EE3220 System-on-Chip Design

Lecture Note 4

ARM Cortex-M4 Processor Architecture

CityU Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Outline

- ARM Architectures and Processors
  - What is ARM Architecture
  - ARM Processor Families
  - ARM Cortex-M Series
  - Cortex-M4 Processor
  - ARM Processor vs. ARM Architectures
- ARM Cortex-M4 Processor
  - Cortex-M4 Processor Overview
  - Cortex-M4 Block Diagram
  - Cortex-M4 Registers

ARM Architectures and Processors

What is ARM Architecture

- ARM architecture is a family of RISC-based processor architectures
  - Well-known for its power efficiency;
  - Hence widely used in mobile devices, such as smartphones and tablets
  - Designed and licensed to a wide eco-system by ARM
- ARM Holdings
  - The company designs ARM-based processors;
  - Does not manufacture, but licenses designs to semiconductor partners who add their own Intellectual Property (IP) on top of ARM's IP, fabricate and sell to customers;
  - Also offer other IP apart from processors, such as physical IPs, interconnect IPs, graphics cores, and development tools.

ARM IP License

- Arm Flexible Access provides up-front, zero or low cost access to a wide range of Arm IP, tools, and training.
- License fees are calculated only on the IP included in the final SoC design.
- Flexible Access includes free use of thousands of Arm Artisan Physical IP libraries for implementing silicon for manufacture across the broadest range of foundries and process technology nodes.

Artisan Physical IP – Free Library Program

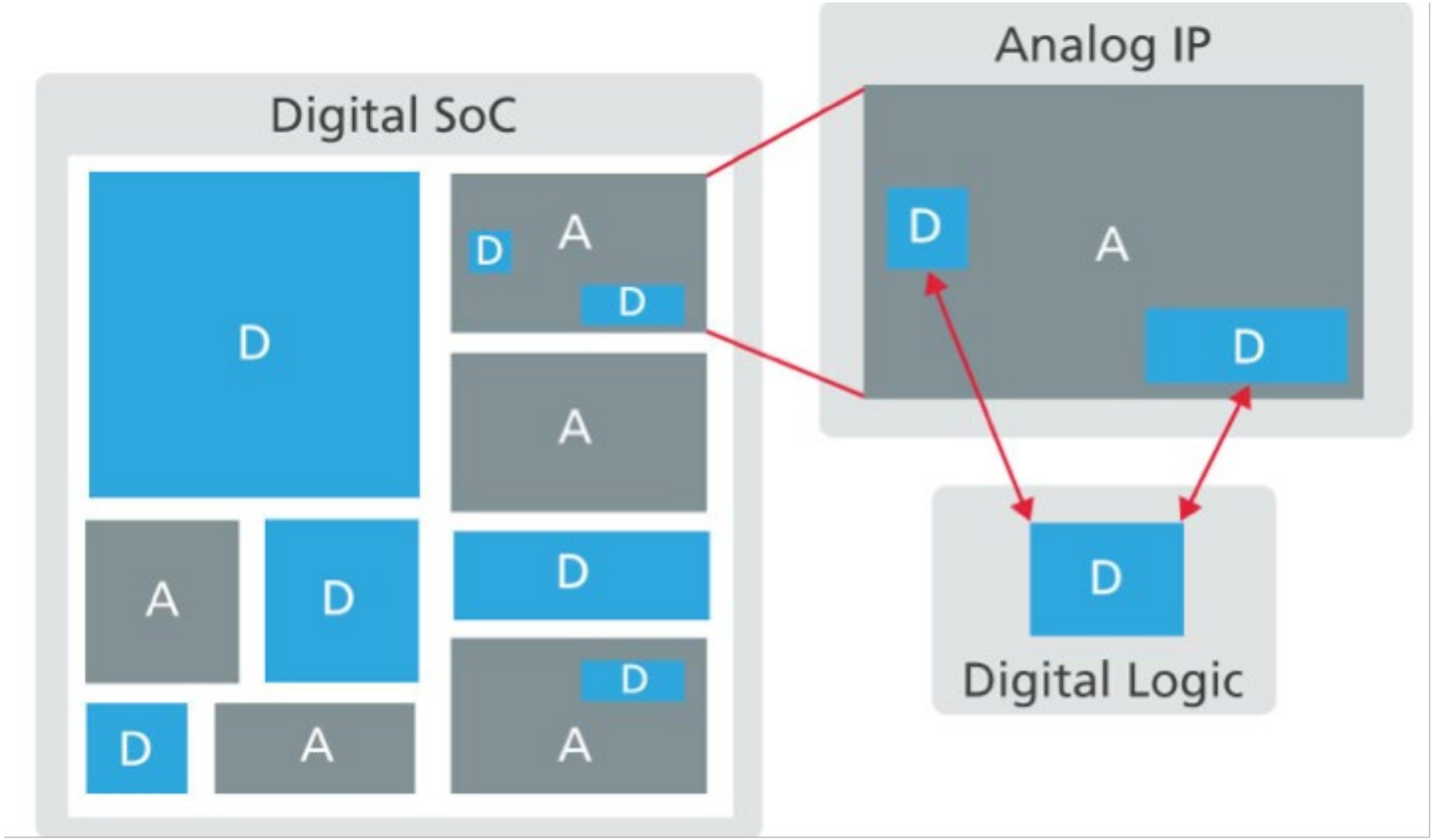
for details of the thousands of Physical IP libraries included in the Artisan Physical IP - Free Library Program see https://www.arm.com/products/silicon-ip-physical

	5 nm	7 nm	12 nm	14 nm	22 nm	28 nm	40 nm	45 nm	55 nm	65 nm	80 nm	90 nm	110 nm	130 nm	150 nm	152 nm	160 nm	180 nm	250 nm
TSMC						•	•		•	•	•	•	•	•	•	•	•	•	•
Samsung	•	•		•		•		•		•									
Global Foundries			•	•		•	•	•	•	•		•	•	•				•	•
UMC					•	•	•		•	•	•	•		•	•			•	•
SMIC						•	•			•		•	•	•	•			•	
XMC									•										
SK hynix												•							
Silterra													•	•	•			•	
HHGrace													•	•				•	
DB HiTek													•	•				•	
Vanguard													•					•	•
MagnaChip														•				•	
CSMC														•					
TowerJazz														•				•	
HeJian																	•	•	
1st Silicon																		•	•
HHNEC																		•	

https://www.arm.com/products/flexible-access

Intellectual Property vs. System-on-Chip

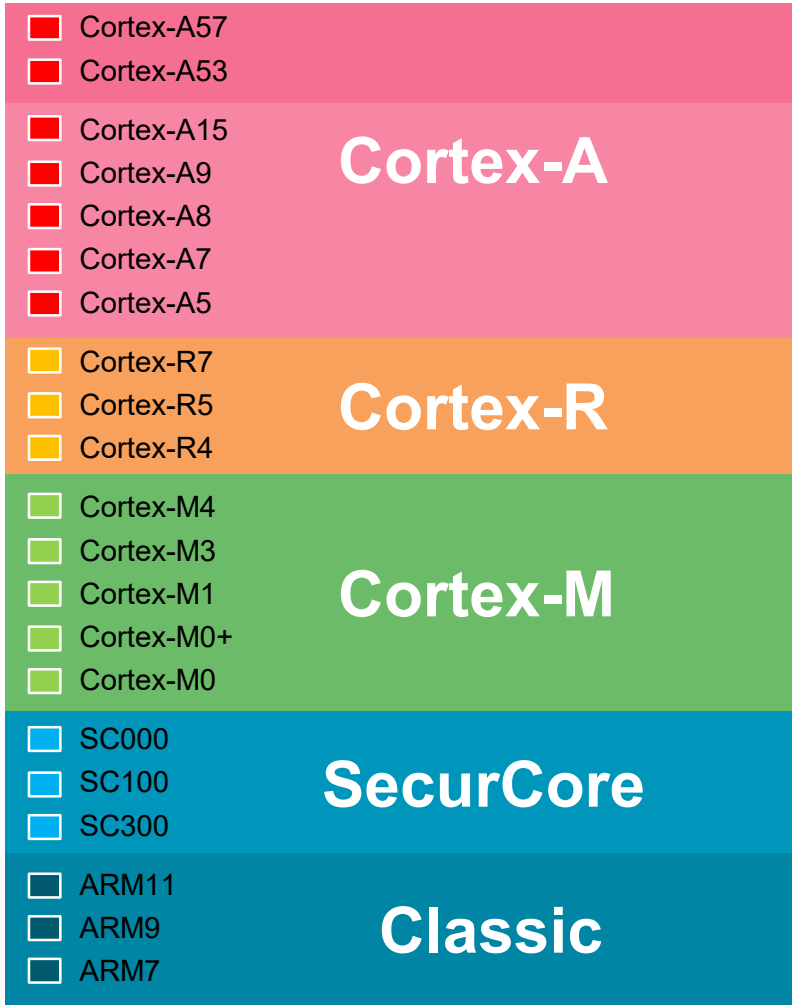
- Intellectual Property (IP) has become the major building blocks of complex, highly integrated systems on chips (SoCs).



https://semiengineering.com/its-all-ip-in-an-soc/

ARM Processor Families

- Cortex-A series (Application)
  - High performance processors capable of full Operating System (OS) support;
  - Applications include smartphones, digital TV, smart books, home gateways etc.
- Cortex-R series (Real-time)
  - High performance for real-time applications;
  - High reliability
  - Applications include automotive braking system, powertrains etc.
- Cortex-M series (Microcontroller)
  - Cost-sensitive solutions for deterministic microcontroller applications;
  - Applications include microcontrollers, mixed signal devices, smart sensors, automotive body electronics and airbags;
- SecurCore series
  - High security applications.
- Previous classic processors
  - Include ARM7, ARM9, ARM11 families

