Laboratorio di Elementi di Architetture e Sistemi Operativi

Compitino del 22 Maggio 2013

Istruzioni:

- Le soluzioni degli esercizi vanno inviate per email a `davide.bresolin@univr.it`, entro *Lunedì 27 Maggio*.
- Nelle soluzioni è sufficiente inviare il codice del programma assembly.

Attenzione: consegnare solamente lo svolgimento degli esercizi 1 e 2!

Esercizio 0. Scrivere il programma che calcola la somma di dodici numeri visto a lezione e provarne il funzionamento con il simulatore LC-3. Verificare che dopo l'istruzione `HALT` il risultato è ancora presente in memoria nella locazione `SUM`.

Esercizio 1. Scrivere un programma in linguaggio assembly che determina il minimo tra dieci numeri contenuti in memoria a partire dall locazione `NUMBERS` e lo pone in una locazione di memoria dedicata ed identificata dal simbolo `MIN`.

1. Provare il funzionamento del programma con il simulatore LC-3.
2. Verificare che dopo l'istruzione `HALT` il risultato è ancora presente in memoria.

Esercizio 2. Scrivere un programma in linguaggio assembly che converta i caratteri di una stringa in MAIUSCOLO. Si assuma che la stringa si trovi in memoria nella locazione `STRING`. Per risolvere l'esercizio si tenga presente che:

1. per caricare la stringa in memoria si può usare la direttiva `.STRINGZ`;
2. le stringhe sono rappresentate come in C (con lo `0` finale);
3. i caratteri minuscoli hanno codice ASCII compreso tra `a=97` e `z=122` (estremi inclusi);
4. per convertire un carattere da minuscolo a MAIUSCOLO basta sottrarre `32`;

Attenzione: nelle istruzioni i valori costanti devono essere compresi tra `-16` e `15` (5 bit in complemento a due)!

Esercizio 3. Scrivere un programma in linguaggio assembly che calcoli il *Massimo Comun Divisore* tra due numeri positivi contenuti nelle locazioni di memoria `A` e `B` e salvi il risultato nella locazione di memoria `MCD`. Il programma deve utilizzare l'algoritmo di Euclide per calcolare l'MCD:

```
int a, b;
while(a != b) {
  if(a < b) {
    b = b -a;
  } else {
    a = a - b;
  }
}
return a;
```

Elenco delle Istruzioni LC-3

`ADD DR,SR1,SR2`	Somma
`ADD DR,SR1,<valore>`	
`AND DR,SR1,SR2`	And bit-a-bit
`AND DR,SR1,<valore>`	
`NOT DR,SR`	Negazione bit-a-bit
`LEA DR,<label>`	Load con indirizzamento immediato
`LD  DR,<label>`	Load con indirizzamento PC-relative
`LDI DR,<label>`	Load con indirizzamento indiretto
`LDR DR,BR,<offset>`	Load con indirizzamento base+offset
`ST  SR,<label>`	Store con indirizzamento PC-relative
`STI SR,<label>`	Store con indirizzamento indiretto
`STR SR,BR,<offset>`	Store con indirizzamento base+offset
`BRn <label>`	Salta se negativo
`BRz <label>`	Salta se zero
`BRp <label>`	Salta se positivo
`BRzp <label>`	Salta se zero o positivo
`BRnp <label>`	Salta se negativo o positivo
`BRnz <label>`	Salta se negativo o zero
`BRnzp <label>`	Salta sempre
`JMP BR`	Salto non condizionato
`HALT`	Arresta l'LC-3

Nota: `<valore>` è un intero a 5 bit in complemento a due (da -16 a 15). `<offset>` è un intero a 6 bit in complemento a due (da -32 a 31).