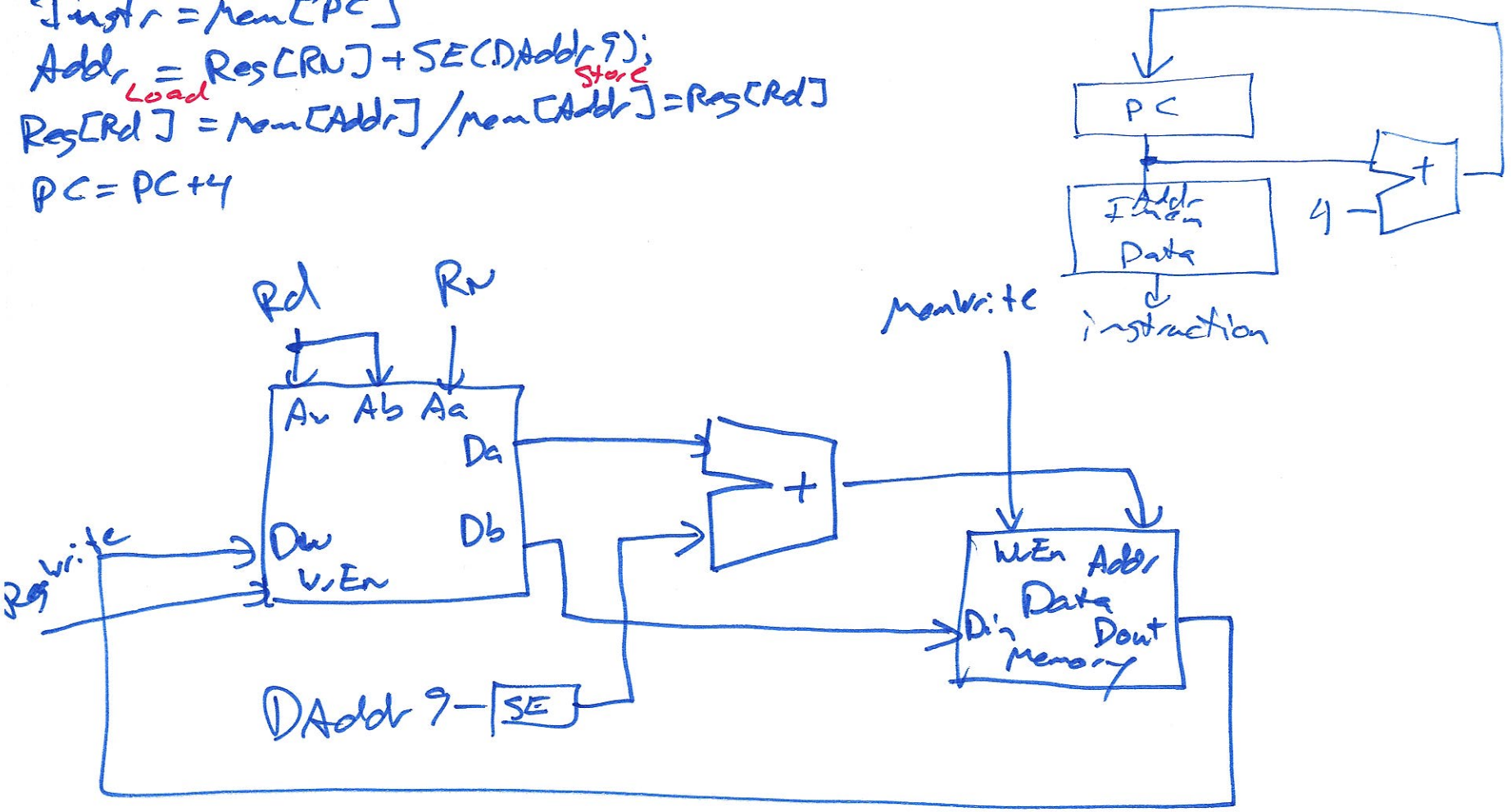
Review Problem 22

- Develop a single-cycle CPU that can do LDUR and STUR (only). Make it as simple as possible

Instr = mem[PC]
Addr = Reg[Rn] + SE(DAddr 9); (Load / Store)
Reg[Rd] = mem[Addr] / mem[Addr] = Reg[Rd]
