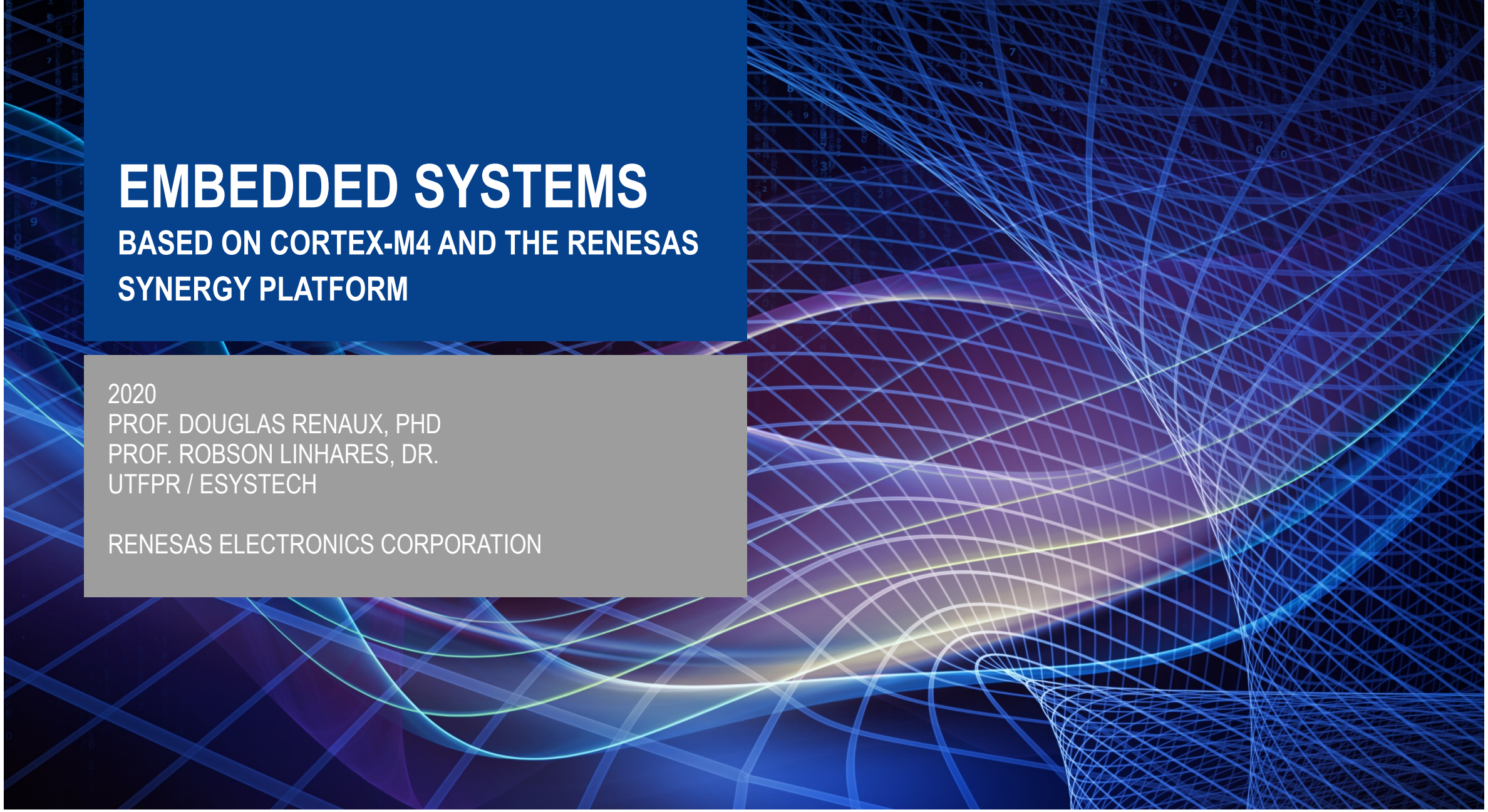


# EMBEDDED SYSTEMS

## BASED ON CORTEX-M4 AND THE RENESAS SYNERGY PLATFORM

2020
PROF. DOUGLAS RENAUX, PHD
PROF. ROBSON LINHARES, DR.
UTFPR / ESYSTECH

RENESAS ELECTRONICS CORPORATION

# 3.1 – ARM HISTORY

- ARM is a British company with focus on microprocessor and GPU design. It currently is owned by the SoftBank Group.
- Although ARM was founded in 1990, its story begins in 1981 in the Acorn Computer Ltd.
  The success of ARM over the last 3 decades is an inspiring story

# 3.1 – ARM HISTORY

- The BBC micro was released by Acorn Computer Ltd in 1981 as part of the BBC Computer Literacy Project.
- The main characteristics of the BBC micro are:
  - CPU: 2 MHz 8-bit processor 6502
  - Memory: up to 128 KBytes
  - Display: PAL/NTSC TV
    640 x 256 pixels (8-bit/pixel)
  - I/O: keyboard, joystick, parallel printer, serial


source: wikimedia.org (CC)

# 3.1 – ARM HISTORY

- In 1984, a new set of requirements for the home computer called for a replacement of the 8-bit 6502. The Motorola 68.000 was considered, however its interrupt response was too slow for the intended use.
  The recent work of Patterson with the Berkeley RISC the MIPS project by Hennessy at Stanford caught the attention of the engineers at Acorn. Hence, they decided to design their own 32-bit RISC processor.
- Starting in October 1984, on April 1985 the ARM1 was born. At this time, ARM meant Acorn RISC Machine and was the name of a project at Acorn Computer.

"Steve Furber is one of the brightest guys I've ever worked with – brilliant " says Acorn CEO Hermann Hauser, "and when we decided to do a microprocessor on our own I made two great decisions – I gave them two things which National, Intel and Motorola had never given their design teams: the first was no money; the second was no people. The only way they could do it was to keep it really simple."

# 3.1 – ARM HISTORY

- ARM2 was released in 1987 and was used in the BBC Archimedes 300, a successor of the BBC Micro.

- Characteristics:
  - CPU: ARM2
  - Memory: up to 16 MBytes

source: wikimedia.org (CC)

# 3.1 – ARM HISTORY

- ARM was founded in 1990 with the name Advanced RISC Machines. It had financial support from VLSI, Acorn and Apple.

- ARM was created with the purpose of designing processors, thus, becoming one of the first Intellectual Property companies in the world.

ARM Ltd. Headquarters - 1990

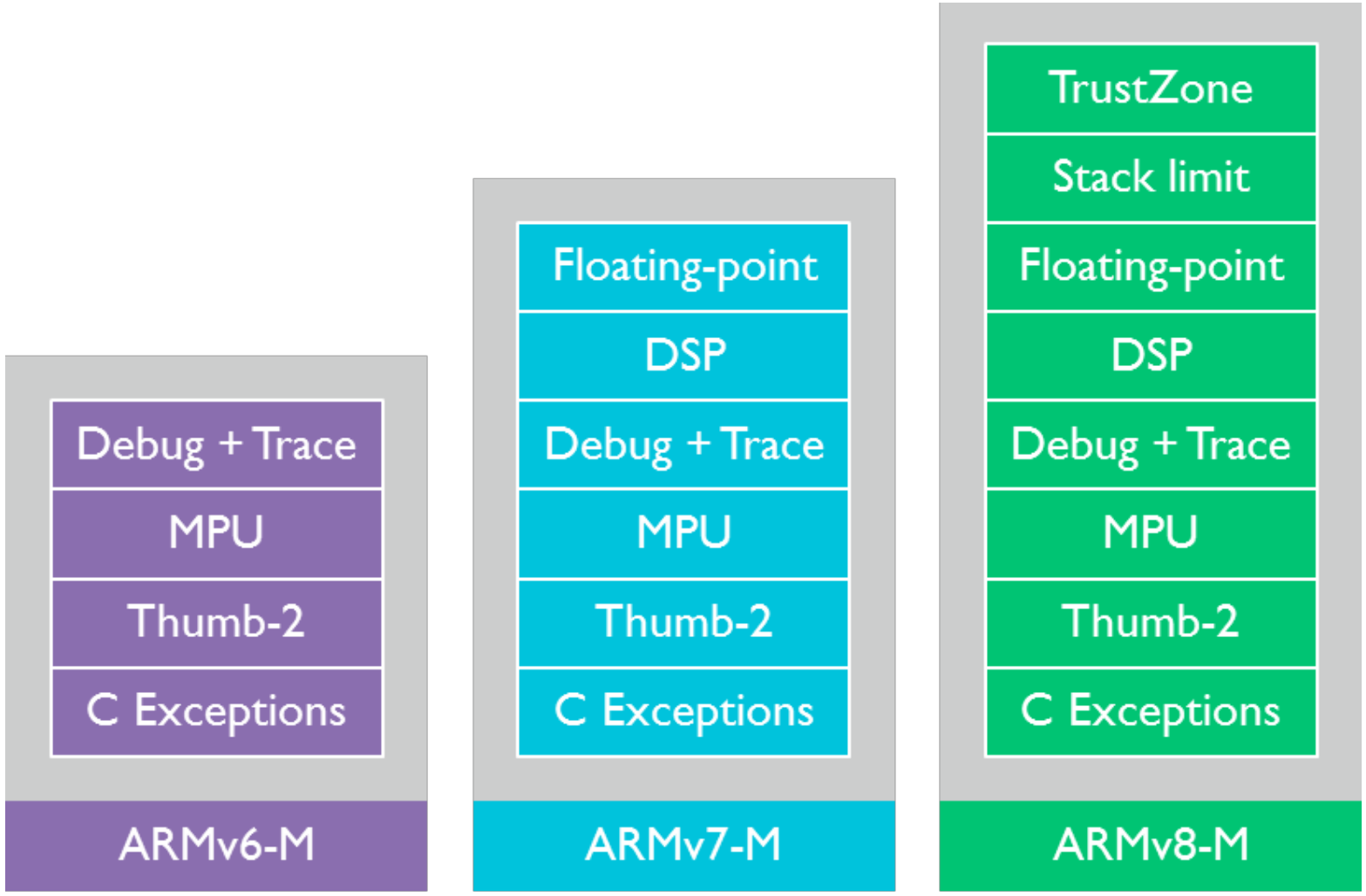
source: ARM

# 3.1 – ARM HISTORY

- In 1994, the ARM7TDMI was released, being a major success on the market.
- In 2004, the Cortex-M, Cortex-R and Cortex-A series were released.

# 3.1 – ARM HISTORY

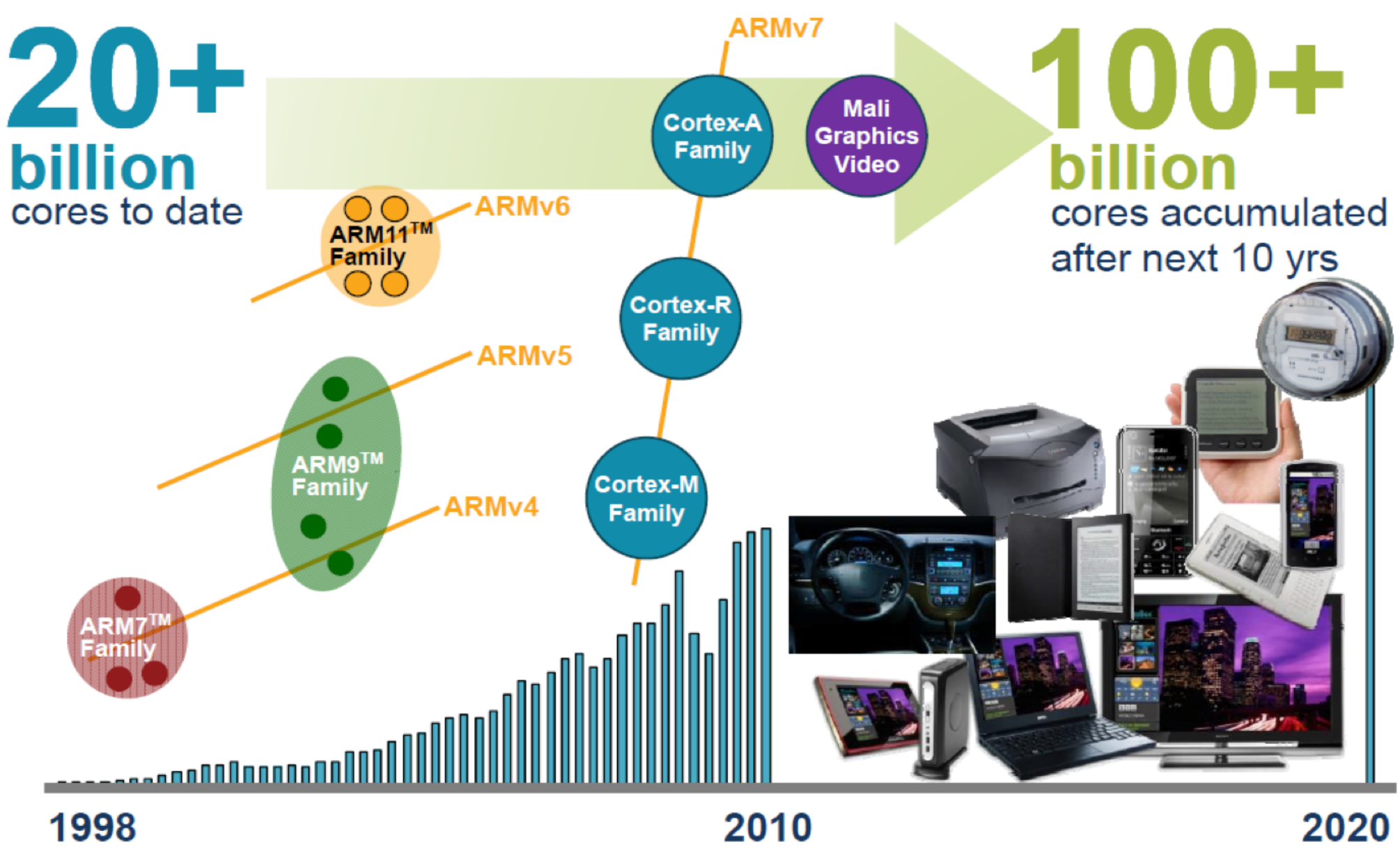
- Over time, several ARM architecture versions were released. From the seminal ARMv1 to the ARMv8.3-A, released in 2017. The documentation of these standards is available on the arm.com website.
- For each architectural version there are several implementations



