UTFPR

Sistemas Embarcados: (ELF74)
Prof: DaLuz

UTFPR

Arquitetura:

ARM Pdfs adicionais:

http://www.elf74.daeln.com.br/Pdfs/Arm-Cortex-M4-Processor-Datasheet.pdf

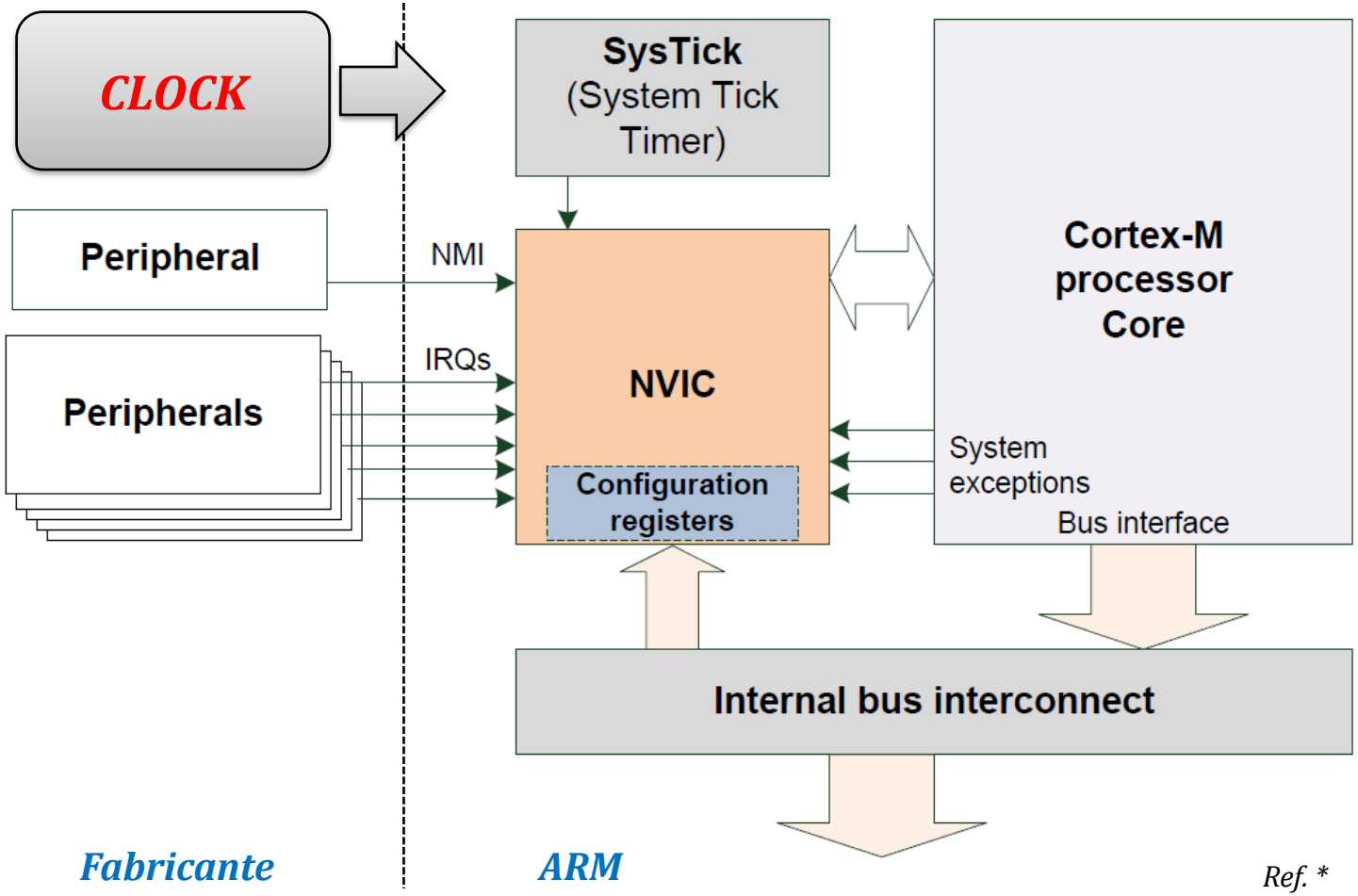
http://www.elf74.daeln.com.br/Pdfs/Arm_Armv7m_Arm.pdf

http://www.elf74.daeln.com.br/Pdfs/Arm_Cortex_M4_r0p0_trm.pdf

UTFPR

Arquitetura:

Cortex-M – Concepção



*Ref. **

UTFPR

Arquitetura:

Cortex-M Instruction Set Architecture

Instruções do núcleo Cortex-M (**Revisão**):

- **Tipo de dados comuns**
- **Modos de operação**
- **Registradores**
- **Instruções**
- **Acesso a Memória**
- **Exceções**

Cortex-M ISA:

Tipos de Dados

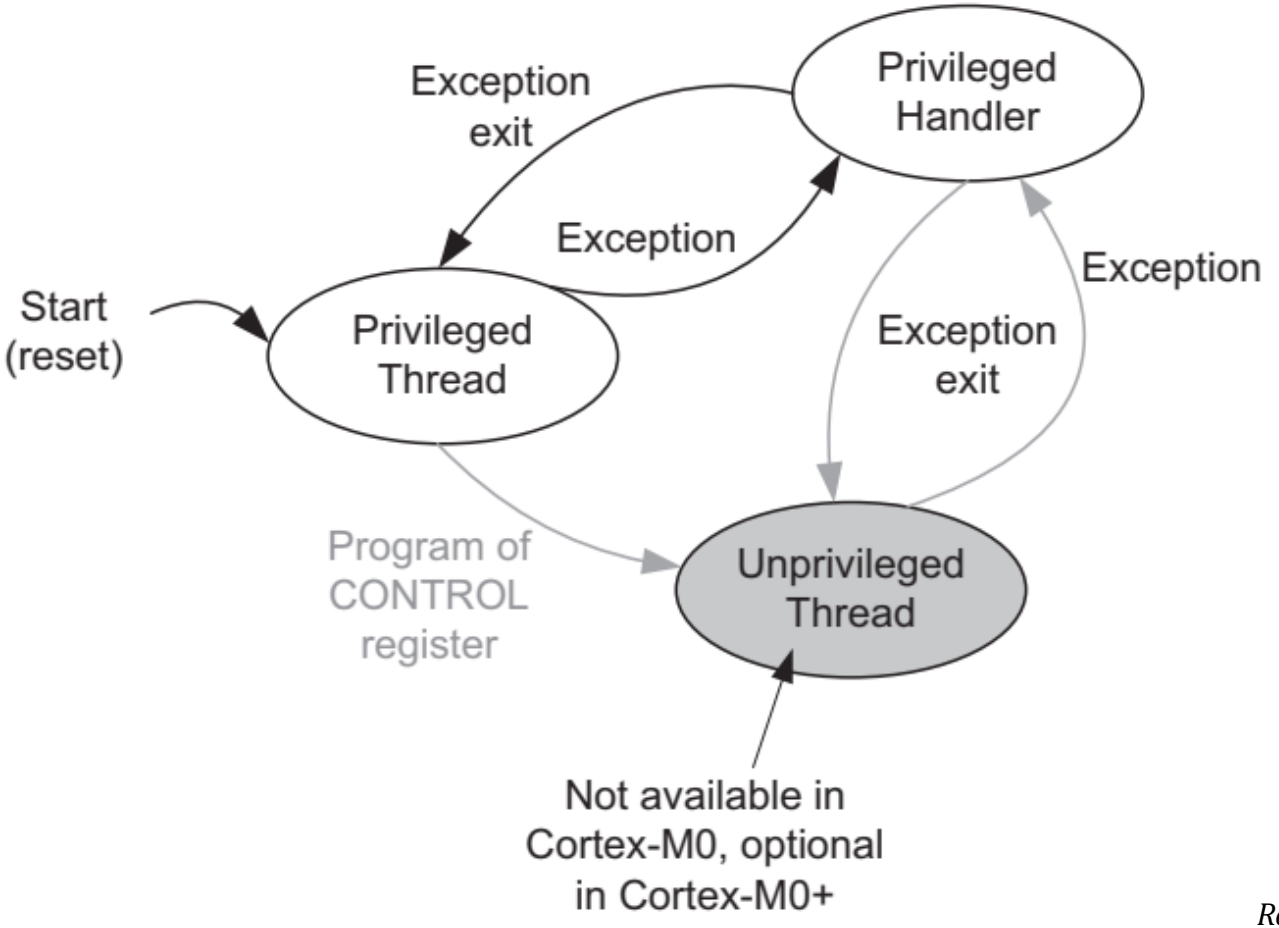
Instruções do núcleo Cortex-M (**Revisão**):

- **bit:** guarda apenas um bit (**0** ou **1**).
- **byte:** conjunto de **8-bit** = 1 x endereço **RAM**
- **half-word:** **16-bit** = 2 x endereço **RAM**
- **word:** **32-bit** = 4 x endereço **RAM**
- **double-word:** **64-bit** = ex. R1:R0 = 8 x endereço **RAM**

UTFPR

Cortex-M ISA:

Modos de Operação

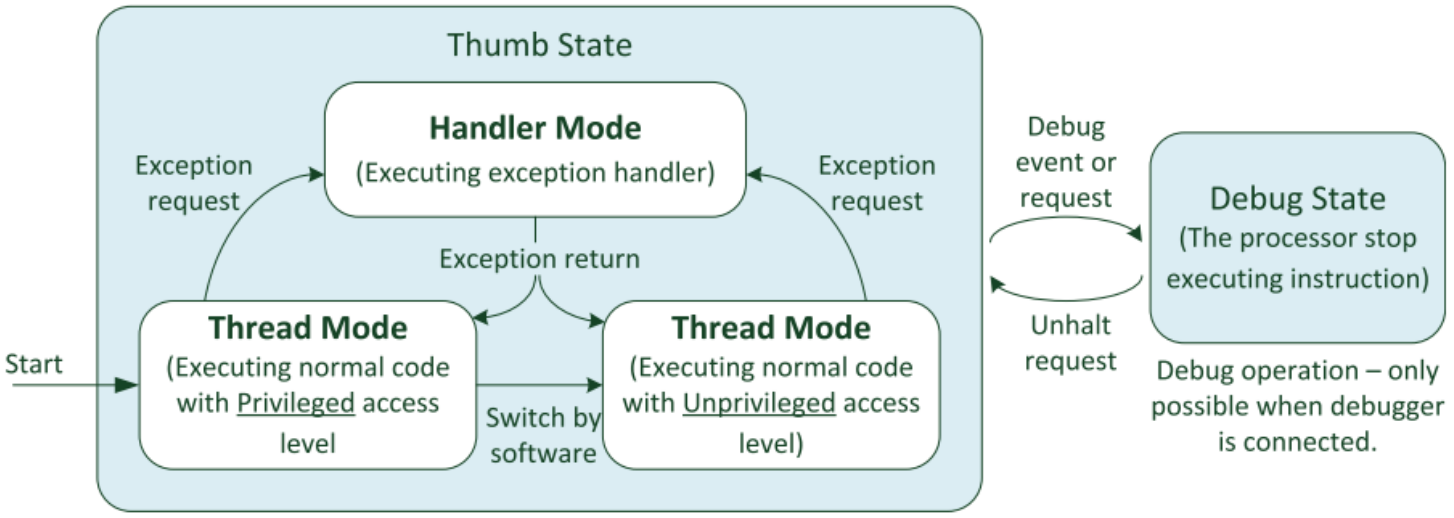


*Ref. **

UTFPR

Cortex-M ISA:

Modos de Operação



- **1) Privilegiado:** Execuções S.O.
- **2) Ñ Privilegiado:** Aplicação, *Thread.*
- **3) Handler:** Exeções, Interrupções, *H. Faults*

*Ref. **

UTFPR

Cortex-M ISA:

Registradores

- 1) Geral (R0-R15)
- 2) F.P. (S0-S31)
- 3) S.R. (xPSR, Primask, Faultmask, Basepri, Control)

- *Low registers: R0-R7*
- *High registers: R8-R15*
- *Special: SP, LR, PC*

- *Obs: Todos 32-bit*

Name		
R0		
R1		
R2		
R3		
R4		
R5		
R6		
R7		
R8		
R9		
R10		
R11		
R12		
R13 (MSP)	R13 (PSP)	Main Stack Pointer (MSP), Process Stack Pointer (PSP)
R14		Link Register (LR)
R15		Program Counter (PC)

Floating Point Unit

S1	S0	D0
S3	S2	D1
S5	S4	D2
S7	S6	D3
S9	S8	D4
S11	S10	D5
S13	S12	D6
S15	S14	D7
S17	S16	D8
S19	S18	D9
S21	S20	D10
S23	S22	D11
S25	S24	D12
S27	S26	D13
S29	S28	D14
S31	S30	D15

FPSCR — Floating Point Status and Control Register

Name	Functions	
xPSR	Program Status Registers	Special Registers
PRIMASK	Interrupt Mask Registers	
FAULTMASK		
BASEPRI		
CONTROL	Control Register	

*Ref. **

Cortex-M ISA:

Registradores

Registradores: R0-R12, R13(SP), R14(LR), R15(PC)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Uso Geral								Uso Geral							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Uso Geral								Uso Geral							

- Registradores R0 a R15 – R0 a R12 ⇨ 13 de uso geral;
- Acessíveis para a maioria das instruções L:R0-R7, H:R8-R15
- **b0** é o *Least Significant bit* (**LSb**) e **b31** é o *Most* S. b. (**MSb**)
- *Capacidade:* **32-bit** ⇨ 8 dígitos hexad. ⇨ 0x1A2B.3C4D

  0 a *4.294.967.295 (U) ou -2,147,483,648 a 2.147.384.647 (S)*
- *Os valores podem representar: números ou endereços.*

UTFPR

Cortex-M ISA:

xPSR = APSR + EPSR + IPSR

APSR - *Application Program Status Register*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	Reservado							GE[3:0]				Reservado															

EPSR - *Execution Program Status Register*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado					ICI/IT		T	Reservado								ICI/IT						Reservado									

IPSR - *Interrupt Program Status Register*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado																							0 ou Número da Exceção								

xPSR - Acesso Combinado - *Application Program Status Register*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	ICI/IT		T	*Reservado*				GE[3:0]				ICI/IT						*R.*	0 ou Número da Exceção								

- Os três registradores **APSR**, **EPSR** e **IPSR** podem ser acessados individualmente ou combinados no **xPSR** ou dois a dois **IAPSR**, **EAPSR ou IEPSR.**

UTFPR

Cortex-M ISA:

Campos dos bits

Bit Fields - Campo de bits no *program status register*	
N	Negative flag
Z	Zero flag
C	Carry (or NOT borrow) flag
V	Overflow flag
Q	Sticky saturation flag (not available ARMv6-M)-**DSP**
GE[3:0]	Greater-Than or Equal flags for each byte lane (ARMv7E-M only; not available in ARMv6-M or Cortex-M3)-**SIMD**.
ICI/IT	Interrupt-Continuable Instruction (ICI) bits, IF-THEN instruction status bit for conditional execution (not available in ARMv6-M).
T	Thumb state, always **1**(Cortex-**M**); trying to clear this bit will cause a fault exception. (0 or 1) in Cortex-R or Cortex-A
Exception Number	Indicates which exception the processor is handling.

UTFPR

Cortex-M ISA:

Registradores Especiais

Registrador: PRIMASK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado																															PM

Registrador: FAULTMASK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado																															FM

Registrador: BASEPRI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado																								BASEPRI							

Campo	bit	Descrição
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).
BASEPRI	BASEPRI[7:0]	Changes the priority level required for exception preemption. Affects only the currently executing code with lower priority than BASEPRI.

UTFPR

Cortex-M ISA:

Registradores Especiais

Registrador: CONTROL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservado																													FPCA	SPCEL	nPRIV

Campo	bit	Descrição
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). In Handler mode this bit is always 0.
FPCA	2	Implemented only when floating point is available. 0 = do not save floating point registers on exception. 1 = save floating point context on exception.

UTFPR

Cortex-M ISA:

Instruções: Cortex-M4 / ARMv7-M

Instruções do núcleo Cortex-M (**Revisão**):

- Somente um set de instruções: **Thumb2**
- Mix de instruções: **16** e **32-bit**
- Alinhamento de Código: endereços **pares**
- Interno: **SysTick**, **NVIC**, **MPU**, **FPU**
- S.O. : PendSV, Modos P./Ñ.P., **MPU**, **SysTick**
- Instruções Bit-band

Cortex-M ISA:

Instruções: Genérica

Label

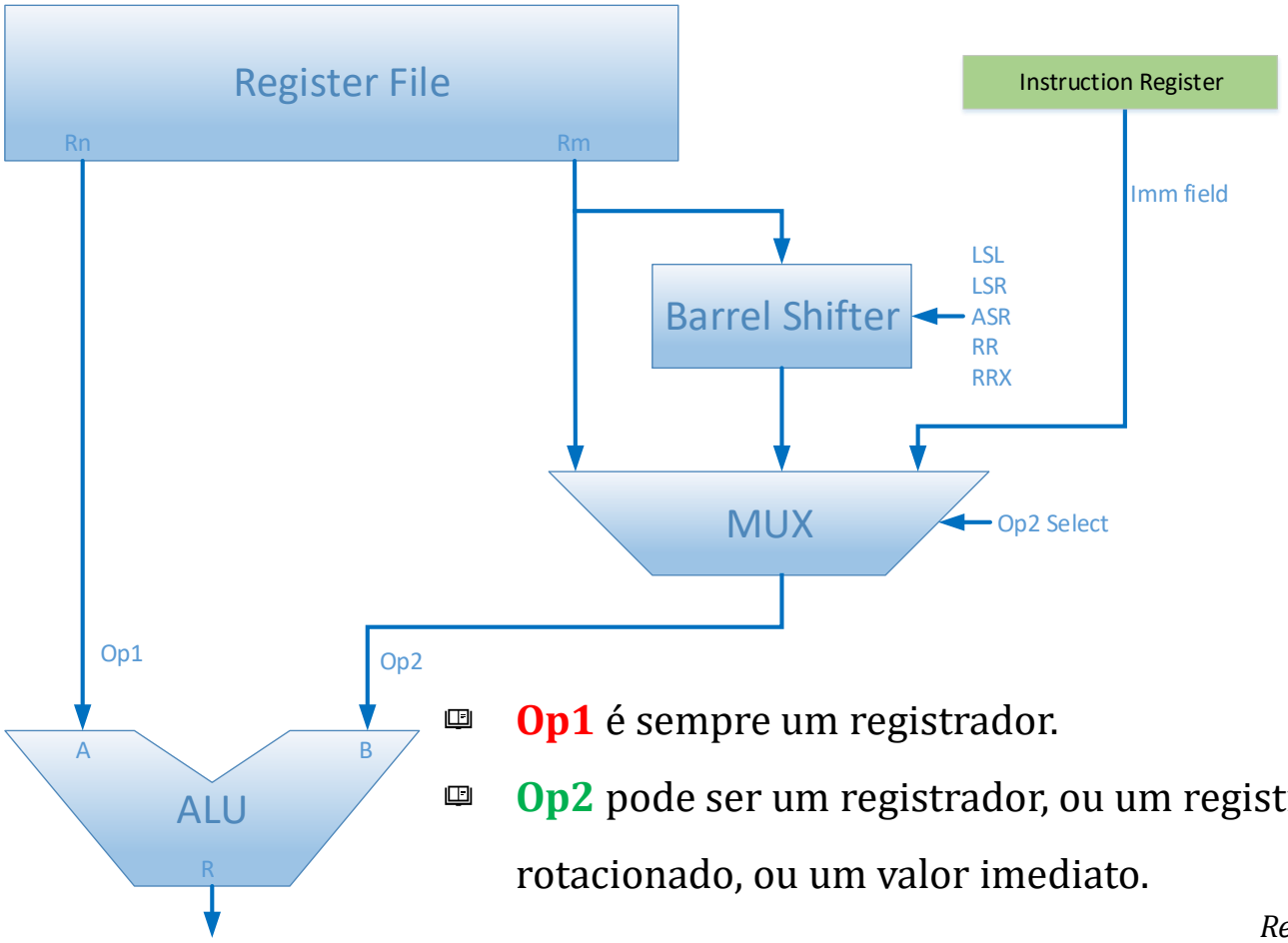
```
MNEMÔNICO Destino, Operando1,              ;Comentário
MNEMÔNICO Destino, Operando1, Operando2    ;Comentário
```

- **Label** é um nome para referência de endereço pelo compilador
- **Mnemônico** é a instrução em si.
- **Operandos** são os registradores, dados , endereços manipulados, etc ...

Ex: `ADD R2, R4, R5` ;R2 = R4 + R5

Cortex-M ISA:

Origem dos operandos



- **Op1** é sempre um registrador.
- **Op2** pode ser um registrador, ou um registrador rotacionado, ou um valor imediato.

