

Tutorial 12: Cache

Problem 1: Direct Mapped Cache

Consider a 128KB of data in a direct-mapped cache with 16 word blocks. Determine the size of the tag, index and offset fields if a 32-bit architecture (ie. 32 address lines) is employed and memory is only word addressable. Redo the calculation if memory is byte addressable.

Word Offset:

To specify the correct word from 16 words (2^4 words) in a block we need 4 bits.

Block Index:

Cache contains 128KB = 2^{17} bytes

One cache block contains 16 words = 64 bytes = 2^6 bytes

No. of rows in cache = (No. of blocks in cache) = (Total no. of bytes in cache) / (No. of bytes in block) = $2^{17} / 2^6 = 2^{11}$ rows/cache

To select 1 out of 2^{11} rows in the cache 11 bits are needed.

Tag Field:

The number of bits for the tag field is (Total No. of Address bits) - (Block Index) - (Word Offset) - (Byte Offset) = $32 - 11 - 4 - 2 = 15$.

The Organisation for this cache is shown in Figure 1.

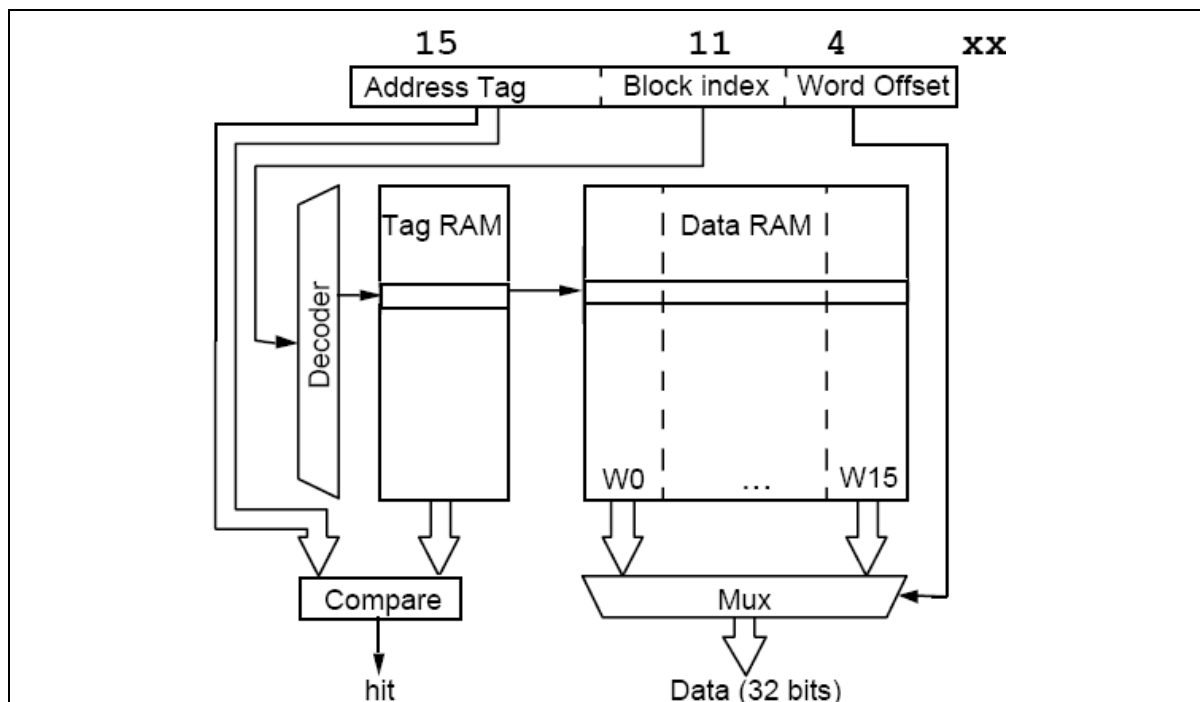


Figure 1: Organisation for 128KB Direct Mapped Cache with 16-word Block Size and Word Addressable

For byte addressable cache we need to select 1 byte in 16 words (64 bytes = 2^6 bytes). This increases byte offset to 6 bits. The Organisation for this byte-addressable cache is shown in Figure 2.

