Cortex-M Architecture

Ian Johnson
Product Manager
ARM

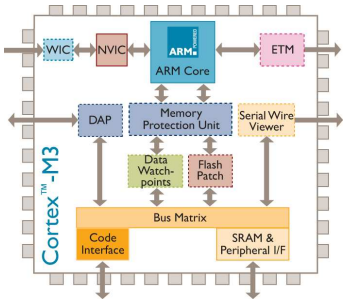
What we develop

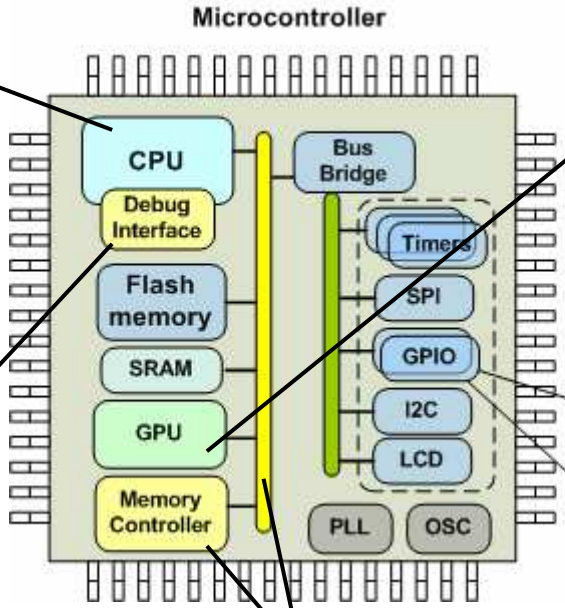
- The processor is only part of the story

Cortex
Low-Power Leadership from ARM

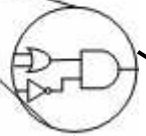
AMBA
System Architecture

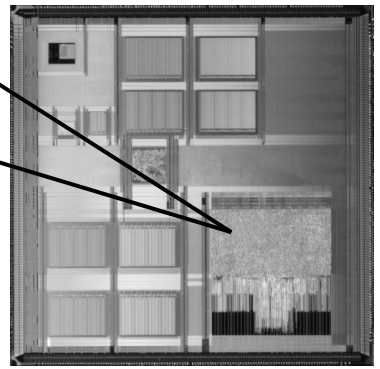
mali





Artisan
Advanced Physical IP by ARM





CoreSight
Debug & Trace IP by ARM

CoreLink
Intelligent System IP by ARM

The Cortex Family

Cortex-A

servers


set top boxes


netbooks


mobile applications

Cortex-R

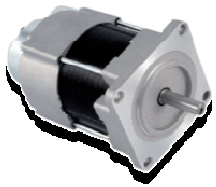

disk drives

digital cameras


mobile baseband

Cortex-M


appliances


motors


audio

Cortex-M – Proven Success in Market

- 90+ licensees of ARM Cortex-M processors
- Over 700 Cortex-M processor-based devices
- >500M shipments to date

Tele-parking

Intelligent toys

Utility Meters

IR Fire Detector

Exercise Machines

Energy Efficient Appliances

Intelligent Vending

ARM Cortex-M family

- **A compatible architecture spanning the embedded application range**
  - Designed for deterministic, low power and low area applications.

