Esercitazione istruzioni MIPS

12 ottobre 2011

Es1

- Dato un vettore A di 100 interi memorizzati a partire dalla locazione 2000, scrivere un programma in assembler MIPS che copii in un vettore B (memorizzato a partire dalla locazione 3000) gli elementi di A che sono minori di 15 ed in un vettore C (memorizzato a partire dalla locazione 4000) tutti gli altri elementi di A (cioè quelli maggiori o uguali a 15).
- Si richiede (1) la descrizione dell'algoritmo usato (in pseudocodice), (2) la descrizione dell'uso dei registri (con la
- indicazione di eventuali valori già memorizzati) e (3) il programma in assembler commentato linea per linea. Non si può assumere la inizializzazione di nessun registro.

Es2

- Sia dato un vettore A di 20 elementi, memorizzato a partire dalla locazione 1000, che contiene i voti riportati agli esami da uno studente. Si può assumere che i voti sono tutti valori positivi tra 18 e 30. Scrivere un programma in assembler MIPS che scriva un vettore B di 20 elementi, memorizzato a partire dalla locazione 2000 in modo che B[i] = 0 se A[i] è compreso tra 18 e 24, B[i] = 1 se A[i] è compreso tra 25 e 27 e B[i]=2 se A[i] è compreso tra 28 e 30.
- Si richiede (1) la descrizione dell'algoritmo usato (in pseudocodice), (2) la descrizione dell'uso dei registri (con la indicazione di eventuali valori già memorizzati) e (3) il programma in assembler commentato linea per linea. Non si può assumere la inizializzazione di nessun registro.

Esercizi facoltativi (1)

- Scrivere un programma MIPS per ordinare i dati contenuti nelle diverse locazioni di memoria mostrate nella tabella riportata sotto, in modo tale da inserire il numero più piccolo nella di memoria all'indirizzo più basso. Si utilizzi il minor numero possibile di istruzioni MIPS e si assuma che l'indirizzo base di *vettore* sia contenuto nel registro $s6.

	Indirizzo	Dato
a.	12	1
	8	6
	4	4
	0	2
	Indirizzo	Dato
b.	16	1
	12	2
	8	3
	4	4
	0	5

Esercizi facoltativi (2)

- Quante istruzioni MIPS sono necessari per ordinare il vettore riportato sopra? Nel caso in cui non si possa utilizzare il campo immediato delle istruzioni lw e sw, quante istruzioni MIPS sarebbero necessarie?

Esercizi facoltativi (3)

- Tradurre in numeri decimali i numeri esadecimali riportati sotto.

- 0x12345678
- 0xbeadf00d

Es1-Solu in C

```
j = 0;   // inizializzo l'indice di B
k = 0;   // inizializzo l'indice di C
for (i = 0; i < 100; i++)  { // per ogni elemento A[i] del vettore A
      if (A[i] < 15)  {  // se A[i] è minore di 15 allora..
       B[j] = A[i];  //  ... scrivi A[i] nel vettore B
       j = j +1;  //  ... ed incrementa l'indice j
 } else {   // altrimenti
       C[k] = A[i]; // ... scrivi A[i] nel vettore C
       k = k + 1;  // ... ed incrementa l'indice k
 }
}
```

Es1-Solu in Pseudo-codice

```
    j = 0 ;
      k = 0;
      i = 0 ;    // inizializza indice del ciclo
Ciclo: if ( ! (A[i] < 15)) goto ScriviC; // se la condizione non è vera salta
    alla parte else
      B[j] = A[i];   //  ... inserisci A[i] nel vettore B
      j = j +1;    //  ... j indica il primo elemento libero di B
      goto Con t;   //  ... salta all'incremento di i
ScriviC: C[k]=A[i];   // altrimenti .. inserisci A[i] nel vettore C
          k = k +1;    //  ... k indica il primo elemento libero di C
Cont: i = i +1;    // incrementa l'indice
    if (i != 100) goto Ciclo;   // se non abbiamo scorso tutto l'array,
    torna a inizio ciclo
```

Es1-registri usati

$10 indice i

$11 spiazzamento relativo all'indice i

$12 spiazzamento relativo all'indice j

$13 spiazzamento relativo all'indice k

$20 valore letto da A[i]

