Hanumantha
Dept. of CSE
4th Sem

Subject Code:- 21CS43

Subject:- MICROCONTROLLER & EMBEDDED SYSTEM

Module-01:- ARM Embedded System & ARM Processor fundamentals

Topic -01

Microprocessors versus Microcontrollers

Questions

1. Differentiate Microprocessor and Microcontroller

Ans:-

Microprocessor	Micro controller
* Microprocessor contains ALU, General purpose Register, Stack pointer, program counter, clock Timing circuit & Interrupt circuit.	* Micro controller contains the circuitry of Micro-processor & in Addition it has built-in- ROM, RAM, I/O Devices, Timers & counters.
* It has Many Instructions to Move data Between Memory & CPU.	* It has one or Two instructions to Move data Between Memory & CPU.

* I * It has one or two bit handling Instructions.	* It has Many bit handling Instructions.
* Access times for Memory & I/O devices are more	* Less Access times for Built-in Memory & I/O Devices.
* Microprocessor Based System Requires more Hardware.	* Microcontroller Based System Requires less hardware Reducing PCB Size & Increasing the Reliability.
* Microprocessor Based System is more flexi-ble in design point of view.	* Less Flexible in design point of view.
* It has single Memory map for data & code.	* It has Separate Memor-y map for data & code.
* Less Number of pins are Multi-functioned	* More Number pins are Multi-functioned.

Topic-02

The RISC Design Philosophy.

Questions

2. Explain Briefly The RISC Design philosophy.

Ans:-

RISC - Reduced Instruction Set Computer.

Features of RISC

* It is intended as a contrast with CISC Machines which are Complex Instruction Set computers
* Most RISC Processors Use Hardwired control.
* The Most of the RISC processors Use 32-bit Instructions.
* They have very few Instructions.
* The Instructions are predominantly Register-Based.

* The limited (3 to 5) Addressing modes are used by these processors.

* The Memory Access Cycle is Broken into pipelined Access operation.

* This Involves the Use of Caches of and working Registers.

* A large Register file & Separate Instruction & data caches are Used.

* Using Hardwired control the clock Cycle per Instruction (CPI) are Reduced to 1. for Most RISC Instructions.

* RISC Architecture is Used in ARM cores.

RISC Design

* RISC processor are Designed to Execute Simple But powerful Instructions within a Single Cycle at a high clock Speed.

* RISC processors follow the Four Major Design Rule

1. Instructions

* Reduced Number of Instructions : provides Limited number of Instructions, which simplifies the design of control unit.

* Simple Instruction format with fixed Instruction length.
    * The Instruction length is Aligned on word Boundaries

* Hardwired Instruction
    * There is no need for micro instructions.
    * The Machine Instructions can be Hardwired.
    * These Instructions are Executed faster than the Instructions Implemented with micro instructions.

* One Instruction per cycle
  * RISC processor Execute one Instruction in a single cycle.
  * In RISC processors, there is an one Instruction per machine cycle

2. Pipelines

* The CPU contains several Independent Units that work in parallel.
* One of them fetches the Instructions & other ones Decode & Execute them.
* All RISC processors provide this pipelining feature.
* RISC processor most Instructions are Register to Register.

There are Two phase

1. Instruction Fetch(I)
2. Execute(E).

* Incase of load & store Instructions Three phases are required.

1. Instruction Fetch (I)
2. Execute (E)
3. Data Transfer (D).

3. Register

* Register to Register Operations

* ~~Regi~~ RISC processors have a large number of general purpose Registers & they Use Efficient compiler Technology to optimize Register usage.
* It is the most Important characteristics of RISC processor.
* This Architecture Encourages the optimization of Register.

4. Load - Store Architecture

* RISC processor Operate on data held in Register.

* Load & Store Instructions Transfer data Between the Register Bank & External Memory.

3. Give the comparaison Between RISC and CISC

Ans:-

Characteristics	RISC	CISC
1. Instruction Size	Fixed	Varies
2. Instruction Length	4 - Bytes	1, 2, 3 or 4 Bytes
3. Number of Instruction	Less	more.
4. Instruction decoding	Easy (quick to decode.	Serial (slow) to decode.
5. Instruction Sementics	Almost Always one simple operation.	Varies from simple to complex. possibly many dependent operations per Instruction.

3. Give the Comparaeison Between RISC and CISC

Ans:-

Characteristics	RISC	CISC
1. Instruction Size	Fixed	Varies
2. Instruction Length	4 - Bytes	1, 2, 3 or 4 Bytes
3. Number of Instruction	Less	more.
4. Instruction decoding	Easy (quick to decode.	Serial (slow) to decode.
5. Instruction Sementics	Almost Always one simple Operation.	Varies from simple to complex. possibly many dependent operation -s per Instruction.

6. Addressing mode.	complex Addressing modes are synthesized in software.	Support complex Address-ing modes.
7. Instruction Execution Speed.	Medium	Slow.
8. Instruction Execution	By Harware. Simple Instructions taking one By cycle.	By Micro program. Complex Instructions taking multi-ple cycles.
9. Registers	many. General purpose	Few. May be special purpose.
10. Memory References	Not combined with operat-ions i.e load/store Archite-cture.	combined with operations in many different types of Instructions.
11. Harware	Simple	Complicated.

12. Hardware design focus	Take the Advantages of Implementations with one pipeline & no microcode	Take Advantage of micro-code Implementations.
13. Memory Access.	Rarely	Frequently.
14. Instruction format	Regular, consistent placement of fields	Field placement varies.
15. Pipelined	Highly pipelined.	Not pipelined or less pipelined.
16. Conditional Jump	Can be Based on a Bit Anywhere in Memory	Conditional Jump is usually Based on Status Register Bit.
17. Compilers	complicated	simple
18. Examples	Intel x86, motorola 6800 Series	ARM 8051, ATMEL, AVR, etc.

Topic-03

The ARM Design philosophy

Questions

4. Explain Briefly the ARM Design philosophy.

Ans :-

The Features of RISC which are Accepted By ARM processors

* A Large Uniform Register file
: ARM processor contains a large number of Register like a RISC.

2. A load - Store Architecture

* ARM processor Uses a RISC Architect -ure.

* It contains a large numbers of Registers

* The Instruction Set contains Separate load & Store Instructions for Transferring data Between The Register Bank & External Memory.

* Simple Addressing Modes, with all load/Store Addresses Being determined from Register contents & Instruction fields only.

4. Uniform & fixed - length (32-bit) Instruction fields

* ARM processor Instruction Set contain a Reduced number of instructions.

