# CS3021/3421 Tutorial 6

Q1. Compute the number of hits and misses if the subsequent list of hexadecimal addresses is applied to caches with the following organisations.

(i) 128 byte 1-way cache with 16 bytes per line (direct mapped)
(ii) 128 byte 2-way set associative cache with 16 bytes per line
(iii) 128 byte 4-way set associative cache with 16 bytes per line
(iv) 128 byte 8-way associative cache with 16 bytes per line (fully associative)

0000 ⇒ 0004 ⇒ 000c ⇒ 2200 ⇒ 00d0 ⇒ 00e0 ⇒ 1130 ⇒ 0028 ⇒
113c ⇒ 2204 ⇒ 0010 ⇒ 0020 ⇒ 0004 ⇒ 0040 ⇒ 2208 ⇒ 0008 ⇒
00a0 ⇒ 0004 ⇒ 1104 ⇒ 0028 ⇒ 000c ⇒ 0084 ⇒ 000c ⇒ 3390 ⇒
00b0 ⇒ 1100 ⇒ 0028 ⇒ 0064 ⇒ 0070 ⇒ 00d0 ⇒ 0008 ⇒ 3394 ⇒

Assume that the first 4 bits of the address is used as the offset within the cache line, the next $\log_2(N)$ bits select the set and the remaining bits form the tag. Furthermore, assume that the all cache lines are initially invalid and that a LRU replacement policy is used.

Q2. Determine the number of compulsory, capacity and conflict misses for each cache organisation in Q1.

Q3. Write a program (in C, C++, Java, …) to solve Q1. Make sure you can create a generalised cache object with parameters L, K and N. Hand in a copy of your source code and evidence that your program works.

Q4. FOR AN EXTRA 5 MARKS modify your program to simulate a pseudo LRU replacement policy. What are the hit rates for the cache organisations specified in Q1?

THIS TUTORIAL IS WORTH DOUBLE MARKS (MARKED OUT OF 20 OR 25/20)

## 128 byte 1-way cache with 16 bytes per line (L = 16, N = 8, K=1)

| tag | 4 word (16bytes) | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Address Format**

| 15 | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| address | set | hit/miss |
|---|---|---|
| 0000 | | |
| 0004 | | |
| 000c | | |
| 2200 | | |
| 00d0 | | |
| 00e0 | | |
| 1130 | | |
| 0028 | | |
| 113c | | |
| 2204 | | |
| 0010 | | |
| 0020 | | |
| 0004 | | |
| 0040 | | |
| 2208 | | |
| 0008 | | |
| 00a0 | | |
| 0004 | | |
| 1104 | | |
| 0028 | | |
| 000c | | |
| 0084 | | |
| 000c | | |
| 3390 | | |
| 00b0 | | |
| 1100 | | |
| 0028 | | |
| 0064 | | |
| 0070 | | |
| 00d0 | | |
| 0008 | | |
| 3394 | | |

## 128 byte 2-way cache with 16 bytes per line (L = 16, N = 4, K=2)

| tag (K=0) | tag(K=1) | 4 word (16bytes) | | | | 4 word (16bytes) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

**Address Format**

| 15 | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| address | set | hit/miss |
|---|---|---|
| 0000 | | |
| 0004 | | |
| 000c | | |
| 2200 | | |
| 00d0 | | |
| 00e0 | | |
| 1130 | | |
| 0028 | | |
| 113c | | |
| 2204 | | |
| 0010 | | |
| 0020 | | |
| 0004 | | |
| 0040 | | |
| 2208 | | |
| 0008 | | |
| 00a0 | | |
| 0004 | | |
| 1104 | | |
| 0028 | | |
| 000c | | |
| 0084 | | |
| 000c | | |
| 3390 | | |
| 00b0 | | |
| 1100 | | |
| 0028 | | |
| 0064 | | |
| 0070 | | |
| 00d0 | | |
| 0008 | | |
| 3394 | | |