Unit VI- Current Trends in Processor Architecture

RISC Design Philosophy

Features of RISC

- **RISC** is an acronym for **Reduced Instruction Set Computer.**
- 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 operations.
- This involves the use of caches and working registers.
- A large register file and separate instruction and data caches are used. This benefits internal data forwarding and eliminates unnecessary storage of intermediate results.
- Using hardwired control, the clock Cycles Per Instruction (CPI) are reduced to 1 for most RISC instructions. This is the advantage of having all instructions of equal size.
- RISC architecture is used in ARM cores.

RISC Design

- RISC processors are designed to execute simple but powerful instructions within a single cycle at a high clock speed.
- RISC processors follow the four major design rules.

Major Design Rules

1. Instructions

Reduced number of instructions : RISC processor provides limited number of instructions, which simplifies the design of control unit.

Simple instruction format : RISC processor uses simple instruction formats with fixed instruction length. The instruction length is aligned on word boundaries. Field locations, especially opcodes are fixed.

This architecture provides following benefits :

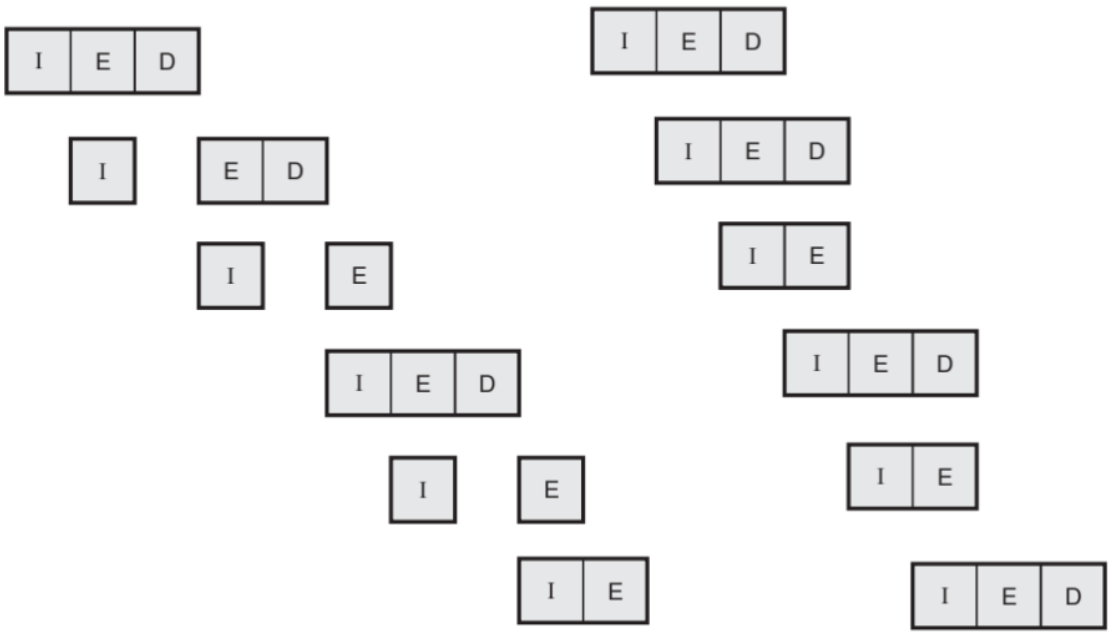
- With fixed fields, opcode decoding and register operand addressing can occur simultaneously.
- It simplifies the design of control unit.
- Instruction fetching is optimized since word-length units are fetched.

Hardwired instructions : With simple, one cycle instruction, there is no need for microinstructions. The machine instructions can be hardwired. These instructions are executed faster than the instructions implemented with microinstructions, since it is not necessary to access a microprogram control memory during instruction execution.

One instruction per cycle : RISC processor execute one instruction in a single cycle. In RISC processors, there is an one instruction per machine cycle. A machine cycle is defined to be the time it takes to fetch two operands from registers, performs an ALU operation, and stores the result in a register. Due to this, RISC machine instructions are not complicated and can execute about as fast as, microinstructions on CISC machines.

2. Pipelines

- Recall the concept of pipelining. The CPU contains several independent units that work in parallel. One of them fetches the instructions, and other ones decode and execute them. At any instant, several instructions are in various stages of processing. All RISC processors provide this pipelining feature.
- In RISC processor most instructions are register to register. These instructions have the following two phases :
  - Instruction Fetch (I)
  - Execute (E)
- The instruction fetch phase fetches the instruction to be executed from memory and then execute phase performs an ALU operation with register input and output to execute the instruction.
- In case of load and store instructions, three phases are required :
  - Instruction fetch (I)
  - Execute (E)
  - Data transfer (D)
- Here, also the instruction fetch phase fetches the instruction to be executed. In execution phase address of memory is calculated and in data transfer phase actual data is transferred from register-to-memory or from memory-to-register depending upon the instruction.



(a) Two way instruction pipelining (b) Three way instruction pipelining

3. Registers

Register to Register Operations : Risc processors have a large number of general purpose registers and 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 use, so that frequently accessed operands remain in high-speed storage.

