Microarchitecture

CH007

Figure 7.1 State elements of ARM processor

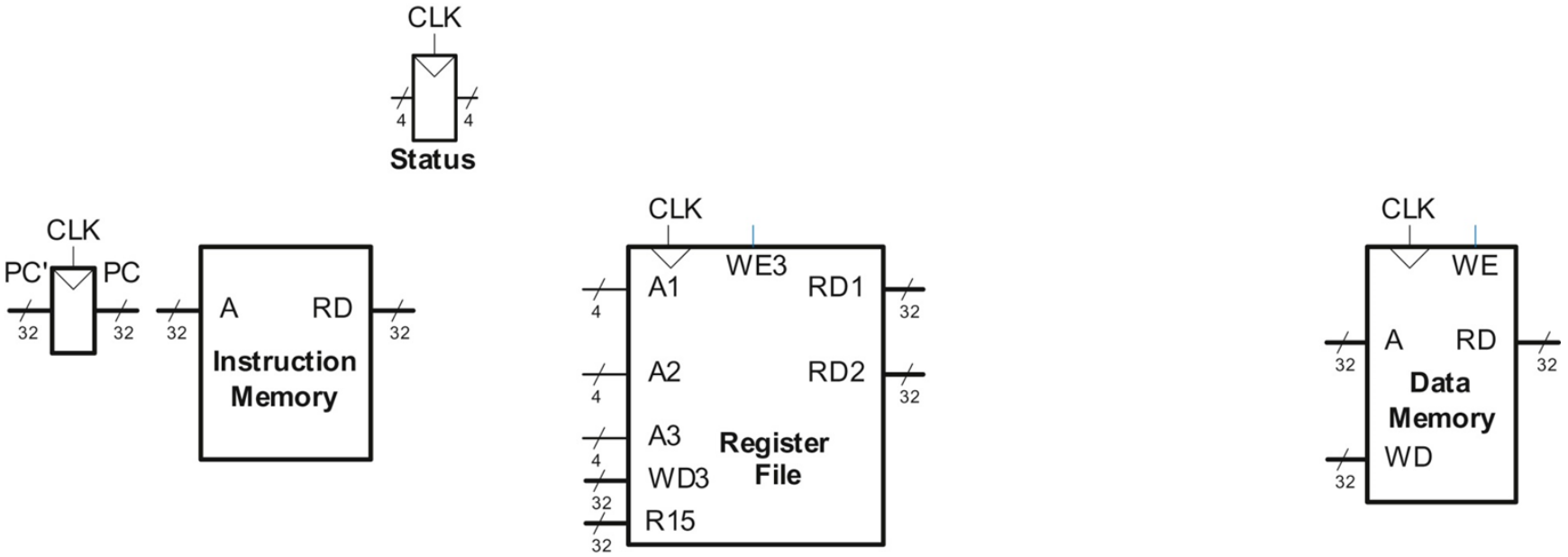


Figure 7.2 Fetch instruction from memory

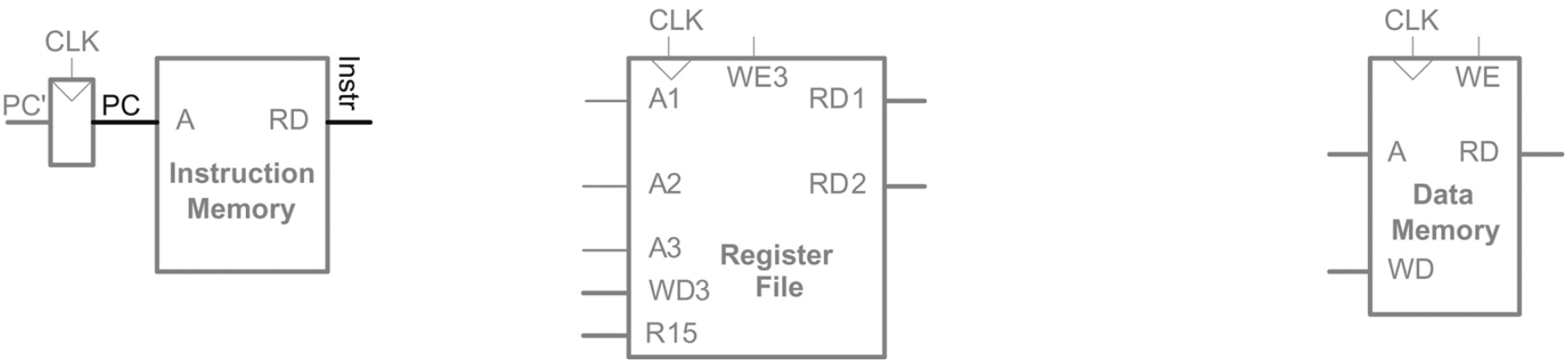


Figure 7.3 Read source operand from register file

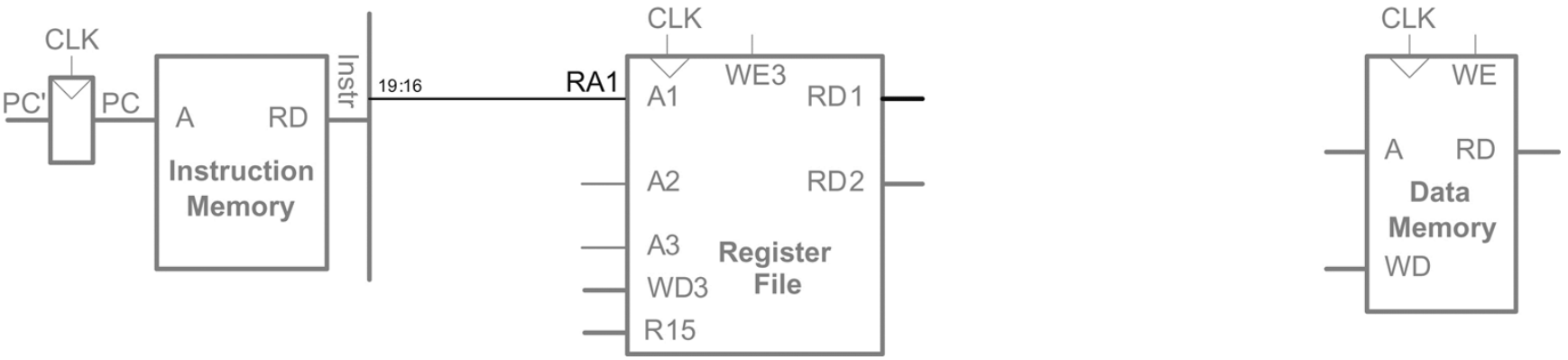


Figure 7.4 Zero-extend the immediate

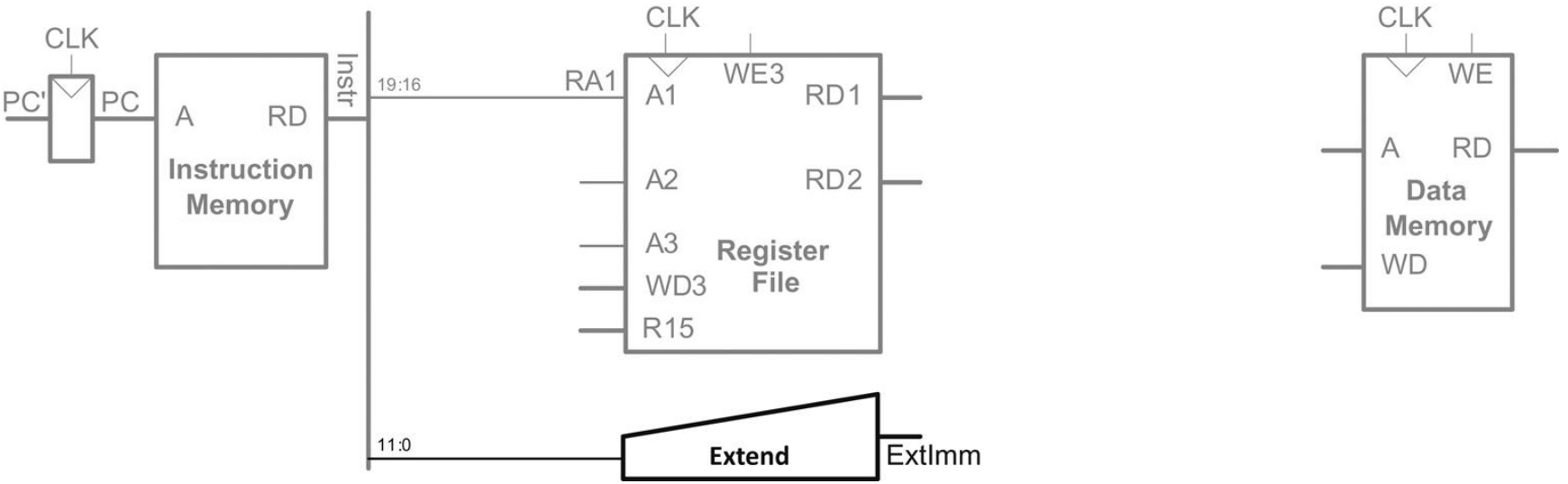


Figure 7.5 Compute memory address

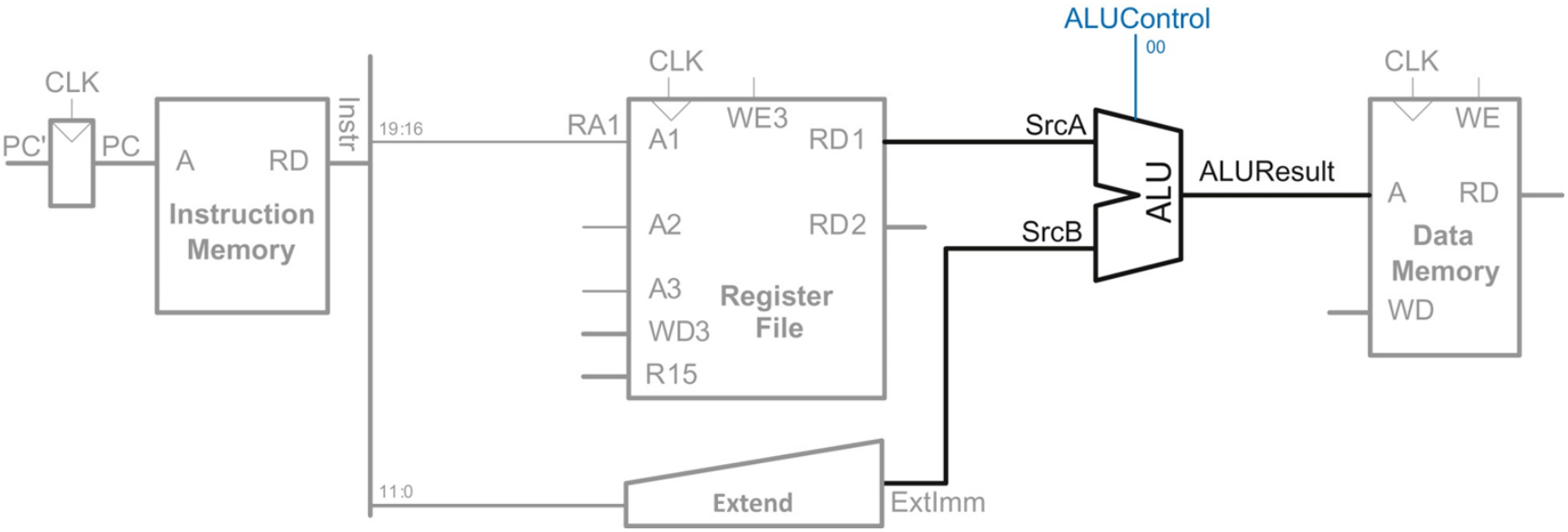


Figure 7.6 Write data back to register file

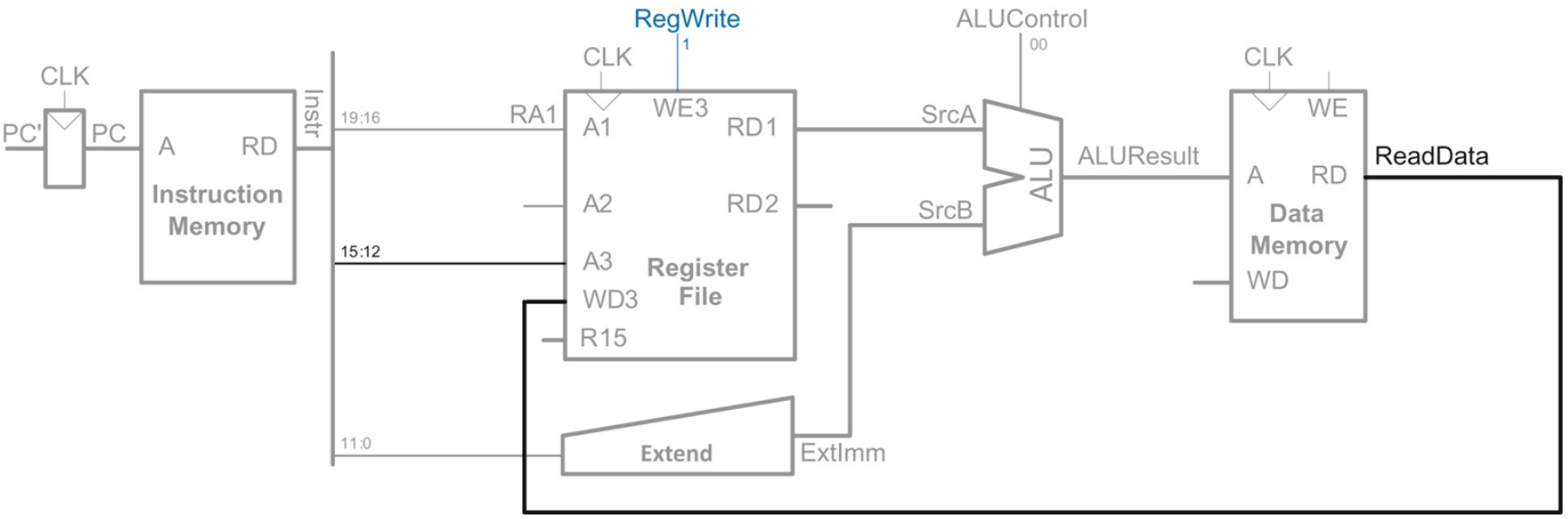


