**Università degli Studi di Milano - Corso Architettura degli elaboratori e delle reti – Turno 1 – Prof. Borghese**

***Cognome e nome dello studente:*** ***Matricola:***

**A.A. 2007-2008 – Seconda prova in itinere**

**A. [9] Progettare una macchina a stati finiti in grado di leggere un testo e di riconoscere all'interno di una stringa di caratteri la sequenza: 'AAB'.**
Supponiamo che:
la macchina legga in ogni istante un carattere e che questo carattere sia o '0', 'A', 'B' o 'C'.
all'inizio la stringa contiene i caratteri '0C'.
Definire lo STG, la STT, codificarla e realizzare il circuito mediante porte logiche. Qual è il cammino critico della macchina a stati finiti?
Quali tempi occorre considerare per definire il periodo del clock? Definire dei valori per questi tempi e calcolare la frequenza del clock che si può ottenere.

**B. [2] Quale di queste frasi è corretta?**
Uno shift register si può realizzare con latch di tipo D
Un flip-flop non si può utilizzare come elemento di memoria.
Un latch è più lento di un flip-flop, a parità di tecnologia.
E giustificare la risposta

**C. [3]** In quali segmenti viene suddivisa la memoria gestita da un processore MIPS? Cosa si intende per codifica big-endian o little endian? Riguarda i dati o le istruzioni? Cosa rappresenta lo stack? Come si può leggere dalla memoria un'istruzione il cui indirizzo è > 2Gbyte?

**D. [3]** Costruire la porta di scrittua del register file e calcolare il cammino critico. Specificare in quali fasi del clock è possibile scrivere sul register file e perché.

**E. [3]** Scrivere il circuito del flip-flop di tipo D, e scrivere la tabella di eccitazione e di transizione. Definire correttamente: bistabili, latch e flip-flop.

**F - [7]** Dati i seguenti due programmi:

```
Main:
Prima:  li $v0,2
        sub $t0, $t1, $t2
        j dopo
        jal Proc_A
        beq $s0, $s1, Prima
        add $t1, $t1, $t3
        lw $s0, 0($gp)
dopo:   ….
        ….
```

```
Proc_A
prima:  sub $t0, $t2, $t3
        j fine:
        beq $t0, $t1, prima
        sw$s0, 0($gp)     ….
fine:
        ….
        ….
        …..
        …..
```

Scrivere il programma eseguibile (linked), risultante, sapendo che il segmento testo del Main è di 2Kbyte e di Proc_A è di 512 Byte; e che il segmento dati del Main è di 12Kbyte e di Proc_A è di 1Kbyte.

**G. [3]** Tradurre in linguaggio macchina il segmento di codice di Proc_A definito al punto E.

**H - [1]** Indicare le modalità di indirizzamento dei dati previste dal MIPS.

**I - [3]** Tradurre in linguaggio Assembly il seguente spezzone di codice C:

```
j = 4;
for (i=j; i<N; i++)
{   A[i] = s + j;
}
```

