Esercizio 1 (11 punti).

Sia data una CPU con processore a **2GHz** e **4 CPI** (Clock per Instruction) che adoperi indirizzi da 32 bit e memoria strutturata su due livelli di **cache** (L1, L2), il cui setup è come segue:
L1 è una cache **set-associativa** a **2 vie** con **4 set** e **blocchi da 2 word**; adopera una politica di rimpiazzo **LRU**. Ricordiamo che consideriamo il set come l'insieme del blocco in cache con tag e bit di validità, mentre la linea è il gruppo di set con il medesimo indice.
L2 è una cache **direct-mapped** con **4 linee** e **blocchi da 16 word**.

1) Supponendo che all'inizio nessuno dei dati sia in cache, indicare quali degli accessi in memoria indicati di seguito sono HIT o MISS in ciascuna delle due cache. Per ciascuna **MISS** indicare se sia di tipo Cold Start (**Cold**), Capacità (**Cap**) o Conflitto (**Conf**). Utilizzare la tabella sottostante per fornire i risultati ed indicare la metodologia di calcolo più in basso.

	Address	128	8	162	40	8000	192	12	10	220	130	160	480
L1	**Block#**												
	Index												
	Tag												
	HIT/MISS												
	Miss type												
L2	**Block#**												
	Index												
	Tag												
	HIT/MISS												
	Miss type												

2) Calcolare le dimensioni in bit (compresi i bit di controllo ed assumendo che ne basti uno per la LRU) delle due cache: (a) L1 e (b) L2.

3) Assumendo che gli accessi in **memoria** impieghino **400 ns**, che gli **hit** nella cache **L1** impieghino **1 ns** e gli **hit** nella cache **L2** impieghino **40 ns**, calcolare (a) il **tempo totale** per la sequenza di accessi, (b) il tempo **medio** per la sequenza di accessi, e (c) **quante istruzioni** vengono svolte nel tempo medio calcolato.

4) Calcolare il word offset del quinto indirizzo (8000) per la cache L1 spiegando i calcoli effettuati.

5) Considerando i tipi di miss che possono verificarsi durante l'uso della memoria virtuale, indicare se possa presentarsi il caso in cui si abbiano sia *cache hit* sia *page table miss*, spiegando brevemente il motivo. Si noti che questa è una domanda teorica non collegata alla domanda (1) di questo esercizio.

Esercizio 2 (11 punti).

Considerare l'architettura MIPS a ciclo singolo nella figura in basso (ed in allegato).
Si vuole aggiungere alla CPU l'istruzione vectorised jump and link safe (vjals), di tipo I e sintassi assembly
`vjals $indice, vettore`
che salta all'indirizzo contenuto nell'elemento $indice-esimo del vettore di word e salva il valore PC+4 in `$ra` (si ricorda che `$ra` è l'ultimo registro). Inoltre se il valore $indice è minore di zero, l'istruzione sceglierà il primo elemento (0-simo) del vettore.

Esempio 1: se in memoria si è definito staticamente il vettore:
`.word 15, 24, 313, 42`
e nel registro $t0 c'è il valore 3 allora
`vjals $t0, vettore`
salterà all'indirizzo 42 (che è l'elemento con indice 3 del vettore) e salverà il valore di PC+4 in $ra.

Esempio 2: Se invece nel registro `$t0` c'è il valore `-8` l'istruzione salterà all'indirizzo 15 (che è l'elemento con indice 0 del vettore) e salverà il valore di PC+4 in $ra.

