Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Eletrônica (DAELN)

SISTEMAS EMBARCADOS

Conjunto de Instruções Thumb2

Prof. André Schneider de Oliveira

andreoliveira@utfpr.edu.br

Cortex-M3

- Utiliza a versão de arquitetura ARMv7-M
  - Conjunto de instruções Thumb2
    - Emprega instruções de 16 e 32 bits

- Material de apoio:
  - Cortex-M3/M4F Instruction Set
  - ARM ® and Thumb ® -2 Instruction Set (quick reference)
  - Livro: The Definitive Guide to The ARM Cortex-M3

Unified Assembler Language (UAL)

- Padroniza as pseudo-instruções a serem resolvidas pelo montador (assembler):
  - POP, PUSH
  - Relaxamento nas definições de Rd e Rn
    - Rd : registro de destino
    - Rn : registo de origem (parametro)
  - Obrigatoriedade do sufixo S para alterar **flags**

- Especificadores de largura da instrução
  - .W força instrução de 32 bits (Wide)
  - .N força instrução de 16 bits (Narrow)

Instruções de Carga

Mnemonic	Operands	Brief description	Flags
MOV, MOVS	Rd, Op2	Move	N,Z,C
MOVT	Rd, #imm16	Move Top	-
MOVW, MOV	Rd, #imm16	Move 16-bit constant	N,Z,C
MRS	Rd, spec_reg	Move from special register to general register	-
MSR	spec_reg, Rm	Move from general register to special register	N,Z,C,V
MVN, MVNS	Rd, Op2	Move NOT	N,Z,C

- Carga de registrador
  MOV Rd, <Op2>
  Rd := <Op2>

- Carga de registrador negada
  MVN Rd, <Op2>
  Rd := /<Op2>

- <Op2> = registrador (deslocado) ou constante

Instruções Lógicas e Aritméticas

Mnemonic	Operands	Brief description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn, #imm12	Add	N,Z,C,V
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right	N,Z,C
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N,Z,C
CMN, CMNS	Rn, Op2	Compare Negative	N,Z,C,V
CMP, CMPS	Rn, Op2	Compare	N,Z,C,V
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C
LSL, LSLS	Rd, Rm, <Rs|#n>	Logical Shift Left	N,Z,C
LSR, LSRS	Rd, Rm, <Rs|#n>	Logical Shift Right	N,Z,C
MLA	Rd, Rn, Rm, Ra	Multiply with Accumulate, 32-bit result	-
MLS	Rd, Rn, Rm, Ra	Multiply and Subtract, 32-bit result	-
MUL, MULS	{Rd,} Rn, Rm	Multiply, 32-bit result	N,Z
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N,Z,C

Instruções Lógicas e Aritméticas

Mnemonic	Operands	Brief description	Flags
ROR, RORS	Rd, Rm, <Rs|#n>	Rotate Right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate Right with Extend	N,Z,C
RSB, RSBS	{Rd,} Rn, Op2	Reverse Subtract	N,Z,C,V
SBC, SBCS	{Rd,} Rn, Op2	Subtract with Carry	N,Z,C,V
SDIV	{Rd,} Rn, Rm	Signed Divide	-
SMLAL	RdLo, RdHi, Rn, Rm	Signed Multiply with Accumulate (32 x 32 + 64), 64-bit result	-
SMULL	RdLo, RdHi, Rn, Rm	Signed Multiply (32 x 32), 64-bit result	-
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn, #imm12	Subtract	N,Z,C,V
TEQ	Rn, Op2	Test Equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
UDIV	{Rd,} Rn, Rm	Unsigned Divide	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply with Accumulate (32 x 32 + 64), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned Multiply (32 x 32), 64-bit result	-

Instruções Lógicas e Aritméticas

- Formato
  - Três operandos explícitos (Rd, Rn, <Op2>)
    ADD R0, R1, R2
    R0 := R1 + R2
  - Dois operandos explícitos e um implícito (Rd, <Op2>)
    ADDS R0, R1 (flags)
    R0 := R0 + R1
  - válido apenas para algumas instruções (UAL).
  - Dois operandos (Rn, <Op2>)
    CMP R7, R8

