Esame di Architetture – Canale MZ – Prof. Sterbini – 30/6/14

Cognome e Nome: _____ Matricola: _____

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

Esercizio 1 (14 punti). 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.

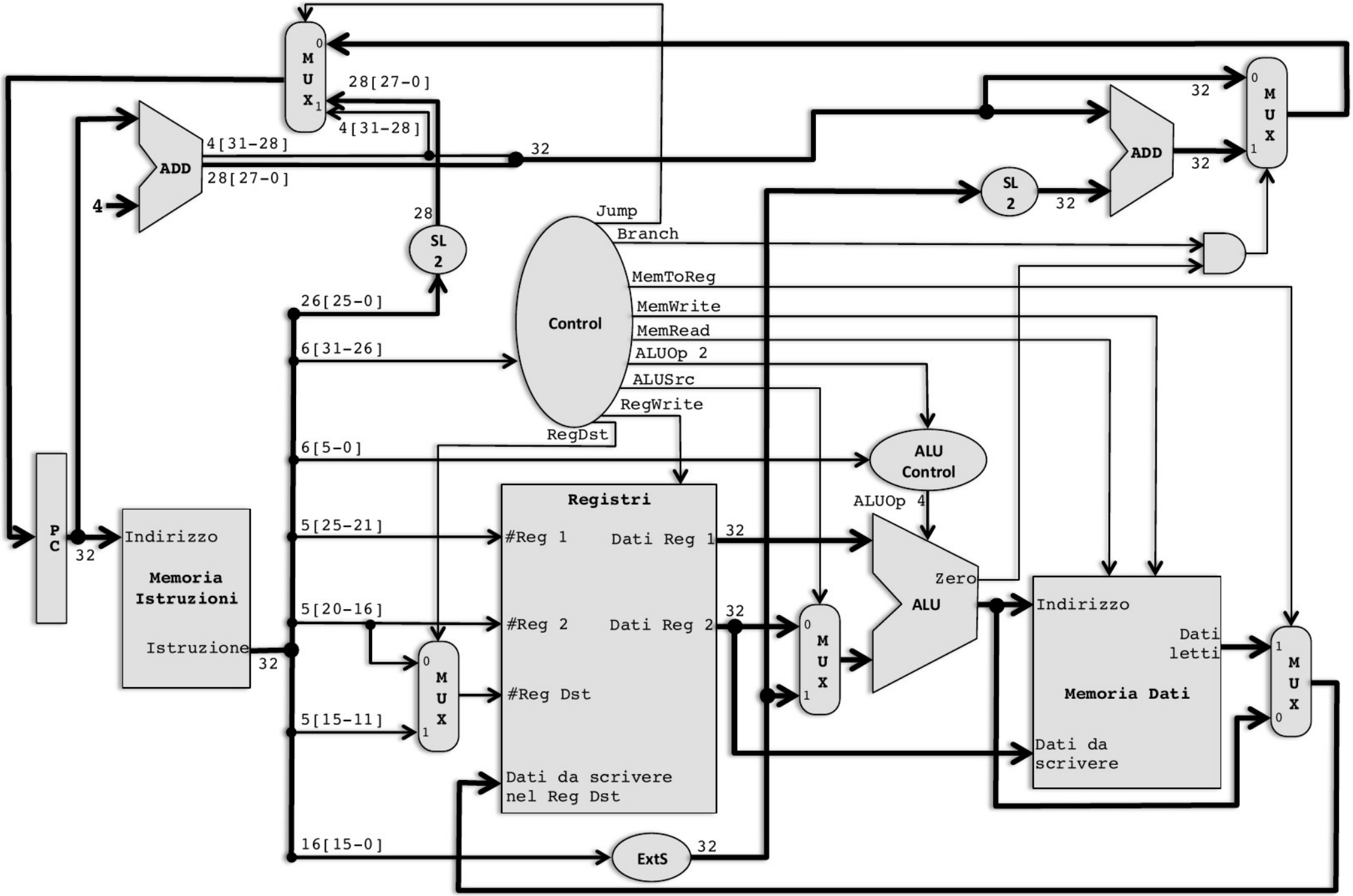
Esercizio 2 (16 punti). Considerate l'architettura MIPS a ciclo singolo in figura (diagramma sul retro). Vogliamo aggiungere l'istruzione di tipo I `indj offset(rs)` (indirect jump) che salta incondizionatamente all'indirizzo contenuto nella memoria nella posizione `offset(rs)`, ovvero corrisponde a svolgere le due istruzioni `lw $at offset(rs) ; jr $at`