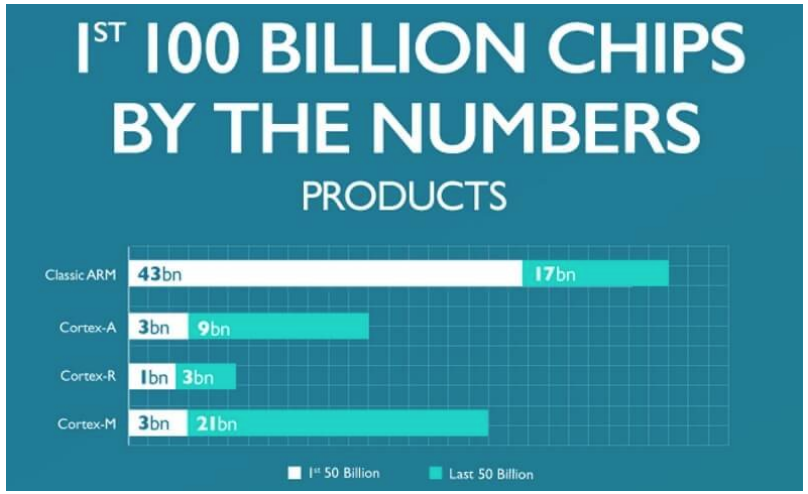

Cortex-M0
Cortex-M0+

Cortex-M3
Cortex-M4
Cortex-M7

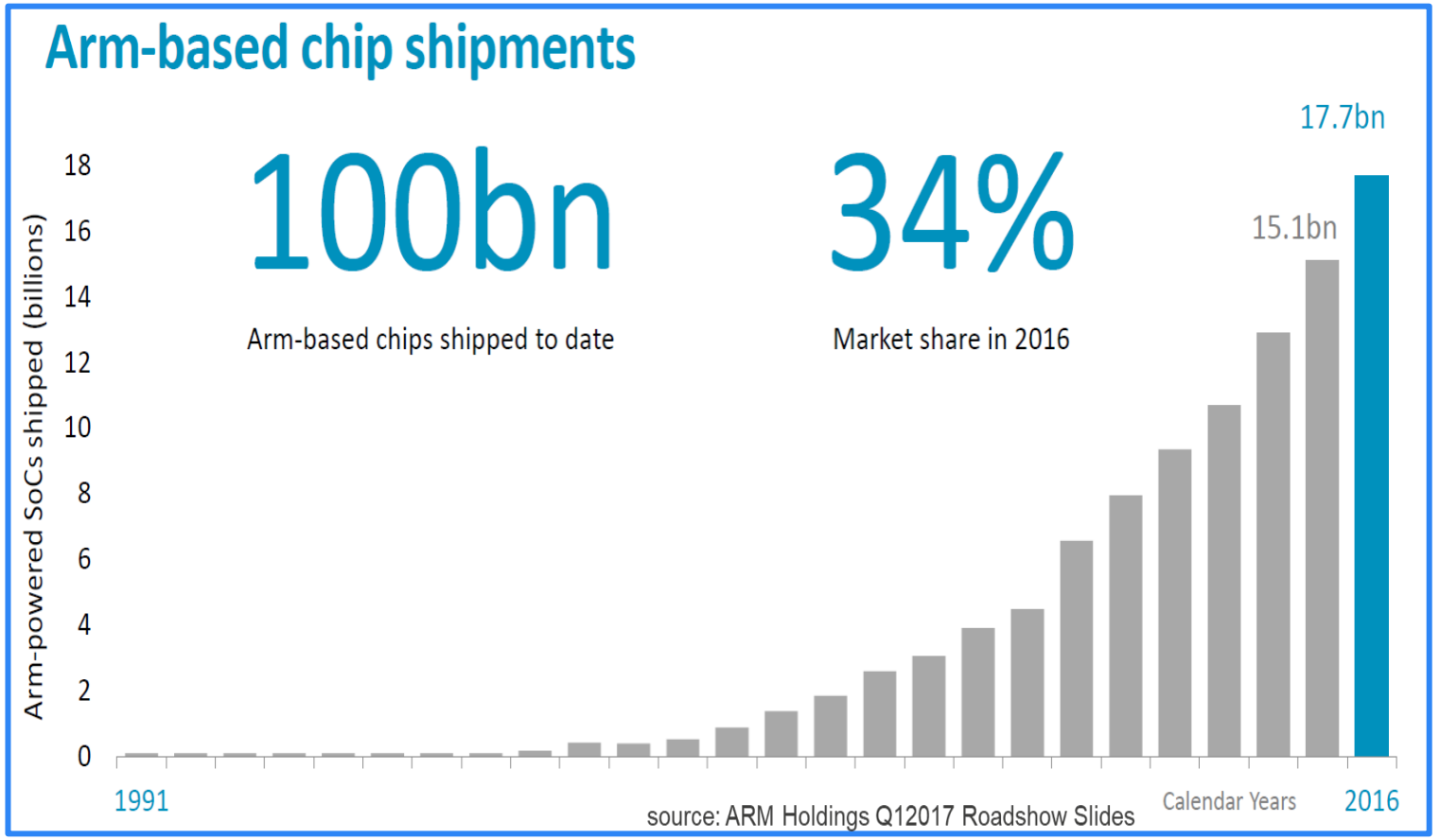
Cortex-M23
Cortex-M33



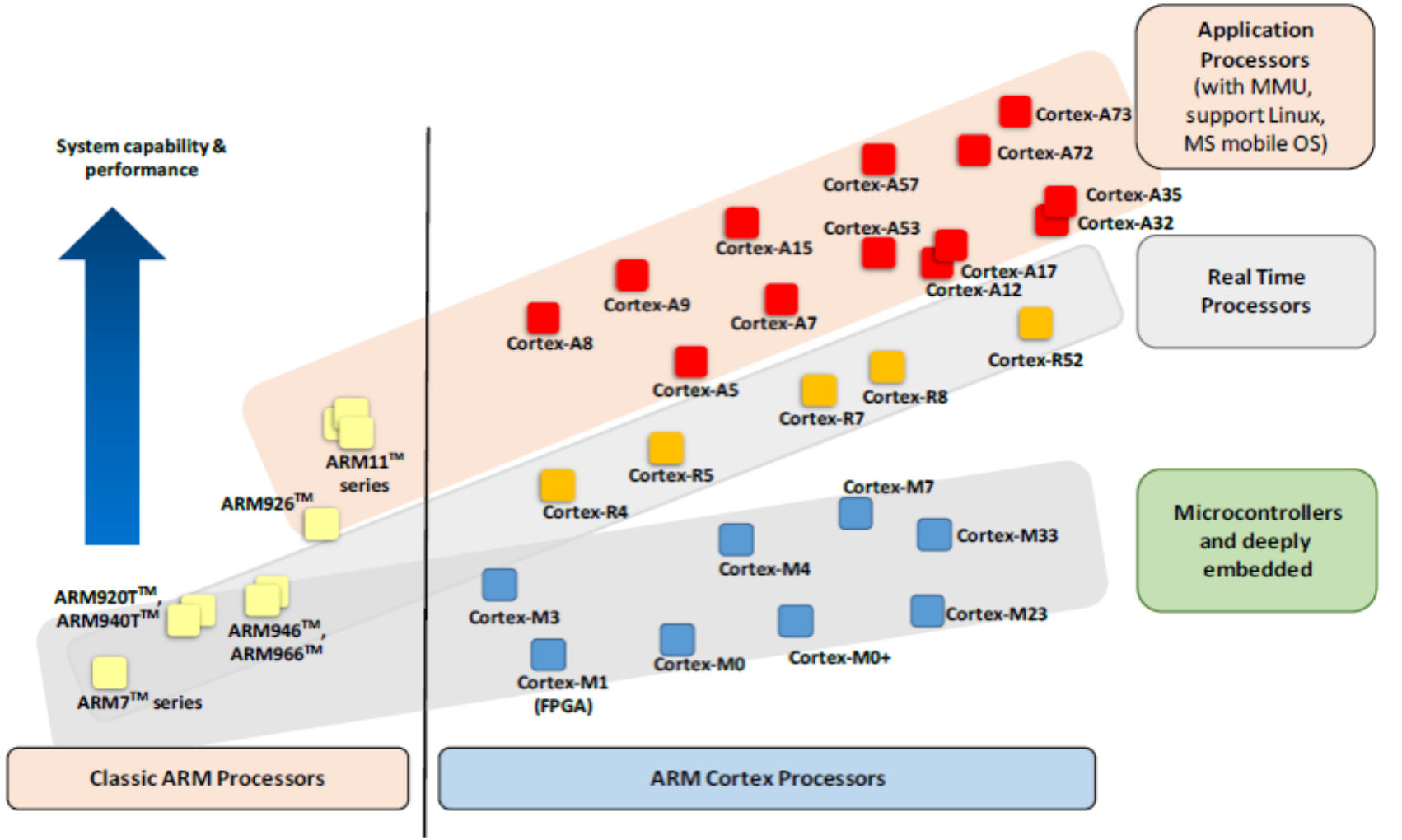

source: ARM
The Future is in Your Hands - Peter Middleton




source: community.arm.com

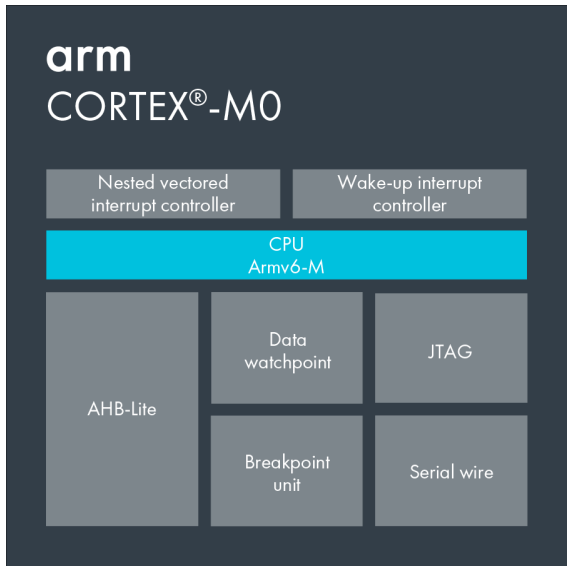


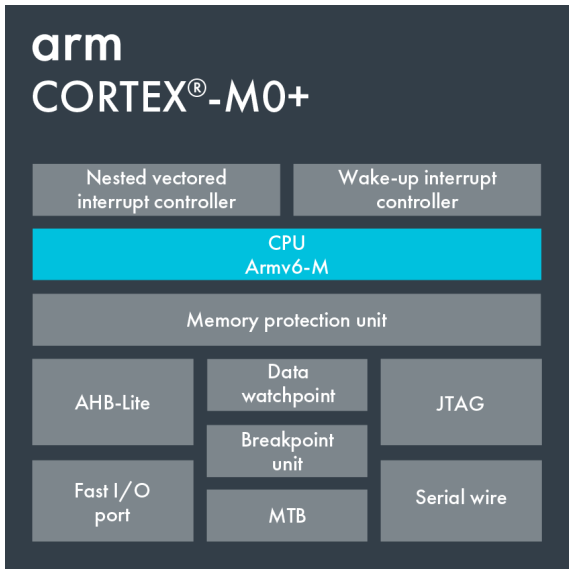
# ARM PROCESSOR FAMILY

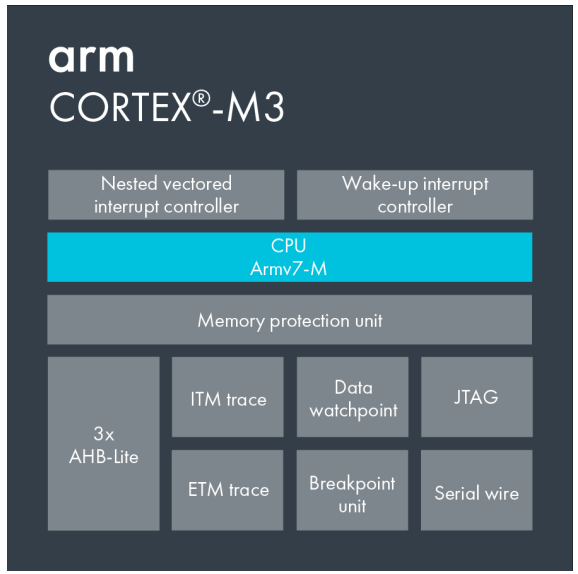


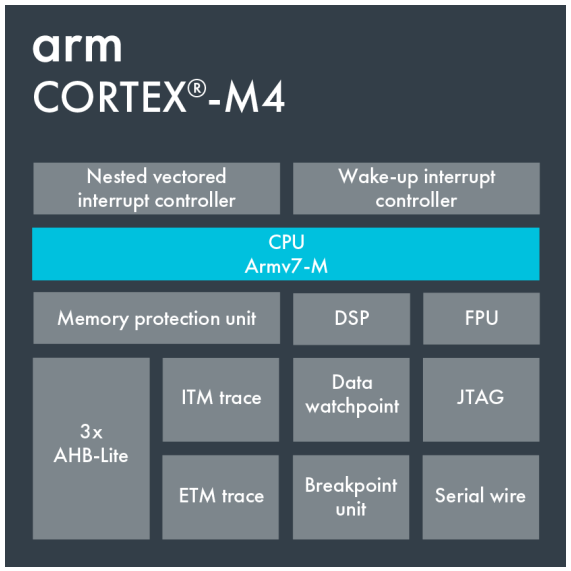

source: ARM
ARM Cortex-M for Beginners - Yiu

# INTRODUCING THE CORTEX-M FAMILY – 1/2



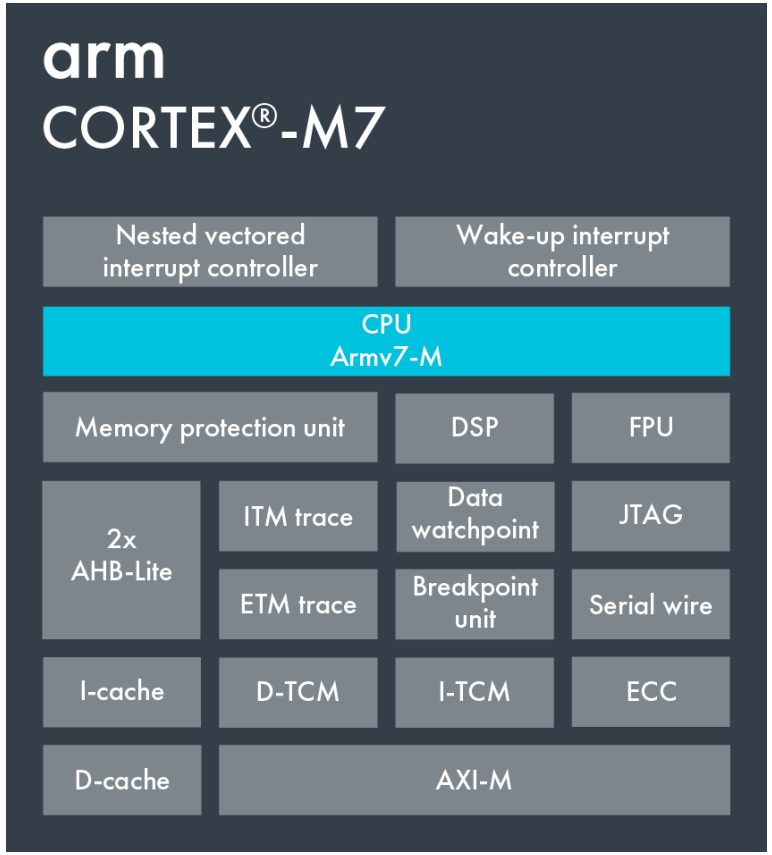



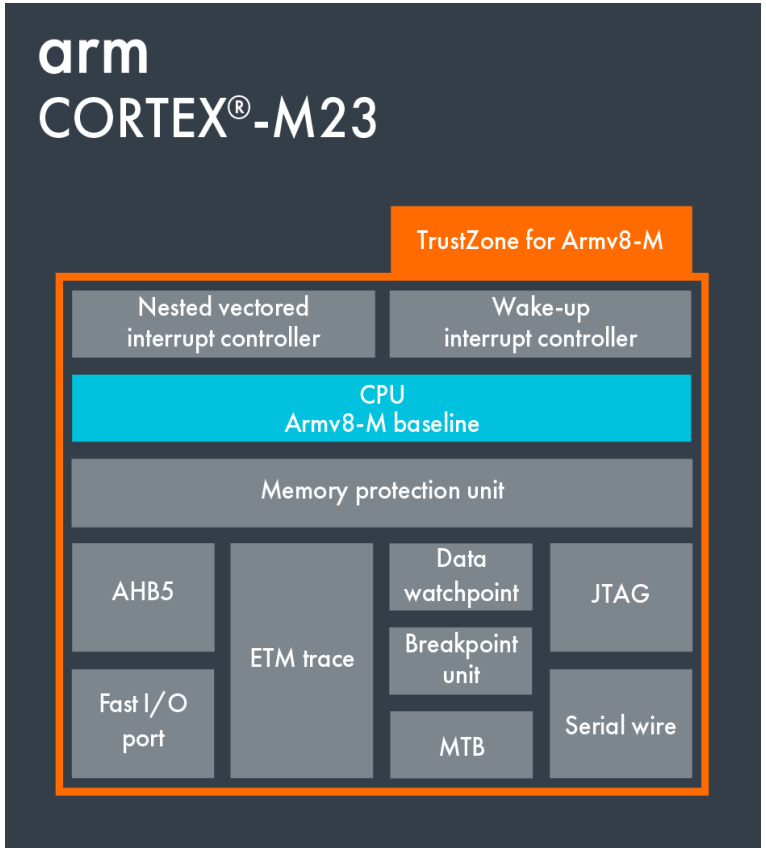



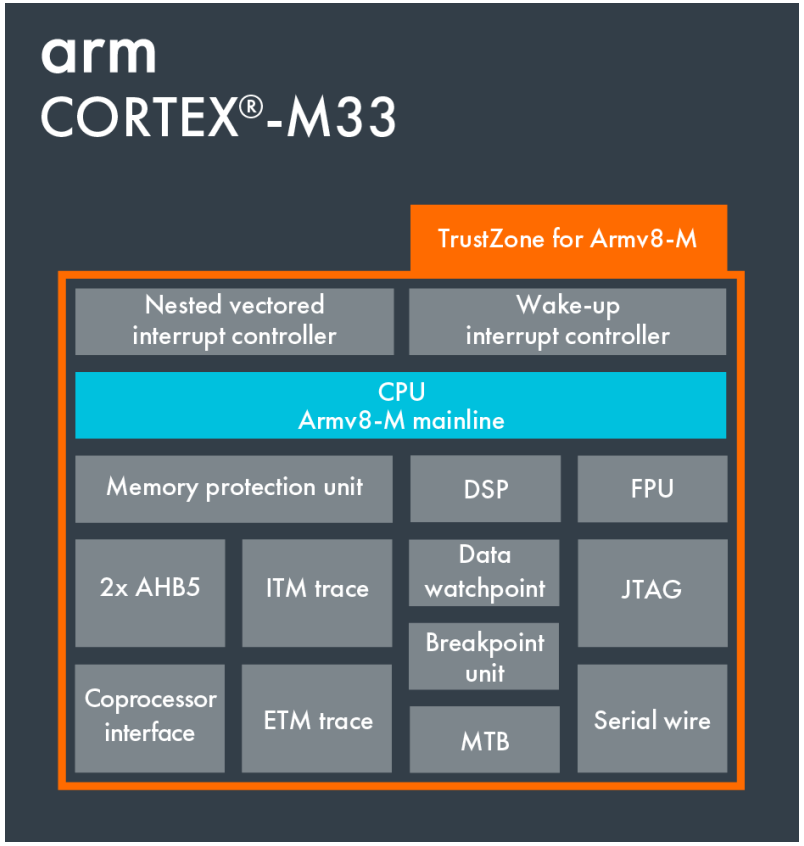




source: community.arm.com

# INTRODUCING THE CORTEX-M FAMILY – 2/2

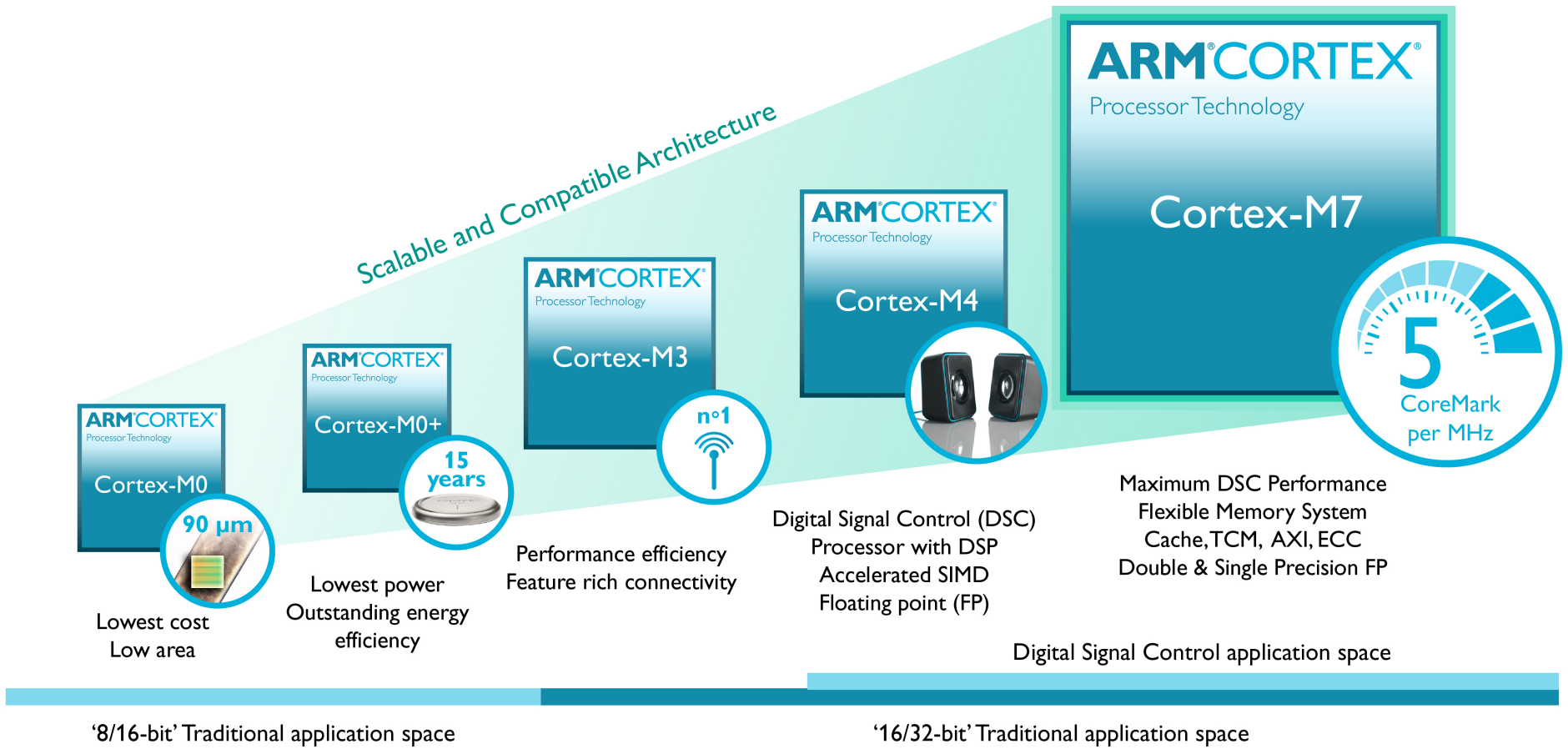



source: community.arm.com

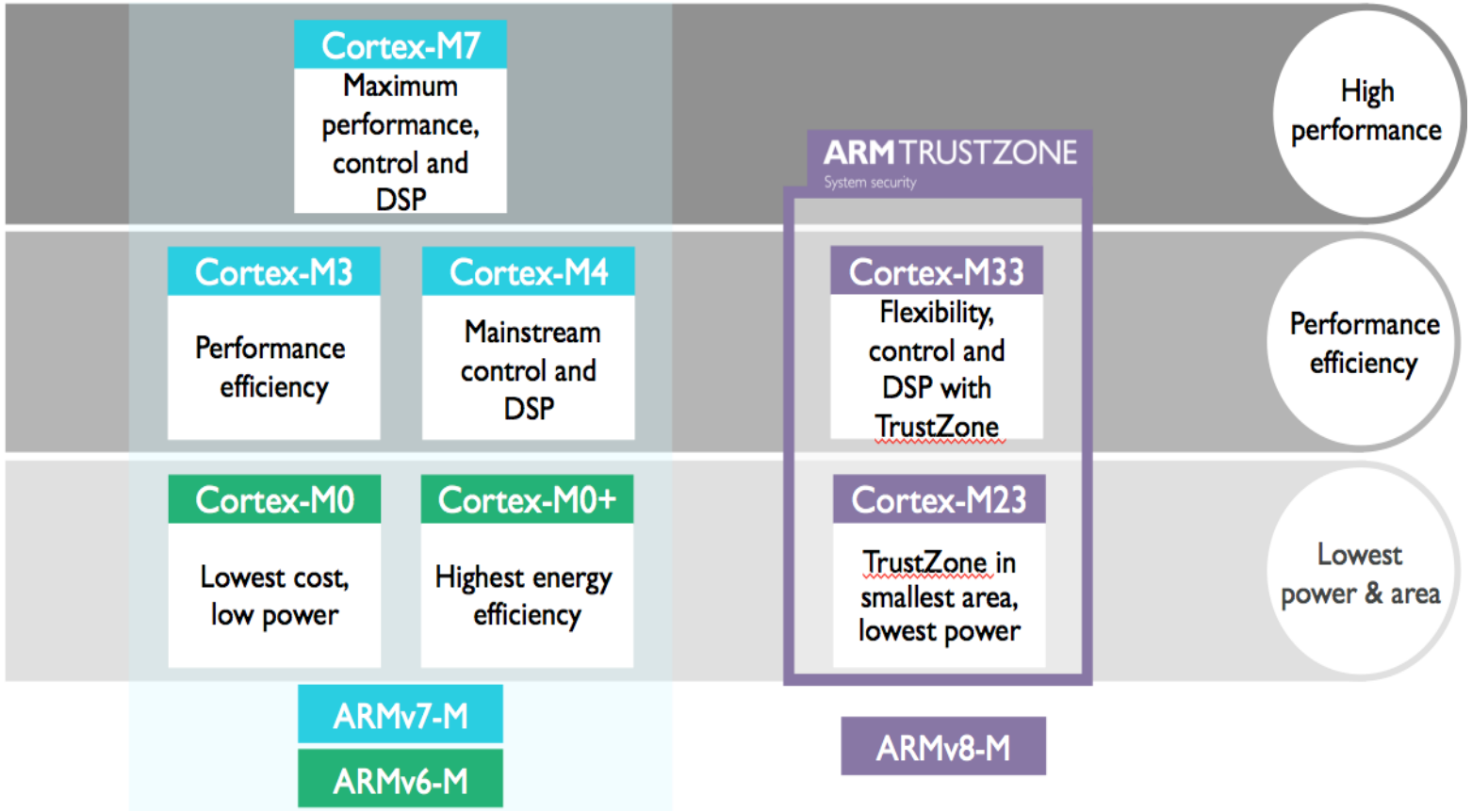
# THE CORTEX-M FAMILY




source: community.arm.com

# THE CORTEX-M FAMILY

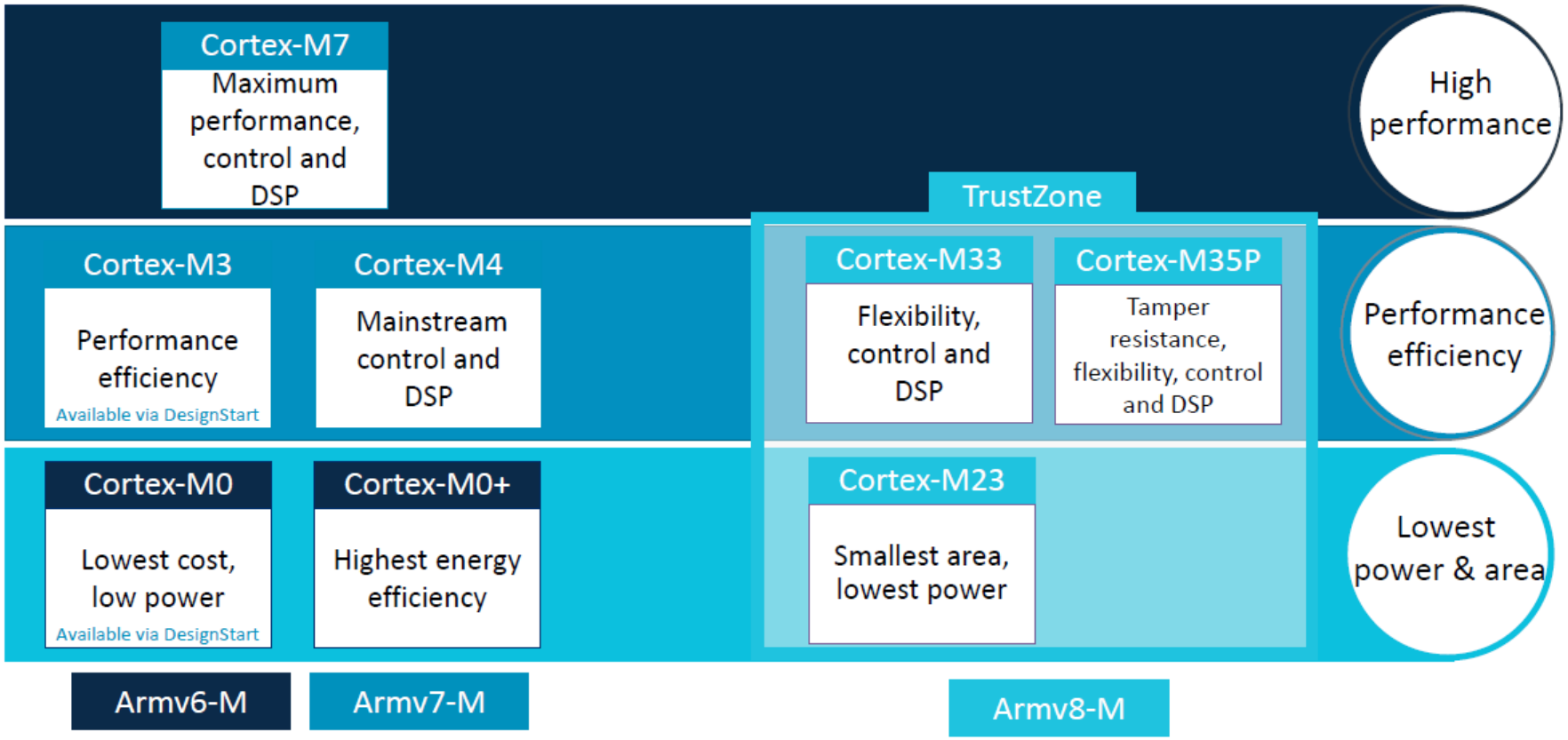
In 2016, the Cortex-M33 and Cortex-M23 were added to the Cortex-M family.




source: community.arm.com

# THE CORTEX-M FAMILY

And in 2018, the Cortex-M35P was added to the Cortex-M family.




source: arm.com

| Processor | Descriptions |
|---|---|
| Cortex-M0 | A very small processor (starting from 12K gates) for low cost, ultra low power microcontrollers and deeply embedded applications |
| Cortex-M0+ | The most energy-efficient processor for small embedded system. Similar size and programmer's model to the Cortex-M0 processor, but with additional features like single cycle I/O interface and vector table relocations |
| Cortex-M1 | A small processor design optimized for FPGA designs and provides Tightly Coupled Memory (TCM) implementation using memory blocks on the FPGAs. Same instruction set as the Cortex-M0 |
| Cortex-M3 | A small but powerful embedded processor for low-power microcontrollers that has a rich instruction set to enable it to handle complex tasks quicker. It has a hardware divider and Multiply-Accumulate (MAC) instructions. In addition, it also has comprehensive debug and trace features to enable software developers to develop their applications quicker |
