10' Esercitazione 2'parte

https://politecnicomilano.webex.com/meet/gianenrico.conti

31 maggio 2021

Gian Enrico Conti

Temi esame

Outline

- Esame 2019 01 24
- Esame 2019 02 15

Esame 2019 01 24

MICROARCHITETTURA

Sia dato un processore MIPS dotato di una pipeline standard a CINQUE STADI, senza propagazione né riconoscimento dei conflitti. Sia inoltre dato il seguente codice assembly.

```
1      ADD   R5, R2, R1
2      LW    R3, 4(R5)
3      LW    R2, 0(R2)
4      OR    R3, R5, R3
5      SW    R3, 0(R5)
```

- Inserire opportunamente delle `nop` per assicurare la corretta esecuzione del codice.

- Risolvere il problema utilizzando le `nop` solo quando i conflitti non si possono evitare neppure riordinando le istruzioni.

Esame 2019 01 24

MICROARCHITETTURA **SOLUZIONE**

Sia dato un processore MIPS dotato di una pipeline standard a CINQUE STADI, senza propagazione né riconoscimento dei conflitti. Sia inoltre dato il seguente codice assembly.

```
1       ADD     R5, R2, R1
2       LW      R3, 4(R5)
3       LW      R2, 0(R2)
4       OR      R3, R5, R3
5       SW      R3, 0(R5)
```

- Inserire opportunamente delle `nop` per assicurare la corretta esecuzione del codice.

```
ADD1
NOP
NOP
LW2
LW3
NOP
OR4
SW5
```

- Risolvere il problema utilizzando le `nop` solo quando i conflitti non si possono evitare neppure riordinando le istruzioni.

```
ADD1
LW3
NOP
LW2
NOP
NOP
OR4
SW5
```

Esame 2019 01 24

- Risolvere infine il problema supponendo che il processore sia dotato di propagazione e identificazione dei conflitti. Indicare inoltre il numero di cicli di clock totali e il CPI.

N° Istruzioni	Cicli di Clock														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Numero cicli di clock:

$CPI =$

Esame 2019 01 24

- Risolvere infine il problema supponendo che il processore sia dotato di propagazione e identificazione dei conflitti. Indicare inoltre il numero di cicli di clock totali e il CPI.

N° Istruzioni	Cicli di Clock														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADD1	F	D	E	M	W										
LW2		F	D	E	M	W									
LW3			F	D	E	M	W								
OR4				F	D	E	M	W							
SW5					F	D	E	M	W						

Numero cicli di clock: 9

$$CPI = \frac{9}{5} = 1.8$$

Esame 2019 01 24

SCHEDULING

Si consideri il seguente insieme di processi:

Processo	Tempo di Arrivo (T_A)	Tempo di Esecuzione (T_E)
P1	0	6
P2	3	2
P3	4	3
P4	7	5
P5	9	1

Si esegua lo scheduling di tali processi secondo i due seguenti algoritmi:

- Round Robin, con un quanto di tempo pari a 3 unità
- Shortest Remaining Time

Per ognuno dei due casi, quindi, si svolgano i seguenti punti:

- Si calcoli il tempo di attesa medio T_W dei processi
- Si calcoli il rapporto di prestazioni R dei processi

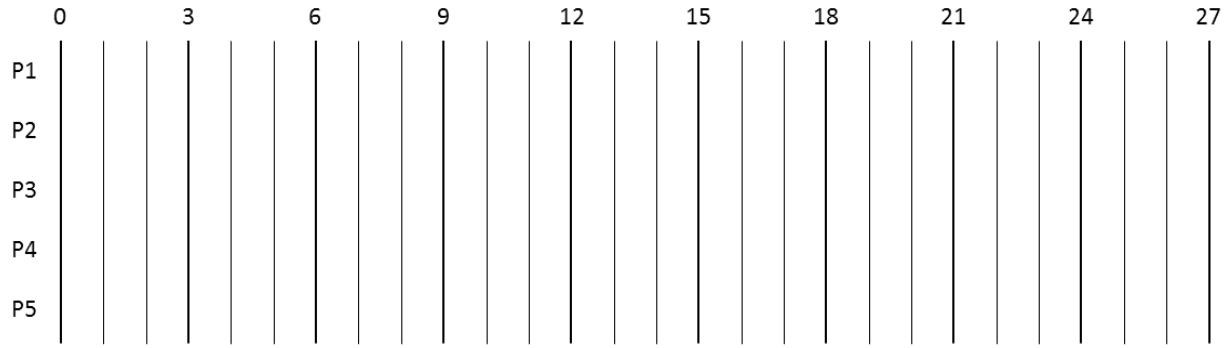
Esame 2019 01 24

SCHEDULING

Si consideri il seguente insieme di processi:

