

Arquitetura de Sistemas Embarcados

Edna Barros (ensb@cin.ufpe.br)



Centro de Informática – UFPE

- O processador ARM é um processador RISC (Reduced Instruction Set Computer)
- ARM foi o primeiro microprocessador RISC desenvolvido para uso comercial
- A combinação de um hardware simples com um repertório de instruções reduzido permite eficiência no consumo de potência e tamanho reduzido

Engenharia de Sistemas Embarcados 4

Introdução a Arquitetura ARM (Advanced RISC Machine)

O Processador ARM

- Um dos *cores* mais licenciados e mais vendidos
- Usado em equipamentos móveis devido ao baixo consumo de potência e desempenho razoável
- Várias extensões:
 - THUMB: instruções de 16 bits
 - JAZELLE: máquina virtual JAVA

Engenharia de Sistemas Embarcados 5

História do ARM

- Originalmente significava:
 - ARM – Acorn RISC Machine (1983 – 1985)
 - Acorn Computers Limited, Cambridge, England
- ARM – Advanced RISC Machine 1990
 - ARM Limited, 1990
 - ARM tem sido licenciado para diferentes fabricantes
 - Tornou-se líder de mercado para aplicações embarcadas de baixa potência

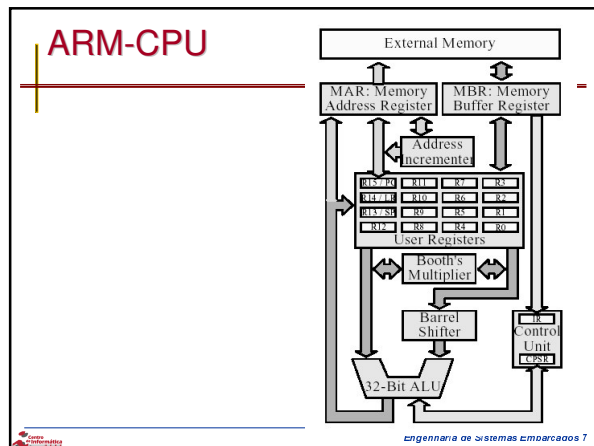
Engenharia de Sistemas Embarcados 3

O Processador ARM

- *Cores*: ARM6, ARM7, ARM9, ARM10, ARM11
- Extensões: THUMB, JAZELLE, etc..
- IP-Blocks: UART, GPIO, controladores de memória

CPU	Description	ISA	Process	Voltage	Area mm ²	Power mW	Clock / MHz	Mips / MHz
ARM7TD-M	Core	V4T	0.18u	1.8V	0.53	<0.25	60-110	0.9
ARM7TD-Mt-S	Synthesizable core	V4T	0.18u	1.8V	<0.8	<0.4	>50	0.9
ARM9TD-M	Core	V4T	0.18u	1.8V	1.1	0.3	167-220	1.1
ARM920T	Macrocell 16+16KB cache	V4T	0.18u	1.8V	11.8	0.9	140-200	1.05
ARM940T	Macrocell 8+8KB cache	V4T	0.18u	1.8V	4.2	0.85	140-170	1.05
ARM9E-S	Synthesizable core	V5TE	0.18u	1.8V	?	~1	133-200	1.1
ARM1020-E	Macrocell 32+32KB cache	V5TE	0.15u	1.8V	~10	~0.85	200-400	1.25

Engenharia de Sistemas Embarcados 6

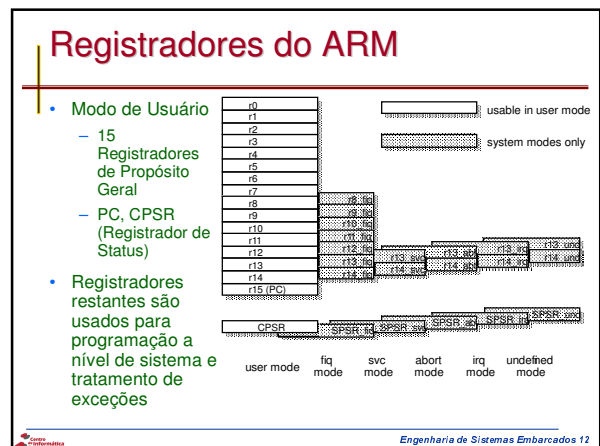


- Exceções:
 - Abort mode executa após a interrupção de busca antecipada de dado ou instrução
 - System mode é um modo privilegiado de usuário para o S.O.
 - Undefined mode executa quando instrução indefinida é executada
- Engenharia de Sistemas Embarcados 10

