CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test - I

Sub:	MICROCONTROLLERS							Code:	BCS402
Date:	05-06-2024	Duration:	90 mins	Max Marks:	50	Sem:	4th C	Branch:	CS(DS)

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1	a. What are the design rules for RISC design philosophy?	6	CO1	L1
	b. If r0=0x00000000, r1=0x80000004 Find the content of the registers r0 and r1 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not. MOVS r0,r1,LSL #1	4	C03	L3
2	With a neat diagram explain the different hardware components of an embedded device based on ARM core and ARM processor based embedded system software.	10	CO2	L2
3	a. Define the concept of pipelining for ARM processors.	6	CO2	L2
	b. If r0=0x00000000, r1=0x00000002 and r2=0x00000001 Find the content of the register r0, r1 and r2 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not. RSB r0,r1,#0	4	CO3	L3
4	Write short notes on: (a) Exceptions, Interrupts and Vector Table (b) Core extensions for ARM processor	10	CO2	L2

CCI **CI** **HOD**

Sub:	MICROCONTROLLERS							Code:	BCS402
Date:	05-06-2024	Duration:	90 mins	Max Marks:	50	Sem:	4th C	Branch:	CS(DS)

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1	a. What are the design rules for RISC design philosophy?	6	CO1	L1
	b. If r0=0x00000000, r1=0x80000004 Find the content of the registers r0 and r1 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not. MOVS r0,r1,LSL #1	4	C03	L3
2	With a neat diagram explain the different hardware components of an embedded device based on ARM core and ARM processor based embedded system software.	10	CO2	L2
3	a. Define the concept of pipelining for ARM processors.	6	CO2	L2
	b. If r0=0x00000000, r1=0x00000002 and r2=0x00000001 Find the content of the register r0, r1 and r2 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not. RSB r0,r1,#0	4	CO3	L3
4	Write short notes on: (a) Exceptions, Interrupts and Vector Table (b) Core extensions for ARM processor	10	CO2	L2

CCI **CI** **HOD**

Sub:	MICROCONTROLLERS							Code:	BCS402
Date:	05-06-2024	Duration:	90 mins	Max Marks:	50	Sem:	4th C	Branch:	CS(DS)

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
5	a. Explain in detail about Register Allocation	6	CO2	L1
	b. If r1=0x00000002 and r2=0x00000002 Find the content of the register r1 and r2 after the following instruction executed in isolation and also mention if the CPSR register is updated or not. TST r1,r2	4	CO3	L3
6	a. Draw and Explain the ARM core Dataflow Model (Architecture).	6	CO1	L3
	b. If r0=0x00000000, r1=0x00000001, r2=0x00000002 and r3=0x00000003 Find the content of the register r0, r1, r2 and r3 after the following instruction is executed in isolation. MLA r0,r1,r2,r3	4	CO3	L3
7	a. Explain in detail the processor modes available for ARM7.	6	CO2	L2
	b. Differentiate between (i) RISC and CISC	4	CO1	L2

Sub:	MICROCONTROLLERS							Code:	BCS402
Date:	05-06-2024	Duration:	90 mins	Max Marks:	50	Sem:	4th C	Branch:	CS(DS)

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
5	a. Explain in detail about Register Allocation	6	CO2	L1
	b. If r1=0x00000002 and r2=0x00000002 Find the content of the register r1 and r2 after the following instruction executed in isolation and also mention if the CPSR register is updated or not. TST r1,r2	4	CO3	L3
6	a. Draw and Explain the ARM core Dataflow Model (Architecture).	6	CO1	L3
	b. If r0=0x00000000, r1=0x00000001, r2=0x00000002 and r3=0x00000003 Find the content of the register r0, r1, r2 and r3 after the following instruction is executed in isolation. MLA r0,r1,r2,r3	4	CO3	L3
7	a. Explain in detail the processor modes available for ARM7.	6	CO2	L2
	b. Differentiate between (i) RISC and CISC	4	CO1	L2

MICROCONTROLLERS IAT I solution

1. a. What are the design rules for RISC design philosophy?

