CS1216 - Monsoon 2022 - Homework 7

Jivansh Sharma
UG 24 1020211193

10/12/2022

Collaborators: None

1. 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.
P1: Write X
P2: Read X
P3: Write X
P2: Write X
P4: Read X

Request	Hit/Miss	Request On Bus	Who Responds ?	State in Cache 1	State in Cache 2	State In Cache 3	State in Cache 4	Why ?
P1: Write X	Miss	Upgrade X	No Response; all invalidate	Modified	Invalid	Invalid	Invalid	No snooping possible, value is not on bus either, only way is to update the value directly in memory.
P2: Read X	Miss	Read X	P1 Responds, following a memory writeback.	Shared	Shared	Invalid	Invalid	When request is on bus, P1 snoops and shares the value of X with P1.
P3: Write X	Miss	Upgrade X	No response. All others invalidate self	Invalid	Invalid	Modified	Invalid	No snooping is possible, value is on the bus but about to be invalidated, only way is to update the value directly in memory.
P2: Write X	Miss	Write X	P3 Responds	Invalid	Modified	Invalid	Invalid	Can be directly updated to modified through snooping by P3
P4: Read X	Miss	Read X	P2 Responds, following a memory writeback	Invalid	Shared	Invalid	Shared	P2 and P4 share the answer value.

2. On a CPU that has a 32-bit architecture, assume that the TLB can hold 64 entries. Each entry has a Virtual Page Number (VPN) and the associated Physical Page number (PPN). 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?
Based on the discussions held in class, what is the biggest drawback of this TLB design? Think about the cases where there might be multiple programs running simultaneously on the machine.

The biggest drawback of this TLB design would be it's inability to handle multiple programs running simultaneously on the machine. Due to having only 64 entries, there can potentially be a lack of space to store the virtual and physical page number mappings for all the programs running on the machine, which could furthere lead to a performance thorttling as the CPU would need to frequently access the main memory to retrieve the necessary page mappings, leading to a slower overall processing speed.

$$\begin{aligned}\text{Total Capacity of TLB} &= \text{Number of Entries} \times \text{Size of each entry} \\ &= 64 \times 8 \text{ KB} \\ &= 512 \text{ KB}\end{aligned} \tag{1}$$

3. Given a 4-way set associative cache, a FIFO replacement policy and a block access pattern shown below, which accesses are hits and which ones misses? Assume that all the blocks map to the same set and it is a cold cache, i.e. there is no data in the cache before the first access to B1 is seen by the cache. Now, assume that the replacement policy is changed to LIFO. What will be the pattern of hits and misses in this case? Show your work.
FIFO (First in First Out) replacement policy is relatively straightforward. Of the various blocks that are present in different ways of a given set, one that will be replaced / evicted, in all likelihood to make space for an incoming block that maps to the same set, will be the one that has been the first to be brought in. Similarly, in the LIFO policy, the block that will be replaced will be the one that has been brought in last.
B1, B2, B3, B4, B5, B1, B2, B3, B4, B5, B1, B2, B3, B4, B5.

In both cases, the first five accesses result in misses because the cache is empty. After that, all the subsequent accesses result in hits as the necessary blocks are already

Both for FIFO and LIFO:

Block	Hit/Miss	Why?
B1	Miss	Cache Empty
B2	Miss	Cache Empty
B3	Miss	Cache Empty
B4	Miss	Cache Empty
B5	Miss	Cache Empty
B1	Hit	Already Present
B2	Hit	Already Present
B3	Hit	Already Present
B4	Hit	Already Present
B5	Hit	Already Present
B1	Miss	All blocks are occupied, need to make space for incoming block
B2	Hit	Already Present
B3	Hit	Already Present
B4	Hit	Already Present
B5	Hit	Already Present

4. In general, cache access time is proportional to capacity. Assume that main memory accesses take 75 ns and that memory accesses are 40% of all instructions. The following table shows data for L1 caches attached to each of two processors, P1 and P2.

(a) Assuming that the L1 hit time determines the cycle times for P1 and P2, what are their respective clock rates?