- ### O Processador ARM
- Processador RISC de 32 bits
 - Instruções de 32 bits
 - 16 registradores de 32 bits (37 registradores internos)
 - Pipeline (ARM7: 3 estágios)
 - Cache
 - Tipos de Dados de 8, 16 e 32 bits
 - 7 modos de operação :
 - Usr, fiq, irq, abt, sys, und
 - Estrutura simples
 - Baixo consumo/ bom desempenho
- Engenharia de Sistemas Embarcados 9

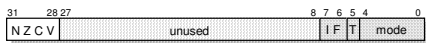
- ### Modelo ARM para o Programador
- 16 Registradores visíveis:
 - Quinze registradores de propósito geral (r0, r1, r2, r3, ..., r12)
 - Registrador r13 é o stack-pointer
 - Registrador r14 guarda endereço de retorno
 - O contador de programa - PC (r15)
 - O Registrador de Status (CPSR)
 - Registradores usados no modo de usuário
 - Outros registradores são usados nos modos privilegiados e de exceção
- Engenharia de Sistemas Embarcados 11

- ### Modos de Operação
- O Processador ARM possui 7 modos de operação (exceções):
- User mode é modo usual de execução de programas de usuário
 - Exceções:
 - Fast Interrupt (FIQ) mode suporta transferência de dados
 - Interrupt (IRQ) mode é usado para tratamento de interrupções
 - Supervisor mode é um modo protegido para o sistema operacional
- Engenharia de Sistemas Embarcados 9



Registrador de Status do ARM CPSR

- N (Negative), Z (Zero), C (Carry), V (Overflow)
- mode – controla modo do processador
- T – controla repertório de instruções
 - T = 1 – repertório de instruções de 16-bit (Thumb instructions)
 - T = 0 – repertório de instruções de 32-bit (ARM instructions)
- I F – desabilita interrupções

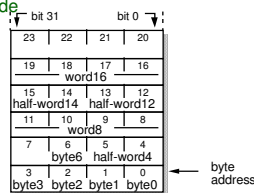


Instruções de Processamento de Dados

- Classes de instruções de Processamento de Dados
 - Operações aritméticas
 - Operações lógicas (nível de bit)
 - Operações de Movimentação entre registradores
 - Operações de Comparação
- Operandos: 32-bits; existem 3 maneiras de especificar os operandos
 - Operandos estão em registradores
 - O segundo operando pode ser uma constante (imediato)
 - Operando no registrador de deslocamento
- Resultado: 32-bits, armazenado em registrador
 - Multiplicação produz um resultado de 64-bits

Organização de Memória do ARM

- Array linear de bytes numerados de 0 a $2^{32} - 1$
- Tipos de dados
 - bytes (8 bits)
 - half-words (16 bits)
 - sempre alinhadas no limite de 2-bytes (iniciam em endereço par)
 - words (32 bits)
 - sempre alinhadas no limite de 4-bytes (iniciam em endereço múltiplo de 4)



Instruções de Processamento de Dados

- Formato

function	operand 1 address	operand 2 address	operand 3 address
----------	----------------------	----------------------	----------------------

- Todos os operandos são de 32 bits e estão em registradores
- O resultado também é armazenado em um registrador

Repertório de Instruções ARM

- Instruções de Processamento de Dados
- Instruções de Transferência de Dados
- Instruções de Fluxo de Controle

Instruções de Processamento de Dados

Operações Aritméticas

ADD r0, r1, r2	r0 := r1 + r2
ADC r0, r1, r2	r0 := r1 + r2 + C
SUB r0, r1, r2	r0 := r1 - r2
SBC r0, r1, r2	r0 := r1 - r2 + C - 1
RSB r0, r1, r2	r0 := r2 - r1
RSC r0, r1, r2	r0 := r2 - r1 + C - 1

Operações Lógicas com Bits

AND r0, r1, r2	r0 := r1 and r2
ORR r0, r1, r2	r0 := r1 or r2
EOR r0, r1, r2	r0 := r1 xor r2
BIC r0, r1, r2	r0 := r1 and (not) r2

Movimentação de Registradores

MOV r0, r2	r0 := r2
MVN r0, r2	r0 := not r2

Operações de Comparação

CMP r1, r2	set cc on r1 - r2
CMN r1, r2	set cc on r1 + r2
TST r1, r2	set cc on r1 and r2
TEQ r1, r2	set cc on r1 xor r2

Instruções Aritméticas

- Operandos Imediatos:
Constante = $(0 \rightarrow 255) \times 2^{2n}$, $0 \leq n \leq 12$

ADD r3, r3, #3	r3 := r3 + 3
AND r8, r7, #&ff	r8 := r7[7:0], & for hex

- Operandos em Registrador de Deslocamento
 - O segundo operando está sujeito a uma operação de deslocamento antes que seja combinado com o primeiro operando

ADD r3, r2, r1, LSL #3	r3 := r2 + 8 x r1
ADD r5, r5, r3, LSL r2	r5 := r5 + 2^{r2} x r3



