3. List and explain seven ARM processor modes. Also, explain ARM core changing from user mode to interrupt request mode on an exception, with neat diagram.

→ Seven processor modes are there. In that six privileged mode they are:
abort, fast interrupt request, interrupt request, supervisor, system and undefined.

- The processor enters abort mode where there is a failed attempt to access memory.
- Fast interrupt request & 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 kernal 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.

non-privileged mode: user is used for program applications.

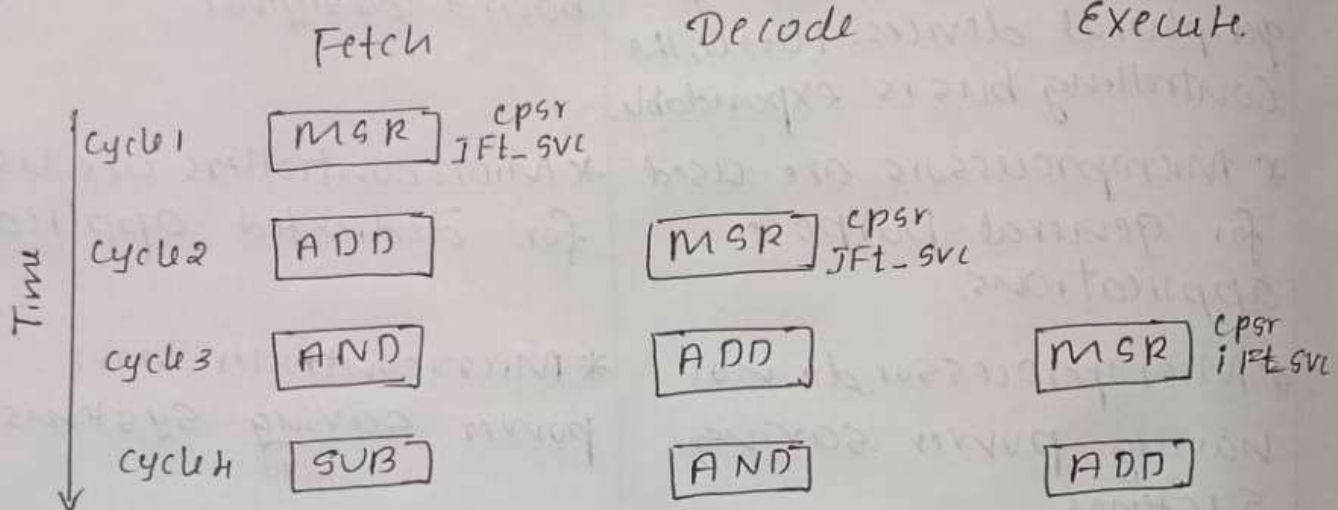
- The figure 1.1 illustrate what happens when an interrupt forces a mode change.
- The fig shows the core changing from user mode to interrupt request mode, which happens when an interrupt request occur due to an external device raising an interrupt to the processor core.

2. Justify the statement: The ARM pipeline will not process an instruction, until it passes completely through the execute stage.

→ • The ARM pipeline will not process an instruction, until it passes completely through the execute stage.

• For example, an ARM7 pipeline has executed an instruction only when the fourth instruction is fetched.

• The following fig shows an instruction sequence on an ARM7 pipeline.



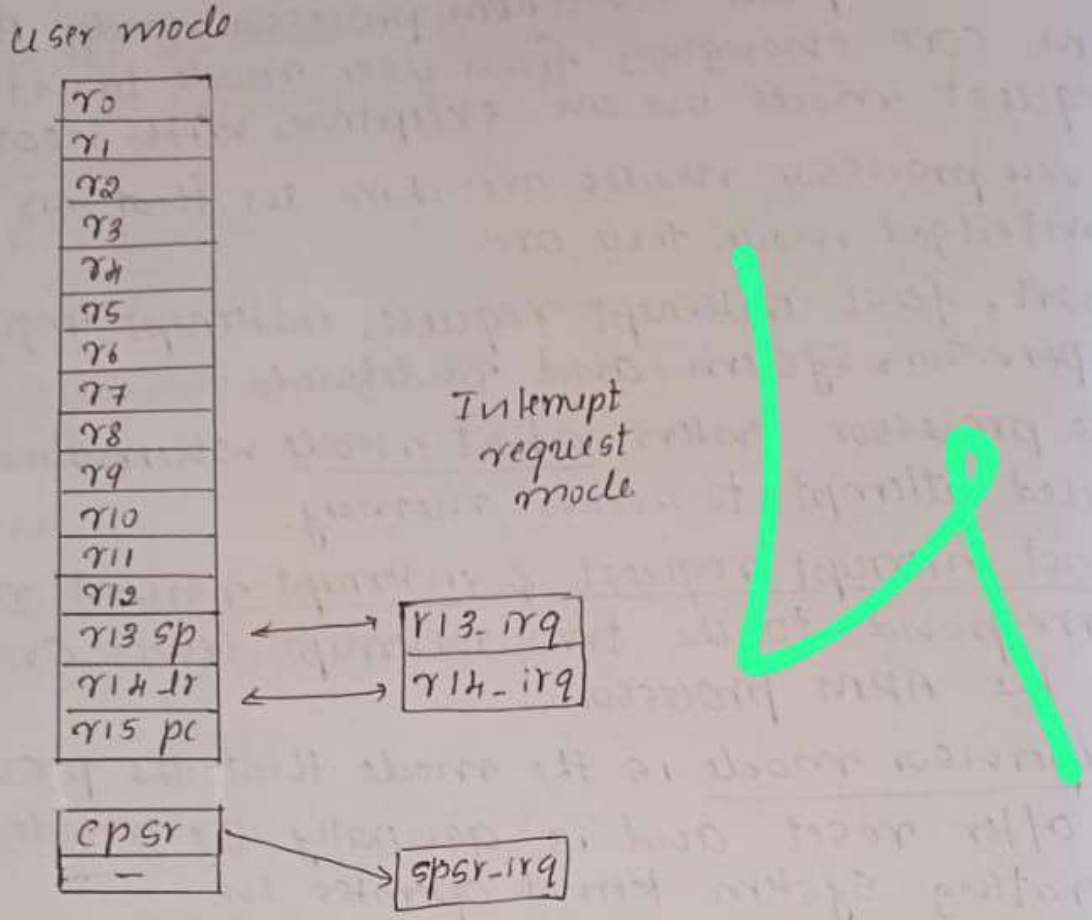
• The MSR instruction is used to enable IRQ interrupts which only occurs once the MSR instruction completes the execute stage of the pipeline. It clear the I bit in the cpsr to enable the IRQ interrupts.

• Once the ADD instruction enters the execute stage of the pipeline, IRQ interrupts are enabled.

Assignment-1

1) List out differences between Microprocessors vs Microcontrollers.

Microprocessors	Microcontrollers
* Microprocessors generally does not have RAM, ROM and I/O pins.	* Microcontrollers is 'all in one' processor, with RAM, I/O ports all on the chip.
* Microprocessor usually use its pin as a bus to interface to RAM, ROM & peripheral devices. Hence, the controlling bus is expandable.	* Controlling bus is internal and not available to the board designer.
* Microprocessors are used for general purpose applications.	* Microcontrollers are used for dedicated applications.
* Microprocessor do not having power saving systems.	* Microcontroller have power saving systems.
* The cost of microprocessor is high as compared to microcontroller.	* Microcontroller is cheaper than microprocessor.
* The processing speed of microprocessor is above 1 GHz.	* processing speed of Microcontroller is about 8 MHz to 50 MHz.
* Microprocessors are based on Von-neumann model.	* Microcontroller are based on Harvard architecture.



- This change causes user register r13 & r14 to be banked. The user registers are replaced with registers r13_irq & r14_irq respectively.
- r14_irq contain the return address & r13_irq contain the stack pointer for interrupt request mode
- The figure also shows a new register appearing in interrupt request mode: the spsr, which stores the previous mode cpsr. The cpsr being copied into spsr_irq.

4) With example, explain why a branch instruction takes three cycle.

→ • The processor must flush the pipeline when jumping to a new address.

5) Show the post condition when MOVS instruction shifts register r1 left by one bit and result is stored in r0. where r0 = 0X00000000, r1 = 0X80000004, and CPSR = nzcvqiFt

→ PRE cpsr = nzcvqiFt
r0 = 0X00000000 ; 0000| ..... |0000
r1 = 0X80000004 ; 1000| ..... |0100

MOVS r0, r1, LSL#1

nzcv 1000| ..... |0100

nzcv [1]|0000| ..... 1000 = 0X00000008

POST : cpsr = nzcvqiFt
r0 = 0X00000008
r1 = 0X80000004

6) Explain negative indexing & logarithmic indexing with examples.

→ negative indexing: This loops structure count from N to 0 in steps of size STEP

```
        RSB  i, N, #0 ; i = -N
loop
        ; loop body goes here and i = -N, -N+STEP, ...
        ADDS i, i, #STEP
        BLT  loop ; Use BLT or BLE to exclude 0 or not
```

logarithmic indexing: This loop structure counts down from 2^N to 1 in power of two. for example, if N = 4, then it counts 16, 8, 4, 2, 1

```
MOV   r1, #1
B     case1
AND   r0, r0, r1
EOR   r0, r2, r3
....
case1
SUB   r0, r0, r1
```

- This example shows why a branch instruction takes three cycles.
- The three executed instruction take a total of 5 cycles.
- The MOV instruction executes on the first cycle.
- On the second cycle, the branch instruction calculates the destination address.
- This causes the core to flush the pipeline and refill it using this new pc value.
- The refill takes two cycles.
- Finally, the SUB instruction executes normally.
- The below fig illustrates the pipeline state on each cycle.
- The pipeline drops the 2 instruction following the branch when the branch takes place.

