26. (*b*) 27. (*a*)

CONCEPTUAL SHORT QUESTIONS WITH ANSWERS

1. What are the different types of parallelism ?

Ans. Parallelism is of two types :

(*a*) **Hardware parallelism.** It is built into the machine's architecture.

(*b*) **Software parallelism**. It is exploited by the concurrent execution of machine language instructions in a program as coded by a programmer.

2. Distinguish between temporal and Data Parallel processing ?

Ans. The following table summarizes the differences between the two :

Temporal Parallelism	Data Parallelism
1. In this parallelism, the job is divided into a set of independent tasks and tasks are assigned for processing.	1. Full jobs are assigned for processing.
2. Tasks should take equal time.	2. Jobs may take different times.
3. Pipeline stages should be synchronized.	3. No need to synchronize beginning of jobs.
4. Bubbles in jobs may lead to idling of PEs.	4. Bubbles in jobs do not cause idling of processors.
5. PEs are specialized to perform tasks.	5. PEs are general purpose.

3. Give some examples of India's supercomputers.

Ans. Some examples of India's Supercomputers are :

1. **PARAM** 10000 (by CDAC ; in 1998). It uses Sun Solaris OS for parallel programming. The peak speed is 10G flops.
2. PACE + by DRDO ; in 1997). It was ANUPAM programming Environment with FORTRAN and C languages.
3. **ANUPAM Model–2** (by BARC ; in 1997). It uses DEC Alpha processors with DEC UNIX environment.

4. How is bit serial memory accessing different from bit parallel accessing ?

Ans. The first electronic digital computers used a bit serial MM. Each bit of a word was read individually from memory. ***For example*** : EDSAC, Pilot ACE, EDVAC and UNIVAC systems (all of them had bit serial memories).

Bit-parallel arithematic became possible once bit-parallel memory was available. *For example :* IBM–701 machine used bit parallel arithematic.

5. What do you mean by Inter leaved memory ? Distinguish between low-order and high-order inter leaving.

Ans. An interleaved memory is a memory unit divided into a number of modules or banks that can be accessed simultaneoulsy. Each memory bank has its own addressing circuitry. **Instruction and data addresses** are interleaved to take advantage of the **parallel fetch capability**. It can be done in two ways.

(*a*) **Low-order interleaving.** This method uses low-order bits of an address to determine the memory bank containing the address : For example

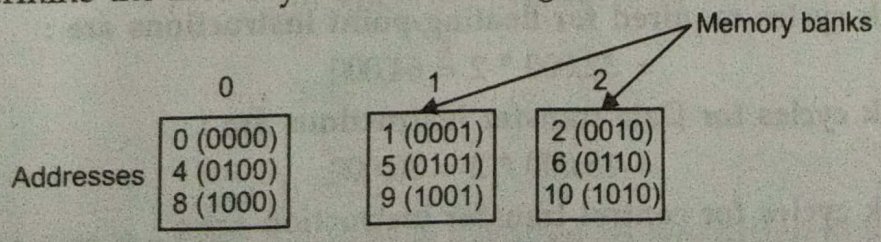


Fig (a). Low-order interleaving lets the low-order address determine the memory bank

(*b*) **High-order interleaving :** This method uses the high-order bits of an address to determine the memory bank.

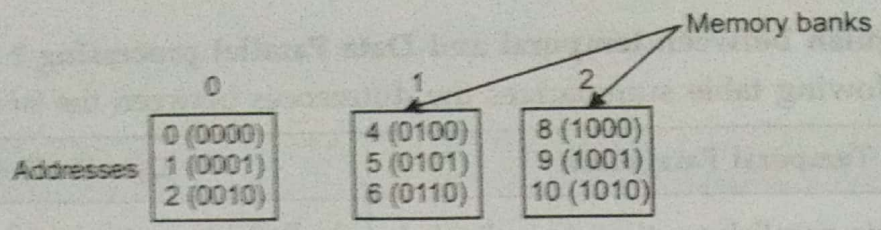


Fig (b) : High-order interleaving lets the high-order address determine the memory bank.

Note :

1. During computer designing, it is very important to match the speed of the various components.
2. IBM STRETCH computer was the first computer to have an interleaved memory.
3. The advantage of having two memory banks was that the maximum data transfer rate to and from the memory was increased by a factor of 2 (2-times)

6. A program is run on a 40MHZ CPU with the instruction mix and corresponding clock cycle count as given in the table below. Determine

(a) Effective CPI.

(b) Execution time.

(c) MIPS rate for the program. **[UPTU, B. Tech (CSE) 7th Sem.; 2003-04] and [GGSIPU, M.Tech (IT)-1st sem., Jan-2011**

The table is shown below :

Instruction Type	Clock cycle Count	Instruction Count
1. Integer Arithematic	1	45,000
2. Floating point	2	32,000
3. Data transfer	2	15,000
4. Control Transfer	2	8,000

Ans. As we know that :

$$CPI = \frac{\text{Total clock cycles requires for given } I_c}{\text{Frequency of clock}}$$

Corresponding to the values in the table given above, we find that :

1. **Clock cycles required for Integer Arithematic instructions are :**
   $= 45,000 * 1 = 45,000$
2. **Clock cycles required for floating-point instructions are :**
   $= 32,000 * 2 = 64,000$
3. **Clock cycles for Data transfer instructions are :**
   $= 15,000 * 2 = 30,000$
4. **Clock cycles for control transfer instruction are :**
   $= 8,000 * 2 = 16,000$

$\therefore$ Total clock cycles = 1,55,000

Also, total number of instructions

$$(I_c) = (45,000 + 32,000 + 15,000 + 8,000)$$
$$= 1,00,000$$

$\therefore \quad I_c = 1,00,000$

(*a*) **Effective CPI** $= \dfrac{1,55,000}{1,00,000}$

$= 1.55$ cycles/instruction

(*b*) **Execution time for the program**

$= \dfrac{\text{Total clock cycle}}{\text{clock period}}$

$= \dfrac{1,55,000}{40\times10^6}$

$= 3.87$ msec.

(*c*) **MIPS** $= \dfrac{\text{Clock Rate}}{\text{CPI} \times 10^6}$

$= \dfrac{40\times10^6}{1.55\times10^6}$

$= 25.8$ MIPS

7. Calculate the average CPI when the following conditions are given :

Instruction Types	Cycles	Relative Frequency
I	2	30%
II	3	25%
III	5	40%
IV	10	5%

Ans. As already explained, in this chapter that —

CPI = No. of cycles/instruction

= cycles * Relative Frequency

$\therefore \quad CPI_I = 2 \times 0.30 = 0.60$ [From table above]

$CPI_{II} = 3 \times 0.25 = 0.75$

$CPI_{III} = 5 \times 0.40 = 2.0$

$CPI_{IV} = 10 \times 0.05 = 0.5$

$\therefore$ Average $\quad CPI = \dfrac{CPI_I + CPI_{II} + CPI_{III} + CPI_{IV}}{4}$

$= \dfrac{(0.60+0.75+2.0+0.50)}{4}$

$= \dfrac{3.85}{4}$

$= 0.9625$

8. Suppose that the same program is executed on two different machines–A and B. Compilers and data set on both the machines are identical. Given that :

Machine–A : Average CPI = 3.5

cycle time = 1.0 ns

Machine–B : Average CPI = 1.2

cycle time = 0.5 ns

Which out of these two machines is slower ? **[GGSIPU, B.Tech.(CSE) –7th sem-, 2009]**

Ans. *Machine–A* : CPI * cycle time
= 3.5 * 1.0
= 3.5

Machine–B : CPI * cycle time
= 1.2 * 0.5
= 0.6

∴ Machine–B is faster as it has a lower CPI.

9. How are performance and execution time related to each other ?

Ans. Performance $\propto \frac{1}{\text{Execution time of CPU}}$

So, a computer that executes a program in half the time taken by another machine has twice the performance. **Please note that the performance and execution time vary in opposite directions. Also note that doubling processor performance does not mean replacing an xGHz CPU with a $2x$ GHz CPU.**

10. Distinguish between Implicit and Explicit parallelism ? **[GGSIPU, B.Tech (CSE) –7th sem., 1st minors, 2012]**

Ans.

Implicit Parallelism	Explicit Parallelism
1. To achieve implicit parallelism, the source code is written in sequential languages like C, FORTRAN 77, LISP or PASCAL.	1. To achieve explicit parallelism, the source code is written in the concurrent dialects of C, FORTRAN, LISP or PASCAL.
2. We need a parallelizing compiler.	2. Here, we need a concurrency preserving compiler.
3. A parallel object code is generated.	3. A concurrent object code is generated.
4. This approach has been used in the programming of shared–memory multiprocessors.	4. This approach has been used in multicomputer development.
5. The compiler must be able to detect parallelism.	5. The compiler need not detect parallelism as it is explicitly specified in the user programs.
Examples :	*Examples :*
(*a*) David Kuck of University of Illinois adopted implicit–parallelism approach.	(*a*) Charles Seitz of California Institute of Technology used this approach in multicomputers development.
(*b*) Ken Kennedy of Rice University used this approach.	(*b*) William Dally of MIT also used this approach.

11. Consider the execution of an object code with 200,000 instructions on a 40 MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles (CPI) needed for each instruction type are given below based on the result of a program trace experiment :

Instruction Type	CPI	Instruction mix
1. Arithematic and logic	1	60%
2. Load/store with cache hit	2	18%
3. Branch	4	12%
4. Memory reference with cache miss	8	10%

(*a*) **Calculate the average CPI when the program is executed on a uniprocessor.**
(*b*) **Also calculate the corresponding MIPS rate ?**

[GGSIPU, B.Tech, (CSE) –7th Sem., 2009]
and [GGSIPU, M.Tech (IT) –7th Sem., 2007]

Ans. (*a*) Average CPI = $1 \times 0.6 + 2 \times 0.18 + 4 \times 0.12 + 8 \times 0.1$
= 2.24 cycles/instruction

(*b*) MIPS rate = $\frac{40}{2.24}$ = 17.86 MIPS

12. A CPU renders a graphic image in 100 ms with graphics card-A and 125 ms with graphics card-B. What is the speed up?

Ans. Speed-up (*s*) = $\frac{125}{100}$ = 1.25
= 25%.

13. Explain I_C and CPI. Why we calculate average value of CPI. Give an example to illustrate.

Ans. IC - Instruction Count : **is defined as the total number of instructions in a program.**

CPI - Clock cycles per Instruction: **It shows, on average, how many cycles are needed for one instruction.**

We calculate an average value of **CPI as CPI is not a constant value.** It varies from instruction to instruction.

For example,

(*a*) Memory access takes more time than register access.

(*b*) Floating point operations take more time than integer operations.

(*c*) Multiplication is more time consuming than addition.

Say, if a RISC machine performs 9 instructions per 10 clock cycles then its average CPI is 1-11 clock periods per instruction.

On the other hand, a CISC machine that performs 4 instructions per 10 clock cycles has an average CPI of 2.5 clock periods per instructions.

Formula to calculate CPI—

Total clock cycles = $\Sigma \text{ CPI}_C * (\#\text{ Instructions})_C$

$$= \begin{bmatrix}\text{Cycles per instruction} \\ \text{for instruction class } C\end{bmatrix} * \begin{bmatrix}\#\text{ instructions of} \\ \text{instruction class } C\end{bmatrix}$$

and Execution Time = (*A* instruction * *X* cycles) +
(*B* instruction * *Y* cycles) +
(C instruction * Z cycles)

where *A*, *B* and *C* are number of different types of instructions
and *x*, *y*, *z* are no. of cycles needed for each type of instruction.

14. Consider the following data for 2-machines *A* and *B*:

	# k Register Instructions (1 cycle each)	# k Memory Access Instructions (2 cycles each)	# k floating point Instructions (3 cycles each)
Machine–A	2	1	2
Machine–B	4	1	1

Compare *A* with *B* in terms of CPI and clock cycles.

Ans. Machine–A

$$\text{Clock-Cycles } (A) = 2 * 1 + 1 * 2 + 2 * 3$$
$$= 2 + 2 + 6$$
$$= \mathbf{10 \text{ cycles}}$$

$$\therefore \quad \text{CPI } (A) = \frac{10 \text{ cycles}}{5 \text{ instructions}} = \mathbf{2}$$

Machine–B

$$\text{Clock-Cycles } (B) = 4 * 1 + 1 * 2 + 1 * 3$$
$$= 4 + 2 + 3$$
$$= \mathbf{9 \text{ cycles}}$$

$$\therefore \quad \text{CPI } (B) = \frac{9 \text{ cycles}}{6 \text{ instructions}} = \mathbf{1.5}$$

Conclusions

Program-A requires fewer instructions but more cycles.
⇒ Higher CPI
and Program-B requires more instructions but fewer cycles.
⇒ Lower CPI.

