COMPUTER ORGANIZATION AND ARCHITECTURE

Shriram K Vasudevan

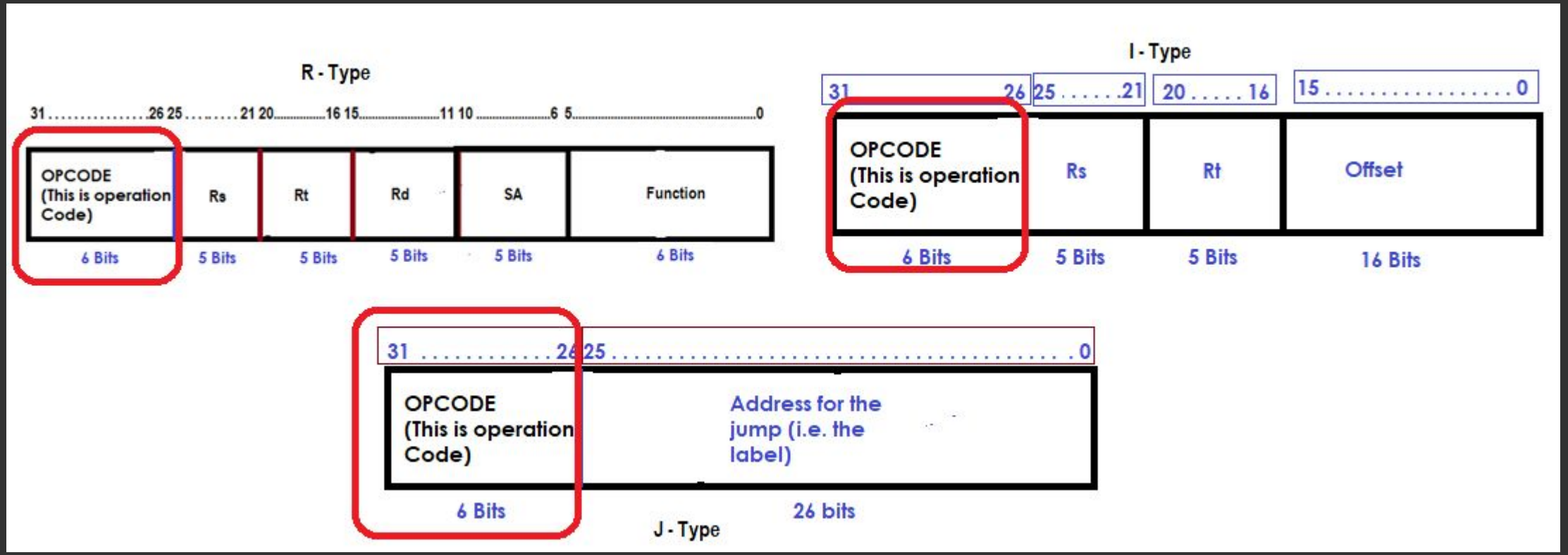
CONTROL SIGNALS

Shriram K Vasudevan

Session - 13

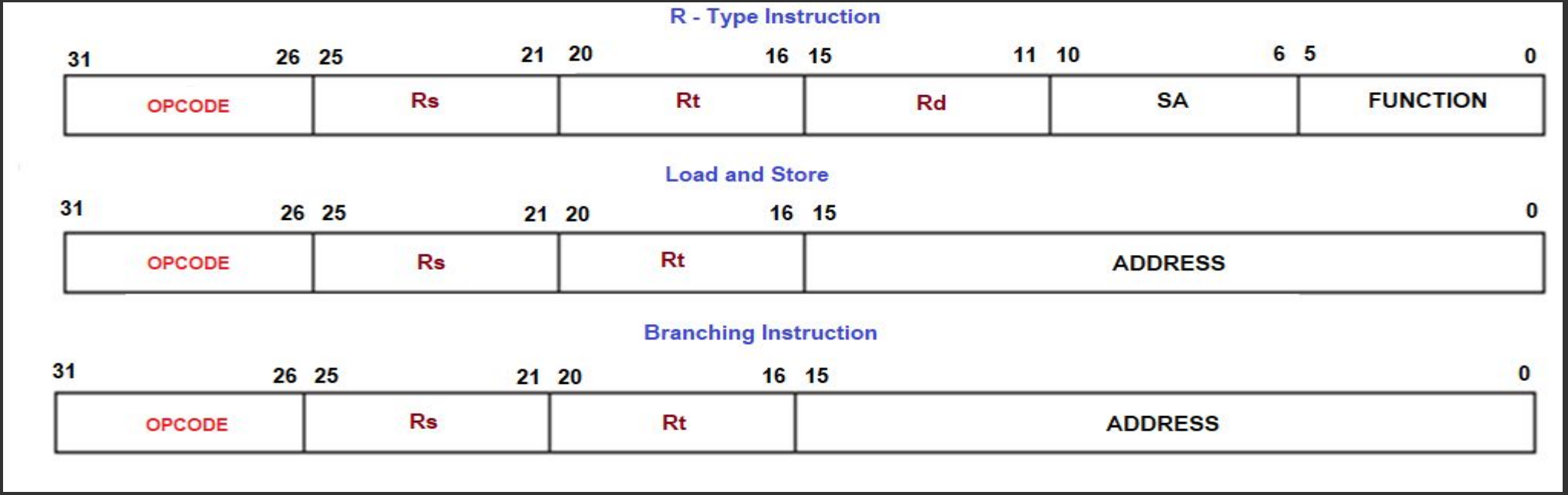
RECOLLECT THIS

- OPCODE is part of the 32 bits and it is positioned between the bits 26 – 31 (0 – 31). R Type / I Type / J Type all have the same position for the OP.



CONTD.,

- When it comes to register representation it is always Rs, Rt being the source registers for all R Type instructions, BEQ or Store instructions. The positions of Rs and Rt never changes and can be seen from the format. (refer previous slide).
- Rs is always consistent in its position for LW and SW.

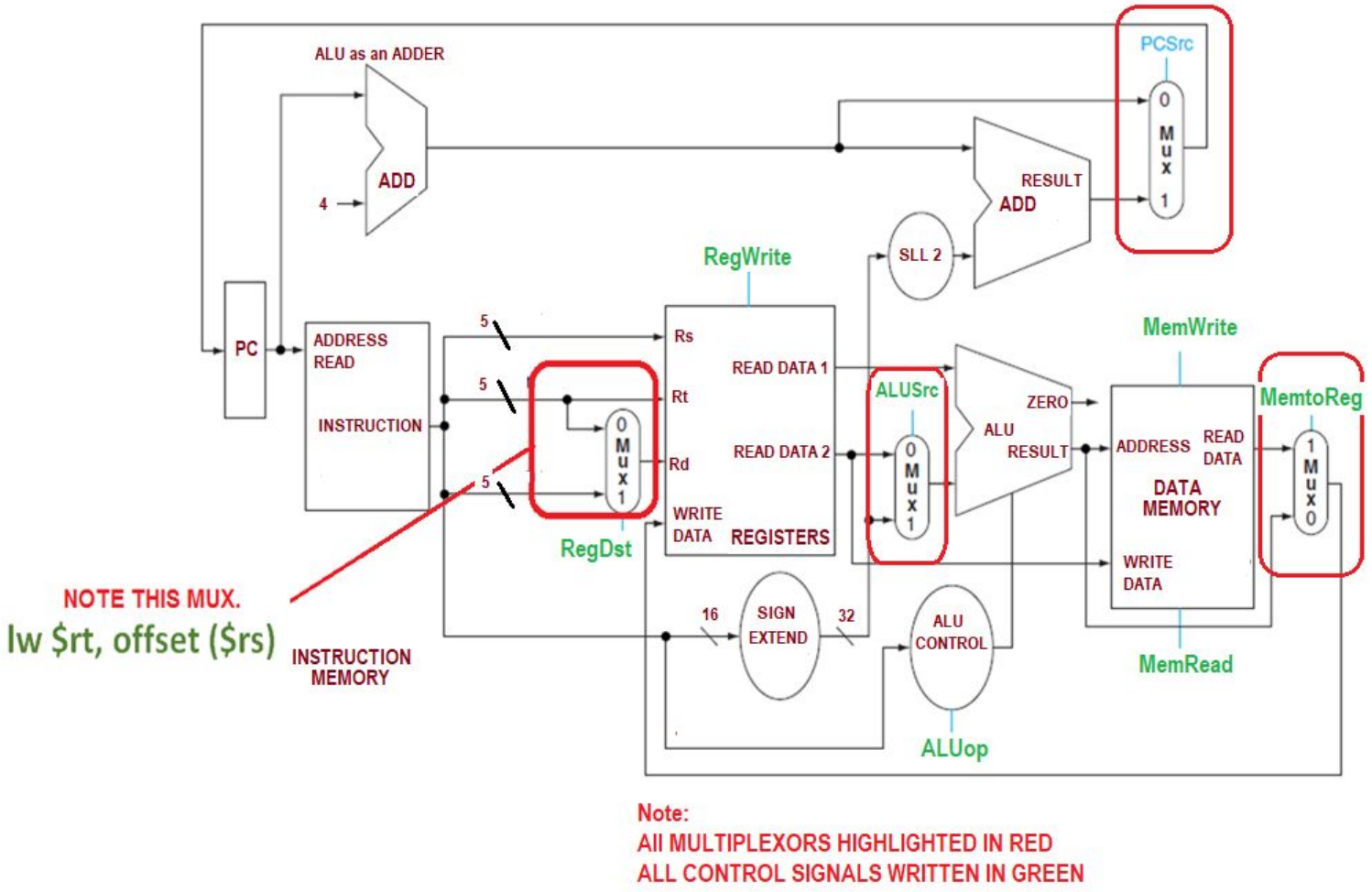


CONTD.,

- Having looked into these identities, we could come to a decision of using multiplexors at appropriate places.
  - Wherever multiplexors are used in this datapath, it will have only 2 inputs.
  - Also, there is definite need for control signal and one control signal shall be there for each multiplexor we use in the data path.
- Also, we could decide on the usage of the control signals for the datapath to work fine.
  - Recollect the control signals we used sometime back.

CONTD.,

Note:
Assert – to represent a signal as logically high.
De-assert – logically low.



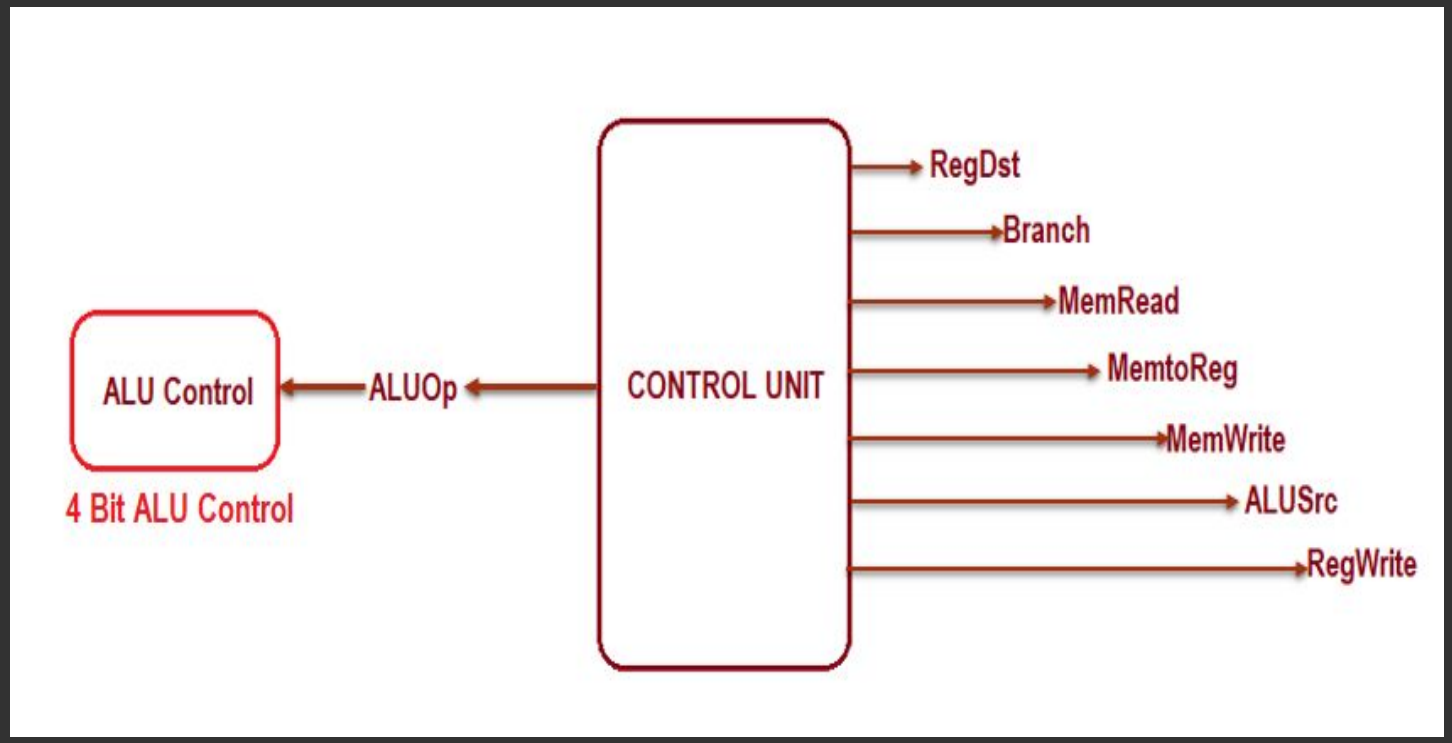
Note this points:

1. The figure is complete now with all the multiplexors included.
2. All the control signals are in place.
3. ALU Control is also added to make the datapath complete.

