b) Suppose the cache has the following design parameters.
Line size (L) = 8 ; Set size (k) = 4; No of sets (N) = 512;
word size = 32 bits; M.M address space = 4M words.

a) What is the total cache size? ⇒ $K \times L \times N = 4 \times 8 \times 512 = 16384 = 16K$.

b) What is the M.M address length? ⇒ $4M = 2^{22} = 22$ bits

c) How many bits of M.M address need to specify tag } ⇒ $3 + 9 + x = 22$, $x = 10$.

d) How many diff lines in M.M can map to the 4 lines in the set #12 } 2^{10} lines

e) How many tag comparators are needed? $K = 4$.

f) If the cache access time is 12ns and the M.M access time is 100ns what hit rate is needed to give an effective access time of the cache mem sys of 18ns?
} $18ns = 12ns + (1-h)\,100$, $h = 94\%$.

) If the cache access time is 15ns and hit rate of 95%, what M.M access time is needed to give an effective access time of the cache mem sys of 22ns
} $22 = 15ns + 0.05 * T_M$, $T_M = 140ns$.

) Implement the classical test-and-set instⁿ using Load Linked/Store Conditional instⁿ pair.

```
L:  MOV  R3, #1
    LL   R2, 0(R1)
    SC   R3, 0(R1)
```
} This code would put in a loop that spins until 1 is returned in R3.

2285 Lets vote. in
8003

* Prof Hema Murthy * waran.
* Prof G Koshy Varghese
* Surya narayan. Civil Prof Computer
* John Agasty
Nandi wade

Unit - 3 Problems

① Suppose you want to achieve speedup of 80 with 100 processors. What fraction of original computation can be sequential?

Ans: Speedup $= \dfrac{1}{(1 - FE + \frac{FE}{SE})}$

Assume the prog operates in 2 modes - Parallel / Serial

Parallel mode → enhanced mode.

no of processors → speedup enhanced.

$$80 = \frac{1}{(1 - \text{Fraction}_{parallel} + \frac{\text{fraction}_{parallel}}{100}}$$

$$0.8 \times \text{Fraction}_{ill} + 80(1 - \text{Fraction}_{ill}) = 1$$

$$80 - 79.2\,\text{Fraction}_{ill} = 1 \Rightarrow \text{Fraction}_{ill} = \frac{80-1}{79.2}$$

$$\text{Fraction}_{ill} = 0.9975$$

To achieve the speedup of 80 with 100 processors only 0.25% of the computⁿ can be sequential.

② A prog repeatedly executes a loop that has 120 iteratⁿ. Each iteration takes 10,000 cycles. on a multiprocessor sys 50,000 cycles are req to sync the processors once all the iterations of the loop have completed.

(a) What is the execution time (of each loop) on a Uniprocessor sys

(b) -"- on a 2-processor sys. What is the speedup over the uniprocessor sys

(c) -"- 4 processor sys. -"-

$$\boxed{\text{Speedup} = \frac{Ex_{old}}{Ex_{new}}}$$

Ans: (a) 120 iteratⁿ * 10000 = 1,200,000 cycles

(b) on a 2-processor sys, each one handle 60 iterⁿ. Base execution time 600,000 cycles. Adding commⁿ time gives 650,000 cycles

(c) Each processor executes 30 iteration for an execution time of 350,000 cycles. Speedup = 3.43.

) A message passing prog. executes on 2 processors. In this sys the delay from when the message is sent until when it is available to be received on the destination process is 1000 cycles. It takes 500 cycles to complete a RECEIVE opn if the message being received is available

(a) What is the shortest possible delay b/w when a message is sent and when the contents of the message are available for use by the prog. on the destn processor

(b) On cycle 100 the prog running on processor 0 sends a message to processor 1. On cycle 200 the prog running on process 1 executes a RECEIVE opn to receive the message. When does the RECEIVE complete?

) When would RECEIVE opn from part (b) of this exercise complete if the prog running on processor 1 execute the RECEIVE on cycle 2000 instead of 200.

Ans:-

(a) 1500 cycles. It takes 1000 cycles for message to arrive at the destn process and another 500 cycles for it to be received by the prog. on destn processor.

(b) RECEIVE opn blocks until the message arrives at process 1 then takes 500 cycles to complete. Since the message was sent on 100 cycle, it arrives at processor 1 on cycle 1100. RECEIVE completes on 1600.

(c) The message still (receive) arrives process 1 on cycle 1100. Since it was sent on cycle 100. The prog running on process 1 waits until 2000 cycle to execute RECEIVE opn. (when) The RECEIVE completes on cycle 2500.

(4) A prog consists of 5 tasks which have execution times of 2000, 4000, 6000, 8000 and 10,000 cycles. It's not possible to divide the execution of one task among multiple processors, but there are no comm^n and sync^n costs. If the tasks are distributed across the processors to achieve the shortest execution time. What is the speedup for executing the prog. on:

(a) 2 processors (b) 4 processors (c) 8 processors.

Ans:- On one processor the prog execution time is 30,000 cycles. The sum of execution times of the tasks.

(a) The most even division of tasks among the processors gives 16000 cycles of work to one processor and 14000 cycles of work to another. So execution time of 2 processors is 16000 cycles. Speedup = 1.875

(b) On 4-processors, the most even distribution assigns 10000 cycles of work on one processor, 8000 to the second and 6000 to each of the other two. This gives execution time of 10000 cycles. for a speedup of 3.

(c) On an 8 processor m/c the best division is to assign one task to each processor (leaving 3 with no work to do) which gives an execut^n time of 10,000 cycles (limited by longest task) and a speedup of 5

(5) Suppose an application is running on a 32-processor multiprocessor, which has 200 ns time to handle ref to a remote memory. For this prog assume that all the ref except those involving comm^n hit in local mem hierarchy which is slightly optimistic. Processors are stalled on a remote ref. The processor clk rate is 3.3 GHz. If the base CPI (assuming that all ref hit in the cache) is 0.5, how much faster is the multiprocessor if there is no communication Vs if 0.2% of the instn involve a remote comm^n ref?

Ans:- The effective CPI for the multiprocessor with 0.2% remote ref is

CPI = Base CPI + Remote ref rate * Remote ref cost

= 0.5 + 0.2% * Remote ref cost.

Remote ref cost = $\frac{\text{Remote (ref) access cost}}{\text{cycle time}} = \frac{200\text{ns}}{0.3\text{ns}} = 666$ cycles

CPI = 0.5 + 0.2 = 1.7

The multiprocessor with all local ref is 1.7/0.5 = 3.4 times faster.