Tutorial 2
CSE 112 Computer Organization

Q1 What is ISA? Explain RISC and CISC ISA.

ISA can be defined as an abstraction over CPU through which a programmer can interact and program the CPU.

(*Refer to tut 1 solution to see more about ISAs*)

RISC	**CISC**
Known as Reduced Instruction Set Computer	Known as Complex Instruction Set Computer
One cycle per instruction	Can use multiple cycle per Instruction
Same Length Instruction	Variable Length Instruction
Less number of Instruction in ISA	Higher number of Instruction in ISA
Higher Size programs	Lower Size programs
Simpler Hardware Implementation	Complex Hardware implementation

NOTE to Section A TAs : This topic is covered in lecture class, so just revise it briefly. No need to spend much time on it.

NOTE to Section B TAs : ISA has not been yet introduced to section B students in class, but in the first tut, this term was introduced to them. So explain it just once again, and clear questions related to it if there are any.

Q2.

1. 4 bits as given in table
2. Since at each location we are storing 2 bytes and maximum memory supported by ISA is 4kb therefore it would require 11 bits.
3. 3 bits as given in the table
4. 3(opcode) + 11(memory) + 4(registers) = 18 bits
5. 3(opcode) + 4*2(4 bits for each register) = 11 bits
6. Yes, Currently we have used one hot encoding for uniquely identifying a register.
   We can use simple binary based representation.
   00 - register A
   01 - register B
   10 - register C
   11 - register D
   Therefore, memory type instructions will have (3 + 11 + 2) 16 bits and register type will have (3 + 2 + 2) 7 bits.
7. Memory - 16 bits and register type - 7 bits. Therefore, memory require more bits for representation
8. Since register type instructions require short length for representation, therefore new syntax for reg type instructions to make it of equal length to mem type instructions, we will add 16 - 7 = 9 bits for unused bits in reg type instructions just after the opcode of the instruction. We will add 0 unused bits for mem type instructions.
9. New syntax for reg type instructions:
   Opcode - 3 bits
   Unused bits - 9 bits
   Reg a - 2 bits
   Reg b - 2 bits
   New syntax for mem type instructions
   Opcode - 3 bits
   Unused bits - 0 bits
   Addr - 11 bits
   Reg - 2 bits
10. Since program counter points to memory to fetch the next instruction, therefore its size should be 11 bits, the same as the number of bits required to represent a memory address uniquely.

NOTE to TAs : The intention of this question is to give a basic understanding of what all comprises in the ISA like arithmetic instructions, branch instruction etc. How it is used to write programs in assembly language. Also explain the basic terminology like opcode, memory operand, registers, one hot encoding etc. Also, briefly explain the memory and reg type instructions as mentioned in question.

Q3 a.

```
temp0 has value 0
temp1 has value 1

A = *X;
B = *Y;
D= *temp0;
if(A - B = 0)
{
    C = *temp1;
}
else
{
     C = *temp0;
}
*Z = C
-------------------------
Ld X A
Ld Y B
Ld temp0 D
Sub A B
Breq Equal A
Ld temp0 C
Breq Exit D
Equal :    -----> label name
     Ld temp1 C
Exit:      ------> label name
      St Z C
```

Size of the code is 9 instructions. Therefore, it is stored in the address range [0, 8]. We need location 9, 10, 11 for the location of X, Y, Z.
We also need two locations for constants temp0 and temp1 to store values 0 and 1 respectively. temp0 is stored at 12 and temp1 is stored at 13.

Note to TAs : Solve this first question to explain to students how to write assembly code with focus on how to deal with branches using labels. Explain briefly the pointer concept to students and also give an idea about branching and how to write assembly code for branching using label.

Q3 b

```
Assume temp1 and temp0 have value 1 and 0 in the physical memory
A = *X
B = *Y
C = *temp1
D = *temp0
E = A
If (A == 0):
      *Z = 0
      Jump to end
if (B == 0):
      *Z = 0
      Jump to end
while(B != D):
      E = E + A
      B -= C
*Z = E
end
-------------------------
Ld X A
Ld Y B
Ld temp1 C
Ld temp0 D
Breq Exit_z0 A
Breq Exit_z0 B
For_loop:
      Add A A
      Sub B C
      Brne For_loop B
St Z A
Breq Exit_code D
Exit_z0:
      St Z D
Exit_code:
```

Code has 12 instructions (we need to jump somewhere at last to end the program). Therefore, the address range for code is [0-11]. The location for X and Y is 12 and 13 respectively. We also need temp1 and temp0 which store values 1 and 0 with address 14 and 15 respectively.

