Module 1: (Cont…)

ARM core data flow model

Q. Explain ARM core data flow model with neat diagram.

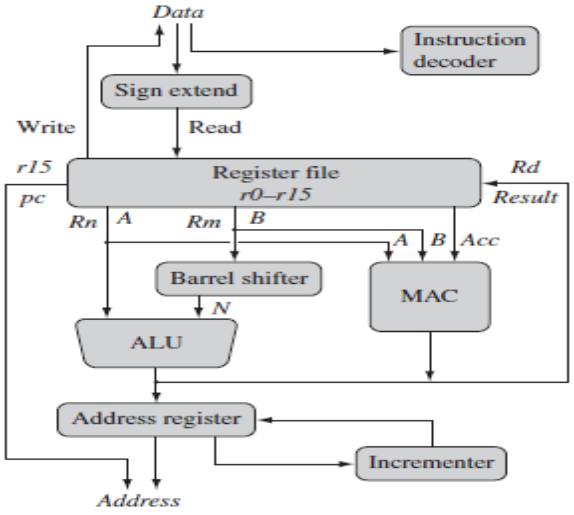


Figure1: ARM core dataflow model

- An ARM core as functional units connected by data buses, as shown in Figure1, where, the arrows represent the flow of data, the lines represent the buses, and the boxes represent either an operation unit or a storage area.
- The instruction decoder translates instructions before they are executed.
- The ARM processor, like all RISC processors, uses a load - store architecture.
- **Load instructions** copy data from memory to registers, and conversely the **store instructions** copy data from registers to memory.
- There are no data processing instructions that directly manipulate data in memory.
- ARM instructions typically have two source registers, Rn and Rm, and a single destination register, Rd. Source operands are read from the register file using the internal buses A and B, respectively.
- The ALU (arithmetic logic unit) or MAC (multiply-accumulate unit) takes the register values Rn and Rm from the A and B buses and computes a result.
- Data processing instructions write the result in Rd directly to the register file.
- Load and store instructions use the ALU to generate an address to be held in the address register and broadcast on the Address bus.
- One important feature of the ARM is that register Rm alternatively can be preprocessed in the barrel shifter before it enters the ALU.
- After passing through the functional units, the result in Rd is written back to the register file using the Result bus.
- For load and store instructions the incrementer updates the address register before the core reads or writes the next register value from or to the next sequential memory location.

REGISTERS

Q5. Explain briefly the active registers available in user mode.
OR
With a neat diagram explain the different general purpose registers of ARM processors.
Answer: Figure shown below shows the active registers available in user mode. All the registers shown are 32 bits in size.

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15

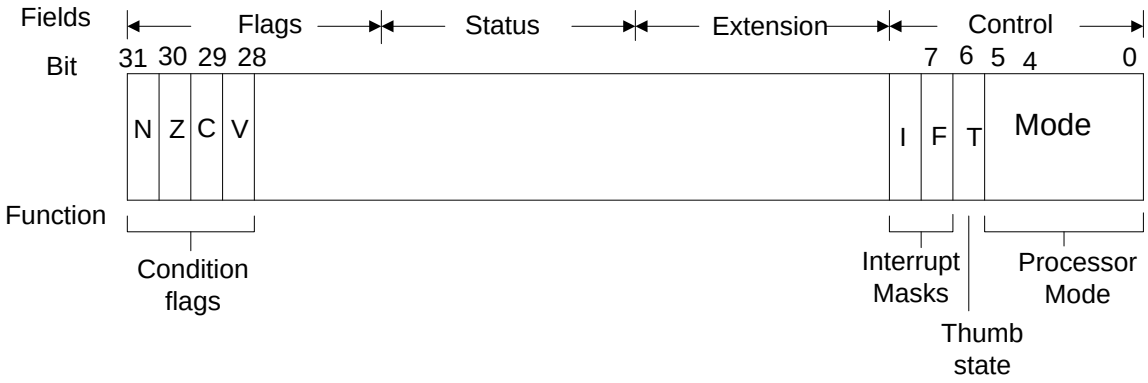
cpsr
-

- There are up to 18 active registers: 16 data registers and 2 processor status registers. The data registers are visible to the programmer as r0 to r15.
- The ARM processor has three registers assigned to a particular task: r13, r14 and r15.
- Register r13: Register r13 is traditionally used as the stack pointer (sp) and stores the head of the stack in the current processor mode.
- Register r14: Register r14 is called the link register (lr) and is where the core puts the return address whenever it calls a subroutine.
- Register r15: Register r15 is the program counter (pc) and contains the address of the next instruction to be fetched by the processor.
- In addition to the 16 data registers, there are two program status registers: current program status register (cpsr) and saved program status register (spsr).