Processo	Tempo di Arrivo (T_A)	Tempo di Esecuzione (T_E)
P1	0	6
P2	3	2
P3	4	3
P4	7	5
P5	9	1

Round Robin



Processo	Tempo di Attesa (T_W)	Rapporto di Prestazioni (R)
P1		
P2		
P3		
P4		
P5		

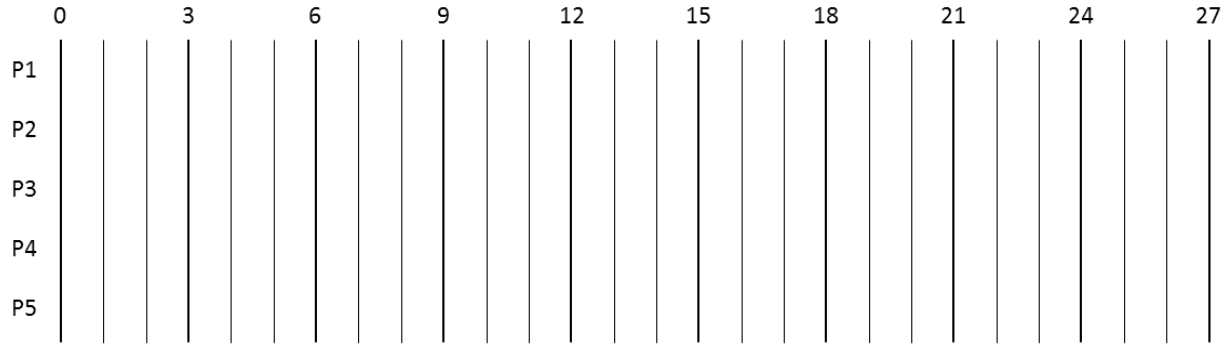
Esame 2019 01 24

SCHEDULING

Si consideri il seguente insieme di processi:

Processo	Tempo di Arrivo (T_A)	Tempo di Esecuzione (T_E)
P1	0	6
P2	3	2
P3	4	3
P4	7	5
P5	9	1

Shortest Remaining Time



Simile ex 09 ...

Processo	Tempo di Attesa (T_W)	Rapporto di Prestazioni (R)
P1		
P2		
P3		
P4		
P5		

Esame 2019 01 24

Sapendo che:

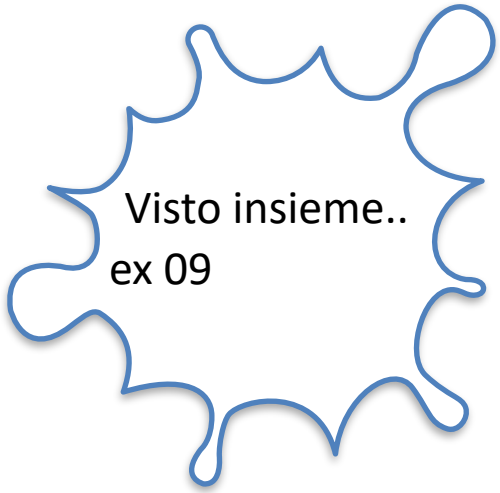
- Il lancio di un programma avviene caricando solamente la pagina di codice con l'istruzione di partenza e una sola pagina di pila;
- Il caricamento di pagine ulteriori è in Demand Paging (cioè le pagine si caricano su richiesta senza scaricare le precedenti fino al raggiungimento del numero massimo di pagine residenti);
- L'indirizzo dell'istruzione di partenza di X è 0x14AF;
- L'indirizzo dell'istruzione di partenza di Y è 0x0231;
- Il numero di pagine residenti R vale 3;
- Viene utilizzato l'algoritmo LRU (ove richiesto prima si de-alloca una pagina di processo e poi si procede alla nuova assegnazione);
- Le pagine meno utilizzate in ogni processo sono quelle caricate da più tempo, con la sola eccezione seguente: se è residente una sola pagina di codice, quella è certamente stata utilizzata recentemente

