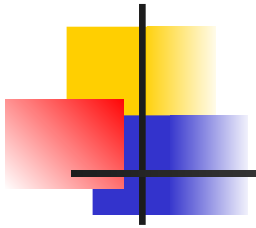
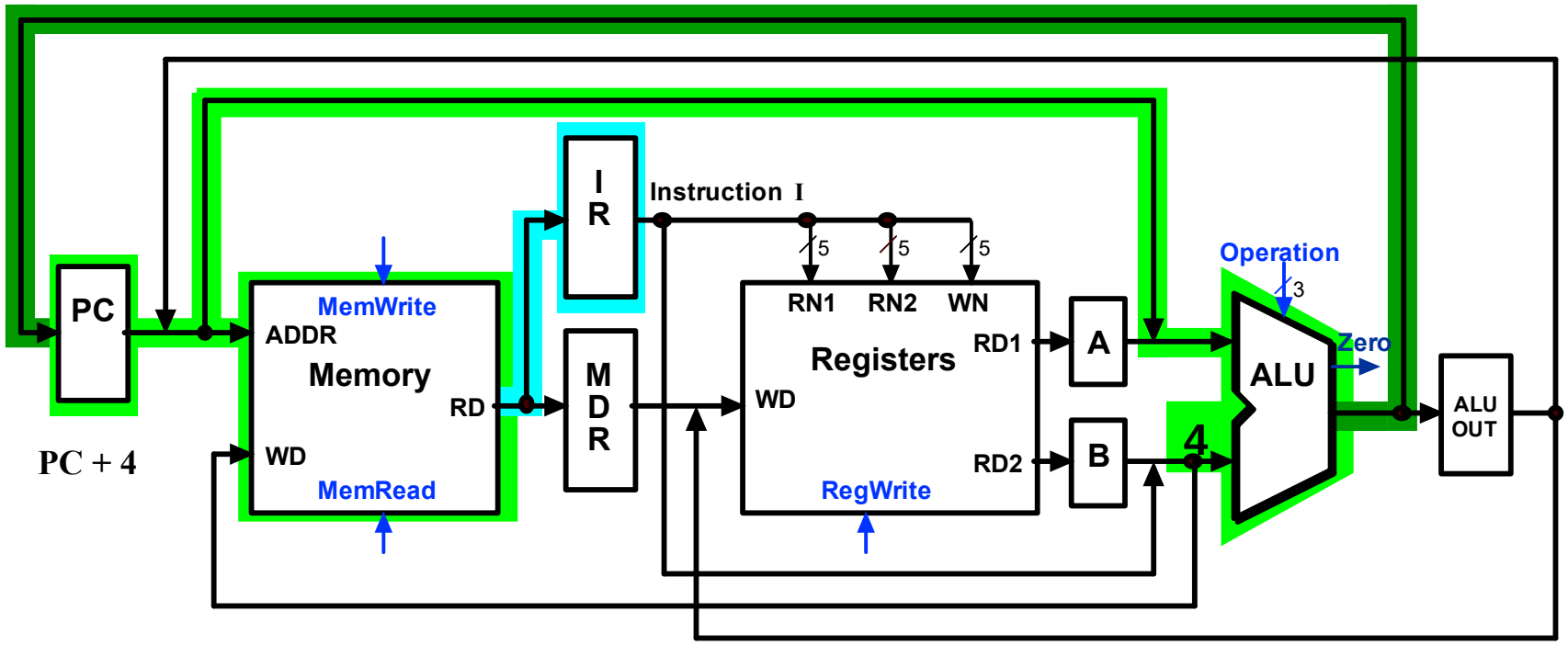


