Politecnico di Milano – Sede di Cremona
Anno Accademico 2018/2019

Architettura dei Calcolatori e Sistemi Operativi

Esame – 19.07.2019

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.
1. È vietato consultare libri, eserciziari, appunti ed utilizzare la calcolatrice e qualunque strumento elettronico (inclusi i cellulari), pena l'invalidazione del compito.
2. Il testo, debitamente compilato, deve essere riconsegnato in ogni caso.
3. Il tempo della prova è di 3 ore

Valutazione

Domanda	Voto	Note
A		
B		
C		
D		
E		
F		

Domanda A

Si consideri la seguente funzione ricorsiva semplificata per il calcolo di una potenza intera:

```
int pow( int b, int e )
{
  if( e == 0 ) {
    return 1;
  }
  if( e == 1 )
    return b;
  }
  return b * pow( b, e-1 );
}
```

Assumendo che sia l'esponente non sia negativo, si traduca la funzione in assembly MIPS

Domanda B

Si consideri un microprocessore MIPS dotato di una pipeline standard a cinque stadi **dotata di percorsi di propagazione.** Si consideri inoltre che nella fase di decode è possibile scrivere un dato nella prima parte del ciclo di clock e leggere nella seconda parte. Sia dato il seguente codice:

```
1.  SLTI    $t1, $t2, 24
2.  ADD     $t4, $t1, $t3
3.  SUB     $t5, $t4, $t1
4.  SW      $t1, 150($t4)
5.  SUBI    $t3, $t5, 1
6.  LW      $t1, 30($t2)
```

In primo luogo si identifichino tutte le dipendenze dati del precedente codice che potrebbero causare conflitti.

RAW:

WAR:

WAW:

Si simuli l'esecuzione del codice inserendo stalli ove necessario, e si calcoli il numero di cicli di clock necessari per eseguire il codice, e il CPI.

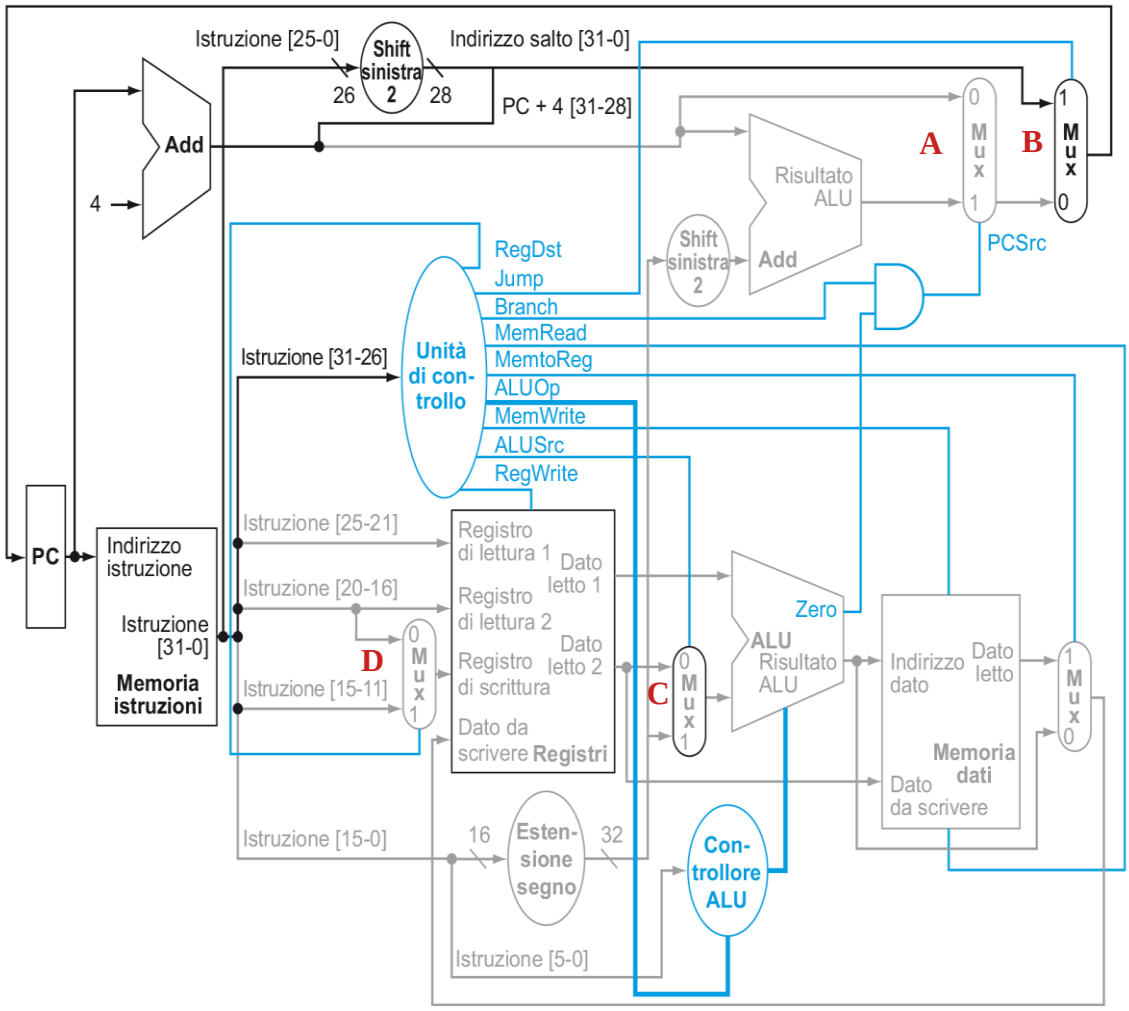
N° Istr.	**Cicli di Clock**														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLTI1															
ADD2															
SUB3															
SW4															
SUBI5															
LW6															

Numero cicli di clock	
CPI	

Si evidenzi nella colonna "Forwarding" quali percorsi di forwarding sono utilizzati tenendo conto di annotare solo le istruzioni che **propagano** un operando tramite tale tecnica e inserendo una o più tra le seguenti 4 diciture: N/A, EX/EX, MEM/EX, MEM/MEM.

Istruzione	**Forwarding**
SLTI1	
ADD2	
SUB3	
SW4	
SUBI5	
LW6	

Infine, data l'architettura di riferimento a singolo ciclo riportata qui di seguito:



si indichino nella tabella seguente quali sono i valori dei segnali di controllo nel caso dell'esecuzione dell'istruzione:

```
BEQ $t1, $t2, LOOP
```

Assumendo che $t1 = 4 e $t2 = 7.

Segnale	Valore
RegDst	
Jump	
Branch	
MemRead	
MemToReg	
PCSrc	
MemWrite	
ALUSrc	
RegWrite	

Domanda C

Dato il seguente programma C:

```
#include ...

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
sem_t           semaphore;
int             shared = 5;

void* thread1( void* arg )
{
      int var1 = *(int*)arg;

      sem_wait( &semaphore );

      pthread_mutex_lock( &mutex );
      shared = shared + (var1 * shared);  // BREAKPOINT A
      pthread_mutex_unlock( &mutex );

      sem_post( &semaphore );
}

void* thread2( void* arg )
{
      int var2 = *(int*)arg;

      pthread_mutex_lock( &mutex );
      shared = shared - var2;             // BREAKPOINT B
      pthread_mutex_unlock( &mutex );

      sem_post( &semaphore );
}

int main()
{
      int startA = 3;
      int startB = 4;
      int startC = 1;
      pthread_t th1, th2, th3;

      sem_init      ( &semaphore, 0, 0 );

      pthread_create( &th1, NULL, &thread1, (void*)&startA );
      pthread_create( &th2, NULL, &thread2, (void*)&startB );
      pthread_create( &th3, NULL, &thread1, (void*)&startC );

      pthread_join  ( th1, NULL );
      pthread_join  ( th2, NULL );
      pthread_join  ( th3, NULL );

      sem_destroy   ( &semaphore );       // BREAKPOINT C

      return 0;
}
```

Si svolgano i seguenti punti:

1. Si completi la seguente tabella, riportando lo stato delle variabili **immediatamente dopo** il breakpoint indicato. Nel caso si passi più di una volta da un breakpoint considerare ogni sua esecuzione nel rispondere. Si utilizzi la seguente classificazione:

   - “ESISTE” se la variabile certamente esiste
   - “NON ESISTE” se sicuramente la variabile non esiste
   - “PUO’ ESISTERE” se potrebbe esistere oppure non esistere

Breakpoint	**var1**	**var2**	**shared**
Breakpoint A			
Breakpoint B			
Breakpoint C			

2. Si completi la tabella seguente indicando tutti i possibili valori che le variabili riportate possono assumere subito dopo i breakpoint specificati.

Breakpoint	**var1**	**var2**	**shared**
Breakpoint A			
Breakpoint B			
Breakpoint C			

Domanda D

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti:

Memoria logica: 32 Kbyte
Memoria fisica: 16 Kbyte
Dimensione pagina: 4 Kbyte

Si considerino due programmi X e Y caratterizzati dalla seguente dimensione iniziale dei segmenti (C: Code, D: Data, P: Pila)

CX: 8K	DX: 12K	PX: 4K
CY: 12K	DY: 4K	PY: 4K

Completare la seguente tabella, riportando la struttura in pagine della memoria virtuale dei due programmi X e Y sapendo che le pagine vengono allocate sequenzialmente.

Indirizzo Pagina Virtuale	Programma X	Programma Y

Ad un certo istante T_0 le seguenti operazioni sono state completate:

1. Il processo P viene creato ed esegue il programma Y
2. Il processo Q viene creato ed esegue il programma X
3. Il processo Q accede ad un dato all'indirizzo 0x259E
4. Il processo P salta all'indirizzo 0x2754
5. Lo stack del processo Q cresce e alloca una nuova pagina
6. Il processo P fa una fork e crea un nuovo processo R
7. Il processo R crea due pagine di dati dinamiche
8. Il processo R termina

Sapendo che:

A) L'esecuzione di un programma avviene caricando inizialmente e in quest'ordine:
   a. La pagina di codice con l'istruzione di partenza
   b. Una pagina di pila
B) Il caricamento di ulteriori pagine in memoria avviene su richiesta (on demand)
C) Il numero di pagine residenti di **R** è pari a **3**
D) L'indirizzo esadecimale di partenza di X è 0x10A0
E) L'indirizzo esadecimale di partenza di Y è 0x07EC

F) Per la sostituzione delle pagine di memoria si utilizza una politica LRU. Almeno una pagina di pila deve sempre rimanere in memoria.

G) L'allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza

H) All'inizio della sequenza di eventi la MMU è vuota

Si riporti nelle tabelle seguenti una descrizione dello stato della memoria fisica e della MMU al termine della sequenza di operazioni di cui sopra.

Memoria Fisica	
Ind. Fisico	**Pag. Allocate**

MMU			
Proc	**NPV**	**NPF**	**Valido**

Domanda E

Per ognuna delle dichiarazioni seguenti, si compili la tabella corrispondente, riportando una breve descrizione del modificatore indicato in grassetto, indicando la quantità di memoria che deve essere riservata a compile-time e a run-time per il simbolo in esame ed infine indicando la sezione cui il simbolo viene assegnato al momento del linking.

```
// A livello di scope globale
static int X;
```

Descrizione	
Memoria a compile-time	
Memoria a run-time	
Sezione	

```
// A livello di scope globale
static const int X = 16;
```

Descrizione	
Memoria a compile-time	
Memoria a run-time	
Sezione	

```
// A livello di scope locale
register int X;
```

Descrizione	
Memoria a compile-time	
Memoria a run-time	
Sezione	

```
// A livello di scope locale
volatile int X;
```

Descrizione	
Memoria a compile-time	
Memoria a run-time	
Sezione	

Domanda F

Si descriva in modo chiaro e completo il processo di compilazione, indicando i vari passaggi, i tool coinvolti ed i formati di ingresso e di uscita di ogni passo.