| Cortex-M4 | It provides all the features on the Cortex-M3, with additional instructions target at Digital Signal Processing (DSP) tasks, such as Single Instruction Multiple Data (SIMD) and faster single cycle MAC operations. In addition, it also have an optional single precision floating point unit that support IEEE 754 floating point standard |
| Cortex-M7 | High-performance processor for high-end microcontrollers and processing intensive applications. It has all the ISA features available in Cortex-M4, with additional support for double-precision floating point, as well as additional memory features like cache and Tightly Coupled Memory (TCM) |
| Cortex-M23 | A small processor for ultra-low power and low cost designs, similar to the Cortex-M0+ processor, but with various enhancements in instruction set and system-level features. It also supports the TrustZone security extension. |
| Cortex-M33 | A mainstream processor design, similar to previous Cortex-M3 and Cortex-M4 processors, but with much better flexibility in system design, better energy efficiency and higher performance. It also supports the TrustZone security extension. |

source: ARM
ARM Cortex-M for Beginners - Yiu

# THE CORTEX-M FAMILY – COMPARISON

## Instruction Set Comparison
*(non-exhaustive list)*

| | Cortex-M0 | Cortex-M0+ | Cortex-M3 | Cortex-M4 | Cortex-M7 |
|---|---|---|---|---|---|
| Single-cycle Multiply | Yes (32bit product) | Yes (32bit product) | Yes (32/64bit product) | Yes (32/64bit product) | Yes (32/64bit product) |
| Hardware divide | - | - | Yes | Yes | Yes |
| Advanced instructions memory accesses (Performance) | - | - | Yes | Yes | Yes |
| Table branch, compare & branch, conditional exec (Performance) | - | - | Yes | Yes | Yes |
| Exclusive access (Performance) | - | - | Yes | Yes | Yes |
| SIMD DSP (DSP Performance) | - | - | - | Yes | Yes |
