4 Set di istruzioni ARM

Questo capitolo descrive il set di istruzioni ARM.

Set di istruzioni ARM

4.1 Riepilogo del set di istruzioni

4.1.1 Riassunto del formato

I formati del set di istruzioni ARM sono illustrati di seguito.

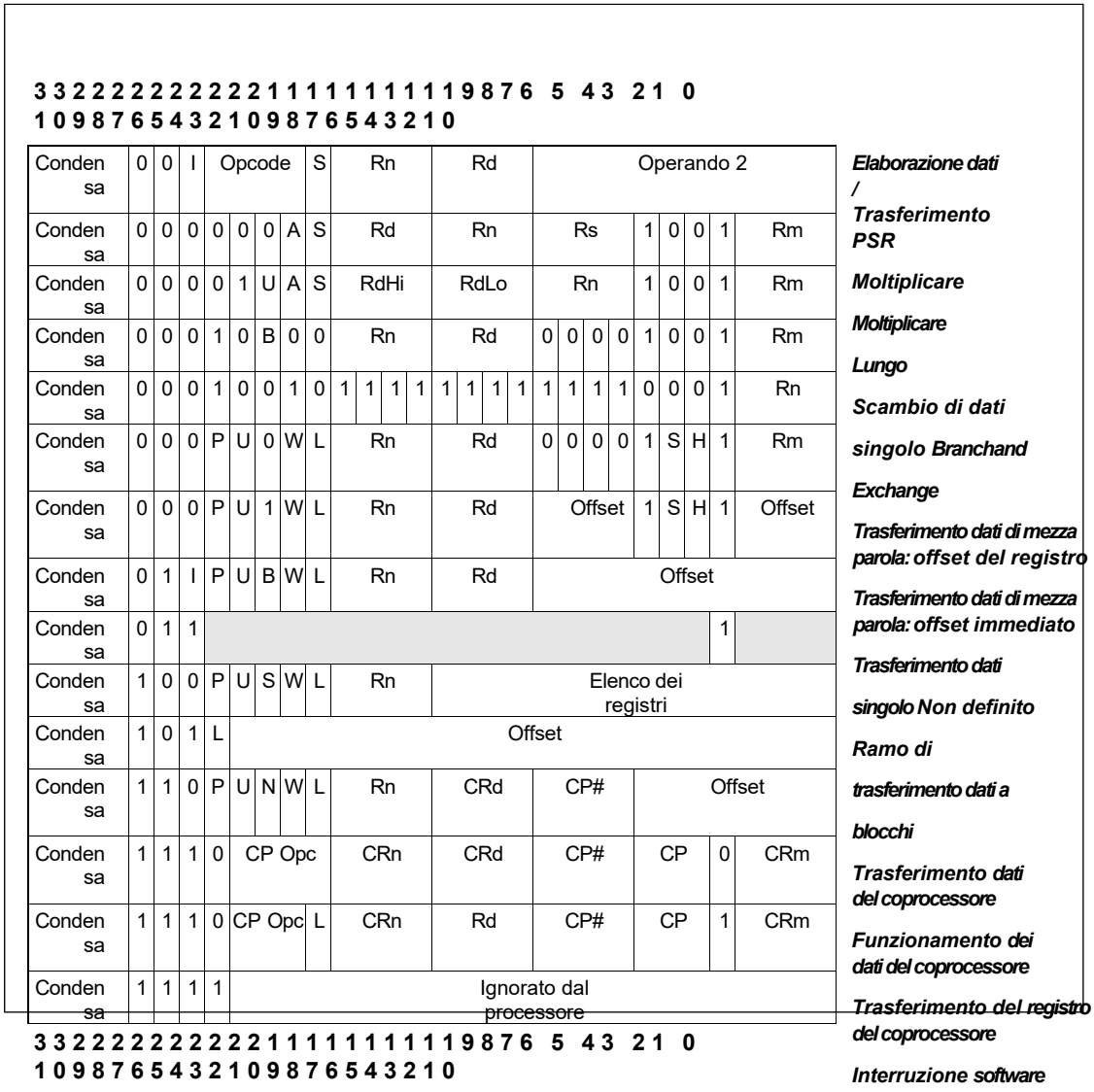


Figura 4-1: Formati del set di istruzioni ARM

Nota *Alcuni codici di istruzioni non sono definiti ma non causano la trappola per istruzioni non definite, ad esempio un'istruzione Moltiplica con il bit 6 cambiato in 1. Queste istruzioni non devono essere utilizzate, poiché la loro azione potrebbe cambiare nelle future implementazioni ARM.*

4.1.2 Riassunto delle istruzioni

Mnemonico	Istruzioni	Azione	Vedere la sezione:
ADC	Aggiungere con riporto	Rd := Rn + Op2 + Carry	4.5
ADD	Aggiungi	Rd := Rn + Op2	4.5
E	E	Rd := Rn E Op2	4.5
B	Ramo	R15 := indirizzo	4.4
BIC	Bit Libero	Rd := Rn E NON Op2	4.5
BL	Ramo con collegamento	R14 := R15, R15 := indirizzo	4.4
BX	Filiali e scambi	R15 := Rn, T bit := Rn[0]	4.3
CDP	Elaborazione dati con coprocessore	(specifico per il coprocessore)	4.14
CMN	Confronta Negativo	Bandiere CPSR := Rn + Op2	4.5
CMP	Confronto	Bandiere CPSR := Rn - Op2	4.5
EOR	Esclusivo OR	Rd := (Rn E NON Op2) OPPURE (op2 E NON Rn)	4.5
LDC	Caricare il coprocessore dalla memoria	Carico del coprocessore	4.15
LDM	Caricare più registri	Manipolazione della pila (Pop)	4.11
LDR	Carica il registro dalla memoria	Rd := (indirizzo)	4.9, 4.10
MCR	Spostare il registro della CPU nel registro del coprocessore	cRn := rRn {<op>cRm}	4.16
MLA	Moltiplicare Accumulare	Rd := (Rm * Rs) + Rn	4.7, 4.8
MOV	Spostare un registro o una costante	Rd : = Op2	4.5
MRC	Spostamento da un registro del coprocessore a un registro della CPU	Rn := cRn {<op>cRm}	4.16
SIGNORA	Spostare lo stato/flag del PSR nel registro	Rn := PSR	4.6
MSR	Spostare il registro in stato/flags PSR	PSR := Rm	4.6
MUL	Moltiplicare	Rd := Rm * Rs	4.7, 4.8
MVN	Spostare il registro negativo	Rd := 0xFFFFFFFF EOR Op2	4.5
ORR	O	Rd := Rn O Op2	4.5
RSB	Sottrazione inversa	Rd := Op2 - Rn	4.5
RSC	Sottrazione inversa con carry	Rd := Op2 - Rn - 1 + Carry	4.5

Mnemonico	Istruzioni	Azione	Vedere la sezione:
SBC	Sottrazione con riporto	Rd := Rn - Op2 - 1 + Carry	4.5
STC	Memorizzazione del registro del coprocessore nella memoria	indirizzo := CRn	4.15
STM	Negozio multiplo	Manipolazione della pila (Push)	4.11
STR	Memorizzare il registro nella memoria	<indirizzo> := Rd	4.9, 4.10
SUB	Sottrarre	Rd := Rn - Op2	4.5
SWI	Interruzione software	Chiamata al sistema operativo	4.13
SWP	Scambiare il registro con la memoria	Rd := [Rn], [Rn] := Rm	4.12
TEQ	Test di uguaglianza bitwise	CPSR flags := Rn EOR Op2	4.5
TST	Bit di prova	flag CPSR := Rn E Op2	4.5

Tabella 4-1: Il set di istruzioni ARM (continua)

4.2 Il campo delle condizioni

Nello stato ARM, tutte le istruzioni vengono eseguite in modo condizionato in base allo stato dei codici di condizione CPSR e al campo di condizione dell'istruzione. Questo campo (bit 31:28) determina le circostanze in cui un'istruzione deve essere eseguita. Se lo stato dei flag C, N, Z e V soddisfa le condizioni codificate dal campo, l'istruzione viene eseguita, altrimenti viene ignorata.

Esistono sedici condizioni possibili, ciascuna rappresentata da un suffisso di due caratteri che può essere aggiunto al mnemonico dell'istruzione. Ad esempio, una diramazione (`B` in linguaggio assembly) diventa `BEQ` per "Branch if Equal" (diramazione se uguale), il che significa che la diramazione verrà eseguita solo se il flag Z è impostato.

In pratica, possono essere utilizzate quindici condizioni diverse, elencate nella ***Tabella 4-2: Riepilogo dei codici di condizione***. Il sedicesimo (1111) è riservato e non deve essere utilizzato.

In assenza di un suffisso, il campo di condizione della maggior parte delle istruzioni è impostato su "Sempre" (prefisso AL). Ciò significa che l'istruzione verrà sempre eseguita, indipendentemente dai codici di condizione CPSR.

Codice	Suffisso	Bandiere	Significato
0000	EQ	Set Z	uguale
0001	NE	Z chiaro	non uguale
0010	CS	C set	non firmato superiore o uguale
0011	CC	C chiaro	non firmato inferiore
0100	MI	Set di N	negativo
0101	PL	N chiaro	positivo o nullo
0110	VS	V set	traboccare
0111	VC	V chiaro	nessun overflow
1000	HI	C impostato e Z libero	superiore senza segno
1001	LS	C chiaro o Z impostato	non firmato inferiore o uguale
1010	GE	N è uguale a V	maggiore o uguale
1011	LT	N non è uguale a V	meno di
1100	GT	Z chiaro AND (N uguale a V)	maggiore di
1101	LE	Z set OR (N non uguale a V)	inferiore o uguale
1110	AL	(ignorato)	sempre

Tabella 4-2: Riepilogo dei codici delle condizioni

4.3 Filiali e scambi (BX)

Questa istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

Questa istruzione esegue una derivazione copiando il contenuto di un registro generale, Rn, nel contatore di programma, PC. La diramazione provoca un flush e un refill della pipeline a partire dall'indirizzo specificato da Rn. Questa istruzione consente inoltre di scambiare il set di istruzioni. Quando l'istruzione viene eseguita, il valore di Rn[0] determina se il flusso di istruzioni verrà decodificato come istruzioni ARM o THUMB.

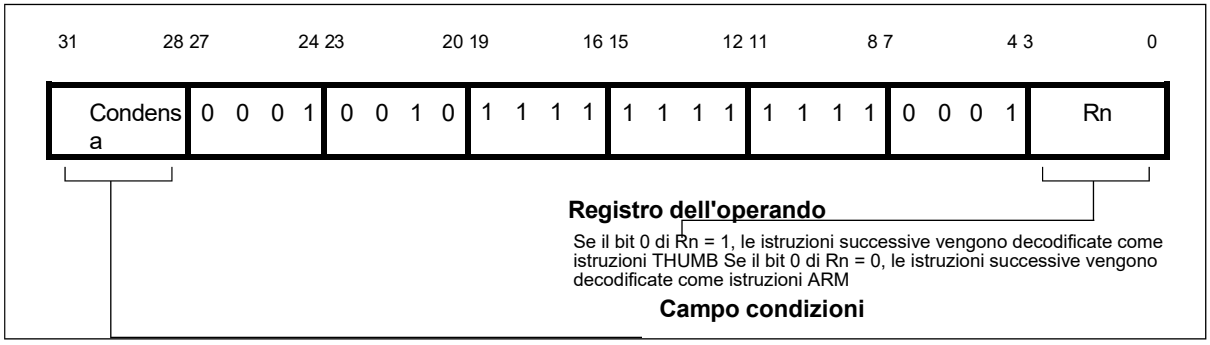


Figura 4-2: Istruzioni per le filiali e gli scambi

4.3.1 Tempi di ciclo delle istruzioni

L'istruzione BX richiede 2S + 1N cicli per essere eseguita, dove S e N sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.3.2 Sintassi dell'assemblatore

BX - branch and exchange.

```
BX{cond} Rn
```

{cond}Mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

Rnis è un'espressione che valuta un numero di registro valido.

4.3.3 Utilizzo di R15 come operando

Se R15 viene utilizzato come operando, il comportamento è indefinito.

Finale - Accesso

4.3.4 Esempi

```
        ADR R0, Into_THUMB + 1; genera l'indirizzo di destinazione della
        derivazione
                        e impostare il bit 0 alto - quindi
                        ; arrivare allo stato di THUMB.
        BX R0           ; diramazione e passaggio a THUMB
                        stato.
        CODE16          ; Assemblare il codice
    successivo come Into_THUMB      ; Istruzioni
    THUMB

    .
    .
        ADR R5, Back_to_ARM: Genera il target di diramazione nella parola
                        : allineato                                    ;
    indirizzo - quindi bit 0
                        è basso e quindi passa di nuovo ad ARM
                        stato.
        BX R5           ; diramazione e ritorno ad ARM
                        stato.
        .
        .
ALIGN               ; allineamento    di parole
        CODE32              ; Assemblare il codice
    successivo come     istruzioni ARM Back_to_ARM

        .
        .
```

Finale - Accesso libero

4.4 Diramazione e diramazione con collegamento (B, BL)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione***, a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-3: Istruzioni di diramazione***.

Figura 4-3: Istruzioni di diramazione

Le istruzioni di diramazione contengono un offset firmato a complemento di 2 a 24 bit. Questo viene spostato a sinistra di due bit, esteso di segno a 32 bit e aggiunto al PC. L'istruzione può quindi specificare una diramazione di +/- 32Mbyte. L'offset della derivazione deve tenere conto dell'operazione di prefetch, che fa sì che il PC sia 2 parole (8 byte) avanti rispetto all'istruzione corrente.

Le diramazioni oltre i +/- 32Mbyte devono utilizzare un offset o una destinazione assoluta che sia stata precedentemente caricata in un registro. In questo caso il PC deve essere salvato manualmente in R14 se è richiesta un'operazione di tipo Branch with Link.

4.4.1 Il bit di collegamento

La diramazione con collegamento (BL) scrive il vecchio PC nel registro di collegamento (R14) del banco corrente. Il valore di PC scritto in R14 viene adattato per tenere conto del prefetch e contiene l'indirizzo dell'istruzione successiva all'istruzione di diramazione e collegamento. Si noti che il CPSR non viene salvato con il PC e R14[1:0] viene sempre cancellato.

Per ritornare da una routine chiamata da Branch with Link si usa MOV PC,R14 se il registro di collegamento è ancora valido o LDM Rn!,{..PC} se il registro di collegamento è stato salvato su una pila puntata da Rn.

4.4.2 Tempi di ciclo delle istruzioni

Le istruzioni di diramazione e di diramazione con collegamento richiedono 2S + 1N cicli incrementali, dove S e N sono definiti in ***6.2 Tipi di ciclo a*** pagina 6-3.

Finale - Accesso

4.4.3 Sintassi dell'assemblatore

Gli elementi in {} sono facoltativi. Gli elementi in <> devono essere presenti.

```
B{L}{cond} <espressione>
```

{L} viene utilizzato per richiedere la forma di diramazione con collegamento dell'istruzione.
Se assente, R14 non sarà influenzato dall'istruzione.

{cond} è un mnemonico a due caratteri, come indicato nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. Se assente, verrà utilizzato AL (ALways). Se assente, verrà utilizzato AL (ALways).

<espressione> è la destinazione. L'assemblatore calcola l'offset.

4.4.4 Esempi

```
qui    BAL    qui       ; assembla a 0xEAFFFFFE (notare l'effetto di
                        ; offset PC).
       B      lì        ; condizione sempre utilizzata come predefinita.
       CMP    R1,#0     ; Confronta R1 con zero e dirama a fred
                        ; se R1 era zero, altrimenti
       continuare BEQ  fred ; continuare all'istruzione
       successiva.

       BLsub+ROM ; Richiama la subroutine all'indirizzo
       calcolato. ADDS R1,#1      ; Aggiungere 1 al registro 1,
       impostando i flag CPSR
                       sul risultato, quindi chiamare la
       subroutine se BLCC   sub    ; il flag C è azzerato, il
       che sarà l'opzione
                       a meno che R1 non contenga 0xFFFFFFFF.
```

4.5 Elaborazione dati

L'istruzione di elaborazione dati viene eseguita solo se la condizione è vera. Le condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

La codifica delle istruzioni è illustrata nella ***Figura 4-4: Istruzioni di elaborazione dati*** di seguito.

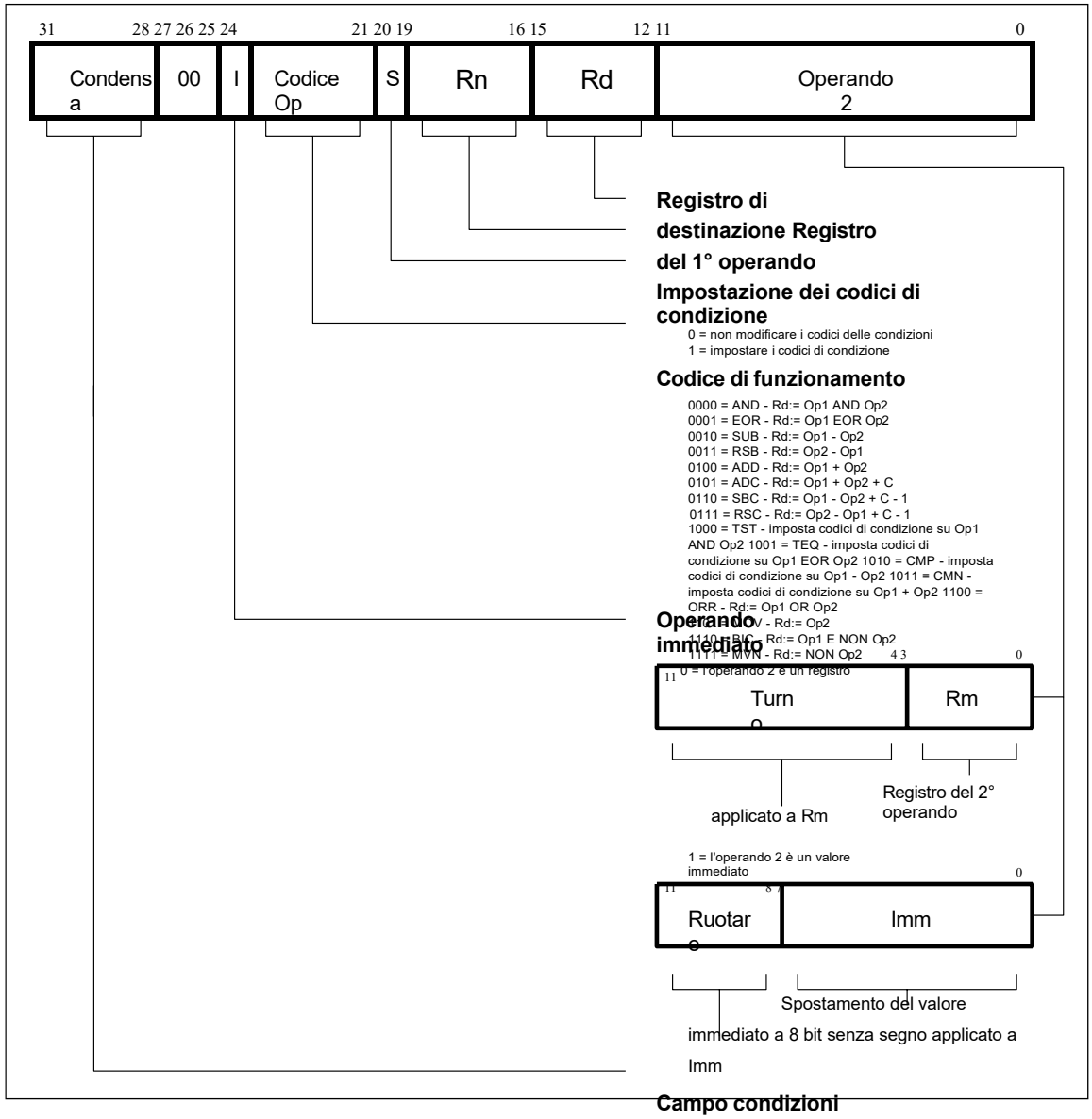


Figura 4-4: Istruzioni per l'elaborazione dei dati

L'istruzione produce un risultato eseguendo un'operazione aritmetica o logica specifica su uno o due operandi. Il primo operando è sempre un registro (Rn).

Il secondo operando può essere un registro shiftato (Rm) o un valore immediato a 8 bit ruotato (Imm), a seconda del valore del bit I dell'istruzione. I codici di condizione nel CPSR possono essere conservati o aggiornati come risultato di questa istruzione, in base al valore del bit S dell'istruzione.

