Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito A

Cognome e Nome: _____ Matricola: _____

Parte 1 (per chi non ha superato l'esonero – 1 ora)

Esercizio 1A. Si ha il dubbio che in una partita di CPU a ciclo di clock singolo (vedi sul retro) la Control Unit sia rotta, producendo il segnale di controllo **RegWrite** attivo **anche quando** è attivo il segnale di controllo **MemWrite**.

a) Si indichino *qui sotto* quali delle istruzioni base (**lw, sw, add, sub, and, or, xor, slt, beq, j**) funzioneranno male e perché.

b) si scriva *qui sotto* un breve programma assembly MIPS (senza pseudoistruzioni) che termina valorizzando il registro $s0 con il valore 1 se il processore è guasto, altrimenti con 0. Si assume che RegDst sia asserito solo per le istruzioni di tipo R e che MemToReg sia asserito solo per l'istruzione lw.

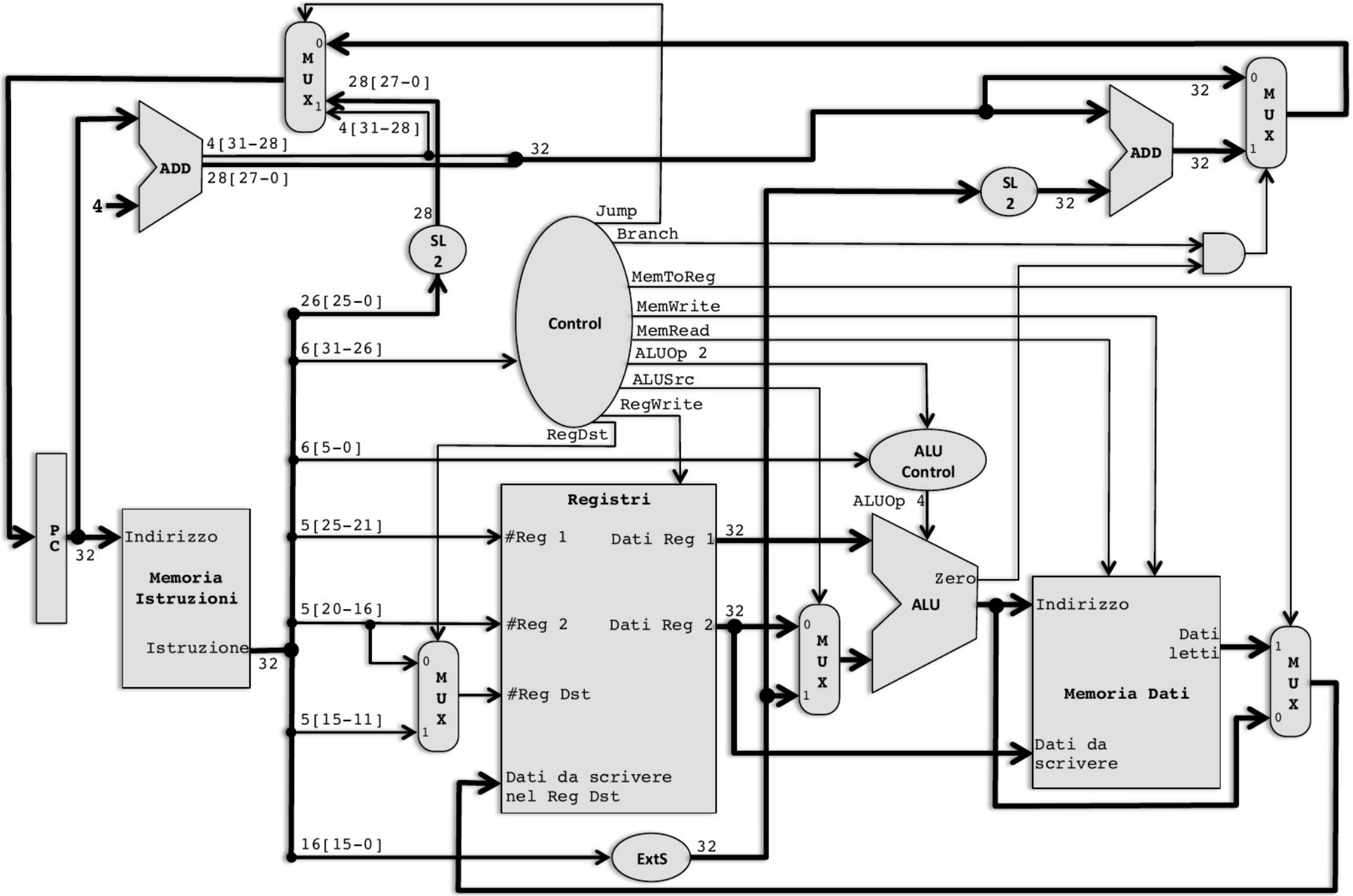
Esercizio 2A. Considerate l'architettura MIPS a ciclo singolo in figura (diagramma sul retro).
Si vuole aggiungere l'istruzione di tipo I `bali rd, label` (branch and link immediato) che salta all'indirizzo della label (relativo come nei beq) e salva nel registro **rd** l'indirizzo della prossima istruzione.

