degree of parallelism & it is suitable in multi-processor system. it requires more bits for control word (1 it for every control signal)
- The vertical micro-programming reduces the size of control words by encoding, control signal pattern before it is stored in control memory. it offers more flexibility than horizontal micro-programming.
- The pattern with vertical-programming is the maximum degree of parallelism is 1(due to the decoder).
- The combinational controlled unit is preferred in most application.

Control word sequencing

- In order to implement, the micro-program, control words are to be sequentially from control memory. One address instruction is used for performing the control word sequencing.

Flag	Control Signal	Address

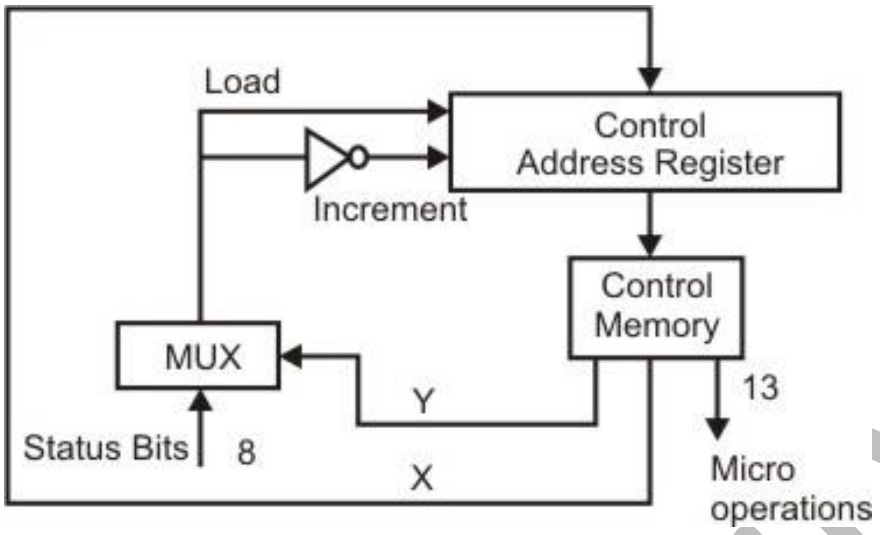
Q Let Micro-programmed control unit has to be support 256 instruction, each of which on average takes 16 micro-operation. The system has to be support 16 flag conditions and generate 48 control signals. if horizontal micro-programming is used
1) how many bits are required for each control word
2) what is the control memory size in byte.

Q A microprogram control unit having the behaviours such that the control signals are divided into 10 mutually exclusive groups. the no of signals for each group is given below

Group	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}
Signal	3	12	7	5	11	11	14	3	5	3

Q Consider a CPU where all the instructions require 7 clock cycles to complete execution. There are 140 instructions in the instruction set. It is found that 125 control signals are needed to be generated by the control unit. While designing the horizontal microprogrammed control unit, single address field format is used for branch control logic. What is the minimum size of the control word and control address register? **(Gate-2008) (2 Marks)**
(A) 125, 7 **(B)** 125, 10 **(C)** 135, 7 **(D)** 135, 10
Answer: (D)

Q The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields: a micro-operation field of 13 bits, a next address field (X), and a MUX select field (Y). There are 8 status bits in the inputs of the MUX.



How many bits are there in the X and Y fields, and what is the size of the control memory in number of words? **(GATE-2004) (2 Marks)**

(A) 10, 3, 1024 **(B)** 8, 5, 256 **(C)** 5, 8, 2048 **(D)** 10, 3, 512

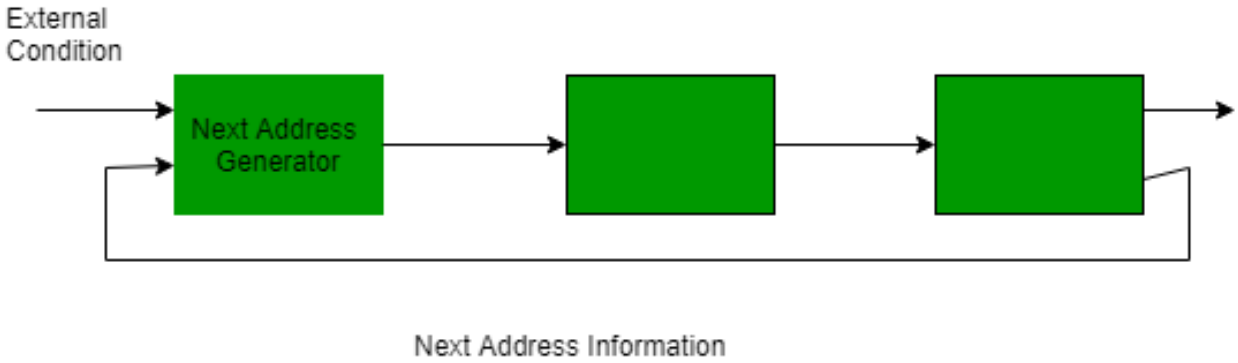
Answer: (A)

Q A micro program control unit is required to generate a total of 25 control signals. Assume that during any micro instruction, at most two control signals are active. Minimum number of bits required in the control word to generate the required control signals will be: (GATE-1996) (2 Marks)

a) 2 **b)** 2.5 **c)** 10 **d)** 12

Ans: c

Q The general configuration of the micro-programmed control unit is given below:



What are blocks B and C in the diagram respectively?(**NET-JAN-2017)**

(A) Block address register and cache memory

(B) Control address register and control memory

(C) Branch register and cache memory

(D) Control address register and random access memory