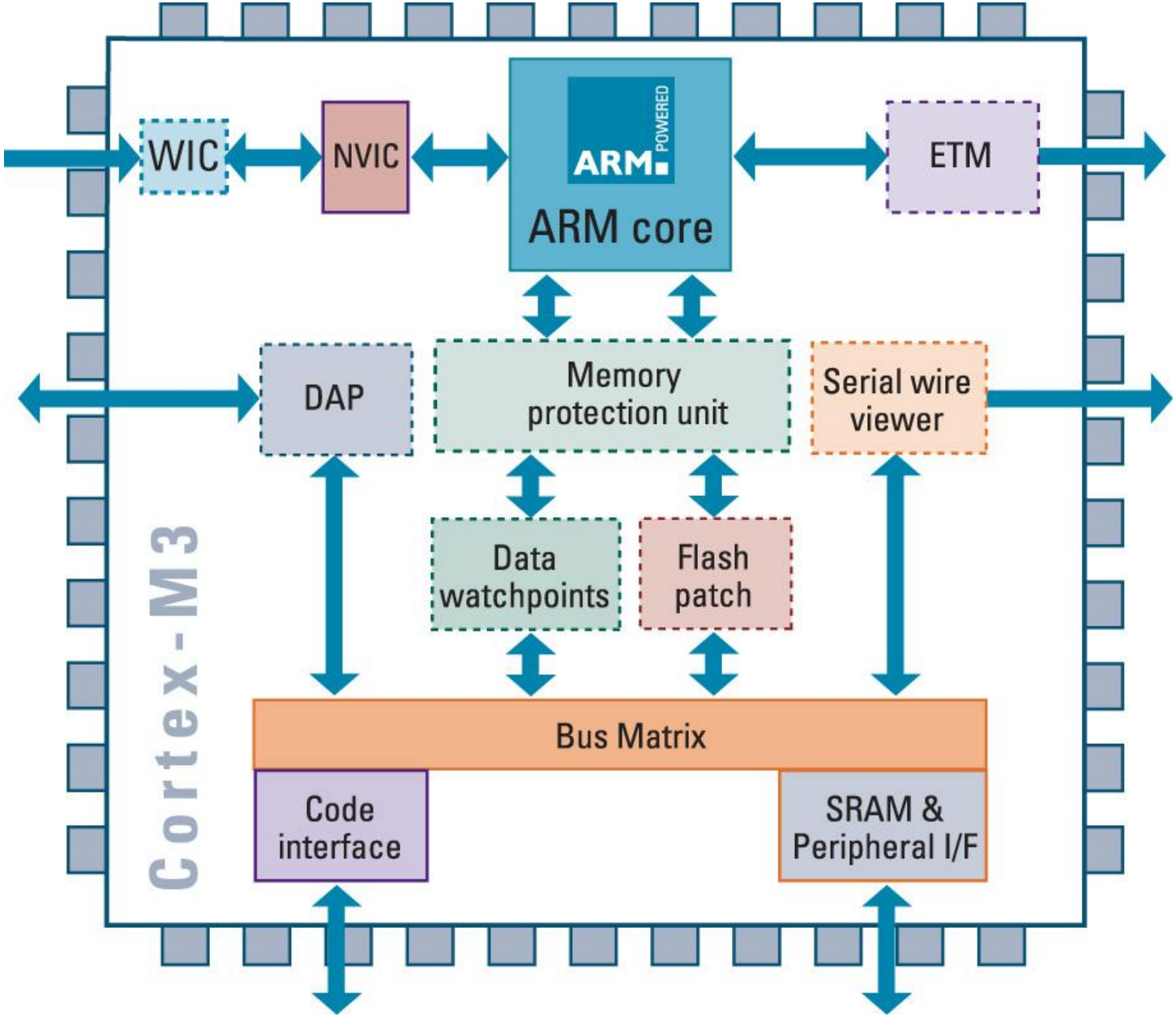
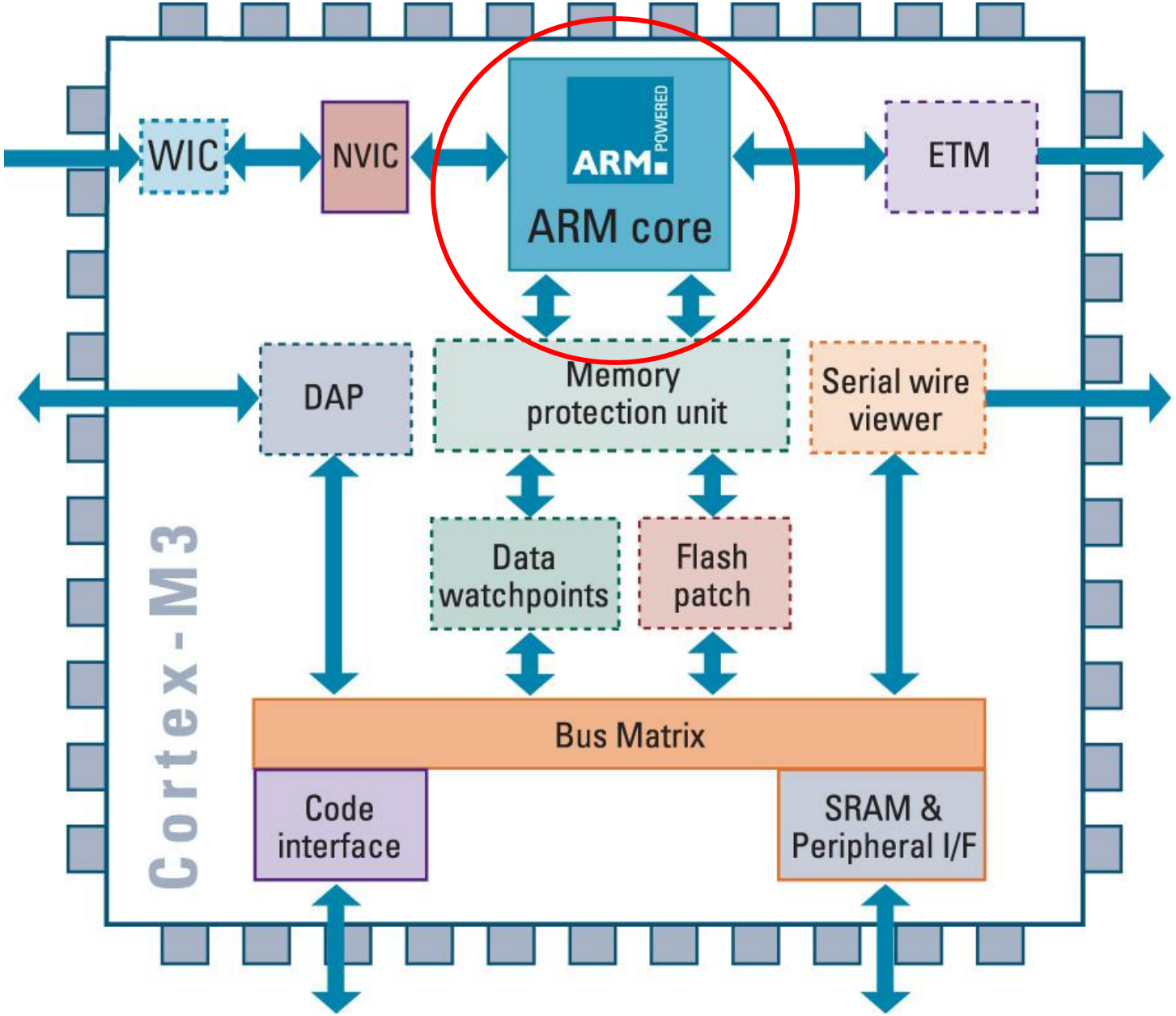
Cortex Familie – Eine Architektur für alle Applikationen

