CSE 112: IIIT-Delhi, Winter 2023

Computer Organization

Quiz I Rubric

Date: April 06, 2023

Problem I: Signed Unsigned Representation **(5 Points)**

Consider the following operation to be performed: Operand1 = $(-255)_{10}$, Operand2 = $(-250)_{10}$.

(a) Write the 1's complement representation of both the operands in 12 bits.

(b) State the rule for getting 2's complement representation from 1's complement representation.

(c) Use this rule to get 2's complement representation of the mentioned numbers.

i) Using 12 bits ii) using 9 bits

(d) Perform the addition operation in 2's complement form itself. Show your computation.

i) Using 9 bits ii) Using 12 bits

(e) Report the minimum bits required to represent the above computed output correctly in 2's complement notation.

Solution I: **1 x 5 Points**

Operand_1 = $(-255)_{10}$

Operand_2 = $(-250)_{10}$

Range of 1's complement representation using n-bits = $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

For n = 9, Range of representable Numbers[-255,255]

For n = 12, Range of representable numbers [-2047, 2047]

a)

Decimal Representation	Unsigned representation	9-bit representation (1's complement)	12-bit representation (1's Complement)
+ 255	$(1111\_1111)_2$	$(0\_1111\_1111)_2$	$(0000\_1111\_1111)_2$
- 255	Not Representable	$(1\_0000\_0000)_2$	$(1111\_0000\_0000)_2$
+ 250	$(1111\_1010)_2$	$(0\_1111\_1010)_2$	$(0000\_1111\_1010)_2$
- 250	Not Representable	$(1\_0000\_0101)_2$	$(1111\_0000\_0101)_2$

b) Rule Of Getting 2's complement representation using 1's complement representation

(2's complement) = (1's complement) + $(1)_2$.

c) Range of 2's complement representation using n-bits = $-(2^{n-1})$ to $+(2^{n-1} - 1)$

	Decimal Representation	Unsigned representation	9-bit representation	12-bit representation
	+ 255	$(1111\_1111)_2$	$(0\_1111\_1111)_2$	$(0000\_1111\_1111)_2$
1's Complement	- 255	Not Representable	$(1\_0000\_0000)_2$	$(1111\_0000\_0000)_2$
2's Complement	- 255	Not Representable	$(1\_0000\_0001)_2$	$(1111\_0000\_0001)_2$
	+ 250	$(1111\_1010)_2$	$(0\_1111\_1010)_2$	$(0000\_1111\_1010)_2$
1's Complement	- 250	Not Representable	$(1\_0000\_0101)_2$	$(1111\_0000\_0101)_2$
2's Complement	- 250	Not Representable	$(1\_0000\_0110)_2$	$(1111\_0000\_0110)_2$

d) **Addition** operation in 2's complement representation.
Rules : Discard the carry in 2's complement.
[ ] : Represents Carry bit generated.
[ ] : Represents Sign Bit.

Number	2's complement representation (9-bits)	2's complement representation (12-bits)
- 255	(1_0000_0001)$_2$	(1111_0000_0001)$_2$
- 250	(1_0000_0110)$_2$	(1111_0000_0110)$_2$
Result	(1_0_0000_0111)$_2$	(1_1110_0000_0111)$_2$
	Note : Two negative numbers after addition is giving a positive number. It's a case of overflow.	

e) Minimum Number of bits required to represent the computed result in 2's complement representation.
-250 + (-255) = -505
When n = 9. Range [-256, 255]
When n = 10. Range [-512,+511]
So, n = 10 will satisfy our requirements.

Problem II: Opcode Assignment and Instruction encoding **(10 Points)**

Imagine a virtual ISA having instruction size of 32-bits, having 32 registers. Suppose the ISA supports an address space of **1MegaBytes** with Byte addressable memory. The number of **unique** Instructions/Operations supported by the processor is 32.

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

a) Assign opcodes to the below-listed Operations/Instructions in Binary/Hex format.
b) Form a complete Binary/Hex instruction code using these opcodes.

Instruction (mnemonic)	Operation	Opcode (Hex)	Instruction Code (Hex)
Mov A,B	Move/Copy the contents of Register B into A(B = A).		
Add C,D	Add registers C and D and store the result in register C(C = C + D).		
LDR A, Address	Load register A with the data stored at the given address(mentioned below).		
STR B, Address	Store the content of register B at the specified address(mentioned below).		

