Esercitazioni di Fondamenti di Informatica - Lez. 2
02/10/2020

Pseudocodifica e Linguaggio di Von Neumann

Tabella riassuntiva di istruzioni della macchina di von Newmann

READ	Dal nastro di input ad ACC
WRITE	Da ACC a nastro di output
LOAD XXX	il contenuto della cella xxx viene trasferito nell'accumulatore
LOAD= X	Inizializza il contenuto dell'ACC
STORE XXX	il contenuto dell'accumulatore viene trasferito nella cella xxx
ADD X	: [ACC] + [X] ---> [ACC] somma il contenuto della cella X allo stesso modo ADD, SUB, MULT, DIV (divisione intera)
ADD= 13	Viene sommato il **valore 13** al contenuto dell'accumulatore
BR XX	la macchina salta all'istruzione xxesima
BEQ XX	la macchina salta ad eseguire l'istruzione xx-esima, se il contenuto dell'accumulatore è 0, altrimenti prosegue con l'istruzione successiva
BGE, BG,BL	altre forme di salto condizionato dipendenti dal contenuto dell'accumulatore
LOAD@ XXX	viene trasferito in accumulatore il contenuto della cella il cui indirizzo è a sua volta contenuto nella cella xxx

Esercizi di pseudo codifica

Scrivere un programma in pseudo codifica che soddisfi le richieste dell'esercizio. Quando richiesto proporre
una versione dello stesso programma nel linguaggio della macchina di Von Neumann.
Nella soluzione il testo preceduto dal simbolo **//** è un commento al codice.

1. **Leggi due numeri e stampa la differenza del maggiore con il minore**

```
0  main ( ){
1   scanf ( numero1 ) // leggi  il primo numero
2   scanf ( numero2 ) // leggi  il secondo numero
3   // controlla se il primo numero e ' maggiore   del secondo
4   i f ( numero1>numero2 ) {
5         differenza = numero1- numero2
6   }
7   else {
8           differenza = numero2 - numero1
9           }
10   printf ( differenza ) // Stampa la differenza
11  }
```

2. **Leggere una sequenza di numeri fino a che si incontra il valore zero e stamparne il massimo.**

Proporre anche la versione in linguaggio di macchina di Von Neumann.

```
0 main ( ) {
1         scanf ( numero1 ) // leggi primo numero
2         massimo = numero1 // inizializza  il massimo con i l primo
3                                   //numero letto
4         while ( numero1 != 0) {
5                 // controlla  se il nuovo numero letto e '
6                 // maggiore del massimo corrente
7           if ( numero1>massimo ) {
8                   massimo = numero1          // aggiorna il massimo
9           }
10                                                  / leggo  il prossimo numero
11                // non ho bisogno di memorizzare tutti i numeri letti
12          scanf ( numero1 )
13        }
14 // stampa il risultato
15       printf ( " il  massimo numero letto è  : " )
16       printf (massimo )
17 }
```

Cella 101 - memorizza il valore massimo M
Cella 102 - memorizza il valore letto da input

```
0  READ            // leggo un numero A
1  STORE 101       // memorizza A come massimo M
2  STORE 102       // memorizza anche in cella temporanea
3  BEQ 10          // se A = 0 vai a stampare M
4   SUB 101        // sottrai M da A
5   BLE 8          // se (A-M)<0    M ancora il massimo
6   LOAD 102       // carica A
7   STORE 101      // M = A
8   READ           // leggi nuovo numero A
9   BR 2            // ripeti il ciclo
10  LOAD 101       // carica M
11  WRITE          // stampa M
12 END
```

3.Leggere due numeri e stampare la differenza del maggiore con il minore

Pseudocodice	**algoritmo**
0 main ( ) { 1 scanf (numero1) 2 scanf (numero2) 3 4 if (numero1>numero2) { 5 differenza=numero1-numero2 6 } 7 else { 8 differenza=numero2-numero1 9 } 9 printf(differenza) 10 }	// leggi il primo numero // leggi il secondo numero // se il primo numero e ' maggiore del secondo **//** calcola differenza e conserva il risultato // altrimenti // calcola la differenza tra il secondo e il primo // stampa il valore calcolato

Codifica Von Newmann

Cella 101 - memorizza il primo valore letto
Cella 102 - memorizza il secondo valore letto

```
0  READ
1 STORE 101     // memorizzo il primo
2 READ
3 STORE 102          // memorizzo il secondo
4 SUB 101       // in ACC ho il secondo numero letto esegue ACC- 101
5 BGE 8              // se il risultato è maggiore di 0 salto a scriverlo
6 LOAD 101           // se non è >= 0 carico il primo in ACC
7 SUB 102            // e sottraggo il  secondo
8 WRITE              // scrivo il res che si trova in  ACC
9. END
```

