



## LECTURE 4: ARM Cortex-M architecture overview

*EMBEDDED SYSTEM COURSE*

# Learning Goals

Introduce about the ARM Cortex M processor.

Explain some core components in Cortex-M including NVIC, SysTick timer and Floating Point Unit.

Explain about the basic concepts on Cortex-M instruction set.

# Table of contents

General Information about the Cortex-M

Introduction to the architecture

Programmer Model

Instruction Set

Summary

General Information about the Cortex-M

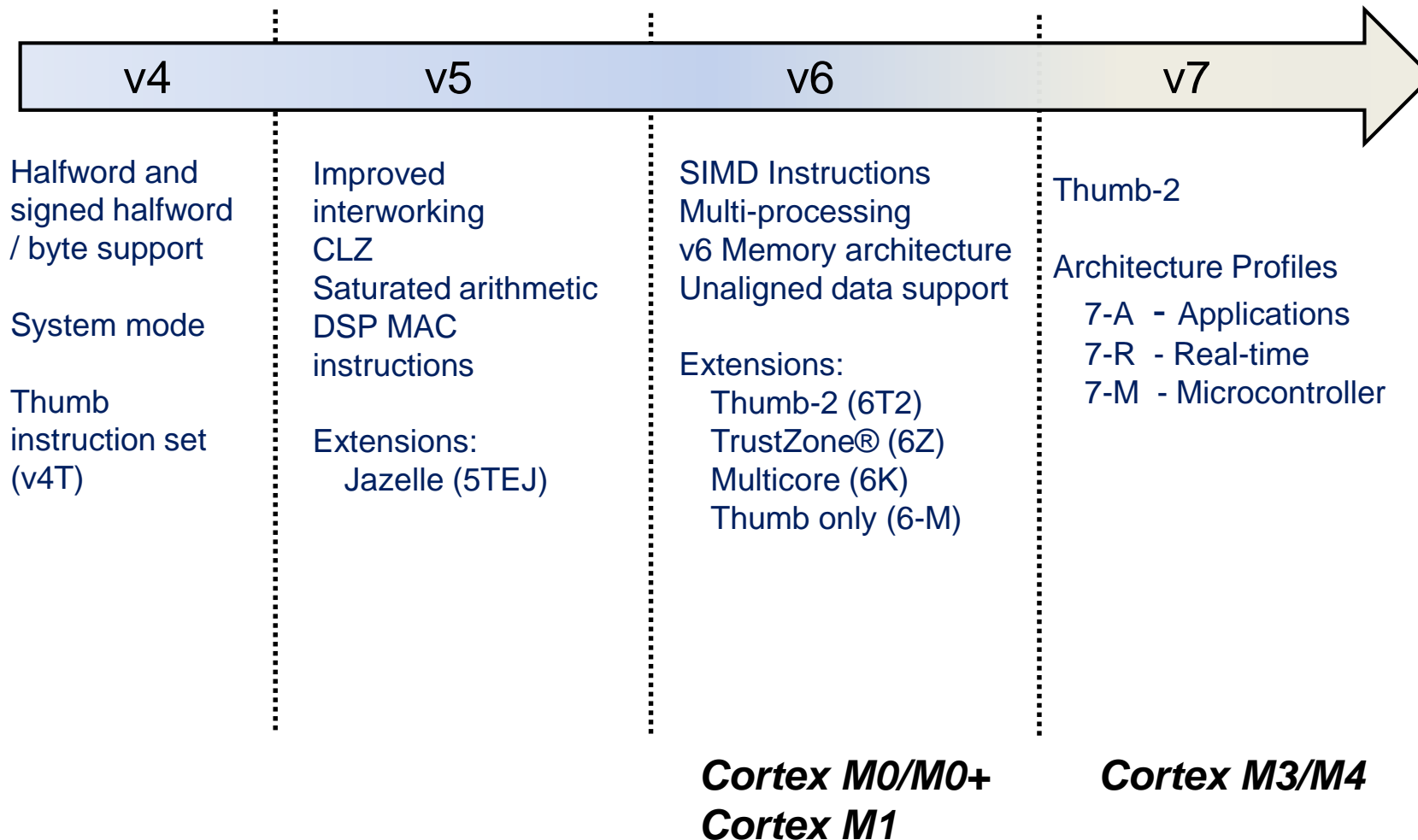
Introduction to the architecture

Programmer Model

Instruction Set

Summary

## Development of the ARM Architecture



## ARM CORTEX

The ARM Cortex family includes processors based on the three distinct profiles of the ARMv7 architecture.