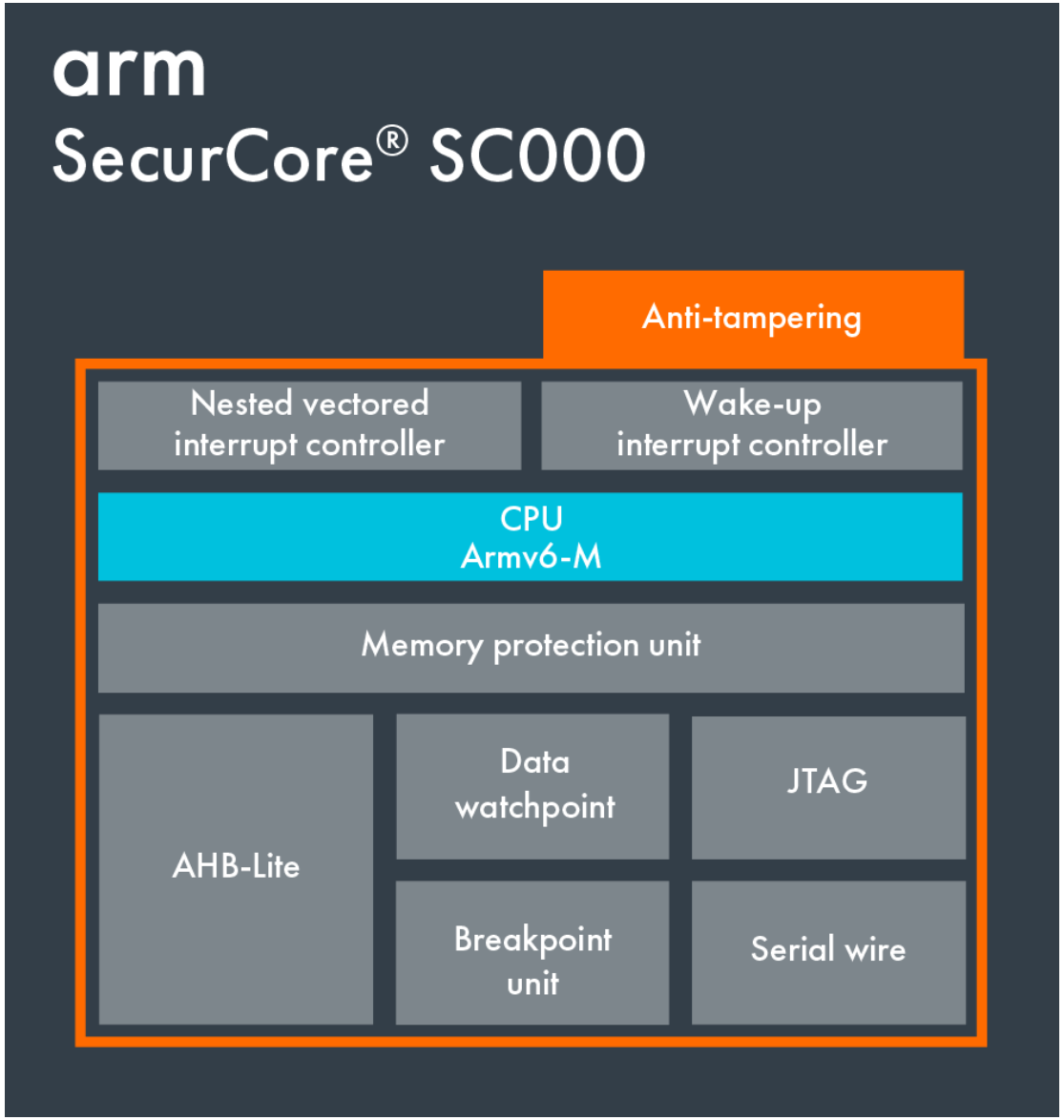


Cortex
Low-Power Leadership from ARM

As of Dec 2013

CityU Department of Electrical Engineering
香港城市大學 City University of Hong Kong

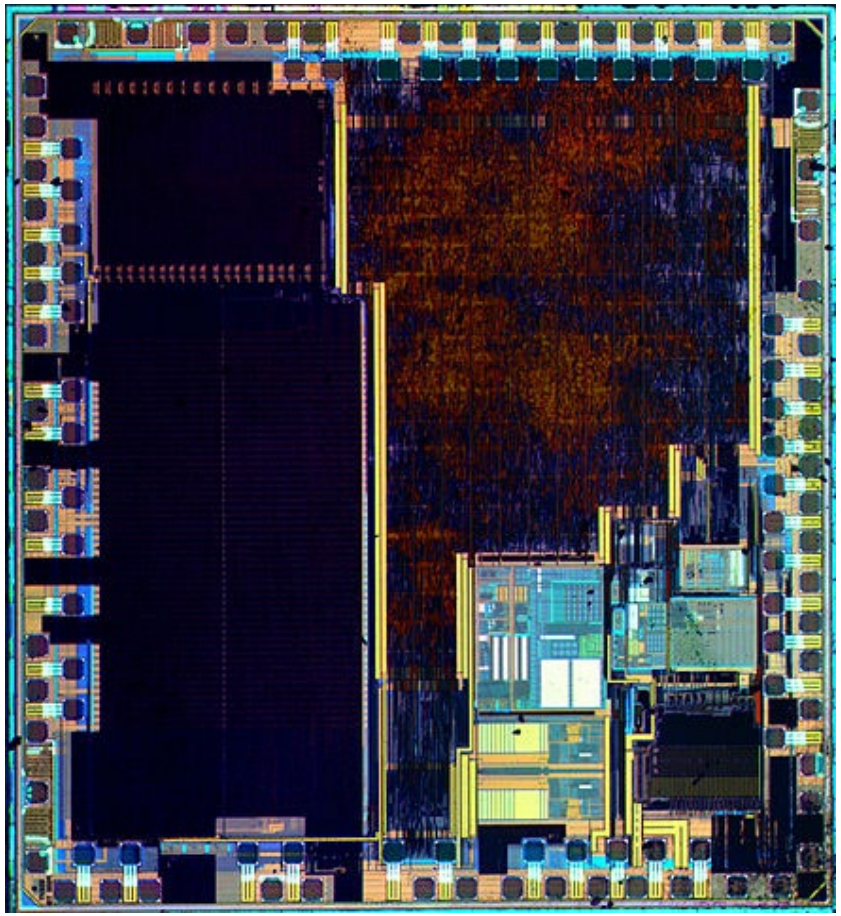
Arm SecurCore SC000 processor

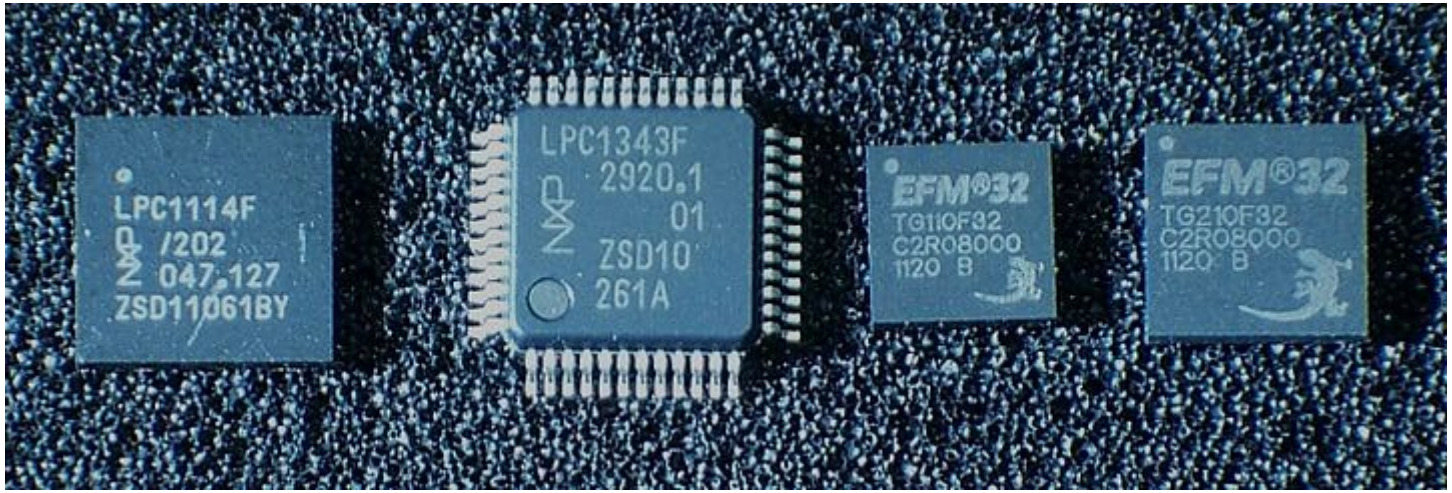


- Arm SecurCore processor is designed for smartcard and embedded security applications.
- Arm SecurCore processors are the most widely licensed 32-bit processors for smartcards worldwide.
- The programmers' model is the same as the Cortex-M0.

ARM Cortex-M3 - Die

- Die photo from a STM32F100C4T6B IC, 24 MHz ARM Cortex-M3 microcontroller with 16 KB flash memory, 4 KB RAM. Manufactured by STMicroelectronics.

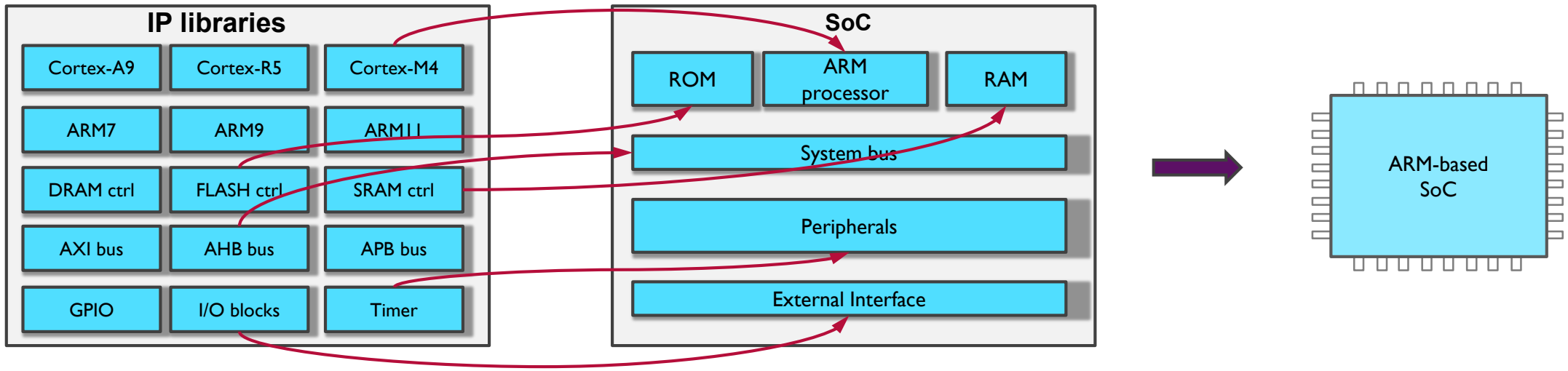




ARM Cortex-M0 and Cortex-M3 microcontroller ICs from NXP and Silicon Labs

Design an ARM-based SoC

- Select a set of IP cores from ARM and/or other third-party IP vendors
- Integrate IP cores into a single chip design
- Give design to semiconductor foundries for chip fabrication



