DATAPATH AND ITS UNDERSTANDING – MIPS – COMPUTER ARCHITECTURE – SESSION – 10

Shriram K Vasudevan

Agenda

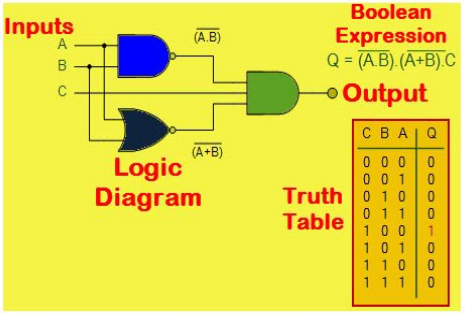
- What a data path is all about?
- Major elements in the data path.
- Building a simple datapath for instructions.
- Branching and related hardware requirements for datapath.
- Building a comprehensive datapath.
- Tracing the datapath for instructions from the comprehensive datapath

Remember this – a must

S.No.	Combinational Circuit	Sequential Circuit
1.	It contains no memory elements	It contains memory elements
2.	The present value of it's outputs are determined solely by the present values of it's inputs	The present value of it's outputs are determined by the present value of it's inputs and it's past state
3.	It's behavior is described by the set of output functions	It's behavior is described by the set of next-state(memory) functions and the set of output functions

Combinational Logic Circuits

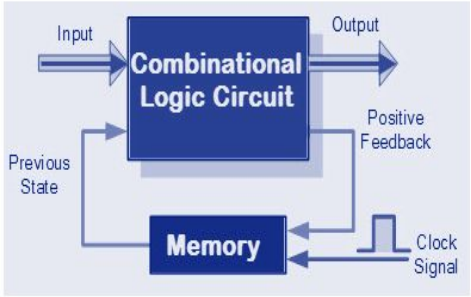
The simple time independent logic circuits that are implemented using Boolean circuits whose output logic value depends only on the input logic values can be called as combinational logic circuits.



Combinational Logic Circuit

Sequential Logic Circuits

The simple logic circuits whose output logic value depends on the input logic values and also on the stored information is called as sequential logic circuits.



Sequential Logic Circuit

Contd.,

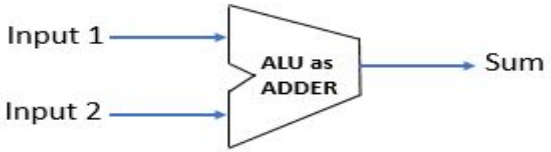
- A state element has at least two inputs and one output.
- The required inputs are the **data value to be written into the element and the clock, which determines when the data value is written.**
- The output from a state element provides the value that was written in an earlier clock cycle
- Example for a state element? – D – Flip-flop. Two inputs (a value and a clock) with exactly one output.
- **In addition to flip-flops, our MIPS implementation also uses two other types of state elements: memories and registers, both of which appear in architecture.**
- **Note: The clock is used to determine when the state element should be written; a state element can be read at any time.**

Introduction

- Let's learn the basic terminologies, first!
  - A data path element is a unit (hardware component) which shall be helpful in holding the data.
  - The data could be the one which has to be processed or could have been already processed.
  - Either case, the data requires a place to stay and that is what we mean as data path element.
  - Some of the data path elements could be memory elements to hold the instruction or data, ALU, registers, buses, adders, multiplexors etc. (I ignore control signals for now, we shall deal it later)

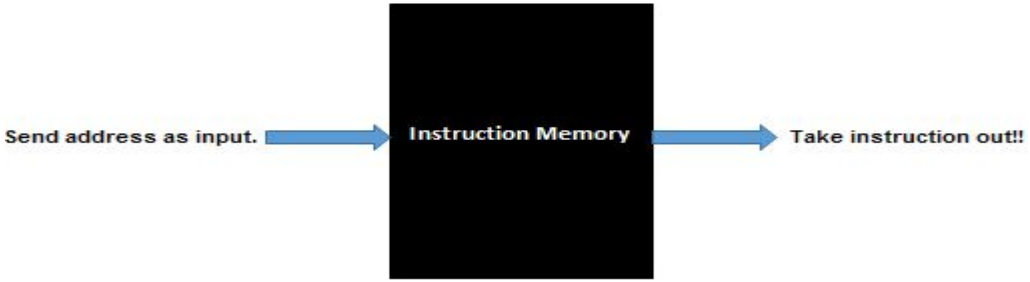
Datapath elements

Program Counter



Adder

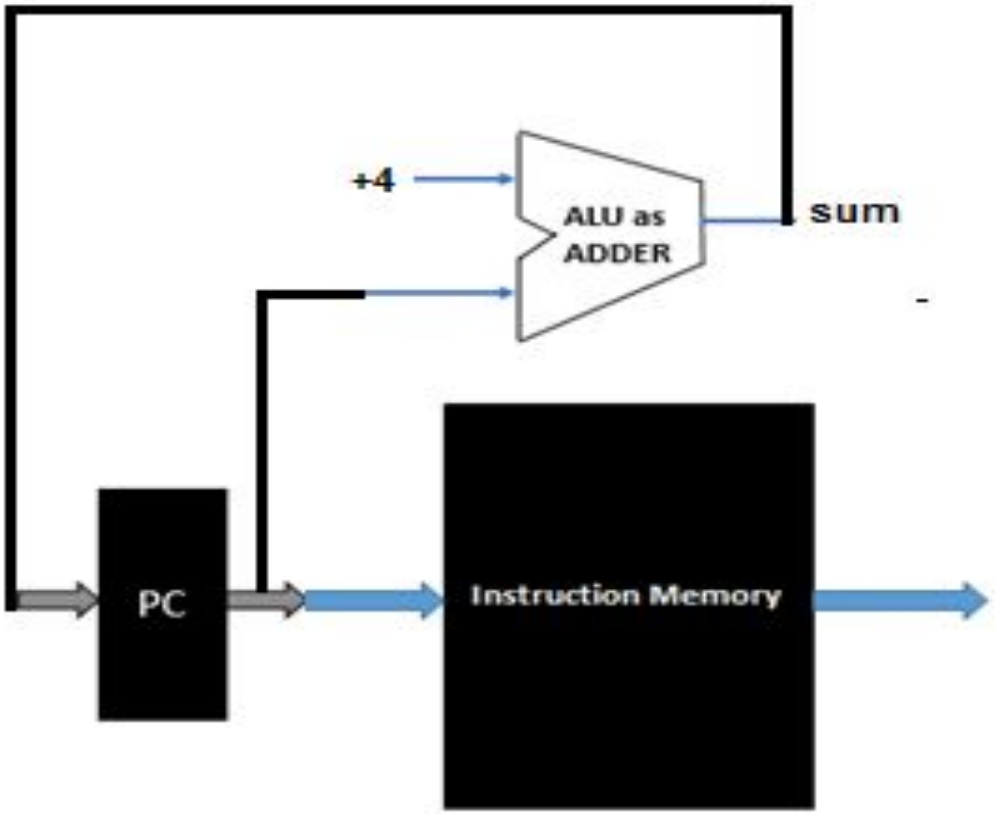
Instruction Memory



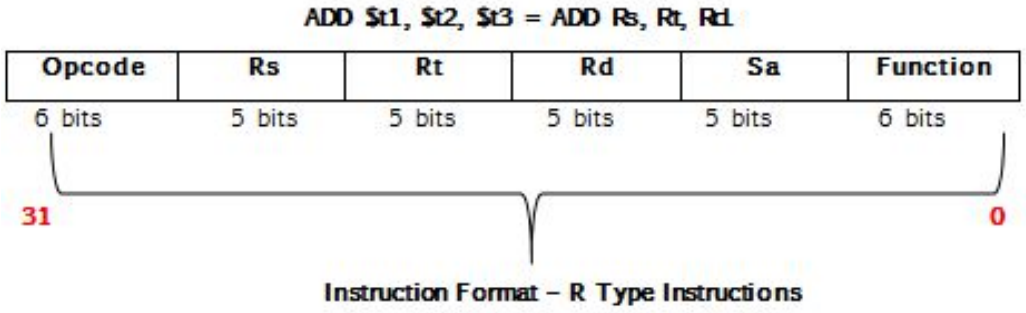
There are more components like multiplexors and control signals to be learnt; We shall introduce them as and when needed appropriately.
Now, it would be good to get a shape with all the datapath elements available in hand. Just to reiterate,