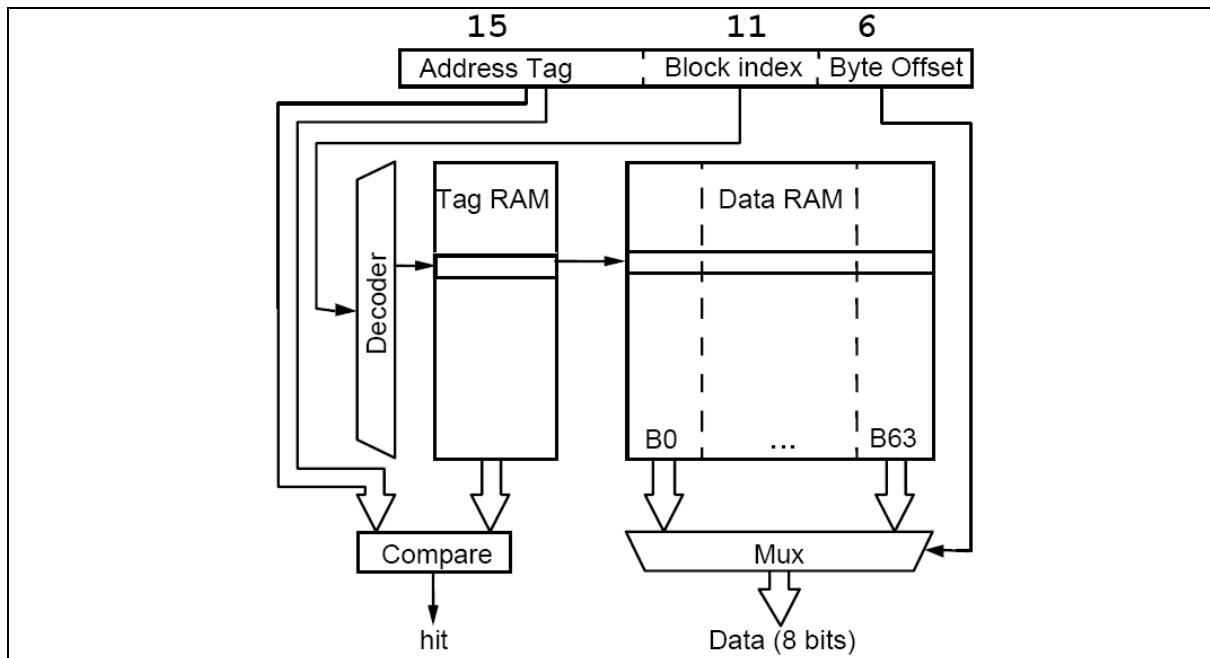


Figure 2: Organisation for 128KB Direct Mapped Cache with 16-word Block Size and Byte Addressable

Problem 2: Data Access in Direct Mapped Cache

Consider a 128KB of data in a direct-mapped cache with 16 word blocks, for a 32-bit address architecture and word addressable memory. Analyse the data access pattern to memory and cache for the accesses to memory locations 0x00000014, 0x0000001C, 0x000001F4, and 0x8000002C.

Figure 3 presents an example memory map for the system under consideration. Memory locations 0x00000014, 0x0000001C, 0x000001F4, and 0x8000002C and their contents are highlighted in blue.

Memory	
Address	Value of Word
...	...
00000010	e
00000014	f
00000018	g
0000001c	h
...	...
000001f0	χ
000001f4	δ
000001f8	ε
000001fc	φ
...	...
80000024	π
80000028	θ
8000002c	ρ
80000030	σ
...	...

Figure 3: The Memory Map for with the Addresses and Data Items for the Accessed Data Highlighted

Figure 4 illustrates division of the address bits into word offset (4 bits), block index (11 bits), and cache tag (15 bits) for the 4 memory addresses. Note that the first 2 bits of the address are not used as the memory is only word addressable.