*Ref. **

UTFPR

Cortex-M ISA:

Origem dos operandos

Exemplos para **Op2**:

```
ADD   R2, R4, R5          ;Op2 é um registrador (R5)
ADD   R2, R4, R5,LSL #2   ;Op2 é um registrador rotacionado
                          ;R5 Logic Shifted Left 2-bit
                          ;equivalente a multiplicar por 4
ADD   R2, R4, #0xFF       ;Op2 é um valor imediato 0xFF
```

Obs: #valor ⇨ #-15, #0xCFC, #2_1010 …

UTFPR

Cortex-M ISA:

Possibilidades de deslocamentos

- **ASR #n** $1 \leq n \leq 32$
- **LSL #n** $1 \leq n \leq 31$
- **LSR #n** $1 \leq n \leq 32$
- **ROR #n** $1 \leq n \leq 31$
- **RRX**

Base	Prefixo	Sufixo	Exemplo
Binário	2_	Y or y	2_1001 ou 1001Y
Decimal:	-	T or none	1234T or 1234
Hexadecimal:	0x or 0X	H or h	1234H or 0x1234
Octal:	(zero)	Q, q, O, or o	0777 or 777q or 777Q or 777o

https://developer.arm.com/documentation/101407/0537/Debugging/Expressions/Constants

Cortex-M ISA:

Constantes no Op2

- Podem variar entre **3** a **16-bit**
- Constantes de 8-bit são mais comuns podem:
  - Ser deslocadas „. (0 a 31-bit)
  - Padrões: **0xXYXYXYXY**, **0x00XY00XY** ou **0xXY00XY00**

- Exemplos:

```
MOVT    R0,#0xACFC          ;16-bit cte
MOV     R0,#0xE100E100      ;8-bit padrão
LSL     R0,#27              ;5-bit cte
```

UTFPR

Cortex-M ISA:

Flags: N,Z,C,V

- Algumas instruções podem possuir uma variação que acrescenta o sufixo **S** ⇨ "*set the flags*"

- Exemplos:

```
ADD    R2, R3, R5   ;Ñ afeta os flags
ADDS   R0, R3, R5   ;Afeta os flags (N,Z,C e V)
                    ;após executar a soma.
```

UTFPR

Cortex-M ISA:

Flags: N,Z,C,V

- Algumas instruções podem possuir uma variação que acrescenta um **condicional**. Um Bloco **IT** normalmente apresenta os condicionais.

- Exemplos:

```
ADDS   R0, R3, R5  ;Afeta os flags (N,Z,C e V)
ITT    EQ          ;Bloco IT com 2 instruções
ADDEQ  R2, R3, R5  ;Se Z=1, executa
ADDSEQ R2, R3, R5  ;Se Z=1, executa
```

UTFPR

Cortex-M ISA:

Bloco IT

- Pode executar uma sequência de 1 a 4 instruções
- IT**xyz** **COND**
- onde **x**, **y**, **z** são **T** ou **E** (**T**hen ou **E**lse) e correspondem a uma instrução dentro do bloco.
- Exemplos:

```
ITTE    EQ          ;Bloco IT com 3 instruções
ADDEQ   R2, R3, R5  ;Se Z=1, executa
SUBEQ   R7,R8       ;Se Z=1, executa
ADDNE   R2, R3, R5  ;Se Z=1, não executa
```

Cortex-M ISA:

Bloco IT

Observações:

- Um salto **BCC** condicional pode estar dentro de um bloco **IT**.
- As instruções: IT, CBZ, CBNZ, CPSIE, CPSID não podem estar presentes em um bloco **IT**.
- Instruções que alteram o **PC** só podem aparecer por último num bloco **IT**.

UTFPR

Cortex-M ISA:

Condicionais - sufixos

Sufixo	*Flags*	Significado	
EQ	Z = 1	Equal - Igual	S/U
NE	Z = 0	Not equal - diferente	S/U
CS or HS	C = 1	Higher or same - maior ou igual	U
CC or LO	C = 0	Lower - menor	U
MI	N = 1	Negative - negativo	S
PL	N = 0	Positive or zero - positivo ou zero	S
VS	V = 1	Overflow - estouro de bit	S
VC	V = 0	No overflow - sem estouro	S
HI	C = 1 e Z = 0	Higher - maior	U
LS	C = 0 ou Z = 1	Lower or same - menor ou igual	U
GE	N = V	Greater than or equal - maior que ou igual	S
LT	N != V	Less than - menor que	S
GT	Z = 0 e N = V	Greater than - maior que	S
LE	Z = 1 ou N != V	Less than or equal - menor que ou igual	S
AL	*any value*	Always - Sempre	S/U

UTFPR

Cortex-M ISA:

Condicionais - sufixos

Observações:

- Instruções são **16-bit** ou **32-bit**.
- Isso implica que o **PC** sempre será par. Os projetistas resolveram utilizar o **b0** do **PC** como sinalizador do "*interworking*". Cortex-M não tem "*interworking*" portanto o **b0** sempre tem que ser **1**. E este valor é armazenado no flag **T** no **xPSR**.
- "BX", "BLX", "POP{PC}", "MOV PC, LR", "ADD PC, PC, R1"

UTFPR

Cortex-M ISA:

Texas ISA (221pgs):
http://www.elf74.daeln.com.br/Pdfs/CortexM_InstructionSet.pdf

ARM ISA cap.11:
http://www.elf74.daeln.com.br/Pdfs/Arm_Asm_User_Guide_v5.06.pdf

Cortex-M ISA:

Aritméticas

Instrução	Descrição	Operação
ADD Rd, Rn, Op2	Add a register to Op2	Rd = Rn + Op2
ADC Rd, Rn, Op2	Add a register to Op2 and to Carry	Rd = Rn + Op2 + CY
SUB Rd, Rn, Op2	Subtract from a register the Op2	Rd = Rn - Op2
SBC Rd, Rn, Op2	Subtract from a register the Op2 and the Borrow (negation of Carry)	Rd = Rn - Op2 - /CY
RSB Rd, Rn, Op2	Subtract from Op2 a register	Rd = Op2 - Rn
RSC Rd, Rn, Op2	Subtract from Op2 a register and the Borrow	Rd = Op2 - Rn - /CY
MOV Rd, Op2	Move to Rd from Op2 (put a copy of Operand2 into Rd)	Rd = Op2
MVN Rd, Op2	Move to Rd /Op2	Rd = /Op2
MOVT Rd,<imm16>	Move to Rd[31:16] from imm16. Lower bits of Rd are unaffected	Rd[31:16] = imm16

Cortex-M ISA:

Comparação e teste

Instrução	Descrição
CMP Rn, Op2	Compare: Subtract from Rn the Op, discard result, change flags
CMN Rn, Op2	Compare negative: Add Rn to Op2, discard result, change flags
TST Rn, Op2	Test: Rn AND Op2, discard result, change flags
TEQ Rn, Op2	Test equivalence: Rn EOR Op2, discard result, change flags

UTFPR

Cortex-M ISA:

Instruções de Deslocamento no Op2

Instrução	Descrição	#imm Sh
ASR Rd, Rn, Sh	Arithmetic Shift Right (preserves signal)	1..32
LSL Rd, Rn, Sh	Logical Shift Left	0..31
LSR Rd, Rn, Sh	Logical Shift Right	1..32
ROR Rd, Rn, Sh	Rotate Right	0..31
RRX Rd, Rn	Rotate Right Extended	
Sh	Pode ser valor imediato de 5-bit (1 a 32) ou (0 a 31)	
	Pode ser os 8-bit(0 a 255) menos significativos de um Registrador geral	

UTFPR

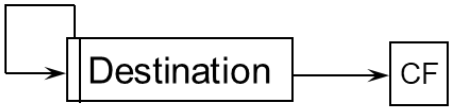
Cortex-M ISA:

Instruções de Deslocamento

LSL : Logical Left Shift

Multiplication by a power of 2

ASR: Arithmetic Right Shift

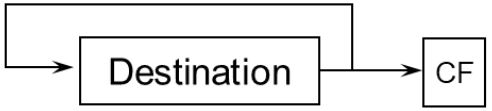


Division by a power of 2, preserving the sign bit

LSR : Logical Shift Right

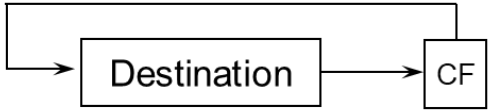
Division by a power of 2

ROR: Rotate Right



Bit rotate with wrap around from LSB to MSB

RRX: Rotate Right Extended



Single bit rotate with wrap around from CF to MSB

*Ref. **

UTFPR

Cortex-M ISA:

Multiplicação e Divisão

Instrução	Descrição	Operação
Instructions that multiply 32-bit by 32-bit resulting 32-bit with wrapping (LSW is preserved and higher bits are discarded)		
MUL Rd, Rm, Rs	Multiply	Rd = Rm * Rs
MLA Rd, Rm, Rs, Rn	Multiply and accumulate	Rd = Rm * Rs + Rn
MLS Rd, Rm, Rs, Rn	Multiply and subtract	Rd = Rm * Rs - Rn
Long multiplication: multiply 32-bit by 32-bit resulting 64-bit		
UMULL RdLo, RdHi, Rm, Rs	Unsigned long multiply	RdHi:RdLo = unsigned(Rm*Rs)
UMLAL RdLo, RdHi, Rm, Rs	Unsigned long multiply and accumulate	RdHi:RdLo = unsigned(RdHi:RdLo + Rm*Rs)
UMAAL RdLo, RdHi, Rm, Rs	Unsigned long multiply and accumulate double	RdHi:RdLo = unsigned(RdHi+RdLo + Rm*Rs)
SMULL RdLo, RdHi, Rm, Rs	Signed long multiply	RdHi:RdLo = signed(Rm*Rs)
SMLAL RdLo, RdHi, Rm, Rs	Signed long multiply and accumulate	RdHi:RdLo = signed(RdHi:RdLo + Rm*Rs)

Instrução	Descrição	Operação
UDIV Rd, Rn, Rm	Unsigned divide	Rd = Rn / Rm
SDIV Rd, Rn, Rm	Signed divide	Rd = Rn / Rm

UTFPR

Cortex-M ISA:

Conjunto de bits

Instrução	Descrição	Operação
BFC Rd,#<lsb>,#<width>	Bit field clear	clear Rd[(width+lsb-1)..lsb], others unchanged
BFI Rd, Rn,#<lsb>,#<width>	Bit field insert. Copy the <width> LSb of Rn to Rd	Rd[(width+lsb-1)..lsb] = Rn[(width-1)..0]
SBFX Rd, Rn,#<lsb>,#<width>	Signed bit field extract.	Copy bitfield from Rn to LSb of Rd and sign extend.
UBFX Rd, Rn,#<lsb>,#<width>	Unsigned bit field extract.	Copy bitfield from Rn to LSb of Rd and zero extend.

- Um "*bit field*" é um conjunto de bits dentro de um registrador.
- "*witdh*"=tamanho, n. de bits do conjunto (1 a 32)
- "lsb"=posição do bit menos significativo do conjunto.

UTFPR

Cortex-M ISA:

Extensão S/U

Instrução	Descrição	Operação
SXTB Rd, Rm	Sign extend a byte	extract bits [7:0] and sign extend to 32 bits
UXTB Rd, Rm	Zero extend a byte	extract bits [7:0] and zero extend to 32 bits
SXTH Rd, Rm	Sign extend a half word	extract bits [15:0] and sign extend to 32 bits
UXTH Rd, Rm	Zero extend a half word	extract bits [15:0] and zero extend to 32 bits

UTFPR

Cortex-M ISA:

Acesso a Memória

Instrução	Descrição
LDRB	Load byte. Read a byte from memory, zero extend and store in register.
LDRH	Load half word. Read a half-word from memory, zero-extend and store in register.
LDR	Load register. Read a word from memory and store in a register.
LDRD	Load double. Read a double word form memory and store in two registers.
STRB	Store byte. Store the LSB of a register into memory (byte wide)
STRH	Store half-word. Store the lower half of a register into memory (16-bit wide)
STR	Store register. Store a register into memory (32-bit wide)
STRD	Store double. Store the two registers into memory (64-bit wide)
LDM	Load multiple. Read several (up to 16) registers from memory.
STM	Store multiple. Store several (up to 16) registers into memory.
PUSH	Store registers in current stack (main or process)
POP	Restore registers in current stack (main or process)
LDREX	Load Register Exclusive
LDREXB	Load Byte to Register Exclusive
LDREXH	Load Half Word to Register Exclusive
STREX	Store Register Exclusive
STREXB	Store Register Exclusive to Byte
STREXH	Store Register Exclusive to Half Word
CLREX	Clear local processor exclusive tag
LDRT	Load unprivileged. With variants for Byte and HalfWord.
STRT	Store unprivileged. With variants for Byte and HalfWord.

UTFPR

Cortex-M ISA:

Little / Big endian

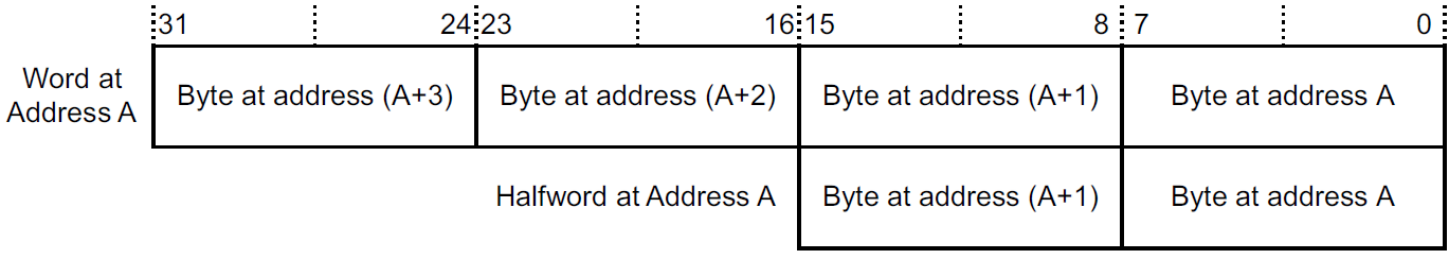


Figure A3-1 Little-endian byte format

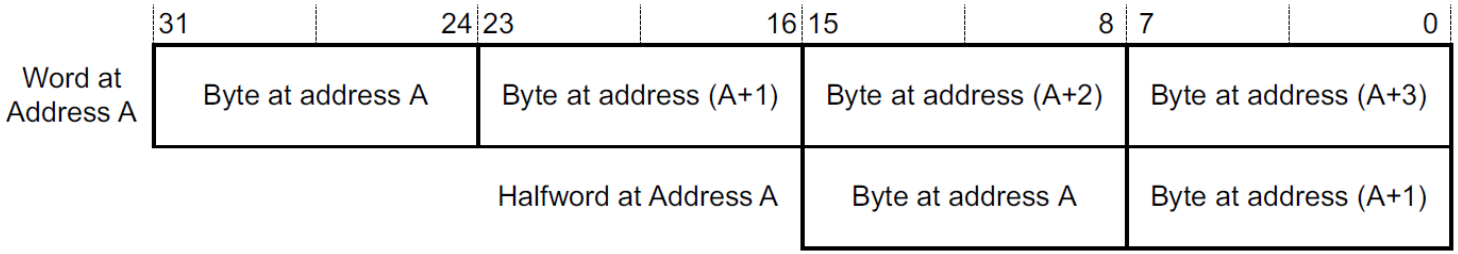


Figure A3-2 Big-endian byte format

*Ref. **

UTFPR

Cortex-M ISA:

Little endian

MSByte	MSByte-1	LSByte+1	LSByte
Word at address A			
Halfword at address (A+2)		Halfword at address A	
Byte at address (A+3)	Byte at address (A+2)	Byte at address (A+1)	Byte at address A

Figure A3-3 Little-endian memory system

*Ref. **

UTFPR

Cortex-M ISA:

LDR – Load

Exemplos:

```
LDR   Rt,[Rn]                ;offset -255 a 2095
LDR   Rt,[Rn,#offset]        ;pre. e suf. -255 a 255
LDRB  Rt,[Rn,#offset]
LDRH  Rt,[Rn,#offset]
LDR   Rt,[Rn,Rm]             ;offset com registrador
LDRB  Rt,[Rn,Rm]
LDRH  Rt,[Rn,Rm]
LDR   Rt,[Rn, Rm, LSL #2]    ;address = Rn + Rm<<2
```

UTFPR

Cortex-M ISA:

LDR – Modos

Modo	Exemplo	Resultado	Alteração Op2
Pre com writeback (!)	LDR R0,[R1,#4] !	R0 = [R1 + 4]	R1 = R1 + 4
	LDR R0,[R1,R2] !	R0 = [R1+R2]	R1 = R1 + R2
	LDR R0,[R1,R2,LSL #2] !	R0 = [R1 + (R2 << 2)]	R1 = R1 + R2 << 2
Pré-indexado	LDR R0,[R1,#4]	R0 = [R1 + 4]	sem alteração
	LDR R0,[R1,R2]	R0 = [R1+R2]	sem alteração
	LDR R0,[R1,R2,LSL #2]	R0 = [R1 + (R2 << 2)]	sem alteração
Pós-indexado	LDR R0,[R1],#4	R0 = [R1]	R1 = R1 + 4
	LDR R0,[R1],R2	R0 = [R1]	R1 = R1 + R2
	LDR R0,[R1],R2,LSL #2	R0 = [R1]	R1 = R1 + R2 << 2

Cortex-M ISA:

LDR – Load

Exemplos:

```
LDR Rd,[Ri,#cte]!      ;pré ⇨ 1:calcula, 2:acessa
LDR Rd,[Ri,Rm]!        ;atualiza Ri
LDR R2,[R3,#4]!
LDR Rd,[Ri],#cte       ;pós ⇨ 1:acessa, 2:calcula
LDR Rd,[Ri],Rm         ;atualiza Ri
LDR R2,[R3],#4
```

UTFPR

Cortex-M ISA:

LDRSB e LSRSH

- Leitura de ***byte*** ou ***half word***:

```
LDRSB Rt,[Rn,#off]     ;formatos
LDRSH Rt,[Rn,#off]
LDRSB Rt,[Rn,Rm]
LDRSH Rt,[Rn,Rm]
```

UTFPR

Cortex-M ISA:

LDR (Relativo ao PC)

- Leitura de dados da memória usando o **PC**:

```
LDR Rt,[PC,#100]        ;formatos
LDR Rt, label
```

- **Obs**: usual na utilização de **tabelas**.

UTFPR

Cortex-M ISA:

LDRD

- Leitura de dados de **64-bit** da memória(**2xRegs**):

```
LDRD Rt, Rt2, label    ;Relativo ao PC
LDRD Rt, Rt2, [...]    ;Mesmos modos end. do LDR
```

- **Obs**:

  64-bit ⇨ Rt2:Rt

  Rt – *least significant word* (LSW)

  Rt2 – *most significand word* (MSW)

UTFPR

Cortex-M ISA:

STR

- Escreve dados na memória:

STR

STRB

STRH

- **Obs**: Mesmos modos do **LDR**

UTFPR

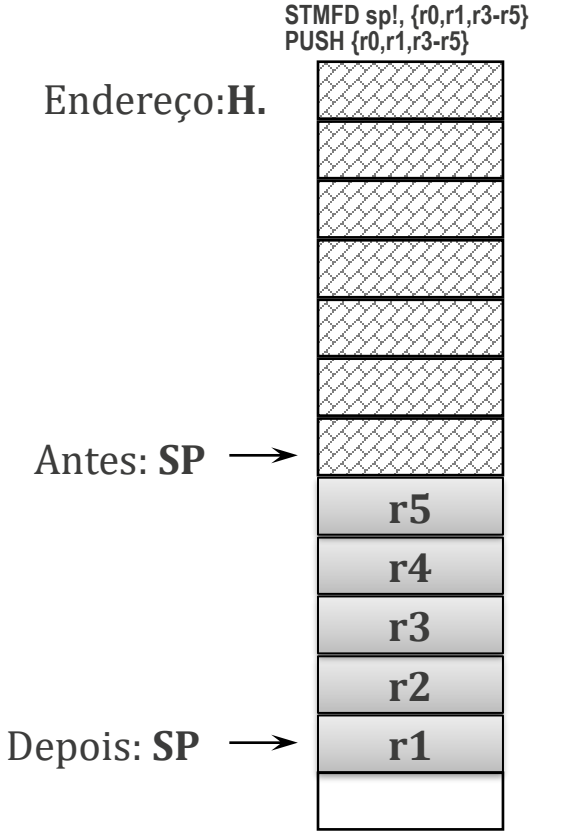
Cortex-M ISA:

PUSH e POP

- Transferência de registradores para a pilha:

```
PUSH {R0-R7}
POP {R1,R3-R6}
```

- **Obs**: Caso o **PC** esteja na lista, um salto será executado, quando terminar o **POP**. O **T-bit** do **xPSR** vem do **b0** do **PC**, deve ser **1** Cortex-M para que não gere exceção.



FD : Full Descending

*Ref. **

UTFPR

Cortex-M ISA:

LDM e STM

- Transferência de múltiplos registradores:

```
LDM R0,{R0,R1,R2}
LDM R1!,{R2-R7}
STM R0!,{R2-R4}
```

- **Obs**: A lista sempre será na ordem ascendente. Na memória, os registradores de índice mais baixos ficam nos endereços inferiores. **Rn ≠ PC,** STM - "*reglist*" não pode conter o **SP**, LDM - "*reglist*" não pode conter o **SP** se contiver o **LR,** "*reglist*" não pode conter **Rn** se existir um sufixo de *writeback*. No bloco **IT** devem ser a última instrução.

UTFPR

Cortex-M ISA:

LDM e STM

- Modificadores: LDMIA, LDMFD e STMIA, STMEA
- IA = *Increment After* – Padrão

  1 - acessa a memória

  2 - incrementa o endereço
- DB = *Decrement Before*

  1 - decrementa o endereço

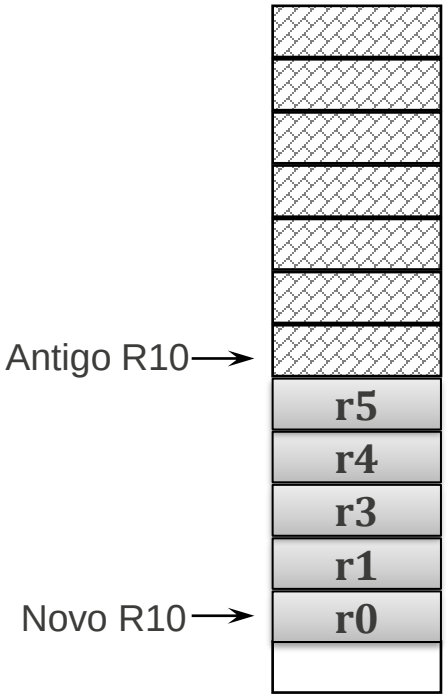
  2 - acessa a memória
- EA = *Empty Ascending*
- FD = *Full Descending*