Also, there is no PCWrite as MemWrite or RegWrite. (This happens automatically at the end of each clock cycle and no one needs to give an alarm to do this task)

CONTD.,

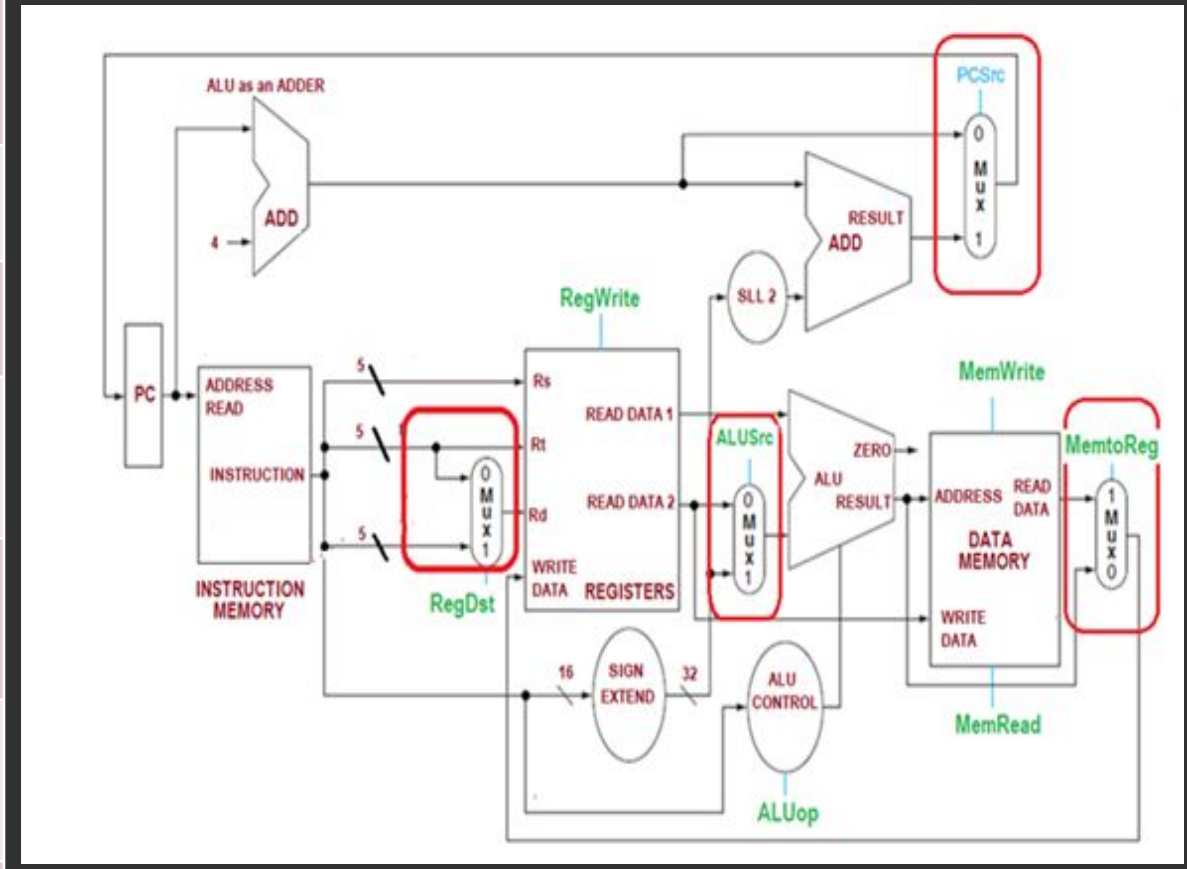
- Now comes the real control unit and the control signals in picture.



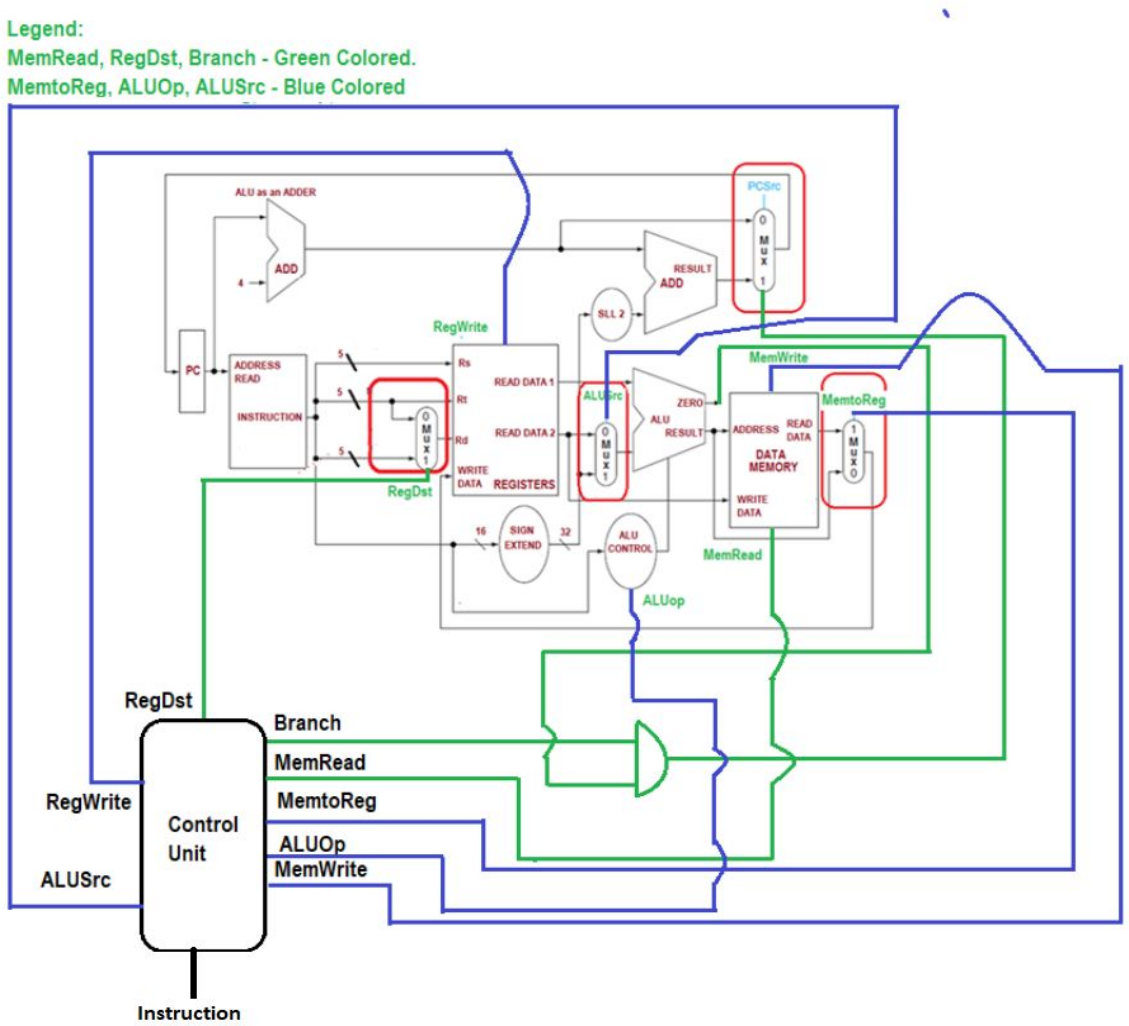
ALU CONTROL LINES	Function
0000	AND
0001	OR
0010	ADD
0110	SUBTRACT
0111	SET ON LESS THAN
1100	NOR

LET US UNDERSTAND THE CONTROL SIGNALS

Control Signal	De-asserted state	Asserted state
RegDst	As one can see from RHS, when de-asserted, destination register is identified by Rt.	If asserted, destination register is identified by Rd.
RegWrite	Not Applicable.	Selected register shall be written with the write data input.
ALUSrc	Input shall be from READ DATA 2 when de-asserted.	Sign extended 32 bit input.
PCSrc	PC shall get value of PC + 4 (This is how the computation happens normally)	This will work for the branching. PC shall be fed with the output of the adder which shall update the branch address.
MemRead	None	Content shall be read from the memory location pointed by the address and made available in Read Data output.
MemWrite	None	Writing into the memory happens from the write data inputs. Memory address generated earlier shall be used.
MemtoReg	Value fed to Register write comes out directly from ALU.	Value fed to Register write comes out directly from data memory.