Engenharia de Sistemas Embarcados 19

Operações de Comparação

- Só afetam os flags (N, Z, C and V) no CPSR

CMP r1, r2	;set cc on r1 - r2
CMN r1, r2	;set cc on r1 + r2
TST r1, r2	;set cc on r1 and r2
TEQ r1, r2	;set cc on r1 xor r2



Engenharia de Sistemas Embarcados 22

Operações Lógicas

- Operações Booleanas

AND r0, r1, r2	; r0 := r1 and r2
ORR r0, r1, r2	; r0 := r1 or r2
EOR r0, r1, r2	; r0 := r1 xor r2
BIC r0, r1, r2	; r0 := r1 and not r2



Engenharia de Sistemas Embarcados 20

Registrador de Deslocamento

- O ARM não possui instruções de deslocamento
- O registrador de deslocamento permite o deslocamento de um operando de instrução aritmética



Engenharia de Sistemas Embarcados 23

Operações de Movimentação de Registradores

MOV r0, r2	; r0 := r2
MVN r0, r2	; r0 := not r2

MVN: o registrador destino recebe o registrador fonte com o bits invertidos



Engenharia de Sistemas Embarcados 21

- Barrel Shifter- Deslocamento para Esquerda
 - LSL #5 => multiplica por 2^5 => multiplica por 32
- Barrel Shifter- Deslocamento para Direita
 - LSR #5 => divide por 2^5 => divide por 32



Engenharia de Sistemas Embarcados 24

Deslocando Operandos

- Por exemplo,
`ADD r3, r2, r1, LSL #3 ; r3 := r2 + 8 x r1`
- 'LSL' indica 'logical shift left pelo número de bits especificado', que é igual a 3
- '#' indica valor imediato.



Engenharia de Sistemas Embarcados 25

Setando os Códigos de Condição – FLAGS

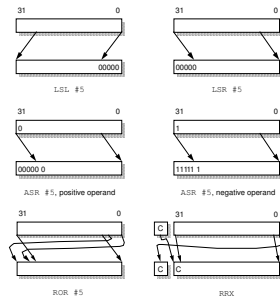
- Operações aritméticas setam todos os flags (N, Z, C, and V)
- Operações lógicas e de move setam N e Z
 - Preserva V e C quando não se trata de operações de deslocamento, ou setam C de acordo com a operação de deslocamento realizada no operando



Engenharia de Sistemas Embarcados 28

Operações de Deslocamento ARM

- LSL – Deslocamento Lógico para Esquerda
- LSR – Deslocamento Lógico para Direita
- ASR – Deslocamento Aritmético para Direita
- ROR – Rotação para Direita
- RRX – Rotação para Direita



Engenharia de Sistemas Embarcados 26

Multiplicação

- Exemplo (Multiply, Multiply-Accumulate)

<code>MUL r4, r3, r2</code>	<code>r4 := [r3 x r2] <31:0></code>
<code>MLA r4, r3, r2, r1</code>	<code>r4 := [r3 x r2 + r1] <31:0></code>

- Nota
 - 32-bits menos significativos são colocados no registrador de resultados, os outros são ignorados
 - Segundo operando imediato não é suportado
 - Registrador de resultado pode ser diferente de registradores fonte
 - Se bit 'S' é setado então V é preservado e C não possui significado



Engenharia de Sistemas Embarcados 29

Setando os Códigos de Condição – FLAGS

- Qualquer instrução pode setar os códigos de condição (N, Z, V, e C)
 - Para todas as instruções (exceto para a operação de comparação) uma requisição deve ser feita explicitamente
 - Em linguagem de montagem esta requisição é indicada pela adição de um 'S' ao opcode
 - Exemplo (`r3-r2 := r1-r0 + r3-r2`)

<code>ADDS r2, r2, r0</code>	<code>; carry out to C</code>
<code>ADC r3, r3, r1</code>	<code>; ... add into high word</code>



Engenharia de Sistemas Embarcados 27

Instruções de Transferência de Dados

- Instruções Simples de load e store
 - Transferência de um dado (byte, half-word, word) entre registradores ARM e memória
- Instruções de Múltiplos load e store
 - Permite a transferência de uma grande quantidade de dados
 - Usado para entrada e saída de subrotina para salvar e restaurar registradores de trabalho e para copiar blocos de dados na memória
- Instruções de swap de registradores simples
 - Toda a transferência entre registrador e memória em uma instrução
 - Usado na implementação de semáforos para garantir exclusão mútua no acesso a dados compartilhados



Instruções de Transferência de Dados

Endereçamento de Registrador Indireto

Load e Store simples

LDR r0, [r1]	r0 := mem ₃₂ [r1]
STR r0, [r1]	mem ₃₂ [r1] := r0

Nota: r1 armazena endereço de palavra (2 LSBs são 0)