CPSR (Current Program Status Register)

Q6. Explain the various fields in current program status register (CPSR) with neat diagram.

Answer: Figure below shows the basic layout of a generic program status register.



- The cpsr is divided into four fields, each 8 bits wide: flags, status, extension and control.
- In current designs the extension and status fields are reserved for future use.

- The control field contains the processor mode, state and interrupts mask bits.
- The flag field contains the condition flags.
- The following table gives the bit patterns that represent each of the processor modes in the cpsr.

Mode	Mode[4:0]	Privilaged
Abort	10111	Yes
Fast interrupt request	10001	Yes
Interrupt request	10010	Yes
Supervisor	10011	Yes
System	11111	Yes
Undefined	11011	Yes
User	10000	No

- When cpsr bit 5, T=1, then the processor is in Thumb state. When T=0, the processor is in ARM state.
- The cpsr has two interrupt mask bits, 7 and 6 (I and F) which control the masking Interrupt request (IRQ) and Fast Interrupt Request (FIR).
- Condition flags are updated by comparisons and the result of ALU operations that specify the S instruction suffix.
- For example, if SUBS subtract instruction results in a register value of zero, then the Z flag in the cpsr is set.

- The following table shows the conditional flags:

Flag	Flag Name	Set when
N	Negative	Bit 31 of the result is a binary 1
Z	Zero	The result is zero, frequently used to indicate equality
C	Carry	The result causes an unsigned carry
V	Overflow	The result causes a signed overflow

Processor Mode

Q7. Explain the various modes of operation of ARM processor.

Answer:

- Each processor mode is either privileged or nonprivileged.
- A privileged mode allows read-write access to the cprs.
- A nonprivileged mode only allows read access to the control field in the cpsr but allows read-write access to the conditional flags.
- There are **seven processor modes** : six privileged modes and one nonprivileged mode.
- The privilege modes are abort, fast interrupt request , interrupt request, supervisor, system and undefined. The nonprivileged mode is user.
  1. The processor enter **abort mode** when there is a failure to attempt to access memory.
  2. **Fast interrupt request and interrupt request modes** correspond to the two interrupt levels available on the ARM processor.
  3. **Supervisor mode** is the mode that the processor is in after reset and is generally the mode that an operating system kernel operates in.

4. **System mode** is a special version of user mode that allows full read-write access to the cpsr.
5. **Undefined mode** is used when the processor encounters an instruction that is undefined or not supported by the implementation. User mode is used for program and applications.

Banked Registers

Q8. Explain programmer's model of ARM processor with complete register sets avaliable.

OR

What are banked registers? Show how the banked registers are utilized when the user mode changes to IRQ mode.

Answer:

- Figure below shows all 37 registers in the register file.
- Of these, 20 registers are hidden from a program at different times. These registers are called banked registers.
- They are available only when the processor is in a particular mode, for example, abort mode has banked registers r13_abt, r14_abt and spsr_abt.
- Banked registers of a particular mode are denoted by an underline character post-fixed to the mode mnemonic.