ARM Cortex-M4

"32-bit/DSC" applications

Efficient digital signal control

ARM Cortex-M3

"16/32-bit" applications

Performance efficiency

ARM Cortex-M1

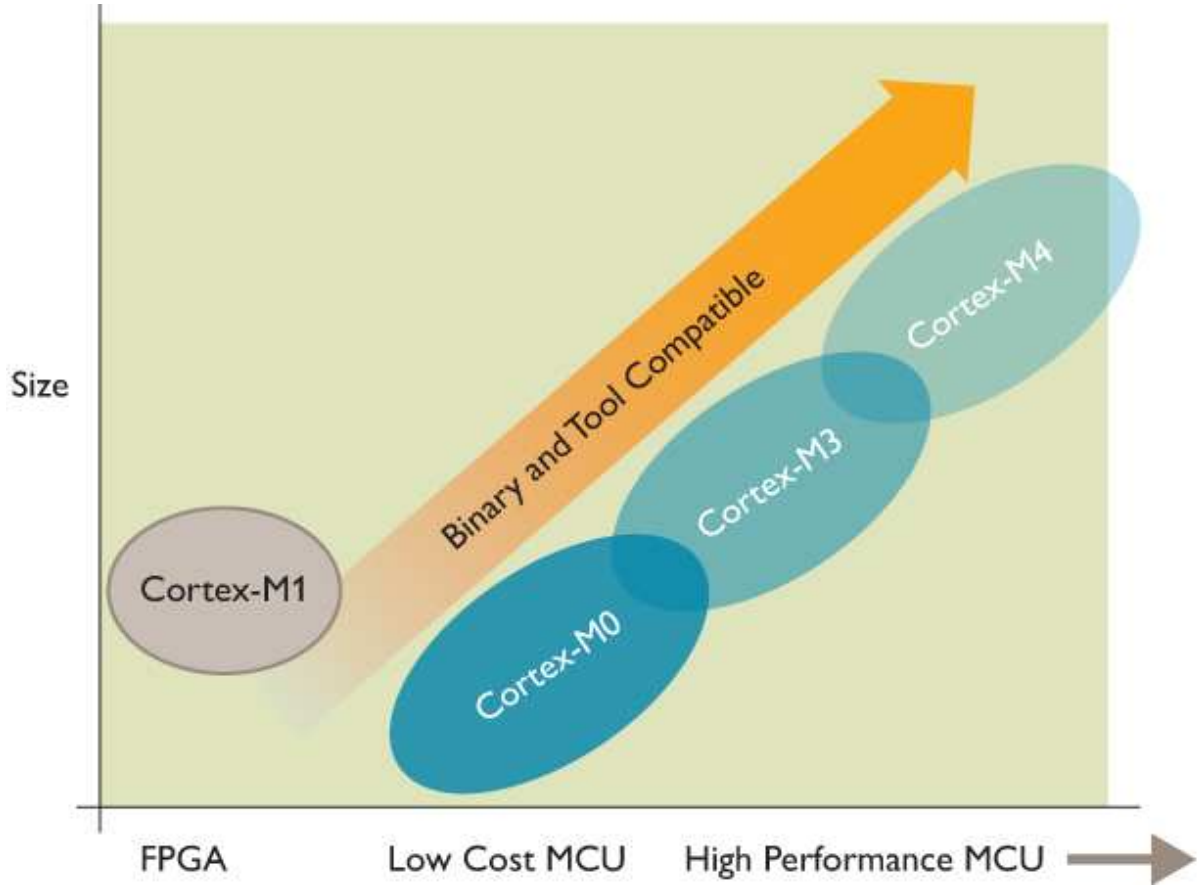
FPGA applications

Optimized for multiple FPGA types

ARM Cortex-M0

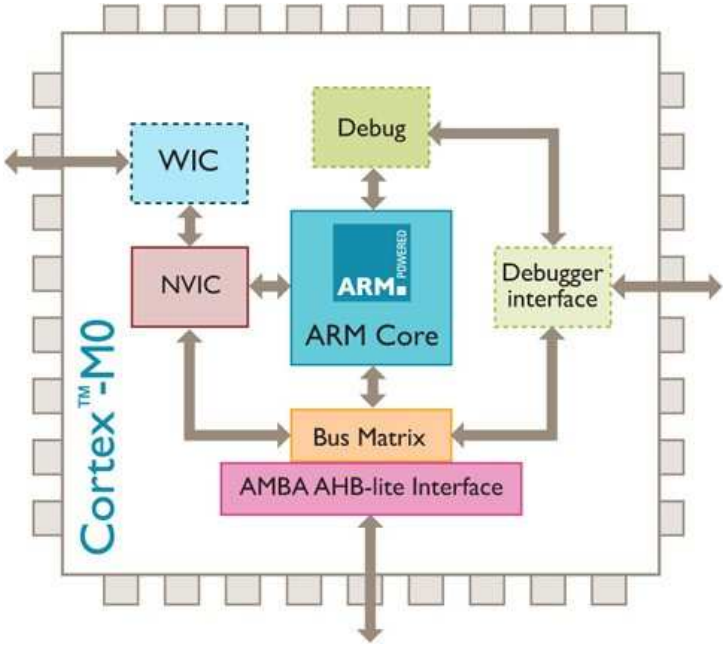
"8/16-bit" applications

Low-cost & simplicity



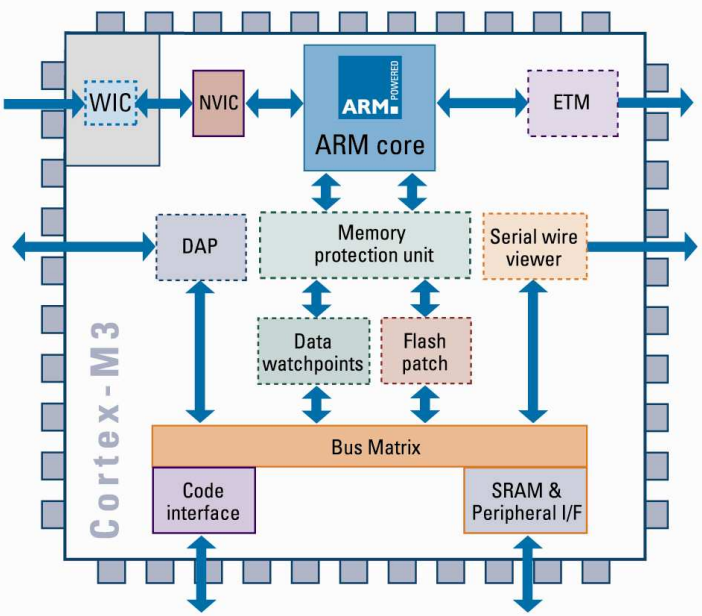
ARM Cortex-M0 Processor

- **32-bit ARM RISC processor**
  - Predominantly 16-bit instruction set
  - 1x AMBA AHB Interface
  - Built-in Nested Vectored Interrupt Controller
  - Optional Wakeup Interrupt Controller
- **Very power and area optimized**
  - Designed for low cost, low power
- **Easy to program and integrate**
  - Just one AHB bus interface, and simple instruction set of 56 instructions
- **Deterministic instruction execution timing**
  - Instructions and interrupts have a fixed timing



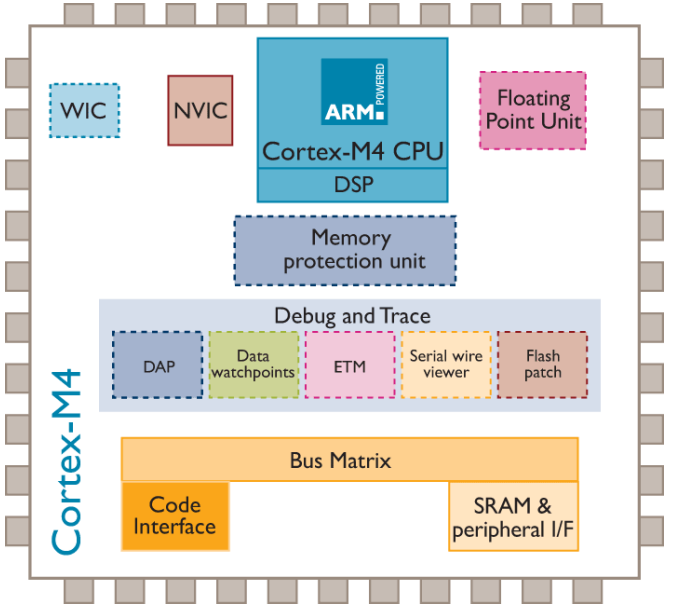
ARM Cortex-M3 processor

- **Rich, unified Thumb-2 high performance instruction set**
  - Smallest code size and reduced memory requirements
  - Fast MAC support
  - Accelerated bit-field processing
- **Harvard architecture**:
  - Allows simultaneous code and data access
  - Reduce interrupt latency
- **Fully configurable to balance features and silicon area**
  - Low latency, integrated Nested Vectored Interrupt Controller (NVIC)
  - Sophisticated debug and trace support
  - Memory Protection Unit (MPU)
  - Embedded Trace Macrocell (ETM)



ARM Cortex-M4 processor

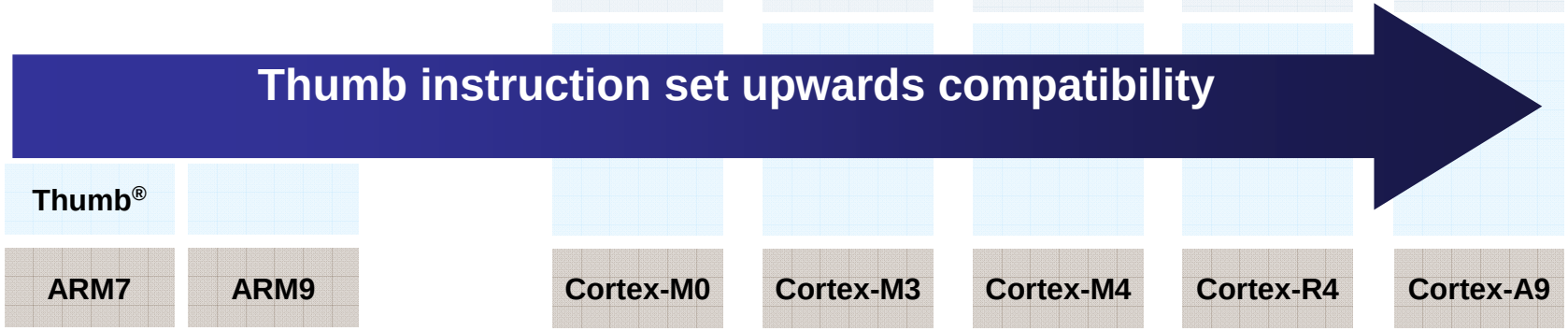
- **Compatible with Cortex-M3 processor**
  - Supports all features in Cortex-M3
- **Additionally:**
- ARMv6 SIMD and DSP
  - Easy migration from ARM9E
- Single cycle MAC
  - Performance improvement for critical algorithms
- Optional single precision FPU
  - Add, subtract, multiply, divide, MAC and sqrt
  - Fused MAC – higher precision
  - Power down option for the Floating Point Unit
- **Launched in early 2010**
- First partner silicon available now