Adição e Subtração

- ADD{S}, ADC{S} Rd, Rn, <Op2> (adição)

  ADD R0, R1, R2
  R0 := R1 + R2

  ADDC R0, R1, R2
  R0 := R1 + R2 + C

- SUB{S}, SBC{S} Rd, Rn, <Op2> (subtração)

  SUB R0, R1, R2
  R0 := R1 - R2

  SBC R0, R1, R2
  R0 := R1 - R2 - C

- RSB{S} Rd, Rn, <Op2> (subtração reversa)

  RSB R0, R1, R2
  R0 := R2 - R1

Operações Lógicas

- AND{S} Rd, Rn, <Op2> – Operação AND
- ORR{S} Rd, Rn, <Op2> – Operação OR
- EOR{S} Rd, Rn, <Op2> – Operação EXCLUSIVE OR
- ORN{S} Rd, Rn, <Op2> – Operação OR NOT
- BIC{S} Rd, Rn, <Op2> – Operação AND NOT (*bit clear*)

- <Op2> é o operando secundário que pode ser uma constante ou um registrador com deslocamento opcional

Comparação

- CMP Rn, <Op2> – afeta flags com Rn - Op2

  (comparação, atualiza flags)

- CMN Rn, <Op2> – afeta flags com Rn + Op2

  (comparação negativa, atualiza flags)

- TST Rn, <Op2> – afeta flags com Rn AND Op2

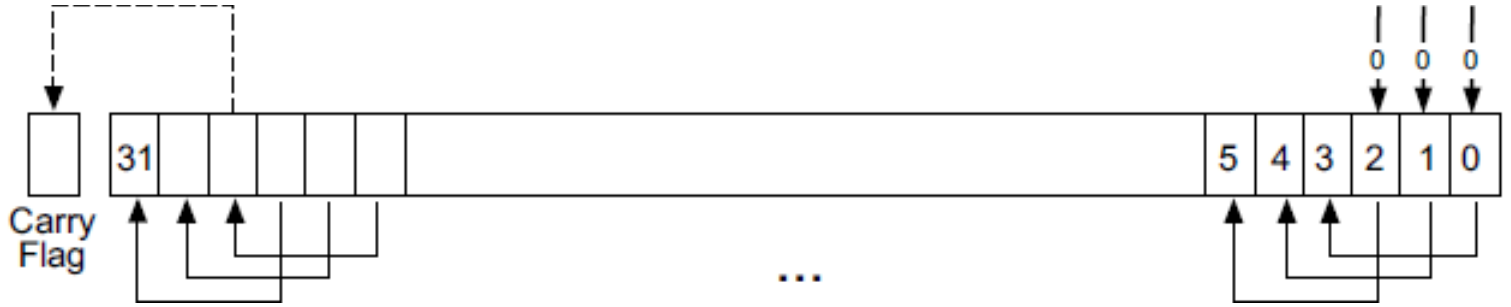
  (testa bits AND bit-a-bit, atualiza flags)

- TEQ Rn, <Op2> – afeta flags com Rn EOR Op2

  (testa equivalência, atualiza flags)

```
CMP     R2, R9
CMN     R0, #6400
TST     R0, #0x3F8
TEQEQ   R10, R9
```

Deslocamento Lógico à Esq. (LSL)



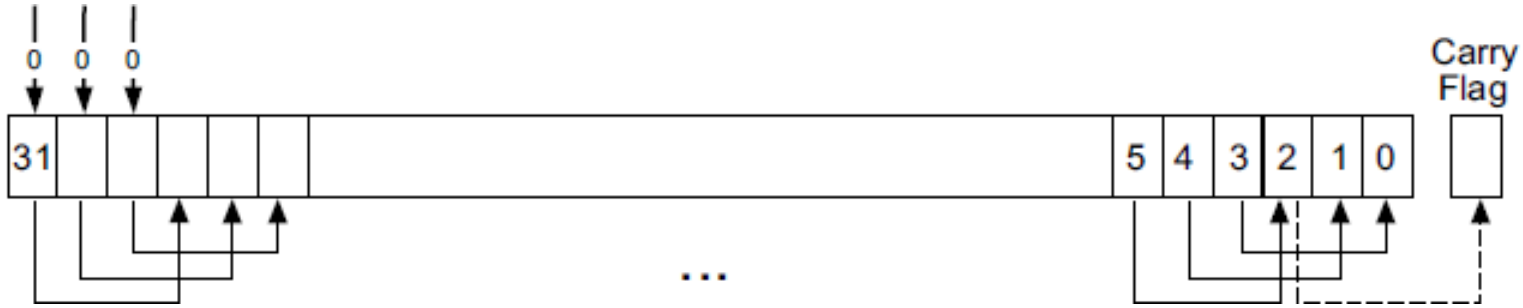
- Variante: LSLS (altera flag C)