Figure 7.7 Increment program counter

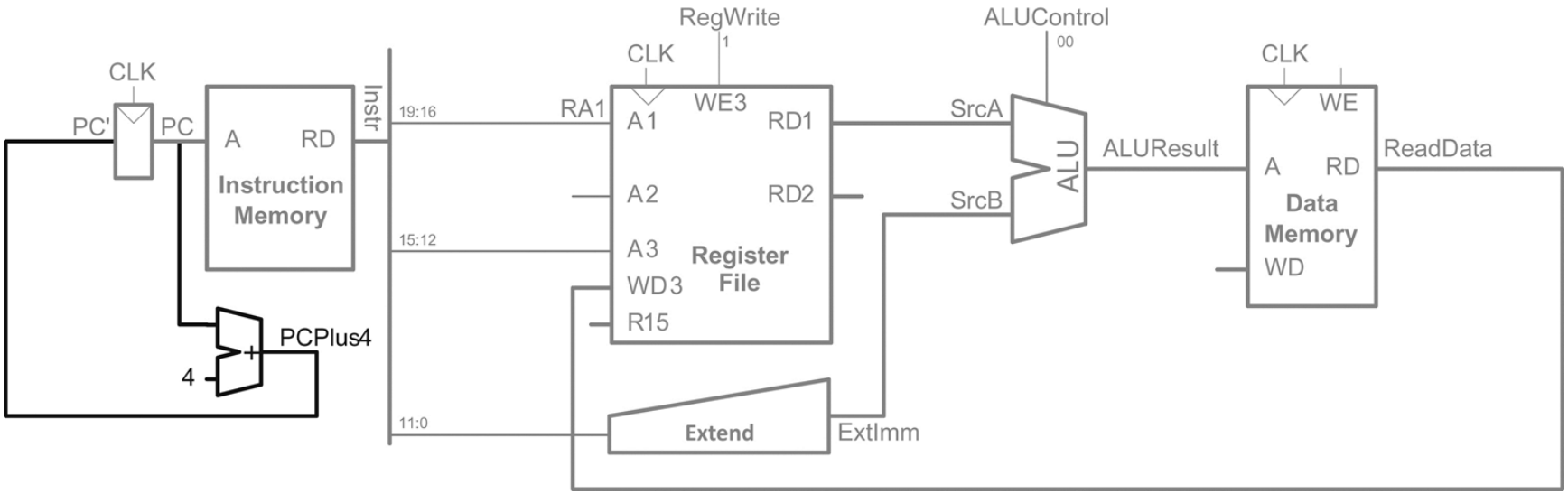


Figure 7.8 Read or write program counter as R15

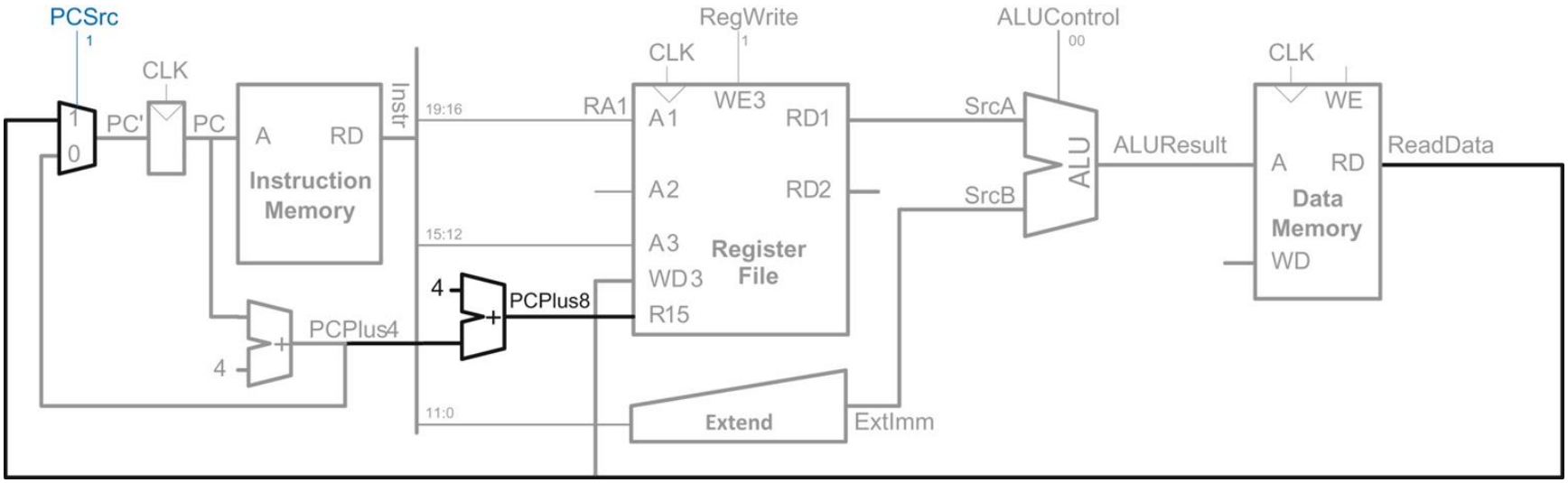


Figure 7.9 Write data to memory for STR instruction

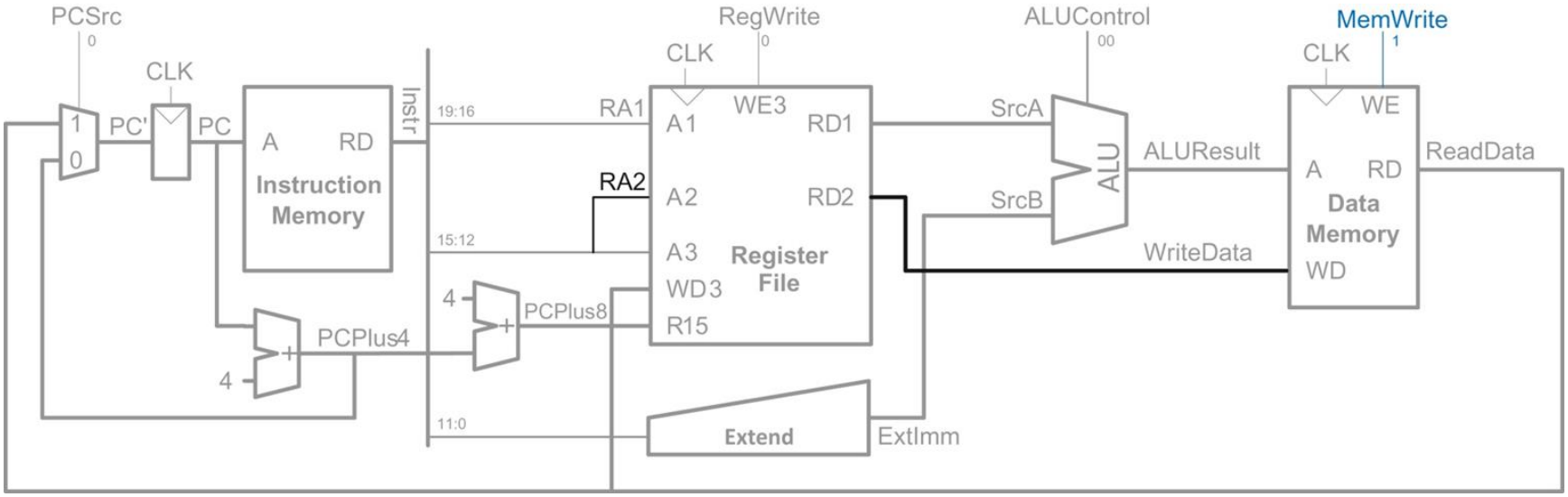


Figure 7.10 Datapath enhancements for data-processing instructions with immediate addressing

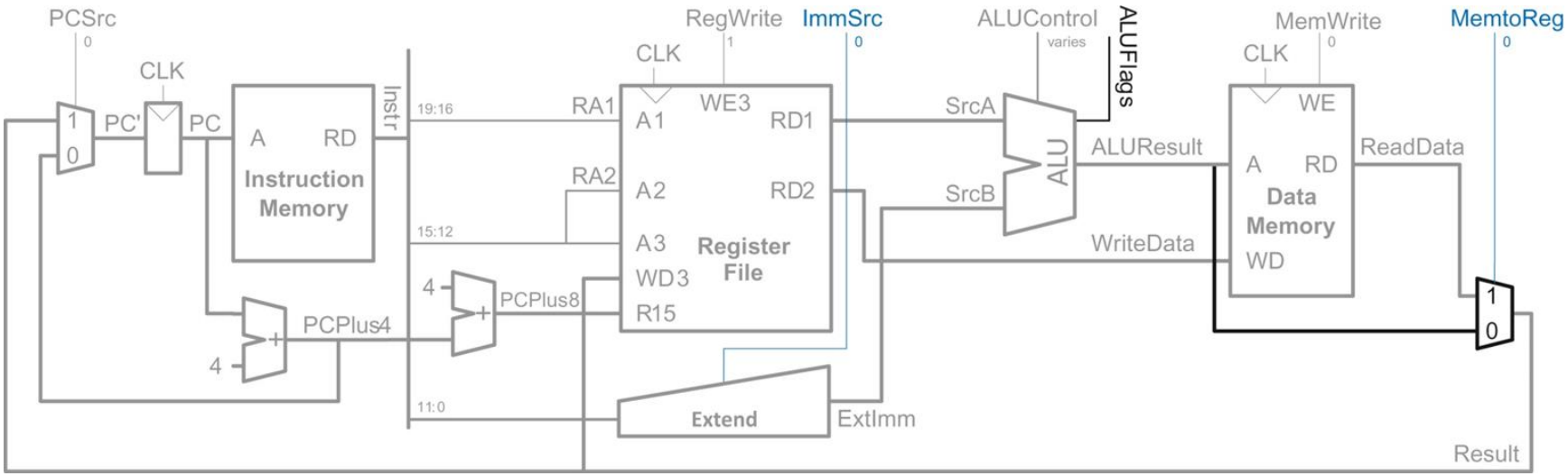


Figure 7.11 Datapath enhancements for data-processing instructions with register addressing

