USN □□□□□□□□□□

CELEBRATING 25 YEARS CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1– July 2023

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sub: | Microcontroller and Embedded Systems | | | | | Sub Code: | 21CS43 | Branch | ISE |
| Date: | 06/07/2023 | Duration: | 90 min’s | Max Marks: | 50 | Sem/Sec: | IV / A, B and C | OBE |

| | | **Answer any FIVE FULL Questions** | MARKS | CO | RBT |
|---|---|---|---|---|---|
| 1 | | Differentiate between i) RISC and CISC Processors ii) Microcontroller and microprocessor. | 10 | CO1 | L2 |
| 2 | | Explain the architecture of a typical embedded device based on ARM core with a neat diagram. | 10 | CO1 | L2 |
| 3 | a | Discuss the following with diagram. i) Von Neuman architecture with cache ii) Harvard architecture with TCM. | 6 | CO1 | L2 |
| | b | Discuss the ARM design philosophy | 4 | CO1 | L2 |
| 4 | | With example, explain the following ARM instructions i) MOV ii) LDR iii) ADD iv) BNE v) STR | 10 | CO2 | L2 |
| 5 | a | Develop an ALP to find factorial of a given number. | 5 | CO2 | L3 |
| | b | Explain the AMBA Bus architecture in detail. | 5 | CO2 | L2 |
| 6 | a | Explain the barrel shifter operation in ARM processor, with a neat diagram. | 5 | CO1 | L2 |
| | b | Develop a program for multiplying two 16 nit numbers. | 5 | CO2 | L3 |

Faculty Signature CCI Signature HOD Signature

USN □□□□□□□□□□

CELEBRATING 25 YEARS CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1– July 2023

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sub: | Microcontroller and Embedded Systems | | | | | Sub Code: | 21CS43 | Branch | ISE |
| Date: | 06/07/2023 | Duration: | 90 min’s | Max Marks: | 50 | Sem/Sec: | IV / A, B and C | OBE |

| | | **Answer any FIVE FULL Questions** | MARKS | CO | RBT |
|---|---|---|---|---|---|
| 1 | | Differentiate between i) RISC and CISC Processors ii) Microcontroller and microprocessor. | 10 | CO1 | L2 |
| 2 | | Explain the architecture of a typical embedded device based on ARM core with a neat diagram. | 10 | CO1 | L2 |
| 3 | a | Discuss the following with diagram. i) Von Neuman architecture with cache ii) Harvard architecture with TCM. | 6 | CO1 | L2 |
| | b | Discuss the ARM design philosophy | 4 | CO1 | L2 |
| 4 | | With example, explain the following ARM instructions i) MOV ii) LDR iii) ADD iv) BNE v) STR | 10 | CO2 | L2 |
| 5 | a | Develop an ALP to find factorial of a given number. | 5 | CO2 | L3 |
| | b | Explain the AMBA Bus architecture in detail. | 5 | CO2 | L2 |
| 6 | a | Explain the barrel shifter operation in ARM processor, with a neat diagram. | 5 | CO1 | L2 |
| | b | Develop a program for multiplying two 16 nit numbers. | 5 | CO2 | L3 |

Faculty Signature CCI Signature HOD Signature

USN □□□□□□□□□□

Internal Assessment Test 1– July 2023

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sub: | Microcontroller and Embedded Systems | | | | Sub Code: | 21CS43 | Branch | ISE |
| Date: | 06/07/2023 | Duration: | 90 min’s | Max Marks: 50 | Sem/Sec: | IV / A, B and C | | OBE |

