Where Can a Block Be Placed?
- At a fixed place in the cache – **direct mapping**
  - Index = BlockAddress mod CacheSize
- Anywhere in the cache – **fully associative**
- Anywhere within a restricted set of places – **set associative**
  - SetIndex = BlockAddress mod NumberOfSets
  - e.g. if there are 4 blocks in a set, we call the cache 4-way set associative

Replacement Strategy
- When a miss occurs we must replace some block from the cache (if the cache is full) with a new block from the memory
  - Random
  - Least-recently used
  - FIFO (round-robin)

Write Strategy
- Block reads are optimized so that they start in parallel with reading and checking tag
  - If this is not the block we want, read data is simply discarded
- Block writes cannot start in parallel with tag checking
- When we modify a block in cache we must decide when to update the memory
  - Write-through – always update the memory
  - Write-back – update the memory only when we delete the block from the cache. We add a dirty bit to the block, this bit is modified when block write occurs and checked on write-back
  - We can write into write buffer and release processor
  - On write miss we can update the cache (write allocate) but we don’t have to (no write allocate)

Example
Assume a fully associative write-back cache with many cache entries that starts empty and performs operations:
- Write Mem[100];
- Read Mem[200];
- Write Mem[200];
- Write Mem[100];
- Write Mem[100];

What are numbers of hits and misses when using no-write allocate vs. write allocate?
see page 402 for solution

Instructions and Data Caches
- We can keep the data and instructions in the same cache – **unified cache**
- Or we can keep them in separate caches – **split cache**
  - This improves hit rate as instructions and data are accessed in a different manner

Cache Performance
To realistically evaluate cache performance we need to look beyond miss rate:
\[ \text{Average} \_ \_ \text{Memory} \_ \_ \text{Access} \_ \_ \text{Time} = \text{Hit} \_ \text{Time} + \text{Miss} \_ \text{Rate} \times \text{Miss} \_ \text{Penalty} \]
Example
Which has the lower miss rate: a 16KB instruction cache with a 16KB data cache or a unified 32KB cache? Use the following data for number of misses per 1000 instructions:

<table>
<thead>
<tr>
<th>size</th>
<th>instruction cache</th>
<th>data cache</th>
<th>unified cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 KB</td>
<td>3.82</td>
<td>40.9</td>
<td>51</td>
</tr>
<tr>
<td>32 KB</td>
<td>1.36</td>
<td>38.4</td>
<td>43.3</td>
</tr>
</tbody>
</table>

Assume 36% of instructions are data transfer instructions, a hit takes 1 clock cycle and penalty is 100 clock cycles. A load or store takes 1 extra clock cycle on a unified cache if there is only one port. What is the average memory access time for both cases? Assume write-through caches with write buffer.

see page 407 for solution

Example
Assume cache miss penalty is 100 cycles and all instructions normally take 1 clock cycle. Average miss rate is 2%, there is an average of 1.5 memory references per instruction, and the average number of cache misses per 1000 instructions is 30. Calculate CPU-time both with average miss rate per memory reference and average miss rate per instruction. What if we had no cache?

see page 409 for solution

Example
What is the impact of two different cache organizations on the performance of a CPU? Assume that CPI for a perfect cache is 2, clock cycle time is 1ns, there are 1.5 memory references per instruction, size of both caches is 64KB and block size is 64B. One cache is direct mapped and the other is 2-way set associative. For a set associative cache assume that clock cycle is 1.25 ns. Cache miss penalty is 75ns for both caches. Calculate memory access time and then CPU-time. Assume that hit time is 1 clock cycle, the miss rate of direct mapped cache is 1.4% and the miss rate of set associative cache is 1%.

see page 410 for solution

Out-of-Order Execution
- When processor supports out-of-order execution, some part of the miss penalty can be overlapped
  - It is a bit tricky to define miss rate and miss penalty
  - We say that processor is stalled whenever it cannot commit maximum number of instructions
    - First instruction that is causing the stall is regarded as a reason for the stall
    - If this is a memory reference waiting because of the cache miss we will count this stall towards miss rate

