Politecnico di Milano – Sede di Cremona
Anno Accademico 2023/2024

Architettura dei Calcolatori e Sistemi Operativi

Esame – 17.06.2024

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 sviluppi in codice assembly la funzione:

```
int sumdigits( unsigned int n )
```

che prende in ingresso un valore intero **n** e produce come risultato la somma delle cifre che compongono la rappresentazione decimale di tale valore. Per esempio, se n=3568 la funzione deve restituire il valore 3+5+6+8, cioè 22.

Nel seguito è riportato un codice di esempio che mostra come la funzione deve essere realizzata

```
        .data
n:      .word 36778421

        .text
main:   la    a0, n
        lw    a0, (a0)
        jal   sumdigits
        li    a7, 1
        ecall
        li    a7, 10
        ecall

sumdigits:
        li    t0, 0             # sum
        li    t1, 10            # constant 10
loop:   beq   a0, zero, end     # if n == 0 finihed
        rem   t2, a0, t1        # t2 = last digit
        add   t0, t0, t2        # updates the sum
        div   a0, a0, t1        # n = n / 10
        b     loop
end:    mv    a0, t0            # returns sum
        jr    ra
```

Domanda B

Si sviluppi in C un programma che prende in ingresso un testo e conta il numero di occorrenze di una data lettera. Per lo sviluppo del programma si considerino le seguenti assunzioni:

- Il testo è memorizzato in un array globale **`text[]`**
- La dimensione dell'array è nota a compile time ed è memorizzata nella variabile **`size`**
- La lettera da ricercare è fornita sulla linea di comando
- Al termine dell'elaborazione il programma deve stampare il conteggio richiesto

Il programma deve essere sviluppato parallelizzando l'elaborazione su due thread. Il codice seguente riporta la parte iniziale del codice.

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

char text[] = "This is vary long text...";
int  size = sizeof(text);

// Must be global
char letter;

void* count( void* arg )
{
  int n    = (int)arg;
  int dim  = size / 2;
  int base = n * dim;
  int cnt  = 0;

  for( int i = 0; i < dim; i++ ) {
    cnt += ( text[base+i] == letter ) ? 1 : 0;
  }
  return (void*)cnt;
}

int main( int argc, char** argv )
{
  if( argc != 2) {
    exit(-1);
  }
  letter = argv[1][0];

  pthread_t tid0, tid1;
  pthread_create( &tid0, NULL, count, (void*)0 );
  pthread_create( &tid1, NULL, count, (void*)1 );

  int *cnt0, *cnt1;
  pthread_join( tid0, (void**)&cnt0 );
  pthread_join( tid1, (void**)&cnt1 );

  int cnttot;
  cnttot = (int)cnt0 + (int)cnt1;

