Politecnico di Milano – Sede di Cremona
Anno Accademico 2023/2024

Architettura dei Calcolatori e Sistemi Operativi

Esame – 09.09.2024

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 RISC-V la **funzione**:

```
int vdiv( unsigned int* X, unsigned int N, unsigned int D )
```

che prende in ingresso un vettore **X** di **N** interi positivi ed un numero intero positivo **D**. La funzione conta il numero degli elementi del vettore **X** che sono esattamente divisibili per **D** e restituisce tale conteggio. Qui di seguito è fornita una traccia di come la funzione deve essere chiamata.

```
        .data
X:      .word 10 22 30 41 11 15 18

        .text
        la      a0, X
        li      a1, 7
        li      a2, 13
        jal     vdiv        # vdiv(X,7,13)
        li      a7, 1
        ecall               # Print result
        li      a7, 10
        ecall               # Exit
```

Soluzione 1

```
vdiv:   mv      t0, a0          # Address of X
        mv      t1, a1          # Number of elements of X
        li      a0, 0           # Will use a0 to count
loop:   lw      t2, (t0)        # t2 = X(i)
        rem     t2, t2, a2      # t2 = X(i) % divisor
        sltiu   t2, t2, 1       # t2 = 1 if reminder is 0
        add     a0, a0, t2      # Updates count
        addi    t0, t0, 4       # Next value in X
        addi    t1, t1, -1      # Decrement elements count
        bne     t1, zero, loop  # Exit condition
        jr      ra              # Returns
```

Soluzione 2

```
vdiv:   mv      t0, a0          # Address of X
        mv      t1, a1          # Number of elements of X
        li      a0, 0           # Will use a0 to count
loop:   lw      t2, (t0)        # t2 = X(i)
        rem     t2, t2, a2      # t2 = X(i) % divisor
        bne     t2, zero, skip  # If reminder not zero does not count
        addi    a0, a0, 1       # Updates count
skip:   addi    t0, t0, 4       # Next value in X
        addi    t1, t1, -1      # Decrement elements count
        bne     t1, zero, loop  # Exit condition
        jr      ra              # Returns
```

Domanda B

Si sviluppi in C un programma run secondo le seguenti specifiche:

- Il programma run accetta due argomenti sulla linea di comando
  - Il primo argomento è il nome di un altro programma eseguibile che si deve trovare in una delle directory elencante nella variabile di ambiente PATH.
  - Il secondo argomento è un parametro che deve essere fornito al programma specificato dal primo argomento al momento dell'esecuzione
- Il programma run deve eseguire il programma ricevuto come primo argomento passando a questo sulla linea di comando il secondo argomento ricevuto.
- Il programma run deve restituire i seguenti stati di uscita
  - 1 se il numero di argomenti passati è errato
  - 2 se il programma eseguito da run termina a causa di un segnale
  - 20 se il programma run non riesce ad eseguire il programma specificato
  - Il doppio del valore ritornato dal programma eseguito da run se tale programma va a buon fine e termina normalmente.

Ecco alcuni esempi di esecuzioni e relativi exit status

Comando	Exit Status	Commento
$> run gnu	1	Il numero di argomenti è errato
$> run gnu 33	16	Il programma gnu esiste, viene eseguito e termina correttamente restituendo come exit status il valore 8
$> run gnatt 21	20	Il programma gnatt non esiste
$> run yak	2	Il programma yak esiste, viene eseguito ma termina a causa del segnale SIGKILL

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main( int argc, char** argv )
{
  if( argc != 3 ) {
    exit( 1 );
  }

  if( fork() == 0 ) {
    execlp( argv[1], argv[1], argv[2], NULL );
    exit( 10 );
  }