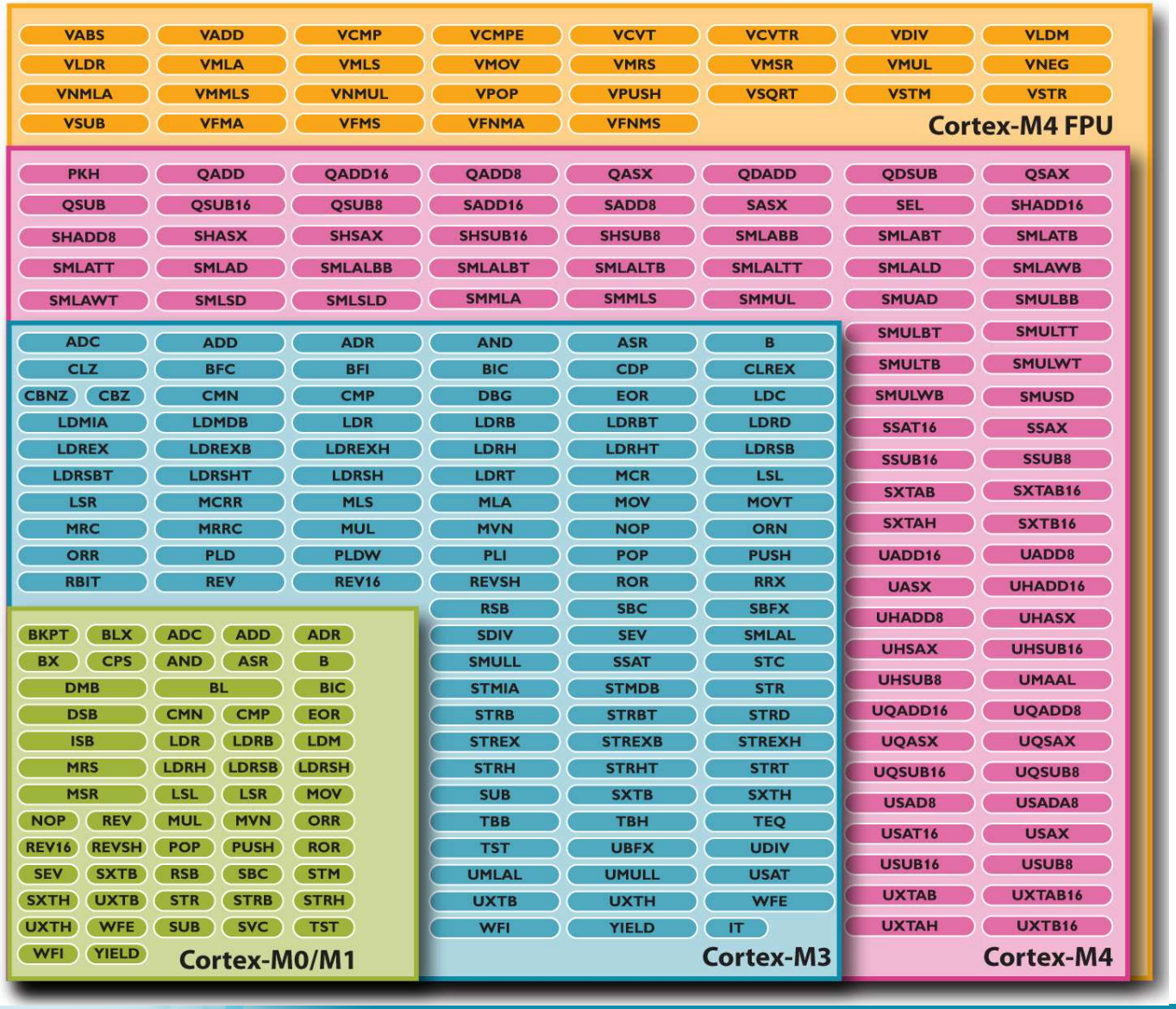
Dotted boxes denote optional blocks

Easy software migration

- **Thumb**
  - 32-bit operations in 16-bit instructions
  - Introduced in ARM7TDMI® processor (‘T’ stands for Thumb)
  - Subsequently supported in every ARM processor developed since
- **Thumb-2**
  - Enables a performance optimised blend of 16/32-bit instructions
  - All processor operations can all be handled in ‘Thumb’ state
  - Supported across the Cortex-M processor range



Powerful Cortex-M instruction set



Instruction Set features

- ARMv6-M
  - Cortex-M0/M1
  - Mostly 16-bit instructions
  - Baseline ALU operations
  - Various Addressing modes
  - Multiply (32x32=32)
  - Data conversion
  - Thumb-2 system instructions
- ARMv7-M(E)
  - Cortex-M3/M4
  - All features in ARMv6-M
  - More addressing modes
  - Larger branch ranges / offset values
  - Hardware Divide
  - MAC, saturation
  - Bit field processing
- Cortex-M4/M4F
  - SIMD
  - Floating point

Data conversions

- Convert between sizes

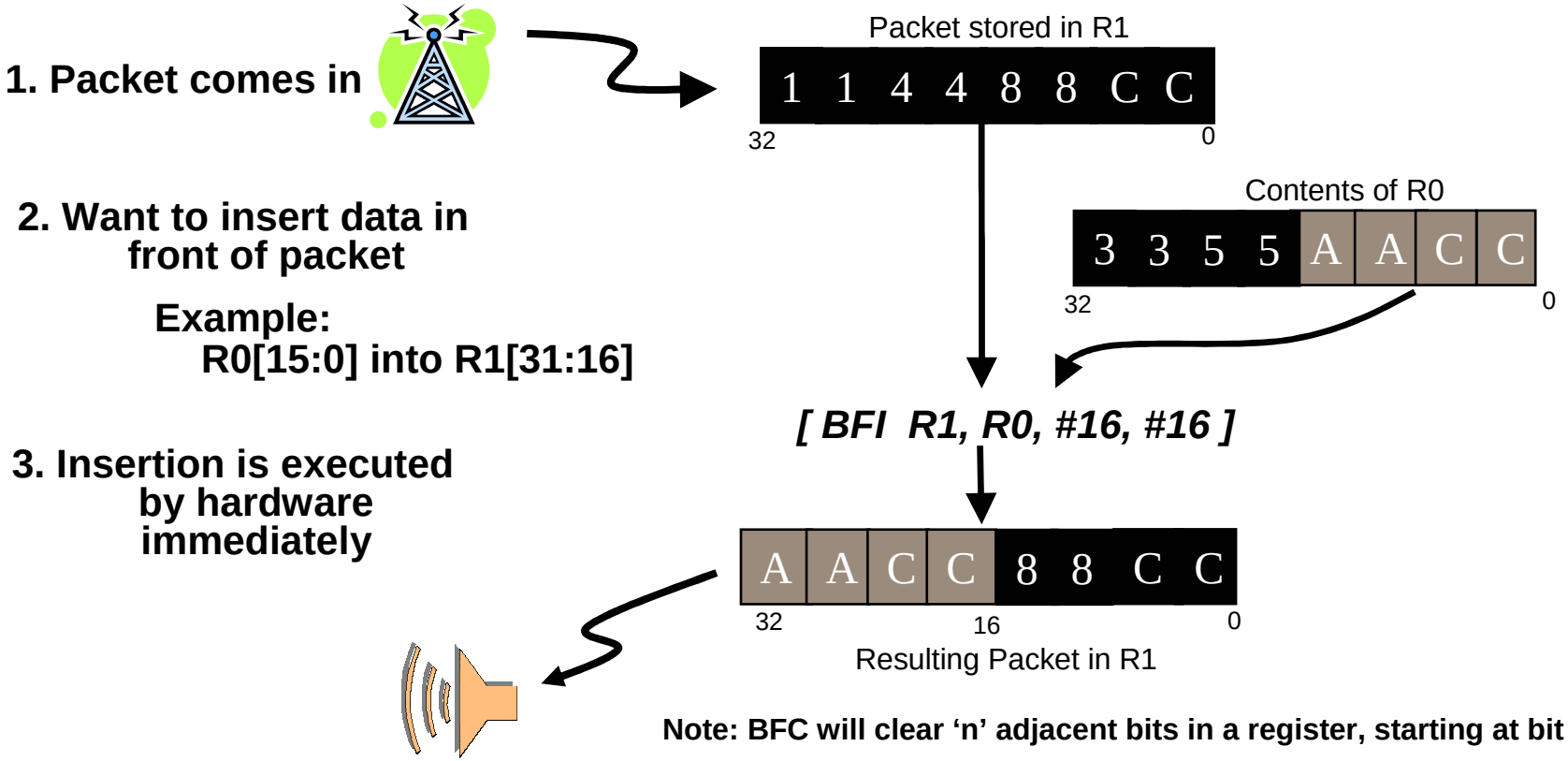
Instructions	Operations
UXTB	Extend byte to word
UXTH	Extend halfword to word
SXTB	Signed extend byte to word
SXTH	Signed extend halfword to word

- Convert between endianess

Instructions	Operations
REV	Reverse byte order in word
REV16	Reverse byte order in half word
REVSH	Reverse byte order in lower half word and signed extend

Bit Manipulation – BFI / BFC

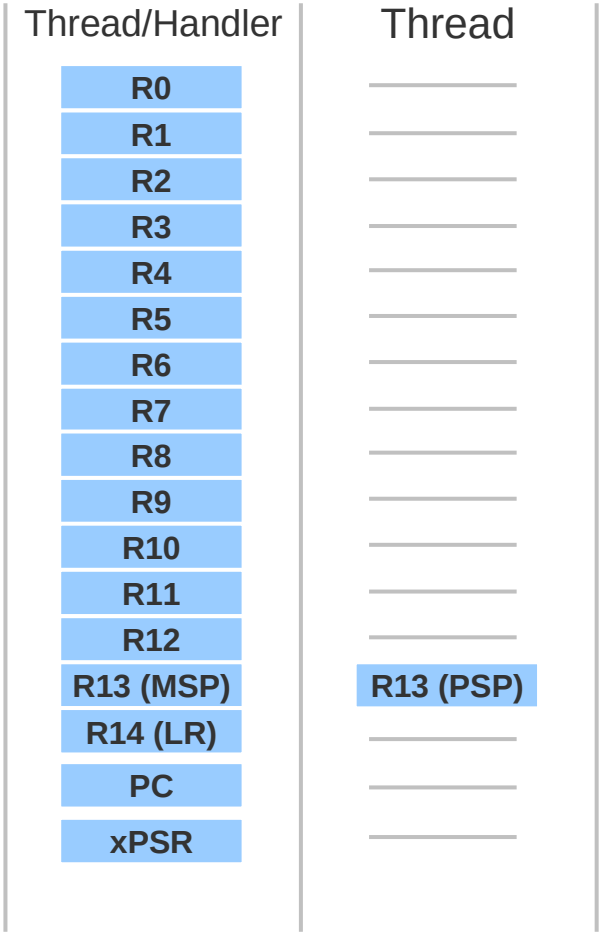
- Insert or clear any number of adjacent bits anywhere in a register
  - Ideal for modifying or stripping packet headers



Note: BFC will clear 'n' adjacent bits in a register, starting at bit 'm'.

