Politecnico di Milano – Sede di Cremona
Anno Accademico 2021/2022

Architettura dei Calcolatori e Sistemi Operativi

Esame – 05.07.2022

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:

`unsigned int fact( unsigned int n )`

che prende in ingresso un intero positivo e restituisce il fattoriale di tale valore, calcolato in modo ricorsivo secondo la relazione:

`fact(n) = n * fact(n-1)`

```
main:
       li      $a0, 5
       jal     fact
       move    $a0, $v0
       li      $v0, 1
       syscall
       li      $v0, 10
       syscall


fact:  addi    $sp, $sp, -8
       sw      $a0, 0($sp)
       sw      $ra, 4($sp)
       li      $v0, 1
       beq     $a0, $zero, end
       addi    $a0, $a0, -1
       jal     fact
       lw      $a0, 0($sp)
       lw      $ra, 4($sp)
       mult    $a0, $v0
       mflo    $v0
end:   addi    $sp, $sp, +8
       jr      $ra
```

Domanda B

Si consideri il seguente programma C che prende in ingresso due vettori dati e scambia il loro contenuto.

```
#include <stdio.h>

#define N 6000

int a[N] = { 1, 3, 5, 7, 9, 11, ... };
int b[N] = { 0, 2, 4, 6, 8, 10, ... };

void swap( int* a, int* b, int n )
{
  int t;
  for( int i = 0; i < n; i++ )
  {
    t = a[i];
    a[i] = b[i];
    b[i] = t;
  }
}

int main()
{
  swap( a, b, N );

  for( int i = 0; i < 6; i++ )
  {
    printf( "%d\t%d\n", a[i], b[i] );
  }

  return 0;
}
```

Si riscrivano la funzione swap() ed il programma main() in modo da parallelizzare l'esecuzione su due thread concorrenti.

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

#define N 6000

int a[N] = { 1, 3, 5, 7, 9, 11, ... };
int b[N] = { 0, 2, 4, 6, 8, 10, ... };

void* swap( void* arg )
{
  arg_t a = *(arg_t*)arg;
  int   t;

  for( int i = 0; i < a.n; i++ ) {
    t = a.a[i];
    a.a[i] = a.b[i];
    a.b[i] = t;
  }
}
```

```
typedef struct {
      int*  a;
      int*  b;
      int   n;
} arg_t;

int main()
{
      arg_t      a1 = { a,       b,       N/2 };
      arg_t      a2 = { a + N/2, b + N/2, N/2 };
      pthread_t  t1;
      pthread_t  t2;

      pthread_create( &t1, NULL, swap, (void*)&a1 );
      pthread_create( &t2, NULL, swap, (void*)&a2 );

      pthread_join( t1, NULL );
      pthread_join( t2, NULL );

      for( int i = 0; i < 6; i++ ) {
            printf( "%d\t%d\n", a[i], b[i] );
      }

      return 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	Associativa a 4 vie	Diretta
Dimensione totale	8 KB	32 KB
Dimensione linea	256 B	128 B
Tempo di accesso	2 ns	1 ns
Hit rate	98%	99 %

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 => 8bit

Index: log2(8K/(256*4)) = log2(8) = 3 => bit

Tag: log2(1G) – 8 – 3 = 30 – 11 = 19 => 19 bit

Struttura dell'indirizzo I-Cache

Offset: log2(128) = 7 => 7bit

Index: log2(32K/128) = log2(2^15 / 2^7) = 8 => bit

Tag: log2(1G) – 7 – 8 = 30 – 15 = 15 => 15 bit

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

TD,miss = 2ns + 100 ns + 20ns * (256/8-1) = 7422 ns

TD,ave = 2 * 0.98 + 722ns * 0.02 = 1.96 + 14.44 = 16.40 ns

Tempo medio di accesso I-Cache

TI,miss = 1ns + 100 ns + 20ns * (128/8-1) = 401 ns

TI,ave = 1 * 0.99 + 401ns * 0.01 = 0.99 + 4.01 = 5 ns

Si consideri un programma in cui il 20% delle istruzioni esegue un accesso alla memoria e si calcolino il tempo di accesso medio in assenza di cache, il tempo di accesso medio in presenza della sola cache istruzioni ed infine il tempo di accesso medio in presenza di entrambe le cache.