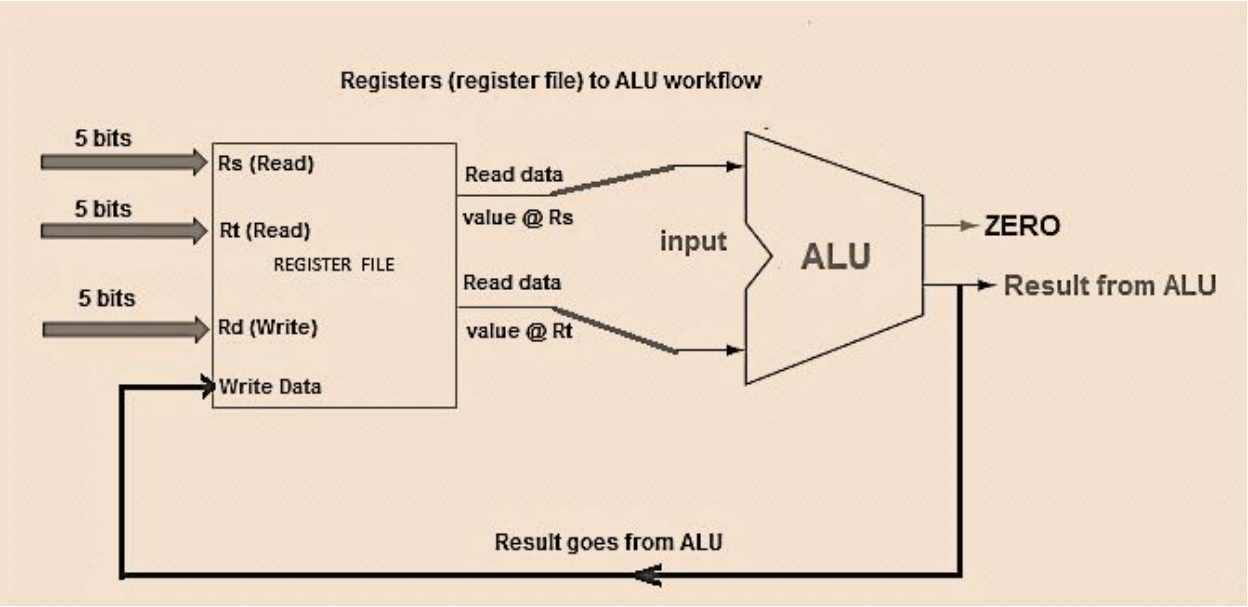
- **Program counter is used to store the address of next instruction.**
- **Adder helps in incrementing the address in the program counter so as to make it hold the address of next instruction.**
- **Instruction memory holds all the instructions and with the input from the program counter, one can identify the instructions one after another.**

Let us connect all the three..

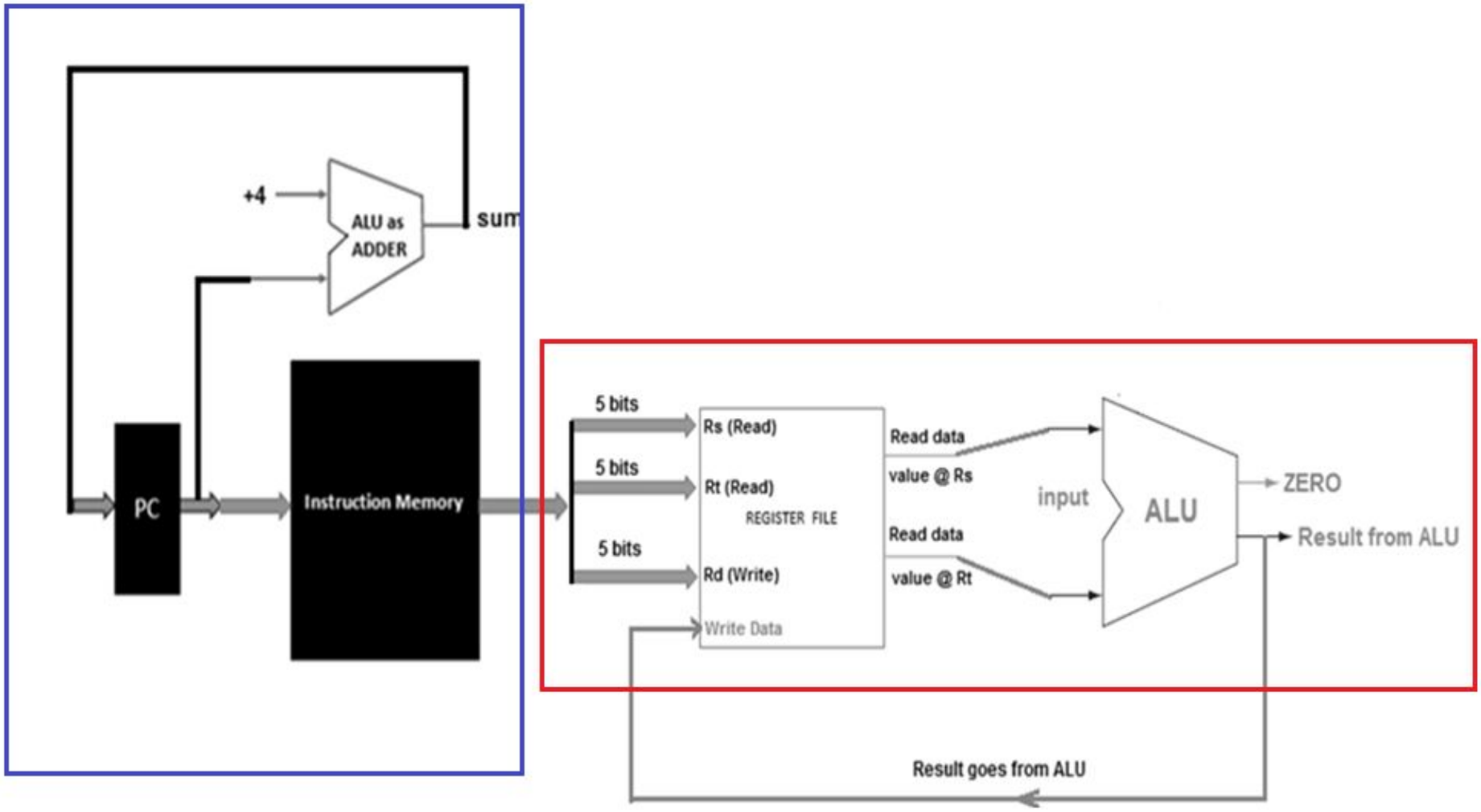


Datapath for ADD/SUB instruction (R – Type)





