

## Sistemi di Calcolo (A.A. 2023-2024)

Corso di Laurea in Ingegneria Informatica e Automatica  
Sapienza Università di Roma



### Compito (15/10/2024) – Durata 1h 30'

Inserire nome, cognome e matricola nel file `studente.txt`.

**ISTRUZIONI PER STUDENTI DSA:** svolgere a scelta due parti su tre.

---

#### Parte 1 (programmazione IA32)

In questo esercizio, si chiede di tradurre in assembly una funzione che copia un array dato in input in uno spazio preallocato, fornito anch'esso in input, applicando la funzione `compute` agli elementi da copiare, solamente se richiesto. Nella directory E1, si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `e1B.s`:

```
char* cond_compute(char* array, unsigned* cond,
                  int n, char* out)
{
    int i = 0;
    char res;

    for (i = 0; i < n; i++) {
        out[i] = array[i];
        if (cond[i]) {
            compute(out[i], &res);
            out[i] = res;
        }
    }
    return out;
}
```

L'unico criterio di valutazione è la correttezza. Generare un file eseguibile `e1A` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1A_main.c`.

**Non** modificare in alcun modo `e1A_main.c`. Prima di tradurre il programma in IA32 si suggerisce di scrivere nel file `e1A_eq.c` una versione C equivalente più vicina all'assembly.

---

#### Parte 2 (programmazione di sistema POSIX)

Si vuole scrivere nel file E2/es2A una funzione `countMinWords` con il seguente prototipo:

```
int countMinWords(const char** s, int n);
```

che, dato un array `s` contenente `n` stringhe, conta il numero minimo di parole presenti nelle stringhe di `s`. Ad esempio se `s` è il seguente array:

```
{ "I'm tired of weakness", "tired of my feet of clay", "tired of days to come", "tired of yesterday" }
```

La funzione dovrà restituire il valore 3, dato che l'ultima stringa contiene 3 parole, e nessuna delle altre stringhe ne contiene un numero minore.

Si assuma che le parole siano separate dal carattere ‘ ‘ (spazio). L’implementazione della funzione deve rispettare il seguente algoritmo:

1. se  $n$  è 0, la funzione restituisce immediatamente il valore -1;
2. la funzione crea  $n$  processi figli, dove il processo  $i$ -esimo conta il numero di parole contenute nella  $i$ -esima stringa di  $s$ ; un nuovo processo figlio viene generato solo quando il precedente ha terminato la sua esecuzione: quando un processo figlio termina restituisce come codice di terminazione il numero di parole contenute nella stringa analizzata;
3. il processo genitore attende la terminazione dell’ultimo figlio creato, e restituisce il valore minimo tra quelli restituiti da tutti i figli.

Per i test, compilare il programma insieme al programma di prova `e2A_main.c` fornito, che **non** deve essere modificato.

---

### Parte 3 (quiz)

Si risponda ai seguenti quiz, inserendo le risposte (A, B, C, D o E per ogni domanda) nel file `e3A.txt`. Una sola risposta è quella giusta. Rispondere E equivale a non rispondere (0 punti).

---

#### Domanda 1 (Assembly)

Si consideri la seguente funzione assembly `foo(char* buf, size_t count)` che scrive su `stdout` il buffer passato come primo argomento.

```
foo:
    push %ebx           #salvo "ebx" per usarlo
    mov 12(%esp), %ebx
    push %ebx           #metto "count" sullo stack
    mov XXX(%esp), %eax
    push %eax           #metto "buf" sullo stack
    push $1             #metto "fd" sullo stack
    call write          #chiamo write(fd, buf, count)
    addl $12, %esp      #epilogo
    pop %ebx
    ret
```

Quale numero andrà messo nella quarta istruzione per far sì che il primo parametro della funzione `foo` (cioè `buf`) venga posizionato sullo stack?

<b>A</b>	16	<b>B</b>	12
<b>C</b>	8	<b>D</b>	4

Motivare la risposta nel file `M1.txt`. **Risposte non motivate saranno considerate nulle.**

---

#### Domanda 2 (Interrupt e System Call)

Si consideri la seguente funzione assembly `void bar()`:

```
bar:
    movl $0, %ebx
    movl $1, %eax
    int $0x80
    ret
```

Quale delle seguenti affermazioni è vera?

<b>A</b>	La funzione esegue la system call numero 0x80 con due parametri (0 e 1)	<b>B</b>	La funzione esegue la system call numero 0 con due parametri (1 e 0x80)
<b>C</b>	La funzione esegue la system call numero 0 con un parametro (1)	<b>D</b>	La funzione esegue la system call numero 1 con un parametro (0)

Motivare la risposta nel file M2.txt. **Risposte non motivate saranno considerate nulle.**

---

### Domanda 3 (Segnali)

Si consideri il caso in cui un processo riceva un segnale SIGSEGV a seguito di un accesso a memoria invalido. Quale delle seguenti affermazioni è vera?

<b>A</b>	Il programma terminerà necessariamente la sua esecuzione.	<b>B</b>	Il programma può gestire il segnale ma non potrà in alcun caso riprendere l'esecuzione.
<b>C</b>	Il programma può gestire il segnale e potrebbe riprendere l'esecuzione.	<b>D</b>	Nessuna delle precedenti.

Motivare la risposta nel file M3.txt. **Risposte non motivate saranno considerate nulle.**

---

### Domanda 4 (Cache)

Si consideri un sistema con una piccola cache completamente associativa contenente 2 sole linee da 8 byte ciascuna e politica di rimpiazzo LRU. Quanti cache miss vengono generati dal seguente frammento di programma? Si assuma che l'array `b` sia allineato a un indirizzo multiplo di 8 byte e che la cache inizialmente non contenga alcun blocco di memoria in uso al processo.

```
short b[16];
b[1] = 7;
b[2] = b[1];
b[4] = 2;
b[0] = 5;
b[12] = 9;
b[2] = b[10];
b[10] = b[1];
```

<b>A</b>	3	<b>B</b>	4
<b>C</b>	5	<b>D</b>	6

Motivare la risposta nel file M4.txt. **Risposte non motivate saranno considerate nulle.**