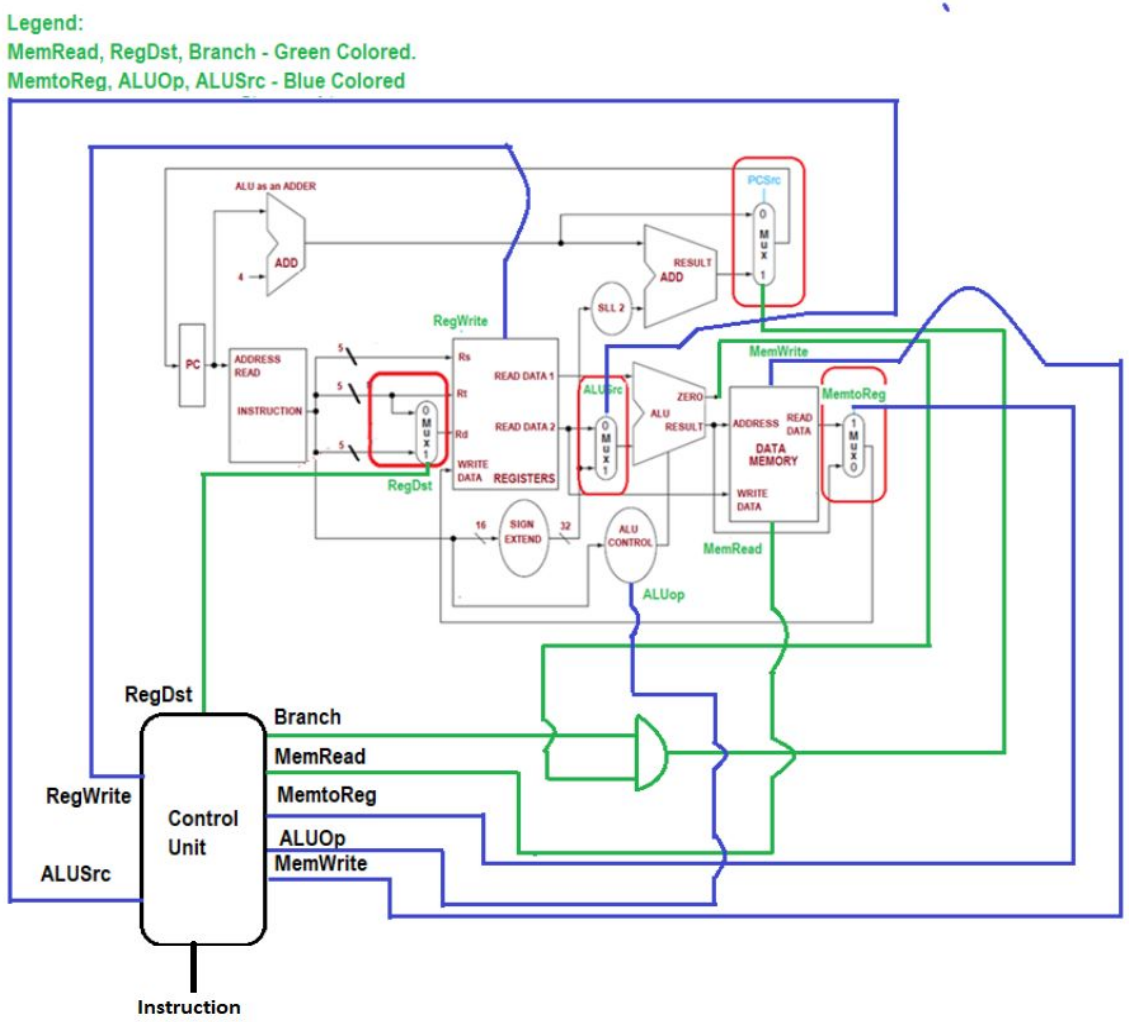


IN DETAIL…



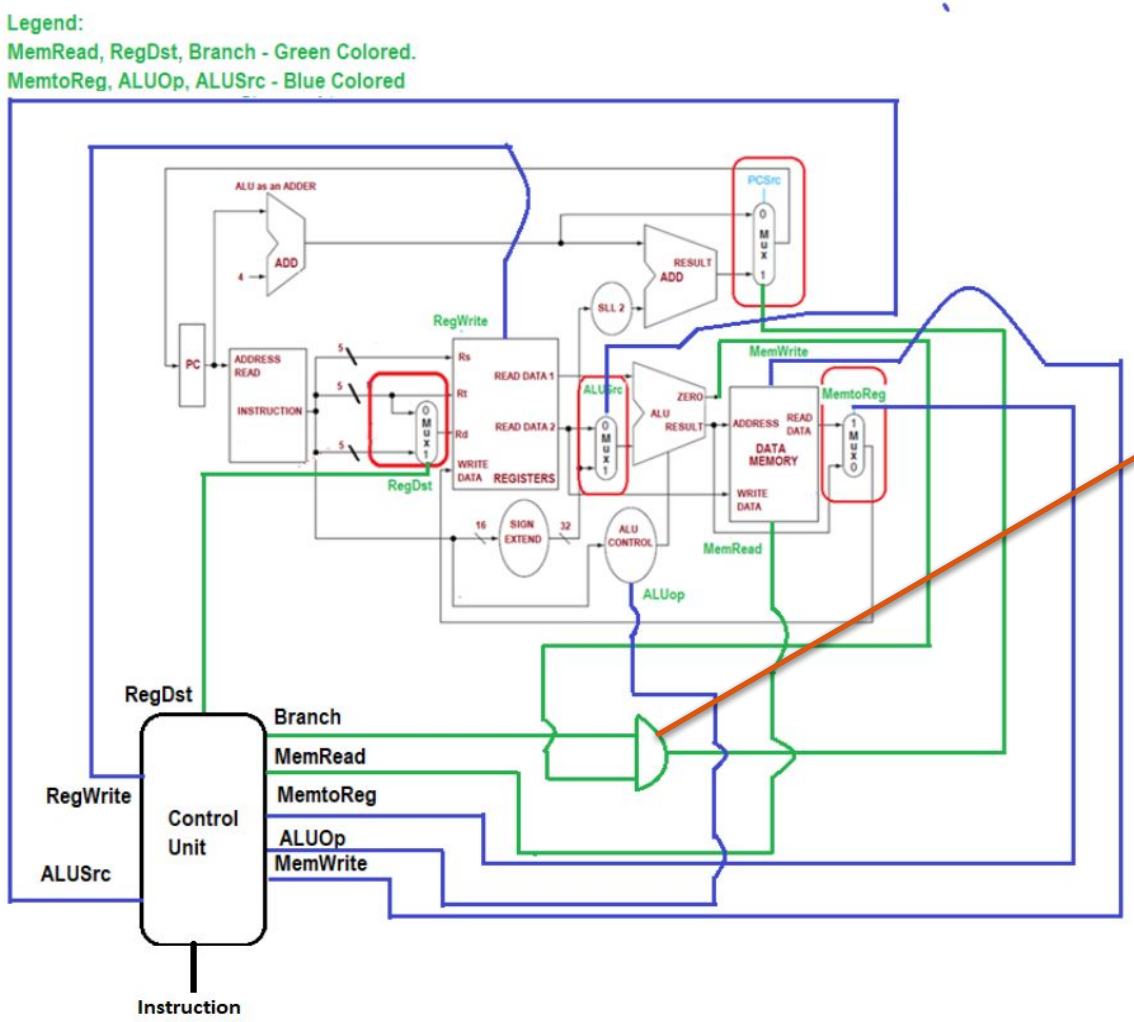
Control Signal	De-asserted state	Asserted state
RegDst	As one can see from RHS, when de-asserted, destination register is identified by Rt.	If asserted, destination register is identified by Rd.
RegWrite	Not Applicable.	Selected register shall be written with the write data input.
ALUSrc	Input shall be from READ DATA 2 when de-asserted.	Sign extended 32 bit input.
PCSrc	PC shall get value of PC + 4 (This is how the computation happens normally)	This will work for the branching. PC shall be fed with the output of the adder which shall update the branch address.
MemRead	None	Content shall be read from the memory location pointed by the address and made available in Read Data output.
MemWrite	None	Writing into the memory happens from the write data inputs. Memory address generated earlier shall be used.
MemtoReg	Value fed to Register write comes out directly from ALU.	Value fed to Register write comes out directly from data memory.

CONTD.,



- The multiplexors are governed by the control signals – RegDst, ALUSrc, MemtoReg.
- There are signals which govern the Read and Write. Read and write can be carried out in the Register and Memory. Signals RegWrite, MemRead, MemWrite are taking care of the read and write and control the same.
- Branch is the signal which would control the branching.
- ALU is controlled by ALUOp.

CONTD.,



- One could spot and AND gate in the diagram. It is fed with the Branch and Zero output from ALU as input.
- This would control what the PC should be. (Means, should it be a branch or should it be a traditional next successive address.
- PCSrc does not directly come from the control unit. (Derived signal)

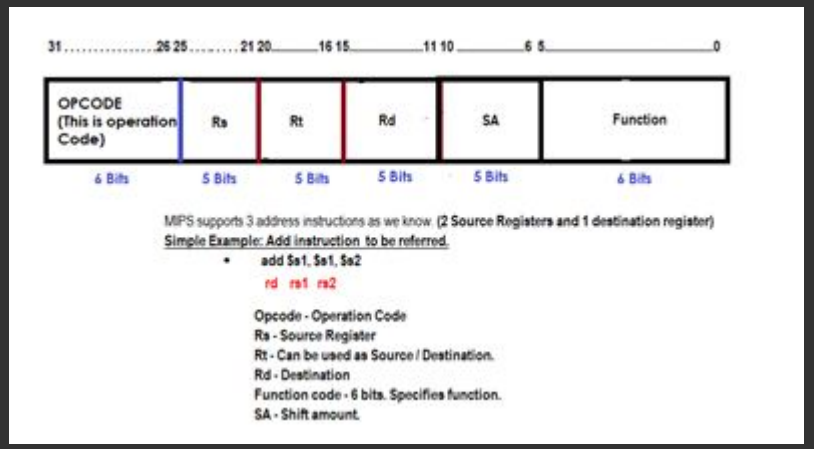
CONTROL SIGNALS – FOR ALL SIGNALS

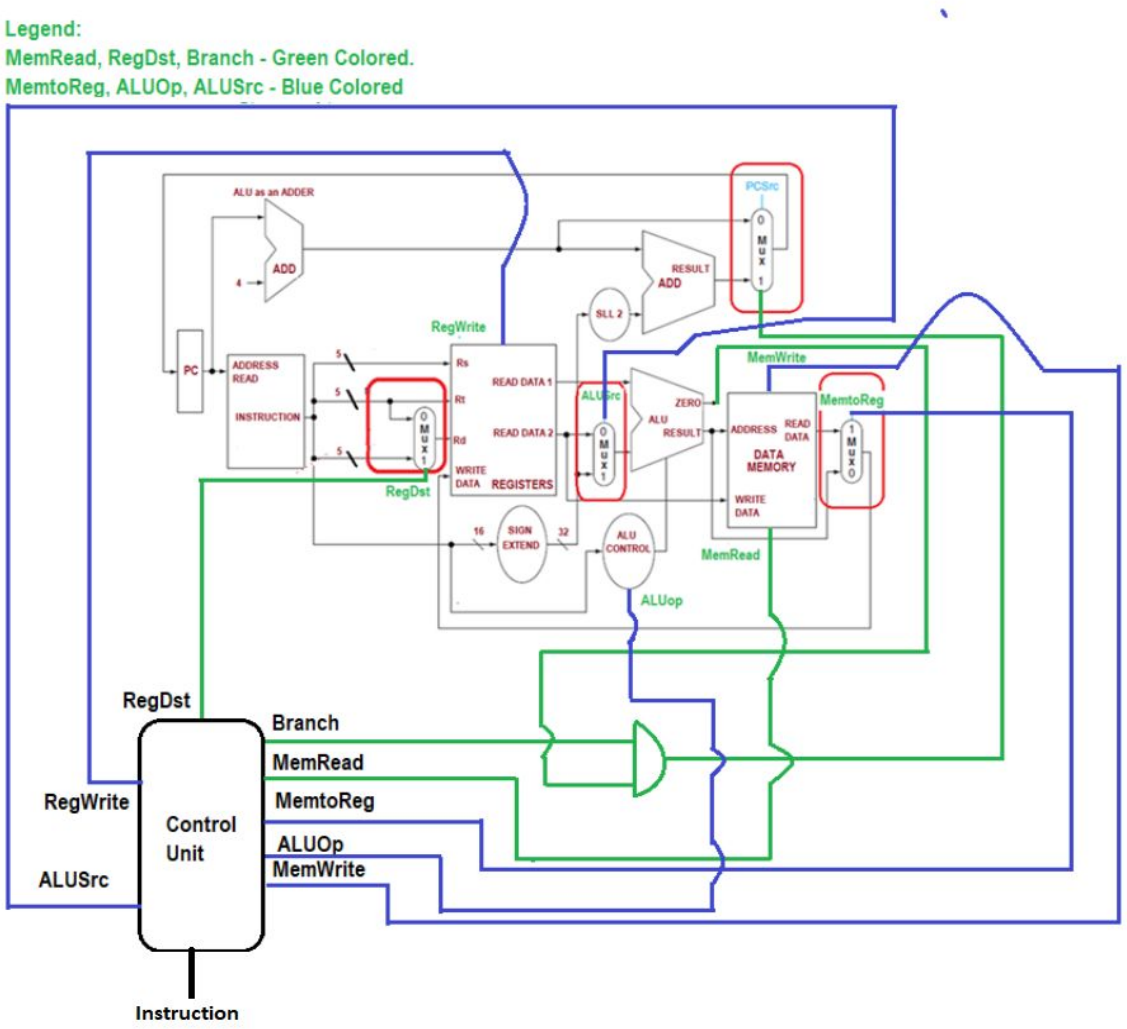
Shriram K Vasudevan

Session - 14

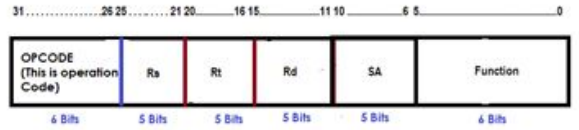
LET US DECODE THE STATUS OF THE CONTROL SIGNALS

- Control Signal Status for R – Format Instructions:
- Let us understand the format once again, with having an instance which we discussed earlier.
- Now, it is to be understood what the signals are which could be high/low for R operations to be performed.





RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP1	ALUOP2
1	0	0	1	0	0	0	1	0



MIPS supports 3 address instructions as we know. **(2 Source Registers and 1 destination register)**

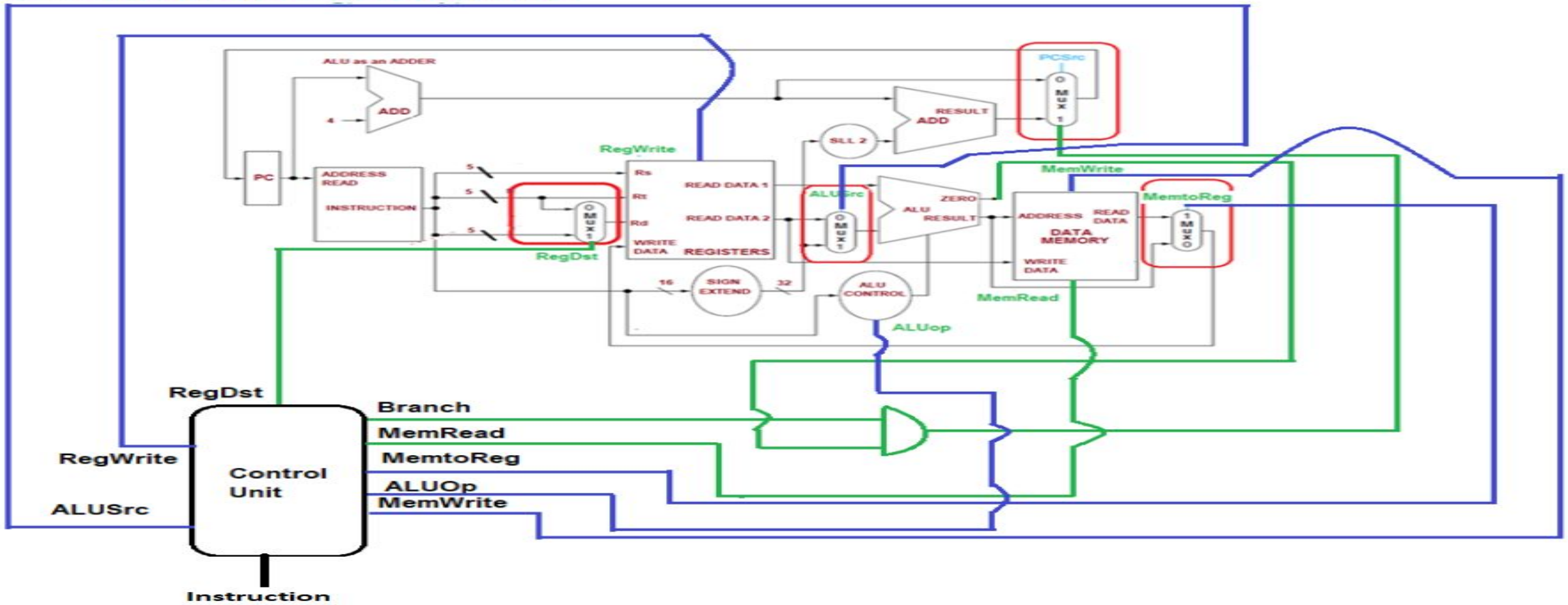
Simple Example: Add instruction to be referred.

- add $s1, $s1, $s2

  rd rs1 rs2

Opcode - Operation Code
Rs - Source Register
Rt - Can be used as Source / Destination.
Rd - Destination
Function code - 6 bits. Specifies function.
SA - Shift amount.

Legend:
MemRead, RegDst, Branch - Green Colored.
MemtoReg, ALUOp, ALUSrc - Blue Colored

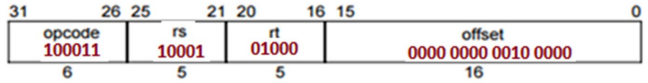


RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP1	ALUOP2
0	1	1	1	1	0	0	0	0

rt rs

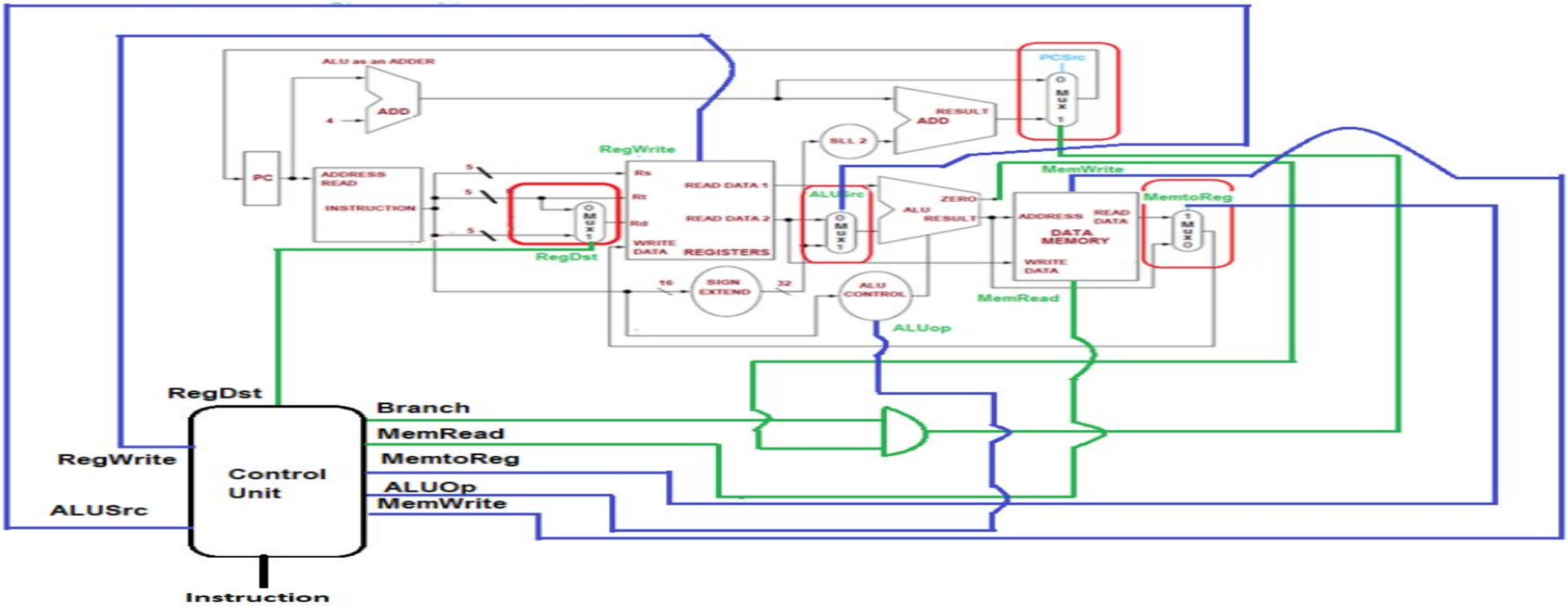
- lw $to, 32 ($s1)
  - 32 is referred to be immediate value. (value being part of instruction is immediately enforced)
  - Assuming $s1 has 500, 500 + 32 = 532 will be the address to look for data. Content of 532 will be retrieved and stored at $t0

I-Type (Immediate).



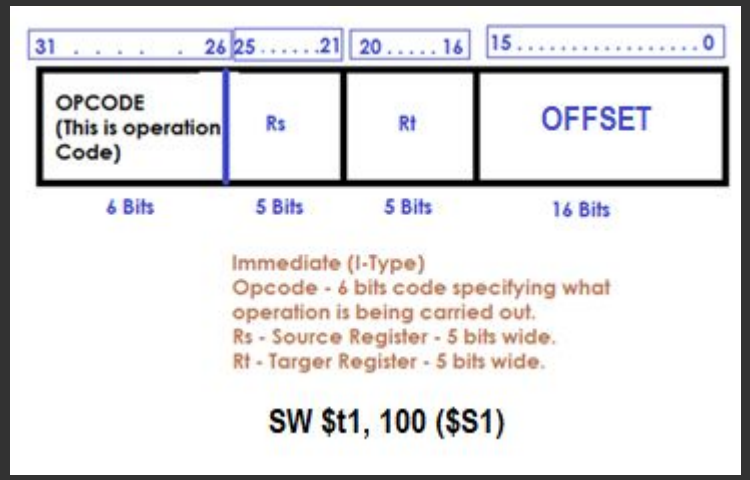
The lw instruction is identified by 35

Legend:
MemRead, RegDst, Branch - Green Colored.
MemtoReg, ALUOp, ALUSrc - Blue Colored

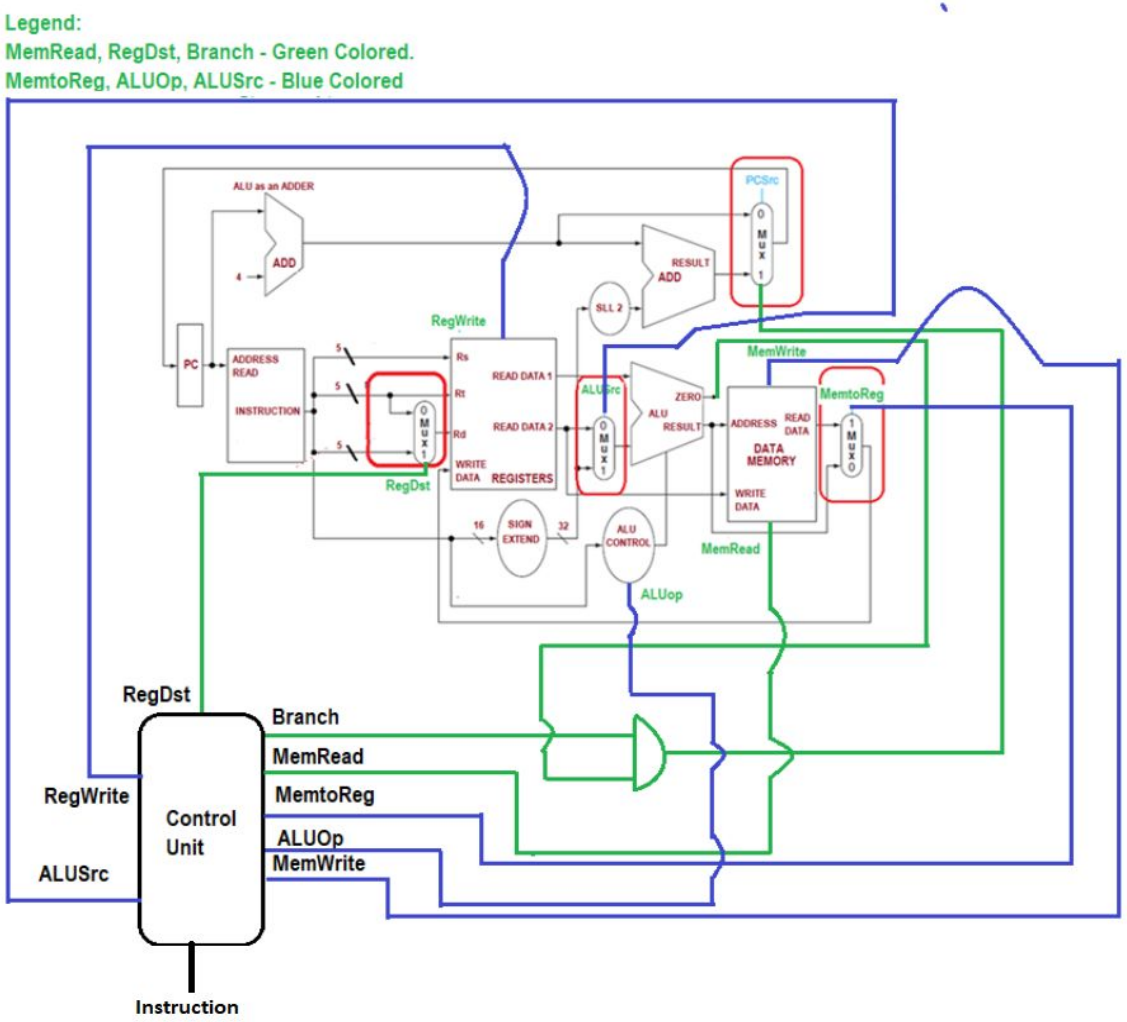


RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP1	ALUOP2
X	1	X	0	0	1	0	0	0

- The status of the control signals shall be as shown above in the table!
- An instance shall help you understanding this better.