Pipeline	Fetch	decode	ALU	LS1	LS2
cycle 1	AND	B	MOV	-	-
cycle 2	EOR	AND	B	MOV	-
cycle 3	SUB	-	-	B	MOV
cycle 4	-	SUB	-	-	B
cycle 5.	-	-	SUB	-	-

```
    MOV  i, #1
    MOV  i, i, LSL N
loop ; loop body
    MOVS i, i, LSR #1
    BNE  loop
```

The following loop structure counts down from an N-bit mask to a one-bit mask. example, N = 4 then it counts 15, 7, 3, 1

```
    MOV  i, #1
    RSB  i, i, i, LSL N ; i = (1<<N)-1
loop ; loop body
    MOVS i, i, LSR #1
    BNE  loop
```

7) Write a program to arrange a series of 32-bit numbers in ascending/descending order.

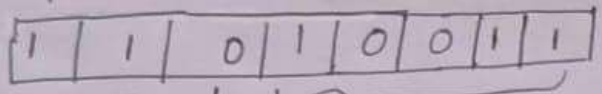
→ Ascending

```
    Area sort, code, readonly
Entry
    MOV  r , #.
    LDR  r2, =cvalue
    LDR  r3, =dvalue
loopo
    LDR  r1, [r2], #4
    STR  r1, [r3], #4
    SUBS r8, r8, #1
    Comp r8, #0
    BNE  loopo
    MOV  r5, #3
NXTPASS LDR r0, A
    MOV  r4, r5
NXTCOMP LDR r2, [r0]
    MOV  r1, r2
    ADD  r0, #4
    LDR  r2, [r0]
    CMP  r1, r2
    BLS  NOEXG
    STR  r1, [r0], #-4
    STR  r2, [r0], #4
```

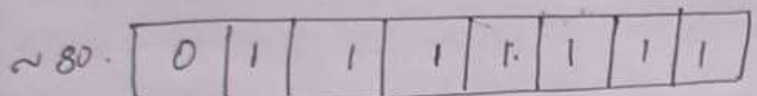
9) Write a code fragment to-
i) Copy the cpsr into register r1
ii) Clear bit 7 of r1
iii) Copy the register r1 back to cpsr
iv) Enable IRQ
v) Disable IRQ
vi) Enable FIQ
vii) Disable FIQ.

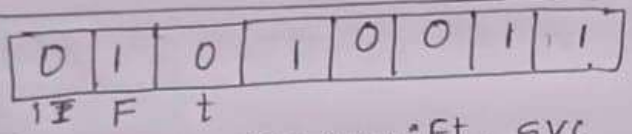
10

→ PRE cpsr = nzcvqIFt_SVC
MRS r1, cpsr
BIC r1, r1, #0x80 ; 0b01000000
MSR cpsr_c, r1
MSR cpsr_c, #0x10
MSR cpsr_c, #0x10



AND





BIC r1, r1, #0x80
r1 = r1 & ~0x80.
80 = 10000000

POST cpsr = nzcvqiFt_SVC

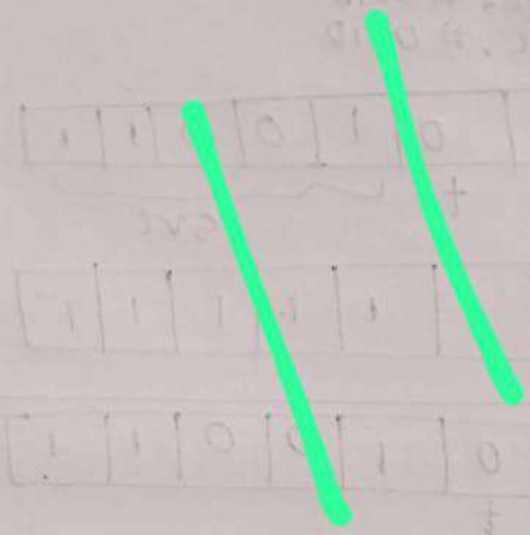
- The MSR first copies the cpsr in to register r1
- The BIC instruction clears bit 7 of r1.
- Register r1 is then copied back into the cpsr, which enables IRQ interrupts.
- If the bit is set to 1, interrupts are disable
- It set to 0, interrupts are enabled.
- The F- bit on the CPSR control FIQ interrupts.

10) If (PRE) cpsr = m2cviFt_USER, r1 = 0x00000001 &
SUBS r1, r1, #1 compute post conditions.

→ PRE cpsr = m2cviFt_USER
r1 = 0x00000001
SUBS r1, r1, #1 ; r1 = r1-1

$$\begin{array}{r} 0000\,|\,\overset{0}{\ldots}\,|\,0001 \\ -\;\; 0000\,|\,\ldots\ldots\,|\,0001 \\ \hline 0000\,1\,|\,\ldots\ldots\,|\,0000 \end{array} = 0x00000000$$

POST cpsr = m2cviFt_USER
r1 = 0x00000000



```
NOEXG SUBS 9H, #1
  BNE NXTCOMP
  SUBS 15, #1
  BNE NXTPASS
BI B BI
A DCD 0XH00000000
 END.
```