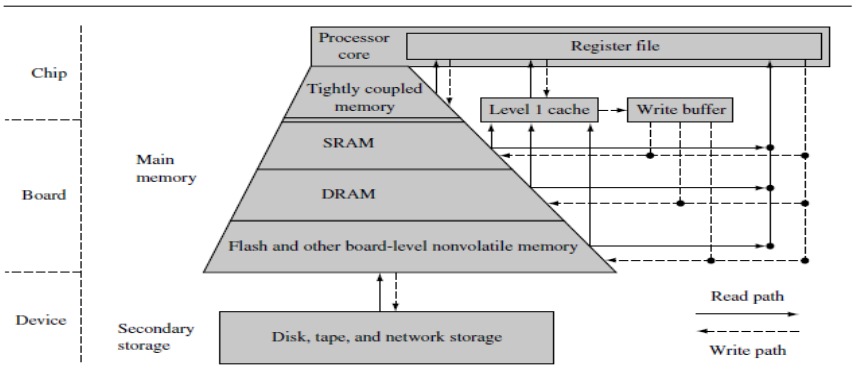
USN

Internal Assessment Test 3 – JULY 2024

Sub:	MICROCONTROLLERS					Sub Code:	BCS402	Branch:	CSE
Date:	06/08/24	Duration:	90 mins	Max Marks:	50	Sem / Sec:	IV Sem A/B/C		OBE

Answer any FIVE FULL Questions — MARKS

			CO	RBT
1. a)	Briefly explain what happens when an IRQ and FIQ exception is raised with an ARM processor. IRQ and FIQ exceptions only occur when a specific interrupt mask is cleared in the cpsr. The ARM processor will continue executing the current instruction in the execution stage of the pipeline before handling the interrupt—an important factor in designing a deterministic interrupt handler since some instructions require more cycles to complete the execution stage. An IRQ or FIQ exception causes the processor hardware to go through a standard procedure (provided the interrupts are not masked): 1. The processor changes to a specific interrupt request mode, which reflects the interrupt being raised. 2. The previous mode's cpsr is saved into the spsr of the new interrupt request mode. 3. The pc is saved in the lr of the new interrupt request mode. 4. Interrupt/s are disabled—either the IRQ or both IRQ and FIQ exceptions are disabled in the cpsr. This immediately stops another interrupt request of the same type being raised. 5. The processor branches to a specific entry in the vector table.The procedure varies slightly depending upon the type of interrupt being raised. We will illustrate both interrupts with an example. The first example shows what happens when an IRQ exception is raised, and the second example shows what happens when an FIQ exception is raised.	[6]	CO4	L2
1. b)	What is interrupt latency and how software handler can minimize the interrupt latency. Interrupt-driven embedded systems have to fight a battle with interrupt latency—the interval of time from an external interrupt request signal being raised to the first fetch of an instruction of a specific interrupt service routine (ISR). Interrupt latency depends on a combination of hardware and software. System architects must balance the system design to handle multiple simultaneous interrupt sources and minimize interrupt latency. If the interrupts are not handled in a timely manner, then the system will exhibit slow response times. Software handlers have two main methods to minimize interrupt latency. The first method is to use a nested interrupt handler, which allows further interrupts to occur even when currently servicing an existing interrupt The second method involves prioritization. You program the interrupt controller to ignore interrupts of the same or lower priority than the interrupt you are handling, so only a higher-priority task can interrupt your handler. You then re enable interrupts.	[4]	CO4	L1
2 a)	Describe the features of Red Hat RedBoot firmware tool. RedBoot is a firmware tool developed by Red Hat. It is provided under an open source license with no royalties or up front fees. RedBoot is designed to execute on different CPUs (for instance, ARM, MIPS, SH, and so on). It provides both debug capability through GNU Debugger (GDB), as well as a bootloader. The RedBoot software core is based on a HAL. RedBoot supports these main features: ■ Communication—configuration is over serial or Ethernet. For serial, X-Modem proto-	[6]	CO4	L2

