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 –8 febbraio 2024 – T3

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. Il sistema deve guidare l’utente affinché esegua una specifica sequenza di pressioni dei pulsanti KEY1 e KEY2. Le indicazioni devono essere date usando i LED 1 e 2. La sequenza di dati è passata a tempo di compilazione tramite l’inizializzazione di un vettore chiamato VETT composto da N elementi di tipo unsigned char su 8 bit. In ciascuna posizione di VETT, se il valore contenuto è 1, allora si riferisce al pulsante KEY1; se 2 allora al pulsante KEY2. Qualunque altro valore deve essere ignorato.

Esempio: nel caso il contenuto del vettore in input sia il seguente,

VETT[N] = [1,1,1,2,0,2,1,3,2]; //con N = 9 in questo esempio

allora la sequenza da introdurre sarà la seguente

3 pressioni di KEY1 🡪 2 pressioni KEY2🡪 1 pressione di KEY1 🡪 1 pressione di KEY2.  
Nota bene: i valori 0 e 3 sono stati ignorati.

1. I LED 1 e 2 visualizzano quale sia il prossimo pulsante da premere:
   * Se LED 1 è acceso e LED 2 spento, KEY1 dovrà essere premuto
   * Se LED 2 è acceso e LED 1 spento, KEY2 dovrà essere premuto
   * Una volta letto il valore premuto dall’utente, tutti i LED lampeggiano con un periodo di 0,25s durante 1s.
   * Una volta terminata la sequenza, entrambi LED 1 e LED 2 saranno resi lampeggianti con un periodo di 0,25 secondi, durante 2 secondi.
   * I valori illegali non devono essere considerate dalla sequenza.
2. Durante le pressioni, il sistema deve inoltre memorizzare i valori inseriti dall’utente, creando il contenuto di un nuovo vettore VETT2 sempre di N elementi su 8 bit:
   * In caso di pressione di KEY1, allora dovrà essere memorizzato il valore 1
   * In caso di pressione di KEY2, allora dovrà essere memorizzato il valore 2
3. Completata la sequenza, dopo una attesa di 3 secondi durante la quale tutti i led lampeggiano con un periodo di 0,125s deve essere lanciata la seguente funzione ASSEMBLER:

unsigned int compare (unsigned char VETT[], unsigned char VETT2[], const unsigned int N);

tale funzione calcola e restituisce il numero di valori differenti tra posizioni “coerenti” di VETT e VETT2 (in altre parole, deve segnalare quanti errori ha fatto l’utente. Il valore calcolato è espresso su 32 bit. La funzione riceve il vettore VETT[] e la costante N originali, quindi deve filtrare i valori illegali in VETT.

1. |Il risultato deve essere visualizzato in binario tramite i led, finché non viene operato il tasto UP del JOYSTICK. A valle di questa azione, il sistema riprende dal punto 1 con VETT2 “svuotato” così come inizialmente.