* These Instruction perform Simple operations which can be Executed in a ~~Single~~ Single Cycle.

* Each Instruction is a fixed length.

* The Uniform & fixed - length Instruction fields Simplify the Instruction decoding.

5. Three - Address Instruction Format

* Most Instruction of RISC & ARM processor have three Address Instruction formats.

* i.e Two Source Operands are Stored in two different Address Locations & Third Operand in a third Address Location.

The Features of RISC which are Rejected By ARM processors

1. Register window.

* The main problem with Register window is the large chip Area occupied by the large number of Registers.

* This feature is Rejected by ARM processors to Reduce cost.

2. Delayed Branches

* when Branch Instruction Appears in a program & hence in a pipeline, a delay slot is created.

* This causes pipeline problems since this is the disturbance to the smooth flow of Instructions.

* This delay slot is filled with some useful Instruction. which in most cases will be Executed.

* Original ARM does not use delayed Branch since they make Exception handling More Complex.

3. Single Cycle Execution of all Instructions

* ARM processor Executes most of the data processing Instruction in a single cycle.

* However, many other Instructions need multiple clock cycle for their Execution.

* More than one clock cycle Becomes the Requirement Even for a simple load & store Instruction.

* Atleast two memory Accesses one for the Instruction & one for the data are needed.

5. Which are the silent features of ARM Instruction set?

Ans:-

1. Variable Cycle Execution for Certain Instructions.

* Some ARM Instructions like Load -Store - Multiple of ~~Registers~~ Instr-uctions vary in the no of Execution Cycles depending upon the no of Registers Being Transferred.

* Load - Store - multiple Instruction Transfer data on Sequential Memory Address. which

* ~~Mult~~ Multiple Register Transfers also Improve the code density.

2. Inline Barrel Shifter to Improve Core performance & Code density

* The ARM Arithmatic Logic Unit has a Barrel Shifter that is capable of ~~Shift~~ Shift & Rotate Operations.

* This Inline Barrel Shifter preprocesses one of the Input Registers Before it is used by an Instruction.

* This Expands the capability of many Instructions to Improve Core performance & code density.

3. ~~Thumb~~ Thumb 16-bit Instruction Set

* The Thumb instruction set consists of 16-bit Instructions that act as a Compact shorthand for a subset of the 32-bit Instructions of the standard ARM.

* These Instructions permit the ARM core to Execute either 16 or 32-bit Instructions.
* These Instruction set Improve code density.

4. Conditional Execution

* These Instruction are Executed when a specific condition has been satisfied.

* This features Improves performances & code density by Reducing Branch Instructions.

5. Enhanced Instructions

* ARM Instruction set also supports the enhanced Digital Signal Processor (DSP) Instructions.

* These Instructions allow ARM processor to serve as a combination of a processor plus a DSP

TOPIC-04

Embedded System Hardware

Questions

6. With Neat Sketch Explain Briefly the ARM processor Based Embedded System Hardware.

(or)

With a neat Diagram Explain the Four Main Hardware components of an ARM Based Embedded Device.

QP Jul/Aug-2022

(or)

Explain the Architecture of a typical Embedded device Based on ARM core with a neat Diagram.

Ans:-

* An Embedded System is a computer system with a dedicated function within a larger Mechanical/Electrical System,

often with Real-Time computing Constraints.

It's characteristics

* low power consumption.
* Small size.
* Rugged Operating Ranges
* low per unit cost.

* Embedded system can control many Different Devices.

* These Devices Use a Combination of Software & Hardware Components.

Following Embedded device Based on an ARM Core device Consists of Four Main Hardware components

1. ARM processor

* It controls the Embedded device.
* The Main Component of an ARM processor is a ARM core.

* ARM core is an Execution Engine that processes Instructions & Manipulates data.

* The other components ARM processor Include Memory Management & caches.

* They allow ARM processor to Interface it with a Bus.

2. <u>Controllers</u>

* Two commonly found Controllers in the Embedded device are interrupt & Memory controllers.

3. <u>Peripherals :-</u>

* They Include Serial I/O, parallel I/O, Timers/ Counter & Clock circuits.

4. <u>Bus:</u>

* It is an Internal Bus Used to Communicate B/w different Components of the device.

