# Indian Institute of Technology (IIT-Bombay)

**AUTUMN Semester, 2025**
**COMPUTER SCIENCE AND ENGINEERING**

**CS230: Digital Logic Design and Computer Architecture**

**Tutorial - V**

**Full Marks: 0**

**Time allowed: $\infty$ hours**

**1.** Given that a CPU has 32 bit word address and 256 KB cache memory. The cache is organized as a 4 way set associative cache with cache block size of 16 bytes

- What is the number of sets in the cache?
- What is the size (in bits) of the tag field per cache block?
- What is the number and size of comparators required for tag matching?
- How many address bits are required to find the byte offset within a cache block?
- What is the total amount of extra memory (in bytes) required for the tag bits?

**Solution:** A CPU has a 32-bit address. The cache parameters are:

- Cache size $C = 256KB = 256 \times 1024 = 262144$ bytes,
- Associativity $A = 4$ (4-way set associative),
- Block size $B = 16$ bytes,
- Address width = 32 bits.

**Assumption:** addresses are byte-addressable (standard interpretation).

(1) Number of sets ($S$): $S = \dfrac{C}{A \cdot B} = \dfrac{262144}{4 \cdot 16} = \dfrac{262144}{64} = 4096 = 2^{12}$.

(2) Number of offset bits (block byte offset) ($b$): $b = \log_2 B = \log_2 16 = 4$ bits.

Index bits ($s$): $s = \log_2 S = \log_2 4096 = 12$ bits.

Tag bits per block ($t$): $t = \text{Address bits} - s - b = 32 - 12 - 4 = 16$ bits.

(3) Comparators for tag matching: One comparator per way in a set.
Number of comparators per set $= A = 4$.
Each comparator must compare $t = 16$ bits.

(4) Byte offset bits (within block): $b = 4$ bits.

(5) Total extra memory required for tag bits:

Number of blocks ($N_b$): $N_b = \dfrac{C}{B} = \dfrac{262144}{16} = 16384$.

Total tag bits: $N_b \cdot t = 16384 \cdot 16 = 262144$ bits.

In bytes: $\dfrac{262144}{8} = 32768$ bytes $= 32$ KB.

## Final answers

(a) **Number of sets:** 4096 sets (= $2^{12}$).

(b) **Tag field size per cache block:** 16 bits.

(c) **Comparators for tag matching:** 4 comparators (one per way), each comparing 16 bits.

(d) **Address bits for byte offset within block:** 4 bits.

(e) **Total extra memory for tag bits:** 32768 bytes = 32 KB (i.e. 262144 bits).

**2.** 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. What is the range of set lines that a main memory block numbered *j* can be mapped to ?

**Solution:**

Number of sets in cache = *v*.
For set 0, the cache lines are numbered *0, 1, . . , k - 1.*
Now for set 1, the cache lines are numbered *k, k + 1, . . . k + k - 1* and so on.
So, main memory block j will be mapped to set *(j mod v)* , which will be any one of the cache lines from *(j mod v) * k to (j mod v) * k + (k - 1)*

**3.** An 8KB direct-mapped write-back cache is organized as multiple blocks, each of size 32-bytes. The processor generates 64-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?

**Solution:**

$$\text{Main Memory size} = 2^{64} \text{ Bytes}$$
$$\text{Cache size} = 8\text{KB} = 2^{13} \text{ Bytes}$$
$$\text{Block size} = 2^{5} \text{ Bytes}$$
$$\text{Valid bit} = 1 \text{ bit}$$
$$\text{Modified bit} = 1 \text{ bit}$$

**Formula:**

$$\text{Number of bits} = \lceil \log_2 n \rceil$$

$$\text{Number of lines in cache} = \frac{\text{Cache size}}{\text{Block size}} = \frac{2^{13}}{2^5} = 2^8$$

$$\text{Main Memory Address (MM)} = \text{Tag} + \text{Index} + \text{Block Offset}$$

$$\text{Total bits required to store meta-data of 1 line} = \text{Valid bit} + \text{Modified bit} + \text{Tag bits}$$

