**Politecnico di Milano – Sede di Cremona**
**Anno Accademico 2020/2021**

# Architettura dei Calcolatori e Sistemi Operativi

## Esame – 17.03.2022

**Prof. Carlo Brandolese**

**Cognome** _____ **Nome** _____

**Matricola** _____ **Firma** _____

### Istruzioni

1. Scrivere con cura, negli spazi sopra segnati, il proprio cognome, nome, numero di matricola e apporre la firma.
2. È vietato consultare libri, eserciziari, appunti ed utilizzare la calcolatrice e qualunque strumento elettronico (inclusi i cellulari), pena l'invalidazione del compito.
3. Il testo, debitamente compilato, deve essere riconsegnato in ogni caso.
4. Il tempo della prova è di 3 ore

### Valutazione

| Domanda | Voto | Note |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |
| F | | |

## Domanda A

Si implementi in codice assembly la funzione:

**`int norm( int x )`**

che prende in ingresso un intero a 32 bit e restituisce il numero di uni presenti nella sua rappresentazione binaria. Nell'implementazione si cerchi di utilizzare il numero minore possibile di registri.

```
norm:       add   $t0, $a0, $zero          # t0 = x
            add   $v0, $zero, $zero        # v0 = number of ones
loop:       beq   $t0, $zero, end          # if x == 0 finished
            andi  $t1, $t0, 0x0001         # selects the lsb of x
            add   $v0, $v0, $t1            # v0=v0+1 if lsb of x is 1
            srl   $t0, $t0, 1              # discards lsb
            beq   $zero, $zero, loop
end:        jr    $ra
```

## Domanda B

Si sviluppi un programma C che prende in ingresso una matrice di interi senza segno M[R][C] e calcola il valore massimo presente in tale matrice. Si assuma che M sia una variabile globale e le dimensioni R e C siano definite come macro. A tale scopo il programma utilizza un thread per ogni riga, in grado di calcolare il massimo degli elementi della riga stessa, quindi dopo che tutti i thread hanno completato il loro lavoro, il programma completa il calcolo del massimo assoluto.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

#define R 4
#define C 3

int M[R][C] = { { 1, 2, 3 }, { 2, 3, 4 }, { 0, 1, 2}, { 3, 5, 1 } };

// Thread function
void* row_max( void *arg )
{
    int row = (int)(arg);
    int max = 0;

    for( int col = 0; col < C; col++ ) {
       max = M[row][col] > max ? M[row][col] : max;
    }
    return (void*)max;

}

int main( int argc, char *argv[] )
{
    pthread_t tid[R];
    void* ret;
    int r_max;
    int m_max;

    // Creates a thred per each row and passes the row index
    for( int row = 0; row < R; row++ ) {
        pthread_create( &tid[row], NULL, row_max, (void*)row );
    }

    m_max = 0;
    for( int row = 0; row < R; row++ ) {
        pthread_join( tid[row], &ret );
        r_max = (int)ret;
        printf( "r_max[%d]: %d\n", row, r_max );
        m_max = r_max > m_max ? r_max : m_max;
    }

    printf( "m_max: %d\n", m_max );