	col is used to communicate with the GNU Debugger (GDB). For Ethernet, TCP is used to communicate with GDB. RedBoot supports a range of network standards, such as bootp, telnet, and tftp. Flash ROM memory management—provides a set of filing system routines that can download, update, and erase images in flash ROM. In addition, the images can either be compressed or uncompressed. ■ Full operating system support—supports the loading and booting of Embedded Linux, Red Hat eCos, and many other popular operating systems. For Embedded Linux, RedBoot supports the ability to define parameters that are passed directly to the kernel upon booting.			
2.b)	Illustrate the steps in the execution flow of sandstone code structure . Sandstone consists of a single assembly file. The file structure is broken down into a number of steps, where each step corresponds to a stage in the execution flow of Sandstone Step 1: Take the Reset Exception Execution begins with a Reset exception. Only the reset vector entry is required in the default vector table. Step 2: Start Initializing the Hardware The primary phase in initializing hardware is setting up system registers. These registers have to be set up before accessing the hardware. Step 3: Remap Memory One of the major activities of hardware initialization is to set up the memory environment. Sandstone is designed to initialize SRAM and remap memory. This process occurs fairly early on in the initialization of the system. Step 4: Initialize Communication Hardware Communication initialization involves configuring a serial port and outputting a standard banner. The banner is used to show that the firmware is fully functional and memory has been successfully remapped. Step 5: Bootloader—Copy Payload and Relinquish Control The final stage involves copying a payload and relinquishing control of the pc over to the copied payload.	[4]	CO4	L3
3 a)	With a neat diagram explain the levels in memory hierarchy and the cache memory .  Figure 12.1 Memory hierarchy. Level 1-Processor Core ● The innermost level of the hierarchy is at the processor core. ● This memory is so tightly coupled to the processor that in many ways it is difficult to think of it as separate from the processor.	[6]	CO5	L2

	● This memory is known as a register file. These registers are integral to the processor core and provide the fastest possible memory access in the system. ● memory components are connected to the processor core through dedicated on-chip interfaces. ● It is at this level we find tightly coupled memory (TCM) and level 1 cache. Level 2-Main memory ● The primary level is the main memory. It includes volatile components like SRAM and DRAM, and nonvolatile components like flash memory. ● The purpose of main memory is to hold programs while they are running on a system. ● Layer 3-secondary memory The next level is secondary storage—large, slow, relatively inexpensive mass storage devices such as disk drives or removable memory. Also included in this level is data derived from peripheral devices.			
3. b)	Describe the terms to measure the cache efficiency. There are two terms used to characterize the cache efficiency of a program: the cache hit rate and the cache miss rate. The hit rate is the number of cache hits divided by the total number of memory requests over a given time interval. The value is expressed as a percentage: $$hit\ rate = \left(\frac{cache\ hits}{memory\ requests}\right) \times 100$$ The miss rate is similar in form: the total cache misses divided by the total number of memory requests expressed as a percentage over a time interval. Note that the miss rate also equals 100 minus the hit rate.	[4]	CO5	L4

4 a)	Write a short note on i) Direct mapped Caches ii) Set Associative Caches	[10]	CO5	L2

i. Direct mapped Caches:

- In a direct-mapped cache each addressed location in main memory maps to a single location in cache memory.
- Since main memory is much larger than cache memory, there are many addresses in main memory that map to the same single location in cache memory.
- Figure 12.5 shows where portions of main memory are temporarily stored in cache memory.
- The figure represents the simplest form of cache, known as a direct-mapped cache.
- The figure shows this relationship for the class of addresses ending in 0x824.