1. Vom M0 zum M4
2. Mögliche Applikationen
3. Cortex-M4 für sicherheitskritische Anwendungen
4. Cortex-M3 Architektur
5. Cortex Mikrocontroller Software Interface Standard CMSIS





Cortex™-M Prozessor-Familie

Cortex™-M4: Thumb®-2 16-/32-Bit Befehlssatz erweitert durch:

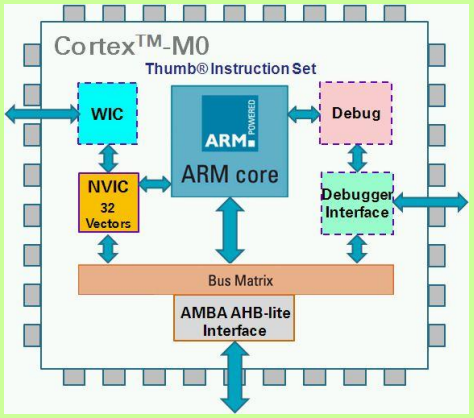
DSP Befehle + Floating Point Befehle (Floating Point Unit)

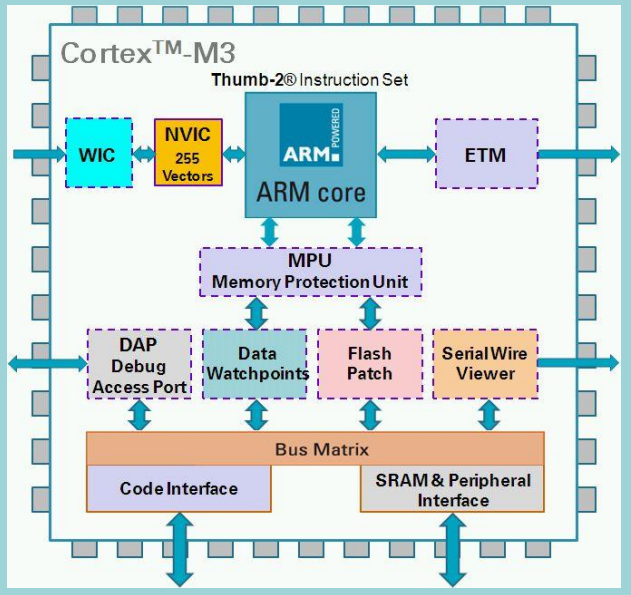
Cortex™-M3: Thumb®-2 16-/32-Bit Befehlssatz

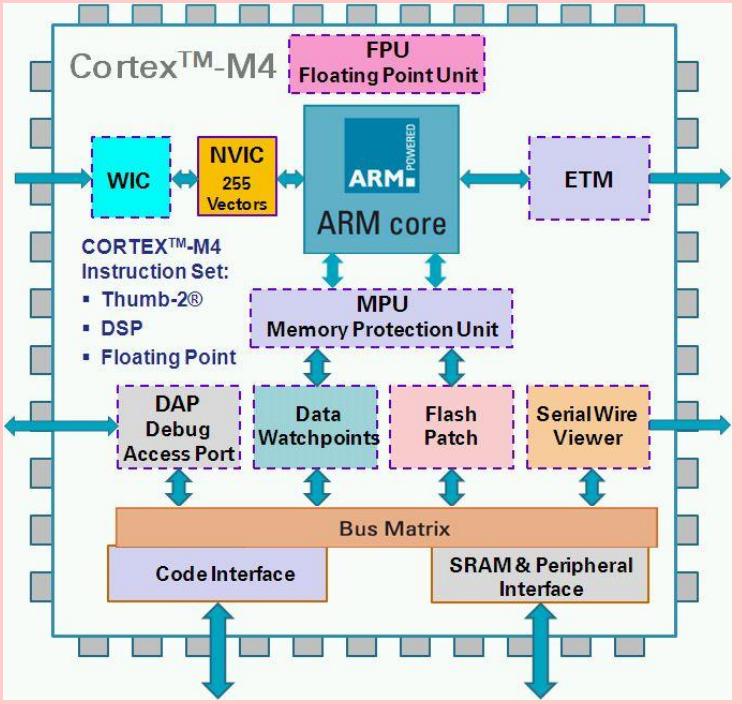
MPU, separater Code-/Data-Bus, ETM, Code-Patch

Cortex™-M0/M1

Thumb® Befehlssatz







Cortex™-M Prozessorfamilie - Applikationen

Cortex™-M4 Core:

- DSP-Filterarithmetik
- Hohe Arithmetik-Performance
- Floating Point Arithmetik

Applikationen:

- Robotik-Applikationen, Antriebssteuerungen und Bildverarbeitung
- Mehrphasen-Motoransteuerungen
- Hard Disc Steuerung
- Modem- und Gateway-Applikationen