| Saturation arithmetic (DSP Performance) | - | - | Adjust only | Full set | Full set |
| Float pointing (optional) (DSP Performance) | - | - | - | SP only | SP and DP |

source: ARM
How to Choose Your ARM Cortex-M Processor – Tim Menasveta

# THE CORTEX-M FAMILY – COMPARISON

## System Features

| | Cortex-M0 | Cortex-M0+ | Cortex-M3/M4 | Cortex-M7 |
|---|---|---|---|---|
| Sleep modes | Yes | Yes | Yes | Yes |
| WIC, SRPG support | Yes | Yes | Yes | Yes |
| OS support (SVC, PendSV) | Yes | Yes | Yes | Yes |
| MPU + Unprivileged | - | Optional (0 or 8 regions) | Optional (0 or 8 regions) | Optional (0, 8, or 16 regions) |
| Fault/Exception Handler | HardFault | HardFault | HardFault+3 configurable | HardFault+3 configurable |
| Fault Status Registers | - | - | Yes | Yes |
| Bit Band | - | - | Yes (Optional) | Yes (Optional) |
| Bus interface | AHB Lite | AHB Lite | AHB Lite, APB | AXI4, AHB Lite, APB, TCM |
| L1 cache | - | - | - | Up to 64kB (I & D each) |
| TCM (Tighly Coupled Memory) | - | - | - | Up to 1MB (I & D each) |

Low Power support

OS support

Advanced OS Handling

Fault handling

interface

source: ARM
How to Choose Your ARM Cortex-M Processor – Tim Menasveta

# THE CORTEX-M FAMILY – COMPARISON

## Cortex-M Performance Considerations

- CoreMark® and Dhrystone (per MHz)
  - Results depends on C compiler used
  - Might not be a good indication of the performance for your real world applications

| | Dhrystone (official) | Dhrystone (max opt) | CoreMark* |
|---|---|---|---|
| Cortex-M0 | 0.87 | 1.27 | 2.33 |
| Cortex-M0+ | 0.95 | 1.36 | 2.46 |
| Cortex-M3 | 1.25 | 1.89 | 3.32 |
| Cortex-M4 | 1.25 | 1.95 | 3.40 |
| Cortex-M7 | 2.14 | 3.23 | 5.01 |

* CoreMark data from ARM website and EEMBC.org website

**Cortex-M7** for high processing requirements
- Up to 2x DSP performance of Cortex-M4
- Highest CoreMark

**Cortex-M3/M4** can be faster because of
- Richer instruction set
- Harvard bus architecture
- Write buffer
- Speculative fetch of branch targ