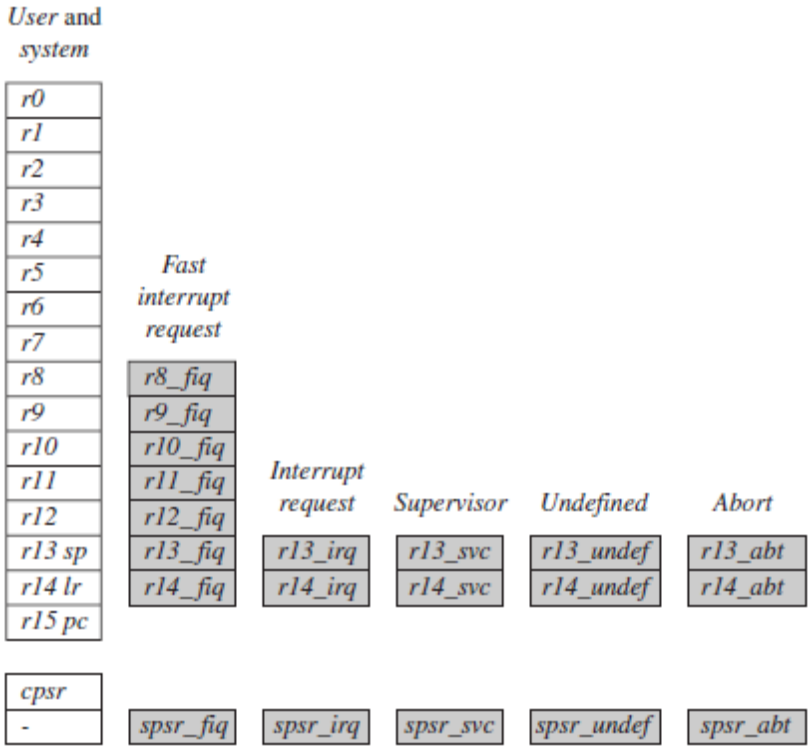


Figure 1: Complete ARM register set

- Every processor mode except user mode can change mode by writing directly to the mode bits of the cpsr.
- All privileged modes except system mode have a set of associated banked registers that are subset of the main 16 registers.
- If the processor mode is changed, a banked register from the new mode will replace an existing register.
- The processor mode can be changed by a program that writes directly to the cpsr when the processor core is in privilege mode.

- The following exception and interrupts causes a mode change: reset, interrupt request, fast interrupt request, software interrupt, data abort, prefetch abort and undefined instructions.
- Exceptions and interrupts suspend the normal execution of sequential instructions and jump to a specific location.
- Following figure 2 illustrates the happening when an interrupt forces a mode change.
- The figure 2 shows the core changing from user mode to interrupt request mode, which happens when an interrupt request occurs due to an external device raising an interrupt to the processor core. This change causes user registers r13 and r14 to be banked.

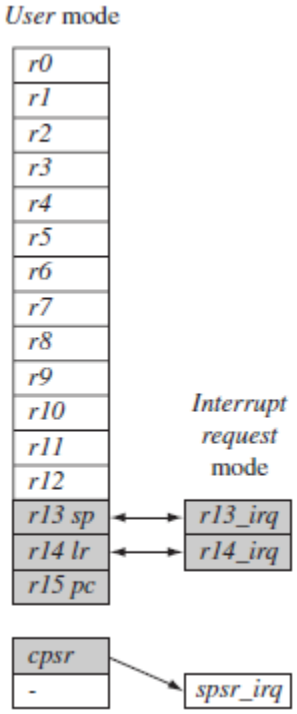


Figure 2: changing mode on an exception

- The user registers are replaced with registers r13_irq and r14_irq respectively.
- r14_irq contains the return address and r13_irq contains the stack pointer for interrupt request mode.
- The saved program status register (spsr), which stores the previous mode's cpsr.

PIPELINE

Q9. With neat diagram explain the various blocks in a 3 stage pipeline of ARM processor organization.
OR
Explain ARM pipeline with 3,5,6 stages.
Answer:

- Pipeline is the mechanism to speed up execution by fetching the next instruction while other instruction are being decoded and executed.
- Figure 1 shows the ARM7 three-stage pipeline.