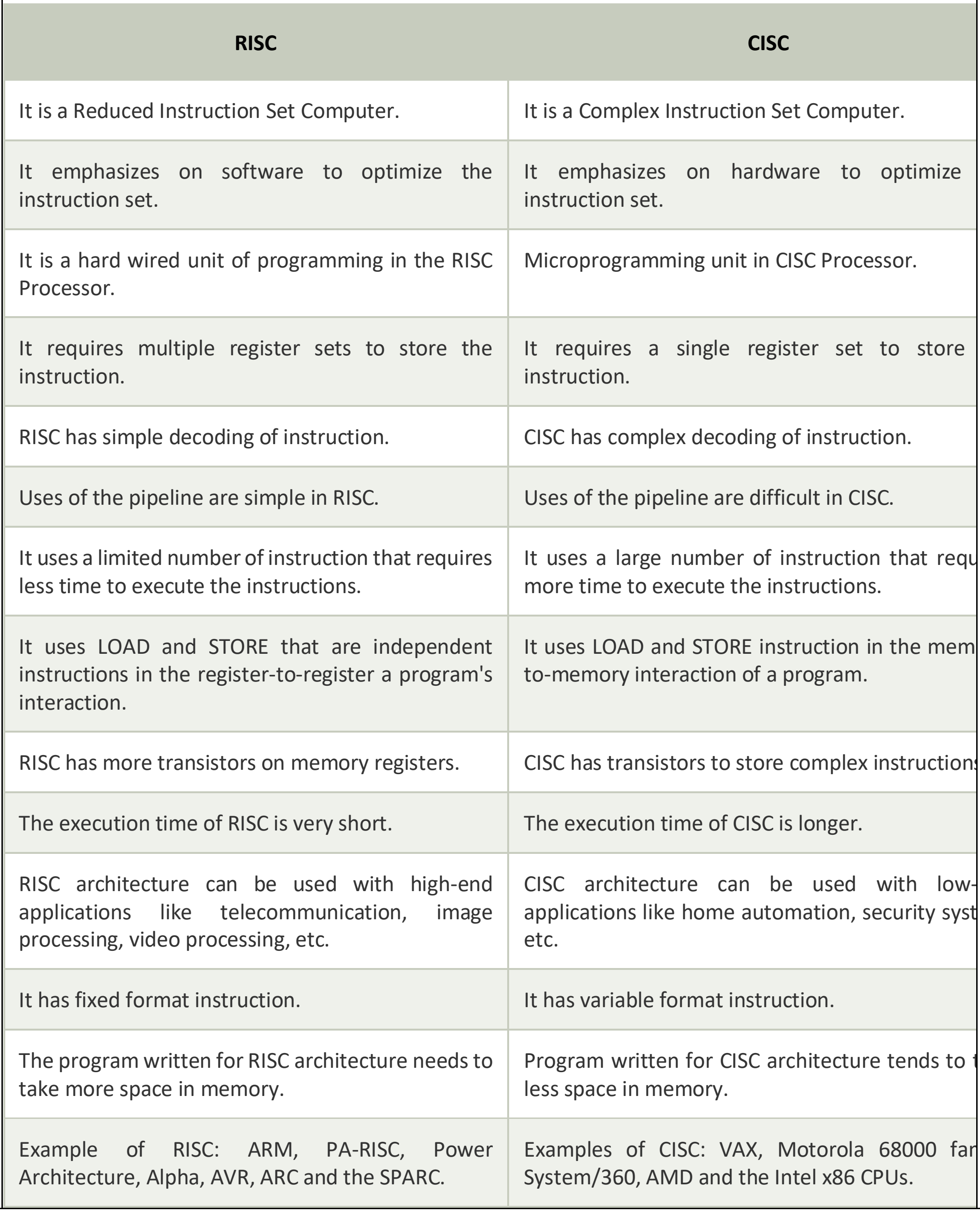
**Answer any FIVE FULL Questions** — MARKS | CO | RBT

1 **Differentiate between i) RISC and CISC Processors ii) Microcontroller and microprocessor.** — 10 | CO1 | L2

