Reducing Miss Rate

- Miss categories:
  - Compulsory – first time when we want to access a block
  - Capacity – if cache cannot hold all blocks needed in a program
  - Conflict – if we use direct-mapped or set-associative strategy two blocks may map to the same record in cache

1. Reducing Miss Rate: Larger Block Size

- Reduce number of compulsory misses
- Larger blocks take advantage of spatial locality
- But increase miss penalty
- Larger block size means that fewer blocks will be in cache – this increases capacity misses and conflict misses

Example
Memory system takes 80 clock cycles of overhead and then delivers 16 bytes every 2 clock cycles. Miss rates for various block sizes are as follows.

<table>
<thead>
<tr>
<th>Block size</th>
<th>Cache size</th>
<th>4K</th>
<th>16K</th>
<th>64K</th>
<th>256K</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.52%</td>
<td>2.84%</td>
<td>2.64%</td>
<td>1.05%</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>7.34%</td>
<td>2.87%</td>
<td>1.33%</td>
<td>0.70%</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>7.90%</td>
<td>2.84%</td>
<td>1.00%</td>
<td>0.51%</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>7.78%</td>
<td>2.77%</td>
<td>1.02%</td>
<td>0.49%</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>0.51%</td>
<td>3.00%</td>
<td>1.15%</td>
<td>0.49%</td>
<td></td>
</tr>
</tbody>
</table>

Which block size gives us the smallest average memory access time?
for solution see page 429

2. Reducing Miss Rate: Larger Caches

- Reduce number of capacity misses
- But increase hit time and have higher cost

Example
Assume higher associativity would increase clock cycle time over direct mapped cache as follows: 2-way 1.36 times, 4-way 1.44 times, 8-way 1.52 times. Hit time is 1 clock cycle. Miss penalty for direct mapped cache is 25 clock cycles. Using the miss rates from figure 5.14, find the average memory access times for different cache sizes.
for solution see page 430 (mistake in solution, 25 should also be multiplied by increased clock cycle)

3. Reducing Miss Rate: Higher Associativity

- Experiments show that:
  - 8-way set associative cache has almost the same miss rate as fully associative cache
  - Direct mapped cache of size N has about the same miss rate as 2-way set associative cache of size N/2 (2:1 cache rule of thumb)
  - Greater associativity can come at the cost of increased hit time

Example
4. Reducing Miss Rate: Way Prediction and Pseudoassociative Caches

- **Way prediction**: High associativity increases hit time
  - Keep predictors with each set to predict which block in the set will be needed on next access, then compare the tag of this block with the tag we are looking for
  - On correct prediction hit time is greatly reduced
  - Simplest prediction is to remember which block was requested last time and change the prediction if we are wrong

- **Pseudoassociativity**
  - For two-way set associative cache, always check the first block
  - If we miss, check the second block before going to memory
  - Upon miss, swap blocks

---

5. Reducing Miss Rate: Compiler Optimizations

- **Make accesses to same block rather than across blocks**

- **Loop interchange**
  
  ```
  for(i=0;i<5000;i++)
  for(j=0;j<100;j++)
  x[i][j]=2*x[i][j]
  ```

- **Blocking**
  
  ```
  for(i=0;i<N;i++)
  for(j=0;j<N;j++)
  {
  r=0;
  for(k=0;k<N;k++)
  r=r+y[i][k]*z[k][j];
  x[i][j]=r;
  }
  ```

---

1. Improving Cache via Parallelism: Nonblocking Caches

- **Nonblocking Caches**
  - Usually, during cache miss, cache waits for data to be read from the memory and stalls all further requests
  - Nonblocking caches continue to serve future requests that result in a hit while the data for the miss is being fetched – hit under one miss
  - Further optimization may allow for multiple outstanding misses (i.e. nonblocking cache would also serve future requests that result in a miss)
  - We could have multiple misses for the same block

---

2. Improving Cache via Parallelism: Hardware Prefetching

- **Prefetch items from the memory before they have been requested by the processor**

- **Data can be placed into the cache or in some buffer**
  - For instructions, processor typically fetches two blocks on a miss – the requested one and the next consecutive one
  - If prefetched data is used, this generates next prefetch request

---

3. Improving Cache via Parallelism: Compiler-Controlled Prefetching

- **Compiler can insert special instructions to request prefetching**
  - Register prefetch request will ask for the data to be loaded into register
  - Cache prefetch request will ask for the data to be loaded into cache
  - Prefetch can be faulting or non-faulting
  - Prefetch only makes sense if processor does not stall while waiting for prefetched data
  - Write hint informs processor of the write miss that writes a whole block, to avoid unnecessary read

---

Example

For the following cache which optimization gives better performance: 2-way set associativity or hit under one miss? Calculate this both for FP and for integer programs. Assume that FP miss rate is 11.4% with direct-mapped cache and 10.4% with 2-way set associative cache. Miss rate for integer programs is 7.4% with direct mapped cache and 6% with 2-way set associative cache. Miss penalty is 16 cycles. Assume that the average memory stall time is simply a product of miss rate and miss penalty. Hit under one miss reduces average memory stall time to 76% for FP programs and to 81% for integer programs.
## Improving Hit Time

### 1. Improving Hit Time: Small and Simple Caches
- Hit time is critical because it determines (limits) the cycle time of the processor.
- It is mostly taken for accessing the tag and comparing it to the block address.

- **For first level cache**
  - Small so that it can fit on the same chip as the processor.
  - Simple, direct-mapped, so that the block read can be overlapped with tag check.

- **For second level cache**
  - Keeping tag memory on chip and data off chip provides fast checks.

### 2. Improving Hit Time: Avoiding Address Translation
- CPU generates requests for virtual addresses.
- Some of those are in main memory but the address must be translated from the virtual address to the physical address.
- To make common case fast, we can store virtual addresses in cache – virtual cache.

- Virtual cache removes address translation overhead but it:
  - May violate page-level protection.
  - Has to be flushed on context switch.
  - Sometimes programs use different virtual addresses for the same physical address.
  - I/O uses physical addresses.

### 3. Improving Hit Time: Pipelined Cache Access
- Pipeline cache access so that it can last multiple clock cycles.
- Now clock cycle can be small.
- But we have greater penalty for mispredicted branches and more stalls for RAW dependencies.

### 4. Improving Hit Time: Trace Cache
- Trace cache collects temporal information about data accessed, then loads this data into cache block (instead of the physical block).
- Branch prediction is folded into cache.
- This solves the problem of low cache utilization – because of branches only a small portion of traditional cache block is used.

## Main Memory Organizations for Improving Performance
- Main memory communicates with cache and I/O devices.
- Performance measures:
  - Latency – how long it takes for the request to be answered.
  - Bandwidth – how much data can be read/written at once.
- Rather than just making memory faster (which is hard) or increasing bandwidth, some organizations better address these issues.
Base Case
- 4 clock cycles to send the address
- 56 clock cycles for access time per word
- 4 clock cycles to send a word of data
- Assume cache block is 4-word long, word is 8 B
  - Miss penalty = 256 cycles

1. Higher Bandwidth: Wider Main Memory
- Wider the memory, less accesses we need on a miss
- In the previous example, if the memory were 2 words long we would need 128 cycles
- Cache width is the same as memory width
  - As CPU still accesses cache a word at a time, words have to go through a multiplexer so that the correct word gets selected
  - Second level cache can help

2. Higher Bandwidth: Simple Interleaved Memory
- Memory is organized into banks
- Interleaved memory organization makes use of that by allowing banks to be read at the same time, interleaving multiple reads
- Miss penalty would be 76 cycles
- Writes can also be interleaved
  - If interleaving is done so that adjacent words are stored in different banks, this optimizes sequential memory accesses
  - To see benefit we need more banks than cycles to access memory

Example
Block size is 1 word, as well as memory bus width.
Miss rate is 3%, memory accesses per instruction 1.2, cache miss penalty 64 cycles and CPI of ideal cache is 2.
If we change block size to 2 words, the miss rate falls to 2%, and a 4-word block has a miss rate of 1.2.
Calculate performance with 2 and 4 words for a block size, with and without interleaving
See solution on page 452

3. Higher Bandwidth: Independent Memory Banks
- Memory is organized into banks, but not sequentially
- Each device, such as I/O and caches, will access one bank

Exercise 5.1
We are given two machines, A and B with same processor (2GHz) and main memory, CPI of 1, and penalty of 100ns. Writing a word to a memory takes 100ns, and writing a block takes 200ns. Cache A is 2-way set associative and has 32B blocks, it is write through and does not allocate a block on write miss. Cache B is direct-mapped and has 32B blocks, it is write back and allocates a block on a write miss.
How to write a benchmark so that A is better than B?
How to write a benchmark so that B is better than A?