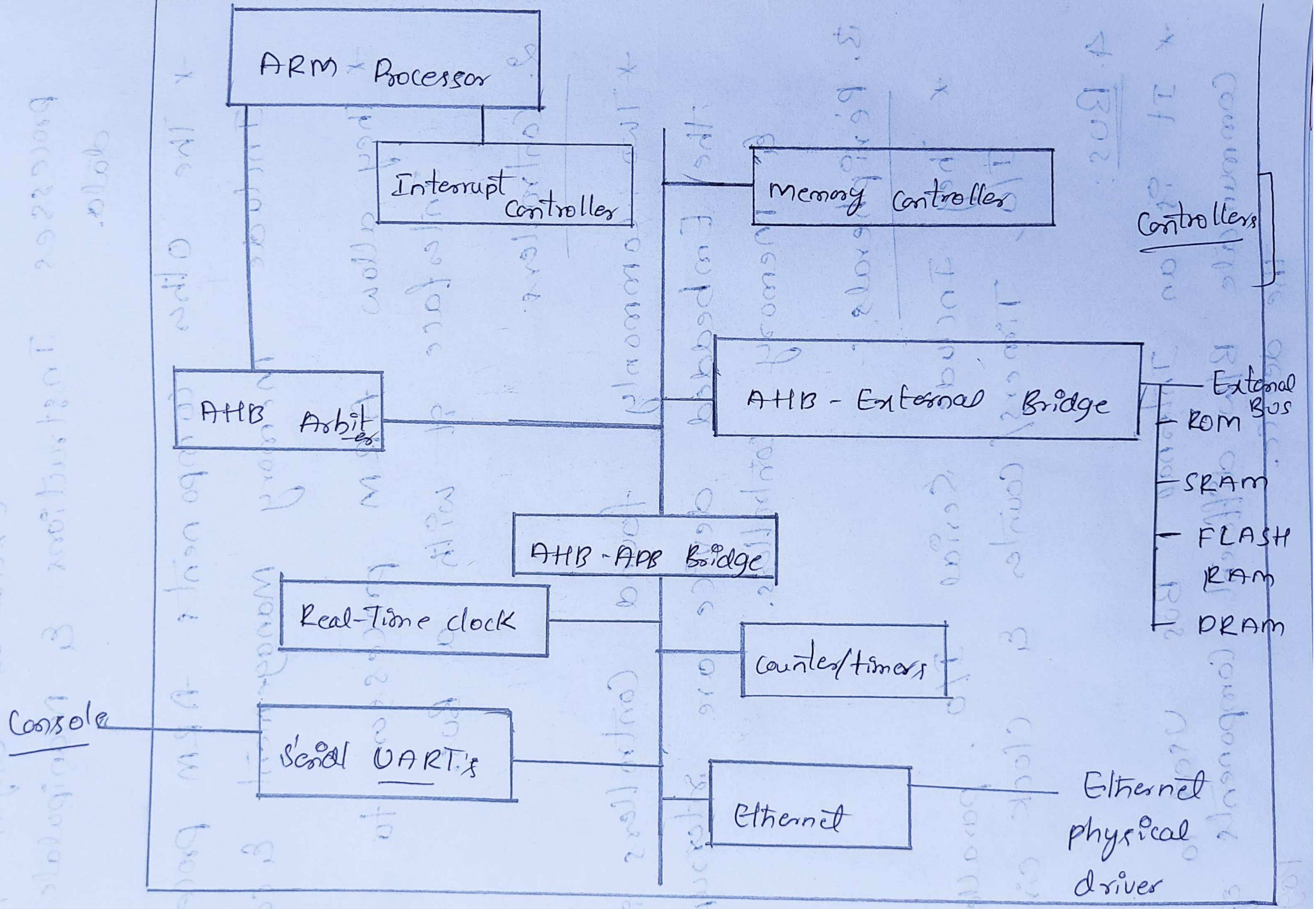

fig:- Typical Embedded device Based on an ARM core

TOPIC - 05

Embedded System Software

Questions

07. Explain Briefly the ARM processor Based Embedded System Software.

(or)

With a neat Diagram Explain the different Software components of an Embedded System.

(or)

Explain the Structure of ARM Cross development kit.

Ans:-

* Embedded System Software drives processor & Associated Hardware components to provide Required System Functionality.

Typically Four Software Abstraction Levels are Used to control an Embedded System.

1. Inilization code

* It is the first code Executed on the Board.

* It configures the Essential parts of the Board Before handling control over to the operating System.

* Usually It is a firmware

2. Device Drivers

* The Software that Directly Interface with & Controls Embedded System hardware is called a device driver

* Different Types of Hardware will have different device driver Requirem-ents that need to be met.

3. Operating System

* An Embedded Operating System is a type of operating system that is Embedded & Specially configured for a certain Hardware configuration.

* It provides an Infrastructure to Control Applications & manage Hardware System Resources.

4. Application

* It performs one of the task required for a device.

* There may be multiple Applicat-ions Running on the same device.

* Application cannot Run on itself But they dependent on the Operating System.

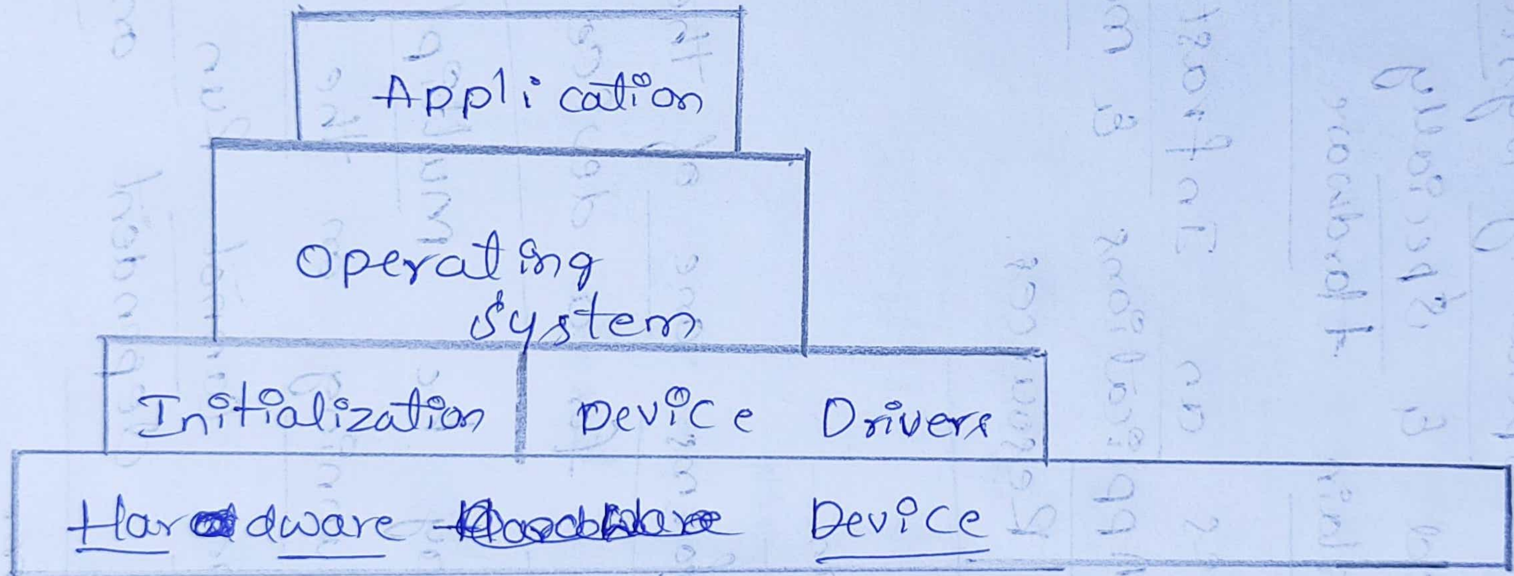

fig:- Four Software Abstraction Levels ~~of~~ Used in Embedded System

08. Explain ARM Core dataflow Model. with a neat Diagram.

Q.P July/Aug 22

Ans:-

* The ARM core dataflow model is Von Neumann Implementation of the ARM.

* Since ARM processor is Basically a RISC processor, it uses a load-store Architecture

LOAD:- This Instruction copies data from memory to Registers in the processor Core.

STORE:- This Instruction copies from Registers in the processor Core to Memory.

Data Bus

* The Data Enters the ARM core through the data Bus.

* The Data is either in the form of an Instruction Opcode (or) Data Item.
* This is in contrast with Hardvard Architecture which uses two different Buses.

Instruction Decoder

* This Unit decodes the Instruction opcode Read from the Memory & then the Instruction is Executed.