**Cortex-M0+** can be faster for simple I/O control tasks because of
- Shorter pipeline
- Single cycle I/O interface

Actual performance depends on
- System level design
- Memory wait states
- Clock configurations, etc

source: ARM
How to Choose Your ARM Cortex-M Processor – Tim Menasveta

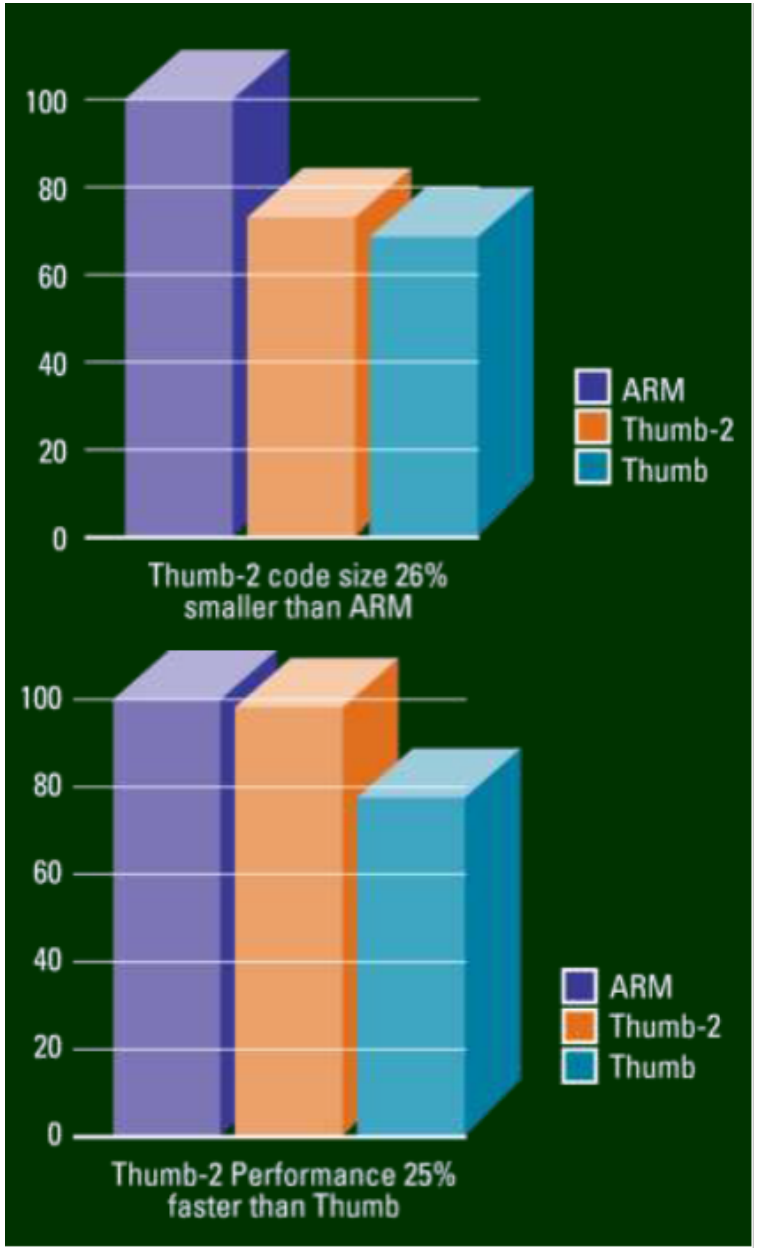
# 3.2 – CORTEX-M FEATURES

Among the distinguishing features of the Cortex-M architecture are:

- 3-stage pipeline (except for Cortex-M7): Fetch, Decode, Execute,
- Harvard architecture (except for Cortex-M0 and M0+),
- Designed for power efficiency (includes an ultra low-power deep sleep),
- Thumb-2 instruction set, combining ARM performance and Thumb code density,
- Interrupt Controller (NVIC) is defined in the architecture; low latency vectored interrupt servicing,
- Interrupt servicing with tail-chaining and late arrival functionalities,
- Bit-banding to provide faster bit operations in memory and memory mapped I/O,
- MPU = Memory Protection Unit,
- Most instructions can be conditional.

# 3.2 – CORTEX-M FEATURES

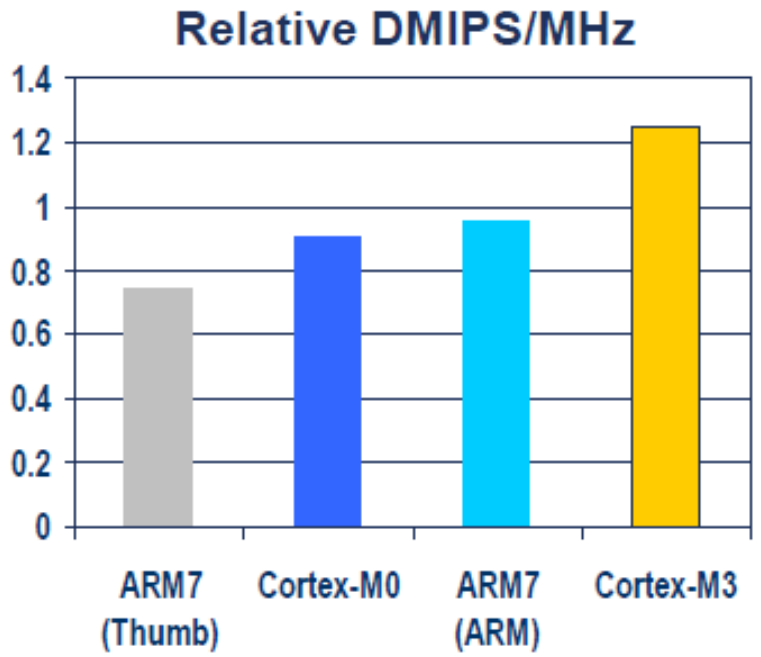
- The ARM7TDMI had two instructions sets:
- ARM - with 32-bit instructions, higher performance and lower code density,
- Thumb - with 16-bit instructions, lower performance and higher code density.
- Cortex-M has a single instruction set: Thumb-2
- It mixes 16-bit and 32-bit instructions. Its code density is similar to Thumb and its performance is similar to ARM.




source: ARM
The ARM Architecture - Joe Bungo

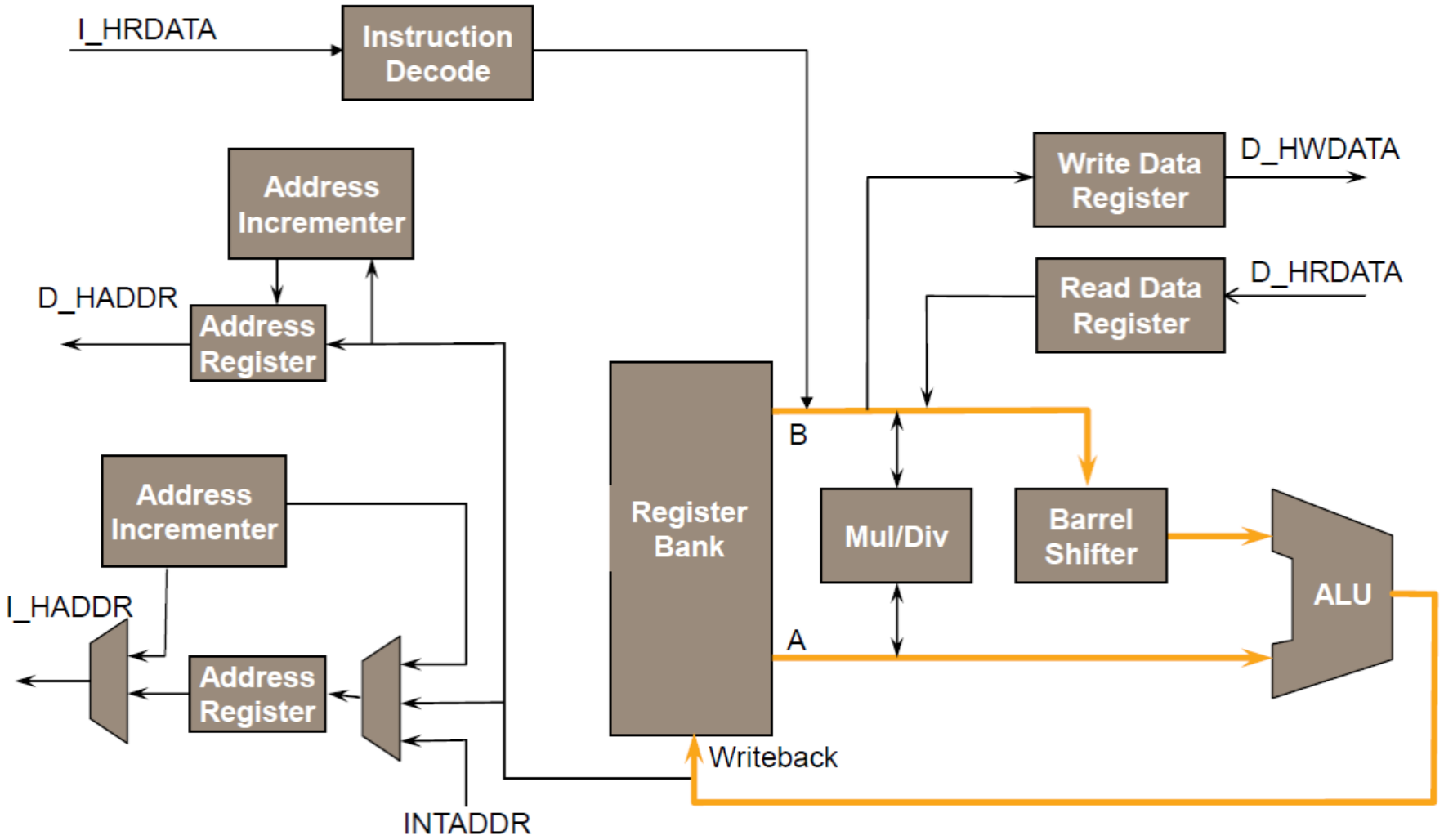
# 3.2 – CORTEX-M FEATURES

| Comparison | Cortex-M0 | Cortex-M3 |
|---|---|---|
| DMIPS/MHz | 0.9 | 1.25 |
| Gate count | 12k | 43k |
| Number interrupts | 1-32 + NMI | 1-240 + NMI |
| Interrupt priorities | 4 | 256 |
| Breakpoints, Watchpoints | 4/2, 2/1 | 8/4, 2/1 |
| MPU, integrated trace option | No | Yes |
| Hardware Divide | No | Yes |