- LSLS {Rd}, Rm, Rs
  LSLS {Rd}, Rm, #imm

- Efeito de LSLS Rm, #n é Rm x 2^n
  - n >= 32: todos os bits são zerados
  - n >= 33: todos os bits e o flag C são zerados

```
LSLS   R1, R2, #3  ; Logical shift left by 3 bits with flag update
```

Deslocamento Lógico à Dir. (LSR)

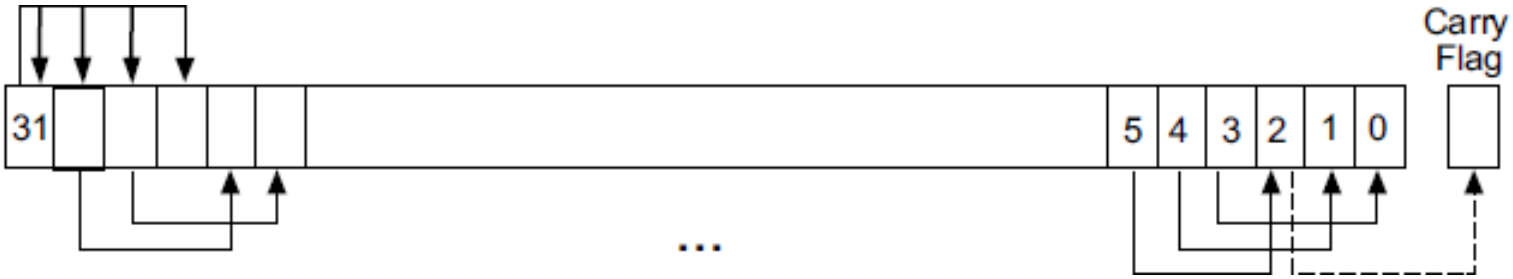


- Variante: LSRS (altera flag C)
  LSRS {Rd}, Rm, Rs
  LSRS {Rd}, Rm, #imm

- Efeito de LSRS Rm, #n é Rm / 2^n
  - n >= 32: todos os bits são zerados
  - n >= 33: todos os bits e o flag C são zerados

```
LSR    R4, R5, #6  ; Logical shift right by 6 bits
```

Deslocamento Aritmético à Dir. (ASR)