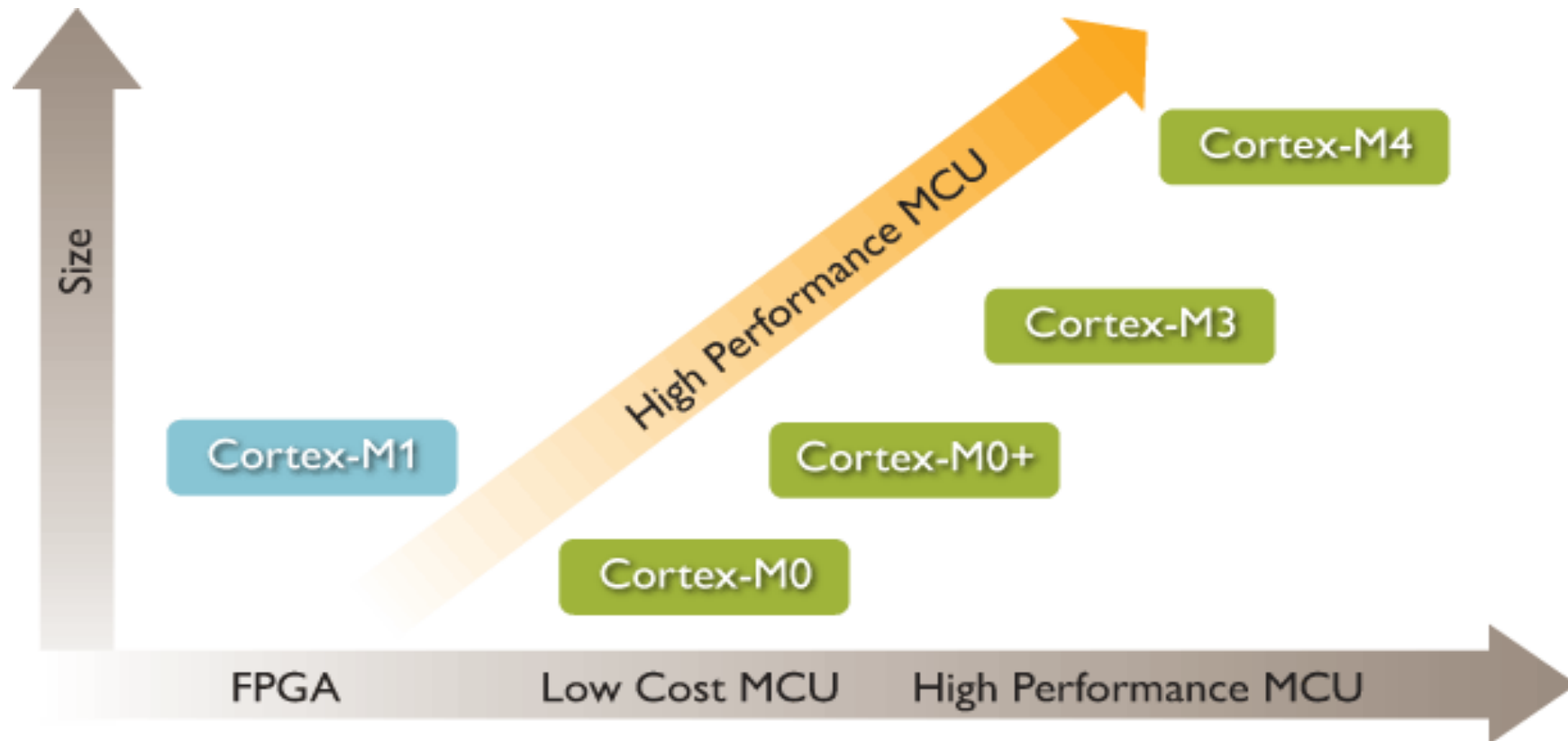
The A profile for sophisticated, high-end applications running open and complex operating systems