In generale:

Tave = (100 * Tfetch + 20 * Trw) / 100

Tempo medio di accesso in assenza di cache

Tfetch = 50ns

Trw = 50ns

Tave = (100 * 50 ns + 20 * 50 ns) / 100 = (5000 ns + 1000 ns) / 100 = 60 ns

Tempo medio di accesso con sola I-Cache

Tfetch = 5ns

Trw = 50ns

Tave = (100 * 5 ns + 20 * 50 ns) / 100 = (500 ns + 1000 ns) / 100 = 15 ns

Tempo medio di accesso con entrambe le cache

Tfetch = 5ns

Trw = 16.40ns

Tave = (100 * 5 ns + 20 * 16.40 ns) / 100 = (500 ns + 328 ns) / 100 = 8.28 ns

Domanda D

Dato il seguente codice, si completi la tabella sottostante riportando, per ogni simbolo, la sezione in cui il simbolo viene allocato, la quantità di memoria richiesta nel file elf e la dimensione richiesta a run-time.

```
#define    SIZE  16

typedef struct {
      float re;
      float im;
} cplx_t;

union {
      int           w[4];
      unsigned char b[16];
} data;

int cplx_t A[SIZE] = { 0 };
int cplx_t B[SIZE];

static const float pi = 3.14159267;

int main( int argc, char** argv )
{
      double norm = malloc( 128 * sizeof(double) );
      ...
      return 0;
}
```

Simbolo	Sezione	Memoria nel file elf	Memoria a run-time
SIZE	nessuna	0	0
cplx_t,	nessuna	0	0
data	bss	0	16
A	data	128	128
B	bss	0	128
pi	nessuna	0	0
main	text	4	4
argc	stack	0	4
argv	stack	0	4
norm	Stack	0	4

Domanda E

Si consideri un sistema dotato di una memoria fisica di 16KB e di uno spazio di indirizzamento logico di 256KB. 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 delle seguenti informazioni:

Numero di pagina virtuale: _____ 8

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: 2K	DX: 6K	PX: 1K	Entry point: 0x0680
CY: 2K	DY: 1K	PY: 1K	Entry point: 0x006C

Si rappresenti la memoria logica dei due processi.

Pagina Virtuale	Contenuto
0	CX0
1	CX1
2	DX0
3	DX1
4	DX2
5	DX3
6	DX4
7	DX5
255	PX0

Pagina virtuale	Contenuto
0	CY0
1	CY1
2	DY0
255	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 11
2. Viene eseguita la prima istruzione del programma X.

PID	NPV	NPF
11	1	0
11	255	1

Pagina Fisica	Contenuto
0	CP1
1	PP0

3. Il processo P esegue la funzione f() all'indirizzo 0x00020
4. All'ingresso di f() lo stack pointer punta all'indirizzo 0x3E080

inf[0x3E080 / 1024] = 248 => Pagina fisica 248 => PX7

PID	NPV	NPF
11	1	0
11	255	1
11	0	2
11	248	3

Pagina Fisica	Contenuto
0	CP1
1	PP0
2	CP0
3	PP7

5. La funzione f() alloca un array automatico di 300 interi
6.
7. inf[(0x3E080 – 1200) / 1024] = 246 => Pagina fisica 246 => PX9

Secondo la politica FIFO scarico la pagina CP1

PID	NPV	NPF
11	255	1
11	0	2
11	248	3
11	246	4

Pagina Fisica	Contenuto
0	vuota
1	PP0
2	CP0
3	PP7
4	PP9

8. La funzione f() termina e ritorna al chiamante

PID	NPV	NPF
11	255	1
11	0	2
11	1	0

Pagina Fisica	Contenuto
0	CP1
1	PP0
2	CP0

9. Il processo P crea il processo Q con PID 22, secondo il meccanismo lazy loading

PID	NPV	NPF
11	255	1
11	0	2
11	1	0
22	0	2
22	1	0
22	255	3

Pagina Fisica	Contenuto
0	CP1 = CQ1
1	PP0
2	CP0 = CQ0
3	PQ0

10. Il processo Q legge 64 bytes a partire dall'indirizzo 0x00880

PID	NPV	NPF
11	255	1
11	0	2
11	1	0
22	0	2
22	1	0
22	255	3
22	2	4

Pagina Fisica	Contenuto
0	CP1 = CQ1
1	PP0
2	CP0 = CQ0
3	PQ0
4	DQ0

11. Il processo P termina

PID	NPV	NPF
22	0	2
22	1	0
22	255	3
22	2	4

Pagina Fisica	Contenuto
0	CQ1
1	vuota
2	CQ0
3	PQ0
4	DQ0

Domanda F

Si disegni la struttura dello stadio di DECODE della pipeline MIPS non ottimizzata. Se ne spieghi il funzionamento, descrivendo i diversi moduli e segnali che lo compongono.

