Architetture dei Sistemi   
di Elaborazione

Nome e Cognome\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matricola\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Il codice compila senza errori: sì [ ] no [ ]

Ho provato il progetto in emulazione: sì [ ] no [ ]

Ho provato il progetto su board: sì [ ] no [ ]

L’ambiente di debug è stato utilizzato : sì [ ] no [ ]

Desidero ritirarmi [ ]

Sistemi basati su ARM T1 – 14 settembre 2022

Leggere con attenzione:

1. Occorre sviluppare un progetto ARM usando l’IDE KEIL µVision.
2. Effettuare login su propria area al LABINF ed usare il software disponibile per editare, compilare e debuggare il codice.
3. Utilizzare l’area desktop sul computer del LABINF per creare il vostro progetto.
4. Utilizzare la scheda LANDTIGER o l’emulatore con tutte le non-idealità abilitate per debuggare il progetto
5. Sono inibiti tutti gli accessi ad internet.
6. Si possono utilizzare progetti esistenti, prelevati dalla propria chiavetta USB, ed è possibile consultare materiale cartaceo.
7. Entro l’orario di consegna, occorre finalizzare il salvataggio di tutti i file (valido anche per la parte di modern architecture) e **copiarli nella propria area personale Z:/ all’interno della cartella che contiene le tracce**. Le consegne in ritardo (con file salvati oltre l’orario massimo di consegna) non vengono considerate valide e conducono in ogni caso all’insufficienza.
8. In caso non sia possibile compilare con successo il progetto consegnato, la prova sarà considerata insufficiente. Si richiede di predisporre l’ambiente di debug con le watch che permettono di seguire il flusso del programma.

Esercizio 1 (max 30 punti)

Sviluppare le seguenti funzionalità per la scheda LANDTIGER ed il system-on-chip LPC1768.

1. All’avvio del sistema, i TIMER1 e TIMER3 sono entrambi programmati in modo ciclico con un periodo iniziale di 50 e 31 µs (microsecondi) rispettivamente, e con interrupt disabilitato.
2. Ad intervalli regolari, con una frequenza di aggiornamento di 2 Hz (la nuova differenza segue la precedente dopo 500 millisecondi), gli 8 LED riportano la differenza dei valori generati da TIMER1 e TIMER3, espressa in complemento a 2.
3. Tutti valori delle differenze riportate sui LED sono inoltre memorizzati in un buffer circolare di N elementi da 32 bit e con N = 128; l’indice del buffer, inizialmente all’indice 0, viene spostato di una posizione dopo ogni acquisizione fino all’indice N-1; se il buffer è saturo si riparte da indice 0, sovrascrivendo quanto memorizzato in precedenza.
4. In qualsiasi momento, mediante la pressione del pulsante INT0, è possibile sospendere il comportamento del sistema e lanciare la seguente funzione scritta in linguaggio ASSEMBLER. La funzione analizza completamente (dalla posizione 0 alla posizione N-1) il buffer a partire da indice 0 e restituisce la differenza tra il numero di valori positivi e negativi memorizzati all’interno del buffer:

char DIFF\_POS\_NEG(int buffer[], unsigned int size, unsigned int \*overflow);  
- dove size indica la dimensione del buffer circolare;

- la funzione restituisce la differenza (valore con segno, espresso in CA2) tra il numero di valori positivi e negativi memorizzati nel buffer (se ci sono valori pari a 0, questi devono essere considerati nel calcolo);

- la variabile overflow viene infine manipolata dalla funzione assembler; all’uscita dalla funzione, la variabile overflow deve avere valore 0 se non si è verificato overflow, oppure 1 in caso si sia verificato overflow.

Il valore restituito deve essere visualizzato usando i LED fino alla pressione di KEY1. In caso di overflow durante l’esecuzione della funzione ASM, i LED mostreranno il risultato blinkando con un periodo di 800ms (400ms on – 400ms off).

1. Una volta terminato il processo di visualizzazione del risultato (alla pressione di KEY1), durante il quale non viene memorizzato nulla nel buffer, il processo riprende da 2), senza ripulire il buffer ma continuando dall’ultimo indice raggiunto prima della pressione di INT0.