Chapter 09: Caches

Lesson 02:

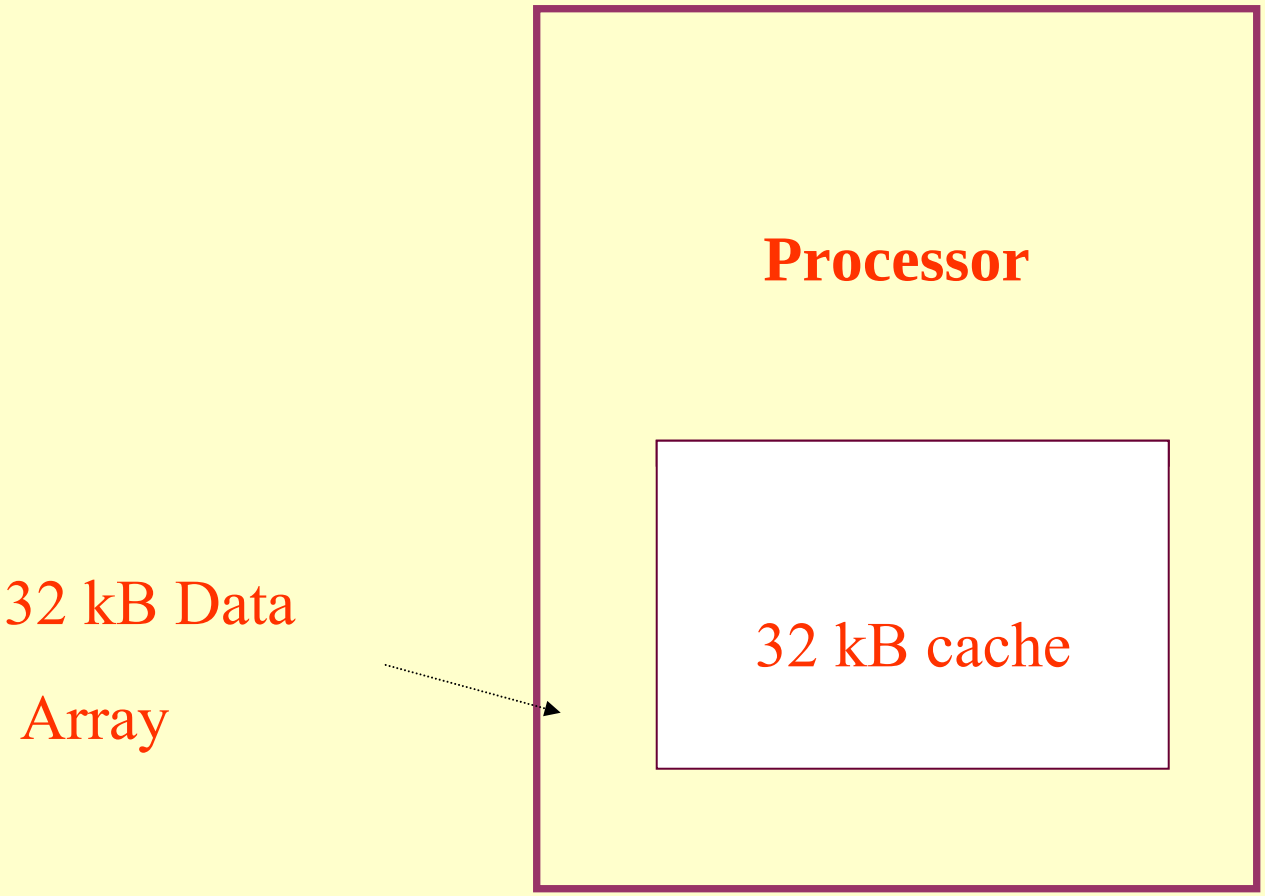
Cache Access and basic Features

Objective

- Learn features of the caches– Capacity, Line-length, Associativity (number of places a given address can reside), Replacement policy, and Whether the cache is write back or write through

Cache capacity

Capacity



Capacity

- The *capacity* of a cache is simply the amount of data that can be stored in the cache

