ARM Cortex core microcontrollers

2nd Cortex-M3 core

Balázs Scherer

Méréstechnika és
Információs Rendszerek
Tanszék

Budapest University of Technology and Economics
Department of Measurement and Information Systems

The Cortex-M3 core

Cortex M3 core and processor

Cortex-M3 core

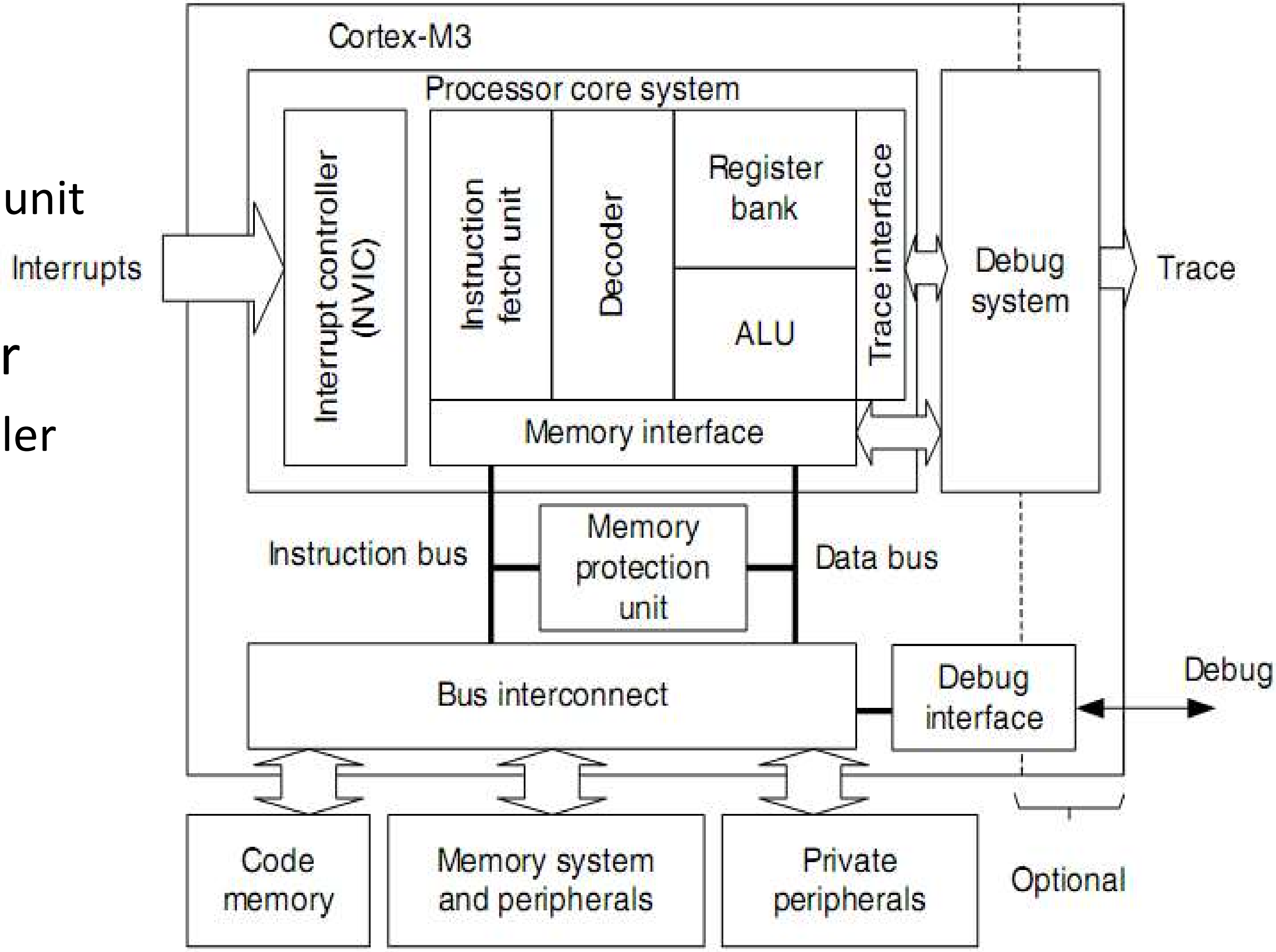
- ALU
- Instruction fetch unit
- Register bank

Cortex-M3 processor

- Interrupt Controller
- Debug system
- Bus Interconnect

Microcontroller

- Peripherals
- Memory
- Clock sources



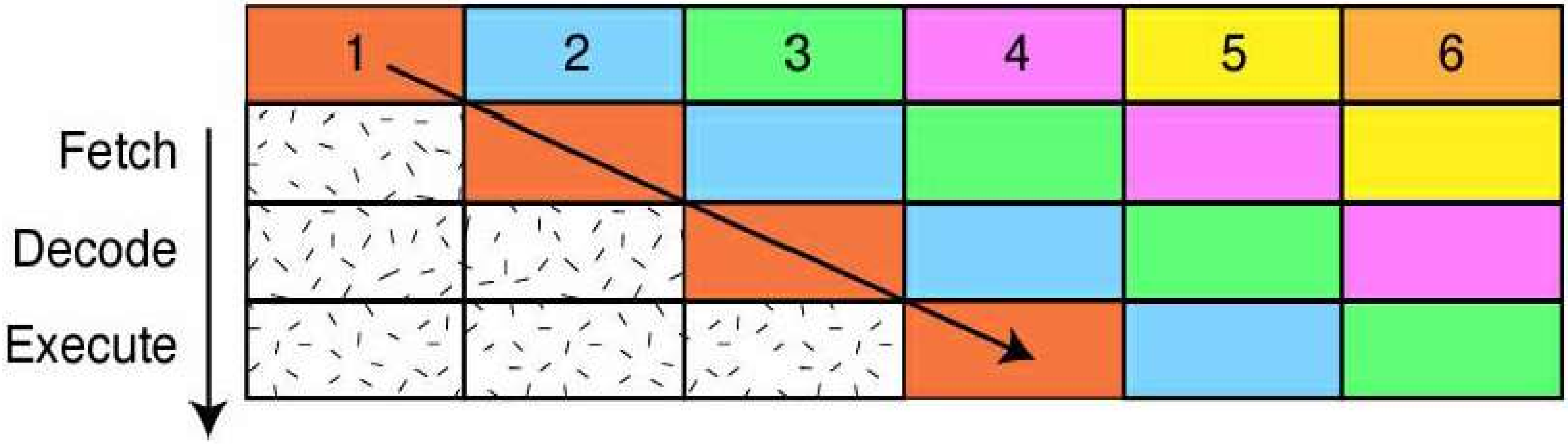
Cortex-M3 main features

- ARMv7M architecture (The ARM7 core has ARMv4 architecture)
- Harvard architecture
  - Separate instruction and data bus: parallel instruction and data fetch
- Thumb-2 instruction set, there is no separate 32-bit ARM and 16-bit Thumb instructions
  - Uses mixed 16-bit and 32-bit instructions
  - More efficient instructions
- Simpler programmer model than ARM 7 core
- System peripherals

Instruction execution and registers

The pipeline of Cortex M3

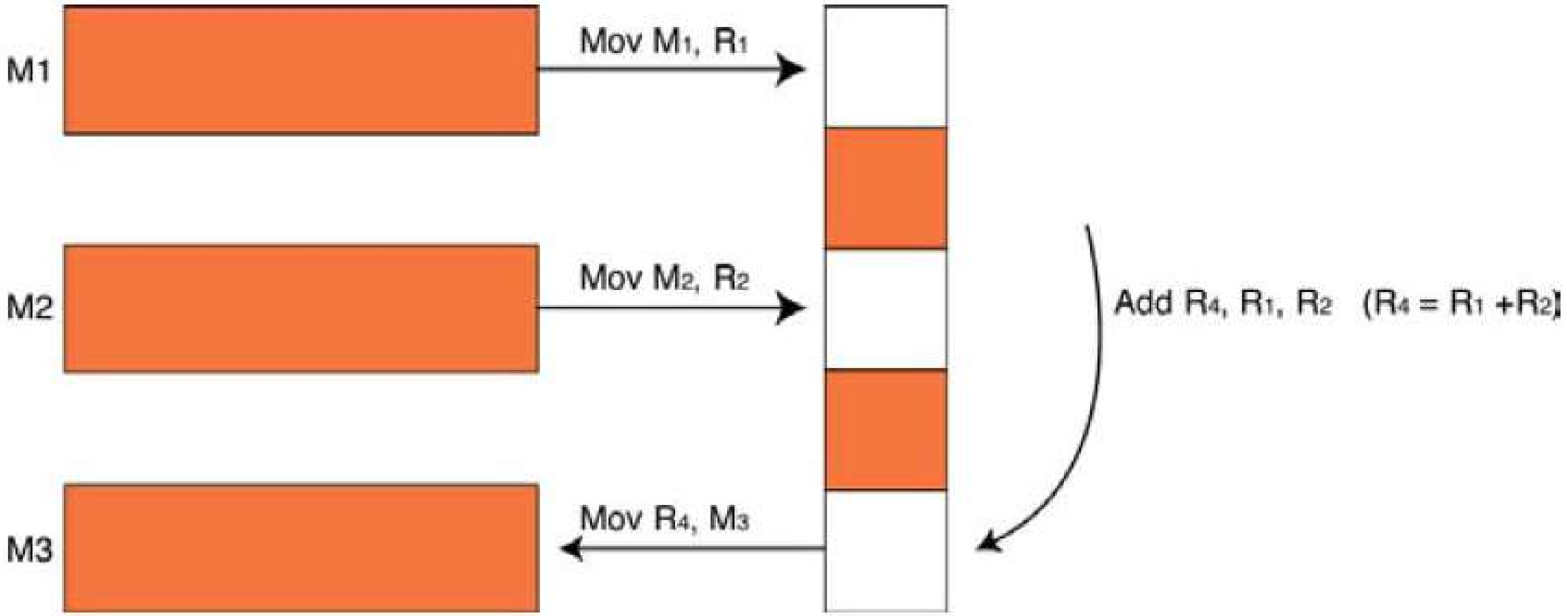
- 3 phase pipeline



- Branch target forwarding
  - In case of branching instruction the target address is identified at the decode phase

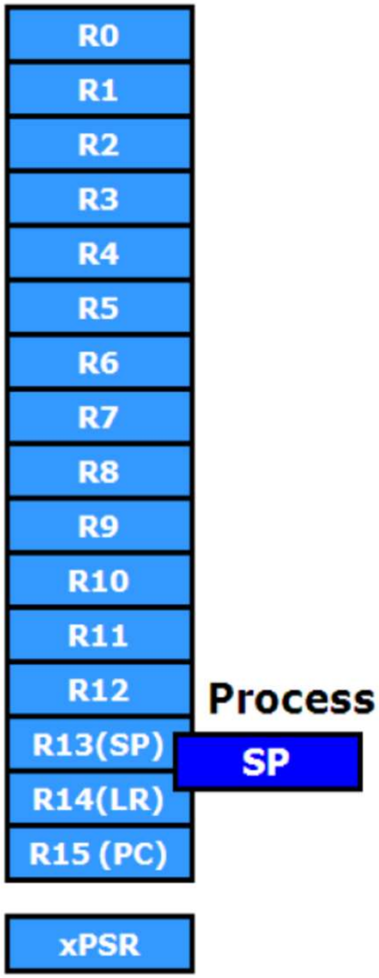
A Cortex-M3 programmer model

- Load and Store architecture
  - First the data needs to be loaded from the memory to a register, then the operation can be executed using the data in the registers, then the result is written back to the memory
  - Same as ARM7 or ARM9 core



Registers of Cortex-M3

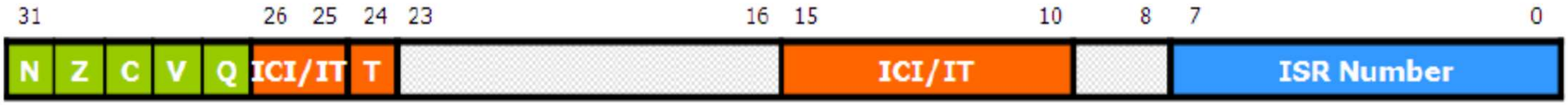
- 16 pieces of 32-bit register



- Similar to the ARM7 and ARM9
  - R13 a stack pointer
  - R14 a link register (return address)
  - R15 a PC (program counter, can be read or written directly)
- The R13 is the stack pointer
  - It is a banked register
  - Process Stack
  - Main Stack-et.

Extended program status register

- The xPSR is not part of the register bank, it is a special purpose register. Only accessable by special instructions.

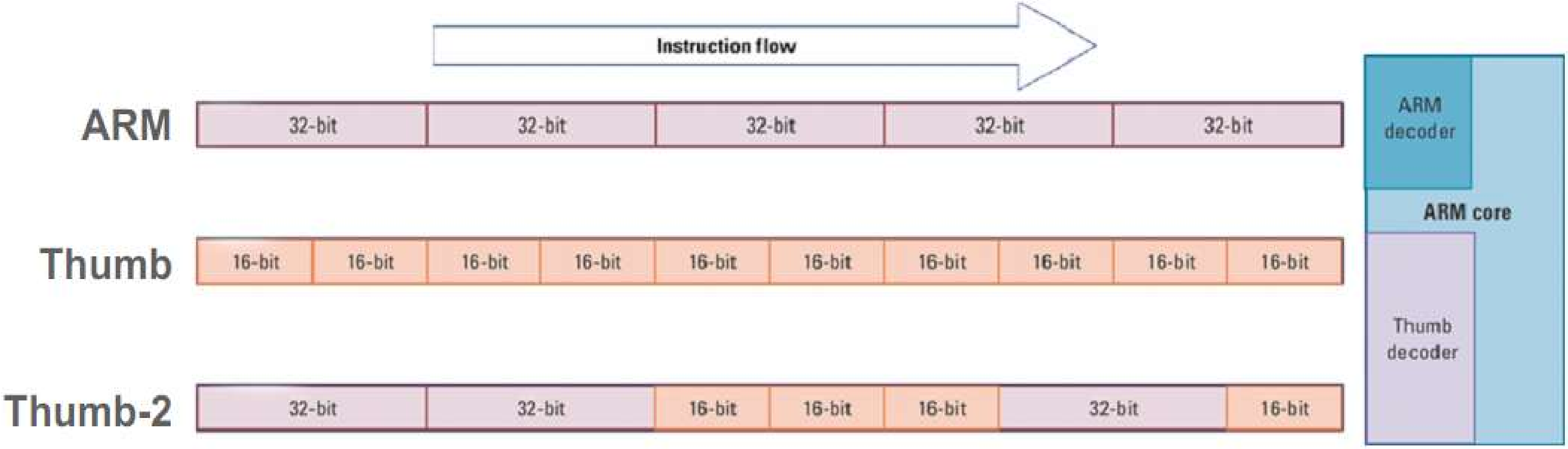


- There are 3 aliases:
- APSR (Application PSR) : Condition code flags (Negatív, Carry, Overflow, Saturated math overflow)
- IPSR (Interrup PSR): The number of the current interrupt
- EPSR (Execution PSR),
  - T: Thumb state (always true)
  - IF-THEN field
    - For simple IF-THEN assembly blocks
  - Interrupt containable instruction field
    - To enable the preemptions of multi cycle operations during interrupts

