CSE 112:
Computer Organization

Mid Semester Exam Rubric

IIIT-Delhi, Winter 2023

Date: April 24, 2023

Problem I: Number Representation **2 + 4 + 4 = 10 Points**

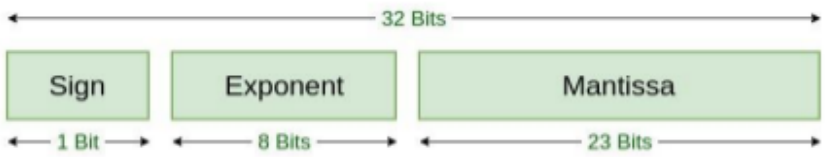
Answer the following problems.

a) What is the condition for overflow in 2's complement addition.
b) Describe the single precision IEEE 754 Floating Point normal number format.
c) Find the range of both positive and negative single precision denormal numbers.

Solution I

a) The condition for overflow in 2's complement addition are
   i) Sum of two positive numbers gives a negative number.
   ii) Sum of two negative numbers gives positive results.

b)



Single Precision
IEEE 754 Floating-Point Standard

The exponent is in biased representation. The exponent (X) is represented by a number E, where E = X + bias. And, the bias is 127.

If the student has clearly mentioned the sign, exponent, mantissa bits, and bias give 4 marks, 1 mark for each component.

c) The largest positive denormal single precision floating point number is 0.111….1 x 2^{-126} (number of 1 after the decimal point is 23).**The result is equal to (1 - 2^{-23}) x 2^{-126} =2^{-126} - 2^{-149}.**

The smallest positive denormal single precision floating point number is 0.000…01 x 2^{-126} = 2^{-149} (number of 0 between the decimal point and 1 is 22).

The smallest negative denormal single precision floating point number is - 0.111….1 x 2^{-126} (number of 1 after the decimal point is 23).

The result is equal to -(1 - 2^{-23}) x 2^{-126} = -(2^{-126} - 2^{-149}).

The largest negative denormal single-precision floating point number is -0.000…01 x 2^{-126} = -2^{-149} (number of 0 between the decimal point and 1 is 22).

Note: Give 1 mark for each.

Problem II : Design and Timing Analysis of Carry Select Adder Circuits **5 + 5 = 10 Points**

a) You need to design two different carry select adder circuits for the addition of two 16-bit unsigned binary numbers.

One carry select adder circuit should be designed using 4-bit binary adder module only and the other carry select adder circuit should be designed using 8-bit binary adder module only.

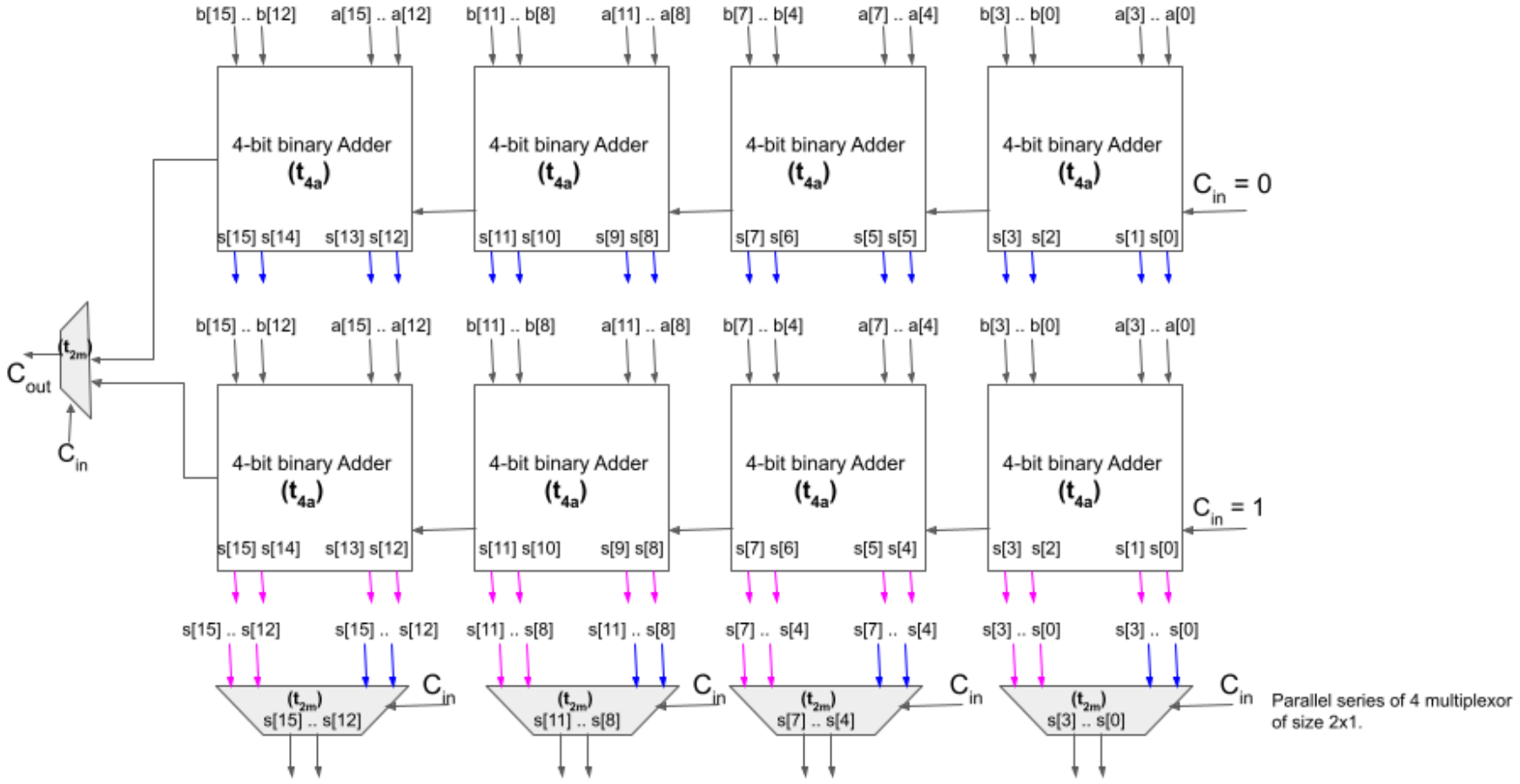
Note: The Hardware components available to us in the form of blocks are 4x1 Multiplexer, 8x1 Multiplexer, 2x1 Multiplexer, 4-bit binary adder, 8-bit binary adder.