**Calculation:**

$$\text{Number of cache lines} = \frac{\text{Cache size}}{\text{Block size}} = \frac{2^{13}}{2^5} = 2^8$$

$$\therefore \text{Index bits} = 8$$

$$64 = \text{Tag bits} + \text{Index bits} + \text{Block Offset bits}$$

$$64 = \text{Tag bits} + 8 + 5$$

$$\therefore \text{Tag bits} = 51$$

| Tag (51 bits) | Index (8 bits) | Block Offset (5 bits) |
|---|---|---|

**Meta-data Calculation:**

$$\text{Total bits per line} = 1(\text{Valid}) + 1(\text{Modified}) + 51(\text{Tag}) = 53 \text{ bits}$$

$$\text{Total number of cache lines} = 2^8 = 256$$

Total bits required for meta-data $= 53 \times 256 = 13568$ bits

$\boxed{\text{Total memory required for meta-data} = 13568 \text{ bits}}$

**4.** 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 as given in Fig. 1. The block size in L2 cache is 16 words. The memory access times are 5 nanoseconds, 15 nanoseconds, and 200 nanoseconds for L1 cache, L2 cache, and main memory unit respectively. When there is a miss in both L1 cache and L2 cache, first a block is transferred


Figure 1

from main memory to L2 cache, and then a block is transferred from L2 cache to L1 cache. What is the total time taken for these transfers?

**Solution:**

$$\begin{aligned}\text{Block size in L1 cache} &= 4 \text{ words}\\ \text{Block size in L2 cache} &= 16 \text{ words}\\ \text{Access time for L1 cache} &= 5 \text{ ns}\\ \text{Access time for L2 cache} &= 15 \text{ ns}\\ \text{Access time for main memory} &= 200 \text{ ns}\end{aligned}$$

**Calculation:**

**Step 1:** Time taken for a block to be transferred from the main memory to the L2 cache is denoted as $T_{mt}$.

As given, the block size of the L2 cache is 16 words.

The bandwidth between **main memory and L2 cache** is 4 words.

Hence, we require 4 memory accesses (for read) and 4 L2 cache accesses (for store).

$$T_{mt} = 4 \times (\text{Memory Access Time} + \text{L2 Cache Access Time})$$

$$T_{mt} = 4 \times (200 + 15) = 860 \text{ ns}$$

**Step 2:** Time taken for a block to be transferred from the L2 cache to the L1 cache is denoted as $T_{ct}$.

$$T_{ct} = \text{Time to access L2 cache} + \text{Time to store in L1 cache}$$

$$T_{ct} = 15 \text{ ns} + 5 \text{ ns} = 20 \text{ ns}$$

**Step 3:**

$$\text{Total Time} = T_{mt} + T_{ct}$$

$$\text{Total Time} = 860 \text{ ns} + 20 \text{ ns} = \boxed{880 \text{ ns}}$$

**5.** The baseline execution time of a program on a 6 $GHz$ single-core machine is 100 nanoseconds (ns). The code corresponding to 90% of the execution time can be fully parallelized. The overhead for using an additional core is 10 ns when running on a multicore system. Assume that all cores in the multicore system run their share of the parallelized code for an equal amount of time. What number of cores minimizes the execution time of this program?

**Solution:**

**Given:** $T_1 = 100$ ns, $f_p = 0.9$, $f_s = 0.1$, $\Delta = 10$ ns per additional core.

Using Amdahl's law with overhead, the execution time on $k$ cores is

$$T(k) = f_s T_1 + \frac{f_p T_1}{k} + \Delta(k-1).$$

Substitute numbers: $f_s T_1 = 0.1 \cdot 100 = 10$ ns and $f_p T_1 = 0.9 \cdot 100 = 90$ ns, so

$$T(k) = 10 + \frac{90}{k} + 10(k-1) = 10k + \frac{90}{k}.$$

Minimise for real $k > 0$: differentiate and set to zero

$$\frac{dT}{dk} = 10 - \frac{90}{k^2} = 0 \quad \Rightarrow \quad k^2 = 9 \quad \Rightarrow \quad k = 3.$$

Check neighbouring integers: $T(2) = 20 + 45 = 65$ ns, $T(3) = 30 + 30 = 60$ ns, $T(4) = 40 + 22.5 = 62.5$ ns. Thus the integer $k$ that minimises $T(k)$ is 3 cores