Instruction set

Mixed 16-bit and 32-bit instructions

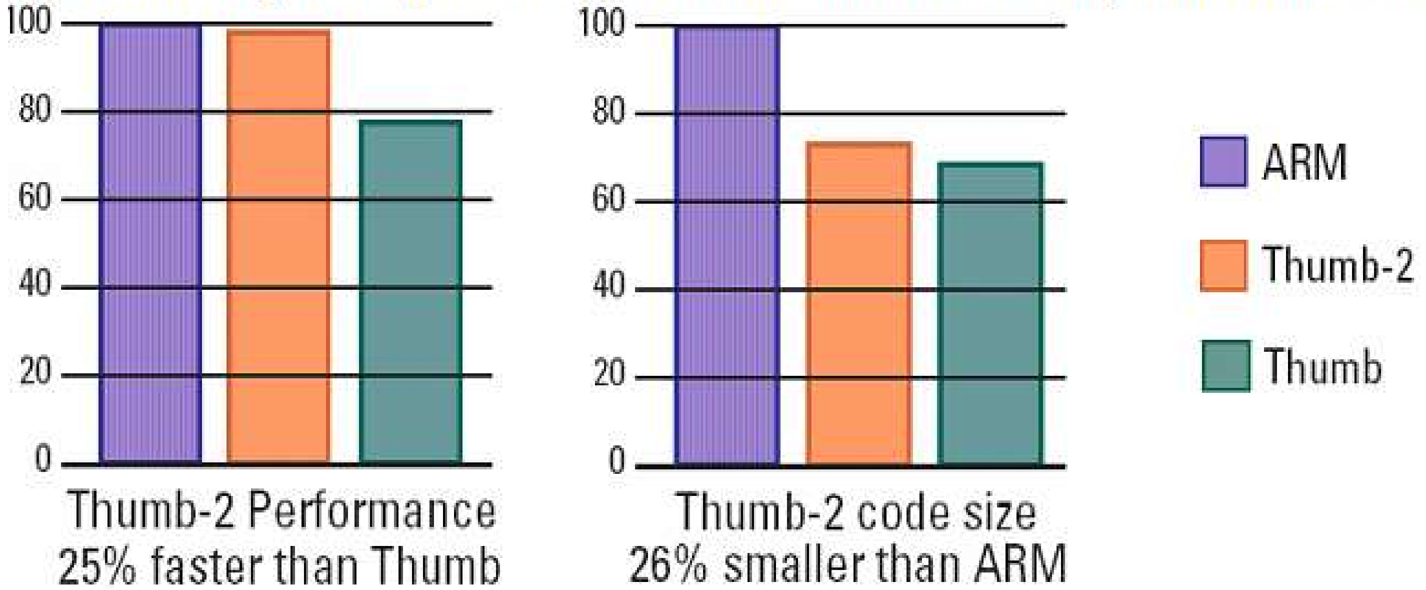
There is no need to fetch instruction in every phase in case of 16 bits instructions



The Thumb2 instruction set

- There is no separate ARM and Thumb instruction set. Only one integrated version the Thumb-2.
  - Thumb-2 is 26% denser than the old 32-bit ARM instruction set
  - Thumb-2 is 25% faster than the old Thumb set.
  - There are instructions for multiplication and dividing

. Relative Dhrystone performance and code size for ARM, Thumb and Thumb-2

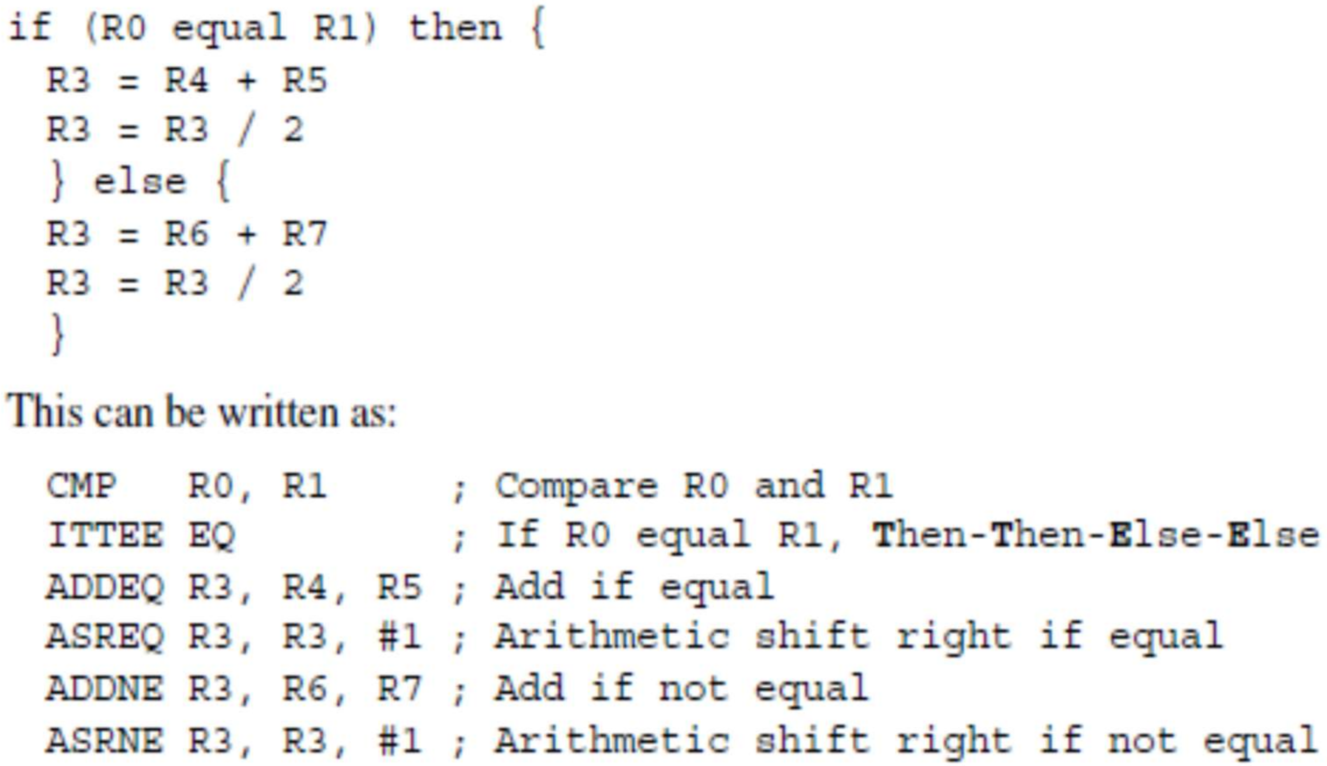


Thumb-2 Performance 25% faster than Thumb

Thumb-2 code size 26% smaller than ARM

Still complex instruction set: IF-THEN block

- Maximum 4 instructions can be placed to an if-then block
  - IT<x><y><z> <cond>
    - <x><y><z>: lehet T:Then, E: else
    - <cond> conditions: EQ: equals, NE: not equals