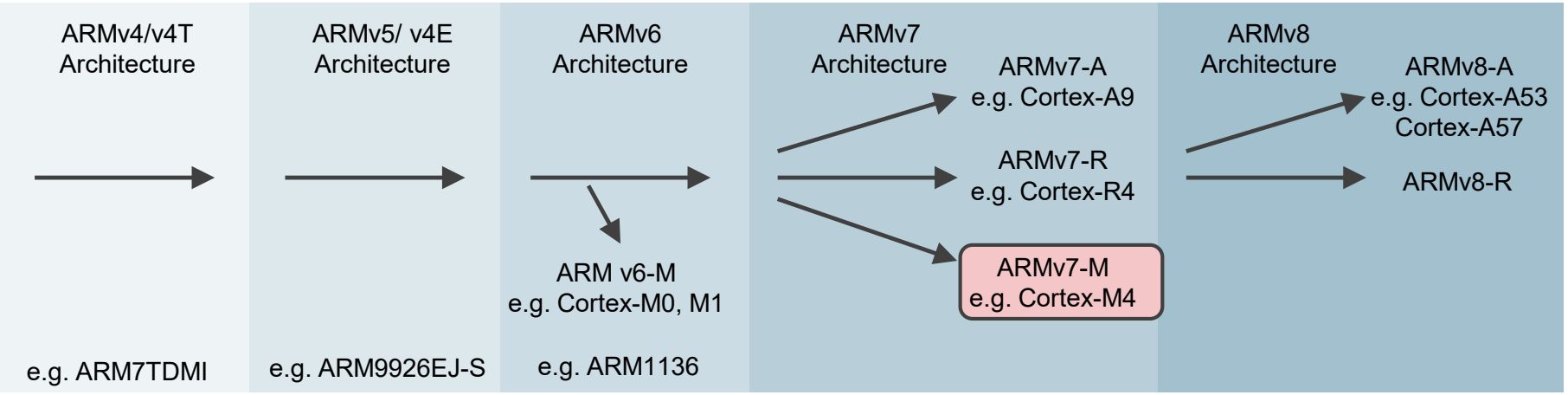
ARM Cortex-M Series

- Cortex-M series: Cortex-M0, M0+, M1, M3, M4.
- Energy-efficiency
  - Lower energy cost, longer battery life
- Smaller code
  - Lower silicon costs
- Ease of use
  - Faster software development and reuse
- Embedded applications
  - Smart metering, human interface devices, automotive and industrial control systems, white goods, consumer products and medical instrumentation

As of Dec 2013

ARM Processors vs. ARM Architectures

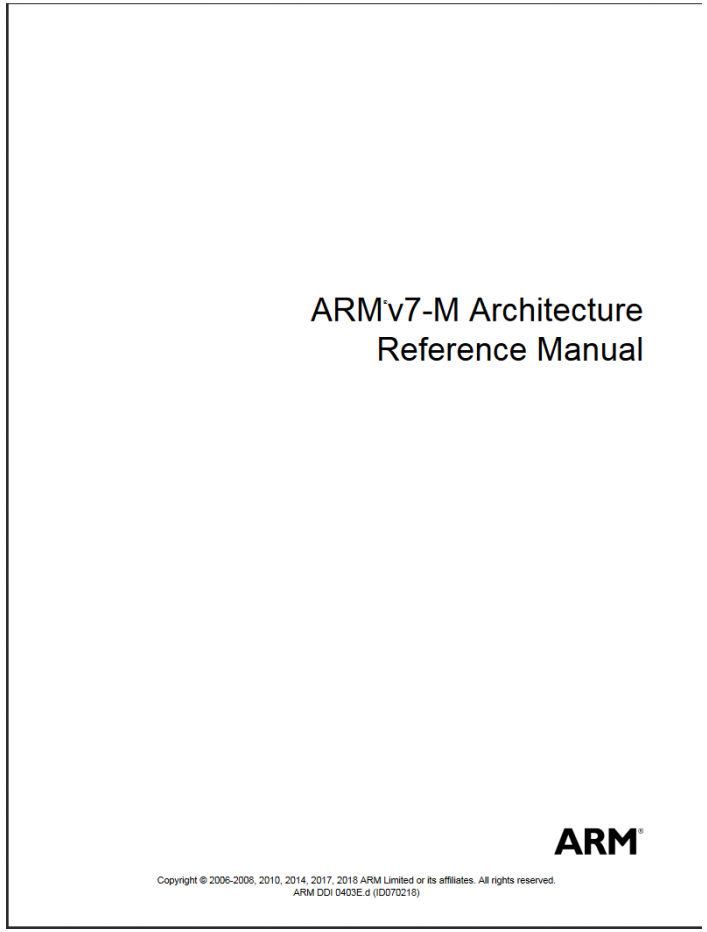
- ARM architecture
  - Describes the details of instruction set, programmer's model, exception model, and memory map
  - Documented in the Architecture Reference Manual
- ARM processor
  - Developed using one of the ARM architectures
  - More implementation details, such as timing information
  - Documented in processor's Technical Reference Manual

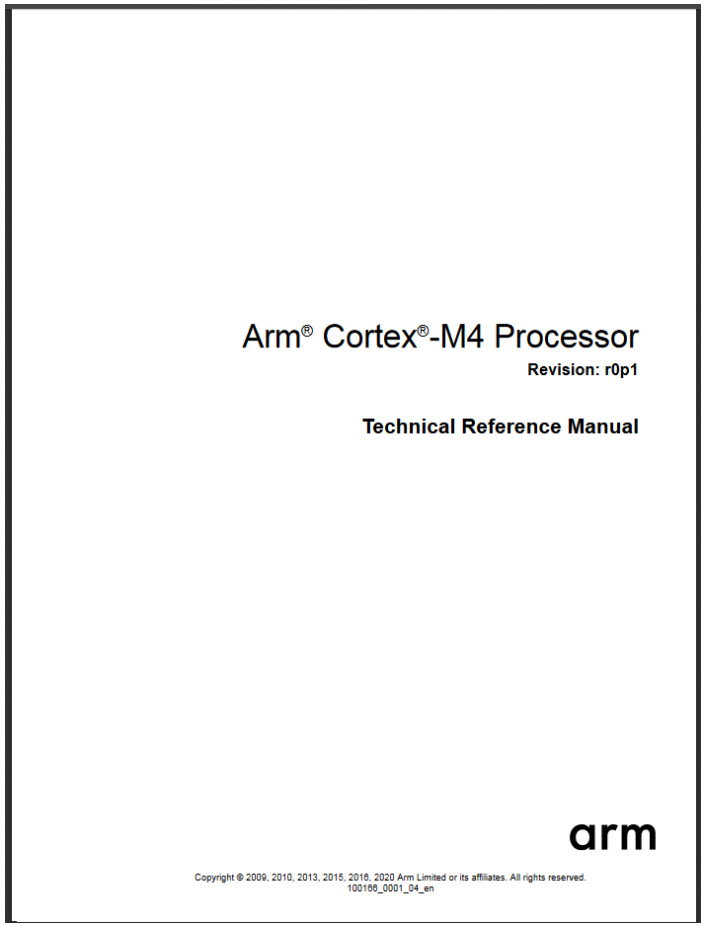


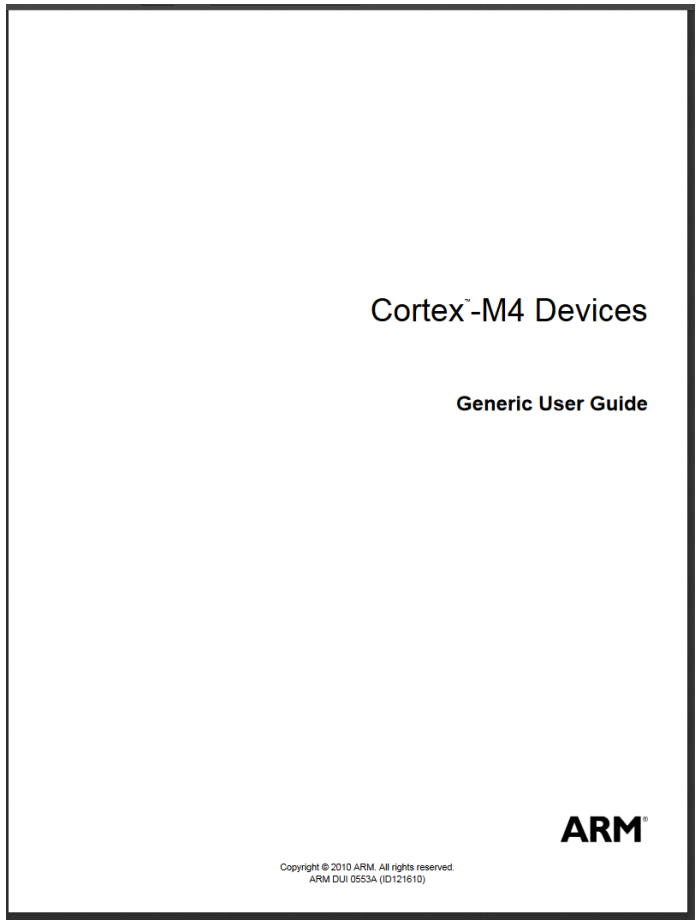
As of Dec 2013

Architecture Reference Manual and User Guide

- 850 pages, 100+ pages, 270 pages







ARM Cortex-M Series Family

Processor	ARM Architecture	Core Architecture	Thumb®	Thumb®-2	Hardware Multiply	Hardware Divide	Saturated Math	DSP Extensions	Floating Point
Cortex-M0	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	Software	No
Cortex-M0+	ARMv6-M	Von Neumann	Most	Subset	1 or 32 cycle	No	No	Software	No
Cortex-M1	ARMv6-M	Von Neumann	Most	Subset	3 or 33 cycle	No	No	Software	No
Cortex-M3	ARMv7-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	Software	No
Cortex-M4	ARMv7E-M	Harvard	Entire	Entire	1 cycle	Yes	Yes	Hardware	Optional

Harvard: computer architecture with physically separate storage and signal pathways for instructions and data.

