Remember in *ANNA* there are 16 instruction bits, with each instruction type having unique splits.

Instruction Formats

Instructions adhere to one of the following three instruction formats:

R-type (`add`, `sub`, `and`, `or`, `not`, `jalr`, `in`, `out`)

15 12	11 9	8 6	5 3	2 0
Opcode	R_D	R_A	R_B	Function code*

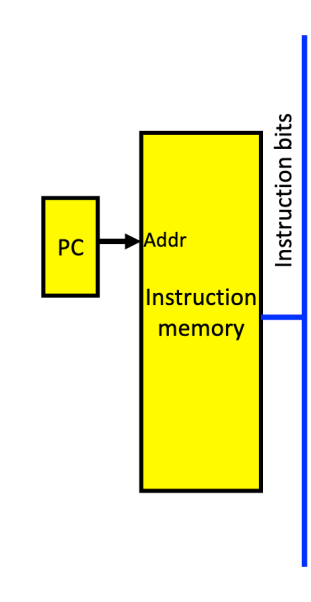
*Function codes for opcode 0000: `add` (000), `sub` (001), `and` (010), `or` (011), `not` (100), `jalr` , `in`, `out` do not use the function code; each has a unique opcode.

I6-type (`addi`, `shf`, `lw`, `sw`)

15 12	11 9	8 6	5 0
Opcode	R_D	R_A	*Imm6*

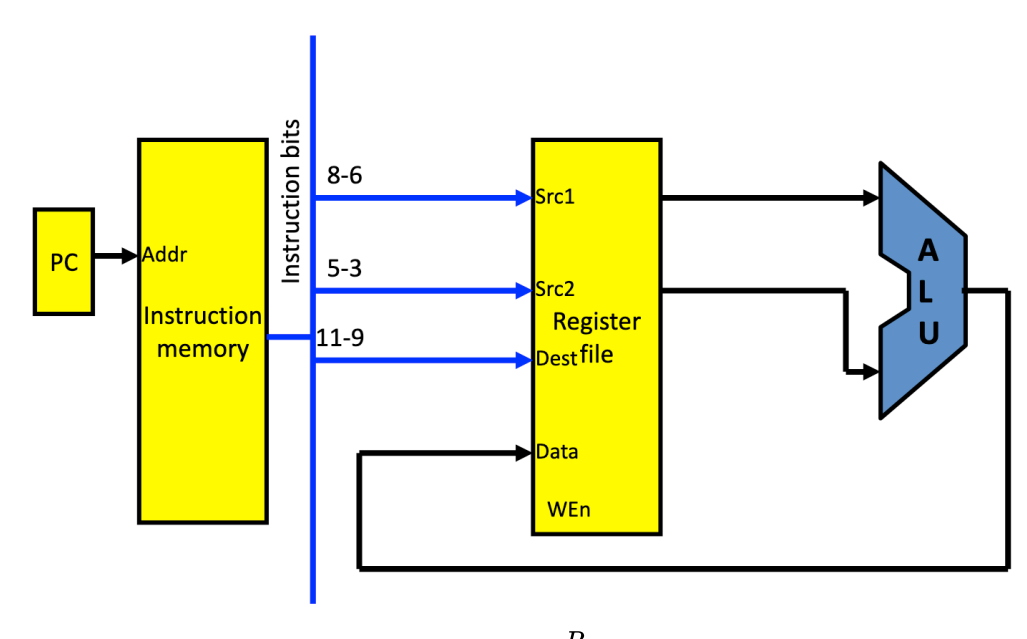
I8-type (`lli`, `lui`, `beq`, `bne`, `bgt`, `bge`, `blt`, `ble`)