LDRB r0, [r1]	r0 := mem ₈ [r1]
---------------	-----------------------------

Nota: nenhuma restrição para r1

Endereçamento de Base+offset (offset até 4K bytes)

LDR r0, [r1, #4]	r0 := mem ₃₂ [r1 + 4]
------------------	----------------------------------

Endereçamento auto-indexado

LDR r0, [r1, #4]!	r0 := mem ₃₂ [r1 + 4] r1 := r1 + 4
-------------------	--

Endereçamento Pós-indexado

LDR r0, [r1], #4	r0 := mem ₃₂ [r1] r1 := r1 + 4
------------------	--

Engenharia de Sistemas Embarcados 31

Instruções de Transferência de Dados

Condition field

Up/Down bit
0 = Down; subtract offset from base
1 = Up; add offset to base

Pre/Post indexing bit
0 = Post; add offset after transfer
1 = Pre; add offset before transfer

Base register

Load/Store bit
0 = Store to memory
1 = Load from memory

Write-back bit
0 = no write-back
1 = write address into base

PSR and force user bit
0 = don't load PSR or force user mode
1 = load PSR or force user mode

Each bit corresponds to a particular register. For example:
• Bit 0 set causes r0 to be transferred.
• Bit 9 unset causes r9 not to be transferred.
At least one register must be transferred as the list cannot be empty.

Engenharia de Sistemas Embarcados 34

Instruções de Transferência de Dados

COPY: ADR r1, TABLE1 ; r1 points to TABLE1
ADR r2, TABLE2 ; r2 points to TABLE2

LOOP: LDR r0, [r1]
STR r0, [r2]
ADD r1, r1, #4
ADD r2, r2, #4
...

TABLE1: ...
TABLE2:...

COPY: ADR r1, TABLE1 ; r1 points to TABLE1
ADR r2, TABLE2 ; r2 points to TABLE2

LOOP: LDR r0, [r1], #4
STR r0, [r2], #4
...

TABLE1: ...
TABLE2:...

Engenharia de Sistemas Embarcados 32

Modos de endereçamento para Transferências Múltiplas

STMB r9!, {r0, r1, r5} STMB r9!, {r0, r1, r5}

STMDA r9!, {r0, r1, r5} STMDB r9!, {r0, r1, r5}

Engenharia de Sistemas Embarcados 35

Instruções de Transferência de Dados

Multiple register data transfers

LDMIA r1, {r0, r2, r5}	r0 := mem ₃₂ [r1] r2 := mem ₃₂ [r1 + 4] r5 := mem ₃₂ [r1 + 8]
------------------------	--

Notas: 1. qualquer subconjunto (ou todos) os registradores podem ser transferidos em uma única instrução
2. a ordem dos registradores é insignificante
3. incluir o r15 na lista causará mudança no fluxo de execução

Organizações de Pilha

FA – ascendente full
EA – ascendente vazia
FD – descendente full
ED – descendente vazia

Cópia de Bloco

- Dado deve ser armazenado antes ou abaixo do endereço guardado no registrador base
- Incremento ou decremento do endereço inicia antes ou depois de armazenamento do primeiro valor

Engenharia de Sistemas Embarcados 33

O Mapeamento entre a pilha e cópia de blocos

		Ascending Full	Ascending Empty	Descending Full	Descending Empty
Increment	Before	STMB STMA			LDMB LDMED
	After		STMA STMEA	LDMIA LDMFD	
Decrement	Before		LDMDB LDMEA	STMDB STMFD	
	After	LDMDA LDMFA			STMDA STMED

Engenharia de Sistemas Embarcados 36

Instruções de Controle de Fluxo

Branch	Interpretation	Normal uses
B	Unconditional	Always take this branch
BAL	Always	Always take this branch
BEQ	Equal	Comparison equal or zero result
BNE	Not equal	Comparison not equal or non-zero result
BPL	Plus	Result positive or zero
BMI	Minus	Result minus or negative
BCC	Carry clear	Arithmetic operation did not give carry-out
BLO	Lower	Unsigned comparison gave lower
BCS	Carry set	Arithmetic operation gave carry-out
BHS	Higher or same	Unsigned comparison gave higher or same
BVC	Overflow clear	Signed integer operation; no overflow occurred
BVS	Overflow set	Signed integer operation; overflow occurred
BGT	Greater than	Signed integer comparison gave greater than
BGE	Greater or equal	Signed integer comparison gave greater or equal
BLT	Less than	Signed integer comparison gave less than
BLE	Less or equal	Signed integer comparison gave less than or equal
BHI	Higher	Unsigned comparison gave higher
BLS	Lower or same	Unsigned comparison gave lower or same