PC = PC + 4

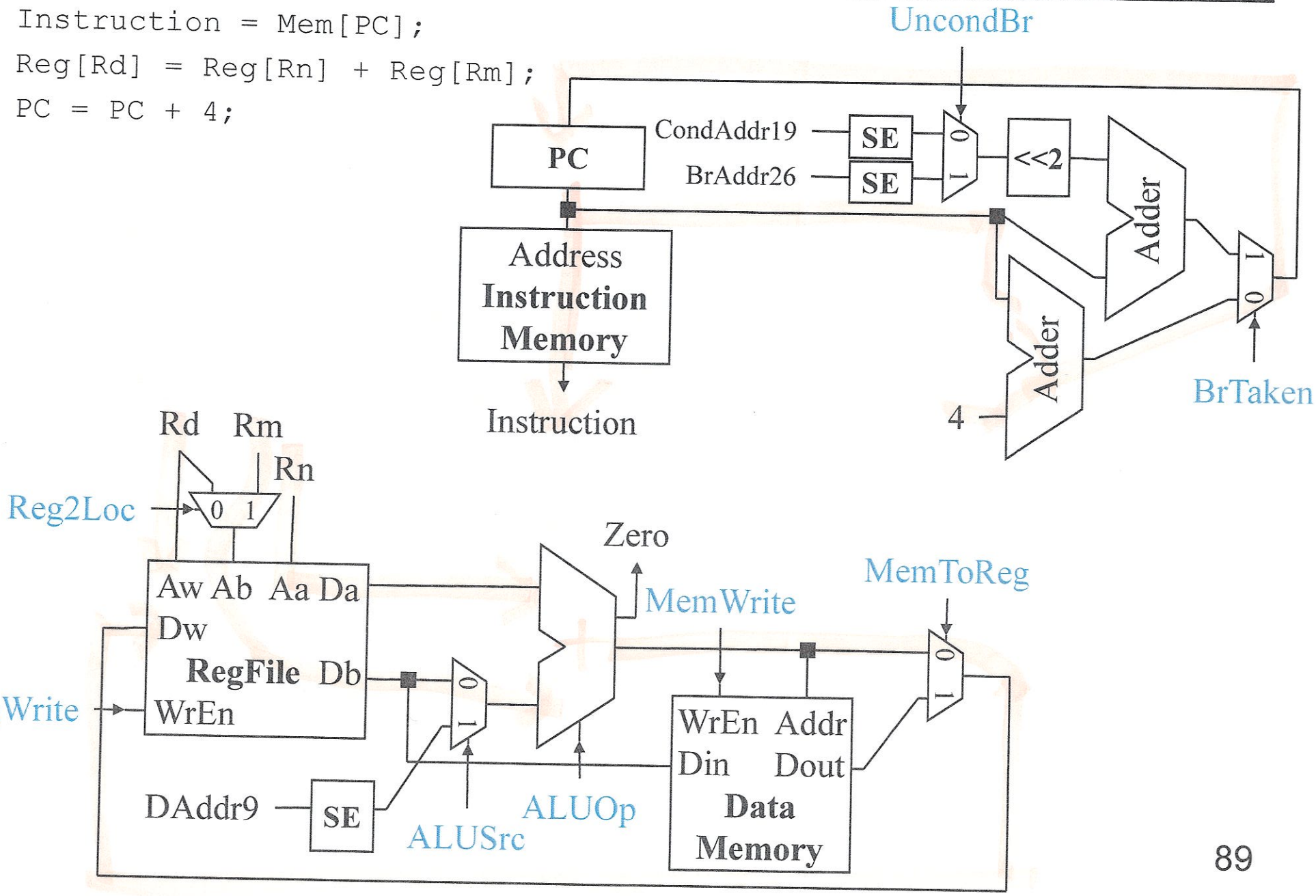


Control Signals

Opcode[31:26] Opcode[25:21]	100010 11000	110010 11000	111110 00010	111110 00000	000101 XXXXX	101101 00XXX
	ADD	SUB	LDUR	STUR	B	CBZ
Reg 2Loc	1	1	X	0	X	0
ALUSrc	0	0	1	1	X	0
MemToReg	0	0	1	X	X	X