A un certo istante di tempo t_0 sono terminati, nell'ordine, gli eventi seguenti:

1. Creazione del processo P e lancio del programma Y ("fork" di P ed "exec" di Y);

a. Creazione del processo Q e lancio del programma X ("fork" di Q ed "exec" di X);

2. Accesso ad un dato all'indirizzo 0x3ABC da parte di P;
3. Creazione di una nuova pagina di pila da parte di P;
4. Accesso ad un dato all'indirizzo 0x2158 da parte di Q;
5. Creazione del processo R come figlio di P ("fork" eseguita da P);
6. Creazione di 1 pagina di pila da parte di R.



Si riporti nella tabella seguente una descrizione dello stato della memoria fisica al termine della sequenza di operazioni di cui sopra. Al termine dell'operazione si riporti lo stato dell'MMU per quanto riguarda il processo P.

Esame 2019 01 24

MEMORIA VIRTUALE

Un sistema dotato di memoria virtuale con paginazione e segmentazione di tipo UNIX è caratterizzato dai parametri seguenti: la memoria centrale fisica ha capacità di 32 Kbyte, quella logica di 32 Kbyte e la pagina ha dimensione 4 Kbyte. Si chiede di svolgere i punti seguenti:

- Si definisca la struttura degli indirizzi fisico e logico indicando la lunghezza dei campi NPF, Spiazzamento fisico, NPL, Spiazzamento logico

---

- Nel sistema vengono creati alcuni processi, indicati nel seguito con P, Q, R, S. I programmi eseguiti da tali processi sono due: X e Y. La dimensione iniziale dei segmenti dei programmi è la seguente:

  CX: 8K DX: 4K PX: 4K

  CY: 12K DY: 8K PY: 4K

Si inserisca in tabella sottostante la struttura in pagine della memoria virtuale.

NPV	Programma X	Programma Y
0		
1		
2		
3		
4		
5		
6		
7		

Esame 2019 02 15

MICROARCHITETTURA

Sia dato un processore MIPS dotato di una pipeline standard a CINQUE STADI, senza propagazione né riconoscimento dei conflitti. Sia inoltre dato il seguente codice assembly.

```
1     ADDI  R1, R2, 8
2     SW    R1, 12(R2)
3     BEQ   R5, R4, ETICHETTA # Si supponga R5!=R4
4     ADDI  R4, R3, 1
5     SLT   R5, R6, R4
```

- Inserire opportunamente delle `nop` per assicurare la corretta esecuzione del codice.

...

- Risolvere il problema utilizzando le `nop` sapendo che al processore è stata aggiunta l'ottimizzazione dei PERCORSI DI PROPAGAZIONE.

....

Esame 2019 02 15

MICROARCHITETTURA

Sia dato un processore MIPS dotato di una pipeline standard a CINQUE STADI, senza propagazione né riconoscimento dei conflitti. Sia inoltre dato il seguente codice assembly.

```
1     ADDI  R1, R2, 8
2     SW    R1, 12(R2)
3     BEQ   R5, R4, ETICHETTA # Si supponga R5!=R4
4     ADDI  R4, R3, 1
5     SLT   R5, R6, R4
```

- Inserire opportunamente delle `nop` per assicurare la corretta esecuzione del codice.

```
ADDI1
NOP
NOP
SW1
BEQ
NOP
NOP
NOP
ADDI
NOP
NOP
SLT
```

- Risolvere il problema utilizzando le `nop` sapendo che al processore è stata aggiunta l'ottimizzazione dei PERCORSI DI PROPAGAZIONE.

```
ADDI1
SW1
BEQ
NOP
NOP
NOP
ADDI
SLT
```

Esame 2019 02 15

MICROARCHITETTURA

- Risolvere infine il problema supponendo che il processore sia dotato di PERCORSI DI PROPAGAZIONE e PREDIZIONE STATICA BRANCH NOT TAKEN. Indicare inoltre il numero di cicli di clock totali e il CPI.

N° Istruzioni	Cicli di Clock														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDI1	F	D	E	M	W										
SW2		F	D	E	M	W									
BEQ3			F	D	E	M	W								
ADDI4				F	D	E	M	W							
SLT5					F	D	E	M	W						

Esame 2019 02 15

MICROARCHITETTURA

- Risolvere infine il problema supponendo che il processore sia dotato di PERCORSI DI PROPAGAZIONE e PREDIZIONE STATICA BRANCH NOT TAKEN. Indicare inoltre il numero di cicli di clock totali e il CPI.

N° Istruzioni	Cicli di Clock														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDI1	F	D	E	M	W										
SW2		F	D	E	M	W									
BEQ3			F	D	E	M	W								
ADDI4				F	D	E	M	W							
SLT5					F	D	E	M	W						

Numero cicli di clock: 9

$$CPI = \frac{9}{5}$$

Esame 2019 02 15

FILESYSTEM

Si consideri un calcolatore dotato di sistema operativo Linux in cui valgono le seguenti specifiche:

- Le dimensioni dei blocchi sono di 4096 byte
- Per l'apertura dei file è sempre necessario accedere a:
  - I-node di ogni cartella o file presente nel percorso
  - Blocco per il contenuto di ogni cartella presente nel percorso
  - Primo blocco dati del file

Dato il contenuto del seguente volume:

I-lista: <0,dir,0> <1,dir,1> <2,dir,2> <3,dir,3> <4,dir,4> <5,norm,{100,…,132}> <6,norm,{200,…,208}> <7,norm,{300,…,353}> <8,dir,8> <9,dir,9> <10,dir,10> <20,norm,{400,401,402}> <21,norm,{500}> …

Blocco 0: … <1,bin> <2,home> <3,usr> …
Blocco 1: … <6, cat> <7, grep> …
Blocco 2: … <8, pippo> <9,pluto> …
Blocco 3: … <4, bin> …
Blocco 4: … <5,ls> …
Blocco 8: … <11,.bashrc> …
Blocco 9: … <10,data> …
Blocco 10: … <20, exam.docx> <21, solutions.docx> …

1. Per ciascuna delle chiamate di sistema sotto riportate, si indichi la sequenza di accessi agli I-Node e ai blocchi (del tipo: I-Node X oppure Blocco Y).

Esame 2019 02 15

FILESYSTEM

Si consideri un calcolatore dotato di sistema operativo Linux in cui valgono le seguenti specifiche:

- Le dimensioni dei blocchi sono di 4096 byte
- Per l'apertura dei file è sempre necessario accedere a:
  - I-node di ogni cartella o file presente nel percorso
  - Blocco per il contenuto di ogni cartella presente nel percorso
  - Primo blocco dati del file

Dato il contenuto del seguente volume:

I-lista: <0,dir,0> <1,dir,1> <2,dir,2> <3,dir,3> <4,dir,4> <5,norm,{100,…,132}> <6,norm,{200,…,208}> <7,norm,{300,…,353}> <8,dir,8> <9,dir,9> <10,dir,10> <20,norm,{400,401,402}> <21,norm,{500}> …

Blocco 0: … <1,bin> <2,home> <3,usr> …
Blocco 1: … <6, cat> <7, grep> …
Blocco 2: … <8, pippo> <9,pluto> …
Blocco 3: … <4, bin> …
Blocco 4: … <5,ls> …
Blocco 8: … <11,.bashrc> …
Blocco 9: … <10,data> …
Blocco 10: … <20, exam.docx> <21, solutions.docx> …

1. Per ciascuna delle chiamate di sistema sotto riportate, si indichi la sequenza di accessi agli I-Node e ai blocchi (del tipo: I-Node X oppure Blocco Y).