1) modificate il diagramma mostrando gli eventuali altri componenti necessari a realizzare l'istruzione

2) indicate sul diagramma i segnali di controllo necessari a realizzare l'istruzione

3) supponendo che l'accesso alle memorie impieghi 133ns, l'accesso ai registri 66ns, le operazioni dell'ALU 200ns, e ignorando gli altri ritardi di propagazione dei segnali, indicate sul diagramma la durata totale del ciclo di clock per permettere l'esecuzione anche della nuova istruzione.

Implementazione ad un ciclo di clock di MIPS (solamente le istruzioni: `add`, `sub`, `and`, `or`, `xor`, `slt`, `lw`, `sw`, `beq`, `j`)



Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito A

Cognome e Nome: _____ Matricola: _____

Parte 2 (per tutti – 2 ore)

Esercizio 3A (16 punti). Considerate l'architettura MIPS con pipeline mostrata in figura (sul retro) ed il frammento di programma qui a destra che somma gli elementi a coordinate x+y pari di una matrice di word.
Indicate:
1) tra quali istruzioni sono presenti data hazard,
2) tra quali istruzioni sono presenti control hazard,
3) tra quali istruzioni sono necessari stalli (data e control)
4) quanti cicli di clock sono necessari a eseguire il programma
5) quanti ne sarebbero necessari se il forwarding non esistesse
6) riordinate le istruzioni per ridurre al massimo gli stalli (mantenendo invariata la sua semantica)
7) calcolate quanti cicli di clock sono necessari a eseguire il programma così ottimizzato

```
.globl main
.data
matrice:    .space 400 # 100 words
DIM:        .word 10  #lato della matrice
.text
main:       xor  $s4, $s4, $s4 # somma=0
            xor  $s2, $s2, $s2 # x=0
            xor  $s0, $s0, $s0 # y=0
            lw   $s1, DIM
loopY:      beq  $s0, $s1, end
loopX:      bge  $s2, $s1, nextY
            mul  $t1, $s0, $s1    # y*DIM
            add  $t1, $t1, $s2    # +x
            muli $t1, $t1, 4      # *4
            lw   $t2, matrice($t1)
            add  $s4, $s4, $t2
            addi $s2, $s2, 2      # x+=2
            j loopX
nextY:      andi $s2, $s0, 0x0001 # x%=2
            addi $s0, $s0, 1      # y+=1
            # qui x=1 se y pari, else x=0
            j loopY
end:        li $v0, 1    # per la syscall
            add $a0, $s4, $zero   # move
```