b) Calculate the longest path delay in both the circuits and write the equation in the form of the below-mentioned variables..

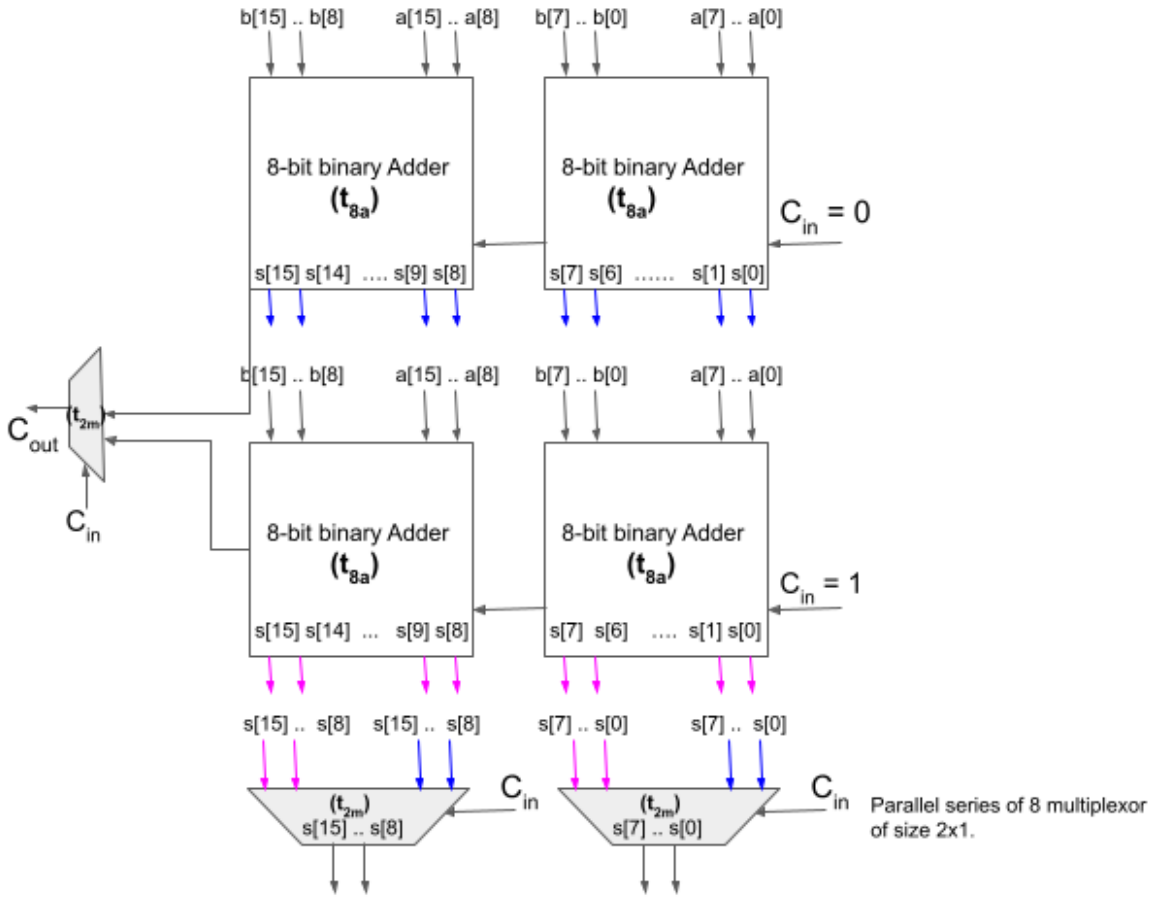
Timing Notations: Time taken by 4x1 Multiplexer = "$\mathbf{t_{4m}}$".
Time taken by 8x1 Multiplexer = "$\mathbf{t_{8m}}$".
Time taken by 2x1 Multiplexer = "$\mathbf{t_{2m}}$".
Time taken by 4-bit binary adder = "$\mathbf{t_{4a}}$".
Time taken by 8-bit binary adder = "$\mathbf{t_{8a}}$".