Address = (0x00_0000_1000)$_{16}$ + (last_three_digit_of_your_roll_number * 4)$_{10}$.
e.g Roll_Number = abcd2468; Address = (468 * 4)$_{10}$ + (0x00_0000_1000)$_{16}$.
Note: Students can opt to answer the question in Binary representation also.

Solution II:

Instruction Size = 32 bits.
=> Each instruction will be formed using 32 bits only.
Opcode_bits + Filler_Bits + Reg_addr_bits + Mem_Address_bits = 32.
Opcode_bits + Filler_bits + Dest_Reg_addr_bits + Source_Reg_addr_bits = 32.

Number of General Purpose Registers = 32.
=> We need to represent each register_addr uniquely using a binary sequence.
We need 32 such unique binary sequences.
$32 = (2)^5$. We need a minimum 5-bits to have 32 unique binary sequences or a unique sequence for each register address.
Implies that the address of each register will be of 5-bits only.

Address space supported by ISA is 1 MegaByte with byte addressable location.
1 MegaByte = 1 Kilo x 1 Kilo x 1 Byte.
1 Kilo = $(2)^{10}$.
1 Mega = $(2)^{20}$.
As all the memory locations are byte addressable we can access one complete byte at once.

So, Number of bits required to represent each address uniquely is 20-bits only.
Note : For the complete above calculation. **2 Points**

a) Assign opcode to the mentioned instructions.
Number of unique Instructions supported by processor = 32.

	Opcode (mnemonic)
Similar Instruction	Mov A,B Mov C,D

Different Instruction	Mov A,B
	Add A,B
	Ldr A,B

We need 32 unique binary representations(Opcodes) to represent each instruction uniquely.Similarly as discussed above we need minimum 5-bits to represent each instruction or each opcode will be of 5-bits only.

2 Points

Instruction	One possible Assignment(5-bit) (to be taken forward)	Other possible Assignment(5-bit)
Mov	$(00000)_2$	$(00100)_2$
Add	$(00001)_2$	$(00101)_2$
Ldr	$(00010)_2$	$(00110)_2$
Str	$(00011)_2$	$(11111)_2$

Note : Any binary sequence whether random or in a sequence can be assigned to the opcodes. But, make sure that no two opcodes have the same binary sequence.

2 Points

Register	One possible Assignment(5-bit) (to be taken forward)	Other possible Assignment(5-bit)
A	$(00000)_2$	$(10100)_2$
B	$(00001)_2$	$(10101)_2$
C	$(00010)_2$	$(10110)_2$
D	$(00011)_2$	(11111

Let a student's Roll Number is: abcd2468
Address := $(468 * 4)_{10}$ = $(1872)_{10}$
$(1872)_{10} = (750)_{16}$
Address = $(0x00\_0000\_1000)_{16} + (0x0750)_{16} = (0x00\_0000\_1750)_{16}$

Now Let's form the complete instruction.

Register Type: **2 Points**

Instruction	Opcode_bits (5-bits)	Filler_bits (17-bits)	Dest_Reg_address (5-bits)	Source_reg_address (5-bits)
MOV A,B	$(00000)_2$	$(000……00000)_2$	$(00000)_2$	$(00001)_2$
ADD C,D	$(00001)_2$	$(000…...00000)_2$	$(00010)_2$	$(00011)_2$

Memory Type: **2 Points**

Instruction	Opcode_bits (5-bits)	Filler_bits (2-bits)	Reg_address (5-bits)	Mem_address (20-bits)
LDR A, Address	$(00010)_2$	$(00)_2$	$(00000)_2$	$(0000\_0001\_0111\_0101\_0000)_2 = (0\_1750)_{16}$
STR B, Address	$(00011)_2$	$(00)_2$	$(00001)_2$	$(0000\_0001\_0111\_0101\_0000)_2 = (0\_1750)_{16}$

Problem III: Assembly Programming **(5 Points)**

a) Write an assembly program to swap two numbers that are stored at two different memory locations. Instructions that are available to the programmer.

Instruction (mnemonic)	Operation
Mov A,B	Move/Copy the contents of Register B into A (A = B).
Add C,D	Add registers C and D and store the result in register C (C = C + D).
LDR A, Address	Load register A with the data stored at the given address.
STR B, Address	Store the content of register B at the specified address.

