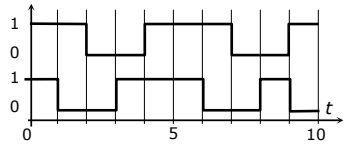
1. [3] Che tipo di numero e che valore rappresenta il codice esadecimale: **0x00000008** nel formato standard IEEE–754, singola precisione?

2. [3] Dare la definizione e scrivere l'espressione generale della seconda forma canonica di una funzione logica. Determinare la seconda forma canonica della seguente funzione: $f(a,b,c) = ab + c(a+b)$

3. [4] Si disegni la struttura circuitale di un decoder a 2 ingressi e se ne calcoli il cammino critico. Tracciare l'andamento delle uscite in corrispondenza degli ingressi in figura.



4. [4] Si progetti un circuito caratterizzato da 3 ingressi A, B e C, ed un'uscita Y. Se A=0, Y assume lo stesso valore di B, mentre se A=1, Y assume il valore negato di C. a) Determinare la tabella di verità di Y; b) esprimerla nella forma canonica più adatta; c) semplificarla mediante mappe di Karnaugh; d) semplificarla ulteriormente, se possibile, mediante semplificazioni algebriche; e) disegnarne lo schema circuitale.

5. [7] Si sintetizzi una macchina a stati finiti di Moore sincrona, caratterizzata da due linee d'ingresso, SET e RESET ed una linea di uscita Z, la quale si porta a "1" in corrispondenza di un fronte di salita su SET, mentre si porta a "0" in corrispondenza di un fronte di salita su RESET (i fronti di salita non sono mai contemporanei). Si considerino inizialmente SET, RESET e Z a "0".
Si determinino: STG, STT, STT codificata e struttura circuitale del sistema completo, non trascurando la gestione del segnale di clock ed avendo cura di semplificare il più possibile le funzioni prima di tradurle in circuito.

6. [7] Si scriva un programma Assembly completo, per ambiente SPIM, il quale richieda all'utente da tastiera gli elementi di un array di 40 numeri interi (allocato staticamente) e, successivamente, li stampi a video in ordine inverso, come appare nell'esempio a lato.

```
Inserisci elemento 1 > 45
Inserisci elemento 2 > 12
...
Inserisci elemento 40 > 9
Elemento(40) = 9
...
Elemento(1) = 45
```

7. [5] Si traduca il seguente frammento di codice dapprima in Assembly MIPS nativo, e quindi in linguaggio macchina, in formato esadecimale. Si supponga che il frammento di codice sia contenuto in memoria a partire dall'indirizzo esadecimale **0x800**.

```
          muli $s1, $s1, 4
ciclo:    beqi $s1, 25, esci  # branch on equal to immediate
          li $t0, 2^13 + 13   # load immediate
          j ciclo
esci:     ...
```


System calls

	codice ($v0)	argomenti	risultato
print_int	1	$a0	
print_float	2	$f12	
print_double	3	$f12	
print_string	4	$a0	
read_int	5		$v0
read_float	6		$f0
read_double	7		$f0
read_string	8	$a0,$a1	
sbrk	9	$a0	$v0
exit	10		

Registri MIPS

0	zero		24-25	t8 – t9
1	at		26-27	k0 – k1
2-3	v0 – v1		28	Gp
4-7	a0 – a3		29	Sp
8-15	t0 – t7		30	s8
16-23	s0 – s7		31	Ra

MIPS Instruction Set: