# UCSC Silicon Valley Extension

## C Programming, Advanced

Assignment 1 : C Review
Instructor : Radhika Grover

Include test cases in all of the following programs to demonstrate that the program executes correctly. The output carries 50% of the points for that problem. No points will be awarded if the program does not compile.

**Problem 4 "Hit or Miss"**
Caches are used to reduce the average time to access data from main memory in a computer. When the memory block requested is present in a cache, it is called a *hit*. When the memory block is not in the cache, it is called a *miss*, and the requested memory block is loaded into the cache. In this problem, you will write a program to determine the hits and misses for four different types of caches, and determine which one has the highest ratio of hits to misses for a given set of memory accesses:

1-way set associative, 2-way set associative, 4-way set associative, 8-way set associative

An *m*-way set associative cache contains *m* cache blocks per set. You are given four types of caches where each cache contains total of 8 cache blocks. The following tables describe the configuration for each type of cache:

**1-way set-associative cache**
8 sets with 1 block each
**Set 0 :** Block1
**Set 1 :** Block1
**Set 2 :** Block1
**Set 3 :** Block1
**Set 4 :** Block1
**Set 5 :** Block1
**Set 6 :** Block1
**Set 7 :** Block1

**2-way set-associative cache**
4 sets with 2 blocks each

**Set 0 :** Block1 Block2
**Set 1 :** Block1 Block2
**Set 2 :** Block1 Block2
**Set 3 :** Block1 Block2

**4-way set-associative cache**
2 sets with 4 blocks each
**Set 0 :** Block1 Block2 Block3 Block4
**Set 1 :** Block1 Block2 Block3 Block4

**8-way set-associative cache**
1 set with 8 blocks
**Set 0:** Block1 Block2 Block3 Block4 Block5 Block6 Block7 Block8

In the 1-way set associative cache, each memory block address is mapped to one cache block whose index is given by:
cache index = (memory block address) % (Number of blocks in the cache)

An *m*-way set associative cache provides more choices to store a memory block. In an *m*-way set associative cache, each memory block is mapped to one set in the cache but can be stored in any cache block that is available in that set. The set index is given by:

Set index = (memory block address) % (Number of sets in the cache)

For example, consider a program that accesses instructions at the following memory block addresses: 0, 0, 1, 1, 1, 1, 8, 9, 16. Next, we briefly discuss how each memory block address is mapped to a cache block for 1-way and 2-way set associative caches, and the number of hits and misses.

**1-way set associative cache**
The cache contains 8 cache blocks and is empty initially. A memory block address is mapped to a cache index as follows:

Cache index= (memory block address) % (Number of cache blocks)

First, the block with address 0 is referenced. This has a cache index of 0 % 8 = 0, and results in a cache **miss** because the cache does not contain this memory block at index 0. This memory block is brought into the cache and stored at index 0. Next, memory block address 0 is referenced again. This is a **hit** because memory block 0 is now present at cache index 0.

The following table describes the cache hits and misses for each memory block address:

| Operation Number | Memory Block Address | Cache Index | Hits/Miss | Reasoning |
|---|---|---|---|---|
| 1 | 0 | 0%8=0 | Miss | Cache block 0 is empty |
| 2 | 0 | 0%8=0 | Hit | As it has been stored at index 0 while performing operation number 1 |
| 3 | 1 | 1%8=1 | Miss | Cache block 1 is empty |
| 4 | 1 | 1%8=1 | Hit | As it has been stored at index 1 while performing operation number 3 |
| 5 | 1 | 1%8=1 | Hit | As it has been stored at index 1 while performing operation number 3 |
| 6 | 1 | 1%8=1 | Hit | As it has been stored at index 1 while performing operation number 3 |
| 7 | 8 | 8%8=0 | Miss | Replaces block 0 from operation number 1 |
| 8 | 9 | 9%8=1 | Miss | Replaces block 1 from operation number 3 |
| 9 | 0 | 0%8=0 | Miss | Replaces block 8 from operation number 7 |
| 10 | 1 | 1%8=1 | Miss | Replaces block 9 from operation number 8 |
| 11 | 8 | 8%8=0 | Miss | Replaces block 0 from operation number 9 |
| 12 | 9 | 9%8=1 | Miss | Replaces block 1 from operation number 10 |
| 13 | 16 | 16%8=0 | Miss | Replaces block 0 from operation number 11 |

Total hits: 4 Total misses: 9

**2-way set associative cache**

This type of cache contains 2 blocks in each set. A memory block can be placed into any one of the two cache blocks that is is empty in that set. If both cache blocks are filled, then one of them should be replaced with the new block. Assume that the **Least Recently Used** (**LRU**) cache replacement policy is used. In this scheme, the block that has not been referenced for the longest time is removed from the set.

The following table shows the hits and misses:

| Operation Number | Memory Block Address | Cache Set Index | Block Id | Hits/Miss | Reasoning |
|---|---|---|---|---|---|
| 1 | 0 | 0%4=0 | 0 | Miss | Block 0 in set 0 is empty |
| 2 | 0 | 0%4=0 | 0 | Hit | As it has been stored in block 0 of set 0 while performing operation number 1 |
| 3 | 1 | 1%4=1 | 0 | Miss | Block 0 of set 1 is empty |
| 4 | 1 | 1%4=1 | 0 | Hit | As it has been stored in block 0 of set 1 while performing operation number 3 |
| 5 | 1 | 1%4=1 | 0 | Hit | As it has been stored in block 0 of set 1 while performing operation number 3 |
| 6 | 1 | 1%4=1 | 0 | Hit | As it has been stored in block 0 of set 1 while performing operation number 3 |
| 7 | 8 | 8%4=0 | 1 | Miss | Block 1 of set 0 is empty |
| 8 | 9 | 9%4=1 | 1 | Miss | Block 1 of set 1 is empty |
| 9 | 0 | 0%4=0 | 0 | Hit | As it has been stored in block 0 of set 0 while performing operation number 1 |
| 10 | 1 | 1%4=1 | 0 | Hit | As it has been stored in block 0 of set 1 while performing operation number 3 |
| 11 | 8 | 8%4=0 | 1 | Hit | As it has been stored in block 1 of set 0 while performing operation number 7 |
| 12 | 9 | 9%4=1 | 1 | Hit | As it has been stored in block 1 of set 1 while performing operation number 8 |
| 13 | 16 | 16%4 =0 | 0 | Miss | Replace block 0 in set 0 as it was accessed before block 1 in set 0 (LRU) |

Total hits: 8 Total misses: 5

Your program should the input file that contains the memory block addresses and report the total hits and missed for each cache configuration and the type of cache with the largest number of hits:

Test case 1: 1-way set associative hits: # misses: #
2-way set associative hits: # misses: #
4-way set associative hits: # misses: #
8-way set associative hits: # misses: #

Best scheme: #-way set associative

Test case 2:
1-way set associative hits: # misses: #
2-way set associative hits: # misses: #
4-way set associative hits: # misses: #
8-way set associative hits: # misses: #

Best scheme: #-way set associative

and so on.

The test file has this format:
# number of test cases
# number of accesses
# memory block addresses