4. Load-store architecture

RISC processors operate on data held in registers. Separate load and store instructions transfer data between the register bank and external memory.

The advantage is that the use of data items which are held in the register does not need memory access.

Comparison between RISC and CISC

Sr. No.	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
6.	Addressing modes	Complex addressing modes are synthesized in software.	Supports complex addressing modes.
7.	Instruction execution speed	Medium	Slow (depend on complexity of instruction)
8.	Instruction execution	By hardware. Simple instructions taking one cycle.	By microprogram. Complex instructions taking multiple cycles.
9.	Registers	Many, general purpose	Few, may be special purpose
10.	Memory references	Not combined with operations, i.e., load/store architecture	Combined with operations in many different types of instructions
11.	Hardware	Simple	Complicated
12.	Hardware design focus	Take the advantage of implementations with one pipeline and no microcode	Take the advantage of microcoded 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.	Compiler	Complicated	Simple
18.	Examples	Intel X86, Motorola 68000 series.	ARM, 8051, ATMEL, AVR, etc.

ARM Design Philosophy

In this section we will discuss the features of RISC which are accepted as well as rejected by ARM processors.

Let us first discuss the features of RISC which are **accepted** by ARM processors.

- **A large uniform register file**
  An ARM processor contains a large number of registers like a RISC.
- **A load-store architecture**
- ARM processor uses a RISC architecture. It contains a large number of registers. The instruction set contains separate load and store instructions for transferring data between the register bank and external memory. When the data is to be operated, it is stored in the register and then processed. The memory accesses are separated from data processing. So we can use data items stored in registers multiple times without multiple memory accesses. This is advantageous since memory accesses are costly. In contract, with a CISC architecture, the data processing instructions can operate on memory directly.
- Simple addressing modes, with all load/store addresses being determined from register contents and instruction fields only.
- **Uniform and fixed-length (32-bit) instruction fields**
  ARM processor instruction set contains a reduced number of instructions. Also, these instructions perform simple operations which can be executed in a single cycle. If the complicated operations, such as division, are to be performed, the compiler or programmer synthesizes them by combining several simple instructions. Each instruction is a fixed length. This allows the pipeline to fetch future instructions before decoding the current instruction in contrast, the CISC processor contains the instructions of variable size, complex and take more cycles to execute. So in CISC complexity is in processor hardware whereas in RISC, complexity is in compiler. The uniform and fixed-length instruction fields simplify the instruction decoding.
- **Three-address instruction formats**
  Most instructions of RISC and ARM processor have three address instruction formats. That is, two source operands are stored in two different address locations and third operand in a third address location.

Let us now discuss the features of RISC which are **rejected** by ARM processors.

- **Register windows**
  The main problem with register windows is the large chip area occupied by the large number of registers. This feature is therefore rejected by ARM processors to reduce cost.
- **Delayed branches**
  When branch instruction appears in a program and 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. In most RISC processors, this problem is tried to reduce by using delayed branches. Here, the branch takes place after the following instruction has executed. This delayed branching technique works well when the processor uses single pipeline. However, it may create problem for super-scalar implementations and also can not work well with branch prediction mechanisms.

Original ARM does not use delayed branch since they make exception handling more complex.

- **Single cycle execution of all instructions**
  ARM processor executes most of the data processing instructions in a single cycle. However, many other instructions need multiple clock cycles for their execution. More than one clock cycle becomes the requirement even for a simple load and store instruction. At least two memory accesses one for the instruction and one for the data, are needed.

In addition, the ARM architecture gives you :

- Control over both the Arithmetic Logic Unit (ALU) and shifter in every data-processing instruction to maximize the use of an ALU and a shifter.
- Auto-increment and auto-decrement addressing modes to optimize program loops.
- Load and store multiple instructions to maximize data throughput.
- Conditional execution of all instructions to maximize execution throughput.

These enhancements to a basic RISC architecture allow ARM processors to achieve a good balance of high performance, low code size, low power consumption and low silicon area.

Introduction to ARM Processor

- ARM has several processors that are grouped into number of families based on the processor core they are implemented with.
- The architecture of ARM processors has continued to evolve with every family. Some of the famous ARM Processor families are ARM7, ARM9, ARM10 and ARM11. Every ARM processor implementation executes a specific Instruction Set Architecture (ISA).
- The ISA has evolved to keep up compatibility so that code written to execute on an earlier architecture revision will also execute on a later revision of the architecture.

ARM Nomenclature

- ARM Nomenclature identifies individual processors and provides basic information about the feature set.
- The letters or words after "ARM" are used to indicate the features of a processor.

Architecture Evolution

- The architecture has continued to evolve since the first ARM processor implementation was introduced in 1985. Table 10.3.1 shows the significant architecture enhancements from the original architecture version 1 to the current version 6 architecture.

