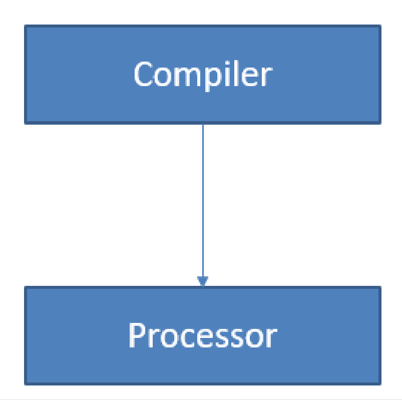
# ARM MICROCONTROLLER

CO 4

# WHAT IS RISC AND CISC?

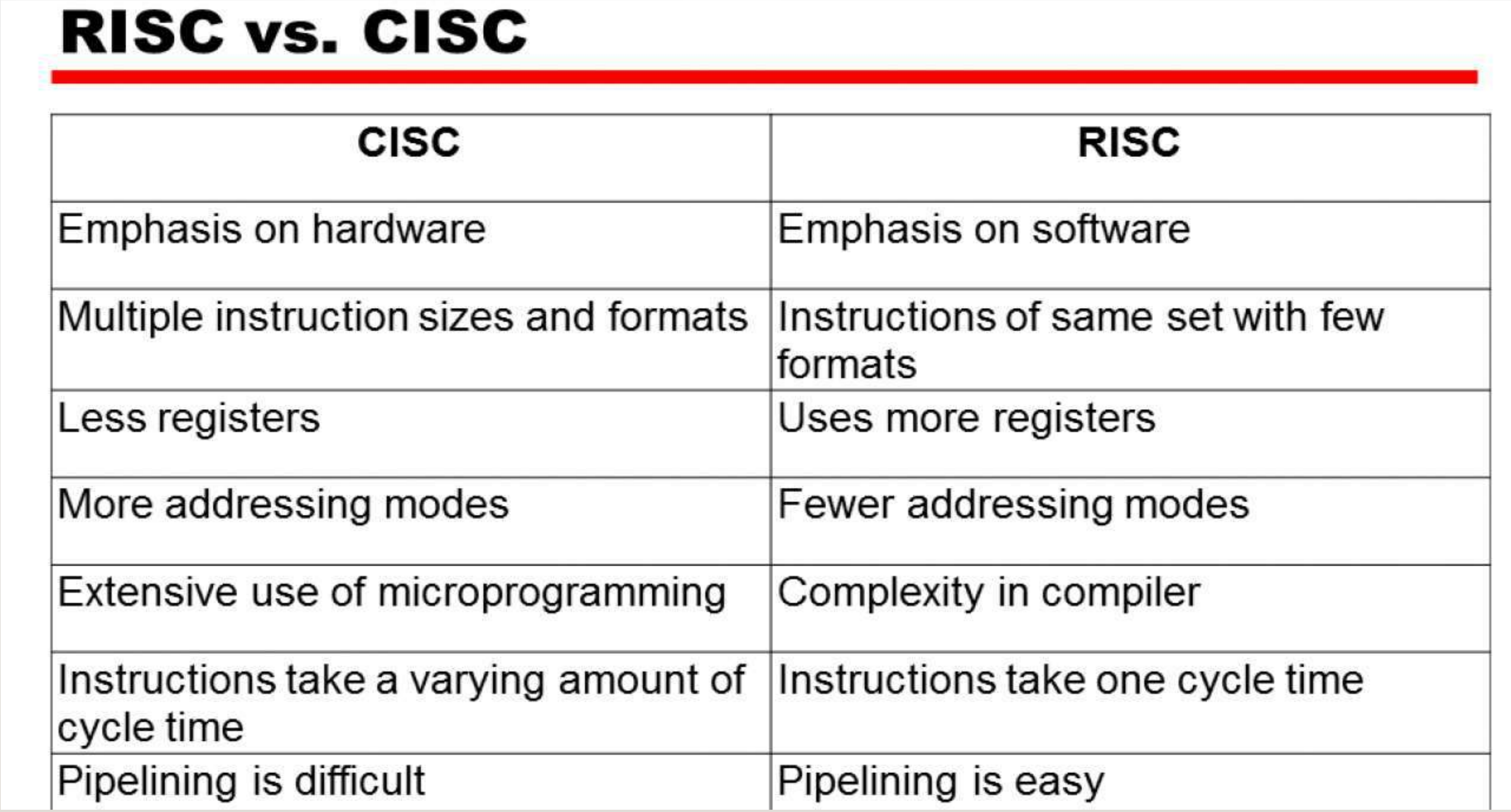
CISC: A complex instruction set computer is a computer where single instructions can perform numerous low-level operations like a load from memory, an arithmetic operation, and a memory store or are accomplished by multi-step processes or addressing modes in single instructions, as its name propose "Complex Instruction Set ".

RISC: A reduced instruction set computer is a computer which only uses simple commands that can be divided into several instructions which achieve low-level operation within a single CLK cycle, as its name propose "Reduced Instruction Set "



Compiler

Processor



# RISC vs. CISC

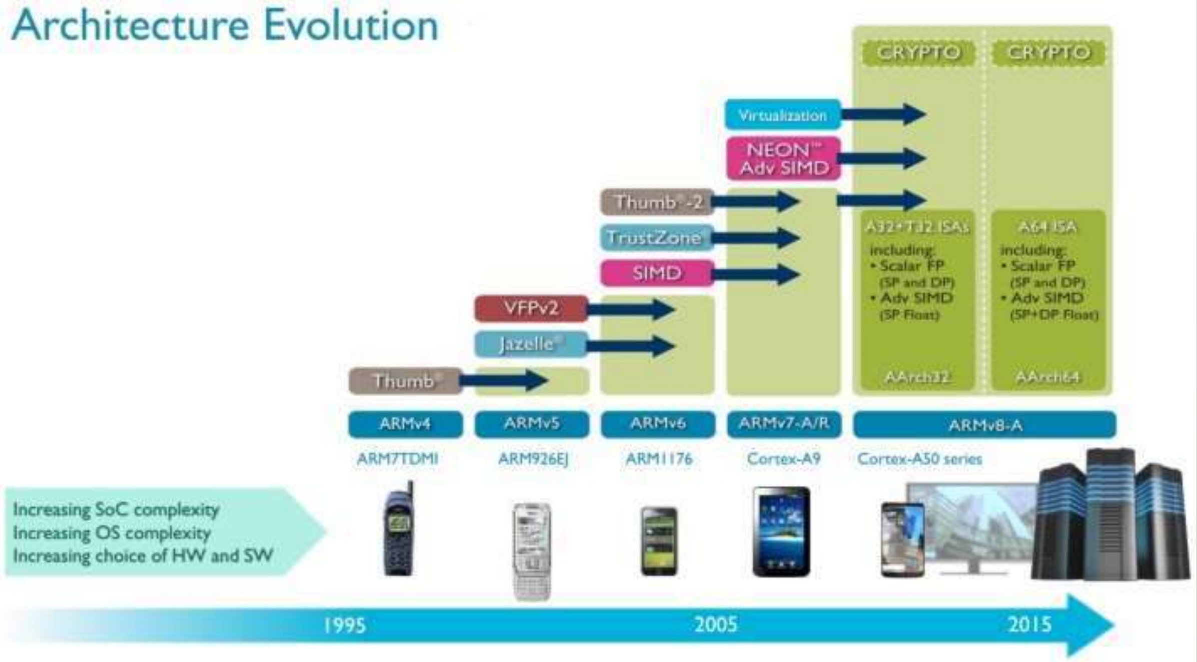
| CISC | RISC |
|---|---|
| Emphasis on hardware | Emphasis on software |
| Multiple instruction sizes and formats | Instructions of same set with few formats |
| Less registers | Uses more registers |
| More addressing modes | Fewer addressing modes |
| Extensive use of microprogramming | Complexity in compiler |
| Instructions take a varying amount of cycle time | Instructions take one cycle time |
| Pipelining is difficult | Pipelining is easy |

# WHAT IS ARM?

- Advanced RISC Machine
- First RISC microprocessor  for commercial use
- Market-leader for low-power and cost-sensitive embedded applications

# THE HISTORY OF ARM

- Developed at Acorn Computers Limited, of Cambridge, England , between 1983 and 1985
- Problems with CISC:
    - Slower memory parts
    - Clock cycles per instruction

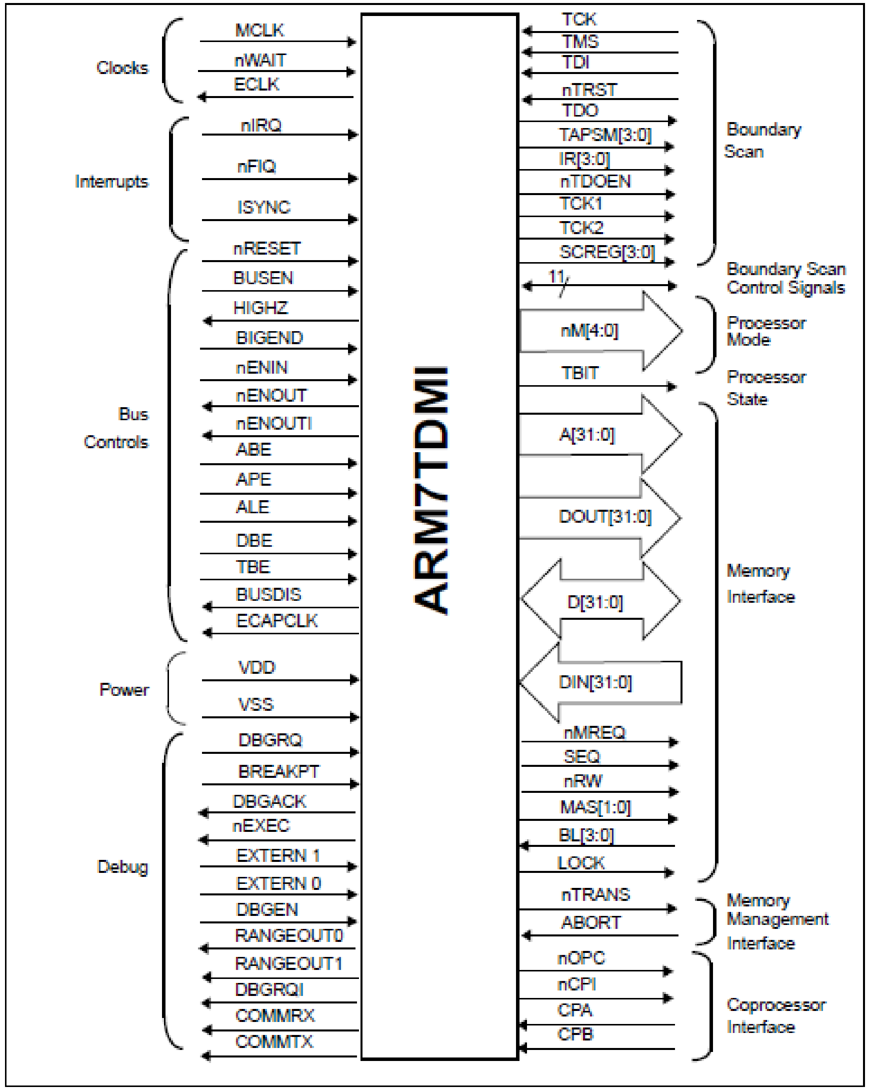


# ARM ARCHITECTURE

- Typical RISC architecture:
  - Large uniform register file
  - Load/store architecture
  - Simple addressing modes
  - Uniform and fixed-length instruction fields
- Enhancements:
  - Each instruction controls the ALU and shifter
  - Auto-increment and auto-decrement addressing modes
  - Multiple Load/Store
  - Conditional execution
  - Enhanced instructions

