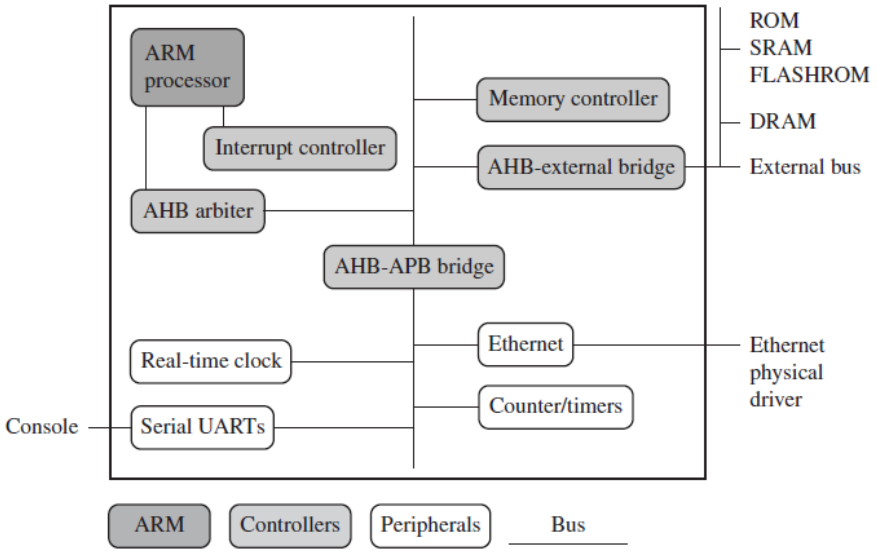
Subject: Microcontroller and Embedded Systems **Code: 21CS43** **Faculty: LAKSHMI R**

Qn. No.		Questions	Marks
1	a)	**With a neat diagram, explain the four main hardware components of an ARM based embedded device.** **Solution:**  **Figure: An ARM-based Embedded Device, a Microcontroller** **Diagram: 3M** We can separate the device into ***four main hardware components***: 1. *The ARM processor* controls the embedded device. An ARM processor comprises a core (the execution engine that processes instructions and manipulates data) plus the surrounding components (memory and cache) that interface it with a bus. 2. *Controllers coordinate important functional blocks of* the system. Two commonly found controllers are interrupt and memory controllers. 3. The *peripherals* provide all the input-output capability external to the chip and are responsible for the uniqueness of the embedded device. 4. A *bus* is used to communicate between different parts of the device. • Advanced System Bus (ASB), Advanced Peripheral Bus(APB). Advanced High Performance Bus (AHB). • Reusable design, run at higher clock speeds. • ARM has introduced two variations on the AHB bus: Multi-layer AHB and AHB-Lite. • 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. **Brief Explanation: 5M**	08

b) Discuss ARM design philosophy. — 04

Solution:

THE ARM DESIGN PHYLOSOPHY:

- **Battery power**: The ARM processor has been specially designed to be small to reduce power consumption and extend battery operation
- **High code density:**useful for applications that have limited on-board memory, such as mobile phones and mass storage devices.
- **Low-cost:**Embedded systems are price sensitive - essential for high-volume applications like digital cameras.
- **Hardware debug technology** within the processor so that software engineers can view what is happening while the processor is executing code. With greater visibility, software engineers can resolve issues faster.

1 M for each point

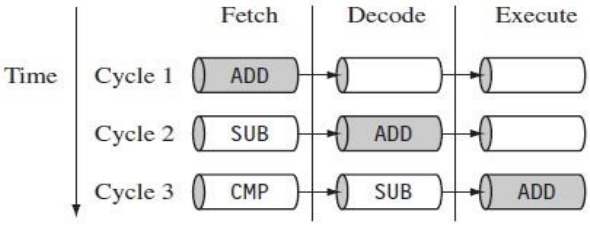
2 a) Define pipeline. Illustrate ARM7 pipeline with suitable example. — 08

Solution:

- A pipeline is the mechanism in a RISC processor, which is used to execute instructions.
- Pipeline speeds up execution by fetching the next instruction while other instructions are being decoded and executed. **Definition: 2M**

ARM7 pipeline

- The above Figure shows a three-stage pipeline:
- Fetch loads an instruction from memory.
- Decode identifies the instruction to be executed.
- Execute processes the instruction and writes the result back to a register.