Alcune operazioni (TST, TEQ, CMP, CMN) non scrivono il risultato su Rd. Vengono utilizzate solo per eseguire test e per impostare i codici di condizione sul risultato e hanno sempre il bit S impostato. Le istruzioni e i loro effetti sono elencati nella ***Tabella 4-3: Istruzioni di elaborazione dati ARM*.**

4.5.1 Bandiere CPSR

Le operazioni di elaborazione dei dati possono essere classificate come logiche o aritmetiche. Le operazioni logiche (AND, EOR, TST, TEQ, ORR, MOV, BIC, MVN) eseguono l'azione logica su tutti i bit corrispondenti dell'operando o degli operandi per produrre il risultato. Se il bit S è impostato (e Rd non è R15, vedi sotto), il flag V nel CPSR non sarà influenzato, il flag C sarà impostato sul carry out dal barrel shifter (o conservato quando l'operazione di shift è LSL #0), il flag Z sarà impostato se e solo se il risultato è costituito da tutti zeri e il flag N sarà impostato sul valore logico del bit 31 del risultato.

Mnemonico dell'assemblatore	Codice Op	Azione
E	0000	operando1 E operando2
EOR	0001	operando1 EOR operando2
SUB	0010	operando1 - operando2
RSB	0011	operando2 - operando1
ADD	0100	operando1 + operando2
ADC	0101	operando1 + operando2 + riporto
SBC	0110	operando1 - operando2 + riporto - 1
RSC	0111	operando2 - operando1 + riporto - 1
TST	1000	come AND, ma il risultato non viene scritto
TEQ	1001	come EOR, ma il risultato non viene scritto
CMP	1010	come SUB, ma il risultato non è scritto
CMN	1011	come ADD, ma il risultato non è scritto
ORR	1100	operando1 O operando2
MOV	1101	operando2 (l'operando1 viene ignorato)
BIC	1110	operando1 E NON operando2 (bit libero)
MVN	1111	NOT operando2 (l'operando1 viene ignorato)

Tabella 4-3: Istruzioni di elaborazione dati ARM

Le operazioni aritmetiche (SUB, RSB, ADD, ADC, SBC, RSC, CMP, CMN) trattano ogni operando come un intero a 32 bit (senza segno o con complemento a 2 firmato, i due sono equivalenti). Se il bit S è impostato (e Rd non è R15), il flag V nel CPSR viene impostato se si verifica un overflow nel bit 31 del risultato; questo può essere ignorato se gli operandi sono considerati senza segno, ma segnala un possibile errore se gli operandi sono a complemento a 2.

complemento firmato. Il flag C sarà impostato sul carry out del bit 31 dell'ALU, il flag Z sarà impostato se e solo se il risultato è zero e il flag N sarà impostato sul valore del bit 31 del risultato (indicando un risultato negativo se gli operandi sono considerati a complemento di 2 firmato).

4.5.2 Turni

Quando il secondo operando è specificato come registro shiftato, il funzionamento del barrel shifter è controllato dal campo Shift dell'istruzione. Questo campo indica il tipo di spostamento da eseguire (logico a sinistra o a destra, aritmetico a destra o rotativo a destra). L'entità dello spostamento del registro può essere contenuta in un campo immediato dell'istruzione o nel byte inferiore di un altro registro (diverso da R15). La codifica dei diversi tipi di shift è mostrata nella ***Figura 4-5: Operazioni di shift ARM***.

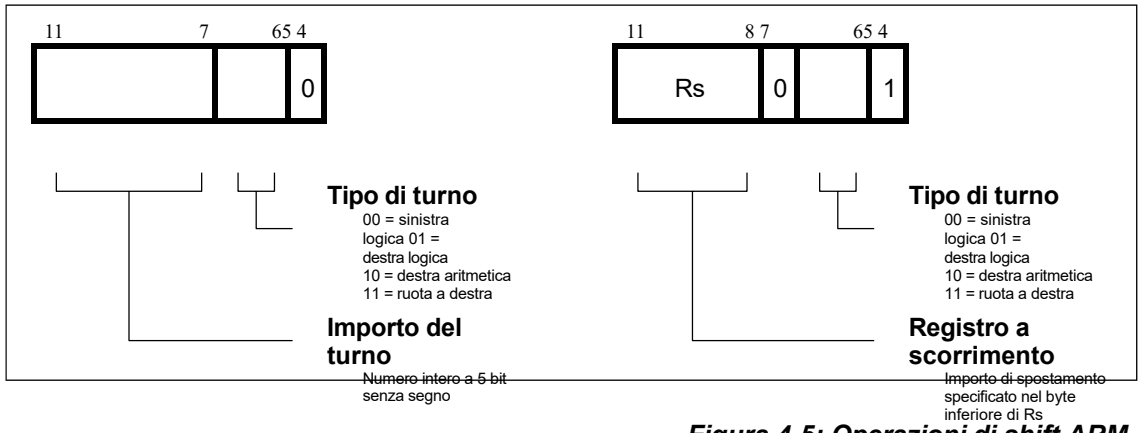


Figura 4-5: Operazioni di shift ARM

Istruzione quantità di turno specificata

Quando la quantità di spostamento è specificata nell'istruzione, è contenuta in un campo di 5 bit che può assumere qualsiasi valore da 0 a 31. Uno spostamento logico a sinistra (LSL) prende il contenuto di Rm e sposta ogni bit della quantità specificata in una posizione più significativa. I bit meno significativi del risultato vengono riempiti di zeri e i bit alti di Rm che non rientrano nel risultato vengono scartati, tranne che per il fatto che il bit meno significativo scartato diventa l'uscita del carry dello shift, che può essere inserito nel bit C del CPSR quando l'operazione dell'ALU è nella classe logica (vedi sopra). Ad esempio, l'effetto di f LSL #5 è mostrato nella ***Figura 4-6: spostamento logico a sinistra***.

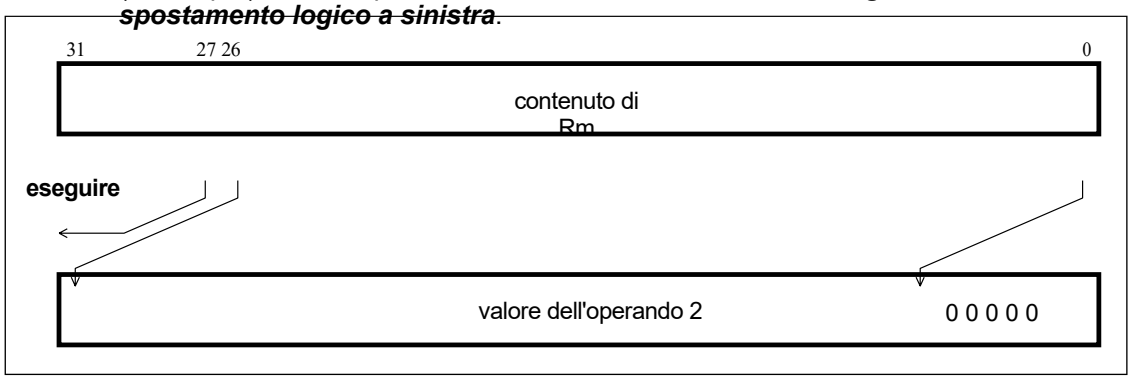


Figura 4-6: Spostamento logico a sinistra

Nota *LSL #0 è un caso particolare, in cui il carry out dello shifter è il vecchio valore del flag CPSR C.*

Il contenuto di Rm viene utilizzato direttamente come secondo operando.

Un logical shift right (LSR) è simile, ma il contenuto di Rm viene spostato in posizioni meno significative nel risultato. LSR #5 ha l'effetto mostrato nella ***Figura 4-7: spostamento logico a destra***.

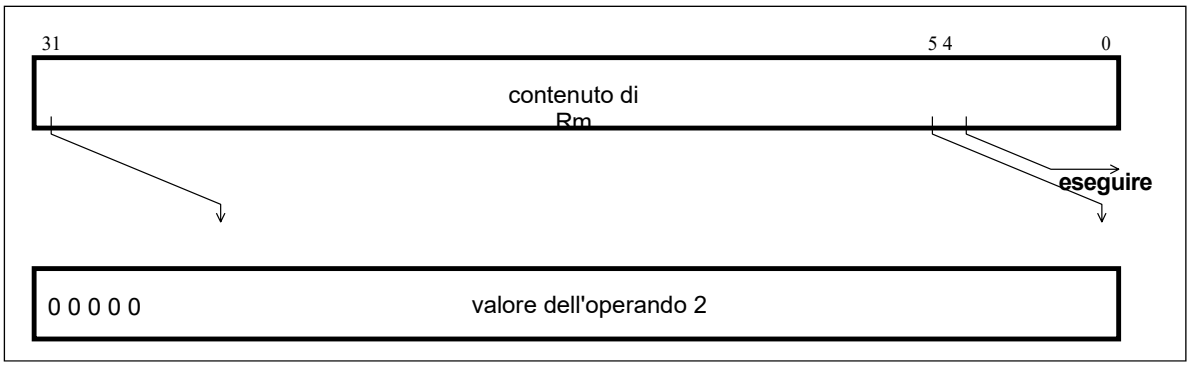


Figura 4-7: Spostamento logico a destra

La forma del campo di spostamento che ci si aspetterebbe corrispondesse a LSR #0 viene utilizzata per codificare LSR #32, che ha un risultato nullo con il bit 31 di Rm come uscita di riporto. Lo shift logico a destra zero è ridondante in quanto è uguale allo shift logico a sinistra zero, quindi l'assemblatore convertirà LSR #0 (e ASR #0 e ROR #0) in LSL #0 e permetterà di specificare LSR #32.

Uno shift aritmetico a destra (ASR) è simile allo shift logico a destra, tranne per il fatto che i bit alti vengono riempiti con il bit 31 di Rm invece che con gli zeri. In questo modo si preserva il segno nella notazione del complemento a 2. Ad esempio, l'ASR #5 è illustrato nella ***Figura 4-8: Spostamento aritmetico a destra***.

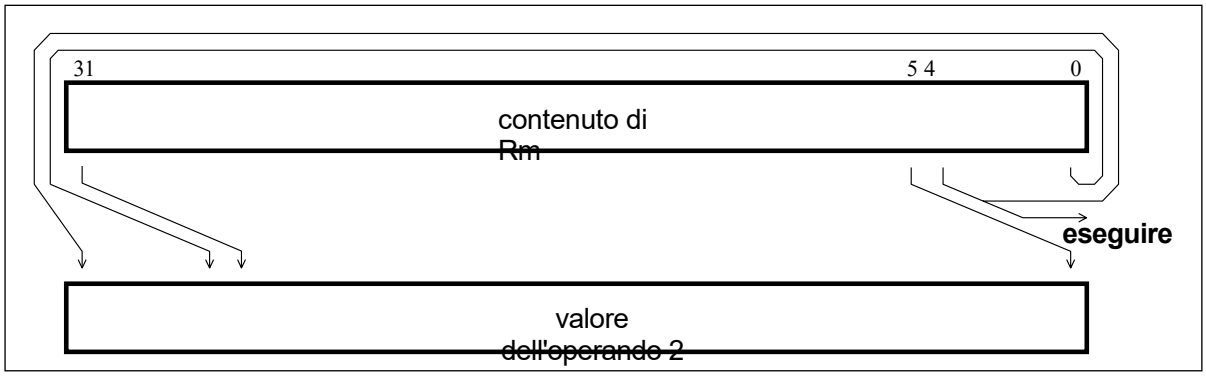
Figura 4-8: Spostamento aritmetico a destra

La forma del campo di spostamento che potrebbe dare ASR #0 è usata per codificare ASR #32. Il bit 31 di Rm viene nuovamente utilizzato come uscita di riporto e ogni bit dell'operando 2 è uguale al bit 31 di Rm. Il risultato è quindi tutto uno o tutto zero, a seconda del valore del bit 31 di Rm.

Le operazioni di rotazione a destra (ROR) riutilizzano i bit che "sforano" in un'operazione di spostamento logico a destra reintroducendoli all'estremità superiore del risultato, al posto degli zeri utilizzati per riempire l'estremità superiore nelle operazioni di destra logica. Ad esempio, ROR #5 è mostrato in ***Figura 4-9: Rotazione a destra*** a pagina 4-13.

Figura 4-9: Ruota a destra

Finale - Accesso libero



La forma del campo di shift che potrebbe dare ROR #0 è utilizzata per codificare una funzione speciale del barrel shifter, la rotazione a destra estesa (RRX). Si tratta di una rotazione a destra di un bit della quantità di 33 bit formata dall'aggiunta del flag CPSR C all'estremità più significativa del contenuto di Rm, come illustrato nella ***Figura 4-10: Rotazione a destra estesa***.

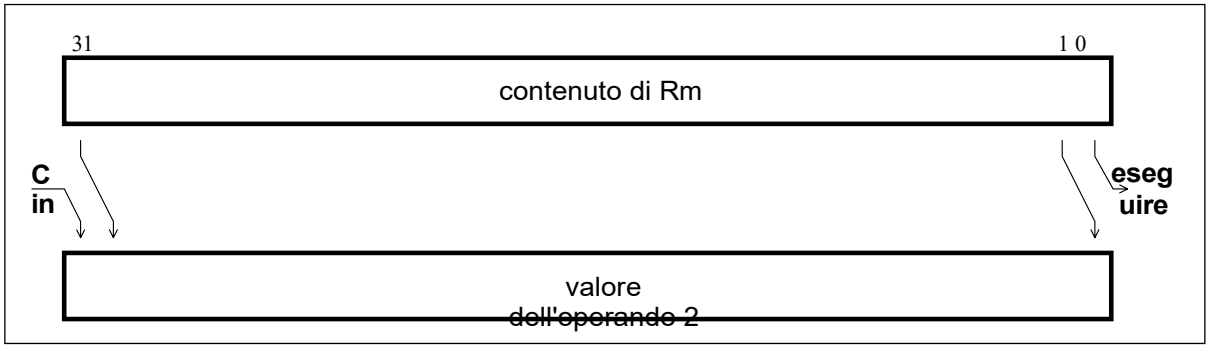


Figura 4-10: Ruota a destra estesa

Registra la quantità di spostamento specificata

Solo il byte meno significativo del contenuto di Rs viene utilizzato per determinare la quantità di spostamento. Rs può essere un qualsiasi registro generale diverso da R15.

Se questo byte è zero, il contenuto invariato di Rm verrà utilizzato come secondo operando e il vecchio valore del flag CPSR C verrà passato come uscita di riporto del traslatore.

Se il byte ha un valore compreso tra 1 e 31, il risultato dello shift corrisponderà esattamente a quello di un'istruzione shift specificata con lo stesso valore e la stessa operazione di shift.

Se il valore del byte è 32 o più, il risultato sarà un'estensione logica dello spostamento descritto in precedenza:

1. LSL per 32 ha risultato zero, carry out uguale al bit 0 di Rm.
2. LSL per più di 32 ha risultato zero, eseguire zero.
3. LSR per 32 ha risultato zero, carry out uguale al bit 31 di Rm.
4. LSR per più di 32 ha risultato zero, eseguire zero.
5. ASR per 32 o più ha risultato riempito e carry out pari al bit 31 di Rm.
6. ROR per 32 ha un risultato pari a Rm, carry out pari al bit 31 di Rm.
7. Il ROR per n dove n è maggiore di 32 darà lo stesso risultato ed esecuzione del ROR per n-32; quindi sottrarre ripetutamente 32 da n fino a quando la somma è nell'intervallo 1-32 e vedere sopra.

Nota *Lo zero nel bit 7 di un'istruzione con shift controllato dal registro è obbligatorio; un uno in questo bit farà sì che l'istruzione sia un'istruzione di moltiplicazione o un'istruzione non definita.*

4.5.3 L'operando immediato ruota

Il campo rotate dell'operando immediato è un intero senza segno a 4 bit che specifica un'operazione di spostamento sul valore immediato a 8 bit. Questo valore viene esteso a 32 bit e quindi sottoposto a un'operazione di rotazione a destra di due volte il valore del campo rotate. In questo modo è possibile generare molte costanti comuni, ad esempio tutte le potenze di 2.

4.5.4 Scrittura su R15

Quando Rd è un registro diverso da R15, i flag del codice di condizione nel CPSR possono essere aggiornati dai flag dell'ALU come descritto sopra.

Quando Rd è R15 e il flag S dell'istruzione non è impostato, il risultato dell'operazione viene collocato in R15 e il CPSR non viene influenzato.

Quando Rd è R15 e il flag S è impostato, il risultato dell'operazione viene collocato in R15 e l'SPSR corrispondente alla modalità corrente viene spostato nel CPSR. Ciò consente di effettuare cambi di stato che ripristinano atomicamente sia il PC che il CPSR. Questa forma di istruzione non deve essere utilizzata in modalità Utente.

4.5.5 Utilizzo di R15 come operando

Se R15 (il PC) viene utilizzato come operando in un'istruzione di elaborazione dati, il registro viene utilizzato direttamente.

Il valore PC sarà l'indirizzo dell'istruzione, più 8 o 12 byte a causa del prefetching dell'istruzione. Se la quantità di shift è specificata nell'istruzione, il PC sarà 8 byte avanti. Se si utilizza un registro per specificare la quantità di shift, il PC sarà 12 byte avanti.

4.5.6 Opcodes TEQ, TST, CMP e CMN

Nota *TEQ, TST, CMP e CMN non scrivono il risultato della loro operazione, ma impostano i flag nel CPSR. L'assemblatore dovrebbe sempre impostare il flag S per queste istruzioni, anche se ciò non è specificato nella mnemonica.*

La forma TEQP dell'istruzione TEQ utilizzata nei precedenti processori ARM non deve essere utilizzata: al suo posto devono essere utilizzate le operazioni di trasferimento PSR.

L'azione di TEQP nell'ARM7TDMI-S è quella di spostare SPSR_<mode> nel CPSR se il processore è in modalità privilegiata e di non fare nulla se è in modalità utente.

4.5.7 Tempi di ciclo delle istruzioni

Le istruzioni di elaborazione dei dati variano per il numero di cicli incrementali effettuati, come segue:

Tipo di elaborazione	Cicli
Elaborazione normale dei dati	1S
Elaborazione dei dati con spostamento specificato dal registro	1S + 1I
Elaborazione dati con PC scritto	2S + 1N
Elaborazione dati con registro specificato shift e scrittura su PC	2S + 1N + 1I

Tabella 4-4: Tempi di ciclo incrementali

S, N e I sono definiti in ***6.2 Tipi di ciclo***, pagina 6-3.

4.5.8 Sintassi dell'assemblatore

1 MOV,MVN (istruzioni a singolo operando).

```
pcode>{cond}{S} Rd,<Op2>
```

2 CMP,CMN,TEQ,TST (istruzioni che non producono un risultato).

```
<opcode>{cond} Rn,<Op2>
```

3 AND,EOR,SUB,RSB,ADD,ADC,SBC,RSC,ORR,BIC

```
<opcode>{cond}{S} Rd,Rn,<Op2>
```

dove:

<Op2> è Rm{,<shift>} oppure,<#espressione>

{cond} è un codice di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

{S}impostare i codici di condizione se S è presente (implicito per CMP, CMN, TEQ, TST).

Rd, Rn e Rmare sono espressioni che valutano un numero di registro.

<#espressione> se viene utilizzata, l'assemblatore cercherà di generare un campo immediato a 8 bit spostato che corrisponda all'espressione. Se ciò è impossibile, viene dato un errore.

<shift> è <nome spostamento> <registro> o <nome spostamento> #espressione, o RRX (rotazione a destra di un bit con estensione).

<shiftname>sono : ASL, LSL, LSR, ASR, ROR. (ASL è un sinonimo di LSL, si assemblano allo stesso codice).

4.5.9 Esempi

```
ADDEQR2,R4,R5        ; se il flag Z è impostato, R2:=R4+R5
TEQSR4,#3            ; verificare l'uguaglianza tra R4 e
3.
                     (La S è di fatto superflua, dato che l'elemento
                     L'assemblatore lo inserisce
automaticamente). SUB R4,R5,R7,LSR R2 ; Spostamento logico a
destra di R7 del numero in
                     ; il byte inferiore di R2, sottrarre il risultato
                     ; da R5 e inserire la risposta in R4.
MOV PC,R14           ; Ritorno dalla subroutine.
MOVSPC,R14           ; Ritorno dall'eccezione e ripristino di CPSR
                     ; da SPSR_mode.
```

4.6 Trasferimento PSR (MRS, MSR)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

Le istruzioni MRS e MSR sono formate da un sottoinsieme delle operazioni di elaborazione dati e sono implementate utilizzando le istruzioni TEQ, TST, CMN e CMP senza il flag S impostato. La codifica è illustrata nella ***Figura 4-11: Trasferimento PSR*** a pagina 4-18.