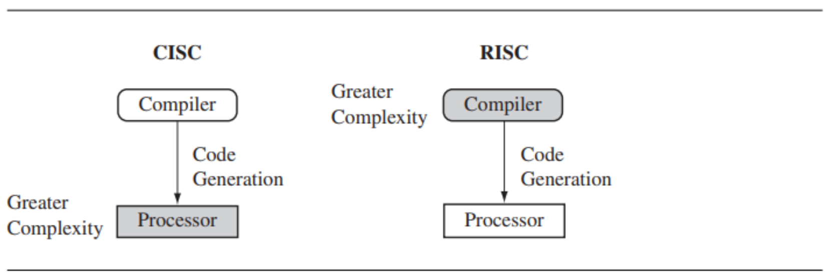
**Scheme: -** (i) Difference between RISC & CISC Processor – 5 Marks

(ii) Difference between Microcontroller & Microprocessor - 5 Marks

**Solution: - (i) Any 5 differences**



| RISC | CISC |
|---|---|
| It is a Reduced Instruction Set Computer. | It is a Complex Instruction Set Computer. |
| It emphasizes on software to optimize the instruction set. | It emphasizes on hardware to optimize instruction set. |
| It is a hard wired unit of programming in the RISC Processor. | Microprogramming unit in CISC Processor. |
| It requires multiple register sets to store the instruction. | It requires a single register set to store instruction. |
| RISC has simple decoding of instruction. | CISC has complex decoding of instruction. |
| Uses of the pipeline are simple in RISC. | Uses of the pipeline are difficult in CISC. |
| It uses a limited number of instruction that requires less time to execute the instructions. | It uses a large number of instruction that requ more time to execute the instructions. |
| It uses LOAD and STORE that are independent instructions in the register-to-register a program's interaction. | It uses LOAD and STORE instruction in the mem to-memory interaction of a program. |
| RISC has more transistors on memory registers. | CISC has transistors to store complex instruction |
| The execution time of RISC is very short. | The execution time of CISC is longer. |
| RISC architecture can be used with high-end applications like telecommunication, image processing, video processing, etc. | CISC architecture can be used with low- applications like home automation, security syst etc. |
| It has fixed format instruction. | It has variable format instruction. |
| The program written for RISC architecture needs to take more space in memory. | Program written for CISC architecture tends to t less space in memory. |
| Example of RISC: ARM, PA-RISC, Power Architecture, Alpha, AVR, ARC and the SPARC. | Examples of CISC: VAX, Motorola 68000 fa System/360, AMD and the Intel x86 CPUs. |



CISC vs. RISC. CISC emphasizes hardware complexity. RISC emphasizes compiler complexity

**Solution: - (ii) Any 5 differences**

| Sl. No | Microprocessor | Microcontroller |
|---|---|---|
| 1. | We need to connect peripherals externally. So it makes circuit bulky. | The presence of peripherals such as RAM, ROM, Inp output, and Timers are In-built. So It is available on single chip. |
| 2. | It increases the overall cost of the system high. | The overall cost of the system is less. |
| 3. | We can connect external memory in ranges of Mbytes and even Gbytes. But speed is less. | The inbuilt finite memory helps to improve the spee of operations. |
| 4. | You can't use it in a compact system. | You can use it in compact systems. |
| 5. | Due to external components, the total power consumption is high. Therefore, it is not ideal for the devices running on stored power like batteries. | As external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries. |
| 6. | Most of the microprocessors do not have power-saving features. | Most of the microcontrollers offer power-saving mo |
| 7. | The microprocessor has a smaller number of registers, so more operations are memory-based. | The microcontroller has more register. Hence the programs are easier to write. |
| 8. | These are based on the von Neumann model where program and data are stored in the same memory module. | These are based on Harvard architecture where program memory and data memory are separate. |
| 9. | It is a central processing unit on a single silicon-based integrated chip. | It is a byproduct of the development of microprocessors with a CPU along with other peripherals. |
| 10 | It uses an external bus to interface to RAM, ROM, and other peripherals. | It uses an internal controlling bus. |
| 11 | Microprocessor-based systems can run at a very high speed because of the technology involved. | Microcontroller based systems run up to 200MHz o more depending on the architecture. |
| 12 | It's useful for general purpose applications that allow you to handle loads of data. | It's useful for application-specific systems. |
| 13 | It's complex and expensive, with a large number of instructions to process. | It's simple and inexpensive with less number of instructions to process. |