Multi cycle execution – animated view

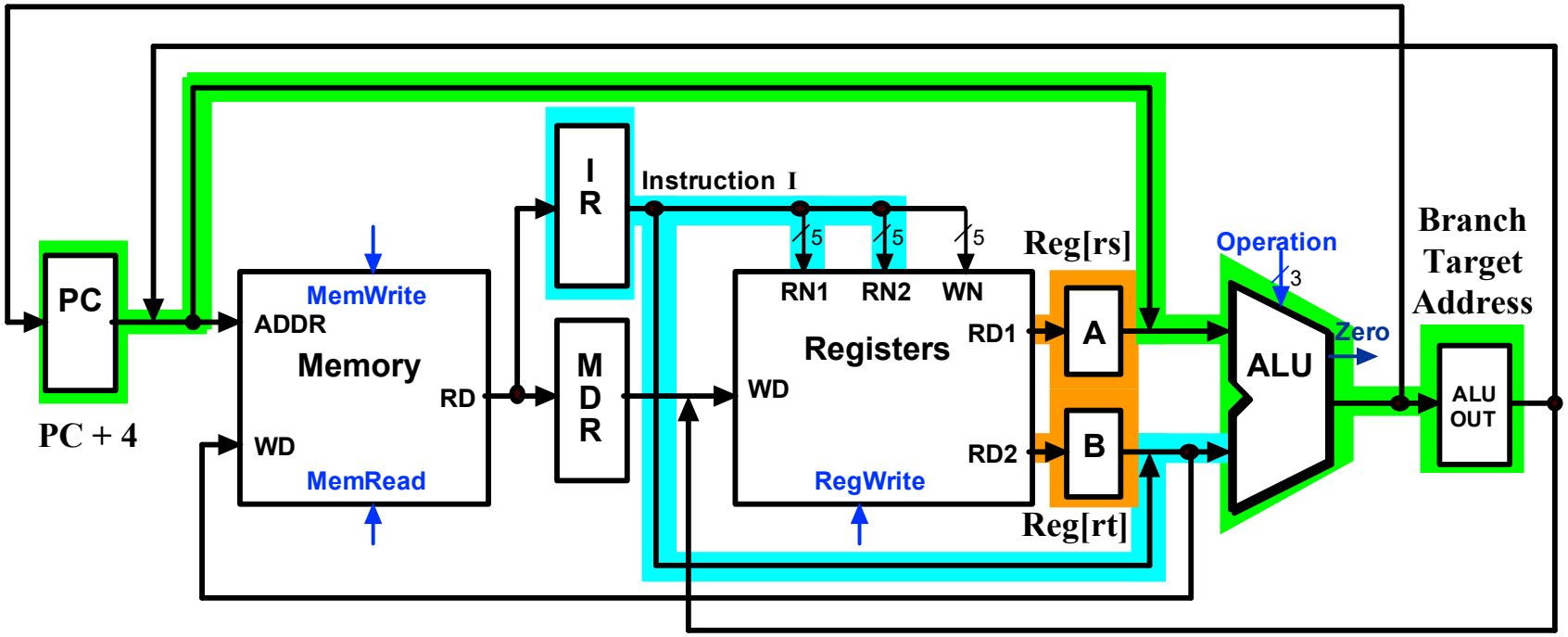
Multicycle Execution Step (1): Instruction Fetch

```
IR = Memory[PC];
PC = PC + 4;
```



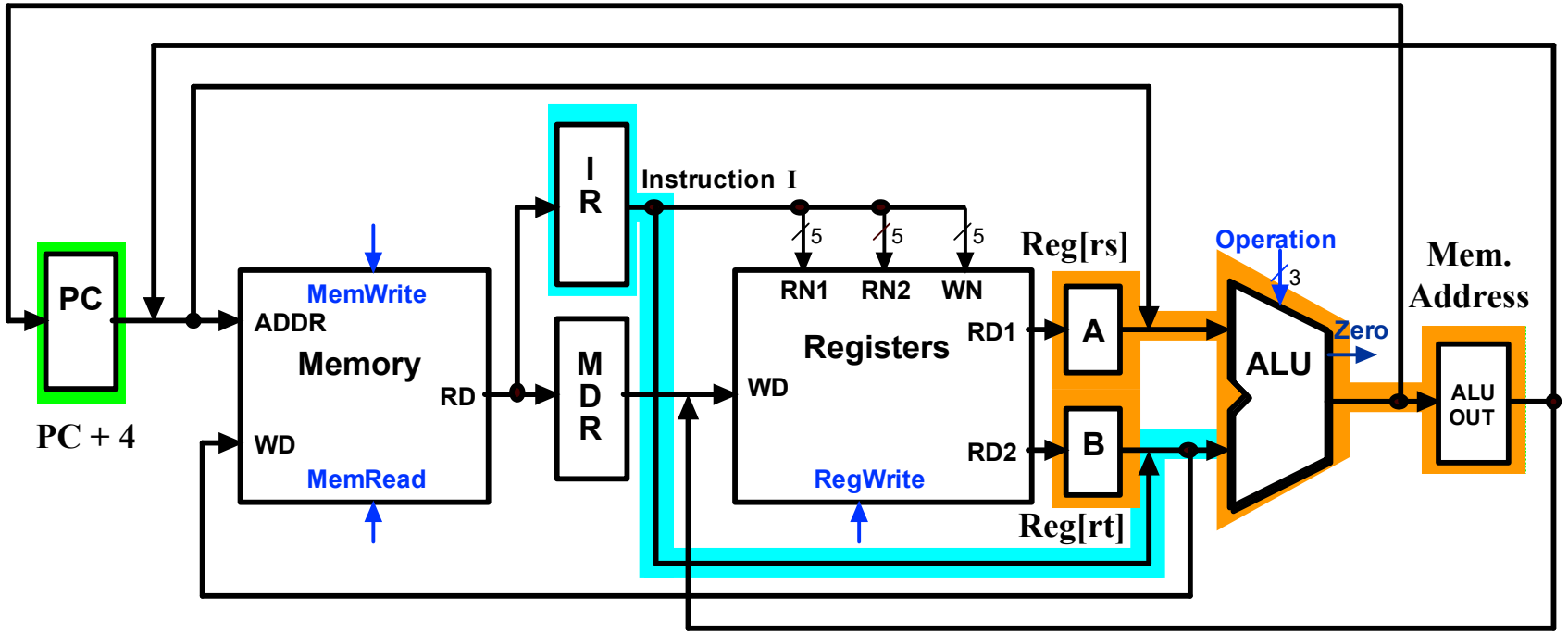
Multicycle Execution Step (2): Instruction Decode & Register Fetch

```
A = Reg[IR[25-21]];          (A = Reg[rs])
B = Reg[IR[20-15]];          (B = Reg[rt])
ALUOut = (PC + sign-extend(IR[15-0]) << 2)
```