Solution II

Design of Carry Select Adder



Total time taken to calculate the sum = $4*t_{4a} + t_{2m}$.



Total time taken to calculate the sum = $2*t_{8a} + t_{2m}$.

Note: Link of google slide of the above figures.

[16-bit_adder]

Problem III: Instruction Encoding **1+ 7 + 2 = 10 Points**

List of available instructions to programmers.

Instruction	**Semantics**	**Syntax**
Addition	reg1=reg1 + reg2	Add reg1, reg2
Subtraction	reg1=reg1 - reg2	Sub reg1, reg2
Multiplication	reg1=reg1 * reg2	Mul reg1, reg2
load	Load data from addr to reg1	Ld reg1 addr
move	Move content of reg2 to reg1	Mov reg1, reg2
Branch if equal to zero	Branch to addr if reg1 is equal to zero	Bz reg1 addr
Branch if not equal to zero	Branch to addr if reg3 is not equal to zero	Bnz reg3 addr

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>

In addition to the given instructions, you have 16 registers. Given that the opcode and register should be encoded in minimum binary representation and ISA can access 8-bit memory addresses.

Answer the following

1. What is the minimum number of bits required to represent the opcode of the given instructions.
2. Represent the above-mentioned instructions in terms of machine code using given syntax and within 16-bits only.

Note: The address to be chosen is $(0xAB)_{16}$.

3. Report the number of unused bits in all cases.

Solution III

1. Since as per the above instruction table given. We have only 7 types of unique instructions possible. Now, to assign a unique binary to each opcode we can follow the below expression.
   n : minimum number of bits required to represent an opcode in binary.
   $2^n \geq$ (No of unique instructions).
   Clearly all possible **n ≥ 3** will satisfy this relation. So, the minimum no of bits required to represent each opcode as per above given instruction table is 3.

2. To begin with firstly we need to assign opcode to each unique instruction operation and map each register to a unique register address.. The only constraint is **"opcode and register should be encoded in minimum binary representation"**. So, we have the liberty to map any binary representation to any instruction operation. And, for the register address mapping we need to be careful as we have 16 registers.We need 4 bits to map each register to a unique address.

   Memory address encoding

   The memory address constraint given to us is **"ISA can access 8-bit memory addresses"**. We have only 8-bits to represent each unique memory address. The memory address given to us is $(0xAB)_{16} = (1010\_1011)_2$.

   Opcode Mapping

Instruction Operation	Add	Sub	Mul	Ld	Mov	Bz	Bnz
Opcode map_1	$(100)_2$	$(101)_2$	$(110)_2$	$(111)_2$	$(011)_2$	$(010)_2$	$(001)_2$
Opcode map_2 (to be taken forward)	$(000)_2$	$(001)_2$	$(010)_2$	$(011)_2$	$(100)_2$	$(101)_2$	$(110)_2$

Register Address Mapping

Register_Number	reg0	reg1	reg2	reg3	reg4	reg5	reg6	reg7
Address_1	$(0100)_2$	$(0101)_2$	$(0110)_2$	$(0111)_2$	$(1000)_2$	$(0000)_2$	$(0001)_2$	$(0010)_2$
Address_2 (to be taken forward)	$(0000)_2$	$(0001)_2$	$(0010)_2$	$(0011)_2$	$(0100)_2$	$(0101)_2$	$(0110)_2$	$(0111)_2$

