Name: Sari Abdul Ghani-Mother Al Hassanie-Nader Al Shamie

1. To code an instruction, we must use two types of instruction: R-Format (and, add, or instructions)and D-Format (li, lw, sw instructions). To start, the number of bits needed for the opcode in both types is 3 bits, since 2^3 = 8, so with 3 bits, we can represent 8 different instructions. But in our design, we have only 6 instructions which is even less than 8. Hence, 3 bits are enough to represent the opcode of our 6 instructions.

   As for the R-format, we have to take into consideration 3 register fields which are : Rm, Rn and Rd. Since we have only 8 registers (X0 through X7), we need only 3 bits to represent the number of the register since 2^3 = 8. And finally for the R-format, we need to talk about the "shamt" field which is about shifting. Since for now, the number of bits needed to code the R-format instruction is: 3 + 3*3 = 12 bits, and we want to make it 16 bits so that we can later use the PC without any problems, (we must make it a power of 2, and 16 is the closet to 12). Thus, we make the shamt felid 4 bits.

   As for the D-format, we have 2 register fields : Rn and Rt, which gives us a total of 6 bits for the register fields (2*3 = 6 bits). As for the op2, we keep it 2 bits. Finally, for the number of bits for the address (immediate value in li instruction, and offset in lw and sw), the range is [-16,15]. Thus, we need 5 bits to represent it, since using 5 bits , we get -2^(5-1) to +2^(n-1)-1 which gives us what we need : -16 till +15. Hence, we get the size of the instruction = 3 (opcode) + 6 (2 registers) + 2 (op2) + 5 (address) = 16 bits. (which is perfect and does not need modification since it is a power of 2).

2. No, it is not mandatory to have the same number of bits for a memory word address and a register. However, we could have them both equal or have the size of the register greater than the memory word size. For example, we can have the size of a register 16 bits and the size of a memory word 16 bits so that every memory word takes exactly one register, or have the size of a register 32 bits, then a one memory word sits in the first 16 bits of the register field. In this design, we take into consideration that the size of a memory word is 16 bits.

3. Although as we mentioned, it is not mandatory to have the size of register and memory word to be the same, we choose the size of the register to be 16 bits, as this size is sufficient to store a memory word, and other times store data and numbers (also, the range of numbers here is -16,15 , which needs only 5 bits to be stored).

4. We will represent each format using a table:

opcode	Rm	shamt	Rn	Rd

3 bits	3 bits	4 bits	3 bits	3 bits

R-format:

D-format:

Opcode	address	Op2	Rn	Rt
3 bits	5 bits	2 bits	3 bits	3 bits

5. **add**: It is R-format instruction which performs addition of Rm and Rn and places the result in Rd.
   Syntax : add
   Opcode: 000

   and: It is R-format instruction which performs AND operation on the bits of Rm and Rn fields and places the result in Rd.
   Syntax : and
   Opcode: 001

   or: It is R-format instruction which performs ORR operation on the bits of Rm and Rn fields and places the result in Rd.
   Syntax : or
   Opcode: 010

   li : It is D-format instruction which loads immediate a constant value into the Rt register.
   Syntax : li
   Opcode: 011

   lw: It is D-format instruction which loads from the memory a value into the Rt register.
   Syntax : lw
   Opcode: 100

   sw: It is D-format instruction which stores the value found in Rt into calculated memory address,
   Syntax : sw
   Opcode: 101

