ARM Cortex®- M0 Processor Core

An Overview

ARM Processors Portfolio

Processor Class	ARM Processor	Architecture	Performance DMIPS/MHz	ARM instructions	Thumb-2 instructions	Jazelle-DBX JAVA bytecode execution	Jazelle-RCT Dynamic compiler support	TrustZone security	E' DSP extensions	Media SIMD extensions	NEON SIMD extensions	Floating point	Caches	Memory Management Unit (MMU)	Memory Protection Unit (MPU)	Hardware Cache coherency	Target OS	Trace support
ARM7 ARM9 ARM11	**ARM7TDMI/ARM7TDMI-S**	ARMv4-T	0.95	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	Real Time	✓
	ARM946E-S	ARMv5-E	1.23	✓	✗	✗	✗	✗	✓	✗	✗	Optional	✓	✗	✓	✗	Real Time	✓
	ARM926EJ-S	ARMv5-EJ	1.06	✓	✗	✓	✗	✗	✓	✗	✗	Optional	✓	✓	✗	✗	Platform	✓
	ARM1136J-S	ARMv6	1.18	✓	✗	✓	✗	✗	✓	✓	✗	Optional	✓	✓	✗	✗	Platform	✓
	ARM1156T2-S	ARMv6-T2	1.45	✓	✓	✗	✗	✗	✓	✓	✗	Optional	✓	✗	✓	✗	Real Time	✓
	ARM1176JZ-S	ARMv6-Z	1.26	✓	✗	✓	✗	✓	✓	✓	✗	Optional	✓	✓	✗	✗	Platform	✓
	ARM11 MPCore	ARMv6	1.25	✓	✗	✓	✗	✗	✓	✓	✗	Optional	✓	✓	✗	✓	Platform/SMP	✓
M	**Cortex-M0+**	ARMv6-M	0.90	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	Real Time	✗
	Cortex-M0	ARMv6-M	0.90	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	Real Time	✗
	Cortex-M1	ARMv6-M	0.79	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	Real Time	✗
	Cortex-M3	ARMv7-M	1.25	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	Optional	✗	Real Time	Instruction only
	Cortex-M4	ARMv7-ME	1.25	✓	✓	✗	✗	✗	✓	✓	✗	Optional	✓	✗	Optional	✗	Real Time	✓
R	**Cortex-A5 MPCore**	ARMv7+MP	1.58	✓	✓	✓	✓	✓	✓	✓	Optional	Optional	✓	✓	✗	✓+ACP	Platform/SMP	✓
	Cortex-R4	ARMv7	1.66	✓	✓	✗	✗	✗	✓	✓	✗	Optional	✓	✗	Optional	✗	Real Time	✓
	Cortex-R5	ARMv7	1.66	✓	✓	✗	✗	✗	✓	✓	✗	Optional	✓	✗	Optional	✗	Real Time	✓
	Cortex-R7	ARMv7	2.53	✓	✓	✗	✗	✗	✓	✓	✗	Optional	✓	✗	Optional	✗	Real Time	✓
A	**Cortex-A7**	ARMv7+MP	1.90	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓+ACP	Platform/SMP	PTM
	Cortex-A8	ARMv7	2.07	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	Platform	✓
	Cortex-A9 MPCore	ARMv7+MP	2.50	✓	✓	✓	✓	✓	✓	✓	Optional	Optional	✓	✓	✗	✓+ACP	Platform/SMP	PTM
	Cortex-A15 MPCore	ARMv7+MP	2.50	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓+ACP	Platform/SMP	PTM
	Cortex-A53	ARMv8	2.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓+ACP	Platform/SMP	PTM
	Cortex-A57	ARMv8	>4.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓+ACP	Platform/SMP	PTM

Equipment Adopting ARM Cores

M

R

A

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	Optional	No	No	No	No
Cortex-M0+	ARMv6-M	Von Neumann	Most	Subset	Optional	No	No	No	No
Cortex-M1	ARMv6-M	Von Neumann	Most	Subset	Optional	No	No	No	No
Cortex-M3	ARMv7-M	Harvard	Entire	Entire	Optional	Yes	Yes	No	No
Cortex-M4	ARMv7E-M	Harvard	Entire	Entire	Optional	Yes	Yes	Yes	Optional

ARM Cortex-M0 Processor

Cortex-M0 Overview

- ARMv6-M Architecture
  - Von-Neumann Architecture
  - 32-bit Architecture
  - Thumb technology
- Nested Vector Interrupt Controller (NVIC)
- AHB-Lite Master Interface
- Optional Ultra-low power Support
- Optional CoreSight-compliant Debug
- RTL is configurable
- Synthesizable
  - Gate count 12 ~ 25K