Revision	Example core implementation	ISA enhancement
ARMv1	ARM1	First ARM processor 26-bit addressing
ARMv2	ARM2	32-bit multiplier 32-bit coprocessor support

ARMv2a	ARM3	On-chip cache Atomic swap instruction Coprocessor 15 for cache management
ARMv3	ARM6 and ARM7DI	32-bit addressing Separate cpsr and spsr New modes-undefined instruction and abort MMU support-virtual memory
ARMv3M	ARM7M	Signed and unsigned long multiply instructions
ARMv4	StrongARM	Load-store instructions for signed and unsigned halfwords/bytes New mode-system Reserve SWI space for architecturally defined operations 26-bit addressing mode no longer supported
ARMv4T	ARM7TDMI and ARM9T	Thumb
ARMv5TE	ARM9E and ARM10E	Superset of the ARMv4T Extra instructions added for changing state between ARM and Thumb Enhanced multiply instructions Extra DSP-type instructions Faster multiply accumulate
ARMv5TEJ	ARM7EJ and ARM926EJ	Java acceleration
ARMv6	ARM11	Improved multiprocessor instructions Unaligned and mixed endian data handling New multimedia instructions

ARM Processor Families - ARM 7, ARM 9 and ARM 11

- ARM has several processors that are grouped into number of families based on the processor core they are implemented with.
- The families are based on the ARM7, ARM9, and ARM11 cores. The postfix numbers 7, 9, and 11 indicate different core designs.

ARM7

- ARM7 family is introduced in 1994 (ARM7TDMI, ARM7EJ-S, ARM720T)
- This family has been immensely successful and has established ARM as the architecture of choice in digital word.
- Over the years more than 10 billion ARM7 processor family based devices have powered a verity of cost and power sensitive applications.

Features of ARM7

1. **Pipeline depth :** 3 stage (Fetch, Decode, Execute)
2. **Operating frequency :** 80 MHz
3. **Power consumption :** 0.06 mW/MHz.
4. **MIPS/MHz :** 0.97
5. **Architecture used :** Von-Neumann
6. **MMU/MPU :** Not present
7. **Cache memory :** Not present
8. **Jazelle instruction :** Not present
9. **Thumb instruction :** Yes (16 bit instruction set)
10. **ARM instruction set :** Yes (32 bit)
11. **ISA (Instruction set architecture) :** V4T (4 TH Version)
12. **Interrupt Controller :** Not Present
13. **ISR entry :** Non Deterministic ISR entry
14. **Power management :** No in built Power Management
15. **Instruction Set performance v/s code size :** Optimal performance code size balance requires interworking between ARM & Thumb code
16. **Ease of application porting from one device to another :** Lack of standardization inhibits application porting.

ARM9 Processor Family

- The ARM9 family was announced in 1997.
- ARM9 family enables single processor solution for microcontroller, DSP & JAVA applications, offering savings in chip area and complexity, power consumption and time to market.
- ARM9 family has enhanced processors and these processors are well suited for applications requiring a mix of DSP+ Microcontroller performance.
- ARM9 family includes - ARM920T, ARM922T, ARM940T, ARM946E-S, ARM966E-S, and ARM926EJ-S processors.

Features of ARM9

- **Pipeline Depth :** 5 stage (Fetch, Decode, Execute, Decode, Write)
- **Operating frequency :** 150 MHz
- **Power Consumption :** 0.19 mW/MHz
- **MIPS/MHz :** 1.1
- **Architecture used :** Harvard. It separates the data D and instruction I buses.
- **MMU/MPU :** Present
- **Cache Memory :** Present (separate 16 K/8 K)
- **ARM/ Thumb Instruction :** Support both

- **ARM940T**
  - It includes a smaller D + I cache and an MPU.
  - It is designed for applications that do not require a platform operating system.
  - It executes the architecture v4T instructions
- ARM946E-S and ARM966E-S
  - Both execute architecture v5TE instructions.
  - They support the optional Embedded Trace Macrocell (ETM), which allows a developer to trace instruction and data execution in real time on the processor.
  - The ARM946E-S includes TCM (Tightly Coupled Memory), cache, and an MPU. The sizes of the TCM and caches are configurable. It is designed for use in embedded control applications that require deterministic real-time response.
  - On the other hand, the ARM966E does not have the MPU and cache extensions but does have configurable TCMs.
- **ARM926EJ-S**
  - It is synthesizable processor core, announced in 2000.
  - It is designed for use in small portable Java-enabled devices such as 3G phones and Personal Digital Assistants (PDAs).
  - Supports the Jazelle technology, which accelerates Java bytecode execution.
  - It features an MMU, configurable TCMs, and D + I caches with zero or nonzero wait state memories.

ARM11 Processors Family

- This family provides the engine that power many smartphones, also widely used in consumer, home and embedded applications.
- It delivers low power and a range of performance from 350 MHz to 1 GHz.
- ARM11 processor software is compatible with all previous generations of ARM processors.
- ARM11 family includes - ARM1176JZ (F)-S and ARM11MP core, ARM1136J(F)-S, ARM1156T2-S processors.