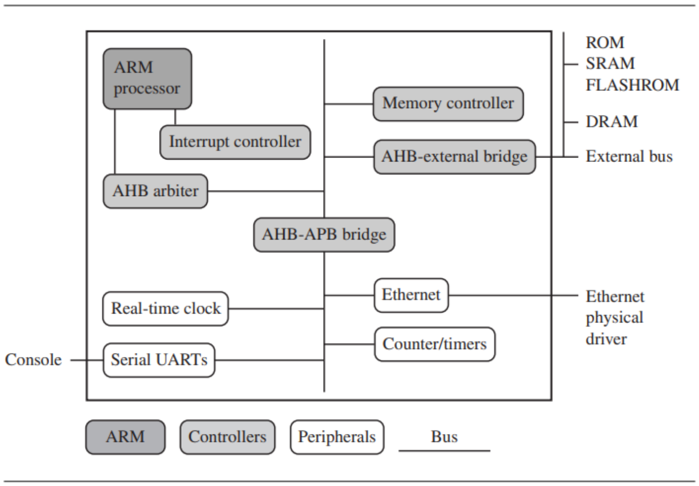
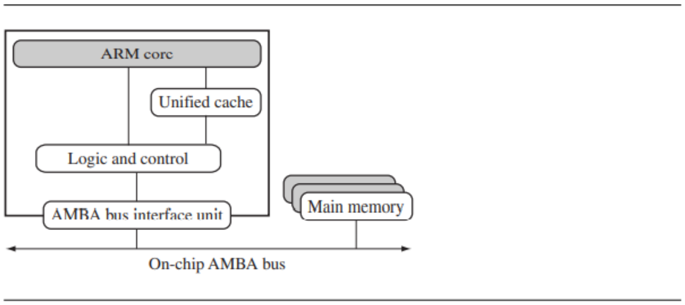
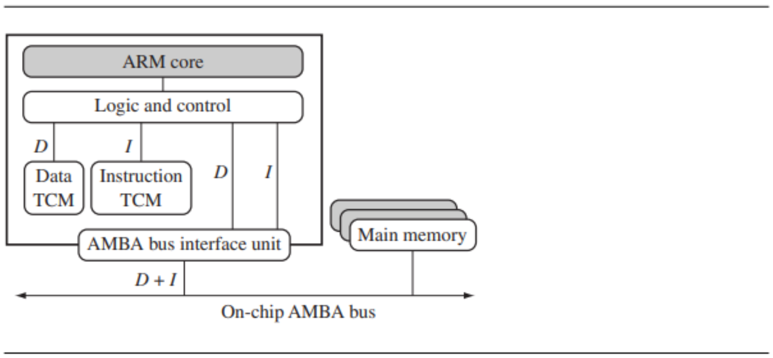
2 Explain the architecture of a typical embedded device based on ARM core with a neat diagram. 10 CO1 L2

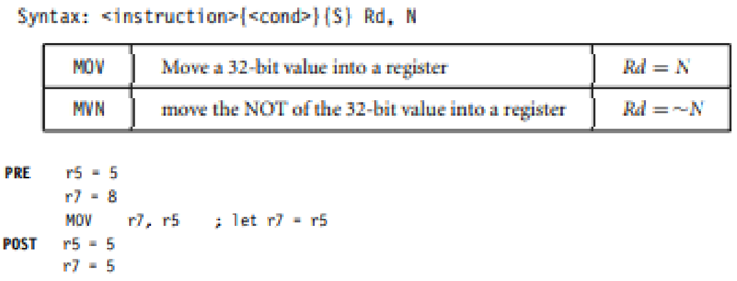
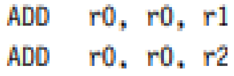
**Scheme: Embedded Device Hardware**