  int status;
  wait( &status );
  if( WIFEXITED(status) == 0 ) {
    exit( 2 );
  }
  exit( 2 * WEXITSTATUS(status) );
}
```

Domanda C

Si consideri un sistema con uno spazio di indirizzamento complessivo pari a 256 MByte ed una struttura di cache disgiunta dati e indirizzi, con le caratteristiche seguenti:

	Instruction Cache	**Data Cache**
Associatività	Diretta	Set associativa a 4 vie
Dimensione totale	32 KB	32 KB
Dimensione linea	128 B	256 B (per ogni via)
Tempo di accesso	1 ns	2 ns
Hit rate	99 %	98 %

Si indichi la struttura dell'indirizzo visto dalle cache, descrivendo i vari campi e il loro significato.

Struttura dell'indirizzo della cache istruzioni

Offset = log2(128) = 7 bit

Index = log2(32K/128) = 8 bit

Tag = log2(256M) – 7 – 8 = 28 – 7 – 8 = 13 bit

Struttura dell'indirizzo della cache dati

Offset = log2(256) = 8 bit

Index = log2(32K/(256*4)) = 5 bit

Tag = log2(256M) – 8 – 5 = 28 – 8 – 5 = 15 bit

Sapendo che:

- L'accesso alla memoria avviene a parole di 64 bit
- Il tempo di accesso alla RAM in modalità normale è di 90 ns
- Il tempo di accesso alla RAM in modalità burst è di
  - 120 ns per il primo accesso
  - 60 ns per gli accessi successivi

Si calcoli il tempo medio di accesso alle due cache.

Tempo medio di accesso alla cache dati

T = 0.98 * 2ns + 0.02 * [2ns + 120ns + (256/8-1) * 60ns]

= 1.96ns + 0.02 * [2ns + 120ns + 1860ns]

= 1.96 + 39.64 = 41.6 ns

Tempo medio di accesso alla cache istruzioni

T = 0.99 * 1ns + 0.01 [1ns + 120ns + (128/8-1)*60ns

= 0.99ns + 0.01*[1ns + 120ns + 900ns]

= 0.99ns +10.21ns = 11.2 ns

Domanda D

Si consideri il programma **transform** costituito dai seguenti file:

main.c **print.c, print.h** **calc.c, calc.h**

e che utilizza la libreria **fourier.a** costituita dai seguenti file:

fft.c, fft.h **dft.c, dft.h** **fourier.h**

In merito alla libreria, si tenga presente che il file header **fourier.h** include i file **fft.h** e **dft.h** e non dichiara altri simboli. Inoltre ogni file sorgente include il corrispondente header ed il file **main.c** include gli header **print.h**, **calc.h** e quanto necessario per utilizzare la libreria. Sulla base di quanto specificato si scriva un **Makefile** che esegue il processo di compilazione. Sfruttando i meccanismi di generalizzazione delle regole offerti dal sistema di make si cerchi di minimizzare il numero di regole.

Soluzione 1

```
transform: main.o print.o calc.o fourier.a
	gcc $? -o $@

fourier.a: fft.o dft.o
	ar -r $@ $?
	ranlib $@

%.o:%.c
	gcc -c $< -o $@

print.c: print.h
calc.c: calc.h
main.c: print.h calc.h header.h
fft.c: fft.h
dft.c: dft.h
```

Soluzione 2

```
transform:
	gcc main.o print.o calc.o fourier.a -o transform

fourier.a: fft.o dft.o
	ar -r fourier.a fft.o dft.o
	ranlib fourier.a

%.o:%.c
	gcc -c $< -o $@

print.c: print.h
calc.c: calc.h
main.c: print.h calc.h header.h
fft.c: fft.h
dft.c: dft.h
```

Domanda E

Si considerino le seguenti definizioni globali di variabili:

```
unsigned int X  = 32;

struct {
  unsigned short A;
  char           B;
  unsigned short C;
  int            D;
} S = { 0x8000, 129, 16, 0xDEADBEEF };

char V[8] = { 1, 2, 3, 4, 5, 6, 7, 8 };
```

Sapendo che il codice viene compilato per una architettura 32 bit con organizzazione della memoria little endian, si compilino le seguenti tabelle riportando il contenuto (in esadecimale) della memoria relativa alle variabili, tenendo conto che gli indici rappresentano l'offset rispetto alla prima locazione di memoria usata da una data variabile e ogni cella rappresenta un byte. Si lascino in bianco le celle non utilizzate.

X

16				
12				
8				
4				
0	00	00	00	20
	+3	+2	+1	+0

S

16				
12				
8	DE	AD	BE	EF
4	00	00	00	10
0	00	81	80	00
	+3	+2	+1	+0

V

16				
12				
8				
4	08	07	06	05
0	04	03	02	01
	+3	+2	+1	+0

Offset dei byte rispetto alla base (indirizzo) della parola

Si compili quindi la seguente tabella in cui la colonna "Esito" deve riportare

- "OK" se è possibile determinare il valore dell'espressione
- "CC ERR" se l'espressione produce un errore in fase di compilazione
- "LD ERR" se l'espressione produce un errore in fase di linking
- "RUNTIME" se l'espressione produce un errore a runtime

Nel caso in cui l'espressione sia corretta, se ne riporti il valore espresso sia in forma decimale sia in forma esadecimale:

Espressione	Esito	Valore (DEC)	Valore (HEX)
((unsigned char*)&X)[3]	OK	0	0x00
***((int*)(&S.C))**	OK	16	0x0010
***((int*)(V[1]))**	RUNTIME		
((short int)S.B)	OK	129	0x0081
((char*)&S)[9]	OK	-66	0xBE
((short int)S)	CC ERR		
***((short int*)&S.B)**	OK	129	0x0081

Domanda F

Considerando una architettura RISC-V priva di pipeline, si disegni e si commenti la porzione di datapath che si occupa della gestione del program counter, sia nella normale esecuzione, sia durante l'esecuzione dei salti.

Vedi materiale didattico