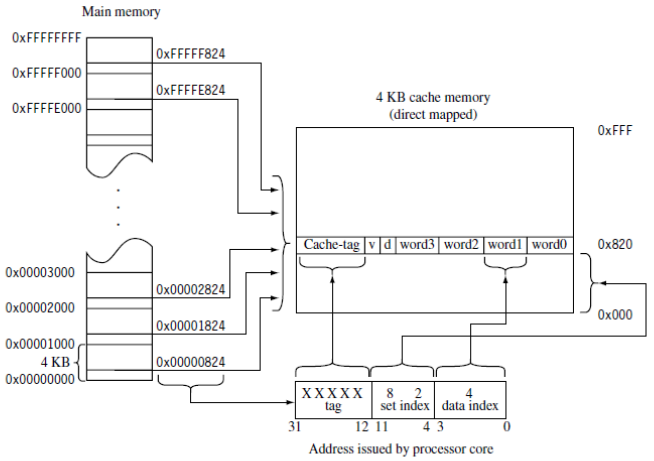


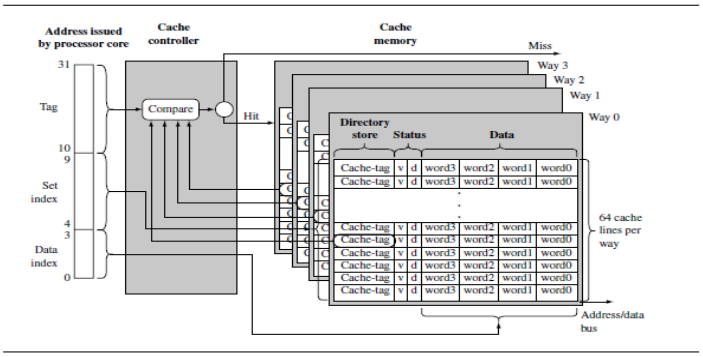
Figure 12.5 How main memory maps to a direct-mapped cache.

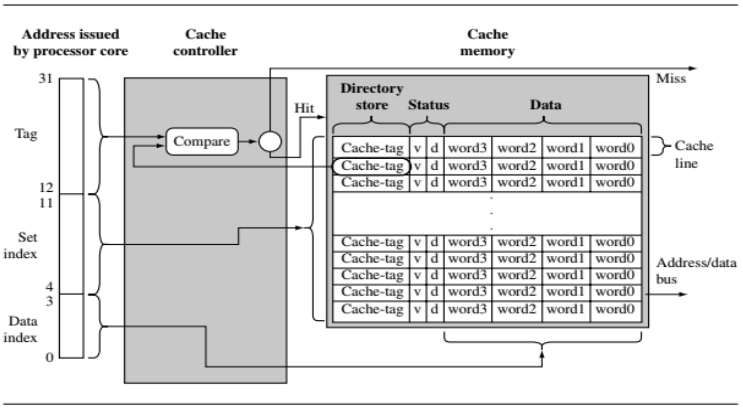
Direct Mapping-Eviction & Thrashing

If valid data exists in this cache line but represents another address block in main memory, the entire cache line is evicted and replaced by the cache line containing the requested address. This process of removing an existing cache line as part of servicing a cache miss is known as eviction—returning the contents of a cache line to main memory from the cache to make room for new data that needs to be loaded in cache.
Direct-mapped caches are subject to high levels of thrashing—a software battle for the same location in cache memory.

Set Associative Caches

Some caches include an additional design feature to reduce the frequency of thrashing.
This structural design feature is a change that divides the cache memory into smaller equal units, called ways.
the set index now addresses more than one cache line—it points to one cache line in each way.
Instead of one way of 256 lines, the cache has four ways of 64 lines.
The four cache lines with the same set index are said to be in the same set, which is the origin of the name “set index.”
The set of cache lines pointed to by the set index are set associative.
Two sequential blocks from main memory can be stored as cache lines in the same way or two different ways.

