Politecnico di Milano – Sede di Cremona
Anno Accademico 2016/2017

Architettura dei Calcolatori e Sistemi Operativi

Esame – 07.07.2017

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 linguaggio assembly la funzione:

```
char* strchr( char* haystack, char needle )
```

che copia cerca la prima occorrenza del carattere **needle** nella stringa **haystack** e restituisce il puntatore a tale carattere. Nel caso in cui il carattere non venga trovato la funzione restituisce un puntatore nullo.

Si traduca quindi in assembly in seguente programma C che utilizza la funzione strchr():

```
char* haystack = "This is the text!";
char* position;

int main( void )
{
  position = strchr( haystack, 't' );
  return (position != 0);
}
```

Domanda B

Si consideri il seguente programma.

```
#include ...

int   a   = 1;
sem_t s1;
sem_t s2;

void* thread1( void* arg )
{
     int var1 = 1;

     sem_wait( &s1 );
     var1    += a;
     a        = var1;                                  Breakpoint 1
     sem_post( &s1 );

}

void* thread2( void* arg )
{
     int var2 = 2;

     sem_wait( &s1 );
     var2    *= a;
     a        = var2;                                  Breakpoint 2
     sem_post( &s1 );
     sem_post( &s2 );
}

void* thread3( void *arg )
{
     int var3 = 3;

     sem_wait( &s2 );
     var3 += a;
     a     = var3;                                     Breakpoint 3
}

int main()
{
     pthread_t th1, th2, th3;

     sem_init      ( &s1, 0, 0 );
     sem_init      ( &s2, 0, 0 );

     pthread_create( &th1, NULL, &thread1, NULL );
     pthread_create( &th2, NULL, &thread2, NULL );
     pthread_create( &th3, NULL, &thread3, NULL );
```

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	var3	a
Breakpoint 1				
Breakpoint 2				
Breakpoint 3				
Breakpoint 4				

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	var3	a
Breakpoint 1				
Breakpoint 2				
Breakpoint 3				
Breakpoint 4				

Domanda C

Si consideri un sistema con uno spazio di indirizzamento di 512MByte, 2 cache set-associative a 4 vie della dimensione rispettivamente di 64Kbyte (DCACHE) e 32Kbyte (ICACHE). Inoltre la dimensione della linea per ogni set è pari a 256 Byte. Sulla base di queste informazioni si indichi la struttura dell'indirizzo visto dalle cache, descrivendo i vari campi e il loro significato.

Sapendo che:

- Il tempo di accesso alla cache in caso di hit è di 1 ns
- L'accesso alla memoria RAM avviene a parole di 32 bit
- Il tempo di accesso alla RAM in modalità normale è di 50 ns
- Il tempo di accesso alla RAM in modalità burst è di 60 ns per la prima parola e 6 ns per le parole successive
- L'hit rate della DCACHE è pari al 80%, mentre l'hit rate della ICACHE è pari al 98 %

Si calcolino i tempi medi di accesso alle due cache.

T_{DCACHE}=

T_{ICACHE}=

Si consideri quindi l'esecuzione di un programma in cui 40% delle istruzioni comporta un accesso in memoria e si calcoli il tempo medio di esecuzione di una istruzione per tale programma, supponendo che in assenza di stalli tutte le istruzioni siano eseguite in 1 ciclo.

T_{AVE}=

Si calcoli infine il miglioramento delle prestazioni per il programma in esame rispetto all'esecuzione sullo stesso sistema ma in assenza di memoria cache.

Domanda D

Sia dato un processore MIPS dotato di una pipeline standard a cinque stadi, percorsi di propagazione e predizione statica BRANCH NOT TAKEN. Sia inoltre dato il seguente codice assembly e si assuma che il l'istruzione BEQ (5) salti durante la prima esecuzione e non salti alla terza.

```
1         ADDI  R1, R0, 0
2   LOOP: LW    R2, 0(R0)
3         SUB   R1, R1, R2
4         ADDI  R3, R3, 1
5         BEQ   R1, R2, LOOP
6         ADD   R4, R3, R3
7         SW    R4, 0(R0)
```