CityU Department of Electrical Engineering
香港城市大學 City University of Hong Kong

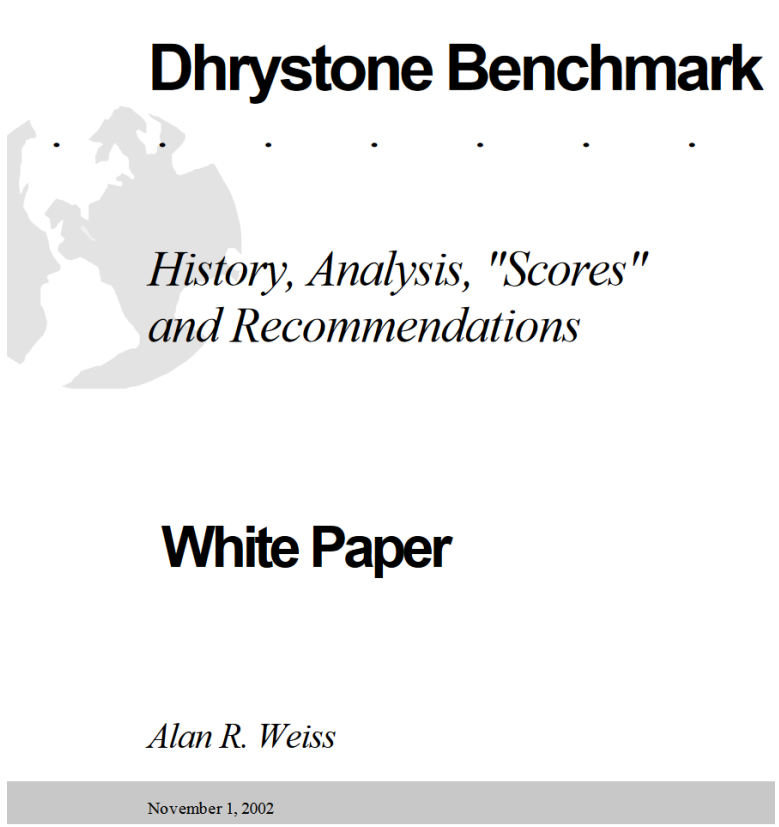
ARM Cortex-M4 Processor Overview

Cortex-M4 Processor Overview

- Cortex-M4 Processor
  - Introduced in 2010
  - Designed with a large variety of highly efficient signal processing features
  - Features extended single-cycle multiply accumulate instructions, optimized SIMD arithmetic, saturating arithmetic and an optional Floating Point Unit.
- High Performance Efficiency
  - 1.25 DMIPS/MHz (Dhrystone Million Instructions Per Second / MHz) at the order of μWatts / MHz
- Low Power Consumption
  - Longer battery life – especially critical in mobile products
- Enhanced Determinism
  - The critical tasks and interrupt routines can be served quickly in a known number of cycles

Dhrystone Million Instructions Per Second / MHz

- Dhrystone is a synthetic computing benchmark program developed in 1984 by Reinhold P. Weicker intended to be representative of system (integer) programming.
- The Dhrystone grew to become representative of general processor performance.




```
*    The average number of parameters in procedure or function calls
*    is 1.82 (not counting the function values aX *
*
*  2. Operators
*  ------------
*                            number     approximate
*                                       percentage
*
*    Arithmetic               32           50.8
*
*       +                      21           33.3
*       -                      7            11.1
*       *                      3             4.8
*       / (int div)            1             1.6
*
*    Comparison               27           42.8
*
*       ==                     9            14.3
*       /=                     4             6.3
*       >                      1             1.6
*       <                      3             4.8
*       >=                     1             1.6
*       <=                     9            14.3
*
*    Logic                     4            6.3
*
*       && (AND-THEN)          1            1.6
*       |  (OR)                1            1.6
*       !  (NOT)               2            3.2
*
*                            --           -----
*                            63           100.1
*
*
*  3. Operand Type (counted once per operand reference):
*  ---------------
*                            number     approximate
*                                       percentage
*
*     Integer                175         72.3 %
*     Character               45         18.6 %
*     Pointer                 12          5.0 %
*     String30                6           2.5 %
*     Array                   2           0.8 %
*     Record                  2           0.8 %
*                            ---         -------
*                            242         100.0 %
*
```

```
/****************** "DHRYSTONE" Benchmark Program ************************/
#define Version "C, Version 2.2"
/*  File:       dhry_1.c (part 2 of 3)
 *  Author:     Reinhold P. Weicker
 *              Siemens Nixdorf, Paderborn/Germany
 *              weicker@specbench.org
 *  Date:       May 25, 1988
 *  Modified:   Steven Pemberton, CWI, Amsterdam; Steven.Pemberton@cwi.nl
 *  Date:       October, 1993; March 1995
 *              Included both files into one source, that gets compiled
 *              in two passes. Made program auto-compiling, and auto-running,
 *              and generally made it much easier to use.
 *
 *              Original Version (in Ada) published in
 *              "Communications of the ACM" vol. 27., no. 10 (Oct. 1984),
 *              pp. 1013 - 1030, together with the statistics
 *              on which the distribution of statements etc. is based.
 .
```

Cortex-M4 Processor Features

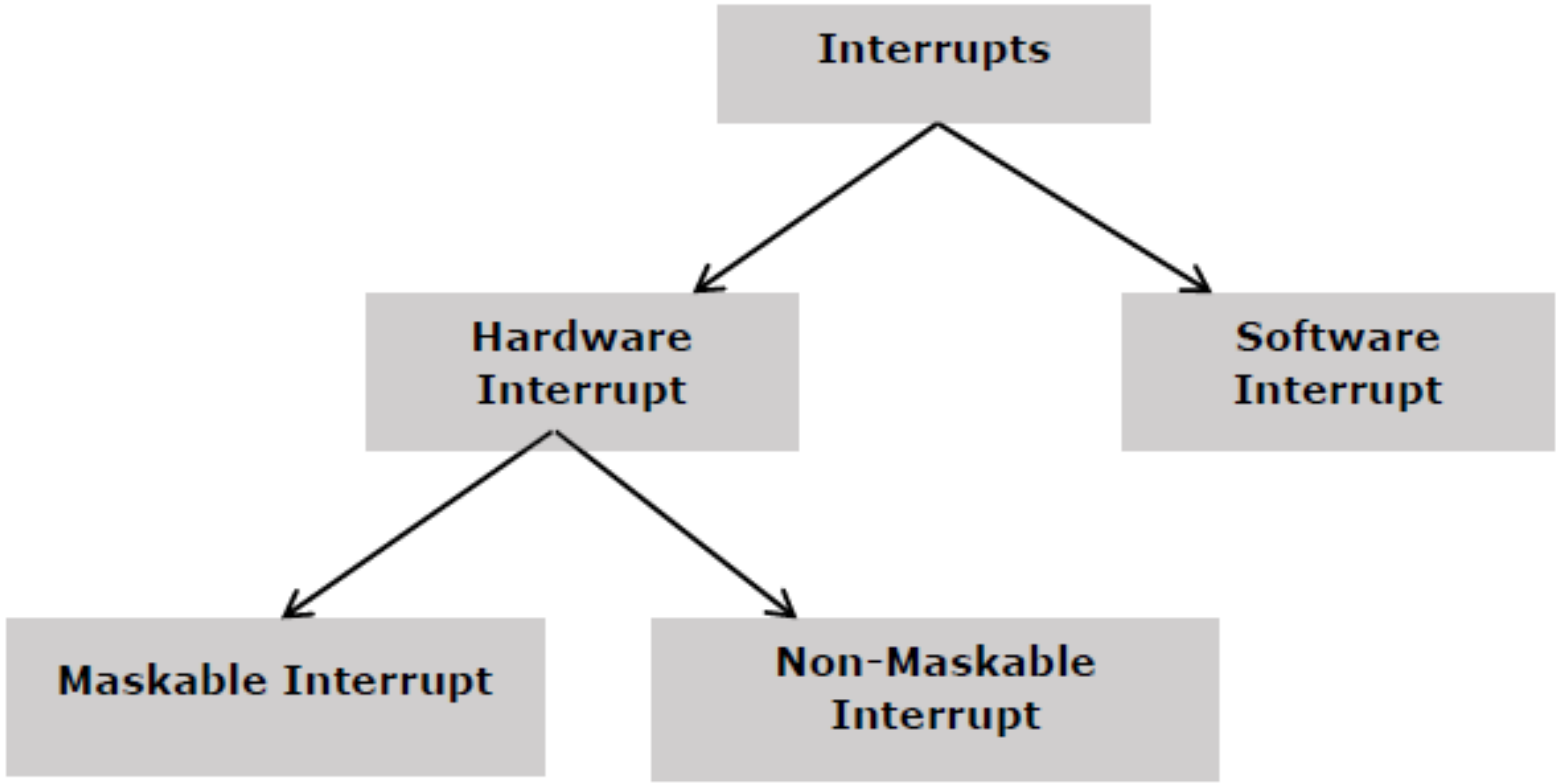
- 32-bit Reduced Instruction Set Computing (RISC) processor
- Harvard architecture
  - Separated data bus and instruction bus
- Instruction set
  - Include the entire Thumb®-1 (16-bit) and Thumb®-2 (16/ 32-bit) instruction sets
- 3-stage + branch speculation pipeline
- Performance efficiency
  - 1.25 – 1.95 DMIPS/MHz (Dhrystone Million Instructions Per Second / MHz)
- Supported Interrupts
  - Non-maskable Interrupt (NMI) + 1 to 240 physical interrupts
  - 8 to 256 interrupt priority levels

Thumb-1 16-bit Instruction Set

Operation		§	Assembler	Updates	Action	Notes
Move	Immediate		`MOVS Rd, #<imm>`	N Z	Rd := imm	imm range 0-255.
	Lo to Lo		`MOVS Rd, Rm`	N Z	Rd := Rm	Synonym of `LSLS Rd, Rm, #0`
	Hi to Lo, Lo to Hi, Hi to Hi		`MOV Rd, Rm`		Rd := Rm	Not Lo to Lo.
	Any to Any	6	`MOV Rd, Rm`		Rd := Rm	Any register to any register.
Add	Immediate 3		`ADDS Rd, Rn, #<imm>`	N Z C V	Rd := Rn + imm	imm range 0-7.
	All registers Lo		`ADDS Rd, Rn, Rm`	N Z C V	Rd := Rn + Rm	
	Hi to Lo, Lo to Hi, Hi to Hi		`ADD Rd, Rd, Rm`		Rd := Rd + Rm	Not Lo to Lo.
	Any to Any	T2	`ADD Rd, Rd, Rm`		Rd := Rd + Rm	Any register to any register.
	Immediate 8		`ADDS Rd, Rd, #<imm>`	N Z C V	Rd := Rd + imm	imm range 0-255.
	With carry		`ADCS Rd, Rd, Rm`	N Z C V	Rd := Rd + Rm + C-bit	
	Value to SP		`ADD SP, SP, #<imm>`		SP := SP + imm	imm range 0-508 (word-aligned).
	Form address from SP		`ADD Rd, SP, #<imm>`		Rd := SP + imm	imm range 0-1020 (word-aligned).
	Form address from PC		`ADR Rd, <label>`		Rd := label	label range PC to PC+1020 (word-aligned).
Subtract	Lo and Lo		`SUBS Rd, Rn, Rm`	N Z C V	Rd := Rn – Rm	
	Immediate 3		`SUBS Rd, Rn, #<imm>`	N Z C V	Rd := Rn – imm	imm range 0-7.
	Immediate 8		`SUBS Rd, Rd, #<imm>`	N Z C V	Rd := Rd – imm	imm range 0-255.
	With carry		`SBCS Rd, Rd, Rm`	N Z C V	Rd := Rd – Rm – NOT C-bit	
	Value from SP		`SUB SP, SP, #<imm>`		SP := SP – imm	imm range 0-508 (word-aligned).
	Negate		`RSBS Rd, Rn, #0`	N Z C V	Rd := – Rn	Synonym: `NEGS Rd, Rn`
Multiply	Multiply		`MULS Rd, Rm, Rd`	N Z * *	Rd := Rm * Rd	* C and V flags unpredictable in §4T, unchanged in §5T and above
Compare			`CMP Rn, Rm`	N Z C V	update APSR flags on Rn – Rm	Can be Lo to Lo, Lo to Hi, Hi to Lo, or Hi to Hi.
	Negative		`CMN Rn, Rm`	N Z C V	update APSR flags on Rn + Rm	
	Immediate		`CMP Rn, #<imm>`	N Z C V	update APSR flags on Rn – imm	imm range 0-255.
Logical	AND		`ANDS Rd, Rd, Rm`	N Z	Rd := Rd AND Rm	
	Exclusive OR		`EORS Rd, Rd, Rm`	N Z	Rd := Rd EOR Rm	
	OR		`ORRS Rd, Rd, Rm`	N Z	Rd := Rd OR Rm	
	Bit clear		`BICS Rd, Rd, Rm`	N Z	Rd := Rd AND NOT Rm	
	Move NOT		`MVNS Rd, Rd, Rm`	N Z	Rd := NOT Rm	
	Test bits		`TST Rn, Rm`	N Z	update APSR flags on Rn AND Rm	
Shift/rotate	Logical shift left		`LSLS Rd, Rm, #<shift>`	N Z C*	Rd := Rm << shift	Allowed shifts 0-31. * C flag unaffected if shift is 0.
			`LSLS Rd, Rd, Rs`	N Z C*	Rd := Rd << Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Logical shift right		`LSRS Rd, Rm, #<shift>`	N Z C	Rd := Rm >> shift	Allowed shifts 1-32.
			`LSRS Rd, Rd, Rs`	N Z C*	Rd := Rd >> Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Arithmetic shift right		`ASRS Rd, Rm, #<shift>`	N Z C	Rd := Rm ASR shift	Allowed shifts 1-32.
			`ASRS Rd, Rd, Rs`	N Z C*	Rd := Rd ASR Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.
	Rotate right		`RORS Rd, Rd, Rs`	N Z C*	Rd := Rd ROR Rs[7:0]	* C flag unaffected if Rs[7:0] is 0.

