Rubric: Quiz 1 CSE 112: Computer Organization

Q1. Imagine a virtual ISA having an instruction size of 32 bits and 64 registers. Suppose the ISA supports an address space of **512 KiloBytes** with byte-addressable memory. The number of **unique** Instructions/Operations supported by the processor is 64. **[10 Points]**

Syntax for Memory type instruction: <Opcode><Memory address> <Register address> <Filler Bits> Syntax for Register type instruction: <Opcode><Source Register Address> <Destination Register Address> <Filler Bits>

- a. What are the minimum bits required to represent the opcode of instruction? [1 Point]
- b. What are the minimum bits required to represent a register uniquely? [1 Point]
- c. What are the minimum bits required to represent a memory address uniquely? [1 Point]
- d. How many filler bits are required in memory type and register type instruction respectively? [1 Point]
- e. For a memory type instruction, with hexadecimal value F4000884, determine the value of opcode, memory address and register address. [1*3 = 3 Points]

(For Set B: Value is E8000886)

f. For a register type instruction, with hexadecimal value 58400000, determine the value of opcode, source register address and destination register address. [1*3 = 3 Points]

(For set B: Value is 68500000)

Answer:

Note: For parts a,b, and c, there is binary marking. Marks are for final answer only.

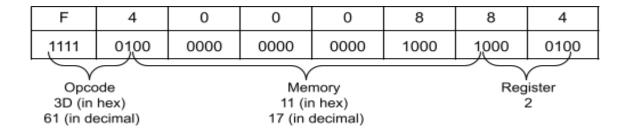
- a. There are 64 unique instructions, hence **6 bits** are needed to represent the opcode of instruction.
- b. There are 64 registers, hence **6 bits** are needed to represent a register uniquely.
- c. For 512 KiloBytes with a byte addressable location, the number of bits can be calculated as:

$$2^{n} = 512 \times 1024 \Rightarrow 2^{n} = 2^{9} \times 2^{10} \Rightarrow 2^{n} = 2^{19} \Rightarrow n = 19$$

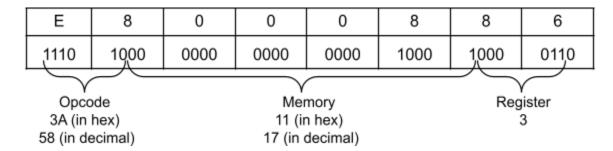
d. Number of filler bits in memory type = 32 - (opcode bits + memory address bits + register address bits) = 32 - (6 + 19+6) = 1 [0.5 Points]

Number of filler bits in register type = 32 - (opcode bits + source register address bits + + source register address bits) = 32 - (6 + 6+6) = 14 [0.5 Points]

e. For set A:

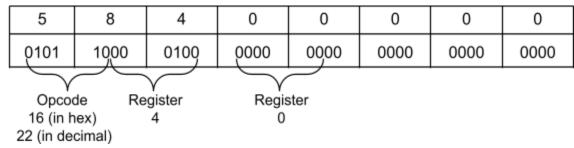


For set B:

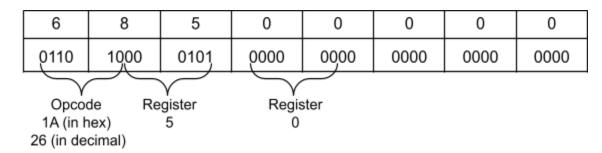


(For both sets, **1 mark each** for opcode, memory address, and register address) (Answers in binary, decimal or hexadecimal)

f. For set A:



For set B:



(For both sets, **1 mark each** for opcode, source register address, and destination register address) (Answers in binary, decimal or hexadecimal)

[10 Points]

Q2. Solve the following:

(a) Find "x" if $(113)_x = 23$ [2 Points] (b) $1100.0101_2 = (?)_{10}$ [2 Points] (c) (Last two digits of your roll no.)₁₀ = $(?)_{16}$ [1 Point] (d) Ideally, how many bits are required to represent 0.4₁₀ in binary form? [1 Point] Use 6 bits after decimal point to represent 0.4₁₀ in binary [1 Point] (ii) Use 6 bits after decimal point to represent 0.2₁₀ in binary [1 Point] (iii) Using part (d)-(ii), compute $0.2_{10} + 0.2_{10}$ in binary and compare with the value obtained in part (d)-(i). Report the error (if any). [2 Points]

Answer:

(a):
$$1.x^2 + 1.x + 3 = 23$$

 $x^2 + x - 20 = 0$
 $(x+5)(x-4) = 0$
 $x = -5,4$ [1 Point to show calculation and arrive at two possible solutions]
 $\Rightarrow x = 4$ [1 Point to choose final correct answer]