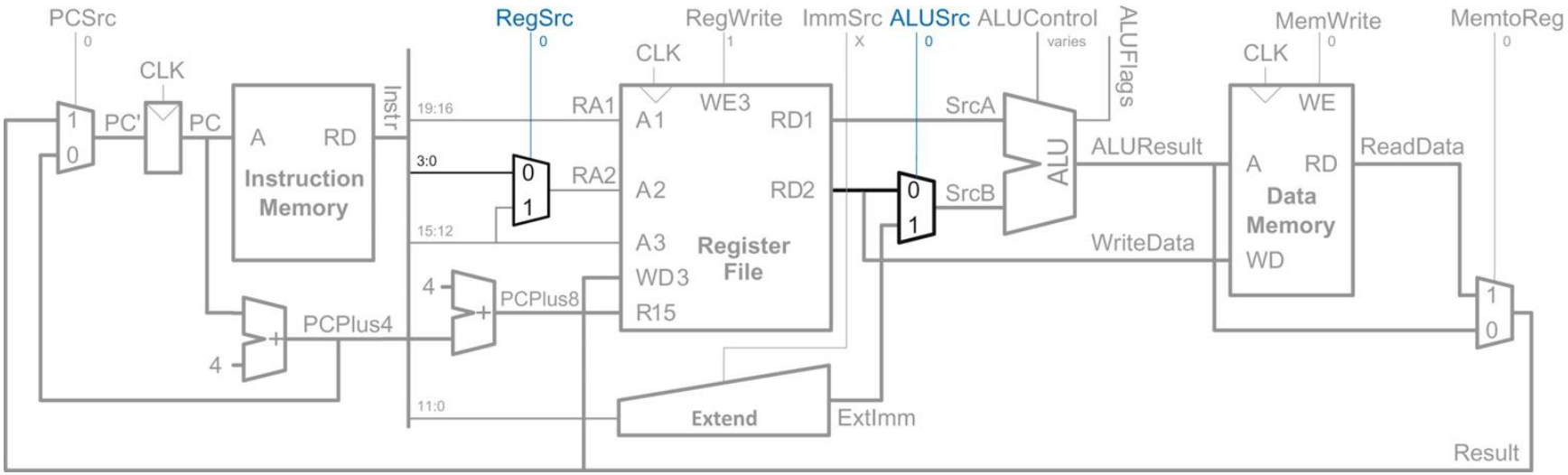


Figure 7.12 Datapath enhancements for B instruction

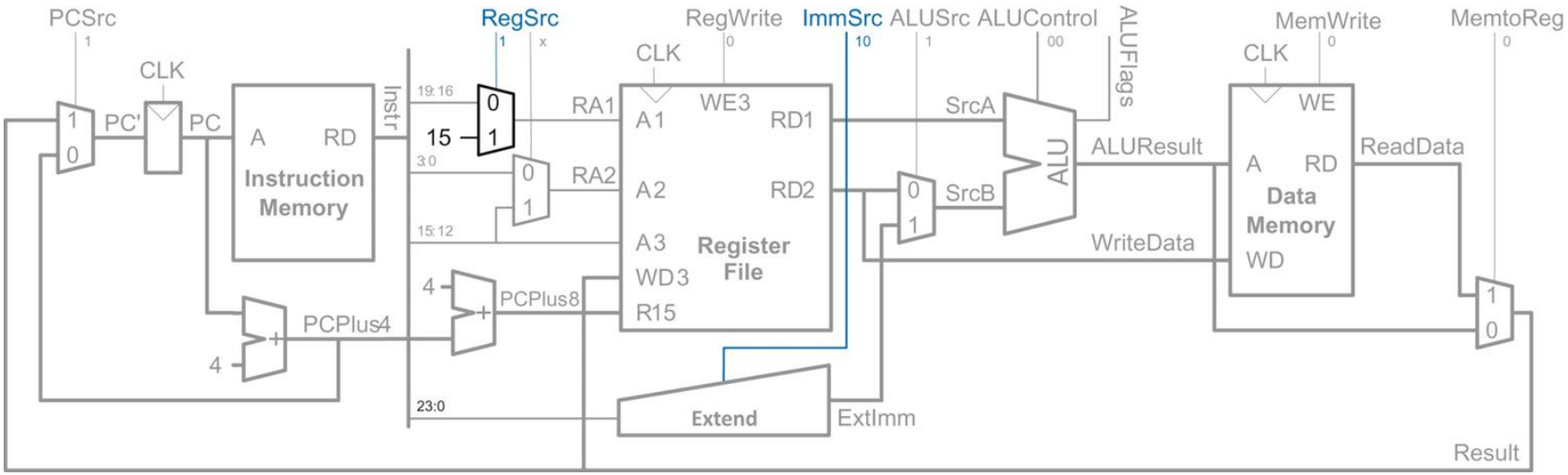


Figure 7.13 Complete single-cycle processor

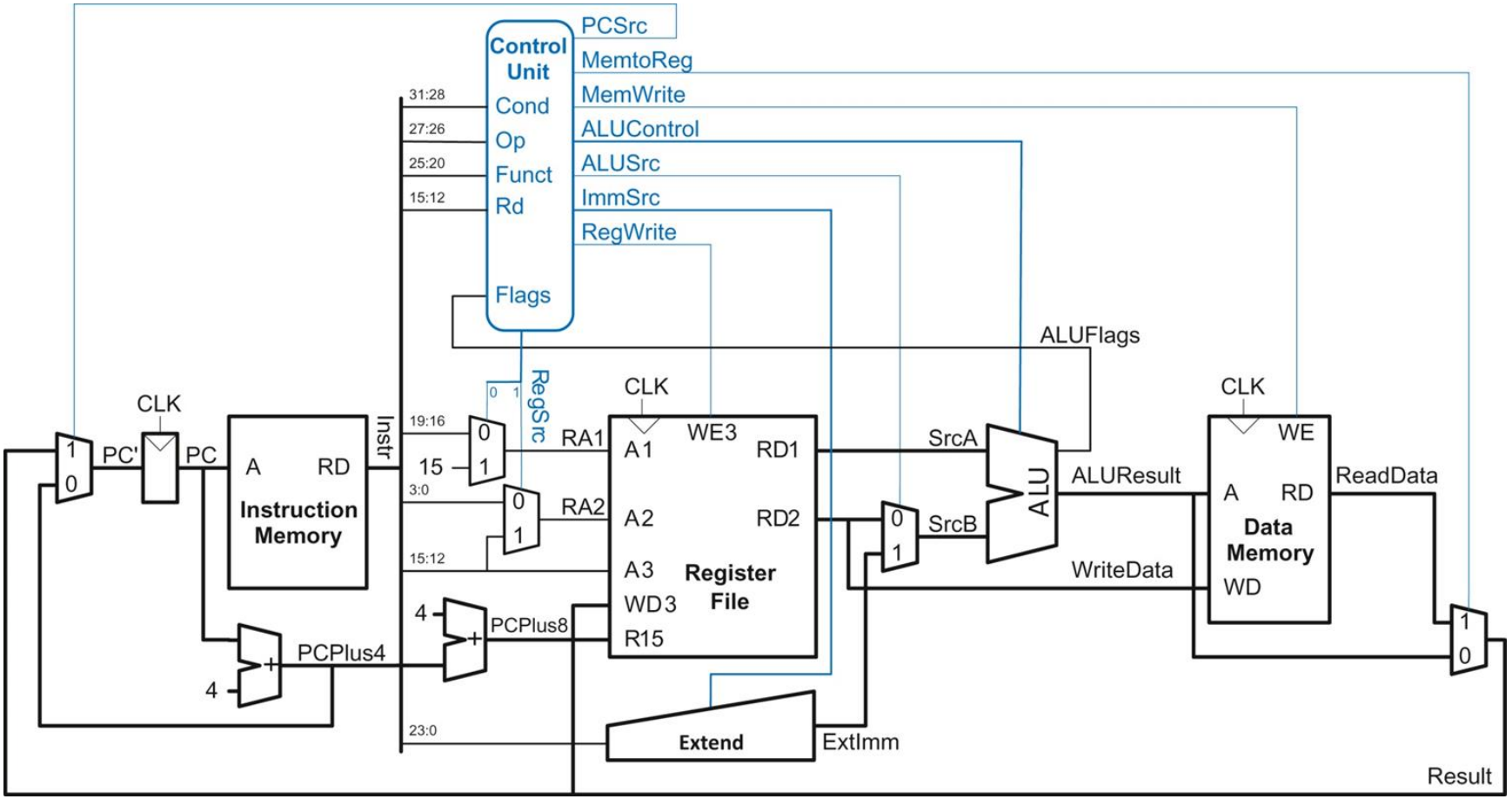


Figure 7.14 Single-cycle control unit

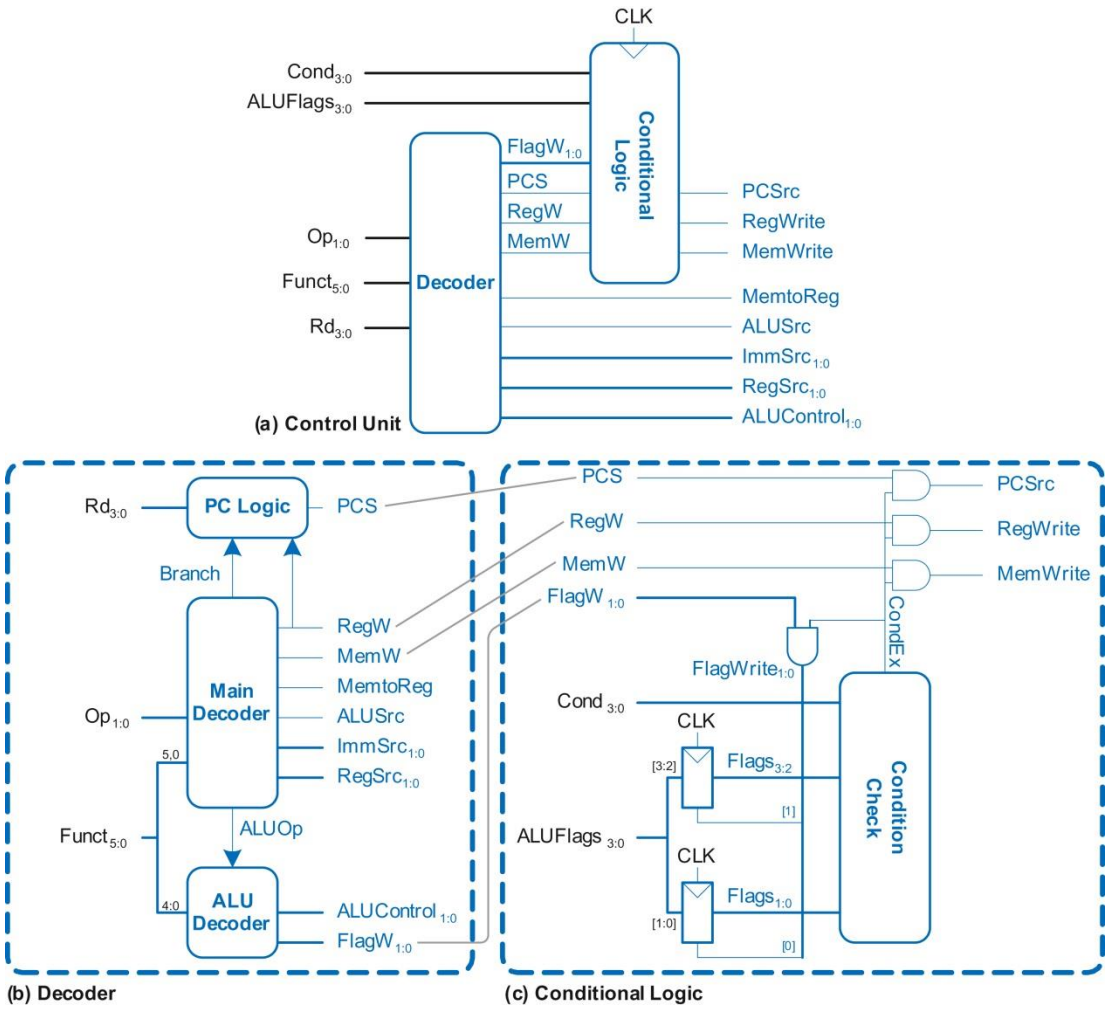


Figure 7.15 Control signals and data flow while executing an ORR instruction

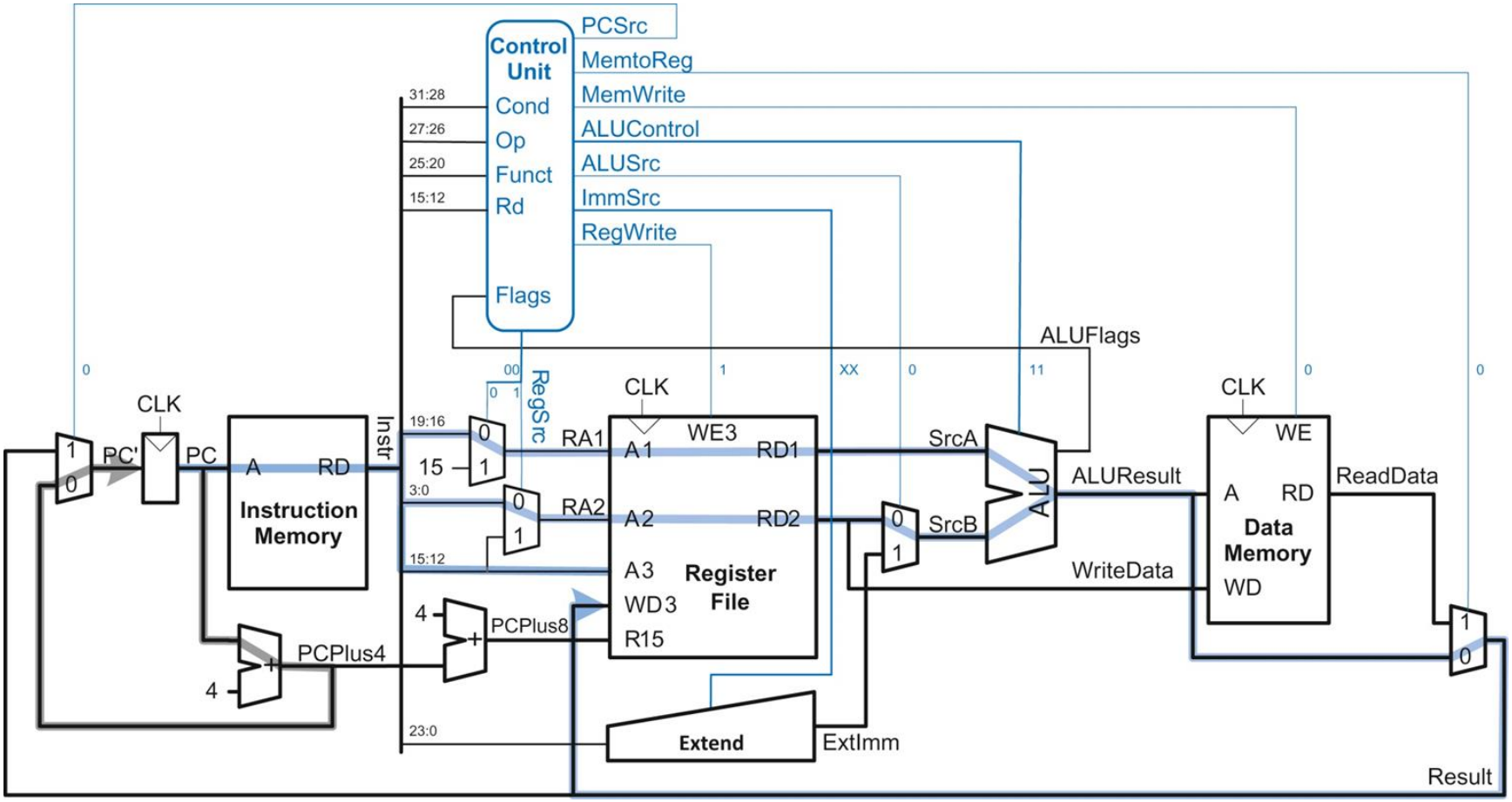


Figure 7.16 Controller modification for **CMP**

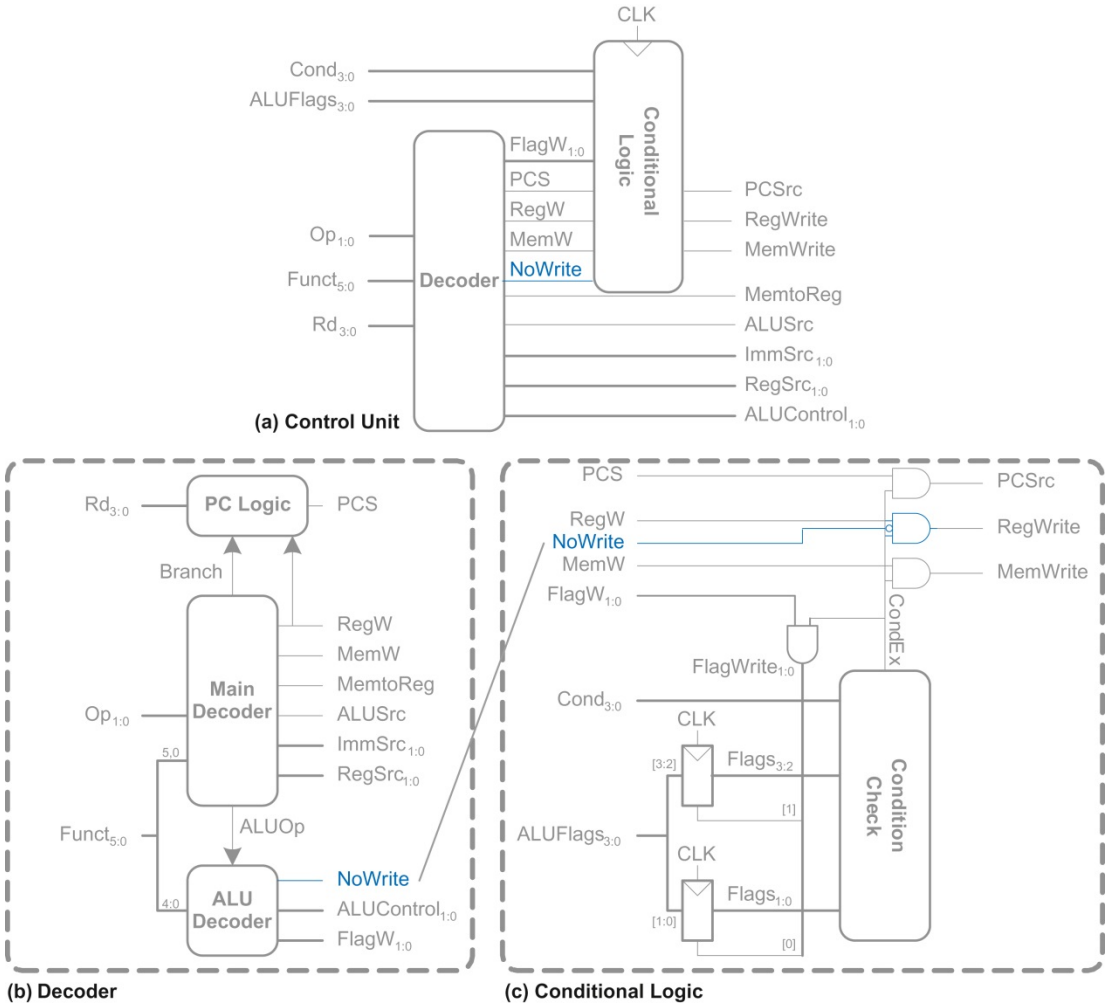


Figure 7.17 Enhanced datapath for register addressing with constant shifts

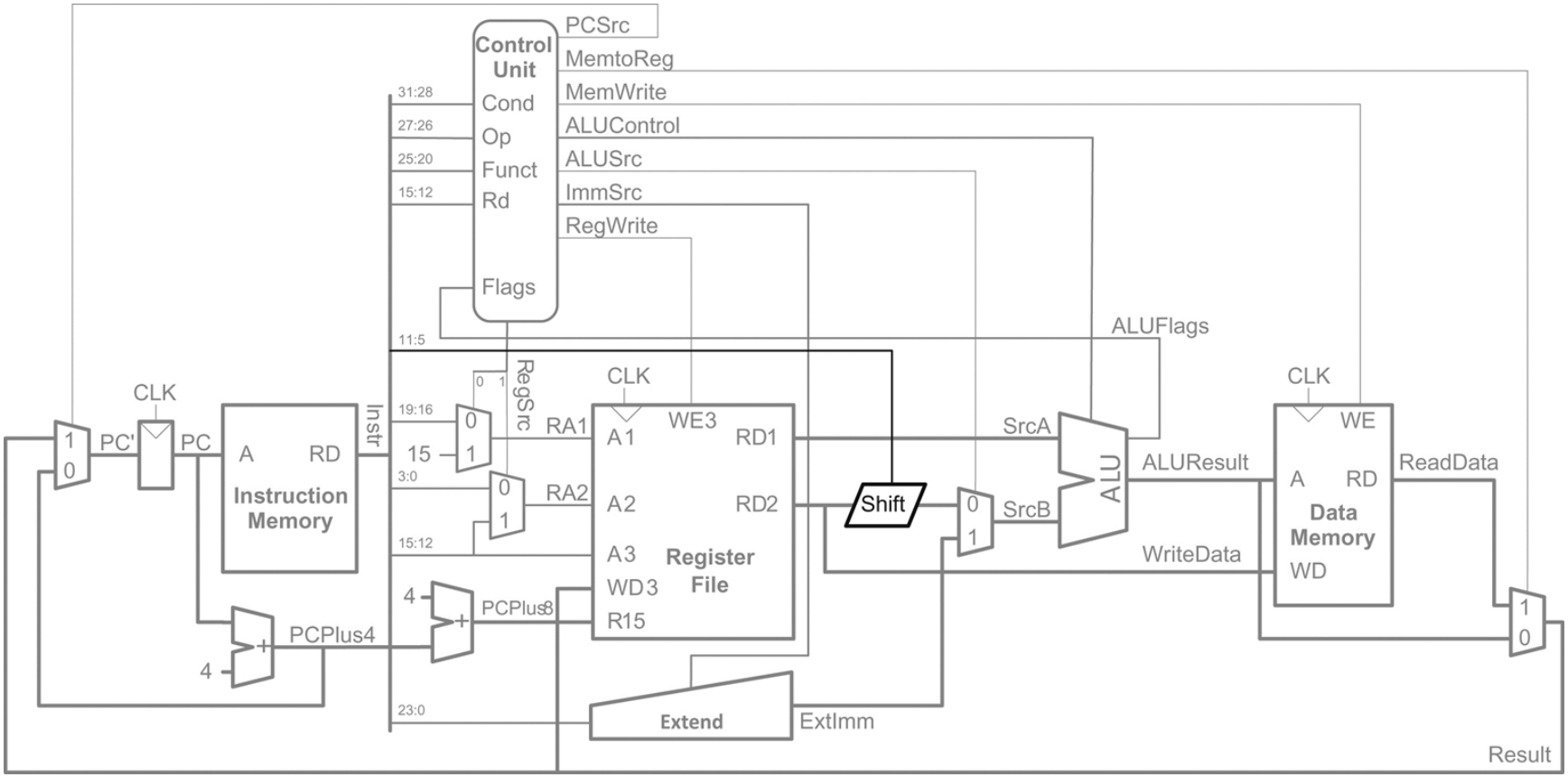


Figure 7.18 LDR critical path

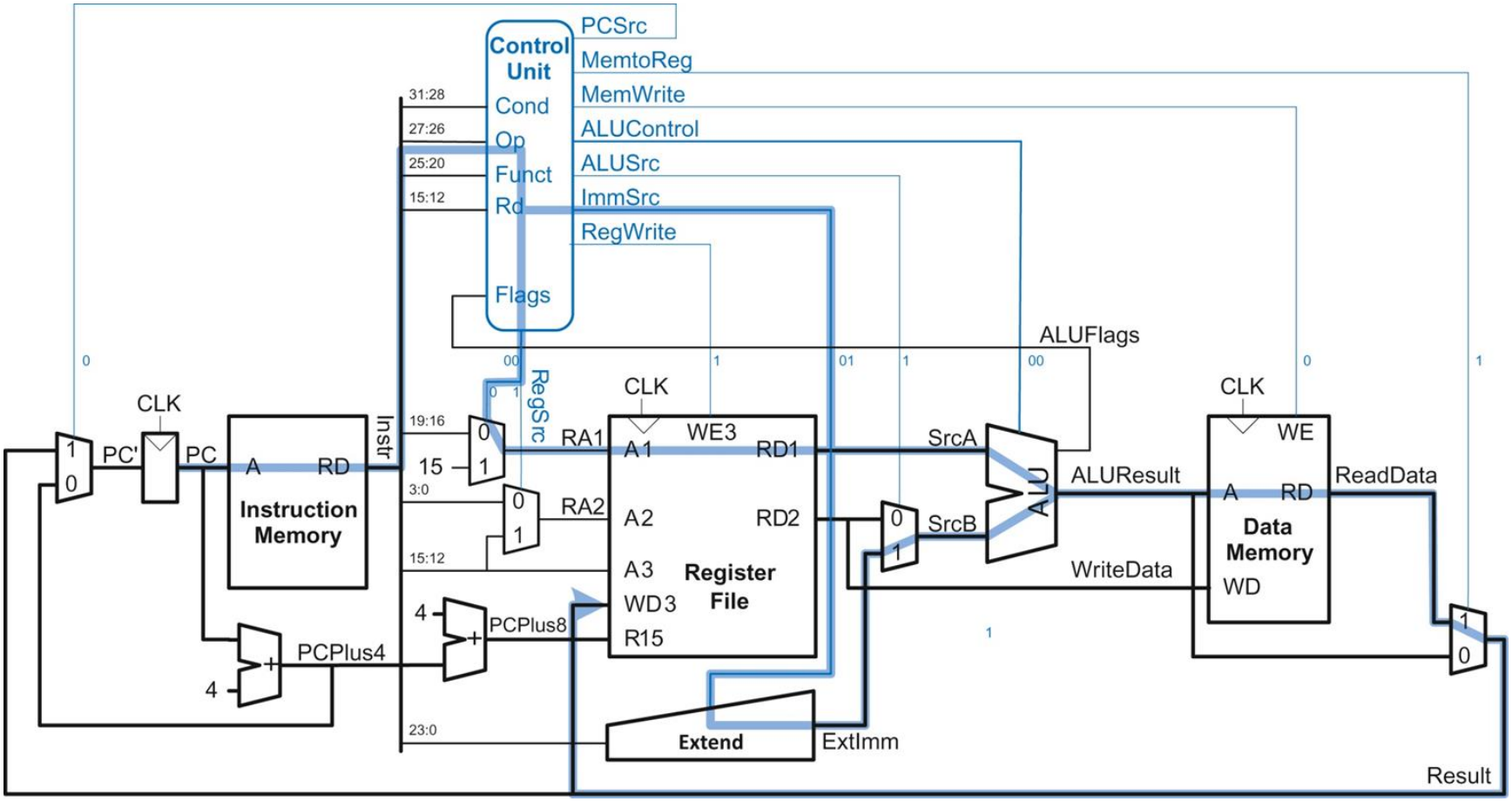


Figure 7.19 State elements with unified instruction/data memory

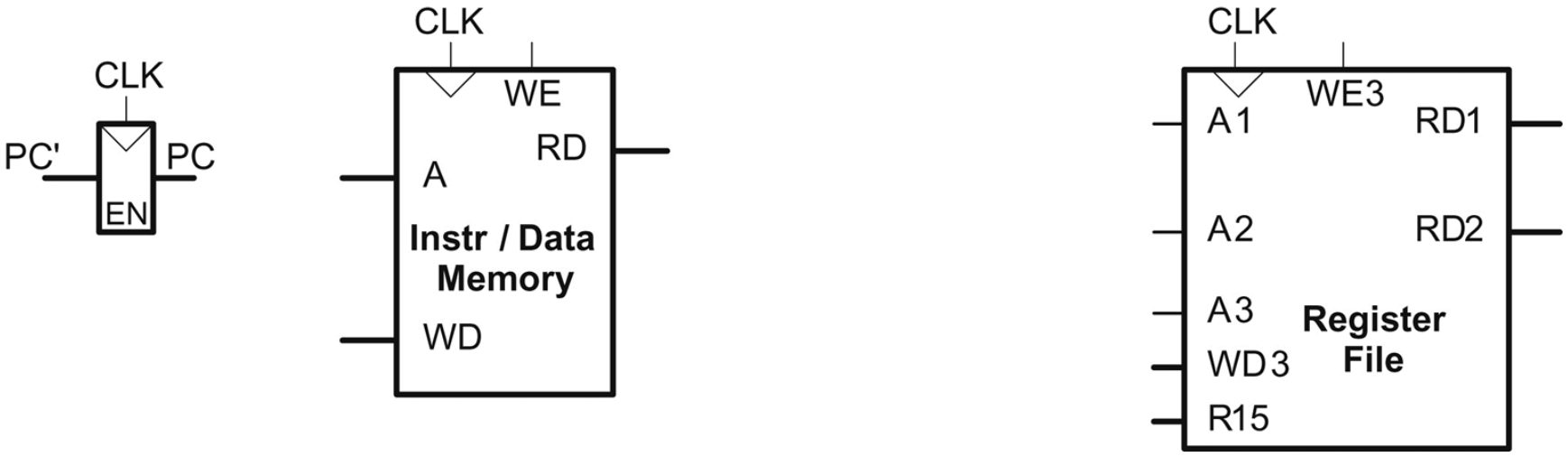


Figure 7.20 Fetch instruction from memory

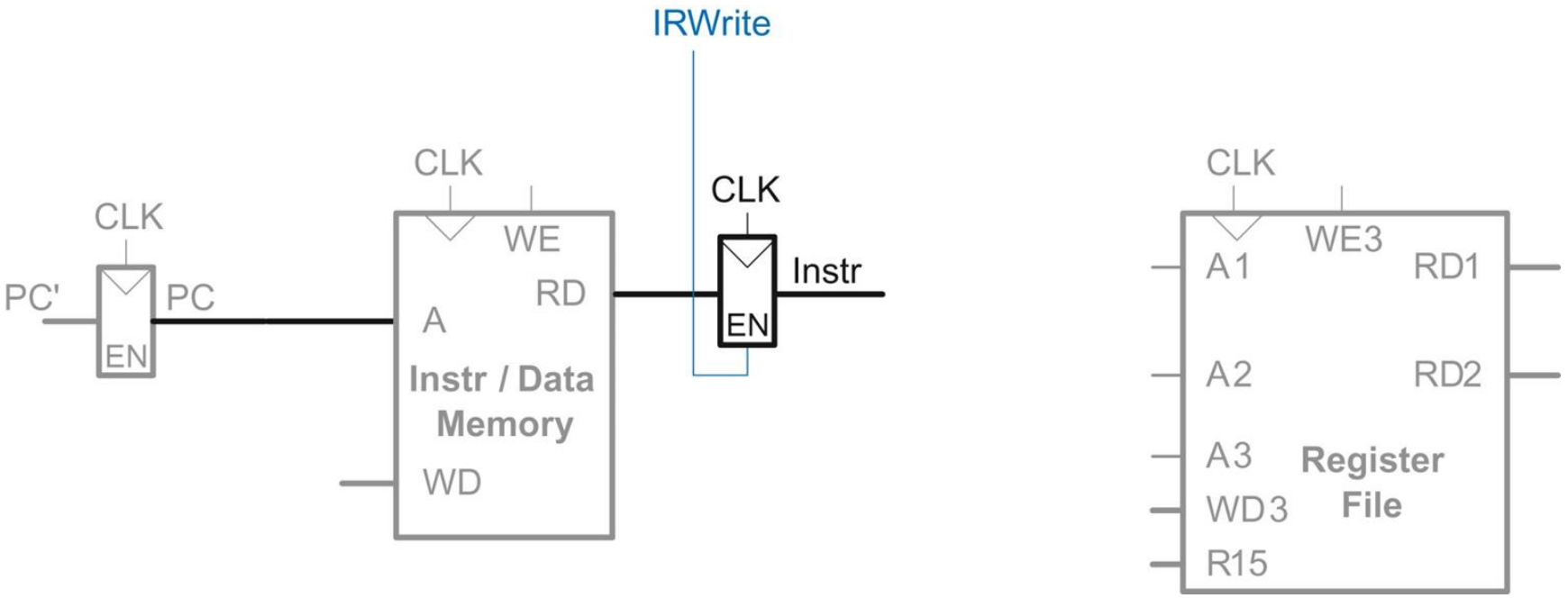


Figure 7.21 Read one source from register file and extend the second source from the immediate field