Cache Line Length

Line Length

- The *line length* of a cache is the cache's block size— the size of the chunks of data that are brought into and thrown out of the cache in response to a cache miss

Example

- When a cache with 32-byte long cache lines has a cache miss, it brings a 32-byte block of data containing the address of the miss into the cache, evicting a 32-byte block of data beforehand if necessary to make room for the new data

Cache Line Alignment

- The address of the first byte in a cache line is always a multiple of the line-length
- Simplifies the process of determining whether or not a cache hit has occurred

Cache Line Alignment

- The low bits of the address determine which byte an address refers to within the line that contains it, and only the higher bits in the address need to be sent to the tag array to determine if a hit has occurred

Number of bits required for detecting cache hit

Exact number of bits required for detecting cache hit

- The size of the cache, the length of the cache lines, and the associativity of the cache determine the exact number of bits that need to be compared in order to detect cache hits

Example

- In a cache with 64-byte cache lines, find how many bits are used to determine which byte within a cache line an address points to

Solution

- The $\log_2$ of 64 is 6, so the low 6 bits of the address determine an address's byte within a cache line

Example

- In a cache with 64-byte cache lines, find the address of the first word in the cache line containing the address 0xbee3de72
- Cache lines are aligned on a multiple of their size, so the address of the first word in a line can be found by setting all of the bits that determine the byte within the line to 0
- $\log_2$ of 64 = 6 bits are used to select a byte within the line

... Solution

- We find the starting address of the line by setting the low 6 bits of the address to 0 in 0xbee3de72
- 0xbee3de40 = the address of the first word in the line

Factors taken into account Line Length Cache designer

Factors taken into account Line Length Cache designer

1. In general, increasing the length of a cache's lines increases the hit rate, due to the property of locality

- Increasing the line length increases the amount of data that is brought into the cache when a cache miss occurs

Cache Designer factor for Line Length

- Since addresses close to the address of a miss are likely to be referenced soon after a miss, using long cache lines means that each miss brings more data that is likely to be referenced soon into the cache, preventing misses if that data is referenced

Cache Designer factor for Line Length

2. However, increasing the length of a system's cache lines too much often increases the amount of time programs take to execute, even if it results in lower miss rates than using shorter lines

Cache Designer factors for Line Length

- Because of the amount of time it takes to bring longer cache lines into the cache, and because the probability that a given byte of data will be needed in the near future goes down as the address of the byte gets further from the address of the cache miss

Cache Designer factors for Line Length

- As a cache's line length increases, the increase in the line fetch time becomes a more significant factor than the decrease in the miss rate

Modern Cache Designs

- Lines in the 32-byte to 128-byte range, which is a good trade-off between hit rate and line fetch time

Example

- If a cache has 64-byte cache lines, find how long does it take to fetch a cache line if the main memory takes 20 cycles to respond to each memory request and returns 2 bytes of data in response to each request

Solution

- Since the main memory returns 2 bytes of data in response to each request, 32 memory requests are required to fetch the 64-Byte line
- At 20 cycles per request, fetching a cache line will take 640 cycles

Example

- Find how does the line fetch time of the above system change if page-mode DRAMs that have a CAS-data delay of 10 cycles are used to implement the main memory
- Assume─ cache lines always lie within a single row of the DRAM, but that each line lies in a different row than the last line fetched)

Solution

- The system still requires 32 memory requests to fetch each cache line
- Using page mode, only the first request takes the full 20 cycles
- Other 31 take only 10 cycles each
- Therefore, the time to fetch a cache line is $20 + (31 \times 10) = 330$ cycles

Analysis of the result

- Page-mode DRAMs significantly reduce the time to fetch longer cache lines
- Designer can increase the line length that gives the best performance