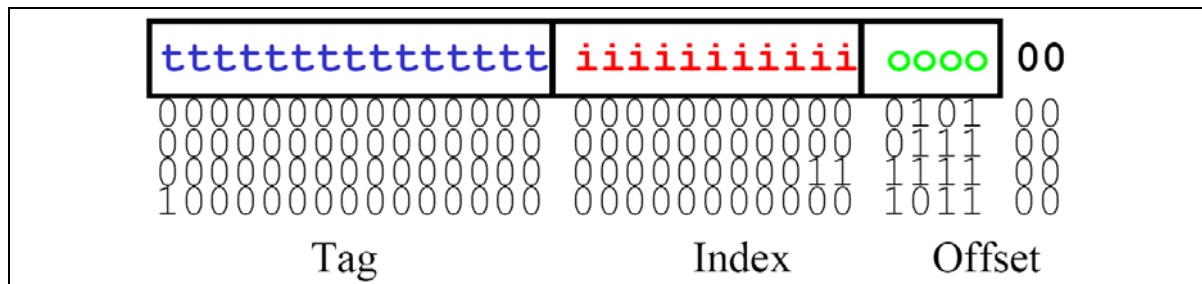


Figure 4: Word Offset, Block Index, and Tag Fields for the 4 Memory Addresses

Figure 5 illustrates organisation of the cache with 16 word blocks and 2048 blocks. In addition to the storage for 16 words, each block of the cache has a bit to indicate if the cache line is valid and a field for the storage of the tag for the memory block currently stored in that block.

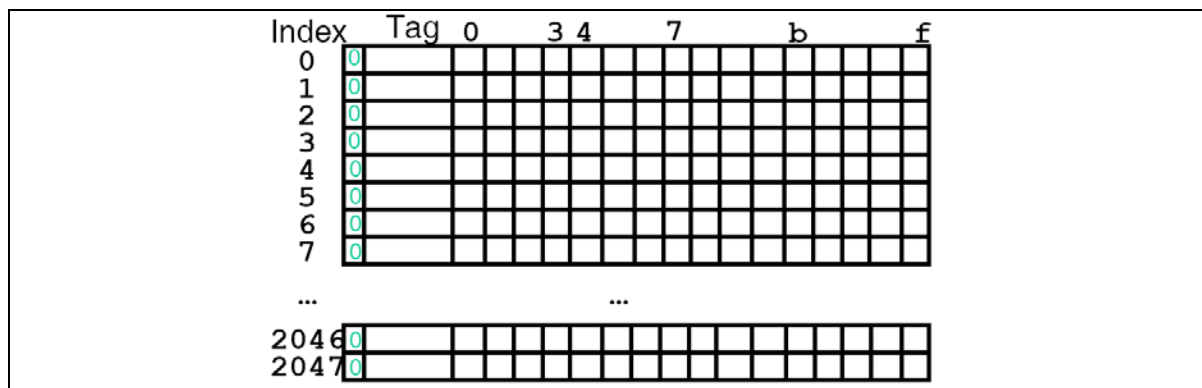


Figure 5: Cache Organisation of a 128KB Cache with 16 Word Blocks and the Tag and Valid Bit Overheads

When the memory location 0x00000014 is accessed, the block index selects cache row 0 and finds the valid bit set to 0. This is called a cache miss as there is nothing in the cache. This cache miss results in transfer of the 16 word memory block from the memory to cache line 0 as shown in Figure 6. After the block transfer, the valid bit is changed to 1, tag field of the address is stored in the tag field of the cache line 0, and the word offset field is used to read the correct word "f".