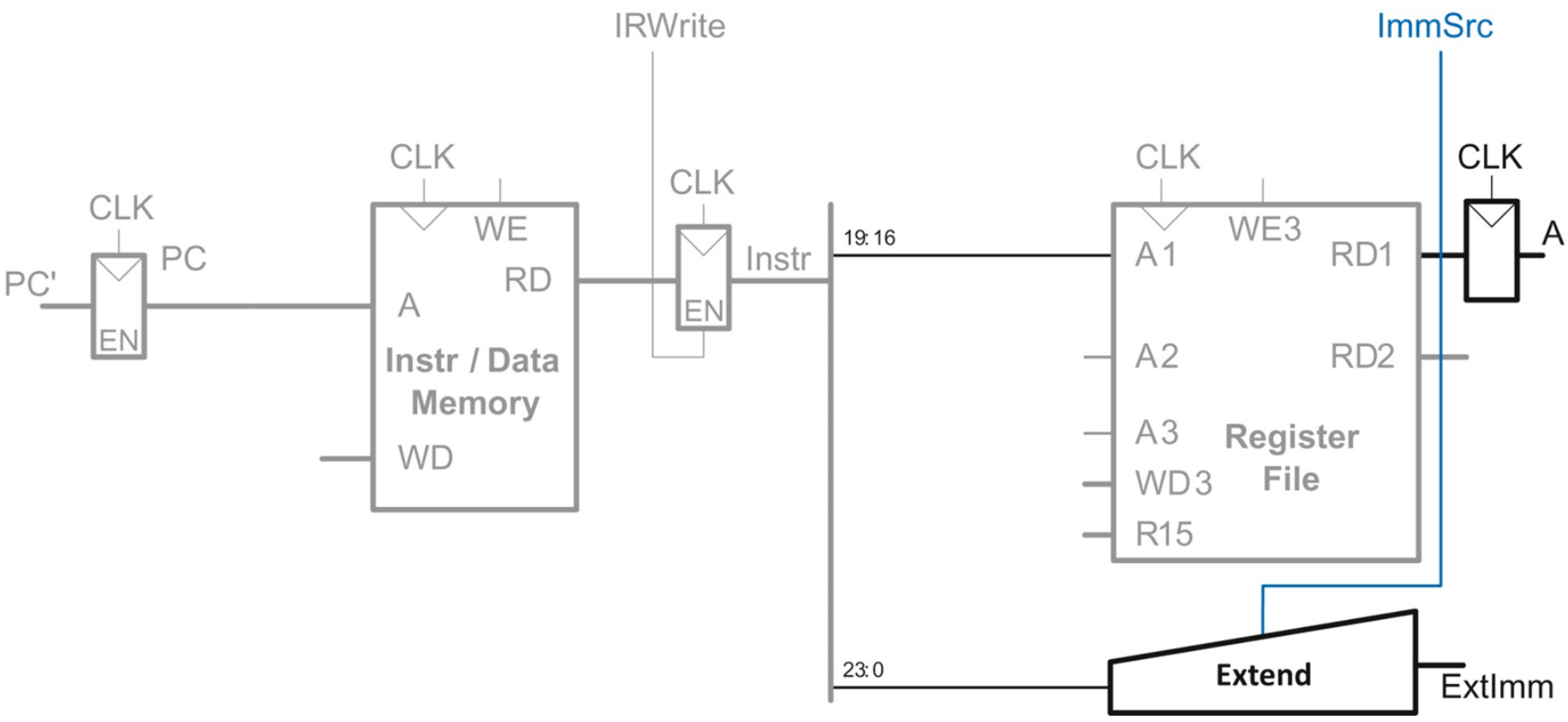


Figure 7.22 Add base address to offset

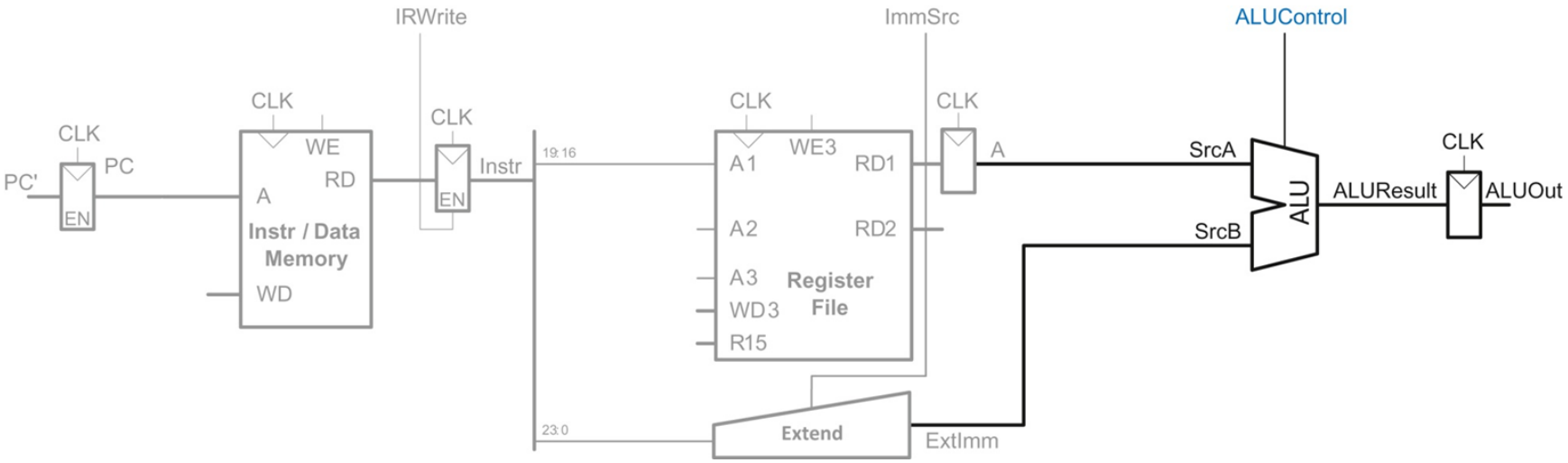


Figure 7.23 Load data from memory

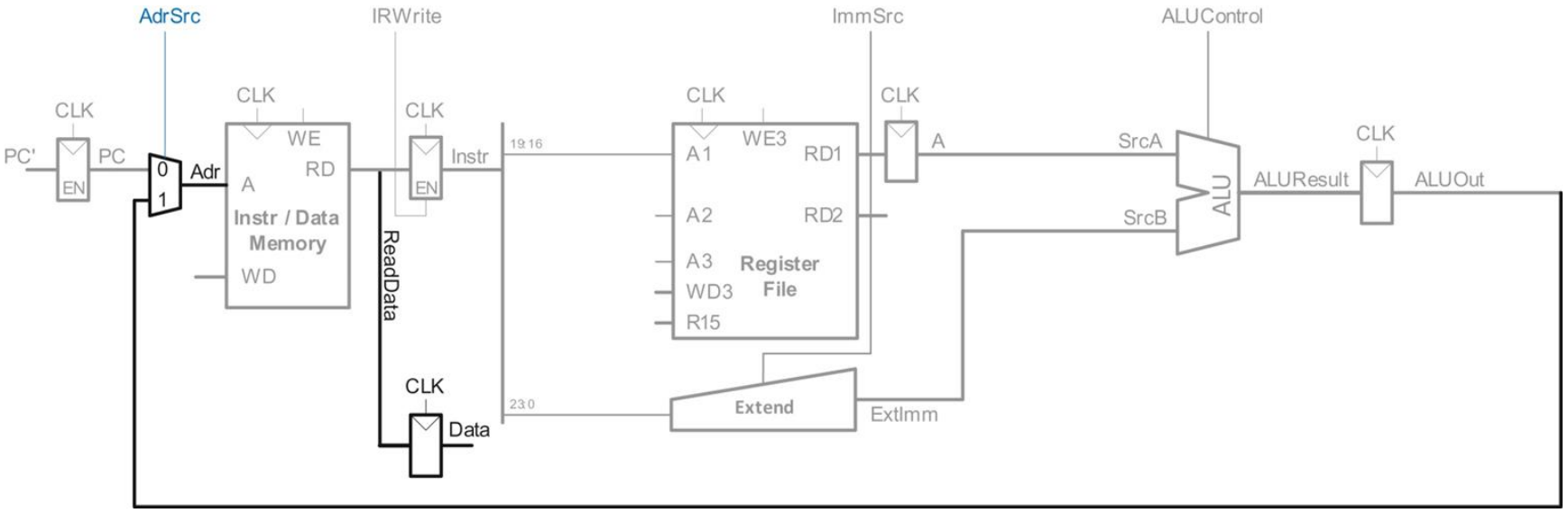


Figure 7.24 Write data back to register file

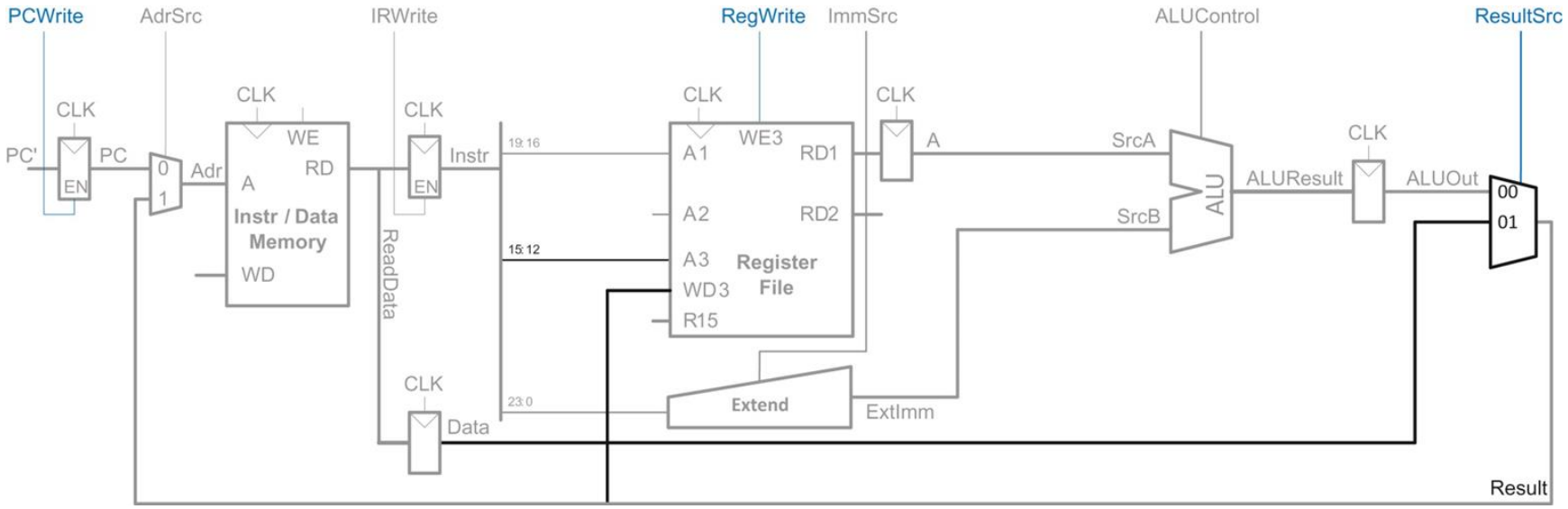


Figure 7.25 Increment PC by 4

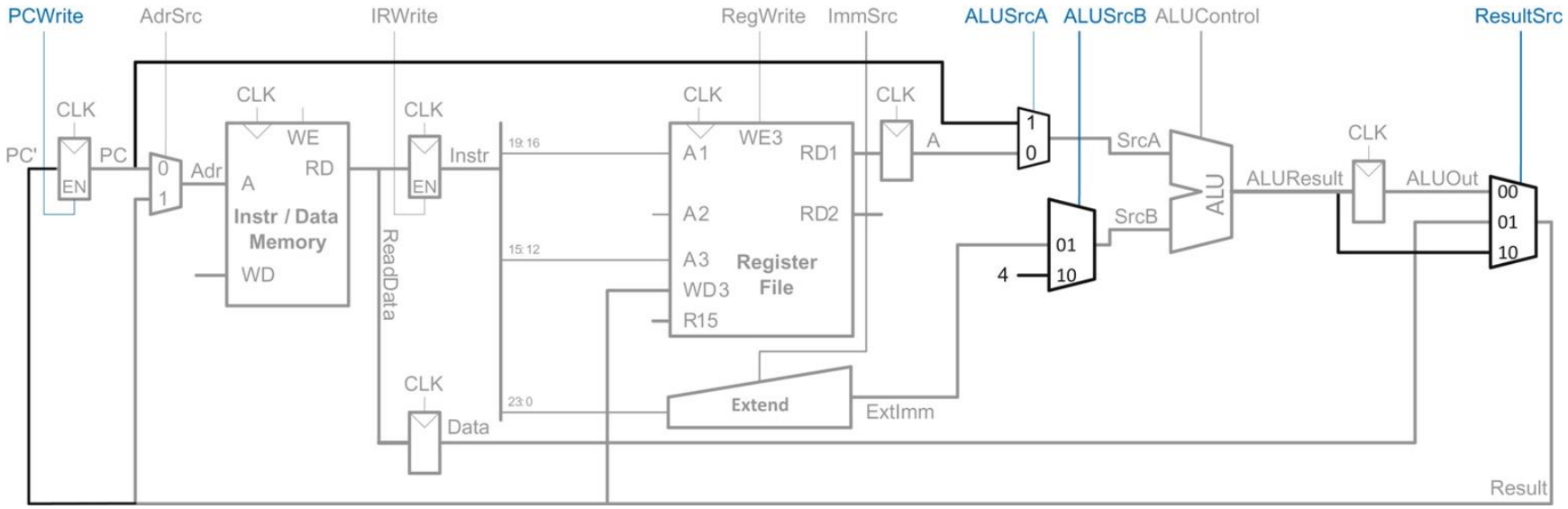


Figure 7.26 Handle R15 reads and writes

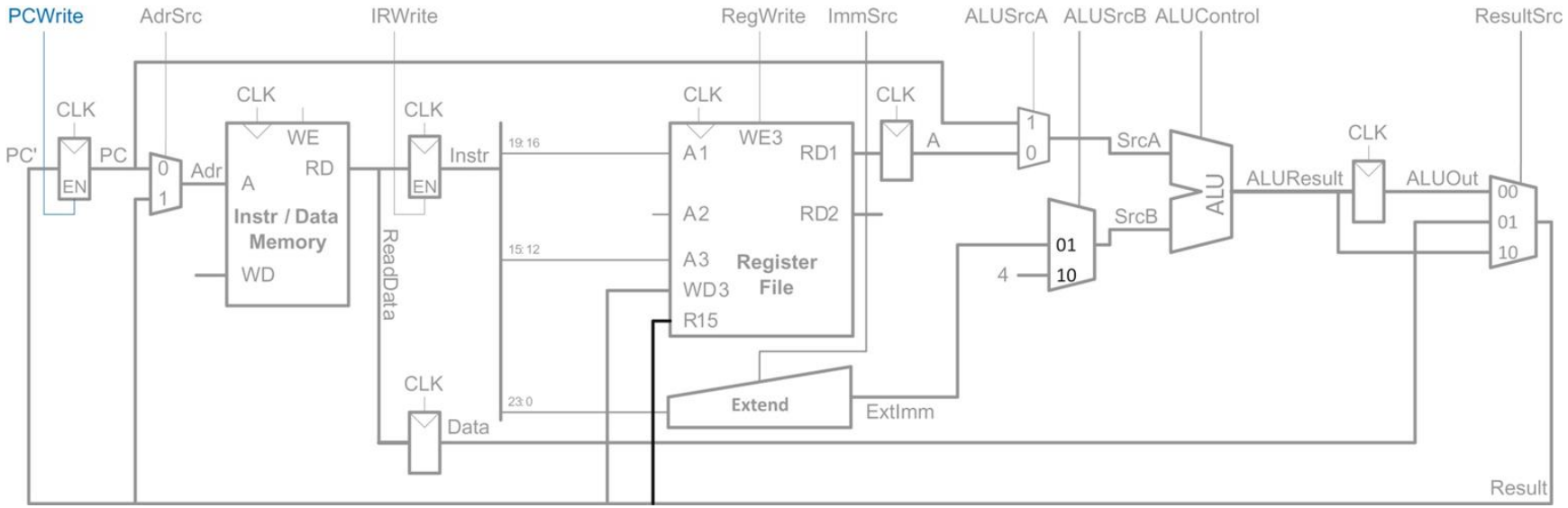


Figure 7.27 Enhanced datapath for STR instruction

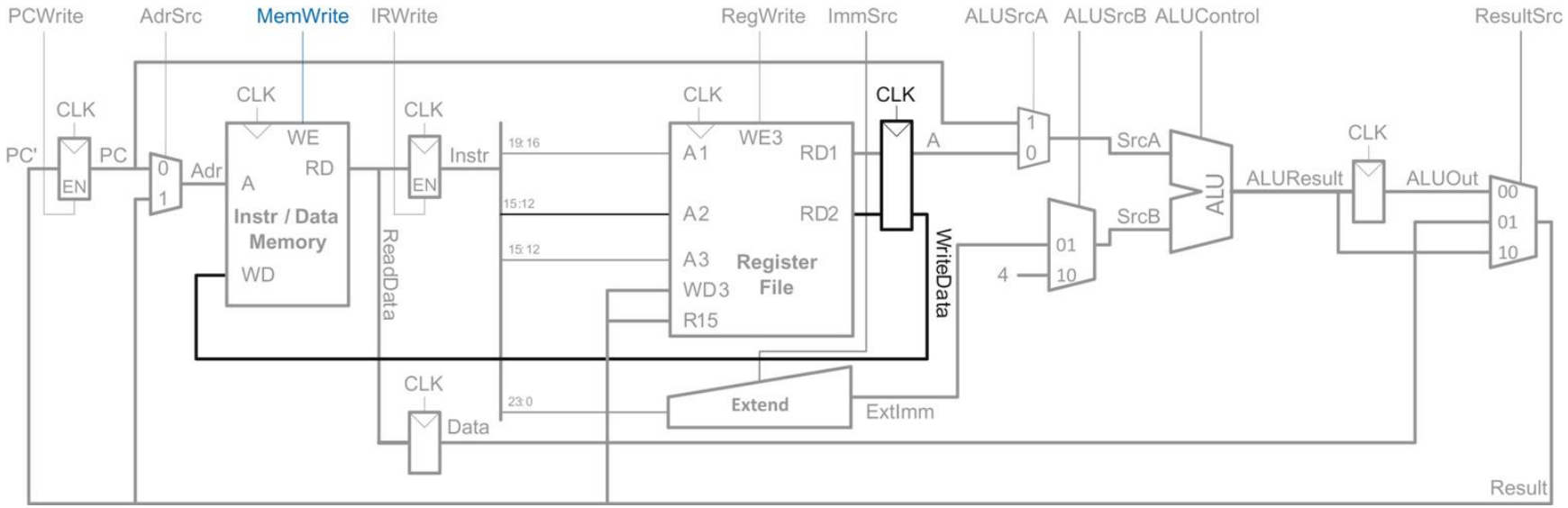


Figure 7.28 Enhanced datapath for data-processing instructions with register addressing