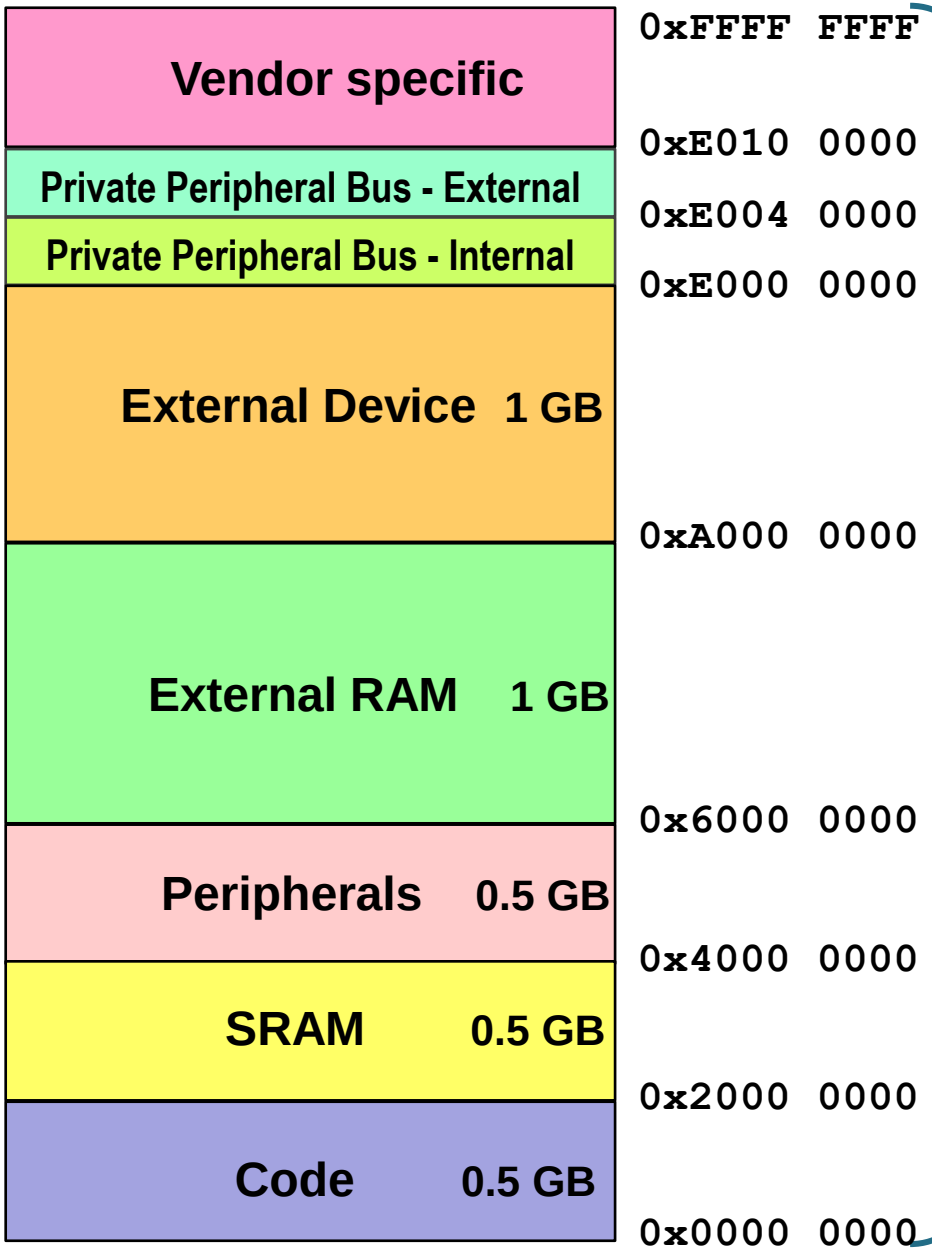
Cortex™-M0/M1 Applikationen:

- **Medizingeräte**
- **Motorsteuerungen**
- **Power Control**
- **Messgeräte-Technik**
- **Beleuchtungstechnik**
- **Spielekonsolen**

Cortex™-M3 Core - Applikationen:

- Applikationen wie Cortex™-M0
- Haushaltsgeräte-Steuerungen
- Automobil- und Industriesteuerungen
- Applikationen für Telekommunikationsgeräte
- Sicherheitstechnische Applikationen

4 Gbyte Adressraum



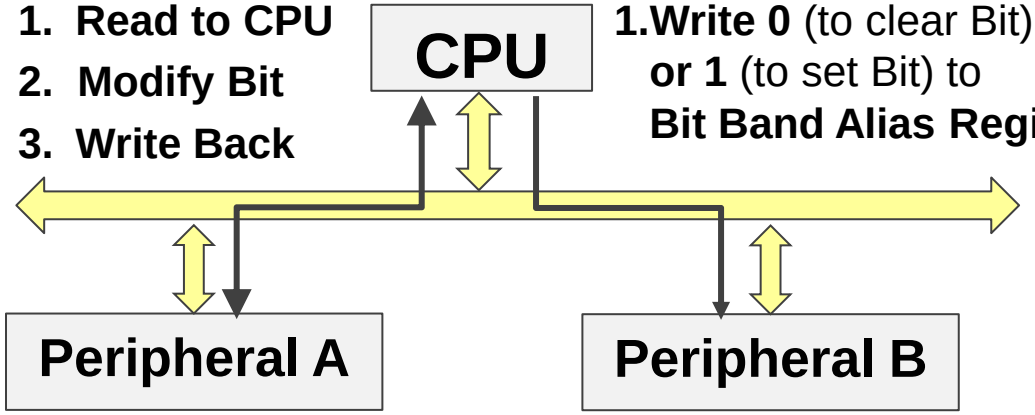
Einheitlicher Adressraum für die gesamte CORTEX™ Bausteinfamilie

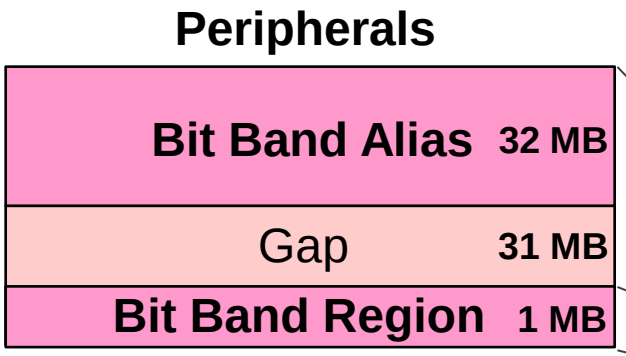
Herkömmliche Änderung eines Bits

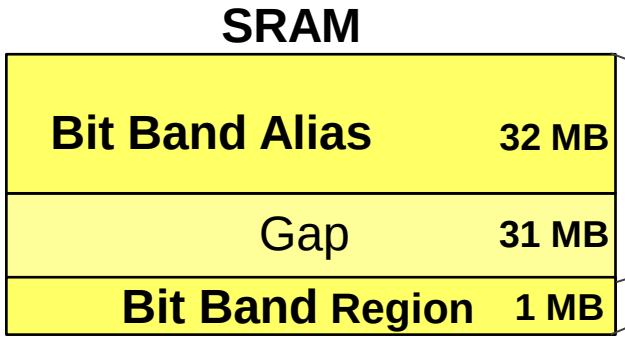
1. Read to CPU
2. Modify Bit
3. Write Back