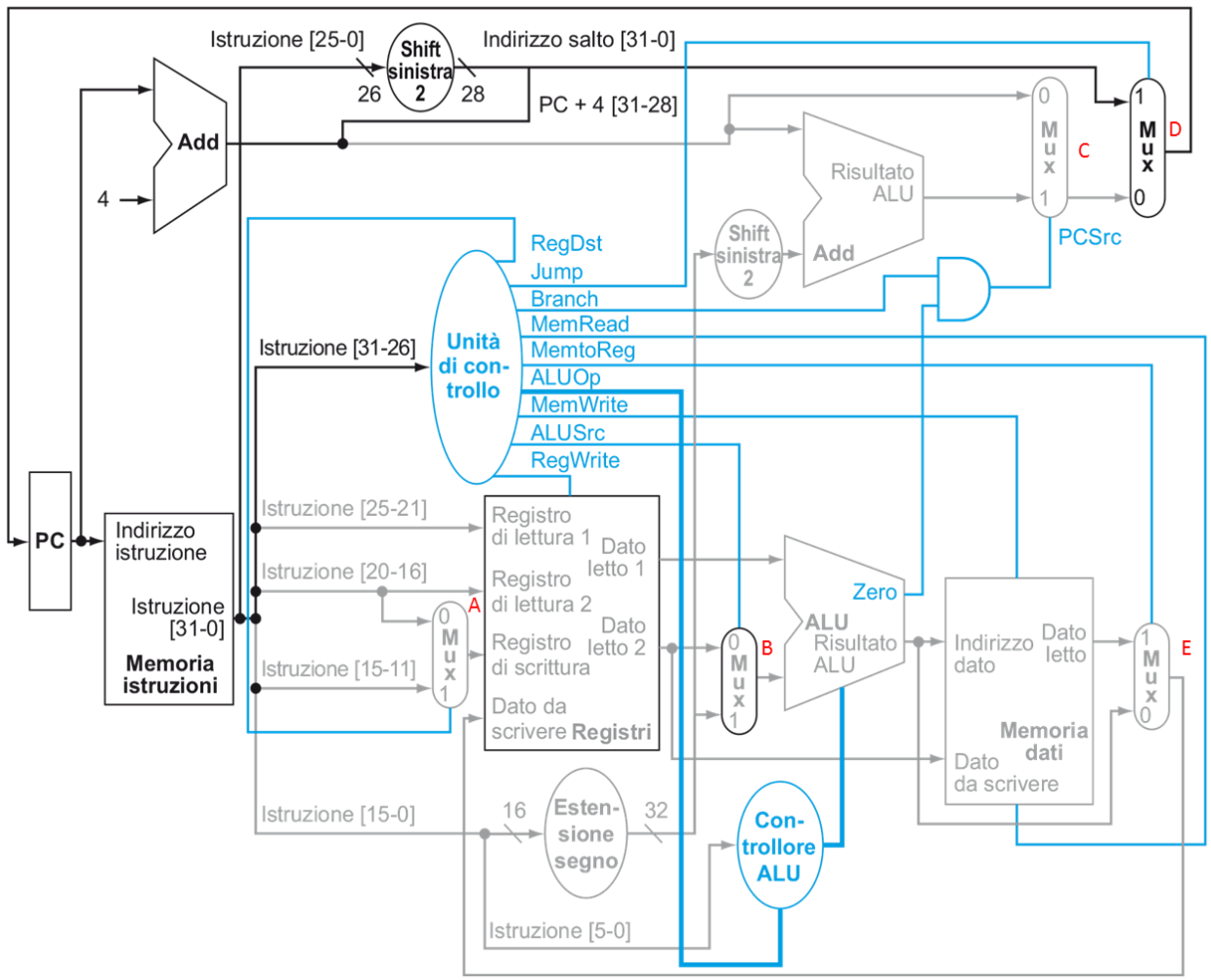
1. Si indichino le dipendenze dati e di controllo che possono causare conflitti.
2. Si esegua il codice inserendo stalli ove necessario, e si calcoli il numero di cicli di clock necessari per eseguire il codice, e il CPI. Per svolgere l'esercizio mostrare mediante uno schema come le istruzioni attraversano i vari stadi del MIPS in corrispondenza dei cicli di clock. Indicare infine quali percorsi di propagazione vengono utilizzati

N° Istr.	Cicli di Clock																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Numero cicli di clock:

$CPI =$

3. Si indichino inoltre, data l'architettura a singolo ciclo di riferimento, quali sono i valori dei segnali di controllo indicarti in figura nel caso dell'esecuzione dell'istruzione LW R2, 0(R0).



Segnale	Valore
RegDst	
Jump	
Branch	
MemRead	
MemToReg	
ALUOp	
MemWrite	
ALUSrc	
RegWrite	
Mux A	
Mux B	
Mux C	
Mux D	
Mux E	

Domanda E

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

Memoria logica: 16 Kbyte

Indirizzo fisico: 16 Kbyte

Dimensione pagina: 2 Kbyte

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

CX: 5K DX: 4K PX: 2K

CY: 8K DY: 3K PY: 2K

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
0	CX0	CY0
1	CX1	CY1
2	DX0	CY2
3	DX1	CY3
4		DY0
5		
6		
7	PX0	PY0

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

1. Il processo P viene creato ed esegue il programma X
2. Il processo P legge un dato all'indirizzo 0x171A
3. Lo stack del processo P cresce fino a 3 KB
4. Il processo Q viene creato ed esegue il programma Y
5. Il processo Q salta all'indirizzo 0x186C
6. Lo stack del processo Q cresce fino a 5 KB
7. Il processo P genera un figlio R
8. Il processo R scrive un dato all'indirizzo 0x17EF
9. Il processo P termina
10. Il processo R chiama una exec ed esegue il programma X, diventando processo S
11. Il processo Q termina

Sapendo che:

1. 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
2. Il caricamento di ulteriori pagine in memoria avviene su richiesta (on demand)
3. Il numero di pagine residenti di R è pari a 3
4. L'indirizzo esadecimale di partenza di X è 0x046C

5. L’indirizzo esadecimale di partenza di Y è 0x15D1
6. Per la sostituzione delle pagine di memoria si utilizza una politica LRU. Almeno una pagina di pila deve sempre rimanere in memoria.
7. L’allocazione delle pagine virtuali nelle pagine fisiche avviene sempre in sequenza
8. 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**
0	
1	
2	
3	
4	
5	
6	
7	

MMU			
Processo	**NPV**	**NPF**	**Valido**

Domanda F

Si descriva nel modo più preciso e chiaro possibile il concetto di thread e si indichino le differenze principali rispetto ad un processo.