1. [2] Si scriva in formato esadecimale la codifica del numero decimale: **–25,5625** secondo lo standard IEEE-754, singola precisione.

2. [4] Si disegni la struttura circuitale di un moltiplicatore di parole di 3 bit. Se ne calcoli inoltre il cammino critico, evidenziando il percorso del segnale che lo determina.

3. [6] Si progetti un circuito caratterizzato da quattro ingressi: $\mathbf{a_0\ a_1\ a_2\ a_3}$ sui quali viene presentato un numero binario di 4 bit, e da un'uscita Y che vale '1' se e solo se il numero in ingresso è divisibile per 5 o per 7 (si ricorda che 0 è divisibile per entrambi).
a) Determinare la tabella di verità di Y; b) esprimerla nella forma canonica più adatta; c) semplificarla mediante mappe di Karnaugh; d) disegnarne lo schema circuitale.

4. [8] Si sintetizzi una macchina a stati finiti di Moore che presenta una linea di ingresso, che viene valutata ogni millisecondo, e una linea d'uscita, che vale "1" se e solo se sulla linea d'ingresso persiste il valore "0" da almeno 3 millisecondi.
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.

5. [6] Si traduca in linguaggio Assembly MIPS nativo, evitando cioè di utilizzare pseudo-istruzioni, la seguente procedura in linguaggio C:

```
int fattoriale_bis( int n )
{
     if( n < 2 )
          return( 1 );
     else
          return( 2*n * fattoriale_bis(n-2) );
}
```

6. [4] Si scriva un programma Assembly completo, per ambiente SPIM, per calcolare il "fattoriale_bis" di un numero positivo. Il programma esegue il calcolo chiamando la funzione **`fattoriale_bis`** descritta nell'esercizio precedente, quindi termina. Il programma deve presentarsi a terminale come nel seguente esempio:

```
Inserisci un numero intero positivo
> 4
Il fattoriale_bis di 4 risulta: 32
```

7. [4] Si traduca il seguente frammento di codice: a) in Assembly MIPS nativo e b) in linguaggio macchina, specificando valore e ampiezza in bit dei campi di ogni istruzione:

```
lw $s1, $s2($s3)
bgei $t0, +20, -20     # branch if greater or equal than immediate
```


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: