Politecnico di Milano – Sede di Cremona
Anno Accademico 2019/2020

Architettura dei Calcolatori e Sistemi Operativi

Esame – 29.01.2020

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:

```
int* find( int* haystack, int needle )
```

che cerca il valore intero **`needle`** nell'array **`haystack`** e restituisce il puntatore alla prima occorrenza. Si tenga presente che l'array **`haystack`** è ordinato per valori numerici crescenti. Nel caso il valore cercato non sia presente, la funzione restituisce un puntatore nullo.

Si traduca quindi in assembly in seguente programma C che utilizza la funzione **`strcat()`**:

```
int   h[64] = { 10, 20, 21, ... };
int   n;
int*  p;

int main( void )
{
  n = 76;
  p = find( h, n );
  return 0;
}
```

Domanda B

Si sviluppi il codice di un programma secondo le seguenti specifiche:

- Il programma accetta tre argomenti sulla linea di comando.
    - Il primo argomento (S) può valere solamente 1 o 2 e indica il segnale che il programma dovrà gestire. Se il valore è 1 il programma gestirà il segnale SIGUSR1, se è 2 il programma gestirà il segnale SIGUSR2.
    - Il secondo argomento specifica il nome di un programma eseguibile (P)
    - Il terzo argomento è un parametro numerico generico (N)
- All'avvio il programma contolla gli argomenti sulla linea di comando e se non sono corretti, termina immediatamente con codice di uscita pari a 1. Non è richiesto di verificare se il nome del programma eseguibile corrisponde effettivamente ad un programma presente nel sistema.
- Ciò fatto, il programma rimane in attesa del segnale specificato dall'argomento S senza svolgere alcuna operazione.
- Quando riceve il segnale il programma esegue il programma (P) specificato sulla linea di comando passando a quest'ultimo come argomento il parametro (N)

Per comodità si riporta il prototipo di alcune funzoni utili.

```
pid_t fork(void);
int execl(const char* path,const char* arg,...);
int execlp(const char* file,const char* arg,...);
pid_t wait(int *wstatus);
pid_t waitpid(pid_t pid,int *wstatus,int options);
sighandler_t signal(int signum,sighandler_t handler);
int kill(pid_t pid,int sig);
```

Domanda C

Si consideri un sistema con uno spazio di indirizzamento di 1 GByte, 2 cache set-associative a 8 vie della dimensione rispettivamente di 128Kbyte (DCACHE) e 32Kbyte (ICACHE). Inoltre la dimensione della linea per ogni singola via è pari a 64Byte. Sulla base di queste informazioni si indichi la struttura dell'indirizzo visto dalle cache, descrivendo i 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 100 ns
- Il tempo di accesso alla RAM in modalità burst è di 150 ns per la prima parola e 30 ns per le parole successive
- L'hit rate della DCACHE è pari al 95 %, mentre l'hit rate della ICACHE è pari al 99 %

Si calcolino i tempi medi di accesso alle due cache.

T_{DCACHE}=

T_{ICACHE}=

Si consideri quindi l'esecuzione di un programma in cui il 20% 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}=

Domanda D

Si considerino i seguenti 5 processi:

Processo	Arrival Time	Service Time
P1	0	7
P2	3	2
P3	5	1
P4	11	3
P5	12	2

Si riporti lo scheduling secondo la politica **Shortest Process Next**. In caso di uguale durata, viene scelto il processo con arrival time minore.

Processo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

Si riporti lo scheduling secondo la politica **Shortest Remaining Time**. In caso di uguale durata, viene scelto il processo con arrival time minore.

Processo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				
P5																				

Per ogni processo nei due casi si riporti il tempo reale necessario a completare il processo.

SPN

Processo	Tempo
P1	
P2	
P3	
P4	
P5	

SRT

Processo	Tempo
P1	
P2	
P3	
P4	
P5	

Per ogni politica di scheduling si riporti infine il tempo medio di attesa dei processi

SPN =

SRT =

Domanda E

Si consideri il codice riportato di seguito ed un un processore MIPS privo di ottimizzazioni e privo di unità di identificazione dei conflitti. Per ogni istruzione si indentifichi:

- La tipologia di conflitto che viene generato (DATI, CONTROLLO, N/A)
- Il numero di cicli di stallo che è necessario inserire dopo l'istruzione per garantire il corretto funzionamento del programma

Istruzione	Tipo di conflitto	Numero di stalli
`ADD R0, R1, R2`		
`LOOP: SUB R1, R0, R3`		
`MUL R2, R0, R3`		
`LW R4, 0(R5)`		
`ADDI R4, R4, 4`		
`SUB R5, R6, R7`		
`ADDI R5, R5, 1`		
`BEQ R0, R6, LOOP`		
`MUL R8, R9, R10`		
`ADDI R9, R9, 5`		

Sempre riferendosi allo stesso processore privo di ottimizzazioni, si introducano le istruzione NOP necessarie alla corretta esecuzione del codice e si riporti il nuovo programma nella Tabella "BASE". Quindi si proceda al riordinamento delle istruzioni in modo da ridurre il tempo di esecuzione totale del programma, riportando il risultato nella Tabella "OTTIMIZZATO"

BASE

OTTIMIZZATO

Domanda F

Si descriva la funzione svolta dal Translation Lookaside Buffer ed il contesto in cui esso viene utilizzato.