4. Leggere base ed esponente ,calcolare la potenza e stampare il risultato.

Controllare che la base sia positiva e l'esponente >= 0 .
Ripetere la lettura di base ed esponente fino a che le condizioni non sono rispettate.
Scrivere anche la versione del codice in linguaggio della macchina di Von Neumann.

```
0 main ( ) {
1      printf ( " Inserisci base " )
2      scanf ( base )
3      while ( base <= 0){
4         printf( "la base deve essere > 0" )
5         scanf( base )
6      }
7      printf("inserisci esponente")
8      scanf(esponente)
9      while ( esponente < 0){
10        printf("deve essere >= 0" )
11        scanf( esponente )
12     }
13   potenza = 1
14   while ( esponente > 0){
15    potenza = potenza * base
16    esponente = esponente - 1
17   }
18  printf ( potenza )
19 }
```

```
// leggi la base
// controlla se la base  è positiva

// continua a leggere  un numero fino a che non è positivo

// continua a leggere  fino a che non è positivo



// inizializzo risultato della potenza
// l'esp deve essere magg di 0
// aggiorno la potenza moltiplicandola un'altra volta per la  base
// decremento esponente
// stampa il risultato
```

Codifica Von Newmann

Cella 101 - memorizza la base
Cella 102 - memorizza il valore di esponente
Cella 103 - contiene il valore calcolato

Notiamo in evidenza i cicli

```
 0 READ          // leggo la base B
 1 BLE 0         // controllo se minore di 0
 2 STORE 101     // memorizzo la base
 3 READ          // leggi
 4 BL 3          // controllo se minore di 0
 5 STORE 102     // memorizzo esponente
 6 LOAD=1        // acc = 1
 7 STORE 103     // memorizzo il valore di ACC
 8 LOAD 102      // leggo esponente
  9 BEQ 16       // se =0 stampo il risultato
  10 SUB=1       // altrimenti sottraggo 1
  11 STORE 102   //memorizzo nuovo esp
  12 LOAD 103    //
  13 MULT 101    //moltiplico il valore parziale x base
  14 STORE 103   // salvo il parziale
 15 BR 8         // ripeto controllando esp
 16 LOAD 103     // a fine ciclo leggo il risultato
 17 WRITE        // e lo stampo
 18 END
```

5.Leggere una sequenza di numeri fino a che si legge uno 0 e fare la media aritmetica dei soli numeri positivi.

0 main(){	
1. somma = 0	// inizializza la somma
2. contatore =0	// inizializza contatore num letti
3. scanf(numero1)	// leggi un nuovo numero
4. **while** (numero1 !=0){	// finchè il numero letto è diverso da 0
5. **if**(numero1 >0)	// se leggo un numero positivo
6. {	
7. somma=somma+ numero1	
8. contatore = contatore +1	// aggiorno somma e contatore
9. }	// inizializza media finale
10 scanf(numero1)	//se ho letto un <0 i valori sono invariati
11 }	**// leggo nuovo num**
12 media =0	// quando ho letto uno 0
13 **if** (contatore >0){	// inizializza media
14 media = somma/ contatore	// posso fare la divisione solo se contatore !=0
15 }	
16 printf(" la media dei numero positivi letti è: ")	
17 printf (media)	
18 }	

6. Data una sequenza di K numeri terminati dallo 0 ($n_0, n_1, n_2, \ldots, n_{k-1}, 0$), calcolare

$$\sum_{i=0}^{\frac{K-1}{2}} n_i * n_{k-1-i}$$ (k-1)/2 preso intero per difetto

Per esempio se ho 5 valori, $k = 5$, dovrò calcolare $n_0 \cdot n_4 + n_1 \cdot n_3 + n_2 \cdot n_2$.

In alternativa se ho 4 valori, $k = 4$, dovrò calcolare $n_0 \cdot n_3 + n_1 \cdot n_2$

```
0   main()                          // devo memorizzare valori letti e quanti valori   letti
1                                    // uso array numeri [ num_valori ]
2   num_valori = 0                  // inizializzo contatore
3   scanf ( numero1 )
4   while ( numero1 !=0 ){          // lo uso quando ho una condizione se che si
                                       ripete
5         numeri [ num_valori ] = numero1        // salva numero in array
6         num_ valori = num_valori + 1
8         scanf ( numero1 )
9  }                          // termina quando legge 0 e non lo memorizza
10  somma = 0
11  i = 0
12  while ( i <= (num_valori -1)/2){
13        somma = somma + numeri [i ]* numeri [ num_valori-1-i ]
14        i = i + 1
15  }
16  printf( "La  somma  finale  e '  : " )
17  printf (somma)
18 } // fine codice
```

7. ESAME 30/01/2018.

Leggere una sequenza di numeri interi positivi terminata da 1.

Letta l'intera sequenza, il programma per ogni numero letto deve stampare la differenza rispetto a 100.

Le stampe devono partire dall'ultimo numero letto e seguire l'ordine inverso rispetto alla lettura.
Ad esempio, se il programma leggesse: 20, 12 e 37, dovrebbe stampare: 63, 88, e 80.

