1. [4] Si progetti un circuito generatore di bit di parità su una parola di 3 bit (b_0, b_1, b_2) in ingresso. Il bit in uscita Y vale 1 se il numero di "1" presenti sugli ingressi è dispari, altrimenti vale 0. a) Determinare le tabelle di verità della funzione Y; b) esprimerla nella forma canonica più adatta; c) semplificare mediante mappe di Karnaugh; d) se possibile, semplificarle ulteriormente mediante passaggi algebrici.

2. [3] Si disegni lo schema di un moltiplicatore hardware di parole di 3 bit e si calcoli il cammino critico, evidenziando i percorsi di segnale che lo determinano.

3. [9] Progettare una macchina a stati finiti di Moore che accetti in ingresso due bit, b_0 e b_1, e sia caratterizzata da una linea d'uscita Y che vale "1" solo quando il valore corrente della parola di due bit in ingresso è uguale alla parola precedente. I valori dell'ingresso vengono valutati ogni centesimo di secondo. Allo stato iniziale, la macchina considera come parola precedente "00".
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 64 GByte e da un bus dati di 64 bit viene dotato di una memoria cache associativa a 2 vie, di capacità totale C = 16 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: $A = 2^{24} + 2^{18} + 2^{12} + 2^6 + 15$.

5. [3] Descrivere le strategie possibili per la scelta del banco nella scrittura di un blocco in cache, in memorie cache n-associative. Si discutano e si confrontino le caratteristiche delle diverse strategie.

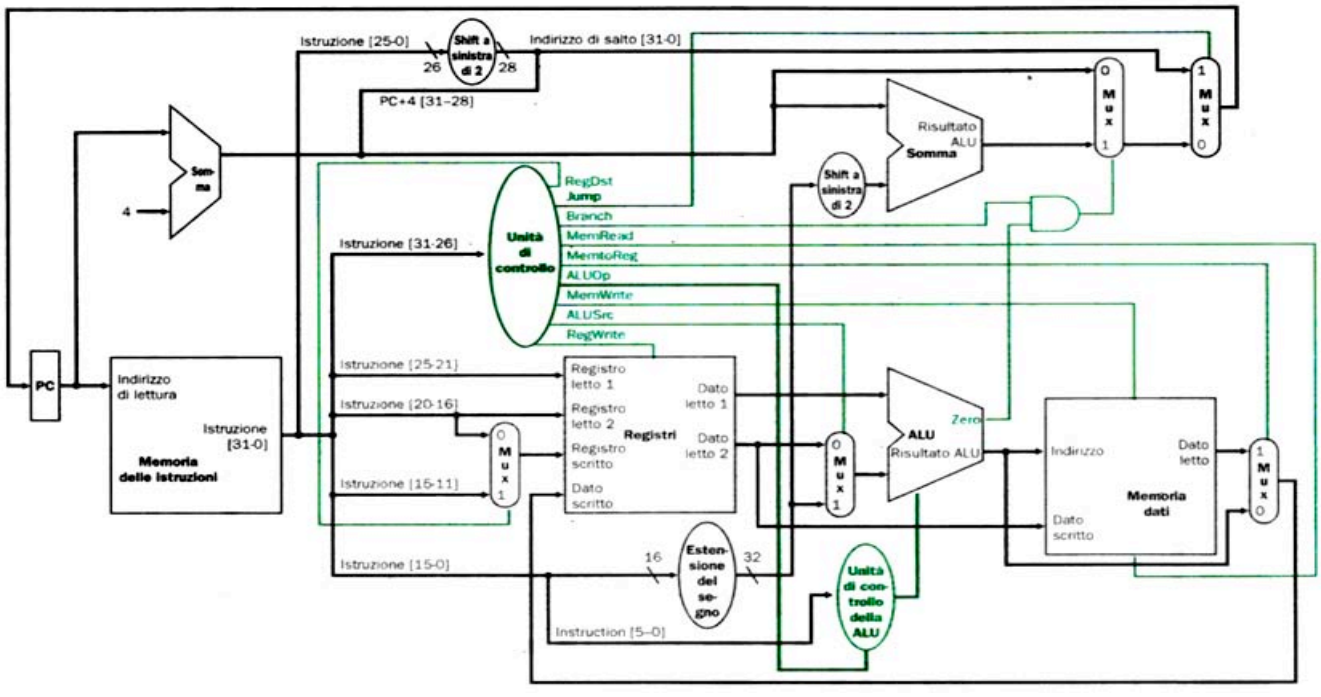
6. [3] Descrivere il meccanismo di "back-off" nel protocollo Ethernet (IEEE 802.3), spiegando in che modo esso si adatta dinamicamente alle condizioni di traffico.

