Week 13: Memory Hierarchies (Solutions)

1 Bandwidth (of memories)

We have 15 minutes (= 2 hours − 1h45) to download the videos.

1.
   - Computer A : We have $800MB/(4bytes/word) = 200 \times 10^6$ words. So we need $5\mu s \times 200 \times 10^6$ words $= 1000s = 16.67$ minutes > 15 minutes. We can't use this computer. *(Note: alternatively you could have computed that* $4bytes/5\mu s = 0.8MB/s$ *and divided* $800MB/(0.8MB/s)$*.)*
   - Computer B : We need $800MB/(400MB/s) = 2s < 15$ minutes. We can use this computer.
   - Computer C : We need $800MB/(2MB/s) = 400s = 6.67$ minutes < 15 minutes. We can use this computer.
2.
   - Computer A : We have $2000MB/(4bytes/word) = 500 \times 10^6$ words. So we need $5\mu s \times 500 \times 10^6$ words $= 2500s = 41.67$ minutes > 15 minutes. We can't use this computer.
   - Computer B : We need $2000MB/(400MB/s) = 5s < 15$ minutes. We can use this computer.
   - Computer C : We need $2000MB/(2MB/s) = 1000s = 16.67$ minutes > 15 minutes. We can't use this computer.

2 Number of cache defects

There are 11 defects in total :

Address	Cache (before access)	Effect
1		cache miss
3	Block(0-3)	cache hit
8	Block(0-3)	cache miss
5	Block(8-11)	cache miss
20	Block(8-11), Block(4-7)	cache miss
18	Block(8-11), Block(20-23)	cache miss
19	Block(16-19), Block(20-23)	cache hit
53	Block(16-19), Block(20-23)	cache miss
9	Block(16-19), Block(52-55)	cache miss
11	Block(8-11), Block(52-55)	cache hit
4	Block(8-11), Block(52-55)	cache miss
43	Block(8-11), Block(4-7)	cache miss
5	Block(40-43), Block(4-7)	cache hit
6	Block(40-43), Block(4-7)	cache hit
9	Block(40-43), Block(4-7)	cache miss
18	Block(8-11), Block(4-7)	cache miss

3 Spatial and temporal locality

Spatial locality is more important for this application, as each element of the vector is accessed only once. In this case, small blocks do not help.

4 Cache Memory

The two versions lead to the same number of cache misses. Both versions, initially load 4 elements into the cache to compute p_{11} (4 cache misses), then in order to compute p_{21} or p_{12} the two missing elements have to be loaded into the cache (2 cache misses), now the cache is full. To compute third element, a_{11} and b_{11} are replaced by the two missing elements (2 cache misses). For the computation of the fourth (and final) entry all required elements are in the cache.

Below are more details, in case the explanation above is not clear enough. For version (1), we have the following memory access requests:

```
copy r0, @a11 //cache miss
copy r1, @b11 //cache miss
copy r2, @a12 //cache miss
copy r3, @b21 //cache miss

copy r0, @a21 //cache miss
copy r1, @b11
copy r2, @a22 //cache miss (cache full)
copy r3, @b21

copy r0, @a11
copy r1, @b12 //cache miss (replace @a11)
copy r2, @a12
copy r3, @b22 //cache miss (replace @b11)

copy r0, @a21
copy r1, @b12
copy r2, @a22
copy r3, @b22
```

For version (2), we have the following memory access requests:

```
copy r0, @a11 //cache miss
copy r1, @b11 //cache miss
copy r2, @a12 //cache miss
copy r3, @b21 //cache miss

copy r0, @a11
copy r1, @b12 //cache miss
copy r2, @a12
copy r3, @b22 //cache miss (cache full)

copy r0, @a21 //cache miss (replace @a11)
copy r1, @b11
copy r2, @a22 //cache miss (replace @b11)
copy r3, @b21

copy r0, @a21
```

```
copy r1, @b12
copy r2, @a22
copy r3, @b22
```

5 Buying a computer

1. The program checks if x is prime.
2. The memory accesses are in the following table :

No	Instruction	Memory Access
1	$y \leftarrow 1$	read @1 (cache miss)
2	$y \leftarrow 1$	read block @0 in memory
3	$y \leftarrow 1$	place block @0 in cache
4	$y \leftarrow 1$	read @1 (in cache)
5	$x = 0$?	read @0 (in cache)
6	$x = 1$?	uses register (x already read)
7	$s \leftarrow 0$	read @13 (cache miss)
8	$s \leftarrow 0$	read block @12 in memory
9	$s \leftarrow 0$	place block @12 in cache
10	$s \leftarrow 0$	read @13 (in cache)
11	$i \leftarrow 2$	read @12 (in cache)
12	$s \leq x$?	uses register (s already read)
13	$s \leq x$?	uses register (x already read)
14	$y = 1$?	uses register (y already read)
...	...	only uses registers

3. We have 4 cache hits (accesses in which the values are in the cache) and 2 cache misses (accesses in which the values are not in the cache).
4. To execute the program, the processor accesses the cache memory each time it wants to read or write a variable. There are 7 cache hits in total, 4 when reading as we saw and 3 when writing at the end (for i, y, and s; x, has not been modified). So we calculate the time needed for each computer to finish this program. For compute A, we need $1ns \times 7 + 120ns \times 2 = 247ns$. For computer B, we need $1.2ns \times 7 + 100ns \times 2 \simeq 208$ ns. If we use this example to decide, we choose computer B; which is expected as the program doesn't need a lot of variables/memory; there are few accesses to it but "many" cache misses (relative to the ratio of memory access time to cache access time, about 100).