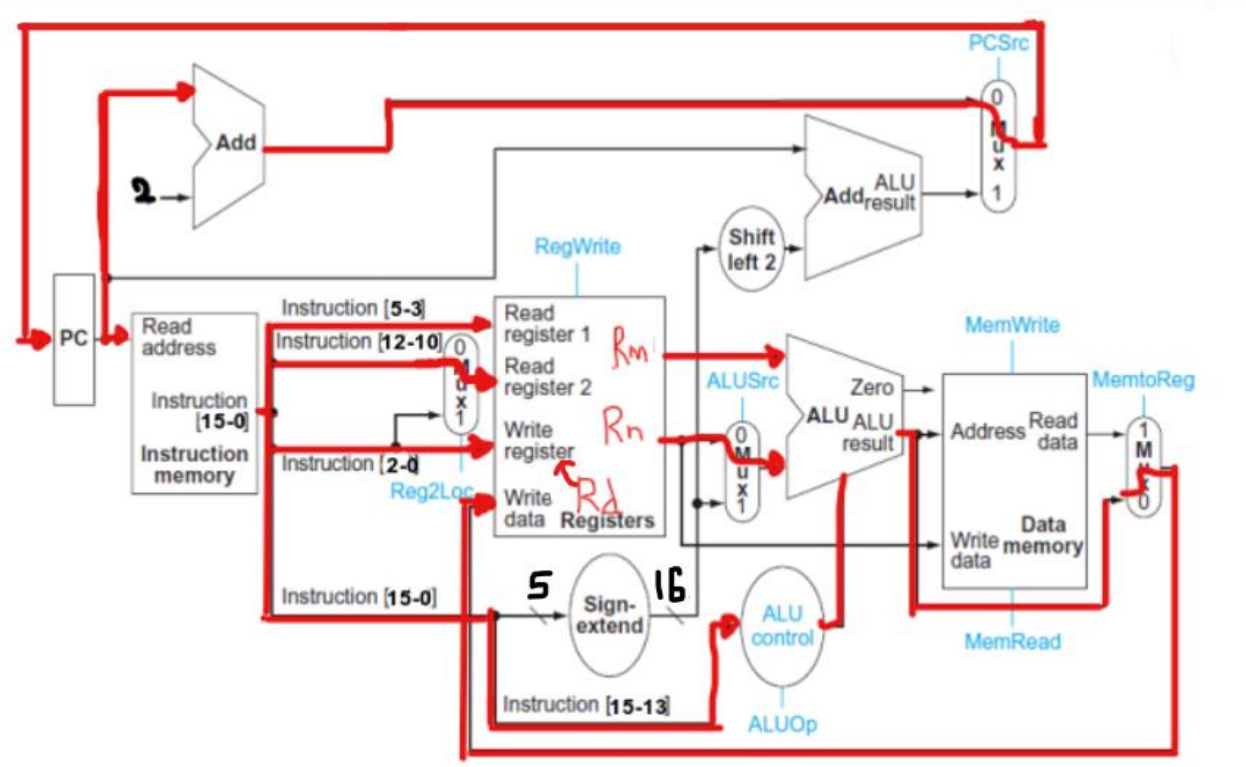
6. Unique identifiers are:

Register	Identifier/corresponding number
X0	$X0 → 000
X1	$X1 → 001
X2	$X2 → 010
X3	$X3 → 011

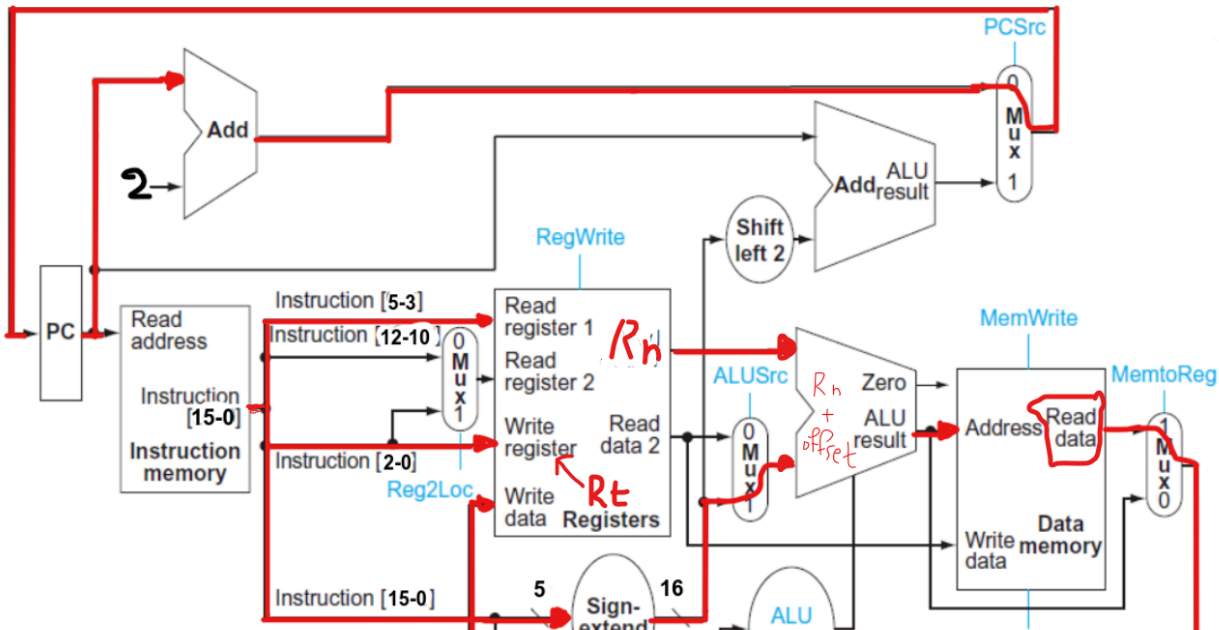
X4	$X4 → 100
X5	$X5 → 101
X6	$X6 → 110
X7	$X7 → 111

