Esempi Assembly

Il ciclo a controllo in coda

ripeti istruzioni finché condizione	*in assembly:* inizio_ciclo: istruzioni Jcondizione inizio_ciclo

```
MOV AX, 0000h
 inizio_ciclo:
   INC AX
   CMP AX, 000Ah      ;confronta AX e il valore 0Ah (10d)
 JNE inizio_ciclo     ;salta all'inizio (e ripete il ciclo) se diverso
```

NB: problema se il valore di inizializzazione è uguale a quellodi confronto

```
MOV AX, 000Ah  ← problema
 inizio_ciclo:
   INC AX
   CMP AX, 000Ah    ;confronta AX e il valore 0Ah (10d)
 JNE inizio_ciclo     ;salta all'inizio (e ripete il ciclo) se diverso
```

Questo spezzone di codice dovrebbe controllore se AX = 10d, e in caso contrario incrementare AX. In caso favorevole uscire dal ciclo. Vediamo però che AX vale già 10d, tuttavia tale registro viene comunque incrementato (alla fine varrà 000Bh o 11d). Inoltre, in questo particolare programma, il ciclo non finirà mai: AX varrà 11, poi 12, poi 13 e non diventerà mai uguale a 10. Sarebbe buona norma, nelle condizioni, evitare di esprimere un'uguaglianza:

```
MOV AX, 000Ah
  inizio_ciclo:
    INC AX
    CMP AX, 000Ah
  JB inizio_ciclo            ; salta se minore (invece di salta se non uguale)
```

In questo modo abbiamo risolto il problema del ciclo infinito. Tuttavia, a causa del fatto che la sequenza viene eseguita almeno una volta,si ha un incremento non voluto; in genere si evita il ciclo a controllo in coda e si utilizza invece quello a ***controllo in testa****.*

Il ciclo a controllo in testa

mentre condizione istruzioni fine ciclo	*in assembly:* inizio_ciclo: Jcondizione fine_ciclo sequenza JMP inizio_ciclo fine_ciclo

```
inizio_ciclo:
   CMP AX,0Ah            ;confronta AX con 10d
   JE fine_ciclo         ;non esegue il ciclo se uguale
       INC AX            ;incrementa AX
   JMP inizio_ciclo      ;salta (cioè esegue il ciclo)se diverso
 fine_ciclo:
```

La differenza tra questa struttura e quella a controllo in coda sta nel fatto che se la condizione è inizialmente verificata, la sequenza di istruzioni non viene eseguita nemmeno una volta.

Il ciclo a contatore

ripeti per N volte
  sequenza
fine ciclo
	in assembly:
	
	CONTATORE = N
	ripeti
	  sequenza
	  decrementa CONTATORE
	finché CONTATORE = 0

*Come contatore si usa di solito il **registro CX** (registro contatore, appunto), perché esiste un'istruzione che esegue le ultime due istruzioni automaticamente: l'istruzione LOOP: decrementa CX e, se CX non è 0, salta all'etichetta specificata.* Grazie all'istruzione LOOP diventa semplice scrivere un ciclo a contatore in assembly:

```
MOV CX, <N>             ; dove N è il numero di ripetizioni da eseguire
 inizio_ciclo:
   sequenza
 LOOP inizio_ciclo
```

AMBIENTE JASMIN

```
; uso istruzione LOOP
data:
dd 10
MOV EBX, data          ; puntatore
MOV ECX, [data]        ; contatore registro C
salta:
  MOV AL, [EBX]
  INC EBX
                       ;  l'istruzione LOOP: decrementa CX e, se CX non è  0, salta all'etichetta specificata
  LOOP salta
```

Dall'URL http://wwwi10.lrr.in.tum.de/~jasmin/commands.html

LOOP-Loop while CX is not zero

Usage: LOOP label

Arguments: label: label that points to the beginning of the iteration

Effects: Decrements CX (not modifying any flags) and jumps to label if CX is not zero

Flags: none

Misc: Create a loop of N iterations by copying N to CX (MOV CX,N), adding a label to the beginning of the block of operations to be repeated and a LOOP statement to its end

Set dell'8088/86

LOOP gira mentre CX non è zero
LOOPE gira mentre è uguale
LOOPNE gira mentre non è uguale
LOOPZ gira mentre è zero
LOOPNZ gira mentre non è zero

LOOP - Salta incondizionatamente CX volte.
LOOPD - Salta incondizionatamente ECX volte.

se **ECX**=00000000 *prosegue con l'istruzione successiva*.

se **ECX** è diverso da 00000000 ***salta*** all'indirizzo suggerito dall'etichetta

non appartiene al Set dell'8088/86; si usa solo con **80386/486**.

LOOPE - Salta CX volte se uguale

LOOPED - Salta ECX volte se uguale

LOOPEW - Salta CX volte se uguale

LOOPNED - Salta ECX volte se non è uguale

LOOPNEW - Salta CX volte se non è uguale

LOOPNZD - Salta ECX volte se non è zero

LOOPNZW - Salta CX volte se non è zero

LOOPW - Salta incondizionatamente CX volte

LOOPZD - Salta ECX volte se è zero

LOOPZW - Salta CX volte se è zero

non appartengono al Set dell'8088/86; si usa solo con **80386/486**.

LOOPNE - Salta CX volte se non uguale.

LOOPNZ - Salta CX volte se non è zero.

LOOPZ - Salta CX volte se zero