15. What happens when the clock period changes per computer?

Ans. Say, if clock period = 700 MHz

$$\text{Then} \quad \text{Cycle Time} = \frac{1}{\text{clock-rate}}$$
$$= \frac{1}{700 \text{ MHz}}$$
$$= \frac{1}{700,000,000 \text{ clock periodes per second}}$$
$$= \mathbf{1.4 \text{ ns}}$$

And if clock period = 450 MHz

$$\text{Then} \quad \text{Cycle Time} = \frac{1}{450 \text{ MHz}} = \frac{1}{450,000,000 \text{ clock periodes per second}}$$
$$= \mathbf{0.2 \text{ ns.}}$$

But please remember that if CPI and I_C are same then the clock rate determines which computer is faster.

$\therefore$ Execution Time = I_C * CPI * t

Also note that clock rate = cycles /second
and 1 Hertz = 1 cycle/sec
$\therefore$ 1 Mega Hertz (MHz) = $1 * 10^6$ cycles/second.

16. A hardware manufacturer company increases the frequency from 700 MHz to 1.2 GHz. What is the potential speed-up achieved?

Ans. We know that

$$\text{Execution Time} = I_C * \text{CPI} * t$$

Given: $t_{old} = \frac{1}{700}$ $t_{new} = \frac{1}{1200}$

$$\therefore \quad \text{Speed-up} = \frac{t_{old}}{t_{new}}$$

$$= \frac{I_C * \text{CPI} * \frac{1}{700}}{I_C * \text{CPI} * \frac{1}{1200}} = \frac{1}{700} \times 1200$$

$$= 1.71428$$

$\therefore$ Speed-up (%) = 71%.

17. Machine-A has a clock cycle time of 1 ns and an average CPI = 2. Machine-B has a clock cycle time of 2 ns and an average CPI = 1.2. Instruction count (I_C) is same which of the two machines is faster?

Ans. Execution Time $(A) = I_C * 2 * 1$ ns
$= 2\ I_C\ ns$
Execution Time $(B) = I_C * 1.2 * 2$ ns
$= 2.4\ I_C$ ns

$$\therefore \quad \frac{\text{Performance-}A}{\text{Performance-}B} = \frac{\text{Execution Time B}}{\text{Exeuction Time A}}$$

$$= \frac{2.4\ I_C\ (ns)}{2\ I_C\ (ns)}$$

$$= 1.2$$

$\therefore$ Machine-A is 1.2 times faster then B.

18. Consider the following data:

Which of the two computers appear faster if it is used to run program-x and program-y:

	Time on Comp-A	Time on Comp-B
Program-x	1	10
Program-y	1000	100

(a) 50% of the time.

(b) 90% and 10% of the time resp.

When would computer-*A* appear faster than computer-*B*?

Ans. (*a*) If the given two programs x and y are used 50% of the time then the usage is:

$$\text{Comp}_A = (0.5) * 1 + (0.5) * 1000$$
$$= 500.5$$
$$\text{Comp}_B = (0.5) * 10 + (0.5) * 100$$
$$= 55$$

$\therefore$ Computer-*B* appears faster than Computer-*A*.

(*b*) If the two programs are used 90% and 10% of the time then, usage is:

$$\text{Comp}_A = (0.9) * 1 + (0.1) * 1000$$
$$= 100.9$$
$$\text{Comp}_B = (0.9) * 10 + (0.1) * 100$$
$$= 19$$

$\therefore$ Computer-*B* is still faster.

Now, Comp_A will run faster than Comp_B if and only if:

$$1\,n + 1000\,(1 - n) < 10\,n + 100\,(1 - n)$$

or $(1000 + 1\,n - 1000\,n) < (10\,n - 100\,n + 100)$

or $900 < 909\ \text{n}$

or $\dfrac{900}{909} < n$

or $n > 0.9900990099$

$\therefore$ $\text{Comp}_A = 0.99\ (1) + 0.01\ (1000) = 10.99$

$\text{Comp}_B = 0.99\ (10) + 0.01\ (100) = 10.9.$

3. **Total cost of memory hierarchy (C_{total}) :** It is given by the fc

$$C_{total} = \sum_{i=1}^{n} C_i \cdot S_i$$

It means that the cost is distributed over n levels. Because $C_1 > C_2 > C_3$ so we have to choose $S_1 < S_2 < S_3 < \ldots S_n$.

An **ideal design** of memory hierarchy should have following parameters :

- $T_{eff} \approx t_1$ (of M_1).
- $C_{total} \approx C_n$ (of M_n).

Practically, this is not possible due to the tradeoffs among n–levels.

Our problem now is to minimize

$$T_{eff} = \sum_{i=1}^{n} f_i \cdot t_i$$

and the constraints are :

$$\left.\begin{matrix} S_i > 0 \\ t_i > 0 \end{matrix}\right\} \text{where } i \leftarrow 1 \text{ to } n$$

So, $$C_{total} = \sum_{i=1}^{n} C_i S_i < C_0.$$

We need to manage the tradeoffs among t_i, C_i, S_i and f_i (or h_i) at all levels ($i \leftarrow 1$ to n).

We are in a position to solve a problem now.

Example 1. *Consider a three level memory hierarchy ($M_1 - M_3$) as shown below :*

Memory level	*Access time*	*Capacity*	*Cost/KB*
M_1 *(cache)*	$t_1 = 25$ *ns*	$S_1 = 512$ *KB*	$C_1 = \$1.25$
M_2 *(MM)*	$t_2 = ?$	$S_2 = 32$ *MB*	$C_2 = \$0.2$
M_3 *(Disk)*	$t_3 = 4$ *ms*	$S_3 = ?$	$C_3 = \$0.0002$

It is desired that :

$$T_{eff} = 10.04\ \mu s$$
$$h_1 = 0.98 \text{ (for } M_1)$$
$$h_2 = 0.9\dot{} \text{ (for } M_2).$$

But $C_{total} \leq \$15000$.

Find S_3, t_2 *(marked as ? in table) ?*

Solution. We know that the cost of memory hierarchy is given by the formula :

$$C = C_1S_1 + C_2S_2 + C_3S_3 \leq 15000$$

or $1.25\ (512) + 0.2\ (32) + 0.0002\ (S_3) = 15000$

or $$S_3 = \textbf{39.8 Gbytes}$$

Now

$$T_{eff} = h_1t_1 + (1 - h_1)h_2t_2 + (1 - h_1)\ (1 - h_2)h_3t_3 \leq 10.04$$

or $$10.04 \times 10^{-6} = 0.98 \times 25 \times 10^{-9} + 0.02 \times 0.9 \times t_2 + 0.02 \times 0.1 \times 1 \times 4 \times 10^{-3}$$

or $$t_2 = \textbf{903 ns.}$$

12. CPU generates :
    (*a*) Virtual addresses
    (*b*) Physical addresses
    (*c*) Login addresses
    (*d*) None of the above.
13. MMU stands for :
    (*a*) Main Memory Unit
    (*b*) Memory Management Unit
    (*c*) Memory to Memory Unit
    (*d*) None of the above.
14. TLB and PTs are :
    (*a*) Cache memories
    (*b*) Translation maps
    (*c*) Auxiliary memory
    (*d*) None of the above.
15. Page fault rate (PFR) is given by :
    (*a*) $PFR = \frac{\text{No. of page faults}}{\text{No. of bits in reference string}}$
    (*b*) PFR = No. of page faults only
    (*c*) PFR = No. of page faults * No. of bits in reference string.
    (*d*) None of the above.
16. In which of the following policies, Belady's anomaly occurs :
    (*a*) FIFO
    (*b*) LRU
    (*c*) LFU
    (*d*) NRU

:: ANSWERS ::

1. (*a*)	2. (*c*)	3. (*a*)	4. (*c*)	5. (*a*)
6. (*c*)	7. (*c*)	8. (*a*)	9. (*b*)	10. (*b*)
11. (*a*)	12. (*a*)	13. (*b*)	14. (*b*)	15. (*a*)
16. (*a*)				

CONCEPTUAL SHORT QUESTIONS WITH ANSWERS

1. Distinguish between sRAM and dRAM ?

[UPTU, B.Tech. (CSE) ; 2004-05]

Ans.

Static RAM (sRAM)	Dynamic RAM (dRAM)
1. Static RAM contains less memory cells per unit area.	1. Dynamic RAM contains more memory cells per unit area.
2. The access times are lesser so they are fast memories.	2. The access times are greater than sRAMs.
3. sRAM consists of a number of flip flops.	3. dRAM stores data as charges on the capacitor.
4. Refreshing circuitry is not required.	4. Refreshing circuitry is required to maintain the charge on the capacitors after every few milliseconds.
5. Cost is more.	5. Cost is less.

2. Consider a two level memory hierarchy (M_1, M_2). M_1 is directly connected to the CPU. Determine the average cost per bit (C) and the average access time (t_a) for the data given below :

Memory level	Capacity (S_i)	Cost (C_i)	Access Time (t_{ai})	Hit (H)
M_1 (Cache)	1024	0.1000	10^{-8}	0.9000
M_2 (Main)	2^{16}	0.0100	10^{-6}	—

Ans. (*a*) Average cost, $C = \dfrac{C_1S_1 + C_2S_2}{S_1 + S_2}$

$$= \frac{0.1\times1024 + 0.01\times2^{16}}{1024 + 12^{16}}$$

$$= 0.01138$$

(*b*) Average access time, $t_a = H^{t_{a_1}} + (1 - H)^{t_{a_2}}$

$$= 0.9000 \times 10^{-8} + (1 - 0.9000)\ 10^{-6}$$

$$= 1.09 \times 10^{-7}$$

3. Consider a two-level memory hierarchy, M_1 and M_2. Denote the hit ratio of M_1 as h. Let C_1 and C_2 be the costs per KB, S_1 and S_2 be the memory capacities and t_1 and t_2 be the access times respectively.

(*a*) Under what conditions will the average cost of entire memory system approaches C_2 ?

(*b*) Find T_{eff} or t_a of this hierarchy.

(*c*) Let $r = t_2/t_1$ be the speed ratio of two memories. Let $E = t_1/t_a$ be the access efficiency of the memory system. Express E in terms of r and h.

(*d*) Plot E against h for r = 5, 20, 100 respectively.

(*e*) What is the required hit ratio (h) to make $E > 0.95$ if $r = 100$?

[GGSIPU, M.Tech., 2nd Minor test ; 2004]

Ans. (*a*) The average cost is :

$$C = \frac{C_1S_1 + C_2S_2}{S_1 + S_2}$$

for C to approach C_2, the conditions are $S_2 \gg S_1$ and $C_2S_2 \gg C_1S_1$.

(*b*) The effective access time is :

$$t_a = \sum f_i\, t_i$$

$$= h_1t_1 + (1 - h_1)h_2t_2$$

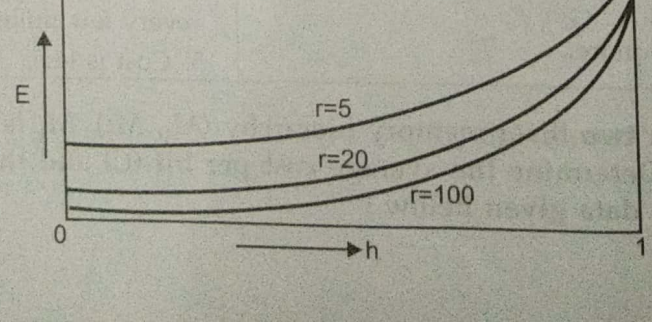
$$= ht_1 + (1 - h)t_2$$

(*c*) If $t_2 = rt_1$,

then $t_a = (h + (1 - h)\ r)t_1$

$\therefore$ $E = \dfrac{t_1}{t_a} = \dfrac{1}{(h + (1-h)\ r)}$

(*d*) The plot is as follows :



(*e*) If $r = 100$, then we get :

$$E = \frac{1}{(h+(1-h)\times 100)} > 0.95$$

Solving this inequality we get this condition :

$$h = \frac{94}{94.05} \approx \mathbf{99.95\%}$$

4. You are asked to perform capacity planning for a two-level memory system. M_1 (cache) has 3 choices – 64 KB, 128 KB and 256 KB. M_2 (MM) has 4MB capacity

Given that : $C_1 = 20\ C_2$

$t_2 = 10\ t_1$

Cache hit ratios are 0.7, 0.9 and 0.98 respectively for above three choices of M_1.

(*a*) Find t_a in terms of t_1 = 20 ns.

(*b*) Find average byte cost of entire system if C_2 = \$0.2/KB.

(*c*) Compare the three memory designs ?

