

Corso di Calcolatori Elettronici I

A.A. 2012-2013

Istruzioni macchina

ing. Alessandro Cilardo

Accademia Aeronautica di Pozzuoli
Corso Pegaso V "GArn Elettronici"

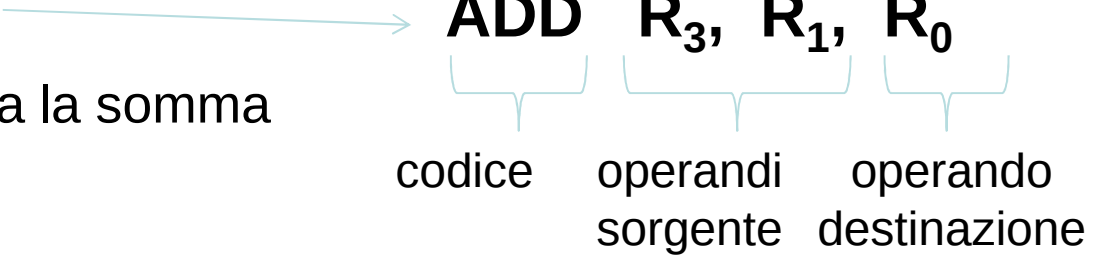
Istruzioni del processore

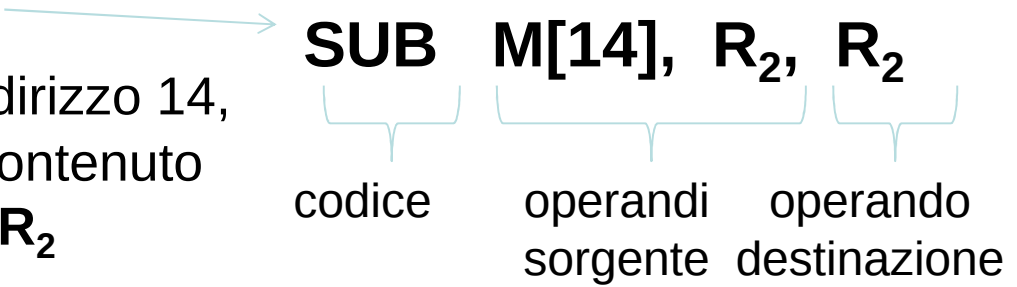
- Abbiamo visto in precedenza alcuni esempi di istruzioni elementari che il processore è in grado di interpretare ed eseguire:
 - **trasferisci** un dato da una locazione di memoria (**M**) ad uno degli n registri di macchina $R_0 \dots R_{n-1}$, e viceversa:
 - es.: $M[18] \rightarrow R_2$, $R_1 \rightarrow M[14]$,
 - **addiziona** i contenuti di due degli n registri $R_0 \dots R_{n-1}$:
 - es.: $R_2 + R_1 \rightarrow R_2$
 - **sottrai** i contenuti di due degli n registri $R_0 \dots R_{n-1}$:
 - es.: $R_3 - R_0 \rightarrow R_0$
 - **transla** (shift) il contenuto di uno degli n registri $R_0 \dots R_{n-1}$:
 - es.: **shiftL** (R_2) $\rightarrow R_2$ (Left-shift, ovvero transla R_2 a sinistra di un bit)

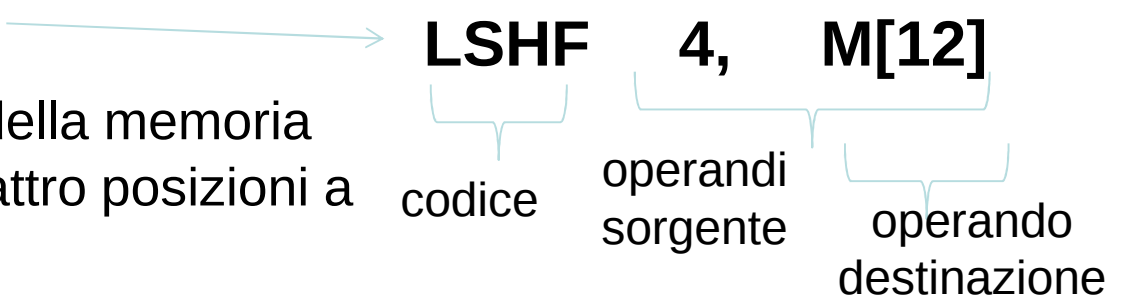
Istruzioni del processore

- Com'è strutturata *in generale* un'istruzione?
- E' in linea di principio una tripla $i = (f, P_1, P_2)$, dove:
 - $f \in F$ è l'insieme dei **codici operativi** del processore, cioè delle operazioni elementari definite al livello del linguaggio macchina;
 - P_1 è un insieme di **operandi-sorgente**, cioè di valori e/o indicazioni dei registri e/o indicazioni delle locazioni di memoria contenenti i valori su cui opera f
 - P_2 è un insieme di **operandi-destinazione**, cioè di indicazioni dei registri o indicazioni delle locazioni di memoria cui sono destinati i risultati dell'istruzione f
 - normalmente è presente un solo operando destinazione, che *può coincidere* con uno degli operandi sorgente

