Architetture dei Sistemi di Elaborazione

23 febbraio 2023 – Turno 2

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 [ ]

Leggere con attenzione:

- Occorre sviluppare un progetto ARM usando l'IDE KEIL µVision.
- Effettuare login su propria area al LABINF ed usare il software disponibile per editare, compilare e debuggare il codice.
- Utilizzare l'area desktop sul computer del LABINF per creare il vostro progetto.
- Utilizzare la scheda LANDTIGER o l'emulatore con tutte le non-idealità abilitate per debuggare il progetto
- Sono inibiti tutti gli accessi ad internet.
- Si possono utilizzare progetti esistenti, prelevati dalla propria chiavetta USB, ed è possibile consultare materiale cartaceo.
- 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.
- 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 ed i breakpoint che permettano di seguire il flusso del programma**.

1) Si consideri una variabile **VAR** ed un vettore **VETT** di N elementi, contenenti valori interi senza segno espressi su 16 bit. N è una costante dichiarata a tempo di compilazione, grande a piacimento (si consiglia di debuggare con N = 8 e 9). La variabile ed il vettore saranno dichiarati di tipo **unsigned short int**, con valore iniziale 0.
2) Si devono generare gli elementi del vettore, utilizzando TIMER3 e KEY1:
   - Quando viene premuto KEY1,
     1. se KEY1 è premuto per meno di 300ms, allora il valore di VAR viene incrementato di una unità
     2. se KEY1 viene premuto per un tempo maggiore, il corrente valore di VAR viene moltiplicato per DUE ogni 300ms di pressione
   - Il TIMER3 deve essere programmato in modo da scatenare una interruzione ogni 10 secondi
     1. Allo scadere di tale conteggio, il valore di VAR viene copiato nella prima posizione libera di VETT
     2. Ad ogni nuovo inserimento in VETT, viene lanciata la seguente funzione assembler.
     3. Quando VETT satura, la funzione viene lanciata normalmente e VETT viene "svuotato" immediatamente dopo. Dalla successiva iterazione, VETT sarà riempito da capo.

```
unsigned int concat_sum(unsigned short int VETT[], unsigned int dim, char* alarm);
```

La funzione deve restituire la somma su 32bit di valori creati a partire da VETT concatenando elementi consecutivi per creare numeri su 32 bit. Esempio: VETT[0] concatenato a VETT[1] crea un elemento, VETT[2] concatenato a VETT[3] crea un elemento, in generale VETT[i] concatenato a VETT[i+1] crea un elemento (dove i è un numero pari). I valori creati devono essere sommati e restituita. Qualora il numero di elementi memorizzati in VETT fosse dispari, allora l'elemento mancante sarà sostituito con 0x0000. L'argomento alarm restituisce indirettamente 0 se non si registra nessuna situazione di overflow, -1 qualora si ci sia overflow.

3) Il valore risultato deve essere presentato usando tutti i LED.
   - Se alarm = 0, allora il contenuto dei bit 4÷11 (little endian) del risultato della funzione viene mostrato fisso in valore binario sui LED
   - Se alarm = -1, allora i tutti i led si lampeggeranno con un periodo di 0,5 secondi.

4) L’output deve essere mostrato fino alla successiva esecuzione della funzione assembler. Durante la visualizzazione del risultato, è possibile operare KEY1 per produrre un nuovo elemento di VETT, dopo aver azzerato il valore di VAR.