ARM architecture

RAMA ASAIRA   12217090

Computer Architecture – Spring 2024/2025

5/13/25

ARM Processor Modes (Operating Modes)

Processor mode		Privilege	Description
User	usr	Unprivileged	Suitable for most application code.
FIQ	fiq	Privileged	Entered as a result of a fast interrupt. [3]
IRQ	irq	Privileged	Entered as a result of a normal interrupt. [3]
Supervisor	svc	Privileged	Suitable for running most kernel code. Entered on Reset, and on execution of a Supervisor Call (**SVC**) instruction.
Monitor	mon	Privileged	A Secure mode that enables change between Secure and Non-secure states, and can also be used to handle any of FIQs, IRQs and external aborts
Abort	abt	Privileged	Entered as a result of a Data Abort exception or Prefetch Abort exception. [3]
Undefined	und	Privileged	Entered as a result of an instruction-related error.
System	sys	Privileged	Suitable for application code that requires privileged access.

Comparison Between ARM and MIPS Instruction Formats

ARM and MIPS are both RISC architectures, but they differ in their instruction formats and encoding. ARM instructions usually measure 32 bits in size, with a consistent structure that is capable of supporting a wide range of operations, while MIPS instructions have a 32-bit structure but with a different coding scheme. ARM instruction set is very flexible and efficient in that it supports a variety of addressing modes and operations. Below is a detailed comparison and a compilation of ARM instruction formats with examples.

ARM Instruction Format:

- ARM instructions are always 32 bits in length, making them simple to decode and execute.
- The ARM architecture supports numerous varied instruction types, including data processing, load/store, and branch instructions, with unique encoding patterns
- ARM instructions typically feature condition codes, which allow instructions to be conditionally executed.

MIPS Instruction Format:

- MIPS instructions are also 32 bits long but are divided into three broad categories: R-type (register), I-type (immediate), and J-type (jump).
- MIPS follows a fixed instruction format with specific locations for operation codes, source and destination registers, and immediate constants.

List of ARM Instruction Formats with Examples

- Data Processing Instructions:
  - Format: [Cond][Opcode][S][Rn][Rd][Operand2]
  - Example: ADD R1, R2, R3 (Adds the values in R2 and R3, stores the result in R1.
- Load/Store Instructions:
  - Format: [Cond][P][U][B][W][L][Rn][Rd][Offset]
  - Example: LDR R1, [R2, #4] (Loads the word from memory at the address in R2 plus 4 into R1.
- Branch Instructions:
  - Format: [Cond][L][Offset]

        - Example: BNE label (Branches to the label if the zero flag is not set).
- Multiply Instructions:
    - Format: [Cond][A][S][Rd][Rn][Rs][Rm]
    - Example: MUL R1, R2, R3 (Multiplies the values in R2 and R3, stores the result in R1.

While ARM and MIPS share similarities as RISC architectures, their instruction formats reflect different design philosophies. ARM's inclusion of condition codes and a wide range of addressing modes provides flexibility, while MIPS focuses on simplicity and regularity. Understanding these differences is crucial for developers working with these architectures.

Identify the ARM Addressing modes with examples

There are multiple addressing modes that can be used for loads and stores:

- *Register addressing*- the address is in a register (1).
- *Pre-indexed addressing* - an offset to the base register is added before the memory access. The base form of this is LDR *Rd, [Rn, Op2]*. The offset can be positive or negative and can be an immediate value or another register with an optional shift applied.(2),(3).
- *Pre-indexed with write-back* - this is indicated with an exclamation mark (!) added after the instruction. After the memory access has occurred, this updates the base register by adding the offset value (4).
- *Post-index with write-back* - here, the offset value is written after the square bracket. The address from the base register only is used for the memory access, with the offset value added to the base register after the memory access (5).

```
(1)  LDR    R0, [R1]             @ address pointed to by R1
(2)  LDR    R0, [R1, R2]         @ address pointed to by R1 + R2
(3)  LDR    R0, [R1, R2, LSL #2]  @ address is R1 + (R2*4)
(4)  LDR    R0, [R1, #32]!       @ address pointed to by R1 + 32, then R1:=R1 + 32
(5)  LDR    R0, [R1], #32    @ read R0 from address pointed to by R1, then
```

Compare between ARM and MIPS use cases

ARM and MIPS architectures possess distinct use cases, which are primarily dictated by their design philosophy and performance characteristics. ARM is suited for low-power consumption and high-performance computing (HPC), while MIPS is typically employed in educational institutions and embedded systems. The use cases for these are elaborated in the subsequent sections.

ARM Use Cases

•**Integrated Low-Power Applications:** ARM processors are particularly dominant in situations where power is critical, i.e., in mobile and IoT usage, due to their power per watt high performance.

•**HPC**: Recent studies show that ARM architectures, like the Amazon Graviton and Fujitsu A64FX, can provide competitive performance in HPC benchmarks, often outperforming energy efficiency compared to previous x86 architectures.

MIPS Applications:

• **Educational Resources**: MIPS is largely employed to educate computer architecture in schools due to its simpler design, which facilitates learning fundamental concepts.

• **Embedded System and IoT**: MIPS processors are designed for embedded virtualization, offering better security and efficiency in IoT technologies. Their architecture is low power consumption and scalable in nature, which makes them suitable for smart devices.

While ARM increases in usage for power-effective and high-performance usage, MIPS is still utilized in education and some embedded systems, highlighting the diversity in processor architecture utilization.

Provide examples of processors in the industry that are implemented based on ARM architecture.

ARM architecture has become a cornerstone in the design of various processors across various industries, particularly embedded systems, mobile, and cloud computing. ARM processors' versatility allows them to find applications in microcontrollers, system-on-chip (SoCs), and even data centers, demonstrating their efficiency and versatility. The following are some instances of ARM-based processors in the industry.

ARM Processors in Embedded Systems

• **Cortex-A9**: Widely used in SoCs, such as those employed on Xilinx Zynq boards, which indicates the integration of ARM into FPGA platforms.

• **ARMv4-based Microprocessors:** Designed for embedded applications, these microprocessors employ a reduced instruction set to enhance performance and efficiency.

ARM Processors in Mobile and Consumer Devices

• **Smartphone Processors**: ARM architecture powers the majority of smartphones, enabling power-efficient performance that has transformed the mobile ecosystem.

• **Microcontrollers**: ARM processors are utilized in various consumer electronics, such as MP3 players and in-car systems, showing their widespread application.

ARM Processors in Cloud Computing

• **Power-efficient Cloud Servers:** ARM processors are increasingly being utilized within data centers because of their low power consumption, which is ideal for 24/7 availability.

While ARM processors dominate power efficiency and embedded applications, x86 processors still retain a large market share in high-performance computing applications, implying heterogeneity in processor architecture adoption.

References:

ARM processor modes

MIPS IV Instruction Set

Addressing modes

industry L1

industry L2

industry L3