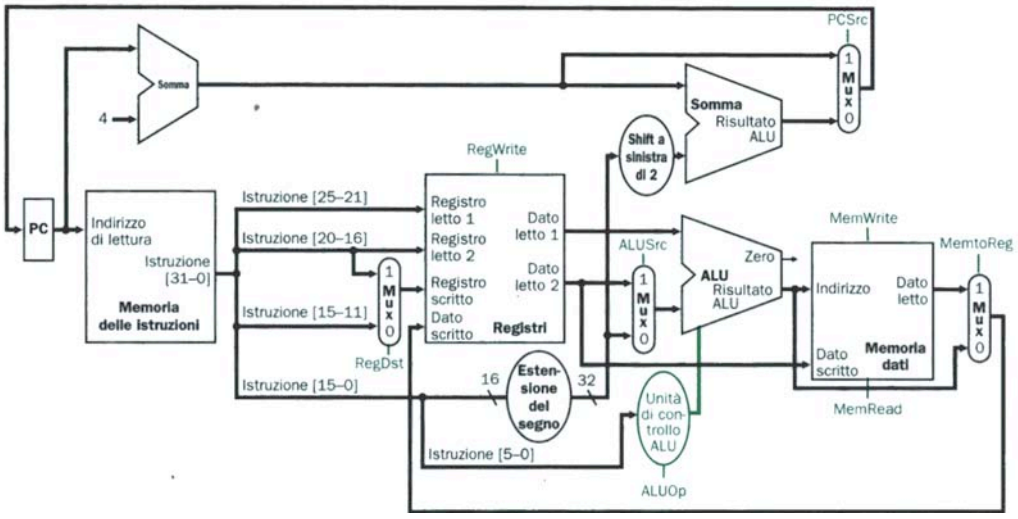
1. [5] Si progetti un circuito caratterizzato da un ingresso a 4 bit che rappresenta un numero binario A, e da un'uscita che vale '1' se: A è divisibile per 4, oppure se A < 4; altrimenti l'uscita vale '0'.
   - Determinare la tabella di verità della funzione logica di uscita;
   - scrivere la funzione nella forma canonica più adatta;
   - semplificarla mediante mappa di Karnaugh e poi, se possibile, semplificarla ulteriormente mediante passaggi algebrici;

2. [4] Evidenziare, nel seguente schema di CPU, i valori all'ingresso ed all'uscita di ogni ALU ed i valori di ogni segnale di controllo, supponendo che la CPU esegua l'istruzione: **`0x300: lw $2, 16($1)`**



3. [8] Si sintetizzi una macchina a stati finiti (di Moore) che realizza un contatore modulo 3, caratterizzato da una linea di ingresso che viene valutata dalla macchina ogni centesimo di secondo. Il contatore viene incrementato ogni qualvolta si verifica che: il valore attuale dell'ingresso è '1' ed il valore precedente è '0'.

   L'uscita è costituita da 2 linee, che rappresentano il valore binario assunto dal contatore.

   Si determinino STG, STT, STT codificata e struttura circuitale del sistema completo, avendo cura di semplificare il più possibile le funzioni prima di tradurle in circuito.

4. [6] Si progetti (esplicitando le dimensioni di tutti i campi) una memoria cache a quattro vie, associata ad un processore con bus dati di 64 bit e bus indirizzi di 32 bit. La capacità totale della cache sia di 2 Mbyte e la dimensione del blocco di 16 parole. Se ne rappresenti lo schema circuitale dettagliato e si calcoli il valore (decimale) di tutti i campi di indirizzamento relativi all'indirizzo: **`0xABC10ABC`**.

5. [4] Disegnare la struttura circuitale di una cella di memoria RAM dinamica. Data una RAM dinamica di 4 Mbit, disegnarne la struttura circuitale globale e calcolare il periodo massimo di refresh, supponendo il tempo di scarica delle celle di memoria pari a 51,2 msec.

