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

A

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 `e1A.s`.

```
#include "e1A.h"

int crc32(char *bytes, int n) {
    int i, j;
    int crc = 0xFFFFFFFF; // valore iniziale
    for (i = 0; i < n; i++) {
        int byte = bytes[i]; // attenzione!
        crc = crc ^ byte;
        for (j = 0; j < 8; j++) {
            int mask = crc & 1;
            mask = -mask; // operazione: neg
            crc = (crc >> 1) ^ (0xEDB88320 & mask);
        }
    }
    return ~crc; // operazione: not
}
```

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 `e1A` con `gcc -m32 -g`. Per i test, compilare il programma insieme al programma di prova `e1A_main.c` fornito.

Nota: **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 `multicount` con il seguente prototipo:

```
int multicount(const char** s, char c, int n)
```

che, dato un array `s` contenente `n` stringhe ed un carattere `c`, verifica il numero massimo di occorrenze di `c` nelle stringhe di `s`. Ad esempio se `c` è il carattere 's' ed `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 2, dato che il carattere compare massimo due volte nelle stringhe in `s` (in particolare compare due volte nella prima stringa). Se il carattere fosse 'l' la funzione dovrebbe restituire 1, dato che il carattere appare una sola volta nella terza stringa.

L'implementazione della funzione deve rispettare il seguente algoritmo:

1. se il carattere `c` è '' (NULL) o 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 occorrenze di `c` nella i-esima stringa di `s`; quando un processo figlio termina restituisce come codice di terminazione il numero di occorrenze di `c` nella stringa analizzata;
3. il processo genitore attende la terminazione di tutti i figli, uno alla volta, e restituisce il valore massimo tra quelli restituiti dai 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 IA32)

Si consideri il seguente frammento di codice:

```
1:    xorl %eax, %eax
2:    movw $1337, %ax
3:    movl $2, %ecx
4:    shrw %cl, %ax
5:    andl  $11, %eax
6:    ???
7:    ret
```

Quale delle seguenti istruzioni può essere inserita a linea 6 per far ritornare alla funzione il valore 20?

A	`orl $10, %eax`	**B**	`xorl $15, %eax`
C	`sarl $1, %eax`	**D**	Nessuna delle precedenti

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

Domanda 2 (paginazione)

Si consideri un sistema di calcolo con spazio logico dei processi a 32 bit. Quanto occupa la tabella delle pagine se ogni pagina è di 2 KB? Si assuma che le entry della tabella delle pagine siano grandi ciascuna 32 bit.

A	32 MB	**B**	64 MB
C	16 MB	**D**	8 MB

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

Domanda 3 (ottimizzazioni)

Si consideri il seguente frammento di codice:

```
int i = 0, x = 7, y = 0;
for(; i < 100; i++)
    y += x * i;
```

Quale delle seguenti ottimizzazioni può essere effettuata dal compilatore?

A	Loop Invariant Code Motion	**B**	Constant Propagation

C	Common Subexpression Elimination	**D**	Tutte le precedenti

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

Domanda 4 (permessi)

Che permessi (in notazione ottale) dovrebbe avere un file per essere accessibile in lettura, scrittura ed esecuzione dall'utente proprietario, lettura e scrittura dal gruppo proprietario, e solo lettura per tutti gli altri utenti?

A	`0764`	**B**	`0724`
C	`0562`	**D**	`0237`

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