The University of Western Australia

Department of Electrical and Electronic Engineering

Computer Architecture 219 (Tutorial 8)

1. [Stallings 2000] Suppose an 8-bit data word is stored in memory is 11000010. Using the Hamming algorithm, determine what check bits would be stored in memory with the data word. Show how you got your answer.

12	11	10	9	8	7	6	5	4	3	2	1
1	1	0	0	0	0	0	1	0	0	1	0

The check bits are in bit numbers 8, 4, 2, and 1.

- Check bit 8 calculated by values in bit numbers: 12, 11, 10, and 9.
- Check bit 4 calculated by values in bit numbers: 12, 7, 6, and 5.
- Check bit 2 calculated by values in bit numbers: 11, 10, 7, 6, and 3.
- Check bit 1 calculated by values in bit numbers: 11, 9, 7, 5, and 3.

2. [Stallings 2000] A set associative cache consists of 64 lines divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of memory addresses.

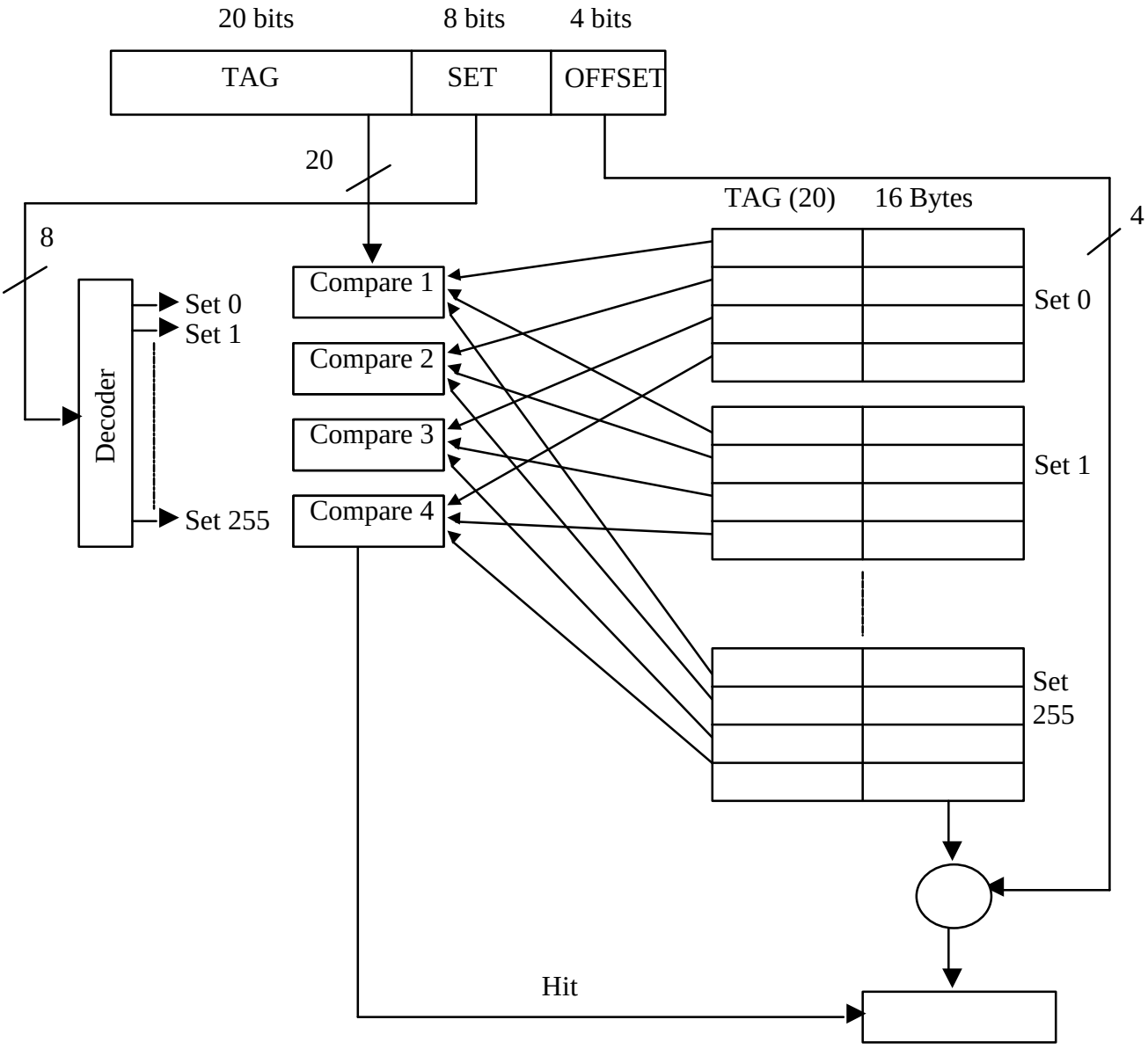
- The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number.
- Main memory consists of 4K = 2^{12} blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits.
- Each block contains 128 words. Therefore, 7 bits are needed to specify the word.