Answer:

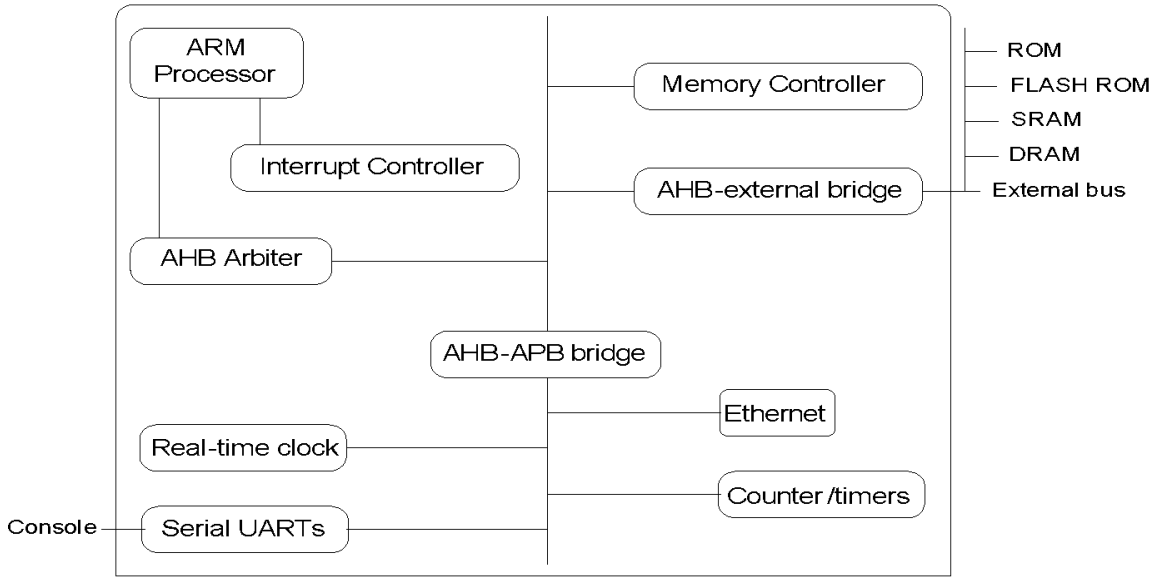
- RISC design philosophy is
  - Aimed at simple but powerful instructions that execute within a single cycle at a high clock speed.
  - Concentrates on reducing the complexity of instructions performed by the hardware.
  - Provides greater flexibility and intelligence in software rather than hardware.
- The RISC philosophy is implemented with four major design rules:
  - **Instructions:** RISC has a reduced number of instruction classes. These classes provide simple operations so that each is executed in a single cycle. Each instruction is a fixed length to allow the pipeline to fetch future instructions before decoding the current instruction.
  - **Pipeline:** The processing of instructions is broken down into smaller units that can be executed in parallel by pipelines.
  - **Register:** RISC machines have a large general-purpose register set. Any register can contain either data or an address.
  - **Load-store architecture:** The processor operates on the data held in registers. Separate load and store instructions transfer data between the register bank and external memory.
- These design rules allow a RISC processor to be simpler, and thus the core can operate at higher clock speed.

b. If r0=0x00000000, r1=0x80000004
Find the content of the registers r0 and r1 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not.
MOVS r0,r1,LSL #1

Before execution of the instruction, r1=0x80000004=1000 0000 0000 0000 0000 0000 0000 0100 in binary. After 1 bit logical shift left(LSL), the value is 0000 0000 0000
0000 0000 0000 0000 1000 which is copied to r0. After the execution of the instruction,
r0=0x00000008 and r1=0x80000004. CPSR is not updated.

2. With a neat diagram explain the different hardware components of an embedded device based on ARM core and ARM processor based embedded system software.

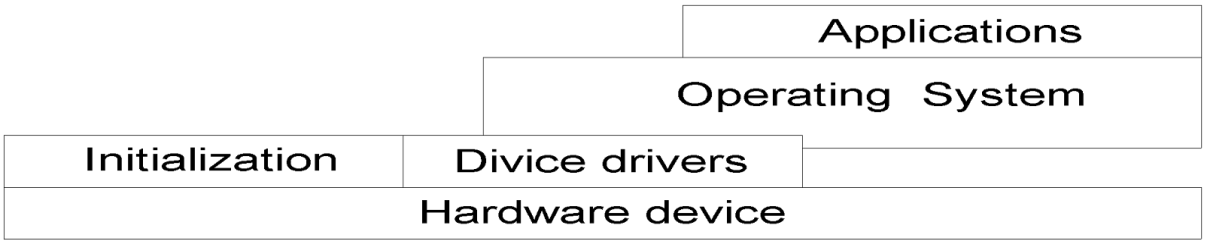
Answer: Figure shown below shows a typical embedded device based on ARM core. Each box represents a feature or function.



- ARM processor based embedded system hardware can be separated into the following four main hardware components:
    - **The ARM processor:** The ARM processor controls the embedded device. Different versions of the ARM processor are available to suit the desired operating characteristics.
    - **Controllers:** Controllers coordinate important blocks of the system. Two commonly found controllers are memory controller and interrupt controller.
    - **Peripherals:** The peripherals provide all the input-output capability external to the chip and responsible for the uniqueness of the embedded device.
    - **Bus:** A bus is used to communicate between different parts of the device.

Embedded System Software

- An embedded system requires software to drive it. Figure below shows typical software components required to control an embedded device.
- Each software components in the stack uses a higher level of abstraction to separate the code from the hardware device.



Initialization (BOOT) code:

- Initialization code (or boot code) takes the processor from the reset state to a state where the operating system can run.
- First code executed on the board and is specific to a particular target or group of targets.
- Handles a number of administrative tasks prior to handling control over to an operating system.
- We can group these different tasks into three phases: initial hardware configuration, diagnostics and booting.
    - **Initial hardware configuration** involves setting up the target platform so it can boot an image.
    - **Diagnostics:** The primary purpose of diagnostic code is fault identification and isolation.
    - **Booting:** involves loading an image and handling control over the image. Loading an image involves copying an entire program including code and data into RAM.

The operating system

- An operating system organizes the system resources: the peripherals, memory and processing time.
- ARM processors support over 50 operating systems.
- We can divide operating systems into two main categories: real time operating systems (RTOSs) and platform operating systems.
- RTOSs provide guaranteed response times to events. Systems running an RTOS generally do not have secondary storage.
- Platform operating systems require a memory management unit to manage large, non-real time applications and tends to have secondary storage.

The device drivers:

- Device drivers are the third component that provides a consistent software interface to the peripherals on the hardware device.

Applications:

- Finally, an application performs one of the tasks required for a device. For example, a mobile phone might have diary application.
- There may be multiple applications running on the same device, controlled by the operating systems.
- An embedded system can have one active application or several applications running simultaneously.
- The software components can run from ROM or RAM. ROM code that is fixed on the device is called firmware, for example the initialization code.

