Assembly ARM

Set di istruzioni

Le istruzioni dell'architettura ARM possono essere suddivise in tre insiemi:

- **Istruzioni aritmetico-logiche** implementano le operazioni matematiche. Queste possono essere di tipo **aritmetiche** (somma, differenza, prodotto), **logiche** (operazioni booleane) o **relazionali** (comparazioni tra due valori).
- **Istruzioni di Branch** cambiano il controllo del flusso delle istruzioni, modificando il contenuto del Program Counter (R15). Le istruzioni di branch sono necessarie per l'implementazione di istruzioni di condizione (if-then-else), cicli e chiamate di funzioni.
- **Istruzioni di Load/Store** muovono dati da (load) e verso (store) la memoria principale.

Istruzioni aritmetico-logiche

In linguaggio assembly ARM le istruzioni aritmetico-logiche hanno una **sintassi** del tipo

opcode{S}{condition} dest, op1, op2

- Per **opcode** si intende l'istruzione che deve essere eseguita.

- Il campo **{S}** è un suffisso opzionale che, se specificato, carica nel registro di stato CPSR il risultato dell'operazione compresi i flag di stato.

Esempi istruzioni aritmetico-logiche

ADD R0, R1, R2

Carico in R0 la somma tra il contenuto del registro R1 e del registro R2 (indirizzamento a registro).

ADD R0, R1, #16

Carico in R0 la somma tra il contenuto del registro R1 e il numero decimale 16 (indirizzamento immediato).

ADD R0, R1, #0xF

Carico in R0 la somma tra il contenuto del registro R1 e il numero esadecimale F (indirizzamento immediato).

Esempi istruzioni aritmetico-logiche

ADDLT R0, R1, R2

Carico in R0 la somma tra il contenuto del registro R1 e del registro R2 solamente se la condizione indicata con il suffisso **LT** è verificata.

ADDS R0, R1, R2

Carico in R0 la somma tra il contenuto del registro R1 e del registro R2, inoltre carico lo stato del risultato nei flags NZCV del registro di stato CPSR.

Per esempio, se il risultato della somma va in overflow i flag C (carry) e V (overflow) verranno settati a 1.

Principali istruzioni aritmetico-logiche

Descrizione	Opcode	Sintassi	Semantica
Addizione	ADD	ADD R_d, R_1, R_2/#imm	$R_d = R_1 + R_2$/#imm
Sottrazione	SUB	SUB R_d, R_1, R_2/#imm	$R_d = R_1 - R_2$/#imm
Moltiplicazione	MUL	MUL R_d, R_1, R_2/#imm	$R_d = R_1 \times R_2$/#imm
Carica nel registro	MOV	MOV R_d, R_1/#imm	$R_d \leftarrow R_1$/#imm
Carica negato nel registro	MVN	MVN R_d, R_1/#imm	$R_d \leftarrow -(R_1$/#imm)
AND logico	AND	AND R_d, R_1, R_2/#imm	$R_d = R_1 \wedge R_2$/#imm
OR logico	ORR	ORR R_d, R_1, R_2/#imm	$R_d = R_1 \vee R_2$/#imm
OR esclusivo	EOR	EOR R_d, R_1, R_2/#imm	$R_d = R_1 \oplus R_2$/#imm
AND con complemento del secondo operando	BIC	BIC R_d, R_1, R_2/#imm	$R_d = R_1 \wedge \neg R_2$/#imm
Shift logico sinistro	LSL	LSL R_d, R_1, R_2/#imm	$R_d = R_1 \ll R_2$/#imm
Shift logico destro	LSR	LSR R_d, R_1, R_2/#imm	$R_d = R_1 \gg R_2$/#imm
Rotazione destra	ROR	ROR R_d, R_1, R_2/#imm	$R_d = R_1 \circlearrowright R_2$/#imm
Confronto	CMP	CMP R_1, R_2	NZCV $\leftarrow R_1 - R_2$
Negato del confronto	CMN	CMN R_1, R_2	NZCV $\leftarrow R_1 + R_2$

Istruzioni di branch

Le istruzioni di **branch** (o **jump**) sono utili per il controllo del flusso di esecuzione delle istruzioni.
Le principali istruzioni sono **B** (branch) e **BL** (branch with link).