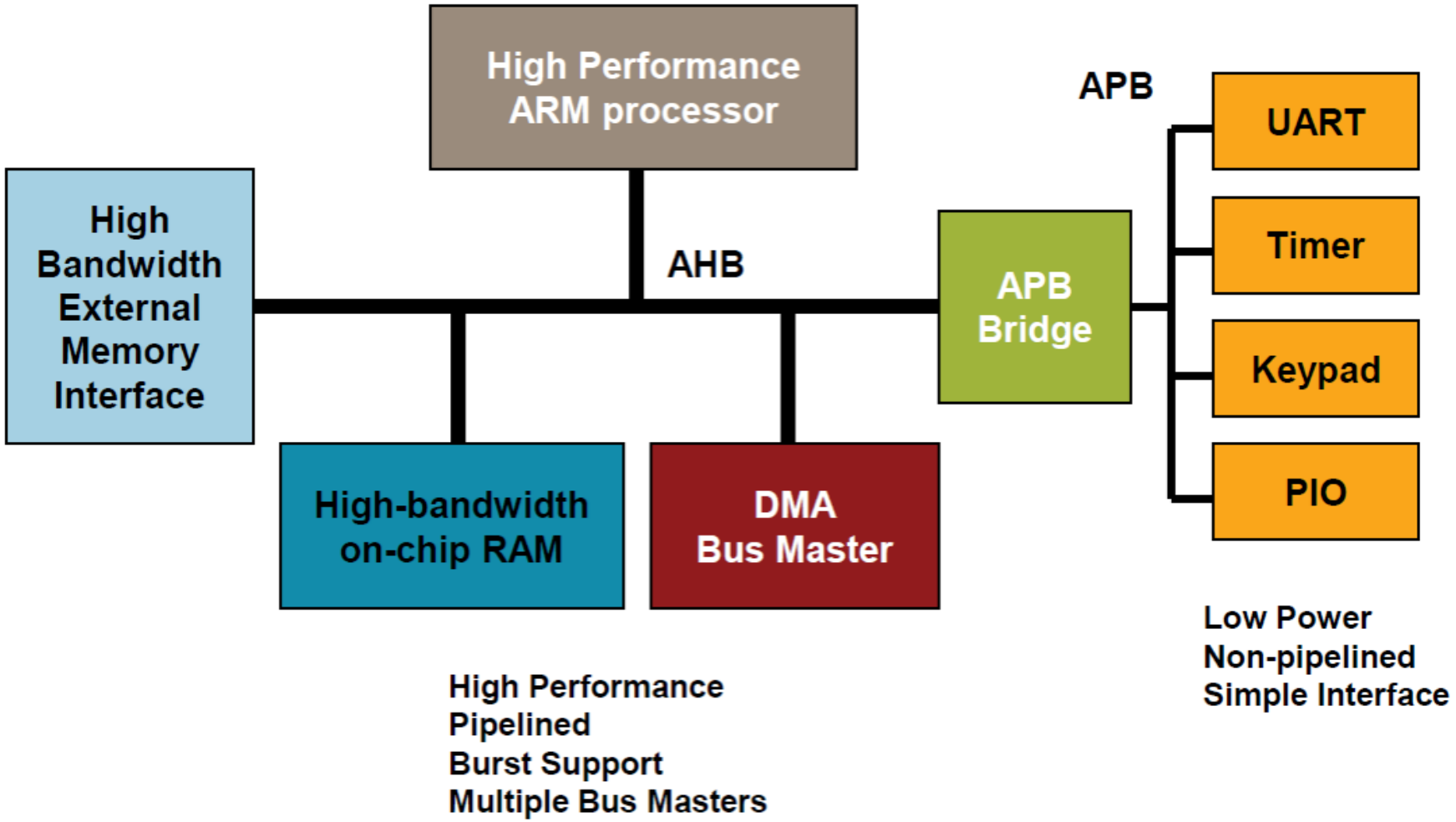

source: ARM
ARM Cortex-M Processor Family

# Cortex-M3 Datapath




source: ARM
The ARM
Architecture - Joe
Bungo

# 3.2 – CORTEX-M FEATURES

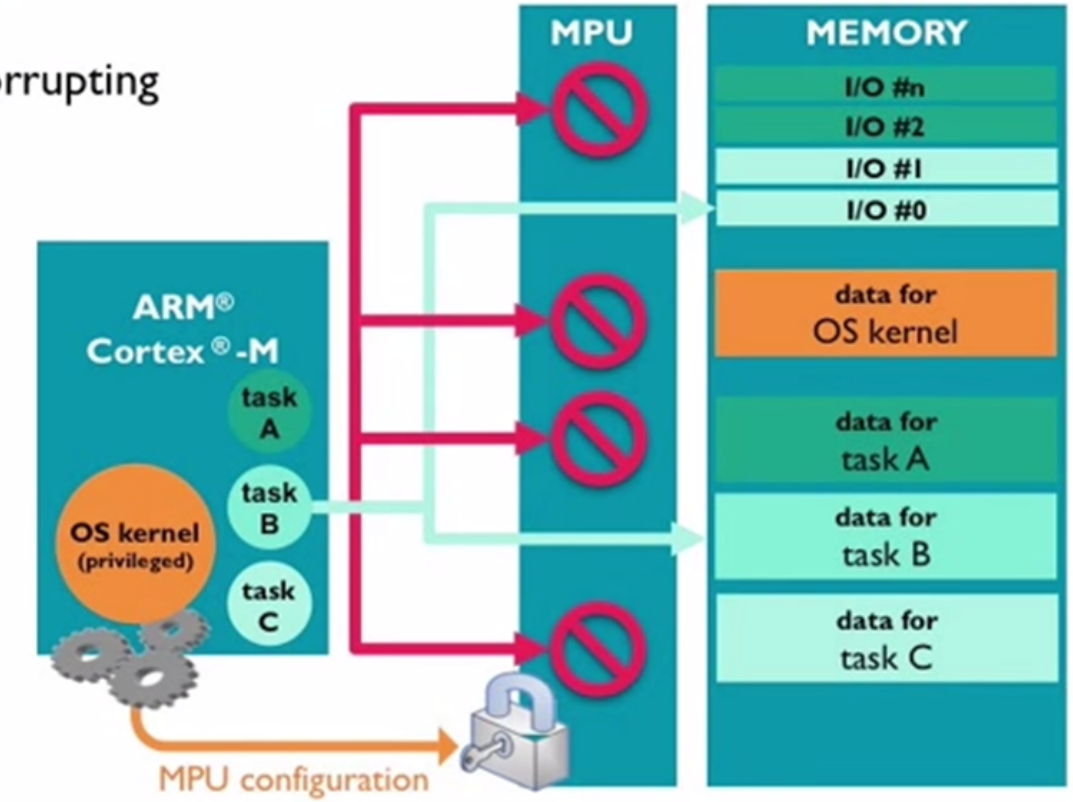



source: ARM The ARM Architecture - Joe Bungo

# 3.2 – CORTEX-M FEATURES

## Memory Protection Unit (MPU)

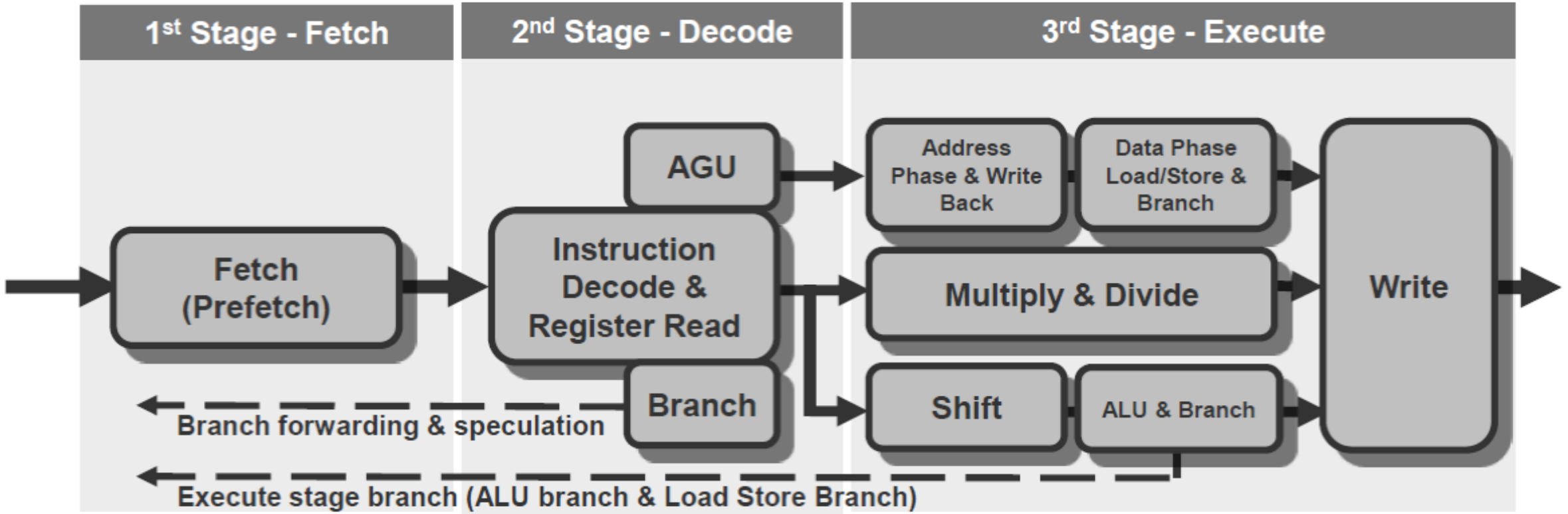
- Prevents application task from corrupting OS or other task data
  - Improves system reliability
- Configurable regions
  - Address
  - Size
  - Memory attributes
  - Access permissions
- Optional in all processors (except Cortex-M0)




source: ARM
How to Choose Your ARM Cortex-M Processor - Tim Menasveta

# 3.2 – CORTEX-M FEATURES

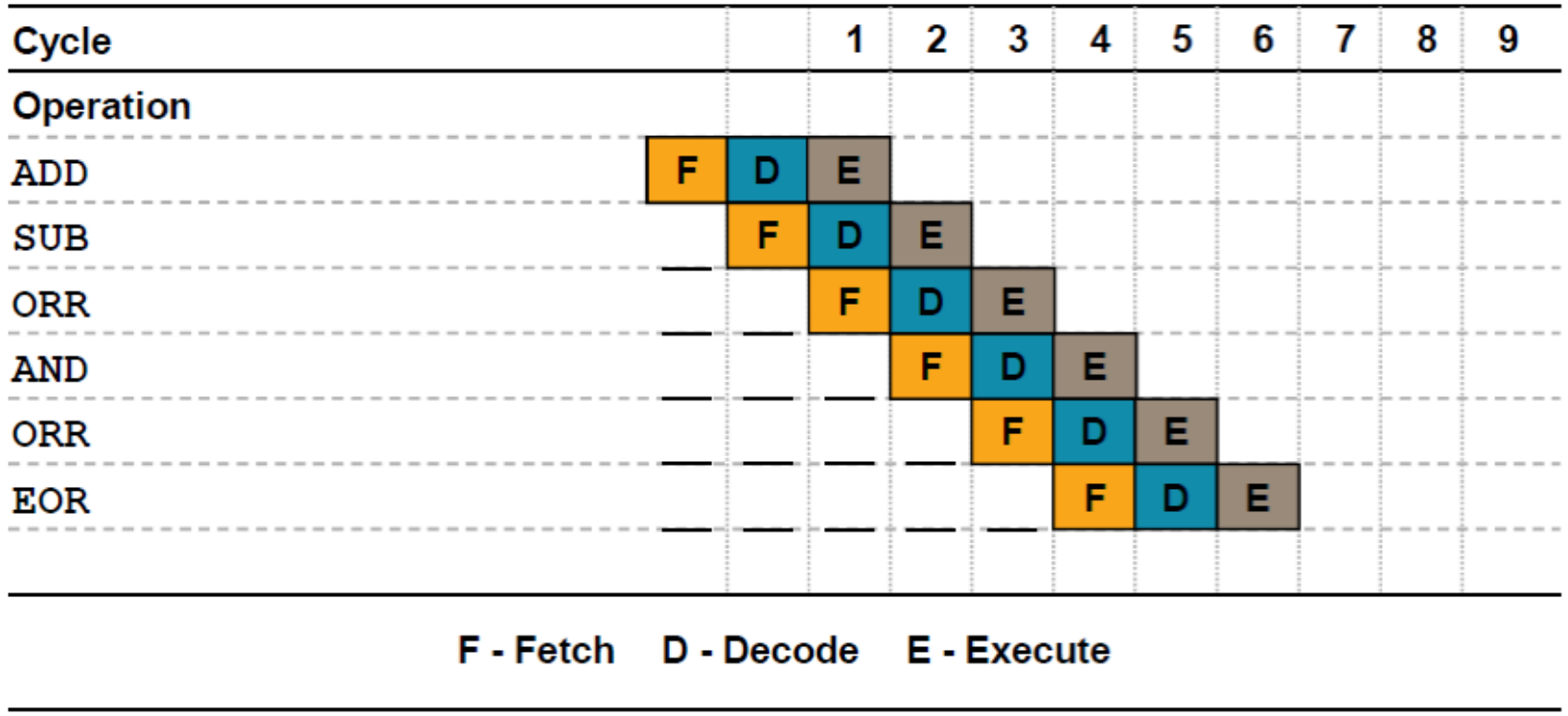
Cortex-M4 Pipeline




source: ARM
The ARM Architecture - Joe Bungo

# 3.2 – CORTEX-M FEATURES

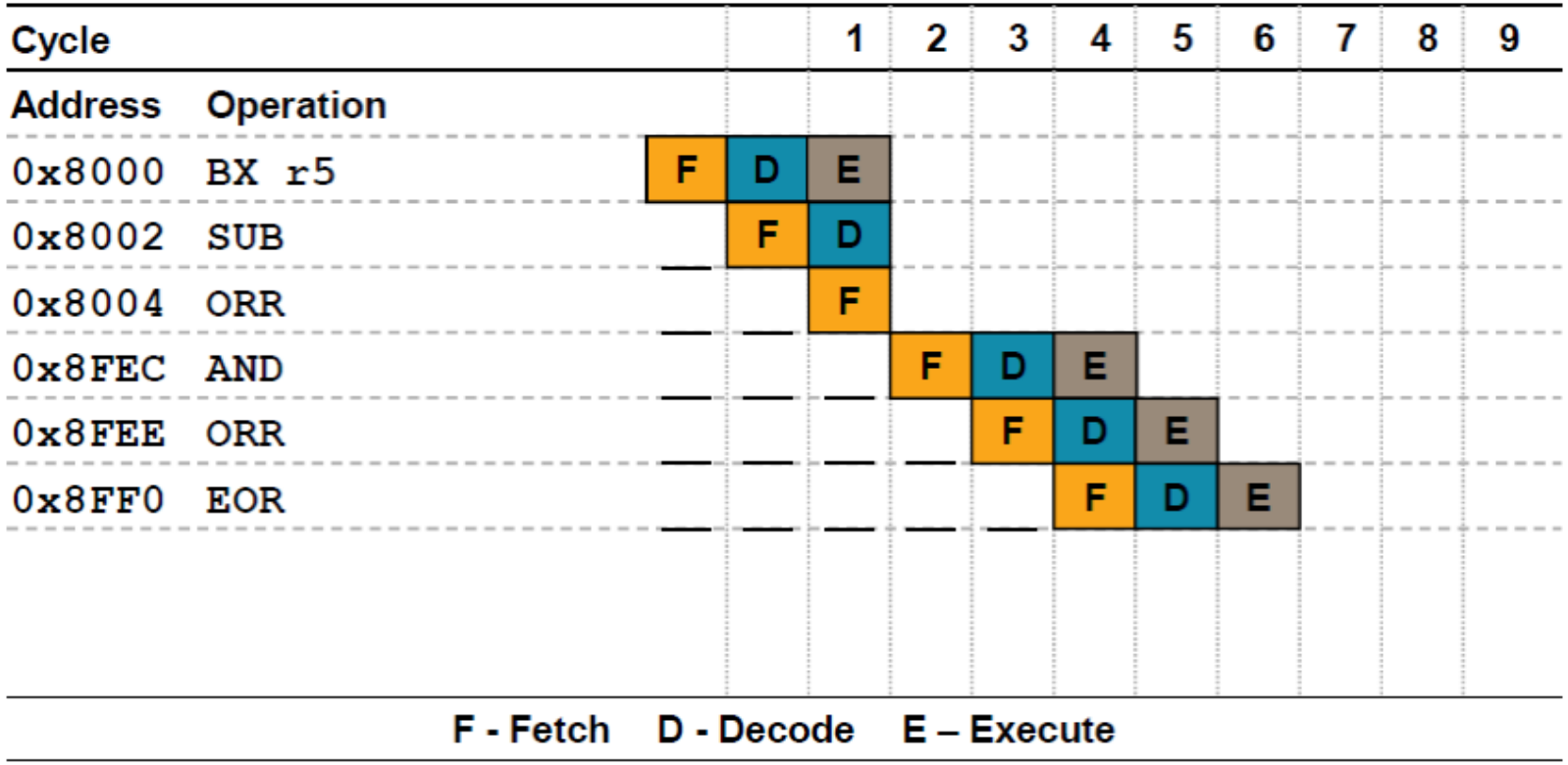
Cortex-M4 Pipeline – example for optimal execution