UTFPR

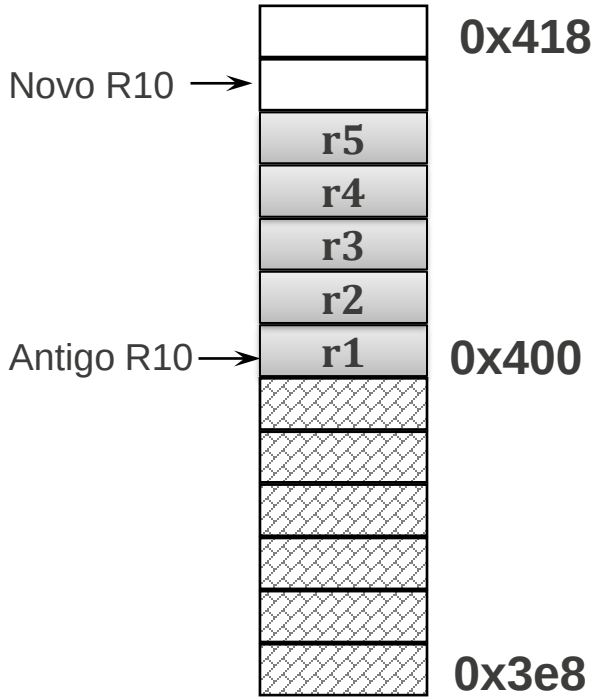
Cortex-M ISA:

LDM e STM

STMDB r10!, {r0,r1,r3-r5}
STMFD r10!, {r0,r1,r3-r5}



STMIA R10!, {r0,r1,r3-r5}
STMEA R10!, {r0,r1,r3-r5}



DB : Decrement Before
FD : Full Descending

Executar

IA : Increment After
EA : Empty Ascending

*Ref. **

UTFPR

Cortex-M ISA:

WRITE BUFFER

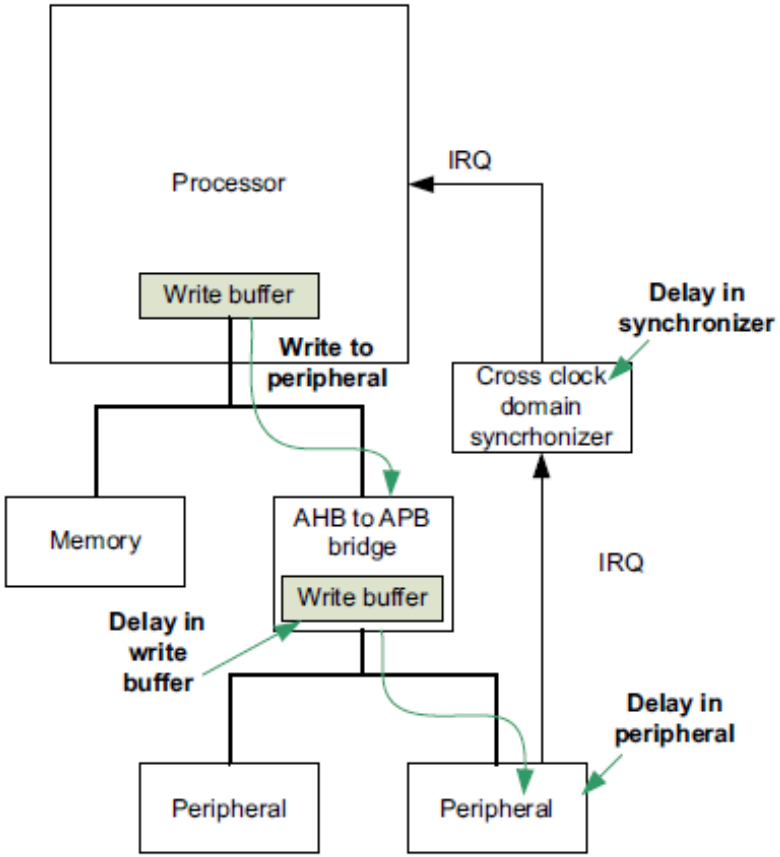


Figure 19 Various sources of delay during interrupt disabling

*Ref. **

UTFPR

Cortex-M ISA:

WRITE BUFFER

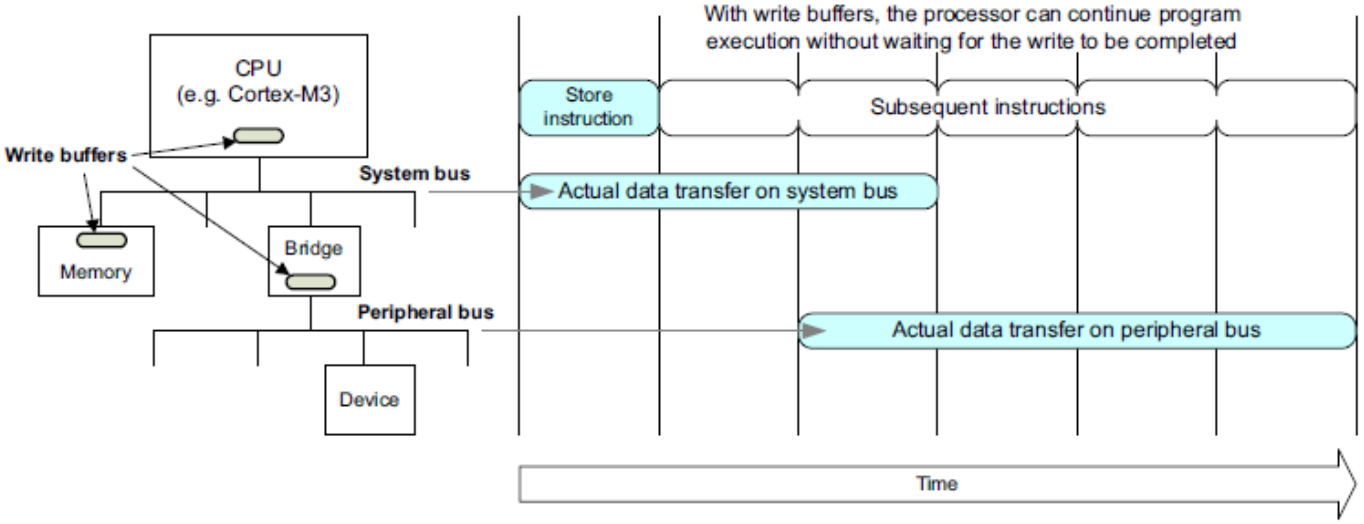


Figure 5 Locations where write buffers might be used and their effect

*Ref. **

Cortex-M ISA:

C vs Assembly

Tipo	Tamanho
char	8 bits
short	16 bits
int	32 bits
unsigned	32 bits
long	32 bits
long long	64 bits
*	32 bits
[ ]	32 bits
struct	ptr = 32 bits
float	32 bits
double	64 bits
enum	=int

UTFPR

Cortex-M ISA:

Saltos - Branches

Instrução	Descrição	Distância
B <label>	Branch to target address.	+/−16 MB
Bcc <label>	Conditional Branch outside IT block	+/−1 MB
	Conditional Branch inside IT block	+/−16 MB
CBNZ Rn,<label>	Compare and Branch on Nonzero.	0-126 B
CBZ Rn,<label>	Compare and Branch on Zero.	
BL <label>	Call a subroutine.	+/−16 MB
BLX <register>	Call a subroutine, optionally change instruction set.	Any
BX <register>	Branch to target address, optionally change instruction set.	Any
TBB [Rn,Rm,LSL #1]	TBB: Table Branch, byte offsets. Base addr, index, optional #1	0-510 B
TBH [Rn,Rm,LSL #1]	TBH: Table Branch, halfword offsets.	0-131070 B

- **Obs**: O **T-bit** do **xPSR** vem do **b0** do **PC**, deve ser **1** Cortex-M para que não gere exceção.
- **MOV PC,... – ADD PC, ... – POP {PC}** – quando executadas também executam saltos!
- Levam de **1** a **4** *clocks* e causam "*flush*" no *pipeline*.

