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

```
sw $s0, -32($s1)          10101110001100001111111111100000
ori $s2, $s0, 0x00001C00  00110110000100100001110000000000
lw $s1, -32($s2)          10001110010100011111111111100000
addi $s2, $s1, -4         00100010001100101111111111111100
```

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 = 0x000800FF, $s1 = 0x00081CFF`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.

ID/EX

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ds	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1
dt	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1

	MemRead	MemWrite
M	0	0

	4	3	2	1	0
t	1	0	0	1	0
s	1	0	0	0	1

	ALUOp	ALUSrc	RegDst
EX	0	0	1	0

	RegWrite	MemToReg
WB	1	0

EX/MEM

	4	3	2	1	0
d					

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dt																																
A																																

STALLO

	MemRead	MemWrite
M	0	0

	RegWrite	MemToReg
WB	0	0

MEM/WB

	4	3	2	1	0
d	1	0	0	0	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
A	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	1	1	1

	RegWrite	MemToReg
WB	1	1

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

```
sw $t0, -64($t1)          10101101001010001111111111000000
andi $s3, $t0, 0x00000FFF 00110001000100110000111111111111
lw $t1, -64($s3)          10001110011010011111111111000000
addi $s3, $t1, -8         00100001001100111111111111111000
```

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 `$t0 = 0x00990AAA, $t1 = 0x00000AAA`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 riempendo lo schema qui sotto riportato.

ID/EX

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
ds	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0
dt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0

	MemRead	MemWrite
M	0	0

	4	3	2	1	0
t	1	0	0	1	1
s	0	1	0	0	1

	ALUOp	ALUSrc	RegDst
EX	0	0	1	0

	RegWrite	MemToReg
WB	1	0

EX/MEM

	4	3	2	1	0
d					

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dt																																
A																																

STALLO

	MemRead	MemWrite
M	0	0

	RegWrite	MemToReg
WB	0	0

MEM/WB

	4	3	2	1	0
d	0	1	0	0	1

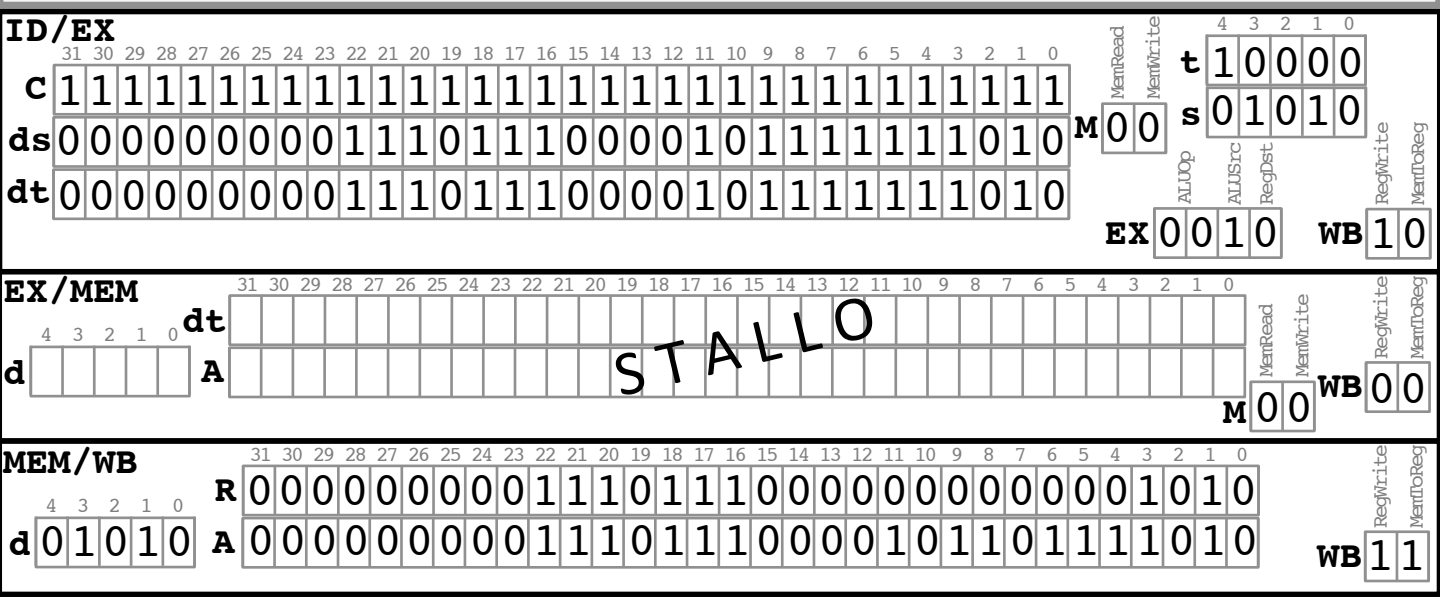
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	0	1	0

	RegWrite	MemToReg
WB	1	1

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

```
sw $t1, -128($t2)          10101101010010011111111110000000
ori $s0, $t1, 0x0BF0       00110101001100000000101111110000
lw $t2, -128($s0)          10001110000010101111111110000000
addi $s0, $t2, -1          00100001010100001111111111111111
```