Note: Students can choose any memory locations where the variables are stored.

Solution III: **5 Points**

Write a Program to swap two numbers that are initially stored at two different memory locations using the instructions mentioned in Question-2.

As the student has the liberty to choose a memory location on their own.
Let first variable stored at : $(0x0\_0000)_{16}$
Let Second variable stored at : $(0x0)0004)_{16}$

Pseudo Code :
- Step_1 : Read both the variables from the memory.
- Step_2 : Store the 1st variable at the 2nd variable's memory location.
- Step_3 : Store the 2nd variable at the 1st variable's memory location.

Assembly Program :

```
LDR A, [0x0_0000]        // load the first variable into register A
LDR B, [0X0_0004]        // load the second variable into register B
STR A, [0X0_0004]        // Store the first variable at 2nd variable's location
STR B, [0X0_0000]        // Store the 2nd variable at 1st variable's location
```

Note : This is just one sample program.Programs following other algorithms are also possible.

Problem IV: Radix Conversion and Binary Algebra **(5 Points)**

a) $(101010\,.\,0110)_2 = (\ \ )_{10}$
b) $(\text{Last\_three\_digit\_of\_your\_roll\_number})_{10} = (\ \ )_8$.
c) $(123.3)_{10} \cong (\ \ )_2$.
d) Determine x if $(10400)_x = (725)_{10}$.
e) Multiply the following numbers after converting them into binary. And, represent the result in binary.
$(24)_{10} * (20)_{10}$

Solution IV: **1 x 5 Points**

a) $(101010.0110)_2 = (?)_{10}$

$(101010)_2 = 1\text{x}2^5 + 0 + 1\text{x}2^3 + 0 + 1\text{x}2^1 + 0$
$= 32 + 8 + 2$
$= 40$

$(.110)_2 = 0 + 1\text{x}2^{-2} + 1\text{x}2^{-3} + 0$
$= 0 + 0.25 + 0.125$
$= 0.375$

$(101010.0110)_2 = (42.375)_{10}$

b) Let a student's roll number = abcd1234
$(234)_{10} = (?)_8$

	234	
8	29	2
8	3	5

$(352)_8$

c) $(123.3)_{10} \approx (?)_2$

2	123	
2	61	1
2	30	1
2	15	0
	7	1

$(123)_{10} = (111\_1011)_2$

$(0.3)_{10} = (?)_2$

0.3 x 2 = 0.6	Step_begin
0.6 x 2 = 1.2	
0.2 x 2 = 0.4	
0.4 x 2 = 0.8	
0.8 x 2 = 1.6	
0.6 x 2 = 1.2	Step_end

Note: we can notice that the sequence is repeating itself for this decimal number. Or, we can't represent that decimal number completely in binary. We need to go with an approximation.

$(0.010011)_2 < (0.3)_{10} < 0.010010)_2$

Any number lying in btw can be used to represent $(0.3)_{10}$ into binary approximately.

$(123.3)_{10} \approx (0111\_1011.0100\_11)_2$

d) $(10400)_x = (725)_{10}$
$1*X^4 + 0*X^3 + 4*X^2 + 0*X^1 + 0 = 7*10^2 + 2*10^1 + 5$
$X^2 (X^2 + 4) = 5 * 5 * 29$
$X = \mp 5$
But, as radix can't be negative so +5 is the suitable answer.

e) $(24)_{10} * (20)_{10}$
$(11000)_2 * (10100)_2$

Any multiplication operation by power of two leads to a left shift of the other number by that power.
E.g multiplication by 2 => left shift by 1-bit
Multiplication by 8 => left shift by 3-bit

X * 20 = X * 16 + X * 4
=> Multiply with 16, left shift X by 4-bits
=> Multiply with 4, left shift X by 2-bits
Then add both of the shifted versions of X as per algorithm.

$(24)_{10} = (11000)_2$

Multiply with 16, left shift (24) by 4-bits	$(11000\_0000)_2$
Multiply with 4, left shift (24) by 2-bits	$(\text{---}11000\_00)_2$
Addition Result	$(11\_110\_0000)_2 = 256 + 128 + 64 + 32 = (480)_{10}$

Note : There can be some typos in the rubric. Please feel free to notify.