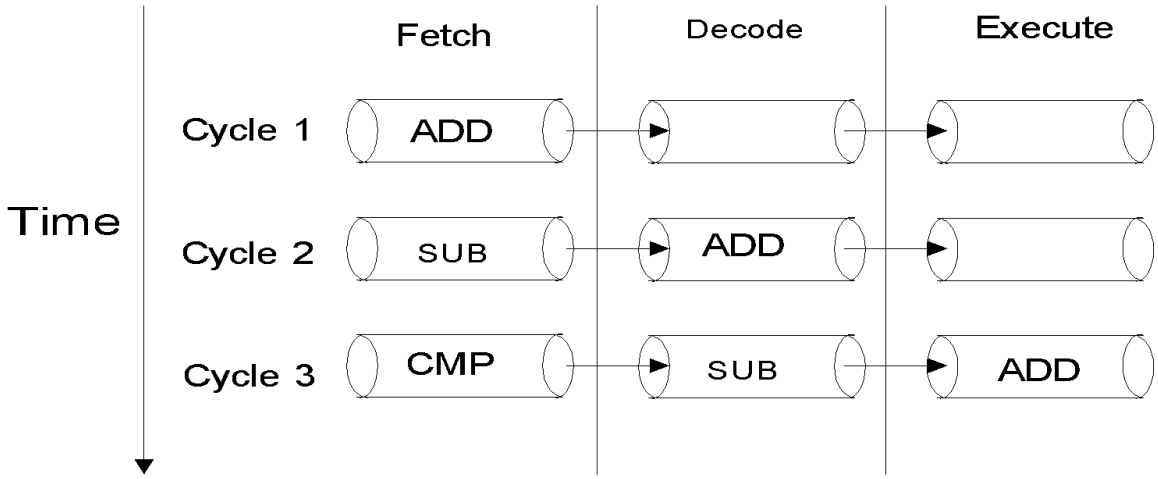
3. a Define the concept of pipelining for ARM processors.

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.

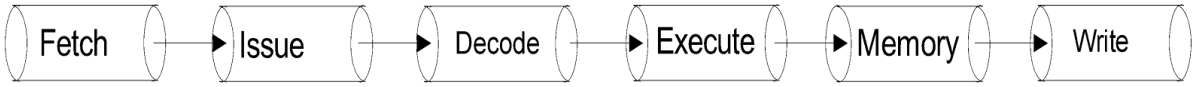
Figure 1: ARM7 Three-stage pipeline

- 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.

b. If r0=0x00000000, r1=0x00000002 and r2=0x00000001 Find the content of the register r0, r1 and r2 after the following instruction is executed in isolation. Mention if the CPSR register is updated or not.
RSB r0,r1,#0

r0=0x0-r1=-r1 which is copied to r0 as 2's complement of r1. r1=0x00000002=0000 0000 0000 0000 0000 0000 0000 0010 1's complement=1111 1111 1111 1111 1111 1111 1111 1101 2's complement=1111 1111 1111 1111 1111 1111 1111 1110=0xfffffffe Hence, r0=0xfffffffe, r1=0x00000002 and r2=0x00000001. CPSR is not updated.

4. Write short notes on:
(a) Exceptions, Interrupts and Vector Table (b) Core extensions for ARM processor

Exceptions, Interrupts, 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

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.

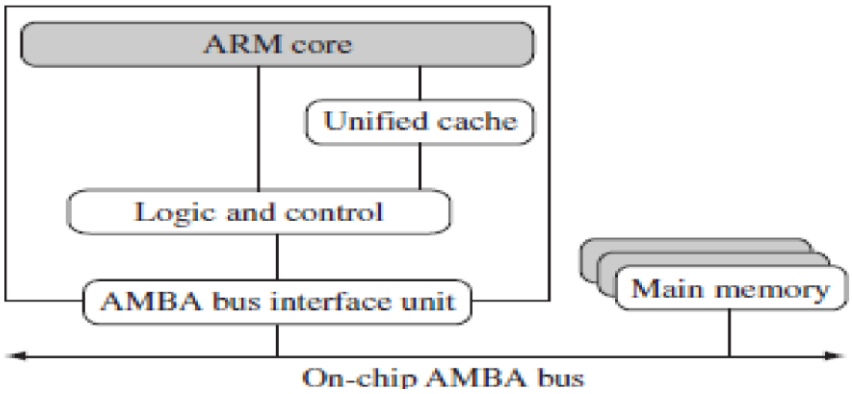


Figure 1: A simplified Von Neumann architecture with cache.

