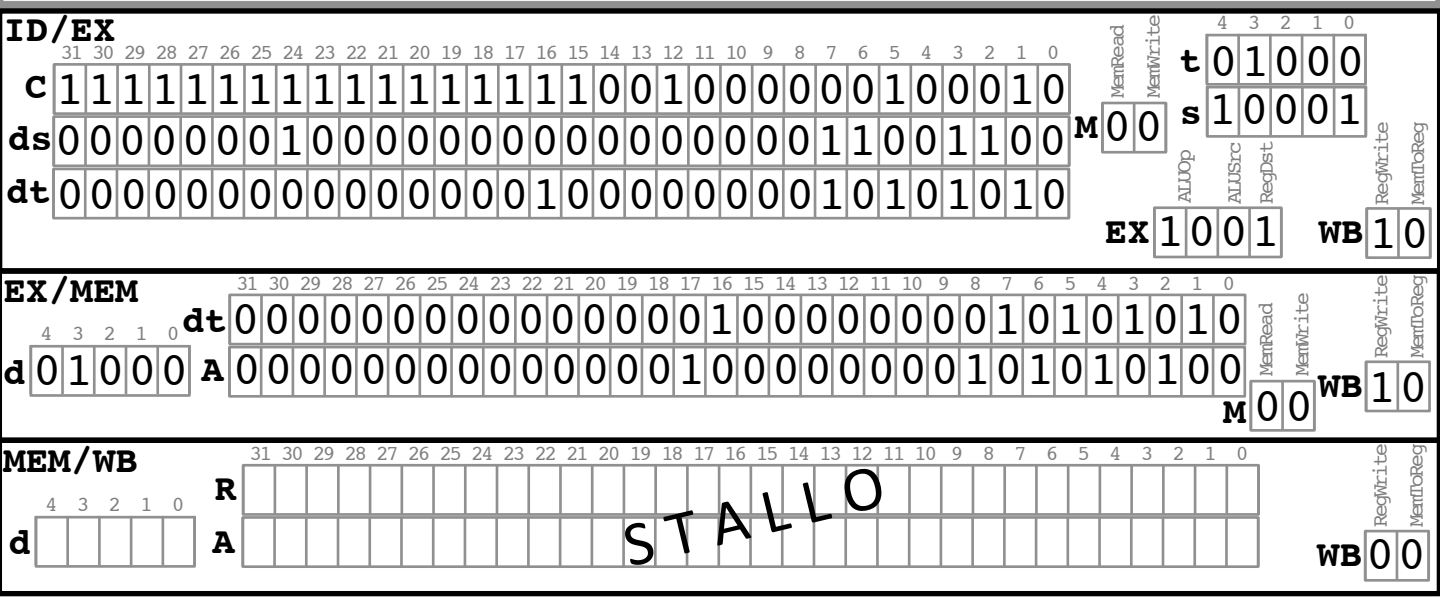
ESERCIZIO 1A Si consideri il seguente programma (accanto ad ogni istruzione è riportata la rappresentazione in binario):

```
sw $s0, 40($s1)          101011 10001 10000 0000000000101000
lw $t0, 40($s1)          100011 10001 01000 0000000000101000
add $t0, $t0, $t0        000000 01000 01000 01000 00000 100000
sub $s2, $s1, $t0        000000 10001 01000 10010 00000 100010
```