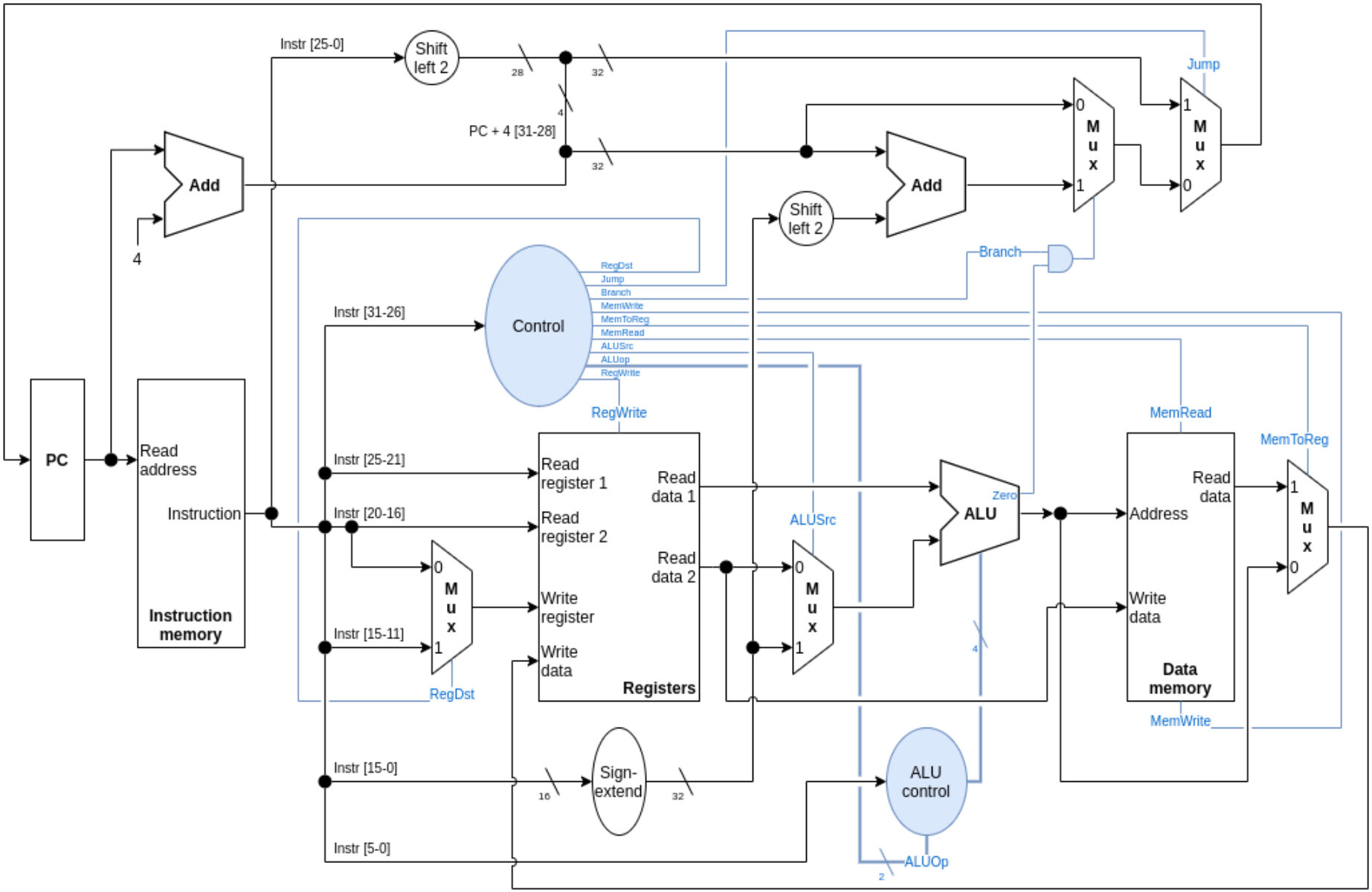
1) Mostrare le **modifiche all'architettura** della CPU MIPS, avendo cura di aggiungere eventuali altri componenti necessari a realizzare l'istruzione. A tal fine, si può alterare la stampa del diagramma architetturale oppure ridisegnare i componenti interessati dalla modifica, avendo cura di indicare i fili di collegamento e tutti i segnali entranti ed uscenti. Indicare inoltre sul diagramma i **segnali di controllo** che la CU genera *per realizzare l'istruzione*.

2) Indicare il contenuto in bit della word che esprime l'istruzione
`vjals` `$16, 0x0FFA`
compilando la tabella sottostante (assumiamo che lo *OpCode* di `vjals` sia `0x38`, e si ricorda che `$ra` è l'ultimo registro).

3) Supponendo che l'accesso alle **memorie** impieghi **75 ns**, l'accesso ai **registri 25 ns**, le operazioni dell'**ALU e** dei **sommatori 100 ns**, e che gli altri ritardi di propagazione dei segnali siano trascurabili, indicare la durata totale del **ciclo di clock** tale che anche l'esecuzione della nuova istruzione sia permessa *spiegando i calcoli effettuati*.

4) Indicando con `vjals` il segnale di controllo che viene asserito per eseguire la nuova istruzione, assumiamo che
a) tutti i segnali di tipo *don't care* siano pari a `0` e che
b) la Control Unit della CPU MIPS modificata per supportare `vjals` sia difettosa e sovrascriva il segnale `RegDst` come segue:
`RegDst ≔ vjals`
In tal caso, quale sarà il valore di $2 al termine dell'esecuzione del seguente frammento di codice, assumendo che `$s2` e il PC siano **inizializzati a `0`**? Spiegare il motivo.

```
addi    $2, $0, -1
vjals   $2, 0x0100
```



Esercizio 3 (11 punti)

Si consideri l'architettura MIPS con pipeline mostrata nella figura in basso (ed in allegato). Il programma qui di seguito (fornito sia in formato testo che come immagine) effettua le operazioni di somma o sottrazione indicate nella stringa fornita in input.

```
1  .globl main
2  .data
3  W: .asciiz "+2-3+4+2"
4
5  .text
6
7  main:
8          and $t1, $t0, $t0
9          and $t0, $t0, $zero
10         la $a0, W
11         or $v0, $zero, $zero
12 Cycle:  lb $t1, 0($a0)
13         beq $t1, $zero, Exit
14         lb $t0, 1($a0)
15         sub $t0, $t0, '0'
16         bne $t1, '-', Add
17         sub $t0, $zero, $t0
18 Add:    add $v0, $v0, $t0
19         addi $a0, $a0, 2
20         j Cycle
21 Exit:   move $a0, $v0
22         li $v0, 1
23         syscall
24         li $v0, 10
25         syscall
```

```
.globl main

.data
W: .asciiz "+2-3+4+2"

.text
main:
        and  $t0, $t0, $zero
        and  $t1, $t0, $t0
        la   $a0, W
        or   $v0, $zero, $zero
Cycle:  lb   $t1, 0($a0)
        beq  $t1, $zero, Exit
        lb   $t0, 1($a0)
        subi $t0, $t0, '0'
        bne  $t1, '-', Add
        sub  $t0, $zero, $t0
Add:    add  $v0, $v0, $t0
        addi $a0, $a0, 2
        j    Cycle
Exit:   move $a0, $v0
        li   $v0, 1
        syscall
        li   $v0, 10
        syscall
```

Si supponga che *tutte le istruzioni impiegate fanno parte del set supportato dalla CPU in figura*, ossia non si fa uso di alcuna pseudoistruzione.

Si indichino (ignorando hazard che possano concernere la syscall):

1) le istruzioni tra le quali sono presenti **data hazard** – indicare i numeri di istruzione N ed M (visibili alla sinistra delle linee di codice in figura) ed il registro coinvolto $xY (nel formato N / M / $xY);