Queste istruzioni consentono di accedere ai registri CPSR e SPSR. L'istruzione MRS consente di spostare il contenuto del registro CPSR o SPSR_<mode> in un registro generale. L'istruzione MSR consente di spostare il contenuto di un registro generale nel registro CPSR o SPSR_<mode>.

L'istruzione MSR consente inoltre di trasferire un valore immediato o il contenuto di un registro ai flag dei codici di condizione (N, Z, C e V) di CPSR o SPSR_<mode> senza influenzare i bit di controllo. In questo caso, i primi quattro bit del contenuto del registro specificato o del valore immediato a 32 bit vengono scritti nei primi quattro bit del PSR corrispondente.

4.6.1 Restrizioni dell'operando

2 In modalità Utente, i bit di controllo del CPSR sono protetti dalla modifica, quindi è possibile modificare solo i flag del codice di condizione del CPSR. In altre modalità (privilegiate) è possibile modificare l'intero CPSR.

   Si noti che il software non deve mai modificare lo stato del bit T nel CPSR. In tal caso, il processore entrerà in uno stato imprevedibile.

3 Il registro SPSR a cui si accede dipende dalla modalità al momento dell'esecuzione. Ad esempio, solo SPSR_fiq è accessibile quando il processore è in modalità FIQ.

4 Non si deve specificare R15 come registro di origine o di destinazione.

5 Inoltre, non tentate di accedere a un SPSR in modalità Utente, poiché non esiste un registro di questo tipo.

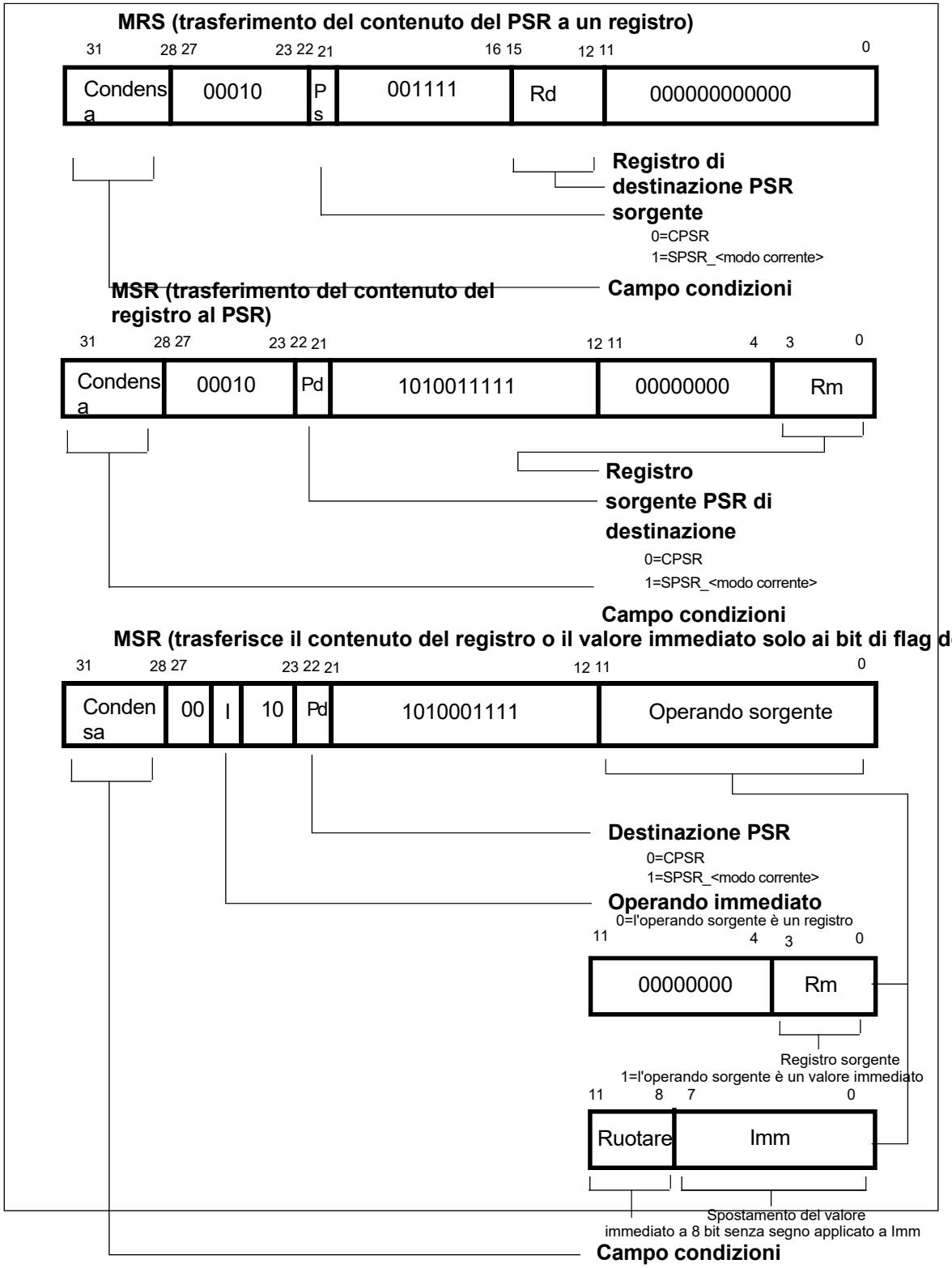


Figura 4-11: Trasferimento PSR

4.6.2 Bit riservati

Solo dodici bit del PSR sono definiti in ARM7TDMI-S (N,Z,C,V,I,F, T e M[4:0]); i restanti bit sono riservati per l'uso in versioni future del processore. Per una descrizione completa dei bit del PSR, consultare la ***Figura 3-6: Formato del registro di stato del programma*** a pagina 3-12.

Per garantire la massima compatibilità tra i programmi ARM7TDMI-S e i processori futuri, è necessario osservare le seguenti regole:

- I bit riservati devono essere conservati quando si modifica il valore in un PSR.
- I programmi non dovrebbero fare affidamento su valori specifici dei bit riservati quando controllano lo stato del PSR, poiché potrebbero essere letti come uno o zero nei processori futuri.

Per modificare i bit di controllo di un qualsiasi registro PSR si deve quindi utilizzare una strategia di lettura-modifica-scrittura, che consiste nel trasferire il registro PSR appropriato in un registro generale utilizzando l'istruzione MRS, modificare solo i bit interessati e quindi trasferire nuovamente il valore modificato al registro PSR utilizzando l'istruzione MSR.

Esempio

La sequenza seguente esegue un cambio di modalità:

```
MRS R0,CPSR                          ; Acquisire una copia del CPSR.
BIC R0,R0,#0x1F                      ; Cancella i bit di modalità.
ORR R0,R0,#nuovamodalità             ; seleziona la nuova modalità
MSR CPSR,R0                          ; Riscrivere l'informazione
modificata
                                     CPSR.
```

Quando lo scopo è semplicemente quello di modificare i flag del codice di condizione in un PSR, è possibile scrivere un valore direttamente sui bit dei flag senza disturbare i bit di controllo. La seguente istruzione imposta i flag N, Z, C e V:

```
MSR CPSR_flg,#0xF0000000             ; Impostazione di tutti i flag
                                     indipendentemente dalla loro
                                     ; stato precedente (non
                                     ; influisce su tutti i bit di
                                     controllo).
```

Non si deve tentare di scrivere un valore immediato di 8 bit nell'intero PSR, poiché tale operazione non può preservare i bit riservati.

4.6.3 Tempi di ciclo delle istruzioni

I trasferimenti PSR richiedono 1S cicli incrementali, dove S è definito in ***6.2 Tipi di ciclo.*** a pagina 6-3.

4.6.4 Sintassi dell'assemblatore

1 MRS - trasferire il contenuto del PSR in un registro

```
MRS{cond} Rd,<psr>
```

2 MSR - trasferire il contenuto del registro al PSR

```
MSR{cond} <psr>,Rm
```

3 MSR - trasferire il contenuto del registro solo ai bit di flag del PSR

```
MSR{cond} <psrf>,Rm
```

I quattro bit più significativi del contenuto del registro vengono scritti rispettivamente nei flag N,Z,C e V.

4 MSR - trasferisce il valore immediato solo ai bit di flag del PSR

```
MSR{cond} <psrf>,<#espressione>
```

L'espressione deve simboleggiare un valore di 32 bit di cui i quattro bit più significativi vengono scritti rispettivamente nei flag N, Z, C e V.

Chiave:

{cond}	Mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.
Rd e Rm	sonoespressioni che valutano un numero di registro diverso da R15
<psr>	è CPSR, CPSR_all, SPSR o SPSR_all. (CPSR e CPSR_all sono sinonimi, così come SPSR e SPSR_all).
<psrf>	è CPSR_flg o SPSR_flg
<#espressione>	quando viene utilizzata, l'assemblatore cercherà di generare un campo immediato a 8 bit spostato che corrisponda all'espressione. Se ciò è impossibile, viene dato un errore.

4.6.5 Esempi

In modalità Utente le istruzioni si comportano come segue:

```
MSR  CPSR_all,Rm  ;  CPSR[31:28]  <-  Rm[31:28]
MSR  CPSR_flg,Rm  ;  CPSR[31:28]  <-  Rm[31:28]
MSR CPSR_flg,#0xA0000000; CPSR[31:28] <- 0xA
                    ;(imposta N,C; cancella Z,V)
MRS Rd,CPSR         ; Rd[31:0] <- CPSR[31:0]
```

Nelle modalità privilegiate le istruzioni si comportano come segue:

```
MSRCPSR_all,Rm      ; CPSR[31:0]  <- Rm[31:0]
MSRCPSR_flg,Rm      ;CPSR[31:28] <- Rm[31:28]
MSR CPSR_flg,#0x50000000; CPSR[31:28] <- 0x5
                    ;(impostare Z,V; cancellare N,C)
MRS Rd,CPSR         ; Rd[31:0] <- CPSR[31:0]
MSR SPSR_all,Rm     ;SPSR_<mode>[31:0]<- Rm[31:0]
MSRSPSR_flg,Rm      ;SPSR_<mode>[31:28] <- Rm[31:28]
MSR SPSR_flg,#0xC0000000; SPSR_<mode>[31:28] <- 0xC
                    ;(imposta N,Z; cancella C,V)
MRS Rd,SPSR         ; Rd[31:0] <- SPSR_<mode>[31:0]
```

4.7 Moltiplicazione e moltiplicazione-accumulazione (MUL, MLA)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-12: Istruzioni di moltiplicazione***.

Le istruzioni di moltiplicazione e moltiplicazione-accumulazione utilizzano un algoritmo di Booth a 8 bit per eseguire la moltiplicazione di interi.

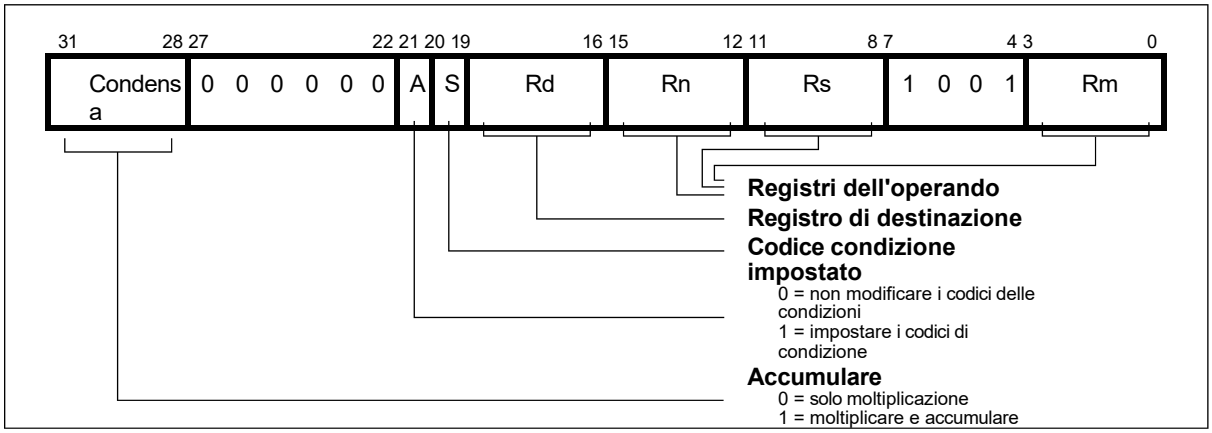


Figura 4-12: Istruzioni di moltiplicazione

La forma moltiplicativa dell'istruzione dà Rd:=Rm*Rs. Rn viene ignorato e dovrebbe essere impostato a zero per compatibilità con eventuali aggiornamenti futuri del set di istruzioni.

La forma moltiplicata-accumulata dà Rd:=Rm*Rs+Rn, che può far risparmiare un'istruzione ADD esplicita in alcune circostanze.

Entrambe le forme di istruzione funzionano su operandi che possono essere considerati come numeri interi firmati (complemento a 2) o non firmati.

I risultati di una moltiplicazione con segno e di una moltiplicazione senza segno di operandi a 32 bit differiscono solo per i 32 bit superiori; i 32 bit inferiori dei risultati con e senza segno sono identici. Poiché queste istruzioni producono solo i 32 bit inferiori di una moltiplicazione, possono essere utilizzate sia per le moltiplicazioni con segno che per quelle senza segno.

Ad esempio, si consideri la moltiplicazione degli operandi:

Operando AOperando BRrisultato

0xFFFFFFF6 0x00000010xFFFF38

Se gli operandi sono interpretati come firmati

L'operando A ha il valore -10, l'operando B ha il valore 20 e il risultato è -200, che viene rappresentato correttamente come 0xFFFF38.

Se gli operandi sono interpretati come non firmati

L'operando A ha il valore 4294967286, l'operando B ha il valore 20 e il risultato è 85899345720, rappresentato come 0x13FFFF38, quindi i 32 bit meno significativi sono 0xFFFF38.

4.7.1 Restrizioni dell'operando

Il registro di destinazione Rd non deve essere uguale al registro dell'operando Rm. R15 non deve essere utilizzato come operando o come registro di destinazione.

Finale - Accesso

Tutte le altre combinazioni di registri daranno risultati corretti e Rd, Rn e Rs possono utilizzare lo stesso registro quando necessario.

4.7.2 Bandiere CPSR

L'impostazione dei flag CPSR è opzionale ed è controllata dal bit S dell'istruzione. I flag N (Negativo) e Z (Zero) vengono impostati correttamente sul risultato (N viene reso uguale al bit 31 del risultato e Z viene impostato se e solo se il risultato è zero). Il flag C (Carry) viene impostato su un valore privo di significato e il flag V (oVerflow) non viene influenzato.

4.7.3 Tempi di ciclo delle istruzioni

MUL impiega 1S + mI e MLA 1S + (m+1)I cicli per essere eseguito, dove S e I sono definiti in ***6.2 Tipi di ciclo a*** pagina 6-3.

mis il numero di cicli dell'array di moltiplicatori a 8 bit necessari per completare la moltiplicazione, controllato dal valore dell'operando del moltiplicatore specificato da Rs. I suoi valori possibili sono i seguenti

1 se i bit [32:8] dell'operando moltiplicatore sono tutti zero o tutti uno.

2 se i bit [32:16] dell'operando moltiplicatore sono tutti zero o tutti uno.

3 se i bit [32:24] dell'operando moltiplicatore sono tutti zero o tutti uno.

4 in tutti gli altri casi.

4.7.4 Sintassi dell'assemblatore

```
MUL{cond}{S} Rd,Rm,Rs
MLA{cond}{S} Rd,Rm,Rs,Rn
```

{cond}mnemonico di condizione a due caratteri. Vedere la ***Tabella 4-2: Riepilogo dei codici delle condizioni*** a pagina 4-5.

{S}imposta i codici delle condizioni se S è presente

Rd, Rm, Rs e Rn sono espressioni che valutano un numero di registro diverso da di R15.

4.7.5 Esempi

```
MUL       R1,R2,R3      ; R1:=R2*R3
MLAEQSR1  ,R2,R3,R4 ; Condizionatamente R1:=R2*R3+R4,
                        ; impostazione dei codici di condizione.
```

4.8 Moltiplicare lungo e moltiplicare-accumulare lungo (MULL,MLAL)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-13: Istruzioni di moltiplicazione lunga***.

Le istruzioni multiply long eseguono una moltiplicazione intera su due operandi a 32 bit e producono risultati a 64 bit. Le moltiplicazioni con e senza segno, ciascuna con accumulo opzionale, danno origine a quattro varianti.

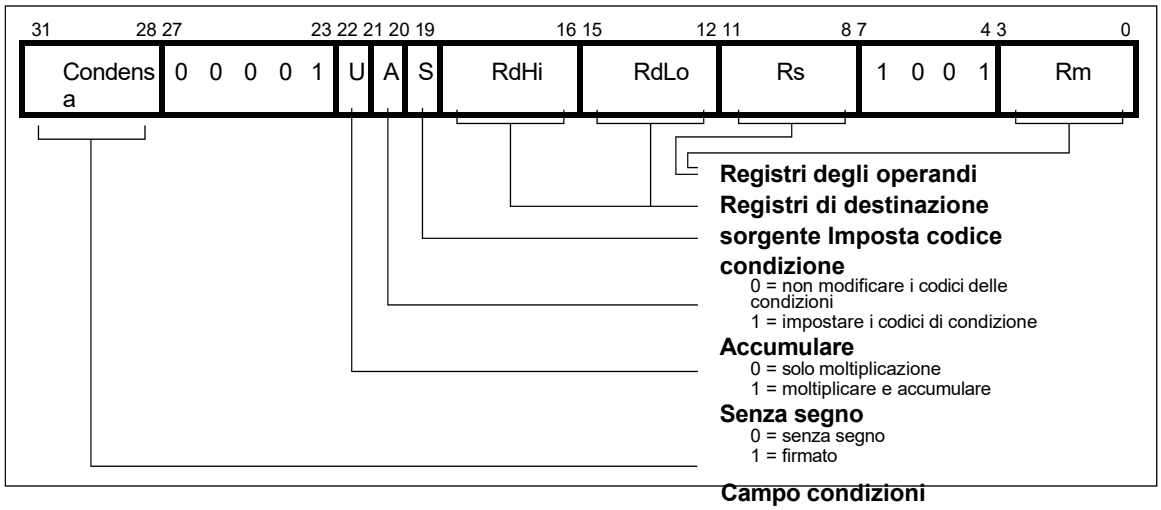


Figura 4-13: Istruzioni di moltiplicazione lunga

Le forme di moltiplicazione (UMULL e SMULL) prendono due numeri a 32 bit e li moltiplicano per produrre un risultato a 64 bit della forma RdHi,RdLo := Rm * Rs. I 32 bit inferiori del risultato a 64 bit vengono scritti in RdLo, i 32 bit superiori del risultato vengono scritti in RdHi.

Le forme di moltiplicazione-accumulazione (UMLAL e SMLAL) prendono due numeri a 32 bit, li moltiplicano e aggiungono un numero a 64 bit per produrre un risultato a 64 bit della forma RdHi,RdLo := Rm
* Rs + RdHi,RdLo. I 32 bit inferiori del numero a 64 bit da sommare vengono letti da RdLo. I 32 bit superiori del numero a 64 bit da sommare vengono letti da RdHi. I 32 bit inferiori del risultato a 64 bit vengono scritti in RdLo. I 32 bit superiori del risultato a 64 bit vengono scritti in RdHi.

Le istruzioni UMULL e UMLAL trattano tutti i loro operandi come numeri binari senza segno e scrivono un risultato a 64 bit senza segno. Le istruzioni SMULL e SMLAL trattano tutti i loro operandi come numeri firmati a complemento di due e scrivono un risultato firmato a complemento di due a 64 bit.

4.8.1 Restrizioni dell'operando

2 R15 non deve essere utilizzato come operando o come registro di destinazione.

3 RdHi, RdLo e Rm devono specificare registri diversi.

4.8.2 Bandiere CPSR

L'impostazione dei flag CPSR è opzionale ed è controllata dal bit S dell'istruzione. I flag N e Z vengono impostati correttamente sul risultato (N è uguale al bit 63 del risultato, Z è impostato se e solo se tutti i 64 bit del risultato sono zero). I flag C e V sono impostati su valori non significativi.

4.8.3 Tempi di ciclo delle istruzioni

L'esecuzione di MULL richiede 1S + (m+1)I e quella di MLAL 1S + (m+2)I cicli, dove *m* è il numero di cicli dell'array di moltiplicatori a 8 bit necessari per completare la moltiplicazione, controllata dal valore dell'operando del moltiplicatore specificato da Rs.

