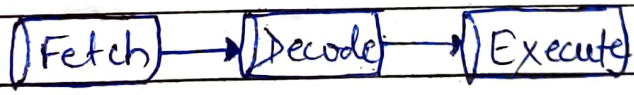
UNIT - 5

1. Define pipeline. Explain 3 stage pipeline execution mechanism in a RISC processor.

→ **Pipeline**: A pipeline is the mechanism a RISC processor uses to execute instru^n. Using a pipeline speeds up the execution by fetching the next instruction while other instructions are being decoded & executed.

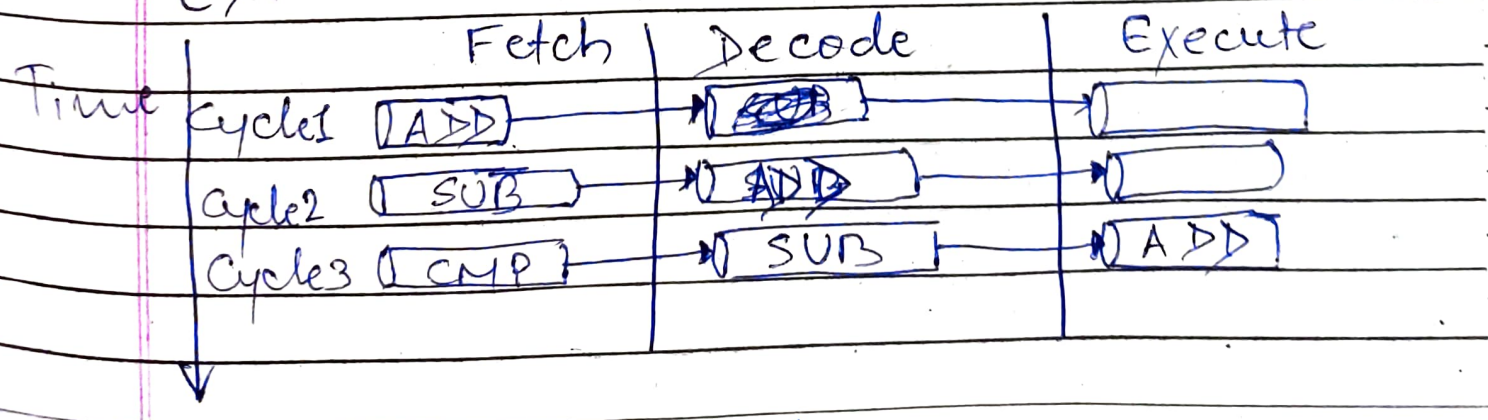
Three stage pipeline:

i) Fetch loads an instruction from memory.
ii) Decode identifies an instruction to be executed.
iii) Execute processes the instruction and writes the result back to a register.



Three stage pipeline

Ex:



Pipeline instruction sequence

2) Discuss with diagram for a simplified data ~~for~~ of a 5 stage RISC pipeline.

Q) RISC design philosphy and design rules.

→ RISC is design philosphy aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed. The RISC philosphy concentrates ~~at delivering~~ on reducing complexities of instr performed by the hardware bcz it's easier to provide greater flexibilities & intelligence in software.

⇒ The RISC philosphy is implemented with 4 major design rules:

1) Instructions: RISC processors have a reduced number of instruction classes. These classes provide simple operation that can each execute in a single cycle. The compiler or programmer synthesizes complicated operations by combining several simple instructions.

2) Pipeline: The processing of instruction is broken down into smaller units that can executed in parallel by pipelines.