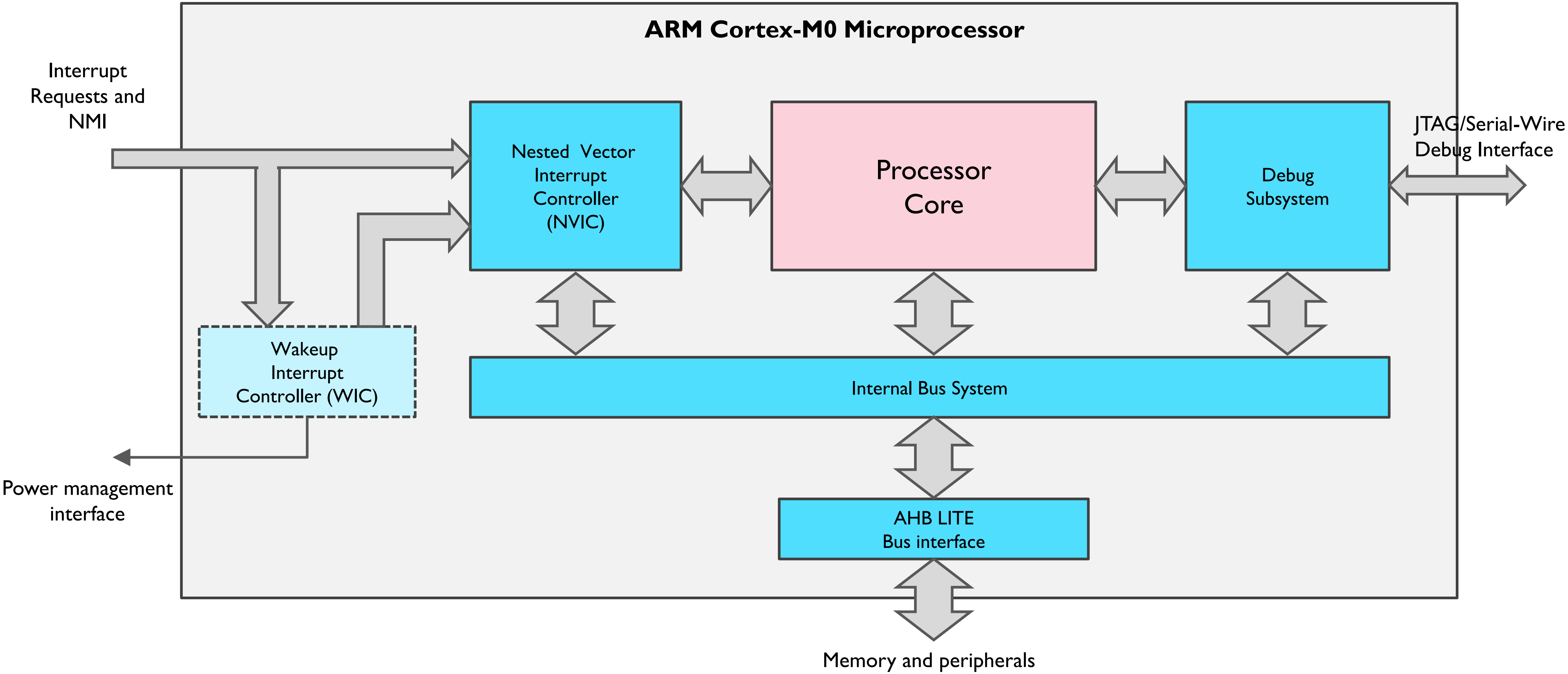
Cortex-M0 Processor Programmers Model

- The Cortex-M0 is designed to be programmed fully in C
  - No need to write assembly code
- Thumb technology
  - 16-bit and 32-bit instructions

- Set of processor core and memory-mapped registers are provided
- Architecturally defined memory map
  - Forwards compatible with other M-profile processors
  - Binary compatible

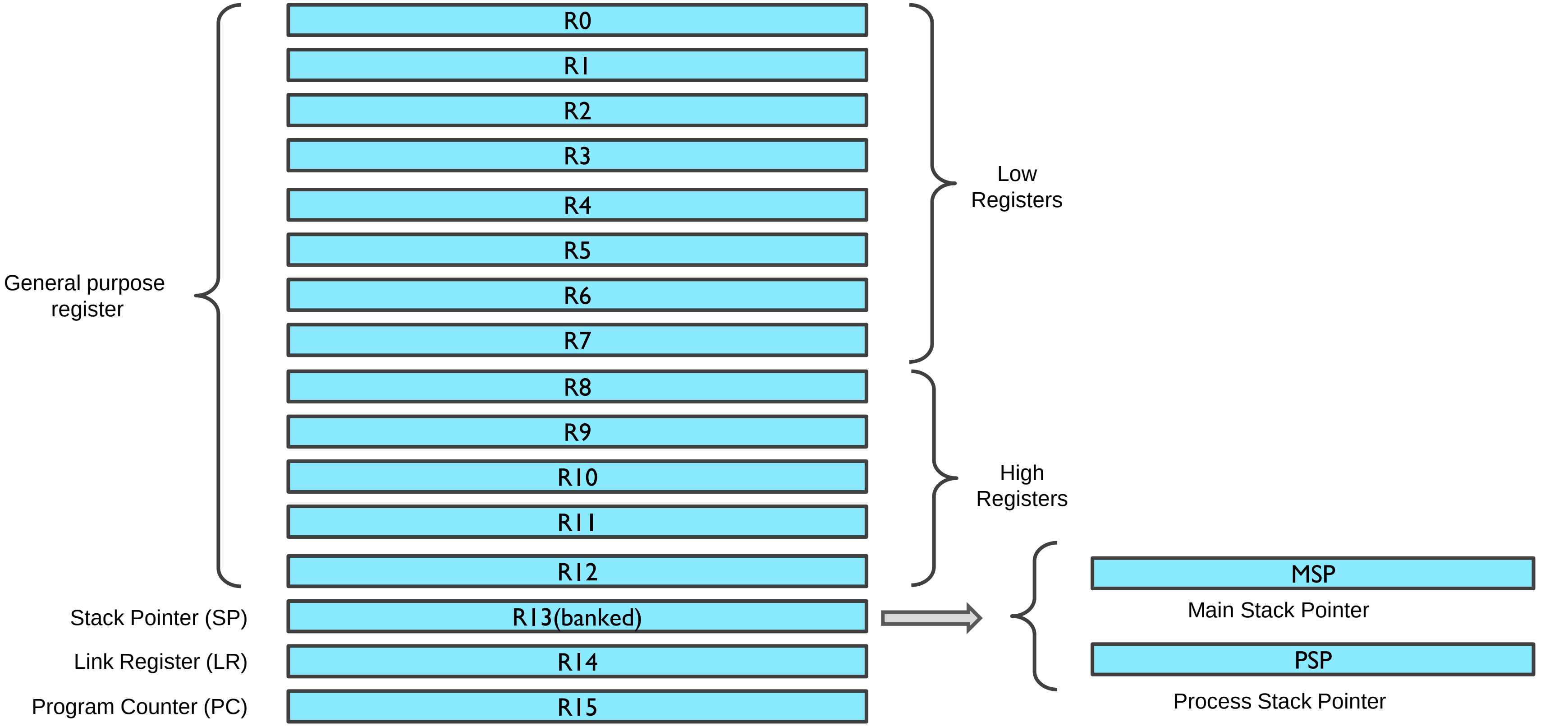
- In-order execution of instructions
- Multi-cycle instructions are abandoned and restarted on interrupt
  - Multiple memory accesses (LDM/STM) are interrupted between transfers
- Deterministic instruction execution