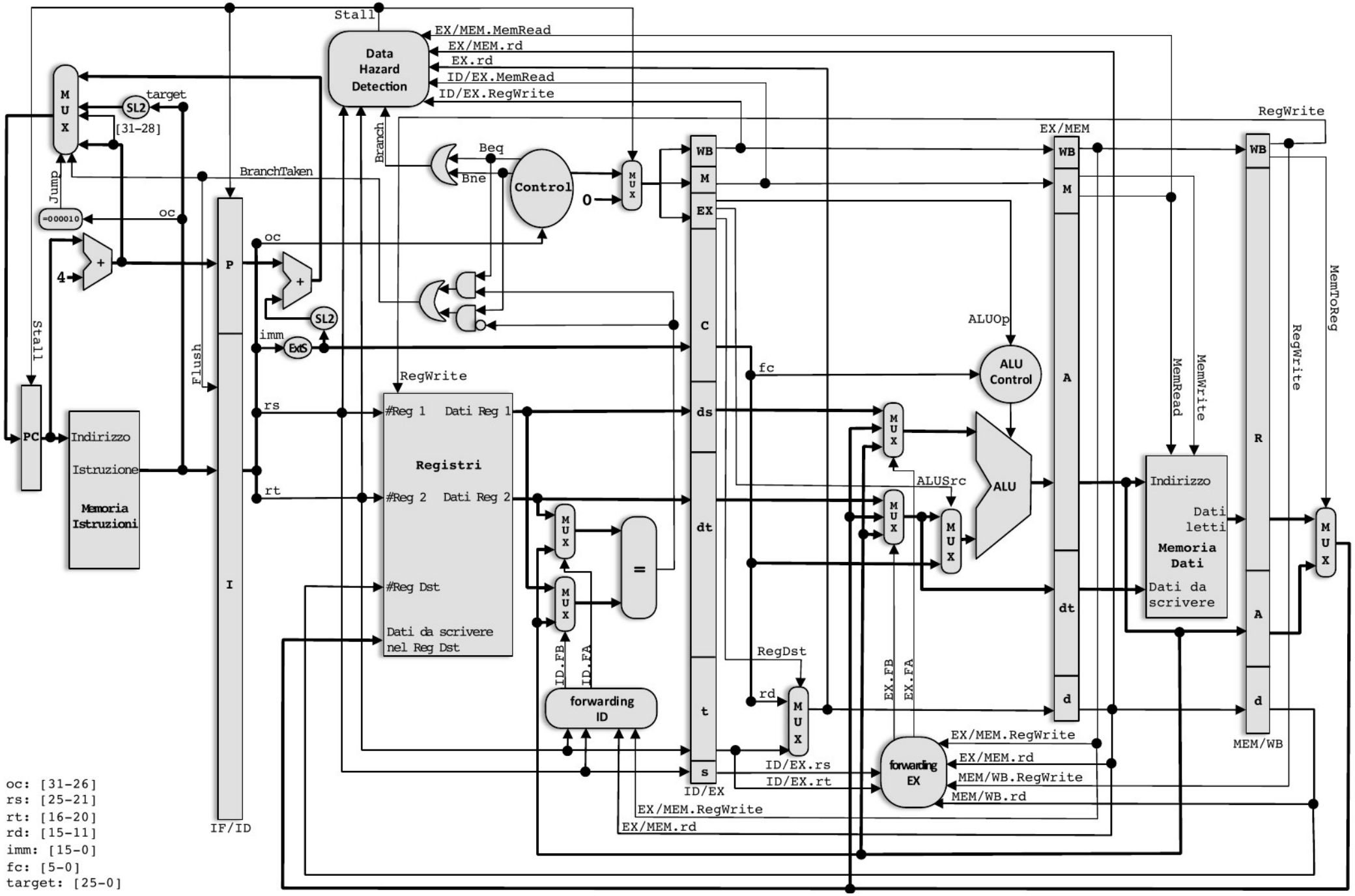
Esercizio 4A (14 punti). Si consideri un sistema dotato di due livelli di cache: CPU <=> L1 <=> L2.
- L1 è una cache 4-way set-associative con 4 linee e blocchi grandi 4 word e strategia di rimpiazzo LRU.
- L2 è una cache 2-way set-associative con 8 linee e blocchi grandi 16 word e strategia di rimpiazzo LRU.