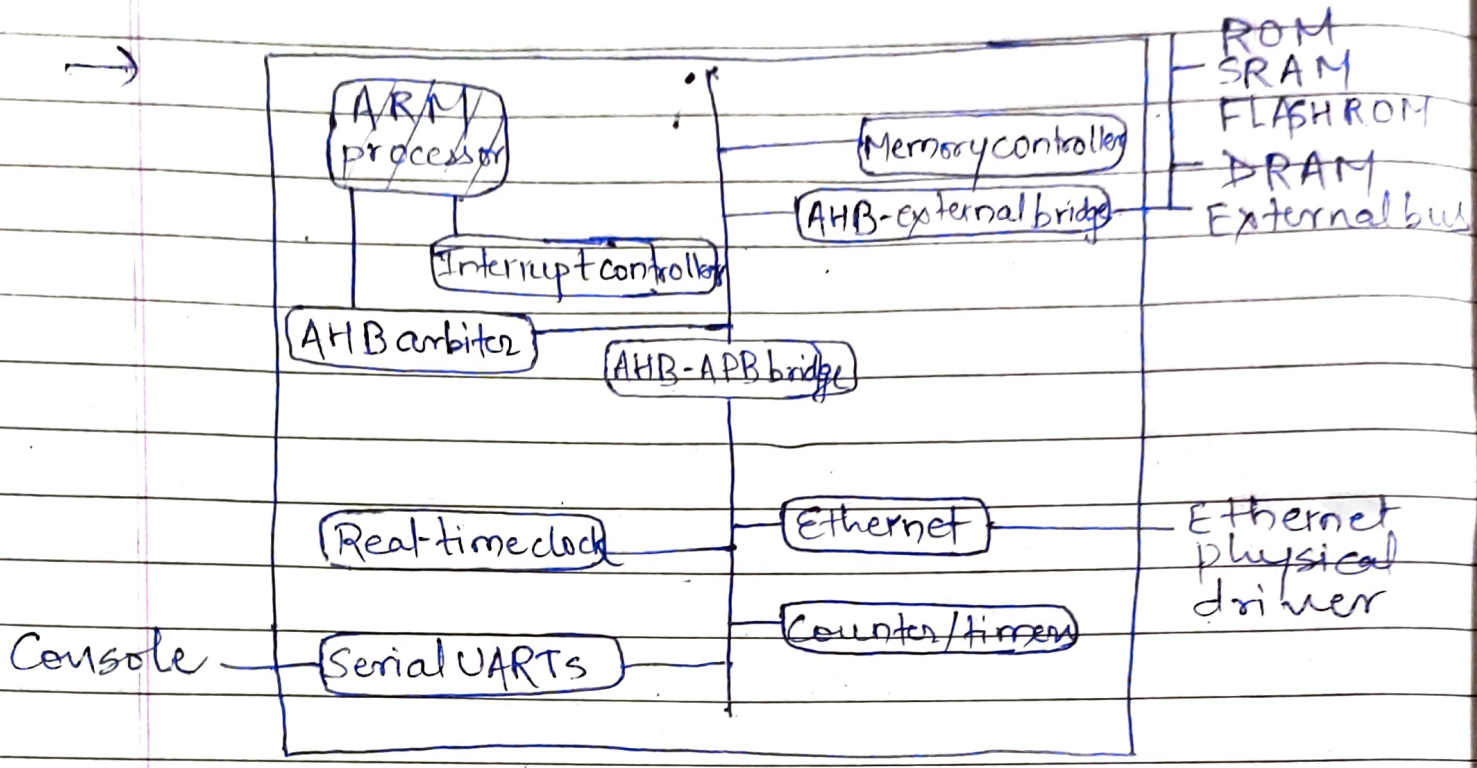
3) Registers: RISC machines have a large general purpose register set. Any register

can contain either data or address. Registers act as the fast local memory store for all data processing operations.

4) Load-store architecture : The processor operates on data held in registers. Separate load & store instructions transfer data b/w register bank & external memory.

Q.) Draw and discuss an ARM based embedded device, as a microcontroller.

→



An ARM based embedded device, as a microcontroller

→ We can separate the device into 4 main components:

i) The ARM processor controls the embedded device. Different versions of the RAM processor are available to suit the desired operating characteristics.

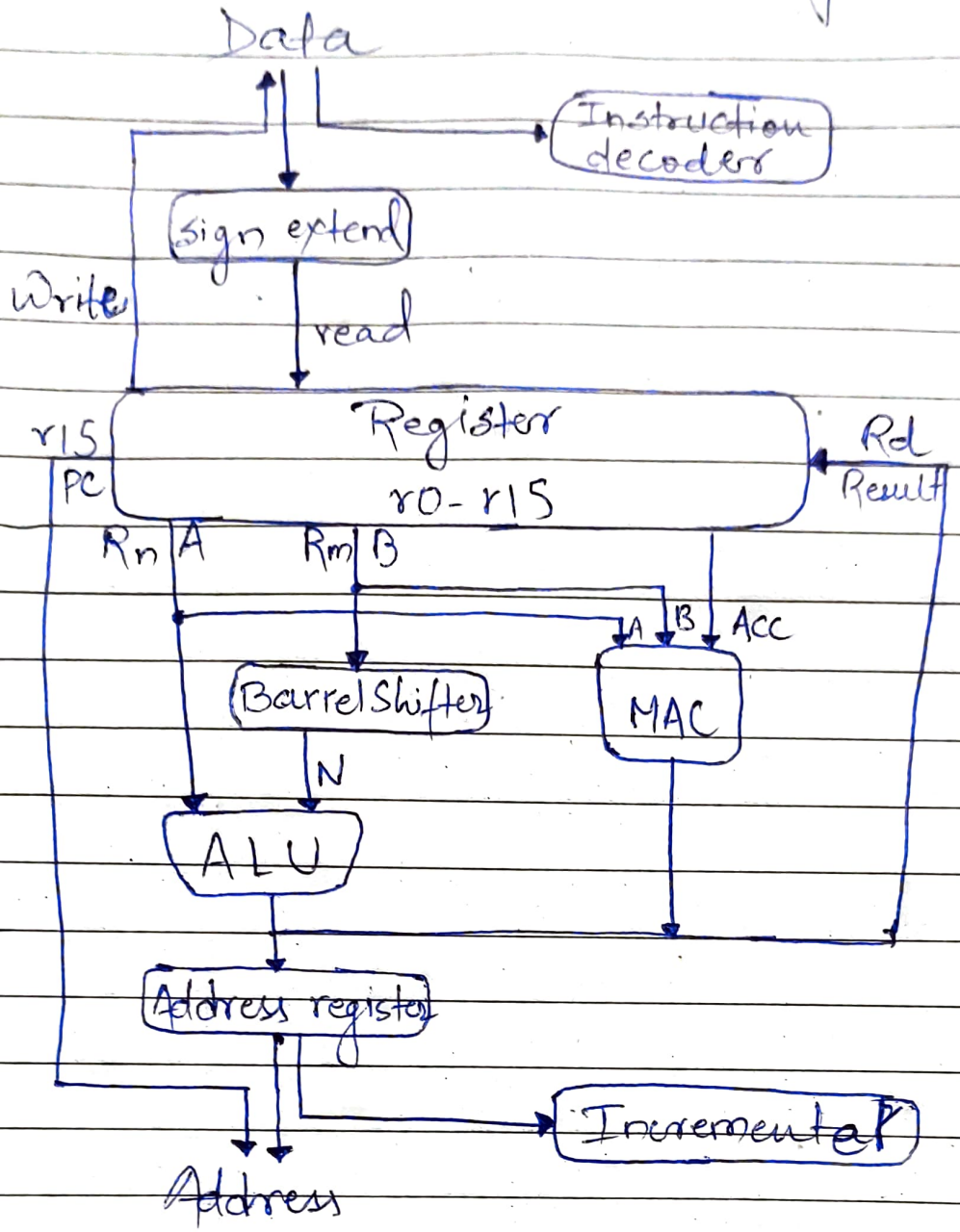
ii) Controllers coordinate important functional blocks of the system. Two ~~important~~ commonly found controllers are interrupt & memory controllers.

