# Esame di Architetture – Canale MZ – Prof. Sterbini – 21/7/14

**Cognome e Nome: _____ Matricola: _____**

---

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

**Esercizio 1 (14 punti).** In una partita di CPU a ciclo di clock singolo (vedi sul retro) la Control Unit potrebbe essere rotta, producendo il segnale di controllo **Jump** attivo se e solo se è attivo il segnale **MemRead**.

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

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.

---

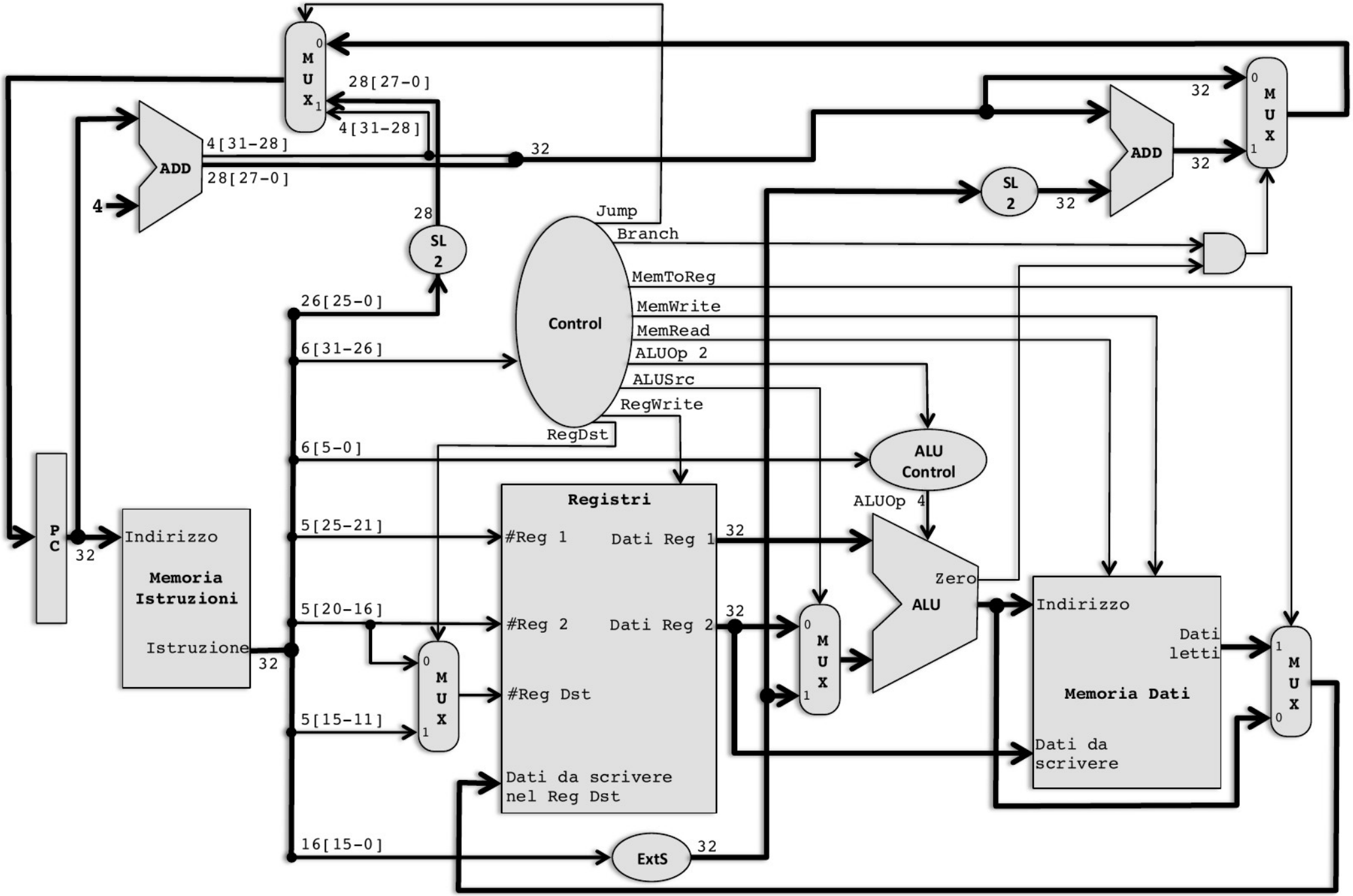
**Esercizio 2 (16 punti).** Considerate l'architettura MIPS a ciclo singolo in figura (diagramma sul retro). Vogliamo aggiungere l'istruzione di tipo R **`jral rs, rt`** (jump to register and link to rt) che salta incondizionatamente all'indirizzo contenuto nel registro **`rs`** e che contemporaneamente memorizza nel registro **`rt`** l'indirizzo della istruzione successiva.