Istruzioni del processore: esempi

- **ADD R_3, R_1, R_0**
calcola $R_3 + R_1$ e sposta la somma nel registro R_0


ADD R_3, R_1, R_0
codice operandi sorgente operando destinazione
- **SUB $M[14], R_2, R_2$**
accedi alla memoria all'indirizzo 14, sottrai a questo valore il contenuto di R_2 e scrivi il risultato in R_2


SUB $M[14], R_2, R_2$
codice operandi sorgente operando destinazione
- **LSHF 4, $M[12]$**
trasla il contenuto della memoria all'indirizzo 12 di quattro posizioni a sinistra


LSHF 4, $M[12]$
codice operandi sorgente operando destinazione

Rappresentazione di un'istruzione

- Il codice operativo e l'indicazione degli operandi sono essi stessi un'informazione da immagazzinare **in memoria** (come i dati)
- Codice operativo e operandi sono quindi oggetto di una **codifica**, che fa corrispondere una stringa di '0' e '1' ad ogni codice operativo e possibile combinazione di operandi

ADD R₃, R₁, R₀  **0010110101010101**

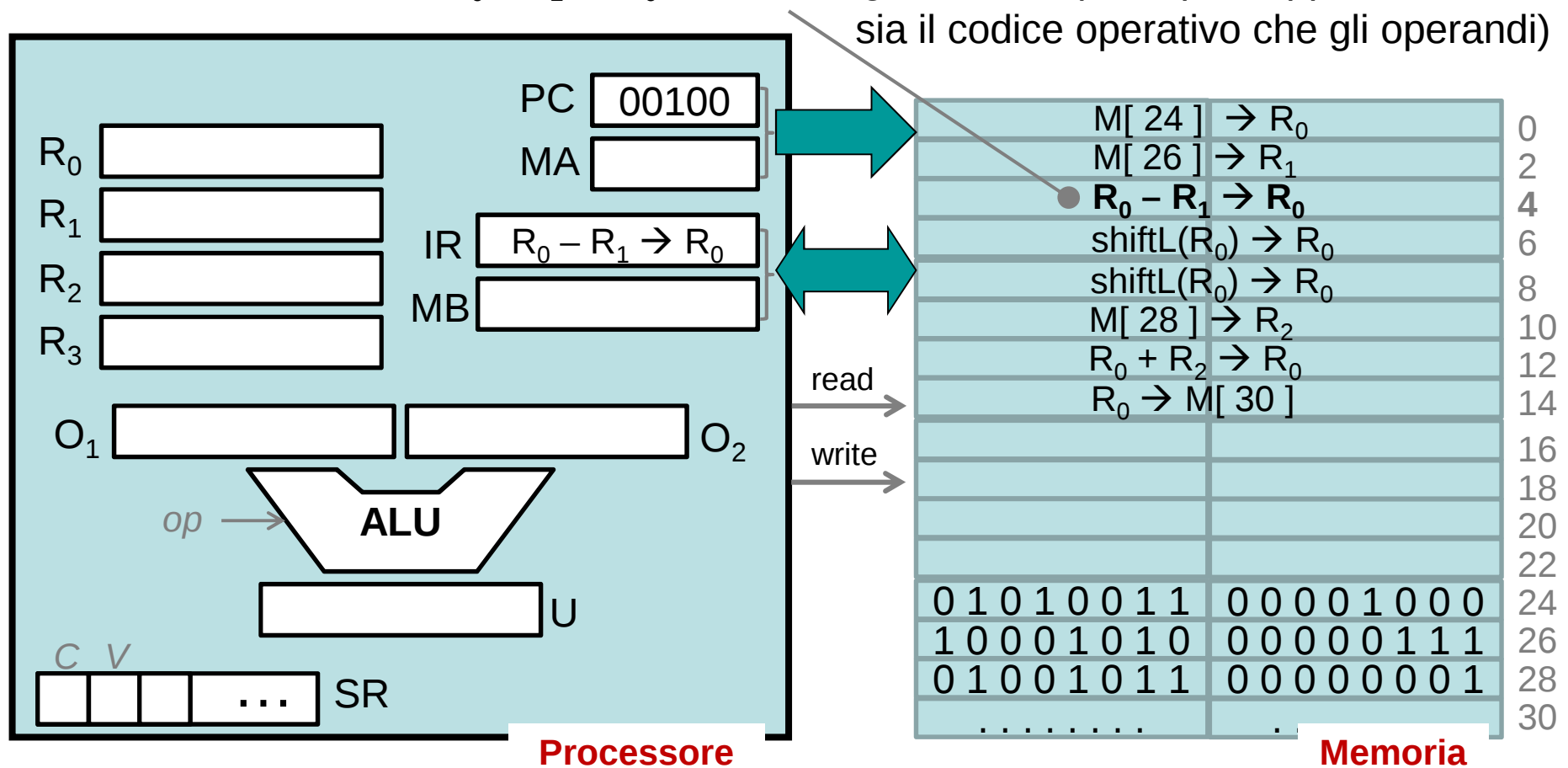
codifica

SUB M[14], R₂, R₂  **1001010111101010**

- Tale informazione è codificata in macchina mediante
 - codici a *lunghezza fissa* (tipicamente 32 bit, es. RISC)
 - o a *lunghezza variabile* (es. nel 68000 multipli di 16 bit)

Rappresentazione di un'istruzione

In questo esempio, la locazione di memoria all'indirizzo 4 contiene materialmente la codifica dell'istruzione $R_0 - R_1 \rightarrow R_0$ come stringa di 16 bit (usati per rappresentare sia il codice operativo che gli operandi)



Operandi delle istruzioni macchina

Rispetto agli operandi su cui operano, le istruzioni macchina si diversificano:

1. per tipo degli operandi (es. intero a 8, 16 o 32 bit);
2. per numero degli operandi espliciti (0, 1, 2 o 3);

e, per ciascun operando:

3. per la "natura" (ad esempio, se è una costante, se è il contenuto di un registro o di una locazione di memoria);
4. per la tecnica di indirizzamento (se è *implicito* o *esplicito*, etc.);

Nel seguito sono introdotti vari possibili criteri di classificazione delle istruzioni macchina

Classificazione delle istruzioni per numero di operandi espliciti

- Tipiche istruzioni macchina hanno 0, 1, 2 o 3 operandi espliciti:
 - OP es. **ClearAccumulator**
 - OP **O₁** es. **Clear R₀**
 - OP **O₁, O₂** es. **Move R₁, R₂**
 - OP **O₁, O₂, O₃** es. **Add R₄, R₆, R₁**ove **O₁, O₂, O₃** sono operandi espliciti
- Un'istruzione può avere operandi *impliciti*
 - ovvero, non indicati espressamente nella codifica dell'istruzione poiché questa fa riferimento sempre allo stesso operando
- Si tratta tipicamente della costante zero oppure di un particolare registro presente nel processore
 - ad esempio il registro *accumulatore*, nelle cosiddette “macchine ad accumulatore” (vedi primo esempio in alto)

Classificazione delle istruzioni per la natura degli operandi

- In funzione della natura degli operandi, le istruzioni sono classificate come:

- Memoria - Immediato
- Memoria - Registro
- Memoria - Memoria
- Registro - Immediato
- Registro - Registro

Operando Registro: l'istruzione indica il nome del registro (ad es. R_3)

Operando Memoria: l'istruzione indica la posizione in memoria (ad es. $M[14]$)

Operando Immediato: l'istruzione indica *direttamente il valore* da usare (ad es. 4), inserito all'interno della codifica in bit dell'istruzione stessa

- In ciascuna coppia, il primo termine indica la natura dell'operando destinazione, mentre il secondo termine indica la natura dell'operando (o degli operandi) sorgente
- Una CPU non supporta necessariamente tutte le possibili combinazioni sopra elencate; eccezioni sono possibili, anche per singole istruzioni

Classificazione delle istruzioni per codici operativi

- Ciascuna CPU è caratterizzata da un proprio repertorio di istruzioni macchina (**Instruction Set**)
- Il repertorio di codici operativi di una CPU può essere più o meno ricco
 - **CISC** (*Complex Instruction Set Computer*): molte istruzioni **complesse**, tendenzialmente più **lente** nella loro esecuzione
 - **RISC** (*Reduced Instruction Set Computer*): poche istruzioni **semplici**, tendenzialmente più **veloci** nella loro esecuzione
 - Nei RISC le istruzioni sono più veloci, ma in genere ne occorrono di più per realizzare un determinato calcolo rispetto ad un CISC
- In entrambi i casi, il repertorio può essere suddiviso tipicamente in poche classi di istruzioni fondamentali