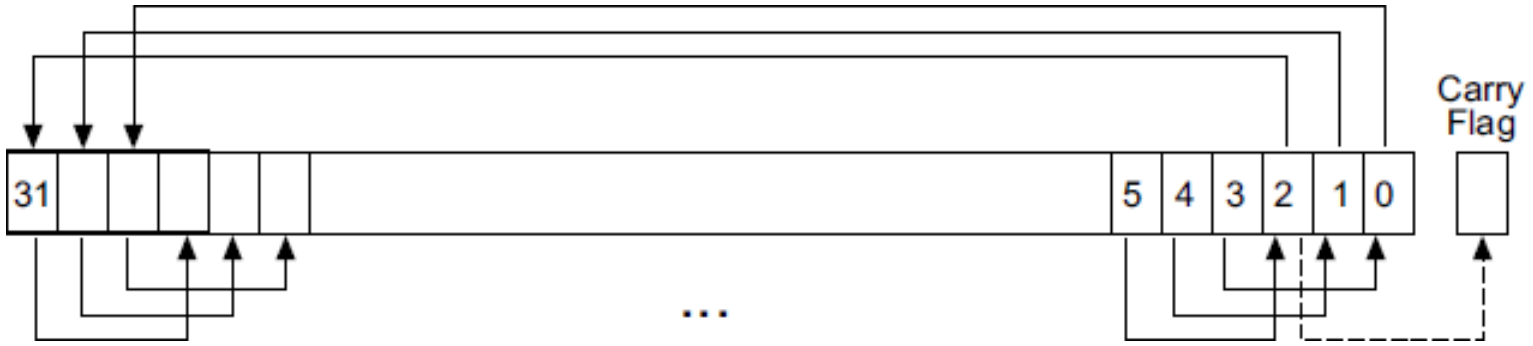


- Variante: ASRS (altera flag C)
  ASRS{Rd}, Rm, Rs
  ASRS{Rd}, Rm, #imm

- Efeito de ASRS Rm, #n é Rm / 2^n
  - n >= 32: todos os bits e o flag C recebem cópia do bit 31

```
ASR    R7, R8, #9  ; Arithmetic shift right by 9 bits
```

Rotação à Direita (ROR)

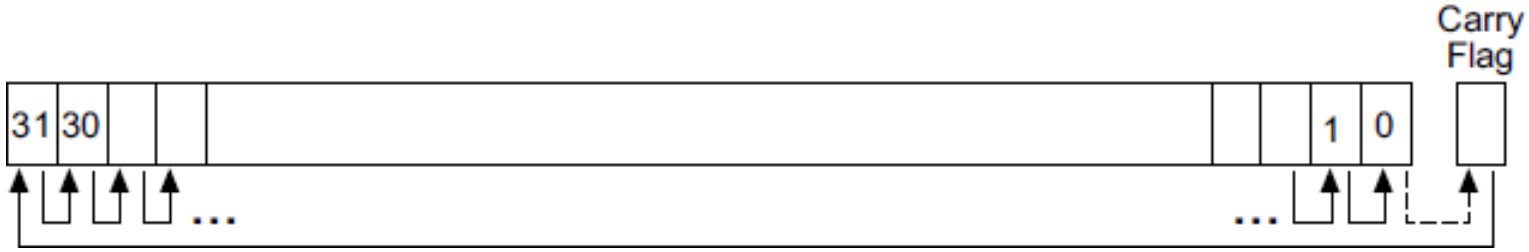


- Variantes: RORS (altera flag C)
  RORS {Rm}, Rm, Rs
  - Apenas dois operandos
  - Apenas registradores baixos (R0 a R7)

- n = 32: o valor não é alterado e flag C := Rm[31]
  n = 33: mesmo efeito que n = 1

```
ROR   R4, R5, R6  ; Rotate right by the value in the bottom byte of R6
```

Rotação à Direita Extendida (RRX)



- Variantes: RRXS (altera flag C)
  RRXS Rd, Rm
  - Inclui o flag C na rotação
  - Rotação de um único bit à direita

```
RRX    R4, R5      ; Rotate right with extend.
```

ARM Barrel Shifter

LSL: Logical Shift Left

Multiplicação por uma potência de 2

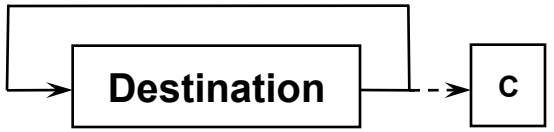
ASR: Arithmetic Right Shift

Divisão por uma potência de 2 preservando o sinal

LSR: Logical Shift Right

Divisão por uma potência de 2

ROR: Rotate Right



Rotação circular de n bits do LSB para o MSB

RRX: Rotate Right Extended

Rotação circular de 1 bit do flag C para o MSB

Operando 2 Flexível <Op2>

- Registrador:
  R4

- Registrador com deslocamento:
  R4, LSL #4
  R5, ASR #1

- Constante:
  #4
  #0xAABB
  #-2

Carga de Constantes de 32 bits

- Instruções MOV.W e MOVT simplificam a carga de constantes que não atendem aos padrões de 8 bits com deslocamento

