CS1216 - Monsoon 2022 - Homework 6

Jivansh Sharma
UG 24 1020211193

29/11/2022

Collaborators: None

1. Like we discussed in class, any memory request that cannot be serviced by the on-chip cache hierarchy then has to be serviced by main memory. 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 500 cycles while data is fetched from main memory. Given these set of assumptions for the program and the memory hierarchy under consideration, what is the CPI for this program if 40% of this program's instructions are load instructions ?

Now, we have the following data to work with:

```
1. Let number of instructions  = 100
2. Therefore, 40% or number of load instructions = 40
3. Number of non-load instructions = 60

We already know,
4. 10% of instructions are L1 cache misses > 40 * 0.1 = 4 Cycles
5. 5% of instructions are L2 cache misses > 40 * 0.05 = 2 Cycles
6. 2% of instructions are L3 cache misses > 40 * 0.02 = 0.8 Cycles

Total L1 Misses = 40 - 4 = 36
So, 96 Instructions take 1 cycle

L1 Hit = 1 Cycle
L1 Miss = 10 Cycles

We know that,
There are 4 L1 Misses and 2 L2 Misses. Therefore L1 Misses with L2 Hits are 2.
These take 2 * (10 + 1) Cycles = 22 Cycles
```

```
We know that,
There are 2 L2 Misses and 0.8 L3 Misses. Therefore L2 Misses with L3 Hits are (2 - 0.8) = 1
These take 1.2 * (20 + 10 + 1) Cycles = 37.2 Cycles

We know that,
There are 0.8 L3 Misses. Therefore L3 Misses with Main Memory Hits are 0.8.
These take 0.8 * (500 + 20 + 10 + 1) Cycles = 424.8 Cycles
```

Calculating the CPI

```
CPI = ((96 * 1) + 22 + 37.2 + 424.8) / 100 = 5.8
```

2. Consider an L1 cache that has 16 sets, is direct-mapped (1-way), and supports a block size of 32 bytes. For the following memory access pattern generated by the CPU (the addresses shown below are byte addresses), show which accesses are hits and which are misses. For each hit, indicate the set that yields the hit. Show your work, and how you arrived at the answers.
0, 48, 84, 32, 96, 360, 560, 48, 84, 600, 84, 48.

Due to spacial locality, each set would have 32 bytes. These would be in the manner 0 - 31, 32 - 63 and so on. After set 16 these addresses would loop over and start replacing addresses from set 0. To calculate the set we can take the floor of address / 32 and the mod with 16. This would give us the set number.

Address	Hit/Miss	Set	Why?
0	Miss	0	Compulsory Miss
48	Miss	1	Compulsory Miss
84	Miss	2	Compulsory Miss
32	Hit	1	Already in Set 1
96	Miss	3	Compulsory Miss
360	Miss	11	Compulsory Miss
560	Miss	1	Compulsory Miss Replaces 32 to 63
48	Hit	1	Compulsory Miss, Was Replaced by 560's row
84	Hit	2	Already in Set 2
600	Miss	2	Compulsory Miss Replaces 64 to 95
84	Miss	2	Compulsory Miss Replaces 600's row
48	Hit	1	Already in Set 1

3. Assume a 512 KB, 4-way set associative cache with a 64 byte block size (cache line 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? Show your work.

Now, we have the following data to work with:

```
1. Blocksize = 64 bytes
2. Cache size = 512 KB  = 512 * 1024 = 5,24,288 bytes
3. Associativity = 4 Ways
4. Address size = 32 Bits
5. Let number of sets  = x
```

Using the formula for Cache Size, we get:
`Cache Size` = `Number of Sets` x `Associativity` x `Block Size`

$$\begin{aligned} 5,24,288 &= x \times 4 \times 64 \\ x &= 5,24,288/(4 \times 64) \\ x &= 5,24,288/256 \\ x &= 2048 \end{aligned} \tag{1}$$

Therefore the number of sets are `2048`. Now, we have to find the number of bits used for the offset, index and tag.

$$\begin{aligned} \texttt{Offset} &= \log_2(64) \\ \texttt{Offset} &= 6Bits \\ \texttt{Index} &= \log_2(2048) \\ \texttt{Index} &= 11Bits \\ \texttt{Tag} &= 32 - 11 - 6 \\ \texttt{Tag} &= 15Bits \end{aligned} \tag{2}$$

Therefore, the number of bits used for the offset, index and tag are `6, 11, 15` respectively.
Calculating the size of the tag array:
`Tag Size` = `Number of Sets` x `Associativity` x `Tag Size`

$$\begin{aligned} \texttt{Tag Size} &= 2048 \times 4 \times 15 \\ \texttt{Tag Size} &= 1,22,880\texttt{Bytes} \\ \texttt{Tag Size} &= 1,22,880/1024\texttt{KB} \\ \texttt{Tag Size} &= 120\texttt{KB} \end{aligned} \tag{3}$$

Therefore, the size of the tag array is `120 KB`.
Now, we have to find the number of bits used for the offset, index and tag in the direct mapped cache.

