1 Domanda I

Usando la rappresentazione binaria, effettua la sottrazione $1100\ 1100\ 0111\ 1110\ 0011_2 - 0100\ 1000\ 1010\ 1001\ 0100_2$

- ☐ $1000\ 0011\ 1101\ 0100\ 1111_2$
- ☐ $0101\ 0011\ 0000\ 0001\ 1101_2$
- ☐ $0100\ 0011\ 1001\ 1010\ 1001_2$
- ☐ $0101\ 0011\ 1000\ 0110\ 1111_2$
- ☐ Nessuna delle risposte

1.1 Risposta

Dobbiamo effettuare la sottrazione:

1100	1100	0111	1110	0011_2	-
0100	1000	1010	1001	0100_2	=
1000	0011	1101	0100	1111_2	

Quindi, la prima risposta risulta essere quella corretta.

Per assicurarci che il risultato sia corretto, possiamo effettuare la sottrazione tra i due numeri convertiti in base 10 e controllare il risultato.

Convertiamo il primo numero:
$1100\ 1100\ 0111\ 1110\ 0011_2 =$
$= 2^{19} + 2^{18} + 2^{15} + 2^{14} + 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^1 + 2^0 = 837603_{10}$

Dopodiché convertiamo il secondo numero:
$0100\ 1000\ 1010\ 1001\ 0100_2 =$
$= 2^{18} + 2^{15} + 2^{11} + 2^9 + 2^7 + 2^4 + 2^2 = 297620_{10}$

Effettuando la sottrazione in base 10 otteniamo: $837603_{10} - 297620_{10} = 539983_{10}$

Ora convertiamo 539983_{10} in binario per assicurarci che i due risultati coincidano.

Dividendo	Divisore	Quoziente	Resto
539983	2	269991	1
269991	2	134995	1
134995	2	67497	1
67497	2	33748	1
33748	2	16874	0
16874	2	8437	0
8437	2	4218	1
4218	2	2109	0
2109	2	1054	1
1054	2	527	0
527	2	263	1
263	2	131	1
131	2	65	1
65	2	32	1
32	2	16	0
16	2	8	0
8	2	4	0
4	2	2	0
2	2	1	0
1	2	0	1

$539983_{10} = 1000\ 0011\ 1101\ 0100\ 1111_2$
Questo ci conferma la correttezza del risultato ottenuto precedentemente.

2 Domanda II

Seleziona la rappresentazione binaria del numero 129_{10}

- ☐ $1010\ 1010_2$
- ☐ $1000\ 0011_2$
- ☐ $1001\ 0001_2$
- ☐ $1000\ 0001_2$
- ☐ Nessuna delle risposte

2.1 Risposta

Convertiamo 129_{10} in base 2.

Dividendo	Divisore	Quoziente	Resto
129	2	64	1
64	2	32	0
32	2	16	0
16	2	8	0
8	2	4	0
4	2	2	0
2	2	1	0
1	2	0	1

Il risultato è quindi $129_{10} = 1000\ 0001_2$. Di conseguenza, la risposta corretta è la numero 4.

3 Domanda III

Convertire il numero 24.875_{10} in binario con rappresentazione a virgola fissa.

- ☐ $0001\ 1000.1010_2$
- ☐ $0001\ 1000.1110_2$
- ☐ $0001\ 1000.1011_2$
- ☐ $0001\ 1000.1111_2$
- ☐ Nessuna delle risposte

3.1 Risposta

Per convertire 24.875_{10} in base 2 con una rappresentazione binaria con virgola fissa, dobbiamo dividere il numero in parte intera e parte decimale.

Convertiamo 24_{10} in base 2:

Dividendo	Divisore	Quoziente	Resto
24	2	12	0
12	2	6	0
6	2	3	0
3	2	1	1
1	2	0	1

Abbiamo ottenuto $24_{10} = 0001\ 1000_2$.
Ora convertiamo 0.875_{10} in base 2.

Moltiplicando	Moltiplicatore	Risultato decimale	Risultato intero
0.875	2	0.75	1
0.75	2	0.5	1
0.5	2	0	1

Abbiamo ottenuto $0.875_{10} = 0.111_2$
Combinando la parte intera e la parte decimale otteniamo $24.875_{10} = 0001\ 1000.1110_2$. Di conseguenza, la risposta corretta è la numero 2.

4 Domanda IV

Quale di queste affermazioni è corretta nella comparazione tra architetture CISC e RISC?