7. The increment performed by the PC counter is based on the size in bytes of an instruction. Since the size of each instruction in our design is 16 bits (which are equal to 2 bytes) ,and there are no branch instructions done, we have the PC incremented by 2 after each instruction starts executing (done in the IF stage at the start of an instruction) . Thus PC = PC + 2 .

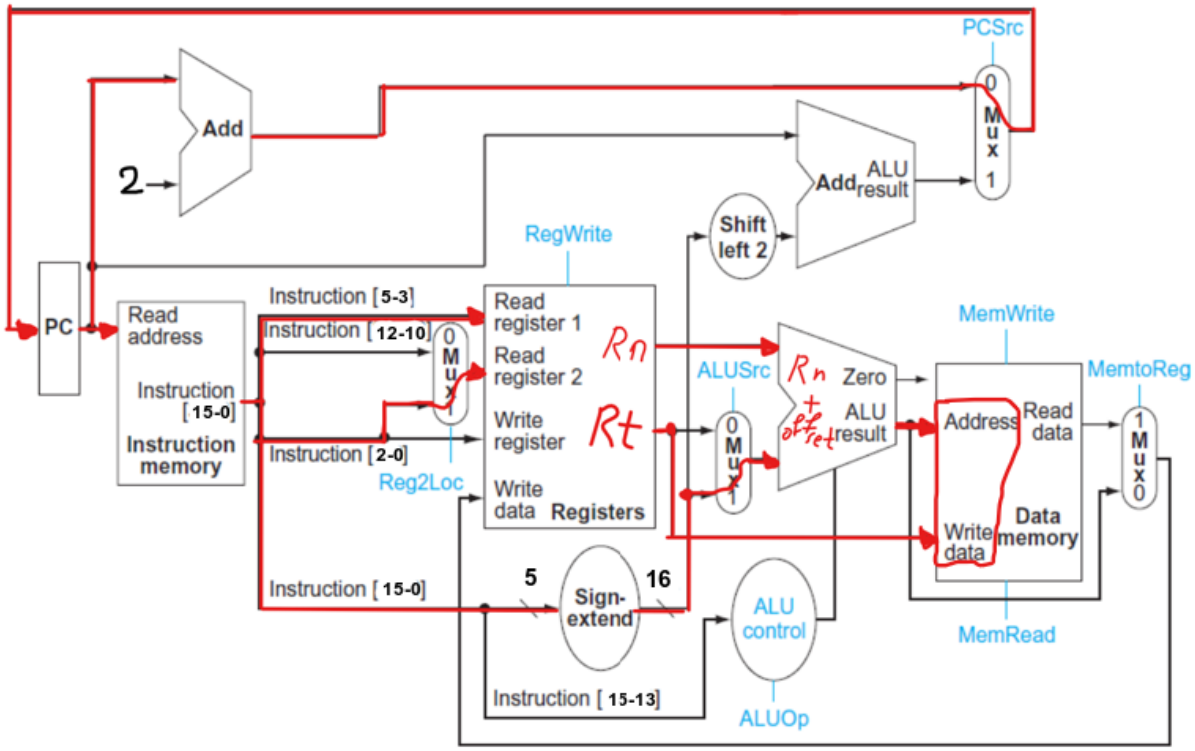
8. **Add/and/or** instructions (All of R-format instructions have the same data path as following) :