The R profile for real-time systems

The M profile optimized for cost-sensitive and microcontroller applications

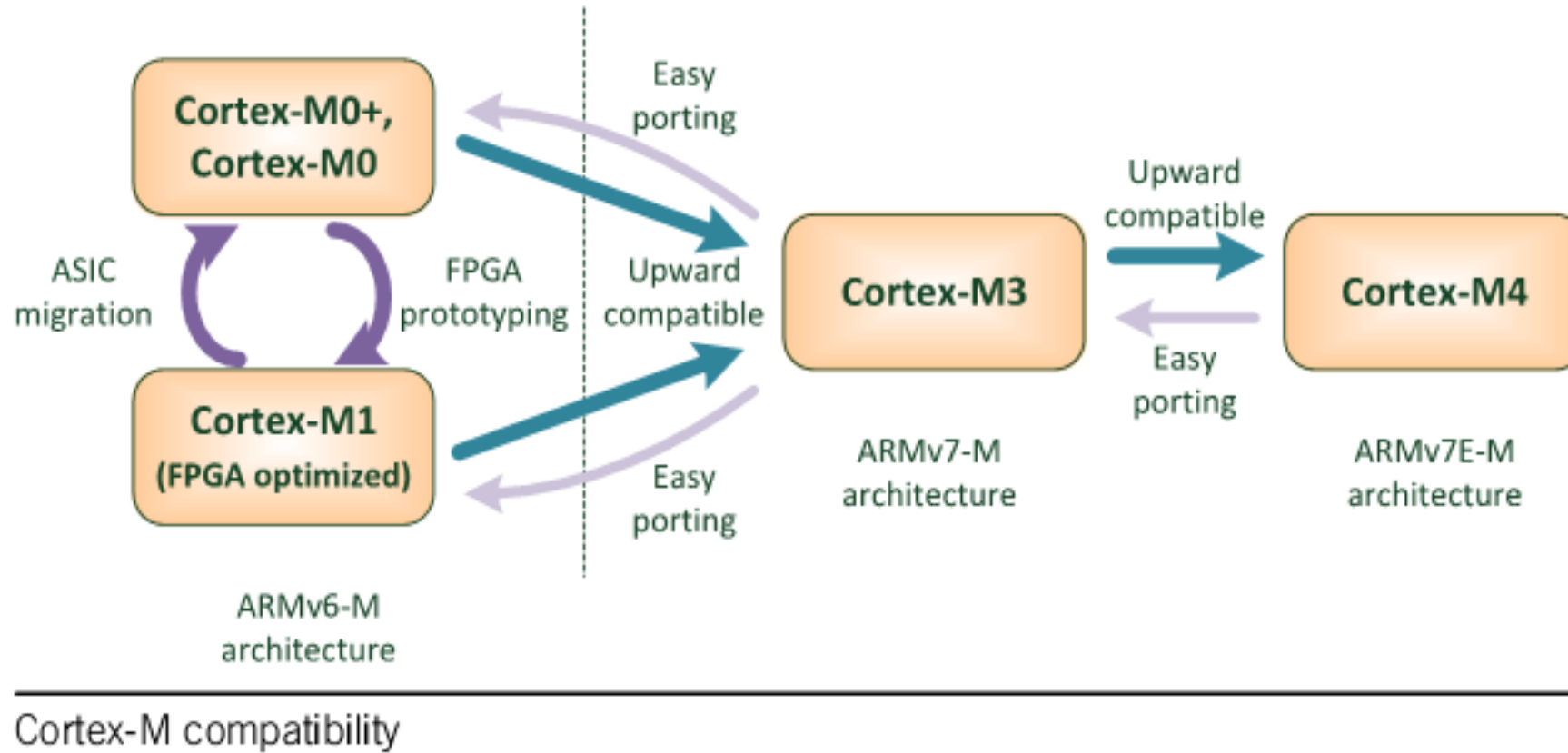


# General Information about the Cortex-M



**Embedded Processors**

# General Information about the Cortex-M

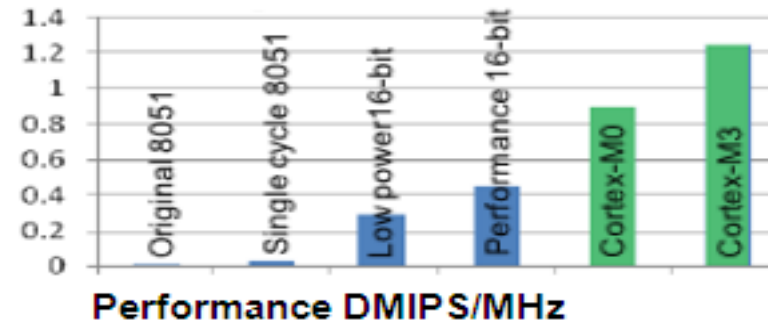
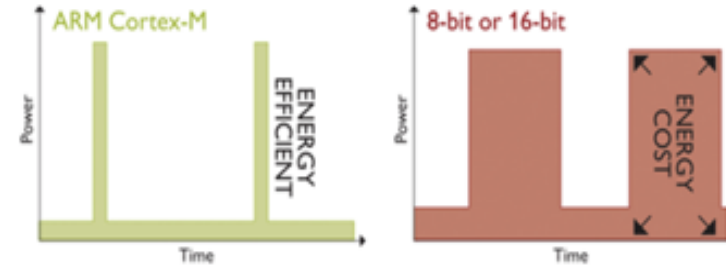


**Cortex M processor**



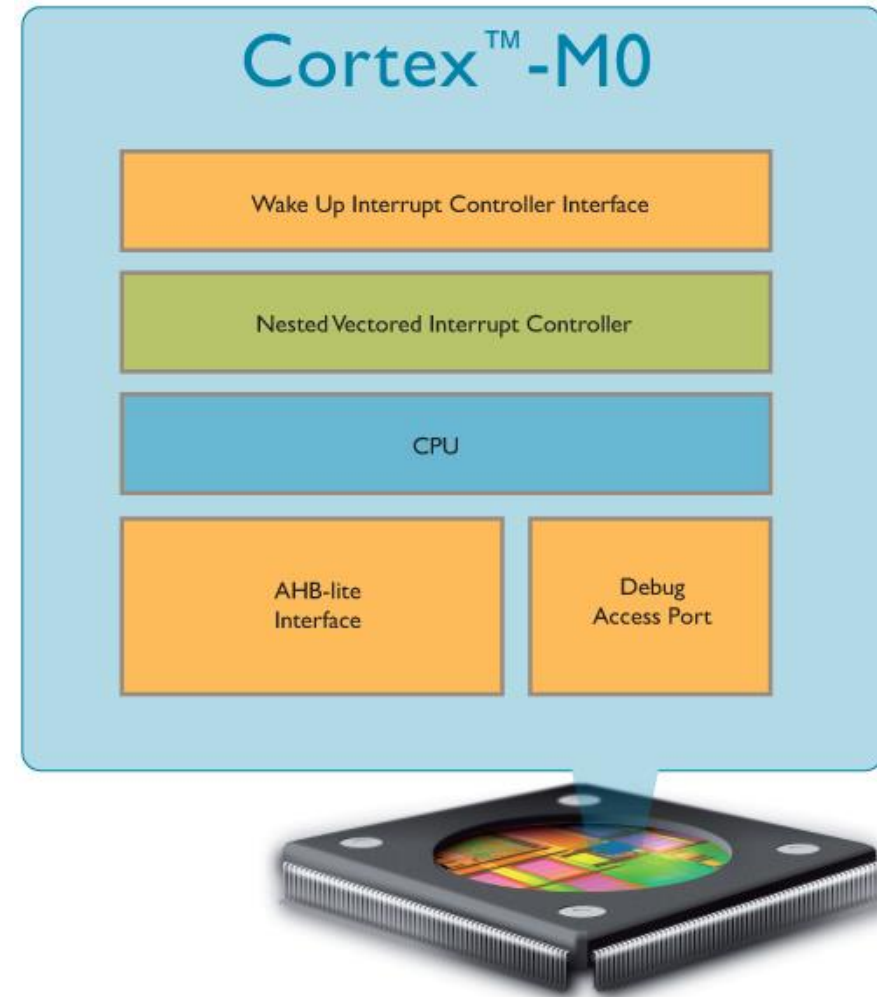
## Cortex-M Advantages:

1. Energy efficiency
2. Smaller code
3. Ease of use
4. High performance



## Cortex M0

- 32-bit RISC processor
- 3-stage pipeline von Neumann architecture
- ARMv6-M architecture
- 16-bit Thumb instruction set with Thumb-2 technology
- Load-Store Architecture
- 56 Instructions
- Low power support



# Table of contents

General Information about the Cortex-M

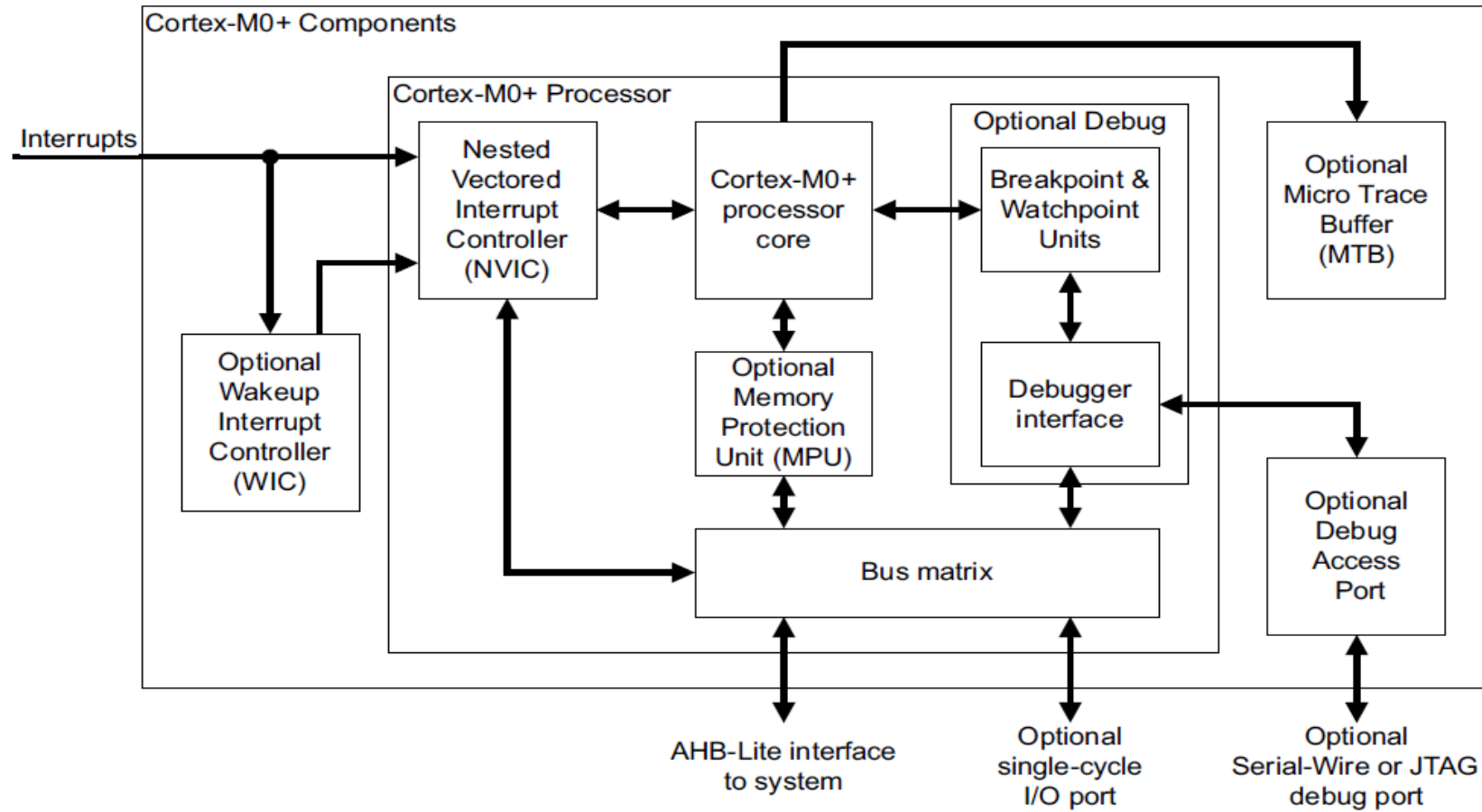
Introduction to the architecture

Programmer Model

Instruction Set

Summary

## Simplified Block Diagram



## Cortex-M0 Functional Blocks

- ARMv6-M Thumb instruction set
- NVIC: 32 external interrupt inputs
- Debug: 4 HD breakpoints, 2 watchpoints.
- Bus interfaces: 32-bit AMBA-3 AHB-Lite system interface

## **Memory model**

- 32-bit address space
- Virtual memory is not supported in ARMv6-M
- Instruction fetches are always half-word-aligned
- Data accesses are always naturally aligned

## Exception model

- Each exception has exception number, priority number and vector address
- An exception may be an interrupt or a hardware error
- Vector table base address is fixed at 0x00