Eg: BFC R0, #4, #8 will clear a byte starting from bit 4.

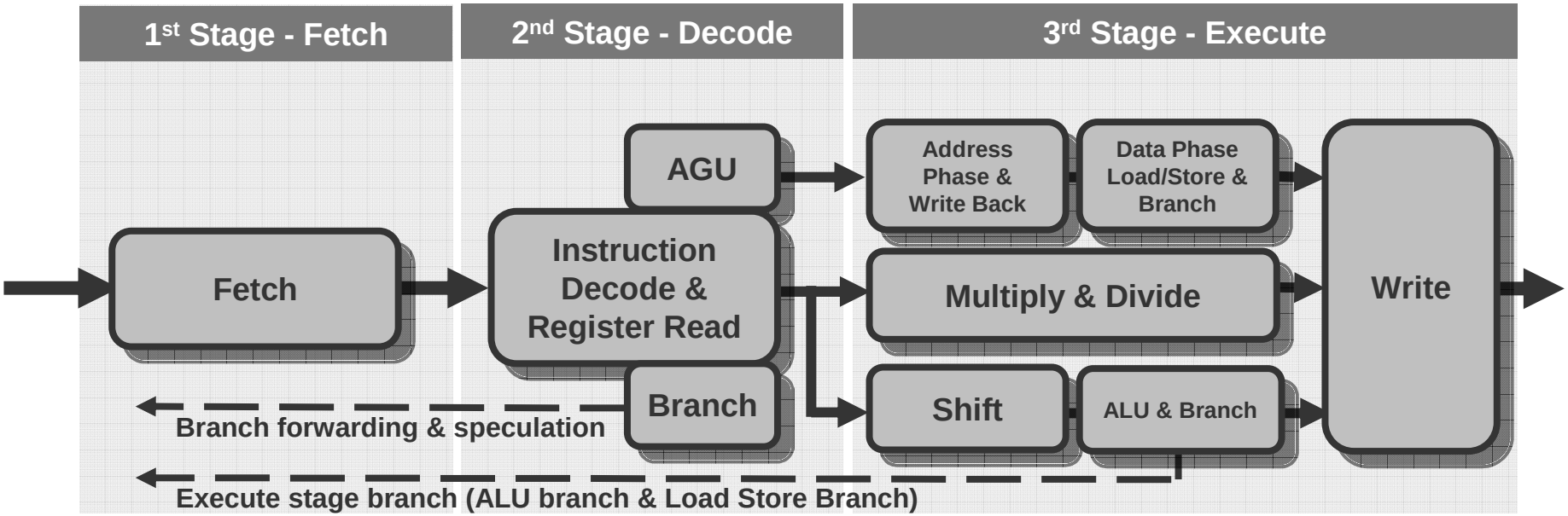
Register File



- **Register set just 18 registers**
  - Compares with 38 registers for traditional ARM
  - Reduces area of Cortex-M3
  - No shadow (mode) registers
- **Same 'visible' register file as traditional ARM**
  - Allows Unified Assembler to mimic ARM instructions in Thumb-2
- **Two stack pointers – Main and Process**
- **Very compiler friendly**
  - Flexible register scheme
  - Single-cycle multiply possible between any of the registers
  - Any register can be used as a pointer to data structures/arrays

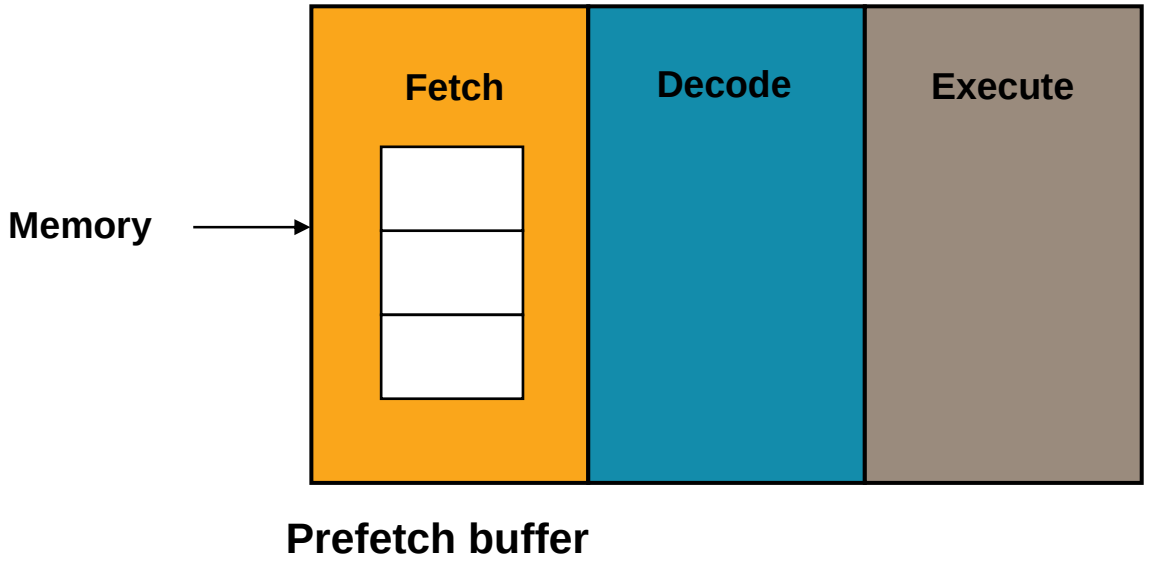
PipelineOverview

- **Harvard architecture**
  - Separate Instruction & Data buses enable parallel fetch & store
- **Advanced 3-Stage Pipeline**
  - Includes Branch Forwarding & Speculation
- **Additional Write-Back via Bus Matrix**