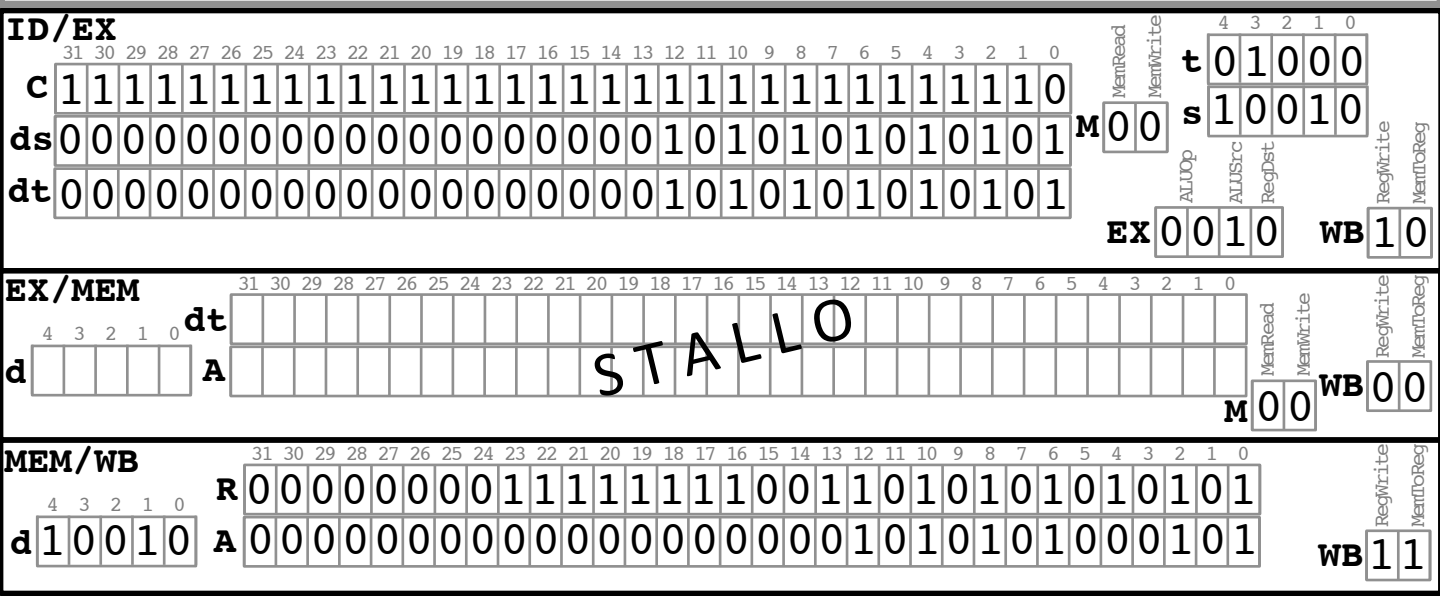
Assumendo un'esecuzione in pipeline secondo lo schema riportato nella pagina seguente (è quello visto a lezione) e che quando inizia l'esecuzione, al ciclo di clock 1, i contenuti dei registri sono `$t1 = 0x0077000A, $t2 = 0x00770BFA`, determinare il contenuto dei registri di pipeline alla fine del ciclo 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 $s1, -16($s2)           10101110010100011111111111110000
andi $t0, $s1, 0x1777      00110010001010000001011101110111
lw $s2, -16($t0)           10001101000100101111111111110000
addi $t0, $s2, -2          00100010010010001111111111111110
```

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 `$s1 = 0x00FF3555, $s2 = 0x00001555`, determinare il contenuto dei registri di pipeline alla fine del ciclo 6 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 riportato nella pagina seguente, e mostrare una traccia del procedimento usato:

```
        addi $s0, $zero, 11
        ori $t0, $zero, 1024
start:  addi $s0, $s0, -1
        addi $t0, $t0, -4
        sw $s0, 2048($t0)
        lw $s1, 2048($t0)
        bne $s1, $zero, start
        bne $s0, $s1, start
        lw $s1, 1048($t0)
        sub $s2, $s1, $t0
```

SOLUZIONE ESERCIZIO 2

```
     5
     6
(10) 7     (9) 15
     8         16
     9         17
     10        18
     13        21

5 + i*8  (10,9,8,...,0)    5 + 11*8 + 4 = 97
```

ESERCIZIO 3 Si consideri una cache a due livelli: L1 è 2-way con 16 insiemi, L2 è 4-way con 32 insiemi e strategia di rimpiazzo LRU. Entrambe le cache hanno blocchi da 16 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.

	1320	3340	1300	5400	3340	7460	3380	5380	1284	5420
L1 (h/m)	m	m	h	m	m	m	h	m	m	h
L2 (h/m)	m	m		m	h	m		h	h	

ESERCIZIO 4 Consideriamo un sistema con memoria virtuale: pagine da 4KB, una TLB dati da 8 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. 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.

	256	800	4200	1200	8400	800	5800	4800	9900	9000
TLB (h/m)	m	h	m	h	m	h	h	h	h	h
Page Fault (s/n)	s	n	s	n	s	n	n	n	n	n
cache (h/m)	h	h	h	h	h	m	h	m	h	m