**Sistemi di Calcolo (A.A. 2021-2022)**
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

B

**Compito (17/06/2022) – Durata 1h 30'**

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

---

**Parte 1 (programmazione IA32)**

Il Cyclic Redundancy Check (CRC) è un algoritmo spesso utilizzato per individuare errori casuali durante la trasmissione di dati. In questo esercizio, viene richiesto di tradurre in assembly un'implementazione semplificata del CRC32. Nella directory E1, si traduca in assembly IA32 la seguente funzione C scrivendo un modulo `e1B.s`.

```
#include "e1B.h"

int crc32b(char *bytes, int n) {
    int magic = ~0; // operazione: not
    int crc = magic;
    while (n--) {
        int value;
        int byte = *bytes++; // attenzione!
        int index = crc ^ byte;
        get_constant(&value, index & 0xFF);
        crc = value ^ (crc >> 8);
    }
    return crc ^ magic;
}
```

Si ricorda che per utilizzare una costante esadecimale `0xABCD` in sintassi GAS/ATT occorre utilizzare l'operando `$0xABCD`. L'unico criterio di valutazione è la correttezza. Generare un file eseguibile `e1B` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1B_main.c` e `table.c` forniti.

**Nota**: **non** modificare in alcun modo `e1B_main.c` e `table.c`. Prima di tradurre il programma in IA32 si suggerisce di scrivere nel file `e1B_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 `wordcount` con il seguente prototipo:

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

che, dato un array `s` contenente `n` stringhe, conta il numero massimo 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 6, dato che la seconda stringa contiene 6 parole, e nessuna delle altre stringhe ne contiene un numero maggiore.

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 massimo tra quelli restituiti da tutti figli.

Per i test, compilare il programma insieme al programma di prova `e2B_main.c` fornito, che **non** deve essere modificato.

---

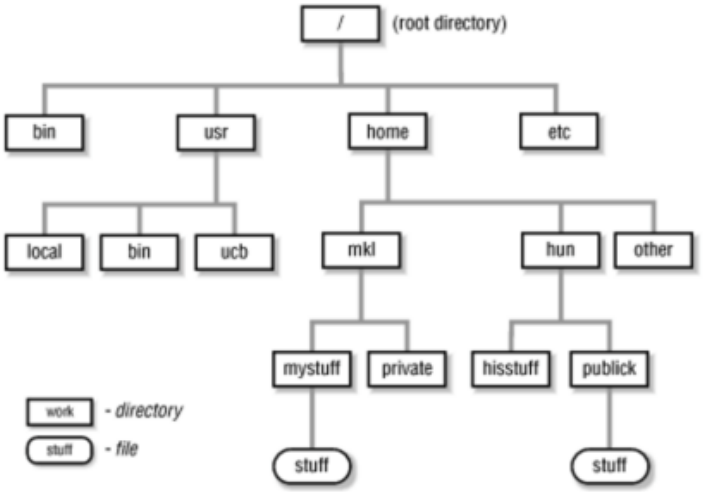
**Parte 3 (quiz)**

Si risponda ai seguenti quiz, inserendo le risposte (A, B, C, D o E per ogni domanda) nel file `e3B.txt`. Una sola risposta è quella giusta. Rispondere E equivale a non rispondere (0 punti).

---

**Domanda 1 (file system)**

Si consideri la seguente struttura di directory:




Assumendo di essere l'utente `hun` e che la directory corrente sia `publick`, quale dei seguenti comandi risulta valido?

| | | | |
|---|---|---|---|
| **A** | `cp -R ../../mkl/mystuff ../other` | **B** | `ls ./~/publick` |
| **C** | `mv stuff ~/../../home/hun/hisstuff` | **D** | `rm -rf ../~/hisstuff` |

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

---

**Domanda 2 (cache)**

Si consideri una cache associativa con 4 linee da 32 byte ciascuna e politica di rimpiazzo LRU, inizialmente vuota. Potendo scegliere fra più linee vuote, si usa la linea con indice più basso. Si ha inoltre un processo che accede in sequenza ai seguenti indirizzi di memoria (senza interruzioni): `37, 432, 258, 793, 773, 935, 268`.

Alla fine della sequenza di accessi, quali sono gli indici dei blocchi contenuti nelle 4 linee di cache? Il trattino indica che la linea di cache rimane vuota.

| | | | |
|---|---|---|---|
| **A** | 13, 24, 29, 8 | **B** | 29, 13, 8, 24 |
| **C** | 29, -, 13, - | **D** | 1, 13, 24, 8 |

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

---

**Domanda 3 (analisi delle prestazioni del software)**

Di quanto è necessario ridurre una porzione di un programma che richiede il 40% del tempo di esecuzione per ottenere uno speedup di 1.25x?

| | | | |
|---|---|---|---|
| **A** | 25% | **B** | 75% |
| **C** | 50% | **D** | 30% |

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

---

**Domanda 4 (allineamento e padding)**

Si consideri la seguente definizione di una struttura S:

```
struct S {
    short a;
    int b;
    char c;
    int d;
};
```

Si consideri un allineamento dei dati a indirizzi multipli di 4. Una sola delle seguenti affermazioni è **vera**. Quale?

| | | | |
|---|---|---|---|
| **A** | La dimensione in memoria di S è di 11 bytes | **B** | Ordinando i campi della struct [b, a, d, c] si minimizza il padding |
| **C** | La struct avrà 3 byte di padding | **D** | Ordinando i campi della struct [b, d, a, c] si minimizza il padding |

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