Instruções Branch and link

- Desvio para subrotina (r14 armazena endereço de retorno)

```

BL SUBR ; branch to SUBR
... ; return here

SUBR: ... ; SUBR entry point
MOV pc, r14 ; return

BL SUB1
...
SUB1: ... ; save work and link register
      STMFD r13!, {r0-r2,r14}
      BL SUB2
      ...
      LDMFD r13!, {r0-r2,pc}
SUB2: ...
      MOV pc, r14 ; copy r14 into r15
  
```

- Subrotinas aninhadas

Execução Condicional

- Execução condicional evita instruções de desvio
- Exemplo

```

CMP r0, #5;
BEQ BYPASS ; if (r0!=5) {
ADD r1, r1, r0 ; r1:=r1+r0-r2
SUB r1, r1, r2 ; }
BYPASS: ...
  
```

Com execução condicional

```

CMP r0, #5 ; ; if ((a==b) && (c==d)) e++;
ADDNE r1, r1, r0 ;
SUBNE r1, r1, r2 ;
...
CMP r0, r1
CMPEQ r2, r3
ADDEQ r4, r4, #1
  
```

Nota: condições são adicionadas ao opcode

Formato das Instruções do ARM

31	26:27	16:15	8:7	0	Instruction type
Cond	0 0	Opcode	Rn	Rd	Operand2
Cond	0 0 0 0 0 0	A S	Rd	Rn	Ra
Cond	0 0 0 0 1 0	U A S	RdHi	RdLo	Ra
Cond	0 0 0 1 0 0	B 0 0	Rn	Rd	0 0 0 0 1 0 0 1
Cond	0 1 1 0 0 0	B W L	Rn	Rd	Offset
Cond	1 0 0 0 0 0	B W L	Rn		Register List
Cond	0 0 0 0 1 1	B W L	Rn	Rd	Offset1 1 0 0 1 Offset2
Cond	0 0 0 0 1 0	B W L	Rn	Rd	0 0 0 0 1 0 0 1
Cond	1 0 1 1				Offset
Cond	0 0 0 1 0 0 1 0	1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 1
Cond	1 1 0 0	B N W L	Rn	CRd	CPHm Offset
Cond	1 1 1 0	Op1	CRn	CRd	CPHm Op2 0 CRm
Cond	1 1 1 0	Op1	CRn	Rd	CPHm Op2 1 CRm
Cond	1 1 1 1				SMT Number

Execução Condicional

Code	Suffix	Flags	Meaning
0000	EQ	Z set	equal
0001	NE	Z clear	not equal
0010	CS	C set	unsigned higher or same
0011	CC	C clear	unsigned lower
0100	MI	N set	negative
0101	PL	N clear	positive or zero
0110	VS	V set	overflow
0111	VC	V clear	no overflow
1000	HI	C set and Z clear	unsigned higher
1001	LS	C clear or Z set	unsigned lower or same
1010	GE	N equals V	greater or equal
1011	LT	N not equal to V	less than
1100	GT	Z clear AND (N equals V)	greater than
1101	LE	Z set OR (N not equal to V)	less than or equal
1110	AL	(ignored)	always

Table 4-2: Condition code summary

Repertório de Instruções ARM

- Arquitetura Load-store
 - operandos armazenados em registradores
 - load/store – únicas instruções que acessam a memória
- Instruções
 - Processamento de Dados – utiliza e modifica valores de registradores
 - Transferência de Dados – copia valores de memória em registradores (load) ou copia valores de registradores em memória (store)
 - Controle de Fluxo
 - branch
 - branch-and-link – salva endereço de retorno
 - trapping – chamada de supervisor

Repertório de Instruções do ARM

- Instruções de processamento de dados com três endereços
- Execução condicional para todas as instruções
- Instruções de load/store para múltiplos registradores
- Habilidade de realizar uma operação de deslocamento e uma operação de ALU em um único ciclo
- Extensão do repertório de instruções através do co-processador incluindo mais registradores e tipos de dados
- Representação das instruções com 16 bits na arquitetura Thumb



Modos de Operação

O Processador ARM possui 7 modos de operação:

- **User mode** é modo usual de execução de programas de usuário
- **Fast Interrupt (FIQ) mode** suporta transferência de dados
- **Interrupt (IRQ) mode** é usado para tratamento de interrupções
- **Supervisor mode** é um modo protegido para o sistema operacional



Engenharia de Sistemas Embarcados 46

Exceções

Os vários modos de operação do processador

- **Abort mode** executa após a interrupção de busca antecipada de dado ou instrução
- **System mode** é um modo privilegiado de usuário para o S.O.
- **Undefined mode** executa quando instrução indefinida é executada