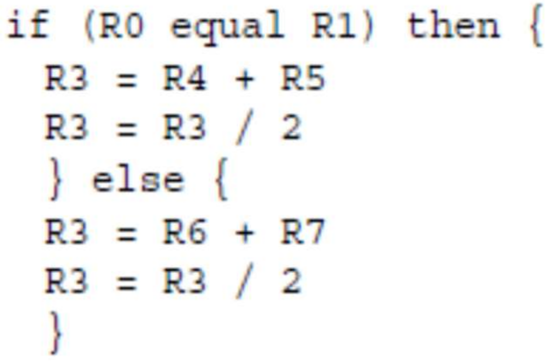
```
if (R0 equal R1) then {
  R3 = R4 + R5
  R3 = R3 / 2
  } else {
  R3 = R6 + R7
  R3 = R3 / 2
  }
```

This can be written as:

```
  CMP   R0, R1      ; Compare R0 and R1
  ITTEE EQ          ; If R0 equal R1, Then-Then-Else-Else
  ADDEQ R3, R4, R5  ; Add if equal
  ASREQ R3, R3, #1  ; Arithmetic shift right if equal
  ADDNE R3, R6, R7  ; Add if not equal
  ASRNE R3, R3, #1  ; Arithmetic shift right if not equal
```

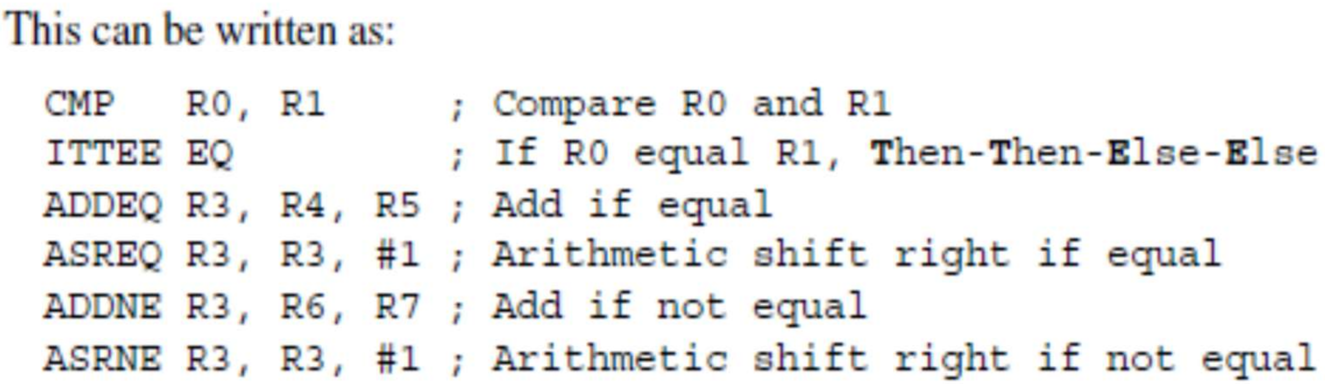
Still complex instruction set: IF-THEN block

- Maximum 4 instructions can be placed to an if-then block
  - IT<x><y><z> <cond>
    - <x><y><z>: lehet T:Then, E: else
    - <cond> conditions: EQ: equals, NE: not equals



```
if (R0 equal R1) then {
  R3 = R4 + R5
  R3 = R3 / 2
  } else {
  R3 = R6 + R7
  R3 = R3 / 2
  }
```

- Instructions in the non selected path Executed as NOP
- Helps the utilization of the pipeline without branching



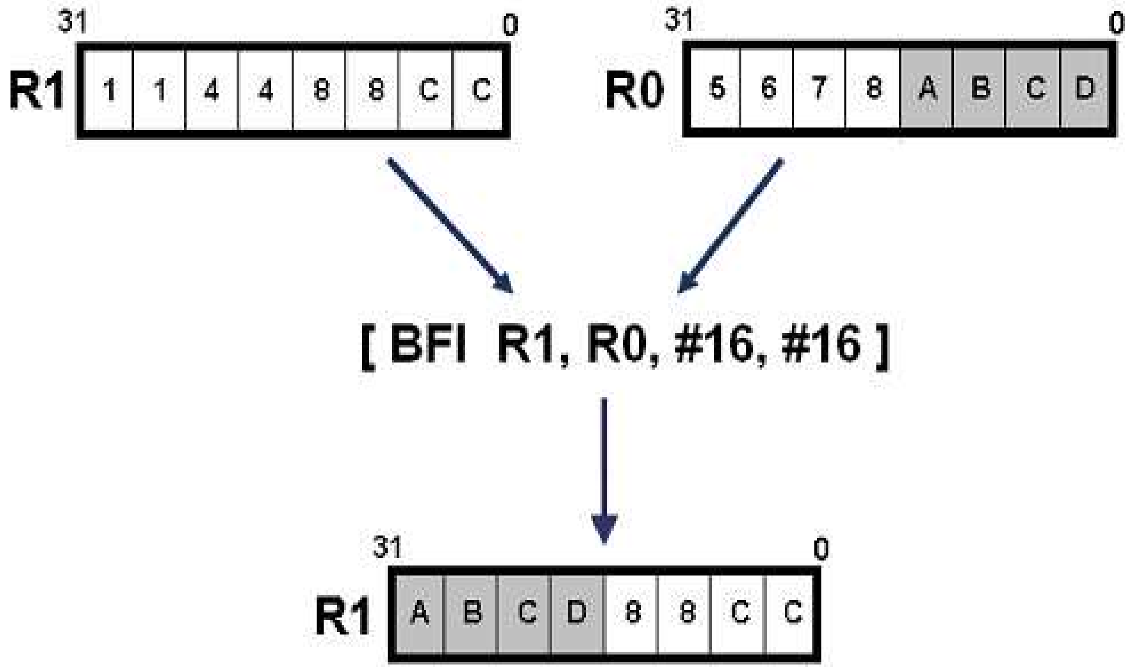
This can be written as:

```
  CMP   R0, R1      ; Compare R0 and R1
  ITTEE EQ          ; If R0 equal R1, Then-Then-Else-Else
  ADDEQ R3, R4, R5 ; Add if equal
  ASREQ R3, R3, #1 ; Arithmetic shift right if equal
  ADDNE R3, R6, R7 ; Add if not equal
  ASRNE R3, R3, #1 ; Arithmetic shift right if not equal
```

Bit field manipulation instructions I.