I valori possibili sono i seguenti:

Per le istruzioni firmate SMULL, SMLAL:

1 se i bit [31:8] dell'operando moltiplicatore sono tutti zero o tutti uno.

2 se i bit [31:16] dell'operando moltiplicatore sono tutti zero o tutti uno.

3 se i bit [31:24] dell'operando moltiplicatore sono tutti zero o tutti uno.

4 in tutti gli altri casi.

Per le istruzioni senza segno UMULL, UMLAL:

1 se i bit [31:8] dell'operando del moltiplicatore sono tutti zero.

2 se i bit [31:16] dell'operando del moltiplicatore sono tutti zero.

3 se i bit [31:24] dell'operando del moltiplicatore sono tutti zero.

4 in tutti gli altri casi.

S e I sono definite in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.8.4 Sintassi dell'assemblatore

Mnemonico	Descrizione	Scopo
UMULL{cond}{S} RdLo,RdHi,Rm,Rs	Senza segno Moltiplica lungo	32 x 32 = 64
UMLAL{cond}{S} RdLo,RdHi,Rm,Rs	Moltiplicare e accumulare senza segno Lungo	32 x 32 + 64 = 64
SMULL{cond}{S} RdLo,RdHi,Rm,Rs	Moltiplicazione firmata Lunga	32 x 32 = 64
SMLAL{cond}{S} RdLo,RdHi,Rm,Rs	Moltiplicazione e accumulazione a segni lunghi	32 x 32 + 64 = 64

Tabella 4-5: Descrizioni della sintassi dell'assemblatore

dove:

{cond}Mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

{S}imposta i codici delle condizioni se S è presente

RdLo, RdHi, Rm, Rs sono espressioni che valutano un numero di registro diverso da di R15.

4.8.5 Esempi

```
UMULLR1   ,R4,R2,R3 ; R4,R1:=R2*R3
UMLALSR1  ,R5,R2,R3 ; R5,R1:=R2*R3+R5,R1 anche l'impostazione
                      Codici di condizione
```

4.9 Trasferimento dati singolo (LDR, STR)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-14: Istruzioni per il trasferimento di dati singoli***.

Le istruzioni di trasferimento di dati singoli vengono utilizzate per caricare o memorizzare singoli byte o parole di dati. L'indirizzo di memoria utilizzato nel trasferimento viene calcolato aggiungendo o sottraendo un offset a un registro di base.

Il risultato di questo calcolo può essere riscritto nel registro di base se è richiesta l'autoindicizzazione.

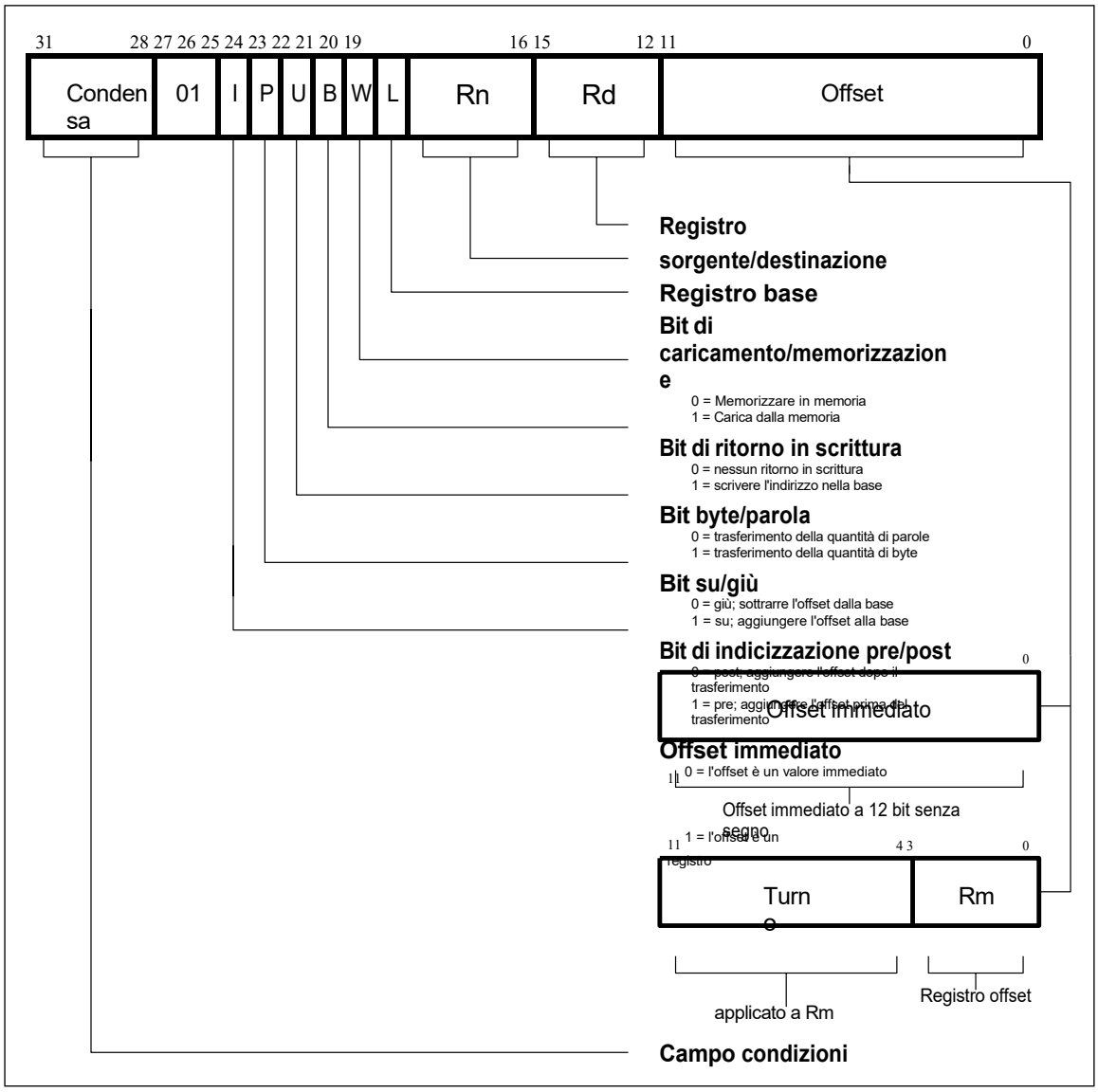


Figura 4-14: Istruzioni per il trasferimento di dati singoli

Finale - Accesso

4.9.1 Offset e indicizzazione automatica

L'offset dalla base può essere un valore immediato binario a 12 bit senza segno nell'istruzione, oppure un secondo registro (eventualmente spostato in qualche modo). L'offset può essere aggiunto (U=1) o sottratto (U=0) al registro di base Rn. La modifica dell'offset può essere eseguita prima (pre-indicizzato, P=1) o dopo (post-indicizzato, P=0) l'utilizzo della base come indirizzo di trasferimento.

Il bit W fornisce modalità di indirizzamento opzionali di incremento e decremento automatico. Il valore di base modificato può essere riscritto nella base (W=1), oppure può essere mantenuto il vecchio valore di base (W=0). Nel caso dell'indirizzamento post-indicizzato, il bit di write back è ridondante e deve essere impostato a zero, poiché il vecchio valore di base può essere mantenuto impostando l'offset a zero. Pertanto, i trasferimenti di dati post-indicizzati riscrivono sempre la base modificata. L'unico uso del bit W in un trasferimento di dati post-indicizzato è nel codice in modalità privilegiata, dove l'impostazione del bit W forza la modalità non privilegiata per il trasferimento, consentendo al sistema operativo di generare un indirizzo utente in un sistema in cui l'hardware di gestione della memoria fa un uso adeguato di questo hardware.

4.9.2 Offset del registro spostato

Gli 8 bit di controllo dello spostamento sono descritti nella sezione delle istruzioni di elaborazione dati. Tuttavia, le quantità di spostamento specificate nel registro non sono disponibili in questa classe di istruzioni. Vedere ***4.5.2 Spostamenti a*** pagina 4-12.

4.9.3 Byte e parole

Questa classe di istruzioni può essere utilizzata per trasferire un byte (B=1) o una parola (B=0) tra un registro ARM7TDMI-S e la memoria.

L'azione delle istruzioni LDR(B) e STR(B) è influenzata dal segnale di controllo **BIGEND.** Le due possibili configurazioni sono descritte di seguito.

Configurazione little endian

Un caricamento a byte (LDRB) si aspetta che i dati siano sugli ingressi del bus dati da 7 a 0 se l'indirizzo fornito è su un confine di parola, sugli ingressi del bus dati da 15 a 8 se si tratta di un indirizzo di parola più un byte, e così via. Il byte selezionato viene inserito negli 8 bit inferiori del registro di destinazione, mentre i bit rimanenti del registro vengono riempiti di zeri. Vedere la ***Figura 3-2: Indirizzi little-endian di byte all'interno di parole a*** pagina 3-4.

Un byte store (STRB) ripete gli 8 bit inferiori del registro sorgente quattro volte attraverso le uscite del bus dati da 31 a 0. Il sistema di memoria esterno deve attivare il sottosistema di byte appropriato per memorizzare i dati.

Un caricamento a parola (LDR) utilizza normalmente un indirizzo allineato alla parola. Tuttavia, un offset dell'indirizzo rispetto al confine della parola causerà la rotazione dei dati nel registro in modo che il byte indirizzato occupi i bit da 0 a 7. Ciò significa che le mezze parole accedute con offset 0 e 2 rispetto al confine della parola saranno caricate correttamente nei bit da 0 a 15 del registro. Ciò significa che le mezze parole a cui si accede con gli offset 0 e 2 dal confine di parola saranno caricate correttamente nei bit da 0 a 15 del registro. Sono quindi necessarie due operazioni di shift per cancellare o estendere il segno dei 16 bit superiori. Questa procedura è illustrata nella ***Figura 4-15: Indirizzamento offset Little endian*** a pagina 4-28.

Finale - Accesso libero

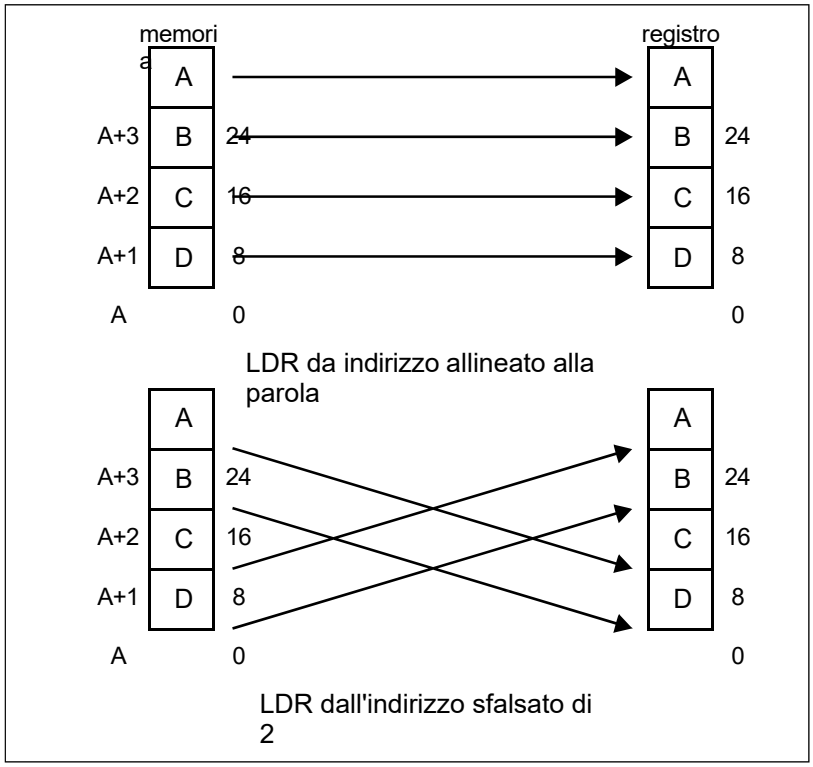


Figura 4-15: Indirizzamento offset little endian

Un word store (STR) deve generare un indirizzo allineato alla parola. La parola presentata sul bus dati non viene influenzata se l'indirizzo non è allineato alla parola. In altre parole, il bit 31 del registro memorizzato appare sempre sull'uscita 31 del bus dati.

Configurazione big endian

Un caricamento a byte (LDRB) si aspetta che i dati siano sugli ingressi del bus dati da 31 a 24 se l'indirizzo fornito è su un confine di parola, sugli ingressi del bus dati da 23 a 16 se si tratta di un indirizzo di parola più un byte, e così via. Il byte selezionato viene inserito negli 8 bit inferiori del registro di destinazione e i bit rimanenti del registro vengono riempiti di zeri. Si veda la ***Figura 3-1: indirizzi big-endian di byte all'interno di parole*** a pagina 3-4.

Un byte store (STRB) ripete gli 8 bit inferiori del registro sorgente quattro volte attraverso le uscite del bus dati da 31 a 0. Il sistema di memoria esterno deve attivare il sottosistema di byte appropriato per memorizzare i dati.

Il caricamento di una parola (LDR) deve generare un indirizzo allineato alla parola. Un offset dell'indirizzo di 0 o 2 rispetto a un confine di parola farà sì che i dati vengano ruotati nel registro in modo che il byte indirizzato occupi i bit da 31 a 24. Ciò significa che le mezze parole accessibili con questi offset verranno caricate correttamente nei bit da 16 a 31 del registro. Ciò significa che le mezze parole a cui si accede con questi offset vengono caricate correttamente nei bit da 16 a 31 del registro. È quindi necessaria un'operazione di shift per spostare (ed eventualmente estendere il segno) i dati nei 16 bit inferiori. Un offset di indirizzo di 1 o 3 rispetto a un confine di parola causerà la rotazione dei dati nel registro in modo che il byte indirizzato occupi i bit da 15 a 8.

Un word store (STR) deve generare un indirizzo allineato alla parola. La parola presentata sul bus dati non viene influenzata se l'indirizzo non è allineato alla parola. In altre parole, il bit 31 del registro memorizzato appare sempre sull'uscita 31 del bus dati.

Finale - Accesso

4.9.4 Uso di R15

Il ritorno in scrittura non deve essere specificato se R15 è indicato come registro di base (Rn). Quando si utilizza R15 come registro di base, occorre ricordare che esso contiene un indirizzo a 8 byte di distanza dall'indirizzo dell'istruzione corrente.

R15 non deve essere specificato come offset del registro (Rm).

Quando R15 è il registro di origine (Rd) di un'istruzione di memorizzazione di registri (STR), il valore memorizzato sarà l'indirizzo dell'istruzione più 12.

4.9.5 Limitazione dell'uso del registro di base

Quando è configurato per le interruzioni tardive, il seguente esempio di codice è difficile da sbrogliare poiché il registro di base, Rn, viene aggiornato prima dell'avvio del gestore delle interruzioni. A volte può essere impossibile calcolare il valore iniziale.

Dopo un'interruzione, l'esempio di codice seguente è difficile da sbrogliare perché il registro di base, Rn, viene aggiornato prima dell'avvio del gestore dell'interruzione. A volte può essere impossibile calcolare il valore iniziale.

Esempio:

```
LDR R0,[R1],R1
```

Pertanto, non si deve utilizzare un LDR o STR post-indicizzato in cui Rm è lo stesso registro di Rn.

4.9.6 Interruzione dei dati

Un trasferimento da o verso un indirizzo legale può causare problemi a un sistema di gestione della memoria. Ad esempio, in un sistema che utilizza la memoria virtuale, i dati richiesti possono essere assenti dalla memoria principale. Il gestore della memoria può segnalare il problema portando ad ALTO l'ingresso **ABORT** del processore, che fa scattare la trappola Data Abort. Spetta al software di sistema risolvere la causa del problema, quindi è possibile riavviare l'istruzione e continuare il programma originale.

4.9.7 Tempi di ciclo delle istruzioni

Le istruzioni LDR normali richiedono 1S + 1N + 1I e le istruzioni LDR PC richiedono 2S + 2N +1I cicli incrementali, dove S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

L'esecuzione delle istruzioni STR richiede 2N cicli incrementali.

4.9.8 Sintassi dell'assemblatore

```
<LDR|STR>{cond}{B}{T} Rd,<Indirizzo>
```

dove:

LDRcarica dalla memoria in un registro

STRstore da un registro in memoria

{cond}mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

{B}se B è presente allora trasferimento di byte, altrimenti trasferimento di parola

{T}se T è presente, il bit W verrà impostato in un'istruzione post-indicizzata, forzando la modalità non privilegiata per il ciclo di trasferimento. T non è ammesso quando è specificata o implicita una modalità di indirizzamento pre-indicizzata.

Rè un'espressione che valuta un numero di registro valido.

Rn e Rm sono espressioni che valutano un numero di registro. Se Rn è R15, l'assemblatore sottrarrà 8 dal valore di offset per consentire il pipelining. In questo caso, la scrittura di base non deve essere specificata.

<Indirizzo> può essere:

1 Un'espressione che genera un indirizzo:

```
<espressione>
```

L'assemblatore cercherà di generare un'istruzione utilizzando il PC come base e un offset immediato corretto per indirizzare la posizione data dalla valutazione dell'espressione. Si tratterà di un indirizzo relativo al PC, preindicizzato. Se l'indirizzo non rientra nell'intervallo, viene generato un errore.

2 Una specifica di indirizzamento preindicizzata:

`[Rn]`	offset di zero
`[Rn,<#espressione>]{!}`	offset di <espressione> byte
`[Rn,{+/-}Rm{,<shift>}]{!}`	offset del contenuto +/- dell'indice registro, traslato da <shift>.

3 Una specifica di indirizzamento post-indicizzata:

`[Rn],<#espressione>`	offset di <espressione> byte
`[Rn],{+/-}Rm{,<shift>}`	offset del contenuto +/- dell'indice registro, spostato come da <shift>.

<shift> operazione generale di spostamento (vedere le istruzioni per l'elaborazione dei dati), ma non è possibile specificare la quantità di spostamento tramite un registro.

{!}riscrive il registro di base (imposta il bit W) se! è presente.

4.9.9 Esempi

```
STR R1,[R2,R4]!         ; memorizzare R1 in R2+R4 (entrambi i quali
                                                          sono
                              registri) e riscrivere l'indirizzo a
                          ; R2.
 STR R1,[R2],R4          ; Memorizza R1 in R2 e scrive indietro
                           ; R2+R4 a R2.
```

```
LDR R1,[R2,#16]          ; caricare R1 dal contenuto di R2+16, ma
                         ; non rispondere.
LDRR1,[R2,R3,LSL#2]      ; Carica R1 dal contenuto di
R2+R3*4LDREQBR1,[R6,#5] ; Carica in modo condizionale il byte
in R6+5 in
                         ; R1 bit da 0 a 7, riempimento bit da 8 a 31
                         ; con zeri.
STR R1,PLACE             ; genera l'offset relativo del PC a
                         indirizzo LUOGO.
    -
LUOGO
```

4.10 Trasferimento di mezze parole e dati con segno (LDRH/STRH/LDRSB/LDRSH)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-16: Trasferimento di dati di mezza parola e firmati con offset di registro***, sotto, e ***Figura 4-17: Trasferimento di mezza parola e dati firmati con offset immediato*** a pagina 4-33.

Queste istruzioni vengono utilizzate per caricare o memorizzare mezze parole di dati e anche per caricare byte o mezze parole di dati con segno esteso. L'indirizzo di memoria utilizzato nel trasferimento viene calcolato aggiungendo o sottraendo un offset a un registro di base. Il risultato di questo calcolo può essere riscritto nel registro di base se è richiesta l'autoindicizzazione.