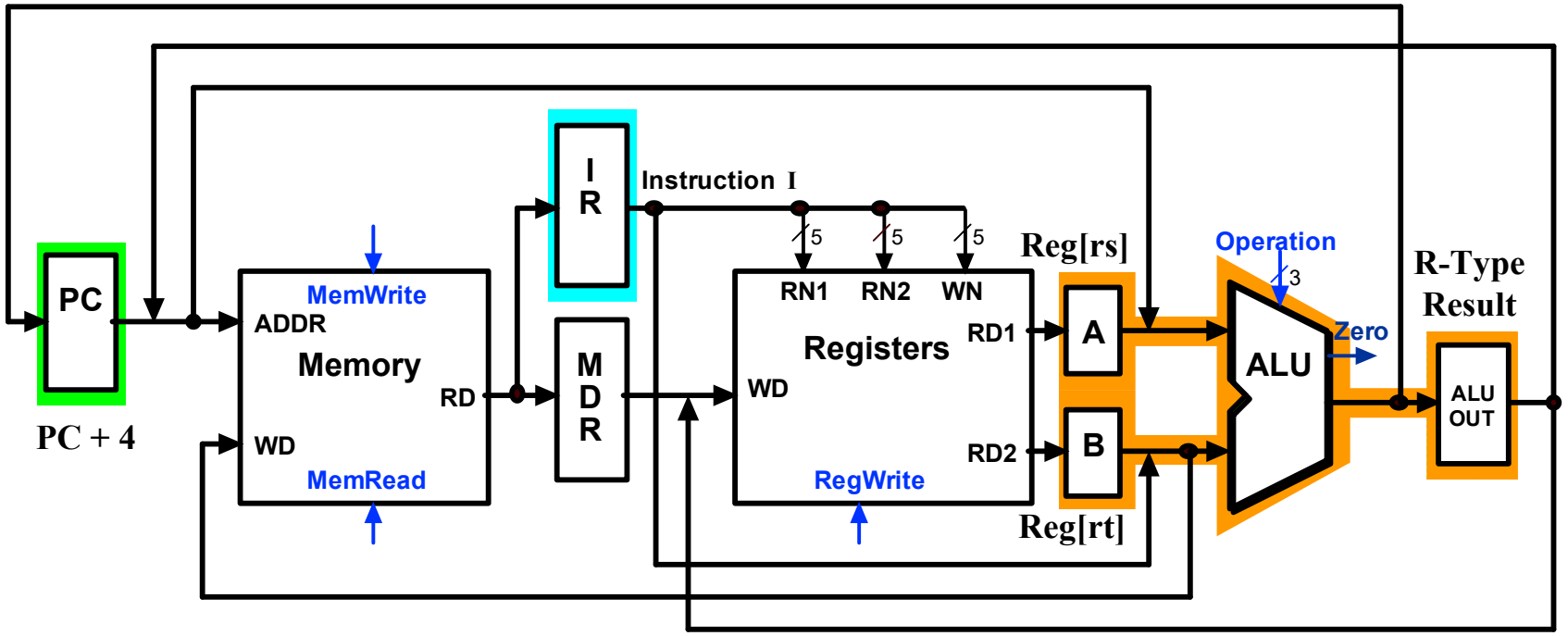
Multicycle Execution Step (3): Memory Reference Instructions

```
ALUOut = A + sign-extend(IR[15-0]);
```



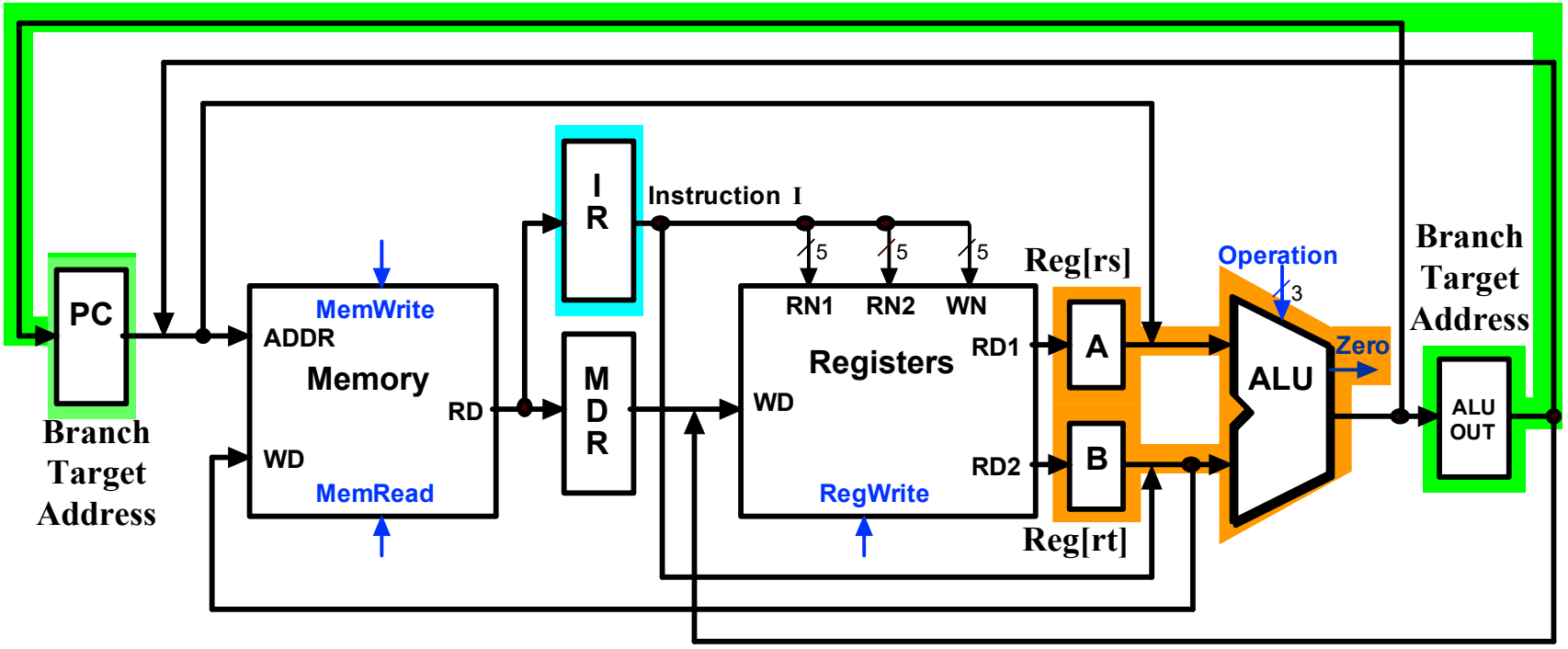
Multicycle Execution Step (3): ALU Instruction (R-Type)

```
ALUOut = A op B
```



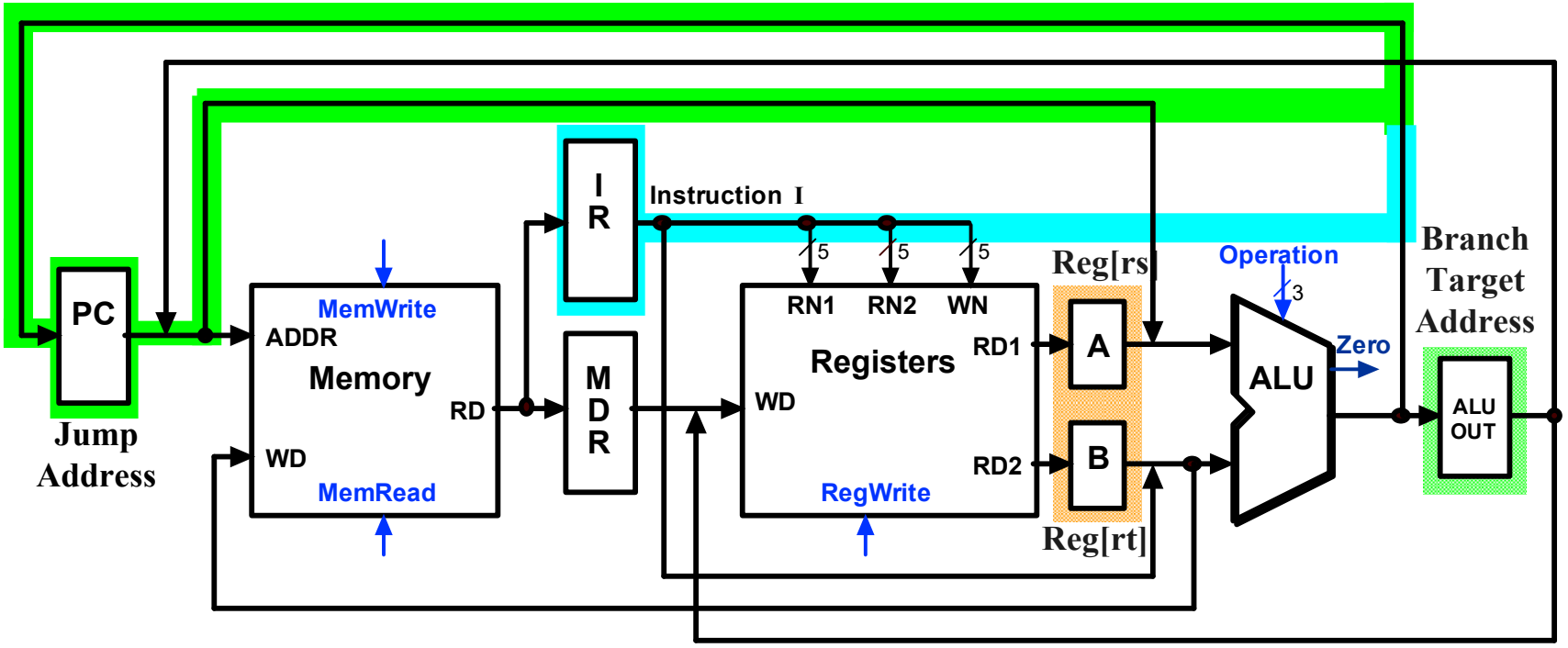
Multicycle Execution Step (3): Branch Instructions

```
if (A == B) PC = ALUOut;
```



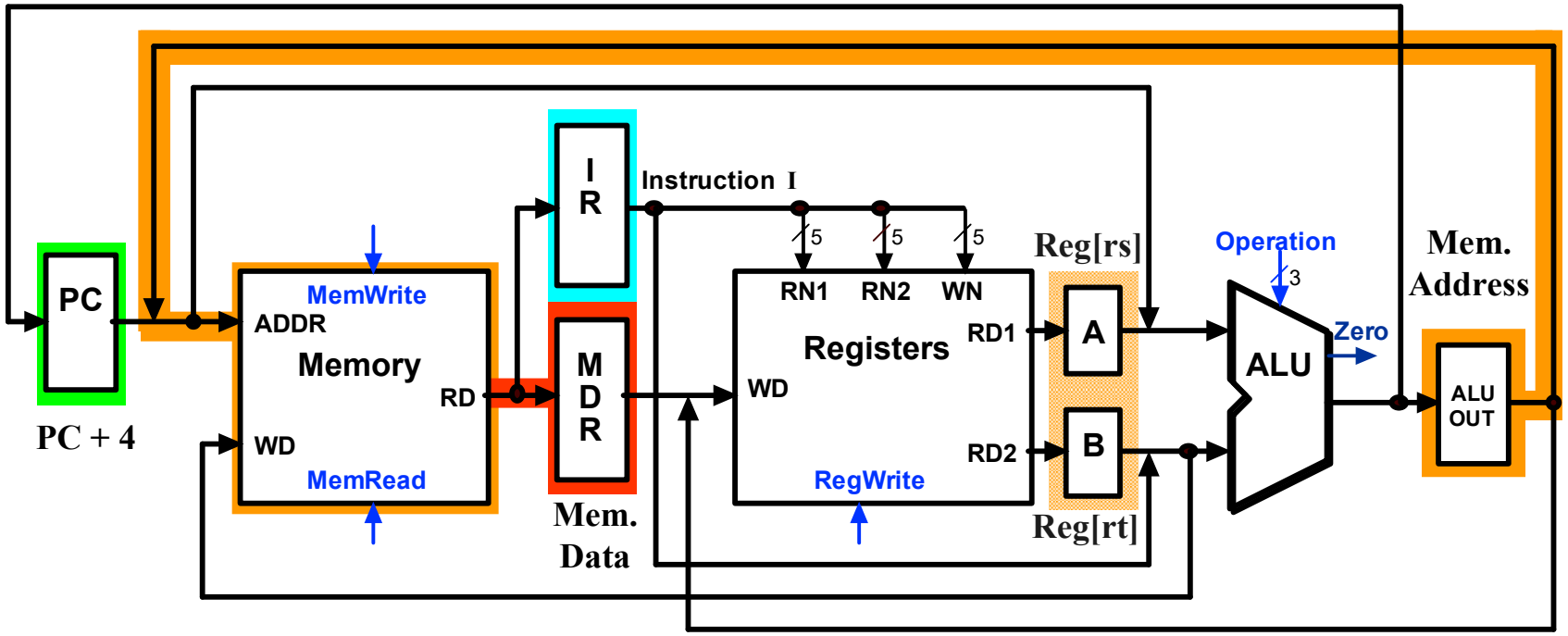
Multicycle Execution Step (3): Jump Instruction

```
PC = PC[31-28] concat (IR[25-0] << 2)
```



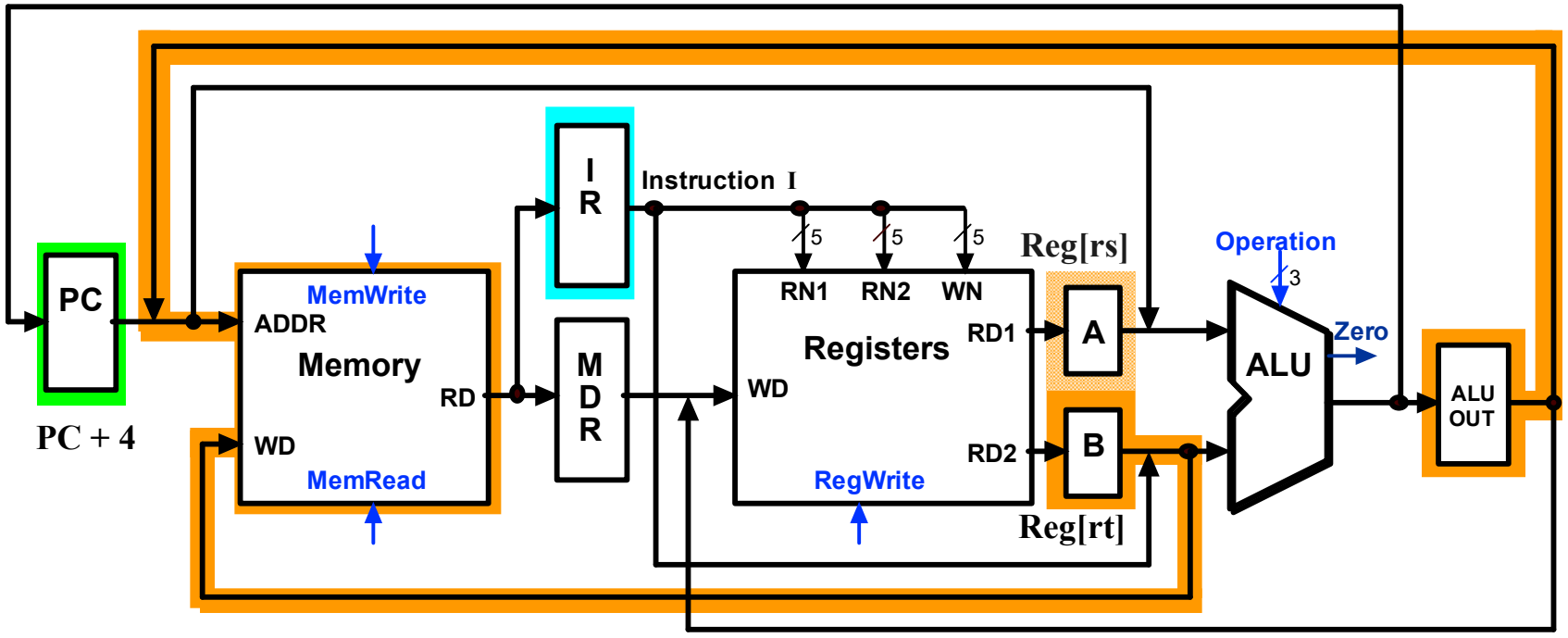
Multicycle Execution Step (4): Memory Access - Read (`lw`)

```
MDR = Memory[ALUOut];
```



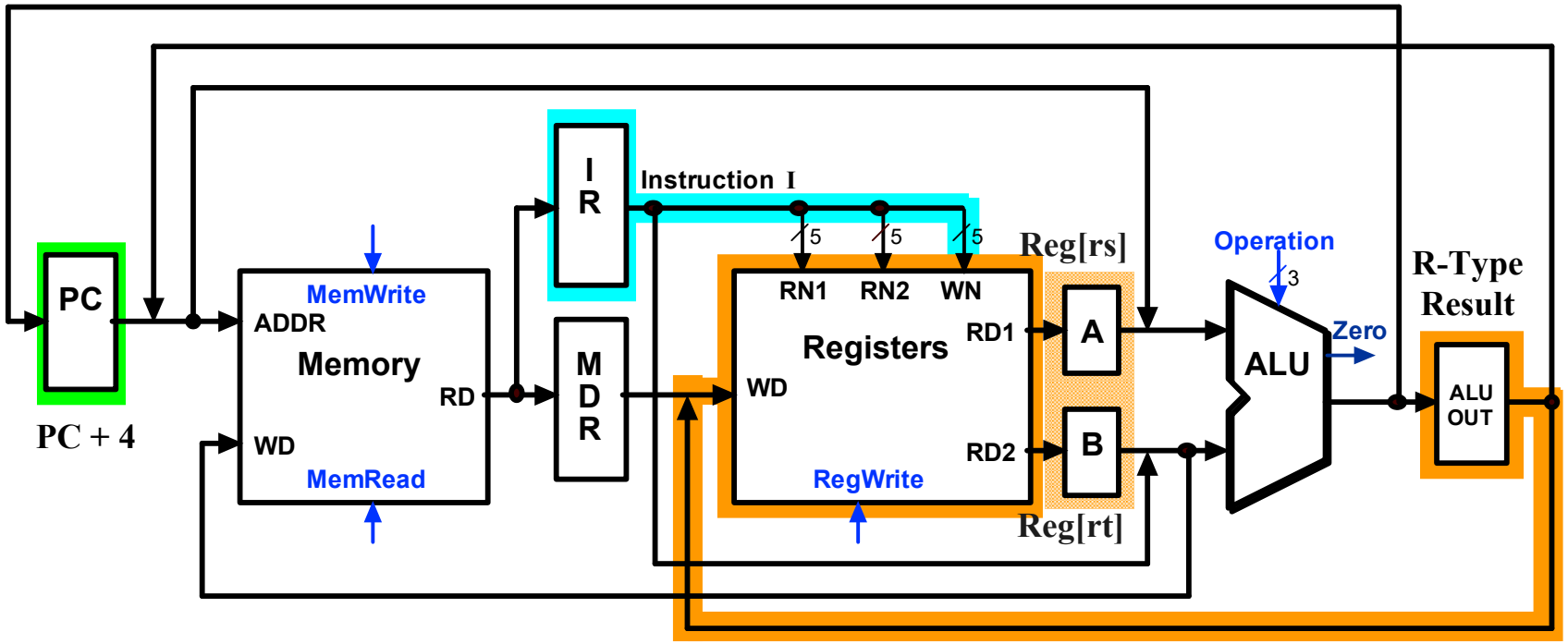
Multicycle Execution Step (4): Memory Access - Write (sw)

```
Memory[ALUOut] = B;
```