Classi fondamentali di istruzioni

- Istruzioni di **trasferimento** dati
 - Copiano un dato dall'operando sorgente all'operando destinazione
- Istruzioni **aritmetiche**
 - Effettuano operazioni aritmetiche sugli operandi sorgente e memorizzano il risultato nell'operando destinazione
 - Operano tipicamente su dati numerici di tipo intero
- Istruzioni **logiche** e di **scorrimento** (*shift*)
 - Effettuano operazioni logiche booleane e di shift sugli operandi sorgente e memorizzano il risultato nell'operando destinazione
 - Operano tipicamente su dati di tipo “stringa di bit”

Classi fondamentali di istruzioni

- Istruzioni di **confronto**
 - Alterano i flag del registro di stato del processore (*Processor Status Word* o *Status Register*, **SR**) in base all'esito del confronto tra due operandi sorgente espliciti (istruzioni di *Compare*) o tra un operando sorgente esplicito ed uno implicito (tipicamente zero, come per l'istruzione *Test*)
- Istruzioni di **salto**
 - Alterano il flusso sequenziale che caratterizza la normale esecuzione delle istruzioni, consentendo la realizzazione di diramazioni (*if-then-else*) e *cicli*
 - Agiscono modificando il registro *Program Counter*
 - Possono essere *condizionate* (eseguite solo se è vera una delle condizioni espresse dai flag del registro di stato **SR**) o *non-condizionate* (eseguite sempre)

Classi fondamentali di istruzioni

- Istruzioni di collegamento a **sottoprogramma**
 - Sono special istruzioni di salto che permettono di passare ad eseguire una differente sequenza di istruzioni, a partire da quella corrente; al termine della nuova sequenza sarà poi possibile tornare al punto da cui è stato effettuato il salto
 - permettono di strutturare il programma in **sottoprogrammi** verso cui si può *saltare* e da cui si può *ritornare*
- Istruzioni di **input/output**
 - Alcune CPU sono dotate di istruzioni apposite per il trasferimento di dati da/verso le interfacce delle periferiche di input/output (*sottosistema di I/O*)

Istruzioni di trasferimento dati

- Copiano un dato dall'operando sorgente all'operando destinazione
- Tipicamente sono istruzioni a due operandi espliciti:

MOVE sorgente, destinazione

- In linea di principio, sorgente e destinazione possono essere un qualsiasi registro/locazione di memoria
- L'operando **sorgente** può anche essere un immediato.
- La **destinazione** NON può essere un immediato! (*perché?*)
- Le istruzioni in cui il *dato* trasferito rappresenta esso stesso un indirizzo di memoria sono spesso considerate a parte

Istruzioni di trasferimento dati

- Le istruzioni di tipo **Clear** (“*scrivi zero in un registro/locazione*”) assumono la costante zero come secondo operando sorgente implicito
 - **Clear R_1** $R_1 \leftarrow 0$
- Esiste una particolare tipologia di architetture, nelle quali è sempre presente un singolo registro speciale detto *accumulator*.
- In tali architetture, uno dei due operandi è implicito: l'accumulator, appunto (indicato qui con **ACC**)
 - **LoadAccumulator #5** $ACC \leftarrow 5$
 - **StoreAccumulator 1000** $M[1000] \leftarrow ACC$

Istruzioni aritmetiche

- Effettuano operazioni aritmetiche *unarie* (cambia segno) o *binarie* (addizione, sottrazione, moltiplicazione, divisione) su dati interi espressi su 8, 16, 32 bit
 - $a = \text{op } b$ operazione unaria (un operando)
 - $a = b \text{ op } c$ operazione binaria (due operandi)
- Alcune CPU sono dotate di istruzioni macchina per l'aritmetica in virgola mobile (IEEE 754)
- In altri casi, un apposito *coprocessore* esterno al processore principale fornisce l'estensione del set di istruzioni per il supporto alla virgola mobile
- Operazioni aritmetiche più complesse (es. radice quadrata) o funzioni trigonometriche ed esponenziali sono di solito supportate da coprocessori o realizzate in software

