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

Architettura dei Calcolatori e Sistemi Operativi

Esame – 27.01.2025

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 oddsum( unsigned int N )
```

che calcola la somma dei primi **N** numeri dispari. Si scriva quindi il codice del programma **main** che chiama la funzione passando a questa il valore **N** e salvando il risultato nella variabile **S**. Qui di seguito è fornita una traccia della sezione dati.

```
        .data
N:      .word   7
S:      .space  4

        .text
main:   la      a0, N
        lw      a0, (a0)
        jal     oddsum
        la      t0, S
        sw      a0, (t0)
        li      a7, 10
        ecall
```

Soluzione 1

```
oddsum: beq     a0, zero, end
        slli    t0, a0, 1       # t0 = 2N
        addi    t0, t0, -1      # t0 = 2N-1 (odd)
        li      a0, 0           # a0 = 0 (sum)
loop:   ble     t0, zero, end   # 2N-1 < 0 ?
        add     a0, a0, t0      # a0 = a0 + 2N-1
        addi    t0, t0, -2      # previous odd
        b       loop
end:    jr      ra
```

Soluzione 2

```
oddsum: li      t0, 1           # t0: odd
        li      t1, 0           # t1: sum
        li      t2, 0           # t2: count
loop:   bge      t2, a0, end    # count >= N ?
        add     t1, t1, t0      # sum
        addi    t0, t0, 2       # next odd
        addi    t2, t2, 1       # counts odds
        b       loop
end:    mv      a0, t1
        jr      ra
```

Domanda B

Si sviluppi in C un programma secondo le seguenti specifiche:

1. Il programma utilizza due thread: **read_text** e **write_text**
2. Il thread **read_text** è costituito da un ciclo che legge una stringa dallo standard input ad ogni iterazione
3. Se la stringa inizia con la lettera **'q'** il ciclo del thread **read_text** termina e restituisce la lunghezza totale delle stringhe lette fino a quel momento

4. Il thread **write_text** stampa su standard output la stringa appena letta dal thread **read_text** e la sua lunghezza
5. La funzione **main()** crea i thread e, quando il thread **read_text** termina, stampa la lunghezza totale lette e termina.

```
#include <...>
#include <pthread.h>
#include <semaphore.h>

char       text[128];
sem_t      sem_read;
sem_t      sem_write;

void* read_text( void* arg )
{
      int count = 0;
      while( 1 ) {
            sem_wait( &sem_write );
            fgets(text, 128, stdin );
            if(text[0] == 'q' ) { break; }
            count += strlen(text) - 1;
            sem_post( &sem_read );
      }
      pthread_exit( (void*) count );
}

void* write_text( void* arg )
{
      while( 1 ) {
            sem_wait( &sem_read );
            printf( "%2d %s", strlen(text), text );
            sem_post( &sem_write );
      }
      pthread_exit( NULL );
}


int main( int argc, char *argv[] )
{
      pthread_t tid_r;
      pthread_t tid_p;
      void*     count;

      sem_init( &sem_read,  0, 0 );
      sem_init( &sem_write, 0, 0 );

      pthread_create( &tid_r, NULL, read_text,  NULL );
      pthread_create( &tid_p, NULL, write_text, NULL );

      sem_post( &sem_write );

      pthread_join( tid_r, &count );
      printf( "Read %d chars\n", (int)count );

      exit(0);
}
```

Domanda C

Si consideri un sistema con uno spazio di indirizzamento complessivo pari a 32 GByte ed una struttura di cache disgiunta dati e indirizzi, con le caratteristiche seguenti:

	Instruction Cache	**Data Cache**
Associatività	Diretta	Completamente associativa
Dimensione totale	128 KB	32 KB
Dimensione linea	256 B	128 B
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(256) = 8 bit

Index = log2(128K/256) = 9 bit

Tag = log2(32G) – 8 – 9 = 35 – 8 – 9 = 18 bit

Struttura dell'indirizzo della cache dati

Offset = log2(128) = 7 bit

Tag = log2(32G) – 7 = 35 – 7 = 28 bit

Sapendo che:

- L'accesso alla memoria avviene a parole di 64 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 il primo accesso
  - 50 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 + 150ns + (128/8-1) * 50ns]

= 1.96ns + 0.02 * [2ns + 150ns + 902ns]

= 1.96ns + 18.04ns = 20 ns

Tempo medio di accesso alla cache istruzioni

T = 0.99 * 1ns + 0.01 * [1ns + 150ns + (256/8-1) * 50ns]

= 0.99ns + 0.01 * [1ns + 150ns + 1550ns]

= 0.99 + 17.01 = 18 ns

Domanda D

Si consideri il seguente codice assembly relativo ad una architettura RISC-V.