[GGSIPU, M.Tech. (CSE) 1st sem., Dec. 2008]

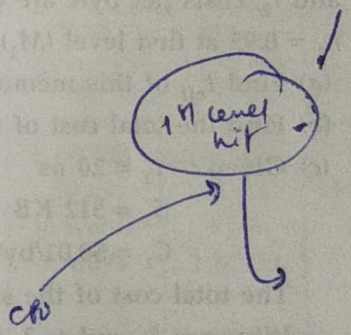
Ans. (*a*) The **average access time** is :

$$t_a = h_1 t_1 + (1 - h_1)\ h_2 t_2$$
$$= ht_1 + (1 - h)\ 10t_1$$
$$= (10 - 9h)\ t_1$$

If $h = 0.7$ then $t_a = 3.7\ t_1 = \mathbf{74\ ns}$

If $h = 0.9$ then $t_a = 1.9\ t_1 = \mathbf{38\ ns}$

If $h = 0.98$ then $t_a = 1.18\ t_1 = \mathbf{23.6\ ns}$



(*b*) The **average byte cost** is :

$$C = \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2}$$

$$= \frac{20 C_2 S_1 + C_2 \times 4}{S_1 + 4000}$$

$$= \frac{20 \times 0.2 S_1 + 0.2 \times 4000}{s_1 + 4000}$$

$$\boxed{C = \frac{4S_1 + 800}{S_1 + 4000}}$$

For S_1 = 64, the average cost is **0.26**

For S_1 = 128, the average cost is **0.32**

For S_1 = 256, the average cost is **0.43.**

(*c*) For the design choices, the product of average access time and average cost is 19.24, 12.16 and 10.15 respectively.

Therefore, the third option is the best choice.

5. Explain the following :

(*a*) Inclusion property.

(*b*) Coherence property.

(c) **Write-through policy.**

(d) **Write-back policy.**

(e) **Paging.**

(f) **Segmentation ?** **[GGSIPU, M.Tech., 6th sem., May 2003]**

Ans. (a) **Inclusion property** refers to the property that information present in a lower level memory must be a subset of that in a higher-level memory.

(b) **Coherence property** requires that copies of an information item be identical throughout the memory hierarchy.

(c) **Write-through policy** requires that changes made to a data item in a lower level memory be made to the next higher level memory immediately.

(d) **Write-back policy** postpones the update at level $(i + 1)$ memory until the item is replaced or removed from level–i memory.

(e) **Paging** divides the virtual memory and physical memory into pages of fixed sizes to simplify memory management and to remove fragmentation problem.

(f) **Segmentation** divides the virtual address space into variable-size segments. Each segment corresponds to a logical unit. The main purpose of segmentation is to facilitate sharing and protection of information among programs.

6. Consider a two level memory hierarchy (M_1 and M_2). Their access times are t_1 and t_2, costs per byte are C_1 and C_2 and capacities are S_1 and S_2 respectively. $h_1 = 0.95$ at first level (M_1)

(a) Find t_{eff} of this memory system.

(b) Find the total cost of this system.

(c) Given : $t_1 = 20$ ns, $t_2 = ?$; $S_1 = 512$ KB, $S_2 = ?$; $C_1 = \$0.01$/byte, $C_2 = \$0.005$/byte

The total cost of the system is upper-bounded by \$15,000.

What are S_2 and t_2 ?

Ans. (a) The t_{eff} is given by :

$$t_{eff} = t_1 h_1 + t_2 (1 - h_1) h_2$$
$$= t_1 h_1 + t_2 (1 - h_1)$$
$$= 0.95\, t_1 + 0.05\, t_2$$

(b) The total cost is :

$$C = C_1 S_1 + C_2 S_2$$

(c) (1) $S_2 = ?$

Now, we have this inequality :

$$0.01 \times 512 \times 1024 + 0.0005 \times S_2 \leq 15000$$

$\therefore$ S_2 cannot exceed **18.6 M bytes**

(2) $t_2 = ?$

Now, we get another inequality :

$$20 \times 0.95 + 0.05 \times t_2 \leq 40$$

$\therefore$ $t_2 \leq$ **420 ns**

7. Compare Symbolic and Numeric processing.

ns.

Symbolic processing	Numeric processing
1. It uses lists, RDBMS, scripts, nets, frames as data objects.	1. It uses integer, floating point numbers, vectors, matrices as the data objects.
2. Locality of reference may not hold.	2. High degree of spatial and temporal localities.
3. Non-deterministic algorithm are used.	3. Deterministic algorithms are used.
4. Inputs can be graphical, audio from the keyboard.	4. Fast I/O is highly desirable.
5. Processors are called symbolic processors which work on large knowledge bases (KB).	5. Processors may be SIMD, MIMD or systolic array types.

8. Compare memory hierarchy on the basis of the following :

(*a*) Access type

(*b*) Capacity in bytes

(*c*) Latency

(*d*) Block size

(*e*) Bandwidth ?

ns.

Memory/Feature	CPU	Cache	M.M	Disk	Tape
1. Access type	Random access	Random access	Random access	Direct access	Sequential access
2. Capacity (bytes)	64–1024	8–256 KB	8–64 MB	1–10 GB	1TB
3. Latency	1–10 ns	20 ns	50 ns	10 ms	10 secs
4. Block size	1 word	16 words	16 words	4 KB	4 KB
5. Bandwidth	System clock rate	8 MB/s	1 MB/s	1 MB/s	1 MB/s

9. Given that in a two-level virtual memory :

$$t_{a1} = 10^{-7} \text{ secs}$$

$$t_{a2} = 10^{-2} \text{ secs}$$

Find h so that the access efficiency e is atleast 90% ?

ns. Given that :

$$t_{a1} = 10^{-7} \text{ secs}$$

$$t_{a2} = 10^{-2} \text{ secs}$$

$$e = 90\% = 0.9$$

$$h = ? \text{ (hit ratio)}$$

Now, **access time ratio (*r*)**

$$= \frac{t_{a2}}{t_{a1}}$$

$$= \frac{10^{-2}}{10^{-7}}$$

$$= 10^{-2+7}$$
$$= 10^5$$

$\because$ $$e = \frac{1}{r+(1-r)h}$$

or $$0.9 = \frac{1}{10^5+(1-10^5)h}$$

or $$h = \mathbf{0.9999}$$

EXERCISE QUESTIONS

1. Discuss memory hierarchy technology. Explain inclusion, coherence and locality properties.

[UPTU, B. Tech (CSE) 8th Sem.; 2008-09 & GGSIPU, B.Tech. 7th sem ; Dec. 2007]

2. (*a*) Describe inclusion, coherence and locality of reference properties of a memory hierarchy.
   (*b*) Explain various address translation mechanism in a virtual memory.

[GGSIPU, M.Tech. 1st sem ; Dec. 2004]

3. In a 2 level memory system there are 8 virtual pages on a disk to be mapped on the 4 page frames in the MM. A certain program generates the following page trace :
   1, 0, 2, 7, 1, 7, 6, 7, 0, 1, 2, 0, 6
   Compute the page faults and the hit ratio. Assume page frames (PF's) are initially empty.

[GGSIPU, M.Tech. 1st Minor ; 2004]

4. Describe TLB as a part of virtual memory technology.

[GGSIPU, M.Tech. 1st Minor ; 2004]

5. Compare private and globally shared virtual memory ?
6. Write short notes on :
   (*a*) Working set (*b*) 90 – 10 rule.
7. Prove that :

$$e = \frac{1}{h+(1-h)r}$$

Where e, h and r have their usual meanings. Assume a two level memory hierarchy as M_1 and M_2.

[Hint :

t_A (average time for CPU to access a word) is given by :

$$t_A = ht_{A1} + (1-h)\, t_{A2} \quad \ldots(1)$$

If t_B is block transfer time then time to access M_2 is given by :

$$t_{A2} = t_B + t_{A1} \quad \ldots(2)$$

Substituting equation (2) in (1) we get :

$$t_A = ht_{A1} + (1-h)\,(t_B + t_{A1})$$
$$= ht_{A1} + (1-h)\, t_B + (1-h)\, t_{A1}$$
$$= ht_{A1} + (1-h)\, t_B + t_{A_1} - ht_{A1}$$
$$= t_{A1} + (1-h)\, t_B$$

— Each sector of MM can be placed in any of the **available** sector–frames. (of cache)

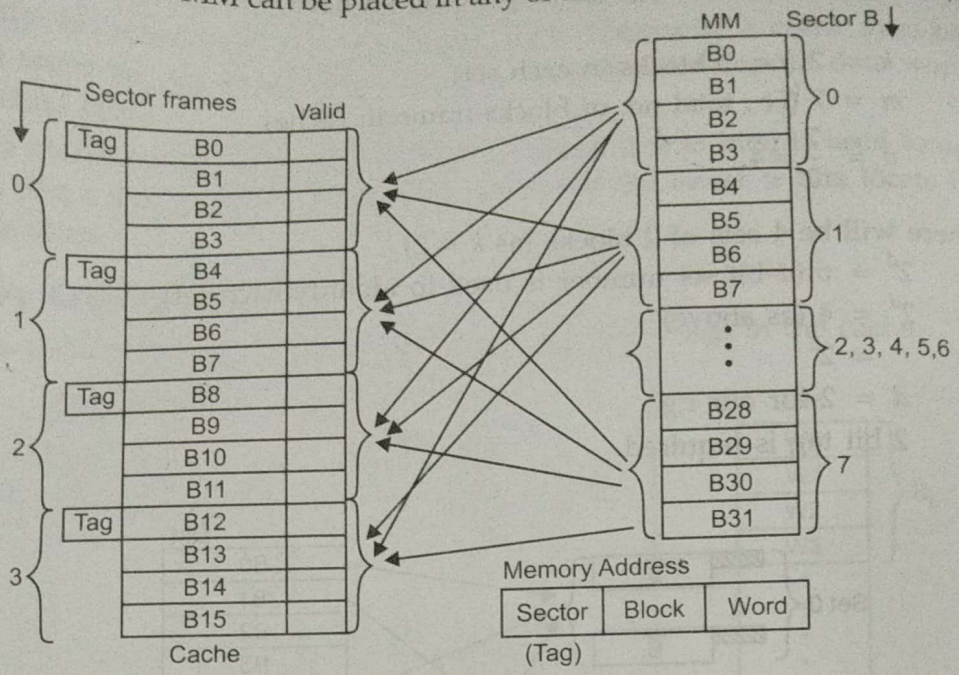


Fig. 6.12. Sector mapping cache.

— If a matched sector–frame is found (a **cache hit**), the block field is used to locate the desired block within the sector frame.

— If a **cache miss occurs**, only the missing block is fetched from the MM and brought into a similar block frame in an available sector.

So, i^{th} block in a sector → is placed into i^{th} block frame in a destined sector frame

Also, a valid bit is attached to each block frame to indicate whether the block is valid or invalid. When the contents of a block frame are replaced, the remaining block frames in the same sector are marked invalid. Only the **most recently replaced block frame in a sector** is marked valid for reference.

We are in a position to solve some problems now.

Example 6.1. Consider system with main memory (MM) consisting of 4 K blocks a cache memory consistory of 128 blocks and a block size of 16 words.

Tag
3
1
0
⋮
31

	Cache
0	384
1	129
2	
⋮	⋮
26	
127	4095

Main Memory (MM)

0	128	256	384	3968
1	129	257	385	
2	130	258	386	
127	255	383		4095
0	1	2	3	31

Fig. 6.13. Mapping main memory blocks to cache blocks.

What will be the word field, block field and Tag field length? How many bits are there in the main memory address?

Solution. It is crystal clear from the figure given in the question that th
total of 32 main memory blocks that map to a given cache block. Like, main
blocks 0, 128, 256, 384 ... 3968 map to cache block 0. ***That is why the direct
technique is is also called as many-to-one mapping technique.*** The main *advanta*
direct-mapping technique is its simplicity in determining where to place an incoming
main memory block in the cache.

Its main *disadvantage* is the inefficient use of the cache. This is so because as per this technique, a number of main memory blocks may compete for a given cache block even if there exist other empty cache blocks. This disadvantage should lead to achieving a low cache hit ratio.

According to the direct-mapping technique, the MMU (Memory Management Unit) interprets the address issued by the processor by dividing it (address) into 3-fields as follows:

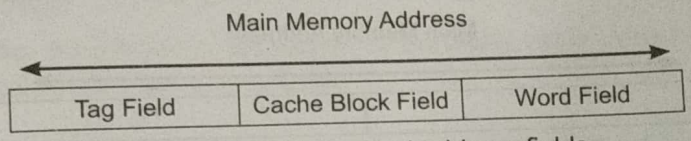


Fig. 6.14. Direct-mapped address fields.

$\therefore$ The length (in bits) of each of the fields are—

(*a*) Word field = $\log_2 B$, where B is the size of block in words ...(1)

(*b*) Block field = $\log_2 N$, where N is the size of cache in blocks. ...(2)

(*c*) Tag field = $\log_2 (M/N)$, where M is the size of MM in blocks. ...(3)

(*d*) Number of bits in the main memory address = $\log_2 (B \times M)$ ...(4)

Please note that the total number of bits as computed by first three equations (1), (2) and (3) should add up to the length of the main memory address. This is a test to check whether your result is correct or not.

$\therefore$ Given : $B = 16, \quad N = 128$

We get—

(*a*) Word field = $\log_2 B = \log_2 16 = \log 2.2^4$ **4 bits.**

(*b*) Block field = $\log_2 N = \log_2 128 = \log 2.2^7$ = **7 bits.**

(*c*) Tag field = $\log_2 (M/N)$

$= \log_2 (2^2 \times 2^{10}/2^7)$

= **5 bits.**

(*d*) Number of bits in the main memory address

$= \log_2 (B \times M)$

$= \log_2 (2^4 \times 2^{12})$

= **16 bits.**

Example 6.2. Calculate the above given parameters in Q 1 for a memory system having the following specifications—

- Size of MM = 4 K blocks
- Size of cache = 128 blocks.
- Block size = 16 words.