Istruzioni aritmetiche

- Molte CPU impongono il vincolo che l'operando destinazione coincida con uno degli operandi sorgente
 - $a = \text{op } a$ operazione unaria (un operando)
 - $a = a \text{ op } b$ operazione binaria (due operandi)
- Ciò consente di lavorare con istruzioni a due soli operandi espliciti (e di risparmiare sui bit per codificare l'istruzione!)
 - Es. nel Motorola 68000:
 - ADD D0,D1 $D1 \leftarrow [D0] + [D1]$
- Il formato di istruzioni a 3 operandi espliciti è tipico delle CPU RISC; in esse, però, c'è il vincolo che i tre operandi siano tutti di tipo registro
- Altre limitazioni sulla natura e sui modi di indirizzamento degli operandi valgono anche per le CPU CISC
 - Ad esempio, nel 68000 le istruzioni aritmetiche devono avere necessariamente un operando di tipo registro

Istruzioni logiche

- Effettuano operazioni logiche booleane “bit a bit” (*bitwise*) sia unarie (NOT) che binarie (AND, OR, XOR) su dati di tipo “stringa di bit” espressi su 8, 16, 32 bit

a = NOT b operazione unaria (un operando)
a = b AND c operazione binaria (due operandi)
a = b OR c operazione binaria (due operandi)
a = b XOR c operazione binaria (due operandi)

ESEMPI

	1 0 1 0 1 0 0 . 0 1 1	1 0 1 0 1 0 0 . 0 1 1	1 0 1 0 1 0 0 . 0 1 1
NOT	AND	OR	XOR
1 1 0 0 1 1 0 . 0 0 1	1 1 0 0 1 1 0 . 0 0 1	1 1 0 0 1 1 0 . 0 0 1	1 1 0 0 1 1 0 . 0 0 1
=	=	=	=
0 0 1 1 0 0 1 . 1 1 0	1 0 0 0 1 0 0 . 0 0 1	1 1 1 0 1 1 0 . 0 1 1	0 1 1 0 0 1 0 . 0 1 0
<i>inverte i bit singolarmente</i>	<i>se c'è almeno uno 0 in ingresso, il bit corrispondente è 0</i>	<i>se c'è almeno un 1 in ingresso, il bit corrispondente è 1</i>	<i>se i due bit in ingresso sono diversi, l'uscita è 1, altrimenti 0</i>

Istruzioni logiche

- L'operazione di AND può essere utilizzata per mettere selettivamente a **zero** alcuni bit in un registro o in una locazione di memoria (ovvero, per “*mascherare*” i bit)
 - **AND 1111111111111100, R₂**
*opera sul registro R₂ (di 16 bit). Il contenuto del registro non verrà modificato, ad eccezione dei due bit meno significativi messi a '0'. Infatti, la **AND** di un bit con 0 produce sempre 0, mentre la **AND** di un bit con 1 non altera il bit (0 **AND** 1 = 0, 1 **AND** 1 = 1)*
- L'operazione di OR può essere utilizzata per mettere selettivamente a **uno** alcuni bit in un registro o in una locazione di memoria
 - **OR 0000000000000011, R₁**
*opera sul registro R₁ (di 16 bit). Il contenuto del registro non verrà modificato, ad eccezione dei due bit meno significativi messi a '1'. Infatti, la **OR** di un bit con 1 produce sempre 1, mentre la **OR** di un bit con 0 non altera il bit (0 **OR** 0 = 0, 1 **OR** 0 = 1)*

Istruzioni logiche

- L'operazione di XOR può essere utilizzata per invertire selettivamente alcuni bit in un registro o in una locazione di memoria:

- **XOR 0000000000000011, R₁**

*opera sul registro R₁ (di 16 bit). Il contenuto del registro non verrà modificato, ad eccezione dei due bit meno significativi, che verranno invertiti (se sono '1' diventano '0' e viceversa). Infatti, la **XOR** è definita come la funzione che è 1 se gli ingressi sono diversi, pertanto: 0 **XOR** 1=1, 1 **XOR** 1= 0.*