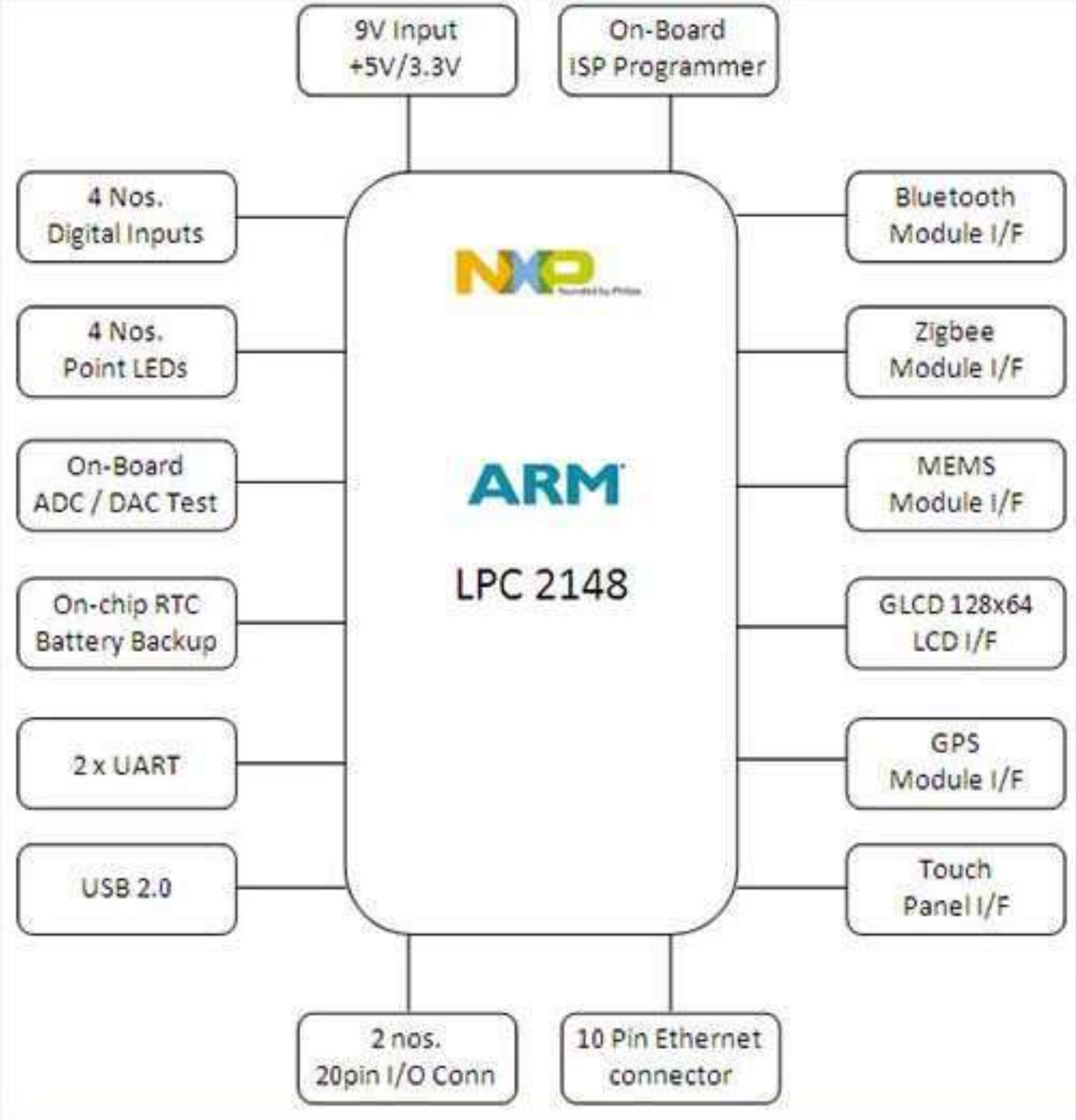
# PIN DIAGRAM

ARM7TDMI Functional Diagram

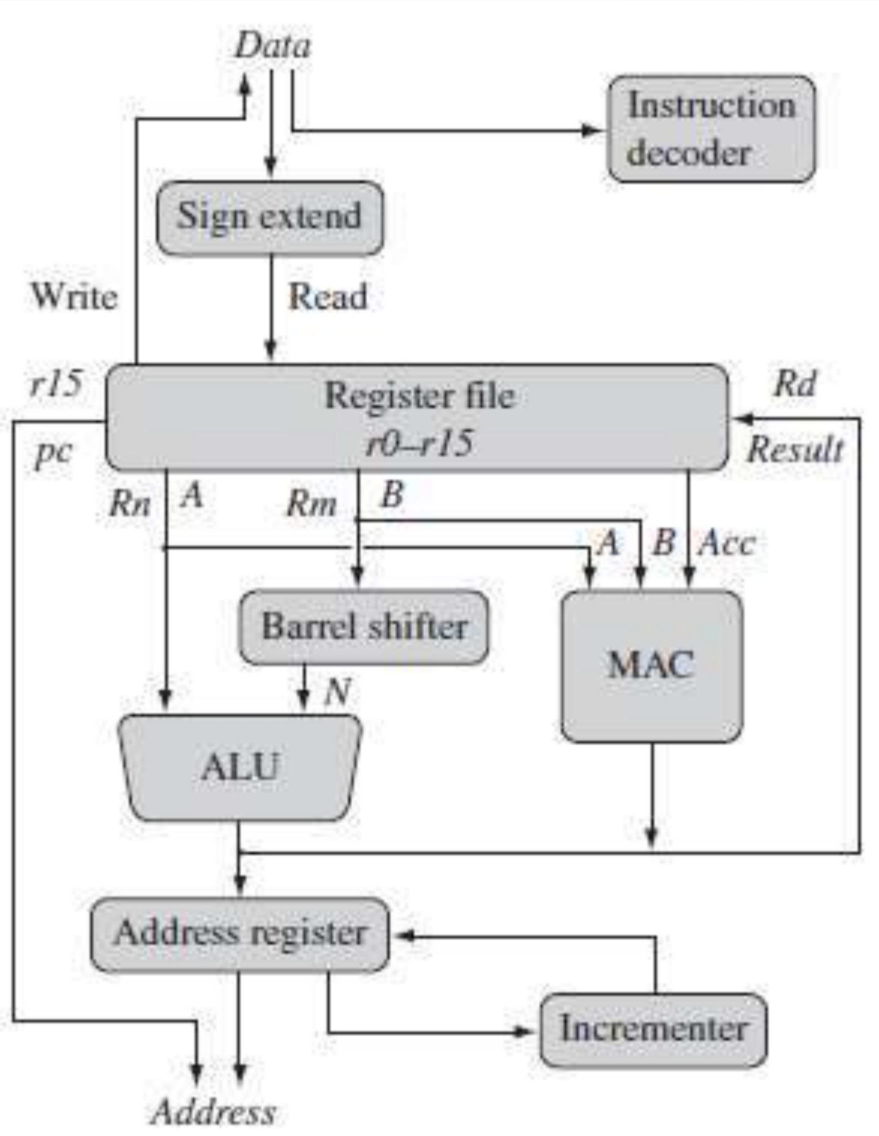


*ARM7TDMI functional diagram*

# ARM ARCHITECTURE



# DATE FLOW MODEL



# DATA SIZES AND INSTRUCTION SETS

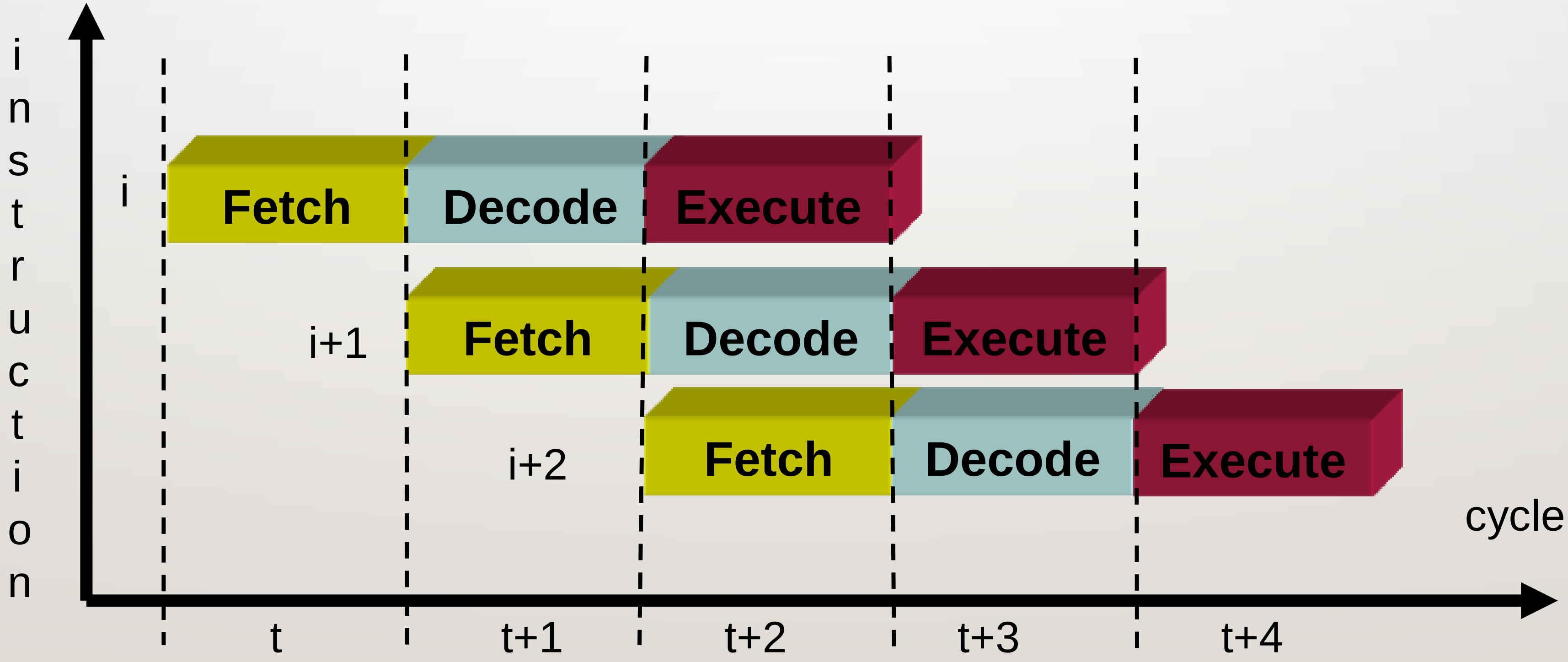
- The ARM is a 32-bit architecture.
- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Half word** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)
- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set
- Jazelle cores can also execute Java byte code