Engenharia de Sistemas Embarcados 47

Exceções

- Exceções são geradas por fontes internas ou externas ao programa em execução: evento de dispositivo ou instrução não definida
- Mais que uma exceção pode acontecer ao mesmo tempo
- Estado do processador antes da exceção deve ser preservado
- ARM suporta 7 tipos de exceções



Engenharia de Sistemas Embarcados 45

Tipos de Exceções

Exception	Mode	Normal Address
Reset	Supervisor	0x00000000
Undefined instruction	Undefined	0x00000004
SWI	Supervisor	0x00000008
Prefetch abort	Abort	0x0000000C
Data abort	Abort	0x00000010
IRQ	IRQ	0x00000018
FIQ	FIQ	0x0000001C



Engenharia de Sistemas Embarcados 48

Exceções ARM

- Tratamento de Exceções
 - Estado corrente é salvo através da cópia de PC no registrador r14_exc e CPSR em SPSR_exc (exc significa para tipo exceção)
 - Modo de operação do processador é mudado para o tipo apropriado de exceção
 - PC é forçado a ter um valor entre 00₁₆ e 1C₁₆, sendo o valor particular dependente do tipo de exceção
 - Instrução na localização do PC é forçada a conter um desvio para a rotina de tratamento de exceções (the vector address); a rotina de tratamento usará o registrador r13_exc, o qual é normalmente inicializado para apontar para pilha na memória, onde registradores serão salvos
 - retorno: restaurar registradores de usuário, então restaurar PC e CPSR (atômicamente)

Instruções de Transferências de registradores de Controle

- As instruções MRS e MSR permitem que o conteúdo de registradores de controle sejam transferidos para registradores de propósito geral
 - Todos os bits ou apenas os bits de flags podem ser transferidos

Sintaxe:

- MRS{<cond>} Rd,<psr> ; Rd = <psr>
- MSR{<cond>} <psr>,<Rm> ; <psr> = Rm
- MSR{<cond>} <psrf>,<Rm> ; <psrf> = Rm

onde

- <psr> = CPSR, CPSR_all, SPSR or SPSR_all
- <psrf> = CPSR_flg or SPSR_flg

Endereçamento imediato também é possível:

- MSR{<cond>} <psrf>,<#Immediate>
- Os quatro bits mais significativos são os flags

Organização dos Registradores

General registers and Program Counter

User/2 / System	FIQ/32	Supervisor/32	Abort/32	IRQ/32	User/4 and 32
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8	r8	r8	r8	r8
r9	r9	r9	r9	r9	r9
r10	r10	r10	r10	r10	r10
r11	r11	r11	r11	r11	r11
r12	r12	r12	r12	r12	r12
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)	r15 (pc)

Program Status Registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
SPSR_flg	SPSR_flg	SPSR_flg	SPSR_flg	SPSR_flg	SPSR_flg

Instruções de Transferências de registradores de Controle



Copies of the ALU status flags (latched if the instruction has the "S" bit set).

Condition Code Flags

- N = Negative result from ALU flag.
- Z = Zero result from ALU flag.
- C = ALU operation Carried out
- V = ALU operation oVerflowed

Interrupt Disable bits.

- I = 1, disables the IRQ.
- F = 1, disables the FIQ.

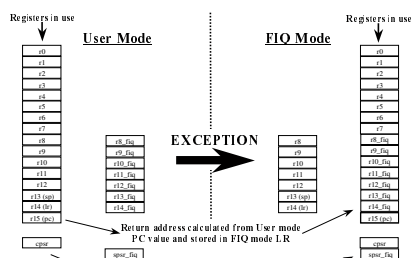
T Bit (Architecture v4T only)

- T = 0, Processor in ARM state
- T = 1, Processor in Thumb state

Mode Bits

- M[4:0] define the processor mode.

Exemplo: Modo de Usuário para FIQ



Usando instruções MRS e MSR

- Os bits reservados não podem ser alterados



Estratégia de uso:

Copia PSR para registrador

- Modifica bits
- Copia registrador atualizado para PSR

Nota:

- No modo usuário apenas os bits de flags podem ser modificados

Quando uma exceção ocorre....

```
R14_<tipo_exceção> = endereço de retorno
SPSR_< tipo_exceção > = CPSR
CPSR[4:0] = Número do modo (exceção)
CPSR[5] = 0      /* Execute in ARM state */
Se < tipo_exceção > == Reset ou FIQ então
    CPSR[6] = 1  /* Desabilita interrupções rápidas */
/* caso contrário CPSR[6] permanece inalterado */
CPSR[7] = 1      /*Desabilita interrupções normais*/
PC = endereço de tratamento
```



Engenharia de Sistemas Embarcados 55

Reset (Cont.)

```
R14_SVC      =      valor imprevisível
SPSR_svc     =      valor imprevisível
CPSR [4:0]   =      0b10011 /*Entra no modo supervisor*/
CPSR [5]     =      0        /* Executa modo ARM */
CPSR [6]     =      1        /* Desabilita interrup. rápidas */
CPSR [7]     =      1        /*Desabilita interrupções
                             normais*/
PC           =      0x00000000
```



Engenharia de Sistemas Embarcados 58

Quando uma exceção ocorre...