(Any example can be considered for explanation)

- The Figure shows a sequence of three instructions being fetched, decoded, and executed by the processor.
- The three instructions are placed into the pipeline sequentially.
- In the first cycle, the core fetches the ADD instruction from memory.
- In the second cycle, the core fetches the SUB instruction and decodes the ADD instruction.
- In the third cycle, both the SUB and ADD instructions are moved along the pipeline. The ADD instruction is executed, the SUB instruction is decoded, and the CMP instruction is fetched.
- This procedure is called filling the pipeline.

About the stages: 3M
Explanation of these stages in pipeline: 3M

b) Compare microprocessors and microcontrollers. — 04

Solution: (Any four of the following)

Microprocessor	Microcontroller
A silicon chip representing a central processing unit (CPU), which is capable of performing arithmetic as well as logical operations	A microcontroller is a highly integrated chip that contains a CPU, scratchpad RAM, special and general purpose register arrays, on chip ROM/

according to a pre-defined set of instructions	FLASH memory for program storage, timer and interrupt control units and dedicated I/O ports		
It is a dependent unit. It requires the combination of other chips like timers, program and data memory chips, interrupt controllers, etc. for functioning	It is a self-contained unit and it doesn't require external interrupt controller, timer, UART, etc. for its functioning		
General purpose in design and operation	Mostly application-oriented or domain-specific		
Doesn't contain a built in I/O port. - external programmable peripheral interface chips like 8255	multiple built-in I/O ports		
Targeted for high end market where performance is important	Targeted for embedded market where performance is not so critical		
Limited power saving options	Includes lot of power saving features		

3 a) With a neat diagram explain ARM core data flow model clearly indicating the significance of barrel shifter and MAC circuit. 08

Solution:

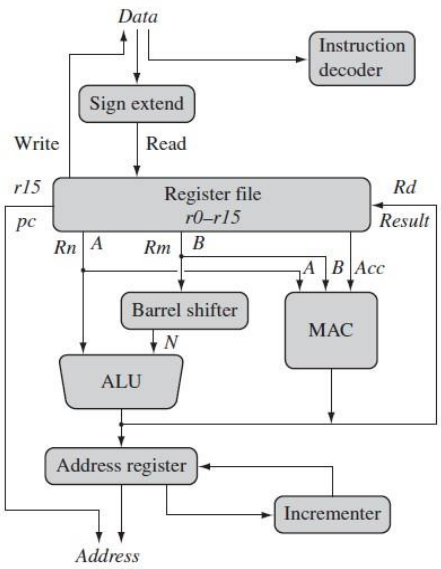
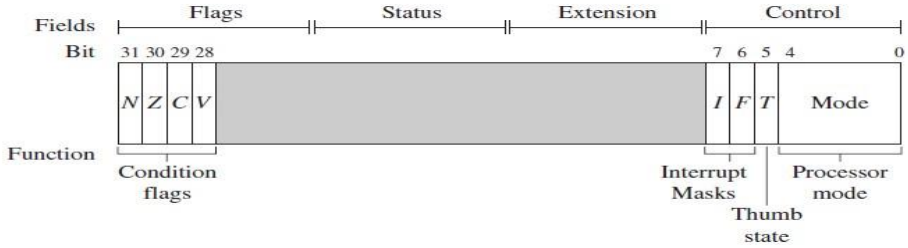


Diagram – 4M

Brief one line introduction to all blocks – 4M

b) With the help of basic layout diagram, explain the current program register. 04

Solution:



The Jazelle J and Thumb T bits in the cpsr reflect the state of the processor.

