1. [5] Si progetti un circuito caratterizzato da 3 bit di ingresso (A, B e C) e due bit d'uscita (y_1 e y_0). Il circuito funziona così:

- se A=1 ➔ Y = (y_1,y_0) = "11"
- se A=0 e B=1 ➔ Y = (y_1,y_0) = "10"
- se A=B=0 e C=1 ➔ Y = (y_1,y_0) = "01"
- se A=B=C=0 ➔ Y = (y_1,y_0) = "00"

a) Determinare le tabelle di verità delle funzioni logiche di uscita; b) esprimerle nella forma canonica più adatta; c) semplificarle mediante mappe di Karnaugh; d) se possibile, semplificarle ulteriormente mediante passaggi algebrici.

2. [4] Si progetti un circuito in grado di calcolare la differenza tra due numeri di 4 bit utilizzando circuiti sommatori (Half adder o Full adder) sfruttando le proprietà della rappresentazione in complemento a 2 per numeri negativi.

3. [8] Si sintetizzi una macchina a stati finiti (di Moore) caratterizzata da un ingresso **A** ed un ingresso **Reset**, e da 2 linee di uscita, **P** e **D**, che vanno a "1" se il numero di fronti di salita pervenuti all'ingresso A è, rispettivamente, pari o dispari. Se RESET è "1", il conteggio dei fronti viene azzerato. Gli ingressi vengono valutati ogni millisecondo.
Si determinino STG, STT, STT codificata e struttura circuitale del sistema completo, senza trascurare la gestione del segnale di clock e avendo cura di semplificare il più possibile le funzioni prima di tradurle in circuito.

4. [5] Un processore caratterizzato da uno spazio di indirizzamento della memoria principale di 1 GByte e da un bus dati di 16 bit viene dotato di una memoria cache associativa a 2 vie, di capacità totale C = 1 MByte e con linee di 8 parole. Dimensionare la cache, evidenziando le dimensioni di tutti i campi, e disegnarne lo schema circuitale dettagliato. Determinare i valori di: byte offset, word offset, index e tag relativi all'indirizzo: A = 2^{20} + 2^{15} + 2^{10} + 2^5 + 5.

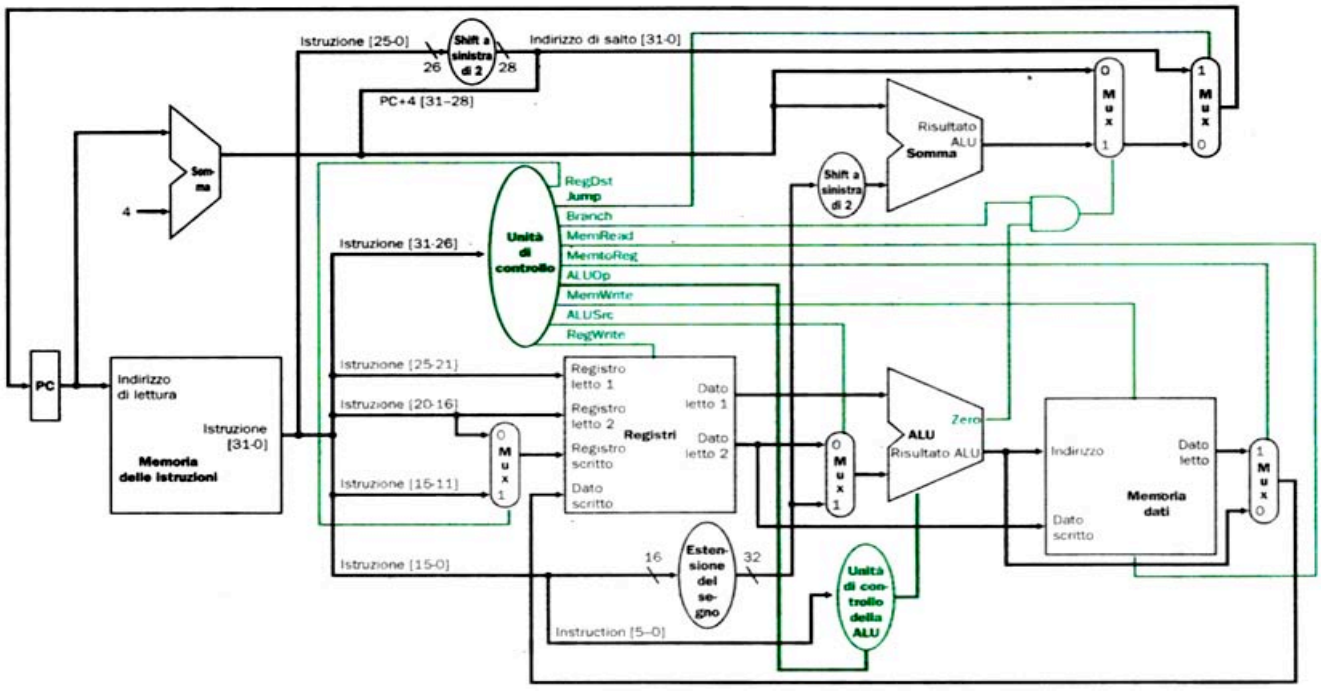
5. [4] Elencare e descrivere i metodi di indirizzamento implementati nell'architettura MIPS rappresentando, per ognuno di essi, le parti di circuito interessate al recupero dell'operando.