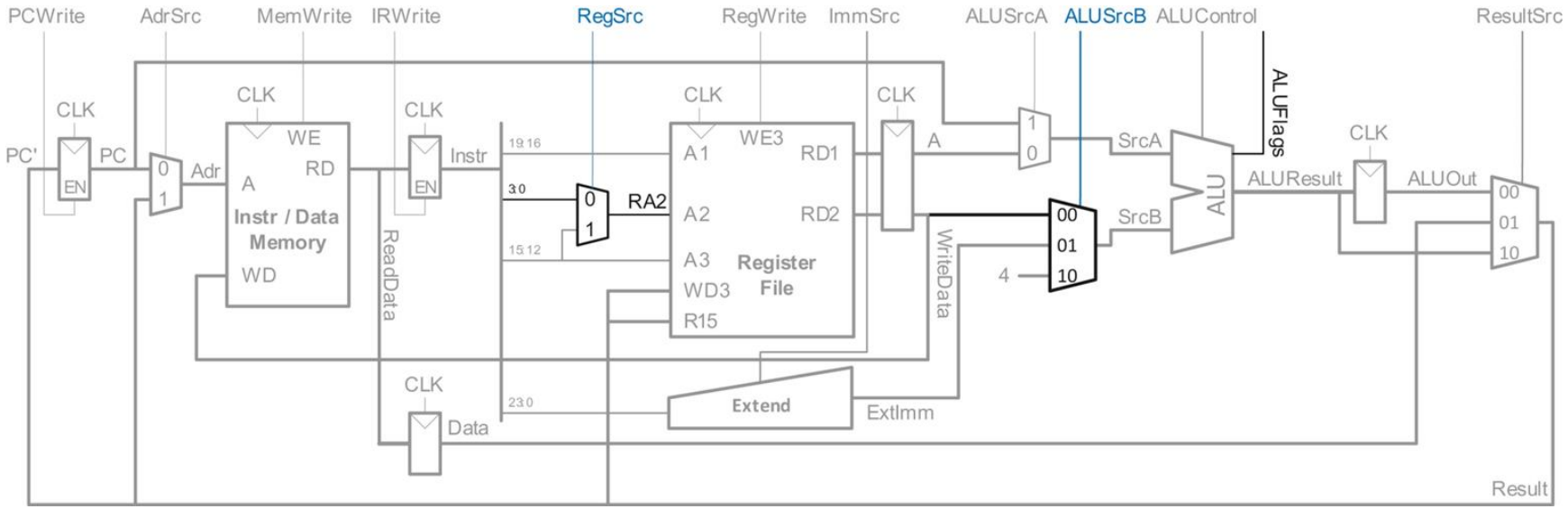


Figure 7.29 Enhanced datapath for the B instruction

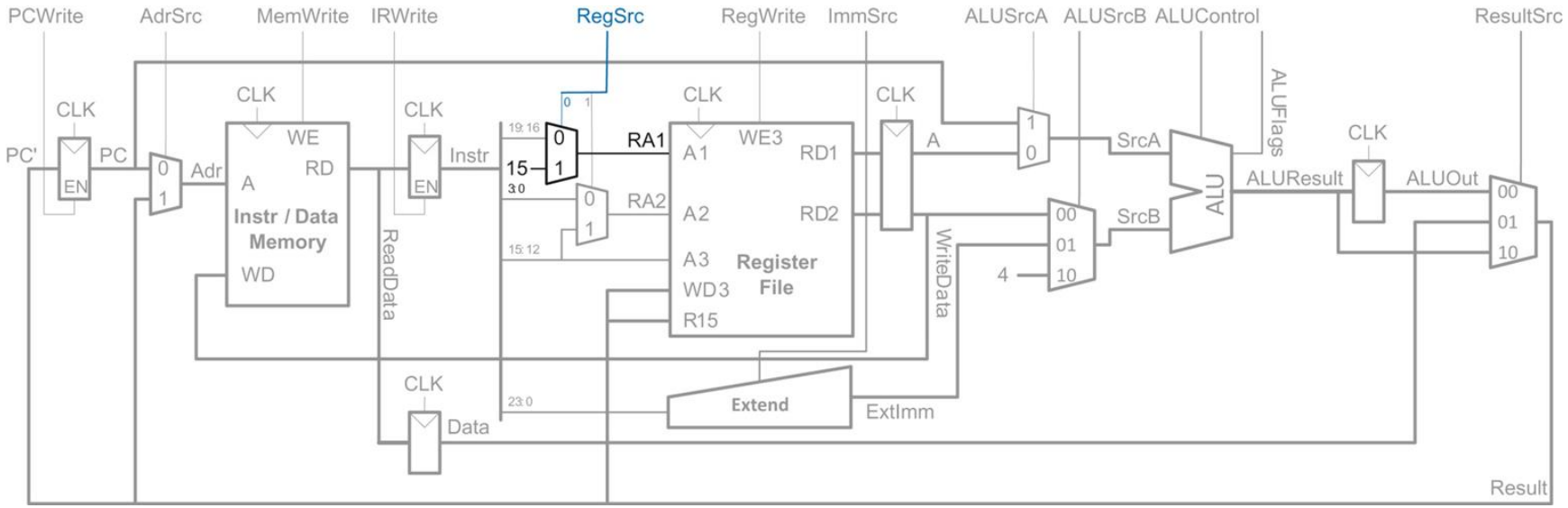


Figure 7.30 Complete multicycle processor

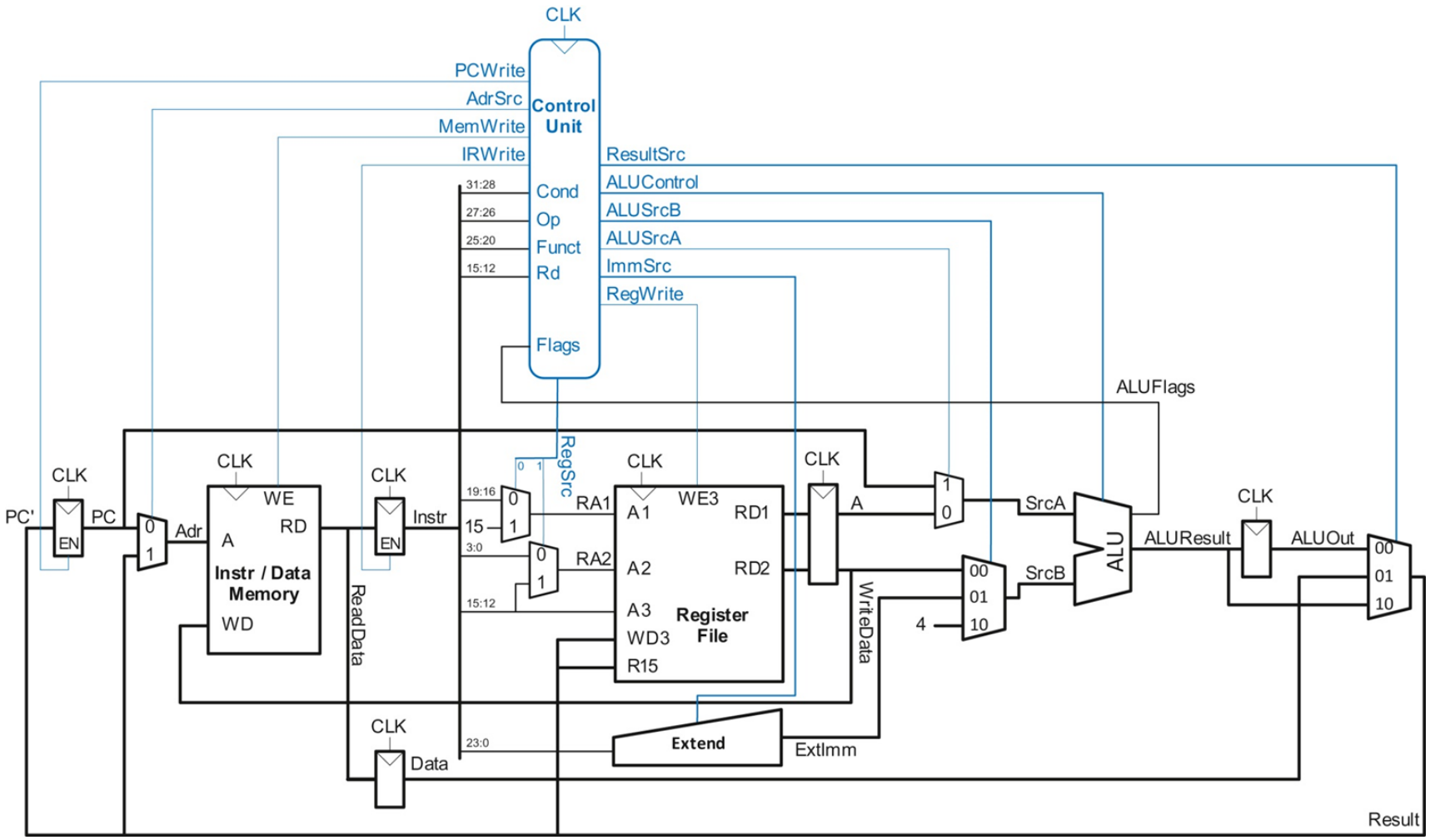


Figure 7.31 Multicycle control unit

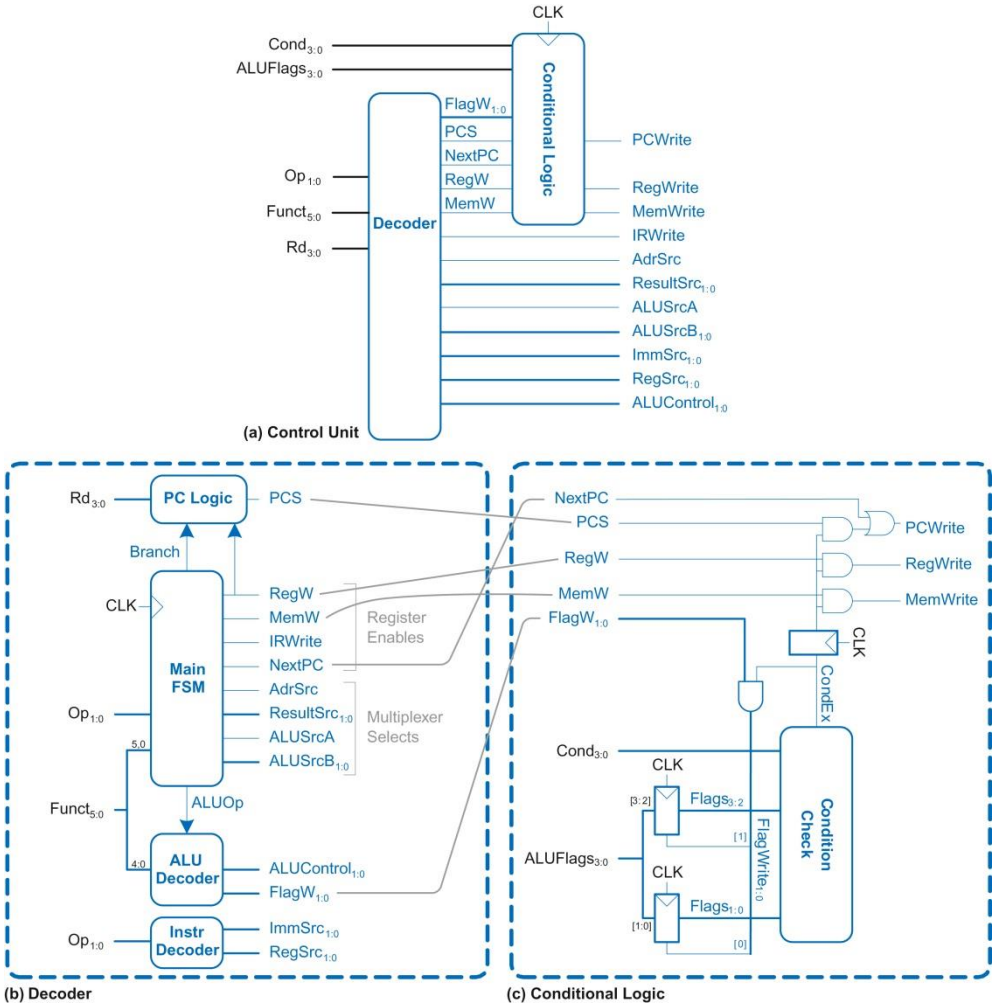


Figure 7.32 Fetch

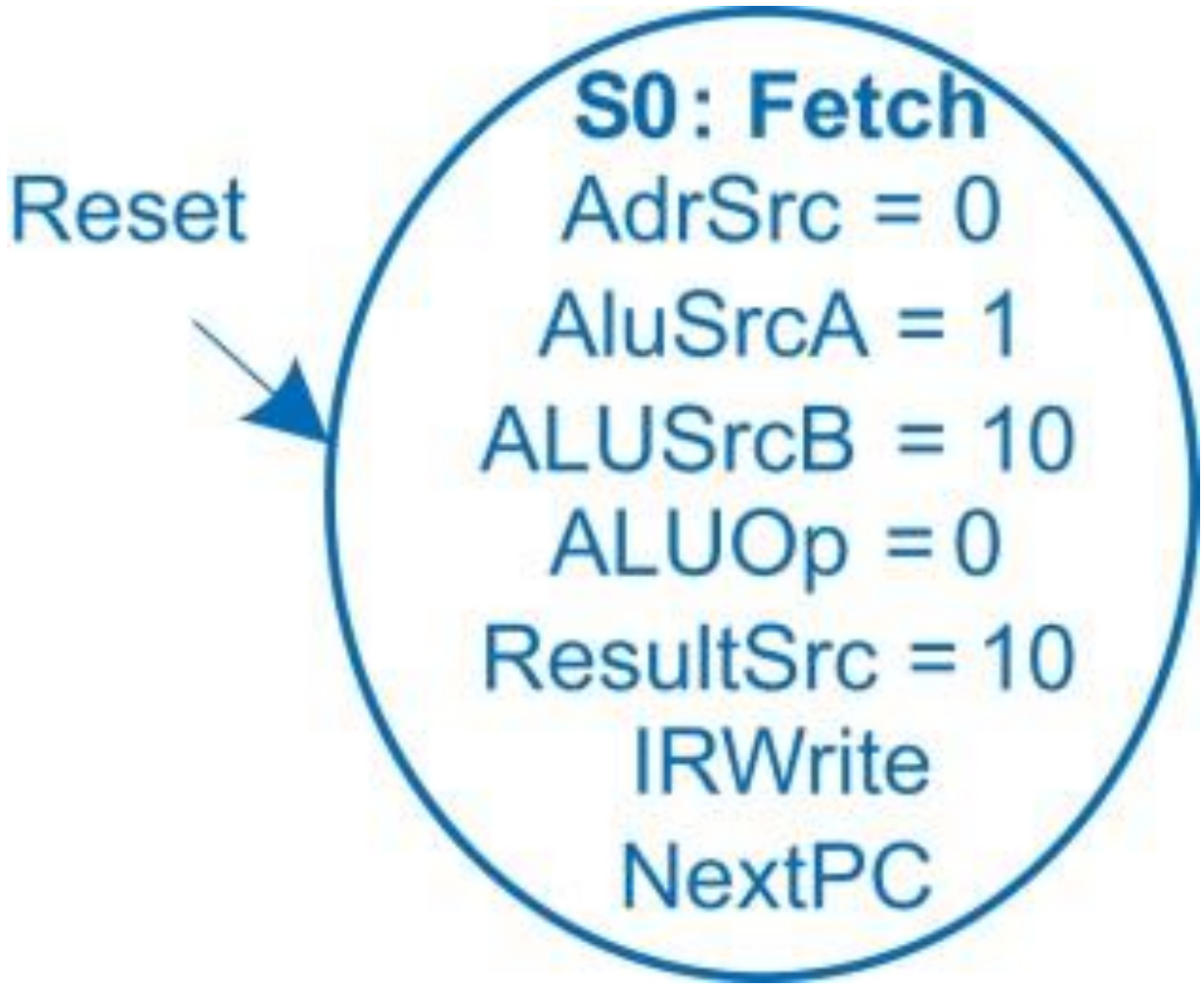


Figure 7.33 Data flow during the fetch step

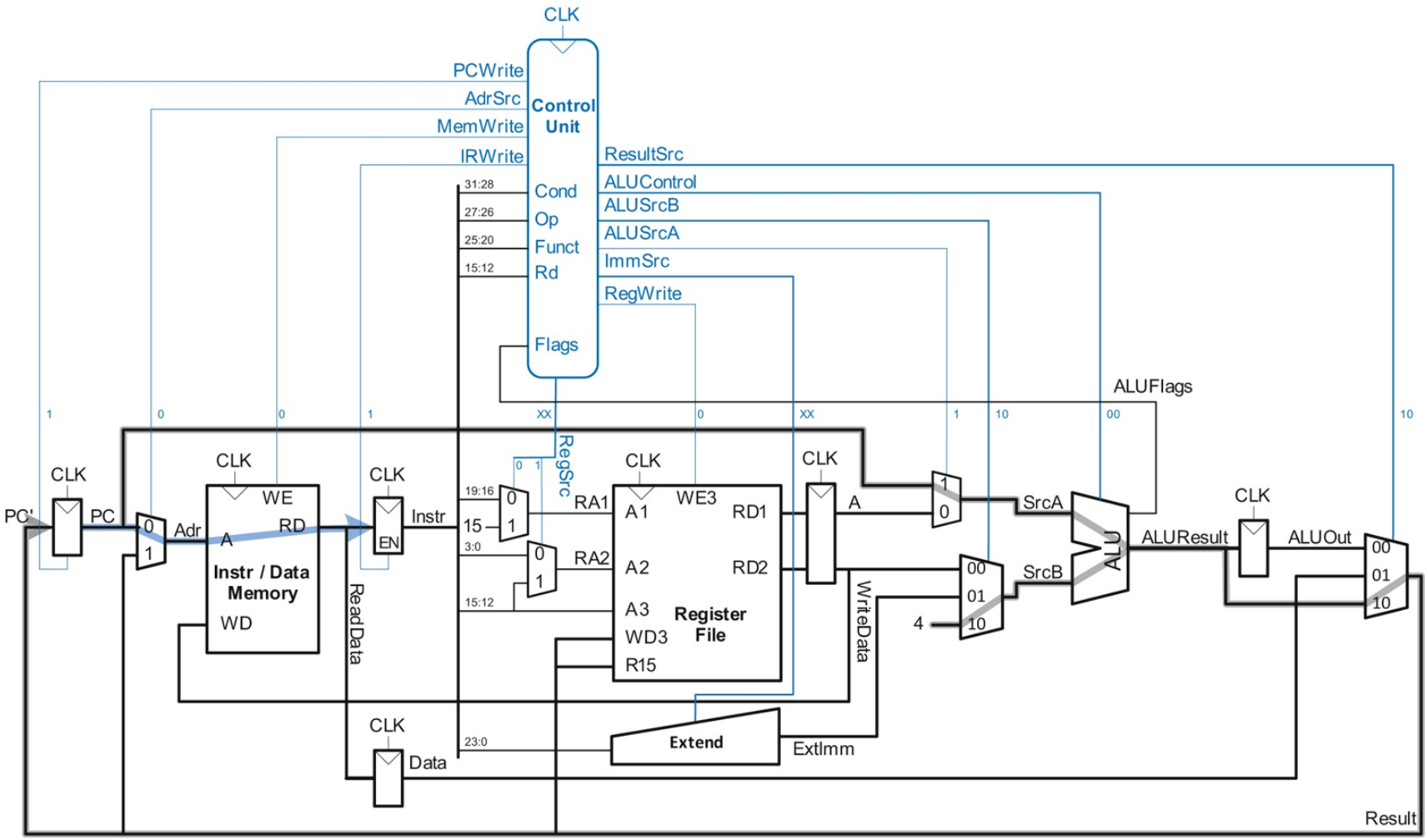


Figure 7.34 Decode

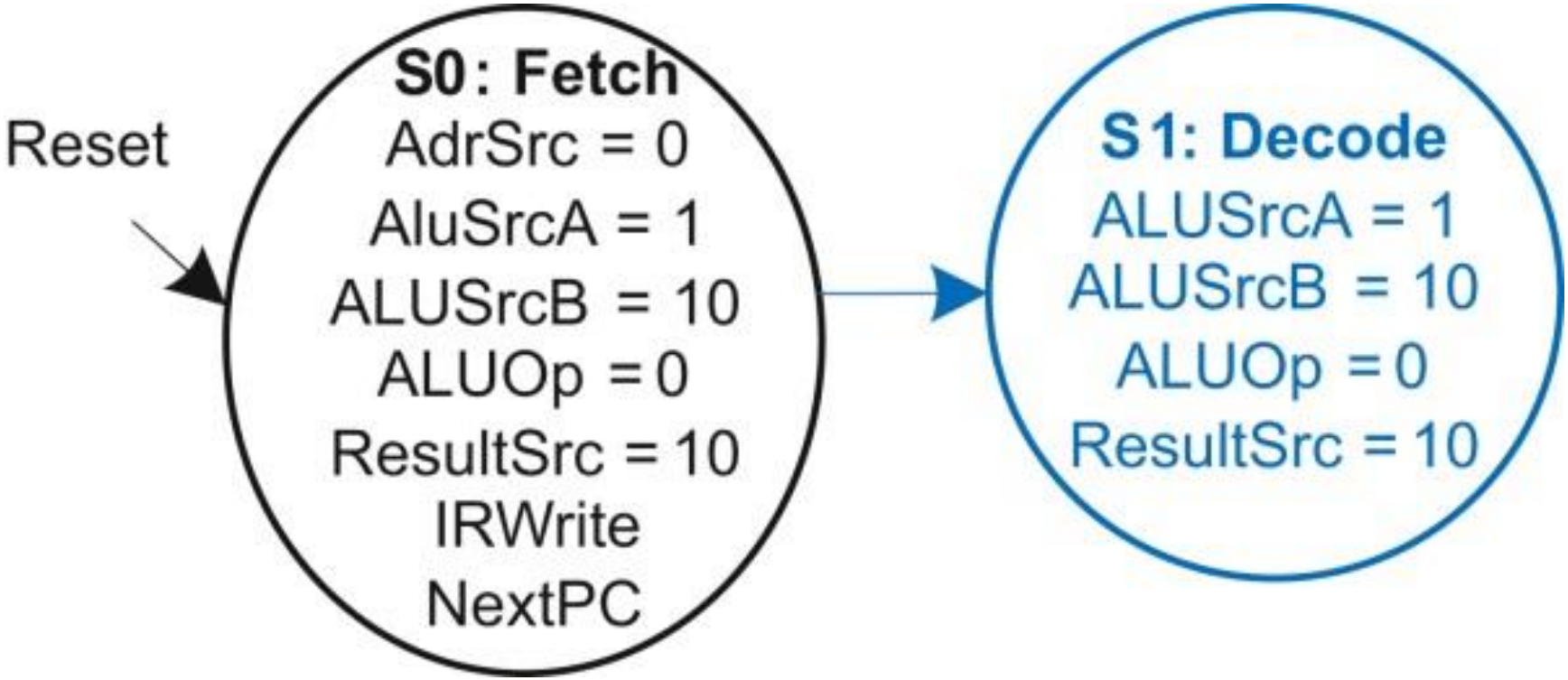


