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 – T2

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)

Si vuole realizzare un sistema in grado di effettuare misure temporali ed elaborarle.

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

1. Il sistema deve contare il numero di pressioni consecutive di KEY1 e KEY2 e popolare un vettore VETT con tali valori:
   * Ad ogni pressione consecutiva di KEY1 o KEY2, una variabile VAR, inizialmente uguale a 0, viene incrementata
   * Ad ogni nuova pressione di KEY2 dopo una sequenza di pressioni di KEY1 (o viceversa), il valore calcolato deve essere memorizzato nella prima posizione libera del vettore VETT di N posizioni (usare N = 1000), dichiarato di tipo unsigned char su 8 bit. I valori vengono memorizzati nel vettore, soltanto se si presenta un cambio di pulsante. Nell’esempio riportato, l’ultima pressione di KEY1 non viene registrata nel vettore.

Esempio:   
  
KEY1 🡪 KEY1 🡪 KEY1 🡪 KEY2 🡪 KEY2 🡪 KEY1 🡪 KEY1 🡪 KEY2 🡪 KEY1  
VETT[N] = [3,2,2,1]

1. I LED devono visualizzare il valore corrente di VAR. Ad ogni nuova pressione del pulsante, il valore aggiornato dovrà essere mostrato lampeggiante con periodo 0,5 secondi per 3 volte, prima di stabilizzarsi (stop lampeggìo dopo 3 volte)
2. Quando viene operato il tasto DOWN del JOYSTICK, deve essere lanciata la seguente funzione ASSEMBLER:

unsigned int compress (unsigned char VETT[], const unsigned int N, char RES[]);

tale funzione riceve come parametri il vettore VETT, N ed un vettore RES di N-1 elementi, inizialmente inutilizzato. Tale vettore RES dovrà contenere, in ogni posizione i, la differenza tra il valore in VETT[i] e VETT[i+1]. Oltre a popolare il vettore RES, la funzione restituisce il numero dei valori strettamente minori di 0 contenuti in RES al termine della funzione. Tale valore restituito è espresso su 32 bit.

1. Il contenuto di RES deve essere visualizzato tramite i led; ciascun valore, partendo dalla posizione RES[0], deve essere visualizzato per una durata di 0,3 secondi sui LED. Durante il periodo di visualizzazione non sarà possibile acquisire altri valori. Una volta finito il tempo di visualizzazione del risultato, il sistema riprende dal punto 1 con VAR, VETT e RES “svuotati” così come inizialmente.