Contd.,



Datapath representation for ADD instruction

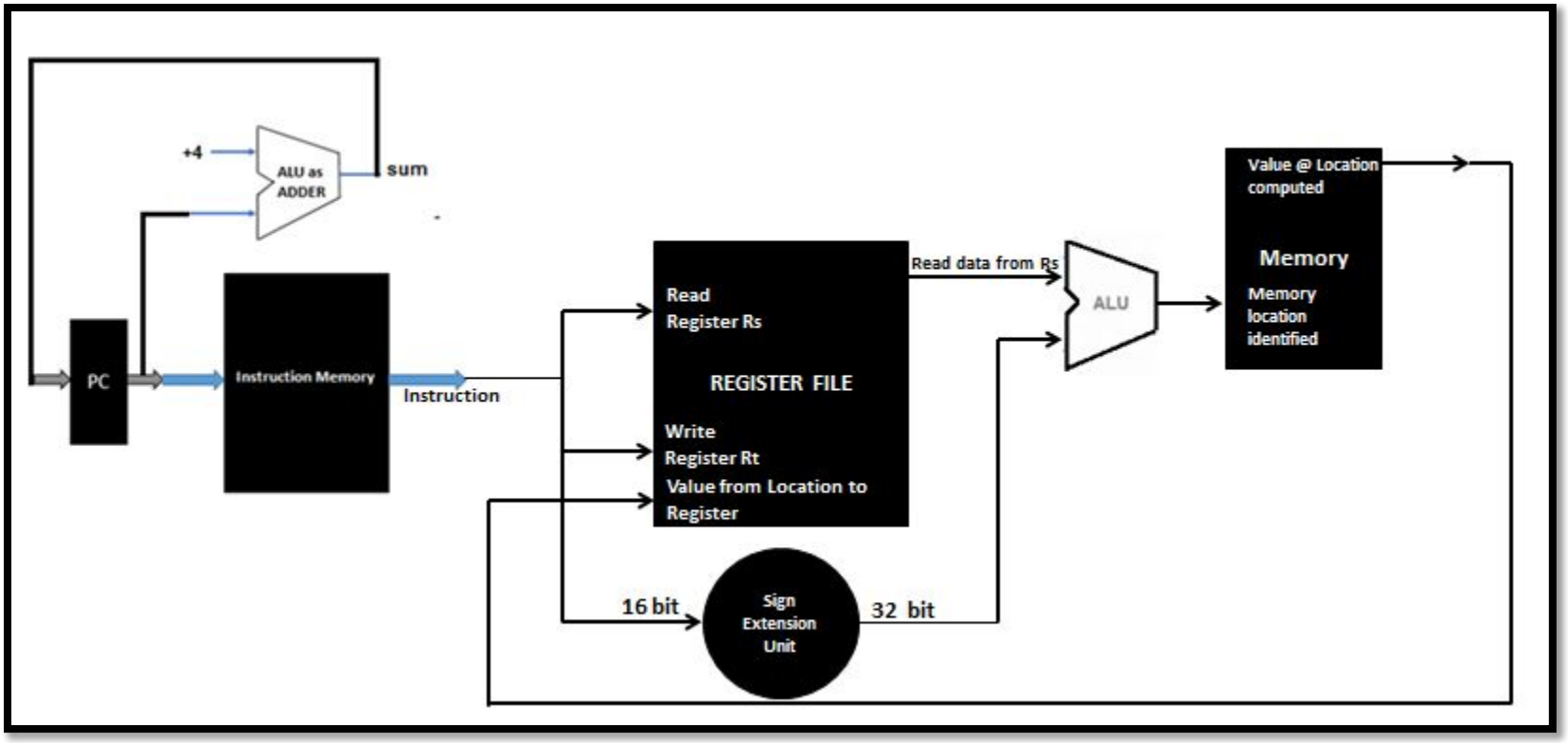
Load Word (LW)

lw $Rt, offset ($Rs)

lw $to, 15 ($t1)

- Assuming t1 has the content 100 and offset as specified is 15, lw computes the sum as 100 + 15 = 115.
- 115 is the location is which has the data to be loaded to the register t0. Content from the location from 115 shall be now copied to $to.

Contd.,



Datapath representation for Load instruction

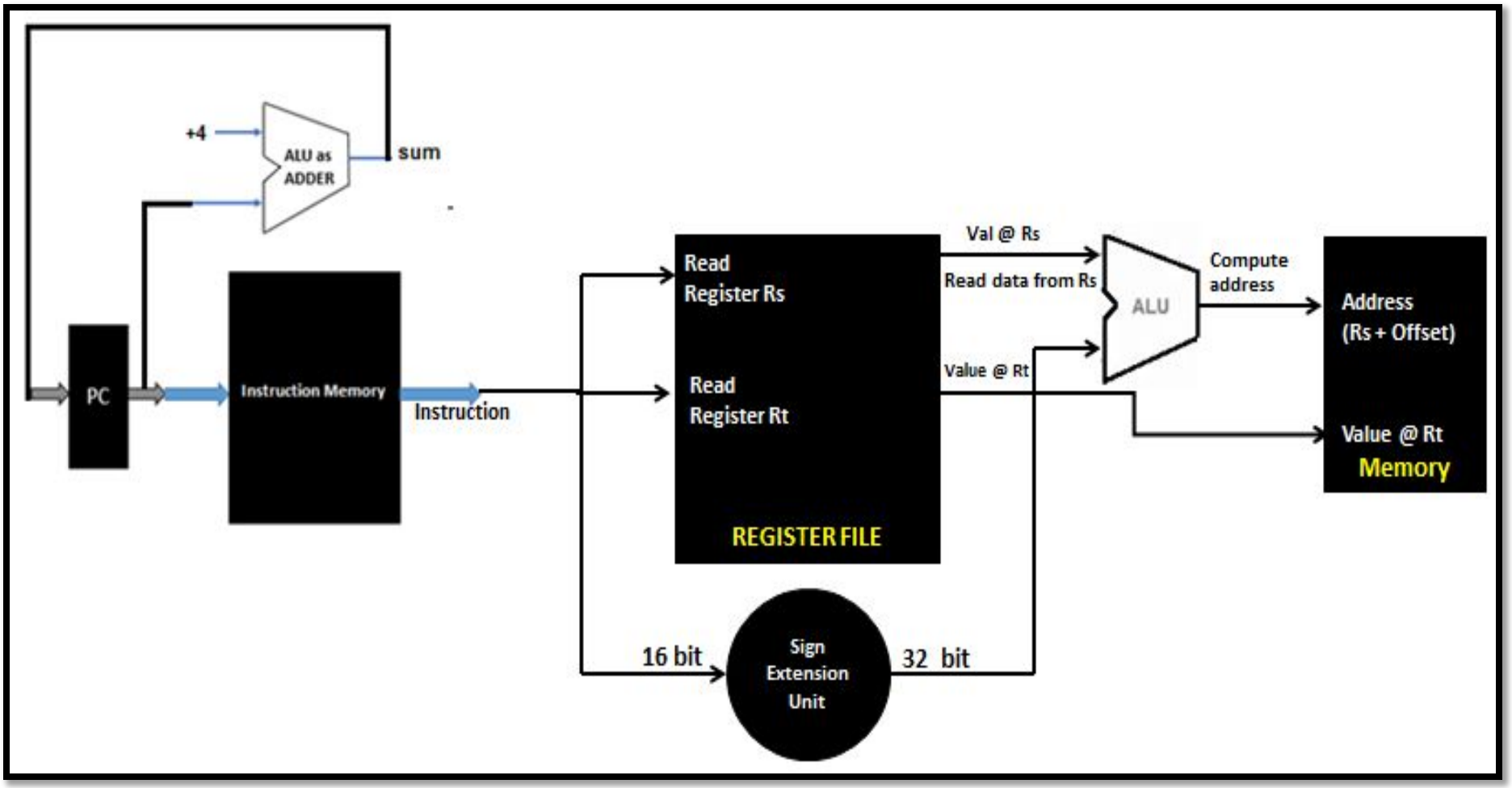
Contd.,

- Understand this!
- Offset is present in the instruction and is 16 bit wide. It has to be converted to 32 bits and sign extension unit is used for the purpose. Eg. **lw $to, 15 ($t1)**
- The register which has the value (in concurrence with the offset) is identified as Rs and the data from Rs is read.
- The Rs data and offset value together gets the result (e.g. 15 from offset and 100 from Rs = 115). 115 is the memory location to be checked for the data to be loaded into the destination register.
- From the location computed the data shall be fetched (retrieved) and loaded into the register specified in the instruction. This is represented in the datapath representation.