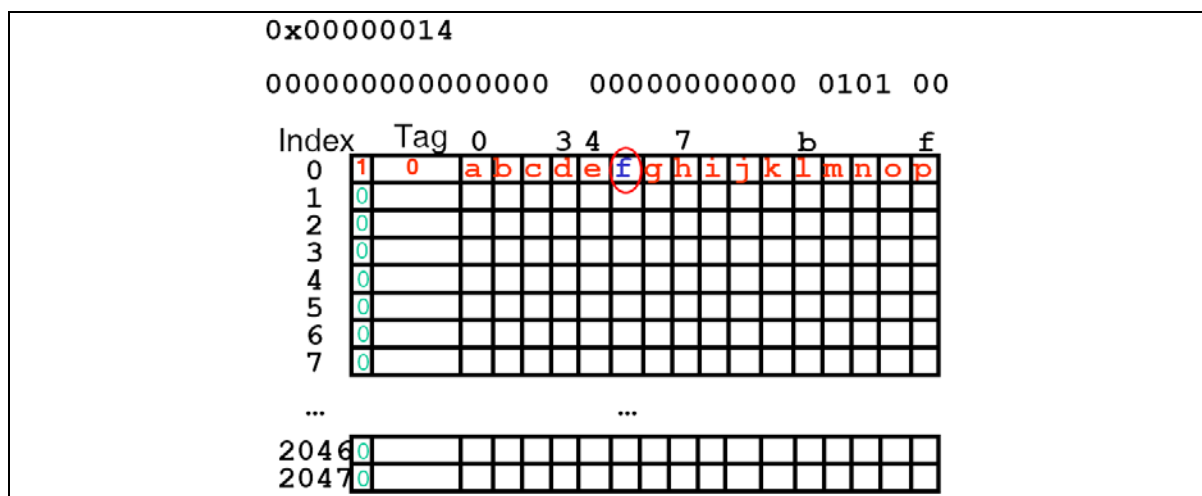


Figure 6: Cache Update after Access to Memory Location 0x00000014

When the memory location 0x0000001C is accessed, the block index selects cache row 0 and finds the valid bit set to 1. Next tag field of the address is compared with the tag field stored in the cache. Identical values of 0 results in a hit and the word offset is used to get the desired value "h". As it is a hit, there is not need to transfer the block from the memory. The cache state after access to memory location 0x0000001C is depicted in Figure 7.

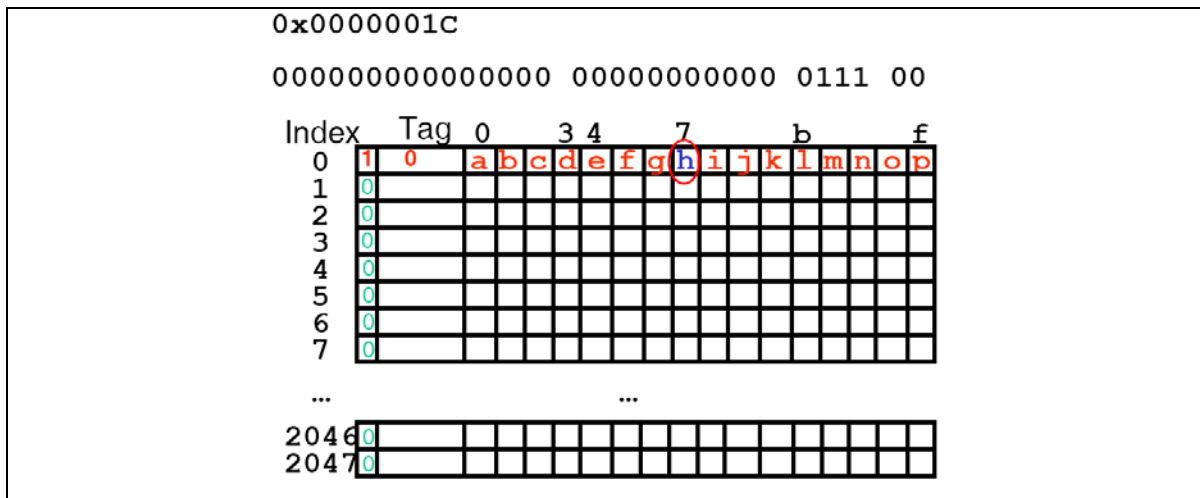


Figure 7: Cache Update after Access to Memory Location 0x0000001C

When the memory location 0x000001F4 is accessed, the block index selects cache row 7 and finds the valid bit set to 0. This is a called cache miss as there is nothing in the cache. This cache miss results in transfer of the 16 word memory block from the memory to cache line 0 as shown in Figure 8. After the block transfer, the valid bit is changed to 1, tag field of the address is stored in the tag field of the cache line 7, and the word offset field is used to read the correct word "w".