Q3c:

```
temp0 has data 0
temp1 has data 1

A = *X;
B = *temp0;
C = *temp1;
D = *temp1;
while (A != 0)
{
      B = B+C;
      C = C+D;
      A = A-D;
}
*Z = B;
-------------------------
Ld X A;
Ld temp0 B
Ld temp1 C
Ld temp1 D
Breq loop_exit A
loop:
      Add B C
      Add C D
      Sub A D
      Brne loop A
loop_exit:
      St Z B
```

Size of the code is 10 instructions. Therefore, it is stored on the address range [0, 9].
We need locations 10, 11 and 12 for locations X, Y and Z. We also need two locations for constants temp0 and temp1 which store the values 0 and 1 respectively. temp0 and temp1 are stored at locations 13 and 14 respectively.

NOTE to TAs : The objective of this question is to familiarize students how to write code in assembly language given the ISA.

Q4

```
0x0000: Ld X A;             000 00000001010 00    (0x0028)
0x0001: Ld temp0 B          000 00000001101 01    (0x0035)
0x0002: Ld temp1 C          000 00000001110 10    (0x003A)
0x0003: Ld temp1 D          000 00000001110 11    (0x003B)
0x0004: Breq 0x0009 A       100 00000001001 00    (0x8024)
0x0005: Add B C             010 000000000 01 10   (0x4006)
0x0006: Add C D             010 000000000 10 11   (0x400B)
0x0007: Sub A D             011 000000000 00 11   (0x6003)
0x0008: Brne 0x0005 A       101 00000000101 00    (0xA014)
0x0009: St Z B              001 00000001100 01    (0x2031)

At the following memory addresses, store the given data:
0x000D:    0x0000
0x000E:    0x0001
```

NOTE to TAs : The objective of this question is to familiarize students with how code is stored in machines.

Q5.

X is at 0xA
Y is at 0xB
Z is at 0xC
temp0 is at 0xD
temp1 is at 0xE
Initial PC is 0.

Cycle	Instruction	A	B	C	D	PC	Memory Access 1 (Data)	Memory Access 2 (Due to PC) (Instruction)	Register Access 1	Register Access 2
1	`Ld X A`	2	-	-	-	1	0xA, 2	0, 0x0028	A, 2	-
2	`Ld temp0 B`	2	0	-	-	2	0xD, 0	1, 0x0035	B, 0	-
3	`Ld temp1 C`	2	0	1	-	3	0xE, 1	2, `0x003A`	C, 1	-
4	`Ld temp1 D`	2	0	1	1	4	0xE, 1	3, `0x003B`	D, 1	-
5	`Breq 0x0009 A`	2	0	1	1	5	-	4, `0x8024`	A, 2	-
6	`Add B C`	2	1	1	1	6	-	5, `0x4006`	B, 0	C, 1
7	`Add C D`	2	1	2	1	7	-	6, `0x400B`	C, 1	D, 1
8	`Sub A D`	1	1	2	1	8	-	7, `0x6003`	A, 2	D, 1
9	`Brne 0x0005 A`	1	1	2	1	5	-	8, `0xA014`	A, 1	-
10	`Add B C`	1	3	2	1	6	-	5, `0x4006`	B, 1	C, 2
11	`Add C D`	1	3	3	1	7	-	6, `0x400B`	C, 2	D, 1
12	`Sub A D`	0	3	3	1	8	-	7, `0x6003`	A, 1	D, 1
13	`Brne 0x0005 A`	0	3	3	1	9	-	8, `0xA014`	A, 0	-
14	`St Z B`	0	3	3	1	10	0xC, 3	9, 0x2031	-	-

NOTE to TAs : The objective of this question is to familiarize students with how code is executed on simple machines. Explain briefly the steps involved in the execution cycle and for now tell them that all steps are executed in a single cycle. Since this has been briefly touched upon in section B, section A TAs make sure that you also explain to them the execution cycle briefly.