Lecture - 27

Associative Caches

* Problem with direct-mapped caches

"Conflict misses"

\[ \text{for (i=0; i < 8; i++)} \]
\[ \text{sum += } x[i] * y[i]; \]

Cache block size = 8 bytes

Cache size = 16 bytes

<table>
<thead>
<tr>
<th>Elem.</th>
<th>Addr</th>
<th>Set index</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x[0])</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x[1])</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(x[2])</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>(x[3])</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>(y[0])</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>(y[1])</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>(y[2])</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>(y[3])</td>
<td>28</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ x[4] \]
\[ x[6] \]
1. Read $a[0]$.

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>$a[0]$</th>
<th>$a[1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8 bytes.

2. Read $b[0]$

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>$b[0]$</th>
<th>$b[1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Read $a[1]$

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>$a[0]$</th>
<th>$a[1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Read $b[1]$

<table>
<thead>
<tr>
<th>$S_0$</th>
<th>$b[0]$</th>
<th>$b[1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Rightarrow$ Trashing!
Set Associative Caches

Each set holds more than one cache line.

\[ 1 < E < \frac{C}{B} \]

E-way set associative cache.

eq 2-way set associative cache.

```
valid  Tag  cache block
S₀

S₁
```

Set S⁻¹
1. Set selection - same as direct-mapped cache.
2. Line matching and word selection.
3. Line replacement on Misses.

Replacement policies
1. Random
2. Least Frequently Used (LFU)
3. Least Recently Used (LRU)

* Fully Associative Caches

Only 1 Set!

\[ E = \frac{C}{B} \]

So

\[ E = \frac{c}{b} \]
lines in the one and only set.
1. Set Selection

trivial: only one set

no set index bits in the address

t bits

m-1

Tag

b bits

Black offset 0

2. Line matching and word selection.

IIIrd to set associative cache.

Fully associative caches \rightarrow\text{app}ropriate for small
caches

\text{tag matching in a large}
cache takes a long time

# HW problem 6.30

HW: Why the middle bits are used for set-index?

Why not the higher order bits?
Writes

Reads - straightforward.

- cache miss? → read from the next lower level → store block in some cache line
- return w

Write-through:
- * more bus traffic
- * simple (no dirty bit)
- * writes block to level K+1 immediately

Write-back:
- * less bus traffic
- * dirty bit needed
- * writes block to level K+1 only when the block is replaced
No-write allocate

Writes

Write-allocate
loads block in cache and writes in cache

Write-back caches

Write-through cache

Real Cache Hierarchy

Core 3

Core 0

Regs
4 cycles
L1 d-cache
32 KB

4 cycles
L1-i-cache (Read Only)
32 KB

11 cycles
L2 unified cache
256 KB

30-40 cycles
L3 unified cache
8 MB
Performance impact of cache parameters

1. miss rate $= \frac{\# \text{ misses}}{\# \text{ references}}$

2. hit rate $= 1 - \text{miss rate}$

3. hit time $= \text{time to deliver a word in the cache to the CPU}$
   $= t_{\text{set selection}} + t_{\text{line identification}} + t_{\text{word identification}}$

4. Miss penalty $= \text{any additional time required because of a miss}$ (eg. choosing a victim for eviction)

   eg. penalty for L1 misses served from:
   1. L2 $\rightarrow$ 10 cycles
   2. L3 $\rightarrow$ 40 cycles
   3. L4 $\rightarrow$ 100 cycles
Impact of cache size
large cache $\Rightarrow$ high hit rate.
" $\Rightarrow$ high hit time
" Faster caches are usually smaller.

Impact of block size
larger blocks $\Rightarrow$ increase hit rate
(for programs with good spatial locality).
" $\Rightarrow$ decrease hit rate
(for programs with good temporal locality)
" For a given cache size,
larger blocks $\Rightarrow$ fewer cache lines.
larger blocks $\Rightarrow$ negative impact on miss penalty.
" larger blocks $\Rightarrow$ larger transfer times.
Impact of Associativity

\[ \text{larger } E \Rightarrow \text{increase hit rate} \]
\[ \Rightarrow \text{conflict misses are reduced.} \]

Choice of \( E \):

hit time vs miss penalty.

Intel i7

\[
\begin{align*}
\text{L1, L2} & \Rightarrow E = 8 \\
\text{L3} & \Rightarrow E = 16
\end{align*}
\]

Impact of Write Strategy

- Use of write-back caches increases.
- Transfer time increases.