Assume that the system uses associative mapping.

Solution. For associative mapping,

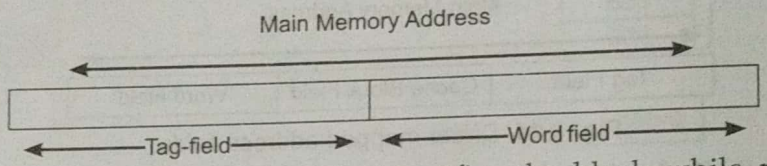
(*a*) Word field = $\log_2 B = \log_2 16$
$= \log 2.2^4$ $\log_2 2^4$
$= 4$ **bits**

(*b*) Tag field = $\log_2 M = \log_2 2^7 \times 2^{10}$
$= 12$ **bits**

(*c*) No. of bits in *MM* address
$= \log_2 (B \times M)$
$= \log_2 (2^4 \times 2^{12})$
$= 16$ **bits**

Please note the following points regarding fully associative mapping—

1. An incoming *MM* block can be placed in any available cache block
2. The address issued by the processor need only have two fields—



Tag field — uniquely identifies the block while residing in the cache.

Word–field — identifies the element within the block that is requested by CPU.

It is MMU hardware that interprets the address issued by the processor by dividing if into two fields as shown in figure.

3. ***Also note that the total number of bits as computed by the first two equations should add up to the length of the main memory address.*** This can be used as a check for correctness of your computations.

Example 6.3. Compute the three parameters–word, set and tag for a memory system having the following characteristic—

- Size of MM = 4K blocks
- Size of cache = 128 blocks
- Block size = 16 words.

Assume that the system uses set–associative mapping with four block per set.

Solution. Again in set–associative mapping,

(*a*) Word field = $\log_2 B$, where B is the size of block in word

(*b*) Set field = $\log_2 32 = 5$ **bits**

(*c*) Tag field = $\log_2 (4 \times 2^{10}/32) = 7$ **bits**

∴ Number of bits in the MM

Address = $\log_2 (B \times M)$
= $\log 2 (2^4 \times 2^{12})$
= **16 bits.**

Note : Why are we talking about blocks and not whole sector because the memory requests are destined for blocks & not for sectors.

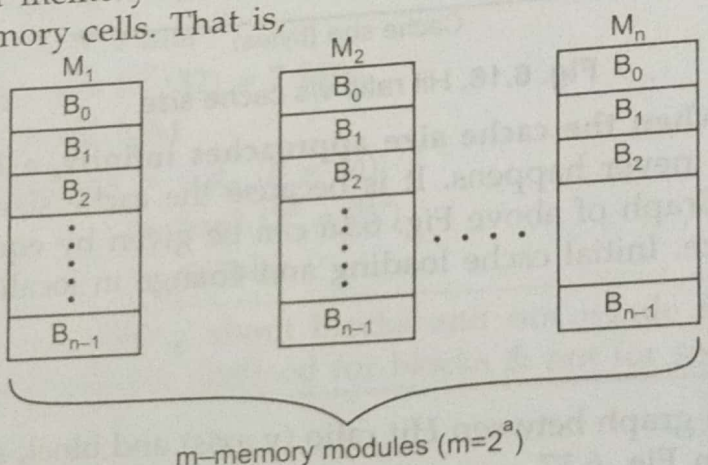
the memory speed [illegible] caches [illegible]
An optimal cache [illegible] design is a must.

6.6 INTERLEAVED MEMORY ORGANIZATION

In this technique, the system memory is divided into a number of memory modules and arranges addressing so that successive words in the address space are placed in different modules. Our objective is to broaden the **effective memory bandwidth** so that more memory can be accessed per unit time. We need to match the memory bandwidth with the bus bandwidth and with the processor bandwidth.

What is Memory Interleaving ?

Consider a main memory with m memory modules wherein $m = 2^a$. Each has $w = 2^b$ words of memory cells. That is,



$\therefore$ Total memory capacity $= m \times w$
$= 2^a \times 2^b$
$= 2^{a+b}$ words.

These memory words are assigned linear addresses.

There are two address formats for memory interleaving :

1. Low–order interleaving.
2. High–order interleaving.

Let us discuss these formats one by one.

I. **Low–order interleaving** spreads the contiguous memory locations across the m modules horizontally. This is shown in Fig. 6.18 below :

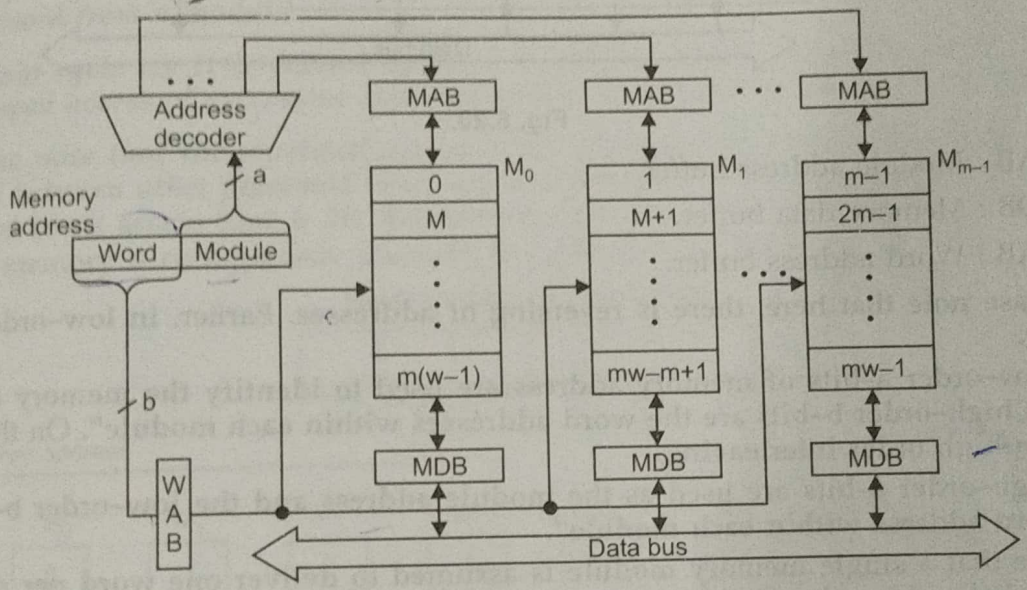


Fig. 6.18. Low order m–way interleaving.

MAB : Module address buffer.
MDB : Memory data buffer.
WAB : Word address buffer.

The **low–order a–bits** of the memory address are used to identify the memory module.

The **high–order** b–bits are the word addresses within each module. **Please note that the same word address is applied to all memory modules simultaneously.** An address decoder is used to distribute module addresses.

II. **High–order interleaving :** It uses the **high–order** a–bits as the module address and the **low–order** b–bits as the word address within each module. Contiguous memory locations are thus assigned to the same memory module. In each memory cycle, only one word is accessed from each module. So, this type of interleaving cannot support block access of contiguous locations. It is shown in Fig. 6.19 below :

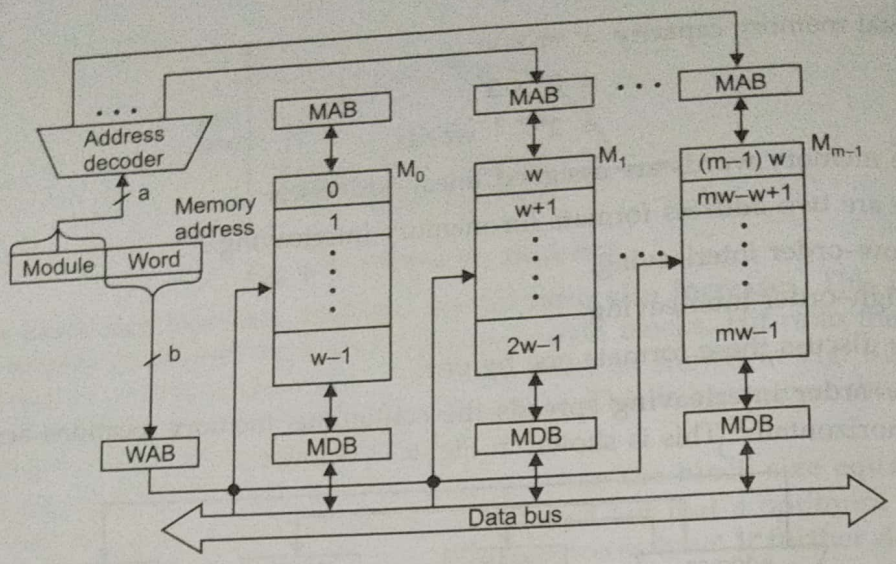


Fig. 6.20.

MAB : Module address buffer.

MDB : Memory data buffer.

WAB : Word address buffer.

Please note that here, there is reversing of addresses. Earlier, **in low–order interleaving.**

"Low–order a–bits of memory address are used to identify the memory module and the high–order b–bits are the word addresses within each module". On the other hand, **in high order interleaving,**

"High–order a–bits are used as the module address and the low–order b–bits as the word address within each module".

Note that a single memory module is assumed to deliver one word per memory cycle and thus has a bandwidth of 1.

What is memory bandwidth ?

The memory bandwidth (B) of an m–way interleaved memory is upper–bounded by m and lower bounded by

1. According to Hellerman, for uniprocessor,

$$B = m^{0.56} \simeq \sqrt{m}$$

Where m is the number of interleaved memory modules.

If $m = 16$ then $B = \sqrt{16} = 4$

This means that if 16 memory modules are used, then the effective memory bandwidth is four times that of a single module.

What is fault tolerance?

It is easier to locate faulty memory modules in a memory bank of m–memory modules due to memory interleaving. However, this fault isolation cannot be carried out in low–order interleaved memory in which a module failure may affect the entire memory bank. Thus, **low–order interleaving memory is not fault–tolerant.**

Pipelined Memory Access

It is also possible to overlap 'm' memory modules in a pipelined fashion. So, the memory cycle or the **major cycle** is subdivided into **'m' minor cycles.**

For eg : An eight way inter based memory (with $m = 8$ and $w = 8$ and thus $a = b = 3$) is shown in Fig. *a*.

Let 'θ' be the major cycle

'τ' be the minor cycle.

Then, these two cycle times are related as follows:

$$\tau = \frac{\theta}{m}$$

Where 'm' is degree of interleaving. Fig (*b*) shows the timing of the pipeline accesses of the eight contiguous memory words. ***Please note note that this type of concurrent access of contiguous words is called as C-access memory scheme.***

We define ***major cycle*** (θ) ***and minor cycle*** (τ) now.

Major cycle (θ): ***It is defined as the total time required to complete the access of a single word from a module.***

Minor cycle (τ): ***It is defined as the actual time needed to produce word, assuring overlapped access of successive memory modules separated in every minor cycle, τ.***

Also note that the pipelined access of the block of or consecutive words is sandwiched between other pipelined block access before and after the present block. Even if the total block access time is 20, the effective access time of each word is reduced to τ as the memory is contiguously accessed in a pipeline fashion.