RegWrite	1	1	1	0	0	0
MemWrite	0	0	0	1	0	0
Br Taken	0	0	0	0	1	(zero)
Uncond Br	X	X	X	X	1	0
ALU Op	+	−	+	+	X	Pass B

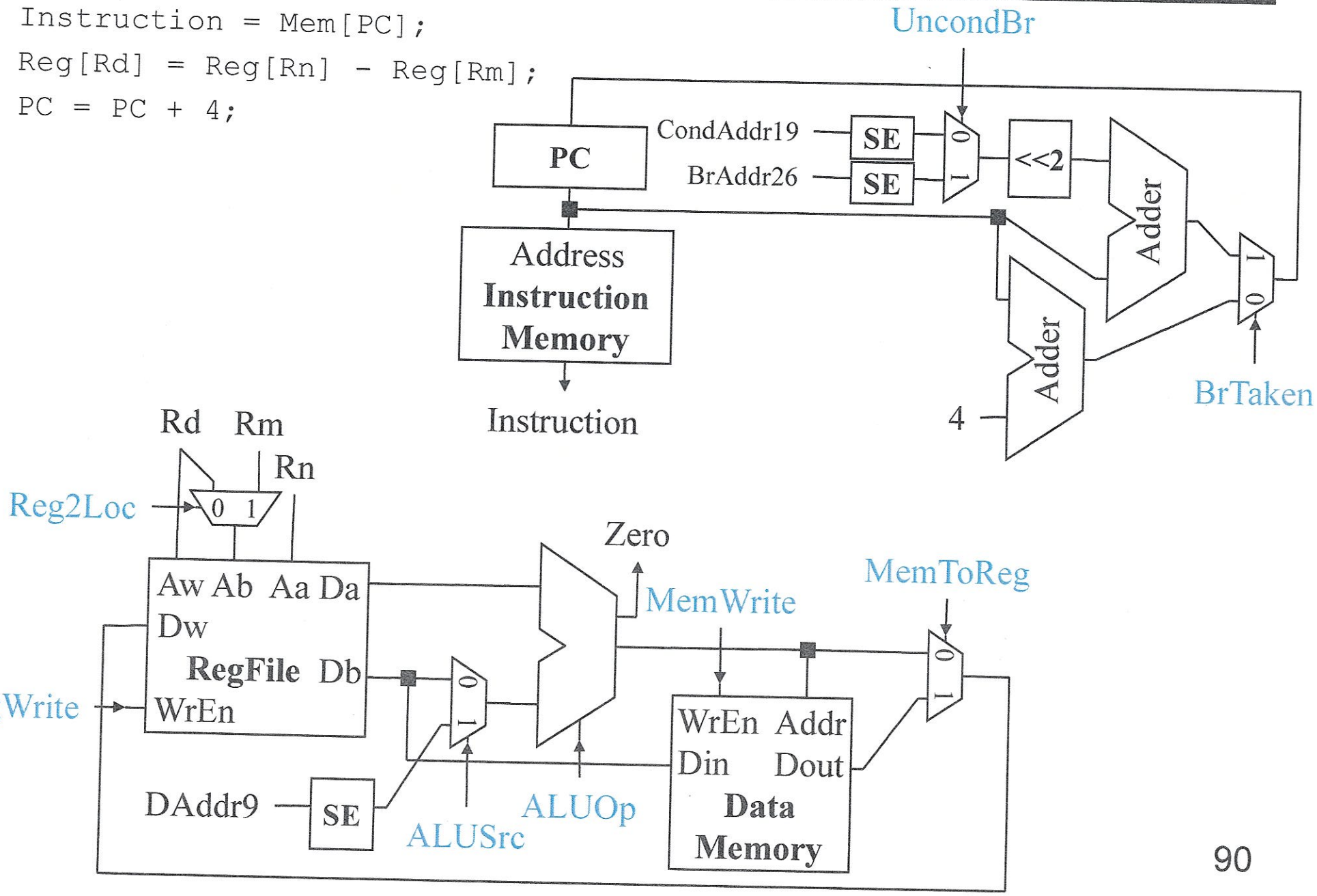
ADD Control

```
Instruction = Mem[PC];
Reg[Rd] = Reg[Rn] + Reg[Rm];
PC = PC + 4;
```