- MOV Ri, #imm16 (MOV.W Ri, #imm16)
  - carrega os 16 LSB de Ri, zerando os 16 MSB

- MOVT Ri, #imm16
  - carrega os 16 MSB de Ri

imm16: Is a 16-bit immediate constant

Multiplicação (32 bits)

- MUL Rd, Rn, Rm

  Rd := Rn x Rm

  Formato preferido: MUL Rd, Rn

  Apenas registradores baixos (R0 – R7)

  Variante MULS afeta flags N e Z, mas neste caso Rm deve ser o mesmo registrador que Rd

- 32-bits x 32-bits -> 32-bits em 1 ciclo de clock

Multiplicação Acumulada (MLA)

- MLA Rd, Rn, Rm, Ra

  Rd := Ra + Rn x Rm

- MLS Rd, Rn, Rm, Ra

  Rd := Ra - Rn x Rm

- 32-bits x 32-bits -> 32-bits em 2 ciclos de clock

Multiplicação Longa (64 bits)

- Multiplicação longa não sinalizada (Unsigned):
  - UMULL RdLo, RdHi, Rn, Rm

    RdHi:RdLo := Rn x Rm

- Multiplicação acumulada longa não sinalizada:
  - UMLAL RdLo, RdHi, Rn, Rm

    RdHi:RdLo := RdHi:RdLo + Rn x Rm

- 32-bits x 32-bits -> 64-bits em 3 a 5 ciclos de clock

Multiplicação Longa (64 bits)

- Multiplicação longa sinalizada (Signed):
  - SMULL RdLo, RdHi, Rn, Rm

    RdHi:RdLo := Rn x Rm

- Multiplicação acumulada longa sinalizada:
  - SMLAL RdLo, RdHi, Rn, Rm

    RdHi:RdLo := RdHi:RdLo + Rn x Rm

- 32-bits x 32-bits -> 64-bits em 3 a 5 ciclos de clock

Divisão

- Divisão não sinalizada (Unsigned):
  - –UDIV Rd, Rn, Rm
  - Rd := Rn / Rm
- Divisão sinalizada (Signed):
  - –SDIV Rd, Rn, Rm
  - Rd := Rn / Rm
- 32-bits / 32-bits -> 32-bits em 4 ciclos de clock
- Pode gerar exceção na divisão por zero (configurável)

Operações em Bits

Mnemonic	Operands	Brief description	Flags
BFC	Rd, #lsb, #width	Bit Field Clear	-
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	-
SBFX	Rd, Rn, #lsb, #width	Signed Bit Field Extract	-
UBFX	Rd, Rn, #lsb, #width	Unsigned Bit Field Extract	-

Operações em Bits

- Bit-field Clear:
  - Zera #w bits a partir de #lsb em Rd

BFC Rd, #lsb, #w

- Bit-field Insert:
  - Copia #w bits de Rn:0 para Rd:#lsb

BFI Rd, Rn, #lsb, #w

Operações em Bits

- Signed Bit-field Extract:
  - Copia #w bits a partir de #lsb para o bit 0 de Rd e estende sinal

SBFX Rd, Rn, #lsb, #w

- Unsigned Bit-field Extract:
  - Copia #w bits a partir de #lsb para o bit 0 de Rd e estende com zeros

UBFX Rd, Rn, #lsb, #w

Operações de Extensão

Mnemonic	Operands	Brief description	Flags
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	-
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	-
UXTB	{Rd,} Rm {,ROR #n}	Zero extend a byte	-
UXTH	{Rd,} Rm {,ROR #n}	Zero extend a halfword	-

Operações de Extensão

- Signed Extend Byte to Word:
  SXTB Rd, Rm // Rd := sign extend Rm[7:0]
- Unsigned Extend Byte to Word :
  UXTB Rd, Rm // Rd := zero extend Rm[7:0]
- Signed Extend Halfword to Word:
  SXTH Rd, Rm // Rd := sign extend Rm[15:0]
- Unsigned Extend Halfword to Word:
  UXTH Rd, Rm // Rd := zero extend Rm[15:0]
- Apenas registradores baixos (R0 a R7)

