

Percorsi abilitativi speciali

Elettronica programmabile

04. Strutture di programmazione

Roberto Roncella



I salti condizionati

- Sono l'atomo di ogni struttura di programmazione
 - Permettono di seguire strade diverse sulla base del risultato delle operazioni precedenti
 - Esistono salti per tutte le diverse condizioni corrispondenti ai flag (C, Z, N, V, S, H, T, I)
 - Vengono usate espressioni che evocano l'evento che ha dato origine a un particolare valore dei flag
 - Es.: "salta se uguale" esegue il salto se $Z=1$, perché l'operazione di confronto funziona eseguendo una differenza tra i due termini

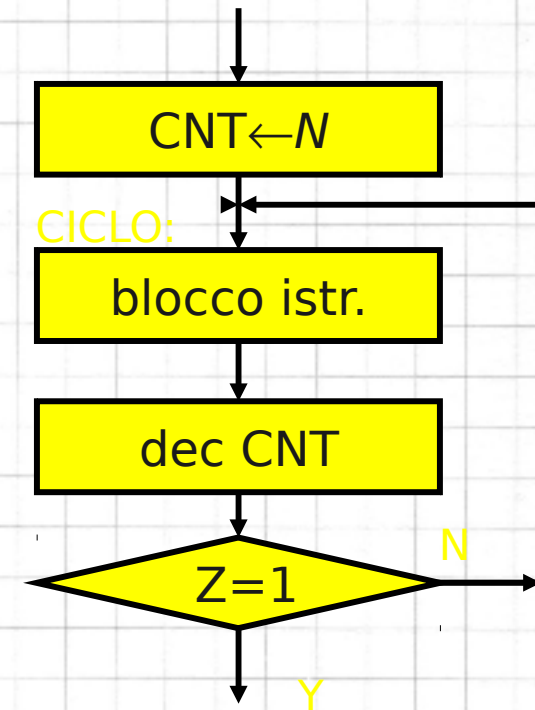
Tipi di salti condizionati

- Skip
 - Se la condizione è vera saltano una istruzione
 - Quella seguente, e poi proseguono
 - Altrimenti non fanno nulla
- Branch
 - Se la condizione è vera saltano all'indirizzo specificato
 - Può essere dato in modo relativo (vai avanti di n)
 - O assoluto (vai all'indirizzo k)
 - Altrimenti non fanno nulla

Ripeti N volte

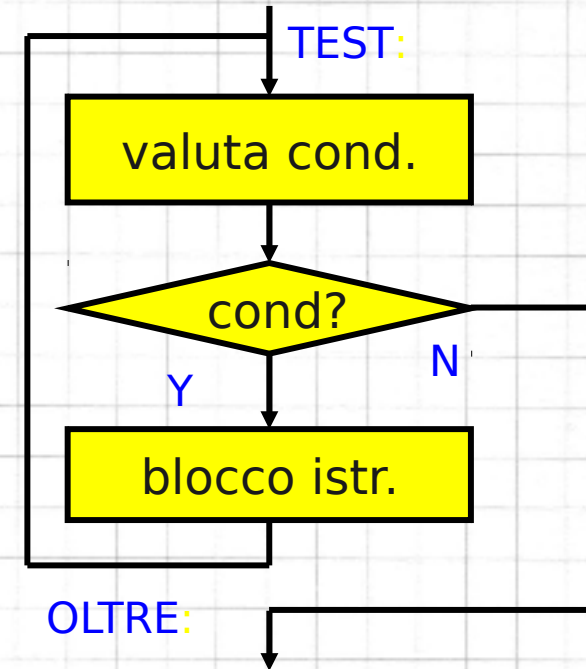
- Ciclo elementare
 - Numero di ripetizioni prestabilito
 - Ha bisogno di un contatore
 - Registro *Cnt*
 - Ha bisogno del flag Z e di un salto condizionato di tipo branch

```
ldi    CNT,4    ;inizializza CNT
CICLO: add   R1,R2 ;blocco di istruzioni
rol    R1       ;(qui a caso...)
ror    R2
dec    CNT      ;decrementa il contatore
brne   CICLO    ;ripeti per 4 volte
nop                    ;il programma prosegue
```



Ripeti mentre ...