The clock rates can be calculated by taking the reciprocal of the Hit Time.

$$\begin{aligned}\text{Clock Rate} &= \frac{1}{\text{Hit Time}}\\ &= \frac{1}{\texttt{Hit Time for P1}}\\ &= \frac{1}{\texttt{0.66ns}}\\ &= 1.5\text{ GHz}\end{aligned} \tag{2}$$

$$\begin{aligned}\text{Clock Rate} &= \frac{1}{\text{Hit Time}}\\ &= \frac{1}{\texttt{Hit Time for P2}}\\ &= \frac{1}{\texttt{0.9ns}}\\ &= 1.1\text{ GHz}\end{aligned} \tag{3}$$

(b) What is the Average Memory Access Time for P1 and P2?

Average Memory Access Time For P1 = L1 hit time + (L1 miss rate * Memory Access Time)

We know that the L1 hit time for P1 is 0.66ns and the L1 miss rate is 0.1. The memory access time is 75ns. Therefore, the Average Memory Access Time for P1 is 0.66ns + (0.1 * 75ns) = 0.66ns + 7.5ns = 8.16ns

Average Memory Access Time For P2 = L1 hit time + (L1 miss rate * Memory Access Time)

We know that the L1 hit time for P2 is 0.9ns and the L1 miss rate is 0.2. The memory access time is 75ns. Therefore, the Average Memory Access Time for P2 is 0.9ns + (0.2 * 75ns) = 0.9ns + 15ns = 15.9ns

(c) Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster?

Total CPI For P1 = Base CPI + [(Memory Access Time*L1 miss rate)/L1 Hit time] * Number of memory instructions

$$\begin{aligned}\text{Total CPI} &= \text{Base CPI} + \frac{\text{Memory Access Time} \times \text{L1 miss rate}}{\text{L1 Hit time}} \times \text{Number of memory instructions}\\ &= 1.0 + \frac{75ns \times 0.1}{0.66ns} \times 40\%\\ &= 1.0 + \frac{7.5ns}{0.66ns} \times 40\%\\ &= 1.0 + 0.4545 \times 40\%\\ &= 1.0 + 0.1818\\ &= 1.1818\end{aligned} \tag{4}$$

Total CPI For P2 = Base CPI + [(Memory Access Time*L1 miss rate)/L1 Hit time] * Number of memory instructions

$$\begin{aligned}\text{Total CPI} &= \text{Base CPI} + \frac{\text{Memory Access Time} \times \text{L1 miss rate}}{\text{L1 Hit time}} \times \text{Number of memory instructions}\\ &= 1.0 + \frac{75ns \times 0.2}{0.9ns} \times 40\%\\ &= 1.0 + \frac{15ns}{0.9ns} \times 40\%\\ &= 1.0 + 0.1667 \times 40\%\\ &= 1.0 + 0.0667\\ &= 1.0667\end{aligned} \tag{5}$$

5. Media applications that play audio or video files are part of a class of workloads called "streaming" workloads; i.e., they bring in large amounts of data but do not reuse much of it. Consider a video streaming workload that accesses a 512 KB working set sequentially with the following address stream

(a) Assume a 64 KB direct-mapped cache with a 32-byte block. What is the miss rate for the address stream above? How is this miss rate sensitive to the size of the cache or the working set?

In a direct mapped cache, with a 32-byte block. Out of 512KB of the working set, 64KB of the working set is mapped to the cache. The remaining 448KB of the working set is not mapped to the cache. This results in a miss rate of 448KB/512KB = 87.5%.

This miss rate is sensitive to the size of the cache and the working set. As the cache size or the working set size increases, the miss rate decreases. Similarly, as the cache size or the working set size decreases, the miss rate increases.

(b) Re-compute the miss rate when the cache block size is 16 bytes, 64 bytes, and 128 bytes. What kind of locality is this workload exploiting?

Block Size	Miss Rate	Locality
16 Bytes	87.5%	Temporal
64 Bytes	87.5%	Temporal
128 Bytes	87.5%	Temporal

In each the Miss Rate is being calculated by Miss Rate = 1 - (Working Set Size / (Cache Size * Block Size))

Because the working set and cache sizes are constant, the miss rate is unaffected in every situation.

Spatial locality, which refers to the reuse of data within a relatively limited region of memory, is being utilised by this workload. The data is retrieved in this situation in a sequential manner, which suggests that the data items that are read simultaneously are probably close to one another in memory. As a result, there are fewer misses since the cache can effectively store and reuse the data.