\[ F: 3 + 0.5 \times 18 = 3 + 0.9 = 3.9 \]
\[ W: 2 + 1 \times 18 = 2 + 1.8 = 3.8 \]
\[ D: 1 + 2 \times 18 = 1 + 36 = 36 \]

<table>
<thead>
<tr>
<th>Hit Rate</th>
<th>4,000 cycles</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>40 cycles</td>
<td>Main Memory</td>
</tr>
<tr>
<td>96%</td>
<td>10 cycles</td>
<td>L2</td>
</tr>
<tr>
<td>90%</td>
<td></td>
<td>L1</td>
</tr>
</tbody>
</table>

**Hit Time**

- Fully Associative: 3 cycle, 95% hit rate
- 2-way Set Associative: 2 cycle, 90% hit rate
- Direct Mapped: 1 cycle, 80% hit rate

**Which is the best L1 cache for this system?**

Review Problem 44
All locations are possible. 

- Fully Associative
- LFU
- Set Associative
- Only one location
- Direct-mapped

If we need to load a new cache line, where does it go?

Replacement Methods
Replacement Strategies

In practice, Random typically only 12% worse than LRU

Use Temporal Locality

Approach #2: Least Recently Used (LRU)

Just arbitrarily pick from possible locations

Approach #1: Random

When needed, pick a location
Slightly higher miss rate because each cache is smaller.

Optimize to usage

Higher bandwidth

Typically split the caches into separate instruction, data caches

CPU

How many accesses/cycle do we need for our pipelined CPU?

\[
\text{Instruction } \times \text{Data accesses}
\]

How do the two compare in usage?

Split Caches
$s = 1 + 2.5 = 3.5$

$(0.1 + 0.5) \times 1 = 0.5 	imes 100 = 6$

Multi-Level Caches
What happens on a write?
Which block is replaced on a miss.
From is a block found.
When can a block be freed.

Low Design Decisions

Conflict

Cache

Cold Start

Three issues:

Types of misses:

Locality: Temporal, Spatial

Illusion of big, fast memory via locality - optimistic
“Page” memory to disk when information won’t fit in main memory

Virtual Memory: View disk as the lowest level in the memory hierarchy

Use Disk as memory?

Disk more cost effective than even DRAM

<table>
<thead>
<tr>
<th>Cost/Capacity</th>
<th>Access Time</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x</td>
<td>10,000,000 cycles</td>
<td>Disk</td>
</tr>
<tr>
<td>15x</td>
<td>100 cycles</td>
<td>Flash</td>
</tr>
<tr>
<td>200x</td>
<td>100 cycles</td>
<td>DRAM</td>
</tr>
<tr>
<td>10,000x</td>
<td>7 cycles</td>
<td>SRAM</td>
</tr>
</tbody>
</table>

Virtual Memory
Memory mapping/address translation – conversion process from virtual memory to physical addresses.

Page fault – "miss" on main memory. Handled as a processor exception.

Page – the block for main memory; moved as a group to/from disk.

Other important terminology:

Virtual address may correspond to address in memory, or to disk.

Virtual addresses are mapped to physical addresses.

Each address the processor generates is a Virtual Address.

Solution: Virtual address

How do they share the address space?

Thought experiment: What happens when you run two programs at once?
Threshing

Translation complex – cache the misses themselves
Huge page fault/cache miss penalty – fully associative, software managed
Large cache lines/page size – spatial locality
Main Memory is a cache

Virtual Memory