- Bit Field Clear (BFC),.
- Bit Field Insert (BFI), from another register

Figure 15. Insertion of any number of adjacent bits using BFI

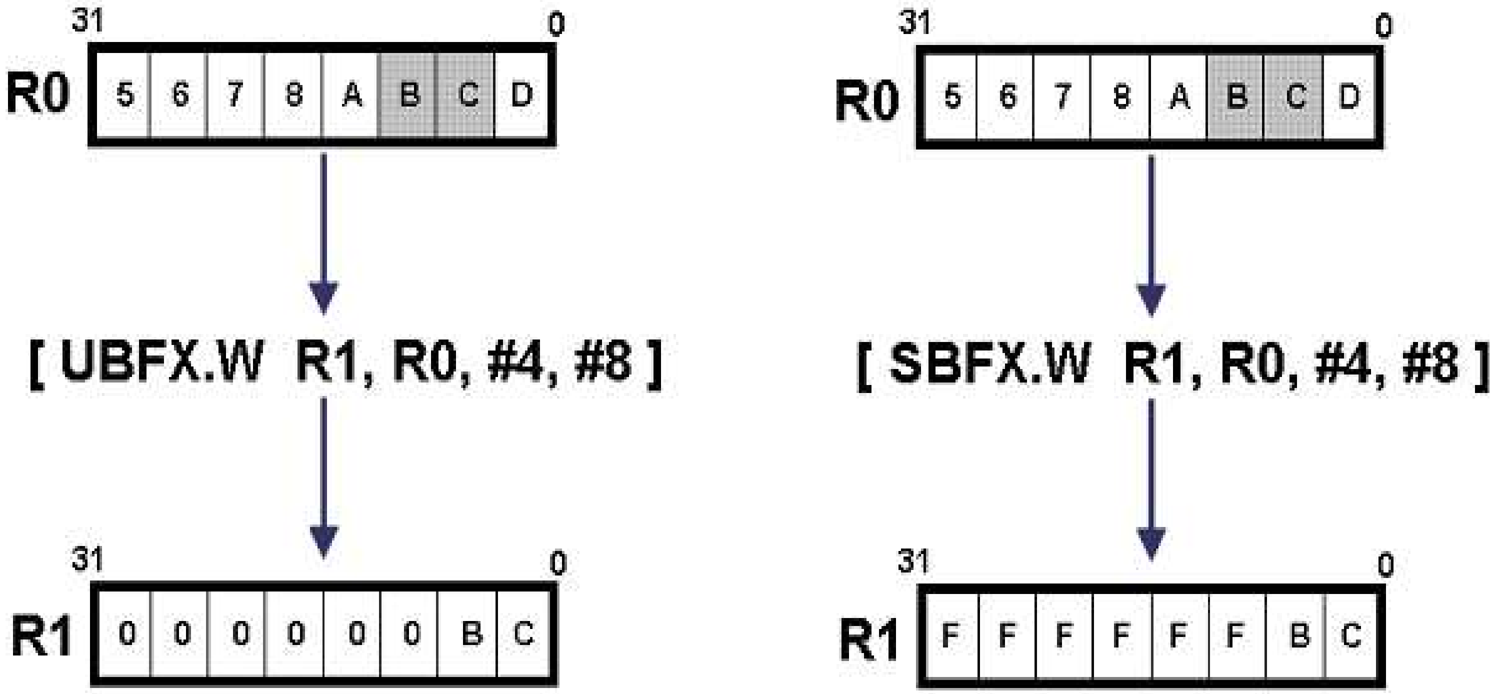


```
[ BFI  R1, R0, #16, #16 ]
```

Bit field manipulation instructions II.

- UBFX (zero extend) and SBFX (sign extend)

Figure 13. Bit extraction with the UBFX and SBFX instructions



Other interesting assembly instructions

- Bit and byte order reversing
- Transferring 64-bit data as one instruction
- One instruction branch table
- Divider instruction

Internal peripherals

System Timer

- 24-bit down counter
- Standardized system timer for all of the ARM Cortex M cores
- Driven by the System clock or System clock / 8
- Three basic control register
  - Counter register
  - Reload
  - Status
    - IT enable
    - Timer config, start/stop

System Timer

- 24-bit down counter
- Standardized system timer for all of the ARM Cortex M cores
  - **Enables easy RTOS porting**
- Driven by the System clock or System clock / 8
- Three basic control register
  - Counter register
  - Reload
  - Status
    - IT enable
    - Timer config, start/stop

Cortex-M3 NVIC

- Nested Vector Interrupt Controller
  - Manufacturer independent standard component: microcontroller manufacturer independent basic interrupt handling
    - Easy porting
  - IT handling is deterministic. Multiple cycle instructions are preempted.
  - Nested interrupt support
    - Multiple interrupt priority level
  - The number of used IT lines are tailored to the given microcontroller
    - The NVIC has a maximum of: 1 non makeable + 15 internal + 240 external peripheral interrupts
    - For example the STM32f107 uses 43 lines

The NVIC vector table

- The reset vector starts from the 0x00000004 address
  - The stack pointer is stored at the 0x00000000 address: enables the usage of C language very early

No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
……	…………………….	…………………….	settable	…………………….
256	Interrupt#240	247	settable	External Interrupt #240