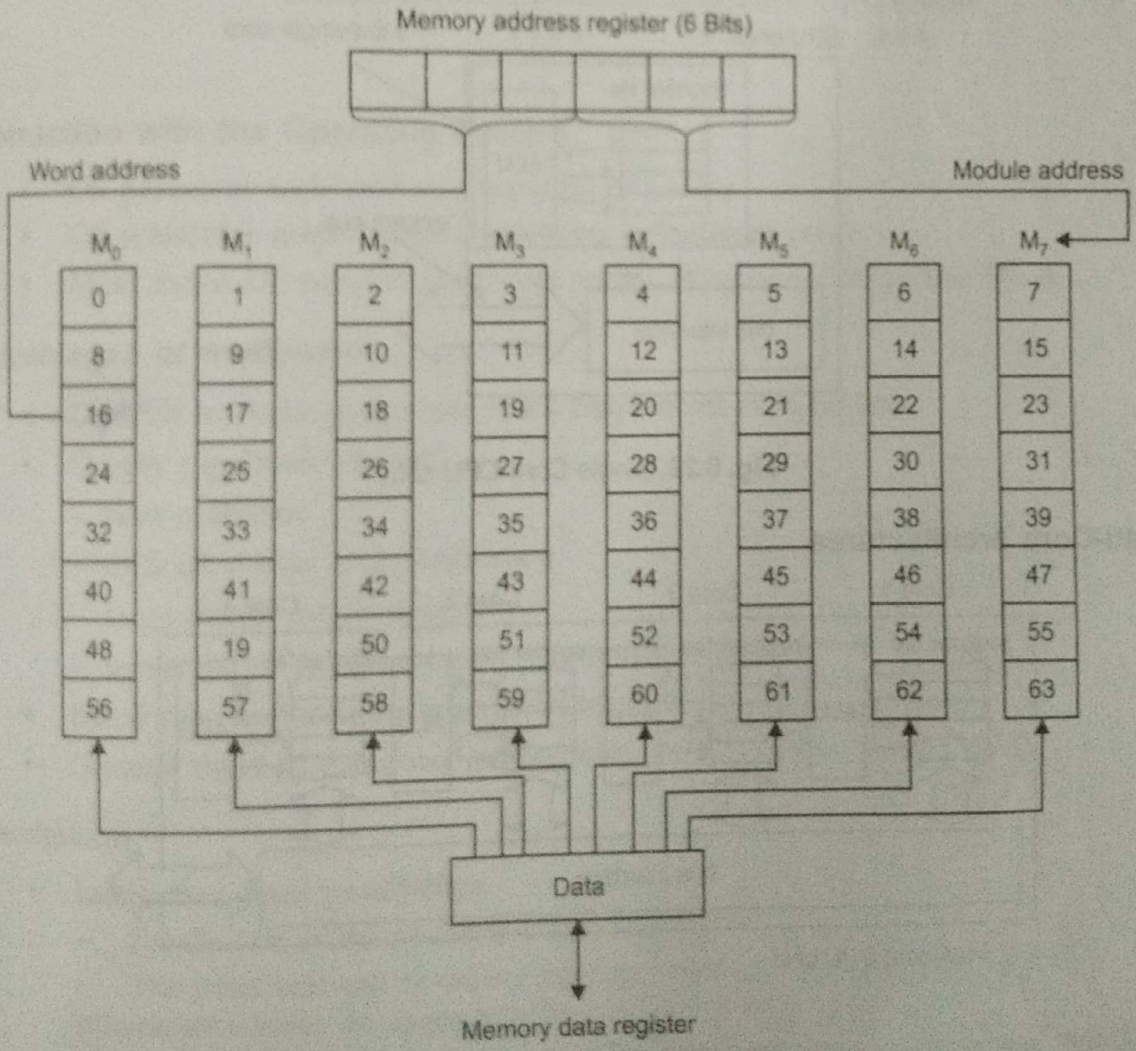


Fig.6.21.(*a*) 8-way low-order interbaving (Absolute address shown in each memory word)

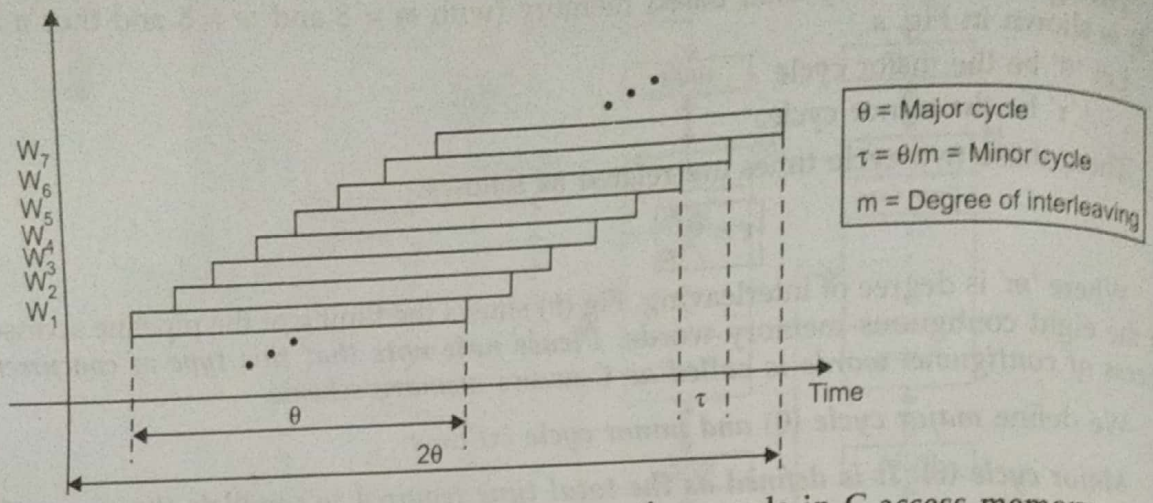


Fig.6.21.(*b*) Pipelined access of 8 consecutive words in C-access memory.

6.7 MULTI-CORE ARCHITECTURES & CACHE COHERENCE PROBLEM

A Multi-Core Architecture is a method of embedding a number of cores on a single chip. A Multi Core Architecture improves the performance of a system by computing a number of tasks at the same time.

Single-Core CPU chip

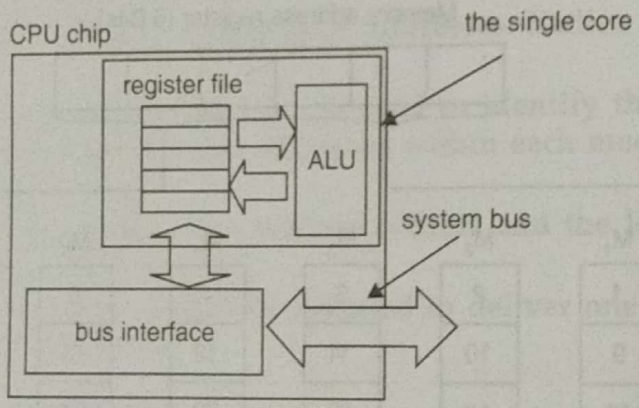


Fig. 6.22. Single Core CPU chip.

Multi-Core architectures

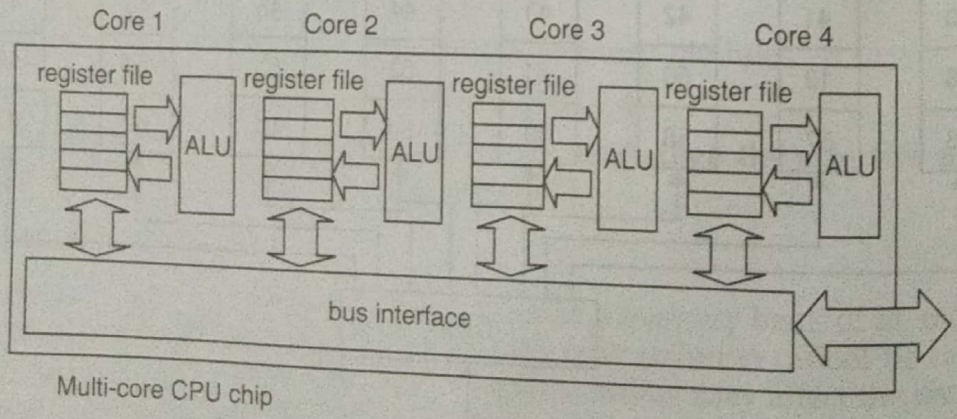


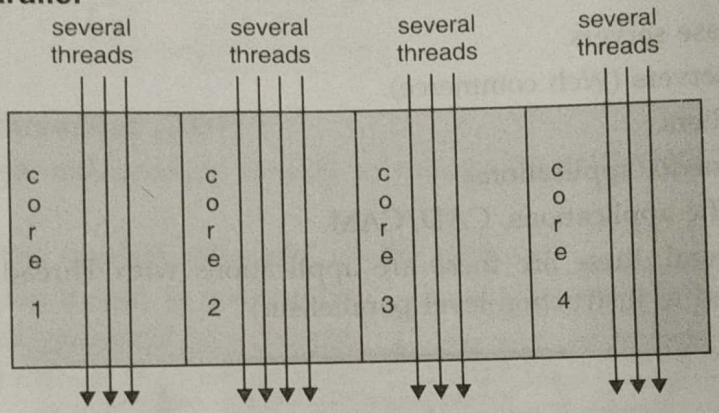
Fig. 6.23. Multi-core CPU chip

Multi-Core CPU chip

- The cores fit on a single processor socket.
- Also called CMP (Chip Multi-Processor)

core 1	core 2	core 3	core 4

The cores run in parallel



Interaction with the Operating System

- OS perceives each core as a separate processor.
- OS scheduler maps threads/processes to different cores.
- Most major OS support multi-core today: Windows, Linux, Mac OS X, ...

Advantages of multi-core

- Difficult to make single-core clock frequencies even higher.
- Deeply pipelined circuits:
  - heat problems.
  - difficult design and verification.
  - large design teams necessary.
  - server farms need expensive air-conditioning.
- Many new applications are multithreaded.
- General trend in computer architecture (shift to wards more parallelism).

Parallelism

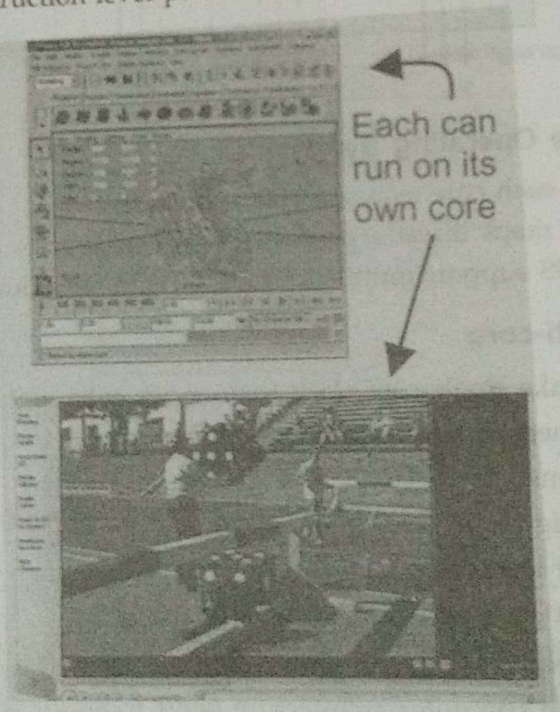
- **Instruction-level parallelism**
  - Parallelism at the machine-instruction level.
  - The processor can re-order, pipeline instructions, split them into microinstructions, do aggressive branch prediction, etc.
  - Instruction-level parallelism enabled rapid increases in processor speeds over the last 15 years.

~4

- **Thread-level parallelism (TLP)**
  - This is parallelism on a more coarser scale.
  - Server can serve each client in a separate thread (Web server, database server).
  - A computer game can do AI, graphics, and physics in three separate threads.
  - Single-core superscalar processor cannot fully exploit TLP.
  - Multi-core architectures are the next step in processor evolution : explicitly exploiting TLP.

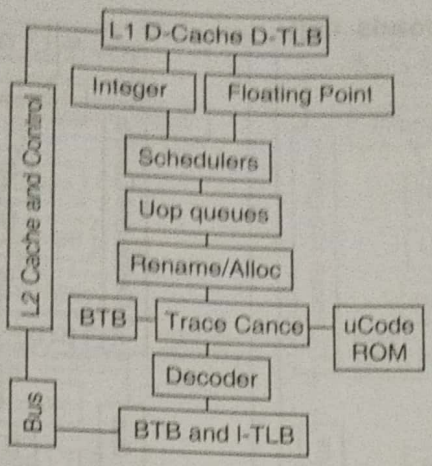
Applications benefit from multi-core

- Database servers.
- Web servers (Web commerce).
- Compliers.
- Multimedia applications.
- Scientific applications, CAD/CAM.
- In general, these are these are applications with Thread-level parallelism (as opposed to instruction-level parallelism)



A technique complementary to multi-core : Simultaneous multithreading

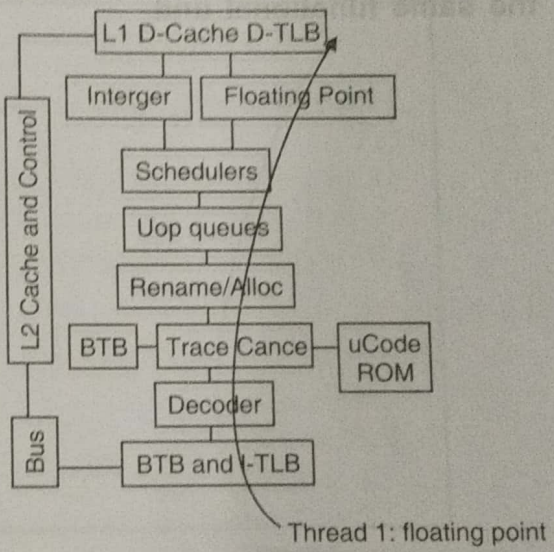
- Problem addressed: The processor pipeline can get stalled:
  - Waiting for the result of a long floating point (or integer) operation
  - Waiting for data to arrive from memory
- Other execution units wait unused.



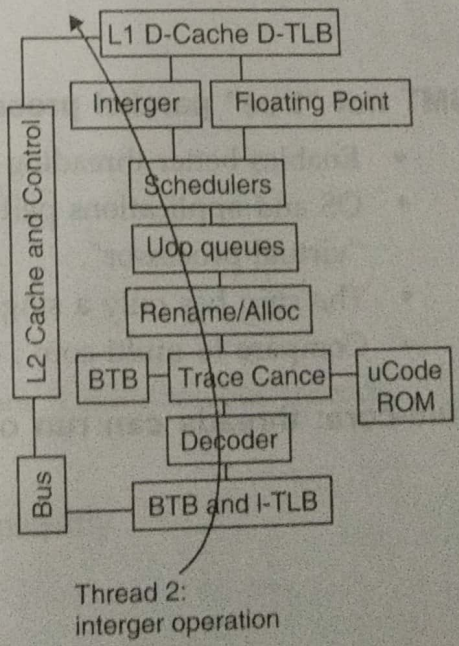
Simultaneous multithreading (SMT)

- Permits multiple independent threads to execute SIMULTANEOUSLY on the SAME core
- Weaving together multiple "threads" on the same core
- **Example :** if one thread is waiting for a floating point operation to complete, another thread can use the integer units.

