Assignment 7 - Semil Jain (u1417989)

1. Large Caches (20 points)
Soln:
OS page size = 16KB
LLC size = 32MB
Blocksize = 64 B
#ways = 16 way set-associative

#Sets = (LLC size) / (#ways * Block size)
= 32 * 2^20 / (16 * 64)
= 2^15 sets

Now assuming a 40 or X bit total address (Doesn't matter what length we assume as our offset and index bits would be the least significant bits of the address on the right hand side).

Blk Offset bits = 6 // log2(Block size)
Index bits = 15 // log2(#sets)

Our physical memory will have the same 40 bit as assumed. Out of which:
Page offset = 14 bits // log2(Page size)
Remaining bits indicate the page number.

In order for OS to have full flexibility to select the tiles our bits that decide this need to be in the **Intersection of Page number bits and index bits = (15 + 6 - 14) = 7 bits.**

The 7 leftmost bits in the index can be used to point to the tiles in order to have the control with the OS.

Since the question asks maximum tiles we can use all 7 bits. Hence,

Total Max Tiles = 2^7 = **128 tiles**

2. Virtually Indexed Cache (20 points)
Soln:

OS min page size: 16KB
Assuming L1 cache must be 2-way set-associative
To find the largest L1 cache that we can design.

Since our OS page size is 16KB the page offset bits are 14 bits.

Now if we want to virtually index our physically tagged cache then our index + block offset bits need to be in sync with our page offset bits. Hence total index + block offset bits can be 14 bits. These 14 bits can be partitioned into block sizes and #sets of any sizes and If we design a 1-way cache then the total cache size adds up to 16KB of size.

Now we know that the number of sets and no. of block sizes are kind of invariant. But we can change the no. of ways. The question mentions that L1 must be a 2-way cache.

Hence, our L1 cache size to design must be 2*16 = 32KB.

3. Organizing Ranks (20 points)
Soln:
No. of processor sockets: 2
No. of memory channels: 6
No. Of ranks: 4
DRAM chip capacity: 2Gb, 4Gb or 8 Gb, To maximize our capacity we will take 8Gb chips.
Data output width per chip - 4, 8 or 16. To maximize our capacity we will select a lesser data output width (4).

Total chips per rank = 64b output / each chip width 4 = 16 chips

Maximum Capacity = [2 sockets * 6 channels * 4 ranks * 16 chips * 8Gb]
= 6144 Gb
= **768 GB (Gigabytes)**

Each memory channel operates at 1.2GHz. Hence,

Memory Bandwidth supported = 2 sockets * 6 channels * 1.2G * 2 (DDR, hence 2 transfers per cycle) * 64 (bits per transfer) = 1843.2 Gbps
= **230.4 GBps**

4. Refresh (20 points)
Soln:

Total system capacity: 1 TB = 1024 GB

We have 32 ranks, Hence:
Each rank = 1024 / 32 = 32 GB rank

Each rank has 16 banks:
Each bank = 32 / 16 = 2GB bank

A row has capacity: 8KB in a bank

Total rows: 2GB / 8 KB = 256 K rows each bank

Acc. To DDR standards there are 8K refresh commands

Each refresh command handles = 256K / 8K
= 32 rows

It takes 40ns to refresh one row, Hence:
Total refresh time = 32 * 40 = 1280 ns

Refresh command is issued every 6.8 micro seconds. Hence,

Memory system unavailable performing refresh: 1280 ns/ 7.8 micro s
= 1280 ns/ 7800 ns
= 0.1641 * 100 %
= **16.41 %**

5. Row Buffers (20 points)
Soln:

Assumptions:
- Bus latencies to be 0
- Bank is already recharged at the start
- Row buffer hit - 20ns
- Row buffer conflict - 60ns
- Empty row/ recharged row - 40ns

Row being accessed	Arrival time at memory controller	Hit/Miss (Open)	Open-Page	Hit/Miss (Close)	Close-Page
X	10 ns	empty	**50ns**	empty	**50ns**
					(Precharge to 70)
X	75 ns	rbh	**95ns**	empty	**115ns**
Y	100 ns	rbc	**160ns**	rbc	**175ns**
					(Precharge to 195)
X	190 ns	rbc	**250ns**	empty	**235ns**
					(Precharge to 255)
X	280 ns	rbh	**300ns**	empty	**320ns**
Y	290 ns	rbc	**360ns**	rbc	**380ns**