CACHES: Organization

- Interpreting the address
- Mappings:
  * Fully associative
  * Direct mapped
  * Set associative
- Structure
- Hit/miss mechanism
- Advantages/disadvantages
- Examples
All three organizations are presented with the following specs:

- organizations to be covered:
  - fully associative cache
  - direct mapped cache
  - set associative cache

- main memory size = $2^{X+W+B}$ bytes
- main memory divided into blocks of size $2^{W+B}$ bytes
- main memory address requires at least X+W+B bits
- the X field of the address (the “block number”) is treated differently by the cache controller depending on the cache organization (more of this later)
- in all three organizations we will use a cache with 8 entries
Address to cache and main memory

(Fig. 5.8a)

- Address bus
  - X bits
  - W bits
  - B bits
  - Byte within “word”
  - “Word” within line
  - Block Number

Tag RAM
- Status bits
- Tag 0
- Block 0
- 
- 
- Status bits
- Tag C-1
- Block C-1

Data RAM

Tag RAM and Data RAM
- Status bits
- Tag 0
- Block 0
- 
- 
- Status bits
- Tag 1
- Block 1
- 
- 
- Status bits
- Tag C-1
- Block C-1

Main memory: $2^{X+W+B}$ bytes
- Block 0
- Block 1
- Block 2
- Block M-1
- Block size: $2^{W+B}$ bytes

Cache
- Block frame size: $2^{W+B}$ bytes
Block diagram of a cache

(Fig. 5.8b)

Address bus

Cache

Address

X  W,B

Upper bits  Lower bits

Tag RAM

Tag RAM (SRAM)

Data cache RAM (SRAM)

Cache controller

Main memory (DRAM)

Data Bus

Selecting within a cache block

Selecting within a cache block

Comparing for a hit/miss decision
Tag interpretation

\[ C = K \times S \]

where:  
- \( C \) = no. of entries (or block frames) in the cache
- \( K \) = associativity (no. of lines per set)
- \( S \) = no. of sets

(Fig. 5.9)

* For those systems that implement paging, the tag is the “page frame number”

\[ C = K \times S \]

where:
- \( C \) = no. of entries (or block frames) in the cache
- \( K \) = associativity (no. of lines per set)
- \( S \) = no. of sets

Fully associative (only one set)
- Associativity (number of lines/set) \( K = C \)
- Number of sets \( S = 1 \)

Direct-mapped (many sets, of one line each)
- Associativity (number of lines/set) is \( K = 1 \)
- Number of sets \( S = C; \ i = \log(s) \)

K-way Set-associative (many sets, each with \( K \) lines)
- Associativity (number of lines/set) is \( K \)
- Number of sets \( S = C/K; \ s = \log(S) \)
Fully associative cache mapping: any block from main memory can be put anywhere in the cache; number of sets $S = 1$, therefore, no "set field" is required; number of lines/set is $K = C = 8$
### Fully associative: example mapping

**TAG** (22 bits) | **BLOCK FRAMES** 32 bits (4 bytes) | **MAIN MEMORY**
---|---|---
FFFFFC | 12345678 | ABCDEF12
000000 | ABCDEF12 | A1B2C3D4
12339C | 123ABCAA | 123ABCAA
000004 | A1B2C3D4 | 123ABCAA

**TAG** (22) | **W+B** (2)
---|---
000000 00 | 00004 00
12339C 00 | 123ABCAA
FFFFFC 00 | 12345678
Because any block can reside anywhere in the cache, an *associative* (content addressable) memory is used. All locations are searched simultaneously.
**Pseudo LRU (least recently user) algorithm**

**Cache line replacement**

- All 4 lines in the set valid?
  - Yes
    - L0 or L1 least recently used
      - B1 = 0? if L0,B1 = 1
        - Yes Replace line L0
        - No Replace line L1
    - Yes B0 = 0?
      - Yes Replace line L1
      - No Replace line L2
  - No Replace invalid line

- L2 or L3 least recently used
  - B2 = 0? if L2,B2 = 1
    - Yes Replace line L2
    - No Replace line L3
Another drawing of the fully associative cache

Address:

(4)        (2)        (1)
Tag  W  B

Cache hit

Tag hit

Cache hit

“Word” select

“Word” select logic

Data out to CPU

Valid

Tag0 V0 Block Fr. 0

Comp0

Tag hit

Cache hit

“Word” select

Tag1 V1 Block Fr. 1

Comp1

Tag hit

Cache hit

“Word” select

Tag7 V7 Block Fr. 7

Comp7

Tag hit

Cache hit

“Word” select

Tag hit

Cache hit

“Word” select

Tag hit

Cache hit

“Word” select

Tag hit

Cache hit

“Word” select

Tag hit

Cache hit

“Word” select

Tag hit

Cache hit

“Word” select
Associative cache: advantages/disadvantages

Advantages:

- Fastest: all tags compared in parallel
- Unordered entries
- Most flexible of all—any memory block can go anywhere in the cache

Disadvantages:

- Most expensive: Need to search entire tag memory simultaneously means lots of hardware
- Replacement Policy is an issue when the cache is full (variety of replacement algorithms used)
- Large tag memory
- Small cache sizes

**Direct-mapped** caches simplify the hardware by allowing each MM block to go into only one place in the cache
Direct mapped cache

The diagram illustrates a direct mapped cache with 8 blocks, each containing 4 words. The address is divided into a tag (T), an index (I), and a block (B) field. The tag RAM is 1 bit, and the data RAM is 8 bytes. Each block frame has a decoder that maps the index to a specific line within the block frame. The cache hit is determined by comparing the tag in the cache with the tag from the address. If a match is found, data is fetched from the corresponding line. The memory size is calculated as $M = 2^X + W$ words, where $X = 4$, $W = 2$, and $B = 1$.

Memory size: $M = 2^4 + 2 = 16 + 2 = 18$ words = 144 bytes

Updating data (from memory)
Direct mapping

• Direct mapping: a block from main memory is mapped to a **specific block frame** in the cache;
• number of sets is \( S = C \);
• number of lines/set is \( K = 1 \);
• index field \( i = \log \text{ of number of entries in the cache} \);
• mapping: cache entry \( e = (\text{block no}) \mod (\text{total no. of entries in cache}) \)
1. Decode the group number of the incoming MM address to select the group
2. If Match AND Valid
3. Then gate out the tag field
4. Compare cache tag with incoming tag
5. If a hit, then gate out the cache line

6. and use the word field to select the desired word.

Cache memory

Main memory address
Tag Group Byte
5 8 3

Cache miss ≠ Cache hit

5-bit comparator

Selector

8–256 decoder

Hit

Bit map

5 5 1 1 1

Tag memory
Valid bits
Tag field, 5 bits

0 2

1 4

256

64

8
Direct-mapped caches: conclusions

- The direct mapped cache uses less hardware, but is much more restrictive in block placement.
- Cache block is available before hit/miss. (Possible to assume a hit and continue; recover later if miss)
- If two blocks from the same group are frequently referenced, then the cache will “thrash.” That is, repeatedly bring the two competing blocks into and out of the cache. This will cause a performance degradation.
- Block replacement strategy is trivial.
- Compromise—allow several cache blocks in each group—the Block-Set-Associative Cache
2-way set associative cache

Memory size: 
\[ M = 2^x + W \] words = 64 words = 128 bytes

From/To CPU

Updating data (from memory)
K-way set associative

• K-way set associative: a block from main memory is mapped to a specific set in the cache (in any of the set’s entries);
• number of lines per set is $K = 2$;
• number of sets is $S = C/K = 8/2 = 4$;
• set field $s = \log$ of number of sets in the cache;
• mapping: cache entry $e = (\text{block no}) \mod (\text{total no. of sets in cache})$
• Data comes after hit/miss decision and set selection
• 2-way and 4-way set associative cache more common
Different drawings for set associatives

GWU: Prof. N. Alexandridis
Different drawings for set associatives

GWU: Prof. N. Alexandridis
Different drawings for set associative caches

(Fig. 5.13)
Different drawings for set associatives

(Fig. 5.13c)
Different drawings for set associatives

(Fig. 5.13b)
Motorola 68040
4-way set-associative cache

\[\text{Local address (LA)}\]
\[31 \quad 12 \quad 11 \quad 9 \quad 0\]

\[\text{S} \quad \text{Page frame} \quad \text{Page offset}\]

\[\text{Supervisor bit} \quad 20\]
\[\text{ATC}\]

\[\text{Physical address} \quad 31\]
\[1110 \quad 9 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0\]

\[\text{Tag (22)} \quad \text{Set (6)} \quad W \quad B\]

\[\text{Set selection}\]

\[\text{Decoder}\]

\[\text{Byte displacement} \quad \text{Doubleword within line}\]

\[\text{Tag} \quad \text{Status}\]
\[1 \quad D0 \quad D1 \quad D2 \quad D3\]

\[\text{MUX}\]

\[\text{Data or instruction} \quad \text{Line select}\]

\[\text{Hit 3} \quad \text{Hit 2} \quad \text{Hit 1} \quad \text{Hit 0}\]

\[\text{Logical OR}\]

\[\text{Line 0} \quad \text{Line 1} \quad \text{Line 2} \quad \text{Line 3}\]

\[\text{Actual, only logical address bits LA31 - LA12 pass through the ATC. The remaining least significant address bits LA11 - LA0 are the same for both the logical and physical address. The structure of the ATC and its entries are given in Figure 6.15.}\]

\[\text{For instruction cache, the status field contains one valid bit for the entire line. For data cache, the status field contains one valid bit for the entire line, plus four additional dirty bits, one for each doubleword.}\]