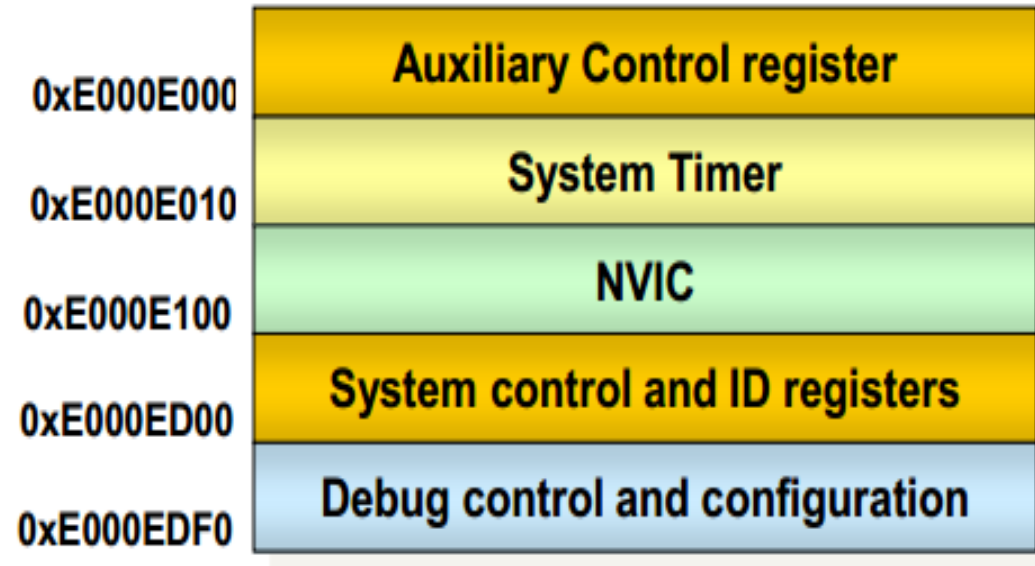
<i>Word 0</i>	Initial value of stack
<i>Word 1</i>	Vector address of exception 1
<i>Word 2</i>	Vector address of exception 2
<i>Word 47</i>	Vector address of exception 47

**Vector Table**

## **System control space**

*Consists of the following groups:*

- CPUID space.
- System control, configuration and status.
- SysTick system timer
- Nested Vectored Interrupt Controller (NVIC)





