Architetture dei Sistemi di Elaborazione

Sistemi basati su ARM T1 – 25 febbraio 2022

Nome e Cognome_____

Matricola_____

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

Il progretto funziona in emulazione: sì [ ] no [ ]

Il progetto funziona su board: sì [ ] no [ ]

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

Desidero ritirarmi [ ]

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'apposita area personale Z: predisposta sul computer del LABINF (dove troverete questo testo) 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) nella propria area personale. 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)

Il codice **Morse** fu inventato da Samuel Finley Breese Morse nel 1836. È formato da combinazioni di segnali lunghi e brevi ("linea" e "punto") con cui si rappresentano tutti numeri e le lettere dell'alfabeto. I simboli del codice Morse sono illustrati nella tabella seguente.

A ·–	J ·–––	S ···	2 ··–––
B –···	K –·–	T –	3 ···––
C –·–·	L ·–··	U ··–	4 ····–
D –··	M ––	V ···–	5 ·····
E ·	N –·	W ·––	6 –····
F ··–·	O –––	X –··–	7 ––···
G ––·	P ·––·	Y –·––	8 –––··
H ····	Q ––·–	Z ––··	9 ––––·
I ··	R ·–·	1 ·––––	0 –––––

Utilizzando la scheda LANDTIGER e il system-on-chip LPC1768 si implementi un sistema che decifri il contenuto di un messaggio trasmesso in codice morse, e ne restituisca:

- La sua traduzione in caratteri ASCII.
- Il numero totale di simboli tradotti.

Il sistema progettato deve funzionare come segue:

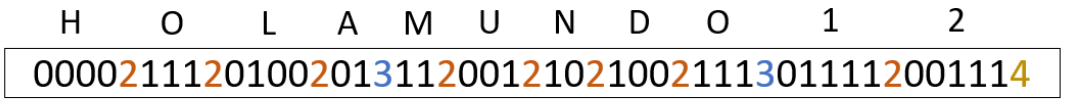
1. *FASE DI ACQUISIZIONE*: Il messaggio viene acquisito tramite l'utilizzo del JOYSTICK e di KEY1.
   - Ad ogni singola pressione del JOYSTICK nella direzione **UP**, viene acquisito un punto (**.**). Si chiede di accendere il LED11 per 1s.
   - Ad ogni singola pressione del JOYSTICK nella direzione **DOWN**, viene acquisito un trattino (-). Si chiede di accendere i LED11 – LED10 – LED9 - LED8 per 1s.

- Ad ogni singola pressione del JOYSTICK nella direzione **RIGHT**, si termina l'acquisizione del simbolo (lettera o numero) corrente.
- Ad ogni singola pressione del JOYSTICK nella direzione **LEFT**, si acquisisce uno spazio.
- Premendo **KEY1**, si termina l'acquisizione della frase. Tutti i LEDs sono accesi per 3s.

Durante la fase di acquisizione, i pulsanti INT0 e KEY2 sono inibiti durante la fase di raccolta dati.

Nello specifico, il messaggio viene trasferito alla funzione che lo decodificherà tramite un vettore *vett_input* di interi senza segno su 8 bits di massimo 100 elementi, che contiene solo i valori 0, 1, 2, 3, 4. Lo **0** viene utilizzato per indicare il punto (**.**), **1** per indicare il trattino (**-**), **2** per indicare il cambio lettera, **3** lo spazio (" "), **4** il termine della frase.

Ad esempio, la frase *HOLA MUNDO 12* viene rappresentata nel seguente modo e contiene in totale 13 simboli. Lo spazio è da considerarsi un carattere ASCII.



KEY1 completa la selezione ed invoca la **funziona ASM** che esegue la traduzione del messaggio.

2. *FASE DI TRADUZIONE*: La funzione ASM da raggiungere tramite invocazione nella parte in linguaggio C ha il seguente prototipo:

```
int traduzione_morse(char* vett_input, char* vett_output);
```

La funzione restituisce il numero totale di segnali tradotti in una variabile intera chiamata RES. Da notare che lo spazio deve essere considerato un carattere.
La funzione salva in *vett_output* la traduzione dei caratteri ASCII del messaggio.

Al rientro dalla funzione ASM:

- I LEDs mostrano il numero totale di segnali tradotti per 2 secondi.
- [**OPZIONALE** → 2 punti extra solo se il messaggio visualizzato è corretto] Il DISPLAY mostra la stringa tradotta visualizzando un nuovo carattere ogni secondo.

Alla pressione di KEY2 il processo ricomincia da 1).

Tabellina ASCII - Morse:

A	01	**J**	0111	**S**	000	**2**	00111
B	1000	**K**	101	**T**	1	**3**	00011
C	1010	**L**	0100	**U**	001	**4**	00001
D	100	**M**	11	**V**	0001	**5**	00000
E	0	**N**	10	**W**	011	**6**	10000
F	0010	**O**	111	**X**	1001	**7**	11000
G	110	**P**	0110	**Y**	1011	**8**	11100
H	0000	**Q**	1101	**Z**	1100	**9**	11110
I	00	**R**	010	**1**	01111	**0**	11111