



# ARM Architecture

*Computer Organization and Assembly Languages*  
*Yung-Yu Chuang*

*with slides by Peng-Sheng Chen, Ville Pietikainen*

# ARM history

- 1983 developed by Acorn computers
  - To replace 6502 in BBC computers
  - 4-man VLSI design team
  - Its simplicity comes from the inexperience team
  - Match the needs for generalized SoC for reasonable power, performance and die size
  - The first commercial RISC implementation
- 1990 ARM (Advanced RISC Machine), owned by Acorn, Apple and VLSI

# ARM Ltd



Design and license ARM core design but not fabricate



# Why ARM?



- One of the most licensed and thus widespread processor cores in the world
  - Used in PDA, cell phones, multimedia players, handheld game console, digital TV and cameras
  - ARM7: GBA, iPod
  - ARM9: NDS, PSP, Sony Ericsson, BenQ
  - ARM11: Apple iPhone, Nokia N93, N800
  - 90% of 32-bit embedded RISC processors till 2009
- Used especially in portable devices due to its low power consumption and reasonable performance



## ARM family comparison



ARM family attribute comparison.

year	1995	1997	1999	2003
	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz <sup>a</sup>	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS <sup>b</sup> /MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

<sup>a</sup> Watts/MHz on the same 0.13 micron process.

<sup>b</sup> MIPS are Dhrystone VAX MIPS.

## ARM is a RISC



- RISC: simple but powerful instructions that execute within a single cycle at high clock speed.
- Four major design rules:
  - Instructions: reduced set/single cycle/fixed length
  - Pipeline: decode in one stage/no need for microcode
  - Registers: a large set of general-purpose registers
  - Load/store architecture: data processing instructions apply to registers only; load/store to transfer data from memory
- Results in simple design and fast clock rate
- The distinction blurs because CISC implements RISC concepts

## ARM design philosophy



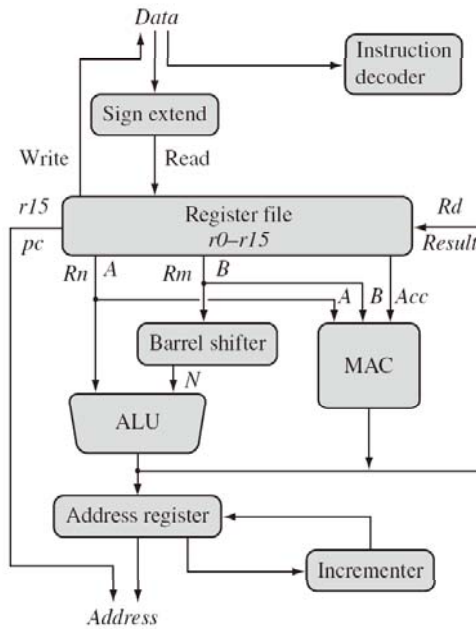
- Small processor for lower power consumption (for embedded system)
- High code density for limited memory and physical size restrictions
- The ability to use slow and low-cost memory
- Reduced die size for reducing manufacture cost and accommodating more peripherals

## ARM features



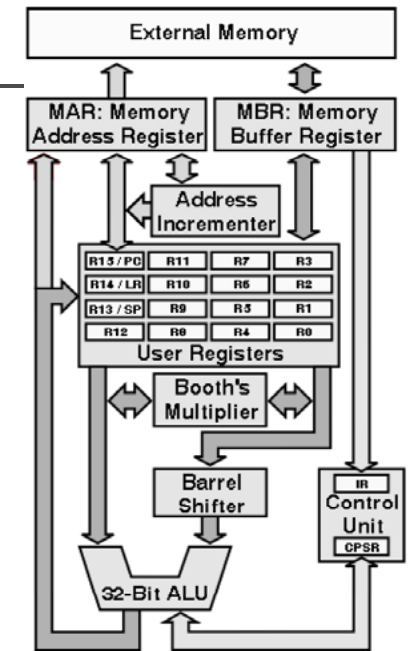
- Different from pure RISC in several ways:
  - Variable cycle execution for certain instructions: multiple-register load/store (faster/higher code density)
  - Inline barrel shifter leading to more complex instructions: improves performance and code density
  - Thumb 16-bit instruction set: 30% code density improvement
  - Conditional execution: improve performance and code density by reducing branch
  - Enhanced instructions: DSP instructions