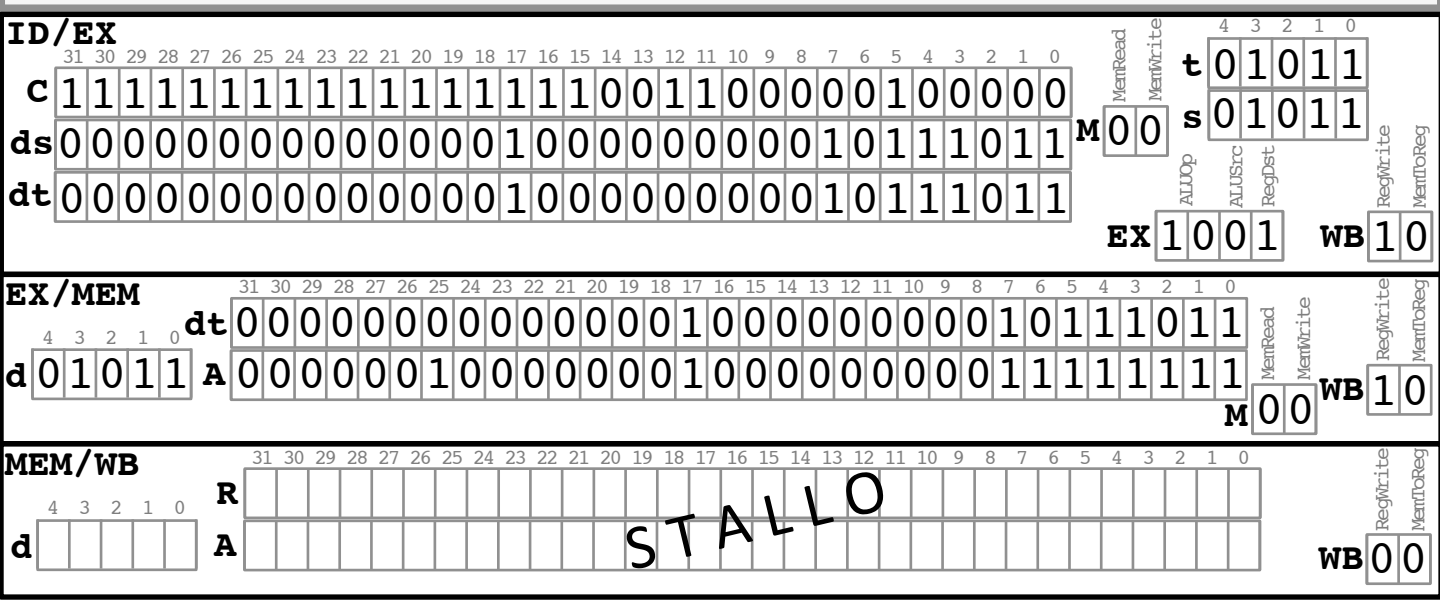
Assumendo un'esecuzione in pipeline secondo lo schema riportato nella pagina seguente (è quello visto a lezione) e che quando inizia l'esecuzione, al ciclo di clock 1, i contenuti dei registri sono `$s0 = 0x000100AA`, `$s1 = 0x010000CC`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.



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

```
sw $s0, 40(s1)           101011 10001 10000 0000000000101000
lw $t3, 40(s1)           100011 10001 01011 0000000000101000
or $t3, $s1, $t3         000000 10001 01011 01011 00000 100101
add $s3, $t3, $t3        000000 01011 01011 10011 00000 100000
```