Cortex-M0 Block Diagram


ARM Cortex-M0 Microprocessor
Interrupt Requests and NMI
Nested Vector Interrupt Controller (NVIC)
Processor Core
Debug Subsystem
JTAG/Serial-Wire Debug Interface
Wakeup Interrupt Controller (WIC)
Internal Bus System
Power management interface
AHB LITE Bus interface
Memory and peripherals

Cortex-M0 Registers

Register bank



General purpose register

R0

R1

R2

R3

R4

R5

R6

R7

R8

R9

R10

R11

R12

Low Registers

High Registers

Stack Pointer (SP) R13(banked)

Link Register (LR) R14

Program Counter (PC) R15

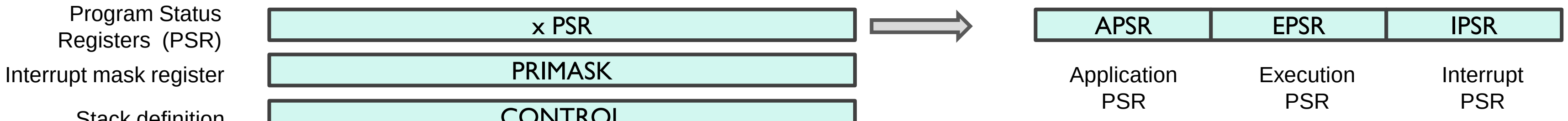
MSP

Main Stack Pointer

PSP

Process Stack Pointer

Special registers



Program Status Registers (PSR) x PSR

Interrupt mask register PRIMASK

Stack definition CONTROL

APSR | EPSR | IPSR

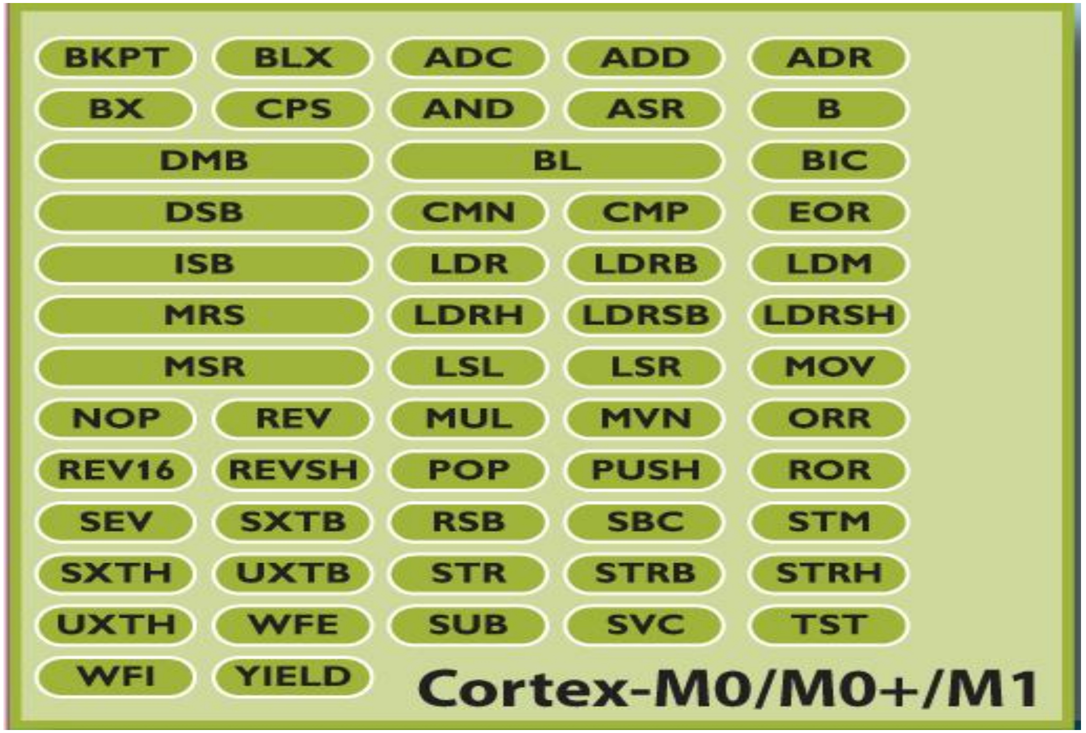
Application PSR

Execution PSR

Interrupt PSR

Reduced Instruction Set

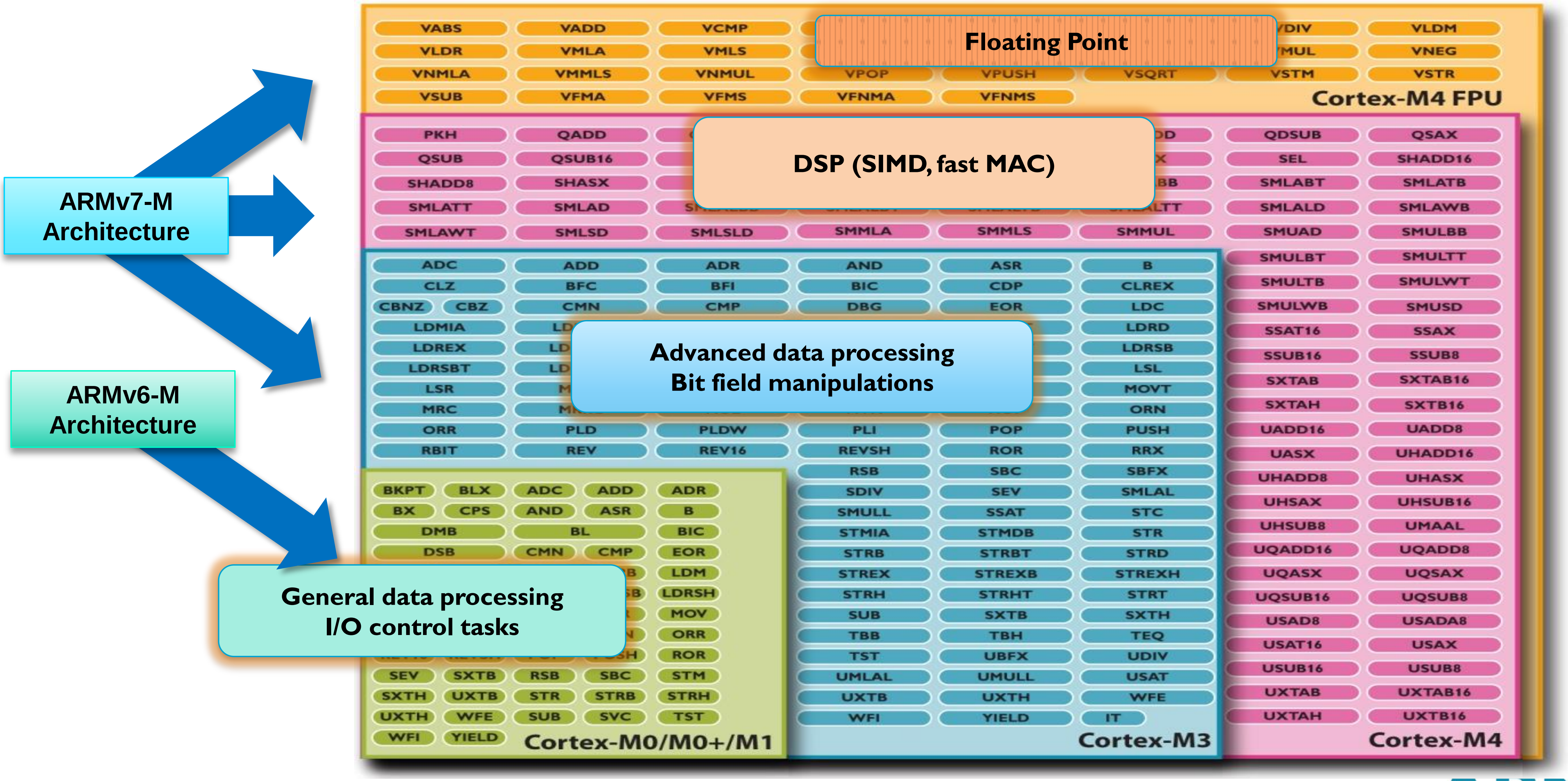
- Only 56 Instructions
- Mostly coded on 16-bit
- Operate on the 32-bit registers
- Option for fast MUL 32x32 bit in 1 cycle
- Instruction types
  - Branch instructions
  - Data-processing instructions
  - Load and store instructions
  - Status register access instructions
  - Miscellaneous instructions



BKPT	BLX	ADC	ADD	ADR
BX	CPS	AND	ASR	B
DMB		BL		BIC
DSB		CMN	CMP	EOR
ISB		LDR	LDRB	LDM
MRS		LDRH	LDRSB	LDRSH