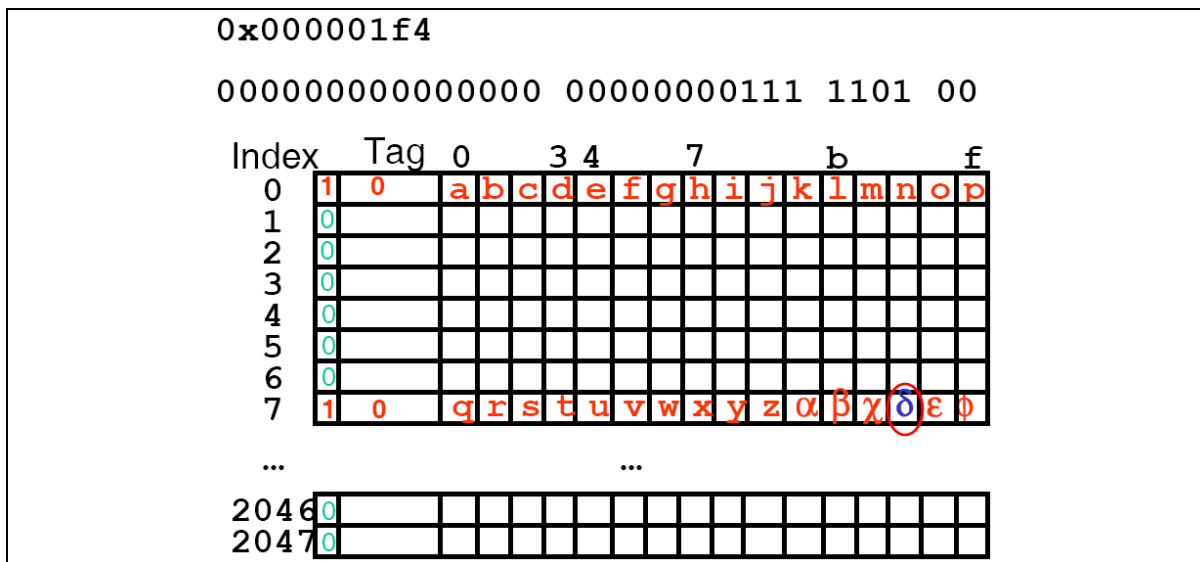


Figure 8: Cache Update after Access to Memory Location 0x000001F4

When the memory location 0x8000002C is accessed, the block index selects cache row 0 and finds the valid bit set to 1. Next tag field of the address (0x4000) is compared with the tag field stored in the cache (0x0000). Different values of the tag field results in cache miss with replacement, as the content of the cache lines needs to be replaced with a block. This cache miss results in transfer of the 16 word memory block from the memory to cache line 0 as shown in Figure 8. After the block transfer, tag field of the address (0x4000) is stored in the tag field of the cache line 0, and the word offset field is used to read the correct word "e".

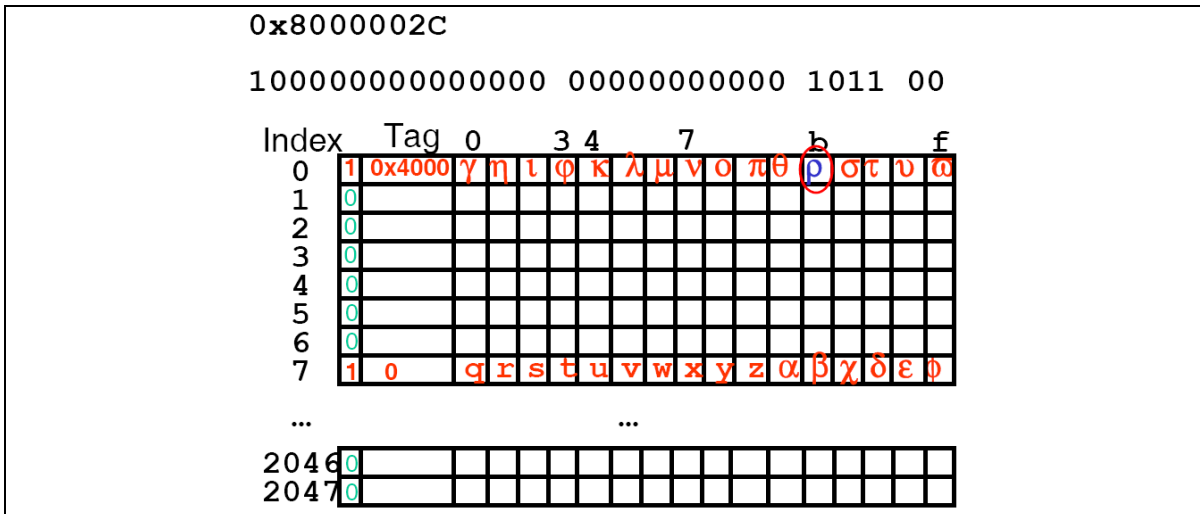


Figure 9: Cache Update after Access to Memory Location 0x8000002C

Problem 3: 2-Way Set-Associated Cache

Consider a 128KB of data in a 2-way set associative cache with 16 word blocks. Determine the size of the tag, index and offset fields if a 32-bit architecture (ie. 32 Address lines) is employed and memory is only word addressable. Redo the calculation if memory is byte addressable.