- No retorno, a rotina de tratamento:
 - Restaura CPSR de SPSR_< tipo_exceção >
 - Restaura PC de LR_< tipo_exceção >



Engenharia de Sistemas Embarcados 56

Exceção de Instrução Indefinida

- Se o processador executa uma instrução de co-processador, ele espera pelo processador externo reconhecer que a instrução será executada. Se não há resposta do co-processador, uma exceção de instrução indefinida ocorre
- Este tipo de exceção pode ser usada para emular um co-processador em software ou aumentar o repertório de instruções com instruções em software



Engenharia de Sistemas Embarcados 59

Reset

- A ativação do RESET para imediatamente a execução da instrução corrente
- O processador inicia a execução nos endereços 0x00000000 ou 0xFFFF0000 no modo supervisor com interrupções desabilitadas



Engenharia de Sistemas Embarcados 57

Exceção de Instrução Indefinida

```
R14_und = endereço de retorno
SPSR_und = CPSR
CPSR [4:0] = 0b11011 ;enter undefined mode
CPSR [5] = 0          ;execute in ARM state
/*CPSR [6] is unchanged */
CPSR [7] = 1          ; disable normal interrupts
PC = 0x00000004
Retorno após emular a instrução:
MOV PC, r14
```



Engenharia de Sistemas Embarcados 60

Chamadas de Supervisor

- Supervisor é um programa que opera em modo privilegiado – pode executar instruções que não são executadas no modo usuário
 - Exemplo: enviar texto para display
- ARM ISA inclui SWI (SoftWare Interrupt)

```
; output r0[7:0]
SWI SWI_WriteC
; retorno de um programa de usuário para
monitor
SWI SWI_Exit
```



Engenharia de Sistemas Embarcados 61

Prefetch Abort

- A exceção de Prefetch Abort é gerada se o processador tenta executar uma instrução inválida
- Na arquitetura ARM, uma exceção Prefetch Abort pode ser gerada quando executado uma instrução BKPT (break-point).



Engenharia de Sistemas Embarcados 64

Software Interrupt (SWI)



Condition Field

- Uma interrupção de software SWI pode ser usada para implementar uma instrução definida pelo usuário
- Ela causa uma mudança para modo supervisor e a rotina de tratamento é executada.
- A rotina de tratamento depende do conteúdo do campo de comentário
- O mecanismo SWI permite que o S.O. implemente um conjunto de instruções privilegiadas, que podem ser executadas no modo de usuário



Engenharia de Sistemas Embarcados 62

Prefetch Abort

```
R14_abt = endereço da instrução abortada + 4
SPSR_abt = CPSR
CPSR [4:0] = 0b10111
CPSR [5] = 0
/* CPSR [6] is unchanged */
CPSR [7] = 1
PC = 0x0000000C
Retorno após resolver a causa da exceção:
SUBS PC, r14, #4
```



Engenharia de Sistemas Embarcados 65

Software Interrupt

- R14_svc = endereço de retorno
- SPSR_svc = CPSR ;Enter Supervisor mode
- CPSR [4:0] = 0b10011 ;Execute in ARM state
- /* CPSR [6] is unchanged */
- CPSR [7] = 1 /*Disable normal interrupts */
- PC = 0x00000008
- Para retornar após executar SWI:


```
MOVS PC, r14
```



Engenharia de Sistemas Embarcados 63

Data Abort exception

- Acesso a dado inválido
- A exceção de Data abort ocorre antes que qualquer outra exceção tenha alterado o estado da CPU
- Data Abort tem maior prioridade entre todas as exceções



Engenharia de Sistemas Embarcados 66

Data Abort exception cont...

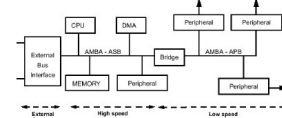
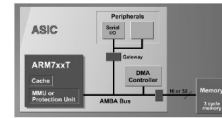
R14_abt = address of the aborted inst. + 8
SPSR_abt = CPSR
CPSR [4:0] = 0b10111 ; Enter abort mode
CPSR [5] = 0 ; Execute in ARM state
/* CPSR[6] is unchanged */
CPSR [7] = 1 ; Disable normal interrupts
PC = 0x00000010
Para retorno após resolver causa da exceção:
SUBS PC, R14, #8



Engenharia de Sistemas Embarcados 67

ARM – AMBA BUS

- AMBA-bus:
- ASB i.e. AMBA System Bus
- APB i.e. AMBA Peripheral Bus
- AHB i.e. AMBA High bandwidth Bus