Register file

* This is a Bank of 32-bit Registers Used for storing data items.

Sign Extend

* The ARM core is a 32-bit processor.
* So most Instruction of the ARM processor Treat Registers as holding signed (or) unsigned 32-bit values.

* when the processor Reads signed 8-bit (or) 16-bit numbers from memory the sign Extend Hardware converts these numbers to 32-bit values & then places them in a Register file.

ALU & MAC

* The ALU (or) MAC reads the operands values from Rn & Rm Registers via A and B Buses Respectively. performs the operation & stores the computed Result via internal C Bus in destination Register Rd & then to the Register file.

Address Register

* This holds the Address Generated by the load & store Instructions and places it on the Address Bus.

Barral shifter

* A wide Range of Expressions & Addresses can be calculated using the Barrel shifter & ALU

Incrementer

* For load & STORE Instructions, the incrementer updates the contents of the Address Register before the processor core reads or writes the next register value from or to the consecutive memory location.

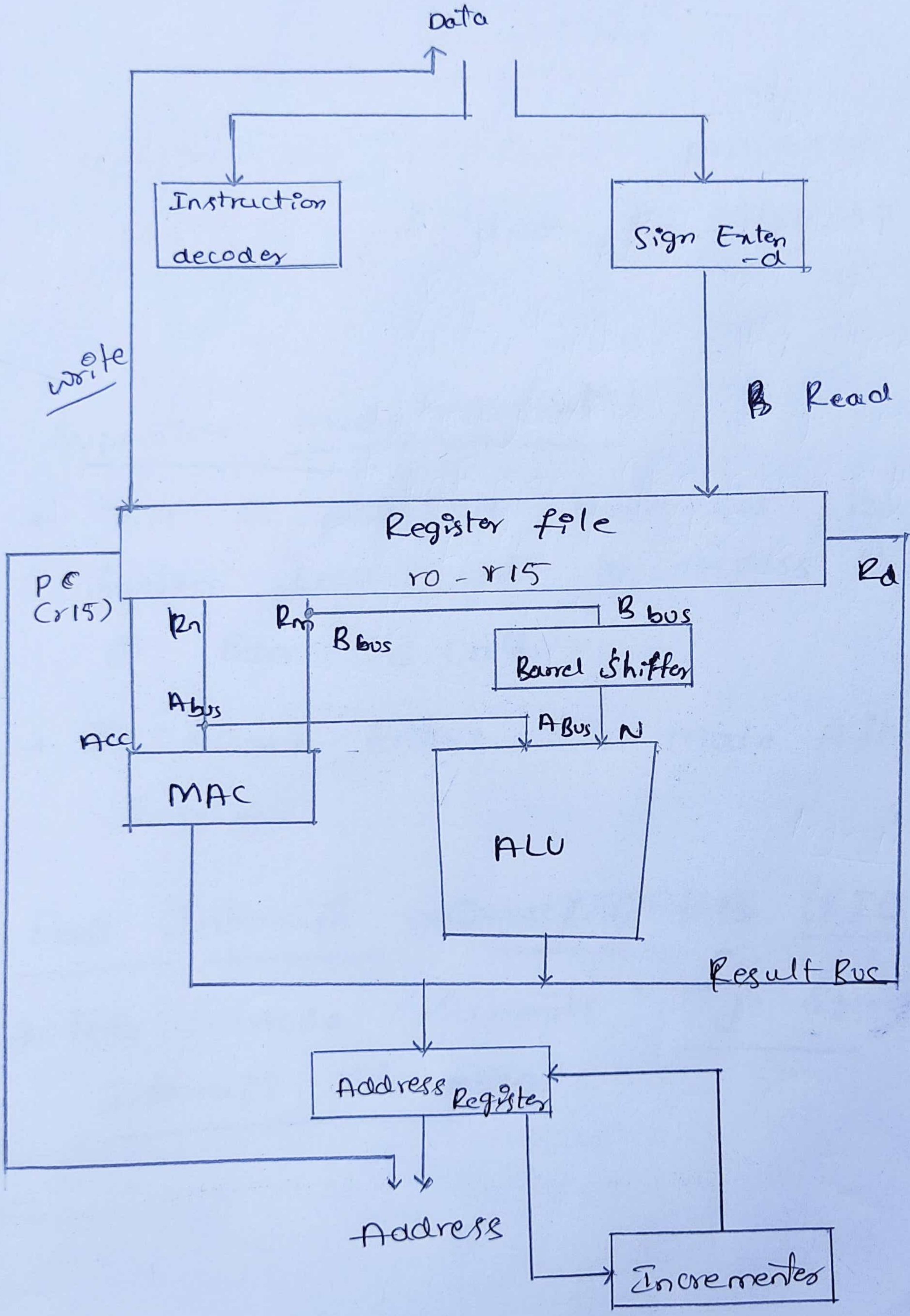

fig:- ARM Core Dataflow Model

Chapter-02

Topic-01 : Registers

Questions

Q1. Explain the Different processor Modes provided By ARM7

Ans:-

1. Supervisor Mode (Default)

* This is protected mode for Running System Level code to Access Hardware or Run O.S Call.

* The ARM7 Enters this Mode After Reset.

2. Fast Interrupt reQuest ~~(FIRQ)~~ (FIQ)

* This mode supports high speed Interrupt Handling.

3. Interrupt ReQuest (IRQ)

* This mode supports all other Interrupt sources in a System

4. Abort

* If an instruction (or) data is fetched from an invalid Memory Location an Abort Exception will be generated.

5. Undefined

~~* This mode is Used to Run the Application code.~~

~~* In the User mode we Cannot~~

* If a fetched opcode is not an ARM instruction, an Undefined instruction Exception will be Generated.

6. User

* This mode is Used to Run the Application code.

* In the User mode we Cannot Change the contents of CPSR & modes can only be changed when an Exception is generated.

* This mode is also known as Unprevileged mode

7. System :-

* This mode is used for Running Operating System tasks.

* It uses the same Registers as User mode

10. Explain Briefly the Active Registers Available in User mode.

~~Answer~~

(or)

Explain the programmer's model of ARM processor with complete Register set Available

(or)

Explain Registers Used Under various modes.

Ans :.

* The ARM processor has a Total of 37 Registers.

* All Register are 32-bit wide.

classified into Two Groups

1. General Purpose Registers.
2. Special purpose Registers.

1. General purpose Registers

* Registers r0 to r12 are Used as General Purpose Registers.

* The General Purpose Registers Hold either Data or an Address.