  printf( "Count: %d\n", cnttot );
}
```

Domanda C

Si consideri un sistema con uno spazio di indirizzamento complessivo pari a 128 GByte e due memorie cache con le caratteristiche seguenti:

	D-CACHE	I-CACHE
Associatività	Set associativa a 16 vie	Diretta
Dimensione totale	1 MB	128 KB
Dimensione linea	256 B per set	128 B
Tempo di accesso	2 ns	1 ns
Hit rate	99.5 %	98 %

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

Struttura dell'indirizzo D-CACHE

Offset: log2(256) = 8 bit

Index: log2(1MB / (256 B * 16)) = log2(2^20 / (2^8 * 2^4)) = 8 bit

Tag: log2(128 GB) – Offset – Index = 37 – 8 – 8 = 21 bit

Struttura dell'indirizzo I-CACHE

Offset: log2(128) = 7 bit

Index: log2(128K / 128 B) = log2(2^17 / 2^7) = 10 bit

Tag: log2(128 GB) – Offset – Index = 37 – 7 – 10 = 20 bit

Sapendo che:

- L'accesso alla memoria avviene a parole di 64 bit
- Il tempo di accesso alla RAM in modalità normale è di 100 ns
- Il tempo di accesso alla RAM in modalità burst è di
  - 200 ns per il primo accesso
  - 40 ns per gli accessi successivi

Si calcoli il tempo di accesso medio alla memoria.

Tempo medio di accesso ai dati

Td,hit = 2ns

Td,miss = 2ns + 200ns + (256 B / 8 B – 1) * 40ns = 202 ns + 31 * 40 ns = 1442 ns

Td = 0.995 * 2 ns + 0.005 * 1442 ns = 1.99 ns + 7.21 ns = 9.2 ns

Tempo medio di accesso alle istruzioni

Ti,hit = 1ns

Ti,miss = 1ns + 200ns + (128 B / 8 B – 1) * 40ns = 201 ns + 15 * 40 ns = 801 ns

Ti = 0.98 * 1 ns + 0.02 * 801 ns = 0.98 ns + 16.02 ns = 17 ns

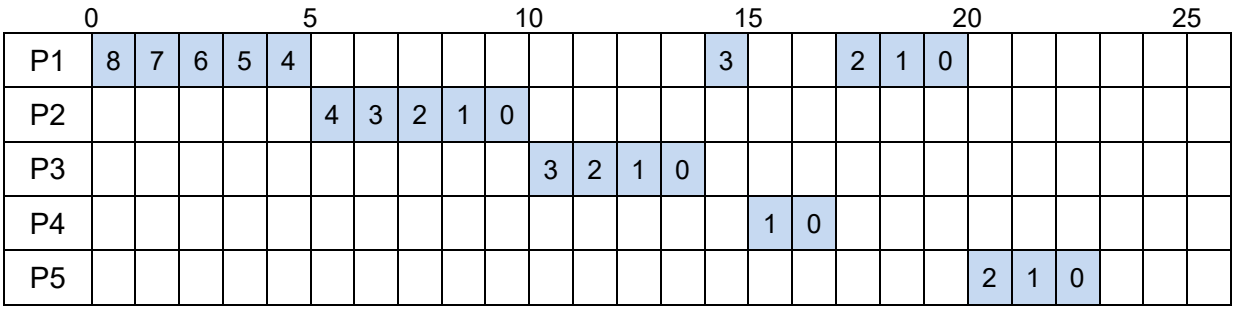
Domanda D

Si considerino i seguenti processi:

Process	Arrival Time	Execution Time
P1	0	9
P2	3	5
P3	4	4
P4	13	2
P5	16	3

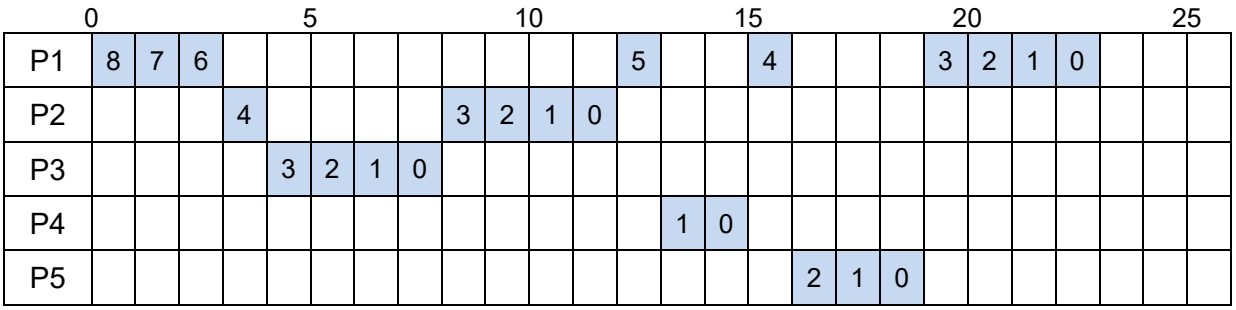
Si simuli lo scheduling di tali processi secondo gli algoritmi indicati e si calcoli il tempo medio di attesa nei due casi. Nel caso in cui due o più processi si trovino nella condizione di essere equivalenti dal punto di vista dello specifico algoritmo di scheduling, si scelga il processo con arrival time più recente.

Round Robin con periodo di 5 unità di tempo



Tempo medio di attesa = (11+2+6+2+4) / 5 = 25 / 5 = 5

Shortest Job First preemptive



Tempo medio di attesa = (14+4+0+0+0) / 5 = 18 / 5 = 3.6

Domanda E

Si consideri un sistema dotato di una memoria fisica di 16KB e di uno spazio di indirizzamento logico di 128KB. 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: _____7_____

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: 6K	DX: 2K	PX: 1K	Entry point: 0x01104
CY: 2K	DY: 2K	PY: 1K	Entry point: 0x00260

Si rappresenti la memoria logica dei due processi.

Programma X

Numero Pagina Virtuale	**Contenuto**	
0	CX0	
1	CX1	
2	CX2	
3	CX3	
4	CX4	Entry
5	CX5	
6	DX0	
7	DX1	
127	PX0	

Programma Y

	Numero Pagina virtuale	**Contenuto**
Entry	0	CY0
	1	CY1
	2	DY0
	3	DY1
	127	PY0

Si supponga che:

- Lo heap viene allocato immediatamente dopo la sezione DATA
- 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 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 TLB al termine delle operazioni riportate nel seguito.

1. Viene creato il processo P con PID 121
2. Viene eseguita la prima istruzione del programma X.

PID	NPV	NPF
121	4	0
121	127	1

Pagina Fisica	Contenuto
0	CP4
1	PP0

3. Il processo P accede ad un intero array globale di 256 interi con base all'indirizzo 0x018C0

PID	NPV	NPF
121	4	0
121	127	1
121	6	2
121	7	3

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1

4. Il processo P chiama una funzione all'indirizzo 0x00508

PID	NPV	NPF
121	127	1
121	6	2
121	7	3
121	1	4

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1

5. Il processo P crea il processo Q con PID 321, secondo il meccanismo di lazy loading

PID	NPV	NPF
121	127	1
121	6	2
121	7	3
121	1	4
321	1	4
321	127	5

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1/CQ1
5	PQ0

6. Il processo Q alloca 512 byte di memoria dinamica

PID	NPV	NPF
121	127	1
121	6	2
121	7	3
121	1	4
321	1	4
321	127	5
321	8	6

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1/CQ1
5	PQ0
6	HQ0

7. 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
121	127	1
121	6	2
121	7	3
121	1	4
321	0	5
321	127	6

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1
5	CQ0
6	PQ0

8. Il processo P alloca una variabile sullo stack spostando la cima all'indirizzo 0x1FB4C

PID	NPV	NPF
121	6	2
121	7	3
121	1	4
321	0	5
321	127	6
121	126	7

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1
5	CQ0
6	PQ0
7	PP1

9. Il processo Q termina e libera la memoria non più utilizzata

PID	NPV	NPF
121	6	2
121	7	3
121	1	4
121	126	7

Pagina Fisica	Contenuto
0	CP4
1	PP0
2	DP0
3	DP1
4	CP1
7	PP1