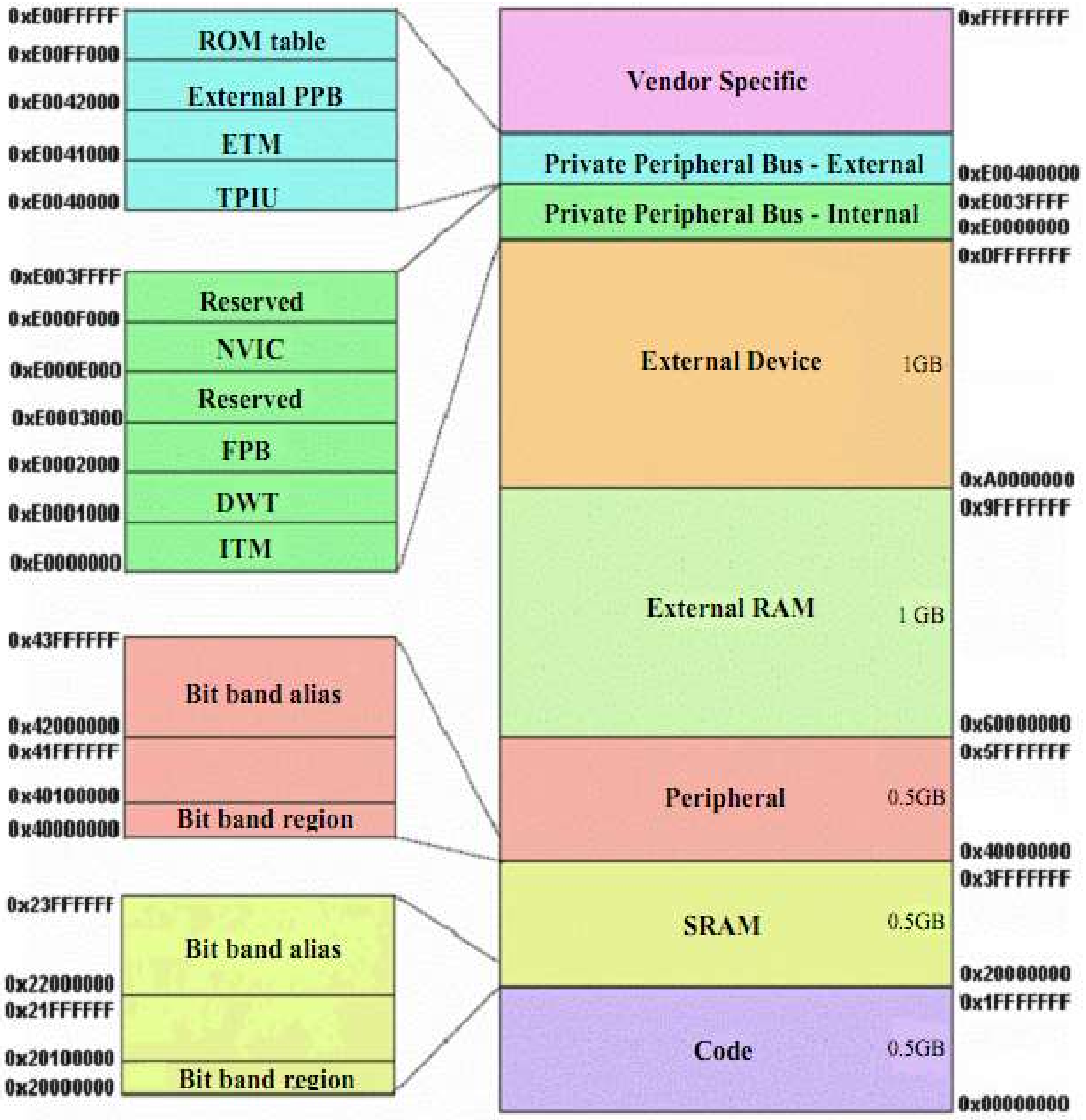
Manufacturer dependent

Memory layout

The memory layout of Cortex M3

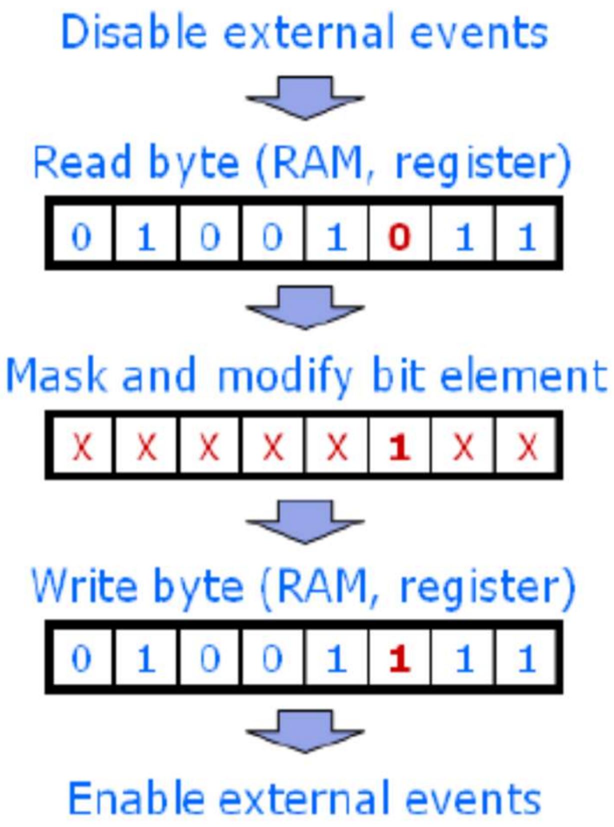
- There was no specified memory layout for ARM 7, or ARM 9
- 4 Gbyte address range
  - 1 Gbyte Code and SRAM area
    - 0.5 G Code area optimized for I-Code bus
    - 0.5 G SRAM area
      - Code can be executed from the SRAM as well
  - 0.5 Gbyte for On-chip peripherals
  - 2 Gbyte for external memories and devices
  - 0.5 Gbyte for Cortex registers and microcontroller specific parts

The memory layout of Cortex M3



Bit banding

- Direct bit control, no need for AND, OR masking



Disable external events

Read byte (RAM, register)

0	1	0	0	1	0	1	1

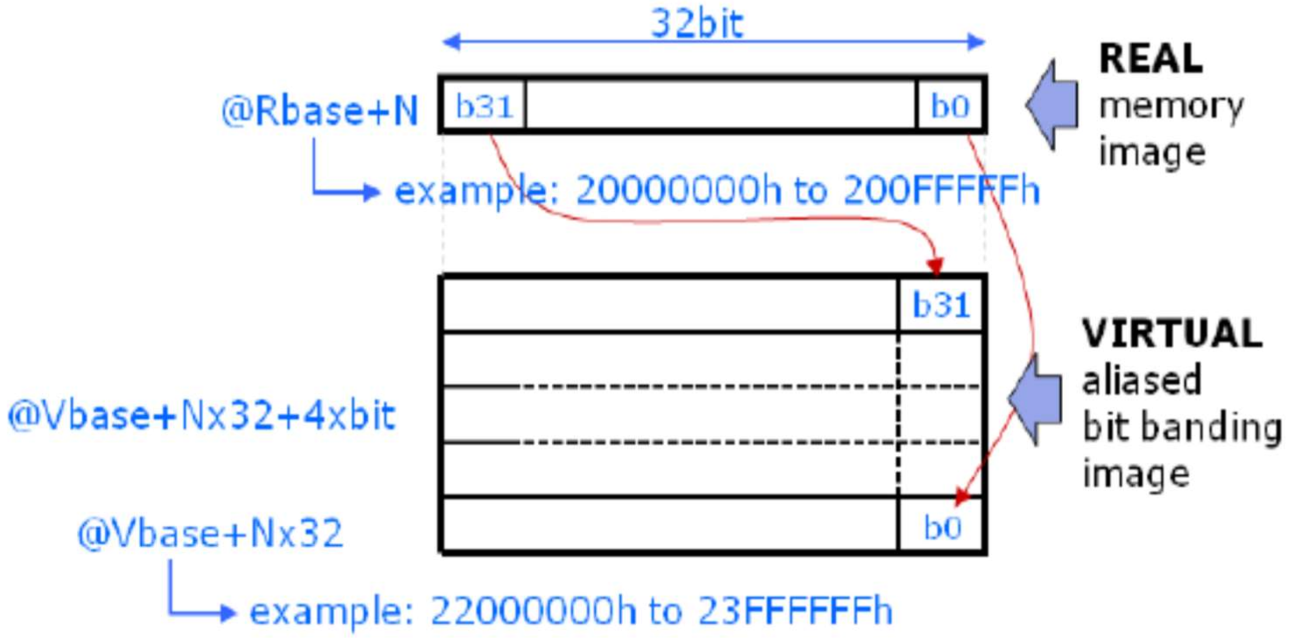
Mask and modify bit element

X	X	X	X	X	1	X	X

Write byte (RAM, register)