source: ARM
ARM Cortex-M3 Introduction
- ARM University Relations

# 3.2 – CORTEX-M FEATURES

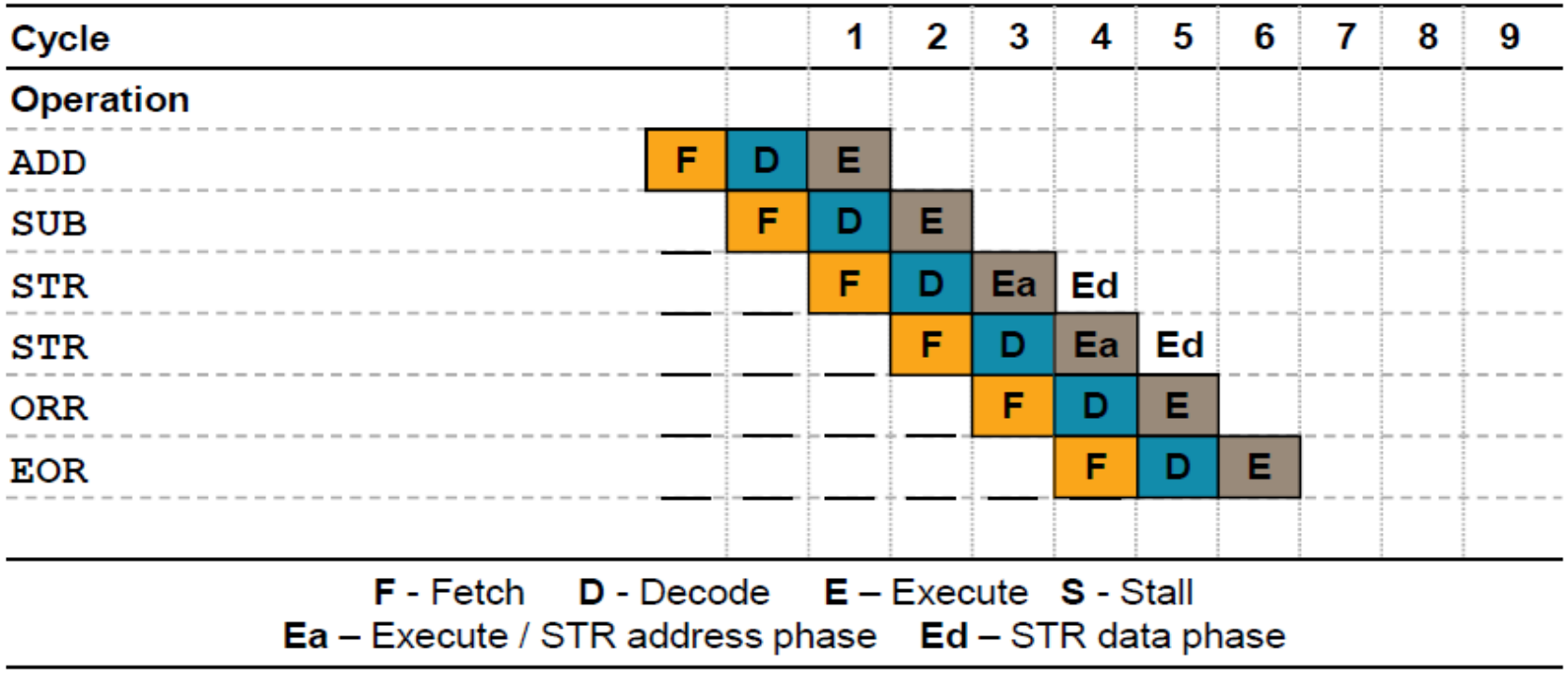
Cortex-M4 Pipeline – pipeline flush due to indirect branch (no forwarding)




source: ARM
ARM Cortex-M3 Introduction
- ARM University Relations

# 3.2 – CORTEX-M FEATURES

Cortex-M4 Pipeline – Harvard architecture allows for concurrent access to code and data memory.




source: ARM
ARM Cortex-M3 Introduction
- ARM University Relations

# 3.3 – CORTEX-M4 INSTRUCTION SET ARCHITECTURE

The Instruction Set Architecture (ISA) presents the Programmer's View of the processor, including:

- Data types
- Processor Modes
- Processor Registers
- Instruction Set
- Memory Accessing
- Exception Processing

# 3.3 – CORTEX-M4 ISA

The Cortex-M4 instruction set can operate on the following data types:

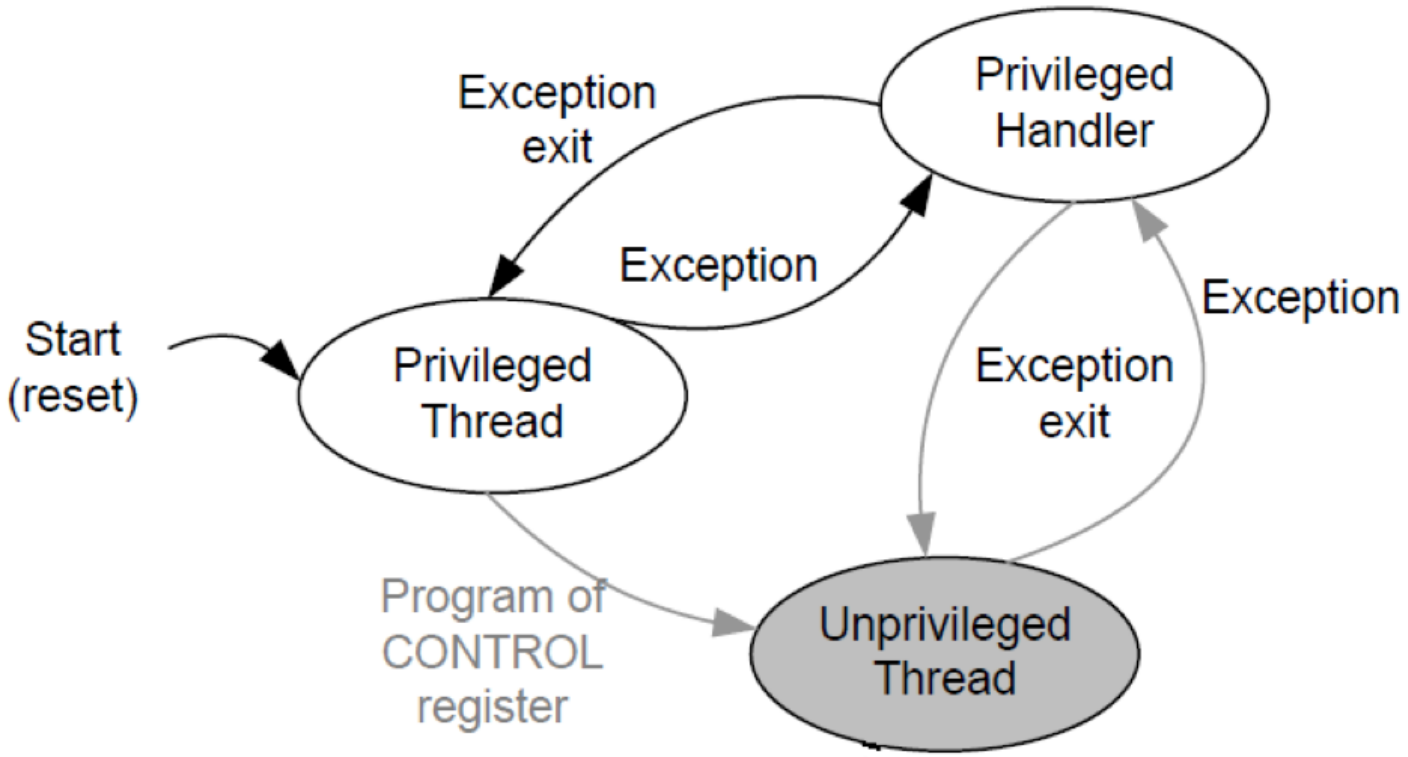
- **bit:** stores a single bit of information (0 or 1).
  Bit banding instructions can set or clear bits in specific memory regions.
- **byte:** 8-bit. Each byte in memory is individually addressable.
- **half-word:** 16-bit. The address of a half-word in memory is the address of its least significant byte.
- **word:** 32-bit. The address of a word in memory is the address of its least significant byte.
- **double-word:** 64-bit. Requires a register pair to be stored, such as R1:R0 (concatenation of R1 and R0 with R1 being the most significant word). The address of a double-word in memory is the address of its least significant byte.

# 3.3 – CORTEX-M4 ISA

The Cortex-M4 processor modes are:

- Privileged Thread -
  mainly for OS execution
- Unprivileged Thread -
  mainly for Application code
- Privileged Handler -
  for exception handling code

Remark: there are two stacks: Main and Process.
The usage of these stacks is related to the processor mode.




source: ARM
ARM Cortex-M for Beginners - Yiu

# 3.3 – CORTEX-M4 ISA

The Cortex-M4 has two stacks:

- Main stack - addressed by MSP (Main Stack Pointer)
  mainly to be used by the Operating System (OS) and exception handlers.
- Process stack - addressed by PSP (Process Stack Pointer)
  mainly to be used by the Application threads.

Only one stack is active at any given time, as selected by the CONTROL register. Register R13 (SP) maps to the active stack pointer, either MSP or PSP.

# 3.3 – CORTEX-M4 ISA

**Table B1-1 Mode, privilege and stack relationship**

| Mode | Privilege | Stack pointer | Typical usage model |
|---|---|---|---|
| Handler | Privileged | Main | Exception handling. |
| Thread | Privileged | Main | Execution of a privileged process or thread using a common stack in a system that only supports privileged access. |
| | | Process | Execution of a privileged process or thread using a stack reserved for that process or thread in a system that only supports privileged access, or that supports a mix of privileged and unprivileged threads. |
| Thread | Unprivileged | Main | Execution of an unprivileged process or thread using a common stack in a system that supports privileged and unprivileged access. |
| | | Process | Execution of an unprivileged process or thread using a stack reserved for that process or thread in a system that supports privileged and unprivileged access. |