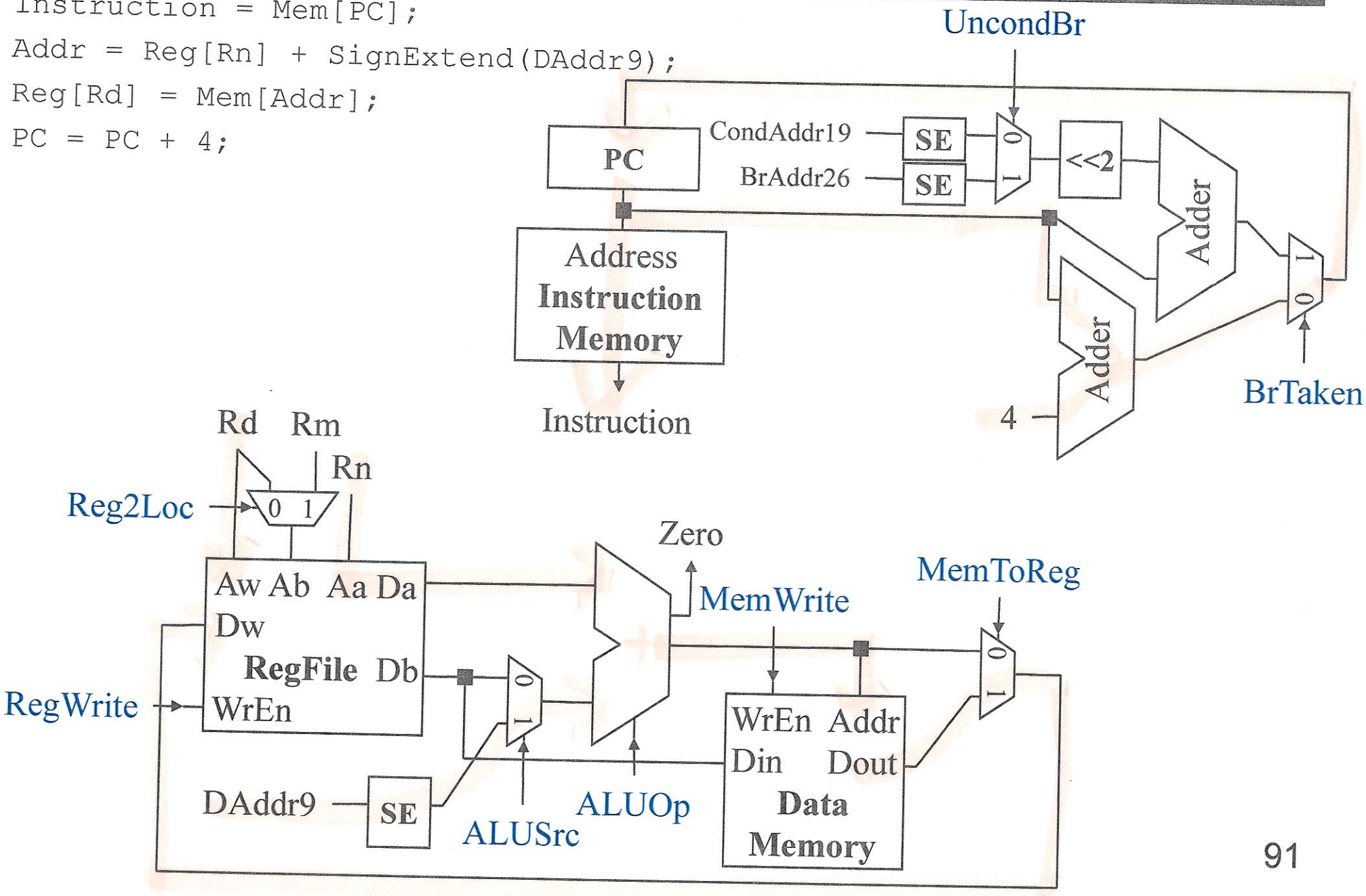
SUB Control

```
Instruction = Mem[PC];
Reg[Rd] = Reg[Rn] - Reg[Rm];
PC = PC + 4;
```



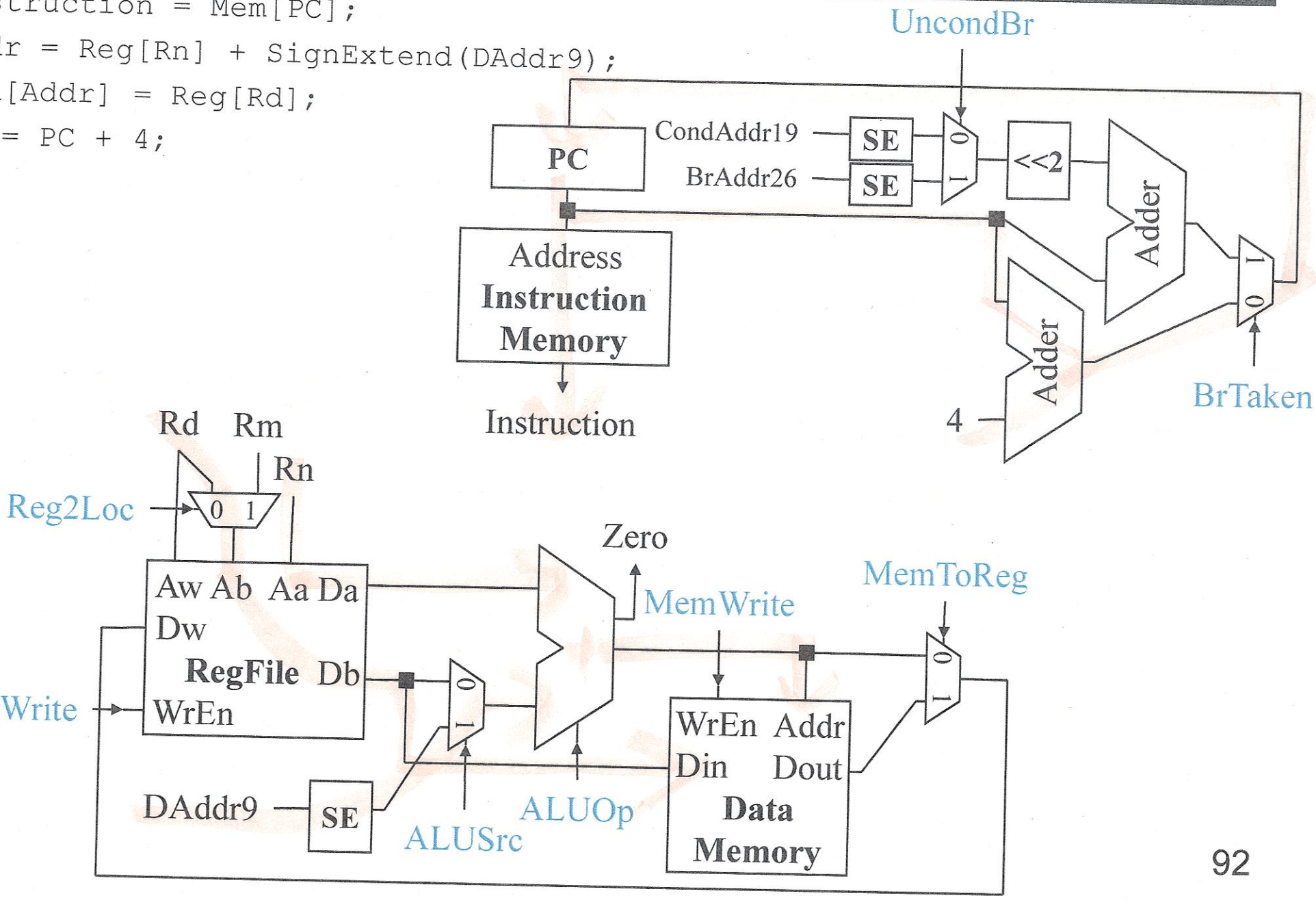
LDUR Control

```
Instruction = Mem[PC];
Addr = Reg[Rn] + SignExtend(DAddr9);