Alla fine, il programma deve anche stampare il numero letto, non stampato, piu` piccolo e quello piu` grande (12 e 37 nel caso precedente).

Proporre la versione in linguaggio della macchina di Von Neumann. E` indispensabile commentare il programma in modo adeguato.

```
0 main() {
1       i = 0
2       scanf( numeri_letti[0] )        //uso una struttura non so quanti num leggo
3       numero max = numeri _letti[ i ]
4       numero min = numeri _letti [ i ]
5
6       while (numero _letti [ i ] !=1 ) {
7          i f ( numeri _letti [ i ]>numero_max ){.          // individuo il max
8                  numeroi_max  = numeri _letti [ i ]
9          }
10         i f ( numeri _letti [ i ] < numero_min ){         // individuo il min
11                 numero_min = numeri _letti [ i ]
12         }
13         i = i + 1
14         scanf( numeri _letti [ i ] )
15      }.                                    // fine while
16      tot_numeri _letti = i
17      while ( i !=0){
18         printf( numeri _letti [i -1]-100)
19         i = i - 1
20      }
21      if ( tot_numeri _letti !=0 ){
22                 printf( "Massimo : " )
23                 printf( numero max )
24                 printf( "Minimo : " )
25                 printf( numero min )
26       }
27 }   // fine codice
```

Codifica Von Newmann

Cella 101 - un indirizzo di cella
Cella 102 - il valore massimo
Cella 103 - memorizza il valore minimo
Cella 104 - un valore temporaneo

```
0       LOAD= 500       //      carica valore prima cella indirizzamento indiretto
1       STORE 101       //      salva il valore in 101
2       READ            //      leggi     numero A
3       STORE 103       //      inizializza         minimo
4       STORE 102       //      inizializzo max
5       SUB=1           //      sottrai  uno
6       BEQ 23           //      s e       A-1 = 0 → vai a stampare
7       ADD=1           //    A-1+1 = A
```

```
8     STORE@ 101          //      salvo A nella cella puntataa da 101
9     SUB    102          // A-max
10    BL     15           // A-max < 0     → salto
11    LOAD@ 101           //      carico A
12    STORE 102           //      salvo   A come massimo
13    LOAD@ 101           //      carico  A
14    SUB    103          //      A -min
15    BG  18              //      A - min > 0      → s a l t o
16    LOAD@ 101   //      carico  A
17    STORE 103   //      salvo   A
18    LOAD 101    //      carico  un  indirizzo in cella
19    ADD=1       //      incremento indirizzamento indiretto
20    STORE 101   //      salvo indirizzo aggiornato
21    READ        //      leggo nuovo     numero
22    BR 5        //      torno a check A - 1 == 0
23    LOAD 101    //      carico indirizzo successivo a quello dell'ultimo num letto
24    SUB=1       //      vi sottraggo 1
25    STORE 101   //      salvo l'indirizzo nella cella
26    SUB=500     //      sottraggo indirizzo base
27    BL 34       //      se <  0  stampo max e min
28    LOAD@ 101   //      carico il valore corrente
29    STORE 104   //      lo  salvo temporaneamente
30    LOAD=100    //      vi  sottraggo 1
31    SUB 104     //      vi sottraggo 104
32    WRITE       //      scrivo su stdout
33    BR 23       //      leggo il numero precedente
34    LOAD 102    //  carico max
35    WRITE       //  stampo max
36    LOAD 103    //  carico  min
37    WRITE       //  stampo  min
38    END
```

8. ESAME 10/09/2012.

Leggere una sequenza di numeri interi, terminata dallo 0

Stampare il valore più grande e più piccolo tra i numeri letti e la media tra i tre valori più grandi.

Commentare opportunamente il programma per spiegare i passi compiuti.

```
0   main ( void){
1     numeri_max1=0;              // inizializzo i valori per i massimi
2     numeri_max2=0;
3     numeri_max3=0;
4     scanf( num_letto ) ;
5     num_min = num_letto;        // inizializzo il valore per il min
6    while ( num_letto!=0){               // controllo se input diverso da 0
7      if ( num_letto< num_min ){
8            num_min  = num_letto;      // memorizza il minimo
9     }
10    if ( num_letto > numeri_max1 ){
11           numeri_max3 =numeri_max2       // memorizza i tre num maggiori
12            numeri_max2 = numeri_max1     // opportunamente
13            numeri_max1= num_letto
14    }
15    elseif  ( num_letto > numeri_max2){     // memorizza il maggiore medio
16           numeri_max3 = numeri_max2
17           numeri_max2= num_letto
18    }
19    elseif ( num_letto> numeri_max3){
20           numeri max 3 = nume ro le t to
21    }
22    scanf( “%d", &num_letto) ;
23  }                                  // fine while
24   printf ( "numero minimo :”)
25   printf( num_min )
26   printf ( "numero massimo” )
33   printf ( numero max )
34   printf ( "media numeri massimi” )
35   printf( (numeri_max1+numeri_max2+numeri_max3)/3 )
36 }
```