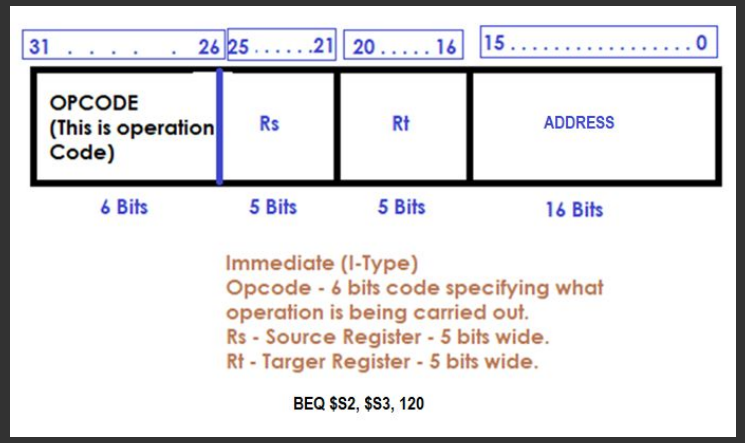


FOR SW INSTRUCTIONS

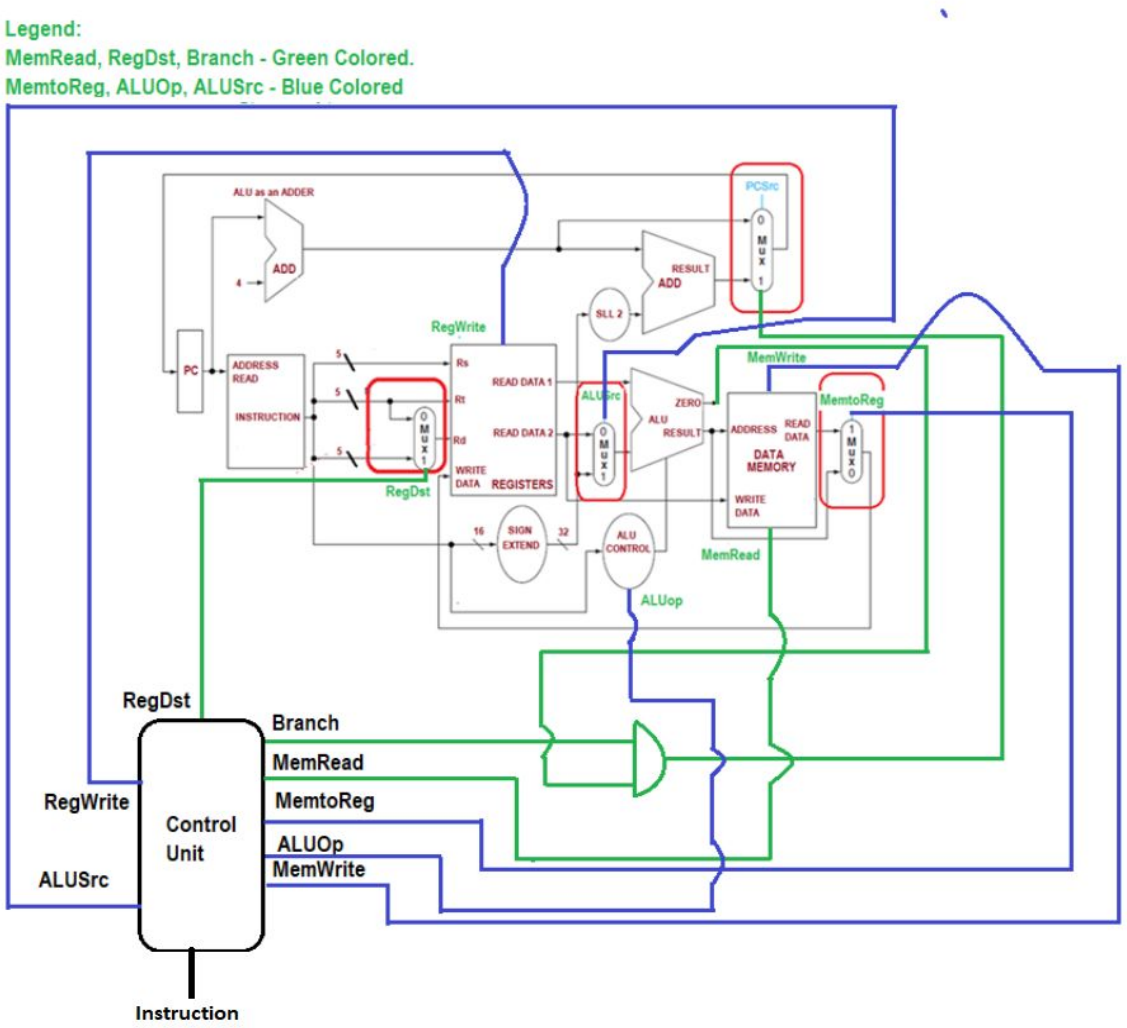


RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOP1	ALUOP2
X	0	X	0	0	0	1	0	1

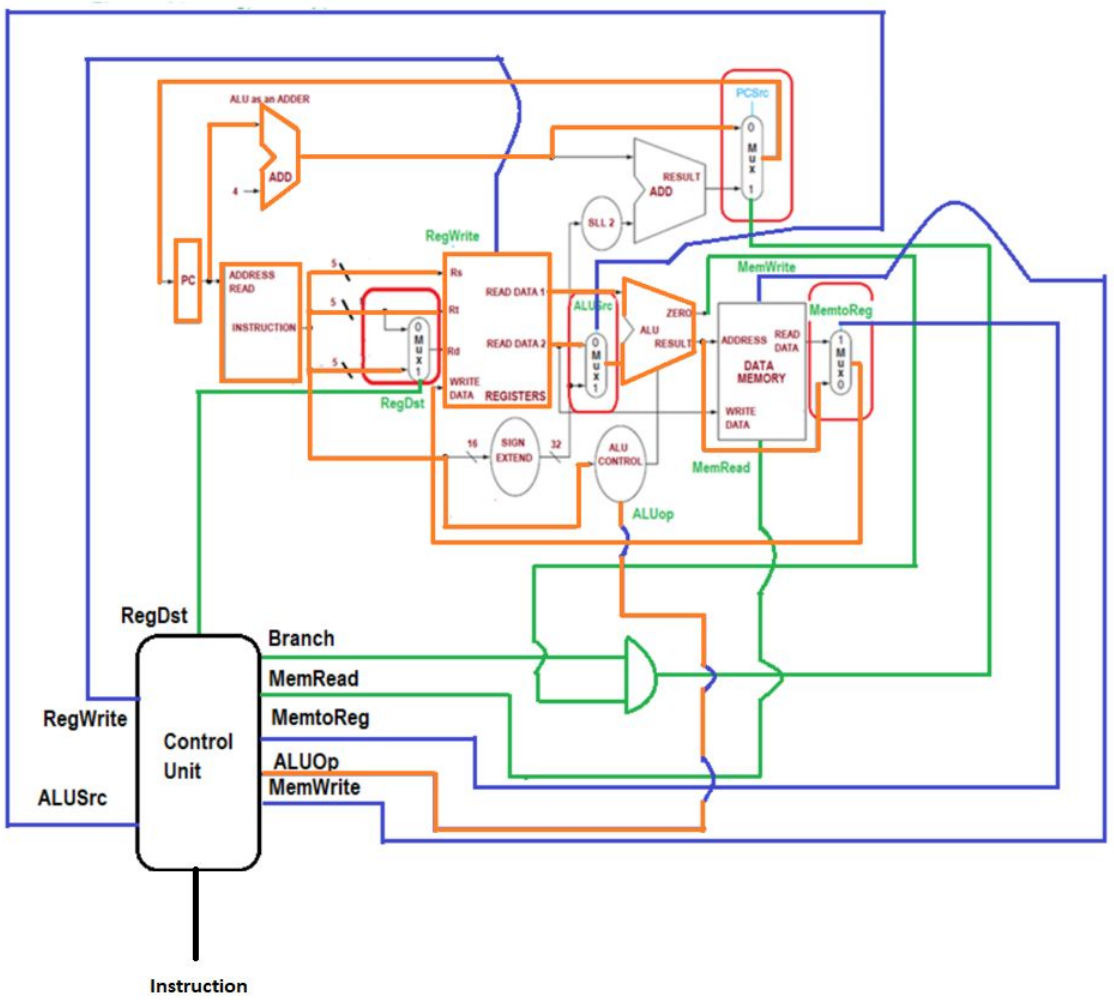
- The status of the control signals shall be as shown above in the table!
- An instance shall help you understanding this better.



FOR BEQ INSTRUCTIONS



OPERATION ELABORATION – R TYPE (WE PREFER R TYPE HERE)



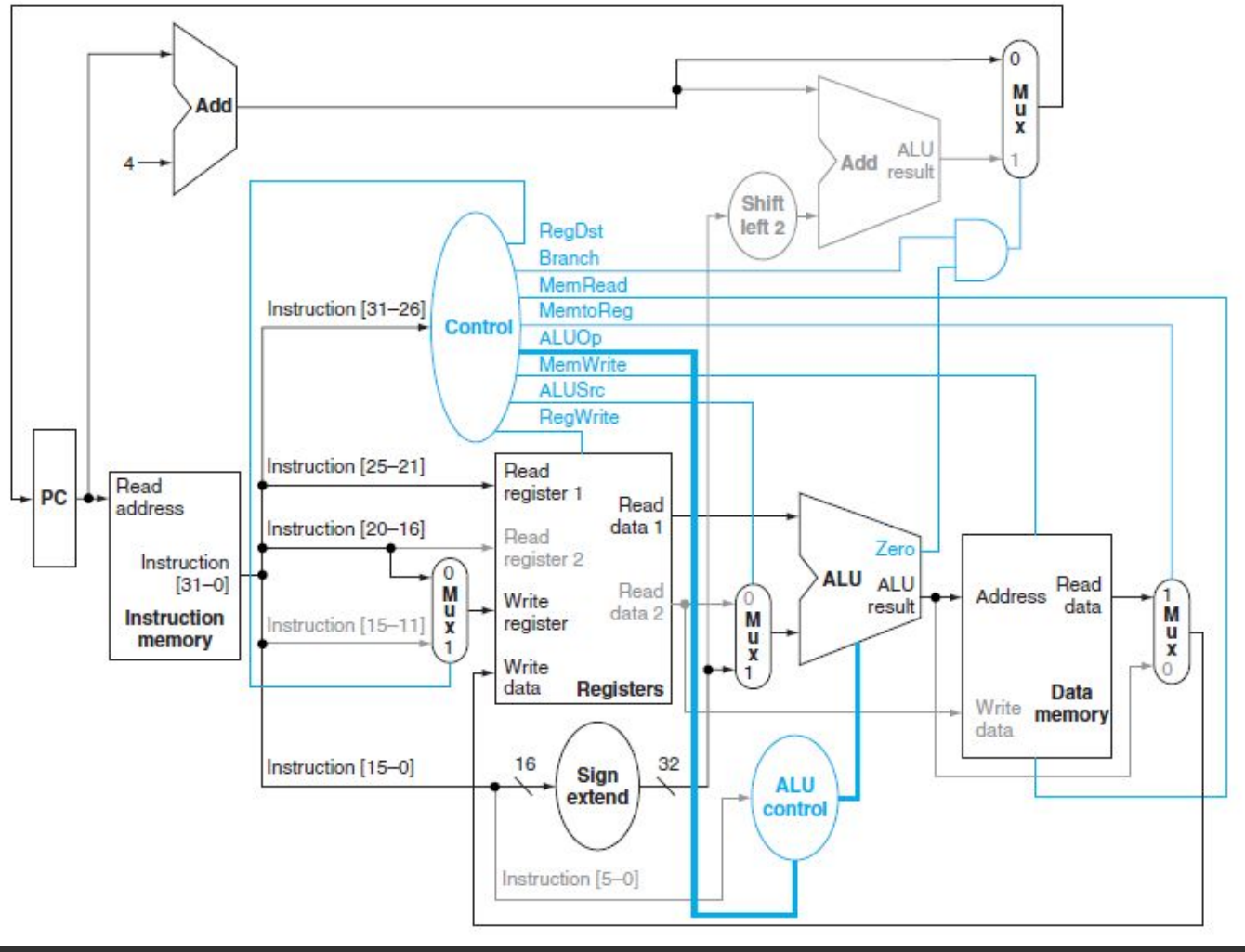
How did we trace the path? (Orange line).

1. When the instruction, add $s1, $s2, $s3 is fetched, the PC gets updated (PC + 4). (This is not a branching instruction, hence branching action would not happen).
2.The source registers (Rs, Rt) are read and shall be fed into the ALU.
3. ALU computes the result and writes the result back to the Register Rd. The entire path is highlighted and traced.

TRACE LOAD/STORE

1. An instruction is fetched from the instruction memory, and the PC is incremented.
2. A register (`$t2`) value is read from the register file.
3. The ALU computes the sum of the value read from the register file and the sign-extended, lower 16 bits of the instruction (`offset`).
4. The sum from the ALU is used as the address for the data memory.
5. The data from the memory unit is written into the register file; the register destination is given by bits 20:16 of the instruction (`$t1`).