7. [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):

```
divi $5, $6, 40        # divide by immediate
li $4, 4^10 + 4^4
```

8. [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 **6 cicli** di clock a partire dal prelievo della prima istruzione.
Ci sono hazard? Se sì, quali?

```
0x100: subi $s4, $a0, 100
       addi $s2, $s4, 2
       lw  $t1, 0($s2)
       beq $s2, $a0, -12
       sub $t0, $s0, $s4
```



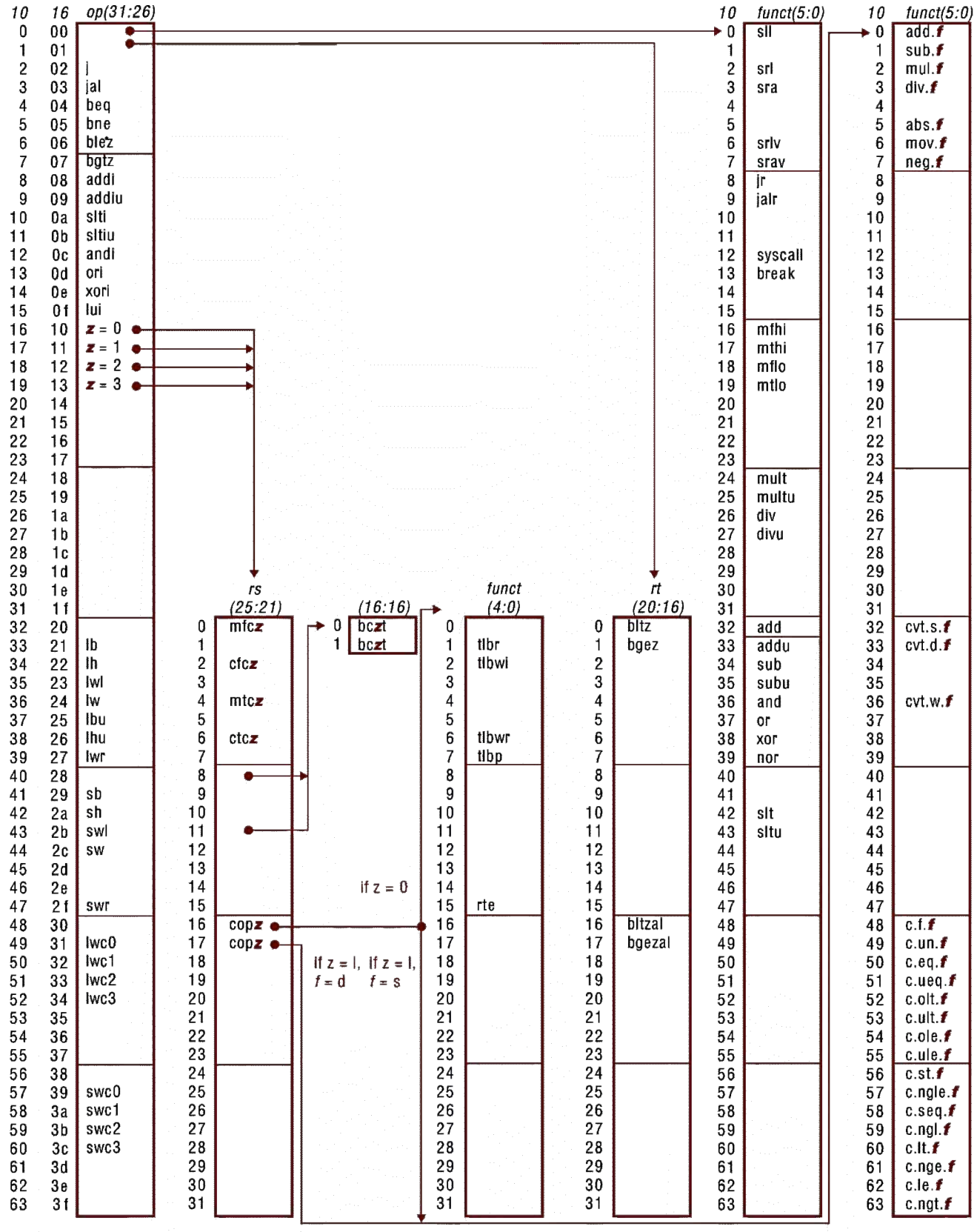


FIGURE A.19 MIPS opcode map. The values of each field are shown to its left. The first column shows the values in base 10 and the second shows base 16 for the op field (bits 31 to 26) in the third column. This op field completely specifies the MIPS operation except for 6 op values: 0, 1, 16, 17, 18,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