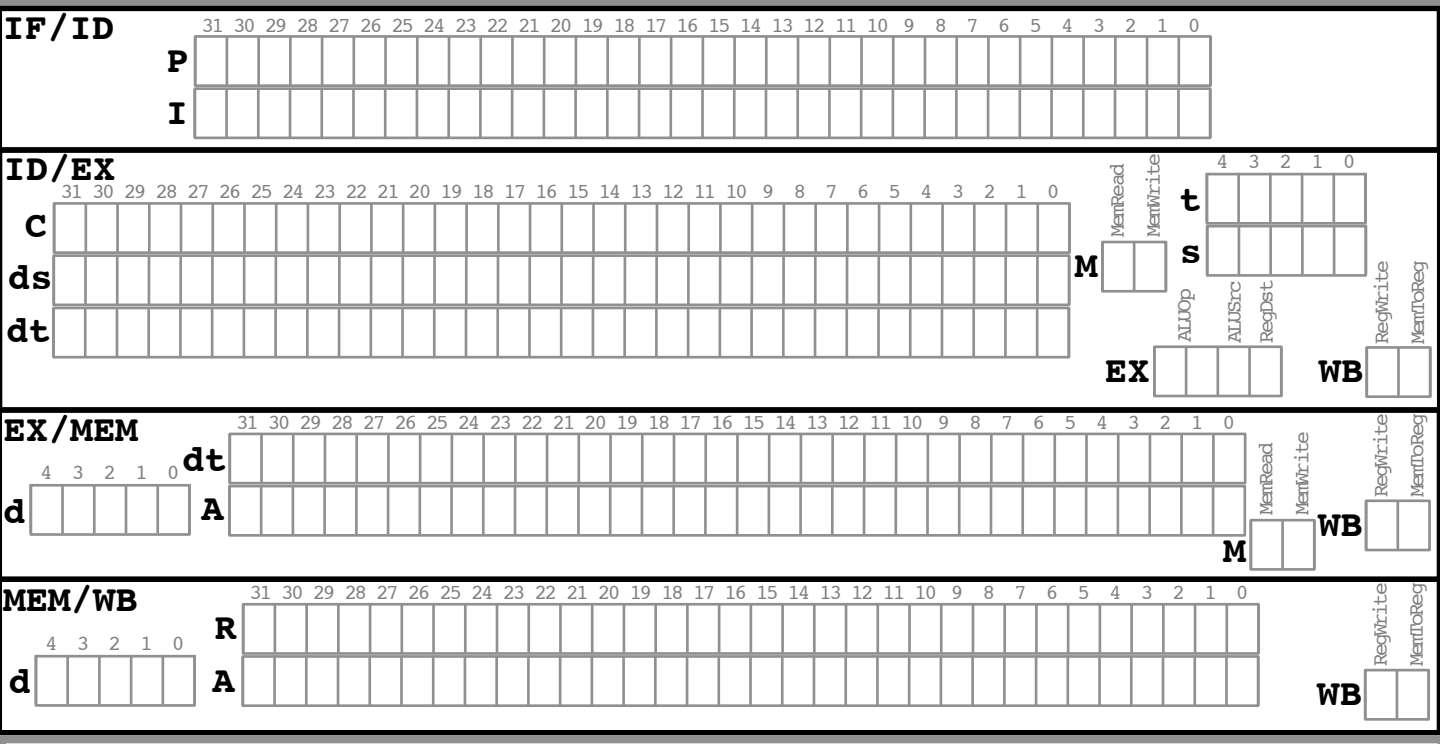
ESERCIZIO 1 Si consideri il seguente programma (accanto ad ogni istruzione è riportata la rappresentazione in binario):

```
0x00400000   lw $s0, ($s1)          100011 10001 10000 0000000000000000
             add $s1, $s1, $s0      000000 10001 10000 10001 00000 100000
             addi $s2, $s1, 4       001000 10001 10010 0000000000000100
             lw $s0, ($s1)          100011 10001 10000 0000000000000000
```

Assumendo un'esecuzione in pipeline secondo lo schema riportato nella pagina seguente (è quello visto a lezione) e che quando inizia l'esecuzione, al ciclo di clock 1, i contenuti dei registri sono `s0 = 0x00000000`, `s1 = 0x10000000`, `s2 = 0x00000000` e la word in memoria all'indirizzo `0x10000000` è `0x00000010`, determinare il contenuto dei registri di pipeline alla fine del ciclo 5 riempendo lo schema qui sotto riportato.



ESERCIZIO 2 Determinare il numero di cicli di clock richiesti per eseguire il seguente programma secondo lo schema di implementazione in pipeline (mostrare una traccia del procedimento usato):

```
         addi $t0, $zero, 11
loop1:   addi $t0, $t0, -1
         addi $t1, $s0, 16
loop2:   beq $t1, $s0, next
         lw $t2, arr1($t1)      #arr1 è l'indirizzo di un array di word
         sw $t2, arr2($t1)      #arr2 è l'indirizzo di un'altro array di word
         addi $t1, $t1, -4
         add $s1, $s1, $t2
         j loop2
next:    addi $s0, $s0, 128
         bne $t0, $zero, loop1
         sub $s1, $s1, $s2
```

ESERCIZIO 3 Si consideri una cache a due livelli. Il primo livello L1 è ha mappatura diretta con 4 blocchi da 8 words. Il secondo livello L2 è 2-way set associative con 8 insiemi, blocchi da 8 words e strategia di rimpiazzo LRU. Si consideri la seguente serie di accessi a word in memoria: 36, 192, 64, 200, 40, 320, 72, 92, 300, 60 (gli indirizzi sono a byte). Determinare per ogni accesso gli hit e i miss su L1 e su L2. Inoltre, sapendo che il tempo di hit su L1 è 1 ciclo di clock, il tempo di miss su L1 è 10 cicli e il tempo di miss su L2 è 150 cicli, calcolare il numero totale di cicli richiesti dalla serie di accessi.

ESERCIZIO 4 Consideriamo un sistema con memoria virtuale: pagine da 4KB, una TLB dati da 16 elementi e una cache dati 2-way set associative con 8 insiemi e blocchi da 128 words. Sia la TLB che la cache adottano una strategia di rimpiazzo LRU. Un processo esegue la seguente serie di accessi a word alla memoria dati: 1024, 2048, 8200, 9400, 4100, 8000, 1024, 12000, 1256, 2048 (gli indirizzi sono virtuali e a byte). Assumendo che inizialmente la TLB e la cache sono vuote e nessuna pagina del processo è in memoria principale, determinare per ogni accesso gli hit/miss su TLB e cache e i page fault. Si assume che nessuna delle pagine caricate in memoria principale durante la serie di accessi venga rimpiazzata. Trascurare i miss sulla cache che accadano durante un page fault.

Esercizi di recupero

ESERCIZIO 5 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 ad un ciclo di clock 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 6 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, arr + 40        #arr è l'indirizzo dell'array
            la $s0, arr
            la $t0, arr
            ?
            la $t2, arr
            lw $t3, ($t2)
loop:       beq $s0, $s1, end
            lw $s2, ($s0)
            addi $s0, $s0, 4
            blt $s2, $t1, min
            ?
            j loop
min:        addi $t0, $s0, -4
            move $t1, $s2
            j loop
max:        addi $t2, $s0, -4
            move $t3, $s2
            j loop
end:        ?
            sw $t3, ($t0)
```