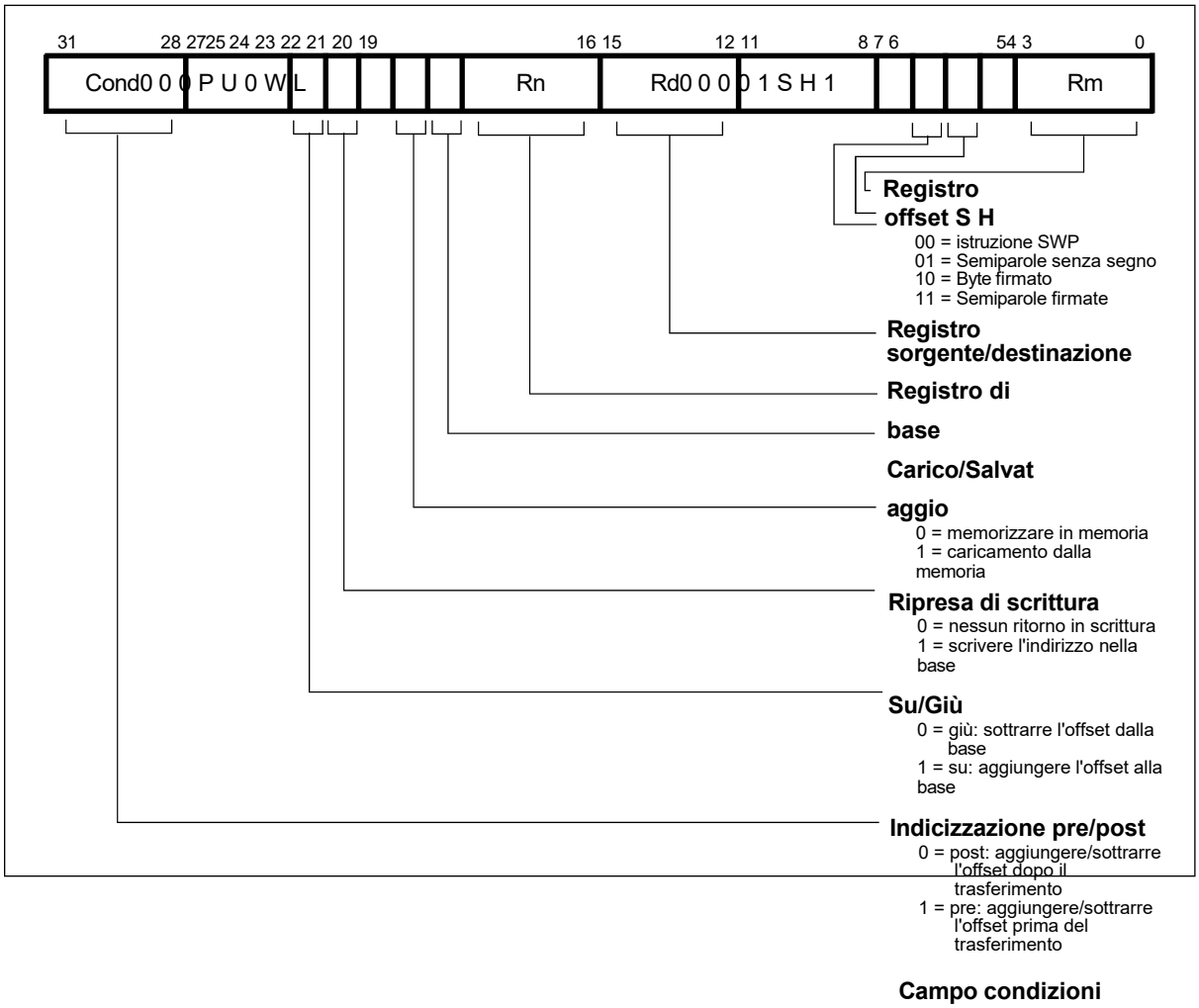


Figura 4-16: Trasferimento di mezza parola e dati firmati con offset di registro

Finale - Accesso

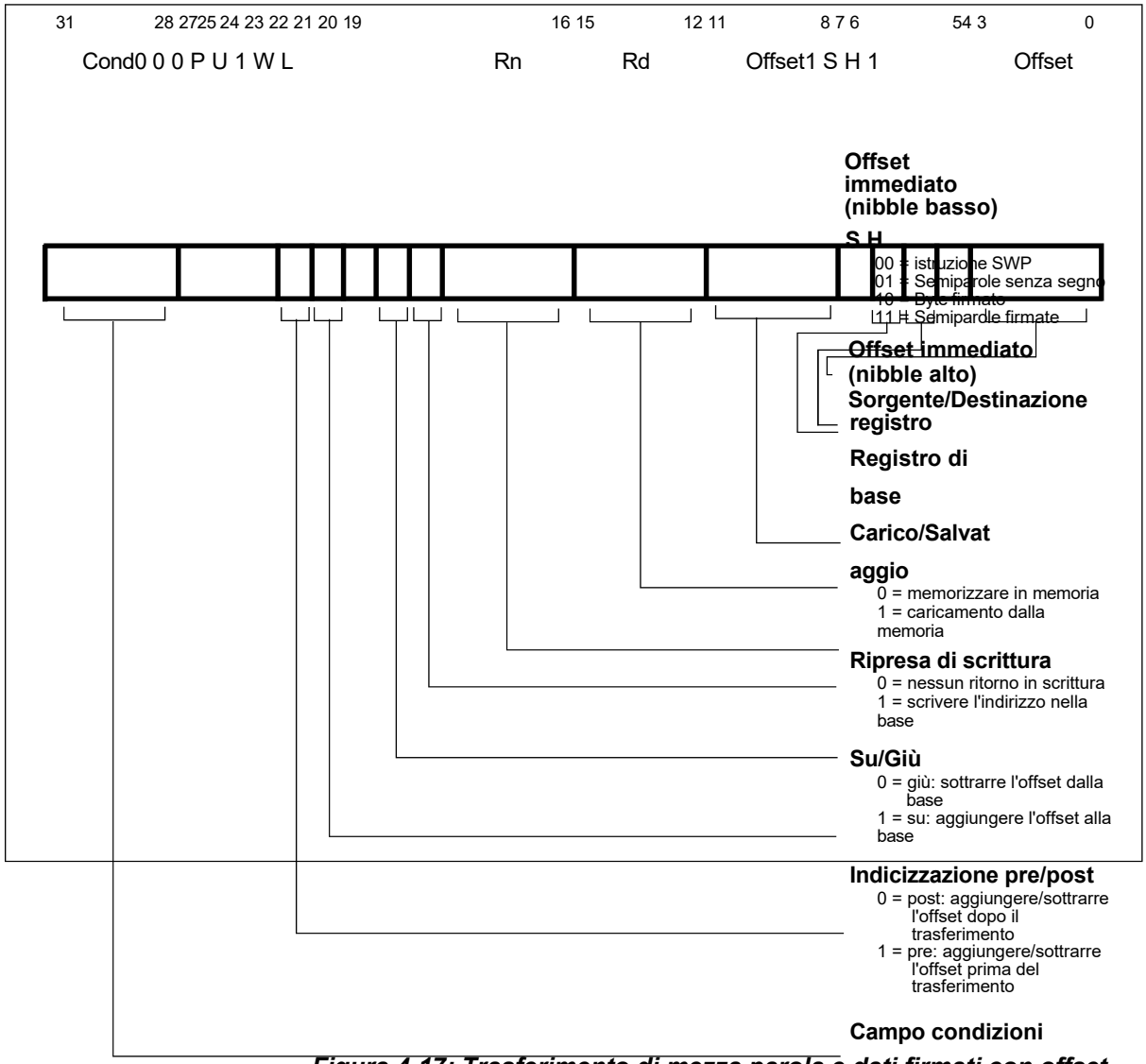


Figura 4-17: Trasferimento di mezza parola e dati firmati con offset immediato

4.10.1 Offset e indicizzazione automatica

L'offset dalla base può essere un valore immediato binario a 8 bit senza segno nell'istruzione o un secondo registro. L'offset a 8 bit si forma concatenando i bit da 11 a 8 e i bit da 3 a 0 della parola dell'istruzione, in modo che il bit 11 diventi il MSB e il bit 0 il LSB. L'offset può essere aggiunto (U=1) o sottratto (U=0) al registro di base Rn. La modifica dell'offset può essere eseguita prima (pre-indicizzato, P=1) o dopo (post-indicizzato, P=0) l'utilizzo del registro di base come indirizzo di trasferimento.

Il bit W consente di scegliere tra le modalità di indirizzamento a incremento e decremento automatico. Il valore di base modificato può essere riscritto nella base (W=1), oppure può essere mantenuta la vecchia base (W=0). Nel caso dell'indirizzamento post-indicizzato, il bit di write back è ridondante ed è sempre impostato a zero, poiché il vecchio valore di base può essere mantenuto se necessario impostando l'offset a zero. Pertanto, i trasferimenti di dati post-indicizzati riscrivono sempre la base modificata.

Il bit di Write-back non deve essere impostato alto (W=1) quando è selezionato l'indirizzamento post-indicizzato.

4.10.2 Caricamento e memorizzazione di mezze parole

L'impostazione di S=0 e H=1 può essere utilizzata per trasferire mezze parole senza segno tra un registro ARM7TDMI-S e la memoria.

Finale - Accesso libero

L'azione delle istruzioni LDRH e STRH è influenzata dal segnale di controllo BIGEND. Le due possibili configurazioni sono descritte nella sezione seguente.

4.10.3 Carichi di byte e di mezza parola con segno

Il bit S controlla il caricamento dei dati con estensione del segno. Quando S=1 il bit H seleziona tra Bytes (H=0) e Half-words (H=1). Il bit L non deve essere impostato basso (Store) quando sono state selezionate operazioni con segno (S=1).

L'istruzione LDRSB carica il Byte selezionato nei bit da 7 a 0 del registro di destinazione e i bit da 31 a 8 del registro di destinazione vengono impostati sul valore del bit 7, il bit di segno.

L'istruzione LDRSH carica la mezza parola selezionata nei bit da 15 a 0 del registro di destinazione e i bit da 31 a 16 del registro di destinazione vengono impostati sul valore del bit 15, il bit di segno.

L'azione delle istruzioni LDRSB e LDRSH è influenzata dal segnale di controllo BIGEND. Le due possibili configurazioni sono descritte nella sezione seguente.

4.10.4 Endianità e selezione di byte/mezzaparola

Configurazione little endian

Un carico di byte firmato (LDRSB) si aspetta dati sugli ingressi del bus dati da 7 a 0 se l'indirizzo fornito è su un confine di parola, sugli ingressi del bus dati da 15 a 8 se si tratta di un indirizzo di parola più un byte, e così via. Il byte selezionato viene inserito negli 8 bit inferiori del registro di destinazione e i bit rimanenti del registro vengono riempiti con il bit di segno, il bit 7 del byte. Si veda la ***Figura 3-2: Indirizzi little-endian di byte all'interno di parole a*** pagina 3-4.

Un caricamento di mezza parola (LDRSH o LDRH) prevede dati sugli ingressi del bus dati da 15 a 0 se l'indirizzo fornito si trova su un confine di parola e sugli ingressi del bus dati da 31 a 16 se si tratta di un confine di mezza parola (A[1]=1). Se il bit 0 dell'indirizzo fornito è ALTO, l'ARM7TDMI-S caricherà un valore imprevedibile. La mezza parola selezionata viene inserita nei 16 bit inferiori del registro di destinazione. Per le mezze parole senza segno (LDRH), i primi 16 bit del registro sono riempiti di zeri, mentre per le mezze parole con segno (LDRSH) i primi 16 bit sono riempiti con il bit di segno, il bit 15 della mezza parola.

Una memorizzazione di mezza parola (STRH) ripete due volte i 16 bit inferiori del registro sorgente attraverso le uscite del bus dati da 31 a 0. Il sistema di memoria esterno deve attivare il sottosistema di mezza parola appropriato per memorizzare i dati. Si noti che l'indirizzo deve essere allineato alla mezza parola; se il bit 0 dell'indirizzo è alto, il comportamento non è prevedibile.

Configurazione big endian

Un carico di byte firmato (LDRSB) si aspetta dati sugli ingressi del bus dati da 31 a 24 se l'indirizzo fornito è su un confine di parola, sugli ingressi del bus dati da 23 a 16 se si tratta di un indirizzo di parola più un byte, e così via. Il byte selezionato viene inserito negli 8 bit inferiori del registro di destinazione e i bit rimanenti del registro vengono riempiti con il bit di segno, il bit 7 del byte. Si veda la ***Figura 3-1: Indirizzi big-endian di byte all'interno di parole*** a pagina 3-4.

Un caricamento di mezza parola (LDRSH o LDRH) prevede dati sugli ingressi del bus dati da 31 a 16 se l'indirizzo fornito è su un confine di parola e sugli ingressi del bus dati da 15 a 0 se è un confine di mezza parola (A[1]=1). L'indirizzo fornito deve essere sempre su un confine di mezza parola. Se il bit 0 dell'indirizzo fornito è ALTO, l'ARM7TDMI-S caricherà un valore imprevedibile. La mezza parola selezionata viene inserita nei 16 bit inferiori del registro di destinazione. Per le mezze parole senza segno (LDRH), i primi 16 bit del registro sono riempiti di zeri, mentre per le mezze parole con segno (LDRSH) i primi 16 bit sono riempiti con il bit di segno, il bit 15 della mezza parola.

Finale - Accesso

Una memorizzazione di mezza parola (STRH) ripete due volte i 16 bit inferiori del registro sorgente attraverso le uscite del bus dati da 31 a 0. Il sistema di memoria esterno deve attivare il sottosistema di mezza parola appropriato per memorizzare i dati. Si noti che l'indirizzo deve essere allineato alla mezza parola; se il bit 0 dell'indirizzo è alto, il comportamento non è prevedibile.

4.10.5 Uso di R15

Il Write-back non deve essere specificato se R15 è indicato come registro di base (Rn). Quando si utilizza R15 come registro di base, occorre ricordare che esso contiene un indirizzo a 8 byte di distanza dall'indirizzo dell'istruzione corrente.

R15 non deve essere specificato come offset del registro (Rm).

Quando R15 è il registro sorgente (Rd) di un'istruzione di memorizzazione di mezza parola (STRH), l'indirizzo memorizzato sarà l'indirizzo dell'istruzione più 12.

4.10.6 Interruzione dei dati

Un trasferimento da o verso un indirizzo legale può causare problemi a un sistema di gestione della memoria. Ad esempio, in un sistema che utilizza la memoria virtuale, i dati richiesti possono essere assenti dalla memoria principale. Il gestore della memoria può segnalare il problema portando ad ALTO l'ingresso ABORT del processore, che attiva la trappola Data Abort. Spetta al software di sistema risolvere la causa del problema, quindi è possibile riavviare l'istruzione e continuare il programma originale.

4.10.7 Tempi di ciclo delle istruzioni

Le istruzioni LDR(H,SH,SB) normali richiedono 1S + 1N + 1I Le istruzioni LDR(H,SH,SB) PC richiedono 2S + 2N + 1I cicli incrementali. S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3. *L'*esecuzione delle istruzioni STRH richiede 2N cicli incrementali.

4.10.8 Sintassi dell'assemblatore

```
<LDR|STR>{cond}<H|SH|SB> Rd,<indirizzo>
```

LDRcaricare dalla memoria in un registro

STRSalvare da un registro in memoria

{cond}mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

H Trasferimento della quantità di mezza parola

SBLoad sign extended byte (valido solo per LDR)

SHCarica la mezza parola estesa del segno (valida solo per LDR)

Rè un'espressione che valuta un numero di registro valido.

<indirizzo> può essere:

1 Un'espressione che genera un indirizzo:

```
<espressione>
```

L'assemblatore cercherà di generare un'istruzione utilizzando il PC come base e un offset immediato corretto per indirizzare la posizione data dalla valutazione dell'espressione. Si tratterà di un indirizzo relativo al PC, preindicizzato. Se l'indirizzo non rientra nell'intervallo, viene generato un errore.

2 Una specifica di indirizzamento preindicizzata:

[Rn] offset di zero

[Rn,<#espressione>]{!}offset di <espressione> byte

[Rn,{+/-}Rm]{!}offset del contenuto +/- del registro indice

3 Una specifica di indirizzamento post-indicizzata:

[Rn],<#espressione> offset di <espressione> byte

[Rn],{+/-}Rmoffset del contenuto +/- del registro indice.

Rn e Rm sono espressioni che valutano un numero di registro. Se Rn è R15, l'assemblatore sottrarrà 8 dal valore di offset per consentire il pipelining. In questo caso, la scrittura di base non deve essere specificata.

{!} riscrive il registro di base (imposta il bit W) se ! è presente.

4.10.9 Esempi

```
LDRHR1,[R2,-R3]!; Caricare R1 dal contenuto di
          ; indirizzo di mezza parola contenuto in
          R2-R3 (entrambi sono registri)
          e riscrivere l'indirizzo a R2
STRHR3,[R4,#14]; memorizzare la mezza parola in R3
a R14+14
          ma non rispondere.
LDRSBR8,[R2],#-223; caricare R8 con il segno esteso
        ; contenuto dell'indirizzo del byte
          contenuti in R2 e riscrivere
          R2-223 a R2.
LDRNESHR11,[R0]; caricare condizionatamente R11 con il segno
          ; contenuto esteso della mezza parola
          ; indirizzo contenuto in R0.
QUI       ; Generare l'offset relativo del PC a
          indirizzo FRED.
          Memorizzare la mezza parola in R5 all'indirizzo
          FRED.
STRHR5, [PC, #(FRED-HERE-8)]
    .
FRED
```

4.11 Trasferimento dati a blocchi (LDM, STM)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-18: Istruzioni di trasferimento dati a blocchi***.

Le istruzioni di trasferimento dati a blocchi vengono utilizzate per caricare (LDM) o memorizzare (STM) qualsiasi sottoinsieme dei registri attualmente visibili. Supportano tutte le possibili modalità di impilamento, mantenendo pile piene o vuote che possono crescere verso l'alto o verso il basso della memoria, e sono istruzioni molto efficienti per salvare o ripristinare il contesto, o per spostare grandi blocchi di dati nella memoria principale.

4.11.1 L'elenco dei registri

L'istruzione può provocare il trasferimento di qualsiasi registro del banco corrente (e I programmi in modalità non utente possono anche trasferirsi da e verso il banco utente, vedi sotto). L'elenco dei registri è un campo di 16 bit nell'istruzione, con ogni bit corrispondente a un registro. Un 1 nel bit 0 del campo dei registri provoca il trasferimento di R0, mentre uno 0 non lo fa; analogamente, il bit 1 controlla il trasferimento di R1, e così via.

È possibile specificare qualsiasi sottoinsieme dei registri o tutti i registri. L'unica restrizione è che l'elenco dei registri non deve essere vuoto.

Ogni volta che R15 viene memorizzato nella memoria, il valore memorizzato è l'indirizzo dell'istruzione STM più 12.

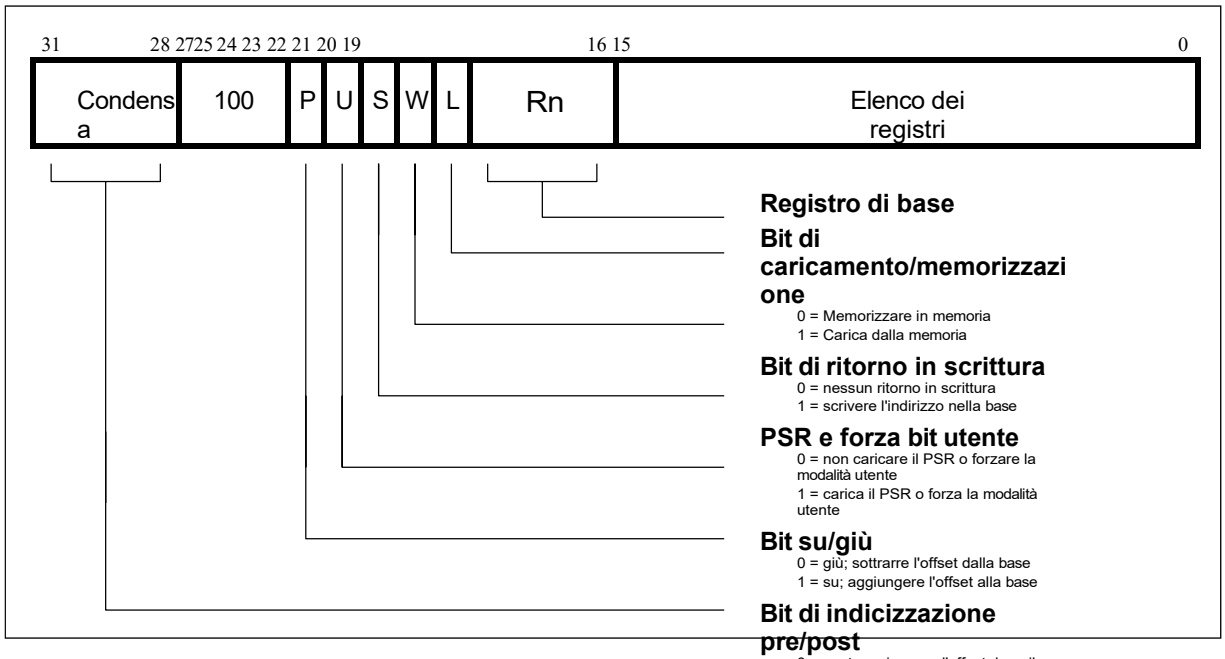


Figura 4-18: Istruzioni di trasferimento dati a blocchi

4.11.2 Modalità di indirizzamento

Gli indirizzi di trasferimento sono determinati dal contenuto del registro di base (Rn), dal bit pre/post (P) e dal bit up/down (U). I registri vengono trasferiti nell'ordine dal più basso al più alto, quindi R15 (se presente nell'elenco) sarà sempre trasferito per ultimo. Anche il registro più basso viene trasferito all'indirizzo di memoria più basso. A titolo di esempio, si consideri il trasferimento di R1, R5 e R7 nel caso in cui Rn=0x1000 e sia richiesto il ritorno in scrittura della base modificata (W=1). ***Figura 4-***

indirizzamento decrescente a pagina 4-39 e la ***Figura 4-22: Indirizzamento pre-decremento a*** pagina 4-40 mostrano la sequenza di trasferimento dei registri, gli indirizzi utilizzati e il valore di Rn al termine dell'istruzione.