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

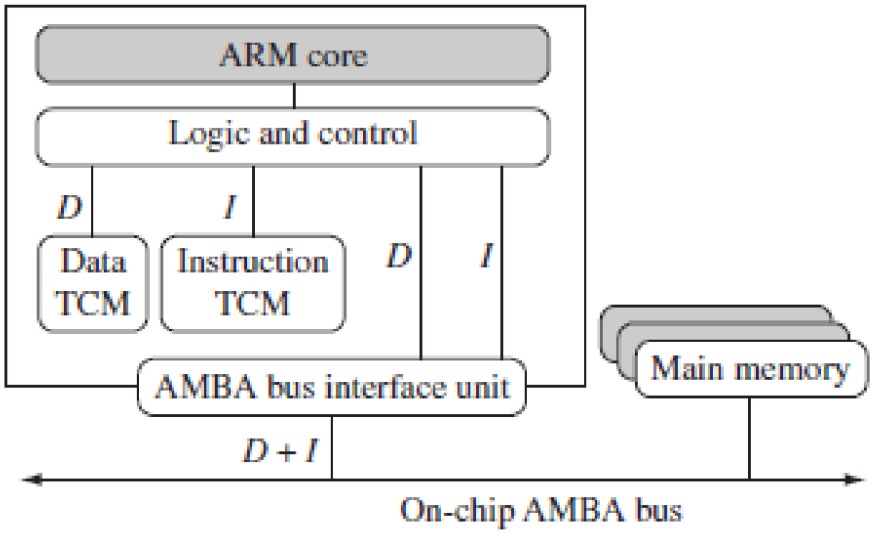


Figure 2: A simplified Harvard architecture with TCMs.

- 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

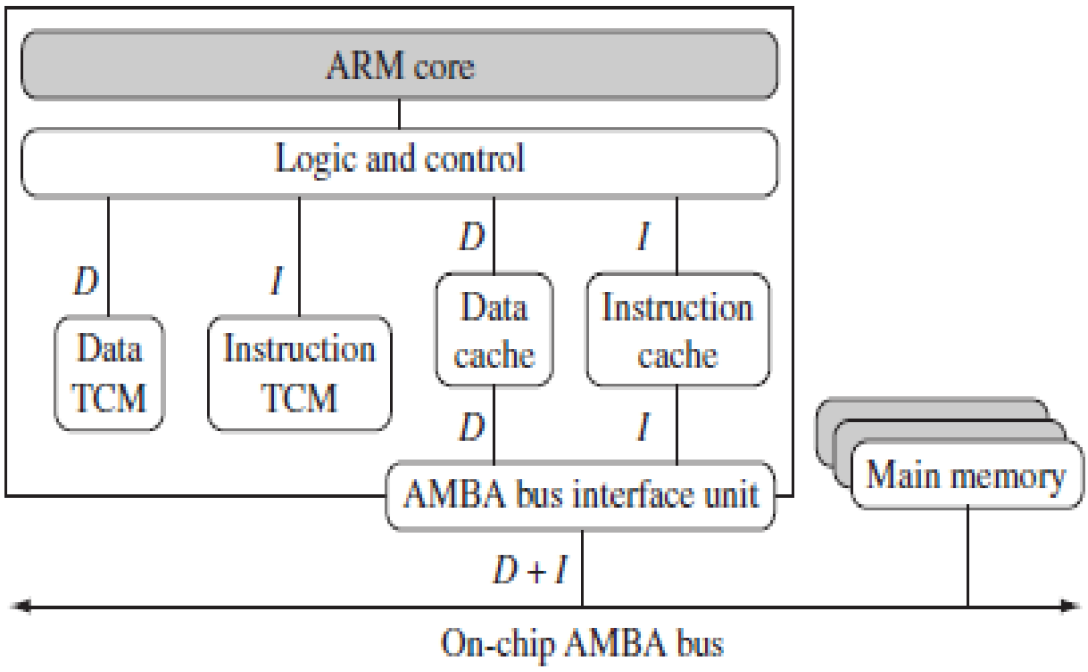


Figure 3: combining both technologies

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.

