CS103 – Monsoon 2018 — Homework 6

Neeraj Pandey

Collaborators: None

Question 1: Consider a program that can execute with no stalls and a CPI of 1 if the underlying processor can somehow magically service every load instruction with a 1-cycle L1 cache hit. In practice, 5% of all load instructions suffer from an L1 cache miss, 2% of all load instructions suffer from an L2 cache miss, and 1% of all load instructions suffer from an L3 cache miss (and are serviced by the memory system). An L1 cache miss stalls the processor for 10 cycles while the L2 is looked up. An L2 cache miss stalls the processor for 20 cycles while the L3 is looked up. An L3 cache miss stalls the processor for an additional 300 cycles while data is fetched from memory. What is the CPI for this program if 30% of the program's instructions are load instructions?.

Solution:

Let's take 100 instructions. We have total 30 load instructions (30%). L1 is 5% of 30, so 1.5 misses. L2 is 2% of 30, so 0.6 misses and L3 is 1% of 30, so 0.3 misses.

There is 1 instruction that is not getting data from L1 and L2 and L3, therefore, 10+20+300 = 330 stall cycles. There is 1 instruction which suffer L1 and L2 and is present in L3. Therefore, 10+20=30 stall cycles. There is 3 instructions that will suffer L1 cache miss, therefore 10*3 = 30 stall cycles.

Total stall cycles = 30+30+330 = 390 stall cycles and total cycles = 100 + 390 = 490 cycles.
Total instructions are 100, therefore average CPI = 490/100 = 4.9

Question 2: Consider an L1 cache that has 8 sets, is direct-mapped (1-way), and supports a block size of 64 bytes. For the following memory access pattern (shown as byte addresses), show which accesses are hits and misses. For each hit, indicate the set that yields the hit. (30 points) 0, 48, 84, 32, 96, 360, 560, 48, 84, 600, 84, 48

Solution: Direct-mapped indexes:

- 1 : 0 - 63
- 2 : 63 - 127
- 3 : 128 - 191
- 4 : 192 - 255
- 5 : 256 - 319
- 6 : 320 - 383
- 7 : 384 - 447
- 8 : 448 - 511

Here, S1, S2.... is set 1, 2, ...
0 : Miss S1
48 : Hit S1
84 : Miss S2
32 : Hit S1
96 : Hit S2
360 : Miss S6
560 : Miss S1
48 : Miss S1
84 : Hit S2
600 : Miss S2
84 : Miss S2
48 : Hit S1

Question 3: Assume a 256 KB, 8-way set associative cache with a 32 byte block size. How many sets does the cache have? How many bits are used for the offset, index, and tag, assuming that the CPU provides 32-bit addresses? How large is the tag array? Please show your steps. Now assume that the cache is change to be a direct mapped cache, while keeping the same block size and cache capacity. In this new cache configuration, how many bits will be needed for index offset and tag bits?

Solution: Cache size = Number of ways * Number of sets * Byte block size
Let us take the number of sets as "k".
Therefore, 256 * 1024 = 8 * x * 32Bytes (Given: No of ways = 8, Size of the cache = 256 * 1024 Bytes, Byte-block size = 32Bytes)
x = 1024 (number of sets)
Now, No. of offset bits; $2^5 = 32 \implies 5$
No. of index bits; $2^{10} = 1024 \implies 10$
No. of tag bits = 32 - 5 - 10 =17
Therefore, the size of the tag array = 1024 * 17 * 32 $\implies 544KB$

Now, the cache capacity and block size remains the same if the cache is changed to direct-mapped cache.
Cache Size: Number of ways * Byte block size * Number of sets
256 * 1024 = 1 * k * 32Bytes
$\implies$ k = 8192
Number of bits used; $2^{13} \implies 13$
Number of offset bits; $2^5 \implies 5$
Number of tag bits; $32 - 13 - 5 = 14$

Question 4: Consider a 4-processor multiprocessor connected with a shared bus that has the following properties: (i) centralized shared memory accessible with the bus, (ii) snooping-based MSI cache coherence protocol, (iii) write-invalidate policy. Also assume that the caches have a writeback policy. Initially, the caches all have invalid data. The processors issue the following five requests, one after the other. Fill the table as shown below. Show your work.

Solution:

Request	Cache Hit / Miss	Request on the Bus	Who respond?	State in Cache 1	State in Cache 2	State in Cache 3	State in Cache 4
P1: Write (X)	Miss	Wr X	Bus	M	Inv	Inv	Inv
P2: Read (X)	Miss	Rd X	P1	Shared	Shared	Inv	Inv
P3: Write (X)	Miss	Wr x	Bus	Inv	Inv	M	Inv
P2: Write (X)	Miss	Wr X	P3	Inv	M	Inv	Inv
P4: Read (X)	Miss	Rd X	P2	Inv	Shared	Inv	Shared

Req1 : P1 writes X, borrows it from from the bus and reqrites it.

Req2 : P1 updated the value, P2 then wants to read the value of X. Also, P2 is in shared state now as P1 changed from modifed to shared state.

Req3 : P1 and P2 are in shared state, so P3 wants to write X and P3 receives the value from the X.

Req4 : P2 write to X and therefore P3 responds because it receives the modified X value. Rest of the values are invalid.

Req5 : P4 reads X because P2 received the updated value and responds.

Question 5: On a CPU that has a 32-bit architecture, assume that the TLB that can hold 32 entries. Each entry has Virtual Page Number and the associated Physical Page number. The Physical Page size is 8 KB. Assuming that these are the only two entities stored per entry in the TLB, and that the virtual and physical address spaces are of the same size, what is the total capacity of the TLB?

Solution: Size of the physical page $= 8 * 1024 Bytes$. Hence, 13 bits are required to express this. We know that the virtual adreess and the physical address are of the same size and 13 bits are require for the virtual address. Therefore, total bits for a entry is 13 +13 = 26. So, total number of bits required for the TLB is 26 * 32 = 832Bits.

Question 6: A 2-way set-associative cache has a 16KB capacity. The cacheline size is 8 words, and the word is 32 bits wide. What is the total number of bits required for the Tag field? Assume a 32 bit address.

Solution:

Total cache size = No. of ways * Byte-block size * No. of sets Total size of the cache = 16 * 1024 Bytes
No. of sets = k
Byte-block size = 32Bytes
Therfore, 1026 * 16 = 2 * k * 32Bytes
$\implies$ k = 256
Nuo. of index bits; $2^8 = 256$
No. of offset bits; $2^5 = 32$
Therefore, Number of tag bits used = 32 - 5 - 8 = 19Bits