Features of ARM11

- **Pipeline depth :** 8 stage pipeline with separate loadstore and arithmetic pipelines.
- **Operating frequency :** 335 MHz.
- **Power Consumption :** 0.4 mW/MHz.
- **MIPS/MHz :** 1.2
- **Architecture used :** Harvard
- **MMU/MPU :** Present
- **Multiplier unit :** 16 × 32 (16 bits of 32-bit size register)
- **Cache Memory :** Present (4 - 64 K size)
- **ARM1136J-S**
  - It was announced in 2003.
  - Designed for high performance and power efficient applications.
  - It executes architecture ARMv6 instructions.
  - Supports Single Instruction Multiple Data (SIMD) extensions for media processing, specifically designed to increase video processing performance.
  - The ARM1136JF-S is an ARM1136J-S with the addition of the vector floating-point unit for fast floating-point operations.
- Supports the thumb instruction set-memory BW & Size requirements reduces by up to 35 %
- Supports Jazelle Technology for efficient embedded JAVA execution
- Supports the DSP extensions
- Supports ARM Trust-Zone Technology for on chip security
- Physically tagged caches to improve OS context switch performance.
- Tightly coupled memories for real-time applications.

Comparison between ARM7, ARM9 and ARM11 Cores

Processor attribute	ARM7	ARM9	ARM11
Pipeline depth	3-stage	5-stage	8-stage
Typical MHz	80	150	335
mW/MHz	0.06	0.19 (+ cache)	0.4 (+ cache)
MIPS/MHz	0.97	1.1	1.2
Architecture	Von neumann	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32
MMU/MPU	Absent	Present	Present
Cache Memory	Absent	Present	Present
Configurable TCM	Absent	Present	Present
Jazelle Technology	Absent	Present	Present

Features and Advantages of ARM Processor

- The features of ARM microcontroller are as follows :
  - Consists lof a **large uniform register file**.
  - Supports **load-store architecture**.
  - Uses **simple addressing modes**.
  - Contains a **reduced number of instructions.**
  - Instructions **perform simple but powerful operations** which can be **executed in a single cycle**. If the complicated operations, such as division, are to be performed, the compiler or programmer synthesizes them by combining several simple instructions.
  - It has **uniform and fixed-length (32-bit) instruction fields**.
  - **Each instruction is a fixed length**. This **allows the pipeline** to fetch future instructions before decoding the current instruction. In contrast, the CISC processor contains the instructions of variable size, complex and take more cycles to execute. So in CISC complexity is in processor hardware whereas in RISC, complexity is in compiler. The uniform and fixed-length instruction fields simplify the instruction decoding.
  - Most instructions have **three-address instruction format**.
  - **Does not use delayed branch** since they make exception handling more complex.
  - **Does not use register windows** to keep chip area small and hence the cost.
  - It has **control over both the Arithmetic Logic Unit (ALU) and shifter** in every data-processing instruction to maximize the use of an ALU and a shifter.
  - Supports **auto-increment and auto-decrement addressing modes** to optimize program loops.
  - Supports **Load and Store Multiple instructions** to maximize data throughput.
  - Supports conditiona execution of all instructions to maximize execution throughput.
- These enhancements to a basic RISC architecture allow ARM processors to achieve a **good balance of high performance, low code size, low power consumption** and **low silicon area**.