Operações de Reversão

Mnemonic	Operands	Brief description	Flags
RBIT	Rd, Rn	Reverse Bits	-
REV	Rd, Rn	Reverse byte order in a word	-
REV16	Rd, Rn	Reverse byte order in each halfword	-
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	-

Operações de Reversão

- Reversão dos bits de uma word:
  RBIT Rd, Rm

- Reversão dos bytes de uma word:
  REV Rd, Rm

- Reversão dos bytes de ambas as halfwords de uma word:
  REV16 Rd, Rm

- Reversão dos bytes da halfword menos significativa de uma word, com extensão de sinal:
  REVSH Rd, Rm

Instruções de Acesso à Memória Load

Mnemonic	Operands	Brief description	Flags
LDM	Rn{!}, reglist	Load Multiple registers, increment after	-
LDMDB, LDMEA	Rn{!}, reglist	Load Multiple registers, decrement before	-
LDMFD, LDMIA	Rn{!}, reglist	Load Multiple registers, increment after	-
LDR	Rt, [Rn, #offset]	Load Register with word	-
LDRB, LDRBT	Rt, [Rn, #offset]	Load Register with byte	-
LDRD	Rt, Rt2, [Rn, #offset]	Load Register with double word	-
LDREX	Rt, [Rn, #offset]	Load Register Exclusive	-
LDREXB	Rt, [Rn]	Load Register Exclusive with byte	-
LDREXH	Rt, [Rn]	Load Register Exclusive with halfword	-
LDRH, LDRHT	Rt, [Rn, #offset]	Load Register with halfword	-
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load Register with signed byte	-
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load Register with signed halfword	-
LDRT	Rt, [Rn, #offset]	Load Register with word	-

Instruções de Acesso à Memória

Store

Mnemonic	Operands	Brief description	Flags
STM	Rn{!}, reglist	Store Multiple registers, increment after	-
STMDB, STMEA	Rn{!}, reglist	Store Multiple registers, decrement before	-
STMFD, STMIA	Rn{!}, reglist	Store Multiple registers, increment after	-
STR	Rt, [Rn, #offset]	Store Register word	-
STRB, STRBT	Rt, [Rn, #offset]	Store Register byte	-
STRD	Rt, Rt2, [Rn, #offset]	Store Register two words	-
STREX	Rd, Rt, [Rn, #offset]	Store Register Exclusive	-
STREXB	Rd, Rt, [Rn]	Store Register Exclusive byte	-
STREXH	Rd, Rt, [Rn]	Store Register Exclusive halfword	-
STRH, STRHT	Rt, [Rn, #offset]	Store Register halfword	-
STRT	Rt, [Rn, #offset]	Store Register word	-

STM and STMEA are synonyms for STMIA. STMEA refers to its use for pushing data onto Empty Ascending stacks.

STMFD is s synonym for STMDB, and refers to its use for pushing data onto Full Descending stacks.

Instruções LDR (Load)

- Leitura de word apontada por registrador:
  LDR Rd, [Rn, {#off}]

- Leitura de halfword apontada por registrador:
  LDRH Rd, [Rn, {#off}]

- Leitura de byte apontado por registrador:
  LDRB Rd, [Rn, {#off}]

- Restrições:
  - Somente registradores baixos (R0 a R7) e SP
  - #off: 0..1020 (SP), 0..124 (R0 a R7), 0..62 (LDRH) 0..31 (LDRB); somente valores positivos
  - #off deve ser *4 para LDR e *2 para LDRH

Instruções LDR (Load)

- Indexação por registrador:
  LDR Rd, [Rn, ±Rm {, <opsh>}]
  LDRH Rd, [Rn, ±Rm {, <opsh>}]
  LDRB Rd, [Rn, ±Rm {, <opsh>}]

- <opsh> = deslocamento opcional

- Exemplo:
  LDR R0, [R1, R2, LSL #2] //end = R1 + R2<<2

Pré e Pós-indexação

- Acesso pré-indexado:
  - Primeiro atualiza endereço, depois acessa memória
  - Usar ! para atualizar o ponteiro

LDR Rd, [Rn {, #off}] {!}

LDR Rd, [Rn, ±Rm {, <opsh>}]{!}

- Exemplos:

LDR R0, [R1, #4]
LDR R0, [R1, R2]!