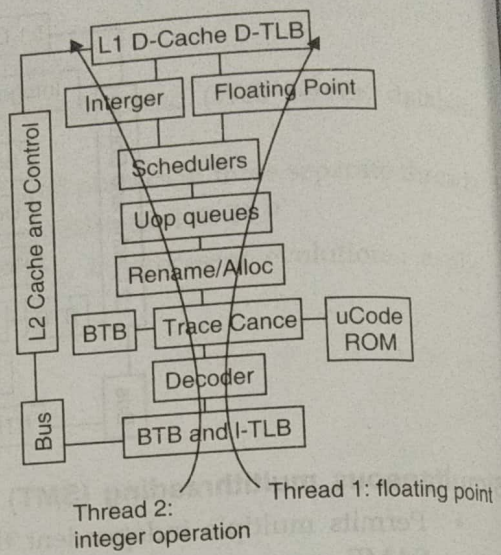
Without SMT, only a single thread can run at any given time



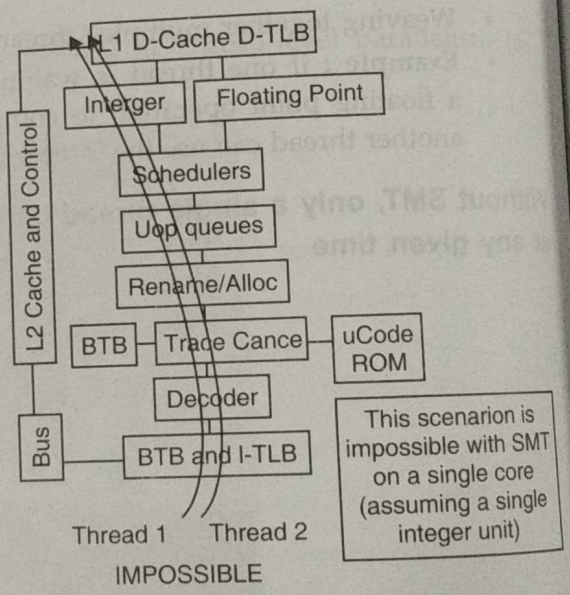
Without SMT, only a single thread can run at any given time



SMT processor : both threads can run concurrently



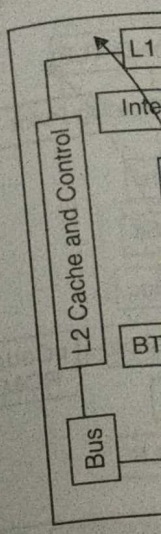
But: Can't simultaneously use the same functional unit



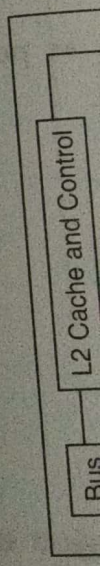
SMT not "true" parallel processor

- Enables better threading (e.g. up to 30%).
- OS and applications perceive each simultaneous thread as a separate "virtual processor".
- The chip has only a single copy of each resource.
- Compare to multi-core : each core has its own copy of resources.

Multi-core: threads can run on separate cores



Multi-core: thread

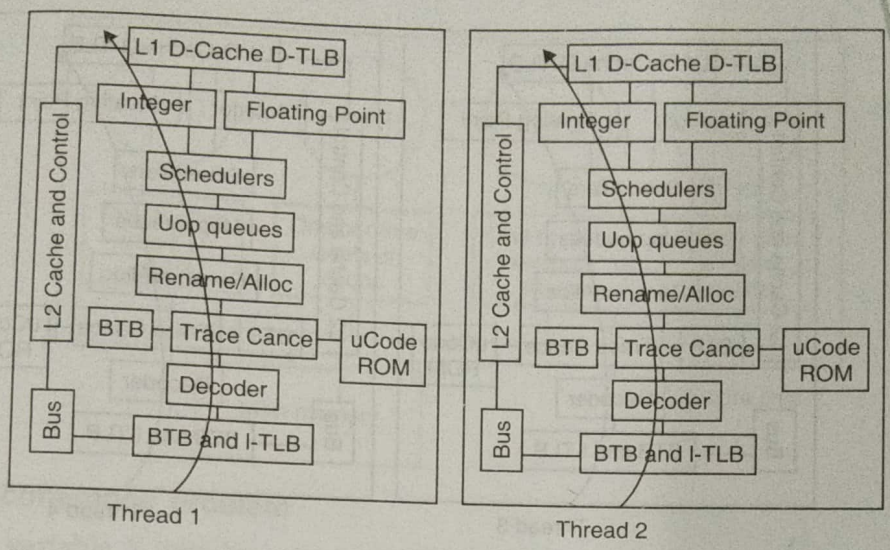


Combining Mu

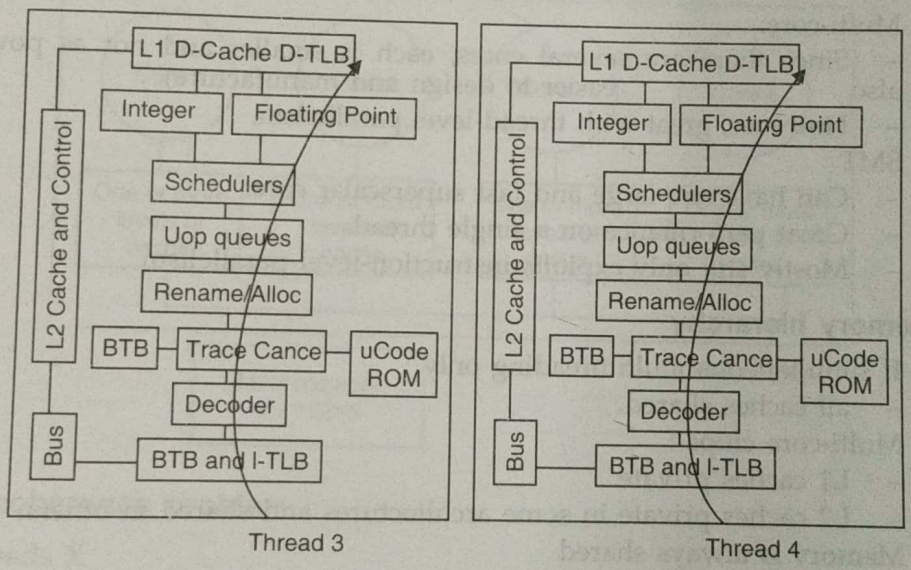
- Cores c
- The dif
  - Sin
  - Sin
  - Mu
  - Mu
- The nu
- Intel c

SMT Dual-co

BACKPLANE BUS SYSTEM



Multi-core: threads can run on separate cores



Combining Multi-core and SMT

- Cores can be SMT -enabled (or not)
- The different combinations :
  - Single-core, non-SMT : standard uniprocessor
  - Single-core, with SMT
  - Multi-core, non-SMT
  - Multi-core, with SMT : our fish machines.
- The number of SMT threads: 2, 4, or sometimes 8 simultaneous threads
- Intel calls them "hyper-threads".

SMT Dual-core : all four threads can run concurrently

word size is expressed in bytes.

(a) 4, 8, 16 bits (b) 8, 16 and 32 bits
(c) 56 bits (d) 64 bits

nsformation of data from main memory to the cache memory is known as:

ersing process. (b) Mapping process.
ct mapping. (d) Associative mapping.

:: ANSWERS ::

1. (b)	2. (a)	3. (a)	4. (b)	5. (b)
6. (a)	7. (a)	8. (b)	9. (b)	10. (b)

CONCEPTUAL SHORT QUESTION WITH ANSWERS

1. 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.

(i) How many bits are required for addressing the main memory ?

(ii) How many bits are needed to represents the TAG, SET, WORD fields ?

[UPTU, B.Tech. 7th sem, 2004 & 2005]

Ans. (i) Main memory consists of 16,384 blocks.

Each block consist of 256 eight–bit words.

Total capacity of main memory

$= 16{,}384 \times 256 \times 8$

$= 4096\ k \times 8$

$= 2^{22} \times 8$

$\therefore$ **22 bits** are required for addressing main memory.

(ii) **Associative mapping** is being used here.

Now, **Word bits** required are as follows :

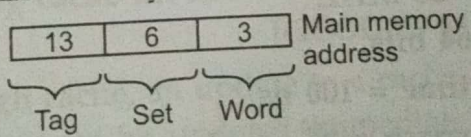
Each block consists of 8 bit words.

$\therefore$ To identify each word we must have $(2^3 = 8)$ **three bits** reserved for it.

Next, let us find out the **set bits** required. Now, there are 64 sets (*i.e.*, $128/2 = 64$). To identify each set $(2^6 = 64)$ **six bits** are required.

Now, how many **tag bits** are required?

The remaining 13 (22-6-3) address bits are tag bits which stores higher address of the main memory. So, the main memory address is formed:



2. What are different cache addressing models ?

Ans. There are 2 cache addressing models:

(i) Physical Address Caches.

(ii) Virtual Address Caches.

When a cache is accessed with a physical memory address, it is called as a **physical address cache**. A unified cache accessed by the physical address is shown in Fig. 1.

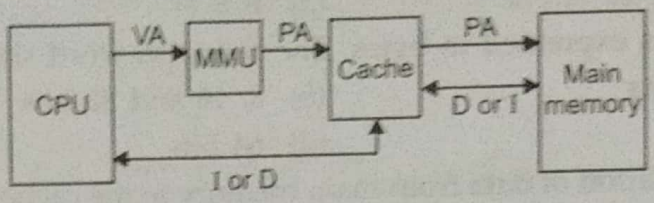


Fig. 1. Physical address caches.

Where VA = Virtual address
PA = Physical address
I = Instructions
D = Data

So, as shown in Fig. 1 above, the cache is indexed and tagged with a physical address. A **cache hit** occurs when required data/instruction is found in the cache. Else a **cache miss** occurs. After a miss, some data comes to cache from the main memory.

On the other hand, when a cache is indexed or tagged with a virtual address then it is called as a **virtual address cache**. Herein, both cache and MMU translation are done in parallel. It is shown in Fig. 2.

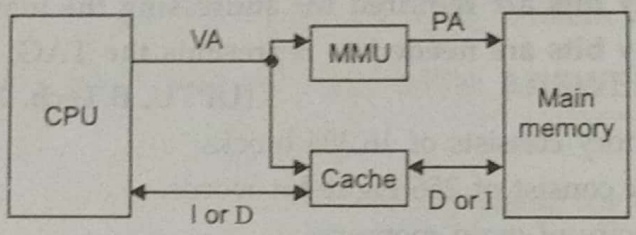


Fig. 2. Virtual address cache.

Remember that CPU generates virtual address (VA) or logical address. This VA is transformed into physical address (or PA) by MMU (as discussed earlier also)

3. **What is the Aliasing problem ?**

Ans. When different logically addressed data have the same index/tag in the cache, it results in the **aliasing problem.** This problem is found in virtual address cache scheme. Now, multiple processes may use the same range of virtual addresses. This aliasing problem may create confusion **if two or more processes access the same physical cache location. The solution to this problem** is to flush the entire cache whenever aliasing occurs. But please remember that excessive **flushing** may reduce cache performance, reduce the hit ratio and may waste lot of time.

4. **A backplane bus multiprocessor has the following design specifications :**
   1. **Bus clock rate = 20 MHz.**
   2. **Word length = 64 bits.**
   3. **Memory access time = 100 ns.**
   4. **Shared address space = 2^{40} words.**
   5. **Maximum no. of signal lines = 96.**
   6. **Synchronous timing protol.**
   7. **Neglect buffer and propagation delays.**

Find out :

(a) Maximum bus bandwidth.

(b) Effective bus bandwidth.

(c) Arbitration scheme.

(d) Name and functionality of each signal line.

(e) Number of slots required on the backplane ?

Ans. (a) Maximum bus bandwidth = $8 \times 20 = 160$ Mbytes/s.

(b) Total time taken by PE to access one word from the memory is :

$= 50\text{ns} + 100\text{ns} + 50\text{ns}$

$= 200\text{ns}$

During this time, the bus cannot be used by other processors.

$\therefore$ The effective bandwidth $= \dfrac{8 \text{ bytes}}{200 \text{ ns}}$

$= 40$ Mbytes/s

If the memory addresses are interleaved then four words can be performed simultaneously. It takes 50 ns to transmit the address to the memory module, 100ns to get data ready in latches. Then it takes four bus cycles to transfer the four words to the requesting processor.

Total time required $= 50 + 100 + 4 \times 50$

$= 350$ ns

Effective bus bandwidth $= \dfrac{4 \times 8 \text{ bytes}}{350 \text{ ns}}$

$= 91.4$ Mbytes/s.