1) Supponendo che gli indirizzi siano da 32 bit (indirizzamento al byte) e che all'inizio nessuno dei dati sia in memoria, indicate quali dei seguenti accessi in memoria sono hit o miss in ciascuna delle due cache:

10 28 200 540 330 210 220 535 190 560 30 36 12 520

2) calcolate le dimensioni in bit delle due cache L1, L2.
3) assumendo che il processore vada a 1Ghz con 1 CPI (Clock Per Instruction), che gli accessi in memoria impieghino 100ns, che gli hit nella cache L1 impieghino 1ns e gli hit nella cache L2 impieghino 20ns, calcolate il tempo totale e il CPI per questa sequenza di accessi.

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]

Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito B

Cognome e Nome: _____ Matricola: _____

Parte 1 (per chi non ha superato l'esonero – 1 ora)

Esercizio 1B. Si ha il dubbio che in una partita di CPU MIPS come quella in figura la Control Unit sia rotta, producendo il segnale di controllo **MemToReg** attivo **se e solo se** è attivo il segnale di controllo **MemWrite**.

a) Si indichino *qui sotto* quali delle istruzioni base (**lw, sw, add, sub, and, or, xor, slt, beq, j**) funzioneranno male e perché.

b) si scriva *qui sotto* un breve programma assembly MIPS (senza pseudoistruzioni) che termina valorizzando il registro $s0 con il valore 1 se il processore è guasto, altrimenti con 0. Si assume che RegDst sia asserito solo per le istruzioni di tipo R e che MemtoReg sia asserito solo per l'istruzione lw.

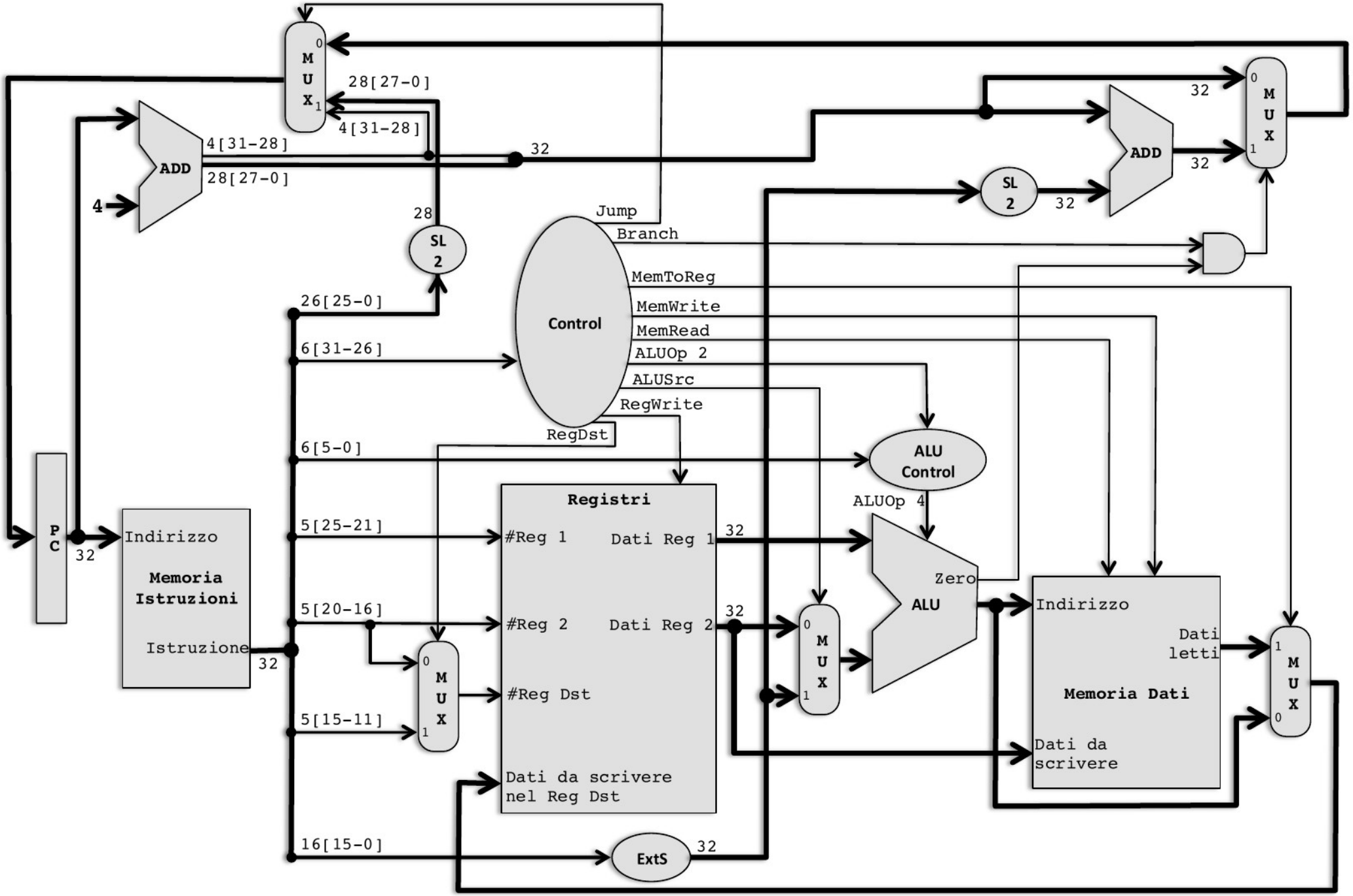
Esercizio 2B. Considerate l'architettura MIPS a ciclo singolo in figura (sul retro). Si vuole aggiungere l'istruzione di tipo I **`baddz rs, rt, label`** che incrementa il registro **`rs`** del valore contenuto nel registro **`rt`** (rs=rs+rt) e salta alla destinazione se la somma è zero.