Pré e Pós-indexação

- Acesso pós indexado:
  - Primeiro acessa memória, depois atualiza endereço
  - Sempre atualiza o ponteiro

LDR Rd, [Rn] {, #off}

LDR Rd, [Rn], ±Rm {, <opsh>}

- Exemplos:

LDR R0, [R1], #4
LDR R0, [R1], R2

Carga com Extensão de Sinal

- Leitura de halfword com extensão de sinal:
  LDRSH Rd, [Rn, #off]
  LDRSH Rd, [Rn, ±Rm {, <opsh>}]

- Leitura de byte com extensão de sinal:
  LDRSB Rd, [Rn, #off]
  LDRSB Rd, [Rn, ±Rm {, <opsh>}]

Instruções STR (Store)

- Escrita de word apontada por registrador: STR
- Escrita de halfword apontada por registrador: STRH
- Escrita de byte apontado por registrador: STRB
- Mesmas variações existentes para LDR

Acessos Múltiplos à Memória

- Leitura de múltiplos registradores da memória:
  LDMIA Rn{!}, <reglist>
  LDMDB Rn{!}, <reglist>

- Escrita de múltiplos registradores na memória:
  STMIA Rn{!}, <reglist>
  STMDB Rn{!}, <reglist>

- Exemplos:
  LDM R0, {R0, R1, R2}
  STM R0!, {R2-R4}

Tipos de Pilha

- Descendente (*descending*): a pilha cresce para baixo, começando em um endereço alto e progredindo para um menor
- Ascendente (*ascending*): a pilha cresce para cima, a partir de um endereço baixo e progredindo para um endereço alto
- Pilha completa (*full*): o ponteiro de pilha aponta para o último item na pilha
- Pilha vazia (*empty*): o ponteiro da pilha aponta para o próximo espaço livre

Tipos de Pilhas

- Normalmente Rn = SP (R13)
- Usar ! para atualizar o ponteiro

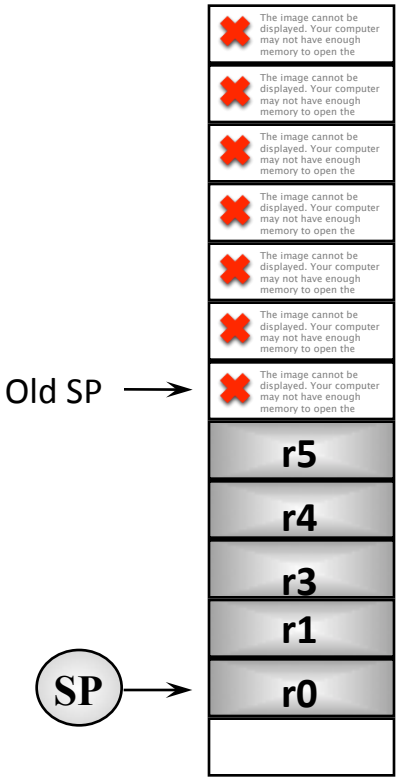
- Full Descending Stack
  - STMFD = STMDB (decrement before)
  - LDMFD = LDMIA (increment after)

- Empty Ascending Stack
  - STMEA = STMIA (increment after)
  - LDMEA = LDMDB (decrement before)