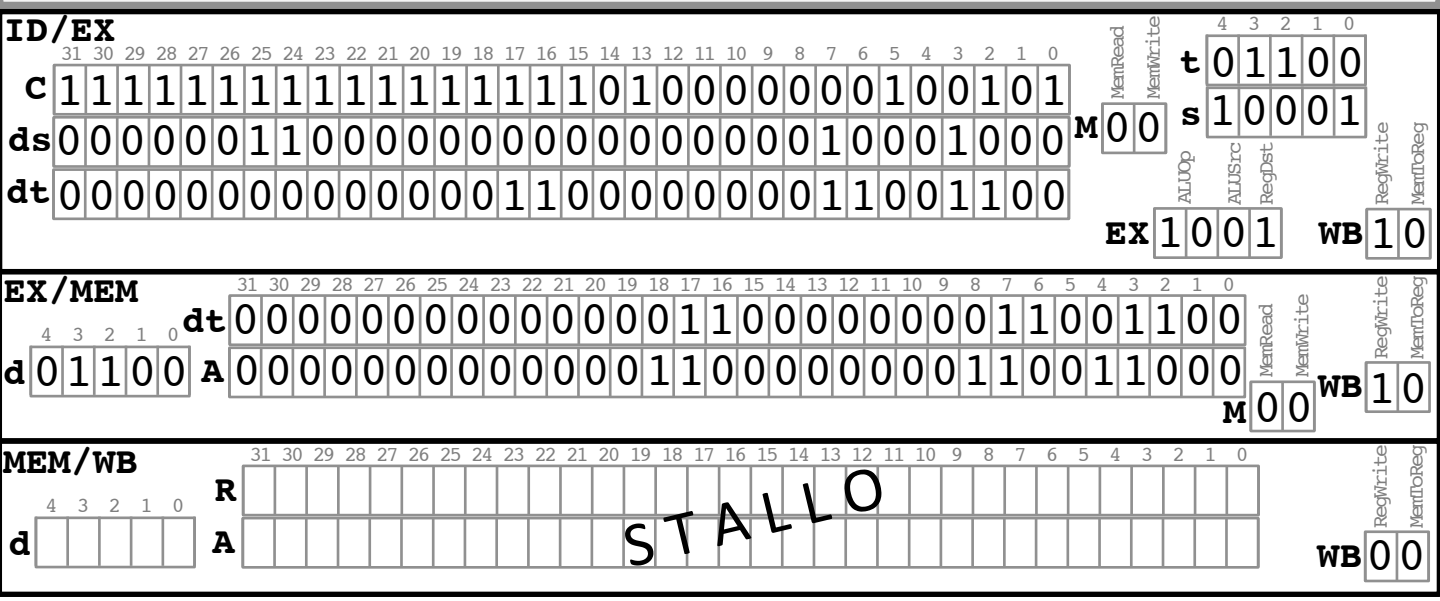
Assumendo un'esecuzione in pipeline secondo lo schema riportato nella pagina seguente (è quello visto a lezione) e che quando inizia l'esecuzione, al ciclo di clock 1, i contenuti dei registri sono `$s0 = 0x000200BB`, `$s1 = 0x02000044`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.



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

```
sw $s0, 40($s1)          101011 10001 10000 0000000000101000
lw $t4, 40($s1)          100011 10001 01100 0000000000101000
add $t4, $t4, $t4        000000 01100 01100 01100 00000 100000
or $s4, $s1, $t4         000000 10001 01100 10100 00000 100101
```

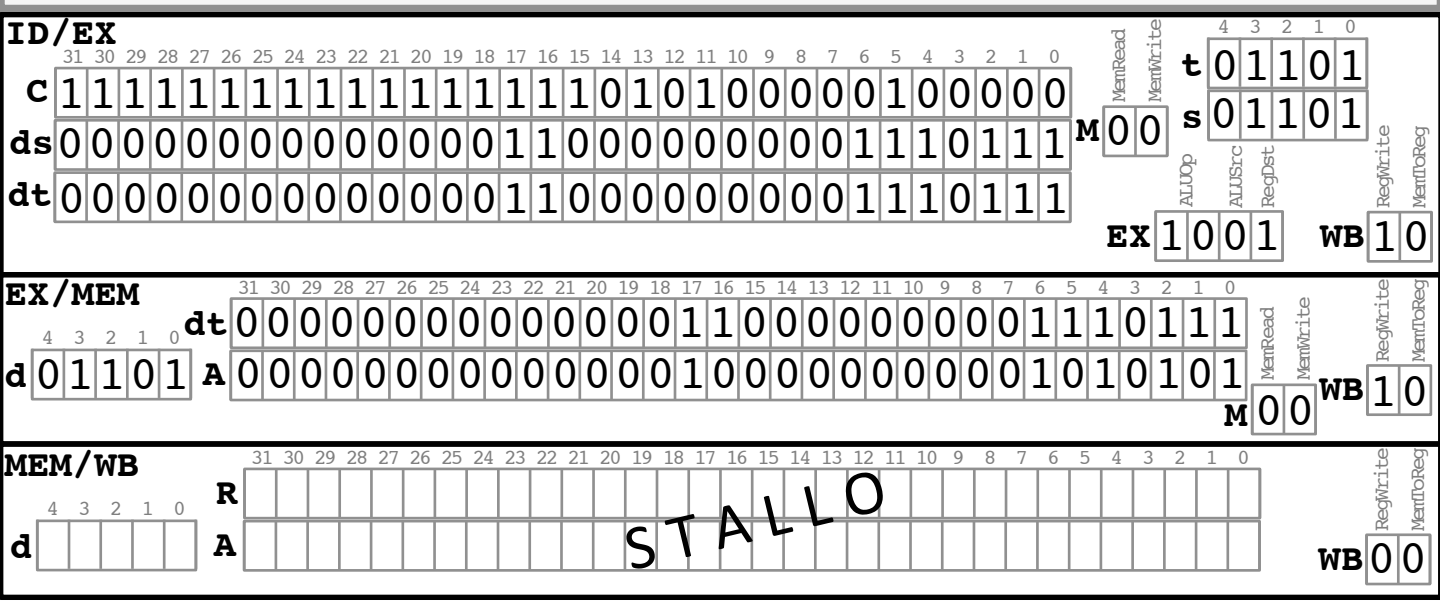
Assumendo un'esecuzione in pipeline secondo lo schema riportato nella pagina seguente (è quello visto a lezione) e che quando inizia l'esecuzione, al ciclo di clock 1, i contenuti dei registri sono `$s0 = 0x000300CC`, `$s1 = 0x03000088`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.



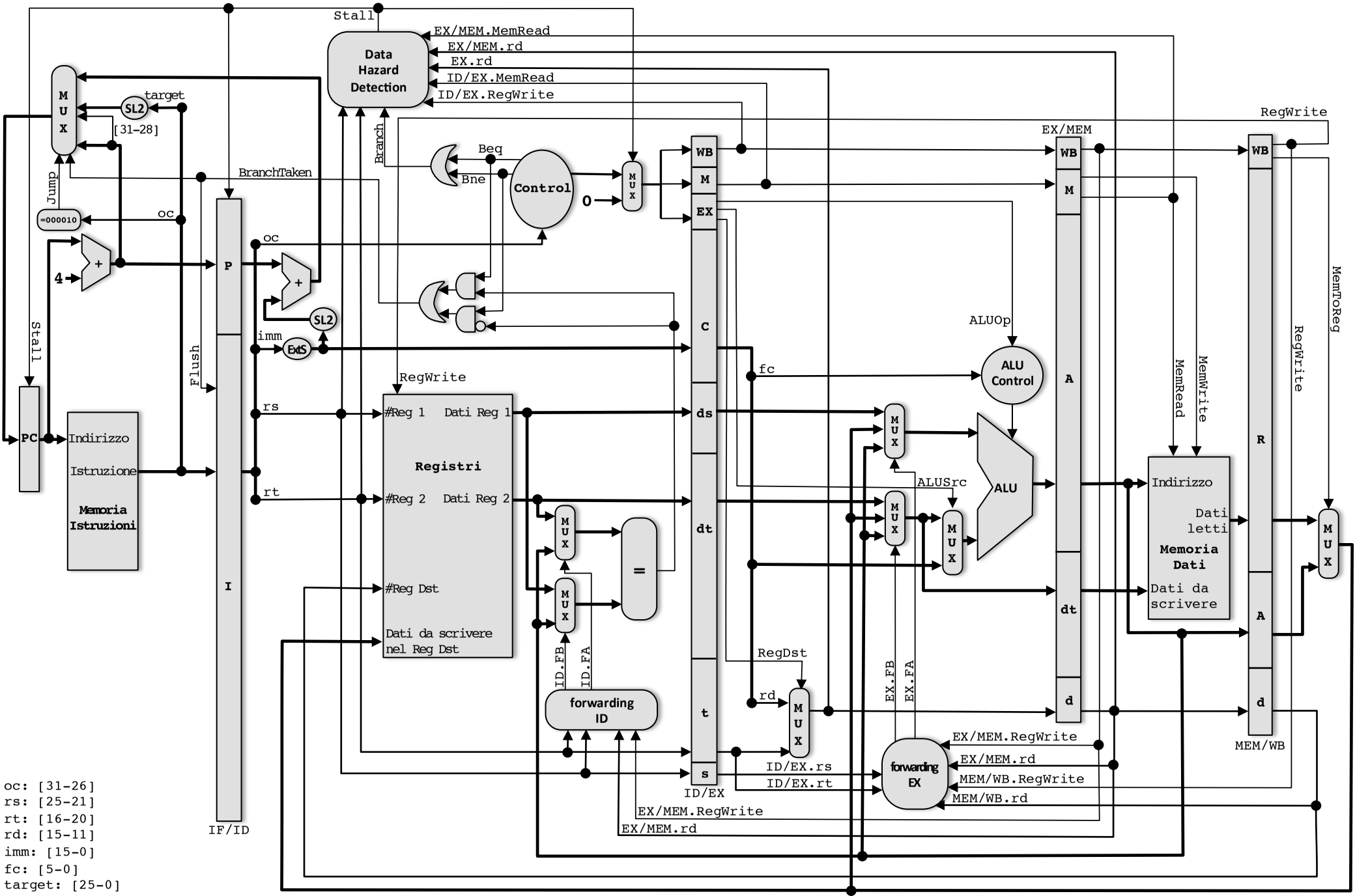
ESERCIZIO 1D Si consideri il seguente programma (accanto ad ogni istruzione è riportata la rappresentazione in binario):

```
sw $s0, 40($s1)          101011 10001 10000 0000000000101000
lw $t5, 40($s1)          100011 10001 01101 0000000000101000
and $t5, $s1, $t5        000000 10001 01101 01101 00000 100100
add $s5, $t5, $t5        000000 01101 01101 10101 00000 100000
```