Word Offset:

To specify the correct word from 16 words (2^4 words) in a block we need 4 bits.

Block Index:

Cache contains 128KB = 2^{17} bytes

One cache block contains 16 word = 64 bytes = 2^6 bytes

No. of rows in cache = (No. of blocks in cache) = (Total No. of bytes in cache) / ((No. of bytes in block) \times (Set Associativity)) = $2^{17} / (2^6 \times 2) = 2^{10}$ rows/cache

To select 1 out of 2^{10} rows in the cache 10 bits are needed.

Tag Field:

The number of bits for the tag field is (Total No. of Address bits) - (Block Index) - (Word Offset) - (Byte Offset) = $32 - 10 - 4 - 2 = 16$.

The Organisation for this cache is shown in Figure 10.

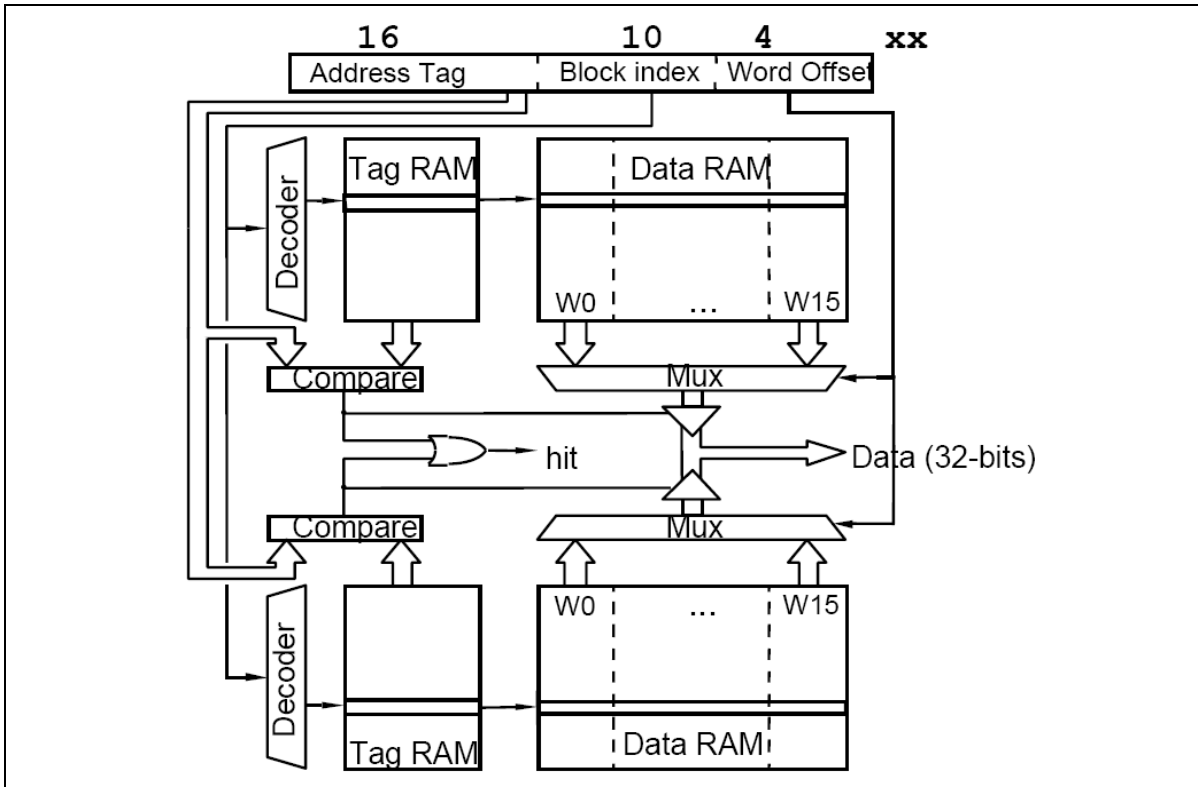


Figure 10: Organisation for 128KB 2-Way Set-Associative Cache with 16-word Block Size and Word Addressable

For byte addressable cache we need to select 1 byte in 16 words (= 64 bytes = 2^6 bytes). This increases byte offset to 6 bits. The Organisation for this byte-addressable cache is shown in Figure 11.