Tipos de Pilha

STMDB SP!, {R0,R1,R3-R5}

STMFD SP!, {R0,R1,R3-R5}

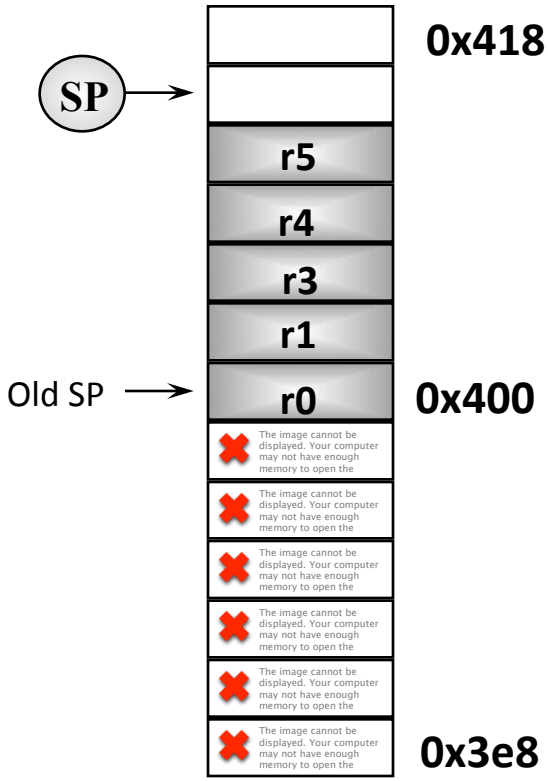


DB : Decrement Before

FD : Full Descending

STMIA SP!, {R0,R1,R3-R5}

STMEA SP!, {R0,R1,R3-R5}



IA : Increment After

EA : Empty Ascending

PUSH e POP

- Implementam uma estrutura de pilha do tipo *Full Descending Stack*
  - PUSH {R0,R1,R3-R5} ↔ STMFD SP!, {R0,R1,R3-R5}
  - POP {R0,R1,R3-R5} ↔ LDMFD SP!, {R0,R1,R3-R5}

- Lista de registradores pode incluir LR (R14) e PC (R15)

Instruções de Salto

Mnemonic	Operands	Brief description	Flags
B	label	Branch	-
BL	label	Branch with Link	-
BLX	Rm	Branch indirect with Link	-
BX	Rm	Branch indirect	-
CBNZ	Rn, label	Compare and Branch if Non Zero	-
CBZ	Rn, label	Compare and Branch if Zero	-

Instruções de Salto

- Salto incondicional:
  B label

- Salto condicional (única instrução condicional que não precisa estar em bloco IT):
  B{cond} label

- Chamada de subrotina (armazena endereço de retorno em LR):
  BL label

- Retorno de chamada de subrotina (PC := LR):
  BX LR

Limites de Abrangência

Instruction	Branch range
`B label`	–16 MB to +16 MB
`B`*cond* `label` (outside IT block)	–1 MB to +1 MB
`B`*cond* `label` (inside IT block)	–16 MB to +16 MB
`BL`{*cond*} `label`	–16 MB to +16 MB
`BX`{*cond*} `Rm`	Any value in register
`BLX`{*cond*} `Rm`	Any value in register

Instruções de Comparação e Salto

- Testa Rn e salta se for diferente de zero:
  CBNZ Rn, label

- Testa Rn e salta se for igual a zero:
  CBZ Rn, label

- Restrições:
  - Apenas registradores baixos (R0 a R7)
  - Apenas saltos para frente de até 130 bytes
  - Não pode ser usado em bloco IT
  - Não afeta flags

Blocos IT (If-Then)

- Um bloco IT consiste de uma a quatro instruções condicionais:
  ITxyz cond

- x, y, z são T ou E (THEN ou ELSE); condições das instruções devem ser coerentes com T ou E

- Exemplo:
  ITTE EQ
  ADDEQ
  SUBEQ
  ORRNE

Blocos IT (If-Then)

- A instrução de salto condicional B{cond} label é a única que não precisa estar em bloco IT
- As instruções IT, CBZ, CBNZ, CPSIE, CPSID não podem estar em bloco IT
- Uma instrução que altera o PC, só pode estar em bloco IT se for a última instrução do bloco

Condições

Suffix	Flags	Meaning
EQ	Z = 1	Equal, last flag setting result was zero
NE	Z = 0	Not equal, last flag setting result was non-zero
CS or HS	C = 1	Higher or same, unsigned
CC or LO	C = 0	Lower, unsigned
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned
LS	C = 0 or Z = 1	Lower or same, unsigned
GE	N = V	Greater than or equal, signed
LT	N != V	Less than, signed
GT	Z = 0 and N = V	Greater than, signed
LE	Z = 1 and N != V	Less than or equal, signed
AL	Can have any value	Always. This is the default when no suffix is specified.