MSR		LSL	LSR	MOV
NOP	REV	MUL	MVN	ORR
REV16	REVSH	POP	PUSH	ROR
SEV	SXTB	RSB	SBC	STM
SXTH	UXTB	STR	STRB	STRH
UXTH	WFE	SUB	SVC	TST
WFI	YIELD			

Cortex-M0/M0+/M1

Cortex-M: binary upwards compatibility



ARMv7-M Architecture

ARMv6-M Architecture

Floating Point

DSP (SIMD, fast MAC)

Advanced data processing
Bit field manipulations

General data processing
I/O control tasks

VABS VADD VCMP DIV VLDM
VLDR VMLA VMLS MUL VNEG
VNMLA VMMLS VNMUL VPOP VPUSH VSQRT VSTM VSTR
VSUB VFMA VFMS VFNMA VFNMS

Cortex-M4 FPU

PKH QADD DD QDSUB QSAX
QSUB QSUB16 X SEL SHADD16
SHADD8 SHASX BB SMLABT SMLATB
SMLATT SMLAD LTT SMLALD SMLAWB
SMLAWT SMLSD SMLSLD SMMLA SMMLS SMMUL SMUAD SMULBB
SMULBT SMULTT
SMULTB SMULWT
SMULWB SMUSD
SSAT16 SSAX
SSUB16 SSUB8
SXTAB SXTAB16
SXTAH SXTB16
UADD16 UADD8
UASX UHADD16
UHADD8 UHASX
UHSAX UHSUB16
UHSUB8 UMAAL
UQADD16 UQADD8
UQASX UQSAX
UQSUB16 UQSUB8
USAD8 USADA8
USAT16 USAX
USUB16 USUB8
UXTAB UXTAB16
UXTAH UXTB16

Cortex-M4

ADC ADD ADR AND ASR B
CLZ BFC BFI BIC CDP CLREX
CBNZ CBZ CMN CMP DBG EOR LDC
LDMIA LD LDRD
LDREX LD LDRSB
LDRSBT LD LSL
LSR M MOVT
MRC M ORN
ORR PLD PLDW PLI POP PUSH
RBIT REV REV16 REVSH ROR RRX
RSB SBC SBFX
SDIV SEV SMLAL
SMULL SSAT STC
STMIA STMDB STR
STRB STRBT STRD
STREX STREXB STREXH
STRH STRHT STRT
SUB SXTB SXTH
TBB TBH TEQ
TST UBFX UDIV
UMLAL UMULL USAT
UXTB UXTH WFE
WFI YIELD IT