1) modificate il diagramma mostrando gli eventuali altri componenti necessari a realizzare l'istruzione

2) indicate sul diagramma i segnali di controllo necessari a realizzare l'istruzione

3) supponendo che l'accesso alle memorie impieghi 100ns, l'accesso ai registri 50ns, le operazioni dell'ALU 150ns, e ignorando gli altri ritardi di propagazione dei segnali, indicate sul diagramma qual è la durata del ciclo di clock per permettere l'esecuzione anche della nuova istruzione.

Implementazione ad un ciclo di clock di MIPS (solamente le istruzioni: `add`, `sub`, `and`, `or`, `xor`, `slt`, `lw`, `sw`, `beq`, `j`)



Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito B

Cognome e Nome: _____ Matricola: _____

Parte 2 (per tutti – 2 ore)

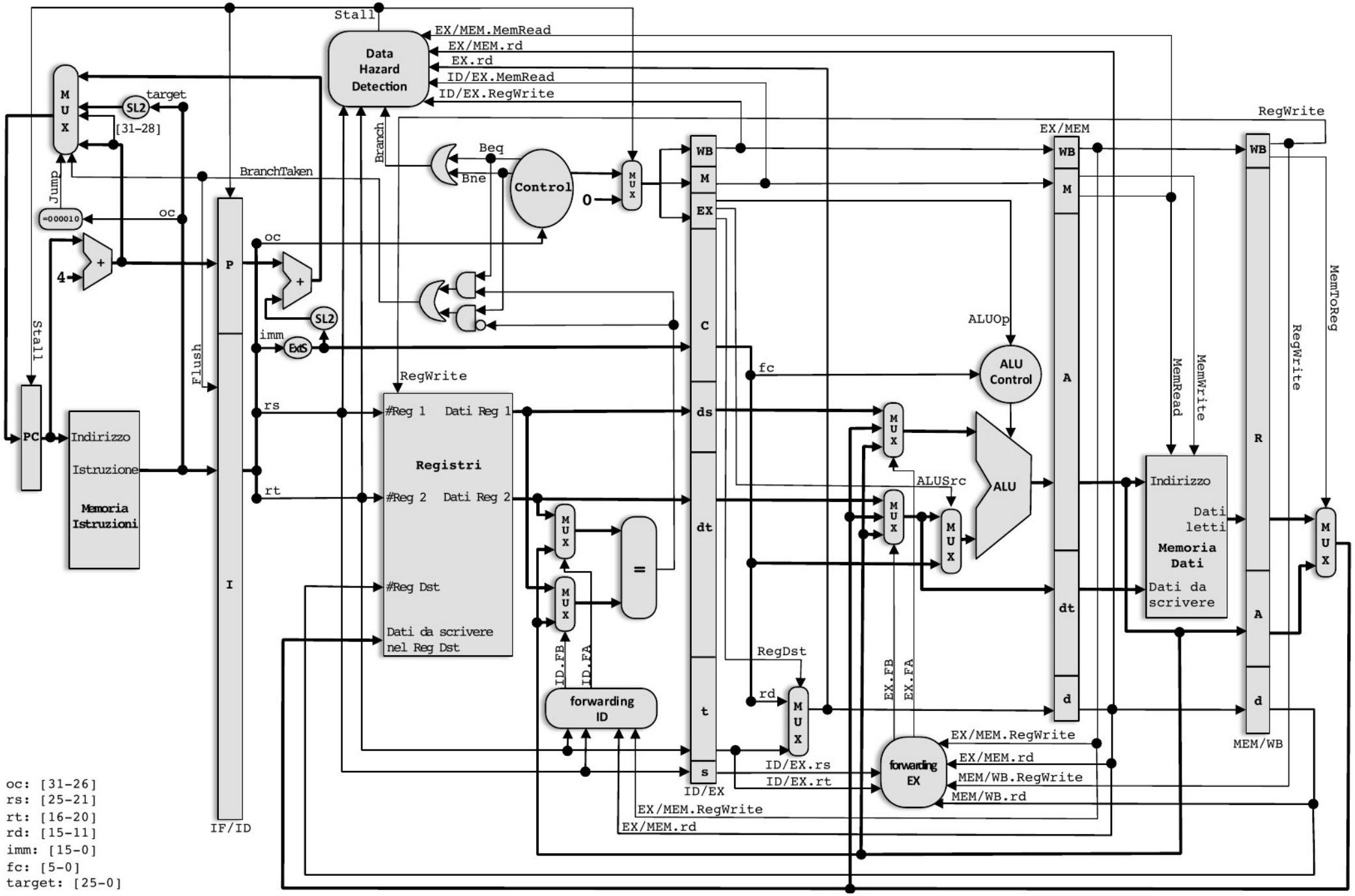
Esercizio 3B (16 punti). Si consideri l'architettura MIPS con pipeline mostrata in figura (sul retro) ed il frammento di programma qui a destra che calcola la somma degli elementi dispari di un vettore di 100 elementi.
Si indichino:
1) tra quali istruzioni sono presenti data hazard,
2) tra quali istruzioni sono presenti control hazard,
3) tra quali istruzioni sono necessari stalli (data e control)