    exit(0);
}
```

## Domanda C

Si consideri un sistema con uno spazio di indirizzamento di 1 GByte ed una architettura di cache con le caratteristiche seguenti:

| | D-Cache | I-Cache |
|---|---|---|
| **Associatività** | Completamente associativa | Diretta |
| **Dimensione totale** | 8 KB | 64 KB |
| **Dimensione linea** | 256 B | 512 B |
| **Tempo di accesso** | 2 ns | 1 ns |
| **Hit rate** | 99% | 98 % |

Si indichi la struttura dell'indirizzo visto dalle cache, descrivendo i vari campi e il loro significato.

Struttura dell'indirizzo D-Cache

*Dimensione memoria 1 G = 2^30 => 30 bit*

*Dimensione linea = 256 B = 2^8 => 8 bit*

*OFFSET = 8 bit*

*TAG = 30 – 8 = 22 bit*

Struttura dell'indirizzo I-Cache

*Dimensione memoria 1 G = 2^30 => 30 bit*

*Dimensione linea = 512 B = 2^8 => 8 bit*

*Numero linee = 64KB / 512 B = 2^16 / 2^9 = 2^7 => 7 bit*

*OFFSET = 9 bit*

*INDEX = 7 bit*

*TAG = 30 – 9 – 7 = 14*

Sapendo che:

- L'accesso alla memoria RAM avviene a parole di 64 bit
- Il tempo di accesso alla RAM in modalità normale è di 50 ns
- Il tempo di accesso alla RAM in modalità burst è
    - 100 ns per la prima parola
    - 20 ns per le parole successive

Si calcoli il tempo di accesso medio alla memoria attraverso le due cache

Tempo medio di accesso D-Cache

Thit = 2 ns

Tmiss = 2 ns + 100 ns + 20 ns x ( 256 / 8 – 1 ) = 722 ns

TD = 0.99 x 2 ns + 0.01 * 722 = 1.98 ns + 7.22 ns = 9.2 ns

Tempo medio di accesso I-Cache

Thit =1 ns

Tmiss = 1 ns + 100 ns + 20 ns x ( 512 / 8 – 1 ) = 1361 ns

TD = 0.98 x 1 ns + 0.02 * 1361 ns = 28.20

## Domanda D

Si consideri il seguente insieme di processi:

| Processo | Arrival time | Execution time |
|---|---|---|
| P1 | 0 | 8 |
| P2 | 3 | 3 |
| P3 | 4 | 6 |
| P4 | 8 | 1 |
| P5 | 10 | 4 |

Si esegua lo scheduling dei processi secondo l'algoritmo Shortest Remaining Time, assumendo che quando il remaining time di due o più processi risulta uguale, il processo selezionato è quello in esecuzione.

| 0 | | | | | | | 5 | | | | 10 | | | | 15 | | | | | 20 | | | | | | | | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 8 | 7 | 6 | | | | 5 | 4 | | 3 | 2 | 1 | 0 | | | | | | | | | | | | | | | |
| P2 | | | | 3 | 2 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | |
| P3 | | | | | 6 | | | | | | | | | | | | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | |
| P4 | | | | | | | | 1 | 1 | 0 | | | | | | | | | | | | | | | | | | |
| P5 | | | | | | | | | | | 4 | | 4 | 3 | 2 | 1 | 0 | | | | | | | | | | | |

Si calcoli il ritardo di ogni processo, il suo tempo di esecuzione e l'overhead di esecuzione, cioè il rapporto tra il ritardo ed il tempo di esecuzione nominale.

| Processo | Ritardo | Tempo di esecuzione | Overhead |
|---|---|---|---|
| P1 | 0 + 3 + 1 = 4 | 12 | 4/8 = 50% |
| P2 | 0 | 3 | 0/3 = 0% |
| P3 | 12 | 18 | 12/6 = 200% |
| P4 | 0 | 1 | 0/1=0% |
| P5 | 2 | 6 | 2/4 = 50 % |

**Domanda E**

Si consideri un sistema dotato di una memoria fisica di 16KB e di uno spazio di indirizzamento logico di 64KB. Tale sistema supporta la memoria virtuale con pagine di 1KB ed è dotato di una MMU con TLB di 8 elementi. Si indichi la dimensione in bit degli indirizzi:

Numero di pagina virtuale: _____6

Offset nella pagina virtuale: _____10

Numero di pagina fisica: _____4

Offset nella pagina fisica: _____10

Si considerino quindi due programmi X e Y così composti:

| | | | |
|---|---|---|---|
| CX: 3K | DX: 4K | PX: 1K | Entry point: 0x0480 |
| CY: 4K | DY: 1K | PY: 1K | Entry point: 0x0C10 |