Cortex-M3

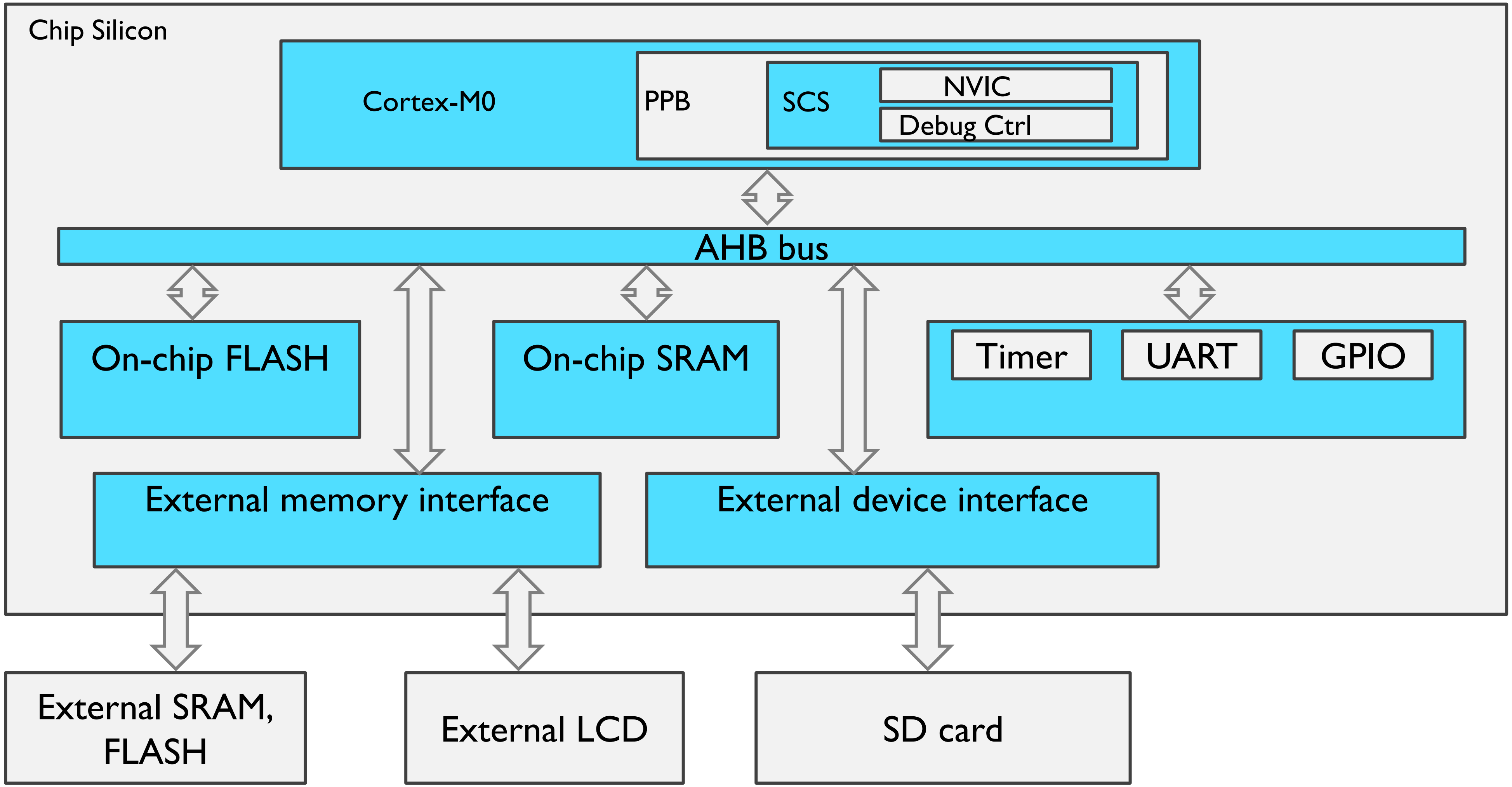
BKPT BLX ADC ADD ADR
BX CPS AND ASR B
DMB BL BIC
DSB CMN CMP EOR
B LDM
SB LDRSH
MOV
ORR
H ROR
SEV SXTB RSB SBC STM
SXTH UXTB STR STRB STRH
UXTH WFE SUB SVC TST
WFI YIELD