Reg[Rd] = Mem[Addr];
PC = PC + 4;
```



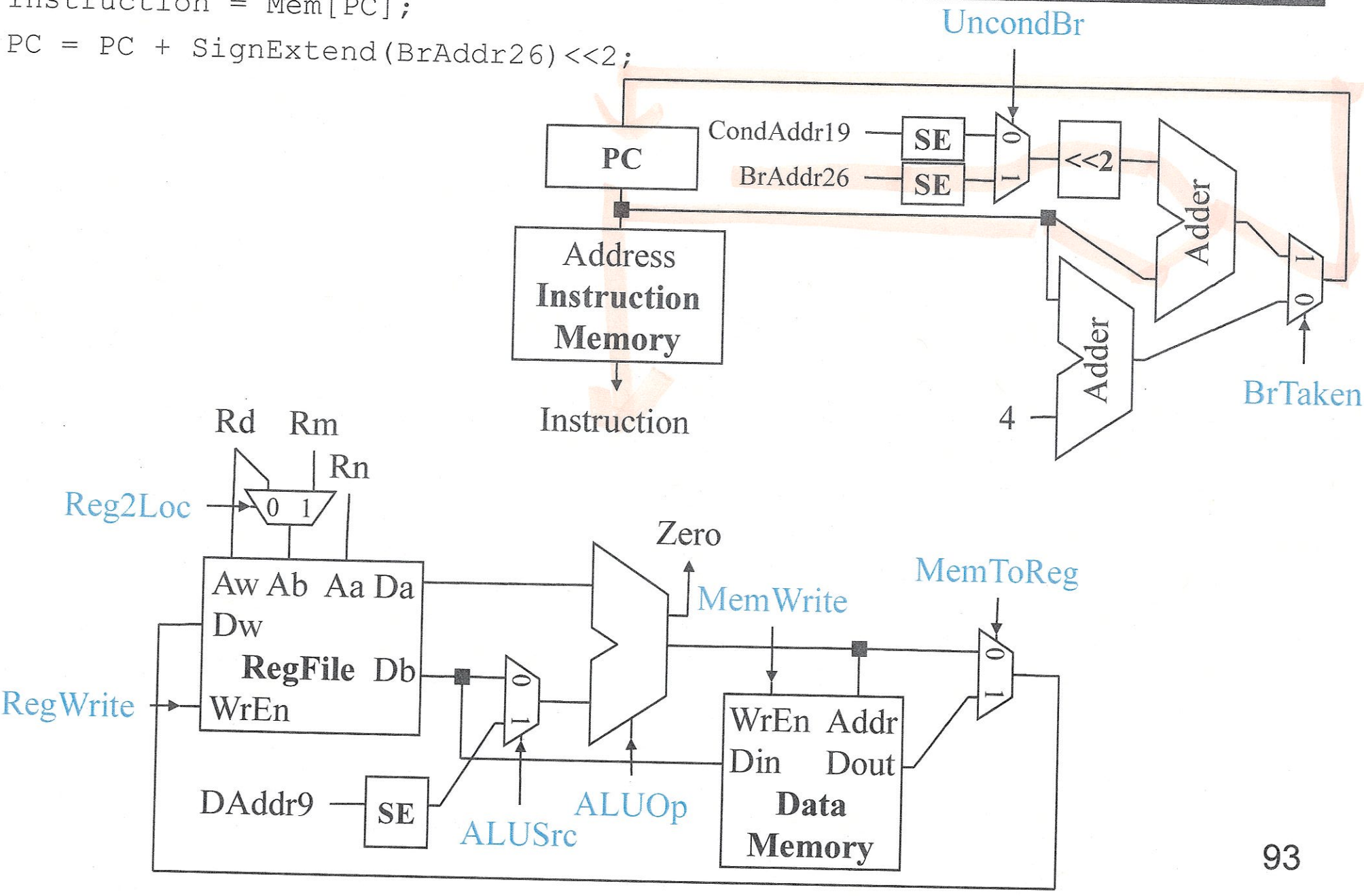
STUR Control

```
Instruction = Mem[PC];
Addr = Reg[Rn] + SignExtend(DAddr9);
Mem[Addr] = Reg[Rd];
PC = PC + 4;
```



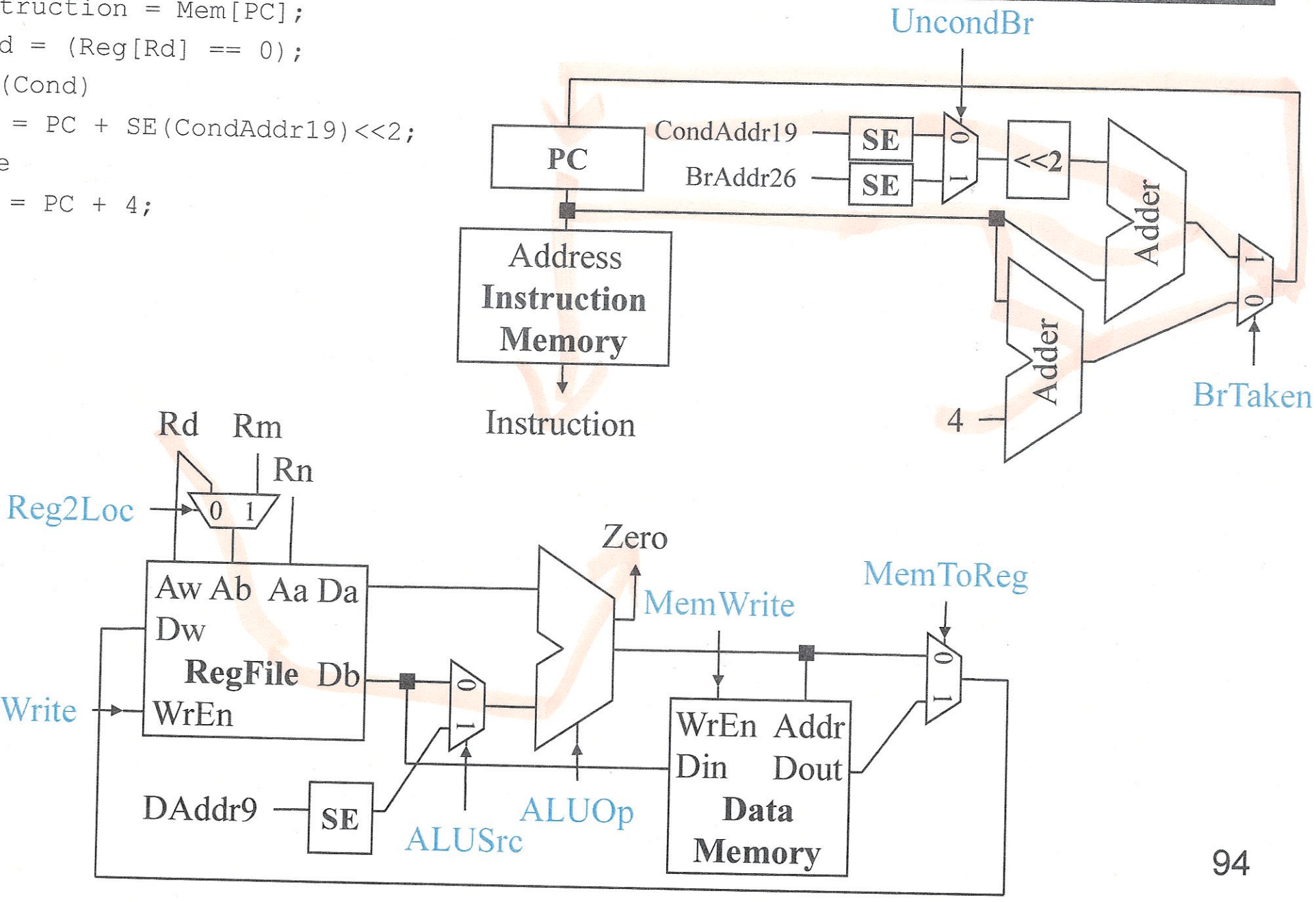
B Control

```