Chiamata di sistema	Sequenza di accessi
fd = **open**("/home/pluto/exam.docx", O_RDWR)	
fd2 = **open**("/home/pluto/solutions.docx", O_RDWR | O_CREAT, S_IRUSR)	
read(fd, buffer, 555)	

Esame 2019 02 15

I-lista: <0,dir,0> <1,dir,1> <2,dir,2> <3,dir,3> <4,dir,4> <5,norm,{100,…,132}> <6,norm,{200,…,208}> <7,norm,{300,…,353}> <8,dir,8> <9,dir,9> <10,dir,10> <20,norm,{400,401,402}> <21,norm,{500}> …

Blocco 0: … <1,bin> <2,home> <3,usr> …
Blocco 1: … <6, cat> <7, grep> …
Blocco 2: … <8, pippo> <9,pluto> …
Blocco 3: … <4, bin> …
Blocco 4: … <5,ls> …
Blocco 8: … <11,.bashrc> …
Blocco 9: … <10,data> …
Blocco 10: … <20, exam.docx> <21, solutions.docx> …

Chiamata di sistema	Sequenza di accessi
fd = **open**("/home/pluto/exam.docx", O_RDWR)	IN0 B0 IN2 B2 IN9 B9 I10 B10 I20
fd2 = **open**("/home/pluto/solutions.docx", O_RDWR | O_CREAT, S_IRUSR)	IN0 B0 IN2 B2 IN9 B9 I10 B10 I21
read(fd, buffer, 555)	B400

Esame xxx : signals

Il processo padre accetta sulla linea di comando un argomento numerico intero N che deve essere minore di 16,

altrimenti il programma termina.

Il processo padre crea quindi N processi figli Ogni processo figlio genera un numero casuale da 1 a 10 e,

se tale numero è maggiore di 5, invia un segnale al processo padre, quindi termina sempre con stato di uscita pari a 0.

Il processo padre conta il numero di segnali ricevuti e lo stampa a video

Esame 15/09/2017 : signals 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <signal.h>
#include <semaphore.h>

// to test against concorrency, use  a fixed value
// we will add a very high number of procvesses to stress "cont++"

#define TEST_MULTIPLE_ACCESS 1

#ifdef TEST_MULTIPLE_ACCESS
int DEFAULT_N = 5000;
#else
int DEFAULT_N = 5;
#endif


sem_t semaforo;
int cont=0;

void signal_handler(int signal) {
    sem_wait(&semaforo);
    cont++;
    sem_post(&semaforo);
}
```

Esame 15/09/2017 : signals 2

```
int myrand(){
#ifdef TEST_MULTIPLE_ACCESS
    return 100;
#else
    int r = arc4random() % 10 + 1;
    return  r;
#endif

}
```

Esame 15/09/2017 : signals 3

```c
int main(int argc, char ** argv) {
    // read cmd line, but with fall back to default. (it will fault if no param added)
    int N = DEFAULT_N;
    if (argc>1)
        N = atoi(argv[1]);

    pid_t parent_pid = getpid();
    pid_t pid, pid_to_wait[N], pid_ritornato;
    int status;

    signal(SIGUSR1, &signal_handler);

    semaforo = sem_init(&semaforo,0,1);
    for (int i = 0; i < N; i++) {
        pid = fork();

        if (pid == 0) {
            //child
            //printf("child pid %d\n", (int)getpid());
            int g = myrand();
            printf("valore casuale %d\n", g);
            if (g > 5) {
                kill(parent_pid, SIGUSR1);
            }
            exit(0);
        }
        else {
            //parent
            pid_to_wait[i] = pid;
        }

    }
    for (int j = 0; j < N; j++) {
        pid_ritornato = wait(&status);
        // printf("parent - tornato il figlio con pid %d, exit status: %d\n", pid_ritornato, WEXITSTATUS(status));
    }

    printf(" segnali: %d\nbatti invio x uscire.", cont);
    char s[10];
    scanf("%s", s);
    return 0;
}
```

Esame 15/09/2017 : signals 4

LAST NOTE:

Con N basso (5) tutto funziona
Cnn N molto alto (x test..) il numero dei conteggi (avendo definito la TEST_MULTIPLE_ACCESS)

Puo rusultare PIU BASSO nonostante il semaforo:

Il kill rimuove il processo e quindi la call back "signal_handler" non e' invocata.