6. [4] Si traducano le seguenti pseudoistruzioni: **a)** in Assembly MIPS nativo e **b)** in linguaggio macchina MIPS (specificando dimensione in bit e valore dei singoli campi istruzione):

```
blei $5, +10, -40          # branch on less or equal than immediate
lw $4, $5($6)
```

7. [5] L'architettura in figura esegue il codice riportato a lato. Si determini lo stato della CPU (i valori all'ingresso ed all'uscita di ogni ALU ed i valori di ogni segnale di controllo) dopo **12 cicli** di clock a partire dal prelievo della prima istruzione.
Ci sono hazard? Se sì, quali?

```
0x100: addi $s4, $0, 50
       subi $s4, $s4, 1
       add $t0, $s0, $s4
       lw  $t1, 0($t0)
       bne $s4, $0, -16
```



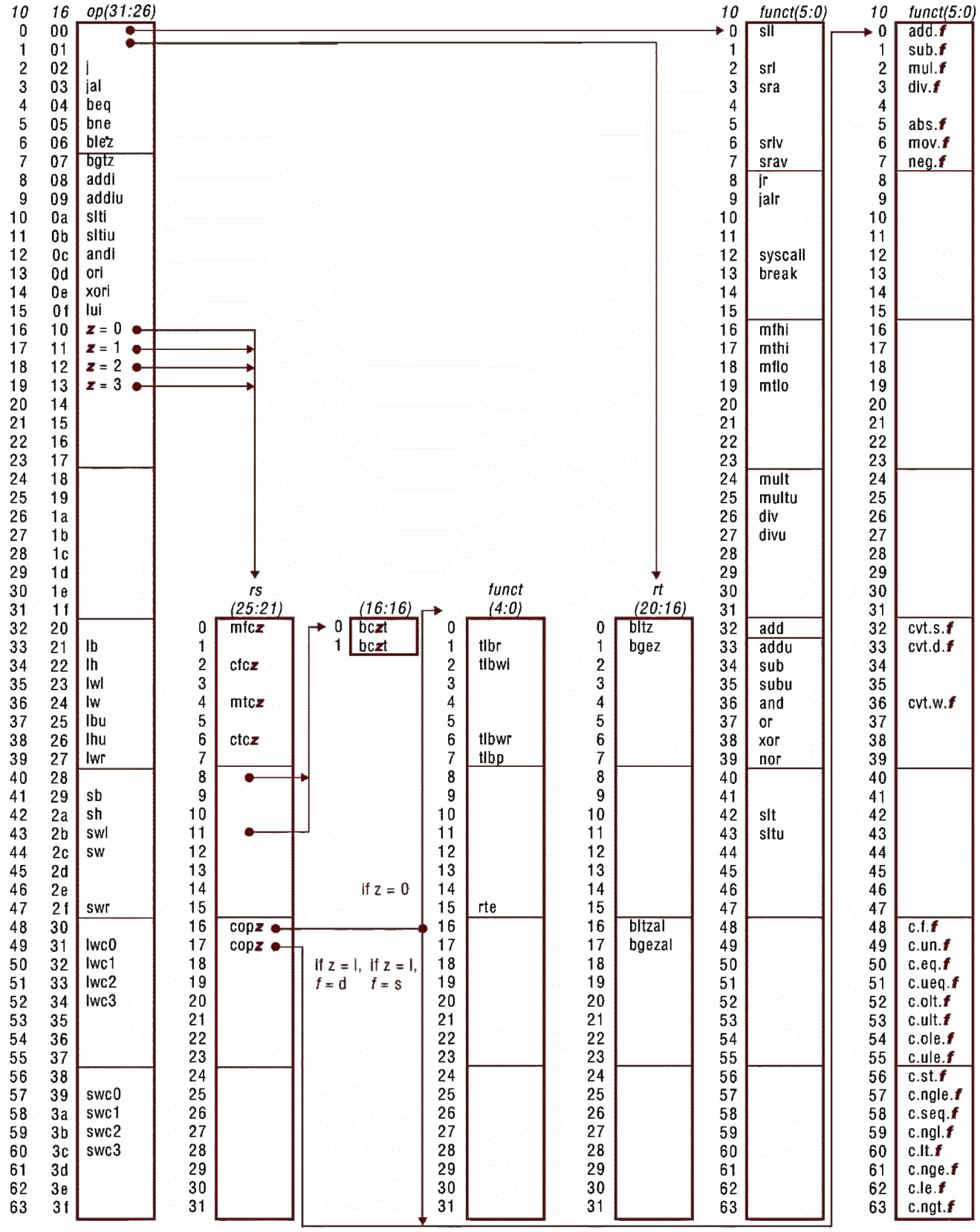


FIGURE A.19 MIPS opcode map. The values of each field are shown to its left. The first column shows the values in base 10 and the second shows base 16 for the op field (bits 31 to 26) in the third column. This op field completely specifies the MIPS operation except for 6 op values: 0, 1, 16, 17, 18, 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)*