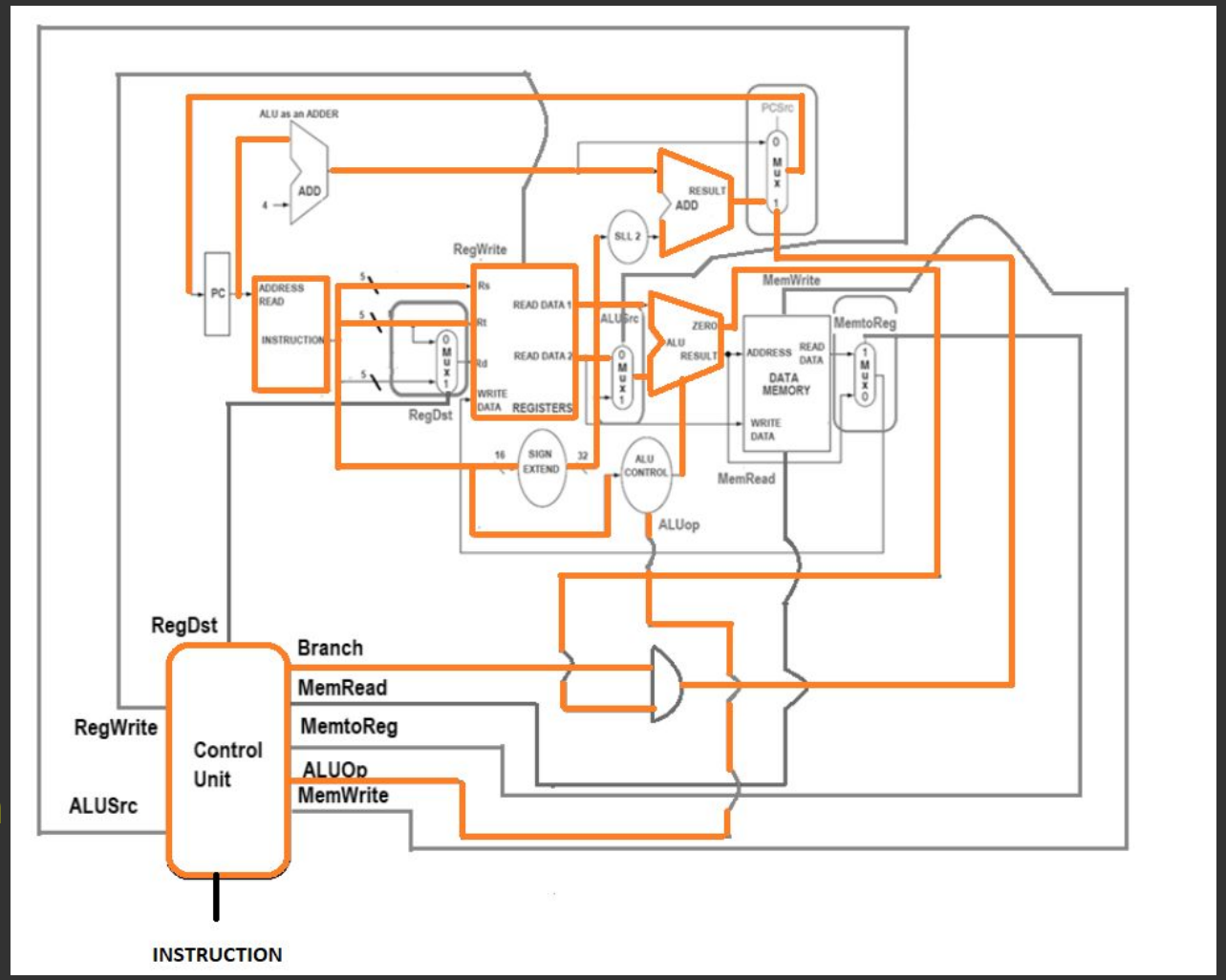
- The control lines, datapath units, and connections that are active are highlighted.
- **A store instruction would operate very similarly.**
- The main difference would be that the memory control would indicate a write rather than a read, the second register value read would be used for the data to store, and the operation of writing the data memory value to the register file would not occur.



TRACE THE PATH FOR BEQ (EASIER)

- Instruction is fetched from the memory. 2 registers are to be compared and we know it already.
- $s1 and $2 are read and they are compared (Literally subtract happens).
- If zero (here we assume it is zero), PC will be updated with appropriate adder result to be stored into the PC. (See the term, Adder result) . The address shall be computed properly by adding PC+4 with sign extended 16 to 32 bit offset.
- If not zero, the regular traditional PC + 4 shall be used.

Note: After using the register file and ALU to perform the compare, the Zero output is used to select the next program counter from between the two candidates.

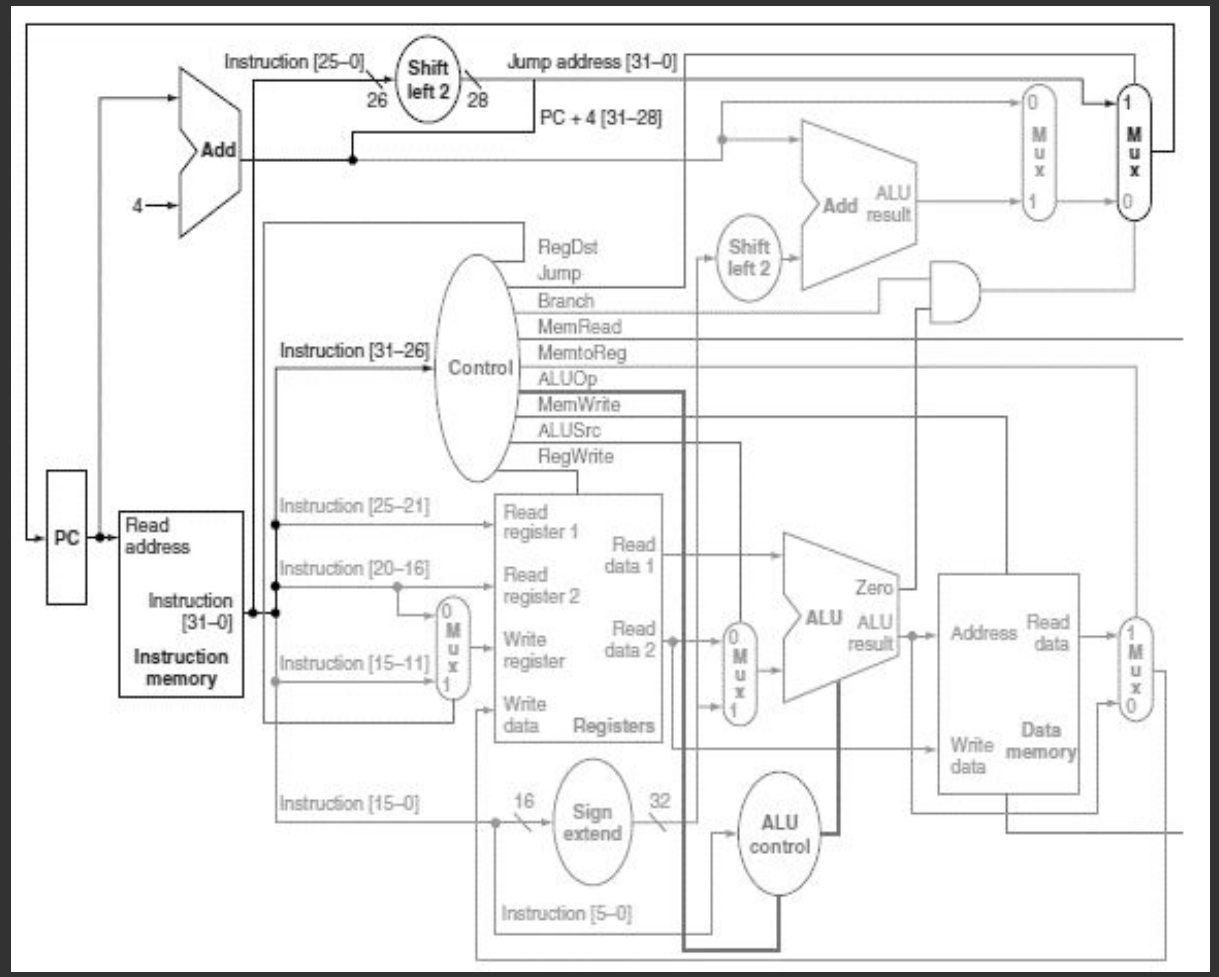


JUMP – A TOUGH ZONE TO PLAY

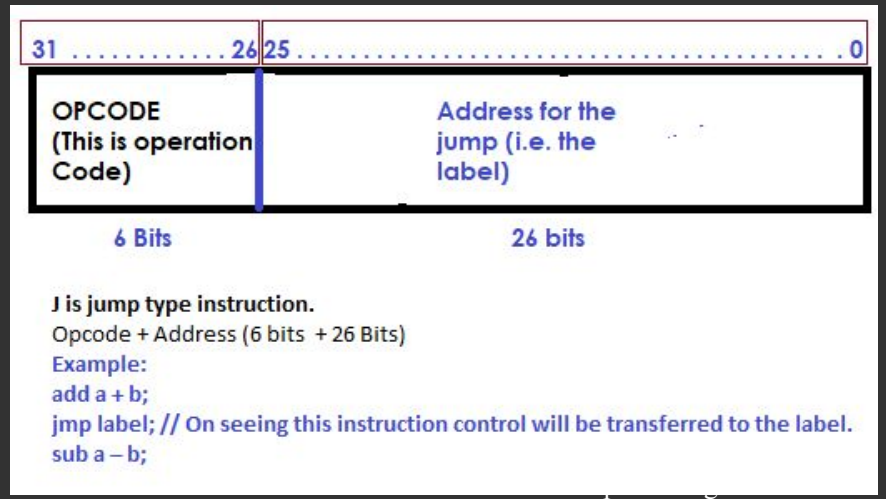
Shriram K Vasudevan

Session – 15

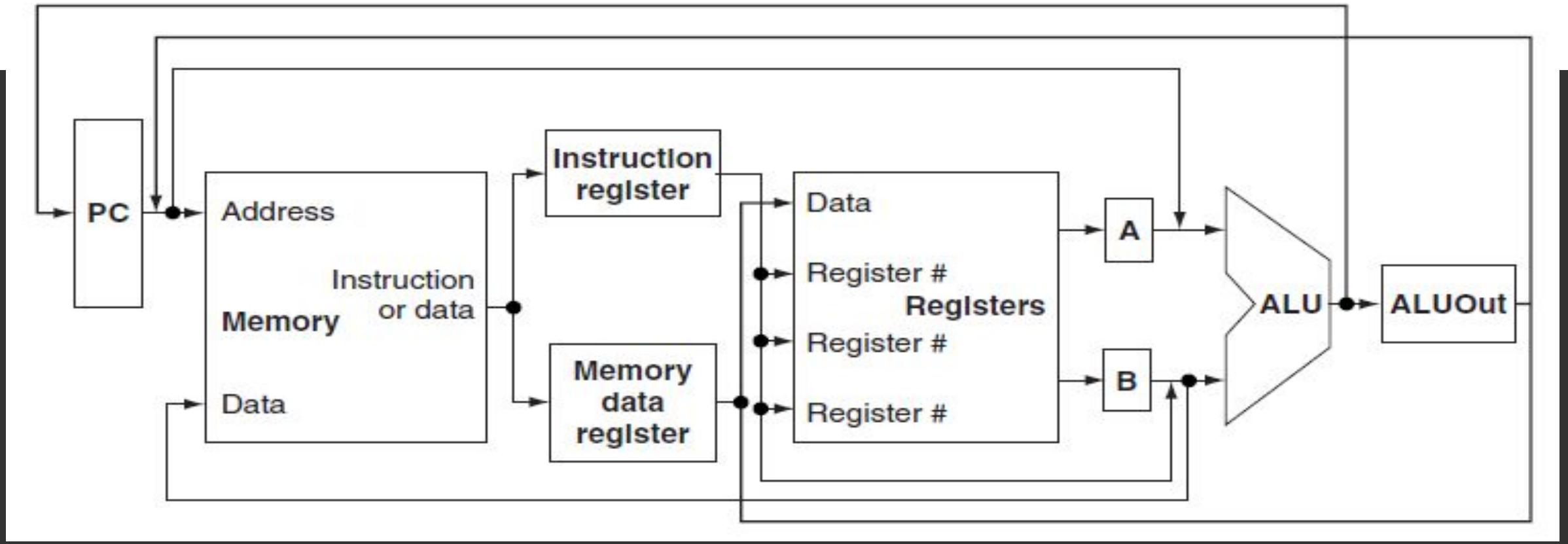
IMPLEMENTING JUMPS!!!! A TOUGH ZONE!