(c) Based on the desired performance and circuit complexity, any of the arbitration scheme can be used.

(d) 40 address lines are needed.

64 data lines are needed.

(e) At least 21 slots are needed, one for each processor board, one for each memory board and one for the bus controller.

5. Explain the following terms :

(a) Write–through versus write–back caches.

(b) Cacheable versus non-cacheable data.

(c) Private caches versus shared caches.

(d) Cash flushing policies.

(e) Factors affecting cache hit ratios.

[GGSIPU, M.Tech. Dec. 2003, May 2003]

Ans. (a) In **write–through cache,** an update to a cache block causes the corresponding memory block to be updated immediately. There is no data loss. In **write–back cache,** the update to the memory block is postponed until the cache block is replaced. But this system has some disadvantages too. The write–back cache controller is not simple. Also, in case of a power failure, the data in the cache memory is lost.

(*b*) Data which are globally shared among several processors and whose values may be updated can be tagged as **non–cacheable.** While instructions, private data and globally shared readable data are called as **cacheable.**

(*c*) Private caches are those that are attached to individual processors. **Shared caches** are shared among processors, much like shared memory modules. These two types of caches can coexist in a system. For instance, a shared cache can be used as second–level cache in a multilevel cache system.

(*d*) Cache flushing deals with the **aliasing problem** in a virtual address cache. These policies determine when **flushing** should be performed and the level at which flushing takes place *i.e.*, page level, segment level or the context level. These policies are related to the OS design also.

(*e*) Cache hit ratio is affected by the factors like cache capacity and block size. A large cache will improve the hit ratio. For a fixed cache size, there is an optimal block size at which hit ratio peaks. A small block size does not take full advantage of locality of reference whereas a large block size can load unneeded data into the cache. Other factors include:

(*a*) The number of sets

(*b*) The set size

These factors affect the hit–ratio particularly in set–associative cache policy.

6. Explain the following terms :

(*a*) Low–order memory interleaving.

(*b*) Physical versus virtual address cache.

(*c*) Atomic versus nonatomic memory accesses.

(*d*) Memory bandwidth and fault tolerance.

[GGSIPU, M.Tech. Dec. 2003]

Ans. (*a*) Please refer to section 6.6 of this chapter.

(*b*) Please refer to Q.2 of this chapter's conceptual problems.

(*c*) In a shared memory, if the update to a memory location is observed by all PEs at the same time then the memory access is **atomic.** If the update is not necessarily observed by all PEs simultaneously then the memory access is **non–atomic.**

(*e*) Please refer to section 6.6 of this chapter.

7. The main memory of the computers has 64 blocks with a block size of 8 words. The cache has 8 block frames. Show the mappings from the blocks in MM to the block–frames in the cache. Draw all lines showing the mappings as clearly as possible for :

(*a*) Direct mapping alongwith address bits.

(*b*) Full associative mapping alongwith address bits.

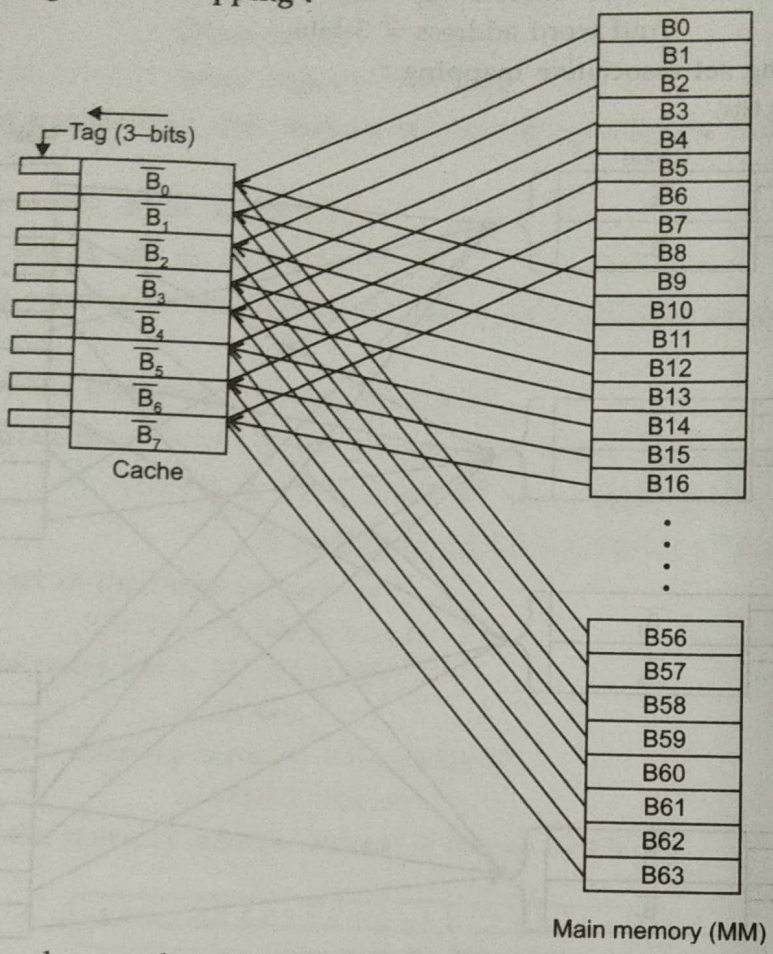
(*c*) Two–way set associative mapping alongwith it's address bits.

(*d*) Sector mapping with 4 blocks/sectors alongwith the address bits.

[GGSIPU, M–Tech. 2nd minor test–2004]

Ans. The main memory blocks are numbered from 0 to 63. The cache block–frames are numbered from 0 to 7. So, we draw now one by one.

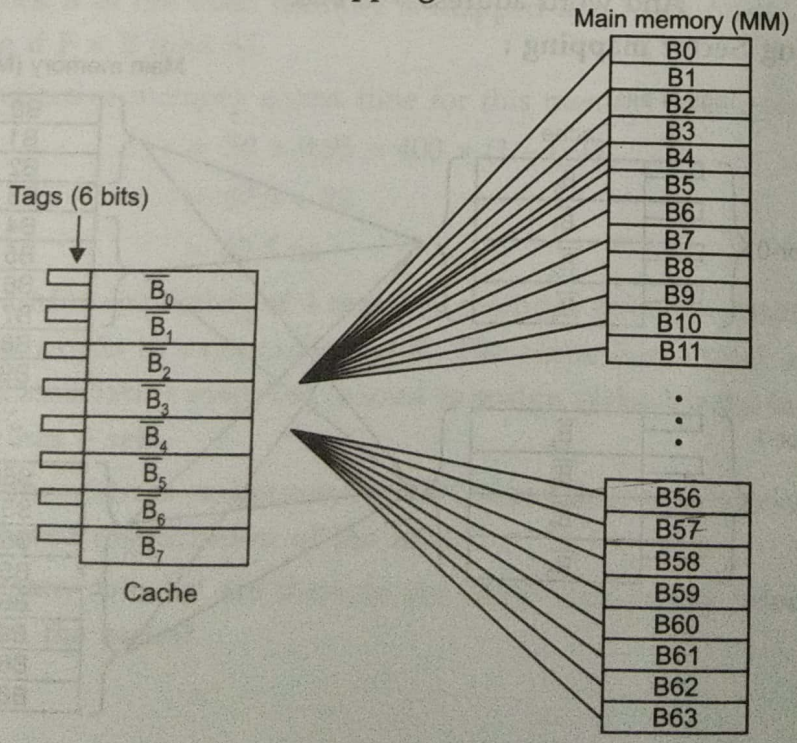
(a) **Using Direct Mapping :**



So, here cache address tag = 3 bits long,

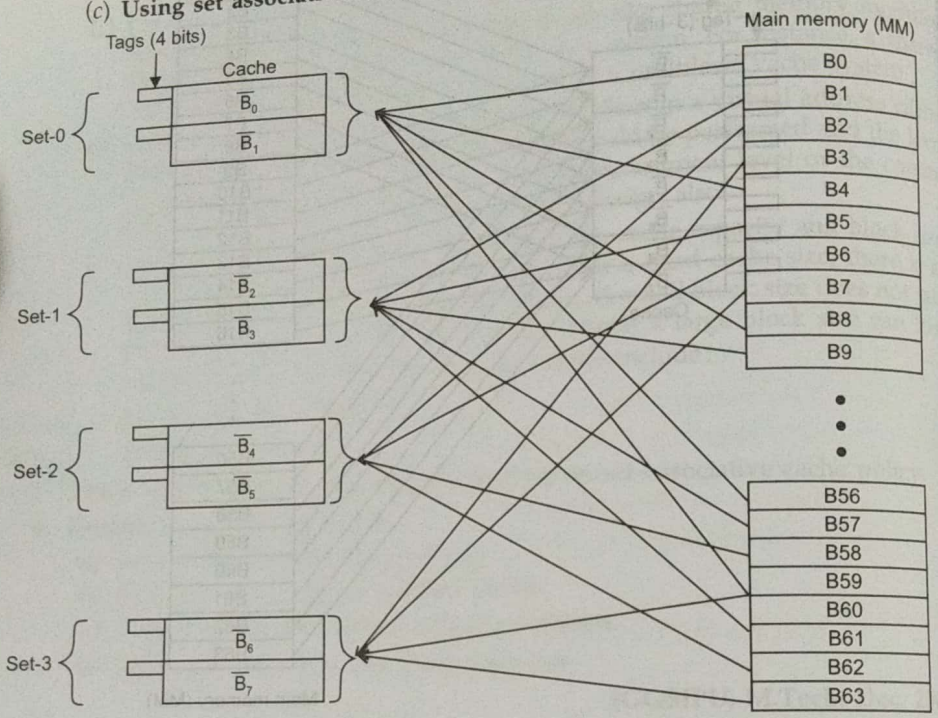
Block address = 3 bits long and word address = 3 bits long.

(b) **Using fully associative mapping :**



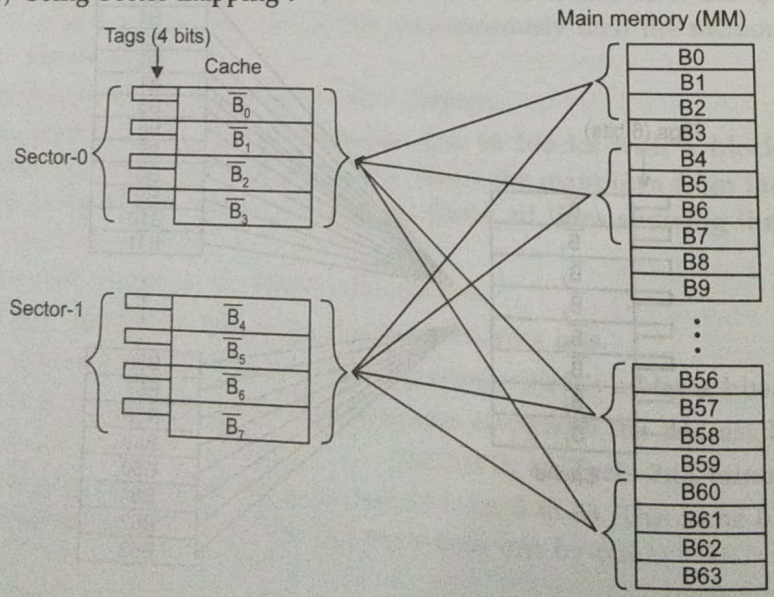
So, In cache address tag = 6 bits
And word address = 3 bits.

(*c*) **Using set associative mapping :**



Herein, cache address tag = 4 bits long.
Set number = 2 bits
And word address = 3 bits.

(*d*) **Using Sector mapping :**



Herein, secto

8. **Consider a cache (M**
teristics :
M_1 : 16k words, 50
M_2 : 1M words, 400
Assume 8–word cach
ping.
(*a*) Show the map
(*b*) Calculate the e
[U

Ans. (*a*) Each set of the
256
Entire cache

Similarly, mer

Thus, the me

A block B of
cache if F =
(*b*) The effective

9. **Consider a MM**
Assume 16 wor
words. Set asso
The cache has
(*a*) Show the a
interleaved
(*b*) How many
there in th

Herein, sector number tag = 4 bits long.
Block address = 2 bits.
Word address = 3 bits.

8. Consider a cache (M_1) and memory (M_2) hierarchy with the following characteristics :

M_1 : 16k words, 50 ns access time.

M_2 : 1M words, 400 ns access time.

Assume 8–word cache blocks and a set size of 256 words with set–associative mapping.

(*a*) Show the mapping between M_2 and M_1.

(*b*) Calculate the effective memory access time with a cache hit ratio of h=0.95?

[UPTU, B. Tech. (CSE) 8th Sem., 2003-04, 2004-05 & 2007-08]
& [GGSIPU, M.Tech-(CSE/IT) 1st Sem., Dec. 2011]

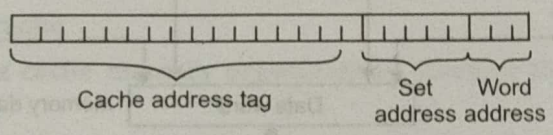
Ans. (*a*) Each set of the cache consists of

256/8 = 32 block frames.