0	1	0	0	1	1	1	1

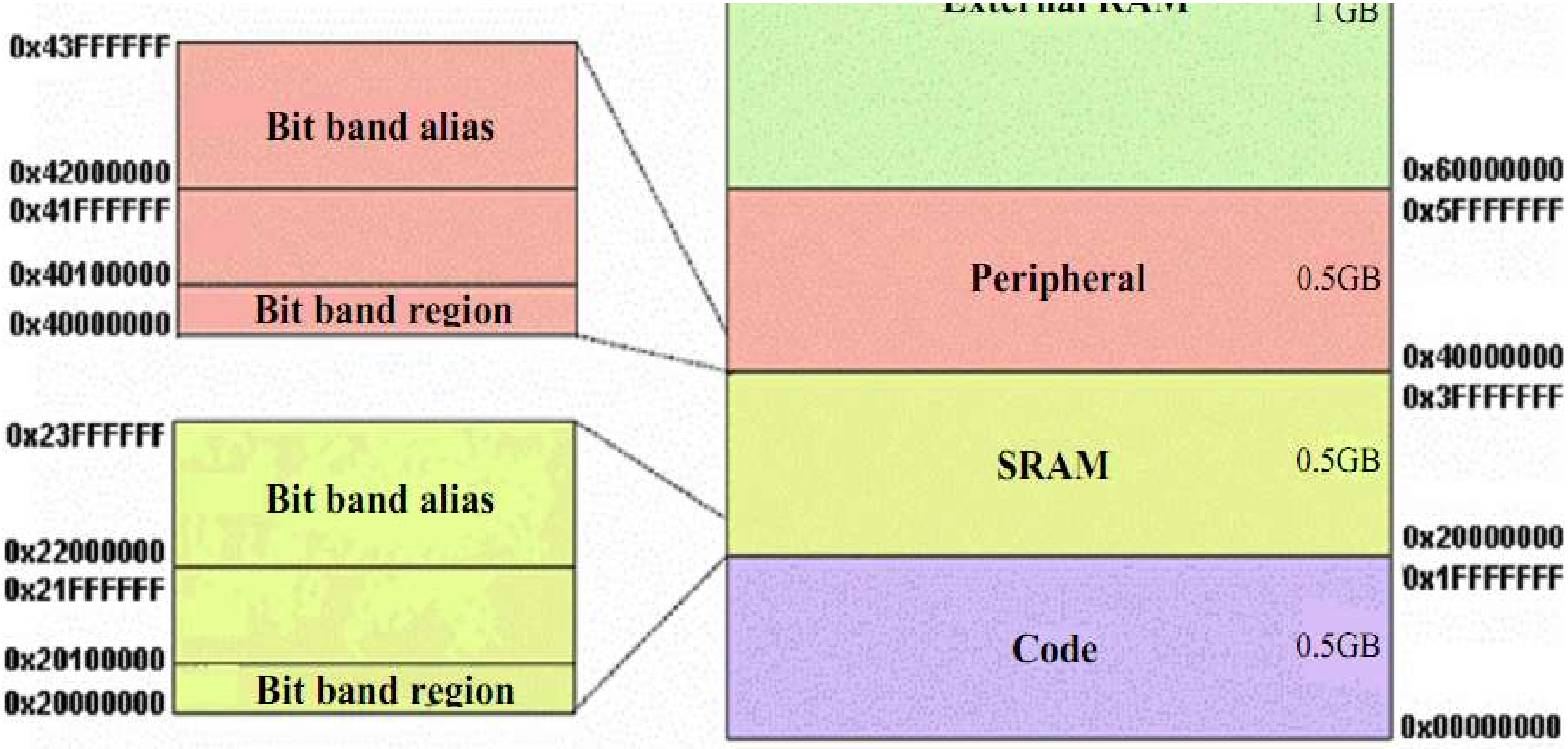
Enable external events


32bit
@Rbase+N
b31
b0
REAL memory image
example: 20000000h to 200FFFFFh
b31
VIRTUAL aliased bit banding image
@Vbase+Nx32+4xbit
@Vbase+Nx32
b0
example: 22000000h to 23FFFFFFh

The bit banding technique allows atomic bit manipulation while keeping the Cortex-M3 CPU to a minimal gate count.

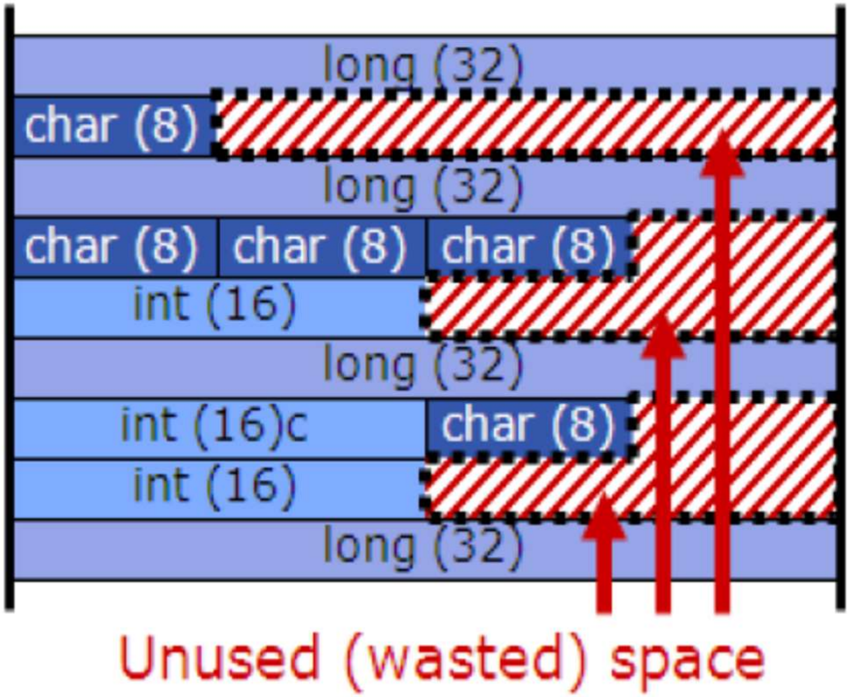
Bit band memory regions

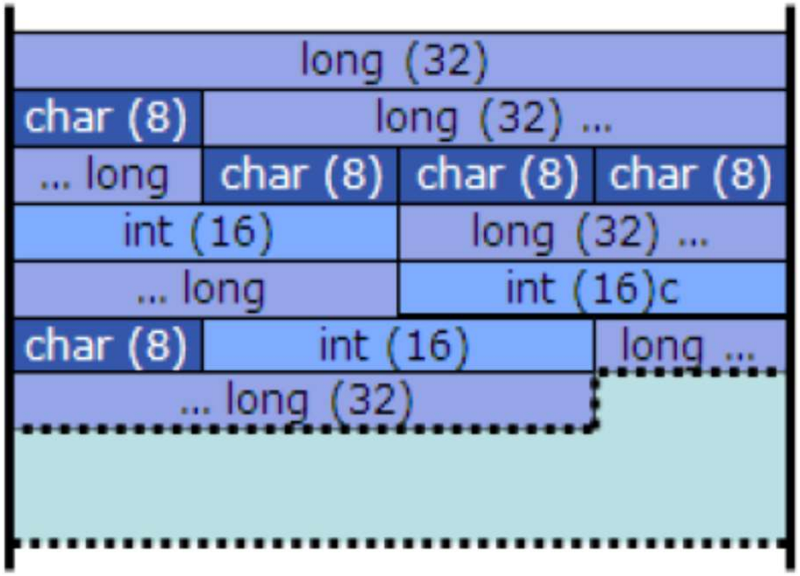
- First 1 Mbyte of SRAM and Peripheral area
  - Practically there is no need for more



Non aligned memory access

- The ARM 7, and ARM 9 used aligned memory access
  - Could lead to wasted RAM space up to 25%
  - Can be very problematic for many code (Vektor CCP stack)