Cortex-M0/M0+/M1

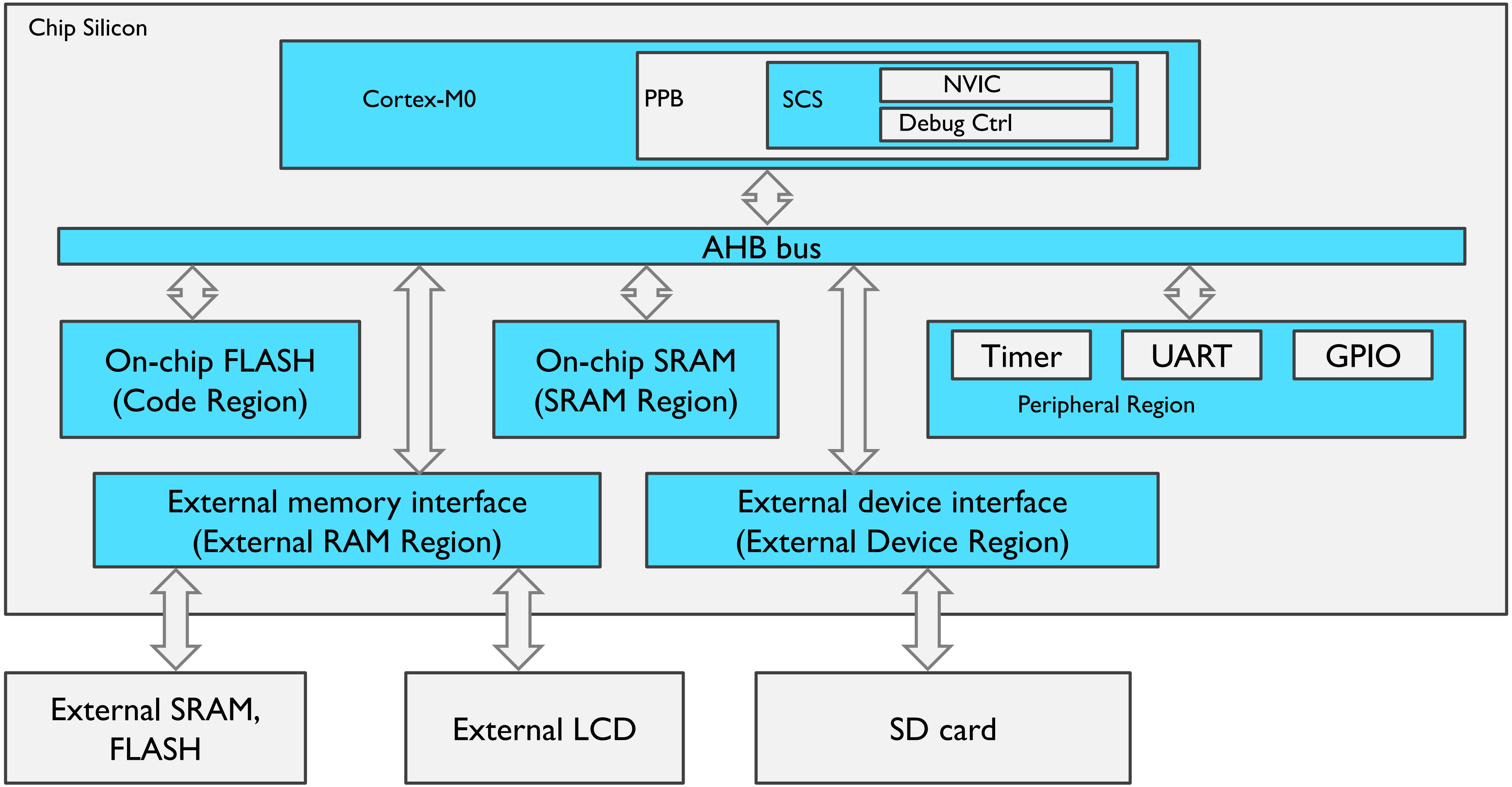
ARM Cortex-M0 Processor

Memory Architecture

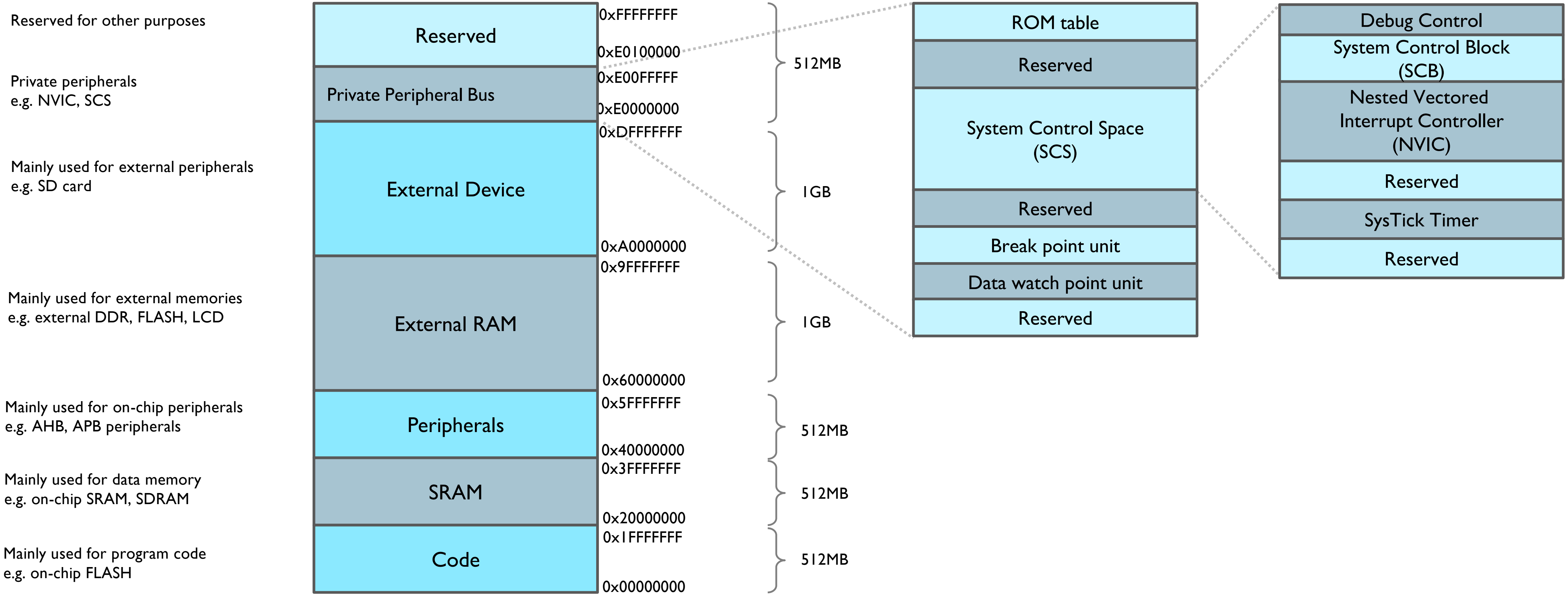
Cortex-M0 Memory Map (Example)



Cortex-M0 Memory Map (Example)