$21 costante 100

$22 risultato della slt

Es1-Solu assembler MIPS

```
      addi  $21, $0, 100 # costante 100 in $21
        addi  $10, $0, 0 # inizializzazione i a 0
        addi  $11, $0, 0 # inizializzazione spiazzamento di i a 0
        addi  $12, $0, 0 # inizializzazione spiazzamento di j a 0
        addi  $13, $0, 0 # inizializzazione spiazzamento di k a 0
Ciclo: lw $20, 2000($11) # leggo A[i] in $20
        slti $22, $20, 15 # se A[i] non è minore di 15...
        beq $22, $0, ScriviC #  ... vai a scrivere A[i] in C
        sw  $20, 3000($12) # scrivi A[i] in B[j]
        addi  $12, $12, 4 # incrementa lo spiazzamento di j
        j Cont  # salta all'incremento di i
```

Es1-Solu assembler MIPS (cont)

```
ScriviC: sw $20, 4000($13) # scrivi A[i] in C[k]
         addi $13, $13, 4 # incrementa lo spiazzamento di k
 Cont: addi $10, $10, 1 # incrementa l'indice i
       addi  $11, $11, 4 # incrementa lo spiazzamento relativo a i
       bne $10, $21, Ciclo # se non abbiamo scorso tutto l'array, torna a
         inizio ciclo
```

Es2

- Sia dato un vettore A di 20 elementi, memorizzato a partire dalla locazione 1000, che contiene i voti riportati agli esami da uno studente. Si può assumere che i voti sono tutti valori positivi tra 18 e 30. Scrivere un programma in assembler MIPS che scriva un vettore B di 20 elementi, memorizzato a partire dalla locazione 2000 in modo che B[i] = 0 se A[i] è compreso tra 18 e 24, B[i] = 1 se A[i] è compreso tra 25 e 27 e B[i]=2 se A[i] è compreso tra 28 e 30.
- Si richiede (1) la descrizione dell'algoritmo usato (in pseudocodice), (2) la descrizione dell'uso dei registri (con la indicazione di eventuali valori già memorizzati) e (3) il programma in assembler commentato linea per linea. Non si può assumere la inizializzazione di nessun registro.

Es2-Solu in C

```
for (i = 0; i < 20; i++) {  // per ogni elemento del vettore
    if (A[i] < 25)  // se il voto è <= 24 (e si assume che sia ≥ 18)...
            B[i] = 0;  //  .... allora B[i] vale 0
    else if (A[i] < 28)   // altrimenti (quindi è >=25)... se A[i] è ≤ 27...
                        B[i] = 1;  //  .... allora B[i] vale 1
            else   // altrimenti (vuol dire che A[i] >= 28 e si assume che sia
    ≤ 30)
    B[i] = 2;  //  .... allora B[i] vale 2
}
```

Es2-Solu in Pseudo-codice

```
        i = 0:
Ciclo: if (! A[i] < 25) goto Else1;
        B[i] = 0;
        goto Cont;
Else1: if (! A[i] < 28) goto Else2;
        B[i] = 1;
        goto Cont;
Else2: B[i] = 2
Cont: i = i +1
        if (i !=20) goto Ciclo;
```

Es2-registri usati

$10 indice i

$11 spiazzamento dell'indice i

$12 registro che contiene il valore da scrivere in B[i]

$13 registro per A[i]

$14 registro per il confronto (slt)

$15 costante 20

Es2-Solu assembler MIPS (cont)

```
      add $10, $0, $0    # i = 0
        add $11, $0, $0  # spiazzamento di i a 0
        addi $15, $0, 20  # costante 20
Ciclo: lw $13, 1000($11) # carica A[i]
        slti $14, $13, 25  # controlla se A[i] < 25 ...
        beq $14, $0, Else1  # ... se ! A[i] < 25 goto Else1
        addi $12, $0, 0  # scrivi 0 in ...
        sw $12, 2000($11) # ... B[i]
        j Cont   # goto Cont
```

Es2-Solu assembler MIPS (cont)

```
Else1: slti $14, $13, 28  # controlla se A[i] < 28 ...
       beq $14, $0, Else2  # ... se ! A[i] < 28 goto Else2
       addi $12, $0, 1  # scrivi 1 in ...
       sw $12, 2000($11) # ... B[i]
       j Cont   # goto Cont
Else2: addi $12, $0, 2  # scrivi 2 in ...
       sw $12, 2000($11) # ... B[i]
       addi $10, $10, 1  # i = i + 1
       addi $11, $11, 4  # spiazzamento = spiazzamento +4
       bne $10, $15, Loop # if (i != 20) goto Ciclo;
```