In tutti i casi, se non fosse stato richiesto il ripristino della base modificata (W=0), Rn avrebbe mantenuto il suo valore iniziale di 0x1000, a meno che non si trovasse anche nell'elenco di trasferimento di un'istruzione di caricamento di registri multipli, quando sarebbe stato sovrascritto con il valore caricato.

4.11.3 Allineamento dell'indirizzo

L'indirizzo deve essere normalmente una quantità allineata alla parola e gli indirizzi non allineati alla parola non influenzano l'istruzione. Tuttavia, i 2 bit inferiori dell'indirizzo appariranno su **A[1:0]** e potrebbero essere interpretati dal sistema di memoria.

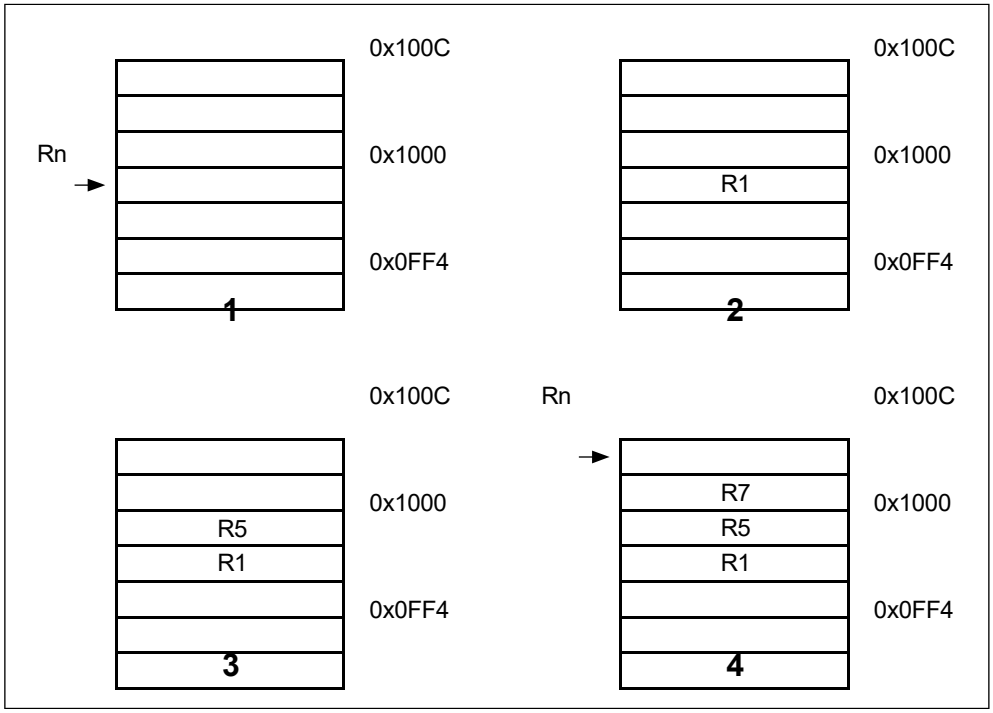


Figura 4-19: Indirizzamento post-incremento

Finale - Accesso

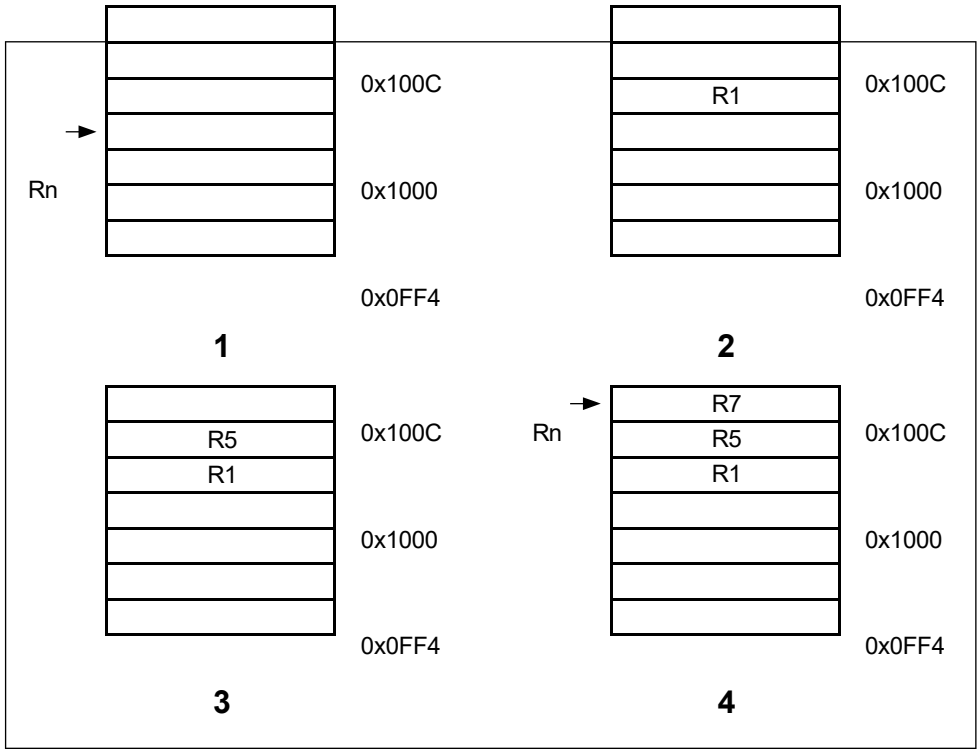


Figura 4-20: Indirizzamento pre-incremento

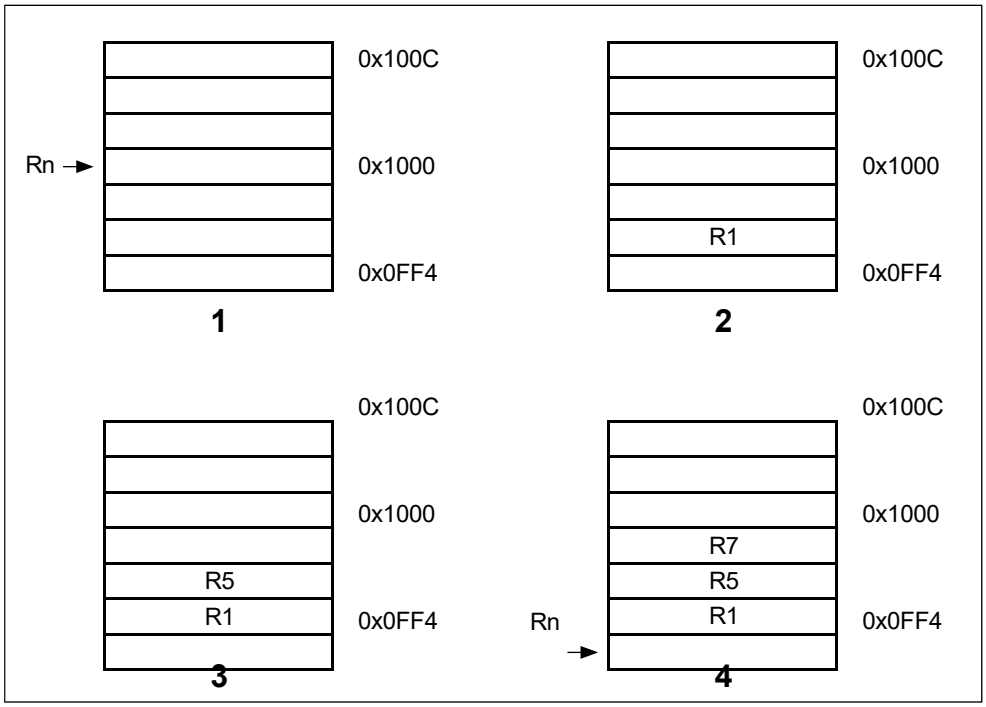


Figura 4-21: Indirizzamento post-decremento

Finale - Accesso libero

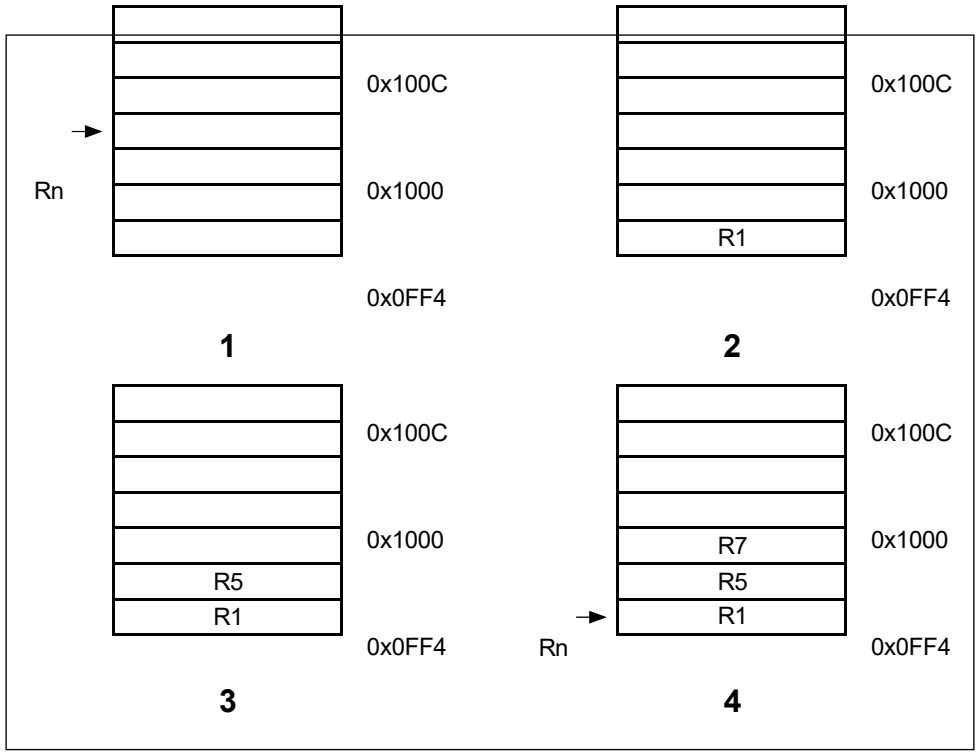


Figura 4-22: Indirizzamento pre-decremento

4.11.4 Uso del bit S

Quando il bit S è impostato in un'istruzione LDM/STM, il suo significato dipende dalla presenza o meno di R15 nell'elenco di trasferimento e dal tipo di istruzione. Il bit S deve essere impostato solo se l'istruzione deve essere eseguita in modalità privilegiata.

LDM con R15 nell'elenco di trasferimento e bit S impostato (modifica della modalità)

Se l'istruzione è un LDM, SPSR_<mode> viene trasferito a CPSR nello stesso momento in cui viene caricato R15.

STM con R15 nella lista di trasferimento e bit S impostato (Trasferimento bancario utente)

I registri trasferiti vengono prelevati dal banco Utente anziché dal banco corrispondente alla modalità corrente. Ciò è utile per salvare lo stato dell'utente in caso di commutazione di processo. Quando si utilizza questo meccanismo, non si deve usare il write-back della base.

R15 non in elenco e bit S impostato (Trasferimento bancario utente)

Per entrambe le istruzioni LDM e STM, vengono trasferiti i registri del banco utente anziché il banco di registri corrispondente alla modalità corrente. Ciò è utile per salvare lo stato dell'utente durante le commutazioni di processo. Il ritorno alla scrittura della base non deve essere utilizzato quando si utilizza questo meccanismo.

Quando l'istruzione è LDM, è necessario prestare attenzione a non leggere da un registro in banco durante il ciclo successivo (l'inserimento di un'istruzione fittizia come MOV R0, R0 dopo l'LDM garantirà la sicurezza).

4.11.5 Utilizzo di R15 come base

R15 non deve essere utilizzato come registro di base in nessuna istruzione LDM o STM.

Finale - Accesso

4.11.6 Inclusione della base nell'elenco dei registri

Quando viene specificato write-back, la base viene riscritta alla fine del secondo ciclo dell'istruzione. Durante un STM, il primo registro viene scritto all'inizio del secondo ciclo. Un STM che prevede la memorizzazione della base, con la base come primo registro da memorizzare, memorizzerà quindi il valore invariato, mentre con la base seconda o successiva nell'ordine di trasferimento, memorizzerà il valore modificato. Un LDM sovrascrive sempre la base aggiornata se questa è presente nell'elenco.

4.11.7 Interruzione dei dati

Alcuni indirizzi legali possono essere inaccettabili per un sistema di gestione della memoria e il gestore della memoria può indicare un problema con un indirizzo portando il segnale **ABORT in** alto. Questo può accadere in qualsiasi trasferimento durante un caricamento o una memorizzazione a registri multipli e deve essere recuperabile se l'ARM7TDMI-S deve essere utilizzato in un sistema di memoria virtuale.

Interruzione durante le istruzioni STM

Se l'interruzione si verifica durante un'istruzione di memorizzazione multipla, ARM7TDMI-S non interviene fino al completamento dell'istruzione, dopodiché entra nella trappola di interruzione dei dati. Il gestore della memoria è responsabile della prevenzione di scritture errate nella memoria. L'unica modifica allo stato interno del processore sarà la modifica del registro di base se è stato specificato il write-back, che deve essere invertita dal software (e la causa dell'interruzione deve essere risolta) prima che l'istruzione possa essere ritentata.

Interruzione durante le istruzioni LDM

Quando ARM7TDMI-S rileva un'interruzione dei dati durante un'istruzione di caricamento multiplo, modifica il funzionamento dell'istruzione per garantire la possibilità di recupero.

1. La sovrascrittura dei registri si interrompe quando si verifica l'interruzione. Il carico che si interrompe non viene eseguito, ma quelli precedenti possono aver sovrascritto i registri. Il PC è sempre l'ultimo registro ad essere scritto e quindi sarà sempre conservato.
2. Il registro di base viene ripristinato al suo valore modificato, se è stato richiesto il ripristino della scrittura. Ciò garantisce la recuperabilità nel caso in cui il registro di base sia anche nell'elenco di trasferimento e possa essere stato sovrascritto prima dell'interruzione.

La trappola di interruzione dei dati scatta al termine del caricamento multiplo e il software di sistema deve annullare qualsiasi modifica della base (e risolvere la causa dell'interruzione) prima di riavviare l'istruzione.

4.11.8 Tempi di ciclo delle istruzioni

Le istruzioni LDM normali richiedono nS + 1N + 1I e le LDM PC richiedono (n+1)S + 2N + 1I cicli incrementali, dove S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3. Le istruzioni STM richiedono (n-1)S + 2N cicli incrementali per essere eseguite, dove *n* è il numero di parole trasferite.

4.11.9 Sintassi dell'assemblatore

`<LDM|STM>{cond}<FD|ED|FA|EA|IA|IB|DA|DB> Rn{!},<Rlist>{^}` dove:

{cond}Mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

Rnis è un'espressione che valuta un numero di registro valido

<Rlist> è un elenco di registri e intervalli di registri racchiusi da {} (ad esempio, {R0,R2-R7,R10}).

{!}se presente richiede il write-back (W=1), altrimenti W=0

{^} se presente, impostare il bit S per caricare il CPSR insieme al PC,

oppure forzare il trasferimento del banco utente quando si è in modalità privilegiata

Nomi delle modalità di indirizzamento

Esistono diverse mnemotecniche di assemblaggio per ciascuna modalità di indirizzamento, a seconda che l'istruzione venga utilizzata per supportare gli stack o per altri scopi. L'equivalenza tra i nomi e i valori dei bit dell'istruzione è riportata nella ***Tabella 4-6: Nomi delle modalità di indirizzamento***:

Nome	Pila	Altro	L bit	P bit	U bit
carico pre-incremento	LDMED	LDMIB	1	1	1
carico post-incremento	LDMFD	LDMIA	1	0	1
carico di pre-decremento	LDMEA	LDMDB	1	1	0
carico post-decremento	LDMFA	LDMDA	1	0	0
negozio di pre-incremento	STMFA	STMIB	0	1	1
negozio post-incremento	STMEA	STMIA	0	0	1
negozio di pre-decremento	STMFD	STMDB	0	1	0
negozio post-decremento	STMED	STMDA	0	0	0

Tabella 4-6: Nomi delle modalità di indirizzamento

FD, ED, FA, EA definiscono la pre- o la post-indicizzazione e il bit di salita/discesa in riferimento alla forma di pila richiesta. F ed E si riferiscono a una pila "piena" o "vuota", cioè se è necessario eseguire una preindicizzazione (piena) prima di memorizzare la pila. A e D indicano se la pila è ascendente o discendente. Se ascendente, un STM salirà e un LDM scenderà; se discendente, viceversa.

IA, IB, DA, DB consentono il controllo quando LDM/STM non vengono utilizzati per le pile e significano semplicemente Incremento dopo, Incremento prima, Decremento dopo, Decremento prima.

4.11.10 Esempi

```
LDMFD SP!,{R0,R1,R2}    ;Disimpilare i 3
registri .STMIA R0,{R0-R15} ; Salva tutti i
registri.
LDMFD SP!,{R15}         ; R15 <- (SP), CPSR invariato.
LDMFD SP!,{R15}^        ;R15 <- (SP), CPSR <- SPSR_mode
                        (consentito solo in modalità privilegiata).
STMFD R13,{R0-R14}^     ; Salva le regs della modalità utente sullo
stack
                        (consentito solo in modalità privilegiata).
```

Queste istruzioni possono essere utilizzate per salvare lo stato all'ingresso della subroutine e ripristinarlo in modo efficiente al ritorno alla routine chiamante:

```
STMED SP!,{R0-R3,R14}   ; salva da R0 a R3 per utilizzarlo come spazio
di lavoro
                        e R14 per il ritorno.
BL    da qualche parte  ; Questa chiamata annidata
sovrascriverà R14 LDMED SP!,{R0-R3,R15}     ; ripristina
l'area di lavoro e ritorna.
```

4.12 Scambio di dati singolo (SWP)

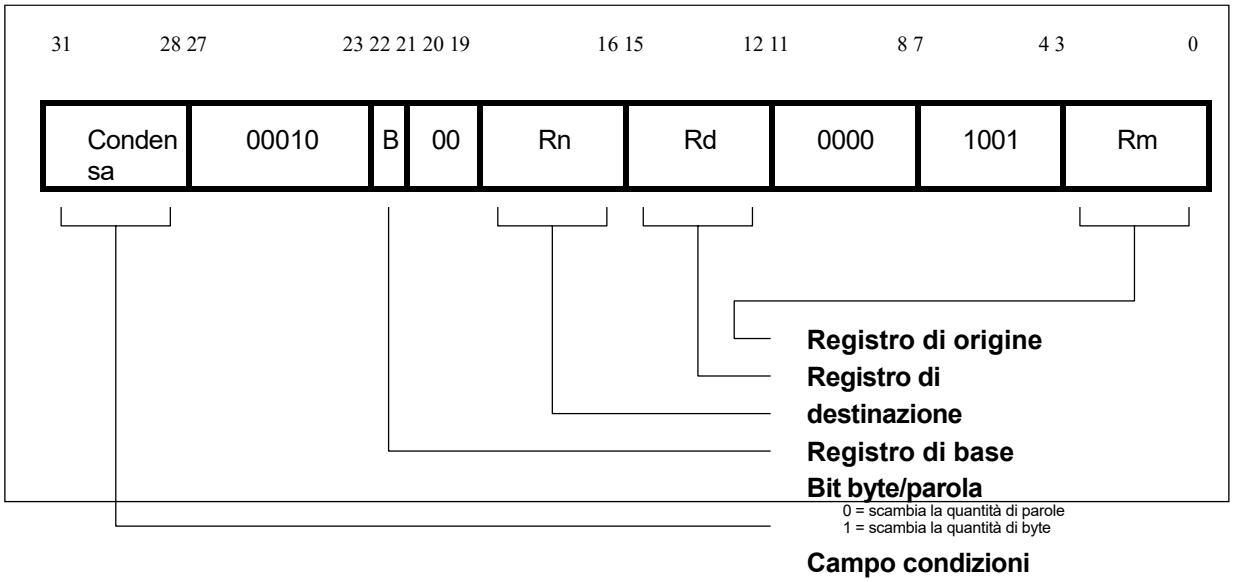


Figura 4-23: Istruzione di swap

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica dell'istruzione è illustrata nella ***Figura 4-23: Istruzione di scambio***.