Engenharia de Sistemas Embarcados 70

Sistema de I/O

- I/O é mapeada em memória
 - Registradores internos de periféricos (controladores de disco, redes e interfaces) são posições de memória endereçáveis e podem ser lidas e escritas por instruções load/store
- Periféricos podem usar dois tipos de interrupção
 - interrupção normal (IRQ) ou
 - entrada rápida de interrupção(FIQ)
 - Normalmente grande parte das entradas compartilham a entrada IRQ enquanto que uma ou duas fontes críticas são conectadas a entrada FIQ
- Alguns sistemas podem incluir hardware externo de DMA para suportar altas taxas de transferências



Engenharia de Sistemas Embarcados 68

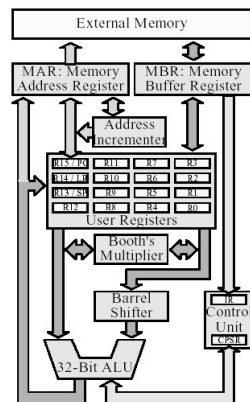
Exemplo: Hello ARM World!

```
AREA HelloWorld, CODE, READONLY ; declare code area
SWI_WriteC EQU &0 ; output character in r0
SWI_Exit EQU &11 ; finish program
ENTRY ; code entry point
START: ADR r1, TEXT ; r1 <- Hello ARM World!
LOOP: LDRB r0, [r1], #1 ; get the next byte
CMP r0, #0 ; check for text end
SWINE SWI_WriteC ; if not end of string, print
BNE LOOP
SWI SWI_Exit ; end of execution
TEXT = "Hello ARM World!", &0a, &0d, 0
END
```



Engenharia de Sistemas Embarcados 71

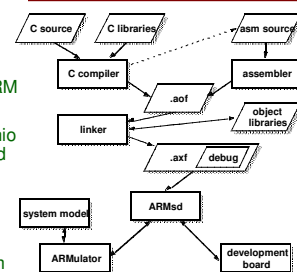
ARM-CPU



Engenharia de Sistemas Embarcados 69

Ambiente de Desenvolvimento ARM

- Desenvolvimento de Software
 - Ferramentas desenvolvidas pela ARM Limited
 - Ferramentas de domínio público (ARM back end para compilador gcc)
- Desenvolvimento Cruzado
 - Ferramentas executam em diferentes arquiteturas para as quais código é produzido.



Engenharia de Sistemas Embarcados 72

Resumindo

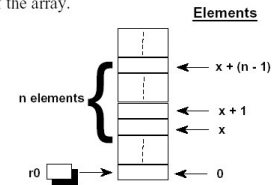
- Todas as instruções possuem 32 bits
- Grande parte das instruções executam em um ciclo
- Todas as instruções são condicionais
- Arquitetura Load/Store
 - Instruções de acesso a memória possui auto indexação
- Instruções de processamento de dados usam apenas registradores e possuem três endereços



Engenharia de Sistemas Embarcados 73

Exercícios

- * Write a segment of code that add together elements x to $x+(n-1)$ of an array, where the element $x=0$ is the first element of the array.
- * Each element of the array is word sized (ie. 32 bits).
- * The segment should use post-indexed addressing.
- * At the start of your segments, you should assume that:
 - $r0$ points to the start of the array.
 - $r1 = x$
 - $r2 = n$



Engenharia de Sistemas Embarcados 76

Resumindo

- Combina operação da ALU com registrador de deslocamento para manipulação de bits com desempenho
- Extensão do repertório de instruções através de instruções de co-processador



Engenharia de Sistemas Embarcados 74

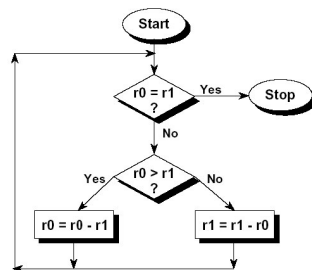
Exercícios

- * The contents of registers $r0$ to $r6$ need to be swapped around thus:
 - $r0$ moved into $r3$
 - $r1$ moved into $r4$
 - $r2$ moved into $r6$
 - $r3$ moved into $r5$
 - $r4$ moved into $r0$
 - $r5$ moved into $r1$
 - $r6$ moved into $r2$
- * Write a segment of code that uses full descending stack operations to carry this out, and hence requires no use of any other registers for temporary storage.



Engenharia de Sistemas Embarcados 77

Exercícios



- * Convert the GCD algorithm given in this flowchart into

- 1) "Normal" assembler, where only branches can be conditional.
- 2) ARM assembler, where all instructions are conditional, thus improving code density.

- * The only instructions you need are CMP, B and SUB.



Engenharia de Sistemas Embarcados 75