Suitability of ARM Processor in Embedded Applications

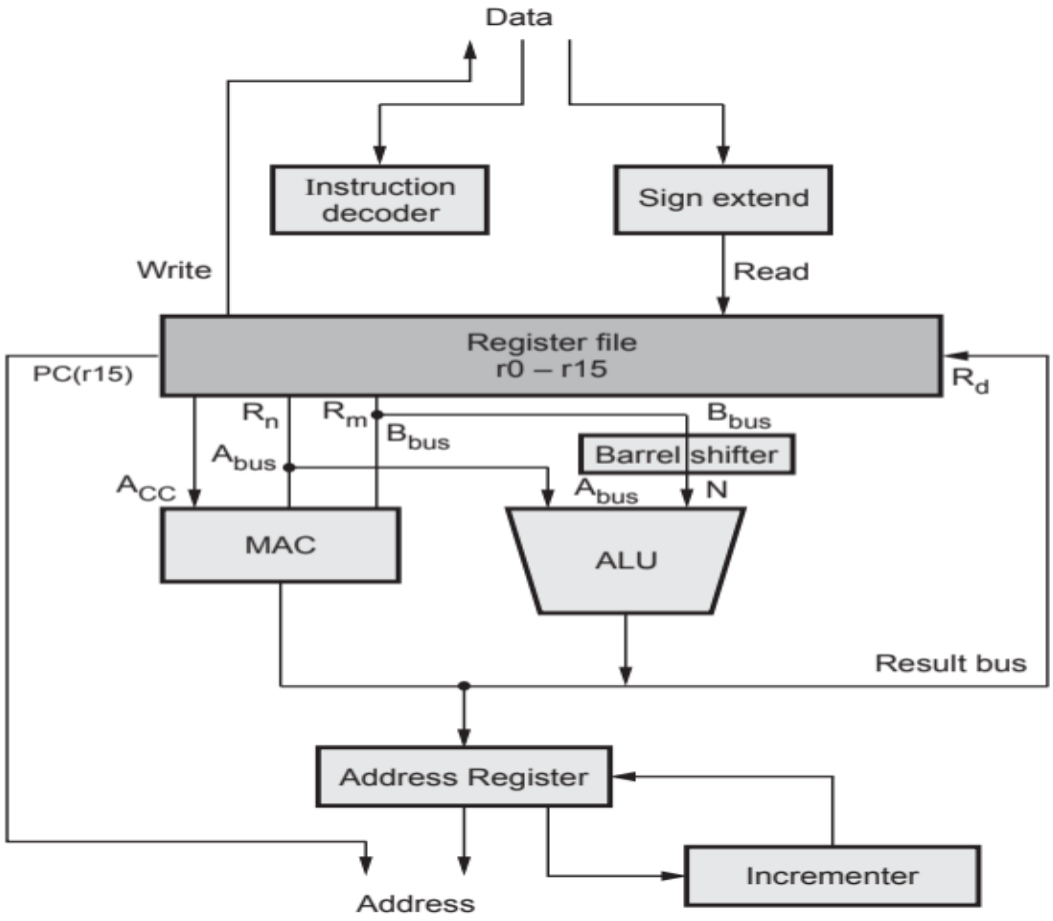
- The ARM instruction set differs from the pure RISC definition in several ways that make the ARM instruction set suitable for embedded applications :
  - **Variable cycle execution for certain instructions** : Some ARM instructions like load-store-multiple instructions vary in the number of execution cycles depending upon the number of registers being transferred. Load-store-multiple instructions transfer data on sequential memory addresses, which increases performance since sequential memory accesses are often faster than random accesses. Multiple register transfers also improve the code density.
  - **Inline barrel shifter to improve core performance and code density** : The ARM arithmetic logic unit has a barrel shifter that is capable of shift and 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 and code density.
  - **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. The 16-bit instructions improve code density by about 30 % over 32-bit fixed-length instructions.
  - **Conditional execution** : These instructions are executed when a specific condition has been satisfied. This feature improves performance and code density by reducing branch instructions.
  - **Enhanced instructions** : ARM instruction set also supports the enhanced Digital Signal Processor (DSP) instructions (fast 16×16-bit multiplier operations and saturation). These instructions allow ARM processor to serve as a combination of a processor plus a DSP.

ARM7 Dataflow Model

- An ARM7 core is an engine within a system that fetches ARM instructions from memory and executes them.
- ARM 7 cores are very small. Typically they occupy a few square millimeters of chip area.
- With advances in modern VLSI technology, it became possible to build additional system components such as cache memory, memory management unit or application specific hardware on the same chip. Application specific hardware may include signal processing hardware or further ARM processor cores.
- While designing a new system, selecting the correct processor core is one of the most critical decision.

ARM7 Core Dataflow Model

- Fig. shows the basic structure of ARM7 core and how data moves between its different parts.



- Since ARM7 processor is basically a RISC processor, it uses a **load-store architecture**. This means, it has two instruction types, load and store, for transferring data in and out of the processor respectively.
  - **LOAD :** This instruction copies data from memory to registers in the processor core.
  - **STORE :** This instruction copies data from registers in the processor core to memory.
- The ARM processor instruction set does not include the instructions that directly manipulate data in memory. The data processing is carried out only in registers.

Data bus :
- The data enters the ARM7 core through the data bus. The data is either in the form of an instruction opcode or a data item.
- Since Von Neumann architecture is used, data items and instructions share the same bus. This is in contrast with Hardvard architecture which uses two different buses.

Instruction decoder :
- This unit decodes the instruction opcode read from the memory and then the instruction is executed.

Register file :
- This is a bank of 32-bit registers used for storing data items.

Sign extend :

- The ARM7 core is a 32-bit processor. So most instructions 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 and then places them in a register file.

ALU (Arithmetic Logic Unit) and MAC (Multiply-Accumulate Unit) :

- Most of the ARM instructions are two operand instructions. The two source registers R_n and R_m are used to store these operands. These source operands are read from the R_n and R_m registers using the internal buses A and B respectively.
- The ALU or MAC reads the operand values from R_n and R_m registers via A and B buses respectively, performs the operation and stores the computed result via internal C bus in destination register, R_d and then to the register file.
- The load and store instructions generate address using ALU and stores it in the address register.

Address register :

- This holds the address generated by the load and store instructions and places it on the address bus.

Barrel shifter :

- The contents of the R_m register alternatively can be preprocessed in the barrel shifter before applying as an input to the ALU.
- A wide range of expressions and addresses can be calculated using the barrel shifter and ALU.

Incrementer

- For load and 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.
- The processor core continues the execution of instruction. Only when an exception or interrupt occurs, the normal execution flow is changed.