# Table of contents

General Information about the Cortex-M

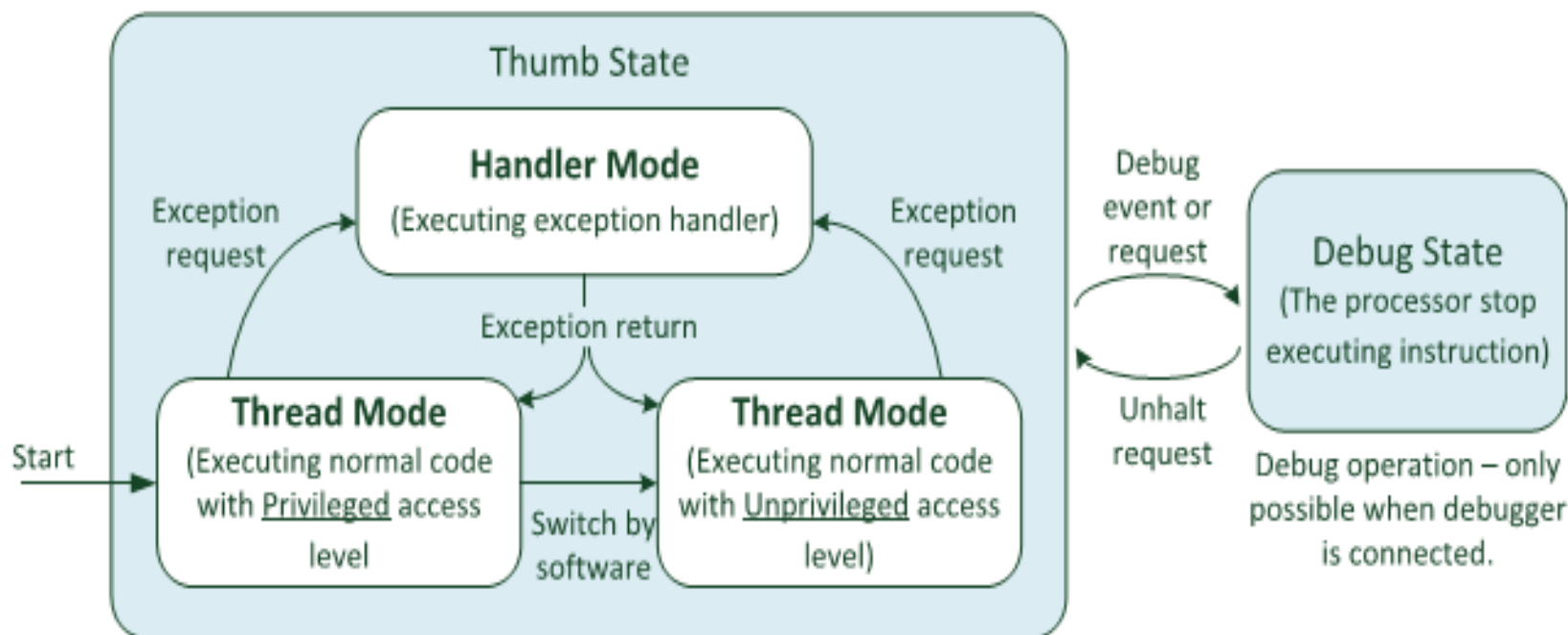
Introduction to the architecture

Programmer Model

Instruction Set

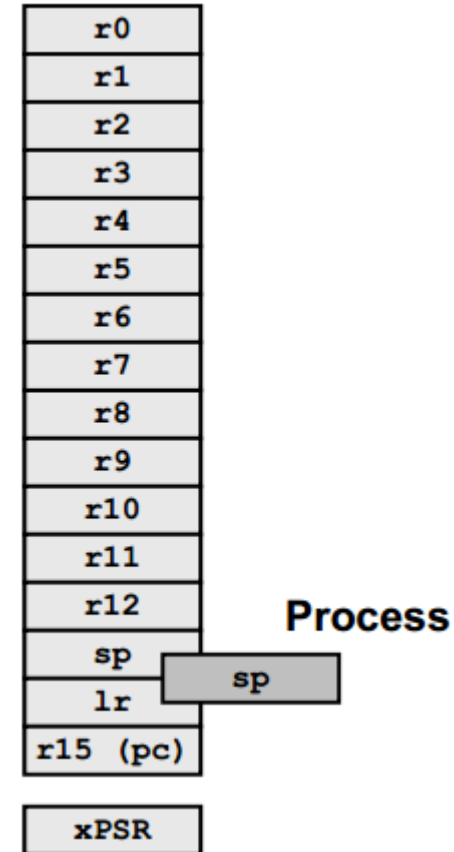
Summary

## Operation Modes & States



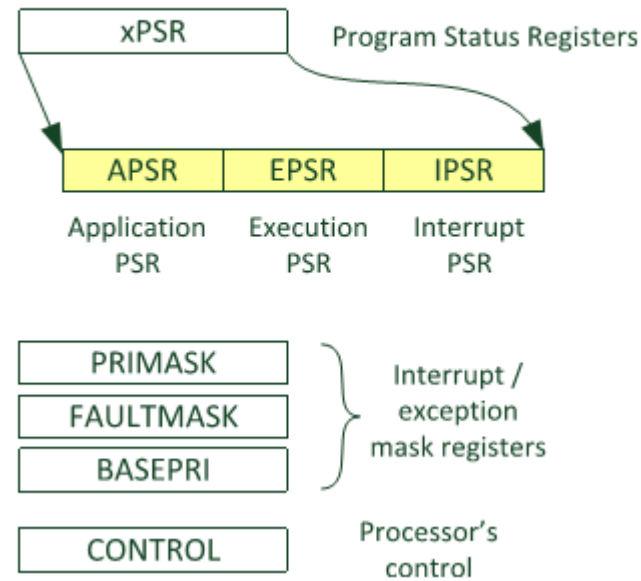
## Core Registers

- All registers are 32 bits wide
- 13 general purpose registers
- Registers r0 – r7 (Low registers)
- Registers r8 – r12 (High registers)
- 3 registers with special meaning/usage
  - ✓ Stack Pointer (SP) – r13
  - ✓ Link Register (LR) – r14
  - ✓ Program Counter (PC) – r15
- Special-purpose registers
  - ✓ xPSR shows a composite of the content of
  - ✓ APSR, IPSR, EPSR



## Special Registers

### Special Registers



APSR, IPSR, and EPSR

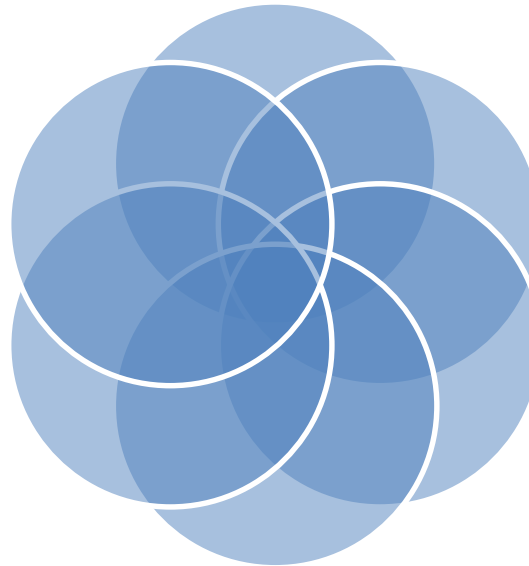
	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
xPSR	N	Z	C	V	Q	ICI/IT	T		GE*	ICI/IT		Exception Number				

# Programmer's Model

## Stack

Full descending: stack pointer indicates the last stacked item on the stack memory.

In an OS environment, ARM recommends that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack



Two stacks, two independent stack pointers.

Handler mode always uses the MSP (Main Stack Pointer)

Thread mode can use MSP (Main Stack Pointer) by default, or PSP (Process Stack Pointer).