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 `$s0 = 0x00030077`, `$s1 = 0x010200DD`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.



Implementazione pipeline di MIPS (solamente le istruzioni: `add`, `addi`, `sub`, `and`, `andi`, `or`, `ori`, `xor`, `xori`, `nor`, `slt`, `slti`, `lw`, `sw`, `beq`, `bne`, `j`).



oc: [31-26]
rs: [25-21]
rt: [16-20]
rd: [15-11]
imm: [15-0]
fc: [5-0]
target: [25-0]

ESERCIZIO 2 Determinare il numero di cicli di clock richiesti per eseguire il seguente programma, secondo lo schema di implementazione in pipeline riportato nella pagina seguente, e mostrare una traccia del procedimento usato:

```
         addi $t0, $zero, 12
start:   addi $t1, $s0, 12
loop:    lw $t2, arr($t1)        #arr è l'indirizzo di un array di words
         add $s1, $s1, $t2
         addi $t1, $t1, -4
         bne $t1, $s0, loop
         addi $t0, $t0, -1
         addi $s0, $s0, 128
         beq $t0, $zero, end
         j start
end:     sub $s1, $s1, $s2
```

SOLUZIONE ESERCIZIO 2

```
       5
  (12) 6                        (11) 31
       7       14      21              .
       9       16      23               .
      10       17      24                .
   (8)12   (4) 19  (0) 26                 .
                       27                  .
                       28                   .
                       29                    .
                       30                     55
L'i-esima iterazione termina al ciclo 5 + i*25. Le iterazioni sono 12 (12,11,...,1),
con l'ultima iterazione che termina con il branch che salta e quindi equivale a un
ciclo in più comunque. Quindi i cicli sono 5 + 12*25 + 1 = 306.
```

ESERCIZIO 3 Si consideri una cache a due livelli: L1 è 2-way con 8 insiemi, L2 è 4-way con 16 insiemi e strategia di rimpiazzo LRU. Entrambe le cache hanno blocchi da 32 words. Determinare per ognuno degli accessi alla memoria riportati nella tabella qui sotto (gli indirizzi sono a byte) gli hit e i miss su L1 e L2. Riportare le risposte nella tabella e mostrare il procedimento usato.

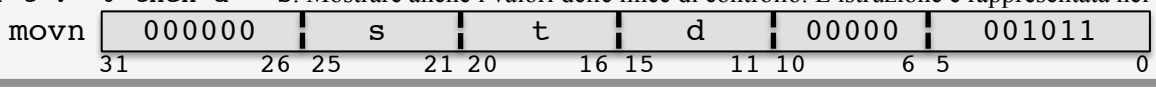
	1300	1400	3400	5400	1400	5500	7500	3428	7440	5380
L1 (h/m)	m	h	m	m	m	h	m	m	h	m
L2 (h/m)	m		m	m	h		m	h		h