Programmers Model

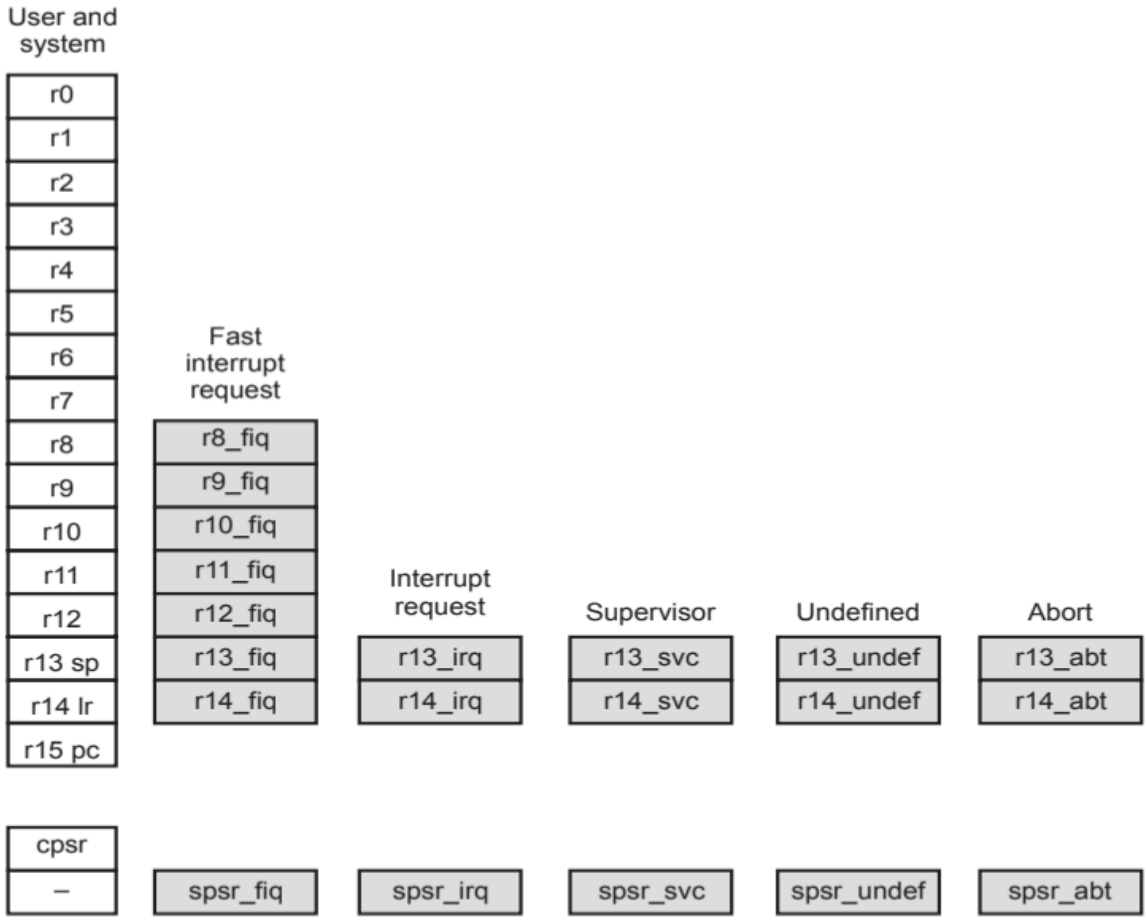
- The register file in the ARM7 core contains all the registers, available to a programmer. The current mode of the processor decides the availability of the registers to the programmer.
- The ARM processor has a total of 37 registers. All registers are 32-bits wide. They can be classified into two groups as,
  - General purpose registers
  - Special purpose registers.

General Purpose Registers

- Registers r0 to r12 are used as general purpose registers. Depending upon the context, registers r13 to r15 can also be used as general purpose registers.
- The general purpose registers hold either data or an address.

Special Purpose Registers

- Registers **r13 to r15**, **CPSR** (Current Program Status Register) and **SPSR** (Saved Program Status Register) are the special purpose registers.



Programming model of ARM processor

Registers r13 to r15

- In user mode, registers r13 to r15 are labeled as r13 sp, r14 lr and r15 pc respectively to differentiate them from other registers. The functions of these registers are given below.
  - **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.
  - **Program counter (r15 pc) :** Register r15 is the program counter and stores the address of the next instruction to be fetched from the memory by the processor.
    - It is used in most instructions as a pointer to the instruction which is two instructions after the instruction being executed.

- ♦ All ARM instructions are four bytes long (one 32-bit word) and are always aligned on a word boundary. This means that the bottom two bits of the PC are always zero, and therefore the PC contains only 30 non-constant bits.
- ♦ It can often be used in place of one of the general-purpose registers r0 to r14, and is therefore considered one of the general-purpose registers. However, there are also many instruction-specific restrictions or special cases about its use. Usually, the instruction is unpredictable if r15 is used in a manner that breaks these restrictions.