7. [4] Supponendo che il frammento di codice riportato a lato venga eseguito da una CPU MIPS pipeline sprovvista di unità di propagazione, nella quale gli hazard vengono gestiti mediante bolle, determinare:

   quanti cicli di clock sono necessari ad eseguire tutto il frammento di codice (dal fetch della prima al WB dell'ultima).

```
addi $s0, $s0, +1
lw  $s0, 0($s5)
add $s6, $s6, $s0
addi $s6, $s6, +1
```

8. [4] Si traducano le seguenti pseudoistruzioni: **a)** in Assembly MIPS nativo e **b)** in linguaggio macchina MIPS (specificando dimensione in bit e valore dei campi di ogni istruzione).

```
li $25, (2^25 + 35)      # load immediate
blei $18, +19, –20       # branch on less than or equal to immediate
```

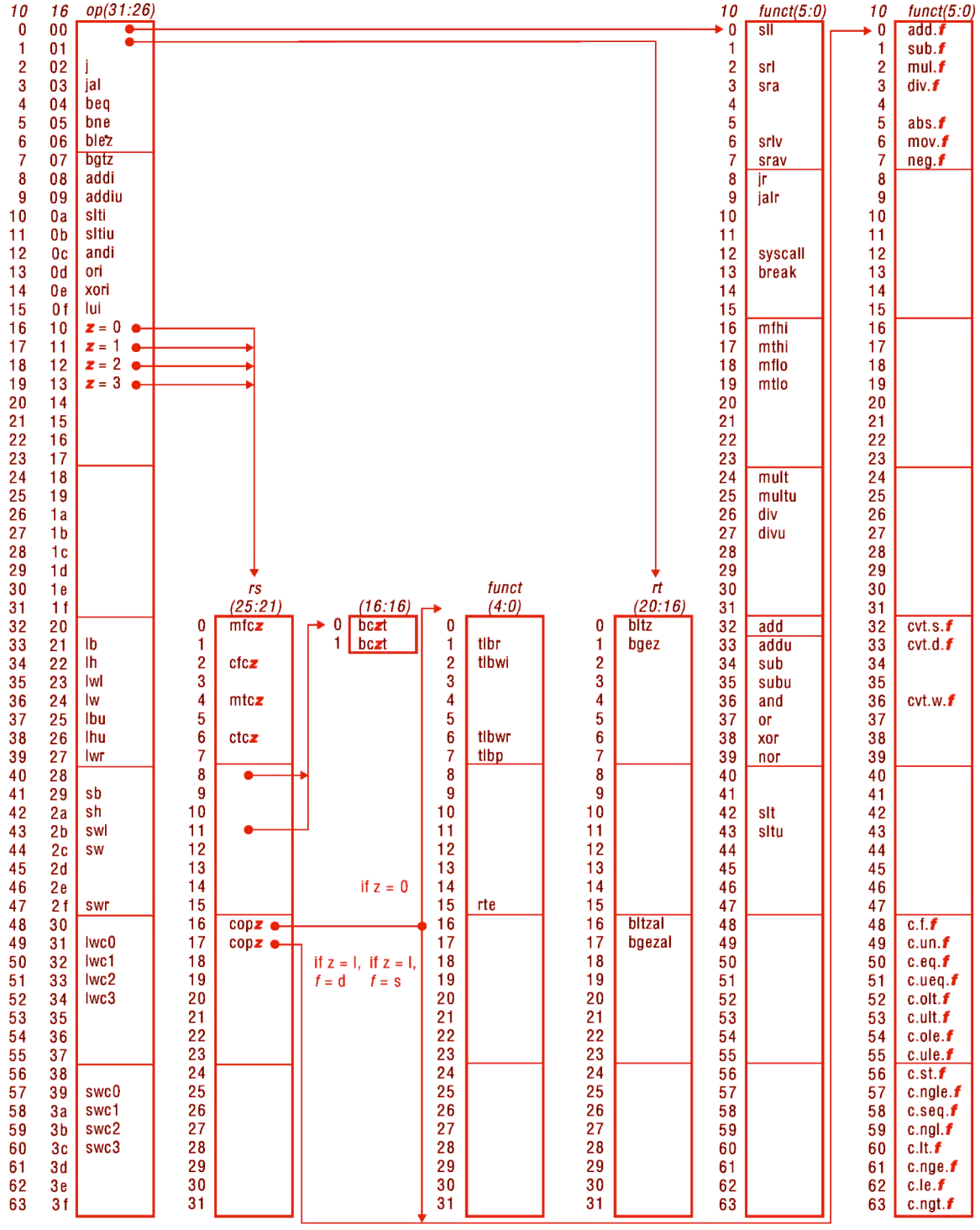


FIGURE A.19 MIPS opcode map. The values of each field are shown to its left. The first column shows the values in base 10 and the second shows base 16 for the op field (bits 31 to 26) in the third column. This op field completely specifies the MIPS operation except for 6 op values: 0, 1, 16, 17, 18,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