15 12	11 9	8	7 0
Opcode	R_D	Unused	*Imm8*



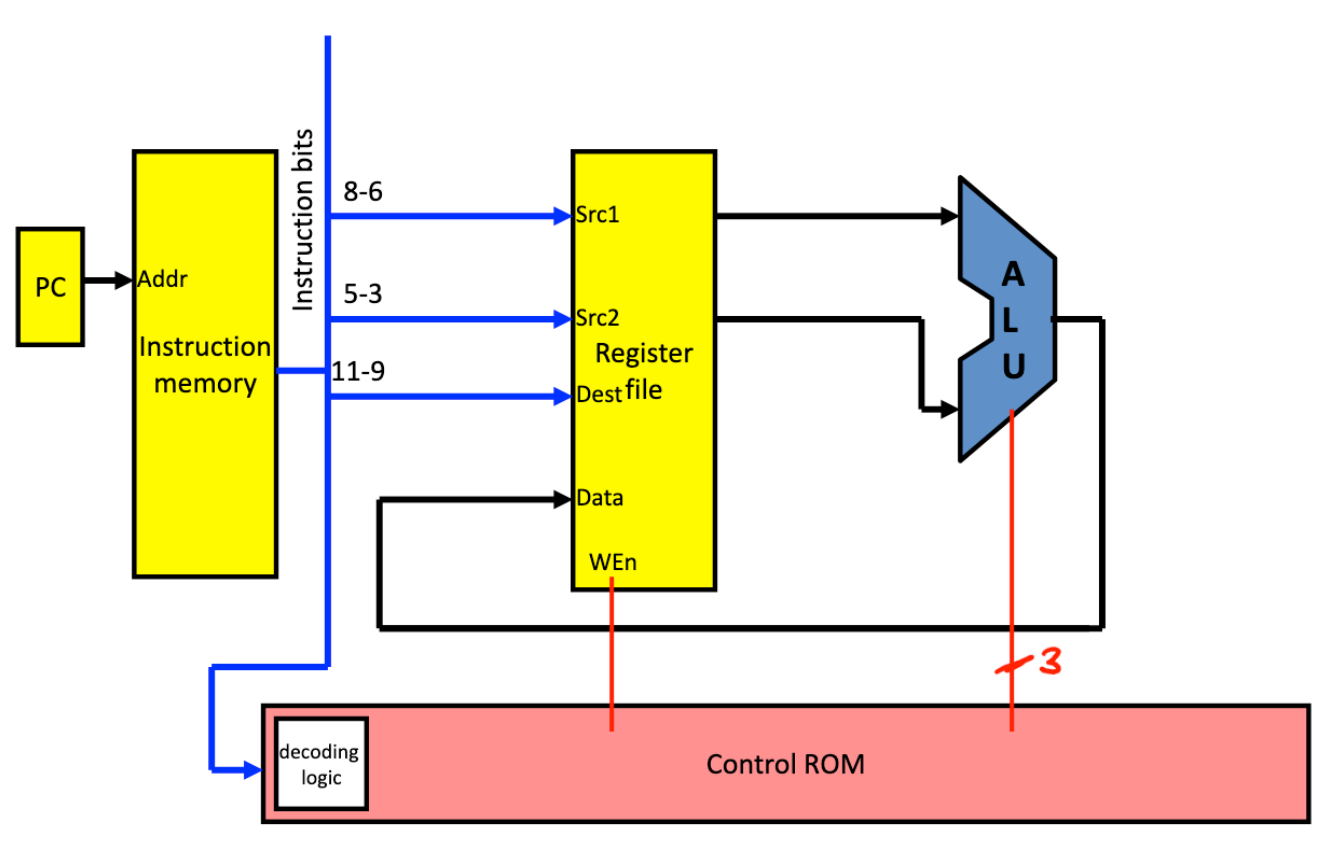
PC (Program Counter):
- Holds the memory address of the next instruction to be executed. Addr is the output of PC.
- Simply a register that stores a number (the address) not the instruction itself.

Instruction Memory:
- Outputs the 16 instruction bits (in *ANNA*) at the memory address given by PC.



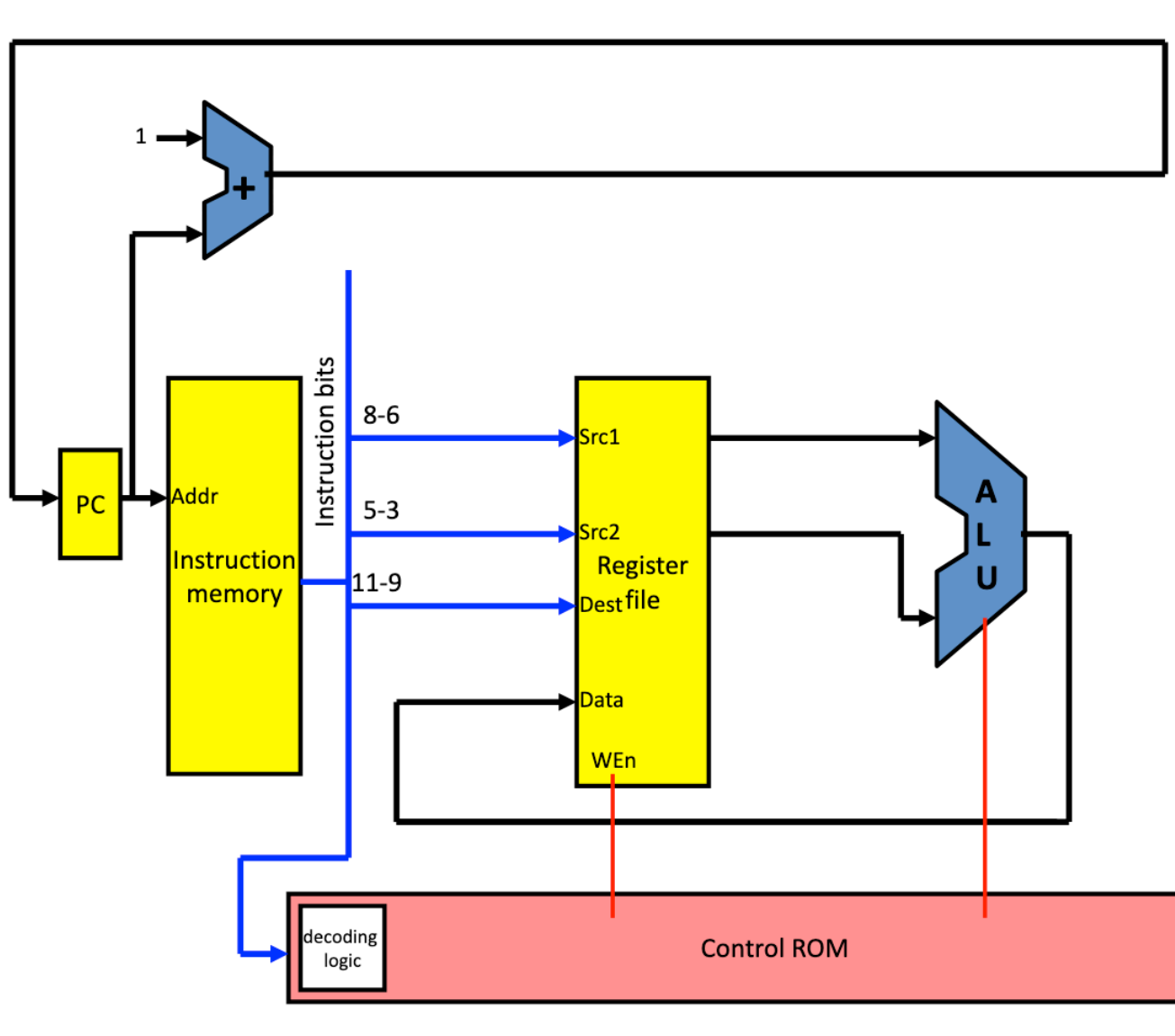
Src1: R-type and I6-type use bits 8-6 to store R_A
Src2: R-type uses bits 5-3 to store R_B
Dest: All instruction types use this to store R_D

Register File: Outputs both Src1 and Src2 to be passed into *ALU*.

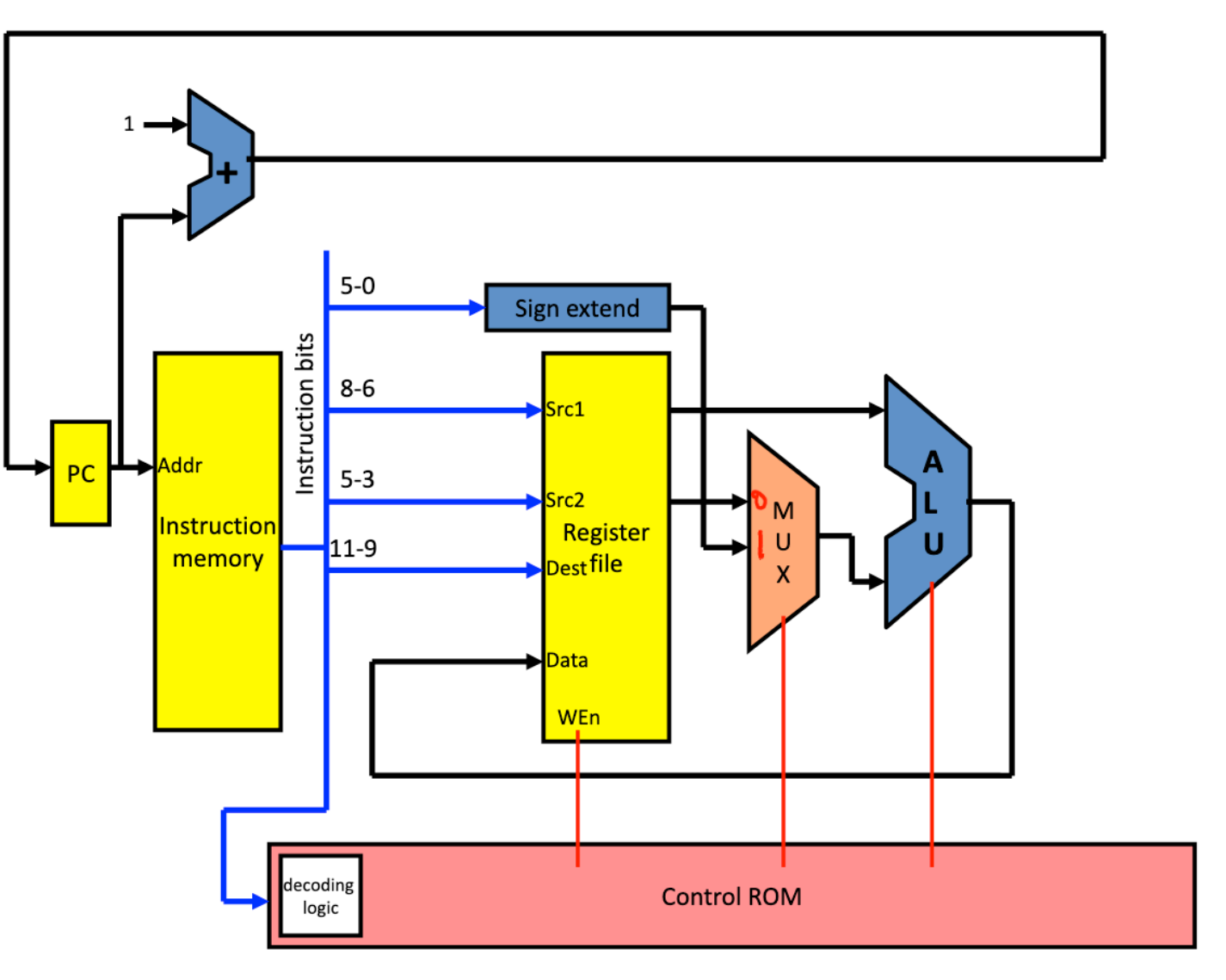


Control ROM:
- Passes in appropriate 3-bit function code to determine what to do with each 16-bit instruction.
- Controls WEn (write enable) to pass Data (output from *ALU*) into Dest.

ALU: Operation with Src1 and Src2 is determined by the function code passed in by the Control ROM.



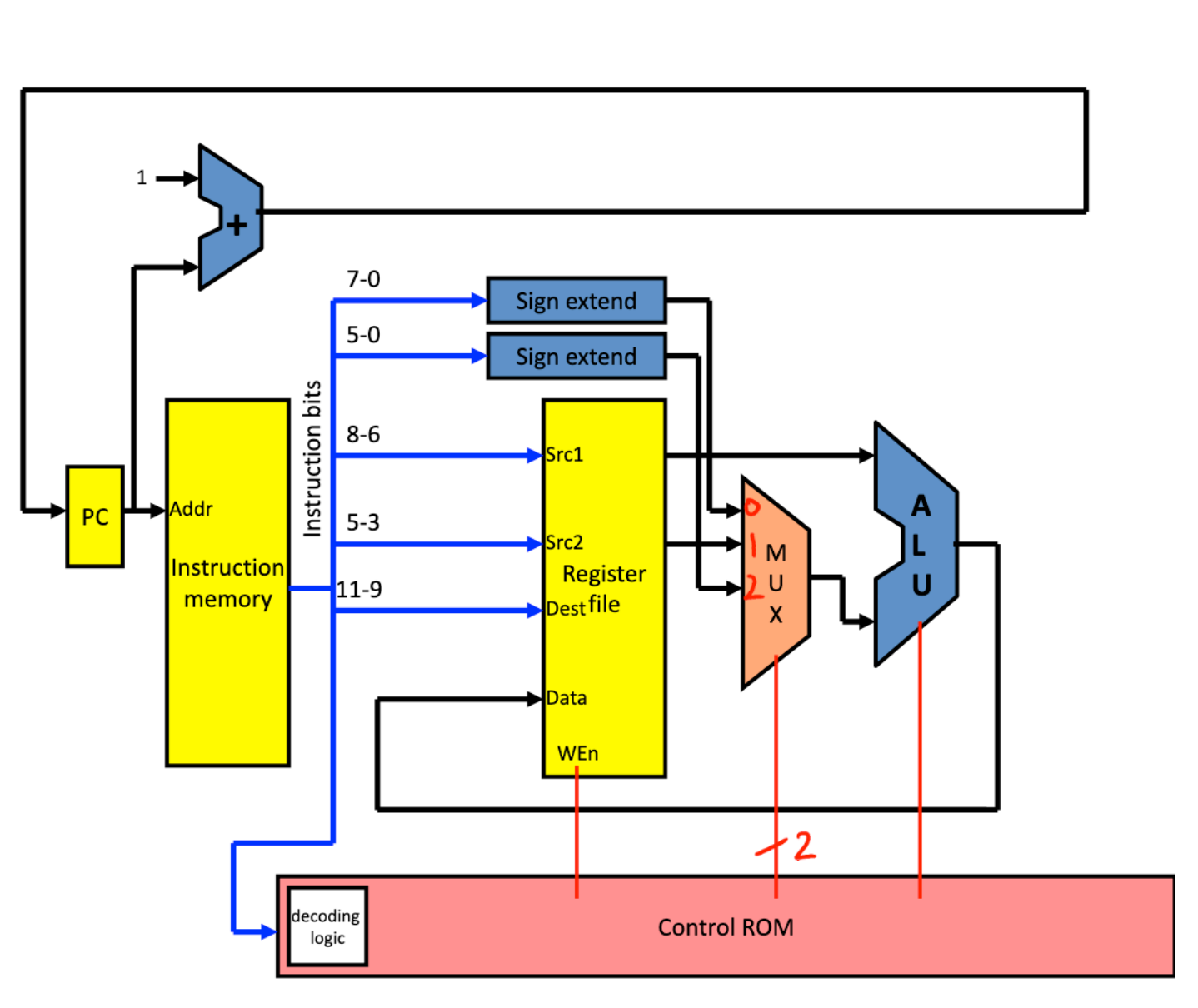
Adder: Increments PC by one word (since *ANNA* is word-Addressable)



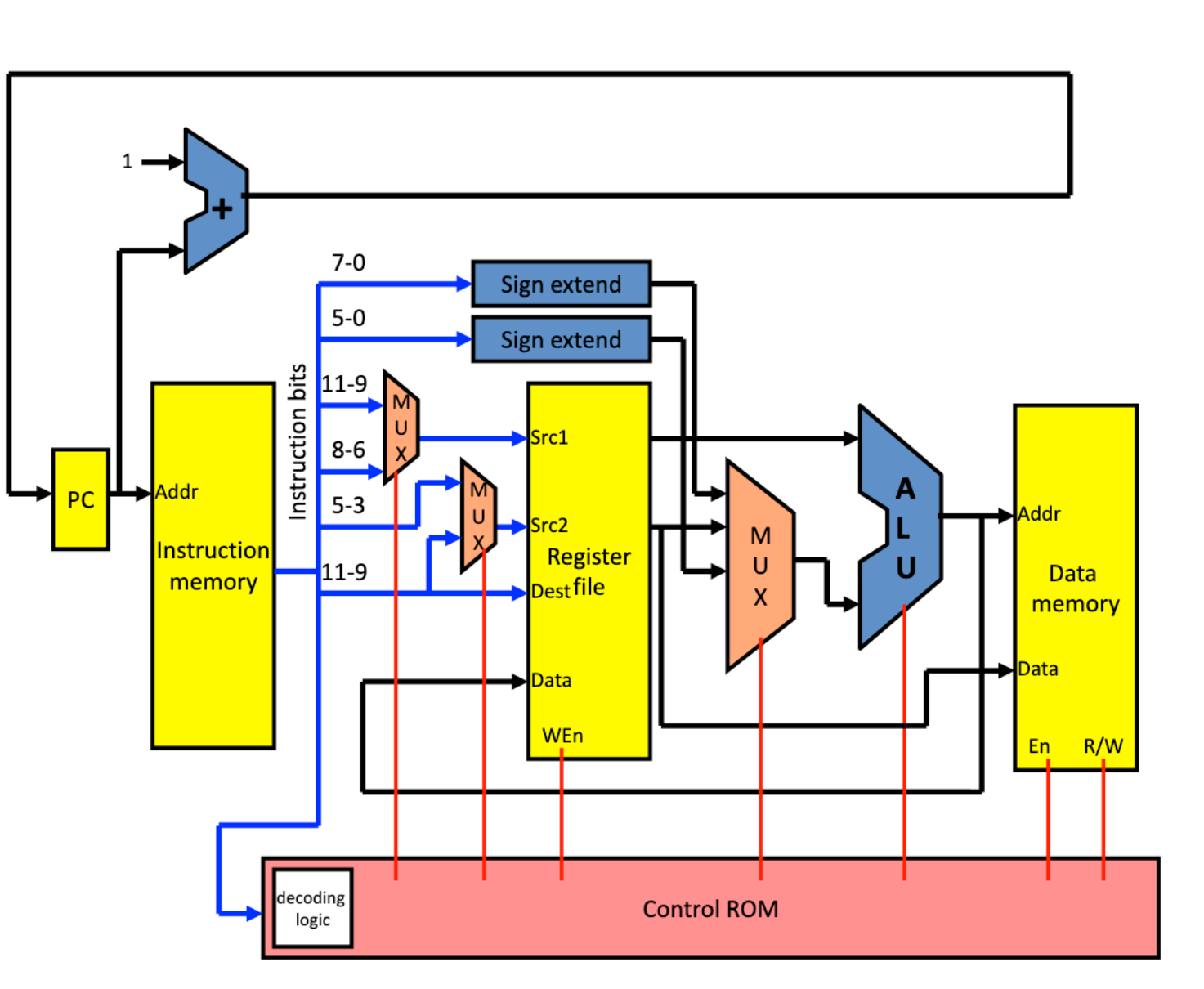
Sign Extension: When I6-type is used sign extension is used to turn that Imm6 -> Imm16.
Multiplexer: Control ROM will know if this is I6-type and will use sign extension output if this is the case. Then, will pass output into *ALU*.

When I6-type: passes in R_A and Imm6 to *ALU*.
When R-type: passes in R_A and R_B to *ALU*.

Note: This multiplexer strategy will be used in a similar fashion later on to help us handle different types.

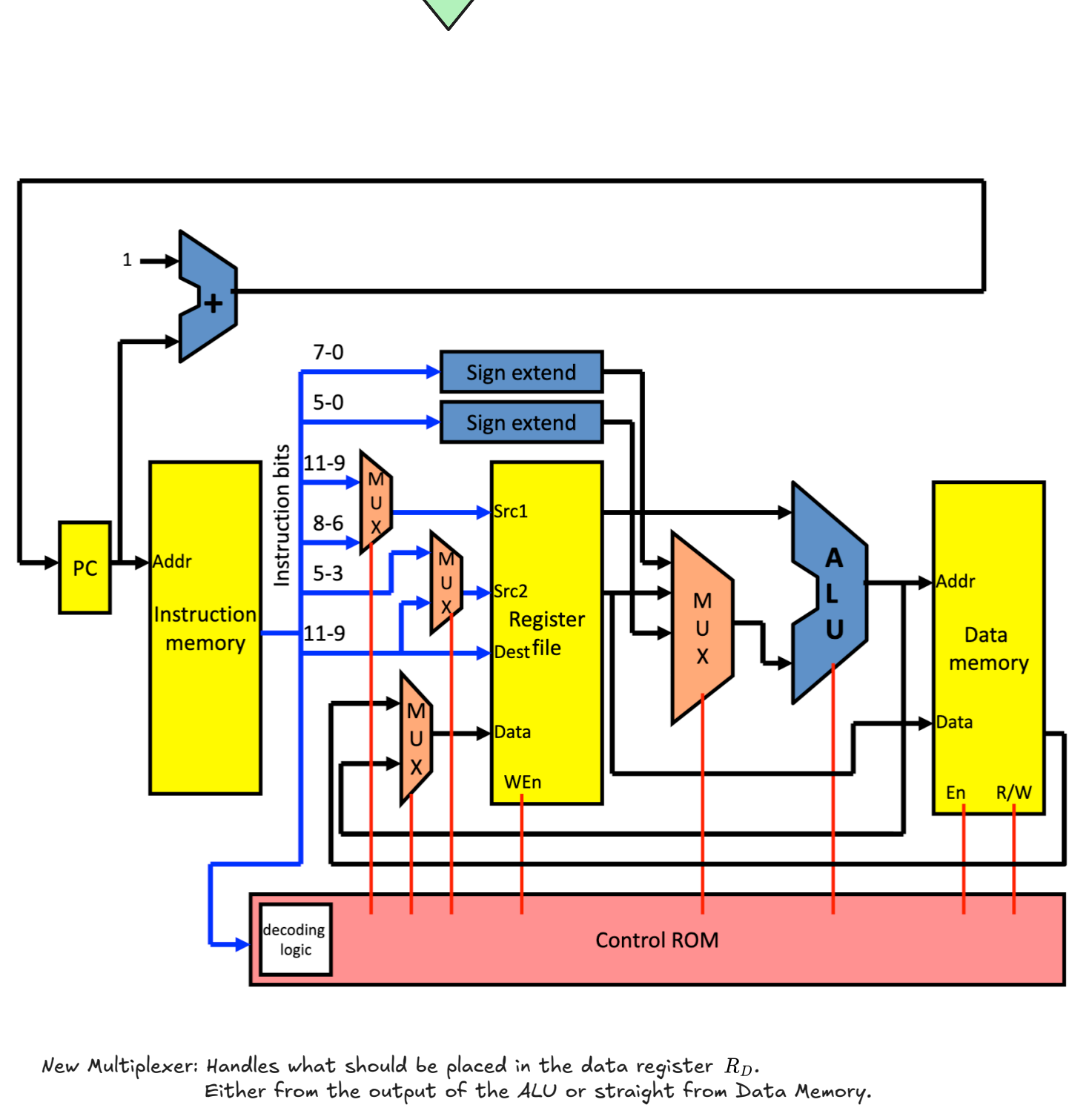


When I8 type:
- Sign Extend will extend Imm8 -> Imm16
- Multiplexer will pass this Imm16 into *ALU* (we will see that Src1 is changed in next part, since R_A is not used in I8-type).