Instruction		Description	Operation	Type	Opcode	Funct
add	rd,rs,rt	Add	rd	R	000000	100000
sub	rd,rs,rt	Subtract	rd = rs - rt	R	000000	100010
addi	rt,rs,imm	Add Immediate	rt = rs + imm	I	001000	
addu	rd,rs,rt	Add Unsigned	rd = rs + rt	R	000000	100001
subu	rd,rs,rt	Subtract Unsigned	rd = rs - rt	R	000000	100011
addiu	rt,rs,imm	Add Immediate Unsigned	rt = rs + imm	I	001001	
mult	rs,rt	Multiply	{hi, lo} = rs * rt	R	000000	011000
div	rs,rt	Divide	lo = rs / rt; hi = rs % rt	R	000000	011010
multu	rs,rt	Multiply Unsigned	{hi, lo} = rs * rt	R	000000	011001
divu	rs,rt	Divide Unsigned	lo = rs / rt; hi = rs % rt	R	000000	011011
mfhi	rd	Move From Hi	rd = hi	R	000000	010000
mflo	rd	Move From Lo	rd = lo	R	000000	010010
and	rd,rs,rt	And	rd = rs & rt	R	000000	100100
or	rd,rs,rt	Or	rd = rs | rt	R	000000	100101
nor	rd,rs,rt	Nor	rd = ~(rs | rt)	R	000000	100111
xor	rd,rs,rt	Exclusive Or	rd = rs ^ rt	R	000000	100110
andi	rt,rs,imm	And Immediate	rt = rs & imm0	I	001100	
ori	rt,rs,imm	Or Immediate	rt = rs | imm0	I	001101	
xori	rt,rs,imm	Exclusive Or Immediate	rt = rs ^ imm0	I	001110	
sll	rd,rt,sh	Shift Left Logical	rd = rt << sh	R	000000	000000
srl	rd,rt,sh	Shift Right Logical	rd = rt >>> sh	R	000000	000010
sra	rd,rt,sh	Shift Right Arithmetic	rd = rt >> sh	R	000000	000011
sllv	rd,rt,rs	Shift Left Logical Variable	rd = rt << rs	R	000000	000100
srlv	rd,rt,rs	Shift Right Logical Variable	rd = rt >>> rs	R	000000	000110
srav	rd,rt,rs	Shift Right Arithmetic Variable	rd = rt >> rs	R	000000	000111
slt	rd,rs,rt	Set if Less Than	rd = rs < rt ? 1 : 0	R	000000	101010
sltu	rd,rs,rt	Set if Less Than Unsigned	rd = rs < rt ? 1 : 0	R	000000	101011
slti	rt,rs,imm	Set if Less Than Immediate	rt = rs < imm ? 1 : 0	I	001010	
sltiu	rt,rs,imm	Set if Less Than Immediate Unsigned	rt = rs < imm ? 1 : 0	I	001011	
j	addr	Jump	PC = PC & 0xF0000000 | (addr<< 2)	J	000010	
jal	addr	Jump And Link	ra = PC + 8; PC = PC & 0xF0000000 | (addr<< 2)	J	000011	
jr	rs	Jump Register	PC = rs	R	000000	001000
jalr	rs	Jump And Link Register	ra = PC + 8; PC = rs	R	000000	001001
beq	rt,rs,imm	Branch if Equal	if (rs == rt) PC += 4 + (imm << 2)	I	000100	
bne	rt,rs,imm	Branch if Not Equal	if (rs != rt) PC += 4 + (imm << 2)	I	000101	
syscall		System Call		R	000000	001100
lui	rt,imm	Load Upper Immediate	rt = imm << 16	I	001111	
lb	rt,imm(rs)	Load Byte	rt = SignExt(MB[rs + imm])	I	100000	
lbu	rt,imm(rs)	Load Byte Unsigned	rt = MB[rs + imm] & 0xFF	I	100100	
lh	rt,imm(rs)	Load Half	rt = SignExt(MH[rs + imm])	I	100001	
lhu	rt,imm(rs)	Load Half Unsigned	rt = MH[rs + imm] & 0xFFFF	I	100101	
lw	rt,imm(rs)	Load Word	rt = MW[rs + imm]	I	100011	
sb	rt,imm(rs)	Store Byte	MB[rs + imm] = rt	I	101000	
sh	rt,imm(rs)	Store Half	MH[rs + imm] = rt	I	101001	
sw	rt,imm(rs)	Store Word	M4[rs + imm] = rt	I	101011	
ll	rt,imm(rs)	Load Linked	rt = MW[rs + imm]	I	110000	
Sc	rt,imm(rs)	Store Conditional	MW[rs + imm] = rt; rt = atomic ? 1 : 0	I	111000	

Pseudo instruction		Description	Operation	Type	Opcode	Funct
bge	rx,ry,imm	Branch if Greather Than or Equal	if (rs >= rt) PC += 4 + (imm << 2)			
bgt	rx,ry,imm	Branch if Greather Than	if (rs > rt) PC += 4 + (imm << 2)			
ble	rx,ry,imm	Branch if Less Than or Equal	if (rs <= rt) PC += 4 + (imm << 2)			
blt	rx,ry,imm	Branch if Less Than	if (rs < rt) PC += 4 + (imm << 2)			
la	rx,label	Load Address	rx = Address(label)			
li	rx,imm	Load 32-bit Immediate	rx = imm			
move	rx,ry	Move	rx = ry			
nop		No Operation				

Register Number	Register name
$0	$zero
$1	$at
$2 – $3	$v0–$v1
$4 – $7	$a0–$a3
$8 – $15	$t0–$t7
$16 – $23	$s0–$s7
$24 – $25	$t8–$t9
$26 – $27	$k0–$k1
$28	$gp
$29	$sp
$30	$fp
$31	$ra
	hi
	lo
	pc