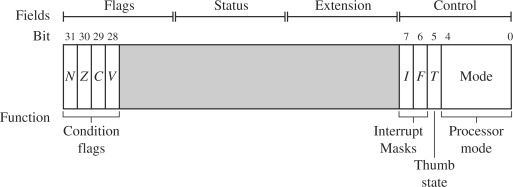
**11) Current Program Status Register (CPSR)**

The ARM architecture utilizes the CPSR (Current Program Status Register) to oversee and manage the processor’s internal operations. This register is 32 bits wide and resides within the core’s register set. Its structure is divided into four 8-bit fields: flags, status, extension, and control. However, in most current ARM implementations, only the flags and control fields are actively used—status and extension are reserved for future enhancements.

- The flags field holds condition flags.

- The control field includes the processor’s current mode, execution state, and interrupt masks.

Some ARM variants feature additional bits. For example, the J bit—used in Jazelle-enabled processors for 8-bit instruction execution—exists within the flags field. It is expected that future versions of the ARM core may include more control bits to support new functionalities.

Processor Modes in ARM:

The processor operates in different modes to manage access rights and control flow:

- Privileged modes allow both read and write access to the CPSR.

- Non-privileged mode (user mode) can only read from the control field, but still read and write the condition flags.

ARM defines seven processor modes:

Privileged modes:

1. Abort mode – triggered during failed memory access attempts.

2. FIQ (Fast Interrupt Request) – handles high-priority interrupts.

3. IRQ (Interrupt Request) – deals with standard interrupts.

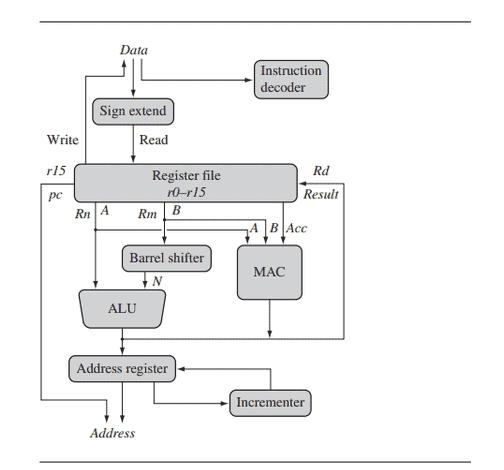
4. Supervisor mode – entered after reset; typically used by operating systems.

5. System mode – similar to user mode but with full access rights.

6. Undefined mode – used for unsupported or illegal instructions.

Non-privileged mode:

* 1)User mode – for running applications with restricted access.

**12) ARM Core Dataflow Model**

The ARM processor is built around a dataflow model that efficiently channels instructions and data through the core using various buses and operational blocks. In a Von Neumann architecture (used by some ARM cores), both instructions and data travel over a shared bus. In contrast, Harvard architectures use separate buses.

When data enters the core through the data bus, it could be either an instruction or a data item. The instruction decoder interprets the incoming instructions, each belonging to a defined instruction set.

ARM follows a load/store architecture, meaning:

- Load instructions transfer data from memory to registers.

- Store instructions move data from registers back to memory.

Data processing happens only in registers, not directly in memory. This architecture ensures that all computations are done within the processor’s register file, which consists of multiple 32-bit registers. When smaller values (like 8-bit or 16-bit) are read from memory, sign-extension logic converts them to 32-bit format before processing.

Most ARM instructions involve:

- Two source registers: Rn and Rm.

- One destination register: Rd.

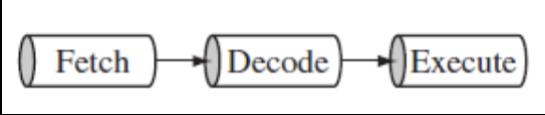
The values are fetched through internal buses A and B, processed using either the ALU (Arithmetic Logic Unit) or MAC (Multiply-Accumulate) unit, and then stored in Rd.

In load/store operations, the ALU computes memory addresses, while the barrel shifter optionally modifies Rm before it’s used—enabling complex address and expression calculations.

After execution, results flow back into the register file via the result bus. For memory operations, the incrementer updates the address register to prepare for the next memory access. Instruction execution continues until halted by an interrupt or exception.

**13) Pipeline in ARM Processors**

Pipelining is a fundamental concept in RISC processors like ARM, enabling parallel execution by dividing instruction processing into multiple stages.

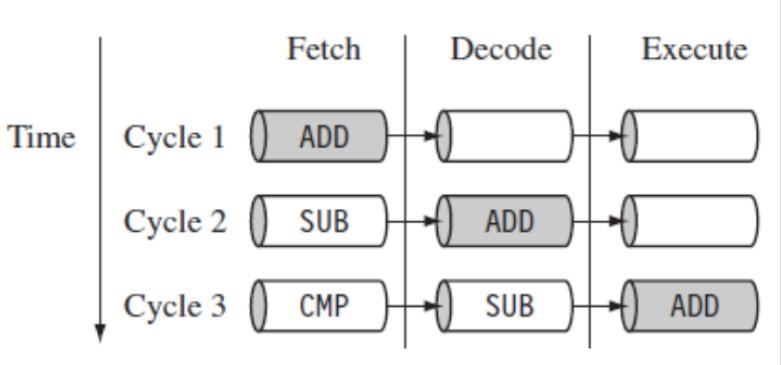


ARM7 Three-stage Pipeline The above Figure shows a three-stage pipeline:

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.



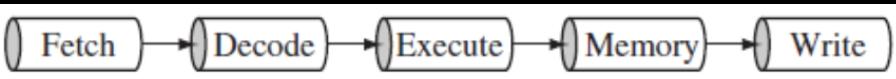
For example, while one instruction is being executed, another is being decoded, and a third is being fetched—this overlapping execution significantly boosts performance.

The pipeline fills as instructions flow through it, eventually reaching a point where one instruction can be completed every clock cycle.

However, longer pipelines mean:

- Higher clock frequencies and better throughput.

- Increased latency and potential data hazards (dependencies between instructions).



ARM9 expands to a five-stage pipeline by adding:

- A memory stage – to access memory.

- A writeback stage – to store results back to registers.

This enhances instruction throughput by around 13% over ARM7.



ARM10 takes it further with a six-stage pipeline, achieving approximately 34% more throughput compared to ARM7. The extra stages enable higher performance but also increase complexity and latency.