**6.** A non-pipelined instruction execution unit operating at 2 $GHz$ takes an average of 6 cycles to execute an instruction of a program $P$. The unit is then redesigned to operate as a 5-stage pipeline at 2 $GHz$. Assume the ideal throughput of the pipelined unit is 1 instruction per cycle. In the execution of program $P$, 20% of instructions incur an average of 2 cycles stall due to data hazards, and 20% of instructions incur an average of 3 cycles stall due to control hazards. The speedup (rounded to one decimal place) obtained by the pipelined design over the non-pipelined design is _____.

**Solution**

Let $\text{CPI}_{\text{non}}$ be the cycles per instruction for the non-pipelined design and $\text{CPI}_{\text{pipe}}$ for the pipelined design.

$$\text{CPI}_{\text{non}} = 6.$$

For the pipelined design the ideal CPI is 1, plus average stall cycles per instruction:

$$\text{average stalls} = 0.20 \times 2 + 0.20 \times 3 = 0.4 + 0.6 = 1.0.$$

Hence

$$\text{CPI}_{\text{pipe}} = 1 + 1.0 = 2.$$

Since both designs run at the same clock frequency, speedup is

$$\text{Speedup} = \frac{\text{CPI}_{\text{non}}}{\text{CPI}_{\text{pipe}}} = \frac{6}{2} = 3.0.$$

**7.** A processor uses a two-level cache hierarchy. The L1 cache has an access time of 2 ns, and the L2 cache has an access time of 10 ns. The main memory access time is 100 ns. The hit rates of L1 and L2 caches are 90% and 80%, respectively. Assume that an access satisfied by L1 cache takes 2 ns, an access satisfied by L2 cache takes 10 ns, and a main memory access (after both misses) includes both L2 and main memory access times.

(a) Calculate the average memory access time (AMAT).

(b) If the L1 hit rate improves by 5% (to 95%), what percentage reduction is achieved in AMAT?

**Solution:**

$$T_{L1} = 2 \text{ ns}, \quad T_{L2} = 10 \text{ ns}, \quad T_{mem} = 100 \text{ ns}$$
$$h_{L1} = 0.90, \quad h_{L2} = 0.80$$

If both caches miss, the total time is

$$T_{miss\_both} = T_{L2} + T_{mem} = 10 + 100 = 110 \text{ ns}$$

**(a) Average Memory Access Time (AMAT):**

$$\begin{aligned} P_{L1-hit} &= 0.90 \\ P_{L1-miss} &= 0.10 \\ P_{L2-hit} &= P_{L1-miss} \times 0.80 = 0.08 \\ P_{miss\ both} &= P_{L1-miss} \times (1 - 0.80) = 0.02 \end{aligned}$$

$$\text{AMAT} = (0.90)(2) + (0.08)(10) + (0.02)(110) = 1.8 + 0.8 + 2.2 = 4.8 \text{ ns}$$

$$\boxed{\text{AMAT} = 4.8 \text{ ns}}$$

**(b) With improved L1 hit rate (95%):**

$$P_{L1-hit} = 0.95, \quad P_{L1-miss} = 0.05$$
$$P_{L2-hit} = 0.05 \times 0.80 = 0.04, \quad P_{miss\ both} = 0.01$$

$$\text{AMAT}_{new} = (0.95)(2) + (0.04)(10) + (0.01)(110) = 1.9 + 0.4 + 1.1 = 3.4 \text{ ns}$$

Percentage reduction in AMAT:

$$\frac{4.8 - 3.4}{4.8} \times 100 = 29.17\%$$

$$\boxed{\text{New AMAT} = 3.4 \text{ ns}, \quad \text{Reduction} = 29.17\%}$$

**8.** A 5-stage pipelined processor operates at 3 GHz. On average, 10% of instructions are branch instructions. Each branch instruction causes a 2-cycle pipeline flush. All other instructions are complete without stalls.

(a) Compute the average CPI (cycles per instruction).

(b) If a branch predictor is added that correctly predicts 80% of the branches, what is the new average CPI?

(c) Determine the speedup obtained by using the branch predictor.