Datapath for Storeword (SW)

- SW is complementary for LW. Store word works this way. (**SW $Rt, Offset ($Rs))**
  - The content from a register specified should be loaded into a memory location.
  - The memory location shall be computed with the offset + register content (As in case of LW). ALU sums up and gets the location to which the content from the source register has to be stored.
  - Once memory location is identified, the value at Rt shall be stored in the location.

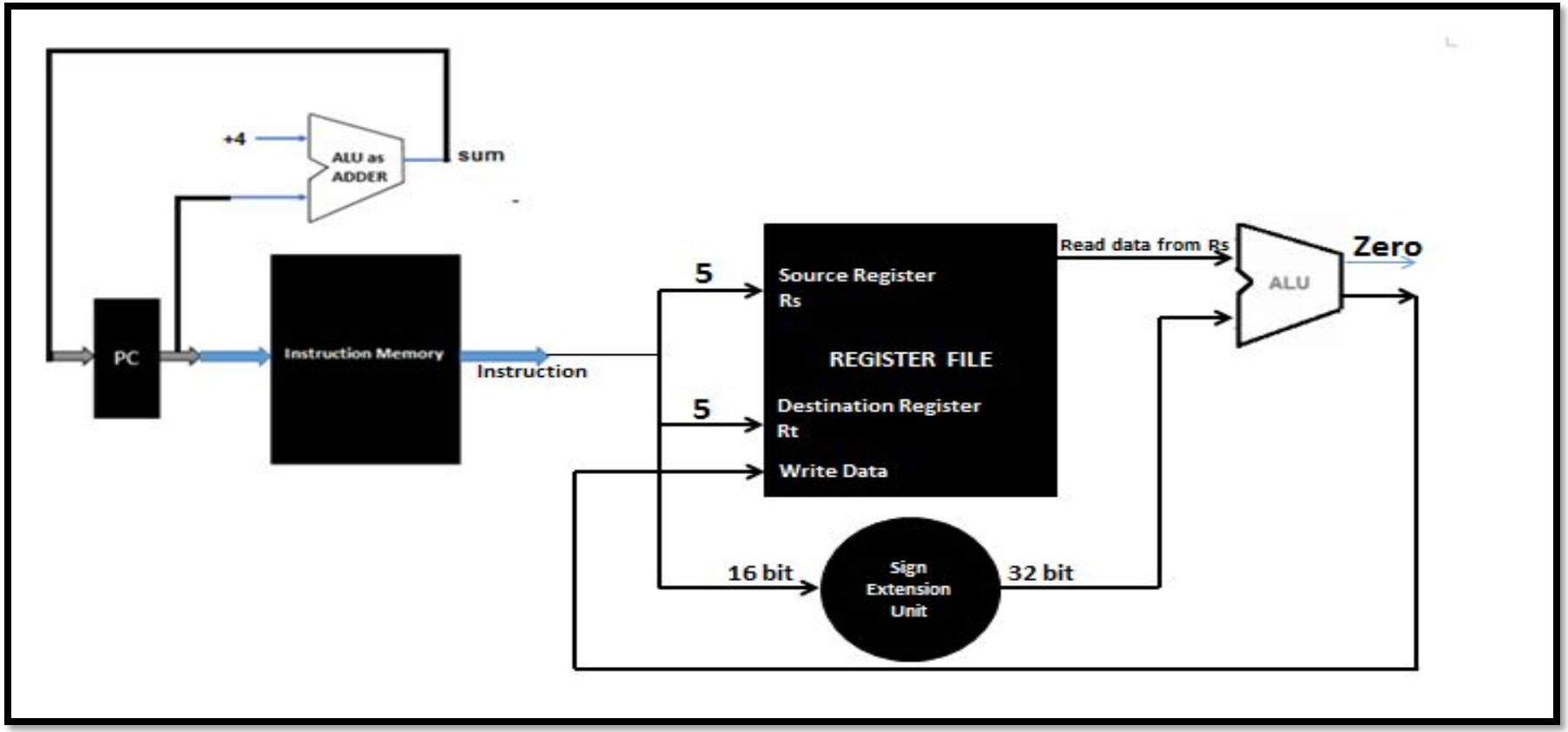
So we draw as this!



Datapath for Storeword (SW)

Datapath for Immediate Instruction

Instruction format: addi $Rs, $Rt, Offset
Sample Instruction: addi $t0, $t0, 5 ($t0 = $t0 + 5)



Lets make it complicated!

- **Single datapath – An approach to be appreciated.**
- Can’t have multiple datapaths for multiple instructions.
- Cost would be more.
- Complexity would be fairly high!
- So, a single datapath should support all the instructions!

LET US SEE THAT IN NEXT SESSION!!! ☺

THANK YOU

SHRIRAM K VASUDEVAN

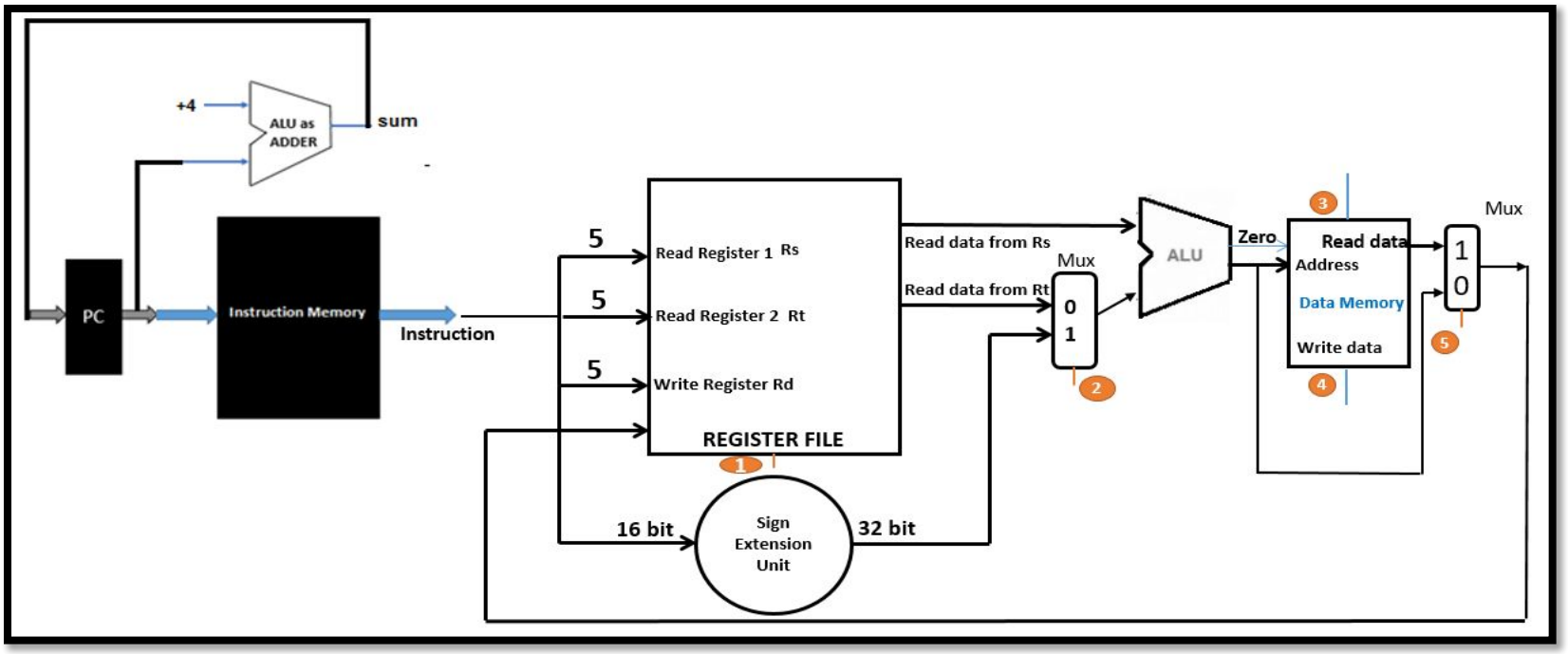
DATAPATH AND ITS UNDERSTANDING – MIPS – COMPUTER ARCHITECTURE – SESSION – 11

Shriram K Vasudevan

Single datapath – An appreciable approach

- Multiplexors needed! No other go!
- Signals – We shall explain later! But, know the names.
  - RegWrite – Register Write Signal.
  - ALUSRC – ALU source.
  - MemRead – Memory Read.
  - MemWrite – Memory Write.
  - MemtoReg – Memory to Register.