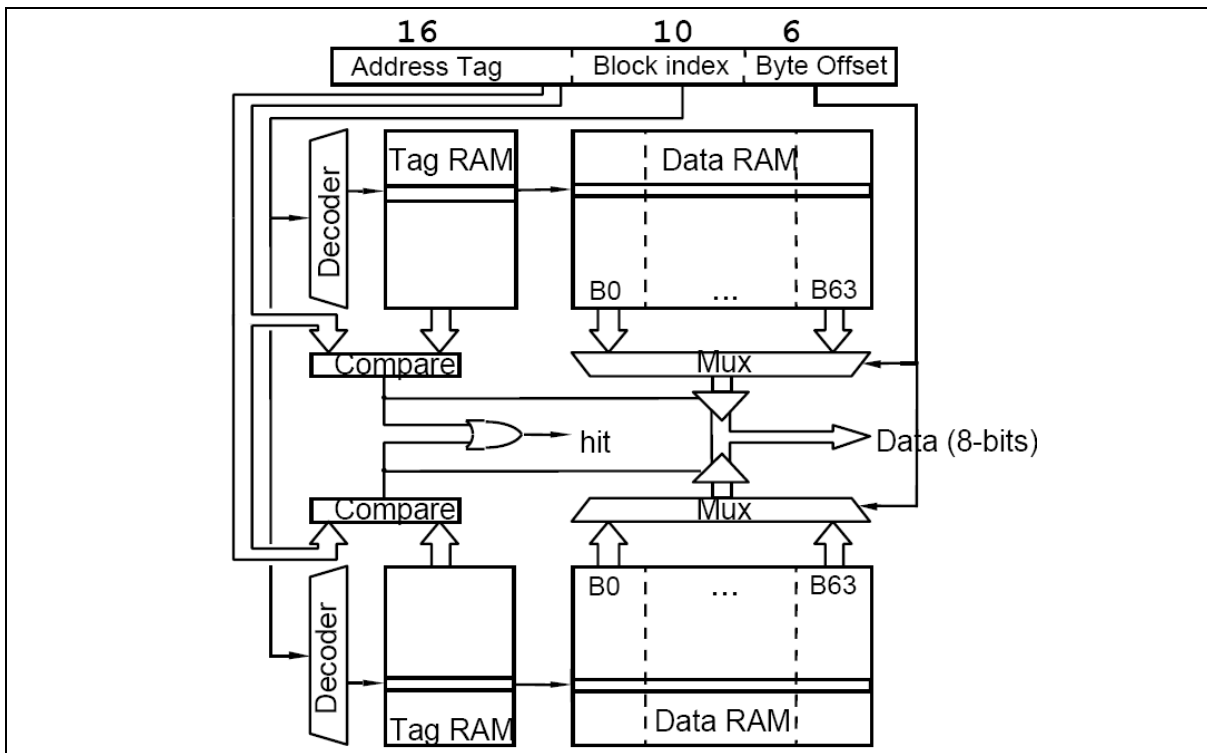


Figure 11: Organisation for 128KB 2-Way Set-Associative Cache with 16-word Block Size and Byte Addressable

Problem 4: Multi-level Cache

Assume a 2-level cache system with the characteristic below. Compute the average the memory access time.

L1 Hit Time = 1 cycle

L1 Miss Rate = 2.5%

L2 Hit Time = 6 cycles

L2 Miss Rate = 17% (% L1 misses that miss)

L2 Miss Penalty = 120 cycles

(Average Memory Access Time) = (L1 Hit Time) × (L1 Hit Rate) + (L1 Miss Rate) × (L1 Miss Penalty)

(L1 Miss Penalty) = (L2 Hit Time) × (L2 Hit Rate) + (L2 Miss Rate) × (L2 Miss Penalty)

(L1 Miss Penalty) = $6 \times (1 - 17\%) + 17\% \times 120 = 4.98 + 20.4 = 25.38$ cycles

(Average Memory Access Time) = $1 \times (1 - 2.5\%) + (2.5\%) \times (25.38) = 0.975 + 0.635 = 1.61$