1. [2] Che numero decimale rappresenta il codice esadecimale: **0x 4580 2800** nel formato standard IEEE–754 a singola precisione?

2. [3] Si enuncino e si dimostrino le due leggi dell'assorbimento.

3. [5] Si progetti un circuito caratterizzato da quattro ingressi (a_0,a_1,a_2,a_3) che rappresentano un numero binario **A** di 4 bit, e da un'uscita **Y** che vale '1' se e solo se:
- A≤8 ed è pari, oppure se:
- A>8 ed è multiplo di 4.

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 trasformazioni algebriche; e) 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 cambia di stato ad ogni fronte di discesa del segnale di ingresso. Si considerino inizialmente sia l'uscita che l'ingresso 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.

5. [6] Si traduca in linguaggio Assembly MIPS nativo (evitando cioè di utilizzare pseudo-istruzioni) la seguente procedura in linguaggio C. Si faccia in modo che la funzione si aspetti l'argomento nel registro **$a0** e restituisca il risultato in **$v0**.

```
int MioFattoriale( int n )
{
     if( n > 1 )
          return( (n+1) * MioFattoriale(n-2) );
     else
          return( 1 );
}
```

6. [4] Si scriva un programma Assembly completo per ambiente SPIM, per calcolare il valore della funzione "MioFattoriale" di cui all'esercizio precedente, di un numero intero fornito da tastiera. Il programma esegue il calcolo chiamando la funzione **MioFattoriale**, quindi termina. Il programma deve presentarsi a terminale come nel seguente esempio:

```
Inserisci un numero intero:
> 5
Il mio fattoriale di 5 vale: 24
```

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.

```
bgei $v0, 12, -24      # branch on greater or equal than immediate
muli $a0, $s1, 16^5    # multiply immediate
```

8. [4] Si disegni la struttura di una ALU a 8 bit in grado di fare AND, OR, addizione e sottrazione. Si rappresenti sia lo schema a blocchi modulare che lo schema circuitale di un singolo modulo di ALU elementare.

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:

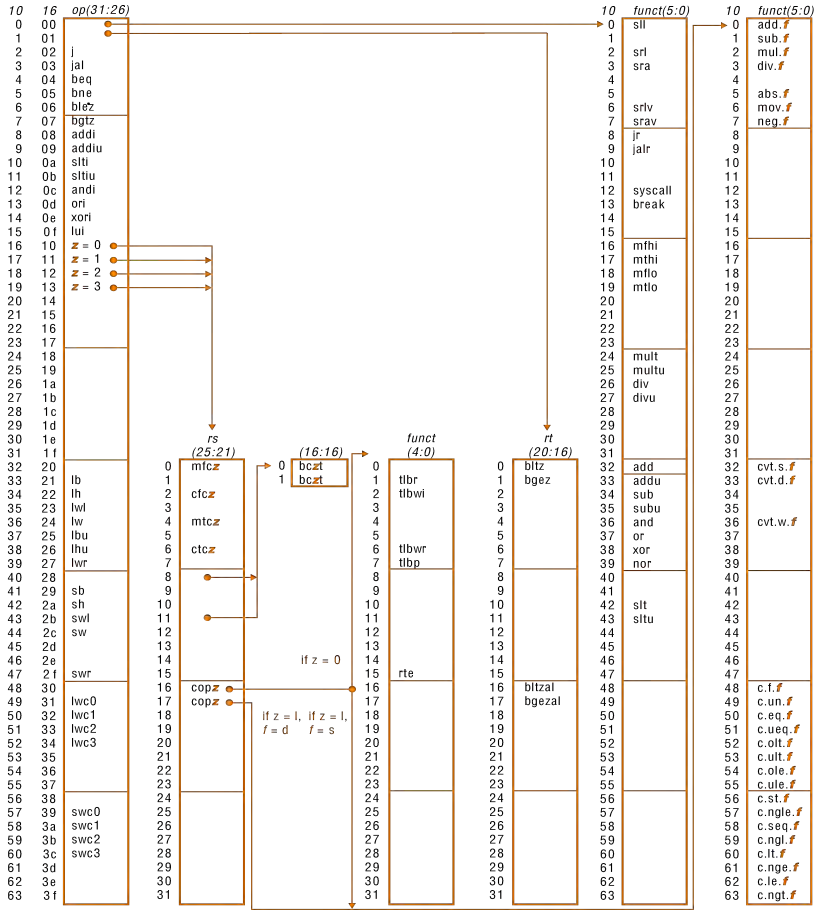

FIGURE A.19 MIPS opcode map. The values of each field are shown to its left. The first column shows the values in base 10 and the second shows base 16 for the op field (bits 31 to 26) in the third column. This op field completely specifies the MIPS operation except for 6 op values: 0, 1, 16, 17, 18, and 19. These operations are determined by other fields, identified by pointers. The last field (funct) uses "f" to mean "s" if rs = 16 and op = 17 or "d" if rs = 17 and op = 17. The second field (rs) uses "z" to mean "0", "1", "2", or "3" if op = 16, 17, 18, or 19, respectively. If rs = 16, the operation is specified elsewhere: if z = 0, the operations are specified in the fourth field (bits 4 to 0); if z = 1, then the operations are in the last field with f = s. If rs = 17 and z = 1, then the operations are in the last field with f = d. *(page A-54)*