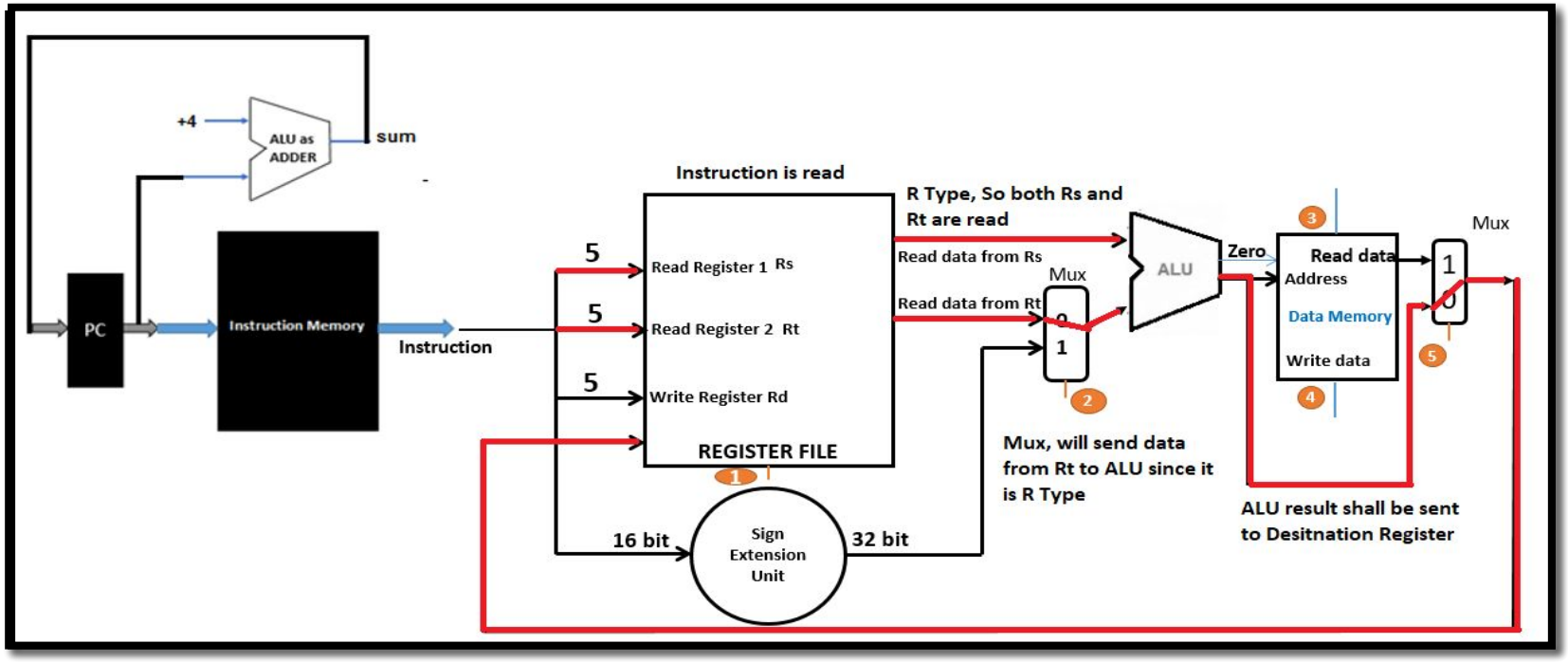
Let us see the versatile datapath!!!



1. RegWrite – Register Write Signal.
2. ALUSRC – ALU source.
3. MemRead – Memory Read.
4. MemWrite – Memory Write.
5. MemtoReg – Memory to Register.

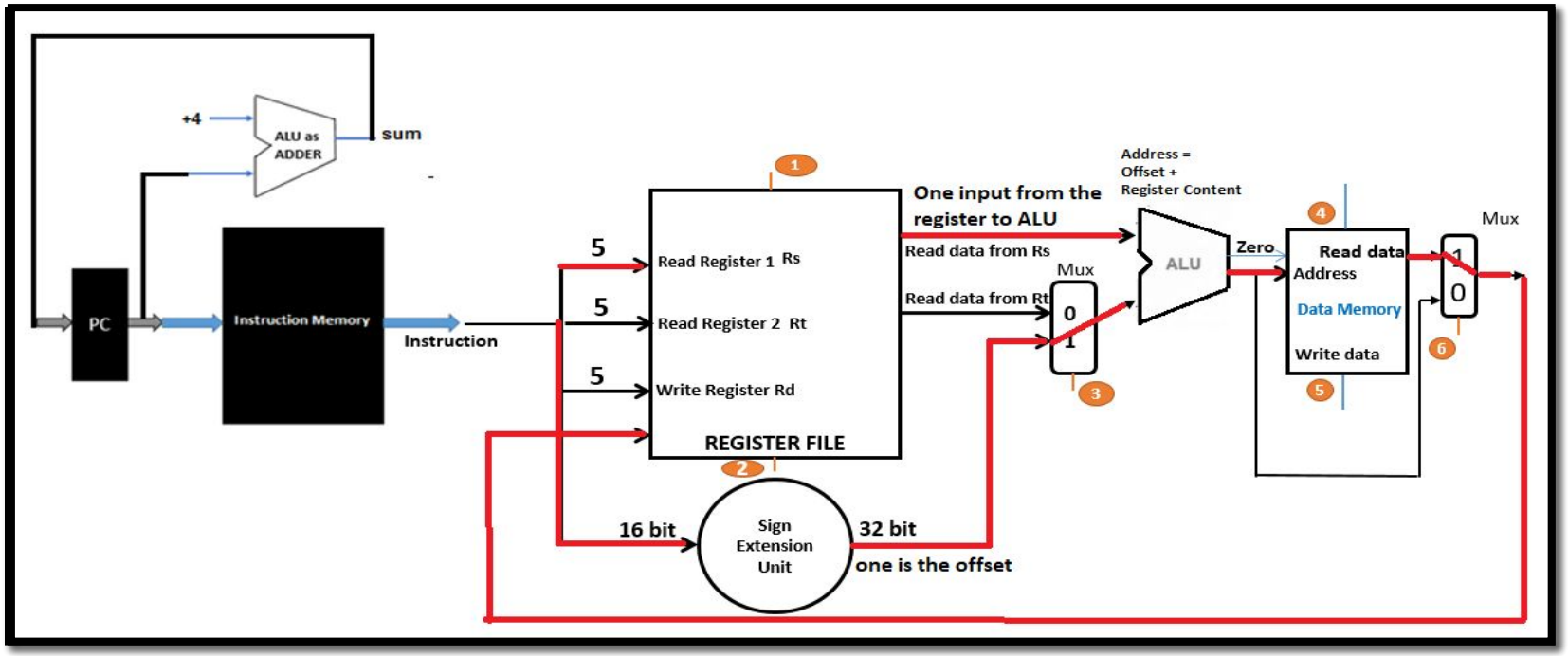
Now starts fun! Lets trace the path!

- Tracing R Type (ADD Instruction, for an instance)



R TYPE (ADD) FROM DATAPATH

Tracing LW from the versatile datapath



LW TRACE FROM DATAPATH

Data path and Branching ! Lets see in the next session, Thank You

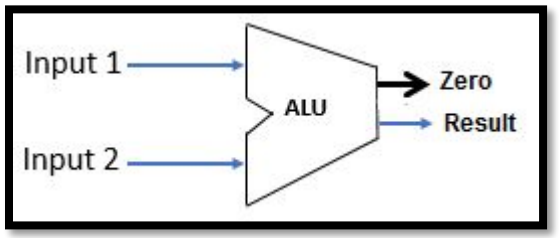
DATAPATH AND ITS UNDERSTANDING – MIPS – COMPUTER ARCHITECTURE – SESSION – 3

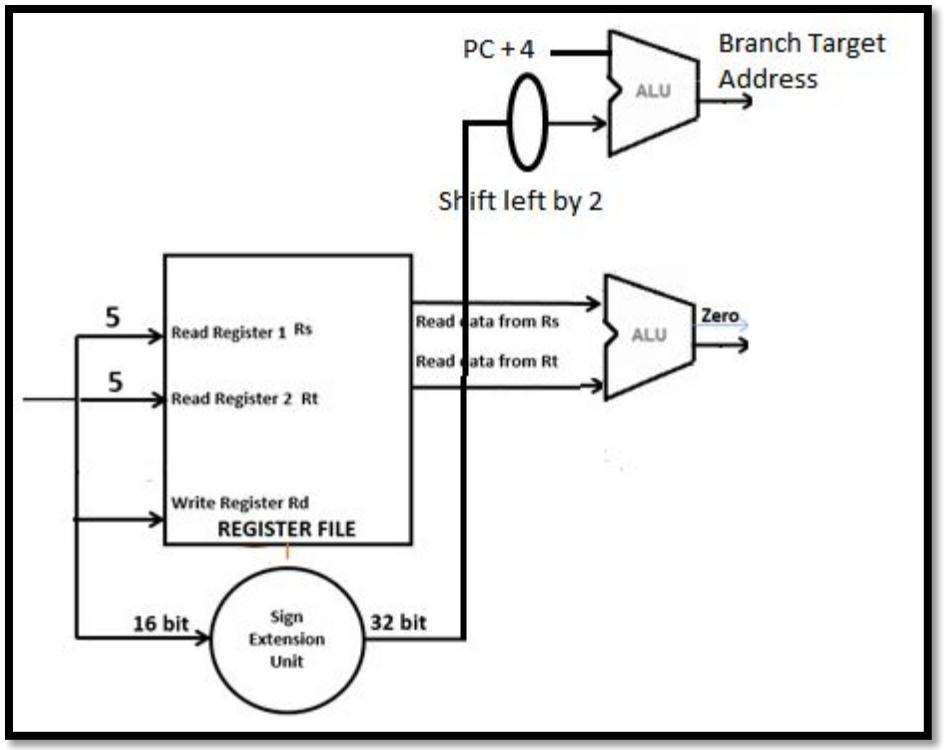
Shriram K Vasudevan

Branching! – A tough zone!

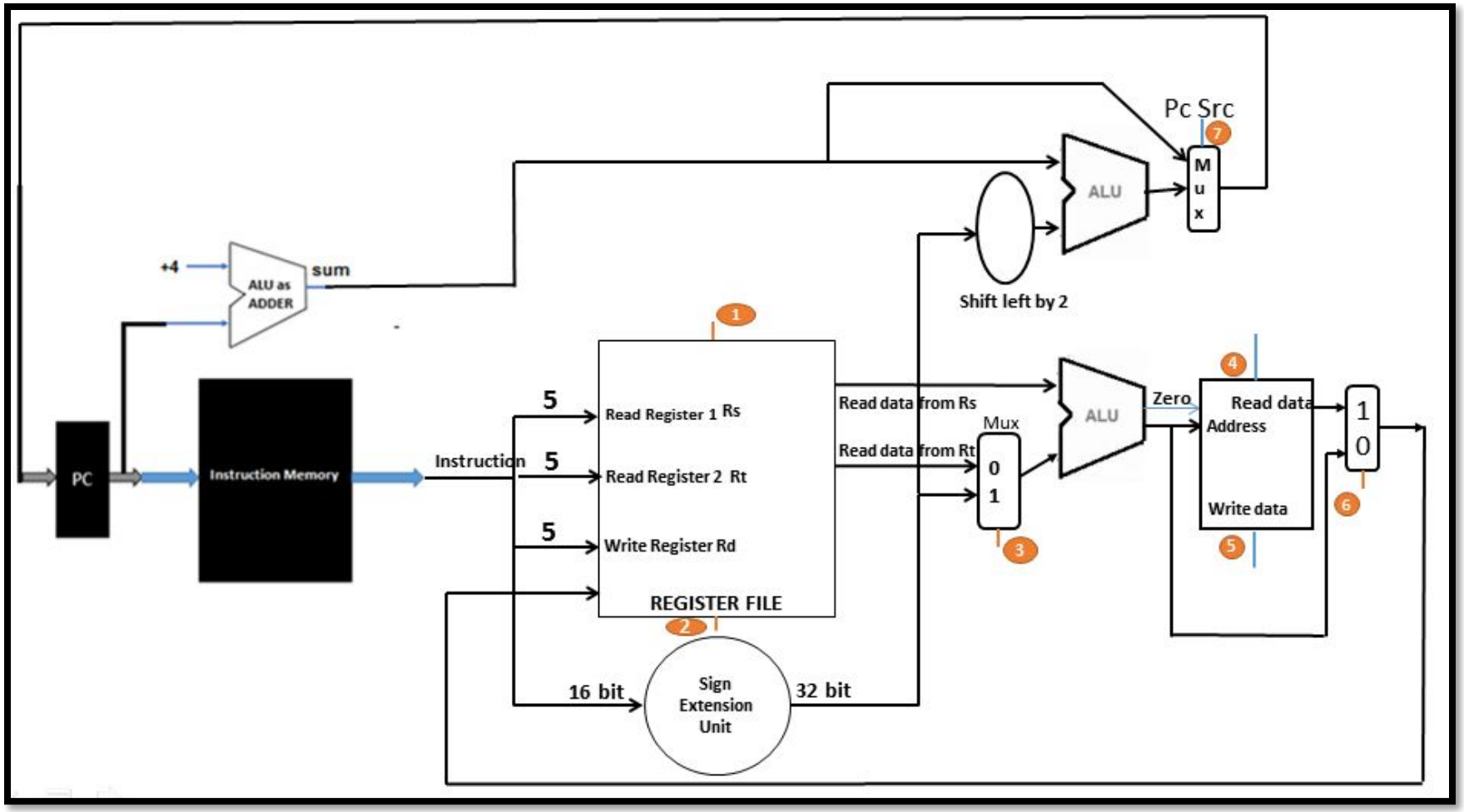
- The instruction to be analyzed is BEQ for establishing a basic understanding about branching operation.
- BEQ is expanded as branch if equal.
- So, the story is simple.
- Two registers are to be compared. There would be an offset made available as part of instruction and it shall help in computing target address to which the branch has to be taken. It also becomes important to understand if the branching would be taken or would the execution be sequential?
- So, the challenges are
  - Compute the branch target address.
  - Identify if the branching is really taken or not.

See this!