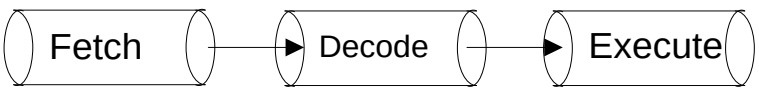
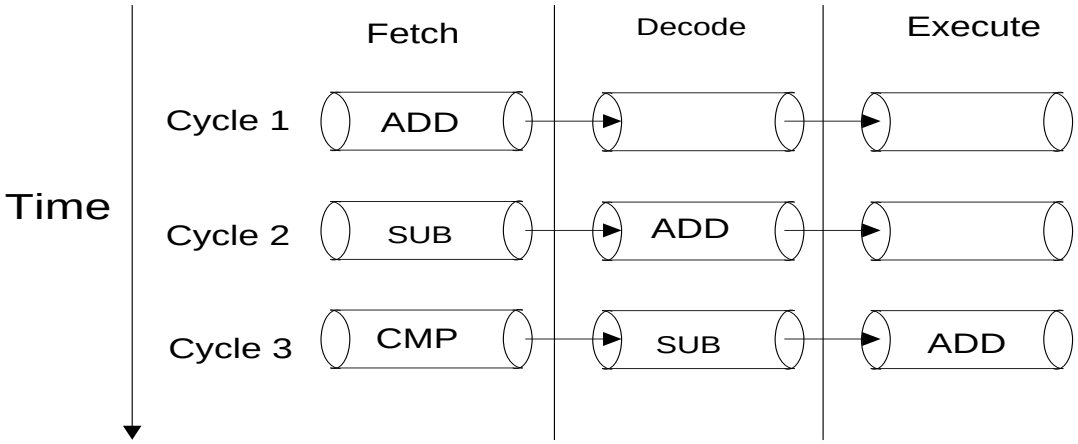


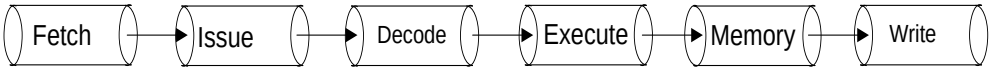
Figure 1: ARM7 Three-stage pipeline

- 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.
- Figure 2 illustrates the pipeline using a simple example. It shows a sequence of three instructions being fetched, decoded and executed by the processor.
- Each instruction takes a single cycle to complete after the pipeline is filled.
    - In the first cycle, the core fetches the ADD instruction from the memory.
    - In the second cycle, the core fetches the SUB instruction and decode the ADD instruction.
    - In the third cycle, the core fetches CMP instruction from the memory, decode the SUB instruction and execute the ADD instruction.
    - The ADD instruction is executed, the SUB instruction is decoded, and the CMP instruction is fetched. This procedure is called **filling the pipeline.**



- The pipeline design for each ARM family differs. For example, the ARM9 core increases the pipeline length to five stages as shown in the figure below.

- The ARM10 increases the pipeline length still further by adding a sixth stage as shown in the figure below.



- As the pipeline length increases the amount of work done at each stage is reduced, which allows the processor to attain a higher operating frequency. This in turn increases the performance.
- **Pipeline Executing Characteristics**
    a. The ARM pipeline has not processed an instruction until it passes completely through the execute stage. For example, an ARM7 pipeline (with three stages) has executed an instruction only when the fourth instruction is fetched. Figure below shows an instruction sequence on an ARM7 pipeline.

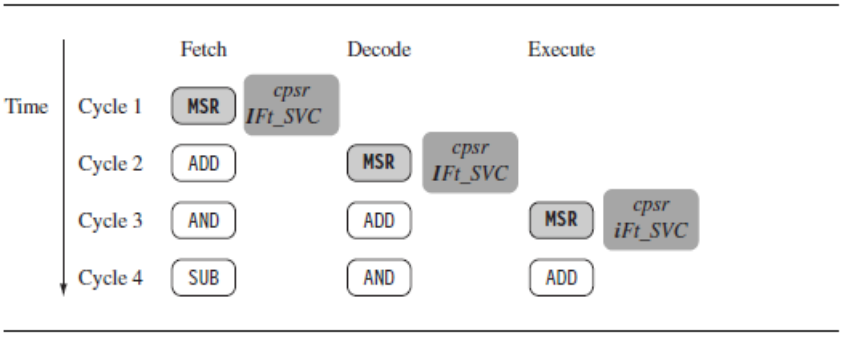


Figure 1: ARM instruction sequence

b. In the execute stage, the pc always points to the address of the instruction plus 8 bytes. In other words, the pc always points to the address of the instruction being executed plus two instructions ahead as shown in figure 2 below