Figure 7.35 Data flow during the Decode step

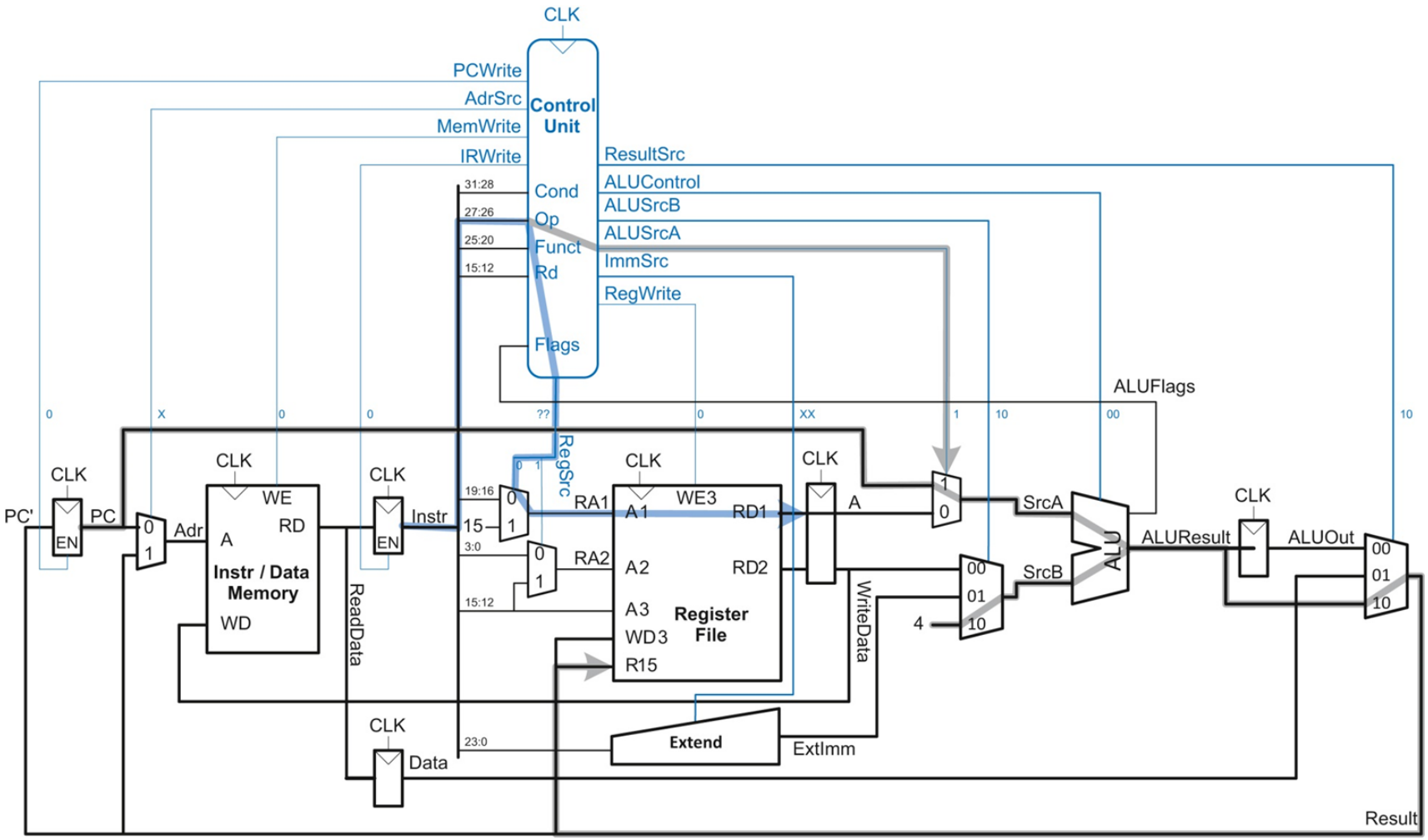


Figure 7.36 Memory address computation

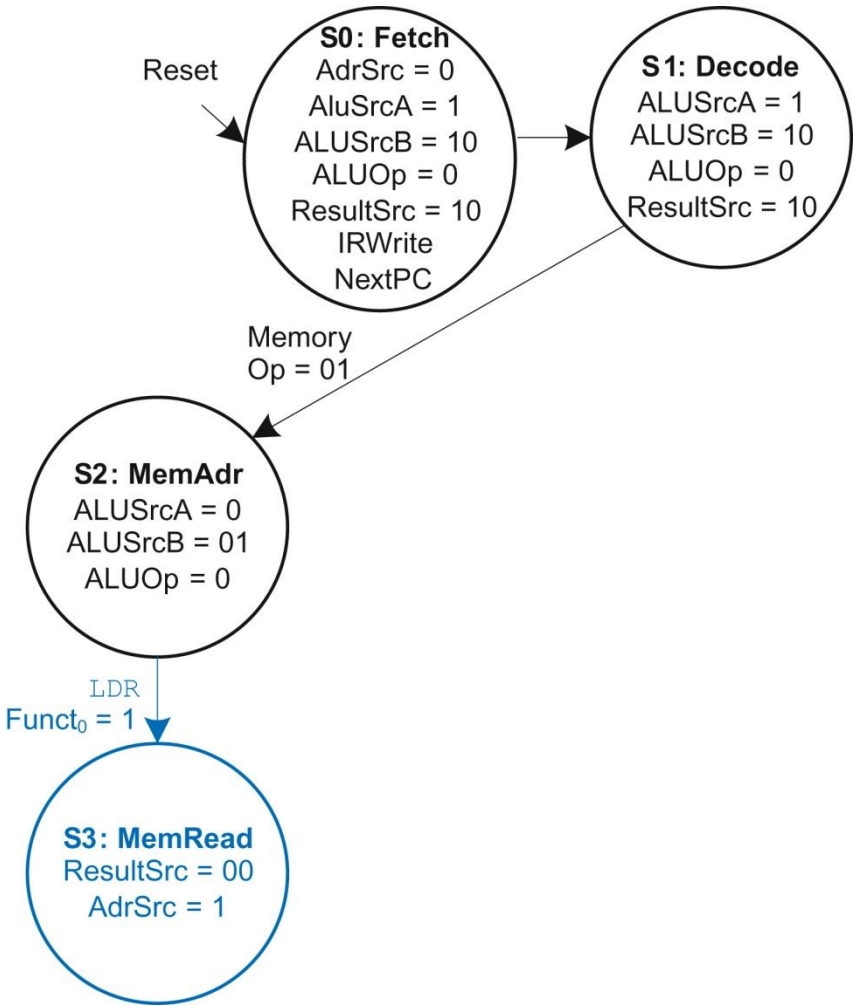


Figure 7.37 Data flow during memory address computation

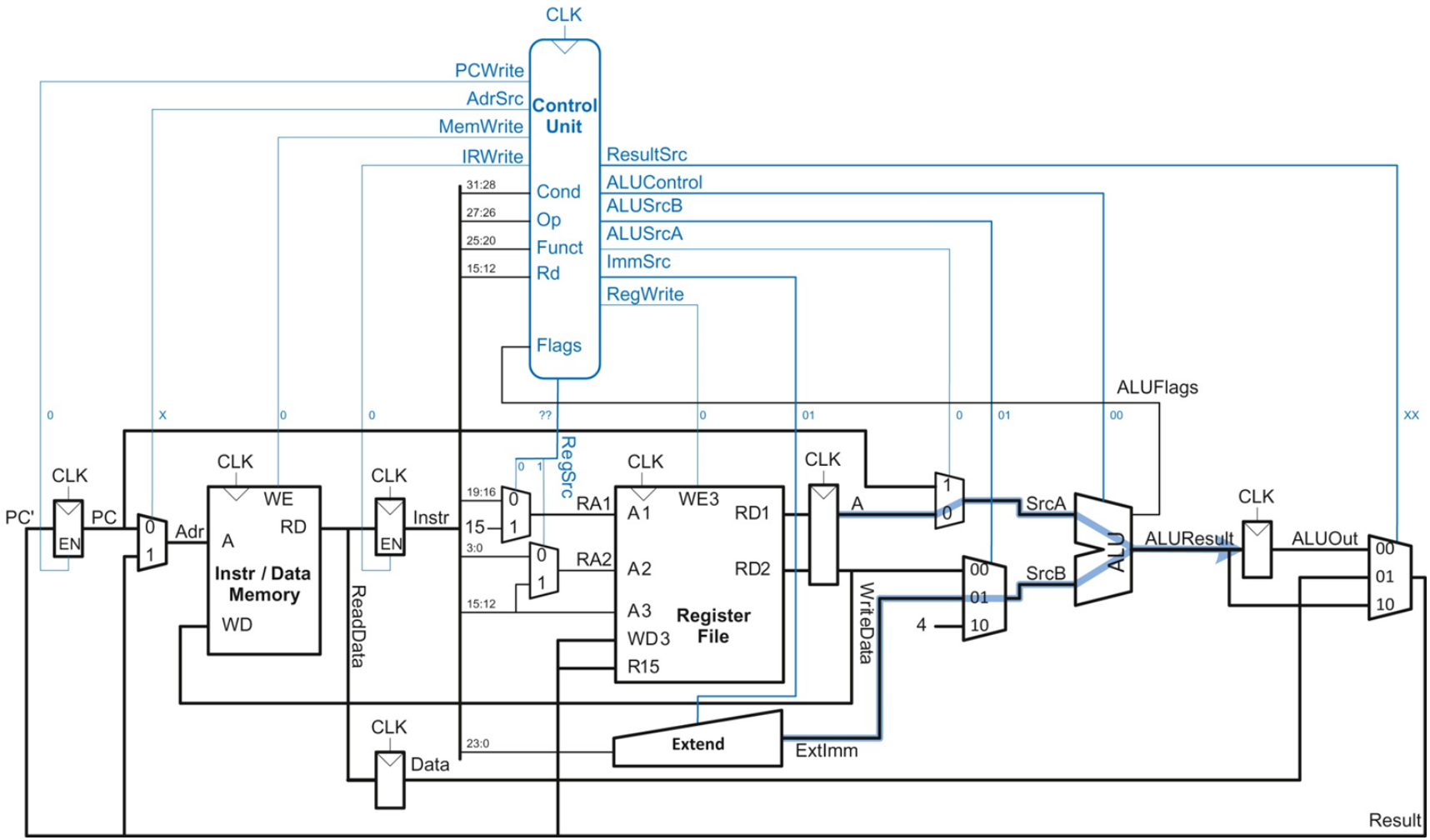


Figure 7.38 Memory read

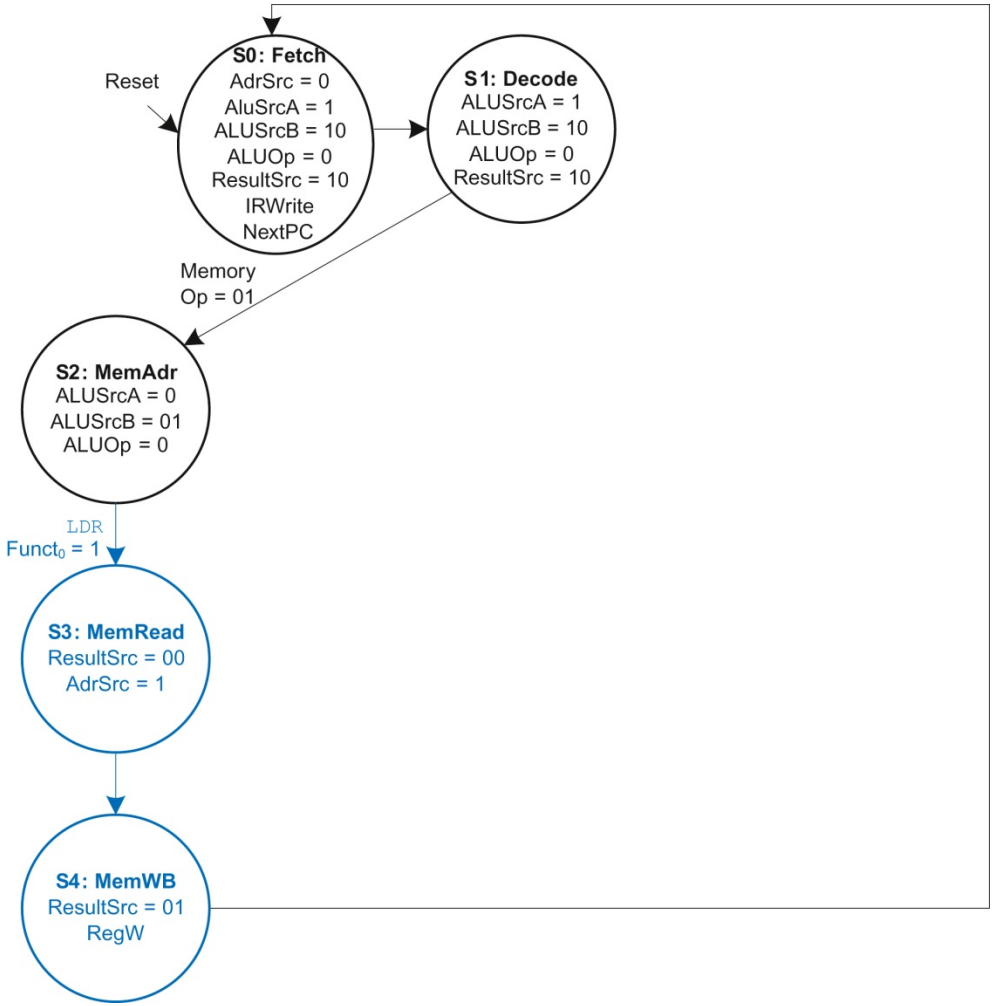


Figure 7.39 Memory write

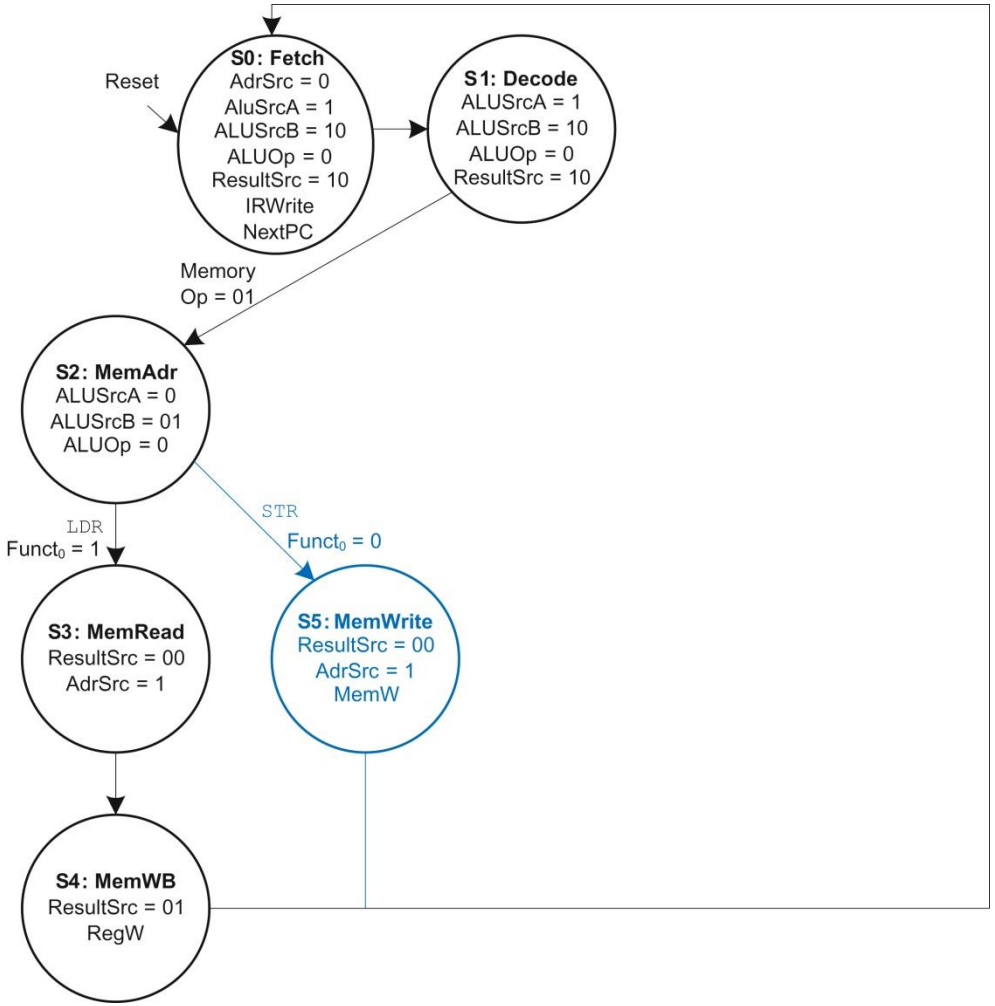


Figure 7.40 Data-processing

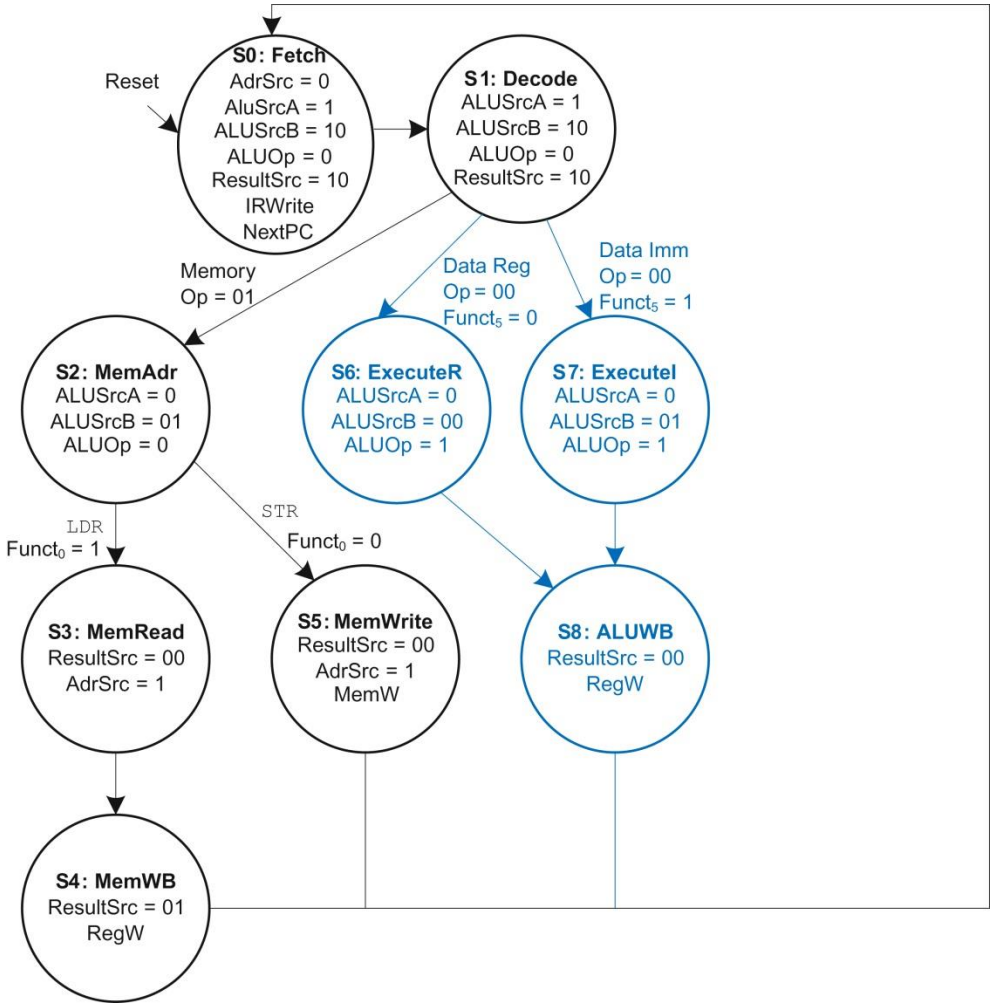
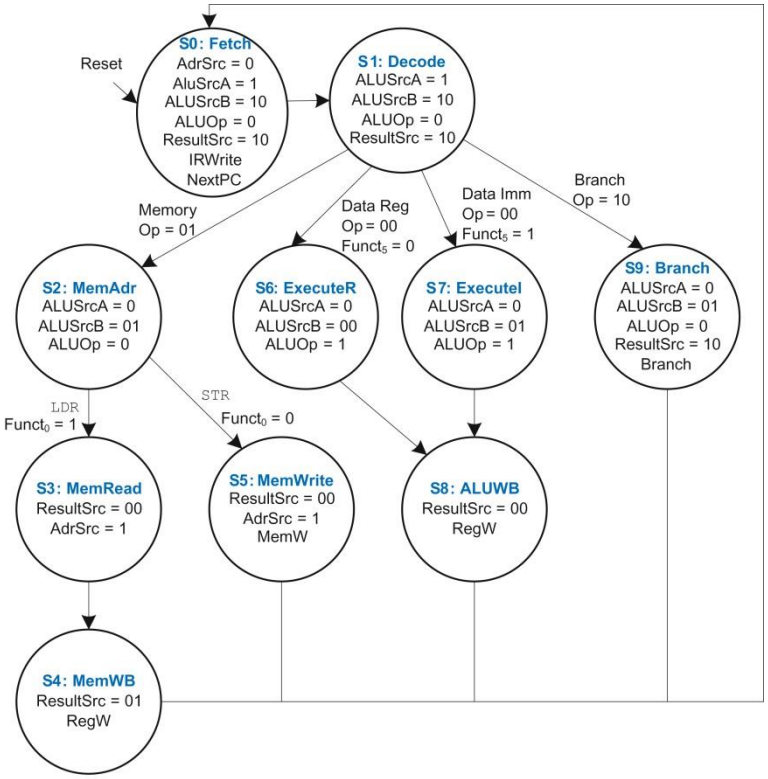