# PIPELINE ORGANIZATION

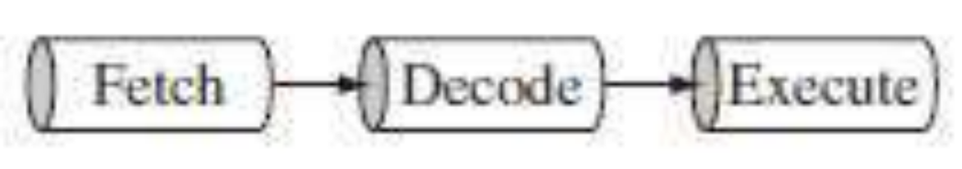
- Increases speed – most instructions executed in single cycle
- Versions:
  - 3-stage (ARM7TDMI and earlier)
  - 5-stage (ARMS,ARM9TDMI)
  - 6-stage (ARM10TDMI)

# Pipeline Organization

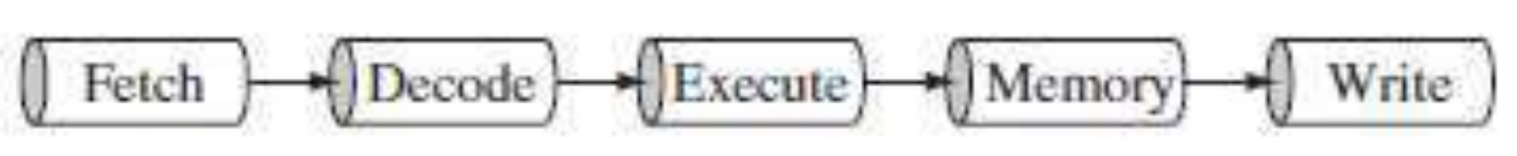
- 3-stage pipeline: Fetch – Decode - Execute
- Three-cycle latency, one instruction per cycle throughput



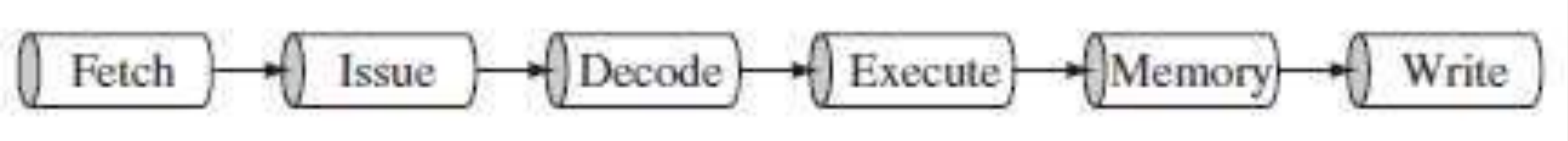
# PIPELINE ORGANIZATION



Fetch → Decode → Execute



Fetch → Decode → Execute → Memory → Write

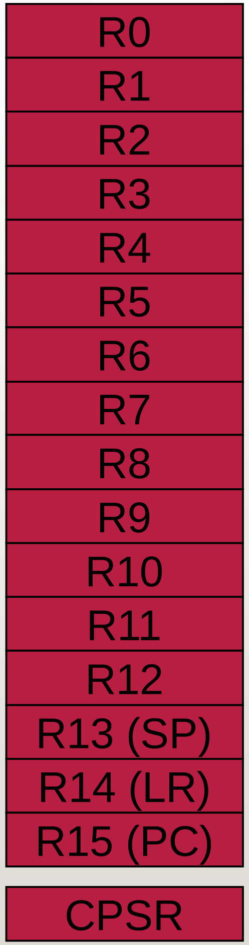


Fetch → Issue → Decode → Execute → Memory → Write

# ARM REGISTERS

- 31 general-purpose 32-bit registers
- 16 visible, R0 – R15
- Others speed up the exception process