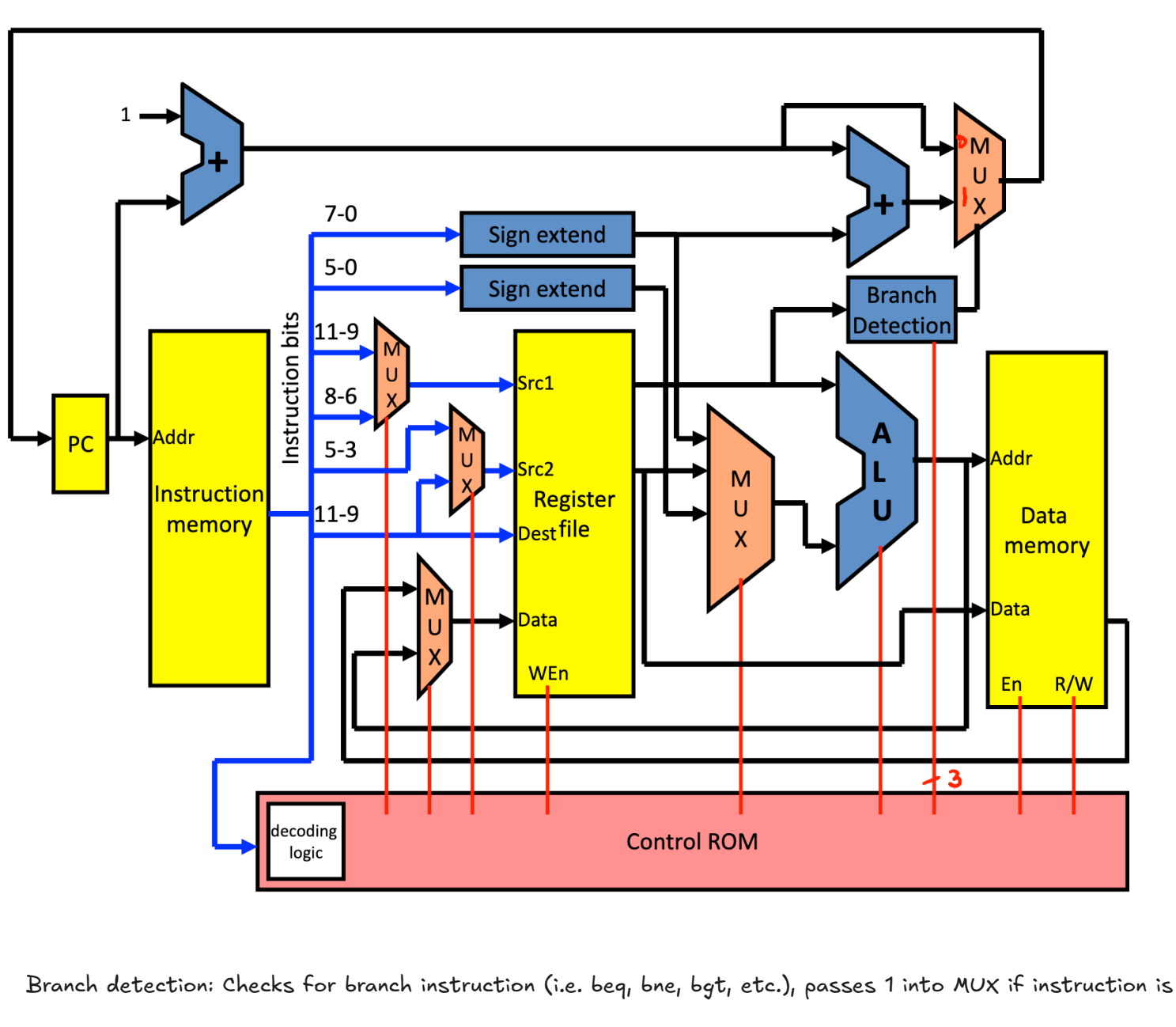


New Multiplexers:
- 11-9 & 8-6 MUX: Separates R-type & I6-type from I8-type (since I8-type does *NOT* use R_A).
- 11-9 & 5-3 MUX: Separates I6-type & I8-type from R-type (since only R-type uses R_B.

Data Memory: Used for lw (load word) and sw (store word) instructions (in *ANNA*), Address will be output of *ALU* with input R_A and Imm6. Data will be stored in R_D.
- Remember:
  - *ONLY* Enabled when lw is called



New Multiplexer: Handles what should be placed in the data register R_D.
Either from the output of the *ALU* or straight from Data Memory.



Branch detection: Checks for branch instruction (i.e. beq, bne, bgt, etc.), passes 1 into MUX if instruction is a branch.