Types of Interrupts

- NMI: Hardware Interrupts that cannot be ignored, cannot be disabled by the CPU.
- Usually used for higher priority tasks, such as watchdog timers.

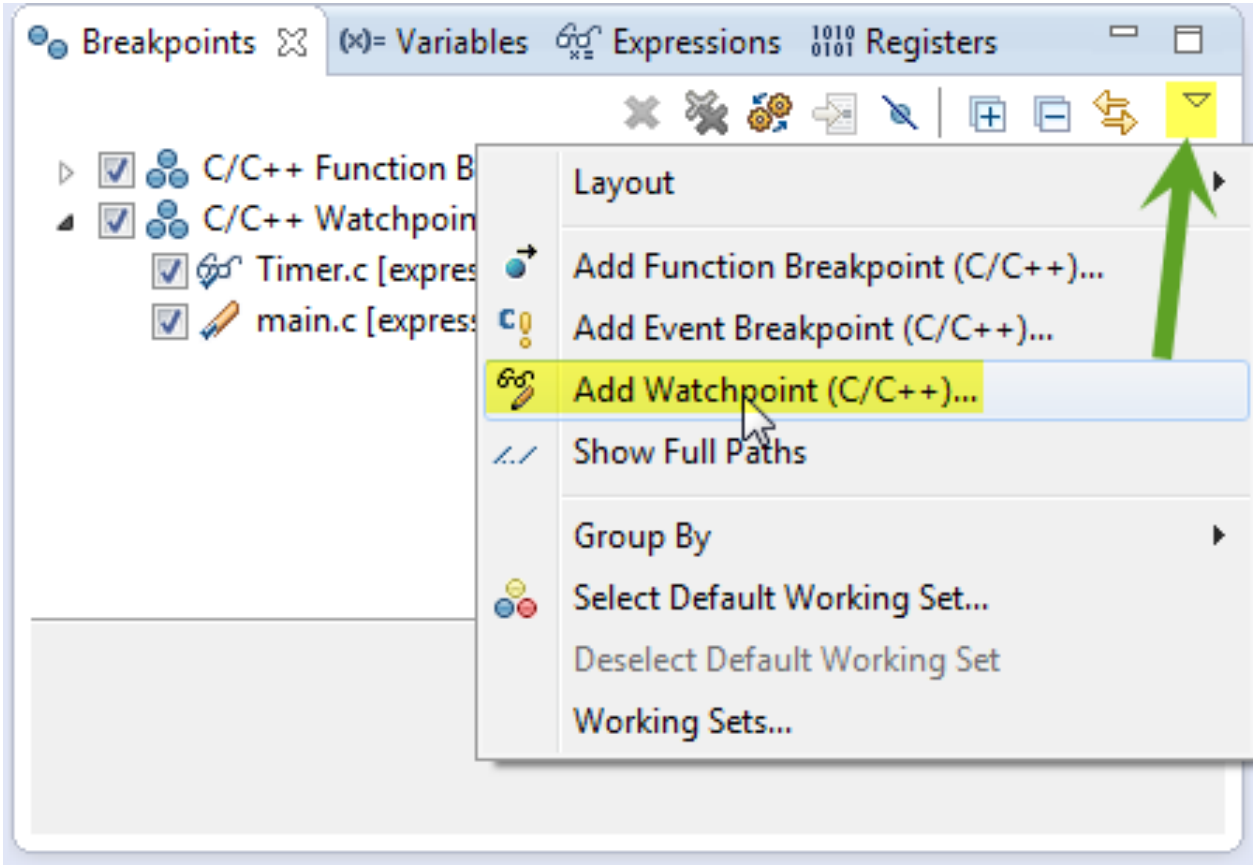


Cortex-M4 Processor Features

- Supports Sleep Modes
  - Up to 240 Wake-up Interrupts
  - Integrated WFI (Wait For Interrupt) and WFE (Wait For Event) Instructions and Sleep On Exit capability (to be covered in more detail later)
  - Sleep & Deep Sleep Signals
  - Optional Retention Mode with ARM Power Management Kit
- Enhanced Instructions
  - Hardware Divide (2-12 Cycles)
  - Single-Cycle 16, 32-bit MAC, Single-cycle dual 16-bit MAC
  - 8, 16-bit SIMD arithmetic
- Debug
  - Optional JTAG & Serial-Wire Debug (SWD) Ports
  - Up to 8 Breakpoints and 4 Watchpoints
- Memory Protection Unit (MPU)
  - Optional 8 region MPU with sub regions and background region

Debug – Breakpoint / Watchpoint

- We can set breakpoints on an executable line of a program.
- When we debug a program, the execution is suspended *before* that line of code is executed.
- We use watchpoint to observe changes to a particular field.

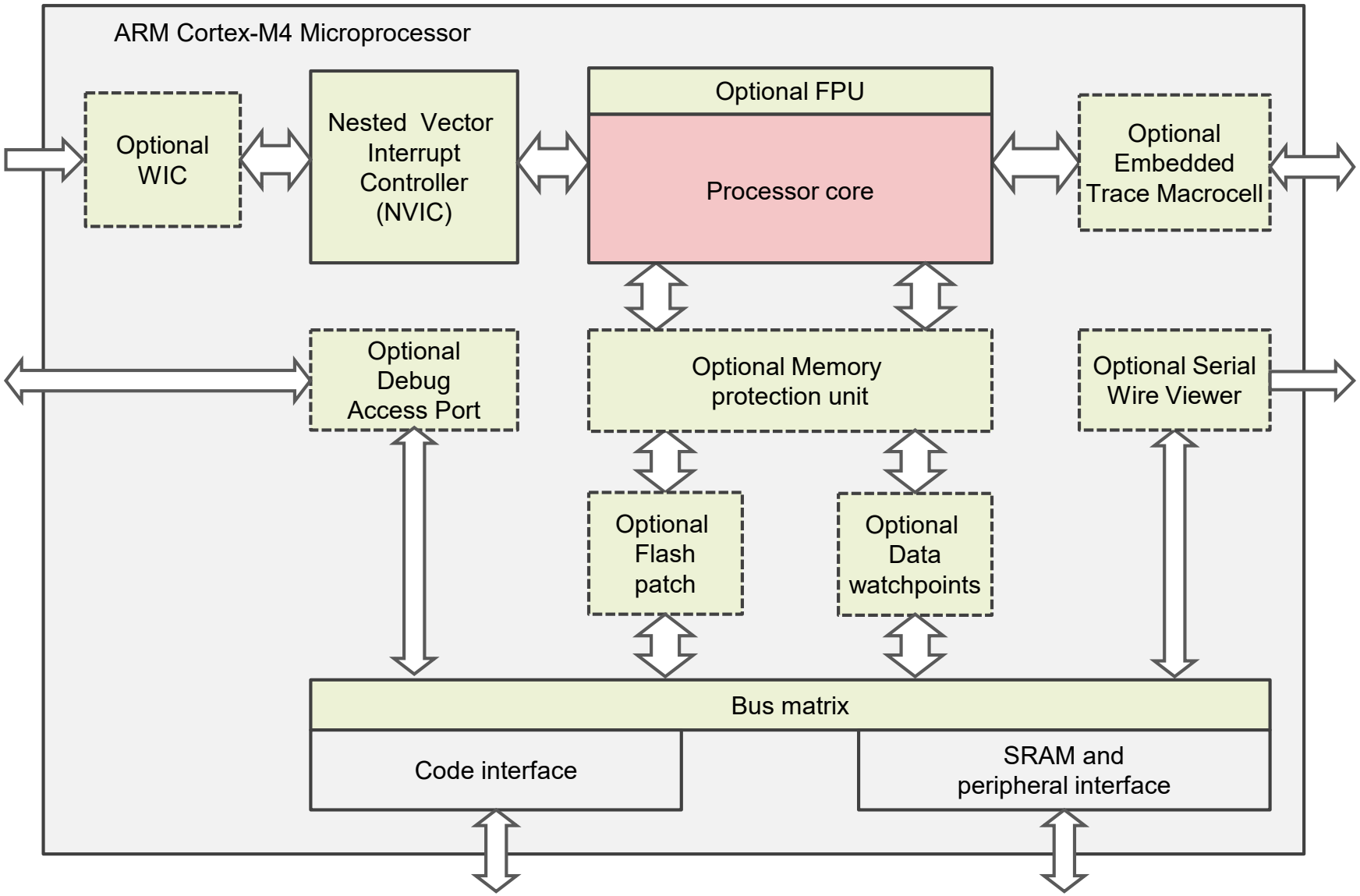


Cortex-M4 Processor Features

- Cortex-M4 processor is designed to meet the challenges of low dynamic power constraints while retaining light footprints
  - 180 nm ultra low power process –157 µW/MHz
  - 90 nm low power process – 33 µW/MHz
  - 40 nm G process – 8 µW/MHz

ARM Cortex-M4 Implementation Data			
Process	180ULL (7-track, typical 1.8v, 25C)	90LP (7-track, typical 1.2v, 25C)	40G 9-track, typical 0.9v, 25C)
Dynamic Power	157 µW/MHz	33 µW/MHz	8 µW/MHz
Floorplanned Area	0.56 mm^2	0.17 mm^2	0.04 mm^2