# Table of contents

General Information about the Cortex-M

Introduction to the architecture

Programmer Model

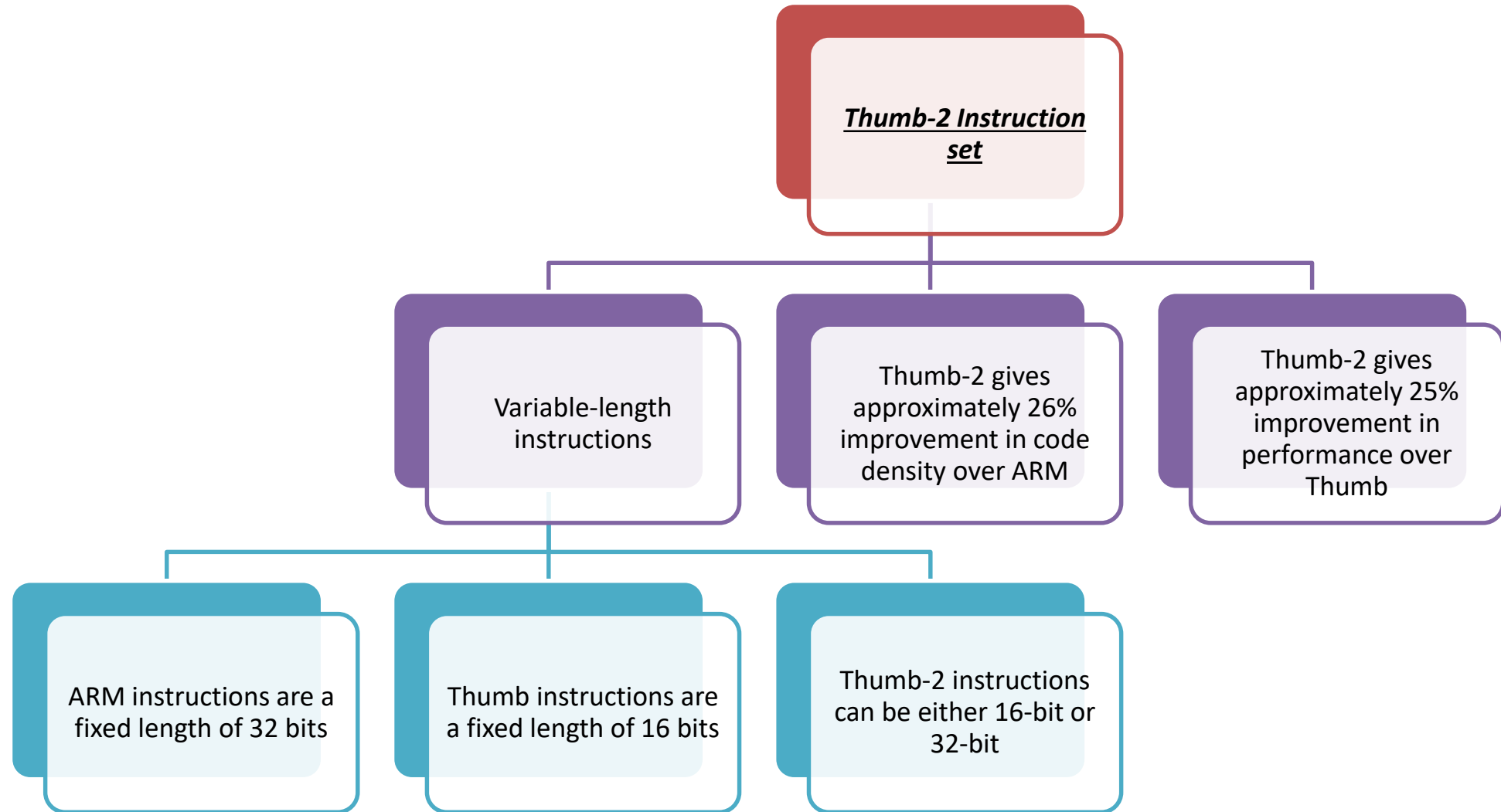
Instruction Set

Summary

No direct  
manipulation of  
memory contents

Memory must be  
loaded into the  
CPU to be  
modified, then  
written back out

The ARM  
Architecture  
is a  
**Load/Store**  
architecture





## **Cortex M0 ISA Overview:**

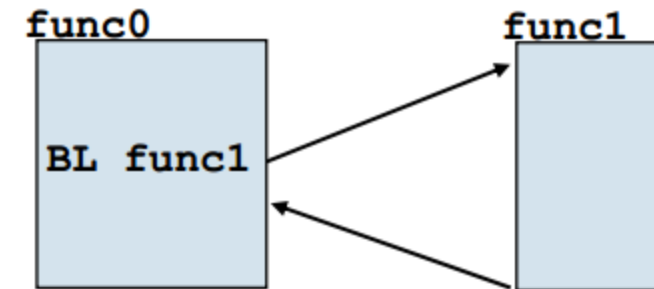
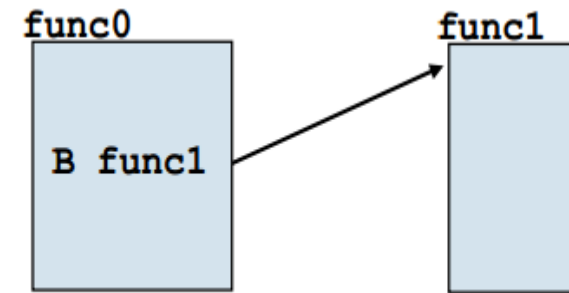
- ARMv6-M supports Thumb-2 technology  
*(The ARM instruction set is not supported)*
- Thumb-2 technology supports mixed 16-bit/32-bit instructions
- Small number of additional 32-bit instructions supported
- Conditional execution is supported
- Optimized for compilation from C
  - ✓ Thumb-2 instructions are not designed to be written by hand
  - ✓ Easy to learn due to small number of mnemonics

## *Instruction Classes*

- Branch instructions
- Data-processing instructions
- Load and store instructions
- Status register access instructions
- Miscellaneous instructions

