Architettura degli Elaboratori a.a. 2021/22

Appello d'esame 24/01/2023 (ASM)

Operazioni preliminari

1. Aprire il file `AE - Uso del computer in laboratorio per la prova ASM.pdf`, contenente informazioni utili per lo svolgimento di questa prova.
2. Aprire tramite MARS il file `program01.asm` in questa directory
3. Completare le seguenti righe con i propri dati in `program01.asm`

```
########################################
# INSERIRE I PROPRI DATI QUI:
# Nome:
# Cognome:
# Matricola:
########################################
```

Esercizio

Si realizzi in assembly MIPS il seguente programma. Sia data in input da utente una stringa di caratteri binari `buffer`. La sequenza termina quando viene letto `'\n'` (codice ASCII `0xA`). Prima del carattere `'\n'`, essa può contenere solo simboli numerici del codice binario: '`0`' (codice ASCII `0x30`) o '`1`'. La stringa consta al più di 24 caratteri (`'\n'` e `'\0'` esclusi).

Il programma deve trasporre la stringa binaria `buffer` in input in una stringa di numeri ottali (ossia, con caratteri numerici da '`0`' a '`7`') scrivendone i caratteri in memoria ad un dato indirizzo `output`. Qualora il numero di simboli binari in `buffer` non sia un multiplo di `3`, il programma deve scartare gli ultimi simboli in che non compaiono in un terzetto. Ad esempio, se `buffer` è `101101111` il programma produrrà la stringa ottale `557`; se `buffer` è `10110111101` il programma produrrà la stessa stringa ottale, `557`, perché gli ultimi due simboli (`01`) vengono scartati.

Il programma deve stampare a video su tre righe separate 0. la stringa prodotta in `output`, seguita dai due seguenti valori: 1. `0` se non sono scartati caratteri al di fuori di un terzetto di simboli, altrimenti `1`; 2. il numero totale di cifre ottali prodotte.

Se, ad esempio, `buffer` è `101101111` allora dovrà essere scritto a schermo su tre linee separate:

```
557
0
3
```

perché tutti i caratteri compaiono in terzetti e tre cifre ottali sono state prodotte.

Se invece `buffer` è `00110111101` allora dovrà essere scritto a schermo su tre linee separate:

```
157
1
3
```

perché due simboli in coda (`01`) vengono scartati.

Nella sezione dedicata al **text segment**, il programma deve avere nel comparto *main* il caricamento dei dati da input (già fornito nel file `program01.asm`: l'indirizzo base della stringa `buffer` verrà

caricato nel registro `$a0` e l'indirizzo base dell'array di caratteri `output` verrà caricato nel registro `$a2`), la chiamata ad una funzione `codificaOttale` definita di seguito e la stampa a terminale delle occorrenze. La stampa a terminale non deve avvenire all'interno della funzione `codificaOttale` ma nel comparto *main* chiamante o in una funzione dedicata di propria stesura.

La funzione `codificaOttale` accetta come argomento:

- `$a0`: l'indirizzo base della stringa `buffer`;
- `$a2`: l'indirizzo base dell'array di caratteri `output`;

e restituisce come risultato:

- `$v0`: `0` se tutta la stringa di input è stata tradotta; altrimenti, `1`
- `$v1`: numero di caratteri nella stringa ottale prodotta.

Come effetto collaterale, deve scrivere in `output` (cioè dall'indirizzo base `$a2`) la stringa ottale prodotta.

Note: Commentare ogni riga di codice avendo cura di spiegare a cosa servano i registri. Si ricorda che la syscall per la stampa di un intero prevede come argomenti `$v0=1` e `$a0` assegnato con il valore da stampare. La syscall per la stampa di un carattere (necessaria per separare gli output con una `'\n'`) prevede come argomenti `$v0=11` e `$a0` assegnato con il carattere da stampare. La syscall per la stampa di una stringa prevede come argomenti `$v0=4` e `$a0` assegnato con l'indirizzo base della stringa da stampare.

Risultato atteso

Per ogni input al programma va stampato l'output della procedura suddetta seguito da accapo come spiegato precedentemente.

Ad esempio, il file `test-02.in` contiene il seguente input:

```
000001010011100101110
```

Il suo output atteso (all'interno di `test-03.expt`) è:

```
0123456
0
7
```

È possibile studiare i casi di test aprendo i file di input (nel formato `test-`*xy*`.in`, dove *xy* è un numero a due cifre) e output atteso (estensione `test-`*xy*`.expt`), confrontandoli col proprio (estensione `test-`*xy*`.out`).

Verifica di corretta esecuzione dell'esercizio

Per verificare che l'esercizio sia stato completato correttamente eseguire `run.sh` (doppio click sul file dal file manager, oppure esecuzione del comando `./run.sh` da terminale) e visualizzare il risultato aprendo il file `test_results.html`. Per ulteriori informazioni, consultare il file `AE - Uso del computer in laboratorio per la prova ASM.pdf`.

Note

- Non è consentito modificare il *data segment*.
- Il limite massimo di istruzioni eseguibili è `305900`. Oltre quel numero, l'esecuzione viene automaticamente terminata.

- Il file `program01.vuoto.asm` contiene una copia del file `program01.asm` che può essere utile in caso sia necessario ripartire da capo.
- Attenzione a non eseguire loop infiniti con leak della memoria: MARS potrebbe andare in crash e cancellare il file che state scrivendo! Sentitevi liberi di salvare un file di backup prima di eseguire del codice rischioso.

Voto

Una soluzione pienamente funzionante, realizzata seguendo una procedura non ricorsiva, corrisponderà ad un voto pari a 18. Una soluzione pienamente funzionante, realizzata seguendo una procedura ricorsiva, corrisponderà ad un voto pari a 30.

Nel caso in cui sia stato risolto il problema con un programma ricorsivo, si deve avere cura di specificarlo con la seguente nota di commento, subito sotto `## INSERIRE IL CODICE QUI`:

```
#### SOLUZIONE RICORSIVA ####
```