Now, we have the following data to work with:

1. Blocksize = 64 bytes
2. Cache size = 512 KB = 512 * 1024 = 5,24,288 bytes
3. Associativity = 1 Way
4. Address size = 32 Bits
5. Let number of sets = x

Using the formula for Cache Size, we get:
Cache Size = Number of Sets x Associativity x Block Size

$$\begin{aligned} 5,24,288 &= x \times 1 \times 64 \\ x &= 5,24,288/(1 \times 64) \\ x &= 5,24,288/64 \\ x &= 8,192 \end{aligned} \tag{4}$$

Therefore the number of sets are 8192. Now, we have to find the number of bits used for the offset, index and tag.

$$\begin{aligned} \texttt{Offset} &= \log_2(64) \\ \texttt{Offset} &= 6Bits \\ \texttt{Index} &= \log_2(8192) \\ \texttt{Index} &= 13Bits \\ \texttt{Tag} &= 32 - 13 - 6 \\ \texttt{Tag} &= 13Bits \end{aligned} \tag{5}$$

Therefore, the number of bits used for the offset, index and tag are 6, 13, 13 respectively.
Calculating the size of the tag array:
Tag Size = Number of Sets x Associativity x Tag Size

$$\begin{aligned} \texttt{Tag Size} &= 8192 \times 1 \times 13 \\ \texttt{Tag Size} &= 1,06,496\texttt{Bytes} \\ \texttt{Tag Size} &= 1,06,496/1024\texttt{KB} \\ \texttt{Tag Size} &= 104\texttt{KB} \end{aligned} \tag{6}$$

Therefore, the size of the tag array is 104 KB.

4. A 64 MB L3 cache has a 128 byte line (block) size and is 32-way set-associative. How many sets does the cache have? How many bits are used for the offset, index, and tag, assuming that the CPU provides 48-bit addresses? How large is the tag array? Show your work.

Now, we have the following data to work with:

```
1. Blocksize = 128 bytes
2. Cache size = 64 MB = 64 * 1024 * 1024 bytes = 67108864 bytes
3. Associativity = 32 Ways
4. Address size = 48 Bits
5. Let number of sets  = x
```

Using the formula for Cache Size, we get:
`Cache Size` = `Number of Sets` x `Associativity` x `Block Size`

$$\begin{aligned} 67108864 &= x \times 32 \times 128 \\ x &= 67108864/(32 \times 128) \\ x &= 67108864/4096 \\ x &= 16384 \end{aligned} \tag{7}$$

Therefore the number of sets are `16,384`. Now, we have to find the number of bits used for the offset, index and tag.

$$\begin{aligned} \texttt{Offset} &= \log_2(128) \\ \texttt{Offset} &= 7Bits \\ \texttt{Index} &= \log_2(16384) \\ \texttt{Index} &= 14Bits \\ \texttt{Tag} &= 48 - 7 - 14 \\ \texttt{Tag} &= 27Bits \end{aligned} \tag{8}$$

Therefore, the number of bits used for the offset, index and tag are `7, 14, 27` respectively.
Calculating the size of the tag array:
`Tag Size` = `Number of Sets` x `Associativity` x `Tag Size`

$$\begin{aligned} \texttt{Tag Size} &= 16384 \times 32 \times 27 \\ \texttt{Tag Size} &= 1,41,55,776\texttt{Bytes} \\ \texttt{Tag Size} &= 1,41,55,776/1024\texttt{KB} \\ \texttt{Tag Size} &= 13,824/1024\texttt{MB} \\ \texttt{Tag Size} &= 13.824\texttt{MB} \end{aligned} \tag{9}$$

Therefore, the size of the tag array is `13.824 MB`.

5. For the following access pattern, (i) indicate if each access is a hit or miss. (ii) What is the hit rate? Assume that the cache has 2 sets and is 2-way set-associative. Assume that block A maps to set 0, B to set 1, C to set 0, D to set 1, E to set 0, F to set 1. Assume an LRU replacement policy.

Now, we have the following data to work with:

```
1. A, C, E Map to Set #0
2. B, D, F Map to Set #1
3. LRU Replacement Policy
4. 2 Way 2 Set Associative Cache
5. In each block, LRU is Rightmost and MRU is Leftmost
```

Serial Number	Address	Hit/Miss	Cache 0 State	Cache 1 State	Why?
1.	A	Miss	A		Compulsory Miss
2.	B	Miss	A	B	Compulsory Miss
3.	C	Miss	C, A	B	Compulsory Miss
4.	B	Hit	C, A	B	
5.	A	Hit	A, C	B	
6.	B	Hit	A, C	B	
7.	E	Miss	E, A	B	Compulsory Miss
8.	C	Miss	C, E	B	Conflict Miss
9.	A	Miss	A, C	B	Capacity Miss
10.	D	Miss	A, C	D, B	Compulsory Miss
11.	B	Hit	A, C	B, D	
12.	C	Hit	C, A	B, D	
13.	A	Hit	A, C	B, D	
14.	F	Miss	A, C	F, B	Compulsory Miss
15.	D	Miss	A, C	D, F	Capacity Miss
16.	B	Miss	A, C	B, D	Conflict Miss
17.	C	Hit	C, A	B, D	
18.	E	Miss	E, C	B, D	Conflict Miss
19.	A	Miss	A, E	B, D	Conflict Miss

Now, we know Hit Rate = (Total Hits/Total Requests) * 100
Using the formula for CPI, we get:
Hit Rate = (Total Hits/Total Requests) * 100

$$\texttt{Hit Rate} = 7/19 * 100 = 36.84\% \tag{10}$$

6. 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, 10% of all load instructions suffer from an L1 cache miss. Every cache miss results in a 300-cycle stall while data is fetched from memory. What is the CPI for this program if 20% of the program's instructions are load instructions?

Now, we have the following data to work with:

```
1. Baseline CPI = 1
2. Miss Rate = 10\%
3. Miss Penalty = 300 cycles
4. Percentage of load instructions = 20\%
```

Using the formula for CPI, we get:
CPI = Baseline + (Miss Rate * Miss Penalty * `Percentage of load instructions`)

$$CPI = 1 + (0.1 * 300 * 0.2) = 1 + 6 = 7 \tag{11}$$

Therefore the CPI would be `7`.