ESERCIZIO 4 Consideriamo un sistema con memoria virtuale: pagine da 16KB, una TLB dati da 8 elementi e una cache dati 2-way set associative con 16 insiemi e blocchi da 256 words. Sia la TLB che la cache adottano una strategia di rimpiazzo LRU. Determinare per ognuno degli accessi riportati nella tabella qui sotto (gli indirizzi sono virtuali e a byte) gli hit/miss su TLB e cache e i page fault. Inizialmente la TLB e la cache sono vuote e nessuna pagina del processo è in memoria principale. Si assume che nessuna delle pagine caricate in memoria principale durante la serie di accessi venga rimpiazzata. Trascurare i miss sulla cache che accadono durante un page fault. Riportare le risposte nella tabella e mostrare il procedimento usato.

	1000	16500	1500	33000	512	2000	18800	34000	17000	32800
TLB (h/m)	m	m	h	m	h	h	h	h	h	h
Page Fault (s/n)	s	s	n	s	n	n	n	n	n	n
cache (h/m)	h	h	h	h	m	h	h	h	m	m

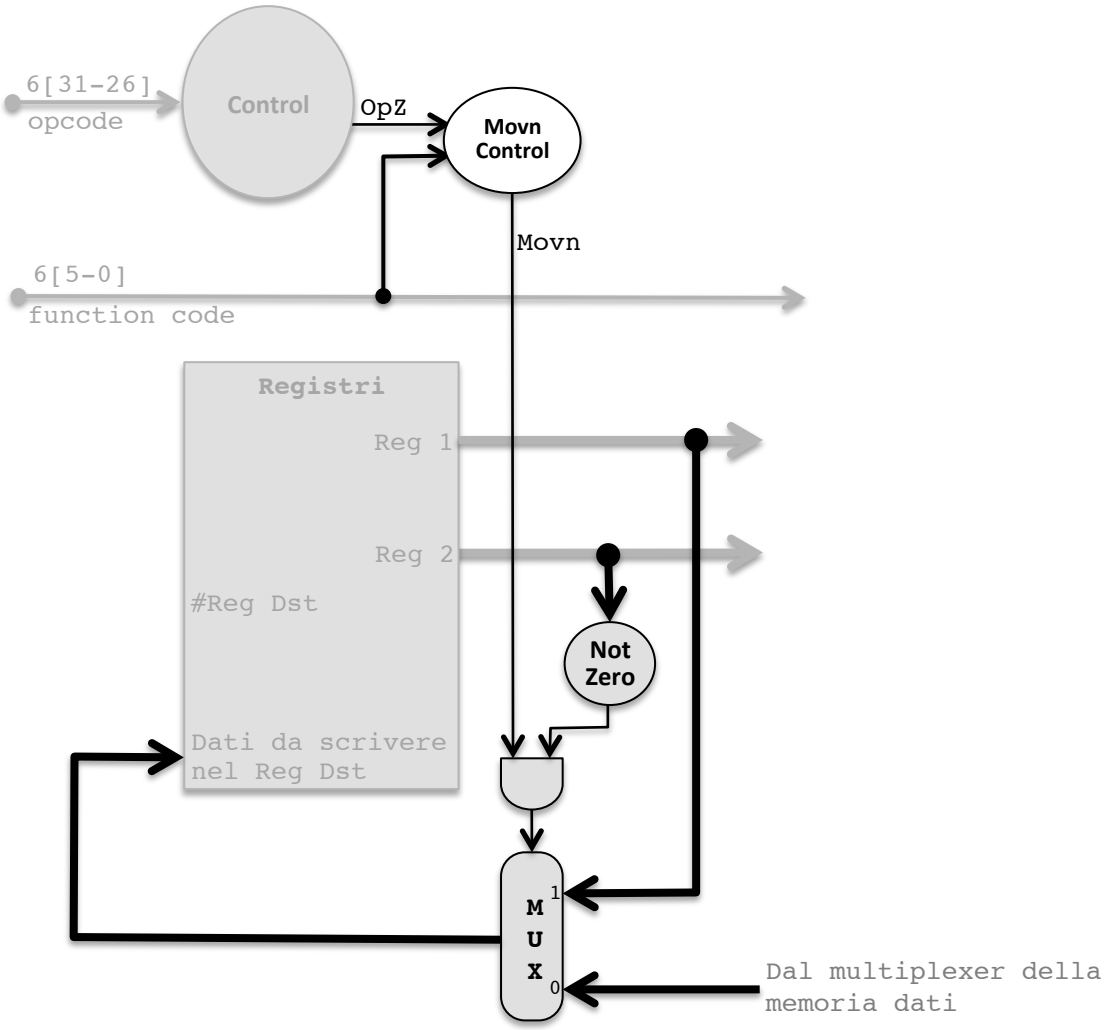
ESERCIZI DI RECUPERO

ESERCIZIO 5 Implementare, relativamente allo schema ad un ciclo di clock, la nuova istruzione `movn d, s, t`, che ha l'effetto: `if t != 0 then d = s`. Mostrare anche i valori delle linee di controllo. L'istruzione è rappresentata nel formato R:



SOLUZIONE ESERCIZIO 5

Aggiungiamo un nuovo elemento `NotZero` che determina se il valore in entrata è diverso da zero. L'input di `NotZero` è il valore del registro 2, cioè il registro `t`. Bisogna aggiungere un multiplexer in entrata a "`Dati da scrivere nel Reg Dst`" che prende come input 1 il valore del registro 1, cioè il registro `s`, ed è governato da un AND tra una nuova linea di controllo `Movn` e l'output di `NotZero`. La linea di controllo `Movn`, è determinata dal opcode (`000000`) e dal function code (`001011`). Risulta più facile introdurre una unità di controllo dedicata, `Movn Control`, che prende in input il function code e una linea `OpZ` dall'unità di controllo principale che è asserita se l'opcode è `000000`. L'output di `Movn Control` è la linea `Movn`.