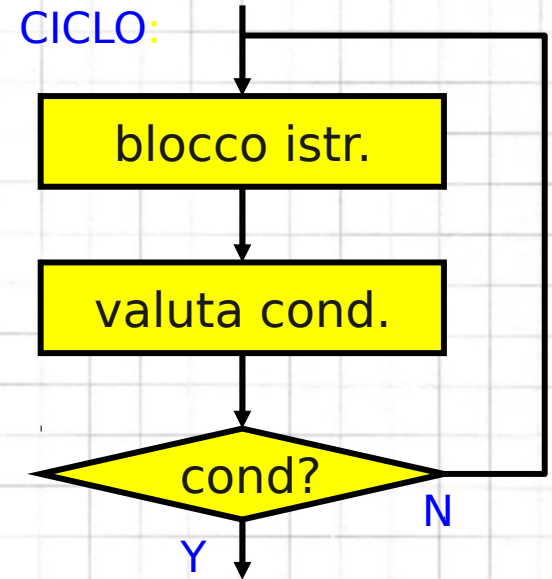
- Struttura di ripetizione sbloccata dal venire meno di una condizione
 - Che causa l'uscita dal loop
- Non richiede registri di appoggio
 - L'operazione che valuta la condizione deve essere eseguita a ogni iterazione



```
TEST: lsr    R16    ;pone R16(0) in C
      brcc   OLTRE ;C=1 per reiterare, C=0 esce
      adc   R1,R2  ;blocco di istruzioni
      asr   R2     ;(qui a caso...)
      com   R1
      rjmp  TEST   ;ripeti ancora una volta
OLTRE:nop                ;il programma prosegue
```

Ripeti finché ...

- Struttura di attesa della condizione
 - Quando si verifica, si esce dal loop
 - Il blocco è eseguito almeno una volta
- Non richiede registri di appoggio
 - L'operazione che valuta la condizione deve essere eseguita a ogni iterazione



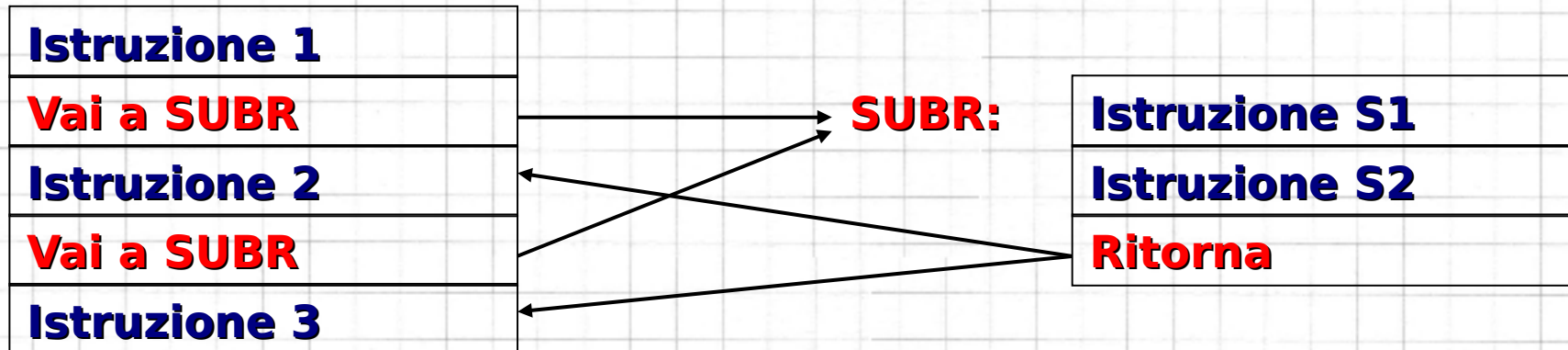
```
CICLO: sts    120,R2 ;blocco di istruzioni
      add    R2,R1  ;(qui a caso...)
      neg    R3
      cpse   R1,R3  ;skip se Z=1 (è la cond.)
      rjmp   CICLO ;reitera
      nop                    ;il programma prosegue
```

Ripetere, ma con dati diversi

- La soluzione banale: le “macro”
 - Riscrivere più volte le stesse istruzioni
 - Insegno queste istruzioni all’assemblatore
 - Le raggruppo e le definisco con un identificatore
 - Le richiamo in blocco usando questo nome
 - La direttiva .MACRO
 - Racchiude le istruzioni da copiare
 - Accetta la definizione di argomenti
- Svantaggio
 - Servono solo ad aiutare il programmatore
 - Non hanno alcun effetto sulla struttura reale del programma

Oltre le macro

- Scrivere una sola volta le istruzioni
 - Inviare il processore a eseguirle e tornare al punto di partenza alla fine
- Struttura di memoria particolare
 - Per tenere memoria del punto in cui tornare



Realizzare un sottoprogramma

- Chiama il sottoprogramma (RCALL)
 - Equivale a eseguire:
 - PUSH PC_H PUSH PC_L
 - RJMP SUBR
- Ritorna dopo il punto di chiamata (RET)
 - Esegue
 - POP PC_L POP PC_H
- Gestione dei dati su cui opera la subroutine
 - Il programmatore deve collocare i dati nelle locazioni in cui la subroutine se le aspetta

