|  |  |  |  |
| --- | --- | --- | --- |
| Instructor |  | Due Date |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | **1** | **2** | **3** | **4** | Total |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) 6 in your Computer Architecture and Organization textbook.

**Part 1 Glossary Terms - Computer Memory**

Define, in detail, each of these glossary terms from the realm of computer architecture and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes

or course textbook(s) as references or the Internet by visiting Web sites such as:

[**http://www.bing.com**](http://www.bing.com) or [**http://www.webopedia.com**](http://www.webopedia.com/)

**(a) Associative Memory**

|  |
| --- |
| Associative memory is an expensive cache mapping schema that allows a main memory (RAM or VRAM) block to be mapped to any block in the cache, which requires the processor to check a main memory tag field against all tags stored in the cache (thus, making it an expensive mapping schema). |

**(b) DRAM**

|  |
| --- |
| Dynamic Random Access Memory (DRAM) is less expensive, slower, and requires less cooling that static RAM (SRAM), and is often used as main memory (whereas SRAM is often used for L1 and L2 caches). |

**(c) EPROM**

|  |
| --- |
| Electrically Programmable Read Only Memory (EPROM) is exactly what it sounds like: read-only memory that can be reprogammed using electricity and special hardware. ROM memory does not lose its contents after it is turned off (a variety of physical / chemical methods are used to achieve this), and EPROM is often used as a form of re-writable storage (e.g., flash memory drives which are a form of EPROM). |

**(d) Memory Hierarchy**

|  |
| --- |
| The memory hierarchy refers to system architecture designs for memory access. For any data access, a processor will begin with checking the smallest, fastest, most expensive forms of memory that are closest to the processor physically (i.e., cache memory), before moving to lower partitions of memory (i.e., main memory, and then secondary memory, and then tertiary memory, and then offline / removable memory drives). This ensures the fastest possible access times.   Often memory access is done in parallel, with an access to cache and main memory made in tandem, so that if an access to a cache misses, the access to main memory is already in progress. |

**(e) Virtual Memory**

|  |
| --- |
| Virtual memory (VRAM) is an extension of main memory address space into secondary storage (SSDs or disks) so that computers can hold more processes in main memory. Solid state drives, which have very fast access times, make VRAM tenable.   VRAM works by mapping virtual addresses in a “page file” on a hard drive to physical addresses in blocks called “pages”, and a process called “paging” is used to copy a page from the drive to an allocated “page frame” in main memory when it is needed.   Virtual memory enables us to write programs whose memory requirements surpass the physical address space available in main memory, which in turn enables us to write inefficient, unoptimized programs such as this 92 mb module for determining if a number is even or not: <https://github.com/samuelmarina/is-even> |

**Part 2 Exercises - Computer Memory**

For each of the following, enter True or False.

**TRUE (1)** All cache mapping schemes require a main memory address to have an offset field.

**TRUE (2)** It is important to know if a computer is byte or word addressable because we need to know how many addresses are contained in main memory, cache and in each block when doing cache mapping.

**FALSE (3)** Caching breaks down when programs exhibit good locality.

**TRUE (4)** The two types of cache write policies are write - through and write - back.

**TRUE (5)** A unified cache is a cache that holds both data and instructions.

**TRUE (6)** When a computer uses paging, there must be a page table for every process.

**FALSE (7)** Memory segmentation can result in internal fragmentation, while paging can result in external fragmentation.

**FALSE (8)** Assuming an 8 - bit virtual address with pages of 32 bytes, the virtual address format is 5 bits for the page and 3 bits for the offset.

**TRUE (9)** Cache replacement policies are necessary to determine which block in cache should be the victim block.

**TRUE (10)** Information can be retrieved fastest from a hard disk as compared to magnetic tapes, optical disks and USB flash drives.

**Part 3 Exercises - Computer Memory**

**(1)** Suppose we have a byte - addressable computer using direct mapping with 16 - bit main memory addresses and 32 blocks of cache. If each block contains 8 bytes, determine the size of the offset field.

**Offset field would 3 bits.**

**(2)** Suppose we have a byte - addressable computer using direct mapping with 16 - bit main memory addresses and 32 blocks of cache. If each block contains 8 bytes, determine the size of the block field.

**Block fields would be 5 bits.**

**(3)** Suppose we have a byte - addressable computer using direct mapping with 16-bit main memory addresses and 32 blocks of cache. If each block contains 8 bytes, determine the size of the tag field.

**Tag field would be 8 bits.**

**(4)** Consider a byte - addressable computer with 24 - bit addresses, a cache capable of storing a total of 64K bytes of data, and blocks of 32 bytes. Show the format of a 24 - bit memory address if the computer uses 4 - way set associative mapping.

|  |  |  |
| --- | --- | --- |
| Tag | Set | Offset |
| **11** | **5** | **4** |

**(5)** Consider a byte - addressable computer with 24 - bit addresses, a cache capable of storing a total of 64K bytes of data, and blocks of 32 bytes. Show the format of a 24 - bit memory address if the computer uses direct mapping.

|  |  |  |
| --- | --- | --- |
| Tag | Set | Offset |
| **3** | **16** | **5** |

**Part 4 Exercises - Computer Memory**

Write a complete answer for each of these.

**(1)** Does a TLB miss always indicate that a page is missing from memory?   
 Explain your answer.

|  |
| --- |
| No, a miss on a translation look-aside buffer (TLB) doesn’t indicate that a page is missing from memory but only that it isn’t in the buffer. After a miss on the TLB, the next step would be to look up the frame number for the page on the page table and use the frame number to look for the page in memory.   The buffer is there to speed up page table lookup and acts as a cache for fetching virtual page / frame pairs. |

**(2)** If you are a computer builder trying to make your system as price - competitive as possible, what features and organization would you select for its memory hierarchy?

|  |
| --- |
| I would save $ on RAM by using virtual memory on SSDs, and make extensive use of caching to create a highly performant machine using the least / least expensive hardware possible. |

**(3)** If you are a computer buyer trying to get the best performance from a system, what features would you look for in its memory hierarchy?

|  |
| --- |
| I would look for large SRAM caches, plenty of DRAM in main memory, and SSDs with VRAM enabled. |

**(4)** Name two ways that, as a programmer, you can improve cache performance.

|  |
| --- |
| As a programmer strictly interested in performance, I could write programs that use good locality of reference, using spatial locality (arrays, lists, loops, etc.) and writing programs that use a sequential flow of control and avoid using object-oriented-programming.  However, as a programmer who must write complex programs and work as part of a team, I would not abandon Object Oriented Programming in favor of marginal performance benefits, unless I was working in a very restrictive architecture (e.g., as an embedded engineer).  I’ve seen some readable, complex, and, I assume, more performant, applications of purely functional / procedural programming, however when it comes to programming at scale and complexity, I don’t think anything can beat OOP and the benefits of class inheritance, abstraction, polymorphism, and encapsulation that it provides. |

**(5)** Look up a specific vendor’s specifications for memory and report the memory access time, cache access time and cache hit rate ( and any other data the vendor provides ) .

|  |
| --- |
|  |