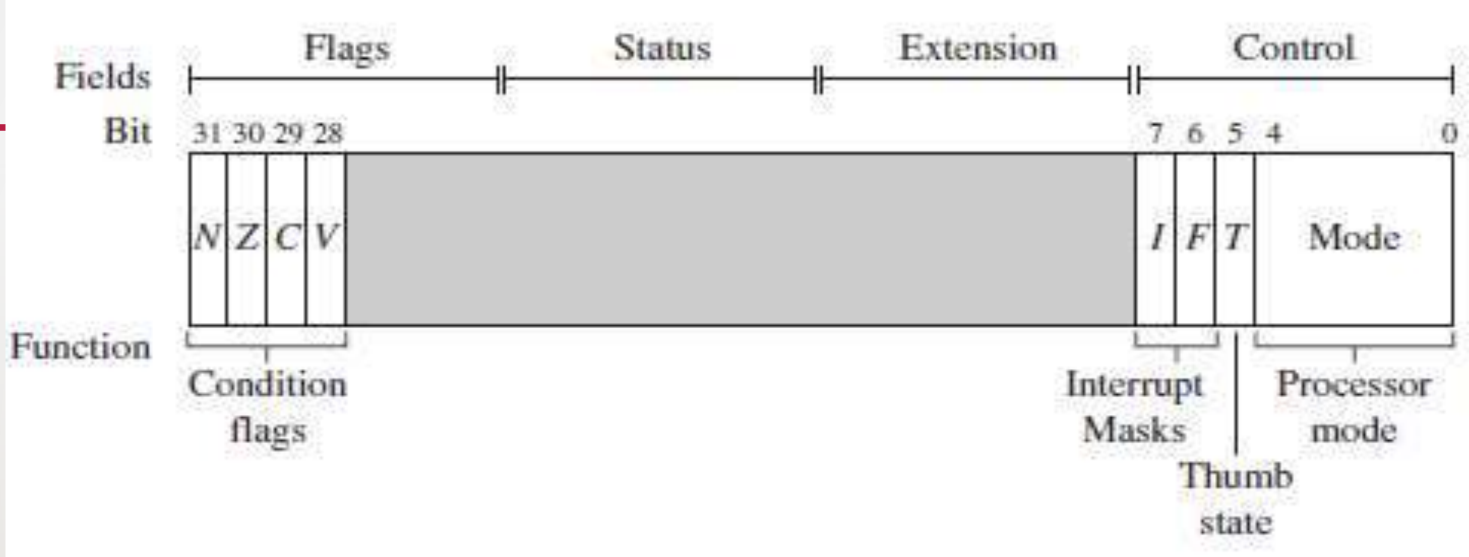
# ARM Registers

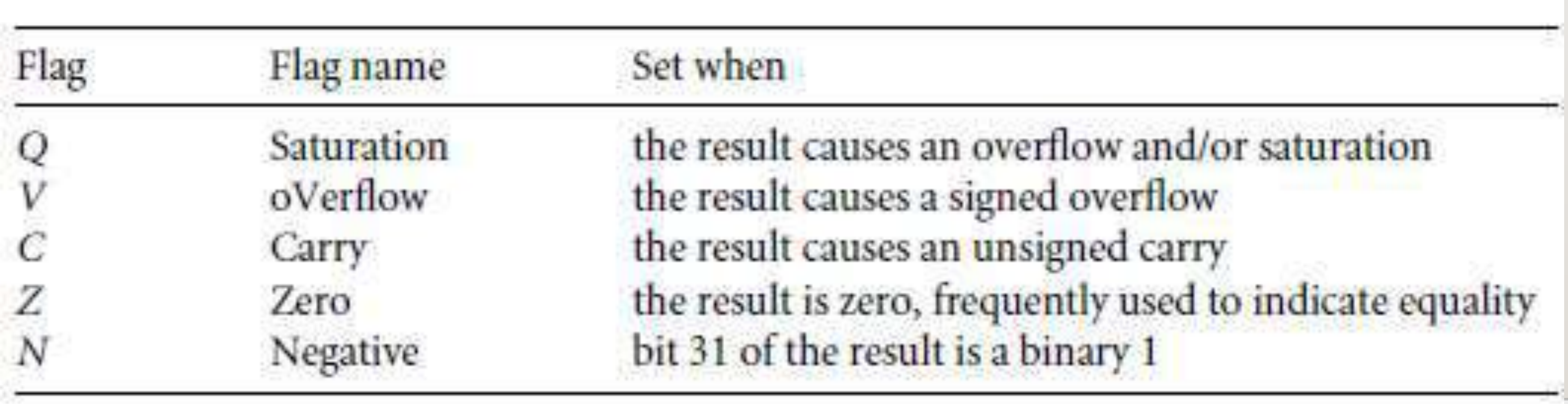


R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)

CPSR

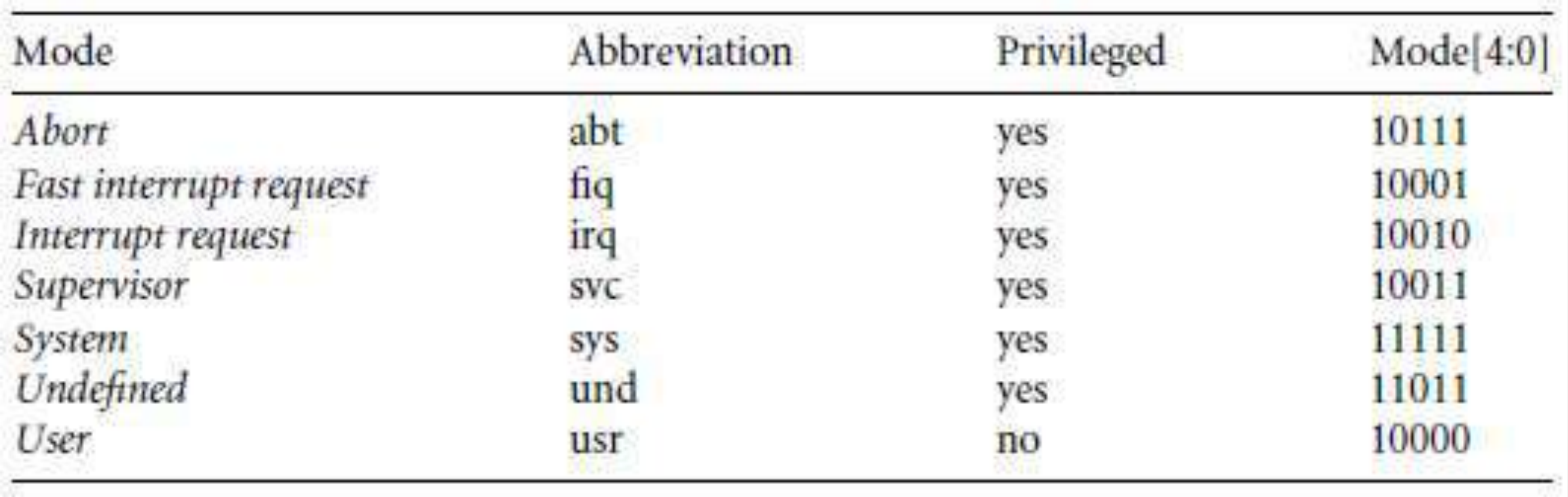
# CPSR





| Flag | Flag name | Set when |
|---|---|---|
| *Q* | Saturation | the result causes an overflow and/or saturation |
| *V* | oVerflow | the result causes a signed overflow |
| *C* | Carry | the result causes an unsigned carry |
| *Z* | Zero | the result is zero, frequently used to indicate equality |
| *N* | Negative | bit 31 of the result is a binary 1 |

# PROCESSOR MODES



| Mode | Abbreviation | Privileged | Mode[4:0] |
|---|---|---|---|
| *Abort* | abt | yes | 10111 |
| *Fast interrupt request* | fiq | yes | 10001 |
| *Interrupt request* | irq | yes | 10010 |
| *Supervisor* | svc | yes | 10011 |
| *System* | sys | yes | 11111 |
| *Undefined* | und | yes | 11011 |
| *User* | usr | no | 10000 |

# PROCESSOR MODES

The ARM has seven basic operating modes:

- **User** : unprivileged mode under which most tasks run
- **FIQ** : entered when a high priority (fast) interrupt is raised
- **IRQ** : entered when a low priority (normal) interrupt is raised
- **Supervisor** : entered on reset and when a Software Interrupt instruction is executed
- **Abort** : used to handle memory access violations
- **Undef** : used to handle undefined instructions
- **System** : privileged mode using the same registers as user mode

# OPERATING MODES

- Seven operating modes:
  - User
  - Privileged:
    - System (version 4 and above)
    - FIQ
    - IRQ
    - Abort
    - Undefined
    - Supervisor

***exception modes***

# OPERATING MODES

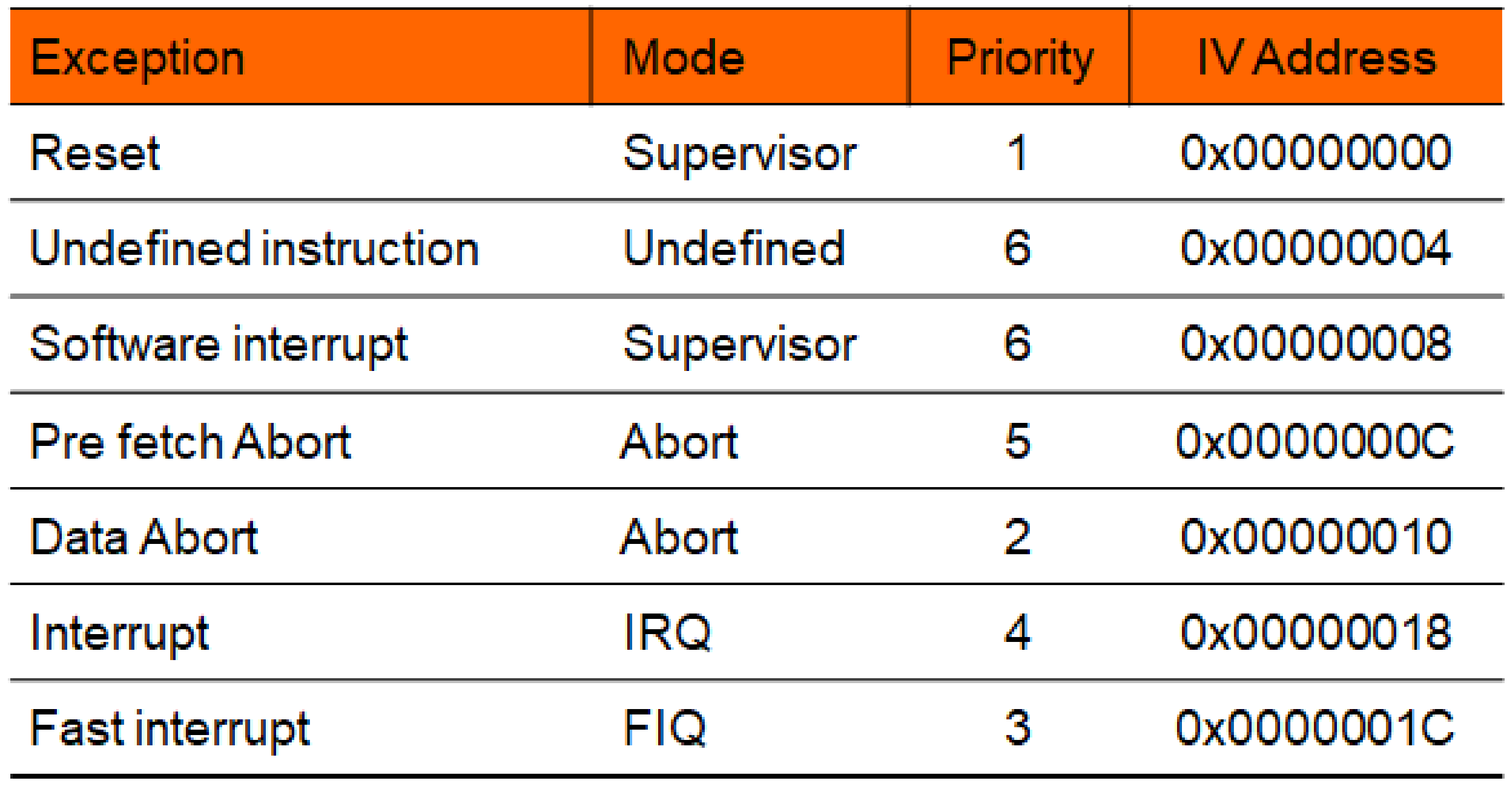
## User mode

- Normal program execution mode
- System resources unavailable
- Mode changed by exception only

