UNIT I:

BLOOM'S LEVEL 2: UNDERSTAND

1. DIFFRENTIATE BETWEEN MICROPROCESSORS AND MICROCONTROLLERS WITH A NEAT BLOCK DIAGRAM.

2. LIST AND EXPLAIN THE FOUR MAJOR DESIGN RULES OF RISC PHILOSOPHY.

3. DIFFERTIATE BETWEEN RISC AND CISC PROCESSORS.

4. LIST AND EXPLAIN IN DETAIL THE ARM DESIGN PHILOSOPHY.

5. JUSTIFY WHY ARM INSTRUCTION SET IS SUITABLE FOR EMBEDDED APPLICATIONS.

6. WITH A NEAT BLOCK DIAGRAM OF AN ARM-BASED EMBEDDED DEVICE, EXPLAIN THE FOLLOWING:

- ARM PROCESSOR
- CONTROLLERS
- PERIPHERALS
- BUS

7. WRITE A NOTE ON THE FOLLOWING:

- ARM BUS TECHNOLOGY
- AMBA BUS PROTOCOL
- MEMORY
- PERIPHERALS

8. WITH A NEAT BLOCK DIAGRAM EXPLAIN THE ARM CORE DATA FLOW MODEL.

9. LIST OUT THE VARIOUS REGISTERS OF ARM 7. COMMENT ON ITS WIDTH, AND SPECIAL PURPOSE OF REGISTERS R13, R14 AND R15.

10. DRAW THE NEAT BLOCK DIAGRAM OF CPSR AND COMMENT ON THE SIGNIFICANCE OF **N, Z, C AND V** FLAGS?

11. LIST THE VARIOUS MODES OF OPERATION OF ARM 7.

12. DEFINE PIPELINE. HOW MANY STAGES OF PPELINE IS AVAILABLE FOR ARM7. ILLUSTRATE THE PIPELINE OPERATION FOR THE FOLLOWING INSTRUCTIONS:

a. ADD R0,R1,R2

b. AND R3,R4,R5

c. SUB R6,R7,R8

BLOOM'S LEVEL 3: APPLY

1. WHICH OF THE FOLLOWING STATEMENTS ARE TRUE WITH RESPECT TO ARM 7 ARCHITECTURE.

a. EACH PROCESSOR MODE IS EITHER PREVILEGED OR NONPREVILEGED.

b. PREVILEGED MODE ALLOWS FULL READ WRITE ACCESS TO THE CPSR.

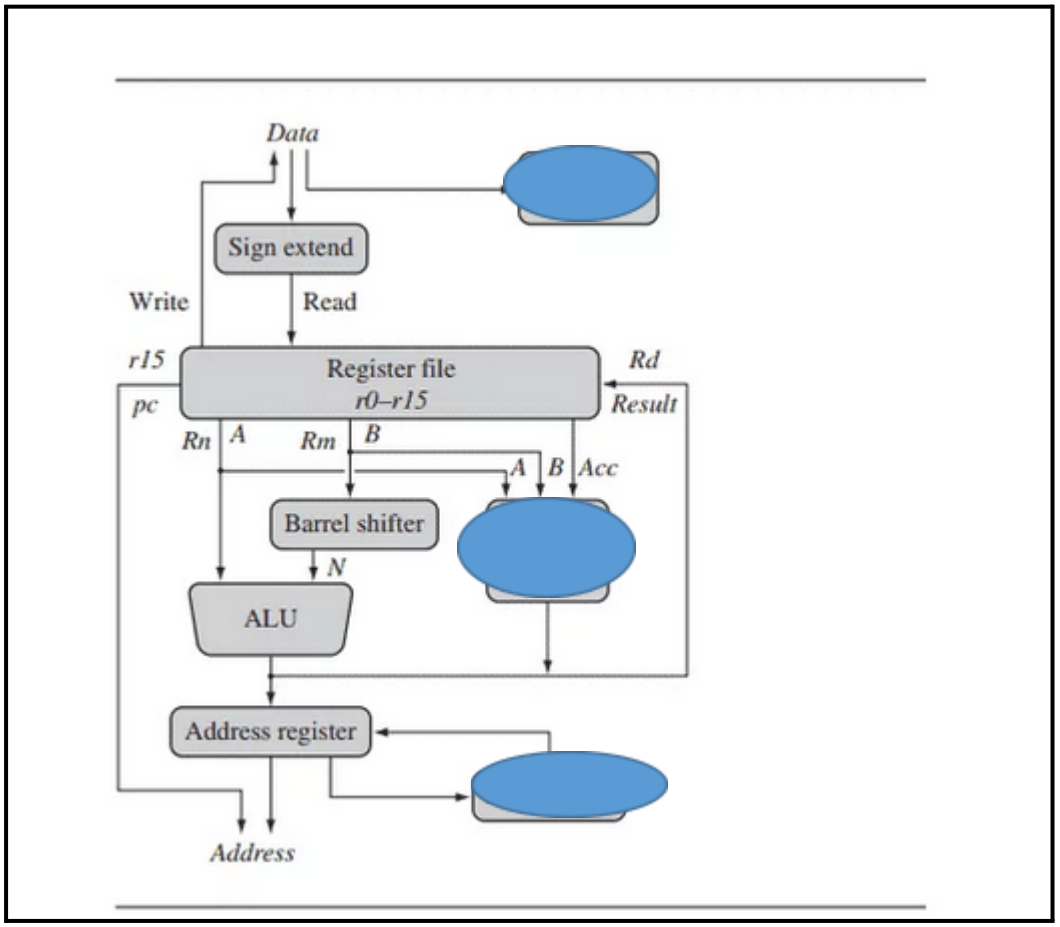
c. THE NEGATIVE FLAG ‘N’ IS SET WHEN BIT 31 OF THE RESULT IS BINARY 1.

d. THE ZERO FLAG ‘Z’ IS USED TO INDICATE EQUALITY.

e. THE CARRY FLAG ‘C’ IS SET WHEN THE RESULT CAUSES AN UNSIGNED CARRY.

f. THE OVERFLOW FLAG ‘V’ IS SET WHEN THE RESULT CAUSES SIGNED OVERFLOW.

BLOOM’S LEVEL 4: ANALYZE: ANALYZE THE ARM CORE DATAFLOW MODEL SHOWN IN FIGURE BELOW AND IDENTIFY THE MASKED BLOCKS AND THEIR SIGNIFICANCE.



UNIT 2:

BLOOM'S LEVEL 2: UNDERSTAND

1. LIST AND EXPLAIN THE VARIOUS DATA TRANSFER INSTRUCTIONS OF ARM7 WITH PROPER SYNTAX AND AN EXAMPLE.

2. WITH A NEAT BLOCK DIAGRAM EXPLAIN THE SIGNIFICANCE OF BARRE SHIFTER AND ALU.

3. LIST AND EXPLAIN THE BASIC 'C' DATA TYPES.

4. LIST AND EXPLAIN THE FOLLOWING INSTRUCTIONS OF ARM7 WITH PROPER SYNTAX AND AN EXAMPLE FOR EACH.

a. SHIFT INSTRUCTIONS
b. ROTATE INSTRUCTIONS
c. ARITHMETIC INSTRUCTIONS
d. LOGICAL INSTRUCTIONS
e. COMPARISON INSTRUCTIONS
f. MULTIPLY INSTRUCTIONS
g. BRANCH INSTRUCTIONS
h. LOAD STORE INSTRUCTIONS
i. SWAP INSTRUCTION
j. PROGRAM STATUS REGISTER INSTRUCTIONS