Instruction = Mem[PC];
PC = PC + SignExtend(BrAddr26)<<2;
```



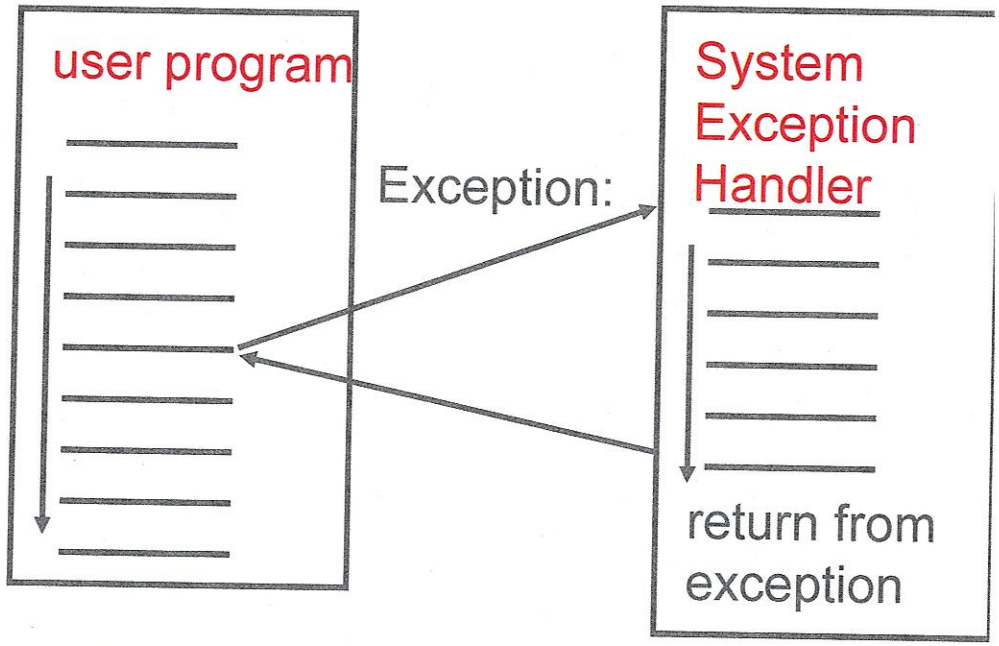
CBZ Control

```
Instruction = Mem[PC];
Cond = (Reg[Rd] == 0);
if (Cond)
 PC = PC + SE(CondAddr19)<<2;
else
 PC = PC + 4;
```



Advanced: Exceptions

Exception = unusual event in processor
- Arithmetic overflow, divide by zero, …
- Call an undefined instruction
- Hardware failure
- I/O device request (called an "interrupt")



Approaches
- Make software test for exceptional events when they may occur ("polling")
- Have hardware detect these events & react:
  - Save state (Exception Program Counter, protect the GPRs, note cause)
  - Call Operating System
    - If (undef_instr) PC = C0000000
    - If (overflow) PC = C0000020
    - If (I/O) PC = C0000040
    - …