2. Special Purpose Registers

* Registers r13 to r15, CPSR & SPSR are the Special Purpose Registers.

* In User Mode Registers r13 to r15 are labeled as r13, SP, r14, lr & r15 pc Respectively to differentiate them from other Registers.

The Functions of These Registers

- **Stack pointer (r13 sp) :-**
  * Register r13 is the stack pointer.
  * It stores the top of the stack in the current processor mode.

- **Link Register (r14 lr)**
  * Register r14 is the Link Register.
  * The Processor stores the Return Address in this Register when a subroutine is called Link Register

- **Program counter (r15, PC) :-**
  * Register r15 is the Program counter & stores the Address of the next Instruction to be fetched from the Memory by the Processor

- **Unbanked Registers r0- r7**
  * Register r0 to r7 are Unbanked Registers
  * This means that each of them Refers to the same 32-bit physical Register in all processor modes.

• Banked Registers

* Register r8 to r14 are Banked Registers
* The Physical Register Referred to by each of them depends on the Current Processor mode.
* where a particular physical Register is intended without depending on the current processor mode, a more specific name is Used.
* Almost all Instructions allow the Banked Registers to be Used wherever a general-purpose Register is allowed.
* Supervisor mode has Banked Register r13_svc, r14_svc & spsr_svc
* Abort mode has Banked Register r13_abt, r14_abt, & spsr_abt.
* Register r8 to r12 have Two Banked physical Registers each.
* Register r13 & r14 have six Banked physical Registers each.

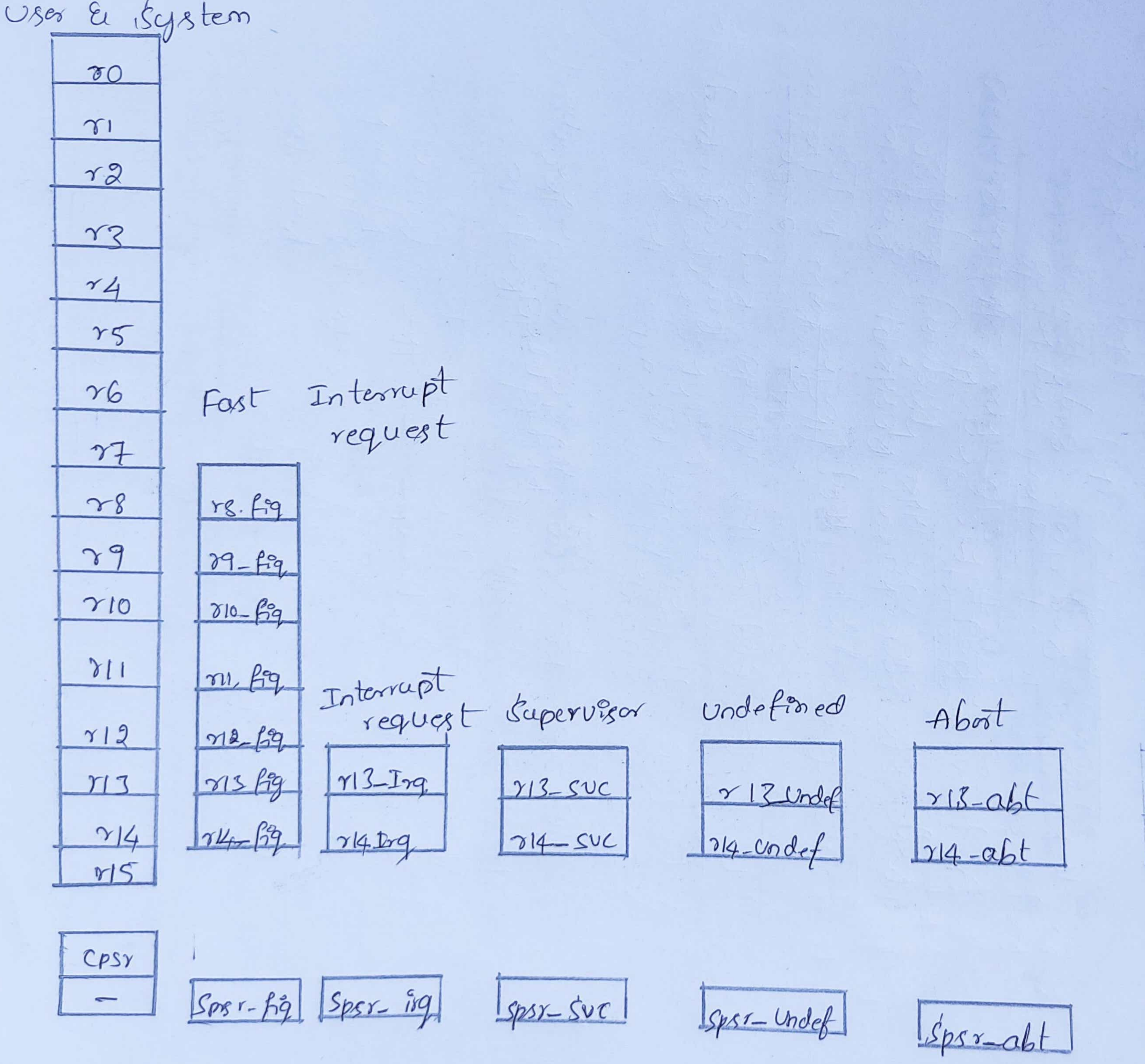

fig:- Programming model of ARM processor

Topic — 02

Current Program Status Register (CPSR)

Questions

1. Explain the various field in Current Program Status Register (CPSR) with neat Diagram.

(or)

Draw the Basic Layout of a Generic program status Register & Briefly Explain the various fields.

Ans :-

* The Current Program Status Register (CPSR) is Accessible in all processor modes.

* It contains condition code flags, Interrupt disable bits, the current processor mode & other status & control Information

* Each Exception mode also has a Saved Program Status Register (spsr).

* That is used to preserve the value of the cpsr when the Associated Exception occurs.

