# Example

A computer system has a word-addressable main memory consisting of 256K 16-bit words. It also has a 4K-word cache organized in a set associative manner with 4 block frames per set and 64-words per block. The cache is 10 times faster than main memory. Assume the cache is initially empty. Suppose the CPU fetches 4352 words from locations 0,1,2,3...,4351 in order. It then repeats this fetch sequence 14 more times. Assume no interleaved memory and no read-through policy.

a) Specify the number of bits in the tag, set and offset fields in the interpretation of a main memory address.

b) Assuming the LRU algorithm is used for block replacement, estimate the speedup (ratio of time without cache to time with cache) resulting from use of the cache.

Cache Structure

| **Set 0** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** | **Set 7** | **Set 8** | **Set 9** | **Set10** | **Set11** | **Set12** | **Set13** | **Set14** | **Set15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 | fr 0 |
| fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 | fr 1 |
| fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 | fr 2 |
| fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 | fr 3 |

Cache is 10 times faster then main memory. Assuming
**Cache Access Time** = ***t*** , then
**Memory Access Time** = ***10*t***

As far as we have **Niter = 15** iterations of references from 0 to 4351 (with total number of references **Nref = 4352**) total number of blocks accessed on each iteration is

$$\mathbf{NBL} = \frac{\mathbf{Nref}}{\mathbf{Block\ Size}} = \frac{4352}{64} = \mathit{68}$$

but we have only *64* block frames in cache.

**Total Time for Fetches Without Cache = Nref * Memory Access Time * Niter**
**= 4352 * 10 * *t* * 15 = *652800*t***
**Total Time for Fetches With Cache = Time for Fetches From Cache (on hits) +**
**Time for Fetches From Memory(on misses);**
**Time for Fetches From Cache (on hits) = Nref * Cache Access Time * Niter;**
**Time for Fetches From Memory(on misses) = Nmisses * Miss Penalty;**

where **Nmisses** is total number of misses for all iterations,
**Miss Penalty** is time to transfer a block from memory to cache assuming 1 word at a time.

**Miss Penalty = Block Size * Memory Access Time =** ***64 * 10 * t***

**Nmisses = SUM (Nmissesi , i = 1..15)**

Now assuming LRU policy we will count number of misses for the first and subsequent iterations to plug in the formula.

## First Iteration

State of the Cache Before Iteration
(initially the cache is empty)

| **Set 0** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** | **Set 7** | **Set 8** | **Set 9** | **Set10** | **Set11** | **Set12** | **Set13** | **Set14** | **Set15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

State of the Cache After Iteration

| **Set 0** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** | **Set 7** | **Set 8** | **Set 9** | **Set10** | **Set11** | **Set12** | **Set13** | **Set14** | **Set15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/64 | 1/65 | 2/66 | 3/67 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

So, we have 64 initial misses to fill the cache and 4 replacements made according to LRU policy (least recently used is the block frame 0 in each set).
**Nmisses1**= 64 + 4 = *68*

## Second Iteration

State of the Cache Before Iteration

| **Set 0** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** | **Set 7** | **Set 8** | **Set 9** | **Set10** | **Set11** | **Set12** | **Set13** | **Set14** | **Set15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **64** | **65** | **66** | **67** | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

(remember that least recently used is now the block frame 1 in each set, shown in black are most recently used fields)

State of the Cache After Iteration

| **Set 0** | **Set 1** | **Set 2** | **Set 3** | **Set 4** | **Set 5** | **Set 6** | **Set 7** | **Set 8** | **Set 9** | **Set10** | **Set11** | **Set12** | **Set13** | **Set14** | **Set15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64/48 | 65/49 | 66/50 | 67/51 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16/0/64 | 17/1/65 | 18/2/66 | 19/3/67 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32/16 | 33/17 | 34/18 | 35/19 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48/32 | 49/33 | 50/34 | 51/35 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

So, we start loading missing blocks 0..3 replacing frame 1 in each set. As a result we miss on each replaced block:
**Nmisses2**= 4 + 4 + 4 + 4 + 4= *20*

## Third Iteration

State of the Cache Before Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | Set 4 | Set 5 | Set 6 | Set 7 | Set 8 | Set 9 | Set10 | Set11 | Set12 | Set13 | Set14 | Set15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **48** | **49** | **50** | **51** | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **64** | **65** | **66** | **67** | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 16 | 17 | 18 | 19 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 32 | 33 | 34 | 35 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

(remember that least recently used is now the block frame 2 in each set)

State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | Set 4 | Set 5 | Set 6 | Set 7 | Set 8 | Set 9 | Set10 | Set11 | Set12 | Set13 | Set14 | Set15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48/32 | 49/33 | 50/34 | 51/35 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 64/48 | 65/49 | 66/50 | 67/51 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 16/0/64 | 17/1/65 | 18/2/66 | 19/3/67 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 32/16 | 33/17 | 34/18 | 35/19 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

So, we start loading missing blocks 0..3 replacing frame 2 this time. As you might notice the resulting number of misses will be the same:

**Nmisses3**= 4 + 4 + 4 + 4 + 4= *20*

Further iterations 4 through 15 as the third iteration will also have the same pattern as the second one.
So, we can calculate the total number of misses on all iterations:

**Nmisses** = 68 + 20*14 = *348*

**Total Time for Fetches With Cache = 4352*15*t + 348*(64*10*t)**
**= 65280*t + 222720*t**
**= *288000*t***

$$\textbf{Speedup} = \frac{\textbf{Total Time for Fetches without Cache}}{\textbf{Total Time for Fetches with Cache}} = \frac{\textbf{652800*t}}{\textbf{288000*t}} = \textbf{\textit{2.26}}$$

c) Repeat part (b) assuming a most recently used (MRU) policy.
As far as the formula for this case is the same as above, we just have to calculate the total number of misses on all iterations.

## First Iteration(MRU)

State of the Cache Before Iteration State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | . . . |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

(initially the cache is empty)

| Set 0 | Set 1 | Set 2 | Set 3 | . . . |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |
| 16 | 17 | 18 | 19 | ... |
| 32 | 33 | 34 | 35 | ... |
| 48/64 | 49/65 | 50/66 | 51/67 | ... |

So, we have 64 initial misses to fill the cache and 4 replacements made according to MRU policy (most recently used is the block frame 3 in each set).

**Nmisses1**= 64 + 4 = *68*

## Second Iteration(MRU)

State of the Cache Before Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | . . . |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |
| 16 | 17 | 18 | 19 | ... |
| 32 | 33 | 34 | 35 | ... |
| 64 | 65 | 66 | 67 | ... |

Notice the change of pattern from the previous example! References to blocks from 0 to 47 are successive hits, but blocks 48 to 51 are missing. Most recently used block frame as the miss occurs is block frame 2, shown in black.

State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |
| 16 | 17 | 18 | 19 | ... |
| 32/48 | 33/49 | 34/50 | 35/51 | ... |
| 64 | 65 | 66 | 67 | ... |

So, we start loading missing blocks 48..51 replacing frame 2 in each set. As a result we have:

**Nmisses2**= *4*

## Third Iteration(MRU)

State of the Cache Before Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |
| 16 | 17 | 18 | 19 | ... |
| 48 | 49 | 50 | 51 | ... |
| 64 | 65 | 66 | 67 | ... |

Most recently used block frame as the miss occurs is block frame 1, shown in black.

State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |
| 16/32 | 17/33 | 18/34 | 19/35 | ... |
| 48 | 49 | 50 | 51 | ... |
| 64 | 65 | 66 | 67 | ... |

So, we start loading missing blocks 32..35 replacing frame 1 in each set. As a result we have:

**Nmisses3**= *4*

## Forth Iteration(MRU)

State of the Cache Before Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ... |

State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 0/16 | 1/17 | 2/18 | 3/19 | ... |

| | | | | |
|---|---|---|---|---|
| 32 | 33 | 34 | 35 | ... |
| 48 | 49 | 50 | 51 | ... |
| 64 | 65 | 66 | 67 | ... |

Most recently used block frame as the miss occurs is block frame 0, shown in black.

| | | | | |
|---|---|---|---|---|
| 32 | 33 | 34 | 35 | ... |
| 48 | 49 | 50 | 51 | ... |
| 64 | 65 | 66 | 67 | ... |

So, we start loading missing blocks 16..19 replacing frame 0 in each set. As a result we have:
**Nmisses4**= *4*

## Fifth Iteration(MRU)

State of the Cache Before Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 16 | 17 | 18 | 19 | ... |
| 32 | 33 | 34 | 35 | ... |
| 48 | 49 | 50 | 51 | ... |
| 64 | 65 | 66 | 67 | ... |

Most recently used block frame as the miss occurs (which occurs right away) is block frame 3, shown in black.

State of the Cache After Iteration

| Set 0 | Set 1 | Set 2 | Set 3 | ... |
|---|---|---|---|---|
| 16 | 17 | 18 | 19 | ... |
| 32 | 33 | 34 | 35 | ... |
| 48/64 | 49/65 | 50/66 | 51/67 | ... |
| 64/0 | 65/1 | 66/2 | 67/3 | ... |

Misses occur at the beginning and at the end of iteration doubling number of misses.
**Nmisses5**= *8*

Further iterations 6 through 15 will also have the same pattern.

Iteration 1 -> number of misses = 68
Iterations 2,3,4,6,7,8,10,11,12,14,15 -> number of misses = 4
Iterations 5,9,13 -> number of misses = 8

So, we can calculate the total number of misses on all iterations:

**Nmisses** = 68 + 11*4 + 3*8 = *136*

**Total Time for Fetches With Cache = 4352*15*t + 136*(64*10*t)**
**= 65280*t + 87040*t**
***= 152320*t***

$$\textbf{Speedup} = \frac{\textbf{Total Time for Fetches without Cache}}{\textbf{Total Time for Fetches with Cache}} = \frac{\textbf{652800*t}}{\textbf{152320*t}} = \mathit{4.28}$$