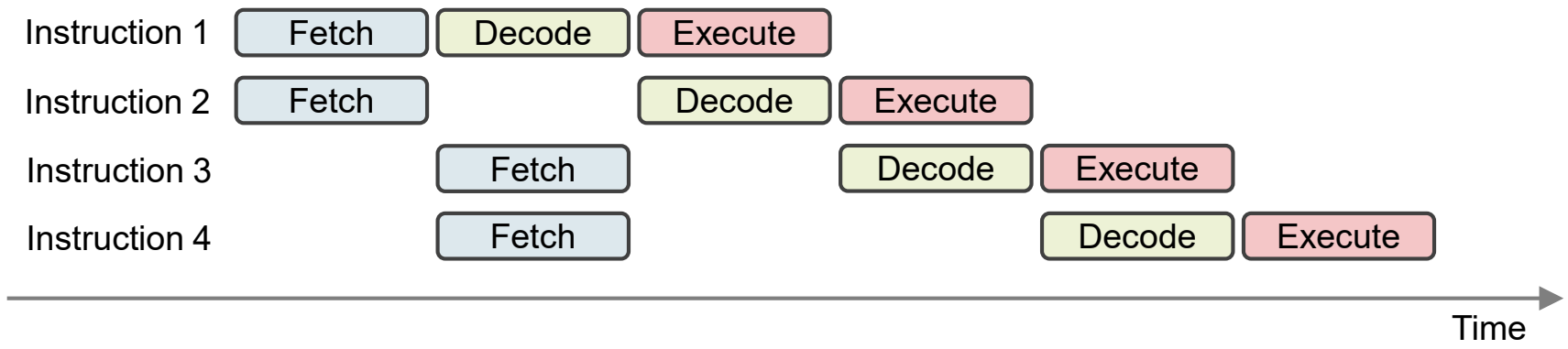
Cortex-M4 Block Diagram



- a 3-stage pipeline Harvard architecture
- optional IEEE754-compliant single-precision floating-point computation
- a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities
- saturating arithmetic and dedicated hardware division

Cortex-M4 Block Diagram

- Processor core
  - Contains internal registers, the ALU, data path, and some control logic
  - Registers include sixteen 32-bit registers for both general and special usage
- Processor pipeline stages
  - Three-stage pipeline: fetch, decode, and execution
  - Some instructions may take multiple cycles to execute, in which case the pipeline will be stalled
  - The pipeline will be flushed if a branch instruction is executed
  - Up to two instructions can be fetched in one transfer (16-bit instructions)

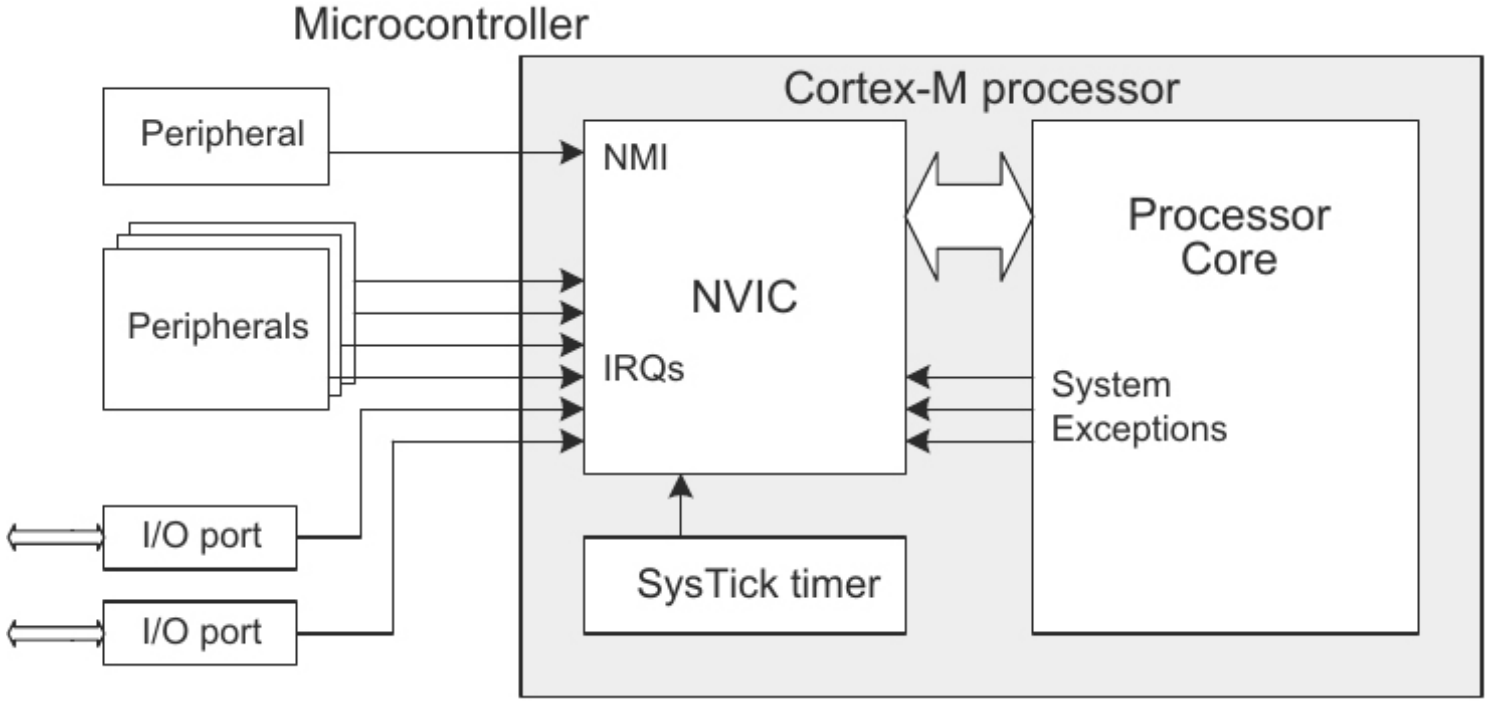


Cortex-M4 Block Diagram

- Nested Vectored Interrupt Controller (NVIC)
  - Up to 240 interrupt request signals and a non-maskable interrupt (NMI)
  - Automatically handles nested interrupts, such as comparing priorities between interrupt requests and the current priority level
- Wakeup Interrupt Controller (WIC)
  - For low-power applications, the microcontroller can enter sleep mode by shutting down most of the components.
  - When an interrupt request is detected, the WIC can inform the power management unit to power up the system.
- Memory Protection Unit (optional)
  - Used to protect memory content, e.g. make some memory regions read-only or preventing user applications from accessing privileged application data

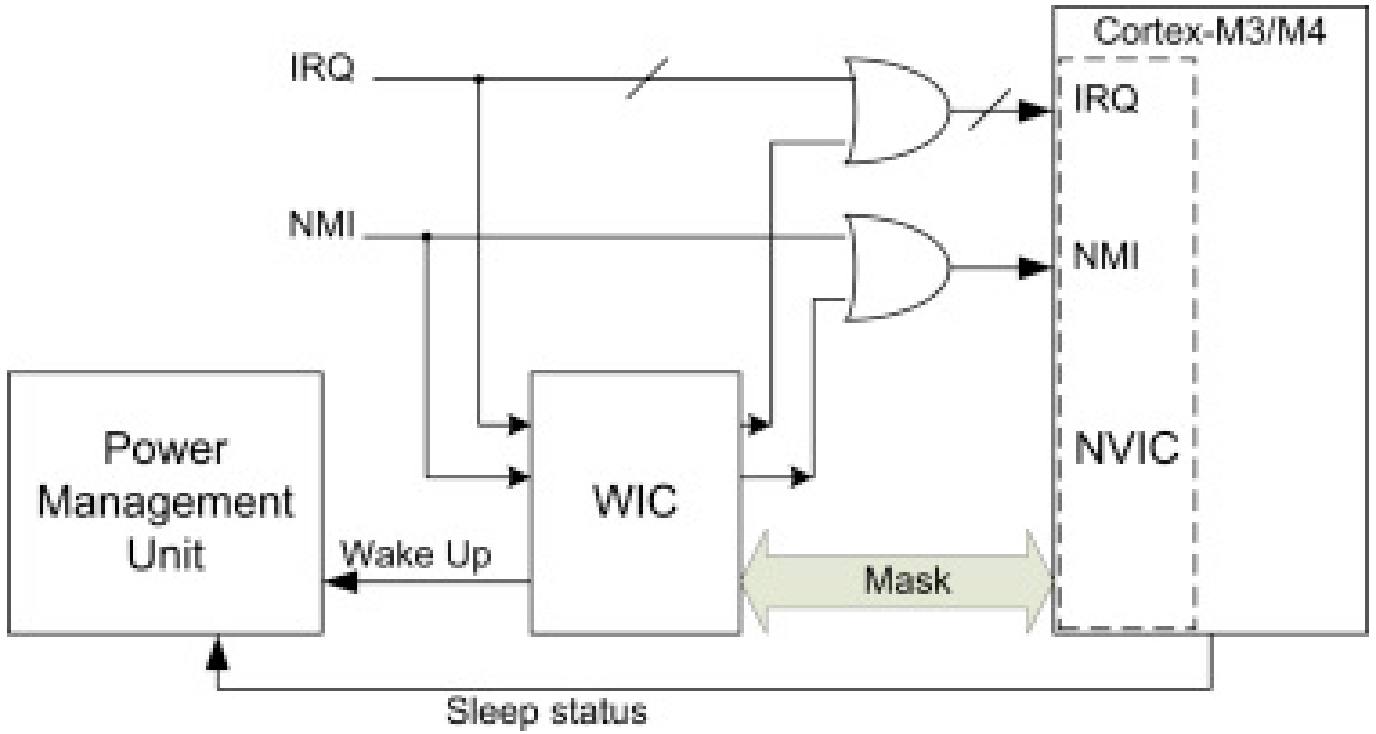
Nested Vectored Interrupt Controller (NVIC)

- Nested vector interrupt control (NVIC) is a method of prioritizing interrupts, improving the MCU's performance as well as reducing interrupt latency.
- "nested" refers to in NVIC, a number of interrupts (up to several hundreds) can be defined, and each interrupt is assigned a priority, with "0" being the highest priority.



Wakeup Interrupt Controller (WIC)

- WIC allows the power consumption of the processor to be further reduced by
  - stopping all the clock signals to the processor;
  - putting the processor into a state retention state.