BLOOM'S LEVEL 3: APPLY

1. DEVELOP AN ASSEMBLY LANGUAGE PROGRAM (ALP) TO PERFORM BLOCK DATA TRANSFER.

2. DEVELOP AN ALP TO GENERATE THE SERIES: 5, 10,15,20,25. HINT: USE MLA INSTRUCTION.

3. DEVELOP AN ALP TO COMPUTE THE FACTORIAL OF A GIVEN NUMBER AND STORE THE RESULT IN RAM LOCATION.

4. DEVELOP AN ALP FIND THE LARGEST NUMBER IN AN ARRAY AND STORE IT IN RAM LOCATION.

5. DEVELOP AN ALP TO ILLUSTRATE THE SIGNIFICANCE OF LOGICAL OPERATIONS.

6. DEVELOP AN ALP TO ILLUSTRATE THE WORKING OF SHIFT AND ROTATE INSTRUCTIONS.

7. DEVELOP AN ALP TO ILLUSTRATE THE WORKING OF SWAP INSTRUCTION.

8. DEVELOP AN ALP TO ILLUSTRATE THE WORKING OF LOAD STORE INSTRUCTIONS

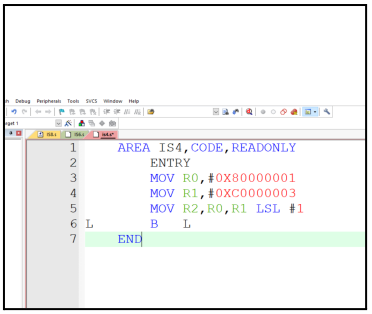
9. DEVELOP AN ALP TO ILLUSTRATE THE WORKING OF PROGRAM STATUS REGISTER INSTRUCTIONS

BLOOM'S LEVEL 4: ANALYZE:

1. ANALYZE THE GIVEN PIECE OF CODE AND ANSWER THE FOLLOWING:

a. WHAT IS THE CONTENT OF R0,R1 AND R2.

b. COMMENT ON THE STATUS OF NZCV FLAGS AFTER EXECUTING THE LAST INSTRUCTION.



```
1       AREA IS4,CODE,READONLY
2           ENTRY
3           MOV R0,#0X80000001
4           MOV R1,#0XC0000003
5           MOV R2,R0,R1 LSL #1
6   L       B   L
7       END
```

2. ANALYZE THE GIVEN PIECE OF CODES ('C' CODE AND COMPILER OUTPUT) AND ANSWER THE FOLLOWING:

```
int checksum_v1(int *data)
{
  char i;
  int sum=0;

  for (i=0; i<64; i++)
  {
    sum += data[i];
  }
  return sum;
}
```

```
checksum_v1
        MOV     r2,r0               ; r2 = data
        MOV     r0,#0               ; sum = 0
        MOV     r1,#0               ; i = 0
checksum_v1_loop
        LDR     r3,[r2,r1,LSL #2]   ; r3 = data[i]
        ADD     r1,r1,#1            ; r1 = i+1
        AND     r1,r1,#0xff         ; i = (char)r1
        CMP     r1,#0x40            ; compare i, 64
        ADD     r0,r3,r0            ; sum += r3
        BCC     checksum_v1_loop    ; if (i<64) loop
        MOV     pc,r14              ; return sum
```

- WHAT IS THE DRAWBACK OF USING CHAR DATA TYPE FOR DECLARING THE LOCAL VARIABLES IN ARM7 'C' PROGRAM?
- IN THE COMPILER OUTPUT HOW CAN WE AVOID THE INSTRUCTION *AND R1,R1,#0XFF*
- WHAT IS THE USE OF BCC INSTRUCTION?
- WHY PC IS UPDATED WITH R14 CONTENT? CAN WE REPLACE R14 BY ANY OTHER REGISTER?