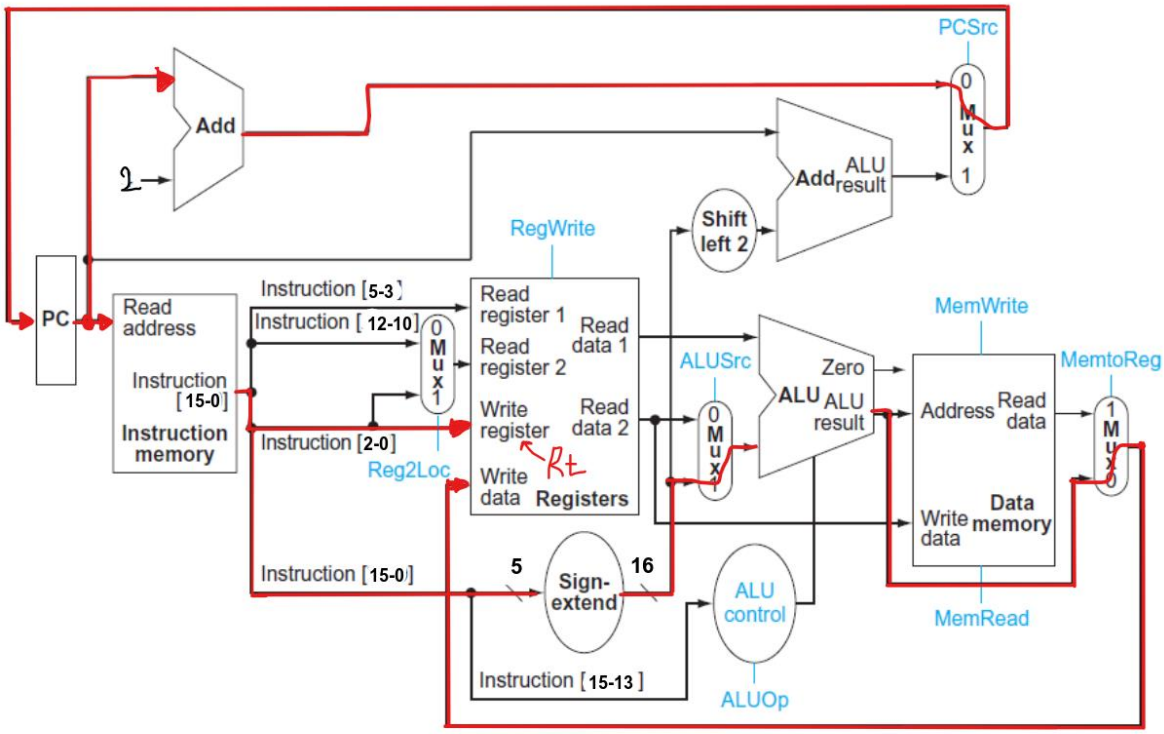
Lw :



Sw:


PCSrc
Add
2
M u x
0
1
ALU Add result
Shift left 2
RegWrite
PC
Read address
Instruction [15-0]
Instruction memory
Instruction [5-3]
Instruction [12-10]
Instruction [2-0]
Reg2Loc
Read register 1
Read register 2
Write register
Write data
Registers
Rn
Rt
ALUSrc
Rn + offset
Zero
ALU result
MemWrite
MemtoReg
Address
Read data
Write data
Data memory
MemRead
Instruction [15-0]
5
Sign-extend
16
ALU control
ALUOp
Instruction [15-13]

Li:


PCSrc
Add
2
M u x
0
1
Add
ALU result
Shift left 2
RegWrite
PC
Read address
Instruction [15-0]
Instruction memory
Instruction [5-3]
Instruction [12-10]
Instruction [2-0]
Reg2Loc
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Rt
ALUSrc
Zero
ALU
ALU result
MemWrite
MemtoReg
Address
Read data
Write data
Data memory
MemRead
Instruction [15-0]
5
Sign-extend
16
ALU control
Instruction [15-13]
ALUOp