- J = 0, T=0 => ARM state
- T=1 => Thumb state

Mode:

- six privileged modes (abort, fast interrupt request, interrupt request, supervisor, system, and undefined), one non-privileged mode (user).

Interrupt Masks:

- Interrupt masks are used to stop specific interrupt requests from interrupting the processor.
- The I bit masks IRQ when set to binary 1; and similarly, the F bit masks FIQ when set to

binary

Condition flags and a short description on what causes them to be set.

Flag	Flag Name	Set When
V	oVerflow	the result causes a signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero
N	Negative	bit 31 of the result is a binary 1

2 M for diagram
2M for explanation for each field

4 a) Explain the different operating modes of ARM processor. — 08

Solution:

- Six privileged modes (abort, fast interrupt request, interrupt request, supervisor, system, and undefined)
- The processor enters abort mode when there is a failed attempt to access memory.
- Fast interrupt request and interrupt request modes correspond to the two interrupt levels available on the ARM processor.
- Supervisor mode is the mode that the processor is in after reset and is generally the mode that an operating system kernel operates in.
- System mode is a special version of user mode that allows full read-write access to the cpsr.
- Undefined mode is used when the processor encounters an instruction that is undefined or not supported by the implementation.
- One non-privileged mode (user):User mode is used for programs and applications.

Mentioning all 7 modes: 2M
Brief explanation of all: 6M

b) Discuss the design rules of RISC architecture. — 04

Solution:

THE RISC DESIGN PHYLOSOPHY:

- **Instructions**—RISC processors have a reduced number of instruction classes and can each execute in a single cycle.
- **Pipelines**—The processing of instructions is broken down into smaller units that can be executed in parallel by pipelines. Ideally the pipeline advances by one step on each cycle for maximum throughput. Instructions can be decoded in one pipeline stage.
- **Registers**—RISC machines have a large general-purpose register set. Any register can contain either data or an address. Registers act as the fast local memory store for all data processing operations.
- **Load-store architecture -** the processor operates on data held in registers. Separate load and store instructions transfer data between the register bank and external memory.

1 M for each point

5 a) Explain Embedded System Software with a neat diagram. — 08

Solution:

EMBEDDED SYSTEM SOFTWARE:

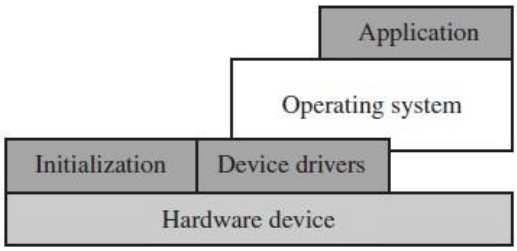


Figure: Software Abstraction Layers Executing on Hardware

- The *initialization code* is the first code executed on the board and is specific to a particular target or group of targets. It sets up the minimum parts of the board before handing control

over to the operating system.

- The *operating system* provides an infrastructure to control applications and manage hardware system resources.
- The *device drivers* provide a consistent software interface to the peripherals on the hardware device.
- An *application* performs one of the tasks required for a device.
  - For example, a mobile phone might have a diary application.
  - There may be multiple applications running on the same device, controlled by the operating system.

Diagram: 1M
Mentioning of above points in brief: 4M
Brief Explanation: 3M

b) Mention any four features of Thumb state. — 04

Solution:

Instruction size	16-bit
Core instructions	30
Conditional execution[a]	only branch instructions
Data processing instructions	separate barrel shifter and ALU instructions
Program status register	no direct access
Register usage	8 general-purpose registers +7 high registers +*pc*

6 a) Define conditional execution. Mention any six conditional mnemonics with their appropriate meaning and condition flags. — 08

Solution:

Conditional execution controls whether or not the core will execute an instruction. Most instructions have a condition attribute that determines if the core will execute it based on the setting of the condition flags.