L'istruzione di scambio di dati viene utilizzata per scambiare una quantità di byte o di parole tra un registro e la memoria esterna. Questa istruzione è implementata come una lettura di memoria seguita da una scrittura di memoria che sono "bloccate" insieme (il processore non può essere interrotto fino a quando entrambe le operazioni non sono state completate e il gestore della memoria è avvertito di trattarle come inseparabili). Questa classe di istruzioni è particolarmente utile per implementare i semafori software.

L'indirizzo di scambio è determinato dal contenuto del registro di base (Rn). Il processore legge prima il contenuto dell'indirizzo di swap. Quindi scrive il contenuto del registro di origine (Rm) all'indirizzo di swap e memorizza il vecchio contenuto della memoria nel registro di destinazione (Rd). Lo stesso registro può essere specificato sia come origine che come destinazione.

L'uscita **LOCK** rimane ALTA per tutta la durata delle operazioni di lettura e scrittura per segnalare al gestore della memoria esterna che sono bloccate insieme e che devono essere completate senza interruzioni. Questo è importante nei sistemi multiprocessore dove l'istruzione swap è l'unica istruzione indivisibile che può essere usata per implementare i semafori; il controllo della memoria non deve essere rimosso da un processore mentre sta eseguendo un'operazione bloccata.

4.12.1 Byte e parole

Questa classe di istruzioni può essere utilizzata per scambiare un byte (B=1) o una parola (B=0) tra un registro ARM7TDMI-S e la memoria. L'istruzione SWP è implementata come un LDR seguito da uno STR e la sua azione è descritta nella sezione sui trasferimenti di dati singoli. In particolare, la descrizione della configurazione Big e Little Endian si applica all'istruzione SWP.

4.12.2 Utilizzo di R15

Non utilizzare R15 come operando (Rd, Rn o Rs) in un'istruzione SWP.

4.12.3 Interruzione dei dati

Se l'indirizzo utilizzato per lo swap è inaccettabile per un sistema di gestione della memoria, il gestore della memoria può segnalare il problema pilotando ABORT HIGH. Questo può accadere sia nel ciclo di lettura che in quello di scrittura (o in entrambi), e in entrambi i casi, verrà attivata la trappola Data Abort. Spetta al software di sistema risolvere la causa del problema, quindi è possibile riavviare l'istruzione e continuare il programma originale.

4.12.4 Tempi di ciclo delle istruzioni

L'esecuzione delle istruzioni di swap richiede 1S + 2N +1I cicli incrementali, dove S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.12.5 Sintassi dell'assemblatore

```
<SWP>{cond}{B} Rd,Rm,[Rn]
```

{cond}Mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

{B} se B è presente allora trasferimento di byte, altrimenti trasferimento di parola Rd,Rm,Rnare espressioni che valutano numeri di registro validi

4.12.6 Esempi

```
SWP    R0,R1,[R2]      ; caricare R0 con la parola indirizzata da R2 e
                       ; memorizzare R1 in R2.
SWPB   R2,R3,[R4]      ; caricare R2 con il byte indirizzato da R4, e
                       ; memorizzare i bit da 0 a 7 di R3 in R4.
SWPEQ R0,R0,[R1]       ; scambia in modo condizionato il contenuto della
                           cartella
                       ; parola indirizzata da R1 con R0.
```

4.13 Interruzione software (SWI)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica dell'istruzione è illustrata nella ***Figura 4-24: Istruzione di interruzione del software***, di seguito riportata.

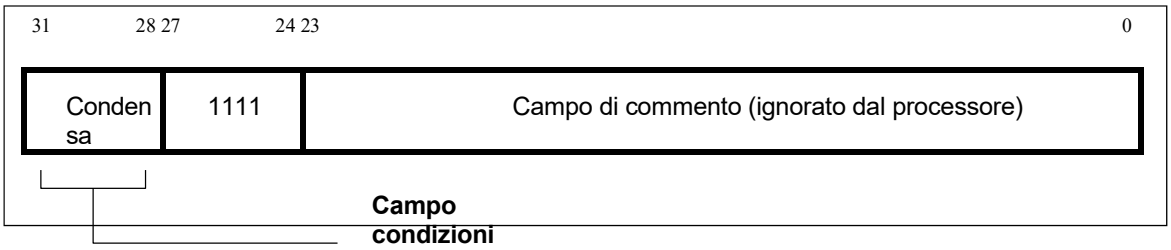


Figura 4-24: Istruzione di interruzione del software

L'istruzione di interrupt software viene utilizzata per entrare in modalità Supervisore in modo controllato. L'istruzione fa scattare la trappola dell'interrupt software, che determina il cambio di modalità. Il PC viene quindi forzato a un valore fisso (0x08) e il CPSR viene salvato in SPSR_svc. Se l'indirizzo del vettore SWI è adeguatamente protetto (da hardware di gestione della memoria esterna) dalla modifica da parte dell'utente, è possibile costruire un sistema operativo completamente protetto.

4.13.1 Ritorno dal supervisore

Il PC viene salvato in R14_svc quando si entra nella trappola di interrupt software, con il PC regolato in modo da puntare alla parola dopo l'istruzione SWI. MOVS PC,R14_svc tornerà al programma chiamante e ripristinerà il CPSR.

Si noti che il meccanismo di collegamento non è rientrante, quindi se il codice supervisore desidera utilizzare gli interrupt software al suo interno deve prima salvare una copia dell'indirizzo di ritorno e dell'SPSR.

4.13.2 Campo di commento

I 24 bit inferiori dell'istruzione sono ignorati dal processore e possono essere utilizzati per comunicare informazioni al codice di supervisione. Ad esempio, il supervisore può osservare questo campo e usarlo per indicizzare un array di punti di ingresso per le routine che eseguono le varie funzioni del supervisore.

4.13.3 Tempi di ciclo delle istruzioni

Le istruzioni di interruzione del software richiedono 2S + 1N cicli incrementali per essere eseguite, dove S e N sono definiti in ***6.2 Tipi di cicli*** a pagina 6-3.

4.13.4 Sintassi dell'assemblatore

```
SWI{cond} <espressione>
```

{cond}mnemonico di condizione a due caratteri, ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

<espressione> viene valutata e inserita nel campo dei commenti (che viene ignorato da ARM7TDMI-S).

4.13.5 Esempi

```
SWI   ReadC             ; Ottiene il carattere successivo dal
flusso di lettura. SWI  WriteI+"k"   ; Emette una "k" nel
flusso di scrittura.
SWINE 0                 ; Chiamata condizionata del supervisore
```

```
;                       nel campo dei commenti.

c
o
n

0
```

Codice del supervisore

Gli esempi precedenti presuppongono, ad esempio, l'esistenza di un codice supervisore adeguato:

```
0x08 B Supervisore      ; punto  di ingresso SWI
EntryTable              ; indirizzi delle routine di
        supervisione DCD ZeroRtn
        DCD ReadCRtn
        DCD WriteIRtn
        . . .
Zero    EQU  0
LeggiC  EQU  256
Scrive  EQU  512
reI


Supervisore

L'SWI ha la routine richiesta nei bit 8-23 e i dati (se presenti) in
; bit 0-7.
Si suppone che R13_svc punti a uno stack appropriato

STMFD R13,{R0-R2,R14}      ; salvare i registri di lavoro e tornare indietro
                           indirizzo.
LDR    R0,[R14,#-4]        ; Ricevere
l'istruzione SWI. BIC      R0,R0,#0xFF000000  ;
Cancella gli 8 bit superiori.
MOV    R1,R0,LSR#8         ; Ottenere l'offset della routine.
ADR    R2,EntryTable       ; Ottenere l'indirizzo iniziale
della tabella di ingresso. LDRR15     ,[R2,R1,LSL#2] ;
Eseguire il branch alla routine appropriata.

     WriteIRtn           ; Inserire il carattere nei bit 0-7 di R0.
          . .   .  .  .  .
LDMFD R13,{R0-R2,R15}^     ; Ripristina l'area di lavoro e ritorna,
                      ; ripristino della modalità e dei flag del processore.
```

4.14 Operazioni con i dati del coprocessore (CDP)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-25: Istruzione di operazione sui dati del coprocessore***.

Questa classe di istruzioni viene utilizzata per indicare a un coprocessore di eseguire un'operazione interna. L'ARM7TDMI-S non riceve alcun risultato e non attende il completamento dell'operazione. Il coprocessore può contenere una coda di istruzioni di questo tipo in attesa di essere eseguite e la loro esecuzione può sovrapporsi ad altre attività, consentendo al coprocessore e all'ARM7TDMI-S di eseguire compiti indipendenti in parallelo.

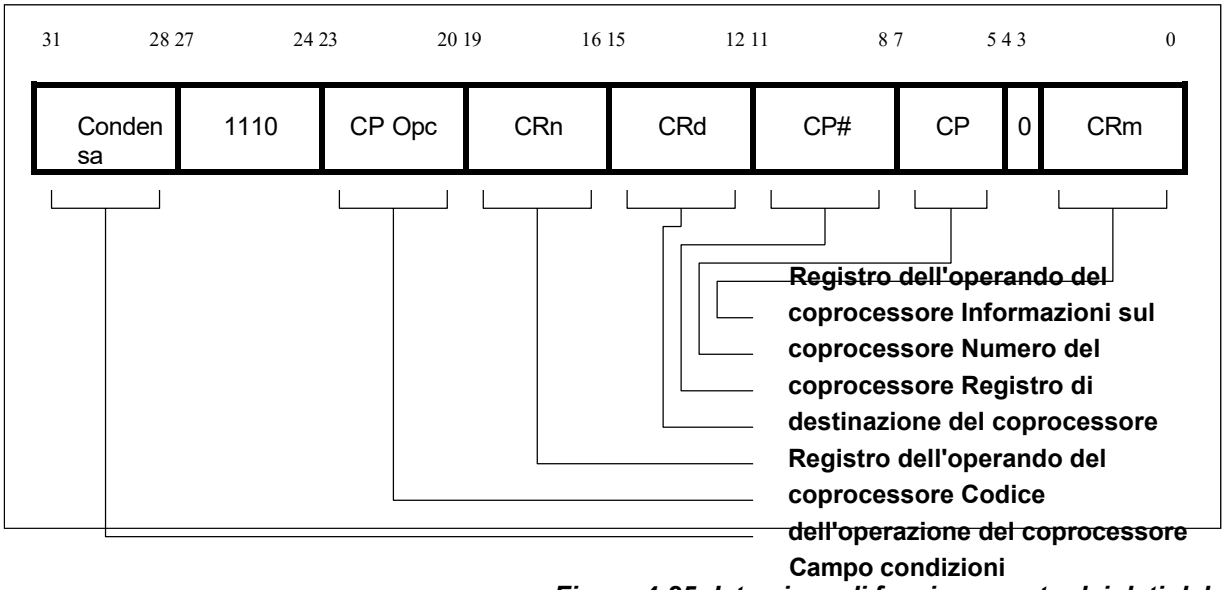


Figura 4-25: Istruzione di funzionamento dei dati del coprocessore

4.14.1 I campi del coprocessore

Solo il bit 4 e i bit da 24 a 31 sono significativi per ARM7TDMI-S. I bit rimanenti sono utilizzati dai coprocessori. I nomi dei campi sopra indicati sono usati per convenzione e i coprocessori possono ridefinire l'uso di tutti i campi tranne CP#, a seconda dei casi. Il campo CP# è utilizzato per contenere un numero identificativo (compreso tra 0 e 15) per ciascun coprocessore; un coprocessore ignorerà qualsiasi istruzione che non contenga il suo numero nel campo CP#.

L'interpretazione convenzionale dell'istruzione prevede che il coprocessore esegua un'operazione specificata nel campo CP Opc (ed eventualmente nel campo CP) sul contenuto di CRn e CRm e collochi il risultato in CRd.

4.14.2 Tempi di ciclo delle istruzioni

Le operazioni sui dati del coprocessore richiedono 1S + bI cicli incrementali per essere eseguite, dove *b* è il numero di cicli spesi nel ciclo busy-wait del coprocessore.

S e I sono definite in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.14.3 Sintassi dell'assemblatore

```
CDP{cond} p#,<espressione1>,cd,cn,cm{,<espressione2>}
```

{cond}mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

p#il numero univoco del coprocessore richiesto

<espressione1>	valutata come costante e inserita nel campo CP Opc
cd, cn e cm	valutanorispettivamente i numeri di registro validi del coprocessore CRd, CRn e CRm
<espressione2	dove present è valutato come una costante e inserito nel campo CP

4.14.4 Esempi

```
  CDP   p1,10,c1,c2,c3     ; Richiesta alla coproc 1 di eseguire
  l'operazione 10
                           su CR2 e CR3, e mettere il risultato
                           in CR1.
CDPEQ p2,5,c1,c2,c3,2    ; se il flag Z è impostato richiedere
                                                  la coproc 2
                    ; effettuare l'operazione 5 (tipo 2) su CR2
                           e CR3, e inserire il risultato in CR1.
```

4.15 Trasferimenti di dati al coprocessore (LDC, STC)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è illustrata nella ***Figura 4-26: Istruzioni di trasferimento dati del coprocessore***.

Questa classe di istruzioni viene utilizzata per caricare (LDC) o memorizzare (STC) un sottoinsieme di registri di un coprocessore direttamente in memoria. L'ARM7TDMI-S è responsabile della fornitura dell'indirizzo di memoria, mentre il coprocessore fornisce o accetta i dati e controlla il numero di parole trasferite.

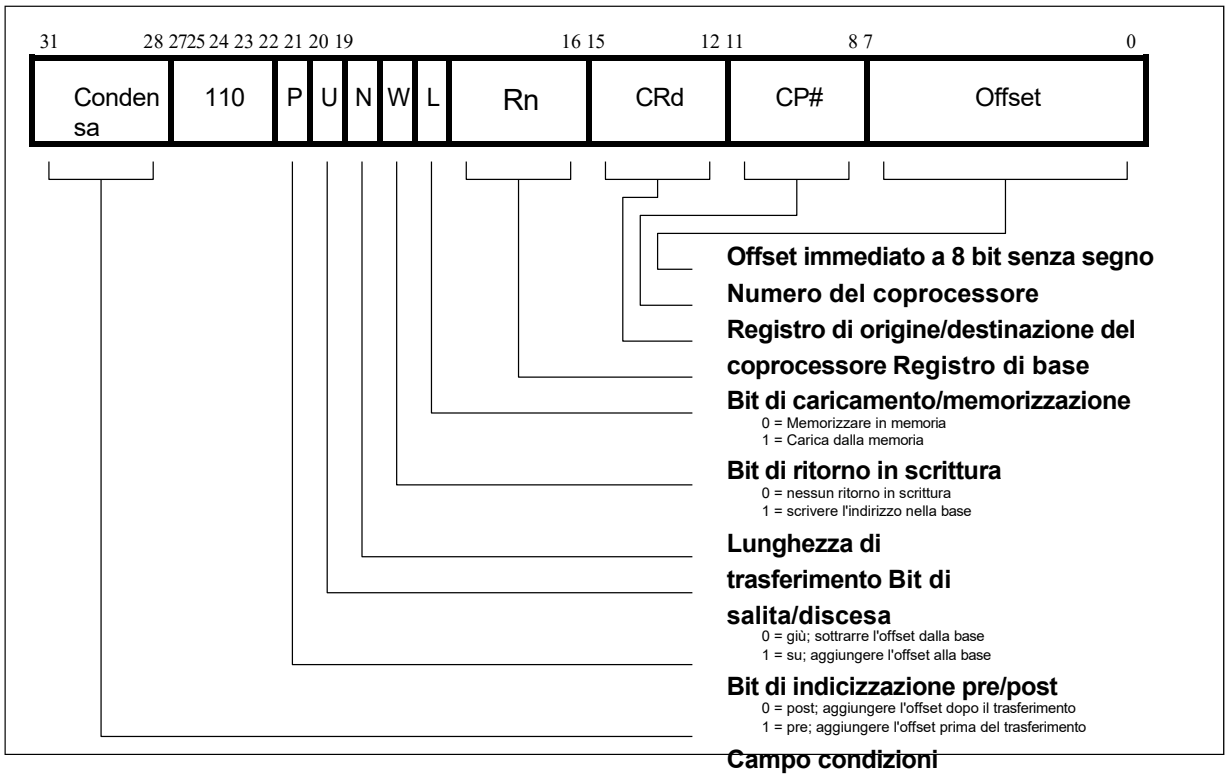


Figura 4-26: Istruzioni di trasferimento dati del coprocessore

4.15.1 I campi del coprocessore

Il campo CP# viene utilizzato per identificare il coprocessore che deve fornire o accettare i dati; un coprocessore risponde solo se il suo numero corrisponde al contenuto di questo campo.

Il campo CRd e il bit N contengono informazioni per il coprocessore che possono essere interpretate in modi diversi da coprocessori diversi, ma per convenzione CRd è il registro da trasferire (o il primo registro se ne deve essere trasferito più di uno) e il bit N viene utilizzato per scegliere una delle due opzioni di lunghezza del trasferimento. Ad esempio, N=0 può selezionare il trasferimento di un singolo registro, mentre N=1 può selezionare il trasferimento di tutti i registri per la commutazione di contesto.

4.15.2 Modalità di indirizzamento

L'ARM7TDMI-S è responsabile di fornire l'indirizzo utilizzato dal sistema di memoria per il trasferimento e le modalità di indirizzamento disponibili sono un sottoinsieme di quelle utilizzate nelle istruzioni di trasferimento di dati singoli. Si noti, tuttavia, che gli offset immediati sono di 8 bit e specificano gli offset di parola per i trasferimenti di dati del coprocessore, mentre sono di 12 bit e specificano gli offset di byte per i

trasferimenti di dati singoli.

L'offset immediato a 8 bit senza segno viene spostato a sinistra di 2 bit e aggiunto (U=1) o sottratto (U=0) al registro di base (Rn); questo calcolo può essere eseguito prima (P=1) o dopo (P=0) che la base sia usata come indirizzo di trasferimento. Il valore modificato della base può essere sovrascritto nel registro di base (se W=1), oppure il vecchio valore della base può essere conservato (W=0). Si noti che le modalità di indirizzamento post-indicizzate richiedono l'impostazione esplicita del bit W, a differenza di LDR e STR che scrivono sempre indietro quando sono post-indicizzate.

Il valore del registro di base, modificato dall'offset in un'istruzione preindicizzata, viene utilizzato come indirizzo per il trasferimento della prima parola. La seconda parola (se ne viene trasferita più di una) andrà o verrà da un indirizzo di una parola (4 byte) più alto del primo trasferimento, e l'indirizzo verrà incrementato di una parola per ogni trasferimento successivo.

4.15.3 Allineamento dell'indirizzo

L'indirizzo di base deve essere normalmente una quantità allineata alla parola. I 2 bit inferiori dell'indirizzo appariranno su **A[1:0]** e potranno essere interpretati dal sistema di memoria.

4.15.4 Uso di R15

Se Rn è R15, il valore utilizzato sarà l'indirizzo dell'istruzione più 8 byte. Il ritorno della base a R15 non deve essere specificato.

4.15.5 Interruzione dei dati

Se l'indirizzo è legale ma il gestore della memoria genera un'interruzione, la trappola dei dati viene eseguita. Avrà luogo il write-back della base modificata, ma tutti gli altri stati del processore saranno preservati. Il coprocessore è in parte responsabile di garantire che il trasferimento dei dati possa essere riavviato dopo che la causa dell'interruzione è stata risolta e deve garantire che qualsiasi azione successiva intrapresa possa essere ripetuta quando l'istruzione viene riproposta.

4.15.6 Tempi di ciclo delle istruzioni

Le istruzioni di trasferimento dati del coprocessore richiedono (n-1)S + 2N + bI cicli incrementali per essere eseguite, dove:

n è il numero di parole trasferite.

b è il numero di cicli trascorsi nel ciclo busy-wait del coprocessore.