## ARM architecture



## ARM architecture

- Load/store architecture
- A large array of uniform registers
- Fixed-length 32-bit instructions
- 3-address instructions

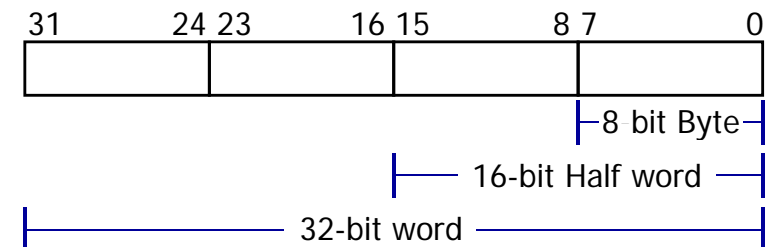


## Registers



- Only 16 registers are visible to a specific mode. A mode could access
  - A particular set of r0-r12
  - r13 (sp, stack pointer)
  - r14 (lr, link register)
  - r15 (pc, program counter)
  - Current program status register (cpsr)
  - The uses of r0-r13 are orthogonal

## General-purpose registers



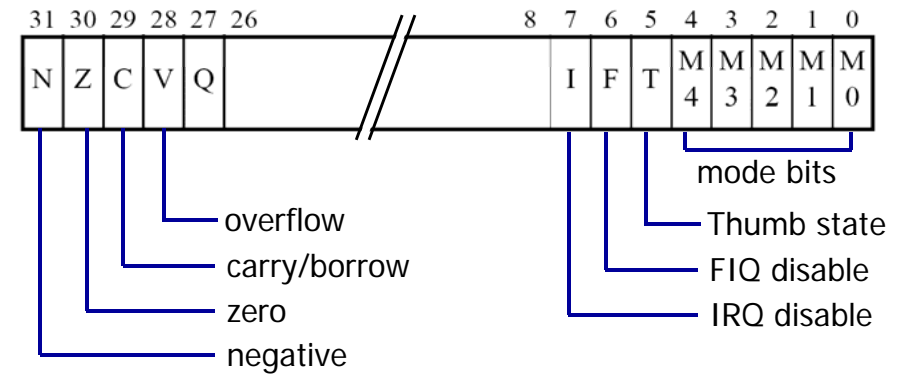
- 6 data types (signed/unsigned)
- All ARM operations are 32-bit. Shorter data types are only supported by data transfer operations.

## Program counter



- Store the address of the instruction to be executed
- All instructions are 32-bit wide and word-aligned
- Thus, the last two bits of pc are undefined.

## Program status register (CPSR)

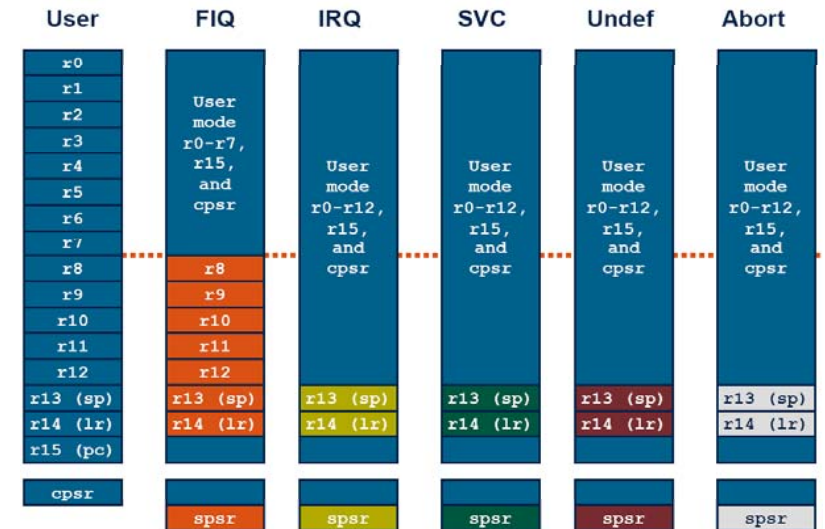


## Processor modes



Processor mode	Description
User	Normal program execution mode
FIQ	Supports a high-speed data transfer or channel process
IRQ	Used for general-purpose interrupt handling
Supervisor	A protected mode for the operating system
Abort	Implements virtual memory and/or memory protection
Undefined	Supports software emulation of hardware coprocessors
System	Runs privileged operating system tasks