4) quanti cicli di clock sono necessari a eseguire il programma
5) quanti ne sarebbero necessari se il forwarding non esistesse
6) riordinate le istruzioni per ridurre al massimo gli stalli (mantenendo invariata la sua semantica)
7) quanti cicli di clock sono necessari a eseguire il programma così ottimizzato

```
.globl   main
.data
vettore: .word ... (100 valori) ...
DIM:     .word 100
.text
main:    xor  $s2, $s2, $s2 # somma=0
         lw   $s0, DIM      # i=100
         subi $s0, $s0, 1   # i= 99
         muli $s0, $s0, 4   # i=396
loop:    lw   $t1, matrice($s0)
         andi $t2, $t1, 1   # val%2
         beq  $t2, $zero, pari
         add  $s2, $s2, $t1 # dispari
pari:    subi $s0, $s0, 4   # el.prec.
         blez $s0, end      # se i<0
         j loop
end:     add  $a0, $s2, $zero # move
         li   $v0, 1   # args syscall
```

Esercizio 4B (14 punti). Si consideri un sistema dotato di due livelli di cache: CPU <=> L1 <=> L2.
- L1 è una cache 2-way set-associative con 16 linee e blocchi grandi 2 word e strategia di rimpiazzo LRU.
- L2 è una cache 1-way set-associative con 4 linee e blocchi grandi 8 word e strategia di rimpiazzo LRU.

