Q1) As is evident from lab 07, caches greatly improve the performance of a program. It provides a high-performance memory which can be used to decrease time required for code execution. It does so by reducing the number of times a program has to access lower level storage by “caching” it. “Caching” simply means that it maintains a copy of the frequently accessed data for easier retrieval. By maintaining a copy in the cache, it speeds up the performance as each instruction including read/write can be performed much quicker. This increases the data retrieval performance and thus improving overall performance as well.

Q2)

|  |  |
| --- | --- |
| Write-through Cache | Write-back Cache |
| (a) When you have to write new changes, the lower level storage and cache memory are updated simultaneously. Hence both will always have the same data. | (a) When you have to write new changes, only the cache memory is updated. Hence both locations can have different value of the same piece of data. Data is only written to main memory when it is deleted from cache. |
| (b) Cache operations are usually slower and require a higher bus bandwidth when compared to write-back cache | (b) It will have only one write operation per update and one extra write when the data is purged from cache memory and written to the main memory. |
| (c) Simple to conceptualize and good for multiple-bus-master systems. | (c)  This may cause issues when we have multiprocessor or multi-bus-master systems |

Q3)

Direct-Mapped Cache: It has a simple structure where a block in the main memory is mapped to a block in the cache memory. It is hence easy to identify individual entry in the cache and its corresponding address in the main memory. A conflict miss issue can arise, this happens when two different addresses in the main memory correspond to the same address in the cached memory. There can also be empty blocks in the cache memory since each block is attached to a block memory but we cannot be sure that the assigned block contains the required data. They require the least amount of SOC area to implement.

Full Associative Cache: It has a more complex structure than Direct-Mapped Cache allowing it to store main memory addresses into any block in the cache memory. This allows for the cache to be utilized fully and prevent conflict misses. The downside is that when we look for a particular address in the cache, we have to check through all the current entries in the cache memory. This can be a time consuming event. These require a lot of SOC area to implement.

N-Way Associative Cache: It works similar to a Direct-Mapped Cache, instead of a one-one map between the main and cache memory there is an N-one map. Multiple blocks in the main memory can be assigned the same block in the cache memory. This allows more of the cache memory to be filled relevant data and reduces conflict misses. The ‘N’ refers to the degree of associativity of the cache, i.e. how many main memory blocks are linked to the same block in cache. This is a compromise between direct mapped and full associative cache.

Q4) Cache provides faster retrieval times by reducing the number of time a program needs to access the slow lower level storage. This done by saving the frequently requested data in the cache memory. The program looks for the data in the cache memory first instead of going directly to the lower level memory. It is hence a very important resource when designing any kind of program.

Prefetching additional data during a cache miss decreases the chances of cache misses.

Cache is partitioned from the RAM making this memory relatively small. Since cache is scarce as well, it is regularly updated and the older data is purged from the memory and replaced by the new data. If a program cannot find the required data in the RAM, it retrieves it from the main storage and then writes it to the cache so that it may be retrieved easier the next time it has to be accessed. This can result in the overwriting of the older data if the cache is full.

For maximum benefit the more frequently accessed data should be given priority to store in the cache. Data which remains constant even after regular access is better to store in the cache. This is because the data has to update in the cache as well as the location in the main memory, making frequent updates more expensive due to the double writes.

Q5)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CODE | Cold Cache  First Run | Loop - CC | Warm Cache  First Run | Loop - WC | Cache Disabled First Run | Loop - CD |
| movs r0, #10 | 10ns | - | 1ns | - | 10ns | - |
| loop: ldr r1, [r2] | (10 + 10)ns | 1ns | 1ns | 1ns | (10 + 10)ns | (10 + 10)ns |
| subs r0, r1 | 10ns | 1ns | 1ns | 1ns | 10ns | 10ns |
| adds r1, r1 | 10ns | 1ns | 1ns | 1ns | 10ns | 10ns |
| str r1, [r2] | (10 + 10)ns | (1 + 10)ns | (1+ 10)ns | (1 + 10)ns | (10 + 10)ns | (10 + 10)ns |
| b loop | 10ns | 1ns | 1ns | 1ns | 10ns | 10ns |