(b): 1100.0101 can be broken down into pre and post-decimal parts.

$$1100 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8 + 4 = 12$$
 [1 Point]

.0101 =
$$0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0.25 + 0.0625 = 0.3125$$
 [1 Point]

Final Answer = 12.3125

(c):

Example - PhD21106 -> 106

106 divided by 16 has a quotient of 6 and a remainder of 10. Final answer: 6A. [1 Point]

Alternative: $106_{10} = 1101010 \Rightarrow 0110_{-}1010 \Rightarrow 6_{-}A \Rightarrow 6A$ [1 Point]

(d):

Ideally, infinite bits or can not be determined or any equivalent answer [1 Point]

Reason: 0.4 = 0.0110011001100...

The underlined part repeats itself.

- (i) $0.4_{10} = 0.011001$ [1 Point]
- (ii) $0.2_{10} = 0.001100$ [1 Point]
- (iii) $0.2 + 0.2 \Rightarrow$

0.001100

- + 0.001100
 - 0.011000 [0.5 Points to show calculation + 0.5 Points for correct answer]

Error:

0.011001 (0.4) - 0.011000 (0.2+0.2) = 0.000001 [0.5 Points to show error calculation + 0.5 Points for correct error]

- Q3. Operand Op1 = $(1000)_2$, Operand Op2 = $(7)_{10}$
 - (a) Evaluate the decimal value of Op1, by considering the given value in Op1 to be already present in, [0.5*4 = 2 Points]
 - (i) 4-bit Unsigned binary form
 - (ii) 4-bit Signed binary form
 - (iii) 4-bit 1's complement binary form
 - (iv) 4-bit 2's complement binary form
 - (b) Mention any one method to calculate the overflow in 2's complement addition.

 Derive its boolean expression also.

 [2 + 1 = 3 Points]
 - (c) Evaluate the following using 4-bit 2's complement arithmetic, assuming Op1 to be already in 2's complement representation, and report the final value in decimal:

[2*2 = 4 Points]

- (i) Op1 + Op2
- (ii) Op1 Op2
- (d) Calculate the presence of overflow in part (c)-(i) & (c)-(ii) using the boolean expression derived in (b). [0.5*2 = 1 Point]

_					
Δ	n	c	۱A	ıΔ	r.

- swer:
 (a) Op1 = (1000)₂ [0.5 Points per sub-part]
 (i) 8
 (ii) -0
 (iii) -7
 (iv) -8
- (b) If two "n" bit operands are added, then over flow is given by checking either of two conditions:
 - (i) Checking if the nth and (n+1)th MSBs carry out have same values, as different values indicate there has been a sign change in final answer but same values just mean **sign-extended** result. **[2 Points]**Boolean Expression:

Overflow: c_out_n XOR c_out_{n+1} [1 Point]

(ii) Checking if the MSB of result matches with MSB of the added operands, if there is a sign change, that implies the presence of extra bit as result has a different sign as compared with the signs of operands. [2 Points]

Boolean Expression (n-bit operands A & B and n-bit sum S): [1 Point]

Overflow: An'Bn'Sn + AnBnSn'

```
(c) Op2 = 0111_2
```

(i) 1000 + 0111

1111 (2's complement form) [1 Point to show calculation & conversion]

Decimal = -1_{10} [1 Point for correct decimal answer with Sign]

(ii) 2's complement of Op2 = 1001_2

Op1 - Op2 = Op1 + 2's comp. Op2 1000 + 1001

= (1)0001 [1 Point to show calculation & conversion]

Two cases:

(i) Discard MSB since it is 4-bit arithmetic.

Decimal (using 4-bits) = $+1_2$ [1 Point for correct decimal answer with Sign]

(ii) Not discarding the overflow bit:

Convert in 2's complement form: 01111 and sign-bit = $1 \Rightarrow$ -ve number

Decimal (using 5-bits) = -15_2 [1 Point for correct decimal answer with Sign]

(d) Overflow = An'Bn'Sn + AnBnSn' .. (1) Overflow = $c_out_n XOR c_out_{n+1}$.. (2)

For both parts.

0.25 Points for showing boolean-calculation

0.25 Points for indicating Overflow/No-overflow

```
(i) Using (1):
    Overflow = 0.0.1 + 1.0.0 = 0
    Using (2):
    Overflow = 0 XOR 0 = 0
    No overflow
```

(ii) Using (1):
 Overflow = 0.0.0 + 1.1.1 = 1
 Using (2):
 Overflow = 0 XOR 1 = 1
 Overflow detected