L'istruzione B carica nel PC (R15) l'indirizzo della prima istruzione della procedura che si desidera eseguire, causando così un salto nel flusso di esecuzione delle istruzioni. Per comodità le procedure vengono identificate univocamente con dei nomi detti **label** (o **etichetta**).

L'istruzione BL è simile all'istruzione B, in più però carica nel registro LR (R14) l'indirizzo di ritorno della procedura, ovvero il valore del PC nel momento in cui viene eseguita l'istruzione BL.

Istruzione di Load e Store

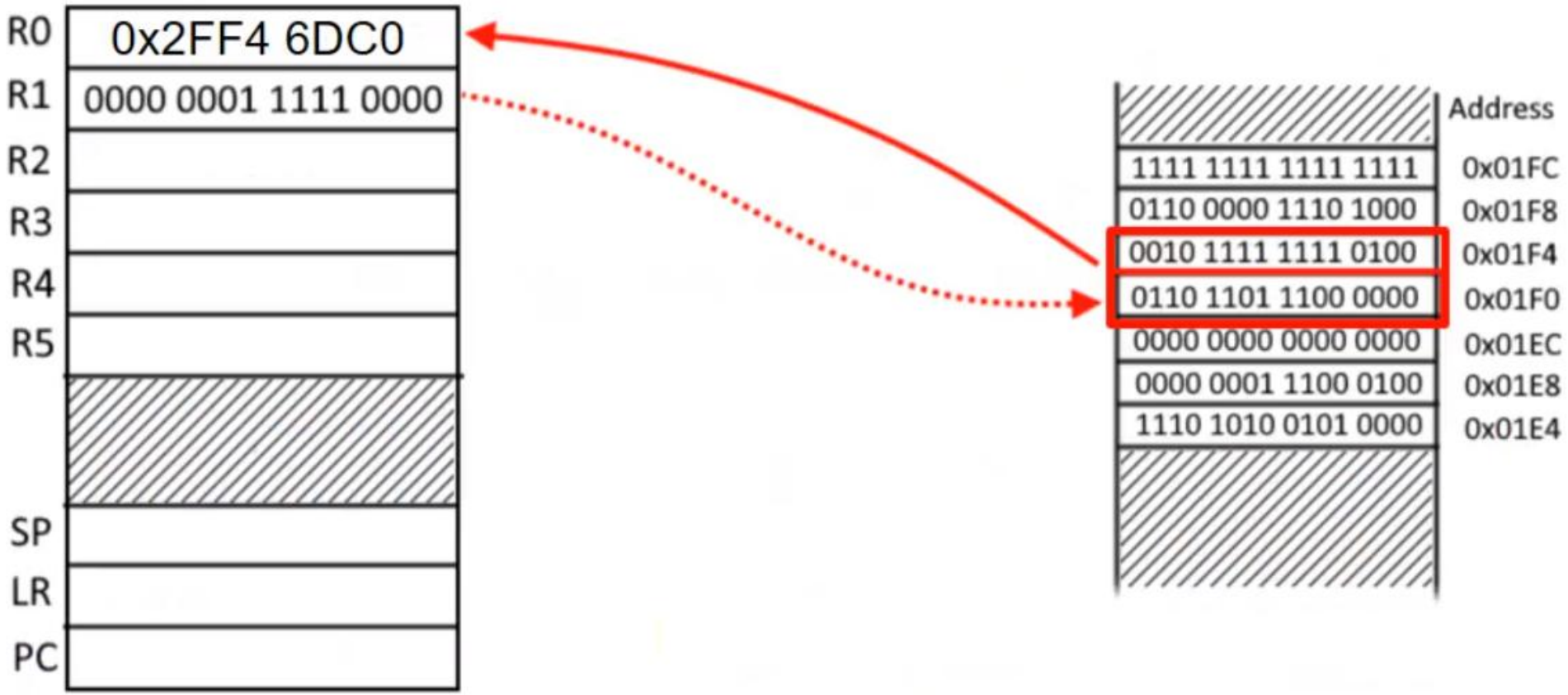
L'istruzione **LDR** (**Load Register**) carica in un registro destinatario un byte, una half world o una full word contenuta in una data locazione della memoria principale. La sintassi è

LDR{type}{condition} dest, [pointer]
LDR{type}{condition} dest, [pointer,offset]

Il campo **dest** è il registro destinatario nel quale caricare il contenuto prelevato dalla memoria. Il campo **[pointer]** indica un **registro puntare** il quale definisce contenuto punta all'indirizzo in memoria della **cella** che si desidera referenziare. É possibile inoltre definire un **offset** il quale verrà sommato all puntatore.