- **An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one.**
- **This multiplexor is controlled by the jump control signal.**
- **The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits, thus yielding a 32-bit address.**

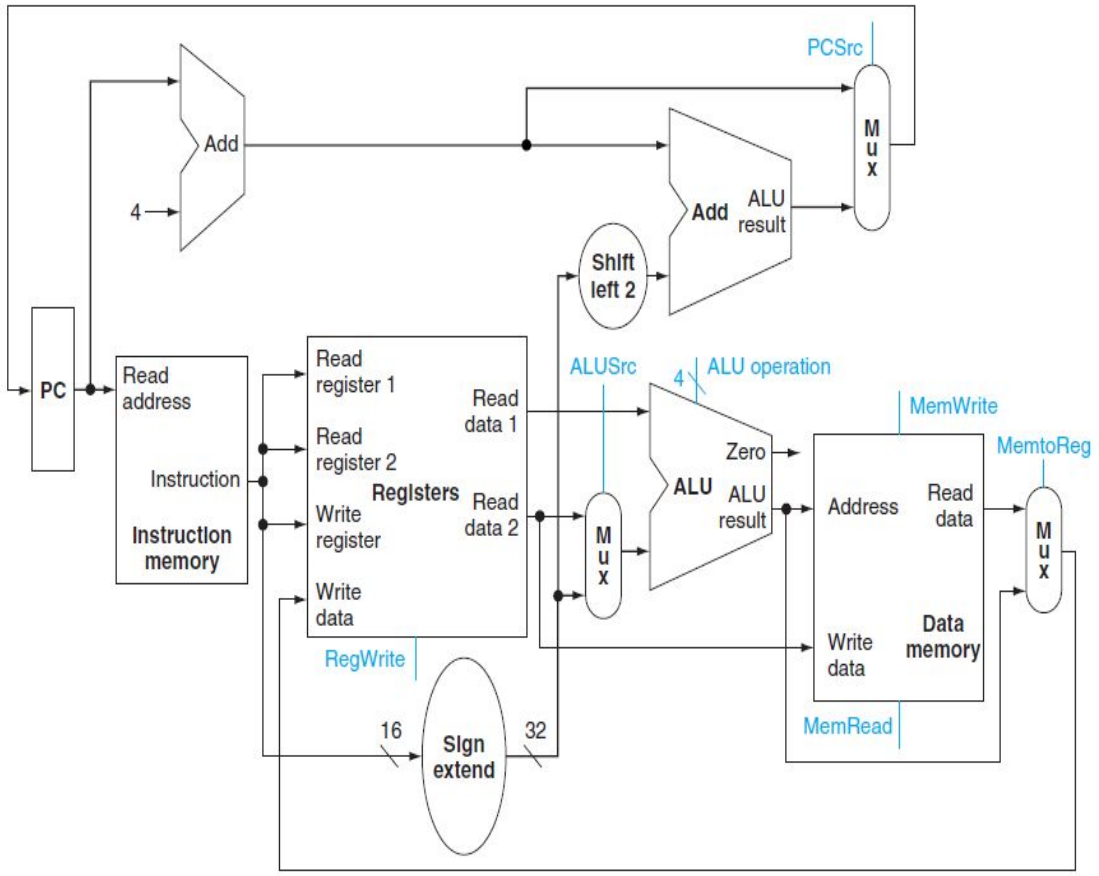


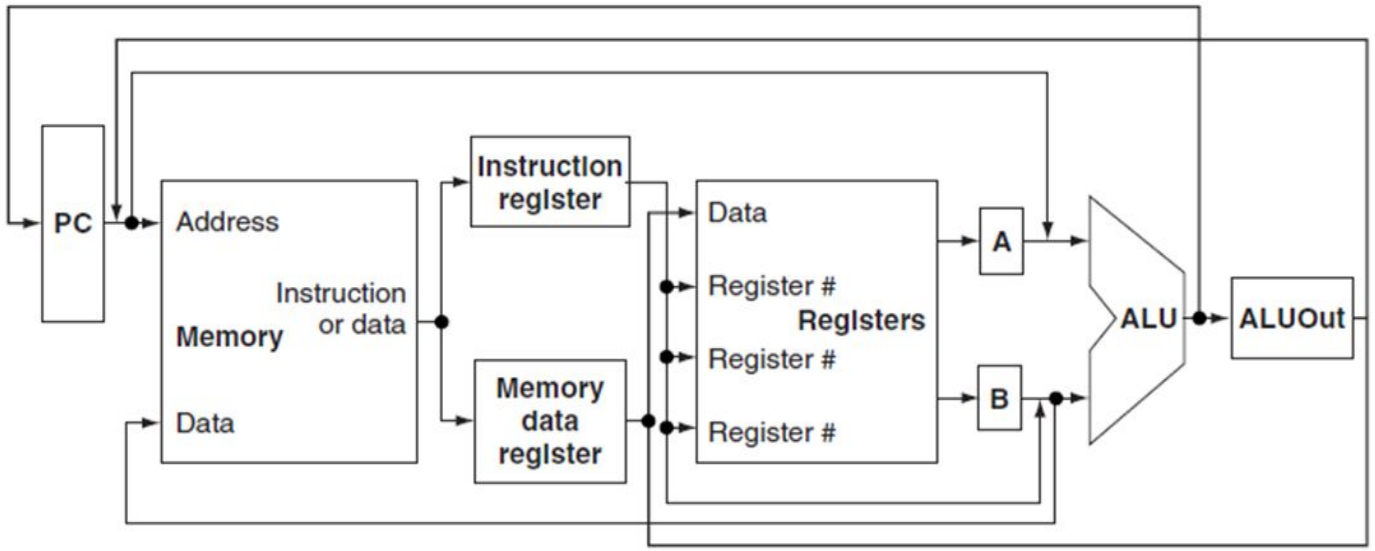
MULTICYCLE IMPLEMENTATION



- This picture shows the key elements of the datapath: **a shared memory unit**, **a single ALU shared among instructions, and the connections among these shared units.**
- The use of shared functional units requires **the addition or widening of multiplexors as well as new temporary registers that hold** data between clock cycles of the same instruction.
- The additional registers are the Instruction register (IR), the Memory data register (MDR), A, B, and ALUOut.

LET US COMPARE…

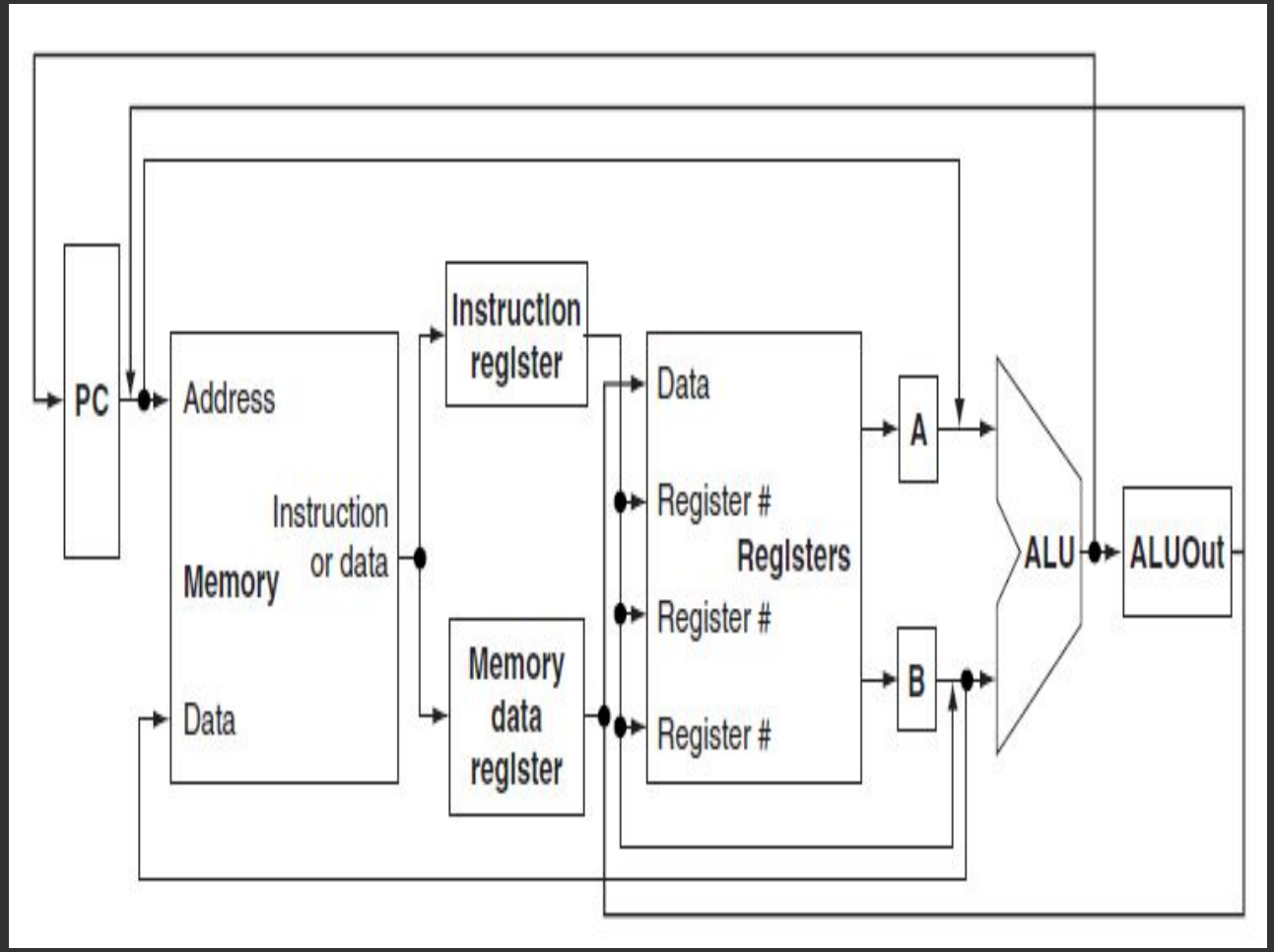




- A single memory unit is used for both instructions and data.
- There is a single ALU, rather than an ALU and two adders.
- One or more registers are added after every major functional unit to hold the output of that unit until the value is used in a subsequent clock cycle.

CONTD.,

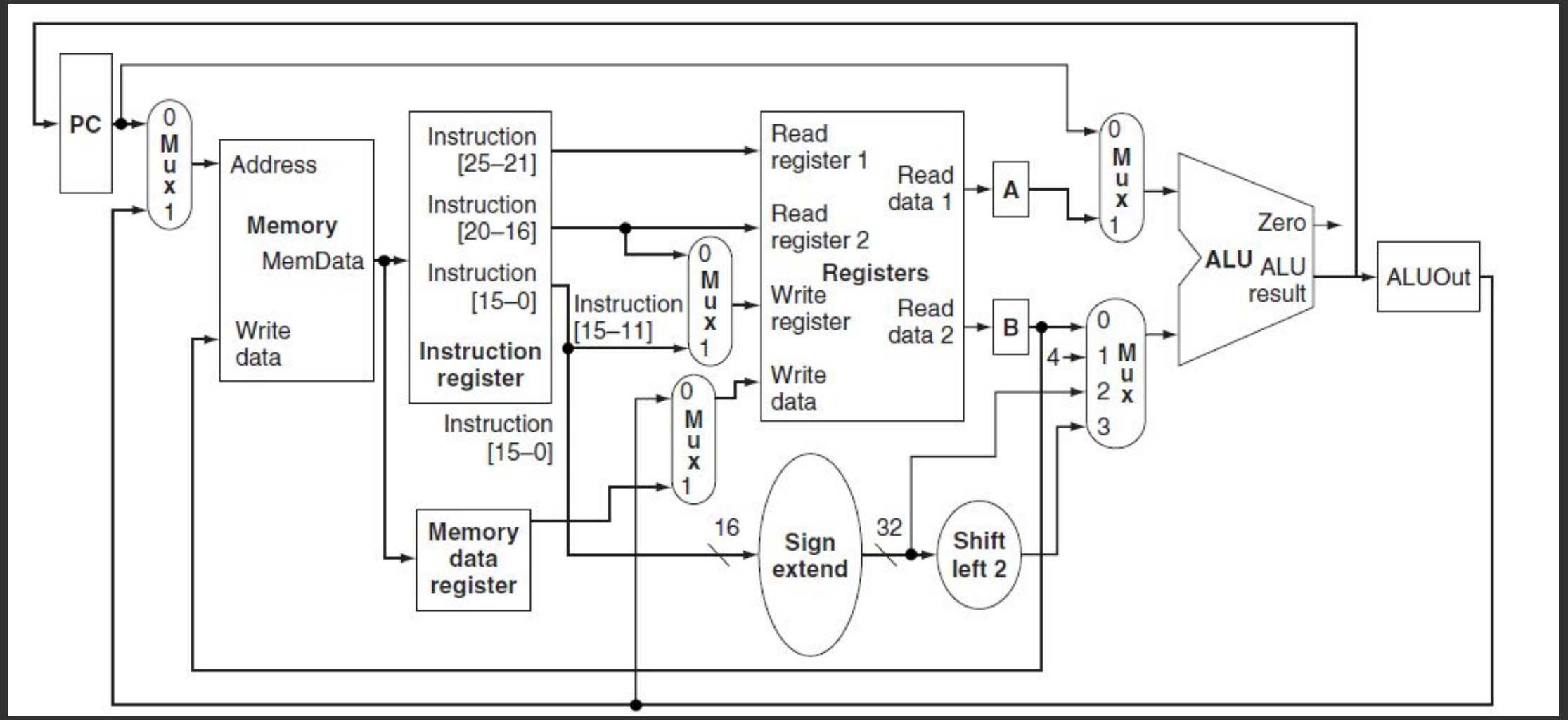
- The Instruction register (IR) and the Memory data register (MDR) are added to save the output of the memory for an instruction read and a data read, respectively.
- Two separate registers are used, both values are needed during the same clock cycle.
- The A and B registers are used to hold the register operand values read from the register file.
- The ALUOut register holds the output of the ALU



CONTD.,

- The IR needs to hold the instruction until the end of execution of that instruction. (Understand this folks).
- Because several functional units are shared for different purposes, we need both to add multiplexors and to expand existing multiplexors.
- For example, since one memory is used for both instructions and data, we need a multiplexor to select between the two sources for a memory address, namely, the PC (for instruction access) and ALUOut (for data access).