Definition: 2M

Condition Mnemonics: (Any six among the following to be mentioned) ----- **6M**

Mnemonic	Name	Condition flags
EQ	equal	*Z*
NE	not equal	*z*
CS HS	carry set/unsigned higher or same	*C*
CC LO	carry clear/unsigned lower	*c*
MI	minus/negative	*N*
PL	plus/positive or zero	*n*
VS	overflow	*V*
VC	no overflow	*v*
HI	unsigned higher	*zC*
LS	unsigned lower or same	*Z* or *c*
GE	signed greater than or equal	*NV* or *nv*
LT	signed less than	*Nv* or *nV*
GT	signed greater than	*NzV* or *nzv*
LE	signed less than or equal	*Z* or *Nv* or *nV*
AL	always (unconditional)	ignored

b) Explain any two pipeline characteristics. — 04

Solution: (Any two must be explained)

- The ARM pipeline will process an instruction only after it passes completely through the execute stage.
- The execution of branch instruction or branching by the direct modification of the PC causes the ARM core to flush its pipeline.

- ARM10 uses branch prediction – reduces the effect of pipeline flush
    - Predicting the possible branches
    - Loading the new branch address prior to the execution of the instruction.
- An instruction in the execute stage will complete even though an interrupt has been raised.

7 a) List and explain all arithmetic instructions of ARM7. (08)

Solution:

```
Syntax: <instruction>{<cond>}{S} Rd, Rn, N
```

ADC	add two 32-bit values and carry	$Rd = Rn + N +$ carry
ADD	add two 32-bit values	$Rd = Rn + N$
RSB	reverse subtract of two 32-bit values	$Rd = N - Rn$
RSC	reverse subtract with carry of two 32-bit values	$Rd = N - Rn -$!(carry flag)
SBC	subtract with carry of two 32-bit values	$Rd = Rn - N -$!(carry flag)
SUB	subtract two 32-bit values	$Rd = Rn - N$

Syntax and description: 5M
Examples: 3M

b) Illustrate the differences between MOV and MVN instructions with relevant examples. (04)

Solution:

```
Syntax: <instruction>{<cond>}{S} Rd, N
```

MOV	Move a 32-bit value into a register	$Rd = N$
MVN	move the NOT of the 32-bit value into a register	$Rd = \sim N$

Syntax and description: 2M
Examples: 2M

8 a) List and give description for the different indexing methods with respect to load and store instructions. (04)

Solution:

Index methods.

Index method	Data	Base address register	Example
Preindex with writeback	*mem[base + offset]*	*base + offset*	`LDR r0,[r1,#4]!`
Preindex	*mem[base + offset]*	*not updated*	`LDR r0,[r1,#4]`
Postindex	*mem[base]*	*base + offset*	`LDR r0,[r1],#4`

Note: ! indicates that the instruction writes the calculated address back to the base address register.

b) Determine the effective address and contents of base address register for the following instructions considering R0 = 0xA0000 and R1 = 0x20. Also, indicate the indexing method used in each of the instructions. (04)

i. LDR R2, [R0, R1, LSL #2]!

Indexing: Pre-index with writeback

Effective Address = Base + offset = 0xA0000 + (0x20 * 4) = 0xA0000 + 0x80 = **0xA0080**

Base = Base + offset = **0xA0080**

ii. STR R3, [R0], #-4

Indexing: Post-index

Effective Address = Base = **0xA0000**

Base = Base + offset = 0xA0000 + (-4)= **0x9FFFC**

c) **Determine the post condition for the following preconditions**

```
R5=0X65, R7=0XA994567B, R8=0XFFFFFF,R9=R10= 0
RSBS R9, R7, R8
SBC R10, R5, #4
```

POST: **RSBS: R9 = R8 – R7** = R8 + 2's complement (R7)

```
                    R8: 0000 0000 1111 1111 1111 1111 1111 1111
+ 2's complement (R7): 0101 0110 0110 1011 1010 1001 1000 0101
                     = 0101 0111 0110 1011 1010 1001 1000 0100
= 5  7 6  B  A  9  8  4 (HEX)
   CARRY IS NOT GENERATED FOR THIS SUBTRACTION: CARRY = 0
SBC: R5 – 4 - !(CARRY)
                  R5 : 0000 0000 0000 0000 0000 0000 0110 0101
+ 2's complement (4): 1111 1111 1111 1111 1111 1111 1111 1100
+ 2's complement (1): 1111 1111 1111 1111 1111 1111 1111 1111
                   =  0000 0000 0000 0000 0000 0000 0110 0000
                =   0      0    0    0    0    0    6    0 (HEX)
```