S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.15.7 Sintassi dell'assemblatore

```
<LDC|STC>{cond}{L} p#,cd,<Indirizzo>
```

LDC caricamento dalla memoria al coprocessore STCstore dal coprocessore alla memoria

{L}quando è presente esegue il trasferimento lungo (N=1), altrimenti esegue il trasferimento corto (N=0)

{cond}mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

p#il numero univoco del coprocessore richiesto

cè un'espressione che valuta un numero di registro del coprocessore valido che viene inserito nel campo CRd

<Indirizzo> può essere:

1 Un'espressione che genera un indirizzo:

Finale - Accesso

`<espressione>`

L'assemblatore cercherà di generare un'istruzione utilizzando il PC come base e un offset immediato corretto per indirizzare la posizione data dalla valutazione dell'espressione. Si tratterà di un indirizzo relativo al PC, preindicizzato. Se l'indirizzo non rientra nell'intervallo, viene generato un errore.

2 Una specifica di indirizzamento preindicizzata:

`[Rn]`	offset di zero
`[Rn,<#espressione>]{!}`	offset di <espressione> byte

3 Una specifica di indirizzamento post-indicizzata:

`[Rn],<#espressione>`	offset di <espressione> byte

{Riscrivere il registro di base (impostare il bit W) se è presente!

Rn	è un'espressione che valuta un numero di registro ARM7TDMI-S valido.

Nota *Se Rn è R15, l'assemblatore sottrarrà 8 dal valore di offset per consentire il pipelining.*

4.15.8 Esempi

```
    LDC p1,c2,tabella; Carica c2 della coproc 1 dall'indirizzo
                     ; utilizzando un indirizzo relativo al PC.
STCEQLp2,c3,[R5,#24]!; memorizza in modo condizionato c3 della coproc 2
                       in un indirizzo a 24 byte da R5,
                       scrivere questo indirizzo in R5 e usare
                       ; opzione di trasferimento lungo (probabilmente a
                       ; memorizzare più parole).
```

Nota *Sebbene l'offset dell'indirizzo sia espresso in byte, il campo di offset dell'istruzione è espresso in parole.*

L'assemblatore regolerà l'offset in modo appropriato.

4.16 Trasferimenti di registro del coprocessore (MRC, MCR)

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5. La codifica delle istruzioni è mostrata nella ***Figura 4-27: Istruzioni di trasferimento dei registri del coprocessore***.

Questa classe di istruzioni viene utilizzata per comunicare informazioni direttamente tra ARM7TDMI-S e un coprocessore. Un esempio di istruzione di trasferimento di registro da coprocessore ad ARM7TDMI-S (MRC) è un FIX di un valore in virgola mobile contenuto in un coprocessore, in cui il numero in virgola mobile viene convertito in un intero a 32 bit all'interno del coprocessore e il risultato viene poi trasferito al registro ARM7TDMI-S. Un FLOAT di un valore a 32 bit nel registro ARM7TDMI-S in un valore in virgola mobile all'interno del coprocessore illustra l'uso del trasferimento dal registro ARM7TDMI-S al coprocessore (MCR).

Un uso importante di questa istruzione è quello di comunicare informazioni di controllo direttamente dal coprocessore ai flag del CPSR dell'ARM7TDMI-S. Ad esempio, il risultato di un confronto tra due valori in virgola mobile all'interno di un coprocessore può essere spostato nel CPSR per controllare il successivo flusso di esecuzione.

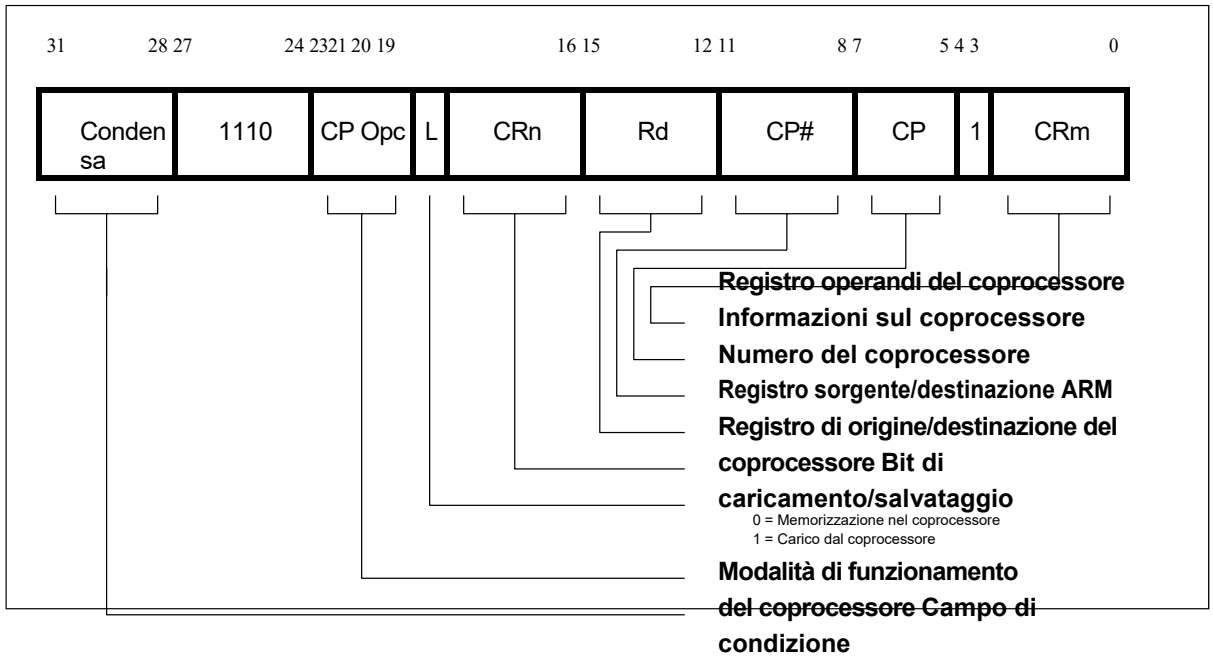


Figura 4-27: Istruzioni di trasferimento dei registri del coprocessore

4.16.1 I campi del coprocessore

Il campo CP# viene utilizzato, come per tutte le istruzioni del coprocessore, per specificare quale coprocessore viene richiamato.

I campi CP Opc, CRn, CP e CRm sono utilizzati solo dal coprocessore e l'interpretazione qui presentata deriva solo da una convenzione. Sono ammesse altre interpretazioni se la funzionalità del coprocessore è incompatibile con questa. L'interpretazione convenzionale è che i campi CP Opc e CP specificano l'operazione che il coprocessore deve eseguire, CRn è il registro del coprocessore che è l'origine o la destinazione dell'informazione trasferita e CRm è un secondo registro del coprocessore che può essere coinvolto in qualche modo che dipende dalla particolare operazione specificata.

4.16.2 Trasferimenti a R15

Quando il trasferimento di un registro del coprocessore all'ARM7TDMI-S ha come destinazione R15, i bit 31, 30, 29 e 28 della parola trasferita vengono copiati rispettivamente nei flag N, Z, C e V. Gli altri bit della parola trasferita vengono ignorati, mentre i bit PC e gli altri bit CPSR non vengono influenzati dal trasferimento.

4.16.3 Trasferimenti da R15

Un trasferimento di registro del coprocessore da ARM7TDMI-S con R15 come registro di origine memorizzerà PC+12.

4.16.4 Tempi di ciclo delle istruzioni

Le istruzioni MRC richiedono 1S + (b+1)I +1C cicli incrementali per essere eseguite, dove S, I e C sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

Le istruzioni MCR richiedono 1S + bI +1C cicli incrementali per essere eseguite, dove *b* è il numero di cicli spesi nel ciclo di attesa del coprocessore.

4.16.5 Sintassi dell'assemblatore

```
<MCR|MRC>{cond} p#,<espressione1>,Rd,cn,cm{,<espressione2>}
```

MRCmuovi dal coprocessore al registro ARM7TDMI-S (L=1)

MCRmuovi dal registro ARM7TDMI-S al coprocessore (L=0)

{cond}mnemonico di condizione a due caratteri. Vedere ***Tabella 4-2: Riepilogo dei codici di condizione*** a pagina 4-5.

p#il numero univoco del coprocessore richiesto

<espressione1> valutata come una costante e inserita nel campo CP Opc

Rè un'espressione che valuta un numero di registro ARM7TDMI-S valido

cn e cm sonoespressioni che valutano rispettivamente i numeri di registro validi del coprocessore CRn e CRm

<espressione2 dove present è valutato come una costante e collocato nel campo CP

4.16.6 Esempi

```
MRC p2,5,R3,c5,c6   ; Richiesta alla coproc 2 di eseguire l'operazione 5
                    su c5 e c6, e trasferire il (singolo)
                    ; parola a 32 bit) il risultato viene riportato in R3.

MCR p6,0,R4,c5,c6   ; Richiesta alla coproc 6 di eseguire l'operazione 0
                    ; su R4 e posizionare il risultato in c6.

MRCEQp3,9,R3,c5,c6,2; richiede in modo condizionato la coproc 3 a
                    ; eseguire l'operazione 9 (tipo 2) su c5 e
                    c6 e trasferire il risultato in R3.
```

4.17 Istruzione non definita

L'istruzione viene eseguita solo se la condizione è vera. Le varie condizioni sono definite nella ***Tabella 4-2: Riassunto del codice di condizione*** a pagina 4-5. Il meccanismo di trappola per istruzioni non definite è previsto per trappolare l'uso di codifiche di istruzioni attualmente non utilizzate. Questo meccanismo può essere utilizzato per fornire compatibilità all'indietro, consentendo l'emulazione di istruzioni incluse in architetture future. I sistemi operativi possono anche utilizzare questo meccanismo per impostare punti di interruzione delle istruzioni, che restituiranno il controllo al sistema operativo attraverso il meccanismo di trappola per istruzioni non definite. Esistono diverse aree di spazio per le istruzioni non definite, la più grande delle quali è codificata come segue: ***Figura 4-28: Istruzione non definita***

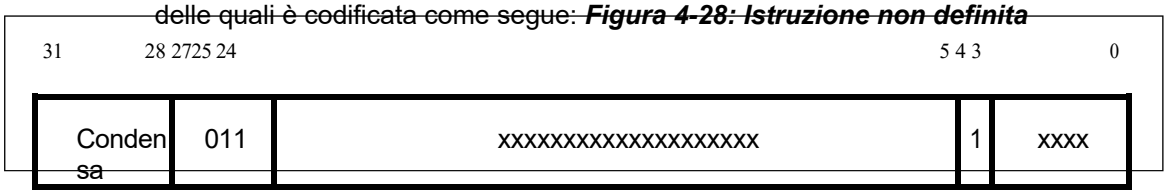


Figura 4-28: Istruzione non definita

Se la condizione è vera, viene eseguita la trappola dell'istruzione undefined. Se è necessaria un'istruzione non definita, si consiglia di utilizzare questa istruzione con tutti i bit contrassegnati da x impostati a 0.

4.17.1 Tempi di ciclo delle istruzioni

Questa istruzione richiede 2S + 1I + 1N cicli, dove S, N e I sono definiti in ***6.2 Tipi di ciclo*** a pagina 6-3.

4.17.2 Sintassi dell'assemblatore

L'assemblatore non dispone di mnemoniche per la generazione di questa istruzione. Se in futuro verrà adottata per un uso specifico, verranno aggiunte all'assemblatore delle mnemotecniche adatte.

4.18 Esempi di set di istruzioni

I seguenti esempi mostrano i modi in cui le istruzioni ARM di base possono essere combinate per ottenere un codice efficiente. Nessuno di questi metodi consente di risparmiare una grande quantità di tempo di esecuzione (anche se possono risparmiarne un po'), per lo più si limita a risparmiare codice.

4.18.1 Utilizzo delle istruzioni condizionali

Utilizzo dei condizionali per l'OR logico

```
CMP    Rn,#p            ; Se Rn=p O Rm=q ALLORA VAI A
                          Etichetta.
BEQ    Etichetta
CMP    Rm,#q
BEQ    Etichetta
```

Questo può essere sostituito da

```
CMP    Rn,#p
CMPNE Rm,#q             ; Se la condizione non è soddisfatta provare
                        ; altro test.
BEQ    Etichetta
```

Valore assoluto

```
TEQ    Rn,#0            ; Segno di prova
RSBMI Rn,Rn,#0          ; e il complemento a 2 se necessario.
```

Moltiplicazione per 4, 5 o 6 (tempo di esecuzione)

```
MOV    Rc,Ra,LSL#2      ; moltiplica per
4, CMP Rb,#5            ; valore di
prova,
ADDCSRc,Rc,Ra           ;completa
moltiplicazione per 5, ADDHI Rc,Rc,Ra ;
completa moltiplicazione per 6.
```

Combinazione di test discreti e test di portata

```
TEQ    Rc,#127          ; test discreto,
CMPNE Rc,#" "-1         ; test di gamma
MOVLS Rc,#"."           ;      IFRc<=" " O Rc=ASCII(127)
                       ; ALLORA Rc:="."
```

Divisione e resto

Una serie di routine di divisione per applicazioni specifiche sono fornite in forma sorgente come parte della libreria ANSI C fornita con il toolkit di sviluppo ARM Cross, disponibile presso il vostro fornitore. Segue una breve routine di divisione di uso generale.

```
                                ; Entrare con i numeri in Ra e Rb.
                                ;
      MOV    Rcnt,#1            ; Bit per controllare la divisione.
   Div1CMP   Rb,#0x80000000     ; sposta Rb finché non è maggiore di Ra.
      CMPCC Rb,Ra MOVCC
      Rb,Rb,ASL#1
      MOVCC Rcnt,Rcnt,ASL#1
      BCC    Div1
```

```
    MOV    Rc,#0
Div2CMP    Ra,Rb                ; Test di possibile sottrazione.
    SUBCS Ra,Ra,Rb              ; sottrarre se ok,
    ADDCSRc,Rc,Rcnt             ;inserire il bit
    corrispondente nel risultato MOVS  Rcnt,Rcnt,LSR#1     ;
    spostamento del bit di controllo
    MOVNERb,Rb,LSR#1            ;dimezza se non è
    finito. BNE                 Div2
                                ;
                                Dividere il risultato in Rc,
                                ; resto in Ra.
```

Rilevamento dell'overflow in ARM7TDMI-S

1 Overflow nella moltiplicazione senza segno con un risultato a 32 bit

```
UMULLO    Rd,Rt,Rm,Rn     da 3 a 6 cicli
TEQ       Rt,#0           ;ciclo +1 e un registro
BNE       traboccare
```

2 Overflow nella moltiplicazione firmata con un risultato a 32 bit

```
SMULL     Rd,Rt,Rm,Rn     da 3 a 6 cicli
TEQ       Rt,Rd ASR#31    ;ciclo +1 e un registro
BNE       traboccare
```

3 Overflow in un accumulo di moltiplicazione senza segno con un risultato a 32 bit

```
UMLAL     Rd,Rt,Rm,Rn     da 4 a 7 cicli
TEQ       Rt,#0           ;ciclo +1 e un registro
BNE       traboccare
```

4 Overflow nella moltiplicazione firmata accumulata con un risultato a 32 bit

```
SMLAL     Rd,Rt,Rm,Rn     ; da 4 a 7 cicli
TEQRt     ,Rd, ASR#31 ;+1 ciclo e un overflow
del registro BNE overflow
```

5 Overflow in un accumulo di moltiplicazione senza segno con un risultato a 64 bit

```
UMULLO    Rl,Rh,Rm,Rn     da 3 a 6 cicli
ANNUNCI   Rl,Rl,Ra1       ;accumulo inferiore
ADC       Rh,Rh,Ra2       ;accumulo superiore
BCS       traboccare      ;1 ciclo e 2 registri
```

6 Overflow nella moltiplicazione firmata accumulata con un risultato a 64 bit

```
SMULL     Rl,Rh,Rm,Rn     da 3 a 6 cicli
ANNUNCI   Rl,Rl,Ra1       ;accumulo inferiore
ADC       Rh,Rh,Ra2       ;accumulo superiore
BVS       traboccare      ;1 ciclo e 2 registri
```

Nota *Il controllo dell'overflow non è applicabile alle moltiplicazioni senza segno e firmate con un risultato a 64 bit, poiché l'overflow non si verifica in questi calcoli.*

4.18.2 Generatore di sequenze binarie pseudocasuali

Spesso è necessario generare numeri (pseudo)casuali e gli algoritmi più efficienti si basano su generatori a turni con retroazione a OR esclusivo, come un generatore di controllo di ridondanza ciclico. Sfortunatamente la sequenza di un generatore a 32 bit ha bisogno di più di un tap di feedback per essere di lunghezza massima (cioè 2^32-1 cicli prima della ripetizione), quindi questo esempio utilizza un registro a 33 bit con tap ai bit 33 e 20. L'algoritmo di base

è newbit:=bit 33 o bit 20, si sposta a sinistra il numero di 33 bit e si inserisce newbit in basso; questa operazione viene eseguita per tutti i newbit necessari (cioè 32 bit). L'intera operazione può essere eseguita in 5 cicli S:

```
                        ; Entrare con il seme in Ra (32
                          bit), Rb (1 bit in Rb lsb), usa
                          Rc.
                        ;
TST   Rb,Rb,LSR#1       ; bit superiore in
carry MOVSRc,Ra,RRX     ;33 bit ruotati a
destra
ADC   Rb,Rb,Rb          ; riporto nel lsb di
Rb EORRc,Rc,Ra,LSL#12   ; (coinvolto!)
EOR   Ra,Rc,Rc,LSR#20   ; (coinvolto in modo simile!)
                        ; nuovo seme in Ra, Rb come prima
```

4.18.3 Moltiplicazione per costante con il barrel shifter

Moltiplicazione per 2^n (1,2,4,8,16,32..)

```
MOVRa  , Rb, LSL #n
```

Moltiplicazione per 2^n+1 (3,5,9,17..)

```
ADDRa  ,Ra,Ra,LSL #n
```

Moltiplicazione per 2^n-1 (3,7,15..)

```
RSBRa  ,Ra,Ra,LSL #n
```

Moltiplicazione per 6

```
ADDRa  ,Ra,Ra,LSL #1 ; moltiplica per
3 MOV  Ra,Ra,LSL#1     ; e poi per 2
```

Moltiplicare per 10 e aggiungere un numero in più

```
ADDRa  ,Ra,Ra,LSL#2 ; moltiplicare per 5
ADDRa  ,Rc,Ra,LSL#1 ; moltiplica per 2 e aggiunge la cifra successiva
```

Metodo ricorsivo generale per Rb := Ra*C, C una costante:

1 Se C è pari, = 2^n*D, D dispari:
diciamo C

```
D=1:     MOVRb ,Ra,LSL #n
D<>1:    {Rb := Ra*D}
         MOVRb    ,Rb,LSL #n
```

2 Se C MOD 4 = 1, diciamo C = 2^n*D+1, D dispari, n>1:

```
D=1:     ADDRb ,Ra,Ra,LSL #n
D<>1:    {Rb := Ra*D}
         ADDRb    ,Ra,Rb,LSL #n
```

3 Se C MOD 4 = 3, diciamo C = 2^n*D-1, D dispari, n>1:

```
D=1:     RSBRb ,Ra,Ra,LSL #n
D<>1:    {Rb := Ra*D}
         RSBRb    ,Ra,Rb,LSL #n
```

Questo metodo non è del tutto ottimale, ma ci si avvicina. Un esempio della sua non ottimalità è la moltiplicazione per 45, che viene eseguita da:

```
RSBRb    ,Ra,Ra,LSL#2 ; moltiplicare per 3
RSBRb    ,Ra,Rb,LSL#2 ; moltiplicare per 4*3-1 =
11 ADDRb ,Ra,Rb,LSL# 2; moltiplicare per 4*11+1
```

piuttosto che da:

```
ADDRb    ,Ra,Ra,LSL#3 ; moltiplicare per 9
ADDRb    ,Rb,Rb,LSL#2 ; moltiplicare per 5*9 = 45
```

4.18.4 Caricamento di una parola da un allineamento sconosciuto

```
                    ; inserire l'indirizzo in Ra (32 bit)
                    ; utilizza Rb, Rc; il risultato è Rd.
                    Nota: d deve essere inferiore a c, ad esempio 0,1.
                    ;
BIC    Rb,Ra,#3     ; ottiene l'indirizzo
allineato alle parole LDMIARb,{Rd,Rc}    ;ottiene 64
bit contenenti la risposta AND    Rb,Ra,#3    ;
fattore di correzione in byte
MOVS   Rb,Rb,LSL#3    ; ... ora in bit e verifica se è
allineato MOVNERd,Rd,LSR Rb ; produce il fondo della
parola risultato
                    ; (se non è allineato)
RSBNERb,Rb,#32      ;ottenere l'altro
valore di spostamento
ORRNE Rd,Rd,Rc,LSL Rb ; combinare le due metà per ottenere il risultato
```