```
0x0040009C       bne   x6, x13, end
...
...
0x004000D8 end:  addi  x3, x5, 22
```

Si codifichino le due istruzioni in formato binario. A tale scopo si completino le tabelle seguenti, riportando la dicitura N/A per quelle voci che non si applicano all'istruzione in esame.

bne x6, x13, end	**Valore**	**Codifica binaria**
Format	B	
Opcode	1100011	1100011
Function code 3	0x1	001
Function code 7	N/A	N/A
Source register 1	x6	00110
Source register 2	x13	01101
Destinatin register	N/A	
Offset	0x004000D8 - 0x0040009C = 0x3C = 60	0000 0011 1100
Immediate	0x1D = 30	0000 0001 1110 (0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm12[12,10:5]							rs2					rs1					funct3			imm12[4:1,11]					opcode						
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**
0				**2**				**D**				**3**					**1**			**E**					**6**			**3**			

addi x3, x5, 22	**Valore**	**Codifica binaria**
Format	I	
Opcode	0010011	0010011
Function code 3	0x00	000
Function code 7	N/A	N/A
Source register 1	x5	00101
Source register 2	N/A	N/A
Destinatin register	x3	00011
Offset	N/A	N/A
Immediate	22	0000 0001 0110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm12[11:0]												rs1					funct3			rd					opcode						
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
0				**1**				**6**				**2**					**8**			**1**					**9**			**3**			

Domanda E

Si consideri un sistema di memoria con uno spazio di indirizzamento virtuale di 16MB ed una dimensione di pagina pari a 8KB.

1. Si indichino le dimensioni in bit di:

   Indirizzo virtuale: _____24_____

   Numero di pagina virtuale: _____11_____

   Offset: _____13_____

2. Si completi la seguente tabella, riportando numero di pagina virtuale e spiazzamento sia in forma binaria, sia esadecimale.

Indirizzo virtuale	Numero di pagina virtuale		Offset	
	Hex	Bin	Hex	Bin
0x380CBA	0x1C0	0011 1000 000	0x0CBA	0 1100 1011 1010
0x249FF2	0x124	0010 0100 100	0x1FF2	1 1111 1111 0010
0x00033A	0x000	0000 0000 000	0x033A	0 0011 0011 1010
0xF0F040	0x787	1111 0000 111	0x1040	1 0000 0100 0000
0xCC33CC	0x661	1100 1100 001	0x13CC	1 0011 1100 1100

Domanda F

Si descriva la funzione delle seguenti parole chiave del linguaggio C: **`extern`**, **`static`**, **`const`** e **`volatile`**.

Si veda il materiale didattico.

Base Integer Instructions						
INST	**NAME**	**FMT**	**OPCODE**	**FUNCT3**	**FUNCT7**	**DESCRIPTION**
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ^ rs2
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0
addi	ADD Immediate	I	0010011	0x0		rd = rs1 + imm
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ^ imm
ori	OR Immediate	I	0010011	0x6		rd = rs1 | imm
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm
slli	Shift Left Logical Imm	I	0010011	0x1	imm[5:11]=0x00	rd = rs1 << imm[0:4]
srli	Shift Right Logical Imm	I	0010011	0x5	imm[5:11]=0x00	rd = rs1 >> imm[0:4]
srai	Shift Right Arith Imm	I	0010011	0x5	imm[5:11]=0x20	rd = rs1 >> imm[0:4]
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]
beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm
bge	Branch ≥	B	1100011	0x5		if(rs1 >= rs2) PC += imm
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm
bgeu	Branch ≥ (U)	B	1100011	0x7		if(rs1 >= rs2) PC += imm
jal	Jump And Link	J	1101111	0x0		rd = PC+4; PC += imm
jalr	Jump And Link Reg	I	1100111	0x00		rd = PC+4; PC = rs1 + imm
lui	Load Upper Imm	U	0110111			rd = imm << 12
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)
ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger

Multiply Extension						
INST	**NAME**	**FMT**	**OPCODE**	**FUNCT3**	**FUNCT7**	**DESCRIPTION**
mul	MUL	R	0110011	0x0	0x01	rd = (rs1 * rs2)[31:0]
mulh	MUL High	R	0110011	0x1	0x01	rd = (rs1 * rs2)[63:32]
mulsu	MUL High (S) (U)	R	0110011	0x2	0x01	rd = (rs1 * rs2)[63:32]
mulu	MUL High (U)	R	0110011	0x3	0x01	rd = (rs1 * rs2)[63:32]
div	DIV	R	0110011	0x4	0x01	rd = rs1 / rs2
divu	DIV (U)	R	0110011	0x5	0x01	rd = rs1 / rs2
rem	Remainder	R	0110011	0x6	0x01	rd = rs1 % rs2
remu	Remainder (U)	R	0110011	0x7	0x01	rd = rs1 % rs2