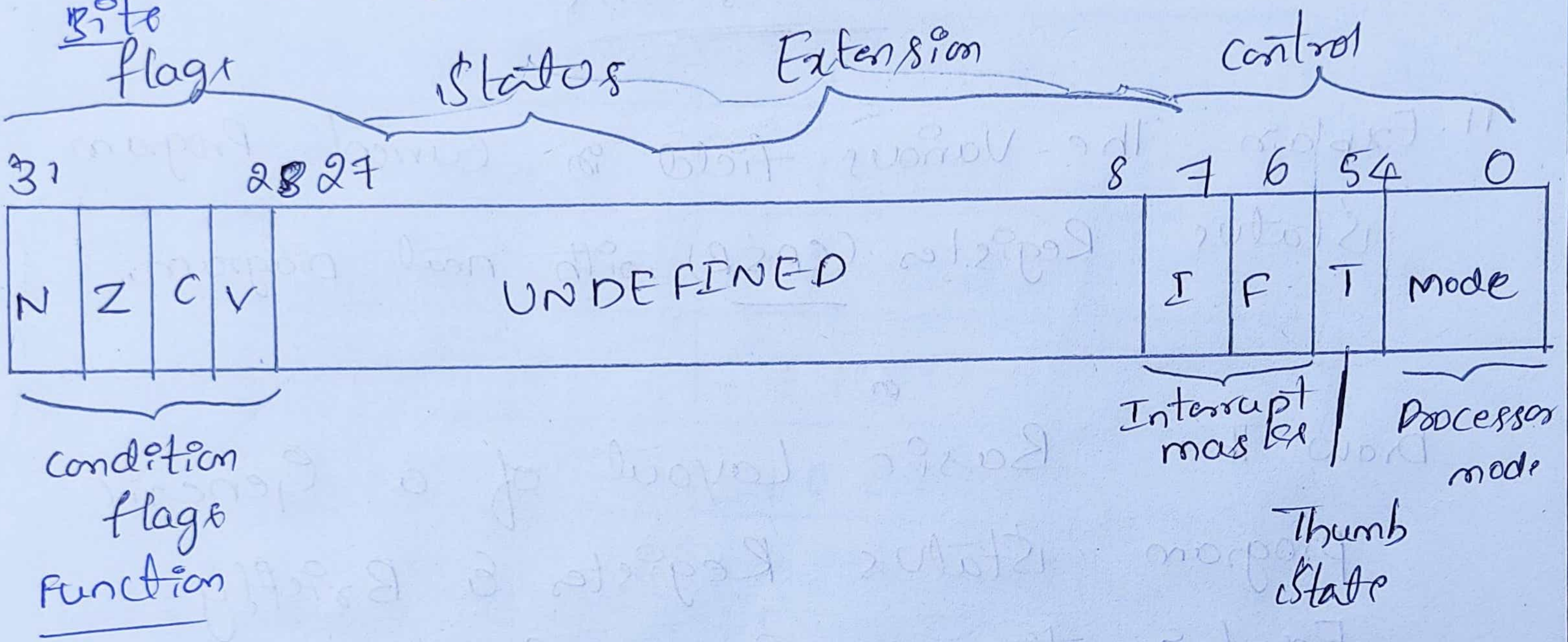

fig.- Format of cpsr & spsr.

Control flags (Bits 0-7)

* The control Bits change when an Exception arises & can be altered by Software only when the processor is in a privileged mode.

Bits 0-4 (mode select Bits) : Processor modes

* These bits determine the processor mode.

Processor mode	Mode selects Bits
Abort	10111
Fast Interrupt request	10001
Interrupt request	10010
Supervisor	10011
System	11111
Undefined	11011
User	10000

Table :- Processor modes

Bit 5 (Thumb state Bit)

* This Bit gives the state of the core.
* The state of the core determines which Instruction set is Being Executed.
* There are Three Instruction sets ARM, Thumb & Jazelle.
* one of the three instruction set is active when the processor is in ARM state, Thumb state & Jazelle state Respectively.

Bit 6 and 7 (Interrupt Masks)

* There are Two Interrupts Available on the ARM processor core:
 1. Interrupt Request (IRQ) and
 2. Faster. Interrupt Request (FIQ).

* These are Maskable Interrupts & their masking is controlled by bit 6 and bit 7 of Cpsr.

* Bit 6(F) controls FIQ and bit 7(I) controls IRQ.

Condition Code flags

* These flags in the cpsr can be tested by Most instruction to determine whether the Instruction is to be Executed.

Bit 27 (saturation flag, Q)

* This flag is Available for the ARM processor Cores which include the DSP Extensions.

* Similarly bit 27 of each spsr is Q flag & is used to preserve & Restore the cpsr. Q flag if an Exception occurs.

Bit 28 (overflow flag, V)

* It is set in one of two ways
  * For an Addition/Subtraction, V is set to 1 if signed overflow occured regarding the Operands & Result as two's Complement signed Integers.
  * For Non-Addition/Subtraction V is normally left unchanged.

Bit 29 (carry flag, C)

* It is set in one of four ways
  * For an addition, including the comparision instruction CMN, C is set to 1 if the Addition produced a carry & to 0 otherwise.
  * For a Subtraction, including the comparision instruction CMP, C is set to 0 if the Subtraction produced a borrow, and to 1 otherwise
  * For addition/subtraction that incorporate a shift operation C is set to the last bit shifted out of the value by the shifter.

* For other non addition/subtractions C is Normally left Unchanged.

Bit 30 (Zero flag, Z)

* It is set to 1 if the Result of the instruction is Zero & to 0 otherwise.

Bit 31 (Negative flag, N)

* It is set to bit 31 of the Result of the instruction.

* If this Result is Regarded as a two's complement signed integer, then N=1 if the Result is Negative and N=0 if it is positive or Zero.

other Bits

* other bits in the program Status Registers are Reserved for future Expansion.

Topic – 03

Pipelines.

Q. 12. Explain detail, the ~~Complete ARM~~ pipeline Concept for ARM processor.

Ans:-

* Fetching the next instruction while other instruction is in Execution is called "pipelining"