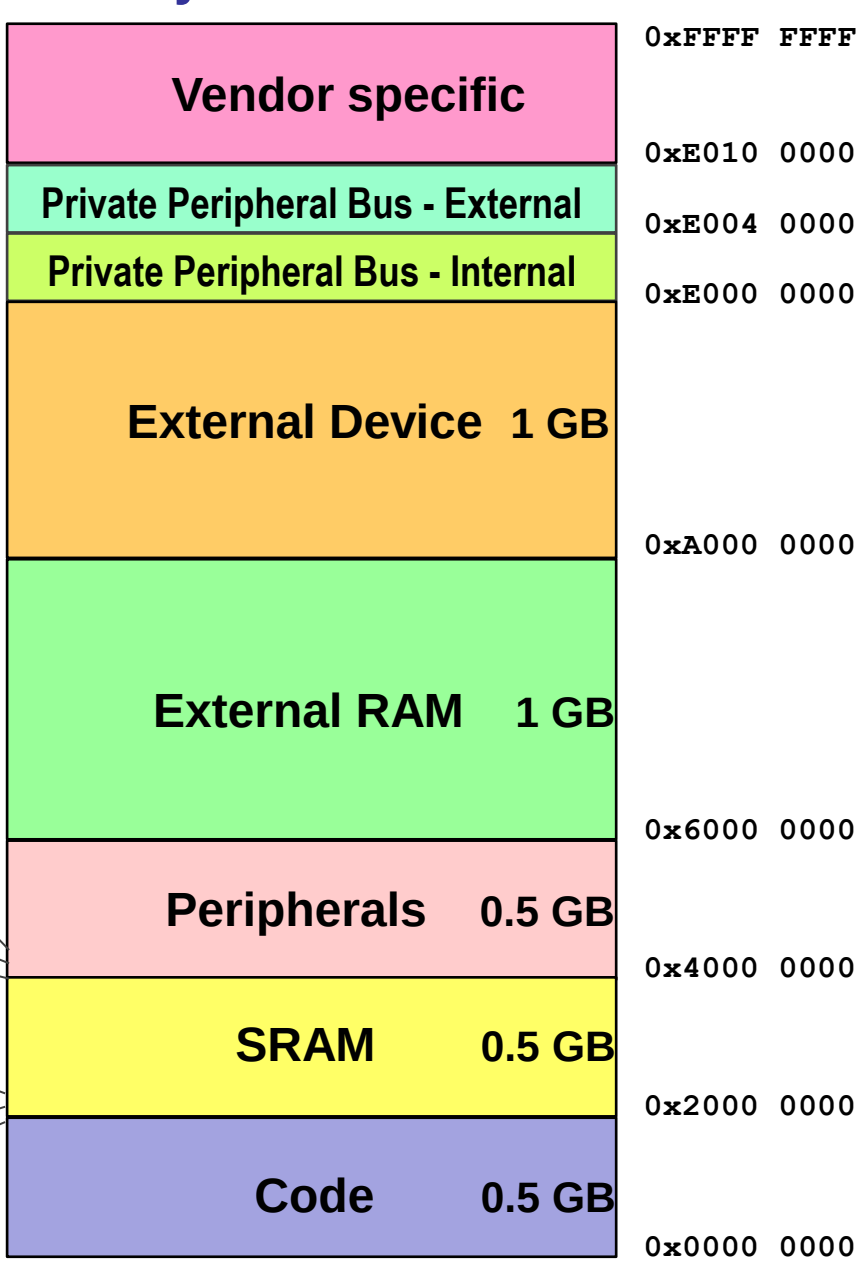
Bit-Änderung über Bit Banding

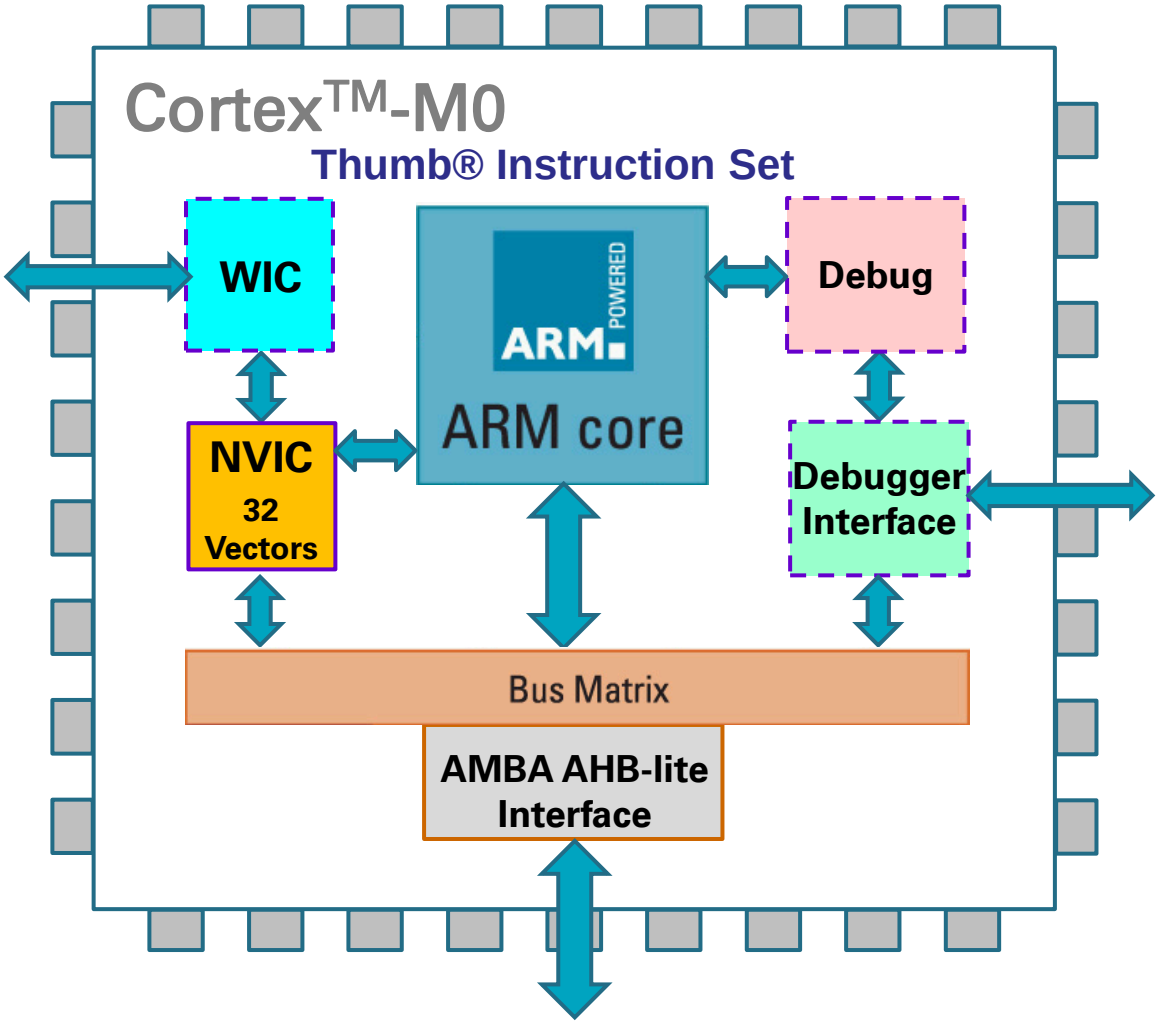
1. **Write 0** (to clear Bit)
   or 1 (to set Bit) to
   Bit Band Alias Region