1) Supponendo che gli indirizzi siano da 32 bit (indirizzamento al byte) e che all'inizio nessuno dei dati sia in memoria, indicate quali dei seguenti accessi in memoria sono hit o miss in ciascuna delle due cache:
10 28 200 540 330 210 220 535 190 560 30 36 12 520
2) calcolate le dimensioni in bit delle due cache L1, L2.
3) assumendo che il processore vada a 2Ghz con 2 CPI (Clock Per Instruction), che gli accessi in memoria impieghino 150ns, che gli hit nella cache L1 impieghino 1ns e gli hit nella cache L2 impieghino 25ns, calcolate il tempo totale impiegato e il CPI per questa sequenza di accessi.

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]

Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito A

Parte 3 (assembler)

Esercizio 5A.

1) Si realizzi la funzione **ricorsiva** che calcola il massimo comun divisore (MCD) con l'algoritmo di Eulero basato sulle differenze definita qui a destra:

$$MCD(x,y)=\begin{cases} x & \text{se x = y} \\ MCD(x-y,y) & \text{se x > y} \\ MCD(x,y-x) & \text{se x < y} \end{cases}$$

2) Si realizzi un programma che usa la funzione MCD per calcolare il minimo comune multiplo (mcm) tra 2 valori positivi:

- legge due valori interi positivi X, Y

- calcola e stampa il minimo comune multiplo tra X, e Y, ovvero mcm(X,Y) = X * Y / MCD(X, Y)

Esempi del MCD:

se X=2 e Y=3 il risultato è MCD(2,3) = MCD(2,1) = MCD(1,1) = **1**

se X=30 e Y=20 il risultato è MCD(30,20) = MCD(10,20) = MCD(10,10) = **10**

se X=15 e Y=81 il risultato è MCD(15,81) = MCD(15,66) = MCD(15,51) = MCD(15,36) =
= MCD(15,21) = MCD(15,6) = MCD(9,6) = MCD(3,6) = MCD(3,3) = **3**

Esempi dell'output:

se X=27, Y=15 il risultato è mcm(27,15) = 27 * 15 / MCD(27, 15) = 27 * 15 / 3 = **135**

se X=54, Y=30 il risultato è mcm(54,30) = 54 * 30 / MCD(54, 30) = 54 * 30 / 6 = **270**

se X=54, Y=72 il risultato è mcm(54,72) = 54 * 72 / MCD(54, 72) = 54 * 72 / 18 = **216**

Esame di Architetture – Canale AL – Prof. Sterbini – 8/7/13 – Compito B

Parte 3 (assembler – 1 ora)

Esercizio 5B.

1) Si realizzi la funzione assembler **ricorsiva** che calcola il massimo comun divisore (GCD) di tre numeri X, Y, Z usando l'algoritmo di Eulero basato sui resti, definita qui a destra (si esegue la prima condizione valida)

$$GCD(x,y)=\begin{cases} x & \text{se y è zero} \\ GCD(y,x) & \text{se x < y} \\ GCD(y,x \bmod y) & \text{altrimenti} \end{cases}$$

2) si realizzi il programma assembler che usa la funzione GCD per calcolare il GCD di 3 valori positivi, e che:

- legge tre valori interi positivi X, Y, Z

- calcola e stampa il minimo comune multiplo tra X, Y e Z, ovvero GCD(GCD(X, Y), Z)

Esempi del GCD:

se X=2 e Y=3 il risultato è GCD(2,3) = GCD(3,2) = GCD(3,1) = GCD(1,0) = **1**

se X=30 e Y=20 il risultato è GCD(30,20) = GCD(20,10) = GCD(10,0) = **10**

se X=15 e Y=81 il risultato è GCD(15,81)=GCD(81,15)=GCD(15,6)=GCD(6,3)=GCD(3,0)=**3**

Esempi dell'output:

se X=27, Y=150 e Z=600 il risultato è GCD(GCD(27,150),600) = GCD(3,600) = **3**

se X=54, Y=150 e Z=600 il risultato è GCD(GCD(54,150),600) = GCD(6,600) = **6**

se X=54, Y=90 e Z=630 il risultato è GCD(GCD(54,90),630) = GCD(18,630) = **18**