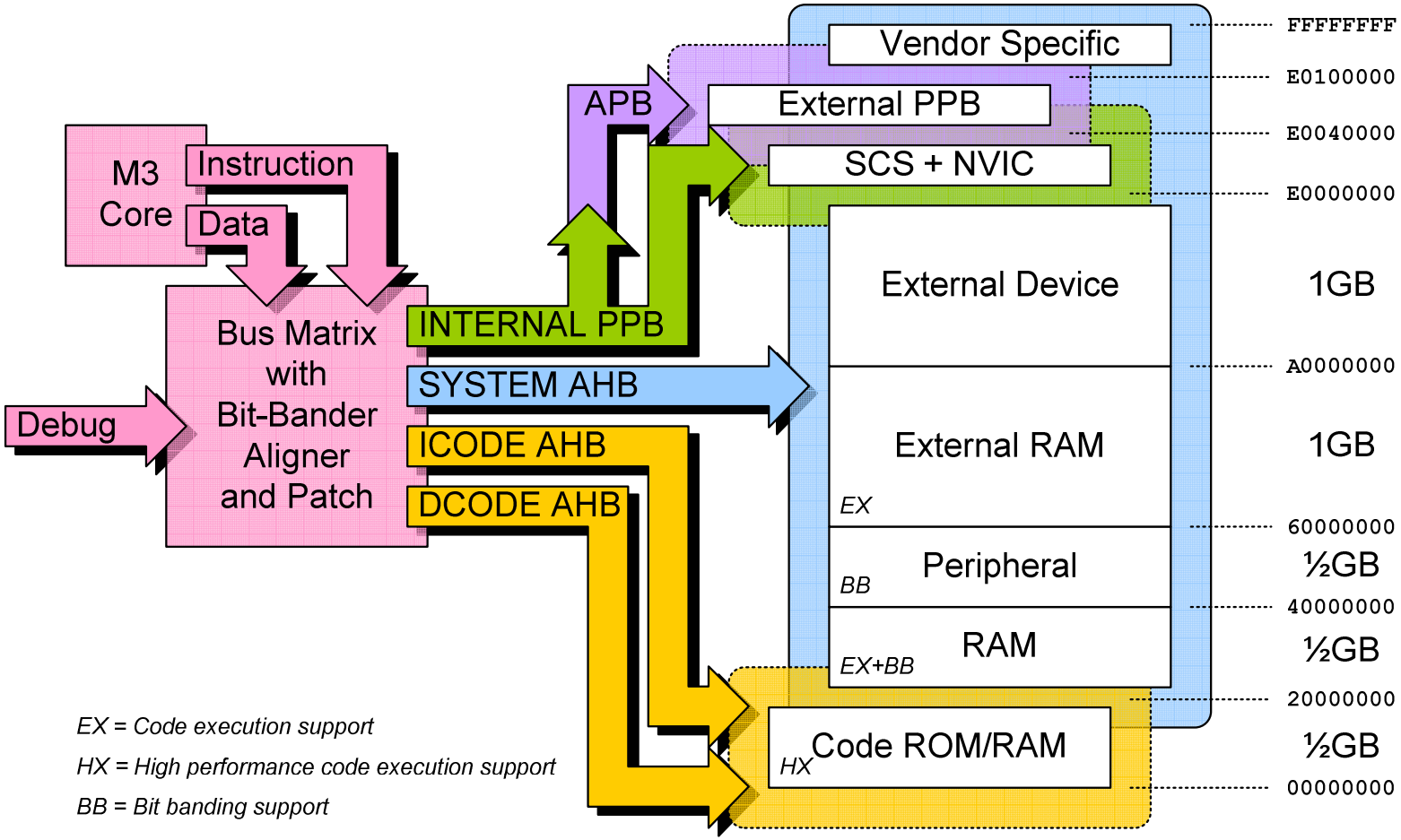
Fetch stage

- Instructions are fetched from Code (or other executable) memory
  - Word-aligned 32-bit read accesses on AHB bus
  - One 32-bit opcode or any combination of 16-bit and halves of 32-bit opcodes
  - Unaligned 32-bit branch target takes two accesses to fetch
- Prefetch buffer holds 3 words for Cortex-M3/M4 and 3 half-words for Cortex-M0
  - Reduce impact of flash wait states and unaligned instruction placement



Memory Map

- Very simple linear 4GB memory map
- Fixed map required to host system components and simplify implementation
- The Bus Matrix partitions memory access via the AHB and PPB buses

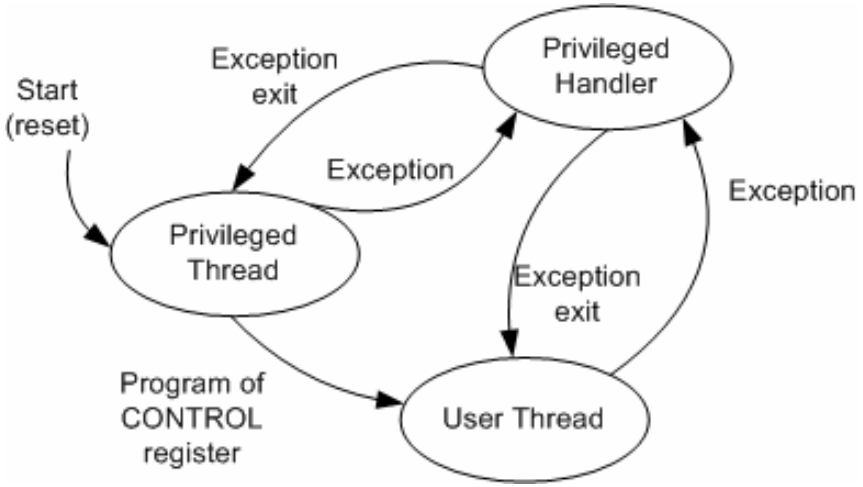


Processor operation modes

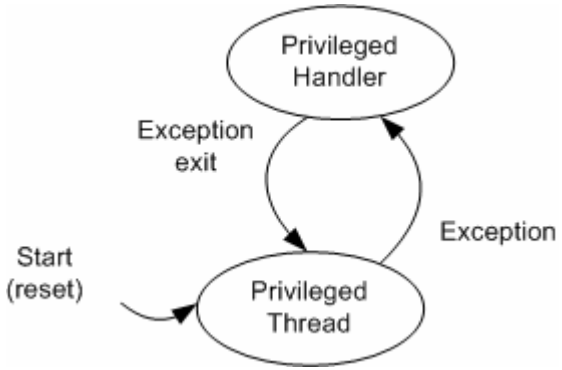
- Cortex-M3/M4:
  - 2 modes (thread, handler)
    - Normal prog or exception handler
  - 2 privilege levels (privileged, user)
    - Basic “security” model
    - Memory access protection

- Cortex-M0:
  - 2 modes (thread, handler)
    - Normal prog or exception handler

Cortex-M3/M4:



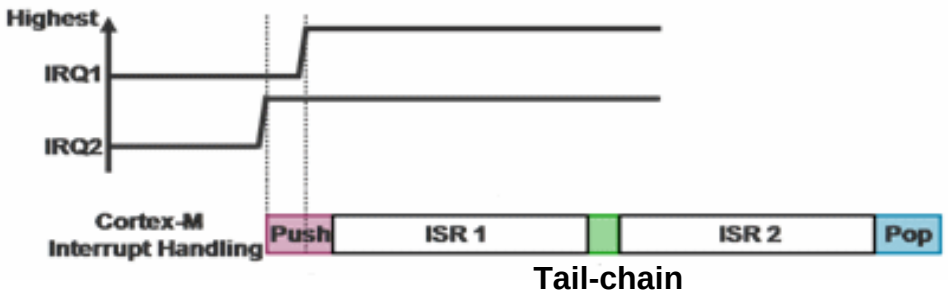
Cortex-M0:



Nested Vectored Interrupt Controller

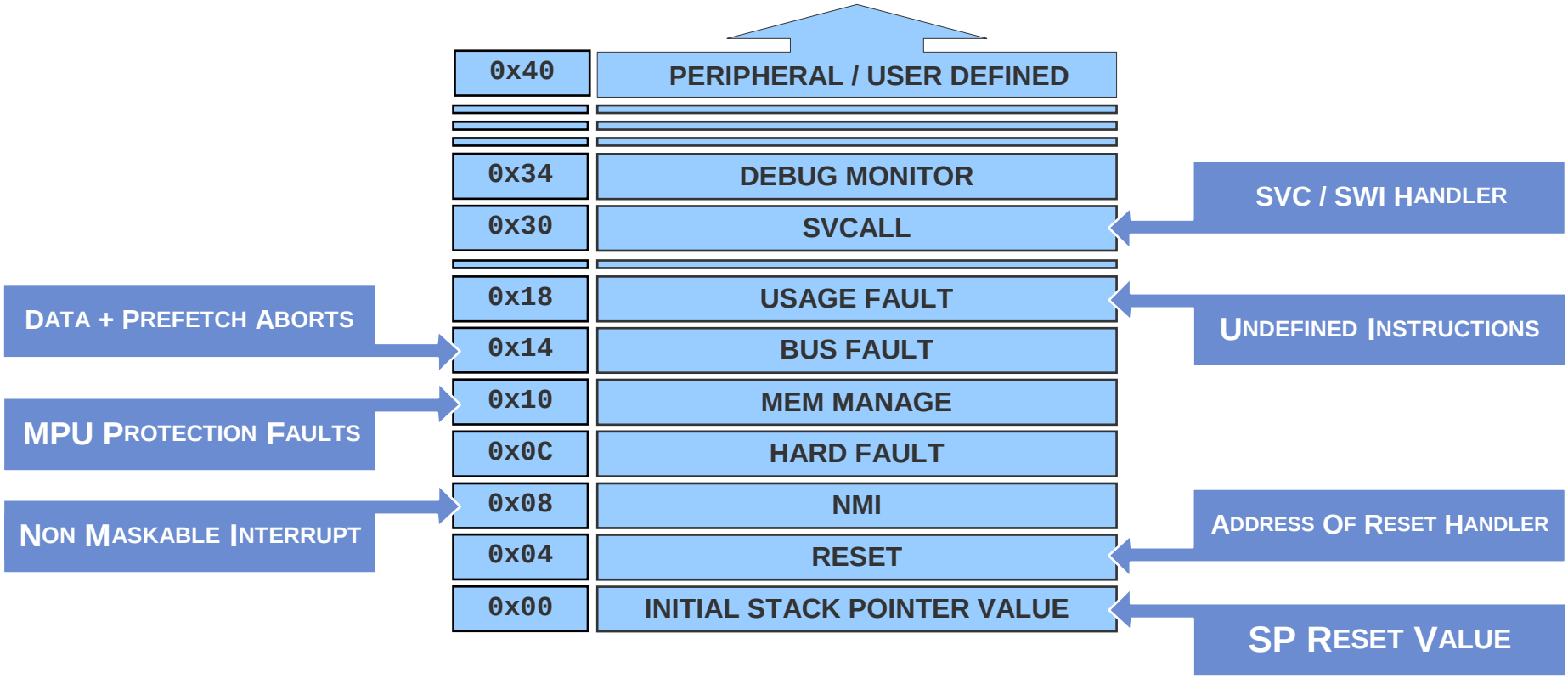
- Faster interrupt response
  - With less software effort
  - 12 cycles, deterministic, low latency
- ISR written directly in C
  - Interrupt table is simply a set of pointers to C routines
  - ISRs are standard C functions
- Integrated NVIC handles:
  - Saving corruptible registers
  - Exception prioritization
  - Exception nesting

