Lecture 24 - Cache Memories

Idea 1:

Every storage device is a "cache" to the slower and larger storage device (at level \( K+1 \)).

Cache Hits

The data object \( d \), that the program needs from level \( K+1 \), is already available at level \( K \).

\[\begin{array}{c}
K & 2 & 4 \\
K+1 & 1 & 2 & 3 & 4 \\
\end{array}\]

Read block 2 \( \rightarrow \) cache hit.

Cache Misses

If the data object \( d \) is not cached at level \( K \),

\[\begin{array}{c}
K & 2 & 4 \\
K+1 & 1 & 2 & 3 & 4 \\
\end{array}\]

Read block 3 \( \rightarrow \) cache miss.

State after reading block 3 from \( K+1 \)

(Before) \( K: 1 \ 2 \ 4 \) \( \rightarrow \) \( K: 2 \ 3 \) (After).
Important terminology

1. replacing/evicting a block
2. victim block
3. replacement policy
4. placement policy

Kinds of Cache Misses

1. **Compulsory misses** (or) **cold misses.**
   - If the cache is empty $\Rightarrow$ cold cache.
   - Not empty $\Rightarrow$ warm cache.

Placement policy

1. **Any block** from level $k+1$ can be stored in **any block** at level $k$. (most flexible)
2. A particular block at level $k+1$ can be placed/stored only in a subset of blocks at level $k$. (more restrictive)

H/W caches ($L_1, L_2, L_3$) use this scheme.

$\oplus$ Is very expensive w.r.t. searching a block.
2. Conflict miss

Even though the cache is large enough to hold the referenced data objects, but because they map to the same cache block, the cache keeps missing.

*Example:* Reading block 0 & 8 repeatedly in the previous page.

3. Capacity misses

if \( \text{size of working set} > \text{cache size} \)

\[ \Rightarrow \text{capacity miss.} \]

The cache is just too small to handle this particular working set.

*Example:*

\[ \begin{array}{c}
0 \\
1
\end{array} \]

\[ \begin{array}{cccc}
0 & 1 & 2 & 3
\end{array} \]

if working set is 0, 1, 2 in the same order again and again.

\[ \Rightarrow \text{capacity miss.} \]
Cache Management

Logic to manage a cache - H/W, S/W or both.

eq. Registers - Compiler

L1, L2, L3 - Hardware Logic.
(built into the caches)

DRAM Main Memory - Operating System

Addr. translation H/W - on the CPU.

Blocks

On 32-bit computers,

Word size = 32 bits or 4 bytes.

Block = a group of words

eg. if block size = 8

⇒ 1 block = 8 words = 32 bytes.

if block size = 16

⇒ 1 block = 16 words = 64 bytes.
Cache Memories

Each memory address = $m$ bits

Unique addresses $M = 2^m$

Cache Sets ($S$)

$S = 2^s$

No. of cache sets $S = 2^s$

Where

Cache Lines ($E$)

Each set has 2 cache lines in the above example

$(E) \ E = 2$

Total cache lines in the above cache = 4
Cache block

Each cache line has 4 blocks. 1 block.

If 1 block = 8 words

⇒ each cache line has 8 words

\[ 4 \times 8 = 32 \text{ words} \]

If 1 word = 4 bytes

⇒ each cache line has

\[ 4 \times 8 = 32 \text{ bytes} \]

\[ 32 \times 4 = 128 \text{ bytes} \]

Size of the above cache

\[ = \frac{128}{512} \text{ bytes} \]

⇒ Size of memory per cache line.
Valid bit

\[
\begin{array}{c|c}
S_0 & 1 \\
 0 & \\
S_1 & 0 \\
 1 & \\
\end{array}
\]

If valid bit = 1 ⇒ The cache line contains meaningful information.
If valid bit = 0 ⇒ Either cold cache (or) cache entry invalidated.

Tag bit

\[ t = m - (b+s) \]

\( t \) - no of. tag bits.

Address: t bits s bits b bits

\[
\begin{array}{c|c|c}
\hline
\text{Tag} & \text{Set index} & \text{Block offset} \\
\hline
\end{array}
\]

To uniquely identify the block stored in the cache line.
4-bit address space

- **Tag (1-bit)**
- **Set index**
  - \( S = 2 \)
  - \( S = 2^2 = 4 \)
  - \( \Rightarrow 4 \text{ Sets} \)
- **Block offset**
  - \( b = 1 \)
  - \( B = 2^b = 2^1 = 2 \) bytes
  - \( \Rightarrow 1 \text{ block} = 2 \text{ bytes} \)

---

To find a word in a cache memory:

1. Find the set in which the word can be. Use the **set index** bits to find this.
2. Find the cache line (if any) within the set which contains the word. Use the **tag (t)** bits to identify this.
A line in the set contains the word
1. iff the valid bit = 1.
and
2. The tag bits in the line match the tag bits in the address A.

Direct Mapped Cache

No of cache lines per set = 1.
i.e. $E = 1$
\( (S, E, B, m)^{10} = (4, 1, 2, 4) \)

- Cache has
- 4 sets
- 1 line per set
- 2 bytes per block
- 4-bit addresses.

**4-bit Address:**

\[
\begin{array}{c}
\text{Tag} \\
\hline
b_3 \quad b_2 \quad b_1 \quad b_0
\end{array}
\]

**Assumption:**

1 word = 1 byte
Cache in action

1. Enumerate the 4-bit tag
2. 2 - 6
3. \( T = 1 \)
4. \( E = 1 \)
5. \( S = 2 \)
6. \( b = 0 \)
7. \( d = 0 \)
8. \( 11 \)
1. Read word at address 0.
2. Read word at address 1.
3. \[ u \ u \ u \ u \ 13 \]
4. \[ u \ u \ u \ u \ 8 \]
5. \[ u \ u \ u \ u \ 0 \]