Matricola: _____

Nome: _____

Cognome: _____

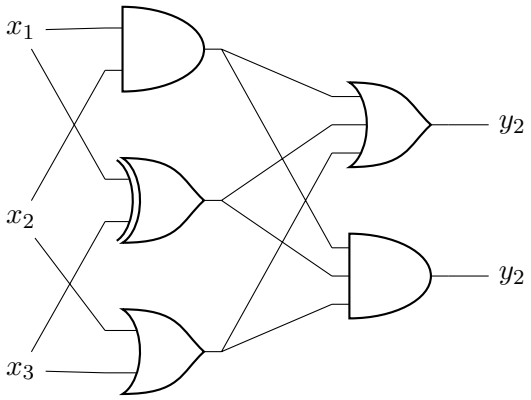
ESAME - Programmazione e Architetture (Modulo B)

4 luglio 2022

L'esame consiste di 15 domande a risposta multipla sugli argomenti del corso. Affinché una risposta sia considerata valida la scelta *deve essere motivata*.

Domanda 1

Si consideri il seguente circuito:



Per quali input $y_1 = 1$ e $y_2 = 1$?

☐ $x_1 = 1$, $x_2 = 0$, $x_3 = 1$

☐ $x_1 = 1$, $x_2 = 1$, $x_3 = 0$

☐ Nessun input

☐ Ogni input per cui $x_1 \neq x_3$

Domanda 2

Si consideri il valore 11101111 in binario. Convertito in base 10 il suo valore è:

☐ 239 ☐ 0.314

☐ 223 ☐ -239

Domanda 3

Dato un multiplexer con 16 input (esclusi quelli di selezione), quante linee di indirizzo/selezione saranno necessarie?

☐ 16 ☐ Un multiplexer è definito solo fino a 8 input

☐ 4 ☐ 1

Domanda 4

Ricordando che ARM-v7a ha 13 registri di 32 bit `R0`, ..., `R12`, quale è il valore contenuto nel registro `R0` al termine dell'esecuzione del seguente programma?

```
        MOV     R0,#0
        MOV     R1,#2
L1      CMP     R1,#0
        BEQ     E1
        MOV     R2,#3
L2      CMP     R2,#0
        BEQ     E2
        SUB     R2,R2,#1
        ADD     R0,R0,#1
        B       L2
E2      SUB     R1,R1,#1
        B       L1
E1
```

Si ricorda che `CMP` effettua una comparazione tra due valori e che `BEQ` effettua un salto condizionale quando l'ultima comparazione era tra valori uguali.

☐ Non termina

☐ Dipende dal valore che era contenuto in `R0` prima dell'esecuzione del programma

☐ 6

☐ 3

Domanda 5

Si consideri un filesystem tipo Unix come quello visto a lezione. Supponendo che ogni blocco possa contenere 1KB di dati o 1024 puntatori ad altri blocchi e che un inode abbia 13 puntatori a blocchi (10 diretti, uno indiretto singolo, uno indiretto doppio e uno indiretto triplo). La dimensione massima dei dati contenuti in un file che faccia uso *solo* di puntatori diretti e indiretti singoli è:

☐ Non è possibile usare solo puntatori diretti e indiretti singoli

☐ 11264KB

☐ 1034KB

☐ 1024KB

Domanda 6

Si considerino tre processi, P_1, P_2 e P_3 (ricevuti in questo ordine) in un sistema first-come first-served, nessuno dei processi effettua operazioni bloccanti. Tutti i processi necessitano di eseguire per 100ms prima di terminare. Ci si attende che P_3 termini dopo:

☐ 200ms

☐ 100ms

☐ 300ms

☐ Dipende dalla dimensione del quanto di tempo

Domanda 7

In un sistema in cui la protezione della memoria è implementata tramite un sistema base + limit, un processo con base 150 e limite 600 che volesse accedere all'indirizzo virtuale 250 accederebbe all'indirizzo fisico:

☐ 850

☐ 400

☐ 250

☐ Dipende dai valori nella page table

Domanda 8

Si consideri la seguente sequenza di comandi per la shell Unix:

```
cat foo.txt | grep hello | sort
```

Supponendo che non vi siano problemi di permessi, il risultato dell'ultimo comando è:

☐ Le righe del file `foo.txt` ordinate lessicograficamente

☐ Le righe del file `foo.txt` contenenti la parola *hello* ordinate lessicograficamente

☐ Le righe del file `hello` ordinate lessicograficamente

☐ Il numero di righe del file `foo.txt` contenenti la parola *hello*

Domanda 9

Si consideri il seguente codice:

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char * argv[])
{
  pid_t pid = fork();
  if (pid == 0) {
    pid_t pid2 = fork();
    if (pid2 == 0) {
      printf("a\n");
    } else {
      fork();
      printf("c\n");
   }
  } else {
    printf("b\n");
  }
  return 0;
}
```

Quale dei seguenti output (ignorando i newline) è possibile?

☐ `b b c c`

☐ `b a a c`

☐ `a b c`

☐ `b c a c`

Domanda 10

Si supponga che il processo P_1 invii un segnale `SIGURS1` al processo P_2. Allora:

☐ P_2 potrà leggere il segnale tramite una funzione di `read`

☐ P_1 rimane bloccato fino a quando P_2 non rilascia il segnale

☐ Verrà chiamato in P_2 il gestore di segnale registrato per `SIGURS1`

☐ Verrà chiamato in P_1 il gestore di segnale registrato per `SIGURS1`

Domanda 11

Dati due thread t_1 e t_2 e due lock L_1 e L_2, se i thread svolgono le seguenti operazioni:

- t_1: acquisisce il lock L_1, rilascia il lock L_1, acquisisce il lock L_2, rilascia il lock L_2;
- t_2: acquisisce il lock L_2, rilascia il lock L_2, acquisisce il lock L_1, rilascia il lock L_1;

Allora quale delle seguenti affermazioni è vera?

☐ Vi è un deadlock

☐ Vi è una race condition

☐ Non è possibile alcun deadlock

☐ tutte le operazioni di acquisizione dei lock devono precedere quelle di rilascio, quindi questo comportamento non è possibile

---

Domanda 12

Si consideri un programma in cui la parte parallela rappresenta il 75%, allora il massimo speedup possibile è:

☐ 0.75

☐ Dipende dal numero di processori

☐ 4

☐ 25

Domanda 13

Si consideri il seguente frammento di codice:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

pthread_mutex_t mylock = PTHREAD_MUTEX_INITIALIZER;

void * f(void * arg)
{
  pthread_mutex_lock(&mylock);
  int * a = (int *) arg;
  pthread_mutex_unlock(&mylock);
  for (int i = 0; i < 1000; i++) {
    *a = *a + 1;
  }
  return NULL;
}

int main(int argc, char * argv[])
{
  int a = 0;
  pthread_t t1;
  pthread_t t2;
  pthread_create(&t1, NULL, f, &a);
  pthread_create(&t2, NULL, f, &a);
  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  printf("%d\n", a);
  return 0;
}
```

Quale dei seguenti è l'output?

☐ 0

☐ 2000

☐ 1000

☐ Vi è una race condition, l'output non è deterministico

Domanda 14

Si consideri il seguente frammento di codice:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

pthread_mutex_t mylock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t mycond = PTHREAD_COND_INITIALIZER;

int *x;
int len = 0;
int done = 0;

void * f(void * arg)
{
  pthread_mutex_lock(&mylock);
  for (int i = 0; i < len; i++) {
    x[i] = i+1;
  }
  done = 1;
  pthread_cond_signal(&mycond);
  pthread_mutex_unlock(&mylock);
  return NULL;
}

void * g(void * arg)
{
  pthread_mutex_lock(&mylock);
  while (done != 1) {
    pthread_cond_wait(&mycond, &mylock);
  }
  for (int i = 0; i < len; i++) {
    printf("%d\n", x[i]);
  }
  pthread_mutex_unlock(&mylock);
  return NULL;
}

int main(int argc, char * argv[])
{
  pthread_t t1;
  pthread_t t2;
  x = (int *) malloc(10*sizeof(int));
  len = 10;
  pthread_create(&t1, NULL, f, NULL);
  pthread_create(&t2, NULL, g, NULL);
  pthread_join(t1, NULL);
```

```
    pthread_join(t2, NULL);
    return 0;
}
```

Quale delle seguenti affermazioni è vera?

- ☐ L'unico output possibile è `1 2 3 4 5 6 7 8 9 10`
- ☐ Il codice esegue correttamente solo se il primo thread riesce ad acquisire il lock per primo
- ☐ Vi è un deadlock
- ☐ Vi è una race condition

Domanda 15

Si consideri l'indirizzo IP `189.128.75.23` con maschera di sottorete `255.255.128.0`. Allora la parte di network dell'indirizzo è:

- ☐ `189.128.0.0`
- ☐ `0.0.75.23`
- ☐ `255.255.128.23`
- ☐ `255.255.75.23`