**Solution:**

Embedded systems can control many different devices, from small sensors found on a production line, to the real-time control systems used on a NASA space probe. All these devices use a combination of software and hardware components. Each component is chosen for efficiency and, if applicable, is designed for future extension and expansion

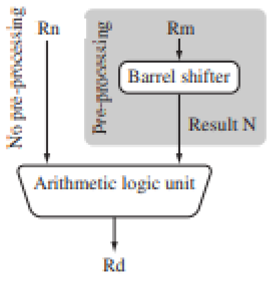
The ARM processor controls the embedded device. Different versions of the ARM processor are available to suit the desired operating characteristics. An ARM processor comprises a core (the execution engine that processes instructions and manipulates data) plus the surrounding components that

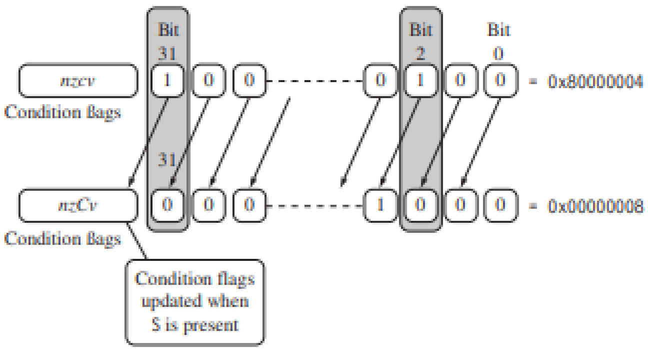
| | | | | | |
|---|---|---|---|---|---|
| | | interface it with a bus. These components can include memory management and caches.<br>■ Controllers coordinate important functional blocks of the system. Two commonly found controllers are interrupt and memory controllers.<br>■ The peripherals provide all the input-output capability external to the chip and are responsible for the uniqueness of the embedded device.<br>■ A bus is used to communicate between different parts of the device<br> ARM processor<br>Interrupt controller<br>AHB arbiter<br>Memory controller<br>AHB-external bridge<br>AHB-APB bridge<br>ROM<br>SRAM<br>FLASHROM<br>DRAM<br>External bus<br>Real-time clock<br>Ethernet<br>Ethernet physical driver<br>Counter/timers<br>Console<br>Serial UARTs<br>ARM<br>Controllers<br>Peripherals<br>Bus | | | |
| 3 | a | Discuss the following with diagram. i) Von Neuman architecture with cache ii) Harvard architecture with TCM.<br>**Scheme: Von Neuman architecture with cache**<br>**Solution:**<br>Von Neuman architecture with cache:<br> ARM core<br>Unified cache<br>Logic and control<br>AMBA bus interface unit<br>Main memory<br>On-chip AMBA bus<br>ARM has two forms of cache. The first is found attached to the Von Neumann–style cores. It combines both data and instruction into a single unified cache, as shown in Figure 2.13. For simplicity, we have called the glue logic that connects the memory system to the AMBA bus logic and control.<br>Harvard architecture with TCM:<br> ARM core<br>Logic and control<br>D<br>I<br>Data TCM<br>Instruction TCM<br>D<br>I<br>AMBA bus interface unit<br>Main memory<br>D + I<br>On-chip AMBA bus | 6 | CO1 | L2 |