Access modes and Execution models

Access modes of Cortex-M3

- The Cortex-M3 has two type of access modes
  - Privileged access mode
    - Also called supervisor
    - This is the default mode after reset
    - Interrupt or event place the core to privileged mode
    - Grant access to every resource
  - Unprivileged operation
    - Also called user mode
    - Restricted access
      - Some operations are denied, like xPSR manipulating ones
      - The System Control Space (SCS) registers are non accessible (NVIC and SysTick)

The operation modes of Cortex-M3

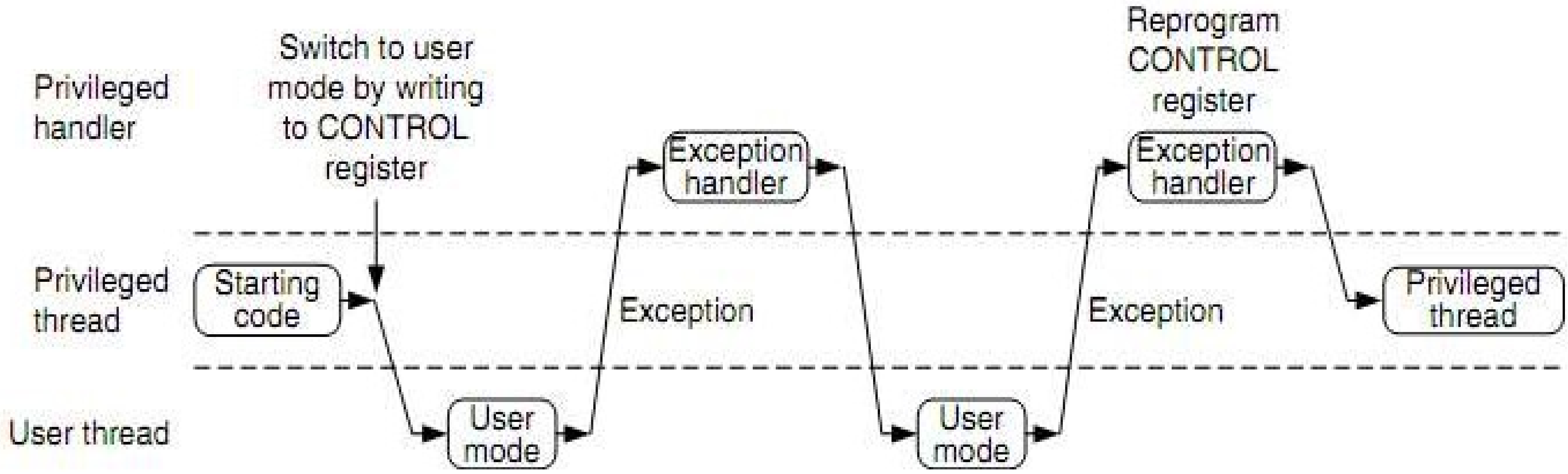
- Thread mode:
  - Normal operation
  - Can use privileged or non-privileged access mode.
    - If switched to non-privileged only the Handler mode can switch it back
- Handler mode:
  - Event and interrupt handling
  - Always privileged access mode

Stack registers of Cortex-M3

- Main stack
  - For RTOS, event and interrupt handling
- Process stack
  - Designed for Thread operation mode. (Thread mode can also use the main stack).

- Separate stack registers can provide protections for RTOS core

Example for privileged code execution



Access and operation modes

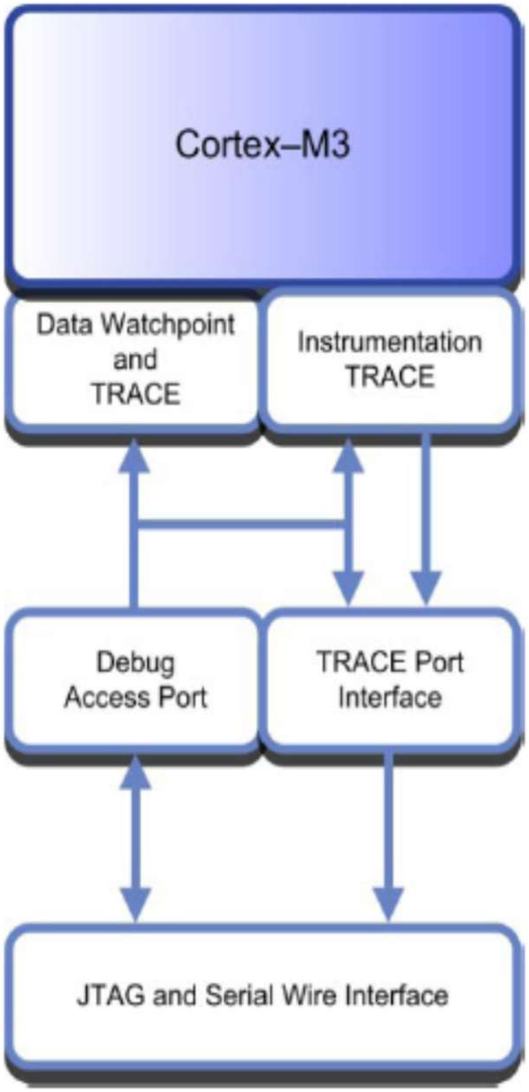
	Operations **(privilege out of reset)**	**Stacks** **(Main out of reset)**
Handler - An exception is being processed	Privileged execution Full control	Main Stack Used by OS and Exceptions
Thread - No exception is being processed - Normal code is executing	Privileged/Unprivileged	Main/Process

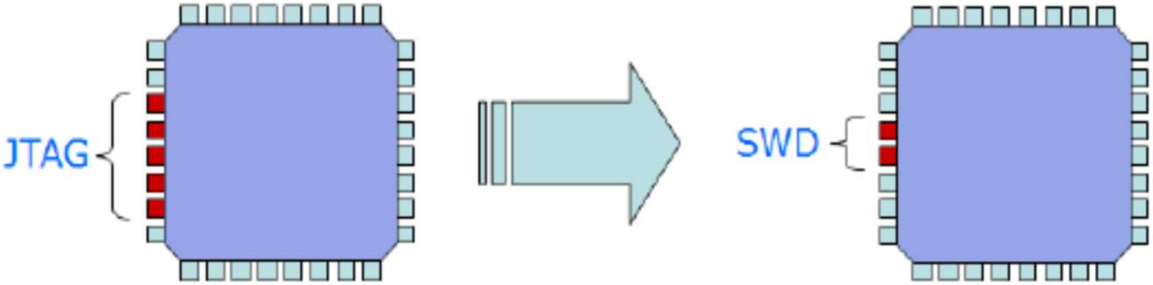
Other specialities

Low power operation

- Sleep mode
  - Processor core is inactive
  - Part of the NVIC is active, and can wake up the processor core
- Sleep now
  - WFI instruction: Wait For Interrupt
    - Goes to Power down, interrupts wake it up
  - WFE utasítás: Wait For Event
    - Peripheral interrupt line can wake the processor, but it do not have to handle the interrupt it can continue with the main program
- Sleep on Exit
  - After IT the core sleeps back
- Deep Sleep
  - The Cortex-M3 core signals the microcontroller specific peripherals the sleep mode

Debug system





The Cortex CoreSight debug system uses a JTAG or serial wire interface. CoreSight provides run control and trace functions. It has the additional advantage that it can be kept running while the STM32 is in a low power mode. This is a big step on from standard JTAG debugging.