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 **150ns**, l'accesso ai registri **75ns**, 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 della nuova istruzione e se la durata totale del ciclo di clock necessario è aumentata rispetto alla CPU senza la 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 MZ – Prof. Sterbini – 30/6/14 – Compito A

Cognome e Nome: _____ Matricola: _____

Parte 2 (per tutti – 2 ore)

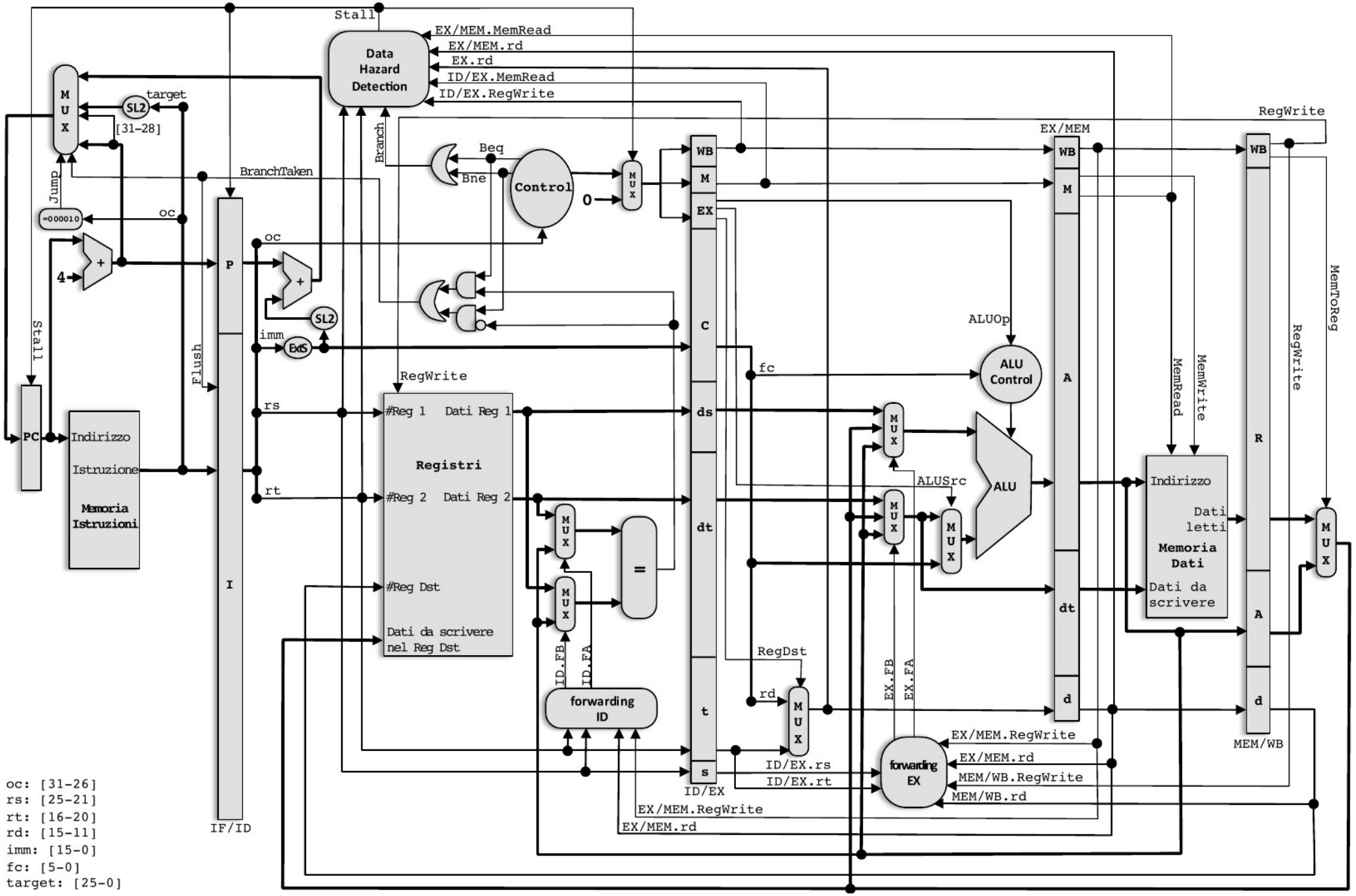
Esercizio 3 (16 punti). Considerate l'architettura MIPS con pipeline mostrata in figura (sul retro) ed il frammento di programma qui a destra che copia in un vettore gli elementi dispari di un vettore

NOTA: NEL VETTORE CI SONO <u>30 NUMERI DISPARI e 70 NUMERI PARI</u> (l'ordine non è dato)
NOTA: assumete che tutte le istruzioni usate nel programma siano di base (nessuna pseudoistruzione).

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) indicate il contenuto della pipeline (quali istruzioni si trovano in quali fasi) alla fine dello **11°** colpo di clock
5) quanti cicli di clock sono necessari a eseguire tutto il programma
6) quanti ne sarebbero necessari se il forwarding non esistesse
7) riordinate le istruzioni per ridurre al massimo gli stalli (mantenendo invariata la sua semantica)
8) calcolate quanti cicli di clock sono necessari a eseguire il programma così ottimizzato

```
.globl main
.data
vettore1: .space 400              # 100 words
vettore2: .space 400              # 100 words
DIM:      .word 100               # N=100
.text
main:     move  $s0, $zero        # i = 0*4
          li    $s1, -4           # j = -4
          lw    $s2, DIM          # N = DIM
          sll   $s2, $s2, 2       # N *= 4
loop:     beq   $s0, $s2, fine    # se i*4==N*4
          lw    $t2, vettore1($s0) # X = v1[i]
          andi  $t3, $t2, 1       # X & 1
          beqz  $t3, next         # se X è pari
          addi  $s1, $s1, 4       # j += 4
          sw    $t2, vettore2($s1) # v2[j] = X
next:     addi  $s0, $s0, 4       # i += 4
          j loop
fine:     li $v0, 10              # per syscall
          syscall                 # fatto
```

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 MZ – Prof. Sterbini – 30/6/14

1°	2°	3°

Cognome e Nome: _____ Matricola: _____ Correzioni ricevute

Esercizio 4 (14 punti). Considerate un sistema con due livelli di cache: **CPU <=> L1 <=> L2 <=> RAM**
- L1 è una cache 4-way set-associative con 2 set e blocchi grandi 8 word e strategia di rimpiazzo LRU.
- L2 è una cache 2-way set-associative con 8 set e blocchi grandi 32 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 cache, indicate quali dei seguenti accessi in memoria sono hit o miss in ciascuna delle due cache:

130 310 1800 150 519 342 1820 127 529 2000 325 516 1794 317

2) per ciascuna MISS indicate se è di tipo **Caricamento, Capacità o Conflitto**

3) calcolate le dimensioni in bit delle due cache L1, L2

4) assumendo che il processore vada a **2Ghz** con **1 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 **30ns**, calcolate il **tempo totale** e il **CPI medio** per questa sequenza di accessi.

Scrivete le risposte qui sotto e sul retro del foglio

Esame di Architetture – Canale MZ – Prof. Sterbini – 30/6/14 – Compito A

Parte 3 (assembler)

Esercizio 5 (30 punti se corretto e ricorsivo, 18 se corretto e iterativo, 0 se non funziona).

Vanno svolti sia la funzione 1) che il main 2)

1) Si realizzi la funzione RICORSIVA **pALinDrOmO** che verifica se una stringa è palindroma **con le seguenti condizioni aggiuntive**:

- gli spazi e gli altri caratteri non-alfanumerici vanno ignorati (sia all'inizio che alla fine)

- una stringa lunga una sola o 0 lettere è sempre palindroma

- le due lettere agli estremi si corrispondono SE E SOLO SE, oltre ad essere la stessa lettera, una è MAIUSCOLA e l'altra minuscola (differiscono di 'A' – 'a')

- se le due lettere agli estremi si corrispondono bisogna controllare che il resto della stringa (senza il primo carattere, lunga 2 caratteri di meno) sia pALinDrOmA

Argomenti da passare:

- indirizzo della stringa
- suo numero di caratteri

Risultato da tornare:

- 1 se VERO
- 0 se FALSO

2) Si realizzi un programma che usa la funzione **pALinDrOmO** che legge da stdin una stringa di massimo 100 caratteri e che stampa la frase “PALINDROMO” oppure “NON PALINDROMO” a seconda del caso

Esempi:

Input: “AmO RoMa”

Output: “PALINDROMO”

Input: “so'ffi.cini al burrO R;R UBL AI!NI CIF'FOS”

Output: “PALINDROMO”

Input: “trentatre TRENTINI entrarono a TRENTO”

Output: “NON PALINDROMO”