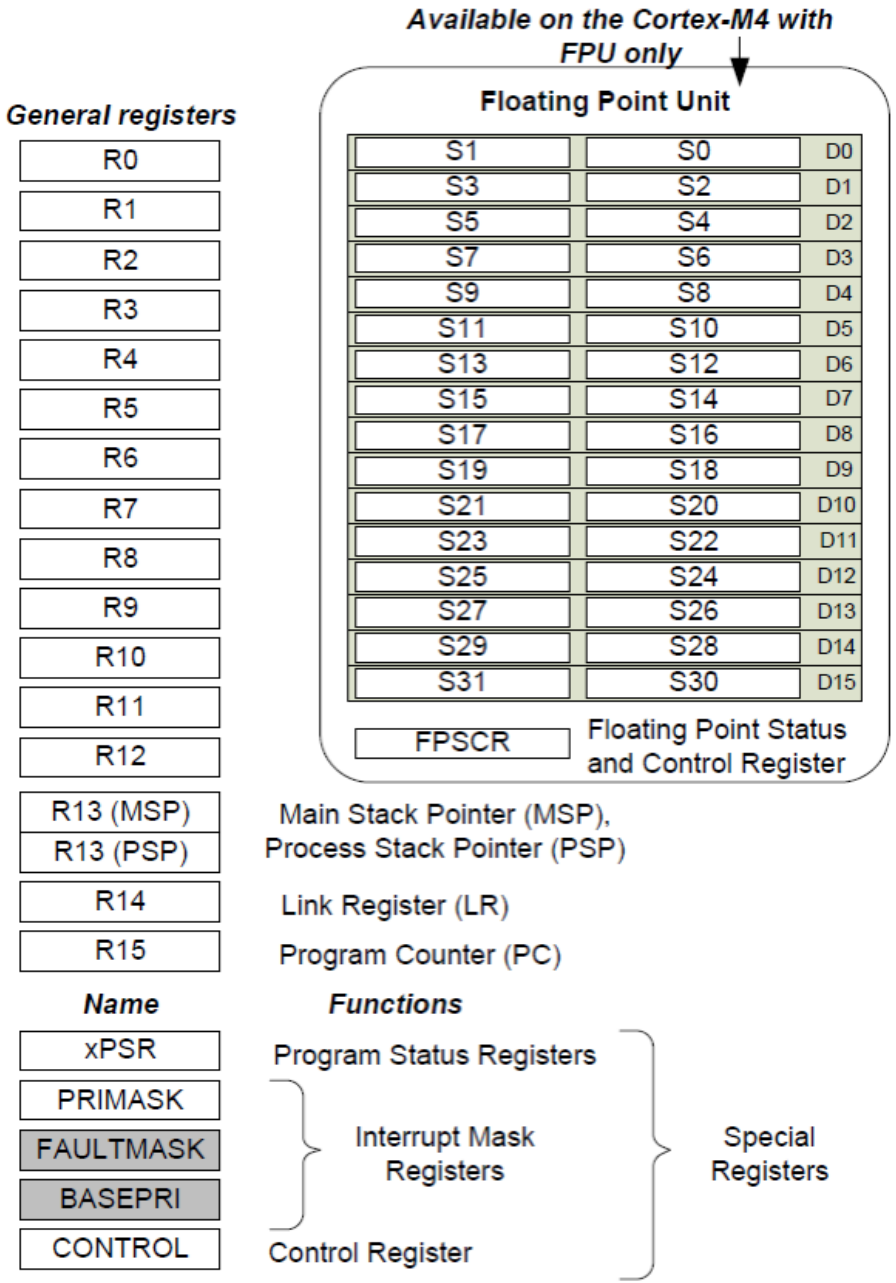
source: ARM ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – REGISTERS

The register set of the Cortex-M4 consists of:

- General Purpose Registers (R0-R15)
- Floating Point Registers (S0-S31)
- Special Registers (xPSR, PRIMASK, FAULTMASK, BASEPRI, CONTROL)

All registers are 32-bit wide. Not all bits of the Special Registers are implemented.




source: ARM
ARM Cortex-M for Beginners - Yiu

# 3.3 – CORTEX-M4 ISA – REGISTERS

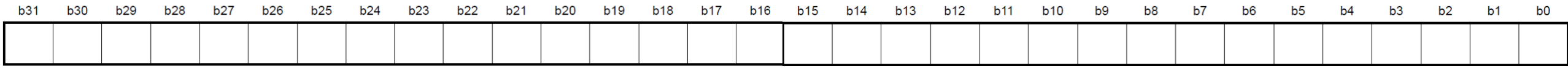
General Purpose Registers:

- 32-bit wide
- Low registers: R0 .. R7
- High registers: R8 .. R15
- Special usage:

  R13 or SP is the Stack Pointer - points to the top of the stack

  R14 or LR is the Link Register - stores the procedure return address

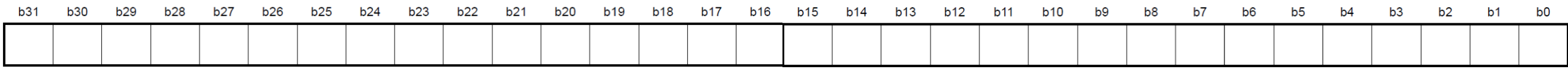
  R15 or PC is the Program Counter - stores the address of the next instruction fetch



# 3.3 – CORTEX-M4 ISA – REGISTERS

General Purpose Registers:

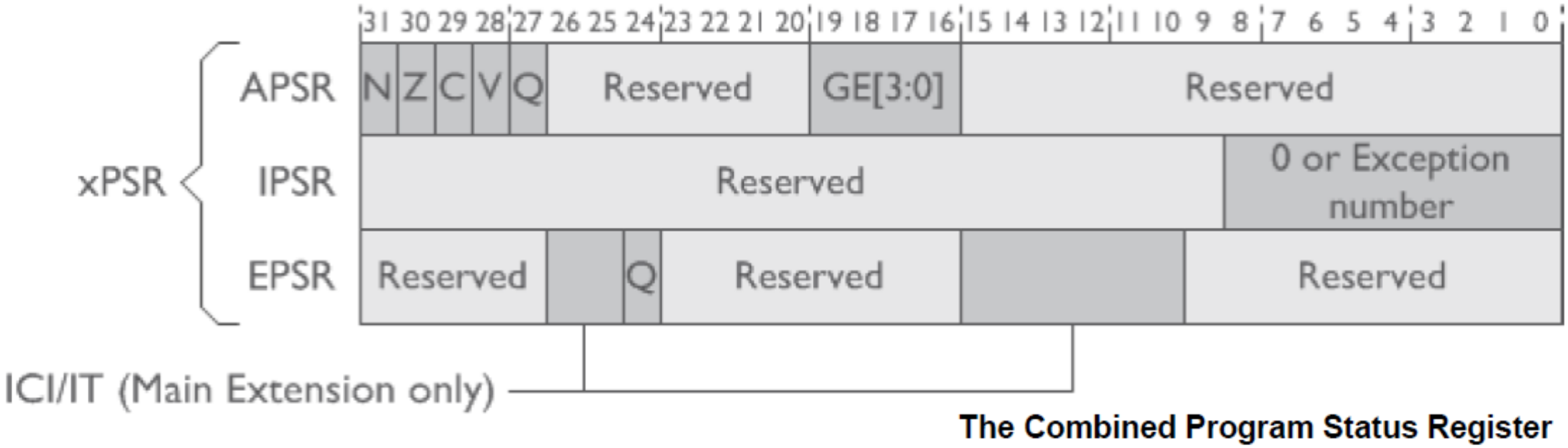
- the general-purpose registers are accessible by most instructions;
- b0 is the Least Significant Bit (LSb) and b31 is the Most Significant Bit (MSb);
- a register can hold an unsigned integer with values from 0 to 4,294,967,295
  or a signed integer with values from -2,147,483,648 to 2,147,483,647;
- when using hexadecimal notation, a 32-bit register holds 8 hex digits. E.g. 0x1234 5678;
- Cortex-M architecture uses 32-bit memory addresses, hence, a single general-purpose register can hold a memory address.



# 3.3 – CORTEX-M4 ISA – REGISTERS

Special Purpose Registers - XPSR

- The three registers APSR, IPSR and EPSR can be accessed individually or combined as XPSR.

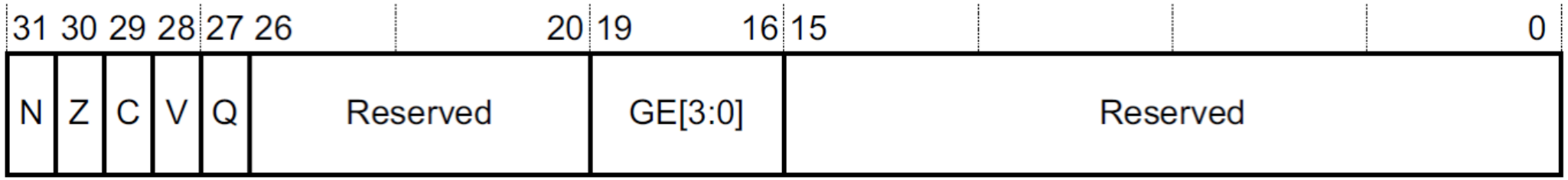



**The Combined Program Status Register**

source: ARM
ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – APSR

## APSR = Application Program Status Register




| Flag | bit | Description |
|---|---|---|
| N | 31 | Negative. For two's complement results this bit is set to indicate that the result is negative. |
| Z | 30 | Zero. This bit is set when the result is zero. After a comparison, this bit is set to indicate that the compared values are equal. |
| C | 29 | Carry. Set to indicate that the result of an unsigned addition produced overflow. Also set to indicate that un unsigned subtraction underflowed. |
| V | 28 | Overflow. Set to indicate overflow in a signed arithmetic operation. |
| Q | 27 | Saturation bit. For DSP extension instructions. |
| GE | 19:16 | Greater than or Equal. For SIMD instructions. |

source: ARM
ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – IPSR

**IPSR = Interrupt Program Status Register**


| Field | bit | Description |
|---|---|---|
| Exception Number | 8 : 0 | This field identifies the exception that is currently active (being serviced). The value 0 indicates that no exception is currently active.<br>If this value is non-zero the processor is in Handler mode.<br>If this value is zero the processor is in Thread mode. |

source: ARM
ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – EPSR

**EPSR = Execution Program Status Register**


| Field | bit | Description |
|---|---|---|
| T | 24 | Thumb. This bit is set to indicate that the instruction set in use is Thumb.<br>On Cortex-M this bit must be set all the time or an exception occurs.<br>On ARM7TDMI, Cortex-R and Cortex-A this bit is 0 when the ARM instruction set is in use. |
| ICI/IT | 26:25<br>15:10 | ICI - used for a interrupted exception-continuable multi-cycle load or store.<br>IT - provide context information for instructions in an IT block. |

source: ARM
ARMv7-M Architecture Reference Manual

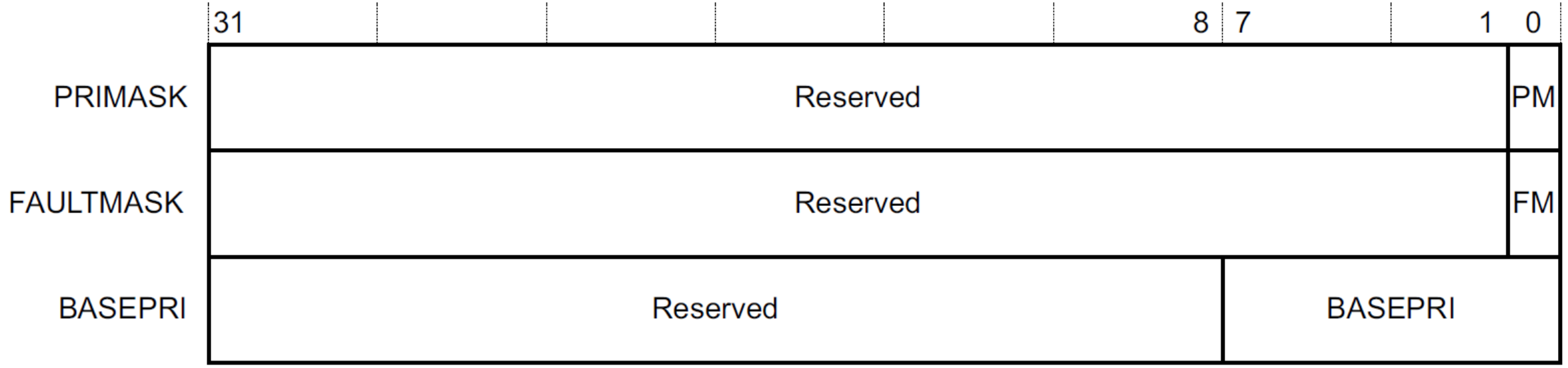
# 3.3 – CORTEX-M4 ISA – XPSR

Mnemonics used to combine the XPSR component registers:

| Mnemonic | Registers accessed |
|---|---|
| IAPSR | IPSR and APSR |
| EAPSR | EPSR and APSR |
| XPSR | All three xPSR registers |
| IEPSR | IPSR and EPSR |

source: ARM
ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – SPECIAL REGISTERS




| Field | bit | Description |
|---|---|---|
| PM | PRIMASK[0] | Set to mask exceptions with configurable priority (priority 0 and lower). Reset to unmask. |
| FM | FAULTMASK[0] | Set to mask the HardFault exceptions and the configurable priorities (prio -1 and lower). |
| BASE PRI | BASEPRI[7:0] | Changes the priority level required for exception preemption. Affects only the currently executing code with lower priority than BASEPRI. |

source: ARM
ARMv7-M Architecture Reference Manual

# 3.3 – CORTEX-M4 ISA – CONTROL REGISTER

CONTROL Register


| Field | bit | Description |
|---|---|---|
| nPRIV | 0 | When the processor is in Thread mode. 0 = privileged mode; 1 = unprivileged mode. |
| SPSEL | 1 | Stack selection. 0 = use MSP (Main Stack); 1 = use PSP (Process Stack).<br>In Handler mode this bit is always 0. |
| FPCA | 2 | Implemented only when floating point is available.<br>0 = do not save floating point registers on exception;<br>1 = save floating point context on exception. |

source: ARM
ARMv7-M Architecture Reference Manual

Renesas.com