	OpZ	**Movn**	Jump	Branch	MemToReg	MemWrite	MemRead	ALUOp	ALUSrc	RegWrite	RegDst
movn	1	1	0	0	0	0	0	XX	0	1	1

ESERCIZIO 6A Il seguente programma calcola nel registro `s0` la somma degli elementi dell'array `arr`, di 20 words, che sono o maggiori di 23 o minori di 11. Trovare le tre istruzioni mancanti:

```
            la $t0, arr
            la $t1, arr + 80
            li $s0, 0
loop:       beq $t0, $t1, end
            ?
            bgt $t2, 23, ok
            ?
            j next
ok:         add $s0, $s0, $t2
next:       addi $t0, $t0, 4
            ?
end:
```

`lw $t2, ($t0)`

`blt $t2, 11, ok`

`j loop`

ESERCIZIO 6B Il seguente programma calcola nel registro `s0` la somma degli elementi dell'array `arr`, di 20 words, che sono maggiori di 9 e minori di 25. Trovare le tre istruzioni mancanti:

```
            la $t1, arr
            la $t2, arr + 80
            li $s0, 0
loop:       ?
            bge $t3, 25, next
            ?
            add $s0, $s0, $t3
next:       addi $t1, $t1, 4
            ?
```

`lw $t3, ($t1)`

`ble $t3, 9, next`

`bne $t1, $t2, loop`

ESERCIZIO 6C Il seguente programma calcola nel registro `s0` il numero degli elementi dell'array `arr`, di 20 words, che sono o maggiori di 27 o minori di 8. Trovare le tre istruzioni mancanti:

```
            la $s1, arr
            la $s2, arr + 80
            li $s0, 0
loop:       ?
            bgt $s3, 27, count
            ?
            j next
count:      add $s0, $s0, 1
next:       addi $s1, $s1, 4
            ?
```

`lw $s3, ($s1)`

`blt $s3, 8, count`

`bne $s1, $s2, loop`

ESERCIZIO 6D Il seguente programma calcola nel registro `s0` il numero degli elementi dell'array `arr`, di 20 words, che sono minori di 29 e maggiori di 7. Trovare le tre istruzioni mancanti:

```
            la $t0, arr
            la $s1, arr + 80
            li $s0, 0
start:      beq $t0, $s1, end
            ?
            bge $t1, 29, next
            ?
            add $s0, $s0, 1
next:       addi $t0, $t0, 4
            ?
end:
```

`lw $t1, ($t0)`

`ble $t1, 7, next`

`j start`