## Branch instructions

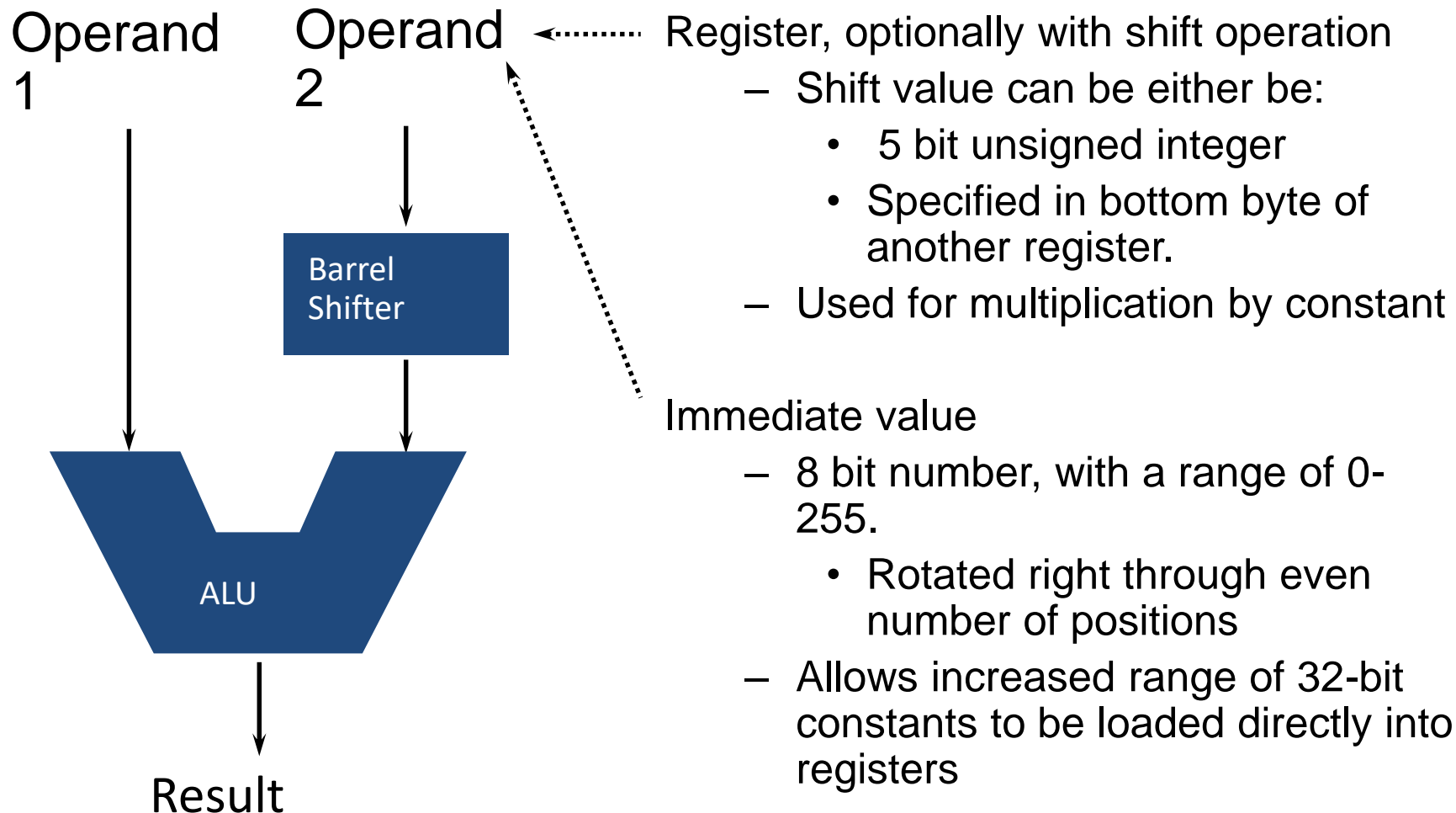
- B – Branch
  - ✓ Absolute branch to a target address, relative to Program Counter (PC)
  - ✓ +/- 256 bytes range, conditional execution supported
  - ✓ +/- 1MB range, no conditional execution supported
- BL – Branch with Link
  - ✓ Branch to a subroutine – Link register is updated
  - ✓ +/- 16MB range, relative to Program Counter (PC)



## **Data Processing Instructions:**

- Consist of:
  - Arithmetic:     **ADD   ADC   SUB   SBC   RSB   RSC**
  - Logical:             **AND   ORR   EOR   BIC**
  - Comparisons:       **CMP   CMN   TST   TEQ**
  - Data movement:     **MOV   MVN**
- These instructions only work on registers, NOT memory.
- Syntax:  
  **<Operation>{<cond>}{S} Rd, Rn, Operand2**
  - Comparisons set flags only - they do not specify Rd
  - Data movement does not specify Rn
  - Second operand is sent to the ALU via barrel shifter.

## Using Barrel Shifter:



## *Load and store instructions:*

<b>LDR</b>	<b>STR</b>	Word
<b>LDRB</b>	<b>STRB</b>	Byte
<b>LDRH</b>	<b>STRH</b>	Halfword
<b>LDRSB</b>		Signed byte load
<b>LDRSH</b>		Signed halfword load

- Memory system must support all access sizes
- Syntax:
  - **LDR**{<cond>}{<size>} Rd, <address>
  - **STR**{<cond>}{<size>} Rd, <address>

e.g. **LDREQB**

## *Status Register Access Instructions*

MRS/MSR - Move data between a general purpose register and status register

- MRS (Register  $\leftarrow$  Status Register)
- MSR (Status Register  $\leftarrow$  Register)

- Syntax:

- **LDR**{<cond>}{<size>} Rd, <address>
- **STR**{<cond>}{<size>} Rd, <address>

General Information about the Cortex-M

Introduction to the architecture

Programmer Model

Instruction Set

Summary



- All the ARM Cortex M processors are 32-bit RISC (Reduced Instruction Set Computing) processors. They have:
  - 32-bit registers
  - 32-bit internal data path
  - 32-bit bus interface
  
- The Cortex-M processors contain the core of the processor, NVIC, the SysTick timer, and optionally the floating point unit (for Cortex-M4).
  
- All the ARM Cortex -M processors are based on Thumb technology, which allows a mixture of 16-bit and 32-bit instructions to be used within

Thanks for your attention !

- This course including **Lecture Presentations, Quiz, Mock Project, Syllabus, Assignments, Answers** are copyright by FPT Software Corporation.
- This course also uses some information from external sources and non-confidential training document from Freescale, those materials comply with the original source licenses.