* ARM7 Uses a simple Three-stage Pipeline

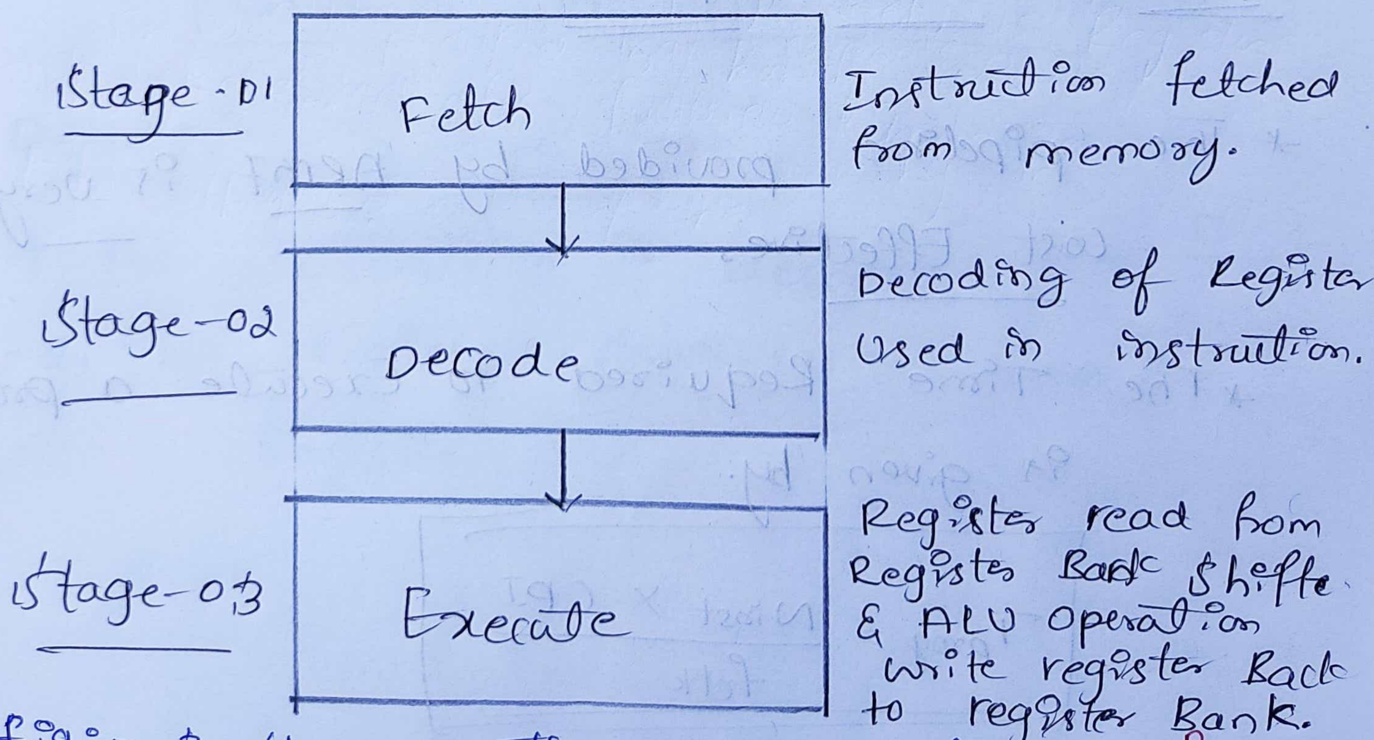

fig:- 3-stages in The ARM7 pipelines

* The Three stages used in the pipelines are

* Fetch :- In this stage the processor fetches the instruction from the memory.

* Decode :- In this stage processor Identifies the instruction which is to be Executed.

* Execute :- In this stage the processor processes the Instruction & stores Result in a Register.

5-stage ARM Pipeline

* The pipeline provided by ARM7 is very cost Effective.

* The Time Required to Execute a program is given by.

$$\boxed{T_{prog} = \frac{N_{inst} \times CPI}{f_{clk}}}$$

where

T_{prog} :- Time Required to Execute a given program.

N_{inst} :- No of ARM instructions Executed in the Program.

CPI :- Average Number of clock cycles per Instruction.

f_{clk} :- Processor clock frequency.

The 5-stages ~~are~~ Pipeline stages are:

* Fetch :- In this stage the Processor fetches instruction from Memory & places in the instruction pipeline.

* Decode :- In this stage
  1. The Instruction is decoded &
  2. The Register Operands read from the Register.

* Execute :- In this stage

1. An Operand is shifted

2. The ALU Result is generated.

3. If the instruction is a load @ store the Memory Address is computed in the ALU.

* Memory :- In this stage, data memory is Accessed if Required.

* otherwise, the ALU Result is simply Buffered for one clock cycle to give the same pipeline flow for all instruction

* Write :-

In this stage the Results Generated by the Instruction are written Back to the Register file including Any data loaded from memory.