Avvertenze sull'uso dello stack

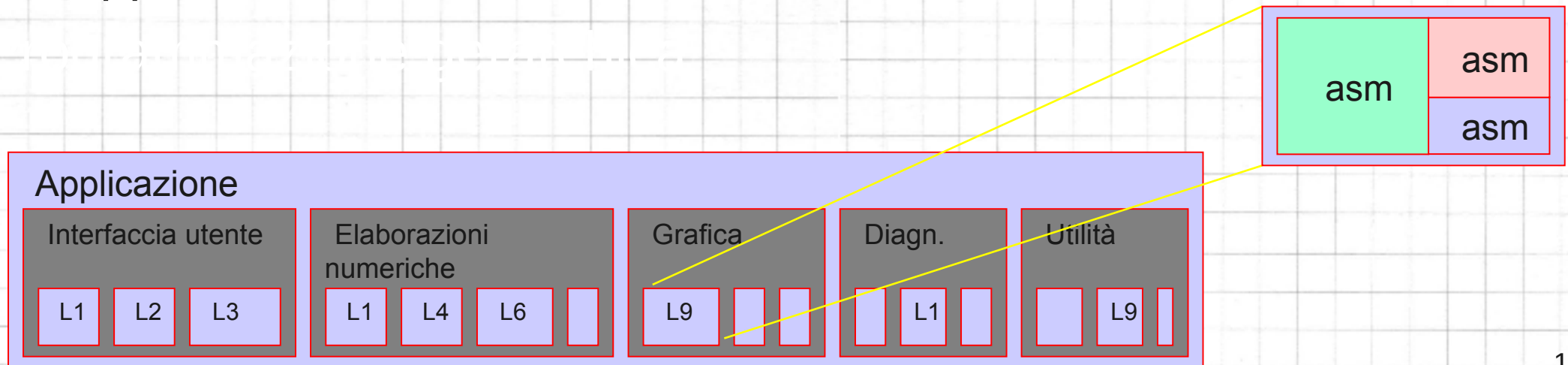
- La chiamata di sottoprogrammi aumenta le dimensioni dello stack
 - Se lo stack si satura si ha “stack overflow”
 - La cima dello stack interagisce con lo spazio riservato ad altri dati
 - Si perdono dati con effetti imprevedibili
- L'uso contemporaneo di sottoprogrammi e memorizzazione nello stack (PUSH e POP) richiede cautela
 - Scritture e letture nello stack devono essere bilanciate
 - Una POP senza PUSH causa la perdita di dati

Uso dei sottoprogrammi (1)

- Realizzazione di procedure e funzioni di uso comune
 - Risparmio di codice e facilità di sviluppo
 - Meccanismi standard di passaggio dei parametri
 - Disponibilità di “librerie”
- Possibilità di realizzare sottoprogrammi in grado di “annidarsi”, grazie allo stack
 - Maggiore flessibilità di uso
 - Possibilità di ricorsione

Uso dei sottoprogrammi (2)

- Partizionamento dei problemi in diversi livelli di astrazione
 - Operazioni semplici di base
 - Funzioni di media complessità
 - Programmi complessi che gestiscono un intero aspetto di una applicazione
 - Applicazione



Interruzioni asincrone

- Il flusso del programma è poco flessibile
 - Non può tenere conto di eventi esterni non prevedibili a priori
 - Si può ottenere una reazione solo controllando periodicamente il verificarsi di un evento (*polling*)
 - Può passare troppo tempo prima di reagire
 - Si può perdere troppo tempo per controllare eventi rari
- Soluzione: “interrupt”
 - Possibilità di interrompere in qualunque momento il flusso di esecuzione normale
 - Ed eventualmente fare dell’altro per poi ripartire da dove ci si era fermati

Tipi di interruzione

- Reset (*“riparti da capo...”*)
 - La macchina riparte come dopo essere stata accesa
 - La richiesta non è eludibile
- Interrupt (*“ho bisogno di te...”*)
 - C'è un evento che richiama l'interesse del processore
 - Il processore è programmato per ascoltare o ignorare la richiesta
 - Dopo aver preso i provvedimenti del caso, il processore riparte da dove si era fermato

Cause di reset

- Assenza o disturbi di alimentazione
 - Un valore scorretto dell'alimentazione potrebbe causare comportamenti imprevedibili
- Attivazione di un apposito segnale esterno
 - Decisione dell'utente per riportare la macchina in una situazione nota
- Azione di meccanismi anti-blocco
 - Sistemi automatici che cercano di evitare situazioni di stallo, non previste dal programmatore (quando “si pianta...”)

Cause di interruzione

- Transizioni di segnali esterni
 - Richieste di attenzione da parte di circuiti esterni
- Richiesta di intervento delle periferiche
 - Interfacce di comunicazione
 - Contatori e timer
 - Contano eventi o misurano intervalli di tempo
 - Comparatori e convertitori
 - Misurano grandezze fisiche (tensioni) esterne

La routine di servizio

- Se l'interruzione è abilitata:
 - Vengono disabilitate interruzioni ulteriori
 - Viene eseguita una RCALL implicita a un indirizzo predefinito (vettore di interrupt)
 - Possono esistere diversi indirizzi in funzione di quale evento ha chiesto l'interruzione
 - Vengono eseguite le istruzioni a partire dal vettore di interrupt sino all'istruzione RETI
 - Riabilita la possibilità di interruzione
- Se non è abilitata:
 - Un flag segnala che l'evento è in corso
 - Il programma procede indisturbato