The Unbanked Registers r0 - r7

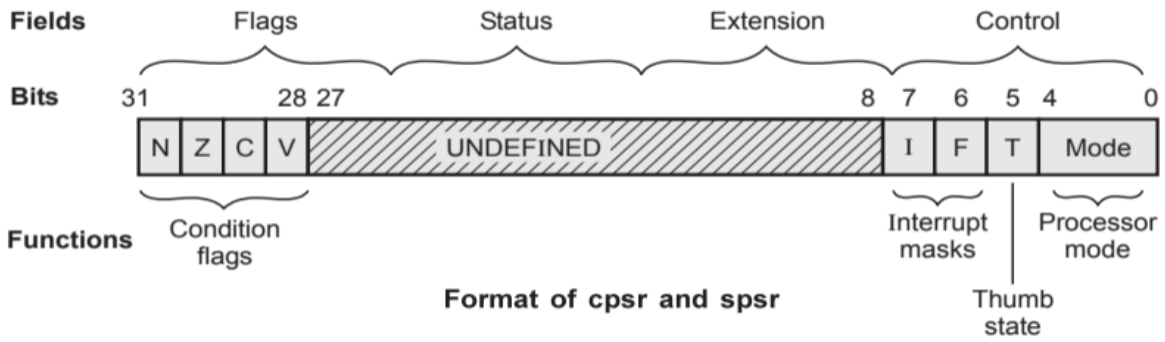
- Registers r0 to r7 are **unbanked registers**. This means that each of them refers to the same 32-bit physical register in all processor modes.
- They are completely general-purpose registers, with no special uses implied by the architecture, and can be used wherever an instruction allows a general-purpose register to be specified.

The Banked Registers, r8 - r14

- Registers 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 (as described below) is used. Almost all instructions allow the banked registers to be used wherever a general-purpose register is allowed.
- Out of 37 registers, 20 registers which are shown shaded in Fig. are the banked registers. Fig. also shows which banked registers are used in which mode. Banked registers of a particular mode are denoted by, r number_mode.
- For example, supervisor, mode has banked registers r13_svc, r14_svc and spsr_svc.
- Registers r8 to r12 have two banked physical registers each. The first group of physical registers are referred to as r8_usr to r12_usr and the second group as r8_fiq to r12_fiq. The r8_usr to r12_usr group is used in all processor modes other than FIQ mode, and the other is used in FIQ mode.
- Registers r13 and r14 have six banked physical registers each. One is used in User and System modes, while each of the remaining five is used in one of the five exception modes.
- The registers r0 to r13 are orthogonal. This means, any instruction which you can apply to r0, you can equally well apply to any of the r1 to r13 registers. This is not the case with r14 and r15 registers.

CPSR and SPSR Registers

- The **current program status register (cpsr)** is accessible in all processor modes. It contains condition code flags, interrupt disable bits, the current processor mode, and other status and 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.



Format of cpsr and spsr

Control flags (Bits 0-7)

- The control bits change when an exception arises and 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 as shown in Table

Processor mode	Mode Select Bits [4 : 0]
Abort	1 0 1 1 1
Fast interrupt request	1 0 0 0 1
Interrupt request	1 0 0 1 0
Supervisor	1 0 0 1 1
System	1 1 1 1 1
Undefined	1 1 0 1 1
User	1 0 0 0 0

Processor mode

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 and Jazelle. One of the three instruction set is active when the processor is in ARM state, Thumb state and Jazelle state respectively.

Thumb

- Thumb instructions are 16 bits (instead of the usual 32 bit). This allows for greater code density in places where memory is restricted.
- The Thumb set can only address the first eight registers, and there are no conditional execution instructions. Also, the Thumb cannot do a number of things required for low-level processor exceptions, so the Thumb instruction set will always come alongside the full ARM instruction set.

- Exceptions and the like can be handled in ARM code, with Thumb used for the more regular code.
- Table gives the comparison of ARM and Thumb instruction set features.

	ARM (cpsr T = 0)	Thumb (cpsr T = 1)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution	Most	Only branch instructions
Data processing instructions	Access to barrel shifter and ALU	Separate barrel shifter and ALU instructions
Program status register	Read-write in privileged mode	No direct access
Register usage	15 general-purpose registers +*pc*	8 general-purpose registers +7 high registers +*pc*

Comparison of ARM and Thumb instruction set features

Jazelle

- The third instruction set introduced by ARM designers is Jazelle. The J bit is the additional flag bit in the flags field only available on Jazelle-enabled processors.
- The Jazelle J and Thumb T bits in the cpsr decide the state of the processor. When both, J and T bits are 0, the processor is in ARM state and executes ARM instructions. When the T bit is 1, the processor is in Thumb state and executes Thumb instructions. When T bit is 0 and J bit is 1, the processor is in Jazelle state and executes Jazelle instructions.
- Jazelle executes 8-bit instructions. It is a hybrid mix of software and hardware. It is designed to increase the speed of execution of Java byte codes. The Jazelle technology and a specially modified version of the Java virtual machine is needed to execute Java byte codes.
- The Jazelle instruction set features are given below.
- Jazelle instructions are 8-bit in size.