**Solution:**

$$f_b = 0.10, \quad \text{Penalty} = 2 \text{ cycles}, \quad \text{Ideal CPI} = 1$$

**(a) Without branch predictor:**

$$\text{CPI} = 1 + f_b \times \text{Penalty} = 1 + 0.10 \times 2 = 1.20$$

$$\boxed{\text{CPI}_{no\ predictor} = 1.20}$$

**(b) With branch predictor (80% accuracy):**

$$\text{Misprediction rate} = 1 - 0.80 = 0.20$$

$$\text{CPI} = 1 + f_b \times \text{Misprediction rate} \times \text{Penalty}$$

$$\text{CPI} = 1 + 0.10 \times 0.20 \times 2 = 1.04$$

$$\boxed{\text{CPI}_{with\ predictor} = 1.04}$$

**(c) Speedup due to branch predictor:**

$$\text{Speedup} = \frac{\text{CPI}_{no\ predictor}}{\text{CPI}_{with\ predictor}} = \frac{1.20}{1.04} = 1.1538$$

$$\boxed{\text{Speedup} \approx 1.15\times \text{ (about 13.3\% reduction in execution time)}}$$

**9.** Neel, a junior engineer on a processor design team, is analyzing a 5-stage pipeline. The latencies are:

| Stage | Latency |
|---|---|
| Fetch | 300 ps |
| Decode | 400 ps |
| Execute | 350 ps |
| Memory | 550 ps |
| Writeback | 100 ps |

Each pipeline register adds an overhead of **20 ps**.

Neel correctly identifies the 'Memory' stage (550 ps) as the main bottleneck, limiting the processor's clock speed. To fix this, he proposes **splitting the 'Memory' stage into two new, equal stages: 'Mem1' and 'Mem2'**.

(Assume all cache accesses hit; a miss would stall the pipeline for 12,000 ps).

Did Neel's change have the effect he wanted? Verify his work by calculating the **cycle time**, **latency of one instruction**, and the **theoretical speedup** of his new 6-stage processor compared to the original *non-pipelined* version.

Which of these results is correct?

(a) Cycle time = 570ps, Latency = 2850ps, Speedup = 2.98

(b) Cycle time = 420ps, Latency = 2100ps, Speedup = 4.05

(c) Cycle time = 420ps, Latency = 2520ps, Speedup = 6.0

(d) Cycle time = 420ps, Latency = 2520ps, Speedup = 4.05

**Solution:**

**1. Calculate New Cycle Time:** Neel splits the 550 ps 'Memory' stage into 'Mem1' (275 ps) and 'Mem2' (275 ps). The new list of stage latencies is: {300, 400, 350, 275, 275, 100}. The new slowest stage is 'Decode' at 400 ps (this is the key trap).

$$\text{Cycle Time} = (\text{New Slowest Stage}) + (\text{Register Overhead})$$

$$\text{Cycle Time} = 400 \text{ ps} + 20 \text{ ps} = \boxed{420 \text{ ps}}$$

**2. Calculate New Latency:** The new design now has $k = 6$ stages (Fetch, Decode, Execute, Mem1, Mem2, Writeback).

$$\text{Latency} = k \times \text{Cycle Time}$$

$$\text{Latency} = 6 \times 420 \text{ ps} = \boxed{2520 \text{ ps}}$$

**3. Calculate Speedup:** Speedup is the ratio of the original non-pipelined time to the new pipelined time (cycle time). The non-pipelined time is the sum of the *original* stages.

$$\text{Time}_{\text{non-pipelined}} = 300 + 400 + 350 + 550 + 100 = 1700 \text{ ps}$$

The pipelined time per instruction is the new cycle time.

$$\text{Time}_{\text{pipelined}} = \text{New Cycle Time} = 420 \text{ ps}$$

$$\text{Speedup} = \frac{\text{Time}_{\text{non-pipelined}}}{\text{Time}_{\text{pipelined}}} = \frac{1700 \text{ ps}}{420 \text{ ps}} \approx 4.0476$$

Speedup $\approx \boxed{4.05}$

The correct results are: Cycle time = 420ps, Latency = 2520ps, Speedup = 4.05. This corresponds to **Option D**.

---