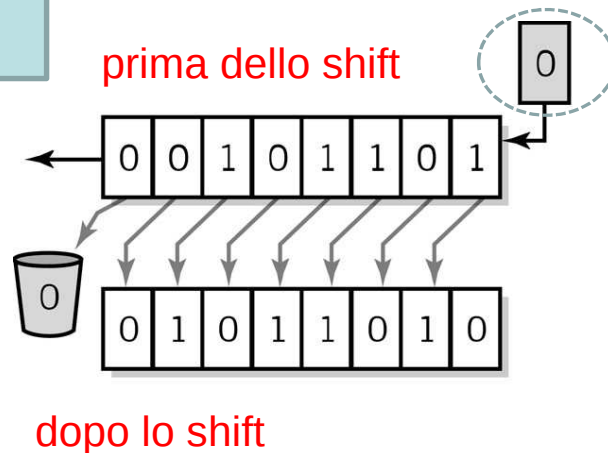
Istruzioni di scorrimento

- Similmente alle operazioni logiche operano su dati di tipo “stringa di bit”, traslandone il contenuto
- Operazioni tipiche:
 - *Shift-Left* logico
 - *Shift-Right* sia aritmetico che logico (vedi lucido successivo)
 - *Circular-Shift-Left*
 - *Circular-Shift-Right*
- Il numero di scorrimenti può essere fisso (tipicamente uno) o variabile (espresso all'interno dell'istruzione da un ulteriore operando, immediato o registro)
- **SHFL** R_1 *trasla a sinistra (Left) il registro R_1 di un bit*
- **ASHFR** R_3 *trasla a destra (Right) il registro R_3 di un bit preservando il segno*
- **SHFR** **3**, R_1 *trasla a destra (Right) il registro R_1 di **tre** bit*

Istruzioni di scorrimento

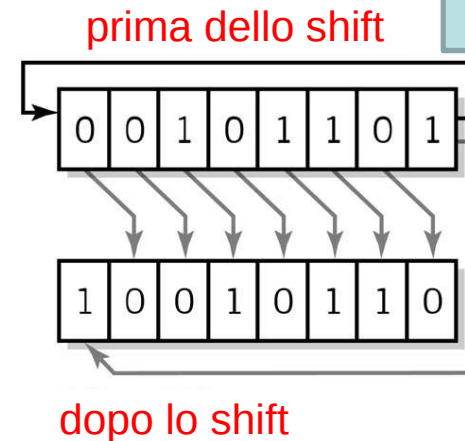
shift logico

Lo shift logico (a sinistra o a destra) **inserisce degli 0** nelle posizioni lasciate vuote



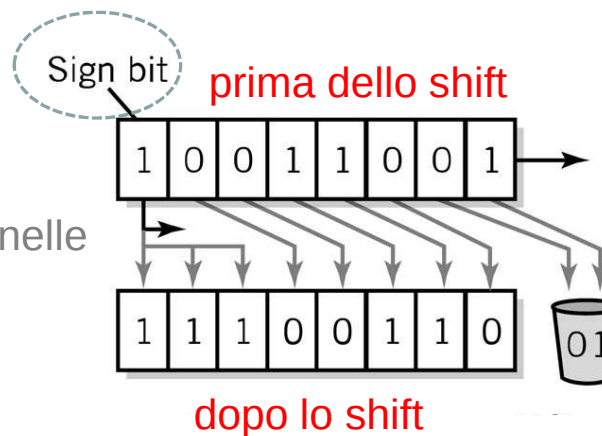
shift circolare

Lo shift circolare (a sinistra o a destra) **reinserisce il bit** eliminato dallo shift all'altra estremità. Nell'esempio, l'1 eliminato a destra ritorna a sinistra.



shift aritmetico

Lo shift aritmetico (solo a destra) **replica a sinistra il bit di segno** nelle posizioni lasciate vuote, per non cambiare il valore del numero se questo è rappresentato in complementi a 2