UTFPR

Cortex-M ISA:

Miscellaneous

Instrução	Descrição
CPSID	Change Processor State, Disable Interrupts.
CPSIE	Change Processor State, Enable Interrupts.
DMB	Data Memory Barrier.
DSB	Data Synchronization Barrier.
ISB	Instruction Synchronization Barrier.
MRS	Move to Register from Special Register.
MSR	Move to Special Register from Register.
NOP	No Operation.
SVC	Supervisor Call.
WFI	Wait for Interrupt.

UTFPR

Cortex-M ISA:

Miscellaneous

Instruction	CMSIS function
CPSIE I	`void __enable_irq(void)`
CPSID I	`void __disable_irq(void)`
CPSIE F	`void __enable_fault_irq(void)`
CPSID F	`void __disable_fault_irq(void)`
ISB	`void __ISB(void)`
DSB	`void __DSB(void)`
DMB	`void __DMB(void)`
REV	`uint32_t __REV(uint32_t int value)`
REV16	`uint32_t __REV16(uint32_t int value)`
REVSH	`uint32_t __REVSH(uint32_t int value)`
RBIT	`uint32_t __RBIT(uint32_t int value)`
SEV	`void __SEV(void)`
WFE	`void __WFE(void)`
WFI	`void __WFI(void)`

Cortex-M ISA:

Miscellaneous

Registrador Especial	Acesso	CMSIS function
PRIMASK	Read	`uint32_t __get_PRIMASK (void)`
	Write	`void __set_PRIMASK (uint32_t value)`
FAULTMASK	Read	`uint32_t __get_FAULTMASK (void)`
	Write	`void __set_FAULTMASK (uint32_t value)`
BASEPRI	Read	`uint32_t __get_BASEPRI (void)`
	Write	`void __set_BASEPRI (uint32_t value)`
CONTROL	Read	`uint32_t __get_CONTROL (void)`
	Write	`void __set_CONTROL (uint32_t value)`
MSP	Read	`uint32_t __get_MSP (void)`
	Write	`void __set_MSP (uint32_t TopOfMainStack)`
PSP	Read	`uint32_t __get_PSP (void)`
	Write	`void __set_PSP (uint32_t TopOfProcStack)`

www.keil.com/pack/doc/cmsis/Core/html/group__intrinsic__CPU__gr.html

UTFPR

Cortex-M ISA:

Miscellaneous

- **CPS**: *Change Processor State*, altera o **PRIMASK**.

```
CPSID i                 ;desabilita IRQs
CPSIE i                 ;habilita IRQs
```

- **DMB:** Usada antes de se alterar o vetor de interrupções, cria barreira.
- **ISB:** Usada após a alteração de código
- **ISB:** Usada depois de terminada as alterações no mapa de memória.
- **MSR:** *Move to Special Register from Register*
- **MRS:** *Move to Register from Special Register*
- ***Especial Register: APSR, IPSR, EPSR, IEPSR, EAPSR, PSR, MSP, PSP, PRIMASK ou CONTROL***

```
MSR special_reg, Rn     ;Rn diferente de SP ou PC
MRS Rd, especial_reg    ;Rd diferente de SP ou PC
```

UTFPR

Cortex-M ISA:

Miscellaneous

- **NOP**: *No operation, pode fazer o micro esperar 1 clock*. Pode ser removida do *pipeline*. Outro uso: alinhamento de código.

```
NOP                 ; 1 ciclo de máquina em espera
```

- **SVC:** *Supervisor call* (S.O.)

```
SCV #imm            ;Gera uma chama de rotina de exceção.
```

- **WFI:** Usada para colocar o micro em *sleep mode*.

```
WFI                 ;Sleep até IRQ ou Exceção ou Debug
```

AAPCS

Antigo ATPCS

- **ATPCS**: *ARM Thumb Procedure Call Standard*.
- **AAPCS**: *ARM Architecture Procedure Call Standard*.

ATPCS:
http://www.elf74.daeln.com.br/Pdfs/ATPCS.pdf

AAPCS:
http://www.elf74.daeln.com.br/Pdfs/AAPCS.pdf

UTFPR

AAPCS

Resumo

- Passagem de **parâmetros** para uma função:

- Primeiros parâmetros em **R0**, **R1**, **R2** e **R3**;

- Demais parâmetros pela pilha.

- Retorno da função:

- **R0** (32 bits)

- **R1**:**R0** (64 bits)

- Alinhamento da pilha deve ser 64 bits:

- **PUSH**/**POP** de número par de registradores antes/depois de chamadas de funções, caso seja necessário salvar e restaurar.

UTFPR

AAPCS

Resumo

```
 THUMB
    IMPORT Conta
    IMPORT A
    AREA |.text|,CODE,READONLY,ALIGN=2
    EXPORT Start

Start
    LDR R2, =A      ;R2 = &A
    MOV R0, #2
    STR R0, [R2]    ;A=2
Loop
    MOV R0, #5      ;Função recebe
                    ;parâmetros em R0 a R3
    BL Conta        ;Chama a função que irá
                    ;multiplicar A variável
                    ;global pelo parâm. 1
                    ;que está em R0 e salva
                    ;resultado em R0


    NOP

    ALIGN
    END
```

```
#include <stdint.h>

uint32_t A;

//Multiplica o parâmetro pelo fator A
uint32_t Conta(uint32_t parametro)
{
        uint32_t resultado;
        resultado = A * parametro;
        return resultado;
}
```

UTFPR

AAPCS

Resumo

```
 THUMB
    AREA DATA, ALIGN=2
    EXPORT A [DATA,SIZE=4]
A   SPACE 4
    AREA |.text|,CODE,READONLY,ALIGN=2
          EXPORT Conta
    EXPORT A

;Retorna a multiplicação de A que está na
;memória pelo parâmetro recebido em R0 e
;retorna em R0 para a AAPCS
Conta
    LDR R2, =A     ;R2 = &A
    LDR R1, [R2]   ;R1 = [A]
          MUL R0, R0, R1
    BX LR

    ALIGN
    END
```

```
#include <stdint.h>

uint32_t Conta(uint32_t parametro);
extern uint32_t A;

int main (void)
{
  uint32_t resultado;
  A = 2;
  while (1)
        {
          resultado = Conta(5);
          A = resultado;
        }
}
```

UTFPR

Referências:

Extras:

Exercício Fixação Assembly:
http://www.elf52.daeln.com.br/Labs/Laboratorio_AP0.pdf

Simular as instruções de LDM e STM
slides desta aula ...

Simular modos da pilha
slides desta aula ...

* *Refs* ↔ Renesas.com, Pixabay.com, wikimedia.org, flickr, community.arm.com, Undergraduated course Renesas (Prof. Douglas P. B. Renaux e Robson Linhares), ytchannel Gustavo W. Dernardin, *ARMv7-M Architecture Reference Manual,*