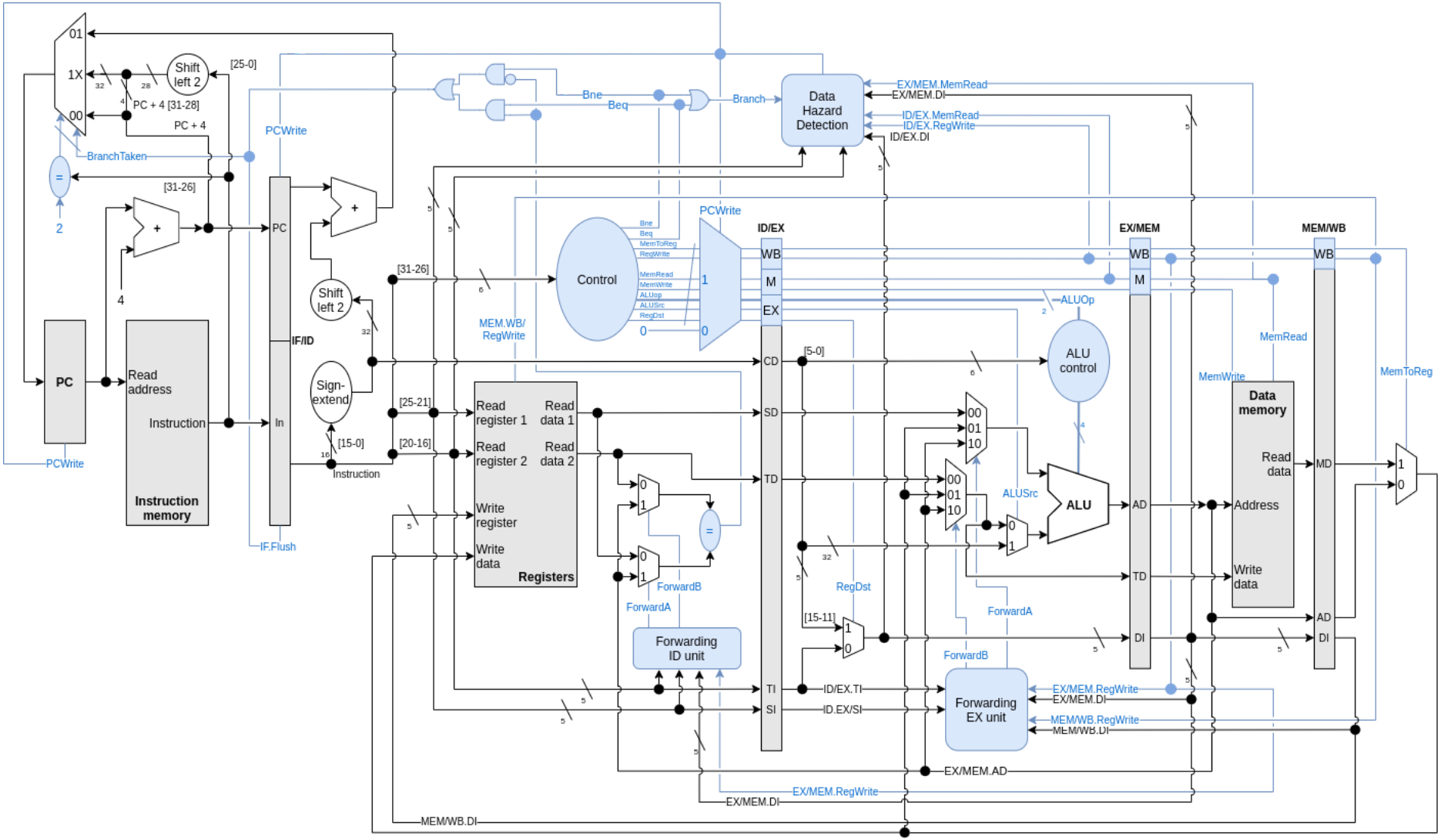
2) le istruzioni tra le quali sono presenti **control hazard** – indicare i numeri di istruzione N ed M (nel formato N / M);

3) quanti **cicli di clock** sono necessari ad eseguire il programma tramite **forwarding**, spiegando il calcolo effettuato;

4) quanti **cicli di clock** sarebbero necessari ad eseguire il programma **senza forwarding**, spiegando il calcolo effettuato;

5) quali sono, per ognuna delle cinque fasi, le **istruzioni** (o le bolle) in pipeline durante il **14° ciclo di clock** (con **forwarding**);
   IF:
   ID:
   EX:
   MEM:
   WB:

6) qualora la decisione delle istruzioni di jump sia compiuta in fase ID, quante fasi di stallo causi potenzialmente ogni control hazard. Chiarire se questo cambiamento abbia altri effetti.


01
1X
00
Shift left 2
[25-0]
32
28
4
PC + 4 [31-28]
PC + 4
BranchTaken
=
2
[31-26]
+
4
PC
PCWrite
IF.Flush
IF/ID
In
Instruction
PC
Read address
Instruction
Instruction memory
PCWrite
Shift left 2
Sign-extend
32
[15-0]
16
[31-26]
[25-21]
[20-16]
Bne
Beq
Branch
Data Hazard Detection
EX/MEM.MemRead
EX/MEM.DI
ID/EX.MemRead
ID/EX.RegWrite
ID/EX.DI
PCWrite
Control
Bne
Beq
MemToReg
RegWrite
MemRead
MemWrite
ALUop
ALUSrc
RegDst
1
0
ID/EX
WB
M
EX
MEM.WB/ RegWrite
CD
SD
TD
TI
SI
[5-0]
ALUOp
ALU control
EX/MEM
MemRead
MemWrite
MemToReg
Data memory
Read data
Address
Write data
MD
AD
DI
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
00
01
10
ALUSrc
ALU
RegDst
[15-11]
ForwardA
ForwardB
Forwarding ID unit
Forwarding EX unit
ID/EX.TI
ID.EX/SI
EX/MEM.RegWrite
EX/MEM.DI
MEM/WB.RegWrite
MEM/WB.DI
EX/MEM.AD
EX/MEM.RegWrite
EX/MEM.DI
MEM/WB.DI