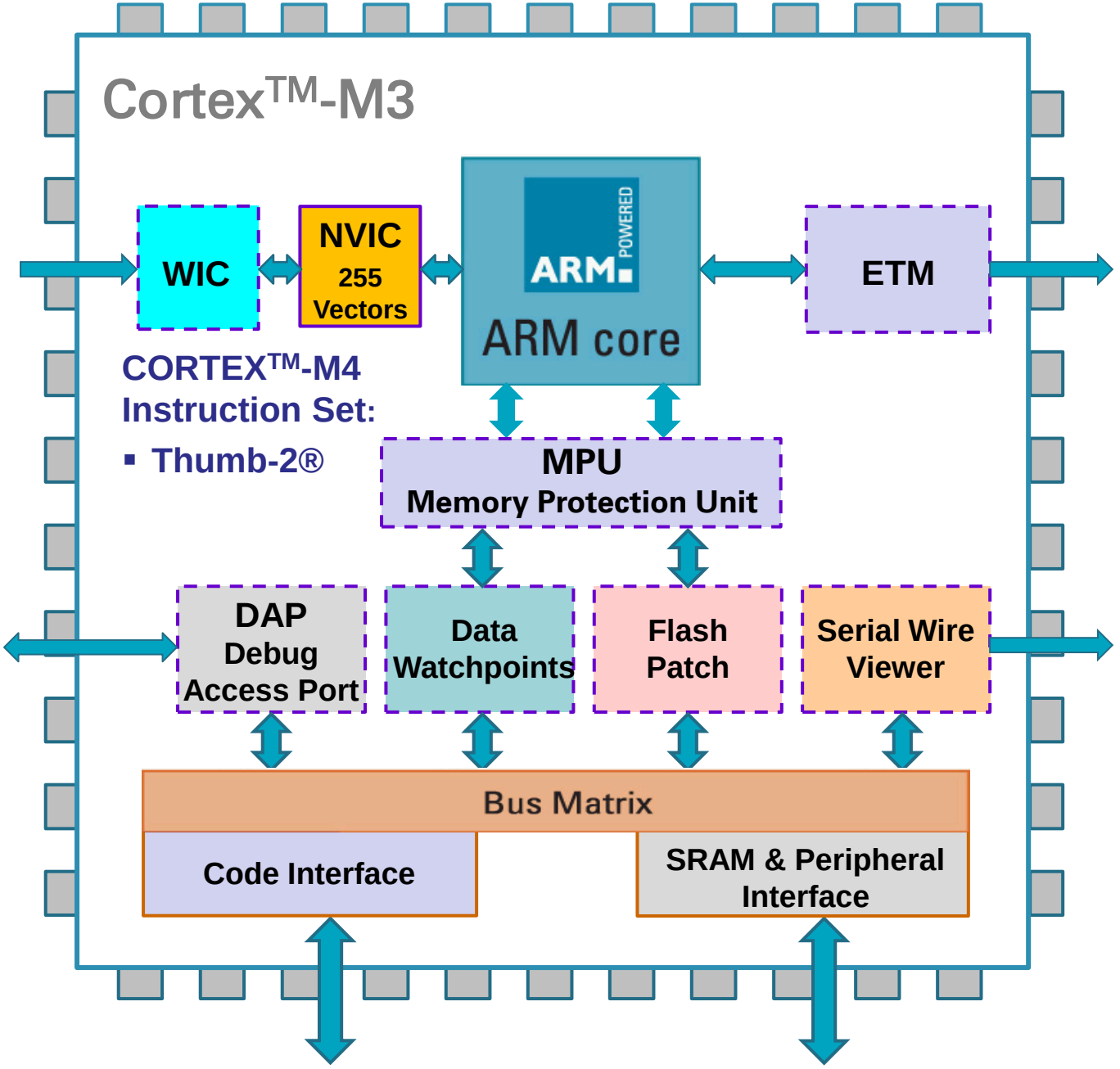


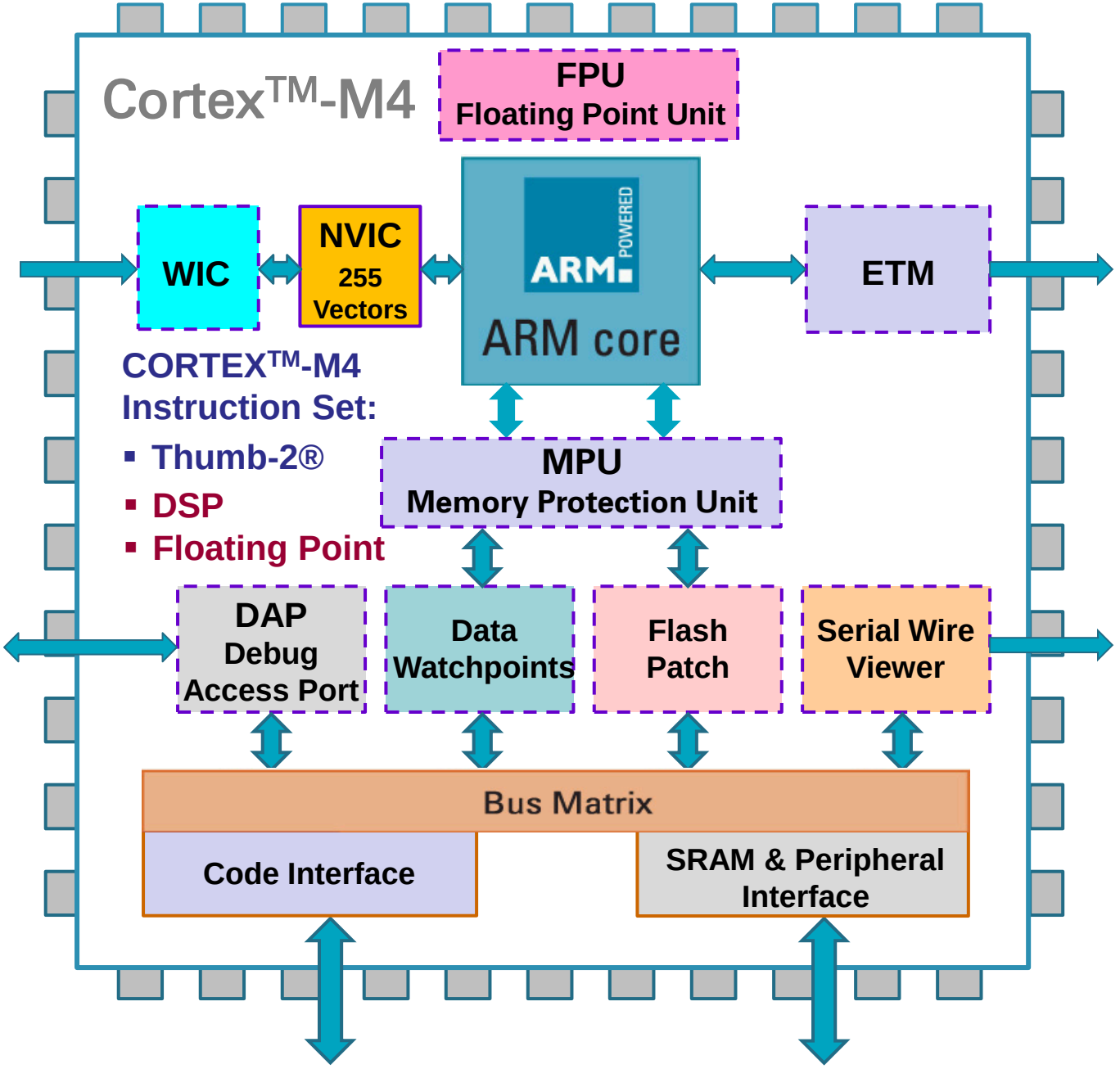










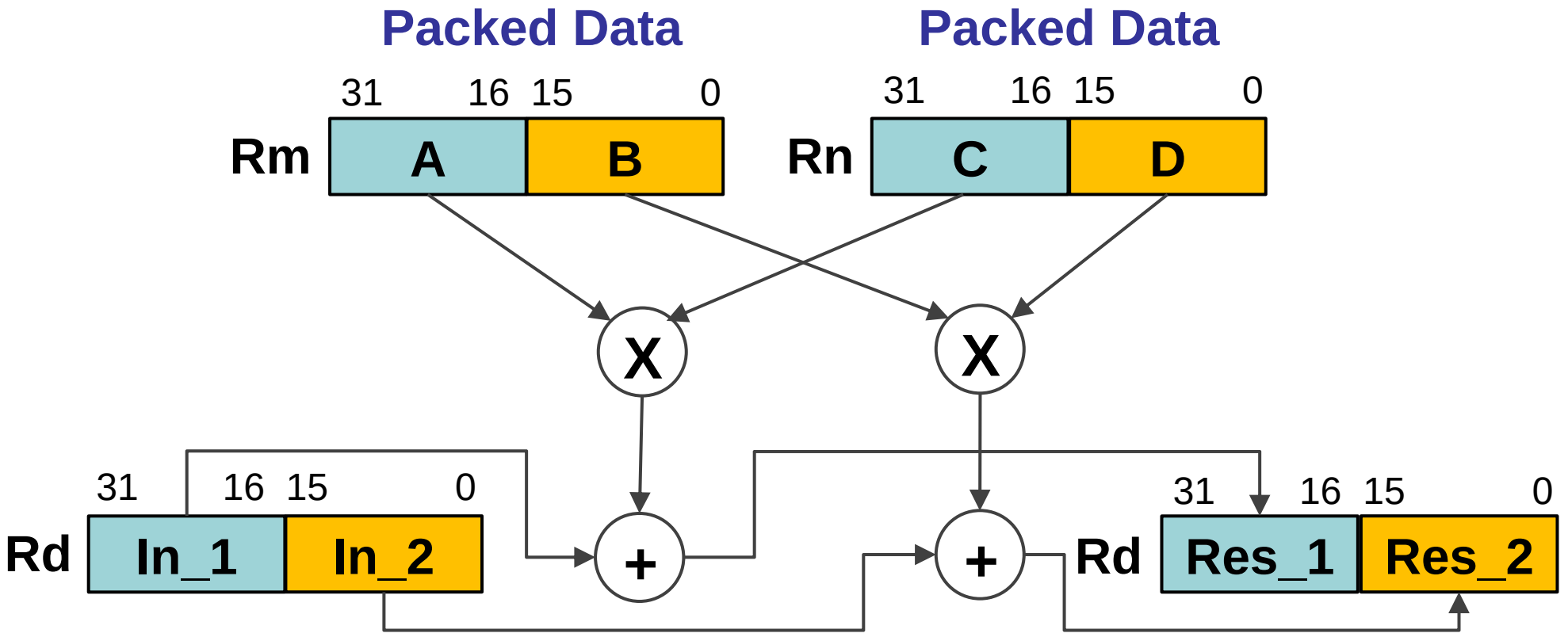


If… Then - bedingte Blöcke in C

```
// C-Code
if (r0 == 0)
   r0 = *r1 + 2;
else
   r1 = *r2 + 4;
```

```
; Assembler Ergebnis
CMP r0,#0
ITTEE EQ
;then
 LDREQ r0, [r1]
 ADDEQ r0, #2
;else
 LDRNE r0, [r2]
 ADDNE r0, #4
```