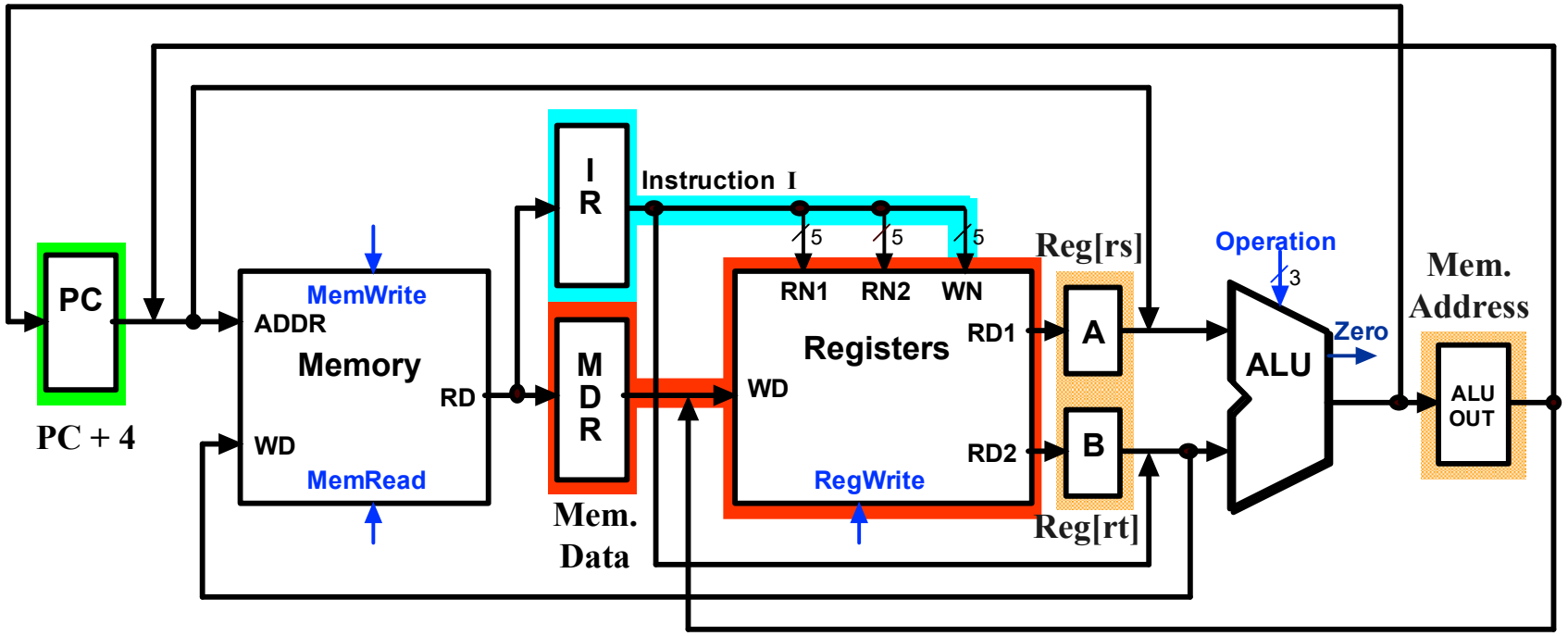
Multicycle Execution Step (4): ALU Instruction (R-Type)

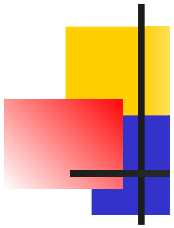
```
Reg[IR[15:11]] = ALUOUT
```



Multicycle Execution Step (5): Memory Read Completion (lw)

```
Reg[IR[20-16]] = MDR;
```





Instruction execution review

- Executing a MIPS instruction can take up to five steps.

Step	Name	Description
Instruction Fetch	IF	Read an instruction from memory.
Instruction Decode	ID	Read source registers and generate control signals.
Execute	EX	Compute an R-type result or a branch outcome.
Memory	MEM	Read or write the data memory.
Writeback	WB	Store a result in the destination register.

- However, as we saw, not all instructions need all five steps.

Instruction	Steps required				
beq	IF	ID	EX		
R-type	IF	ID	EX		WB
sw	IF	ID	EX	MEM	
lw	IF	ID	EX	MEM	WB

Break data path into 5 stages