Example
Assume that CPI for a perfect cache is 2, clock cycle time is 1.25ns, there are 1.5 memory references per instruction. Cache is direct mapped and supports out-of-order execution. Cache miss penalty is 75ns but 30% of that can be overlapped. The miss rate is 1.4%. How much is CPU-time?

see page 411 for solution

Sample Cache Access Diagram
64-bit address, 2-way set associative, cache size 128KB (only for the data part), block size 16B

![Sample Cache Access Diagram](image-url)
Improving Cache Performance

- There are several possible directions for improvement:
  - Reduce miss penalty
  - Reduce miss rate
  - Increase parallelism
  - Reduce hit time

1. Reducing Miss Penalty: Multilevel Caches

- There is a trade-off between the size of the cache and the access time
  - Small caches are faster but cannot hold much data, therefore having larger miss rate
  - Large caches can hold enough data but are slower, thus increasing hit time
  - We can add another, smaller cache (L1) between the current cache (L2) and the CPU
  - Since everything that is in L1 is also in L2, it only makes sense to add L2 if it is much bigger than L1

Multilevel Caches

\[
\text{Average Memory Access Time} = \text{Hit Time}_{L2} \cdot \text{Miss Rate}_{L2} + \text{Miss Penalty}_{L2}
\]

\[
\text{Miss Penalty}_{L1} = \text{Hit Time}_{L2} \cdot \text{Miss Rate}_{L2} \cdot \text{Miss Penalty}_{L2}
\]

- We now have local and global miss rate
  - Local miss rate is number of misses in the given cache level divided by number of accesses to this level and it is equal to \( \text{Miss Rate}_{L1} \) for L1 and \( \text{Miss Rate}_{L2} \) for L2
  - Global miss rate is the number of misses in the given cache level divided by total number of accesses generated by the CPU. It is equal to \( \text{Miss Rate}_{L1} \) for L1, but it is \( \text{Miss Rate}_{L1} \cdot \text{Miss Rate}_{L2} \) for L2

\[
\text{Average Memory Stalls per Instruction} = \text{Misses per Instruction}_{L2} \cdot \text{Hit Time}_{L2} + \text{Misses per Instruction}_{L2} \cdot \text{Miss Penalty}_{L2}
\]

Example

- Assume that in 1000 memory references there are 40 misses in L1 cache and 20 misses in L2 cache. Assume the miss penalty for L2 is 100 clock cycles, hit time of L2 is 10 clock cycles, hit time of L1 is 1 clock cycle and there are 1.5 memory references per instruction. What are various miss rates, the average memory access time and the average memory stalls per instruction?

Example

- Given the data below, what is the impact of second-level cache associativity on its miss penalty:
  - Hit time for direct mapped L2 is 10 clock cycles
  - Two-way associativity increases hit time by 0.1 clock cycles
  - Local miss rate in L2 for direct mapped is 25%
  - Local miss rate in L2 for two-way set associative is 20%
  - Miss penalty for L2 is 100 clock cycles

2. Reducing Miss Penalty: Critical Word First and Early Restart

- CPU is usually stalled waiting for one word, not the whole block of data
  - Critical word first: Request from the memory and send this one word first to CPU thus reducing miss penalty, then continue loading the block
  - Early restart: Request block from the beginning from the memory, but when the desired word arrives send it immediately to CPU
  - These techniques do not help considerably as there is high chance that subsequent CPU accesses will be to other words in the block
Example
Assume that a computer has 64B block and L2 cache takes 11 clock cycles to get critical 8 bytes and then 2 clock cycles per 8 bytes to fetch the rest of the block. Calculate the average miss penalty with and without critical word first technique. What if the following instructions wait for the block to load then read the words following the first one, 8 bytes each.

3. Reducing Miss Penalty:
Giving Priority to Read Miss over Write
- If we have a write-through cache we can avoid stalls on write by writing data into a write buffer and then releasing CPU
- What happens if we do this but the subsequent read miss needs the data from the write buffer than has not been written yet?
  SW R3, 512(R0)  ; cache index 0
  LW R1, 1024(R0) ; cache index 0
  LW R2, 512(R0)  ; cache index 0
- The solution is to check the write buffer before going to memory
- If we have write-back cache, we can store the dirty block into write buffer, read data from memory into cache and release CPU, then perform the write from write buffer to memory