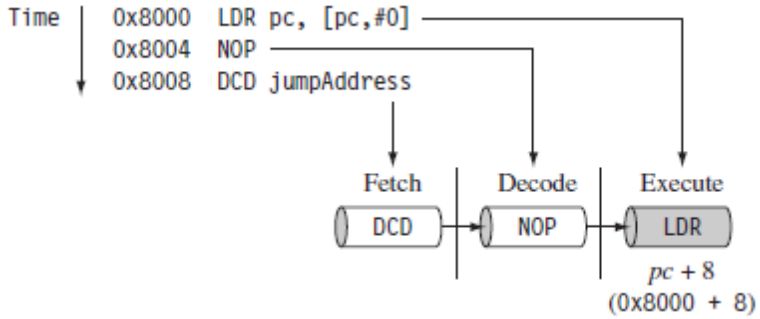


Figure 2: Example: pc = address + 8

c. The execution of a branch instruction or branching by the direct modification of the pc causes the ARM core to flush its pipeline.
d. ARM10 uses branch prediction, which reduces the effect of a pipeline flush by predicting possible branches and loading the new branch address prior to the execution of the instruction.
e. An instruction in the execute stage will complete even though an interrupt has been raised.

Exceptions, Interrupts, and the Vector Table

Q10. Explain briefly the interrupt and the vector table.

Answer:

- When an exception or interrupt occurs, the processor sets the program counter (pc) to a specific memory address.
- The address is within a specified address range called the vector table.
- The entries in the vector table are the instructions that branch to specific routines designed to handle particular exception or interrupt.
- The memory map address 0x00000000 is reserved for the vector table, a set of 32-bit words.
- On some processors, the vector table can optionally located at higher address in memory starting at the 0xffff0000.
- When an exception or interrupt occurs, the processor suspends normal execution and starts loading instructions from the exception vector table.
- Each vector table entry contains a form of branch instruction pointing to start of a specific routine.

- Following is the vector table:

Exception/Interrupt	Shorthand	Address	High address
Reset	RESET	0x00000000	0xffff0000
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved	---	0x00000014	0xffff0014
Interrupt request	IRQ	0x00000018	0xffff0018
Fast interrupt request	FIQ	0x0000001c	0xffff001c

- **Reset vector** is the location of the first instruction executed by the processor when power is applied. This instruction branches to the initialization code.
- **Undefined instruction vector** is used when the processor cannot decode the instruction.
- **Software interrupt vector** is called when SWI instruction is executed. The SWI is frequently used as the mechanism to invoke an operating system routine.
- **Prefetch abort vector** occurs when the processor attempts to fetch an instruction from an address without the correct access permissions.
- **Data abort vectors** is similar to a prefetch abort but is raised when an instruction attempts to access data memory without the correct access permissions.
- **Interrupt request vector** is used by external hardware to interrupt the normal execution flow of the processor.
- **Fast interrupt request vector** is similar to the interrupt request but is reserved for hardware requiring faster response times.

Core Extensions

Q11. Discuss the following with neat diagrams

a. Von Neumann architecture with cache
b. Harvard architecture with TCM

OR

Discuss all 3 core extensions.

Answer:

There are three core extensions wrap around ARM processor: cache and tightly coupled memory, memory management and the coprocessor interface.

1. Cache and tightly coupled memory: The cache is a block of fast memory placed between main memory and the core. With a cache the processor core can run for the majority of the time without having to wait for data from slow external memory.

- ARM has two forms of cache. The first found attached to the Von Neumann-style cores. It combines both data and instruction into a single unified cache as shown in the figure 1 below.

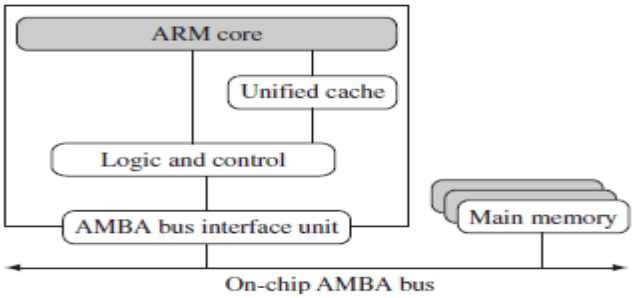


Figure 1: A simplified Von Neumann architecture with cache.

