Microprocessors – Lab 08

Cormac Sharkey - 20333458

# Question 1

Briefly describe why software developers should care about caches.

Answer: Caches are a crucial part of all computers and microcontrollers, as they improve the speed of computer operations by a large amount. Without using a cache, a CPU would need to store any data it is currently using in the non-volatile memory, like the hard drive, which is very slow to read and write to. As a result, operations with the cache disabled are very slow compared to normal computer operations.

Conversely, with the cache enabled, the CPU can quickly read and write to the volatile memory of the cache, which makes data operations very fast. Since software developers are focused primarily on software, they should care about caches because the appropriate use of caches can increase their software runtime.

# Question 2

Briefly describe the difference between a write-through and a write-back cache.

Answer: With write-through cache, any updates to any data in the cache is almost immediately written to the non-volatile memory, like the hard drive, too. As mentioned in Question 1’s answer, writing to non-volatile memory takes more time than writing to volatile memory, so this form of cache is slower than the other type. However, it is good when you’re running multiple processes on that device, as there is always an up-to-date copy of whatever is currently in the cache, in the non-volatile memory.

With write-back cache, not all data in the cache is stored directly into non-volatile memory at every change. Data in the cache is only stored in non-volatile memory when that specific block of data, or cache entry, is being removed from the cache. At this stage, discarding it would cause the data to be lost, thus it is written onto the hard drive. This form of cache is faster than write-through cache, due to the lack of constant non-volatile memory updating. However, it isn’t good for multiple processes running on the device, as not all changed/modified data is accessible to every processor in the device.

# Question 3

Describe and contrast the differences between *Full-Associative* / *Direct-Mapped* and *N-Way Set-Associative* cache organizations.

Answer: Full-Associative cache is a form of cache organization where each block of memory in the non-volatile memory can be stored in any location in the cache, each of which is called a cache entry. This is a more complicated form of cache organization, as each cache entry must be checked when looking for a specific block of memory in the cache, since any block of memory can be anywhere in the cache.

Direct-Mapped cache is a form of cache organisation where each block of memory can only go to one cache entry, meaning that a specific block goes to a specific entry in cache. This is an opposite to the previous form of cache organization and has a good benefit. Since each block of memory can only go in one cache entry, it is obvious where any block of memory stored in cache is at any time. However, since the non-volatile memory is larger than the cache, an error can occur where two blocks of memory map to the same cache entry. When this occurs, the second block of memory cannot be simply forced into another entry, as doing this breaks the rules of the cache organization.

Finally, N-Way-Set-Associative cache is a form of cache organization that combined both of the previous forms of organization. Each block of memory can fit into N number of cache entries, meaning that the error of two blocks mapping to one cache entry can be reduced and even avoided. At the same time, when looking for a block of memory in cache, it is only necessary to look in the N number of places the memory block may be, opposed to every cache entry.

# Question 4

Caches are an important and scarce resource – describe how they are managed for maximum benefit.

Answer: There are many different ways caches are managed to ensure they perform as efficiently as possible. One attribute of caches that are all operating in a single system is coherency. This is like a measure of how well the caches are sharing data and how well the data is managed between them and the things accessing the data, such as the CPU, other caches and non-volatile memory. Two common ways of doing this is using a directory-based approach and using a bus snooping-based approach.

The directory approach consists of storing any data shared between caches in a common place, called a directory, where it is naturally managed and kept coherent. The bus snooping approach involves having a controller monitor the bus between caches. If there is a change to data that is being shared, the controller can alert the other caches to the change and update them appropriately, ensuring coherency.

Another aspect of cache management is based on locality, which involves accessing data based on connections to currently accessed data. This can take the form of spatial locality and temporal locality. Spatial locality is derived from the idea that memory blocks will be accessed if a nearby block has already been accessed. As a result, nearby memory blocks may be loaded into cache pre-emptively.

With temporal locality, the idea is that memory blocks that are more frequently or more recently accessed are more likely to be accessed again. As a result, the memory blocks that were the least accessed most recently may be removed from the cache to free up room for other memory blocks.

# Question 5

Mov r0, #0 @Set r0 to 0

For\_loop:

Ldr r3, [r1] @Load the value at address of r1 into r3

Ldr r4, [r2] @Load the value at address of r2 into r4

Add r3, r4 @Add r4 to r3

Str r3, [r1] @Store r3 back into the value at address of r1

Add r2, #1 @Increment the address by 1

Adds r0, #1 @Increment the counter by 1

Cmp r0, #5 @Compare the counter to 5

Bne For\_loop @If not equal, loop again

Cold cache: 192 ns

Warm cache: 91 ns

No cache: 560 ns