R5 = 0X65, R7 = 0XA994567B, R8 = 0XFFFFFF, R9 = 0x 576BA984, R10 = 0x00000060

(Students have to show complete calculations to get 3M)

04

9 **a)** **List and explain barrel shifter instructions with syntax and relevant examples.**

Solution:

Mnemonic	Description
LSL	logical shift left
LSR	logical shift right
ASR	arithmetic right shift
ROR	rotate right
RRX	rotate right extended

Mnemonic with description: 3M

Examples: 5M

08

b) **Write ARM assembly program to compute factorial of a number.**

Solution:

```
        AREA FACT, CODE, READONLY
        MOV R0 ,  #9
        MOV R1,  #1
REPEAT  MUL R2, R1, R0
        MOV R1, R2
        SUBS R0, #1
        BNE REPEAT
STOP B STOP
        END
```

04

10 **a)** **Compare B and BL instructions.**

Solution:

B	branch	*pc* = *label*
BL	branch with link	*pc* = *label* *lr* = address of the next instruction after the BL

02

b) **Consider the following pre conditions:**

```
R3 = 0x789ABCDE     Mem32[0x000C0000] = 0x11111111
R4 = 0x0            Mem32[0x000C0004] = 0x22222222
R0 = 0x000C0000     Mem32[0x000C0008] = 0x33333333
R1 = 0x04           Mem32[0x000C000C] = 0x44444444
                    Mem32[0x000C0010] = 0x55555555
```

Determine the post conditions for the following instructions. (Note: Indicate the updated

06

		memory contents) **i. LDR R4, [R0, R1, LSL #1]!** **Solution:** EA = Base + offset = 0x000C0000 + (4 * 2) = 0x000C0008 R4 ← Mem32[0x000C0008] R0 = base + offset = 0x000C0008 **POST: R4 = 0x33333333, R0 = 0x000C0008, R1 = 0x00000004** **Calculation: 3M** **ii. STR R3, [R0, R1, LSL #2]!** **Solution:** EA = Base + offset = 0x000C0000 + (0x04 * 0x04) = 0x000C0000 + 0x10 R3 → Mem32[0x000C0010] R0 = base + offset = 0x000C0010 **POST: R3 = 0x789ABCDE, R0 = 0x000C0010, R1 = 0x00000004** **Mem32[0x000C0010] = 0x789ABCDE** **Calculation: 3M**	
	c)	**Pre: R1 = 0x7846, R2 = 0x1111, R3 = 0xFFFFFFFF, R4 = 0x44** **Instruction: UMLAL R1, R2, R3, R4. Determine the post condition.** **Solution:** Effect: [RdHi RdLo] = [RdHi RdLo] + (Rm * Rs) = [R2 R1] = [R2 R1] + (R3 * R4) R3 * R4 = 0XFFFFFFFF * 0x44 3 F F F F F F F F C + 3 F F F F F F F F C = 4 3 F F F F F F F B C To perform [R2 R1] = [R2 R1] + (R3 * R4) : [R2 R1] = 0 0 0 0 1 1 1 1 0 0 0 0 7 8 4 6 R3 * R4 = 0 0 0 0 0 0 4 3 F F F F F F F B C SUM: 0 0 0 0 1 1 5 5 0 0 0 0 7 8 0 2 Higher 32-bits will be moved to RdHi = R2 and lower 32-bits will be moved to RdLo = R1 Therefore, R1 = 0x00007802 and R2 = 0x00001155 **POST**: R1 0x00007802 R2 0x00001155 R3 0xFFFFFFFF R4 0x00000044	04