- ☐ Entrambe consentono di avere operandi sia in memoria che nei registri.
- ☐ Entrambe codificano le istruzioni con un numero di bits costante.
- ☐ L'architettura RISC non supporta operandi "immediati", mentre CISC li supporta.
- ☐ L'architettura RISC non supporta l'esecuzione di istruzioni diverse in base al valore di veritá di una condizione.
- ☐ Entrambe dispongono di operazioni che consentono di scambiare dati tra la memoria e i registri.

4.1 Risposta

Analizziamo le risposte.

1) Le architetture RISC possono accedere alla memoria usando esclusivamente due operazioni: `load` e `store`. Di conseguenza, questa risposta è errata.

2) Le architetture CISC non richiedono che le istruzioni siano codificate con un numero costante di bits. Quindi, anche questa risposta non è quella corretta.

3) Questa opzione non è corretta: entrambe permettono di usare operandi immediati.

4) Le architetture RISC implementano il branching, Quindi è possibile eseguire diverse istruzioni in base al valore di verità di una condizione. Quindi, questa risposta è errata.

5) Questa risposta è corretta: entrambe dispongono di operazioni per spostare i dati tra i registri e la memoria.

5 Domanda V

Il registro `x8` contiene il valore `x8` = 1111 0000 0101 0001 0100 1100 0100 0110. Nota: per semplicità, si consideri l'architettura RV32I. Quale sarà il vaore di `x8` dopo l'esecuzione delle seguenti istruzioni in assembly RISC-V?

```
srli  x5, x8, 28
slli  x5, x5, 4
xor   x8, x8, x8
```

- ☐ `x8` = 1111 0000 0101 0001 0100 1100 0100 0110
- ☐ `x8` = 0000 0000 0000 0000 0000 1100 0000 1011
- ☐ `x8` = 0000 0000 0000 0000 0000 0000 0000 0000
- ☐ `x8` = 1101 0000 0000 0000 1100 0000 0000 0000
- ☐ Nessuna delle risposte

5.1 Risposta

Siamo interessati al valore assunto dal registro `x8` dopo l'esecuzione delle istruzioni mostrate precedentemente.

Le prime due istruzioni modificano il contenuto del registro `x5`, mentre solo l'ultima modifica il contenuto del registro `x8`. Di conseguenza, dato che l'ultima operazione effettua una xor tra `x8` e `x8`, il risultato consisterá nel settare il registro a zero.

Quindi, la risposta corretta è la terza.

6 Domanda VI

La funzione `setV` prende in input il puntatore a due vettori di interi `u`, `v`, e la `dimensione` dei vettori. L'obiettivo è di sommare l'i-esimo elemento di `u` con 1 e salvare il risultato dell'operazione nell'i-esimo elemento di `v`. Il compilatore restituisce la seguente traduzione in assembly RISC-V che, però, è incompleta: manca la riga X1 (Le convenzioni per le gli argomenti e i valori di ritorno specificati dall'ABI, come visto durante il corso, vengono rispettate). Quali degli X1 proposti è adatto alla parte mancante?

```
void sumV(int * u, int * v, unsigned int size){
  for (unsigned int i = 0; i < size; i++) {
    *v++ = *u++ + 1;
  }
  return;
}
```

```
sumV:
        beq     a2, zero, END
        mv      a3, zero
FOR:
        X1
        addi    a0, a0, 4
        addi    a5, a5, 1
        sw      a5, 0(a1)
        addi    a1, a1, 4
        addi    a3, a3, 1
        blt     a3, a2, FOR
END:
        ret
```

- ☐ X1: lw a5, 0(a0)
- ☐ X1: lw a1, 0(a6)
- ☐ X1: lw a5, 4(a0)
- ☐ X1: lw a5, -4(a0)
- ☐ Nessuna delle risposte

6.1 Risposta

L'obiettivo consiste nel trovare la riga mancante.

Come possiamo vedere dallo snippet mostrato precedentemente, il programma aggiunge 1 al valore contenuto in `a5` e ne salva il valore in `v[i]`. Possiamo quindi inferire che il valore di `u[i]` è conservato nel registro `a5`.

Inoltre, per recuperare il valore contenuto in `u[i]`, dobbiamo effettuare una load all'indirizzo specificato dal registro `a0`, quindi l'offset deve essere 0.

Di conseguenza, la risposta corretta è la prima.