ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

Institute of Technology of Cambodia
Department of Foundation Year

TP04

GROUP: I2-GIC-2B

Name of Students	ID of Students	Score
1. SOUNG VICHKA	e20240672	…….

Lecturer: Mr. ROEUN Pacharoth

Cache Memory — Theory & Exercises

Theory Questions

Q1: L1 vs L2 Cache

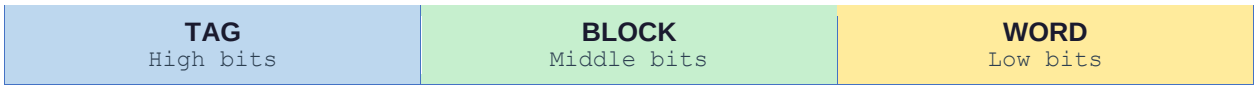
Speed: L1 is faster than L2.

Size: L1 is smaller than L2.

Why smaller? L1 sits directly on the CPU die, where silicon space is extremely expensive. The closer the cache is to the processor, the faster it operates — but physical space on the chip strictly limits its size. L2 is larger but sits slightly further away, introducing more wire delay.

Q2: Three Fields in a Direct-Mapped Cache Address

A memory address is divided into three fields:



Q3: Associative Memory vs Regular Memory

Regular memory is address-accessed: you specify a location and retrieve its contents.

Associative (content-addressable) memory is value-accessed: you broadcast a key and the hardware searches all entries simultaneously for a match.

Associative memory is more expensive because every storage cell requires its own comparison circuit. Regular memory needs only an address decoder — a far simpler and cheaper structure.

Q4: Fully Associative vs Direct-Mapped Cache

Direct-mapped: Each memory block maps to exactly one cache slot (determined by the block field). Simple and fast, but prone to conflict misses when two blocks compete for the same slot.

Fully associative: Any memory block can be placed in any cache slot. There is no block index field — the hardware compares the tag against every slot simultaneously (using associative memory). This eliminates conflict misses but is expensive due to the parallel comparison hardware required.

Exercise 1a — Direct Mapped Cache

Given: **2^{20} words** of main memory | **32 cache blocks** | **16 words/block**

1. Number of Main Memory Blocks

Formula	Total words / Words per block = 2^{20} / 2^{4}
Answer	**2^{16} = 65,536 blocks**

2. Address Format

Word field = $\log_2(16)$ = **4 bits** | Block field = $\log_2(32)$ = **5 bits** | Tag = 20 − 5 − 4 = **11 bits**

TAG (11 bits)	BLOCK (5 bits)	WORD (4 bits)

↓

20 bits

3. Mapping 0x0DB63

Hex address	`0x0DB63`
Binary	`0000 1101 1011 0110 0011`
Word (bits 3–0)	`0011 = 3`
Block (bits 8–4)	`00110 =` **6**
Cache block	**Block 6**

Exercise 1b — Direct Mapped Cache

Given: **2^{32} words** of main memory | **1024 cache blocks** | **32 words/block**

1. Number of Main Memory Blocks

Formula	2^{32} / 2^{5} = 2^{27}
Answer	**2^{27} = 134,217,728 blocks**

2. Address Format

Word field = $\log_2(32)$ = **5 bits** | Block field = $\log_2(1024)$ = **10 bits** | Tag = 32 − 10 − 5 = **17 bits**

TAG (17 bits)	BLOCK (10 bits)	WORD (5 bits)

↓

20 bits

3. Mapping 0x000063FA

Hex address	`0x000063FA`
Binary	`0000 0000 0000 0000 0110 0011 1111 1010`
Word (bits 4–0)	`1 1010 = 26`
Block (bits 14–5)	`00 0001 1111 =` **31**
Cache block	**Block 31**

Exercise 2a — Fully Associative Cache

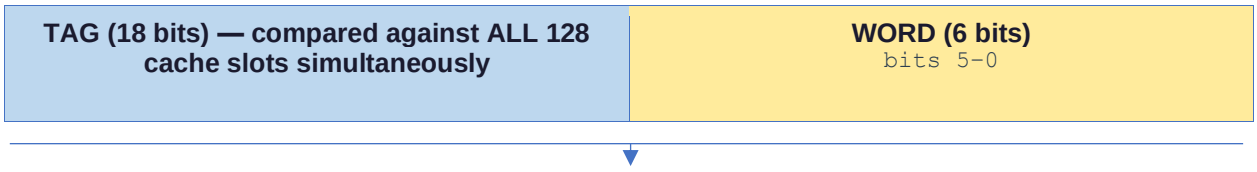
Given: **2^{24} words** of main memory | **128 cache blocks** | **64 words/block**

1. Number of Main Memory Blocks

Formula	$2^{24} / 2^{6} = 2^{18}$
Answer	**2^{18} = 262,144 blocks**

2. Address Format (No Block Field)

Word field = $\log_2(64)$ = **6 bits** | No block index field | Tag = 24 − 6 = **18 bits**



24 bits

3. Mapping 0x01D872

Hex address	`0x01D872`
Binary	`0000 0001 1101 1000 0111 0010`
Word (bits 5–0)	`11 0010 = 50`
Tag (bits 23–6)	`0x0762`
Cache block	Fully associative — **any free block** may be used

The tag `0x0762` is broadcast to all 128 comparators simultaneously on a lookup.

Exercise 2b — Fully Associative Cache

Given: **2^{16} words** of main memory | **64 cache blocks** | **32 words/block**

1. Number of Main Memory Blocks

Formula	$2^{16} / 2^5 = 2^{11}$
Answer	**2^{11} = 2,048 blocks**

2. Address Format (No Block Field)

Word field = $\log_2(32)$ = **5 bits** | No block index field | Tag = 16 − 5 = **11 bits**

TAG (11 bits) — compared against ALL 64 cache slots simultaneously `bits 15–5`	**WORD (5 bits)** `bits 4–0`

3. Mapping 0xF8C9

Hex address	`0xF8C9`
Binary	`1111 1000 1100 1001`
Word (bits 4–0)	`0 1001 = 9`
Tag (bits 15–5)	`111 1100 0110 = 0x7C6`
Cache block	Fully associative — **any of the 64 slots** may be used

Tag `0x7C6` is broadcast to all 64 comparators simultaneously on a lookup.