LDR R0, [R1, #2]

Carico in R0 la **parola** in memoria puntata dal registro R1

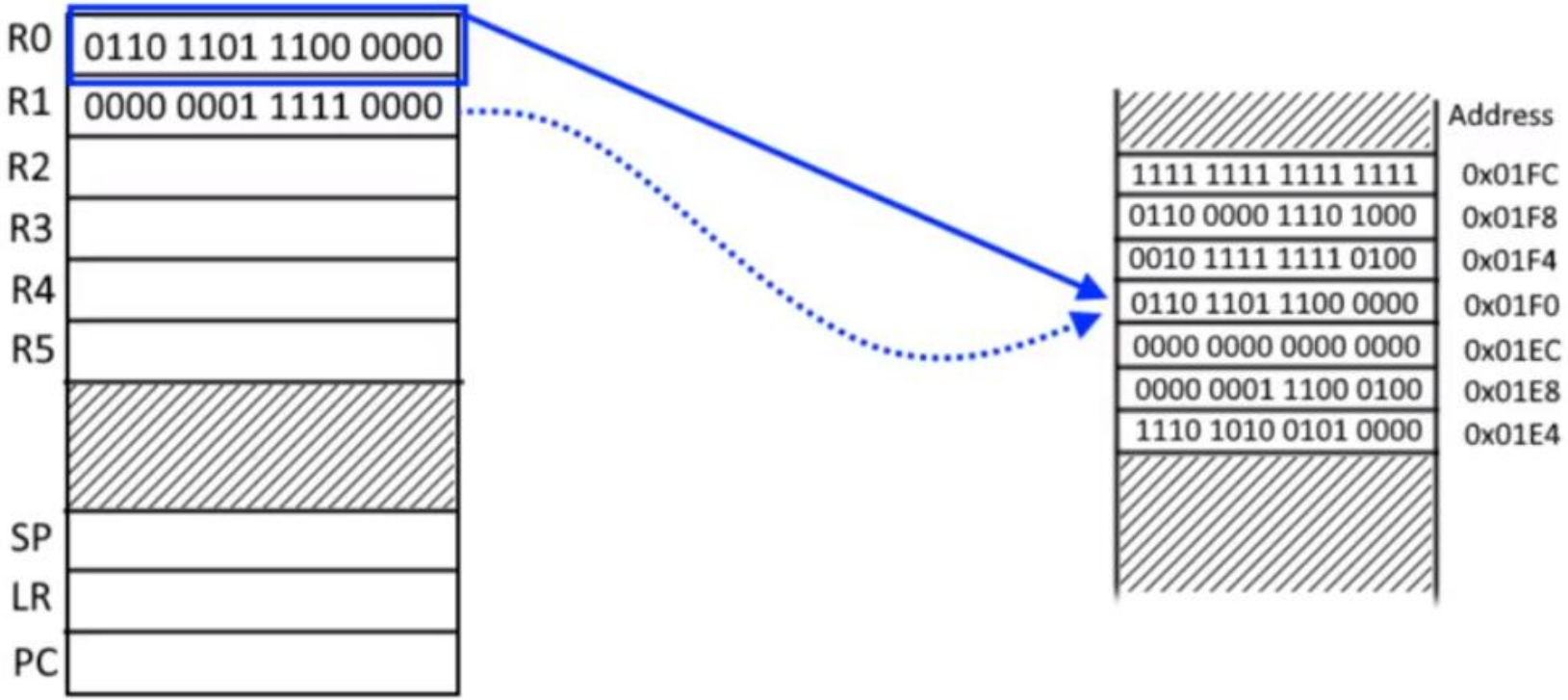


L'istruzione **STR** (**Store Register**) esegue esattamente l'operazione inversa di LDR; carica in memoria il contenuto di un registro **sorgente** all'indirizzo definito dal **puntatore**.

STR{condition} source, [pointer,offset]

STR R0, [R1]

Carico all'indirizzo puntato da R1 la parola contenuta nel registro R0



0x01F0 = 0000 0001 1111 0000