# [T06] Esercitazione 6

## Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ciascun esercizio di programmazione. Non modificate in alcun modo i programmi di test `*_main.c`.
- Rinominare la directory chiamandola `cognome.nome`. Sulle postazioni del laboratorio sarà `/home/biar/Desktop/cognome.nome/`.
- È possibile consultare appunti/libri e il materiale didattico online.
- Rispondete alle domande online sul modulo di consegna.
- **Finiti gli esercizi**, e non più tardi della fine della lezione:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione).
  - fate **upload** del file `cognome.nome.zip`.
  - **importante**: fate logout dal vostro account Google!
- **Se avete domande** accedete a Zoom agli orari stabiliti per l'esercitazione, accedendo con la **mail istituzionale** uniroma1.it dello studente. Troverete online i docenti ed il tutor del corso. In alternativa, scrivete via mail ai docenti.
- **Suggerimento** Non traducete direttamente da C a IA32, ma scrivere prima una versione C intermedia `E1/e1_eq.c` equivalente a quella di partenza, ma più semplice da tradurre in assembly. Testatela con il main di prova prima di passare a scrivere la versione `.s`. E' inutile tradurre la versione C equivalente se è errata!

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

## Esercizio 1 (Ricerca binaria in un array ordinato)

Tradurre nel file `E1/e1.s` la seguente funzione C contenuta in `E1/e1.c` che implementa il classico algoritmo ricerca binaria (o dicotomica) restituento 1 se `x` appartiene all'array `v` di `n` interi `int` e 0 altrimenti. Usare il file `E1/e1_eq.c` per sviluppare la versione C equivalente.

```c
#include "e1.h"

int binsearch(int *v, int n, int x) {
    int i=0, j=n;
    while (i<j) {
        int m = (i+j)/2;
        if (v[m]==x) return 1;
        if (v[m]>x) j=m;
        else i=m+1;
    }
    return 0;
}
```

*Suggerimento*: usare lo shift per dividere per 2.

Usare il main di prova nella directory di lavoro E1 compilando con `gcc -m32 e1_main.c e1.s -o e1`.

## Esercizio 2 (Minimo Comune Multiplo)

Tradurre nel file E2/e2.s la seguente funzione C contenuta in E2/e2.c che, dati due interi, ne calcola il minimi comune multiplo. Usare il file E2/e2_eq.c per sviluppare la versione C equivalente.

```c
#include "e2.h"

int lcm(int x, int y) {
    int greater = y;
    if (x > y)
        greater = x;
    while (1) {
        if ((greater % x == 0) && (greater % y == 0))
            return greater;
        greater++;
    }
}
```

*Suggerimento*: usare le istruzioni CMOVcc e SETcc.

Usare il main di prova nella directory di lavoro E2 compilando con `gcc -m32 e2_main.c e2.s -o e2`.

## Esercizio 3 (Frequenza caratteri)

Tradurre nel file E3/e3.s la seguente funzione C contenuta in E3/e3.c che, data una stringa C, calcola la frequenza dei caratteri nel testo e restituisce il carattere ASCII più frequente. Usare il file E3/e3_eq.c per sviluppare la versione C equivalente.

```c
#include <string.h>
#include "e3.h"

char charfreq(const char* s) {
    unsigned freq[256];
    memset(freq, 0, 256*sizeof(unsigned));

    while (*s) freq[*s++]++;

    unsigned maxi = 0;
    unsigned maxf = freq[0];
    int i;
    for (i=1; i<256; ++i){
        if (freq[i]>maxf) {
            maxi = i;        // A1
            maxf = freq[i];  // A2
        }
    }
    return maxi;
}
```

*Suggerimento*: usare l'istruzione CMOVcc per fare i due assegnamenti condizionali A1 e A2.

Usare il main di prova nella directory di lavoro E3 compilando con `gcc -m32 e3_main.c e3.s -o e3`.

## Esercizio 4 (Domande)

Rispondi alle seguenti domande, tenendo conto che una risposta corretta vale 1 punti, mentre una risposta errata vale 0 punti.

**Domanda 1**. Siano %eax=20 (decimale), %edx=0 (decimale) e %ecx=8 (decimale). Con riguardo alle due istruzioni "idivl %ecx" e "sarl $3, %eax", quale delle seguenti affermazioni risulta vera:

- A. Le due istruzioni scrivono lo stesso valore in %eax, ma "idivl" è più efficiente di "sarl"
- B. Le due istruzioni scrivono lo stesso valore in %eax, ma "sarl" è più efficiente di "idivl"
- C. Le due istruzioni scrivono valori diversi in %eax
- D. In questo caso la "idivl" non modifica il valore del registro %edx
- E. Nessuna delle precedenti

**Domanda 2**. Assumendo %eax=0xFF000000, %ecx=1 (decimale) e %edx=10 (decimale), dopo aver eseguito l'istruzione "testl %eax, %eax" quale delle seguenti affermazioni risulta vera:

- A. Se eseguiamo "cmovnzl %edx, %ecx" viene scritto il valore 10 nel registro %ecx
- B. Se eseguiamo "cmovzl %edx, %ecx" viene scritto il valore 0 nel registro %ecx
- C. L'istruzione "cmovnzw %dx, %cx" non modifica il valore del registro %ecx
- D. L'istruzione "cmovel %edx, %ecx" è equivalente all'istruzione "cmovnzl $10, %ecx"
- E. Nessuna delle precedenti

**Domanda 3**. Assumendo %eax=10 (decimale), %ecx=7 (decimale) e %edx=2 (decimale), quale delle seguenti affermazioni risulta vera:

- A. Se eseguiamo "cmpl %ecx, %eax" e "cmovlel %edx, %eax" viene scritto il valore 2 nel registro %eax
- B. Se eseguiamo "cmpl %ecx, %eax" e "cmovbl %edx, %eax" viene scritto il valore 2 nel registro %eax
- C. Se eseguiamo "cmpb $0, %al" e "cmovel %edx, %eax" viene scritto il valore 2 nel registro %eax
- D. Se eseguiamo "subl %ecx, %eax" e "cmovgl %edx, %eax" viene scritto il valore 2 nel registro %eax
- E. Nessuna delle precedenti

**Domanda 4**. Quale delle seguenti è un'istruzione valida:

- A. cmovzb %al, %cl
- B. cmovzl $3, %eax
- B. cmovgew %ax, (%ecx)
- B. cmpl %ecx, $10
- B. Nessuna delle precedenti