## Exception mode

- – Entered upon exception
- – Full access to system resources
- – Mode changed freely

# EXCEPTIONS



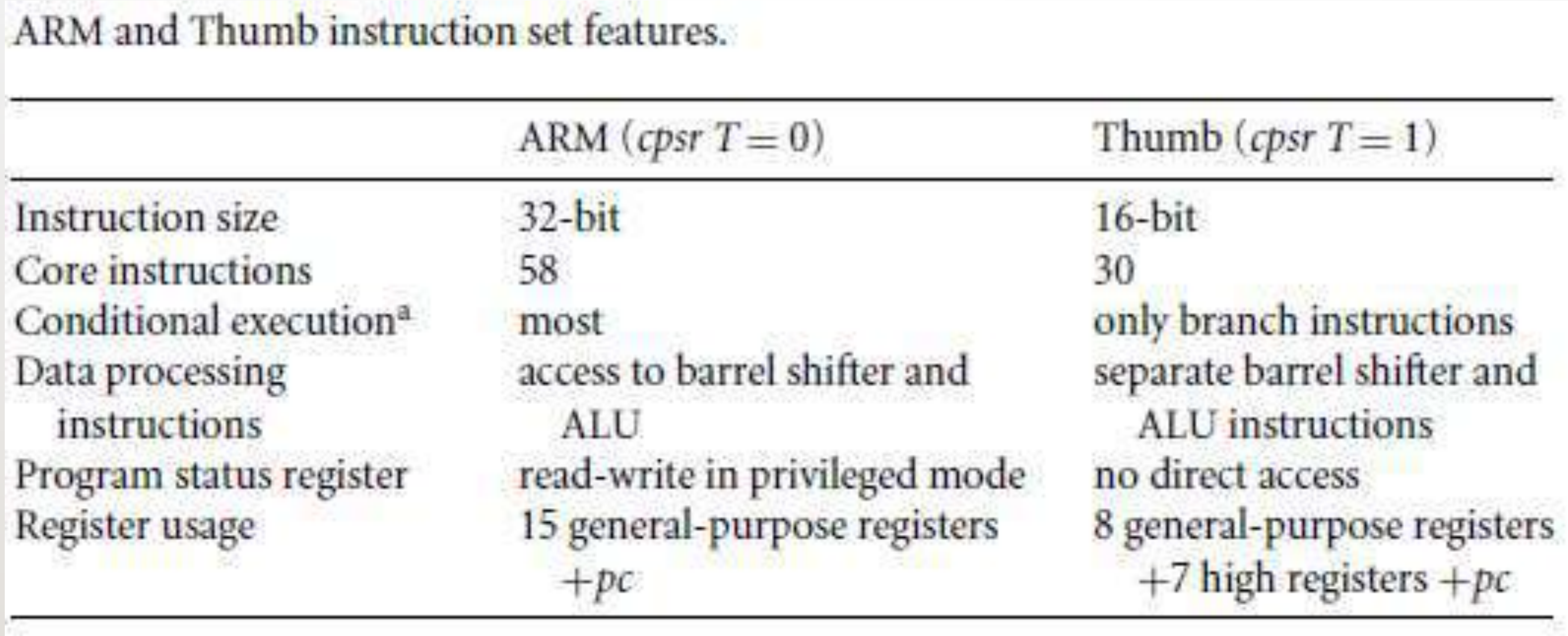
| Exception | Mode | Priority | IV Address |
|---|---|---|---|
| Reset | Supervisor | 1 | 0x00000000 |
| Undefined instruction | Undefined | 6 | 0x00000004 |
| Software interrupt | Supervisor | 6 | 0x00000008 |
| Pre fetch Abort | Abort | 5 | 0x0000000C |
| Data Abort | Abort | 2 | 0x00000010 |
| Interrupt | IRQ | 4 | 0x00000018 |
| Fast interrupt | FIQ | 3 | 0x0000001C |

Exception types, sorted by Interrupt Vector addresses

# ARM Registers

| System & User | FIQ | Supervisor | Abort | IRQ | Undefined |
|---|---|---|---|---|---|
| R0 | R0 | R0 | R0 | R0 | R0 |
| R1 | R1 | R1 | R1 | R1 | R1 |
| R2 | R2 | R2 | R2 | R2 | R2 |
| R3 | R3 | R3 | R3 | R3 | R3 |
| R4 | R4 | R4 | R4 | R4 | R4 |
| R5 | R5 | R5 | R5 | R5 | R5 |
| R6 | R6 | R6 | R6 | R6 | R6 |
| R7 | R7_fiq | R7 | R7 | R7 | R7 |
| R8 | R8_fiq | R8 | R8 | R8 | R8 |
| R9 | R9_fiq | R9 | R9 | R9 | R9 |
| R10 | R10_fiq | R10 | R10 | R10 | R10 |
| R11 | R11_fiq | R11 | R11 | R11 | R11 |
| R12 | R12_fiq | R12 | R12 | R12 | R12 |
| R13 | R13_fiq | R13_svc | R13_abt | R13_irq | R13_und |
| R14 | R14_fiq | R14_svc | R14_abt | R14_irq | R14_und |
| R15 (PC) | R15 (PC) | R15 (PC) | R15 (PC) | R15 (PC) | R15 (PC) |
| CPSR | CPSR | CPSR | CPSR | CPSR | CPSR |
|  | SPSR_fiq | SPSR_svc | SPSR_abt | SPSR_irq | SPSR_und |

# ARM VS THUMB MODE



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[a] | 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 $+pc$ |