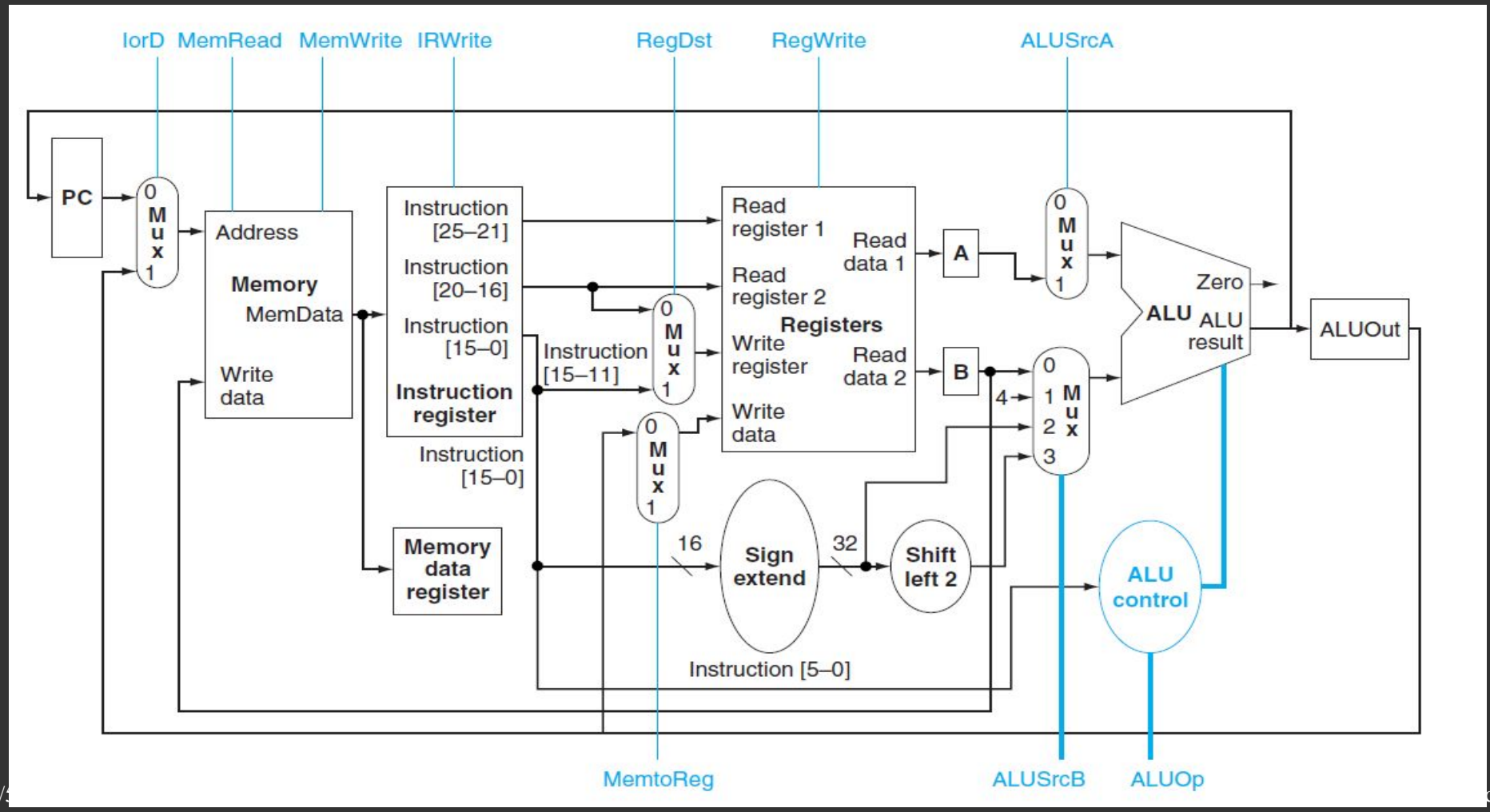
CONTD.,


PC
0
M
u
x
1
Address
Memory
MemData
Write
data
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Memory
data
register
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 1
A
B
4
16
Sign
extend
32
Shift
left 2
0
1
2
3
ALU
Zero
ALU
result
ALUOut

CONTD.,

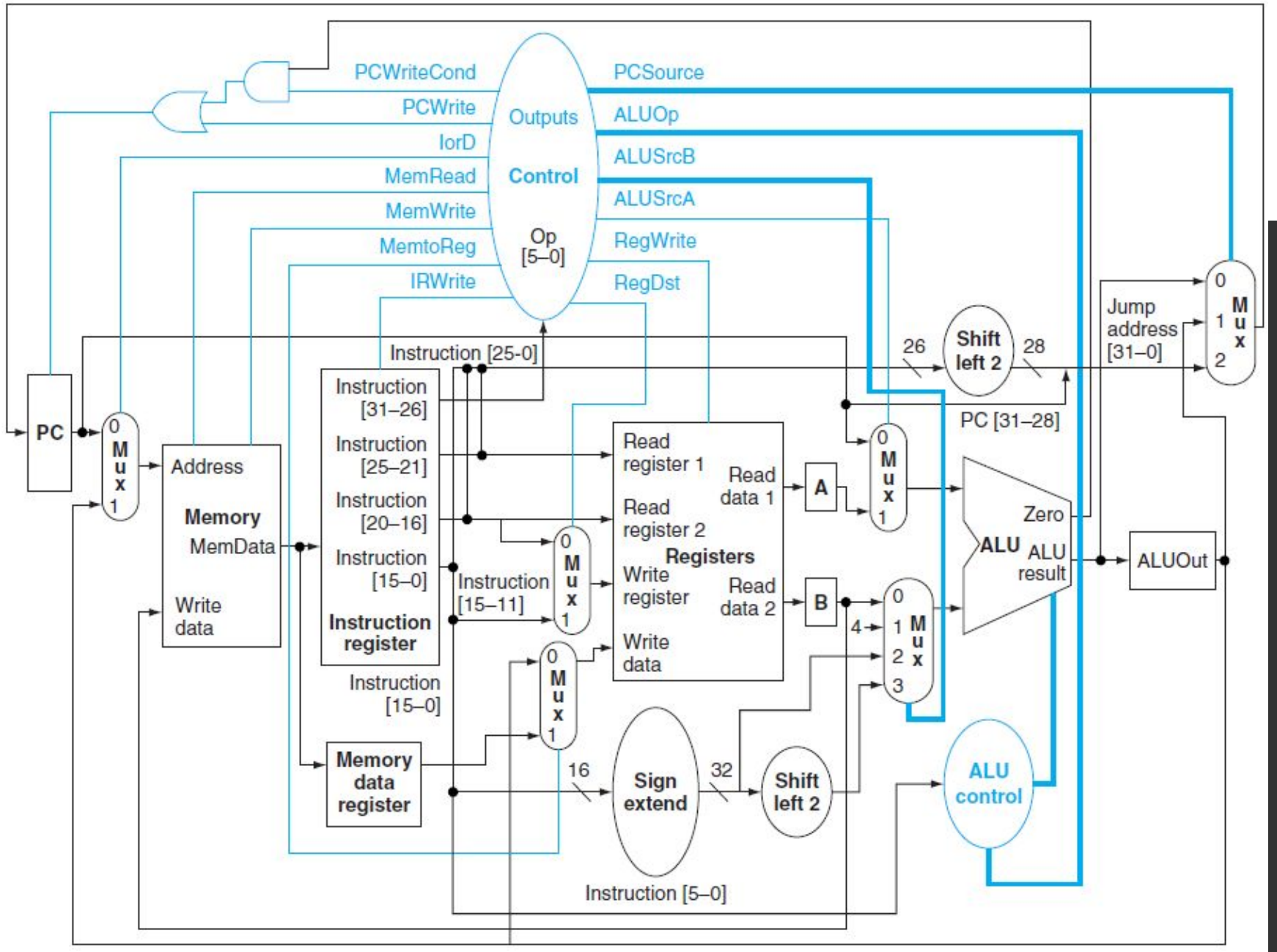
- The multicycle datapath still requires additions to support branches and jumps; after these additions, we will see how the instructions are sequenced and then generate the datapath control.
- With the jump instruction and branch instruction, there are three possible sources for the value to be written into the PC:
- The output of the ALU, which is the value PC + 4 during instruction fetch. This value should be stored directly into the PC.
- The register ALUOut, which is where we will store the address of the branch target after it is computed.
- The lower 26 bits of the Instruction register (IR) shifted left by two and concatenated with the upper 4 bits of the incremented PC, which is the source when the instruction is a jump.

WITH CONTROL SIGNALS..



A BIT MORE DETAILED..

As we observed when we implemented the single-cycle control, the PC is written both unconditionally and conditionally. During a normal increment and for jumps, the PC is written unconditionally. If the instruction is a conditional branch, the incremented PC is replaced with the value in ALUOut only if the two designated registers are equal. Hence, our implementation uses two separate control signals: PCWrite, which causes an unconditional write of the PC, and PCWriteCond, which causes a write of the PC if the branch condition is also true.



CONTD.,

Actions of the 1-bit control signals

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register file destination number for the Write register comes from the rt field.	The register file destination number for the Write register comes from the rd field.
RegWrite	None.	The general-purpose register selected by the Write register number is written with the value of the Write data input.
ALUSrcA	The first ALU operand is the PC.	The first ALU operand comes from the A register.
MemRead	None.	Content of memory at the location specified by the Address input is put on Memory data output.
MemWrite	None.	Memory contents at the location specified by the Address input is replaced by value on Write data input.
MemtoReg	The value fed to the register file Write data input comes from ALUOut.	The value fed to the register file Write data input comes from the MDR.
IorD	The PC is used to supply the address to the memory unit.	ALUOut is used to supply the address to the memory unit.
IRWrite	None.	The output of the memory is written into the IR.
PCWrite	None.	The PC is written; the source is controlled by PCSource.
PCWriteCond	None.	The PC is written if the Zero output from the ALU is also active.

Actions of the 2-bit control signals

Signal name	Value (binary)	Effect
ALUOp	00	The ALU performs an add operation.
	01	The ALU performs a subtract operation.
	10	The funct field of the instruction determines the ALU operation.
ALUSrcB	00	The second input to the ALU comes from the B register.
	01	The second input to the ALU is the constant 4.
	10	The second input to the ALU is the sign-extended, lower 16 bits of the IR.
	11	The second input to the ALU is the sign-extended, lower 16 bits of the IR shifted left 2 bits.
PCSource	00	Output of the ALU (PC + 4) is sent to the PC for writing.
	01	The contents of ALUOut (the branch target address) are sent to the PC for writing.
	10	The jump target address (IR[25:0] shifted left 2 bits and concatenated with PC + 4[31:28]) is sent to the PC for writing.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

@ALL, THANKS! LETS ANALYZE THE PERFORMANCE. SHRIRAM K V