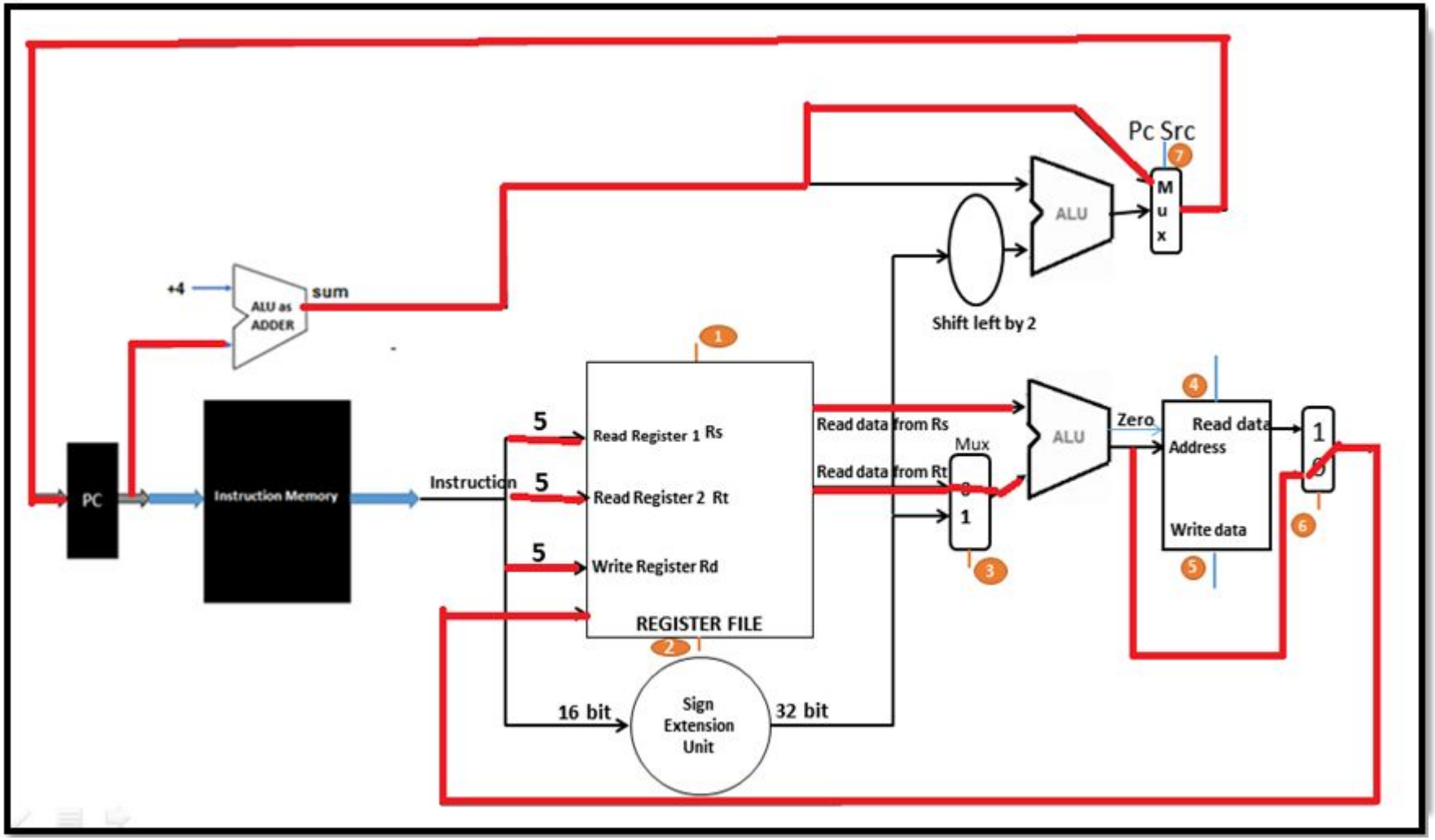

Input 1
Input 2
ALU
Zero
Result


PC + 4
ALU
Branch Target Address
Shift left by 2
5
5
Read Register 1 Rs
Read Register 2 Rt
Write Register Rd
REGISTER FILE
Read data from Rs
Read data from Rt
ALU
Zero
16 bit
Sign Extension Unit
32 bit

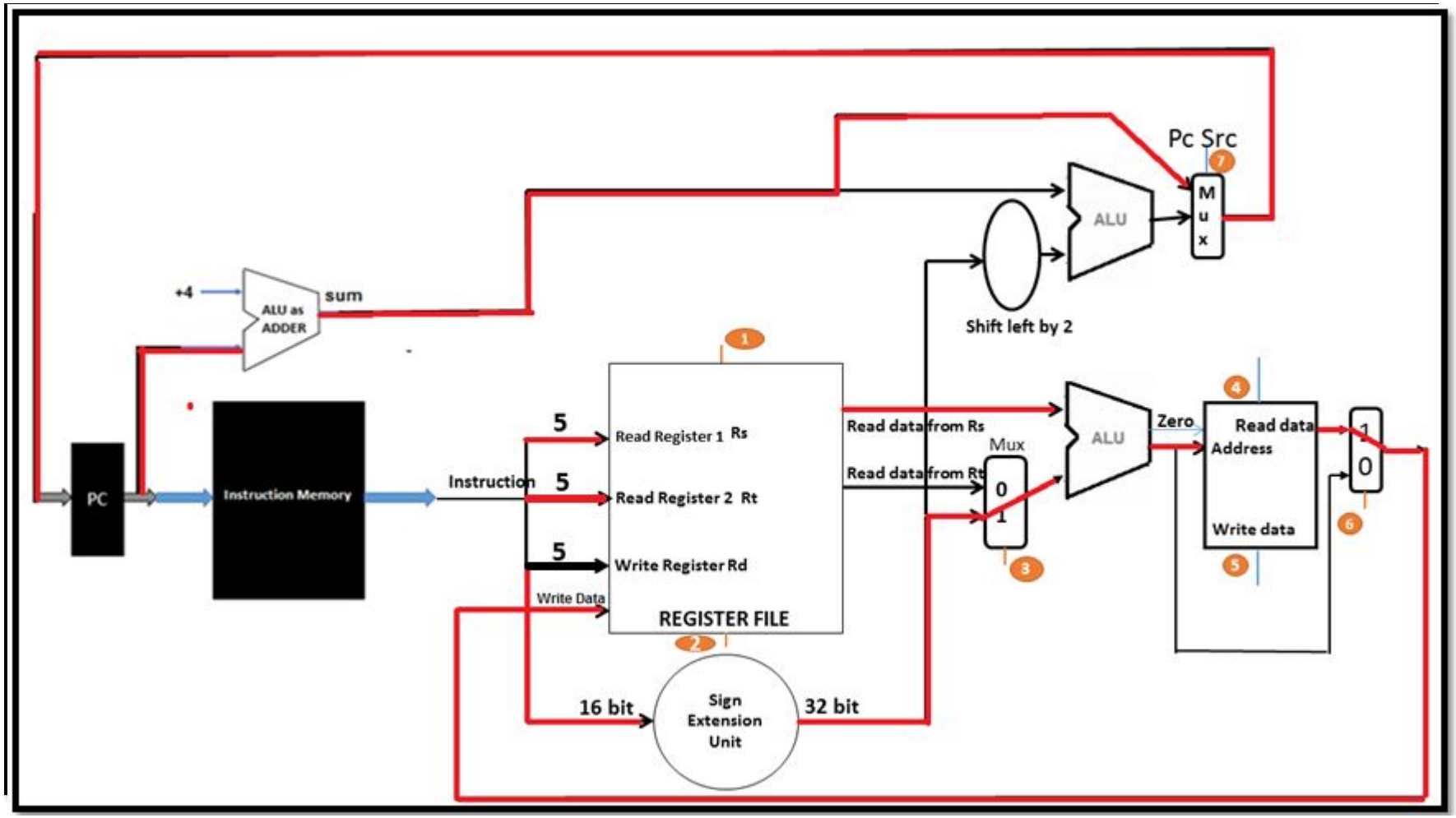
A Complex Datapath With Branching Support!!!!