Associativity

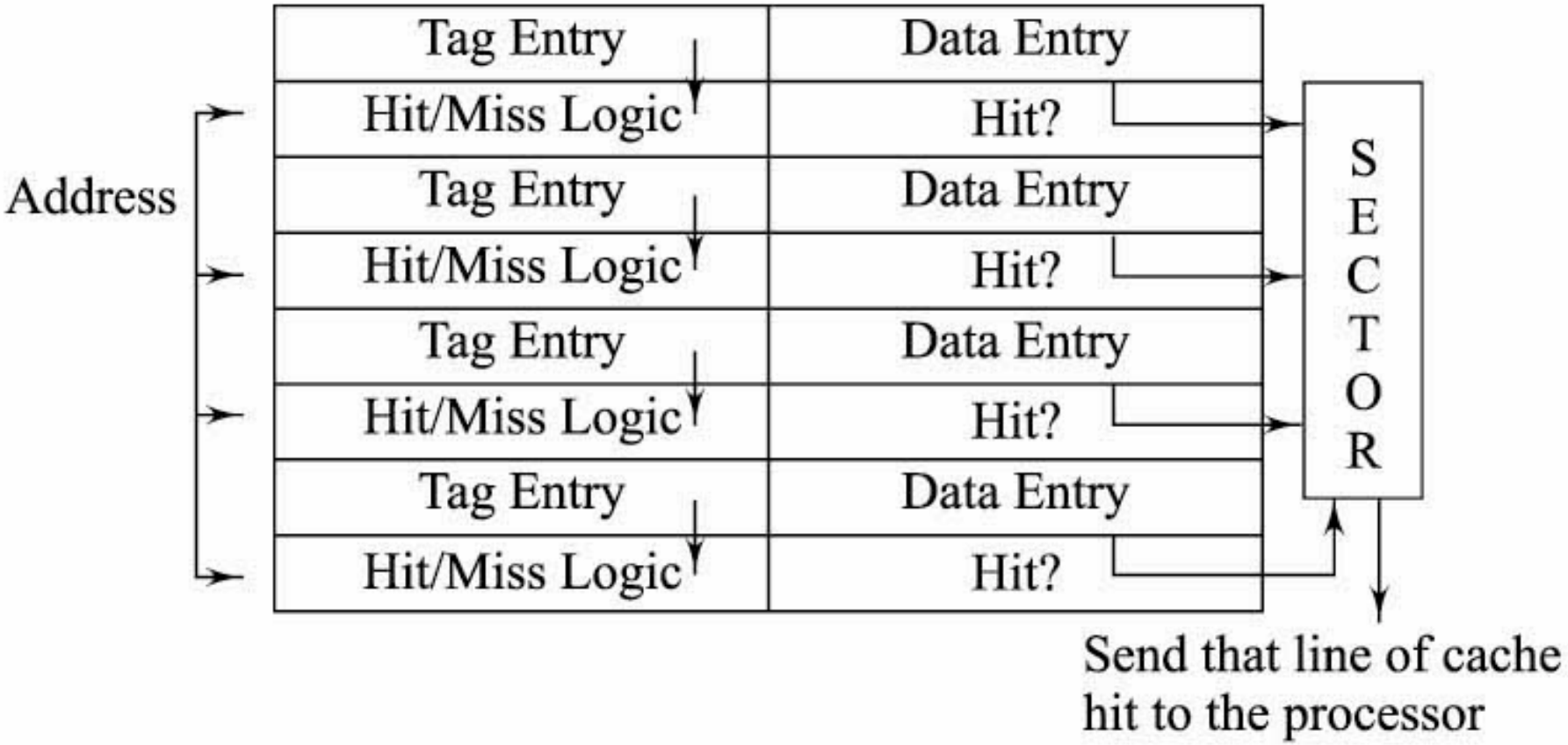
Associativity

- The *associativity of* a cache determines how many locations within the cache may contain a given memory address
- Caches with high associativity allow each address to be stored in many locations in the cache, which reduces cache misses caused by conflicts between lines that must be stored in the same set of locations

Low Associativity

- Caches with low associativity restrict the number of locations an address can be placed in, which increases the number of cache misses but simplifies the cache's hardware, reducing the amount of space taken up by the cache and often reducing the access time

Associativity



Other features

Other features

- Replacement Policy
- Write through or write back policy

Summary

We learnt

- Features of the caches— Capacity, Line-length, Associativity (number of places a given address can reside), Replacement policy, and Whether the cache is write back or write through

End of Lesson 02on

Cache Basic Features