So, main memory address:

TAG	SET	WORD
8	4	7

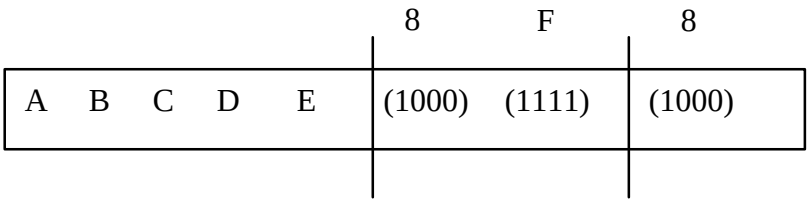
3. [Stallings 2000] Consider a 32-bit microprocessor that has an on-chip 16-kbyte four-way set associative cache. Assume that the cache has a line size of four 32-bit words. Draw a block diagram of this cache, showing its organization and how the different address fields are used to determine a cache hit/miss. Where in the cache is the word from memory location ABCDE8F8 mapped?

Block frame size = 16 bytes
Number of block frames in cache = 16Kbytes/16 Bytes = 1024.
Number of sets = Number of block frames/Associativity = 1024/4 = 256 sets.



For example, location ABCDE8F8 is mapped onto:

- Set 143, any line, byte 8.



4. [Stallings 2000] Describe a simple technique for implementing an LRU replacement algorithm in a four-way set associative cache.

- Associate a 2-bit counter with each of the four blocks in a set.
- Initially, arbitrarily set the four values to 0, 1, 2, and 3 respectively.
- When a hit occurs, the counter of the block that is referenced is set to 0. The other counters in the set with values originally lower than the referenced counter are incremented by 1; the remaining counters are unchanged.
- When a miss occurs, the block in the set whose counter value is 3 is replaced and its counter set to 0. All other counters in the set are incremented by 1.

5. [Stallings 2000] Consider the following code:

```
for ( i = 0; i  <  20; i++ )
   for ( j = 0; j  <  10; j++ )
   a[i] = a[i] * j
```

a. give one example of the spatial locality in the code.
b. Give one example of the temporal locality in the code.

a. A reference to the first instruction is immediately followed by a reference to the second.

b. The ten accesses to a[i] within the inner “for loop” which occur within a short interval of time.

6. [Stallings 2000] A computer has a cache, main memory, and a disk for virtual memory. If a referenced word is in the cache, 20ns are required to access it. If it is in main memory but not in the cache, 60ns are needed to load it into the cache, and then the reference is started again. If the word is not in main memory, 12ms are required to fetch the word from disk, followed by 60ns to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main memory hit ratio is 0.6. What is the average time in nanoseconds required to access a referenced word on this system?

Location of referenced word	Probability	Total time for access in ns
In cache	0.9	20ns
Not in cache, but in main memory	(0.1)(0.6) = 0.06	60 + 20 = 80ns
Not in cache or main memory	(0.1)(0.4) = 0.04	12ms + 60ns + 20ns = 12000080ns

So, the average access time would be:

(0.9)(20) + (.06)(80) + (0.04)(12000080) = 480026ns.

7. [Baron and Higbie 1992] Before caches were in use, many compiler writers had found that compiling for minimum size tended to produce faster-running code and conversely. Do caches increase or decrease the likelihood of this correlation?

- Compiling for minimum memory usage tends to make the code more compact, which should increase the cache hit rate and thus produce faster running code on machines with caches.

8. [Baron and Higbie 1992] If the speed of the main memory were to exceed the speed of the CPU, then cache memory would be useless. True or False? Justify your answer.

- False. Cache is or can be used to serve three purposes: reduced access time, increased bandwidth and decreased memory contention.

- Now, the faster memory would have no use for cache if it is as fast as the CPU, can provide sufficient bandwidth to keep the CPU busy all the time (both instructions and data), and allows simultaneous or near simultaneous access to multiple locations.

- Actually, many banks of the very fast memory might serve all the needs.