→ The peripherals provide all the input output capability external to the chip & are responsible for the uniqueness of the embedded device.

→ A Bus is used to communicate b/w different parts of the device.

Q) Discuss ARM core data flow model.

→



ARM core data flow

The instructor decoder translates instructions before they are executed.

The ARM processor, like all RISC processors uses a load-store architecture. Load instruction copy data from memory to registers in core, & conversly

the store instruction copy data from register to memory.

→ Data items are ~~stored~~ placed in register file - a storage bank made up of 32-bit registers.

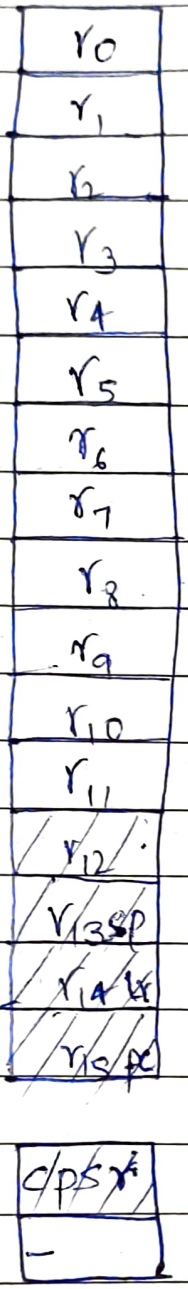
→ ARM instructions typically have two source registers, Rn & Rm and a single destination register, Rd

→ The ALU or MAC takes the register value Rm & Rn from the A & B buses & compute a result. Data processing instructions write the result in Rd directly to the register file. Load & store instr^n use the ALU to generate an address to be held in address register & broadcast on the Address Bus.

→ Rm alternatively can be preprocessed in the barrel shifter before it enters the ALU.

Q) Discuss with a neat diagram for ARM's register organization.

→



- Registers available in user mode

A protected mode normally used when executing applications.

There are upto 18 active registers: 16 data register & 2 processor status

registers. The data registers are visible to programmers as r_0 to r_{15}.

→ r_{13}, r_{14}, r_{15} are assigned to a particular task.

- r_{13} is traditionally used as the stack pointer (sp) & stores the head of the stack in current processor mode.

- r_{14} is called the link register (lr) and is where the core puts the return address whenever it calls a subroutine.

→ r_{13} & r_{14} can also be used as general purpose register. But it's dangerous to use r_{13} as a general register when the processor is running any form of operating system bcz OS often assumes that r_{13} is always points to a valid stack frame.

→ In addition to these, there are 2 program status register: CPSR & SPSR.

Q) Conditional flags of ARM processor.

→

Flag	Flag name	set when
Q	Saturation	result causes overflow
V	overflow	the result causes signed overflow
C	carry	result causes unsigned carry
Z	zero	result is zero
N	negative	bit 31 of result is 1

Q) Addressing modes for load-store.

→ There are different modes for addressing memory. These modes incorporate one of the indexing methods: preindex, preindex with writeback, postindex.

Addressing modes & index method

Preindex with immediate offset
Preindex with register offset
Preindex with scaled register offset
Preindex writeback with imm. offset
Preindex writeback with register offset
Preindex writeback with scaled register offset
Immediate postindex
Register postindex
Scaled register postindex

Addressing mode for single-register load store

Addressing mode for load store multiple instructions

Addressing mode	Description
IA	increment after
IB	increment before
DA	decrement after
DB	decrement before