Register_Number	reg8	reg9	reg10	reg11	reg12	reg13	reg14	reg15
Address_1	$(0011)_2$	$(1001)_2$	$(1010)_2$	$(1011)_2$	$(1100)_2$	$(1101)_2$	$(1110)_2$	$(1111)_2$
Address_2 (to be taken forward)	$(1000)_2$	$(1001)_2$	$(1010)_2$	$(1011)_2$	$(1100)_2$	$(1101)_2$	$(1110)_2$	$(1111)_2$

Instruction Encoding

Note:

a) The below-mentioned green colored instructions come under the category of control instructions. But, here as per syntax they are kept in the Memory instruction category.
b) It is not specified that filler bits should be chosen as Zero/One. So, any choice will work.
c) It is given that the instruction must be encoded within 16-bits only. But, there is no restriction on choosing less than 16-bits for instruction encoding.while evaluating make sure that the encoded instructions (both memory and register type) must be of equal length.

Memory Type Instruction.

Instruction	<Opcode>	<Filler Bits>	<Register address>	<Memory address>
Ld reg1 addr	$(011)_2$	$(0)_2$	$(0001)_2$	$(1010\_1011)_2$
Bz reg1 addr	$(101)_2$	$(0)_2$	$(0001)_2$	$(1010\_1011)_2$
Bnz reg3 addr	$(110)_2$	$(0)_2$	$(0011)_2$	$(1010\_1011)_2$

Register Type Instruction.

Instruction	<Opcode>	<Filler Bits>	<Destination Register Address>	<Source Register Address>
Add reg1, reg2	$(000)_2$	$(00000)_2$	$(0001)_2$	$(0010)_2$
Sub reg1, reg2	$(001)_2$	$(00000)_2$	$(0001)_2$	$(0010)_2$
Mul reg1, reg2	$(010)_2$	$(00000)_2$	$(0001)_2$	$(0010)_2$
Mov reg1, reg2	$(100)_2$	$(00000)_2$	$(0001)_2$	$(0010)_2$

3. Number of Unused bits = Number of filler bits.
    a) Number of unused bits in memory type instruction = 1.
    b) Number of unused bits in register type instruction = 5.

Problem IV: Assembly to Machine Code Translation **5 Points**

Assume a processor with 16 registers. Each of the registers is called by the processor by its corresponding value. Ex: R0 is called 0, R1 as 1, R2 as 2 and so on.

Mnemonic	add	mul	beq	sub
Binary	0001	0010	0100	1000

The instructions are represented as <opcode><destination register><source register 1><source register 2>.

For the following program, convert the given assembly code to machine code.

Program	add R7, R8, R10
	sub R15, R13, R2
	mul R5, R11, R14
	add R1, R3, R7
	mul R7, R8, R9

Solution IV

Note: If any student maps the registers to different register addresses other than the one specified in question, then still evaluate the question with a penalty of 1-point.

Here we have 16 general purpose registers. So, minimum 4-bits will be required to uniquely encode them.

Register Address Mapping

Register_Number	R0	R1	R2	R3	R4	R5	R6	R7
Address_1 (random mapping evaluate with penalty)	$(0100)_2$	$(0101)_2$	$(0110)_2$	$(0111)_2$	$(1000)_2$	$(0000)_2$	$(0001)_2$	$(0010)_2$
Address_2 (Correct Mapping)	$(0000)_2$	$(0001)_2$	$(0010)_2$	$(0011)_2$	$(0100)_2$	$(0101)_2$	$(0110)_2$	$(0111)_2$

Register_Number	R8	R9	R10	R11	R12	R13	R14	R15
Address_1 (random mapping evaluate with penalty)	$(0011)_2$	$(1001)_2$	$(1010)_2$	$(1011)_2$	$(1100)_2$	$(1101)_2$	$(1110)_2$	$(1111)_2$
Address_2 (Correct Mapping)	$(1000)_2$	$(1001)_2$	$(1010)_2$	$(1011)_2$	$(1100)_2$	$(1101)_2$	$(1110)_2$	$(1111)_2$

Machine code translation of the program

Instruction	<opcode>	<destination register>	<source register 1>	<source register 2>
add R7, R8, R10	0001	$(0111)_2$	$(1000)_2$	$(1010)_2$
sub R15, R13, R2	1000	$(1111)_2$	$(1101)_2$	$(0010)_2$
mul R5, R11, R14	0010	$(0101)_2$	$(1011)_2$	$(1110)_2$
add R1, R3, R7	0001	$(0001)_2$	$(0011)_2$	$(0111)_2$
mul R7, R8, R9	0010	$(0111)_2$	$(1000)_2$	$(1001)_2$

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