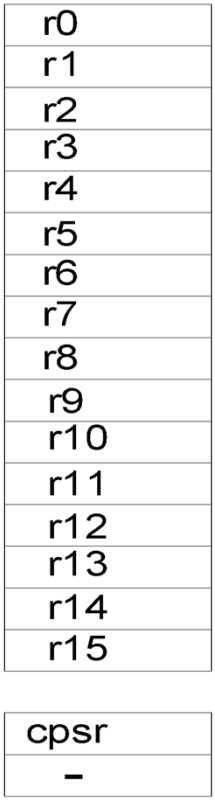
3. **Coprocessors:**

- 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.

5. a. Explain in detail about Register Allocation.

Answer: Figure shown below shows the active registers available in user mode. All the registers shown are 32 bits in size.



- 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).

b. If r1=0x00000002 and r2=0x00000002 Find the content of the register r1 and r2 after the following instruction executed in isolation and also mention if the CPSR register is updated or not.
TST r1,r2

After execution, r1=0x00000002 and r2=0x00000002.CPSR is updated (Zero flag is set if Exclusive OR of r1and r2=0)

6.a Draw and Explain the ARM core Dataflow Model (Architecture).

Answer:

.

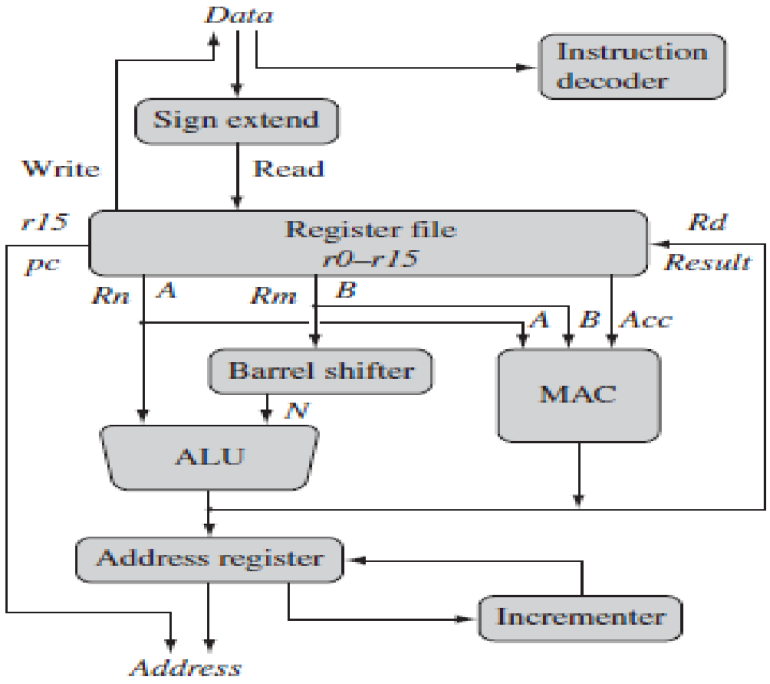


Figure1: ARM core dataflow model

- 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 pre-processed 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 incrementor updates the address register before the core reads or writes the next register value from or to the next sequential memory location.

b. If r0=0x00000000, r1=0x00000001, r2=0x00000002 and r3=0x00000003 Find the content of the register r0, r1, r2 and r3 after the following instruction is executed in isolation.
MLA r0,r1,r2,r3

After execution, r0=r1*r2+r3 Hence, after execution, r0=0x00000005, r1=0x00000001, r2=0x00000002 and r3=0x00000003

7. a. Explain in detail the processor modes available for ARM7.