8051	Cortex-M
1. SJMP/L JMP from vector table to handler 2. PUSH PSW 3. ORL PSW, #00001000b (to switch register bank) 4. Starting real handler code	1. Starting real handler code



Vector Table

- **ARMv7M architecture implements a relocatable vector table**
  - Contains the address of the function to execute for a particular handler
  - The address is fetched via instruction port allowing parallel register stacking and lookup
  - The first sixteen entries are special with the others mapping to specific interrupts



Exception & Pre-emption Ordering

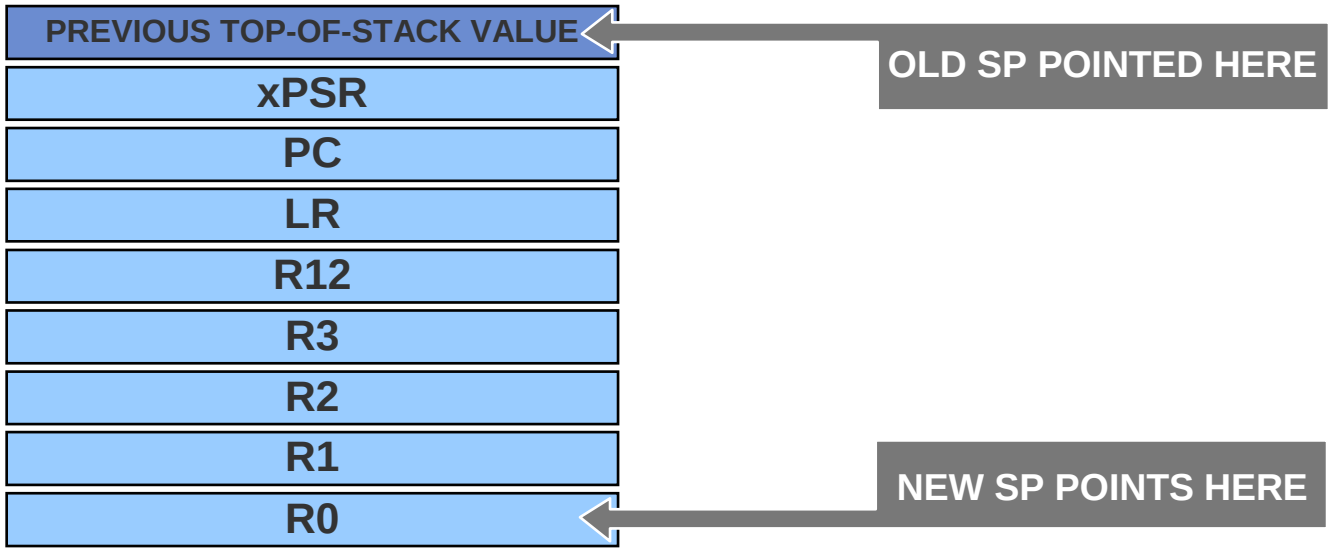
- **Exception handling order is defined by programmable priority**
  - Reset, Non Maskable Interrupt (NMI) and Hard Fault have predefined pre-emption.
  - NVIC catches exceptions and pre-empts current task based on priority
  - Program Counter set to exception address in vector table which directs to handler code

	Exception	Name	Priority	Descriptions
Fault Mode & Start-up Handlers	1	Reset	-3 (Highest)	Reset
	2	NMI	-2	Non-Maskable Interrupt
	3	Hard Fault	-1	Default fault if other hander not implemented
	4	MemManage Fault	Programmable	MPU violation or access to illegal locations
	5	Bus Fault	Programmable	Fault if AHB interface receives error
	6	Usage Fault	Programmable	Exceptions due to program errors
System Handlers	11	SVCall	Programmable	System SerVice call
	12	Debug Monitor	Programmable	Break points, watch points, external debug
	14	PendSV	Programmable	Pendable SerVice request for System Device
	15	Systick	Programmable	System Tick Timer
Custom Handlers	16	Interrupt #0	Programmable	External Interrupt #0
	…	…	…	…
	…	…	…	…
	…	…	…	…
	255	Interrupt #239	Programmable	External Interrupt #239

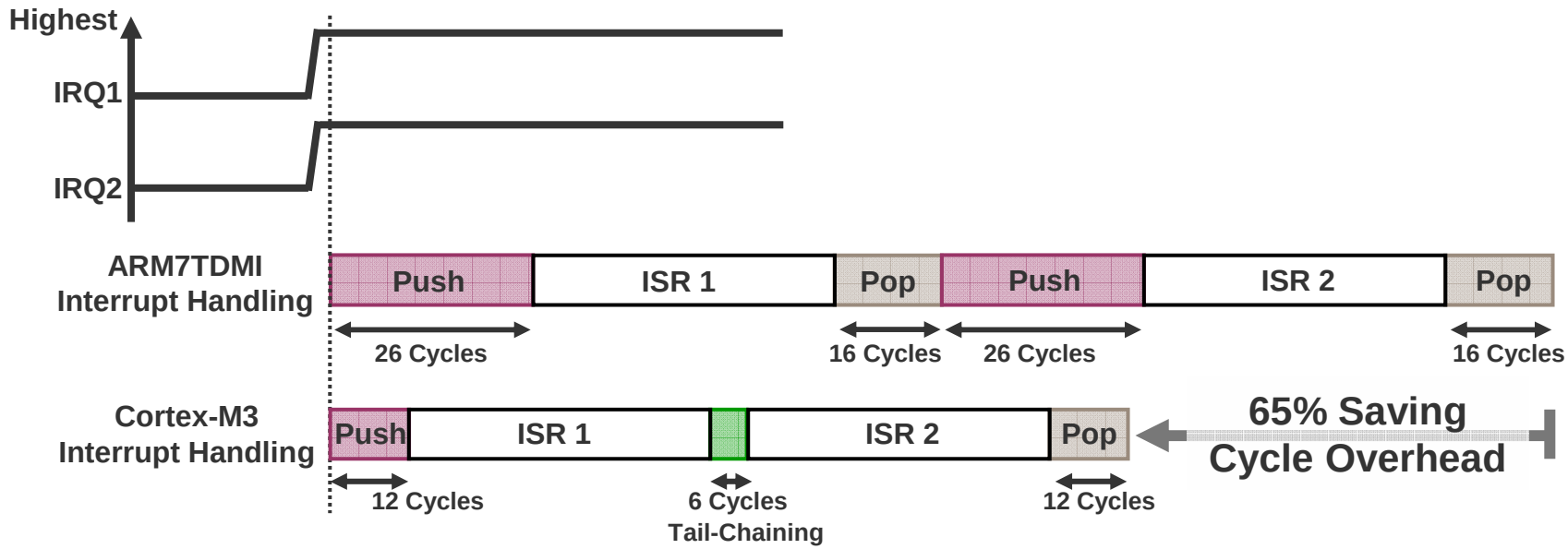
Exception Model

- **Exception Model handles all interrupts, synchronous faults and SVC exceptions**
  - Exceptions cause current machine state to be stacked
  - Stacked registers conform to AAPCS (ARM Architecture Procedure Calling Standard)
- **Exception handlers are trivial as register manipulation carried out by hardware**
  - No assembler code required
  - Simple ‘C’ interrupt service routines

```
void IRQ(void) { /* my handler */ }
```