- The second form, attached to the Harvard-style cores, has separate cache for data and instruction as shown figure 2

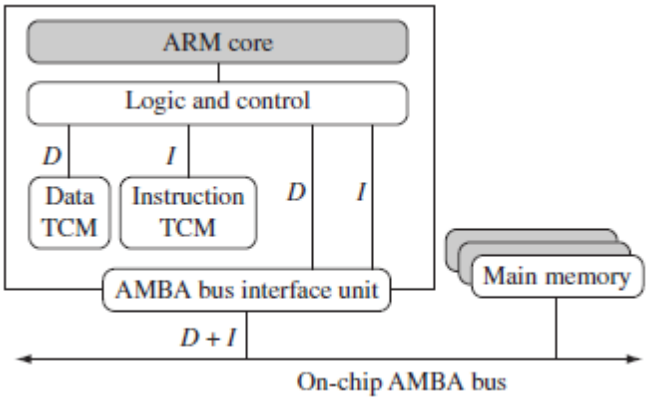


Figure 2: A simplified Harvard architecture with TCMs.

- A cache provides an overall increase in performance but will not give predictable execution.
- But for real-time systems it is paramount that code execution is *deterministic.*
- 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.
- By combining both technologies, ARM processors can behave both improved performance and predictable real-time response. The following diagram shows an example of core with a combination of caches and TCMs as shown in figure 3

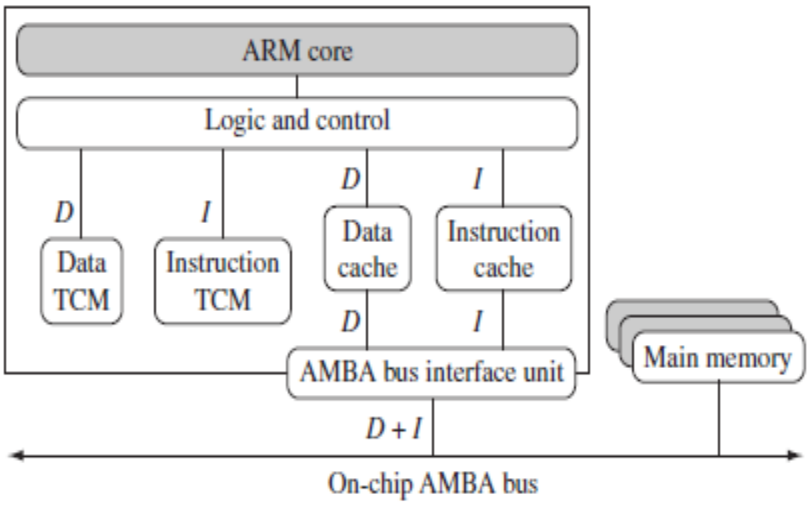


Figure 3: combining both technologies

2. Memory management:

- Embedded systems often use multiple memory devices. It is usually necessary to have a method to help organize these devices and protect the system from applications trying to make appropriate accesses to hardware.

- This is achieved with the assistance of memory management hardware.
- ARM cores have three different types of memory management hardware- no extensions provide no protection, a memory protection unit (MPU) providing limited protection and a memory management unit (MMU) providing full protection.
    - **Nonprotected memory** is fixed and provides very little flexibility. It normally used for small, simple embedded systems that require no protection from rogue applications.
    - **Memory protection unit (MPU)** employs a simple system that uses a limited number of memory regions. These regions are controlled with a set of special coprocessor registers, and each region is defined with specific access permission but don't have a complex memory map.
    - **Memory management unit (MMU)**are the most comprehensive memory management hardware available on the ARM. The MMU uses a set of translation tables to provide fine-grained control over memory.
        - These tables are stored in main memory and provide virtual to physical address map as well as access permission. MMU designed for more sophisticated system that supports multitasking.

Briefly explain how coprocessors can be attached to ARM processor.

3. **Coprocessors:**

- A coprocessor extends the processing features of a core by extending the instruction set or by providing configuration registers.
- More than one coprocessor can be added to the ARM core via the coprocessor interface.
- The coprocessor can be accessed through a group of dedicated ARM instructions that provide a load-store type interface.
- The coprocessor can also extend the instruction set by providing a specialized instructions that can be added to standard ARM instruction set to process vector floating-point (VFP) operations.
- These new instructions are processed in the decode stage of the ARM pipeline. If the decode stage sees a coprocessor instruction, then it offers it to the relevant coprocessor.
- But, if the coprocessor is not present or doesn't recognize the instruction, then the ARM takes an undefined instruction exception.