Bits 6 and 7 (Interrupt Masks)

- There are two interrupts available on the ARM processor core :
  - Interrupt Request (IRQ) and
  - Fast Interrupt Request (FIQ).
- These are maskable interrupts and their masking is controlled by bits 6 and 7 of cpsr. Bit 6(F) controls FIQ and bit 7(I) controls IRQ.
- When the bit is set to binary 1, the corresponding interrupt request is masked and when bit is 0, the interrupt is available.

Condition code flags

- These flags in the cpsr can be tested by most instructions to determine whether the instruction is to be executed.
- The condition code flags are usually modified by : Execution of a comparison instruction (CMN, CMP, TEQ or TST).
- Execution of some other arithmetic, logical or move instruction, where the destination register of the instruction is not r15. Most of these instructions have both a flag-preserving and a flag-setting variant, with the latter being selected by adding an S qualifier to the instruction mnemonic. Some of these instructions only have a flag-preserving version.

Bit 27 (Saturation flag, Q)

- This flag is available for the ARM processor cores which include the DSP extensions. If an overflow and/or saturation occurs in an enhanced DSP instruction, the Q bit is set to 1. The flag is 'sticky' which means the hardware only sets this flag. We need to write to the cpsr directly to clear the flag bit.
- Similarly, bit [27] of each spsr is a Q flag and is used to preserve and 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 or subtraction, V is set to 1 if signed overflow occurred, regarding the operands and result as two's complement signed integers.
  - For non-addition/subtractions, V is normally left unchanged

Bit 29 (Carry flag, C)

- It is set in one of four ways :
  - For an addition, including the comparison instruction CMN, C is set to 1 if the addition produced a carry (that is, an unsigned overflow), and to 0 otherwise.
  - For a subtraction, including the comparison instruction CMP, C is set to 0 if the subtraction produced a borrow (that is, an unsigned underflow), and to 1 otherwise.
  - For non-addition/subtractions 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 (which often indicates an *equal* result from a comparison), and 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.
- The N, Z, C and V flags can be modified in these additional ways :
  - Execution of an MSR instruction, as part of its function of writing a new value to the cpsr or spsr.
  - Execution of MRC instructions with destination register r15. The purpose of such instructions is to transfer coprocessor-generated condition code flag values to the ARM processor.
  - Execution of some variants of the LDM instruction. These variants copy the spsr to the cpsr, and their main intended use is for returning from exceptions.
  - Execution of flag-setting variants of arithmetic and logical instructions whose destination register is r15. These also copy the spsr to the cpsr and are mainly intended for returning from exceptions.

Other Bits

- Other bits in the program status registers are reserved for future expansion.

Modes of Operation

- In the ARM7, there are seven operating modes. These modes are protected or exception modes which have associated interrupt sources and their own register set.

1. **Supervisor mode (Default) :** This is protected mode for running system level code to access hardware or run OS calls. The ARM7 enters this mode after reset.
2. **FIQ (Fast Interrupt reQuest) :** This mode supports high speed interrupt handling.
3. **IRQ (Interrupt ReQuest) :** 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 :** 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 (Current Program Status Register) and modes can only be changed when an exception is generated. This mode is also known as **Unprivileged mode**.
7. **System :** This mode is used for running operating system tasks. It uses the same registers as user mode.

- All the above modes, except user mode, are privilege modes.
- For all operating modes, user registers r0 - r7 are common. However, FIQ mode replaces the r0 - r7 registers by its own registers r8 to r14. Similarly, each of the other modes have their own r13 and r14 registers so that each operating mode has its own unique stack pointer and link register.

Difference between PIC and ARM:

S.No.	PIC	ARM
01.	PIC micro-controller refers to Peripheral Interface Controller.	ARM micro-controller refers to Advanced RISC Machine.
02.	PIC micro-controllers are available in 8-bit, 16-bit, and 32-bit.	ARM micro-controllers are available in 32-bit mostly also available in 64-bit.
03.	It supports PIC, UART, USART, CAN, LIN, Ethernet, SPI, I2S communication protocol.	It supports UART, USART, SPI, CAN, LIN, I2C, Ethernet, I2S, DSP, SAI communication protocol.
04.	It has an effective instruction rate of 4 clock cycles per instruction.	It has an effective instruction rate of 1 clock cycles per instruction.
05.	It uses SRAM, Flash memory.	It uses Flash, SDRAM, EEPROM memory.
06.	It is based on some feature of RISC.	It is based on RISC instruction set architecture.
07.	It is based on Harvard memory architecture.	It is based on modified Harvard architecture.
08.	PIC micro-controller family includes PIC16, PIC17, PIC18, PIC24, PIC32.	ARM micro-controller family includes ARMv4, 5, 6, 7 and series.
09.	It has a very good community support.	It has a vast community support.
10.	Its manufacturer is Microchip.	Its manufacturers are Apple, Nvidia, Qualcomm, Samsung Electronics, and TI etc.
11.	It is available with an average cost as compared to the features.	It is available with a low cost as compared to the features.
12.	Popular micro-controllers include PIC18fXX8, PIC16f88X, PIC32MXX.	Popular micro-controllers include LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc.

****