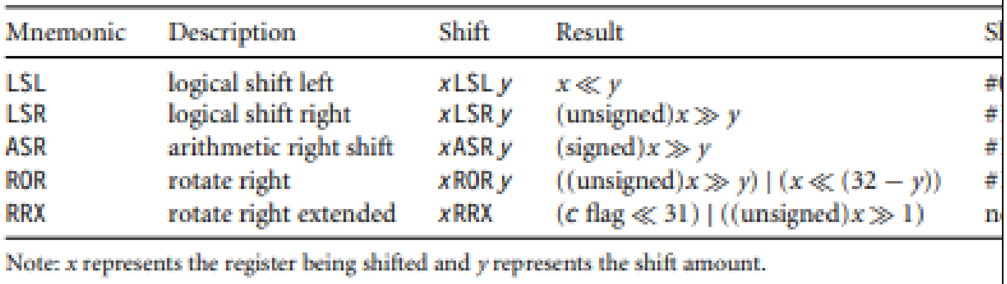
<table>
<tr><td></td><td></td><td>By contrast, the second form, attached to the Harvard-style cores, has separate caches for data and instruction. A cache provides an overall increase in performance but at the expense of predictable execution. But for real-time systems it is paramount that code execution is deterministic— the time taken for loading and storing instructions or data must be predictable. This is achieved using a form of memory called tightly coupled memory (TCM). TCM is fast SRAM located close to the core and guarantees the clock cycles required to fetch instructions or data—critical for real-time algorithms requiring deterministic behavior. TCMs appear as memory in the address map and can be accessed as fast memory.</td><td></td><td></td><td></td></tr>
<tr><td></td><td>b</td><td>Discuss the ARM design philosophy<br><b>Scheme: - To write ARM design philosophy</b><br><b>Solution:</b><br>There are several physical features that have driven the ARM processor design.<br>• Instruction Set for Embedded Systems,Variable cycle execution for certain instructions, Inline barrel shifter leading to more complex instructions, Thumb 16-bit instruction set, Conditional execution, Enhanced instructions<br>• Portable ES require battery power – ARM extended battery operations<br>• Appln’s – Mobile, PDA<br>• High code density<br>• ES - Price sensitive - Use of low-cost memory devices<br>• Reducing the dyeing area (board) – reduces the cost the design and manufacturing<br>• Incorporated Hardware debug technology</td><td>4</td><td>CO1</td><td>L2</td></tr>
<tr><td>4</td><td></td><td>With example, explain the following ARM instructions i) MOV ii) LDR iii) ADD iv) BNE v) STR<br><b>MOV:</b><br><br>Syntax: &lt;instruction&gt;{&lt;cond&gt;}{S} Rd, N<br><table><tr><td>MOV</td><td>Move a 32-bit value into a register</td><td>$Rd = N$</td></tr><tr><td>MVN</td><td>move the NOT of the 32-bit value into a register</td><td>$Rd = \sim N$</td></tr></table><br>PRE r5 = 5<br>r7 = 8<br>MOV r7, r5 ; let r7 = r5<br>POST r5 = 5<br>r7 = 5<br><b>ADD:</b><br><br>ADD r0, r0, r1<br>ADD r0, r0, r2<br><b>LDR:</b><br>Load Register (register) calculates an address from a base register value and an offset register value, loads a word from memory, and writes it to a register.<br>LDR&lt;c&gt;&lt;Rt&gt;[&lt;Rn&gt;,&lt;Rm&gt;]<br><b>BNE:</b><br>Branches when they are unequal.<br>BNE overflow<br><b>STR:</b><br>STR instructions store a register value into memory.<br><b>STR&lt;c&gt; &lt;Rt&gt;,[&lt;Rn&gt;{,/-&lt;imm12&gt;}]</b></td><td>10</td><td>CO2</td><td>L2</td></tr>
<tr><td>5</td><td>a</td><td>Develop an ALP to find factorial of a given number.</td><td>5</td><td>CO2</td><td>L3</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| | | AREA CODE1,CODE,READONLY<br>ENTRY<br>MOV R2,#10 ; number for factorial<br>MOV R3,#1 ; fact=1<br>REPEAT MUL R3,R2,R3 ; FACT =FACT * N<br>SUBS R2,R2,#1 ; N=N-1<br>BNE REPEAT ; REPEAT N != 0<br>LDR R6,=fact ; load the address of fact variable<br>STR R3,[R6] ; store the factorial value in memory<br>STOP B STOP<br>AREA DATA1,DATA,READWRITE<br>fact DCD 0<br>END | | | |
| | b | Explain the AMBA Bus architecture in detail.<br>The Advanced Microcontroller Bus Architecture (AMBA) was introduced in 1996 and has been widely adopted as the on-chip bus architecture used for ARM processors. The first AMBA buses introduced were the ARM System Bus (ASB) and the ARM Peripheral Bus (APB). Later ARM introduced another bus design, called the ARM High Performance Bus (AHB). Using AMBA, peripheral designers can reuse the same design on multiple projects. Because there are a large number of peripherals developed with an AMBA interface, hardware designers have a wide choice of tested and proven peripherals for use in a device. A peripheral can simply be bolted onto the on-chip bus without having to redesign an interface for each different processor architecture. This plug-and-play interface for hardware developers improves availability and time to market.<br>AHB provides higher data throughput than ASB because it is based on a centralized multiplexed bus scheme rather than the ASB bidirectional bus design. This change allows the AHB bus to run at higher clock speeds and to be the first ARM bus to support widths of 64 and 128 bits. ARM has introduced two variations on the AHB bus: Multi-layer AHB and AHB-Lite. In contrast to the original AHB, which allows a single bus master to be active on the bus at any time, the Multi-layer AHB bus allows multiple active bus masters. AHB-Lite is a subset of the AHB bus and it is limited to a single bus master. This bus was developed for designs that do not require the full features of the standard AHB bus. AHB and Multi-layer AHB support the same protocol for master and slave but have different interconnects. The new interconnects in Multi-layer AHB are good for systems with multiple processors. They permit operations to occur in parallel and allow for higher throughput rates.<br>The example device shown in Figure 1.2 has three buses: an AHB bus for the high performance peripherals, an APB bus for the slower peripherals, and a third bus for external peripherals, proprietary to this device. This external bus requires a specialized bridge to connect with the AHB bus. | 5 | CO2 | L2 |
| 6 | a | Explain the barrel shifter operation in ARM processor, with a neat diagram.<br>A unique and powerful feature of the ARM processor is the ability to shift the 32-bit binary pattern in one of the source registers left or right by a specific number of positions before it enters the ALU. This shift increases the power and flexibility of many data processing operations. There are data processing instructions that do not use the barrel shift, for example, the MUL (multiply), CLZ (count leading zeros), and QADD (signed saturated 32-bit add) | 5 | CO1 | L1 |