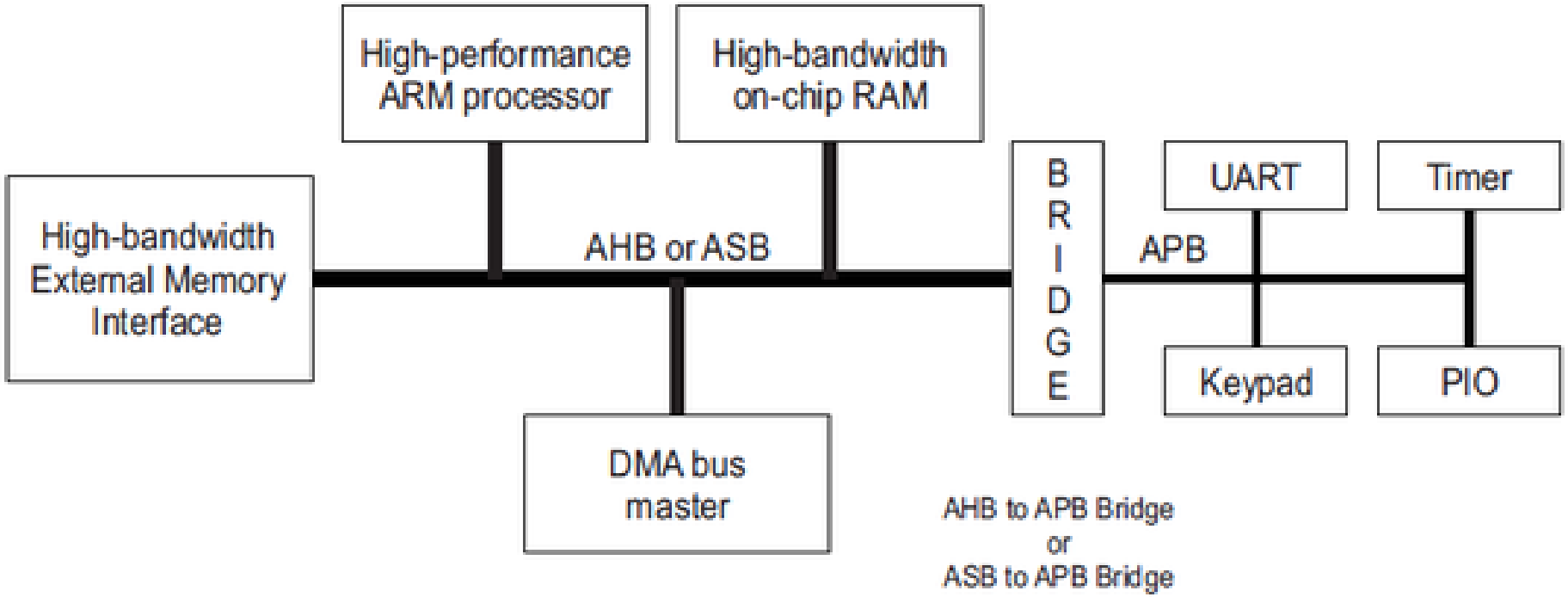


Cortex-M4 Block Diagram

- Bus interconnect
  - Allows data transfer to take place on different buses simultaneously
  - Provides data transfer management, e.g. a write buffer, bit-oriented operations (bit-band)
  - May include bus bridges (e.g. AHB-to-APB bus bridge) to connect different buses into a network using a single global memory space
  - Includes the internal bus system, the data path in the processor core, and the AHB LITE interface unit
- Debug subsystem
  - Handles debug control, program breakpoints, and data watchpoints
  - When a debug event occurs, it can put the processor core in a halted state, where developers can analyse the status of the processor at that point, such as register values and flags

Advanced Micro controller Bus Architecture (AMBA)

- AHB (Advanced High performance) or ASB (Advanced System Bus) protocols for high bandwidth interconnect;
- APB (Advanced Peripheral Bus) protocol for low bandwidth peripheral interconnects.



ARM Cortex-M4 Processor Registers

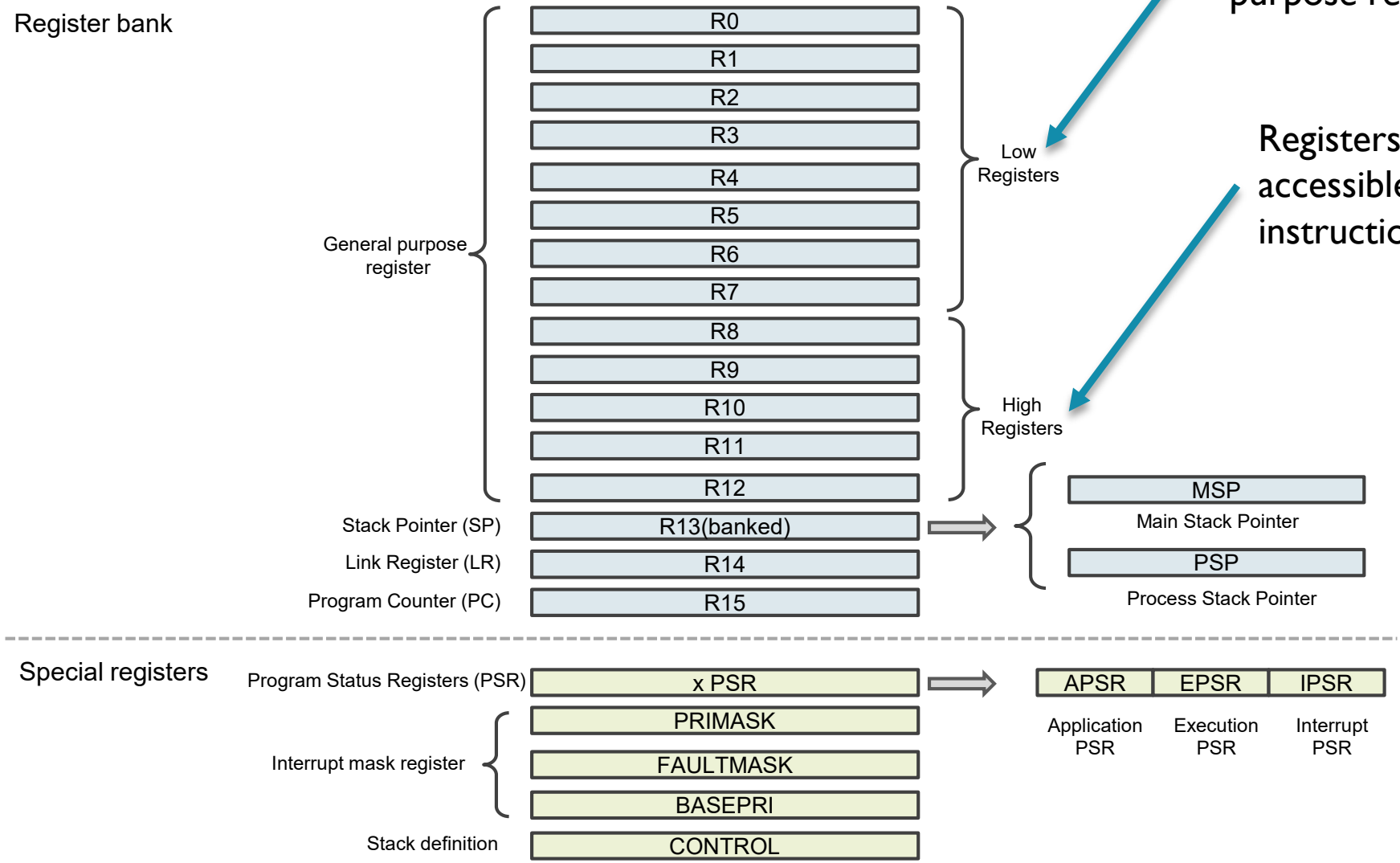
Cortex-M4 Registers

- Processor registers
  - The internal registers are used to store and process temporary data within the processor core
  - All registers are inside the processor core, hence they can be accessed quickly
  - Load-store architecture
    - To process memory data, they have to be first loaded from memory to registers, processed inside the processor core using register data only, and then written back to memory if needed
- Cortex-M4 registers
  - Register bank
    - Sixteen 32-bit registers (thirteen are used for general-purpose);
  - Special registers

Cortex-M4 Registers

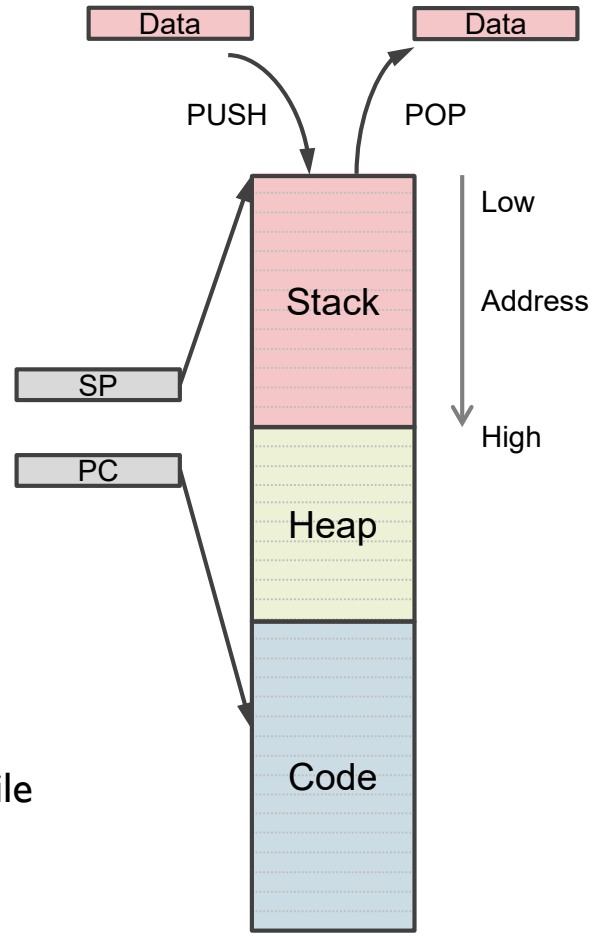
Registers R0-R7 are accessible by all instructions that specify a general-purpose register.

Registers R8-R12 are not accessible by all 16-bit instructions.