## Register organization



## Instruction sets



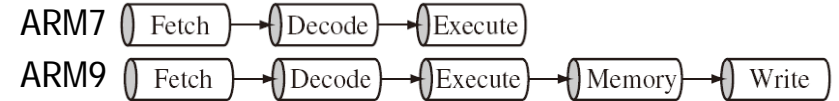
- ARM/Thumb/Jazelle

	ARM ( <i>cpsr</i> $T = 0$ )	Thumb ( <i>cpsr</i> $T = 1$ )
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution <sup>a</sup>	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 + <i>pc</i>	8 general-purpose registers + 7 high registers + <i>pc</i>

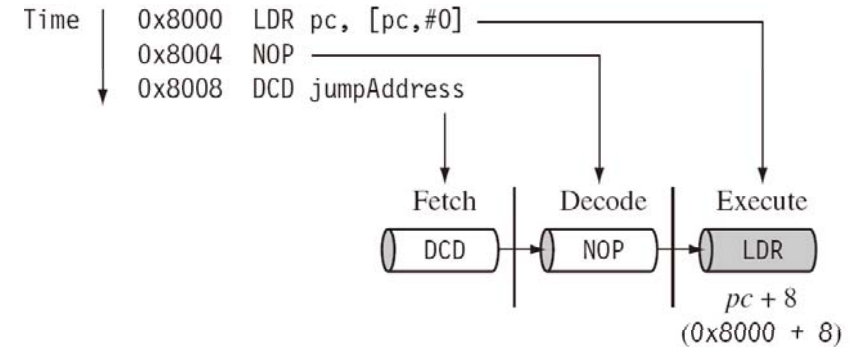
---

Jazelle ( <i>cpsr</i> $T = 0, J = 1$ )	
Instruction size	8-bit
Core instructions	Over 60% of the Java bytecodes are implemented in hardware; the rest of the codes are implemented in software.

## Pipeline



In execution, *pc* always 8 bytes ahead

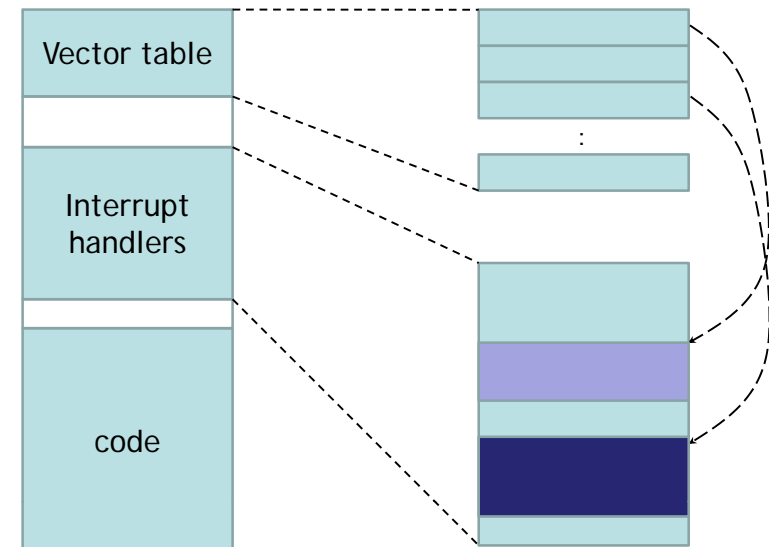


## Pipeline



- Execution of a branch or direct modification of *pc* causes ARM core to flush its pipeline
- ARM10 starts to use branch prediction
- An instruction in the execution stage will complete even though an interrupt has been raised. Other instructions in the pipeline are abandoned.

## Interrupts





# Interrupts



Exception/interrupt	Shorthand	Address
Reset	RESET	0x00000000
Undefined instruction	UNDEF	0x00000004
Software interrupt	SWI	0x00000008
Prefetch abort	PABT	0x0000000c
Data abort	DABT	0x00000010
Reserved	—	0x00000014
Interrupt request	IRQ	0x00000018
Fast interrupt request	FIQ	0x0000001c

# References