Si rappresenti la memoria logica dei due processi.

| Pagina Virtuale | Contenuto |
|---|---|
| 0 | CX0 |
| 1 | CX1 |
| 2 | CX2 |
| 3 | DX0 |
| 4 | DX1 |
| 5 | DX2 |
| 6 | DX3 |
| | |
| | |
| | |
| | |
| | |
| 63 | PX0 |

| Pagina virtuale | Contenuto |
|---|---|
| 0 | CY0 |
| 1 | CY1 |
| 2 | CY2 |
| 3 | CY3 |
| 4 | DY0 |
| | |
| | |
| | |
| | |
| | |
| | |
| 63 | PY0 |

Si supponga che:

- Il caricamento di un programma avviene caricando solo la prima pagina di codice necessaria ed una pagina di stack. Il resto delle pagine viene caricato secondo lo schema di demand paging.
- Le pagine residenti di un processo sono al massimo 4.
- La dimensione della TLB è di 8 righe.
- La politica di sostituzione delle pagine nella TLB è FIFO.
- Nella memoria fisica le pagine vengono riempite a partire dall'indirizzo più basso

Si mostri lo stato della memoria e della MMU al termine delle operazioni riportate nel seguito.

1. Viene creato il processo P con PID 133
2. Viene eseguita la prima istruzione del programma X.

| PID | NPV | NPF |
|---|---|---|
| 133 | 1 | 0 |
| 133 | 63 | 1 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | CP1 |
| 1 | PP0 |
| | |
| | |
| | |
| | |
| | |
| | |

3. Il processo P accede ad una variabile globale all'indirizzo 0x1808

| PID | NPV | NPF |
|---|---|---|
| 133 | 1 | 0 |
| 133 | 63 | 1 |
| 133 | 6 | 2 |
| | | |
| | | |
| | | |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | CP1 |
| 1 | PP0 |
| 2 | DP3 |
| | |
| | |
| | |
| | |
| | |

4. Il processo P crea il processo Q con PID 266, secondo il meccanismo lazy loading

| PID | NPV | NPF |
|---|---|---|
| 133 | 1 | 0 |
| 133 | 63 | 1 |
| 133 | 6 | 2 |
| 266 | 1 | 0 |
| 266 | 6 | 2 |
| 266 | 63 | 3 |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | CP1/CQ1 |
| 1 | PP0 |
| 2 | DP3/DQ3 |
| 3 | PQ0 |
| | |
| | |
| | |
| | |

5. Il processo Q scrive un dato all'indirizzo 0x1820

| PID | NPV | NPF |
|---|---|---|
| 133 | 1 | 0 |
| 133 | 63 | 1 |
| 133 | 6 | 2 |
| 266 | 1 | 0 |
| 266 | 6 | 4 |
| 266 | 63 | 3 |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | CP1/CQ1 |
| 1 | PP0 |
| 2 | DP3 |
| 3 | PQ0 |
| 4 | DQ3 |
| | |
| | |
| | |

6. Il processo Q esegue la sostituzione del codice con il programma Y, libera la memoria non più richiesta ed inizia l'esecuzione.

| PID | NPV | NPF |
|---|---|---|
| 133 | 1 | 0 |
| 133 | 63 | 1 |
| 133 | 6 | 2 |
| 266 | 3 | 4 |
| 266 | 63 | 3 |
| | | |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | CP1 |
| 1 | PP0 |
| 2 | DP3 |
| 3 | PQ0 |
| 4 | CQ3 |
| | |
| | |
| | |

7. Il processo P termina

| PID | NPV | NPF |
|---|---|---|
| | | |
| | | |
| | | |
| 266 | 3 | 5 |
| 266 | 63 | 4 |
| | | |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | Libera |
| 1 | Libera |
| 2 | Libera |
| 3 | PQ0 |
| 4 | CQ3 |
| | |
| | |
| | |

8. Il processo Q, che utilizza 640 bytes di stack, chiama una funzione che contiene un array locale di 800 interi.

Lo stack totale diviene 640 + 800*4 = 640 + 3200 = 3840, cioè richiede complessivamente sup(3240 / 1024) = 4 pagine.

| PID | NPV | NPF |
|---|---|---|
| 266 | 62 | 0 |
| 266 | 61 | 1 |
| 266 | 60 | 2 |
| 266 | 3 | 5 |
| | | |
| | | |
| | | |
| | | |

| Pagina Fisica | Contenuto |
|---|---|
| 0 | PQ1 |
| 1 | PQ2 |
| 2 | PQ3 |
| 3 | PQ0 |
| 4 | CQ3 |
| | |
| | |
| | |

## Domanda F

Si descrivano il meccanismo di cambio di contesto e le funzioni coinvolte nel caso in cui il processo in esecuzione richieda un servizio di sistema operativo che non può essere completato.