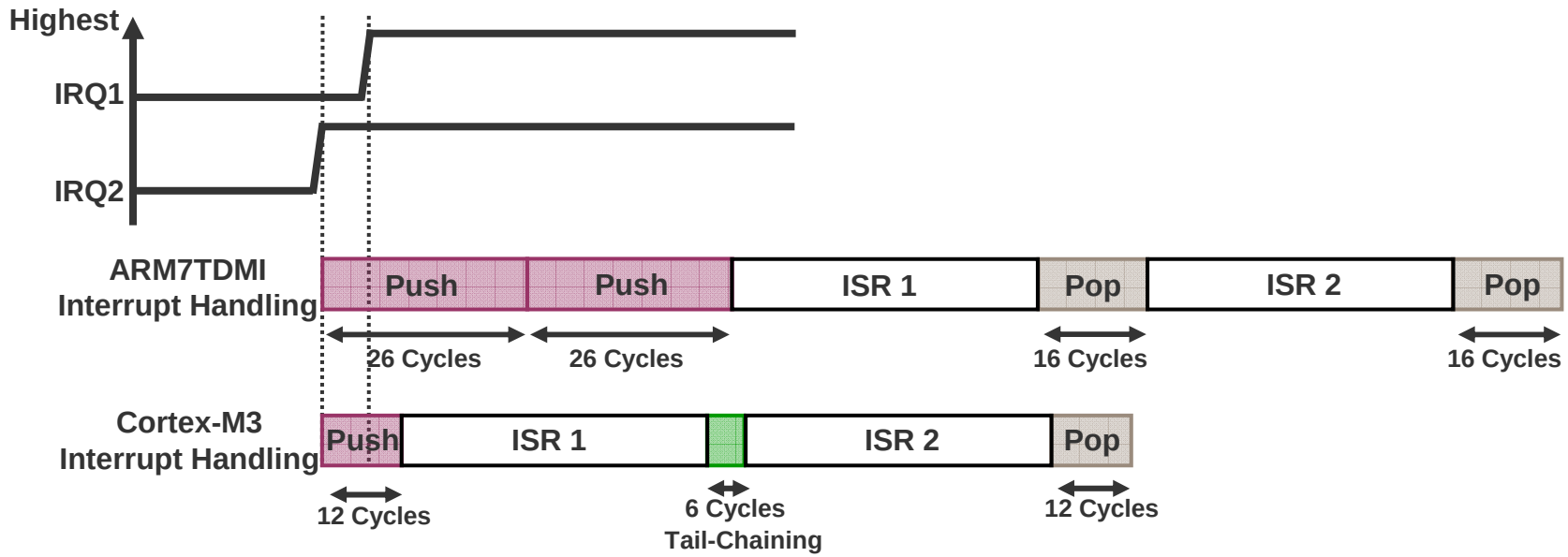


Interrupt Response – Tail Chaining



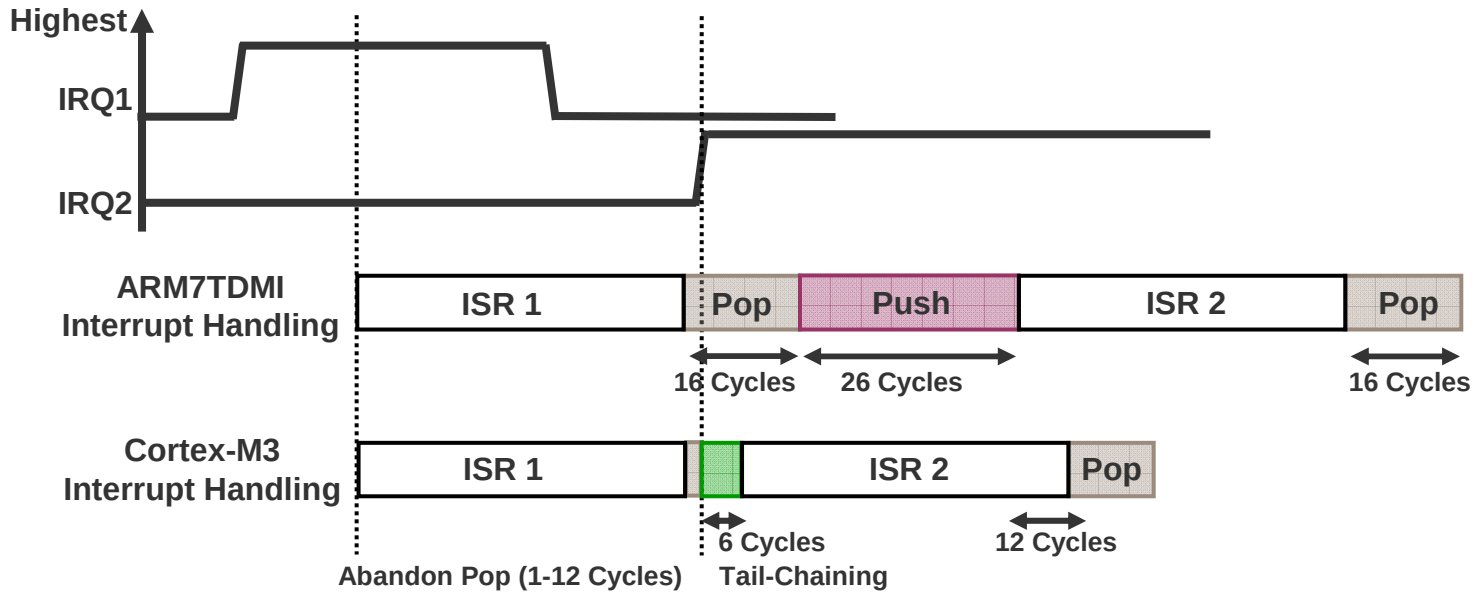
ARM7TDMI	Cortex-M3
▪26 cycles from IRQ1 to ISR1 (up to 42 cycles if in LSM) ▪ 42 cycles from ISR1 exit to ISR2 entry ▪ 16 cycles to return from ISR2	▪ 12 cycles from IRQ1 to ISR1 (Interruptible/Continual LSM) ▪ 6 cycles from ISR1 exit to ISR2 entry ▪ 12 cycles to return from ISR2

Interrupt Response – Late Arriving



ARM7TDMI	Cortex-M3
▪26 cycles to ISR2 entered ▪ Immediately pre-empted by IRQ1 Additional 26 cycles to enter ISR1. ▪ ISR 1 completes Additional 16 cycles return to ISR2.	▪12 cycles to ISR entry ▪Parallel stacking & instruction fetch ▪Target ISR may be changed until last cycle (PC is set) ▪When IRQ1 occurs new target ISR set