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 **50ns**, l'accesso ai registri **25ns**, le operazioni dell'ALU **100ns**, 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 – 21/7/14

**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 sotto che somma gli elementi dispari di un vettore usando puntatori in memoria.

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

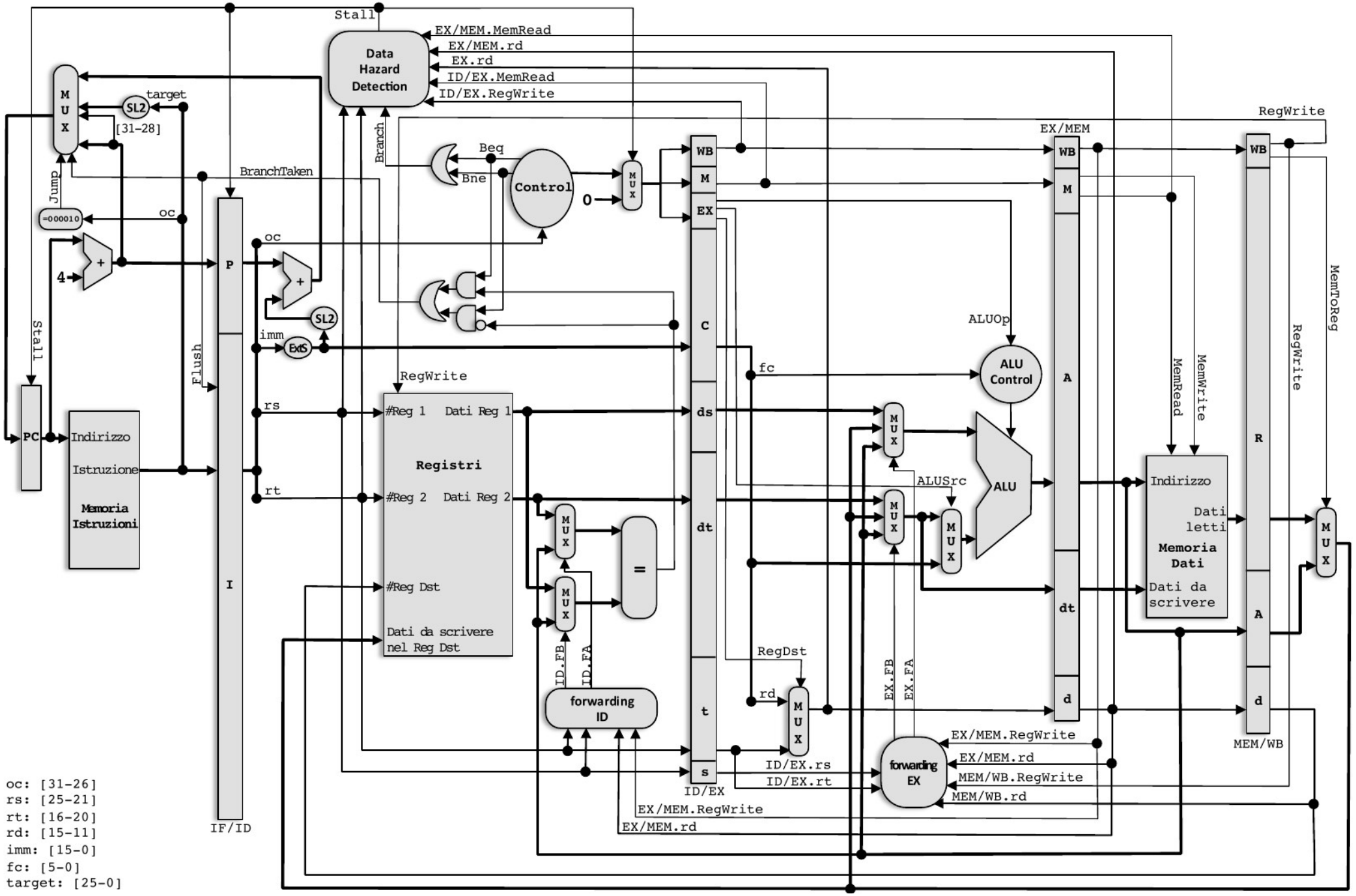
Indicate:
**1)** tra quali istruzioni sono presenti data hazard,
**2)** tra quali istruzioni sono presenti control hazard,
**3)** tra quali istruzioni sono necessari stalli (data e control)
**4)** indicate il contenuto della pipeline (quali istruzioni si trovano in quali fasi) alla fine dello **16°** 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
V:      .space 400              # 100 words
somma:  .word 0                 # somma=0
DIM:    .word 100               # N=100
.text
main:   la   $s1, V             # i = &V[0]
        lw   $s2, DIM           # N = DIM
        sll  $s2, $s2, 2        # N *= 4
        add  $s2, $s1, $s2      # j = &V[N]
loop:   beq  $s1, $s2, fine     # se i==j
        lw   $t2, ($s1)         # X = V[i]
        sll  $t3, $t2, 31       # X << 31
        beqz $t3, next          # se X è pari
        lw   $t3, somma         # Y = somma
        add  $t3, $t2, $t3      # Y += X
        sw   $t3, somma         # somma = Y
next:   addi $s1, $s1, 4        # i += 4
        j loop
fine:   li $v0, 10              # per syscall
        syscall                 # fatto
```

NOTA se necessario usate un foglio protocollo per la risposta

---

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 – 21/7/14

| 1° | 2° | 3° |
|---|---|---|
| | | |

Correzioni ricevute

**Solo la matricola: _____**

**Esercizio 4 (14 punti).** Considerate un sistema con due livelli di cache: **CPU <=> L1 <=> L2 <=> RAM**
- L1 è una cache 2-way set-associative con 2 set e blocchi grandi 4 word e strategia di rimpiazzo LRU.
- L2 è una cache 2-way set-associative con 4 set 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 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 compresi i bit di controllo

4) assumendo che il processore vada a **1Ghz** con **2 CPI** (Clock Per Instruction), che gli accessi in memoria impieghino **50ns**, che gli hit nella cache L1 impieghino **2ns** e gli hit nella cache L2 impieghino **10ns**, 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 – 21/7/14 – 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 **raggruppalettere** che raggruppa le lettere di una stringa mettendo:

- prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')

- poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)

**NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante

**Esempio di algoritmo ricorsivo da realizzare:**

- se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (la stringa è già in ordine)

- se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la lettera è già al suo posto)

- se il primo carattere non è una lettera scambiala con l'ultima ed esegui raggruppalettere sul resto della stringa più corta di un carattere (l'ultima è stata messa a posto)

**Esempio di algoritmo iterativo:**

- sposta tutti i caratteri alfabetici all'inizio

- tutti gli altri vanno in fondo

**Argomenti da passare (o altro se preferite):**

- indirizzo della stringa
- suo numero di caratteri

**Risultato da tornare:**

- nessuno (la stringa va modificata in memoria)

**2)** Si realizzi un programma main che usa la funzione **raggruppalettere** e che legge da stdin una stringa di massimo 100 caratteri e che stampa la stringa risultante

**Esempi:**

**Input:** “Paolino Paperino 113”

**Output:** “PaolinooPaperin 113 ”

**Input:** “Grisù il dr4gh3tto”

**Output:** “Grisotiltdrhg34  ù”

**Input:** “0123456789”

**Output:** “1234567890”

**Input:** “Ciabatta”

**Output:** “Ciabatta”

# Esame di Architetture – Canale MZ – Prof. Sterbini – 21/7/14 – Compito B

## *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 **contaLettere** che conta il numero di lettere, cifre ed altri caratteri nella stringa:

- se la stringa è di 0 caratteri torna la tripla <0, 0, 0>

- se la prima lettera è una lettera (tra 'a' e 'z' e tra 'A' e 'Z') chiama contaLettere sul resto della stringa e somma 1 al primo valore tornato

- se la prima lettera è una cifra (tra '0' e '9') chiama contaLettere sul resto della stringa e somma 1 al secondo valore tornato

- altrimenti chiama contaLettere sul resto della stringa e somma 1 al terzo valore tornato

**Argomenti da passare:**

- indirizzo della stringa
- suo numero di caratteri

**Risultato da tornare:**

la terna < #lettere, #cifre, #altro >

**2)** Si realizzi un programma main che usa la funzione **contaLettere** che legge da stdin una stringa di massimo 100 caratteri e che dopo averla esaminata stampa il numero di lettere, cifre, altro

**Esempi:**

**Input:** “La volpe grigia s'alza la mattina alle 8:43”

**Output stampato:** 31 3 9