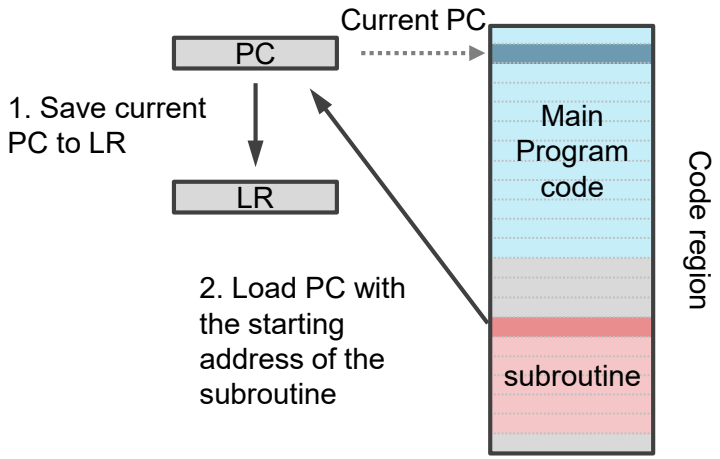
Cortex-M4 Registers

- R0 – R12: general purpose registers
  - Low registers (R0 – R7) can be accessed by any instruction
  - High registers (R8 – R12) sometimes cannot be accessed e.g. by some Thumb (16-bit) instructions
- R13: Stack Pointer (SP)
  - Records the current address of the stack
  - Used for saving the context of a program while switching between tasks
  - Cortex-M4 has two SPs: Main SP, used in applications that require privileged access e.g. OS kernel, and exception handlers, and Process SP, used in base-level application code (when not running an exception handler)
- Program Counter (PC)
  - Records the address of the current instruction code
  - Automatically incremented by 4 at each operation (for 32-bit instruction code), except branching operations
  - A branching operation, such as function calls, will change the PC to a specific address, meanwhile it saves the current PC to the Link Register (LR)

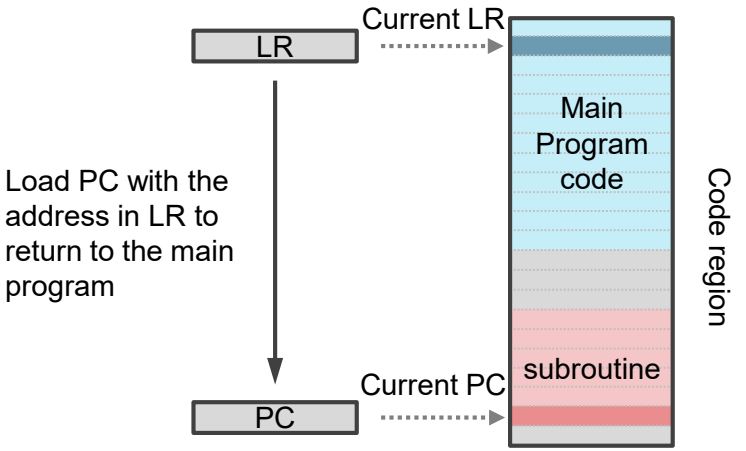


Cortex-M4 Registers

- R14: Link Register (LR)
  - The LR is used to store the return address of a subroutine or a function call
  - The program counter (PC) will load the value from LR after a function is finished



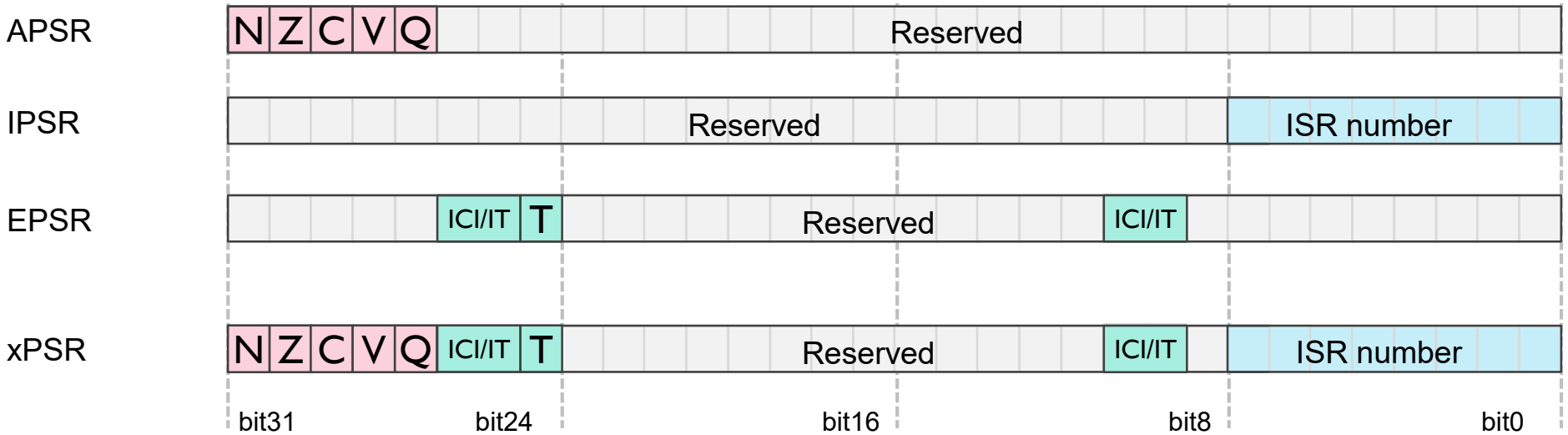
Call a subroutine



Return from a subroutine to the main program

Cortex-M4 Registers

- xPSR, combined Program Status Register
  - Provides information about program execution and ALU flags
  - Application PSR (APSR): You can access the APSR using the MSR and MRS instructions.
  - Interrupt PSR (IPSR): the *Interrupt Service Routine* (ISR) number of the current exception activation.
  - Execution PSR (EPSR): contains the *Thumb state bit* (T-bit).



Cortex-M4 Registers

- APSR
  - N: negative flag – set to one if the result from ALU is negative
  - Z: zero flag – set to one if the result from ALU is zero
  - C: carry flag – set to one if an unsigned overflow occurs
  - V: overflow flag – set to one if a signed overflow occurs
  - Q: sticky saturation flag – set to one if saturation has occurred in saturating arithmetic instructions, or overflow has occurred in certain multiply instructions
- IPSR
  - ISR number – current executing interrupt service routine number
- EPSR
  - T: Thumb state – always one since Cortex-M4 only supports the Thumb state (more on processor states in the next module)
  - IC/IT: Interrupt-Continuable Instruction (ICI) bit, IF-THEN instruction status bit

Cortex-M4 Registers

- Interrupt mask registers
  - 1-bit PRIMASK
    - Set to one will block all the interrupts apart from non-maskable interrupt (NMI) and the hard fault exception
  - 1-bit FAULTMASK
    - Set to one will block all the interrupts apart from NMI
  - 1-bit BASEPRI
    - Set to one will block all interrupts of the same or lower level (only allow for interrupts with higher priorities)
- CONTROL: special register
  - 1-bit stack definition
    - Set to one: use the process stack pointer (PSP)
    - Clear to zero: use the main stack pointer (MSP)

Special-Purpose Priority Mask Register

- https://developer.arm.com/documentation/ddi0413/c/programmer-s-model/registers/special-purpose-priority-mask-register?lang=en

Use the Special-Purpose Priority Mask Register for priority boosting.

Figure 2.5 shows the bit assignments of the Special-Purpose Priority Mask Register.

Figure 2.5. Special-purpose Priority Mask Register bit assignments

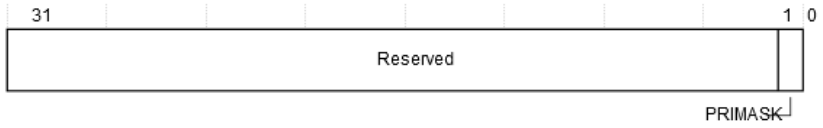
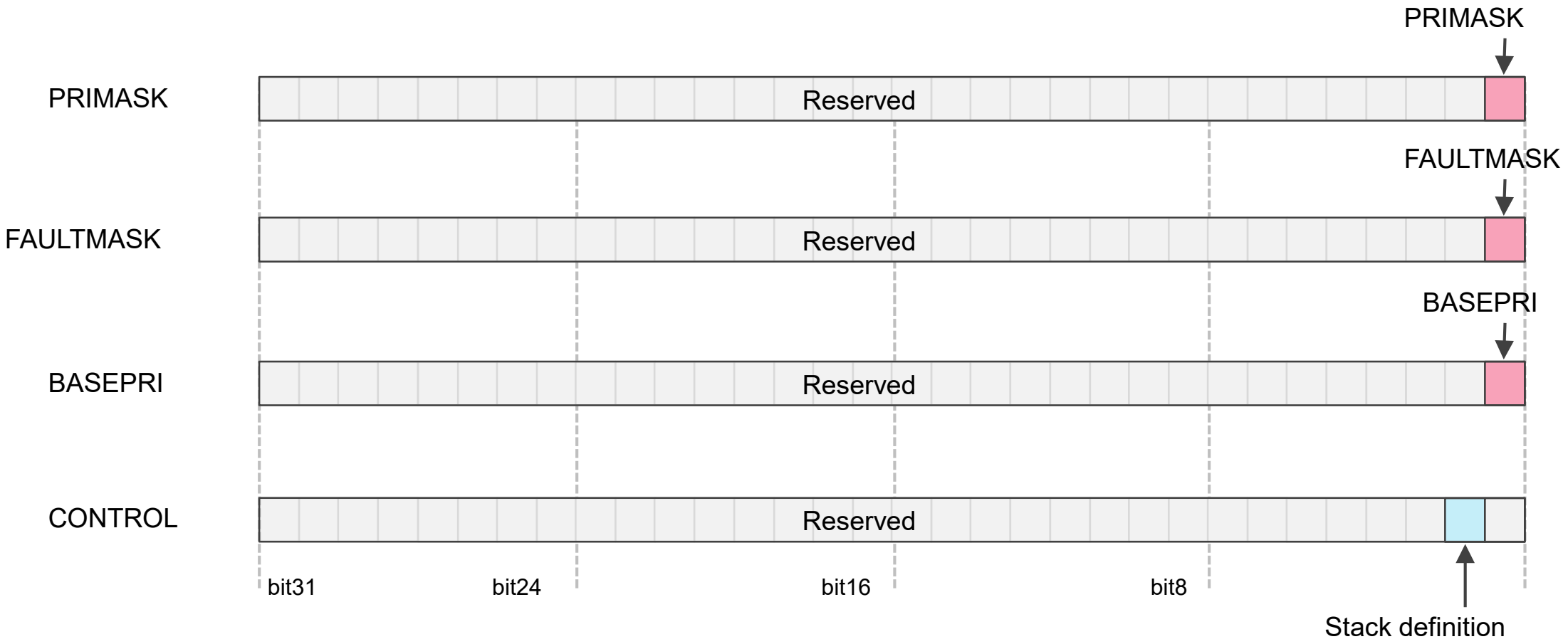


Table 2.4 lists the bit assignments of the Special-Purpose Priority Mask Register.

Field	Name	Function
[31:1]	-	Reserved
[0]	PRIMASK	When set, raises execution priority to 0

You can access the Special-Purpose Priority Mask Register using the `MSR` and `MRS` instructions. You can also use the `CPS` instruction to set or clear PRIMASK.

Cortex-M4 Registers



Useful Resources

- Architecture Reference Manual:

  https://developer.arm.com/documentation/ddi0403/c/
- Cortex-M4 Technical Reference Manual:
  - https://developer.arm.com/documentation/100166/0001/?search=5eec6e71e24a5e02d07b259a
- Cortex-M4 Devices Generic User Guide:

  https://developer.arm.com/documentation/dui0553/a/

Summary

- ARM Architectures and Processors
  - ARM Processor Families
  - ARM Processor vs. ARM Architectures
  - ARM-based System-on-Chip (SoC)

- ARM Cortex-M4 Processor
  - Cortex-M4 Processor Overview
  - Cortex-M4 Registers
  - Cortex-M4 Special Registers