Cortex-M0 Memory Map

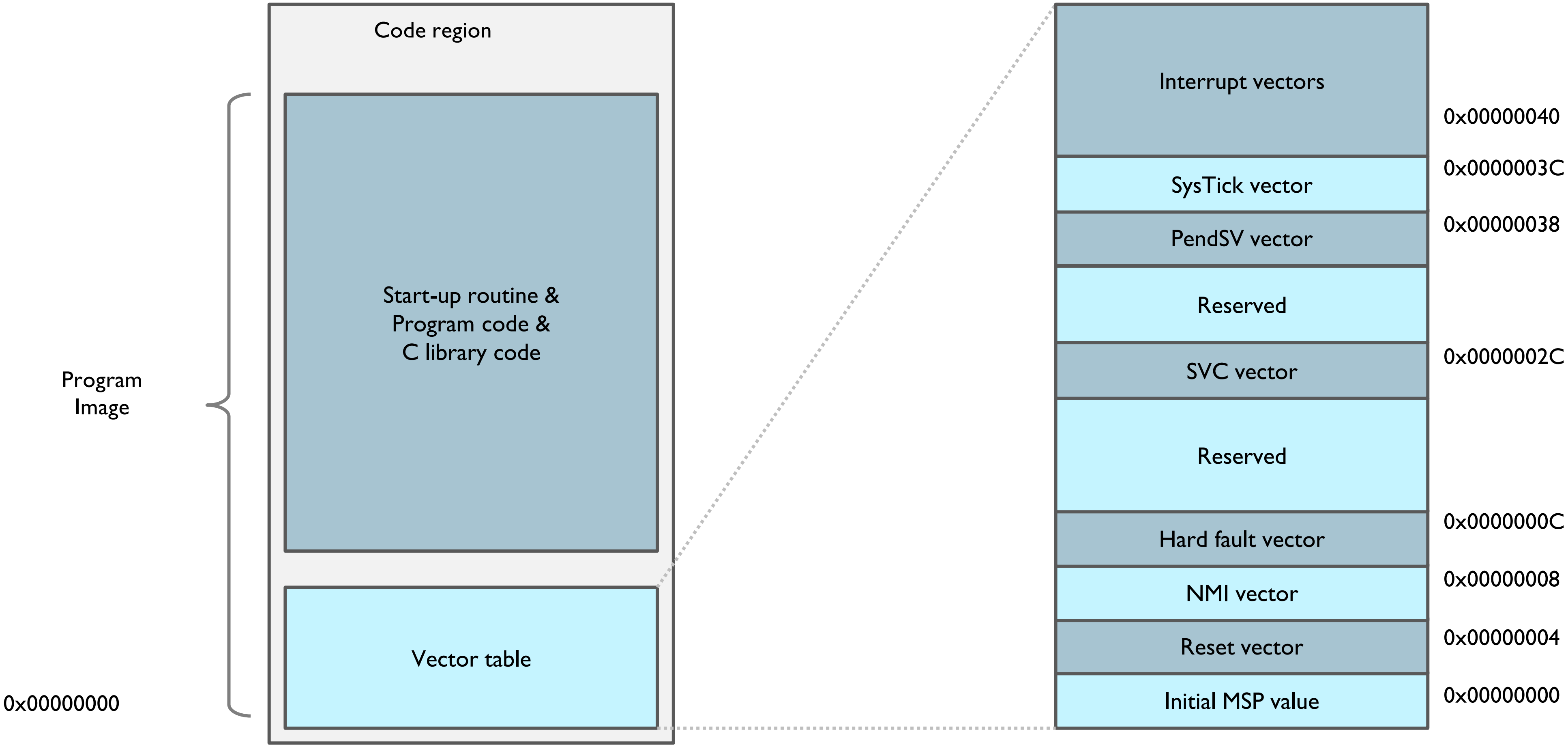


Reserved for other purposes	Reserved	0xFFFFFFFF	0xE0100000	512MB
Private peripherals e.g. NVIC, SCS	Private Peripheral Bus	0xE00FFFFF	0xE0000000	
Mainly used for external peripherals e.g. SD card	External Device	0xDFFFFFFF	0xA0000000	1GB
Mainly used for external memories e.g. external DDR, FLASH, LCD	External RAM	0x9FFFFFFF	0x60000000	1GB
Mainly used for on-chip peripherals e.g. AHB, APB peripherals	Peripherals	0x5FFFFFFF	0x40000000	512MB
Mainly used for data memory e.g. on-chip SRAM, SDRAM	SRAM	0x3FFFFFFF	0x20000000	512MB
Mainly used for program code e.g. on-chip FLASH	Code	0x1FFFFFFF	0x00000000	512MB

- ROM table
- Reserved
- System Control Space (SCS)
- Reserved
- Break point unit
- Data watch point unit
- Reserved

- Debug Control
- System Control Block (SCB)
- Nested Vectored Interrupt Controller (NVIC)
- Reserved
- SysTick Timer
- Reserved

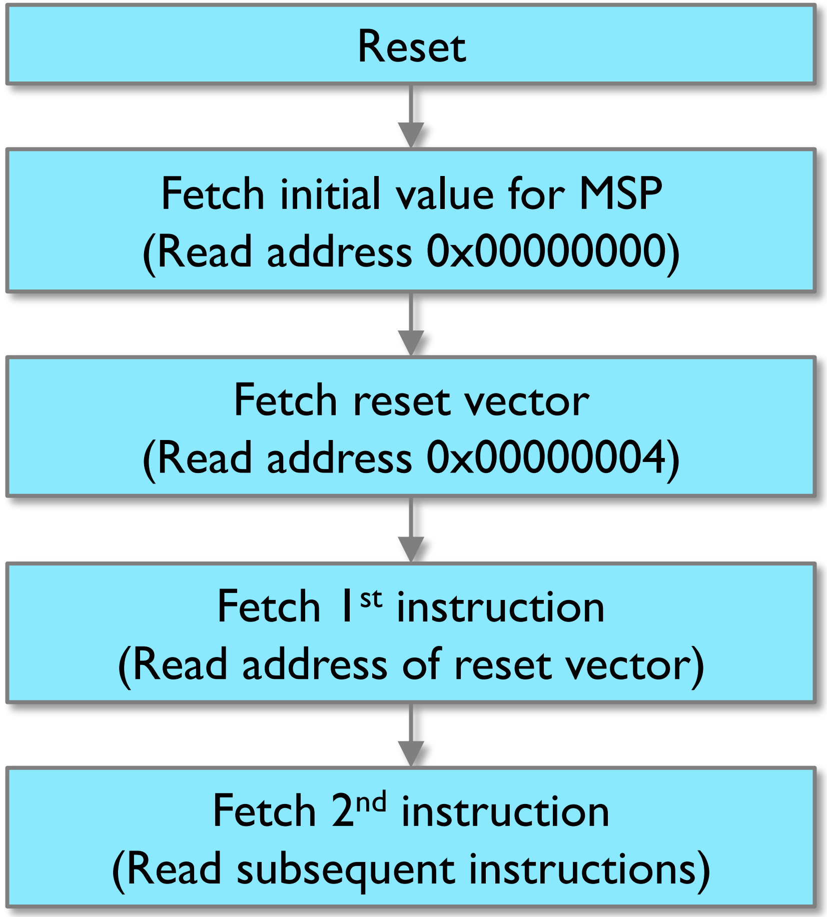
Cortex-M0 Program Image



Cortex – M0 Reset Behaviour

- After reset:

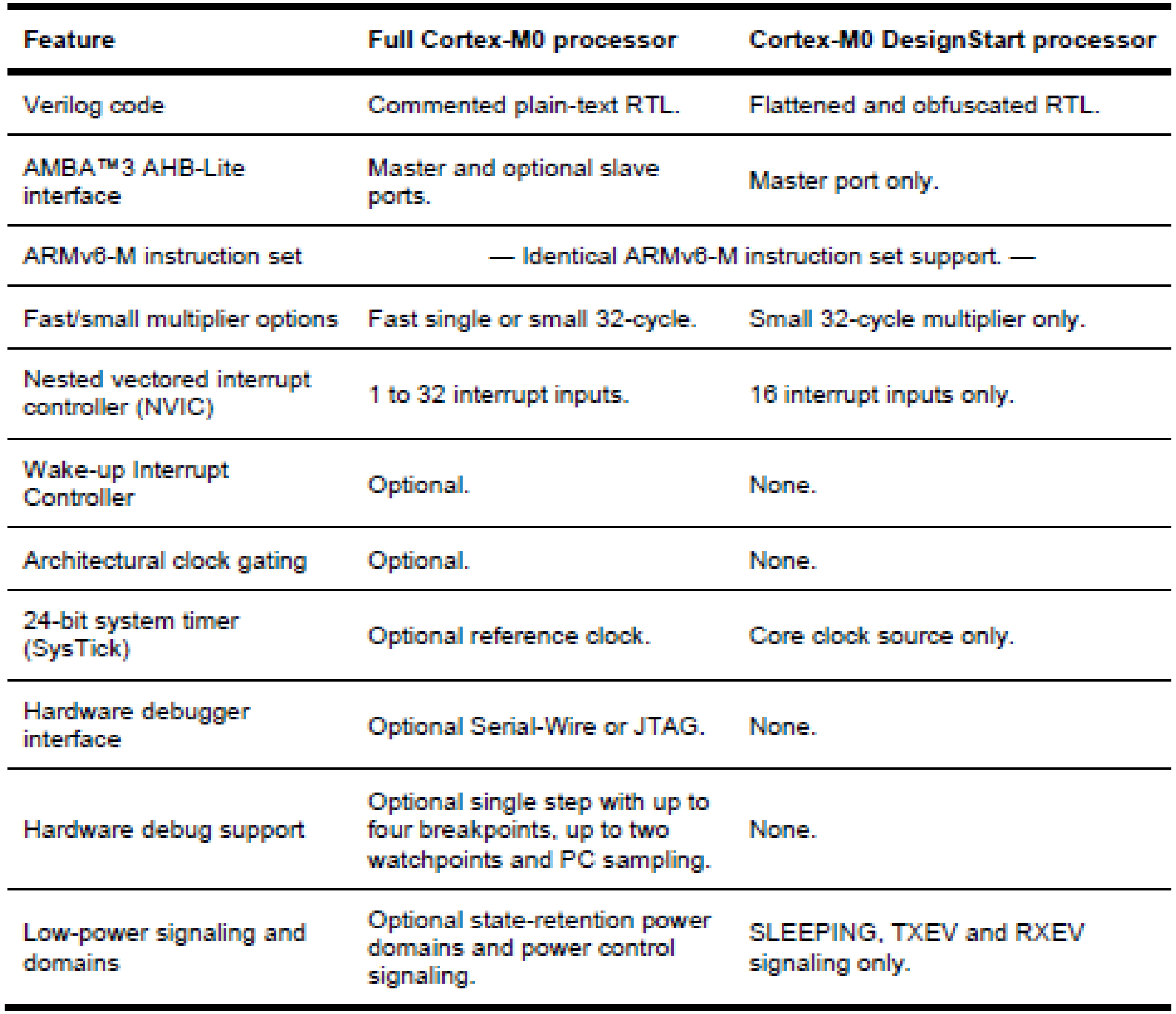
  1. First reads the initial MSP value;
  2. Then reads the reset vector;
  3. Branches to the starting of the programme execution address (reset handler);
  4. Subsequently executes program instructions.


Reset
Fetch initial value for MSP
(Read address 0x00000000)
Fetch reset vector
(Read address 0x00000004)
Fetch 1st instruction
(Read address of reset vector)
Fetch 2nd instruction
(Read subsequent instructions)

Cortex M0 Design Start

- Fixed configuration of the Cortex-M0 processor
- Low cost access to Cortex-M0 processor technology by offering a subset of the full product.
- Pre-configured and obfuscated, but synthesizable, Verilog version of the full Cortex-M0 processor
- Fully working version of the Cortex-M0 processor
- Used as the basis for production hardware and software designs.
- Gate count is ~16K
- Cortex M0 Processor gate count is 12k to 26K depending on the configuration

Cortex M0 Design Start vs Full Cortex M0



Feature	Full Cortex-M0 processor	Cortex-M0 DesignStart processor
Verilog code	Commented plain-text RTL.	Flattened and obfuscated RTL.
AMBA™3 AHB-Lite interface	Master and optional slave ports.	Master port only.
ARMv6-M instruction set	— Identical ARMv6-M instruction set support. —	
Fast/small multiplier options	Fast single or small 32-cycle.	Small 32-cycle multiplier only.
Nested vectored interrupt controller (NVIC)	1 to 32 interrupt inputs.	16 interrupt inputs only.
Wake-up Interrupt Controller	Optional.	None.
Architectural clock gating	Optional.	None.
24-bit system timer (SysTick)	Optional reference clock.	Core clock source only.
Hardware debugger interface	Optional Serial-Wire or JTAG.	None.
Hardware debug support	Optional single step with up to four breakpoints, up to two watchpoints and PC sampling.	None.
Low-power signaling and domains	Optional state-retention power domains and power control signaling.	SLEEPING, TXEV and RXEV signaling only.

Lab

- Consider you have a simple Cortex M0 based Micro-controller with the following memory map
  - Internal SRAM – 1 KB – Starting Address 0x0000 0000
  - A LED Peripheral – One word – Address 0x5000 0000

- We will write assembly program to toggle the LEDs and simulate the program

- We will analyse the resulting binary file and the disassembly file

- In the Next Section we will see how to design AHB-Lite Compliant LED Peripheral