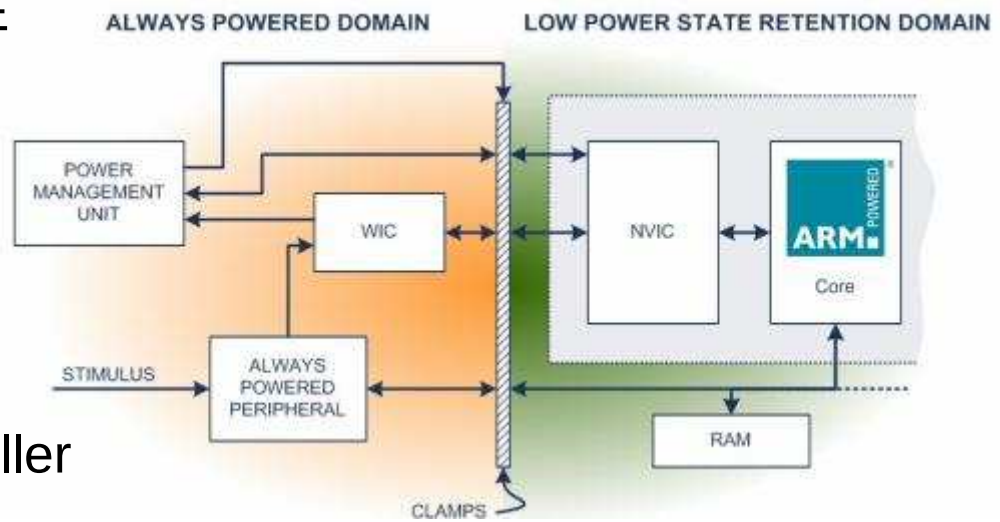
Interrupt Response – Pop Pre-emption



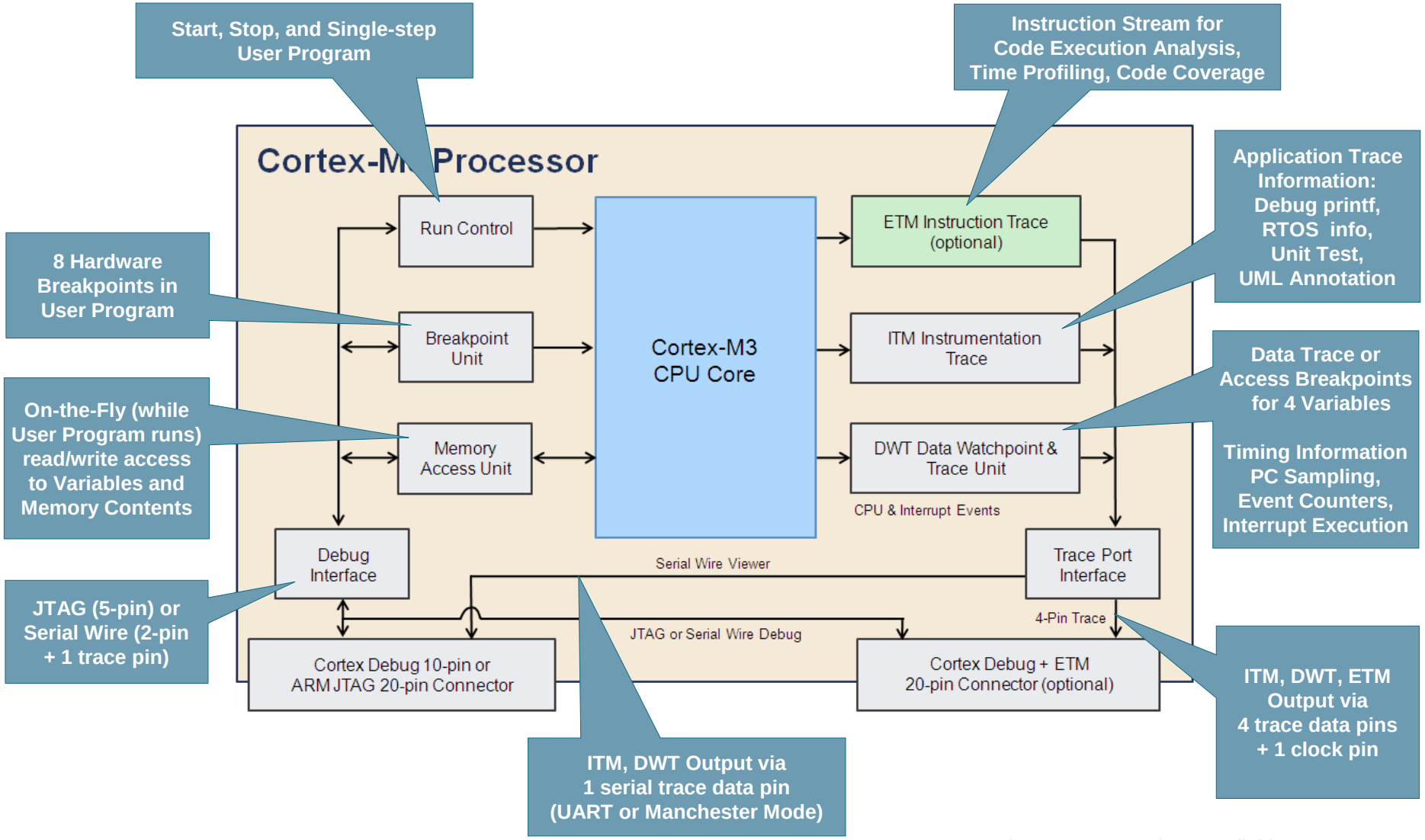
ARM7TDMI	Cortex-M3
▪Load multiple not interruptible ▪ Core must complete the recovery of the stack then re-stack to enter the ISR	▪ Hardware un-stacking interruptible ▪ If interrupted only 6 cycles required to enter ISR2

Cortex-M low power technologies

- Sleep modes are supported architecturally:
  - Sleep mode
  - Deep sleep mode
- Enter sleep mode using WFI/WFE instruction, or
- “Sleep-on-exit” interrupt handling:
  - enables the processor to sleep whenever all outstanding Interrupts are complete
- Optional Wakeup Interrupt Controller (WIC)
  - Allow processor to be powered down and power up when requested
  - enables nW power consumption in deep sleep mode with instant wakeup



CoreSight™ Debug Technology (Cortex-M)



Trace (ETM, ITM, DWT) not available on Cortex-M0

Summary: ARM Cortex-M family

- **A compatible architecture spanning the embedded application range**
  - Designed for deterministic, low power and low area applications.

ARM Cortex-M4

"32-bit/DSC" applications

Efficient digital signal control

ARM Cortex-M3

"16/32-bit" applications

Performance efficiency

ARM Cortex-M1

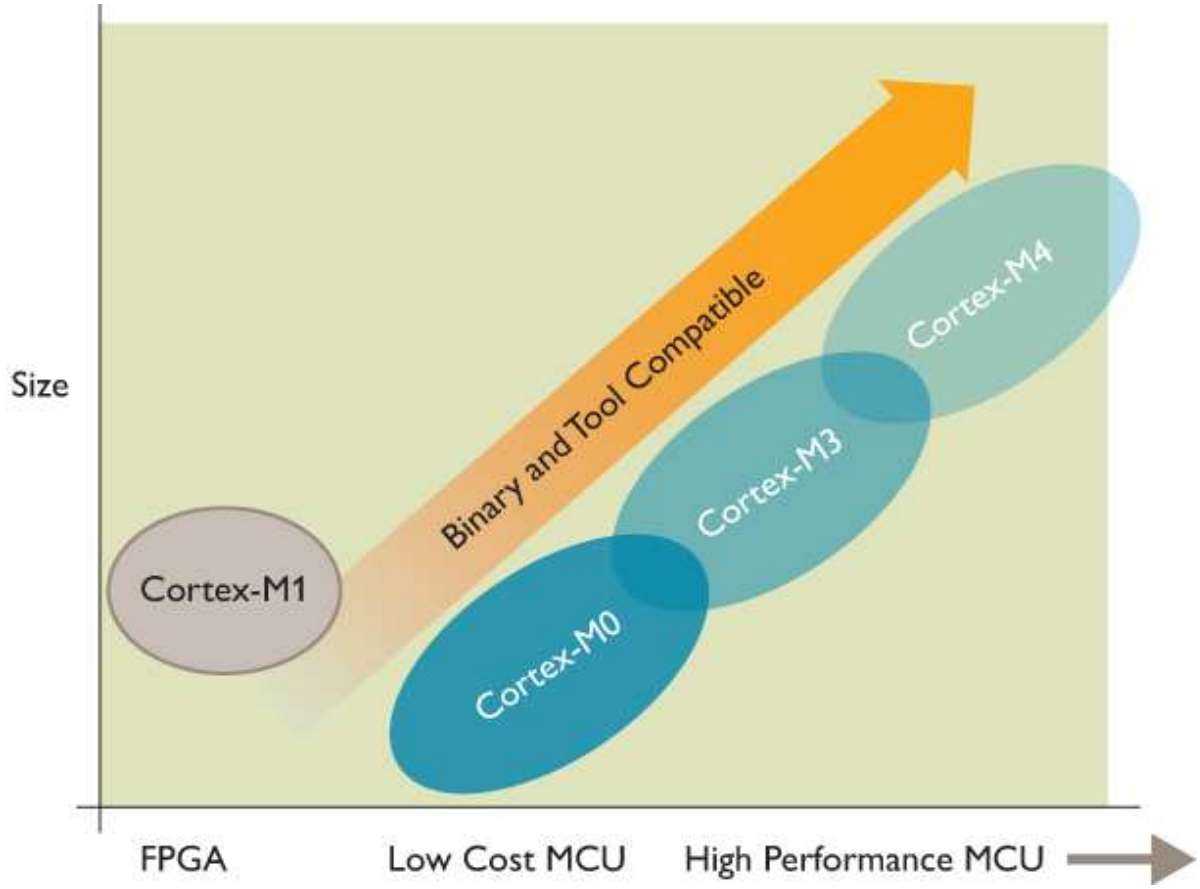
FPGA applications

Optimized for multiple FPGA types

ARM Cortex-M0

"8/16-bit" applications

Low-cost & simplicity



BACKUP SLIDES

Cortex-M0 Pipeline

- Unaligned instruction does not cause stall to the pipeline