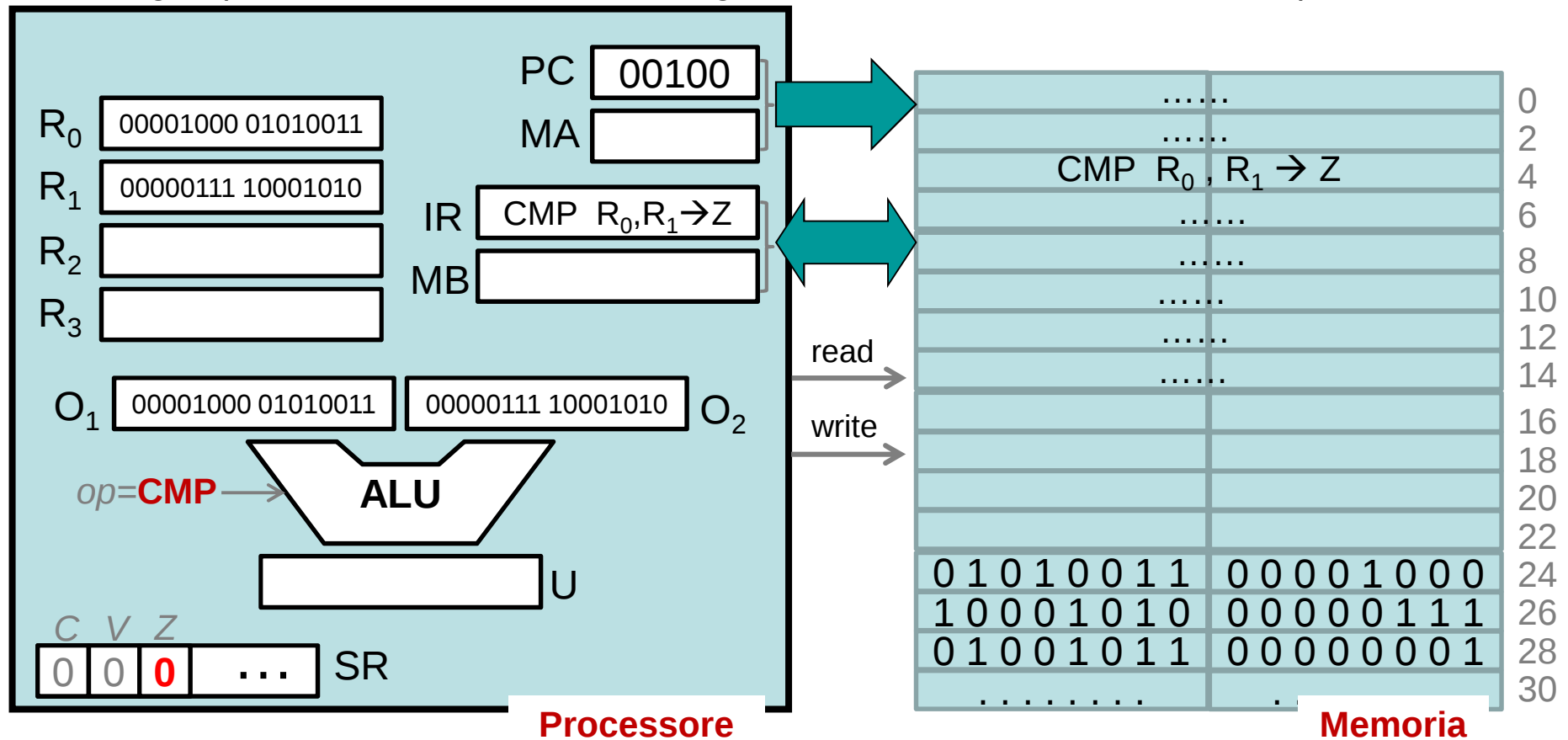


Istruzioni di confronto

- Alterano i flag del registro di stato del processore (*Processor Status Word* o *Status Register*) in base all'esito del confronto tra due operandi sorgente espliciti (istruzioni di *Compare* propriamente dette) o tra un operando sorgente esplicito ed uno implicito (tipicamente zero, come per l'istruzione *Test*)
- Tipicamente queste istruzioni precedono le istruzioni di salto condizionato, e congiuntamente ad esse consentono di realizzare strutture di programmazione quali le strutture di controllo *if-then-else* ed i *cicli*, tipici dei linguaggi di programmazione di alto livello

Istruzioni di confronto

Istruzione di confronto tra R_0 e R_1 . All'ALU viene comandato un confronto, che in realtà è una sottrazione. Se il risultato della sottrazione è zero (ovvero se i due valori confrontati sono uguali) viene messo a **1** il bit **Z** del registro di stato **SR**, altrimenti viene impostato a **0**.