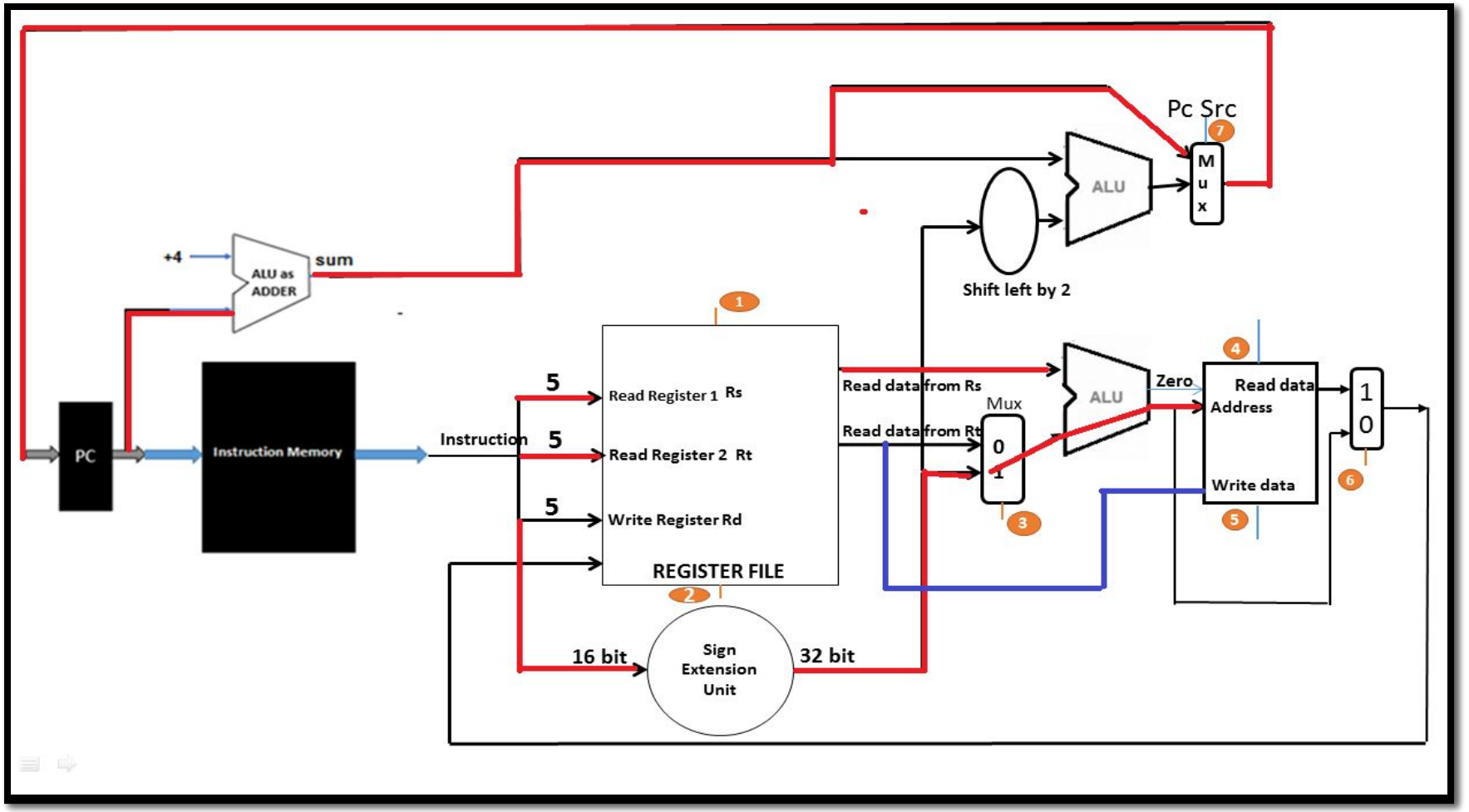
Tracing the data path for ADD instruction



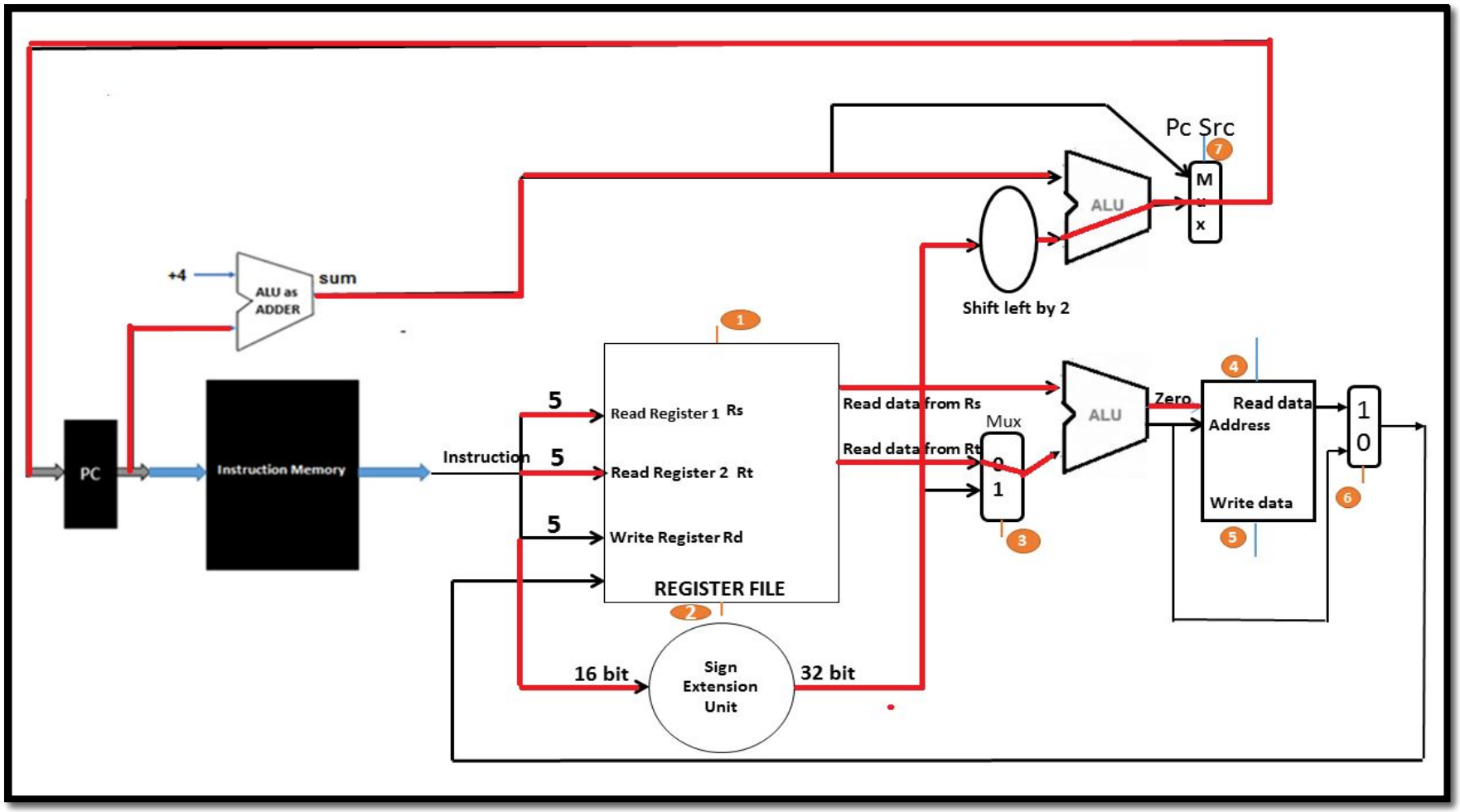
Tracing the data path for LW instruction



Tracing the data path for SW instruction



Tracing the data path for Branching instruction (**BEQ path trace**)



LET US SEE THAT IN NEXT SESSION!!! ☺

THANK YOU

SHRIRAM K VASUDEVAN