ESERCIZIO 1 Si consideri una nuova istruzione `addm d, off(s)` che ha l'effetto `d = d + Mem[s + off]`, cioè addiziona al valore del registro `d` il valore in memoria all'indirizzo dato dalla somma del valore del registro `s` e `off`. Implementare la nuova istruzione `addm`, eventualmente aggiungendo allo schema generale linee di controllo, addizionatori, multiplexer, ecc. e tenendo presente che è rappresentata nel formato I con opcode = `100111` (cioè, un opcode differente da quello di tutte le altre istruzioni).

ESERCIZIO 2 C'è il sospetto che una partita di processori abbia la linea di controllo `RegDst` difettosa essendo sempre uguale a `RegWrite`, cioè `RegDst` è asserita se e solo se `RegWrite` è asserita.

1. Quali istruzioni tra quelle di base (`lw`, `sw`, `add`, `sub`, `and`, `or`, `xor`, `slt`, `beq`, `j`) non funzionano correttamente, se il processore è difettoso.
2. Scrivere un programma in assembly (senza pseudo-istruzioni) che quando termina, il valore del registro `$s0` è 1 se il processore è difettoso e 0 altrimenti.

ESERCIZIO 3 Quali sono i valori dei registri `$s0` e `$s1` quando il seguente programma termina? Mostrare una traccia del procedimento usato.

```
            li $t1, 0
            li $t0, 5
loop_array: beq $t1, 8, end_loop
            sll $t2, $t1, 2
            sw $t0, array($t2)      #array è l'indirizzo di un array di words
            addi $t0, $t0, 5
            andi $t0, $t0, 7
            addi $t1, $t1, 1
            j loop_array
end_loop:   li $s1, 0
loop_val:   sll $t1, $s1, 2
            lw $s0, array($t1)
            blt $s0, $s1, end
            move $s1, $s0
            j loop_val
end:
```

ESERCIZIO 4 Il programma seguente scambia i valori minimo e massimo di un array di 10 interi. Però, tre istruzioni nelle posizioni indicate dai punti interrogativi sono andate perse. Trovare le istruzioni mancanti.

```
            la $s1, array + 40        #array è l'indirizzo dell'array
            la $s0, array
            la $t0, array
            ?
            la $t2, array
            lw $t3, ($t2)
loop:       beq $s0, $s1, end
            lw $s2, ($s0)
            addi $s0, $s0, 4
            blt $s2, $t1, min_update
            ?
            j loop
min_update: addi $t0, $s0, -4
            move $t1, $s2
            j loop
max_update: addi $t2, $s0, -4
            move $t3, $s2
            j loop
end:        ?
            sw $t3, ($t0)
```