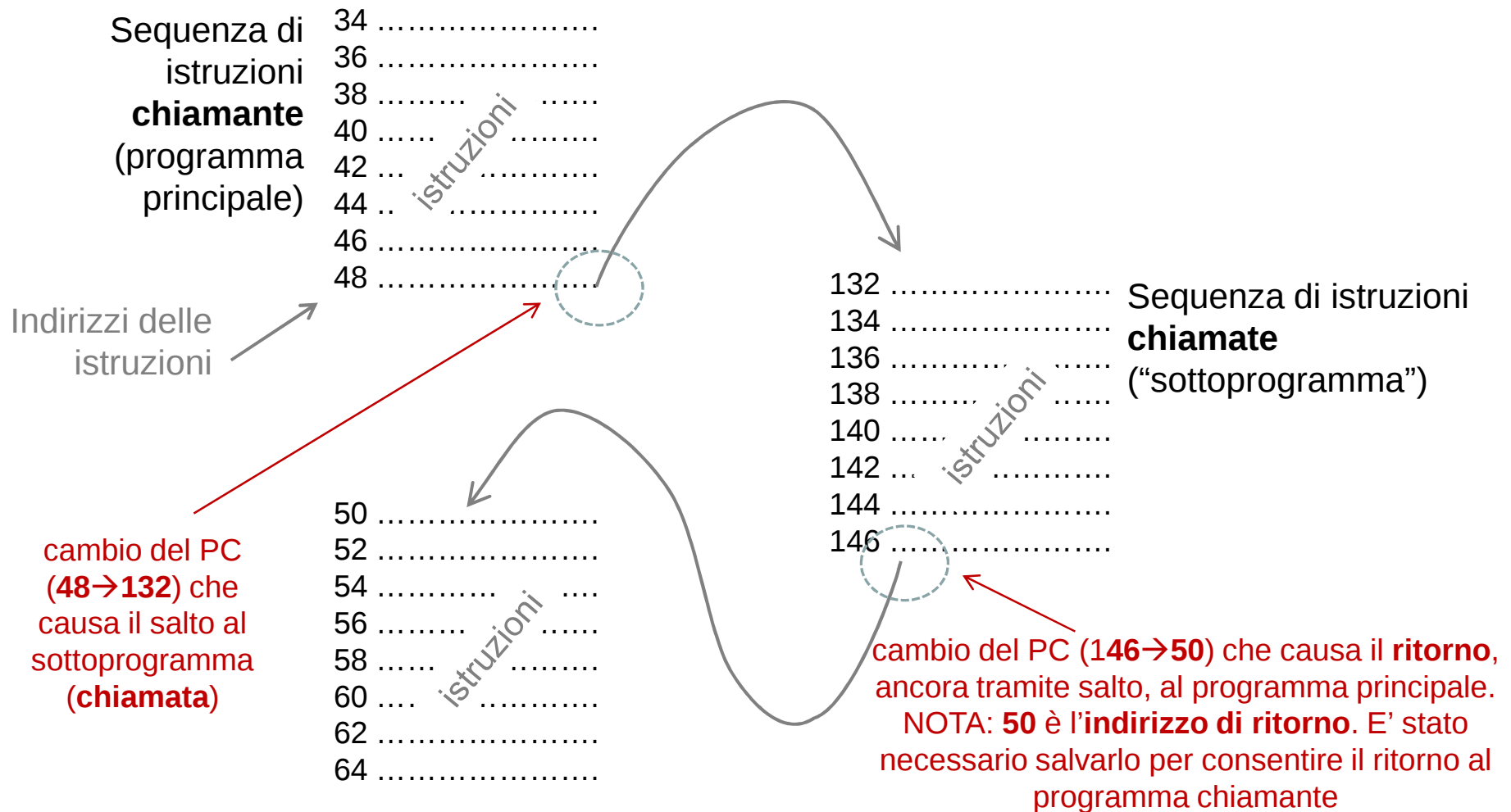
Istruzioni di salto

- Alterano il flusso sequenziale che caratterizza la normale esecuzione delle istruzioni
- Agiscono modificando il registro *Program Counter*
- Possono essere *condizionate* (dal fatto che sia vera una delle condizioni codificate dai flag del registro di stato, ad esempio il bit **Z** visto prima) o *non-condizionate*
- Si distinguono spesso in salti **assoluti** (*Jump*) e **relativi** (*Branch*)
 - Le istruzioni di **Jump** contengono nella codifica dell'istruzione l'indirizzo destinazione
 - Le istruzioni di **Branch** contengono nella codifica dell'istruzione uno spiazzamento (offset) 'x' che, sommato al **PC** attuale, determina l'indirizzo destinazione. Ci si sposta quindi non ad un specifico indirizzo, ma 'x' locazioni *prima* o *dopo* il valore attuale del **PC**

Istruzioni di collegamento a sottoprogramma

- Le istruzioni di salto a sottoprogramma (*Jump To Subroutine* o *Call*) salvano il valore del PC per consentire il ritorno al programma chiamante
- Le istruzioni di ritorno da sottoprogramma (*Return From Subroutine*) ripristinano il valore del PC salvato per realizzare il ritorno al programma chiamante
- Il valore del PC può essere salvato in un apposito registro (*Link Register*, LR) e/o in una opportuna area di memoria chiamata stack

Istruzioni di collegamento a sottoprogramma



Istruzioni di input/output

- Alcune CPU sono dotate di istruzioni apposite per il trasferimento di dati da/verso le interfacce delle periferiche di input/output
 - Istruzioni IN e OUT
- Si tratta in sostanza di istruzioni di trasferimento dati che operano su uno *spazio di indirizzamento* (quello delle interfacce di I/O) distinto da quello della memoria
- Nei sistemi nei quali spazio di indirizzamento di I/O e spazio di indirizzamento di memoria coincidono (sistemi con I/O “*memory mapped*”) le operazioni di I/O vengono eseguite tramite normali istruzioni di trasferimento dati