Entire cache has = 16 ×1024/256
= 64 sets.

Similarly, memory contains 1024 ×1024/8
= 131072 blocks.

Thus, the memory address format is drawn as follows :



A block B of the main memory is mapped to a block frame in set F of the cache if F = B mod 64.

(*b*) The effective memory access time for this memory hierarchy is given by :

= 50 × 0.95 + 400 × (1 – 0.95)
= 47.5 + 20
= 67.5 ns

9. Consider a MM consisting of 4 memory modules with 256 words per module. Assume 16 words in each cache block. The cache has a total capacity of 256 words. Set associative mapping is used to assign cache blocks to block-frames. The cache has 4 sets.

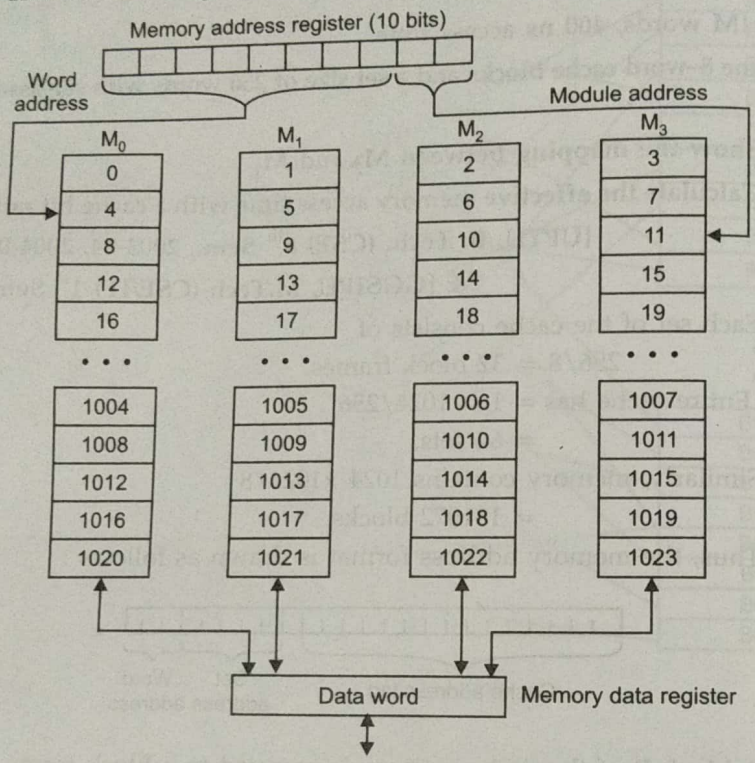
(*a*) Show the address assignments for all 1024 words in a four–way low–order interleaved organization of the MM.

(*b*) How many 'blocks' are there in the MM? How many 'block–frames' are there in the cache?

(c) Explain bit fields needed for addressing each word in the 2–level memory system.

(d) Show the mapping from the blocks in main memory to the sets in the cache and explain how to use the tag field to locate a block frame within each set? **[GGSIPU, M. Tech. 6th sem ; 2004]**

Ans. (a) The address assignment is shown below :



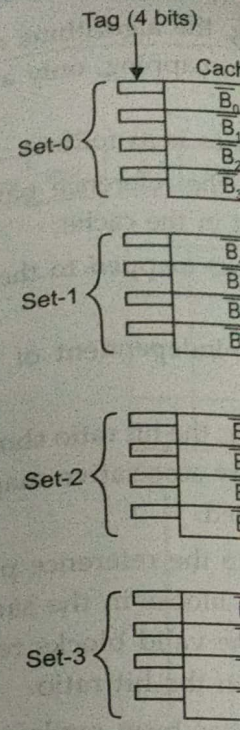
(b) There are 1024/16 = 64 blocks in the main memory

There are 256/16 = 16 blocks-frames in the cache.

(c) 10 bits are required to address each word in the main memory.

2–bits for selecting the memory module and 8 for the offset of a word within the module. 6–bits are required to select a word in the cache – 2 bits to select the set number and 4–bits to select a word within a block. Also that each block frame needs a 4–bit address tag to determine the block resident in it.

(d) The mapping of memory blocks to the block frames in cache is shown below:



After the set in w
address tag of th
search with the p
is in the cache.

10. There are four cache with reasons :

(a) Rank the four c cost.

(b) Rank the four algorithms.

(c) Explain the effe

(d) Explain the effe mance of a set

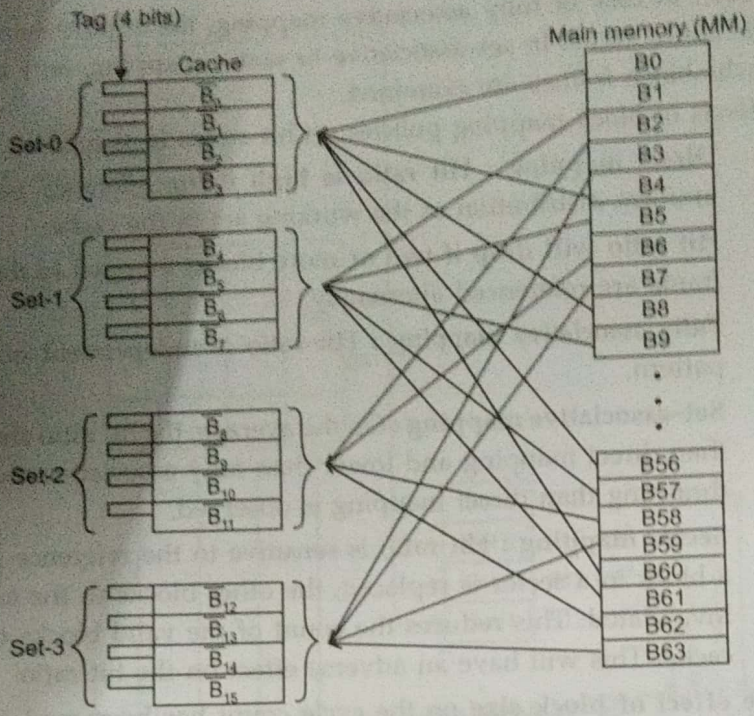
Ans. (a) Direct mapping is sufficient.

Fully associativ search on all blo

In set–associativ set.

In sector mappi the size of each

(b) **In direct mappi** ment algorithms



After the set in which a memory block can be mapped into is identified, the address tag of the block–frames in that set is compared by an associative search with the physical memory address to determine if the desired block is in the cache.

10. **There are four cache memory organizations Answer the following questions with reasons :**
    (*a*) **Rank the four cache organizations in terms of hardware complexity and cost.**
    (*b*) **Rank the four cache organizations with respect to block replacement algorithms.**
    (*c*) **Explain the effects of block mapping policies on the hit ratio issues.**
    (*d*) **Explain the effect of block size, set number and cache size on the performance of a set associative cache organization.**

Ans. (*a*) Direct mapping has the lowest cost. It is because a simple mod operation is sufficient.

Fully associative mapping has highest cost. It is because an associative search on all block–frames is needed.

In set–associative mapping, the associative search is needed within each set.

In sector mapping, it is needed to find the sector. For a fixed cache size, the size of each set or sector will make a difference in the cost.

(*b*) **In direct mapping,** the block replacement is rigid and trivial. Any replacement algorithms can be implemented with any of the three mappings of

cache. In case of fully associative mapping, the algorithms are applied to the entire cache. In set–associative or sector mapping, only a subset of the cache block frames are examined.

(*c*) Effects of block mapping policies on hit–ratios is as follows :

1. **Direct mapping : Hit ratio is high** if the reference pattern leads to uniform distribution of the working set in the cache.

   Hit ratio will drop if two or more blocks mapped to the same block-frame are referenced alternately.

2. **Fully associative mapping :** Hit ratio is independent of the reference pattern.

3. **Set–associative mapping :** On the average, the hit ratio should be higher than direct mapping and lower than fully associative mapping. Lesser thrashing than direct mapping is observed.

4. **Sector mapping :** Hit ratio is sensitive to the reference pattern. When a block in a sector is replaced, the other blocks in the same sector are invalidated. This reduces the count of the valid blocks residing in the cache. This will have an adverse effect on the hit ratio.

(*d*) **The effect of block size** on the cycle count has been explained in section 6.5 of this chapter.

The effect of set number and associativity is here. For a fixed cache size, the two parameters are inversely proportional to each other. When the cache number is small it behaves more like a fully associative cache. When the number of sets is large, its behaviour is close to that of direct mapping. The hit ratio is expected to become lower now.

Cache size effect is also given next. With a larger cache both the hit ratio and the cycle count will increase. This is so because now more data (D) and instructions (I) can be stored in the cache memory.

11. A 128 KB Cache has 64 B lines, 8B physical word, 4k byte pages and is from way set-associative. It uses copy back and LRU replacement. The processor creates 30-bits (byte-addressed) virtual addresses that are translated into 24-bit (byte-addressed) real byte addresses (labelled–$A0_1$ .....A_{23}) from least to most significant.

(*a*) Show a complete layout of cache.

(*b*) Which address bits are unaffected by translation?

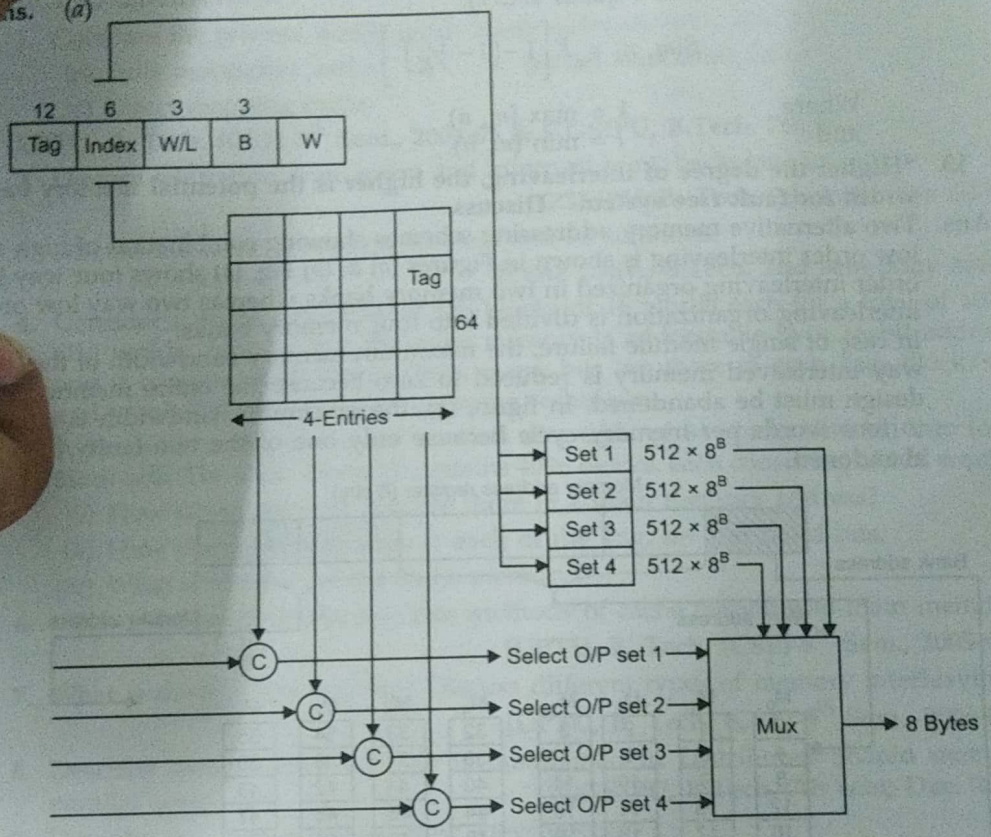
(*c*) Which address bits are compared to entries in cache directories?

(*d*) Which address bit are used to address the cache directories?

(*e*) Which address bits are appended to address bits in (*d*) to address the cache array?

[MDU, B.E(CSE)–7th Sem., May 2009]

ns. (a)



(b) Four (4) address bits.
(c) W/L address bits,
(d) B/W and Index
(e) Index & W/L

12. **Suppose two processors (in a Multiprocessor System) make a total of exactly two references to memory, every memory cycle (T_c = 100ms). The memory consists of 8 low-order interleaved memory modules. Find:**
(a) Expected waiting time
(b) Total Access Time.
(c) Mean total number of queued requests.
(d) Offered memory bandwidth (references/sec)
(e) Achieved memory bandwidth. **[MDU, BE(CSE)–7th Sem., May 2009]**

Ans. (a)

$$T_w = \frac{1}{\lambda}\left[\frac{p^2 - p^p}{2(1-p)}\right]$$

$$= \frac{1}{\lambda}\left(\frac{1}{\mu}\right)\left[\frac{p-p}{2(1-p)}\right]$$

$$= \frac{1}{\lambda}\left[\frac{p-p}{2(1-p)}\right] = \left[\frac{p-p}{2(1-p)}\right] \times T_s$$

(b) $$T = T_s + T_w$$

Where T_w – Waiting time
T_s – Average Service function.