Answer:

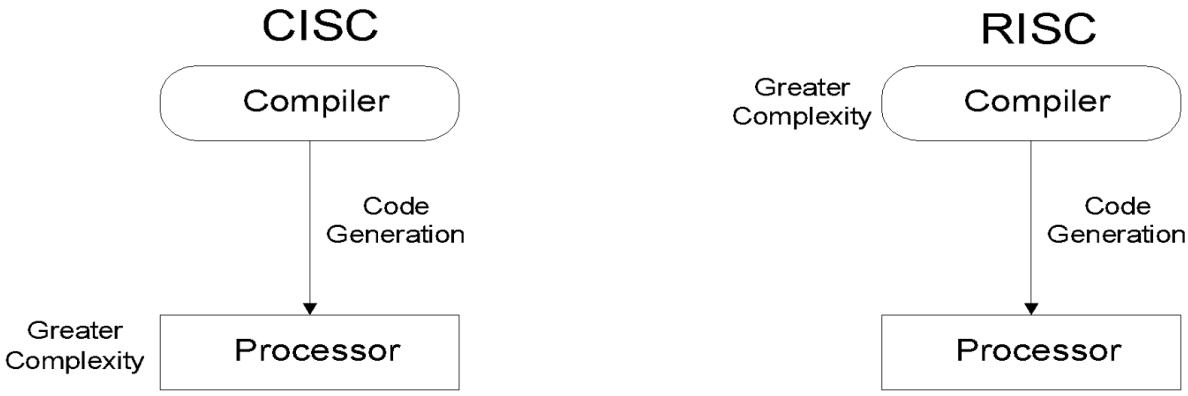
- Each processor mode is either privileged or nonprivileged.
- A privileged mode allows read-write access to the cprs.
- A nonprivileged mode only allows read access to the control field in the cpsr but allows read-write access to the conditional flags.
- There are **seven processor modes** : six privileged 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.

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.

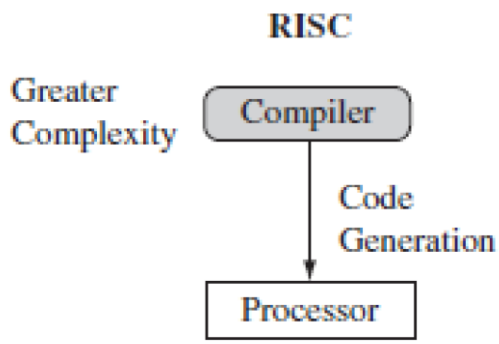
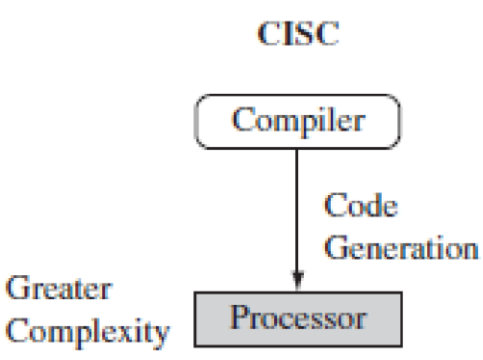
b. Differentiate between (i) RISC and CISC

Answer:

Figure below shows the major difference between CISC and RISC processors, CISC emphasizes on hardware complexity, whereas RISC emphasizes on compiler complexity.



Difference between RISC and CISC

RISC	CISC
	
Emphasizes on compiler complexity	Emphasizes on processor complexity

Simple but powerful instructions	Instructions are more complicated
Executes instruction in single cycle	Takes many cycle to execute
Instructions are of fixed length	Instructions are of variable length
Have large set of general purpose registers	Have limited set of general purpose registers
Any register can contain either data or an address	Dedicated registers for specific purpose
Separate load and store instructions transfer data between the register and external memory.	MOV instructions can be used to transfer between register and memory.