	The important thing to note is that the data or code blocks from a specific location in main memory can be stored in any cache line that is a member of a set. The placement of values within a set is exclusive to prevent the same code or data block from simultaneously occupying two cache lines in a set.  Figure 12.7 A 4 KB, four-way set associative cache. The cache has 256 total cache lines, which are separated into four ways, each containing 64 cache lines. The cache line contains four words.			
5. a)	Discuss the cache replacement and write policies that determine the operation of cache with a proper example. Cache Policy-Write Policy—Writethrough: When the processor core writes to memory, the cache controller has two alternatives for its write policy. The controller can write to both the cache and main memory, updating the values in both locations; this approach is known as writethrough. Alternatively, the cache controller can write to cache memory and not update main memory, this is known as writeback or copyback. When the cache controller uses a writethrough policy, it writes to both cache and main memory when there is a cache hit on write, ensuring that the cache and main memory stay coherent at all times. Under this policy, the cache controller performs a write to main memory for each write to cache memory. Because of the write to main memory, a writethrough policy is slower than a writeback policy. Cache Policy-Write Policy-Writeback : When a cache controller uses a writeback policy, it writes to valid cache data memory and not to main memory. Consequently, valid cache lines and main memory may contain different data. The cache line holds the most recent data, and main memory contains older data, which has not been updated. When a cache controller in writeback writes a value to cache memory, it sets the dirty bit true. If the core accesses the cache line at a later time, it knows by the state of the dirty bit that the cache line contains data not in main memory. If the cache controller evicts a dirty cache line, it is automatically written out to main memory. The controller does this to prevent the loss of vital information held in cache memory and not in main memory. Cache Line Replacement Policies: On a cache miss, the cache controller must select a cache line from the available set in cache memory to store the new information from main memory.	[10]	CO5	L2

	The cache line selected for replacement is known as a victim. If the victim contains valid, dirty data, the controller must write the dirty data from the cache memory to main memory before it copies new data into the victim cache line. The process of selecting and replacing a victim cache line is known as eviction. The strategy implemented in a cache controller to select the next victim is called its replacement policy. The replacement policy selects a cache line from the available associative member set. ARM cached cores support two replacement policies, either pseudorandom or round-robin. ■ Round-robin or cyclic replacement simply selects the next cache line in a set to replace. The selection algorithm uses a sequential, incrementing victim counter that increments each time the cache controller allocates a cache line. When the victim counter reaches a maximum value, it is reset to a defined base value. Pseudorandom replacement randomly selects the next cache line in a set to replace. The selection algorithm uses a nonsequential incrementing victim counter. In a pseudorandom replacement algorithm the controller increments the victim counter by randomly. Selecting an increment value and adding this value to the victim counter. When the victim counter reaches a maximum value, it is reset to a defined base value.			
6.a)	Interpret the basic architecture of cache memory.  Figure 12.4 A 4 KB cache consisting of 256 cache lines of four 32-bit words. It has three main parts:a directory store, a data section, and status information. All three parts of the cache memory are present for each cache line. The cache must know where the information stored in a cache line originates from in main memory. It uses a directory store to hold the address identifying where the cache line was copied from main memory. The directory entry is known as a cache-tag. A cache memory must also store the data read from main memory. This information is held in the data section. There are also status bits in cache memory to maintain state information. Two common status bits are the valid bit and dirty bit. A valid bit marks a cache line as active, meaning it contains live data originally taken from main memory and is currently available to the processor core on demand. A dirty bit defines whether or not a cache line contains data that is different from the value it represents in main memory. Basic Operation of a Cache Controller The cache controller is hardware that copies code or data from main memory to cache memory automatically. It performs this task automatically to conceal cache operation from the software it supports. Thus, the same application software can run unaltered on systems with and without a cache.	[6]	CO5	L4

	The cache controller intercepts read and write memory requests before passing them on to the memory controller. It processes a request by dividing the address of the request into three fields, the tag field, the set index field, and the data index field.			
6. b)	Identify the importance of Write buffer in improving the performance of the memory system. A write buffer is a very small, fast FIFO memory buffer that temporarily holds data that the processor would normally write to main memory. In a system without a write buffer, the processor writes directly to main memory. In a system with a write buffer, data is written at high speed to the FIFO and then emptied to slower main memory. The write buffer reduces the processor time taken to write small blocks of sequential data to main memory. The FIFO memory of the write buffer is at the same level in the memory hierarchy as the L1 cache. ● The efficiency of the write buffer depends on the ratio of main memory writes to the number of instructions executed. ● Over a given time interval, if the number of writes to main memory is low or sufficiently spaced between other processing instructions, the write buffer will rarely fill. ● If the write buffer does not fill, the running program continues to execute out of cache memory using registers for processing, cache memory for reads and writes, and the write buffer for holding evicted cache lines while they drain to main memory. ● A write buffer also improves cache performance; the improvement occurs during cache line evictions.	[4]	CO5	L4

Faculty Signature CCI Signature HOD Signature