3. ANALYZE THE GIVEN PIECE OF CODE AND ANSWER THE FOLLOWING:

- WHICH DATA TYPE IS USED TO DECLARE THE LOCAL VARIABLE?
- WHAT IS THE MODIFICATION THAT IS REQUIRED IN THE COMPILER OUTPUT IF THE VARIABLE SUM IS 16-BIT.

```
checksum_v2
        MOV     r2,r0               ; r2 = data
        MOV     r0,#0               ; sum = 0
        MOV     r1,#0               ; i = 0
checksum_v2_loop
        LDR     r3,[r2,r1,LSL #2]   ; r3 = data[i]
        ADD     r1,r1,#1            ; r1++
        CMP     r1,#0x40            ; compare i, 64
        ADD     r0,r3,r0            ; sum += r3
        BCC     checksum_v2_loop    ; if (i<64) goto loop
        MOV     pc,r14              ; return sum
```

4. ANALYZE THE GIVEN PIECE OF CODES (‘C’ CODE AND COMPILER OUTPUT) AND ANSWER THE FOLLOWING:

```
short checksum_v3(short *data)
{
 unsigned int i;
 short sum=0;

 for (i=0; i<64; i++)
 {
   sum = (short)(sum + data[i]);









    }
    return sum;
  }
```

```
checksum_v3
        MOV     r2,r0               ; r2 = data
        MOV     r0,#0               ; sum = 0
        MOV     r1,#0               ; i = 0
checksum_v3_loop
        ADD     r3,r2,r1,LSL #1     ; r3 = &data[i]
        LDRH    r3,[r3,#0]          ; r3 = data[i]
        ADD     r1,r1,#1            ; i++
        CMP     r1,#0x40            ; compare i, 64
        ADD     r0,r3,r0            ; r0 = sum + r3
        MOV     r0,r0,LSL #16
        MOV     r0,r0,ASR #16       ; sum = (short)r0
        BCC     checksum_v3_loop    ; if (i<64) goto loop
        MOV     pc,r14              ; return sum
```

- HOW CAN WE REDUCE THESE INSTRUCTIONS IN THE COMPILER OUTPUT?
    - `ADD r3,r2,r1,LSL #1`
    - `MOV r0,r0,LSL #16`
    - `MOV r0,r0,ASR #16`
- REWRITE THE ‘C’ CODE TO REDUCE THESE INSTRUCTIONS.

UNIT 3:

LEVEL 2:

1. LIST AND EXPLAIN THE VARIOUS REGISTERS OF ARM 7 USED FOR CONFIGURING PORTS AS GPIO, INPUT/OUTPUT AND SET/CLEAR .
2. WHAT VALUE HAS TO BE LOADED INTO THE REGISTERS
    - TO CONFIGURE PORT 0 (P0.0-P0.15) PINS AS INPUT?
    - TO CONFIGURE PORT 0 (P0.15-P0.31) PINS AS INPUT?
    - TO CONFIGURE PORT 0 (P0.0-P0.15) PINS AS OUTPUT?
    - TO CONFIGURE PORT 0 (P0.15-P0.31) PINS AS OUTPUT?
    - TO CONFIGURE PORT 0 (P1.16-P1.31) PINS AS INPUT?
    - TO CONFIGURE PORT 0 (P1.16-P1.31) PINS AS OUTPUT?
    - WHETHER THE PINS P1.0-P1.15 ARE AVAILABLE AS GPIO?
3. LIST AND EXPLAIN THE REGISTERS USED TO CONTROL GPIO REGISTERS(IOPIN, IODIR,IOSET,IOCLR)

LEVEL 3:

1. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO BLINK THE LEDS CONNECTED TO PORT 0 PINS(P0.16-P0.23).
2. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO IMPLEMENT 8-BIT BINARY COUNTER ON PORT 0 PINS (P0.16-P0.23).
3. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO INTERFACE DAC WITH ARM7 TO GENERATE THE FOLLOWING WAVEFORMS:
    - SQUARE WAVE
    - TRIANGULAR WAVE
4. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO INTERFACE THE RELAY WITH ARM7.
5. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO BLINK THE BUILTIN LED CONNECTED TO PIN NUMBER 5 OF ARDUINO UNO.
6. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO INTERFACE LDR SENSOR CONNECTED TO PIN NUMBER 13 OF ARDUINO UNO.
7. DEVELOP AN EMBEDDED ‘C’ PROGRAM TO INTERFACE BUZZER CONNECTED TO PIN NUMBER 9 OF ARDUINO UNO.