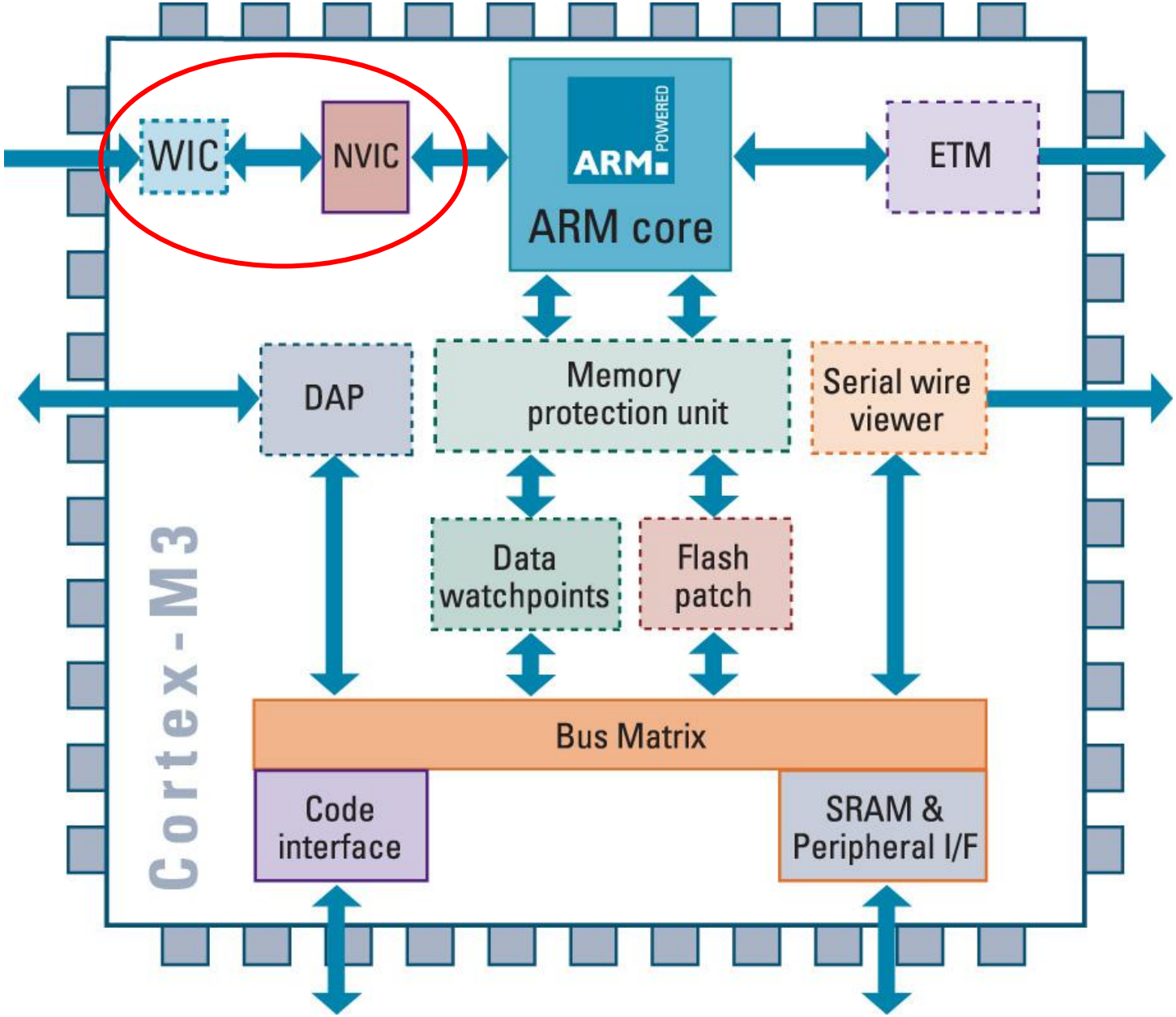
Single Instruction Multiple Data SIMD



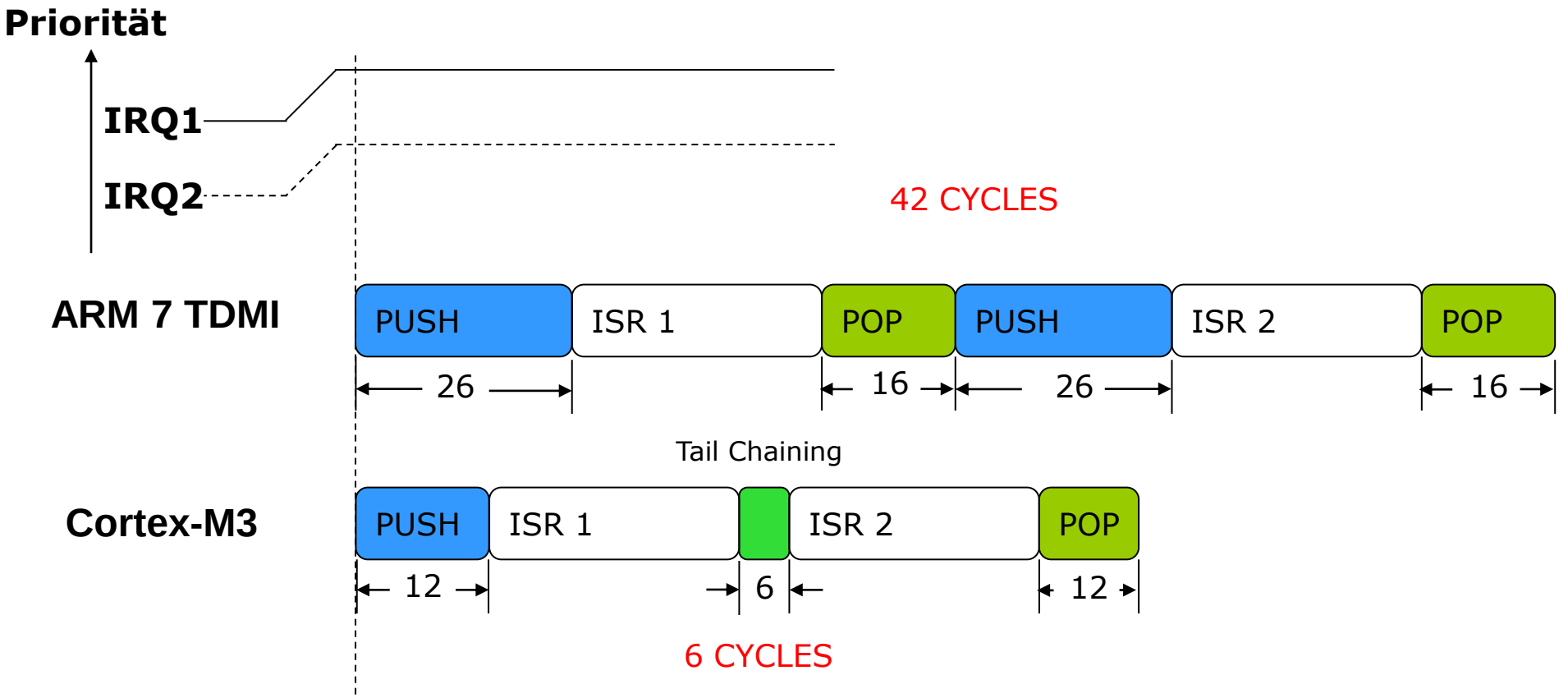
Beispiele:

```
UADD16   Rd, Rm, Rn
SADD16   Rd, Rm, Rn
```

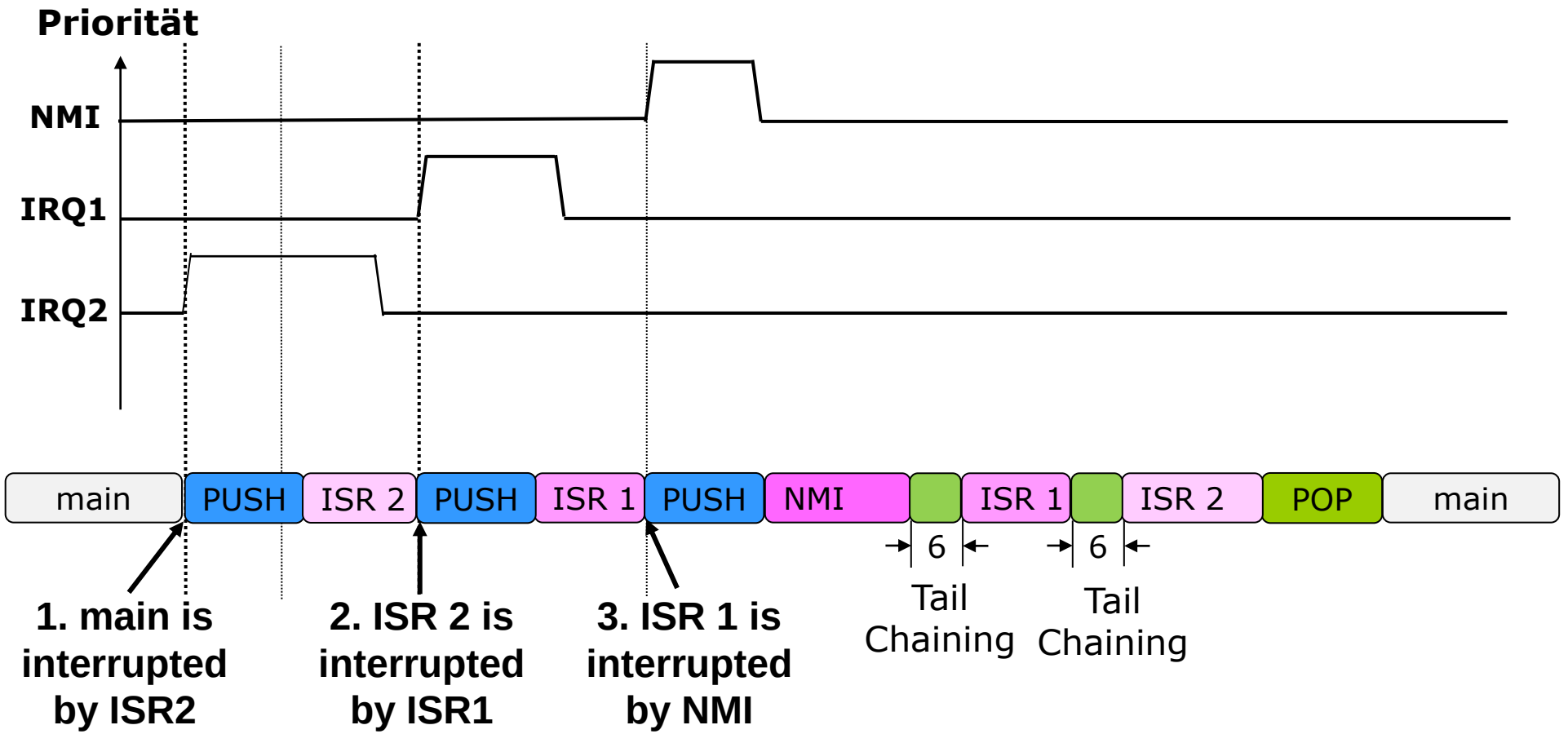
Res_1 = (A * C) + In_1
Res_2 = (B * D) + In_2



Interrupt Response- Tail Chaining



Interrupt Response- Nested Interrupts



Privilege, Modes und Stacks

Modes Thread Mode nach Reset		Operations Privilege Mode nach Reset	Stacks Main Stack nach Reset
	Handler Mode bei Ausführung von Interrupt und Exception	**Privileged execution** volle Systemkontrolle	**Main Stack** durch OS und Interrupts verwendet
	Thread Mode bei normaler Programm-ausführung (kein Interrupt/Exception)	**Privileged / Unprivileged**	**Main Stack/ Process Stack**

Registersatz im Core