instructions. Pre-processing or shift occurs within the cycle time of the instruction. This is particularly useful for loading constants into a register and achieving fast multiplies or division by a power of 2.







| Mnemonic | Description | Shift | Result | Sl |
|---|---|---|---|---|
| LSL | logical shift left | *x* LSL *y* | $x \ll y$ | #( |
| LSR | logical shift right | *x* LSR *y* | $(\text{unsigned})x \gg y$ | # |
| ASR | arithmetic right shift | *x* ASR *y* | $(\text{signed})x \gg y$ | # |
| ROR | rotate right | *x* ROR *y* | $((\text{unsigned})x \gg y) \mid (x \ll (32 - y))$ | # |
| RRX | rotate right extended | *x* RRX | $(c \text{ flag} \ll 31) \mid ((\text{unsigned})x \gg 1)$ | n |

Note: *x* represents the register being shifted and *y* represents the shift amount.

b Develop a program for multiplying two 16 nit numbers. 5 CO2 L3

```
AREA CODE1,CODE,READONLY
ENTRY
LDR R0,=N1 ; Load the address of N1
LDR R1,=N2 ; Load the address of N2
LDRH R2,[R0] ; Load the value of N1
LDRH R3,[R1] ; Load the value of N1
MUL R4, R2,R3 ; multiply N1 and N2
LDR R6,=RESULT ; load the address of RESULT variable
STR R4,[R6]
; store the RESULT OF MULTIPLICATION value in memory
STOP B STOP
N1 DCW 0X1234
N2 DCW 0X5678
AREA DATA1,DATA,READWRITE
RESULT DCD 0
END
```

Faculty Signature CCI Signature HOD Signature