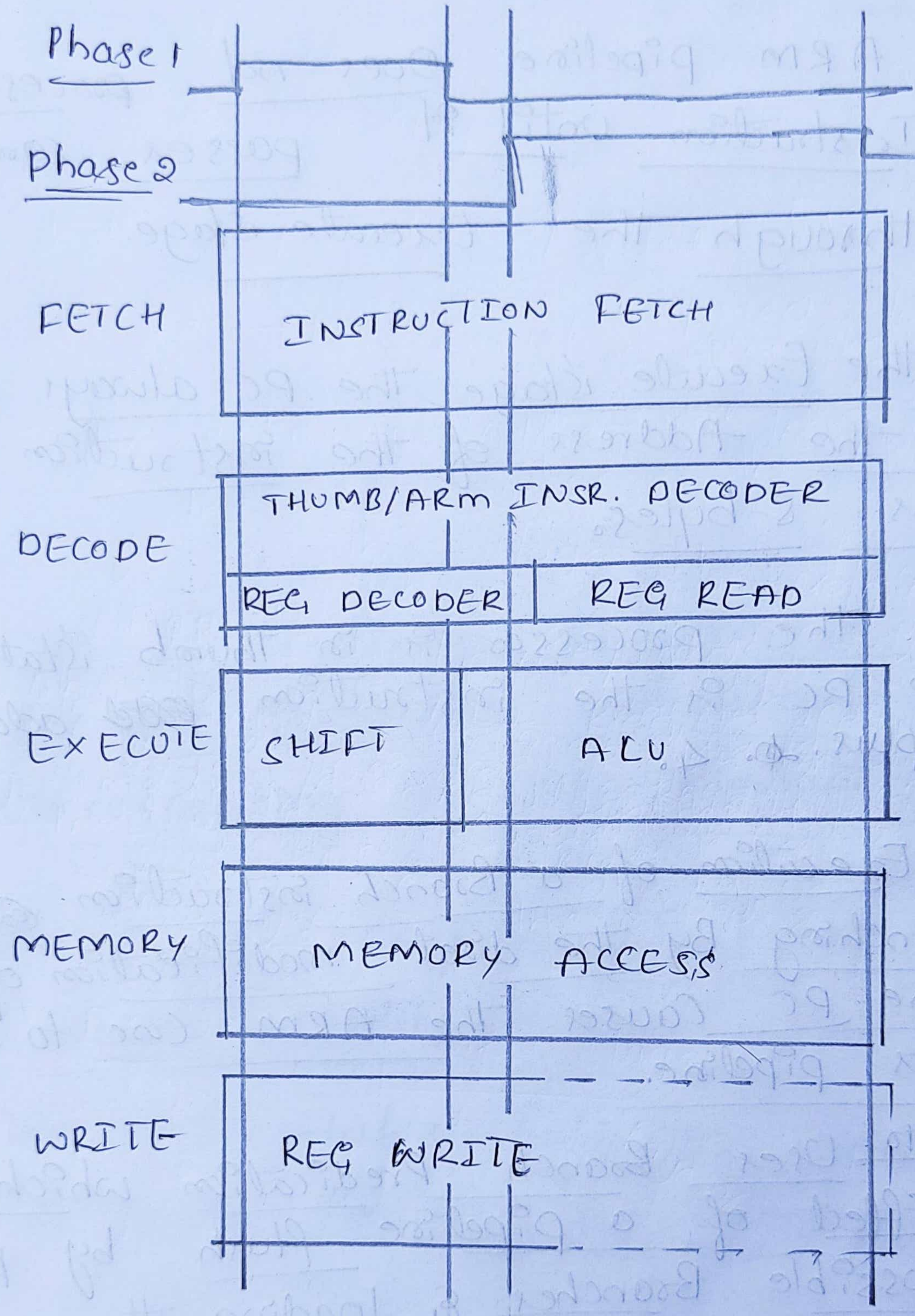

fig:- five stage pipeline

13. Explain Briefly Pipeline Execution Characteristics

Ans:-

* The ARM pipeline Does not process an Instruction until it passes completely through the Execute stage.

* In the Execute stage the PC always points to the Address of the instruction plus 8-bytes.

* When the processor is in thumb state the PC is the instruction ~~...~~ address plus. 4.

* The Execution of a Branch instruction @ Branching By the direct modification of the PC Causes the ARM Core to flush its pipeline.

* ARM10 Uses Branch Predication which Reduces the Effect of a pipeline flush by predicting possible Branches & loading the new Branch address prior to the Execution of the ~~...~~ instruction.

* An Instruction in the Execute stage will Complete Even through an Interrupt has Been Raised.

TOPIC-04

Exceptions, Interrupts & The Vector Table.

Questions

14. Explain Interrupt, Exception & Vector Table.

Ans:-

* When an Exception or interrupt occurs the PC is loaded with specific Address corresponding to the interrupt / Exception.

* The specific Address is known as Vector Address.

* The Vector table holds the Vector Addresses for all the interrupts/ Exceptions.

* In some processors the Vector Table is located at the higher Address in memory, starting at the offset FFFF0000 H.

Exception/Interrupt	Address	High Address
Reset (RESET)	00000000H	FFFF0000H
Undefined Instruction (UNDEF)	00000004H	FFFF0004H
Software Interrupt (SWI)	00000008H	FFFF0008H
Prefetch abort (PABT)	0000000CH	FFFF000CH
Data abort (DABT)	00000010H	FFFF0010H
Reserved	00000014H	FFFF0014H
Interrupt Request (IRQ)	00000018H	FFFF0018H
Fast Interrupt Request (FIQ)	0000001CH	FFFF001CH

Table :- Vector Table

Exception / Interrupts

* **Reset :-**
  - It occurs when power is Applied.
  - In Response to RESET processor Executes Branch Instruction located at Address (0 0000000H) to Transfer program control to the Initialization code.

* **Undefined Instruction :-**
  - It occurs when processor cannot decode the Instruction.
  - In Response processor Executes Branch Instruction located at Address 00000004H.

* **Software Interrupt**
  - The software Interrupt vector is Used when we Execute a SWI Instruction.
  - It is frequently Used to invoke an Operating System Routine.

* Prefetch Abort

- The prefetch abort vector is Used when the processor attempts to fetch an instruction from an Address without the correct Access permissions.

* Data Abort

- The Data Abort Vector is Used when an instruction attempts to Access data Memory, without the correct Access permissions.

* Interrupt Request

- The Interrupt Request Vector is Used by the External hardware to interrupt the Normal instruction Execution flow of the processor.

* Fast ~~Instruction~~ Interrupt Request

- The fast ~~instruction~~ Interrupt vector is Used by External hardware which Requires faster Response Time.

Topic- 05

Core Extensions

Questions

15. What are the different Techniques of Core Extensions.

Ans:-

* The Hardware Extensions to ARM core Improve Performance, manage Resource, & provide Extra Functionality & are Designed to provide flexibility in Handling Particular Applications.

There Are Three ~~Ex~~ Hardware Extensions

1. Cache & Tightly Coupled Memory
2. Memory Management
3. Co-processor Interface

1. cache ~~memory~~ & Tightly coupled Memory

cache Memory

* cache Memory is a small-sized type of ~~vol~~ volatile Memory placed B/w Main Memory & the core.

* It provides High-speed data Access to processor core & stores frequently Used ~~to~~ program & data.

* ARM has Two forms of Cache
  1. single unified cache
  2. separate caches for data & Instruction

Tightly coupled memory

* For Real-Time systems, the Time taken for loading & storing Instruction/data must be predictable

* The Predictable Performance is achieved using a memory called "Tightly Coupled Memory"

* TCM is fast SRAM Located close to the processor core.

* It Requires fixed Amount of clock cycles to fetch Instruction / data & instruction Execution is fast enough to satisfy the needs real-time Algorithms.

2. Memory Management

* Embedded systems mostly use Multiple memory devices.
* ARM processors Use Memory management Hardware to organize these Memory devices & protect the system from Applications Trying to make Inappropriate Access to Hardware.

* ARM Cores have three different Types of Memory Management Hardware.

1. No Extensions - Do not Provide any protection.

2. Memory Protection Unit (MPU) - Provides limited protection.

3. Memory Management Unit (MMU) - Provides full protection

---

3. Co-processors

* The processing features of a Core can be Enhanced by attaching Co-processors to the ARM Core

* The Coprocessors Extends the Instruction set & ~~provide~~ provides Configuration Registers to add processing features.

* If the Coprocessor is not present / doesn't Recognize the Instruction, then the ARM takes an Undefined Instruction Exception.

* The ARM processor core supports several different types of closely coupled coprocessors, Including floating point, SIMD, & System control & Cache maintainance

* Each coprocessor present in an ARM System has a Unique 4-bit ID code.

* One of the Primary Goals of the ARM Coprocessor interface is not to slow down the CPU core.

* 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 ~~the~~ providing a Specilized group of new Instruction.