Architetture dei sistemi di elaborazione
Esame del 27.2.2019 - parte II

B2

Durata: 105 minuti.

E' possibile consultare:
- qualunque materiale cartaceo
- i lucidi scaricati dalla pagina del corso sul portale della didattica
- il codice dei laboratori eventualmente caricato fra gli elaborati sul portale della didattica.

Gli studenti sorpresi a comunicare fra loro saranno immediatamente allontanati dal laboratorio.

Si consideri una sequenza di numeri naturali in cui, scelto arbitrariamente il primo numero della sequenza c_0, gli elementi successivi della sequenza sono ottenuti nel modo seguente:

$$c_{i+1} = \begin{cases} \frac{c_i}{2} & se\ c_i \text{ è } pari \\ 3 * c_i + 1 & se\ c_i \text{ è } dispari \end{cases}$$

La sequenza termina quando si ottiene il numero 1.

Esempio 1. Se c_0= 12, la sequenza è: 12, 6, 3, 10, 5, 16, 8, 4, 2, 1. La sequenza contiene 10 elementi.
Esempio 2. Se c_0= 19, la sequenza è: 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. La sequenza contiene 21 elementi.

La congettura di Collatz afferma che, indipendentemente dal valore iniziale c_0, la sequenza termina sempre, ossia si raggiunge sempre 1 passando attraverso un numero finito di elementi. La congettura di Collatz non è mai stata dimostrata teoricamente, però è stata verificata sperimentalmente per tutti i numeri naturali fino a $87 * 2^{60} \approx 10^{21}$.

Si richiede di aprire con Keil il progetto **progetto.uvproj** presente all'interno della cartella **template** e scrivere subroutine **debuggate e funzionanti** che rispondano alle 3 specifiche seguenti.
Nota 1: non si deve modificare il codice all'interno di `Reset_Handler` che chiama le subroutine. E' richiesto solamente di implementare le subroutine. Si consiglia inoltre di non inserire breakpoint nelle istruzioni di `Reset_Handler`, ma di metterli all'inizio delle subroutine implementate.
Nota 2: le specifiche devono essere svolte in ordine. Si può passare alla specifica 2 solamente dopo aver verificato che la soluzione alla specifica 1 funzioni correttamente. Lo stesso per la specifica 3.

Specifica 1 (8 punti). Scrivere in Assembly ARM una subroutine `iterativeCollatz` che riceve in input un numero naturale, tramite un ciclo calcola la sequenza di Collatz, e restituisce il numero di elementi che compongono la sequenza.
La subroutine deve essere conforme allo standard AAPCS (ARM Architecture Procedure Call Standard), in particolare per quanto riguarda il passaggio del parametro in input/output e il salvataggio dei registri.

Esempio 1: se il parametro in input è 12, il valore restituito è 10.
Esempio 2: se il parametro in input è 19, il valore restituito è 21.

Nota: Siccome l'unica operazione di divisione consiste nel calcolare la metà di un numero, non si possono usare le istruzioni `UDIV` e `SDIV`.

Specifica 2 (6 punti). Scrivere in Assembly ARM una subroutine `recursiveCollatz` che riceve in input due parametri:

P1: un numero naturale corrispondente all'elemento c_i della sequenza
P2: il numero *i*

La subroutine `recursiveCollatz` modifica i due parametri in input svolgendo le seguenti operazioni:

- incrementa di 1 il secondo parametro: $P2_{new} = P2 + 1$
- se il primo parametro è uguale a 1, la subroutine pone $P1_{new} = 1$ e poi termina (ritorna al programma chiamante)
- altrimenti, la subroutine calcola l'elemento c_{i+1} della sequenza, pone $P1_{new} = c_{i+1}$ e poi richiama se stessa passando come parametri $P1_{new}$ e $P2_{new}$.

La subroutine deve essere conforme allo standard AAPCS (ARM Architecture Procedure Call Standard), in particolare per quanto riguarda il passaggio dei parametri in input/output e il salvataggio dei registri.

Esempio 1: se i parametri in input sono P1 = 12 e P2 = 0, la subroutine (dopo 9 chiamate ricorsive) restituisce al programma chiamante $P1_{new} = 1$ e $P2_{new} = 10$.
Esempio 1: se i parametri in input sono P1 = 19 e P2 = 0, la subroutine (dopo 20 chiamate ricorsive) restituisce al programma chiamante $P1_{new} = 1$ e $P2_{new} = 21$.

Nota: Siccome l'unica operazione di divisione consiste nel calcolare la metà di un numero, non si possono usare le istruzioni `UDIV` e `SDIV`.

Specifica 3 (4 punti). Scrivere opportune istruzioni / funzioni in C per:

1. avviare il timer 0 della scheda LPC 1768. Il timer non deve scatenare interrupt.
2. alla pressione del tasto INT0:
    - leggere il valore corrente del registro timer counter del timer 0.
    - chiamare la subroutine `iterativeCollatz` passando il valore del timer counter.
    - memorizzare in una variabile il minimo tra 255 e il valore restituito dalla subroutine `iterativeCollatz`.
    - visualizzare la rappresentazione binaria del valore della variabile sui led 4-11 (il led 11 corrisponde al bit meno significativo).