**J. [2]** Definire come si può tradurre in Assembly il costrutto "switch" utilizzando la Jump Address Table.

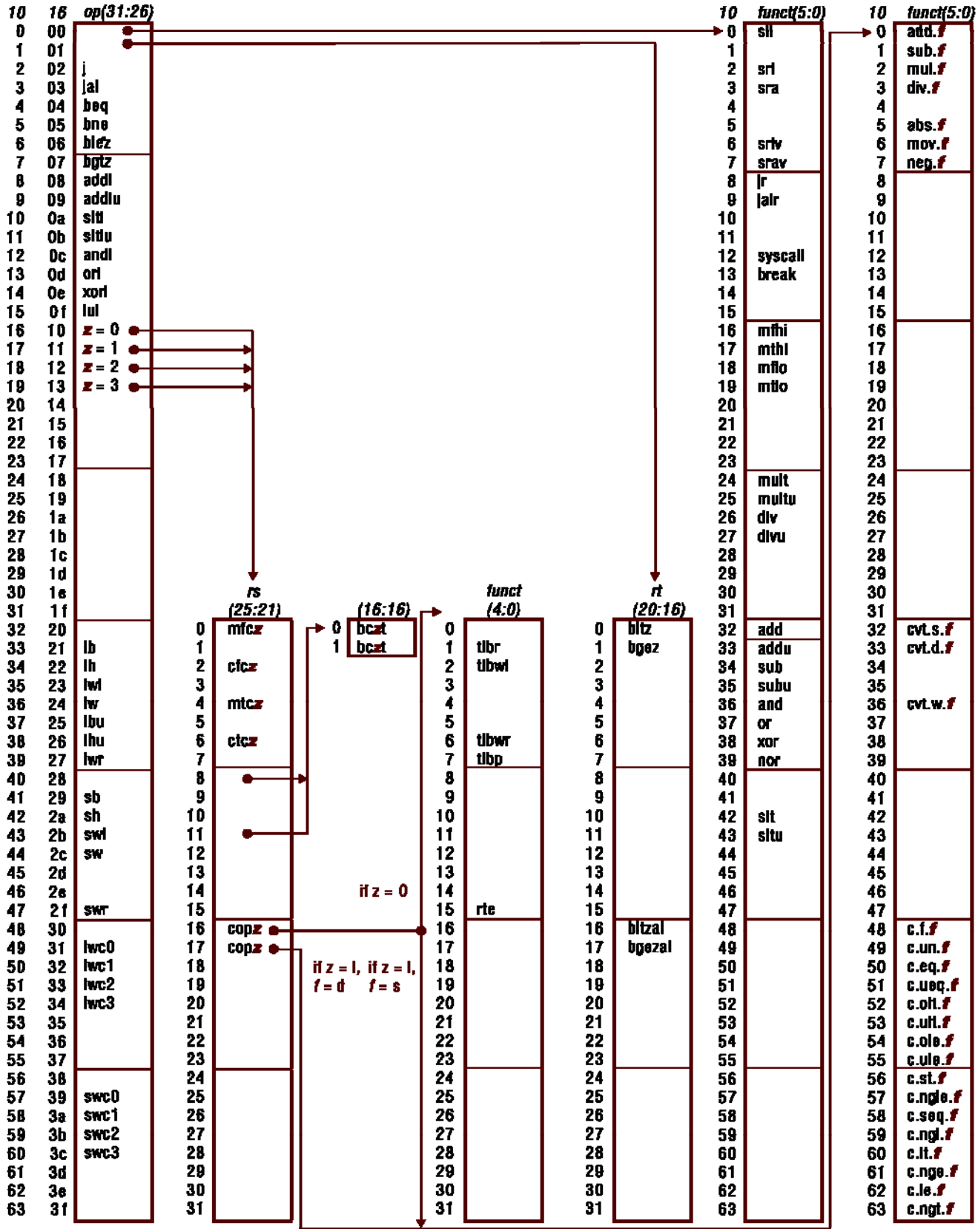
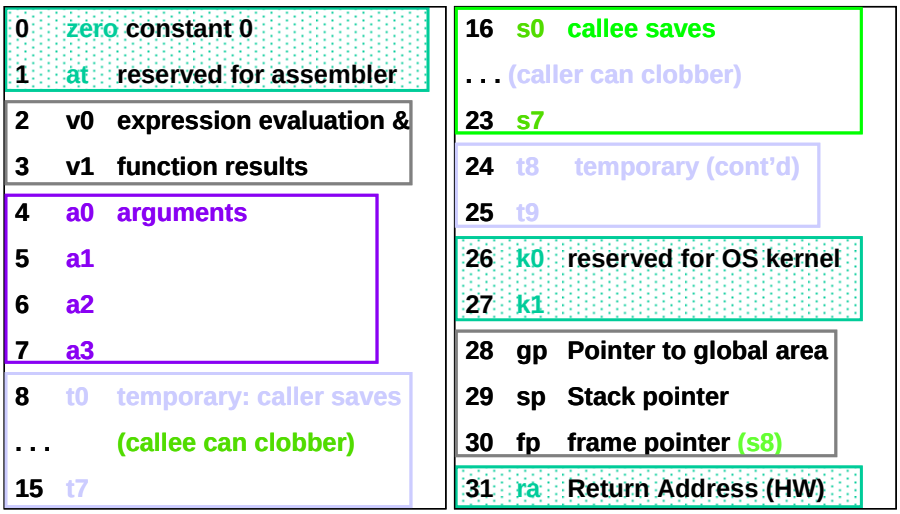



**FIGURE A.19 MIPS opcode map.** The values of each field are shown to its left. The first column shows the values in base 10 and the second shows base 16 for the op field (bits 31 to 26) in the third column. This op field completely specifies the MIPS operation except for 6 op values: 0, 1, 16, 17,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

| Number | Name | Cause of exception |
|---|---|---|
| 0 | Int | interrupt (hardware) |
| 4 | AdEL | address error exception (load or instruction fetch) |
| 5 | AdES | address error exception (store) |
| 6 | IBE | bus error on instruction fetch |
| 7 | DBE | bus error on data load or store |
| 8 | Sys | syscall exception |
| 9 | Bp | breakpoint exception |
| 10 | RI | reserved instruction exception |
| 11 | CpU | coprocessor unimplemented |
| 12 | Ov | arithmetic overflow exception |
| 13 | Tr | trap |
| 15 | FPE | floating point |




| Nome del registro | Numero del registro in coprocessore 0 | Utilizzo |
|---|---|---|
| Bad/Addr | 8 | Registro contenente l'indirizzo di memoria a cui si è fatto riferimento |
| Count | 9 | Timer |
| Compare | 11 | Valore da comparare con un timer. Genera un interrupt. |
| Status | 12 | Maschera delle interruzioni e bit di abilitazione. Stato dei diversi livelli di priorità (6 HW e 2 SW). |
| Cause | 13 | Tipo dell'interruzione e bit delle interruzioni pendenti |
| EPC | 14 | Registro contenente l'indirizzo dell'istruzione che ha causato l'interruzione. |