Figure 7.41 Complete multicycle control FSM



State	Datapath μOp
Fetch	Instr ← Mem[PC]; PC ← PC+4
Decode	ALUOut ← PC +4
MemAdr	ALUOut ← Rn + Imm
MemRead	Data ← Mem[ALUOut]
MemWB	Rd ← Data
MemWrite	Mem[ALUOut] ← Rd
ExecuteR	ALUOut ← Rn op Rm
ExecuteI	ALUOut ← Rn op Imm
ALUWB	Rd ← ALUOut
Branch	PC ← R15 + offset

Figure 7.42 Timing diagrams: (a) single-cycle processor and (b) pipelined processor

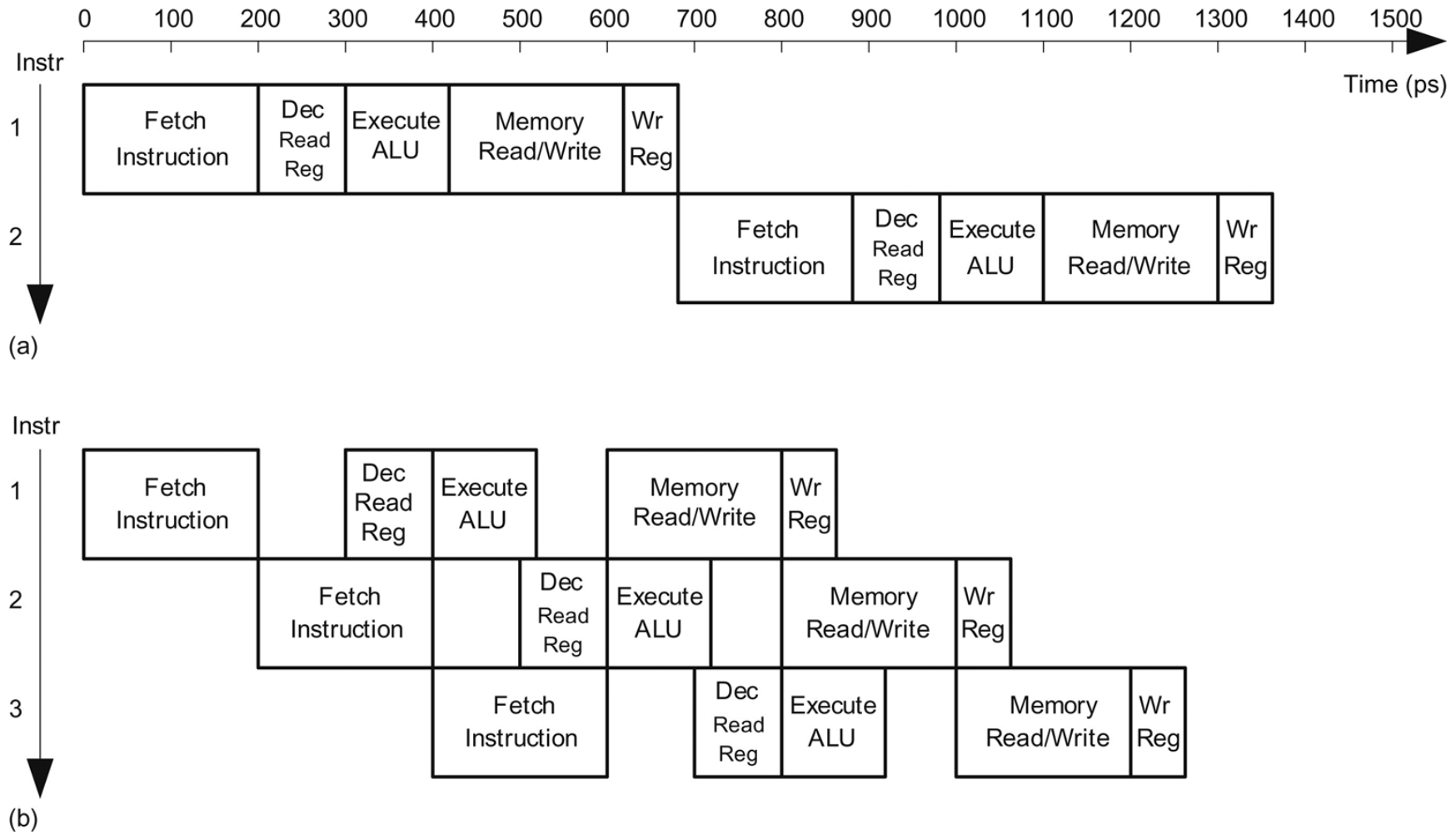


Figure 7.43 Abstract view of pipeline in operation

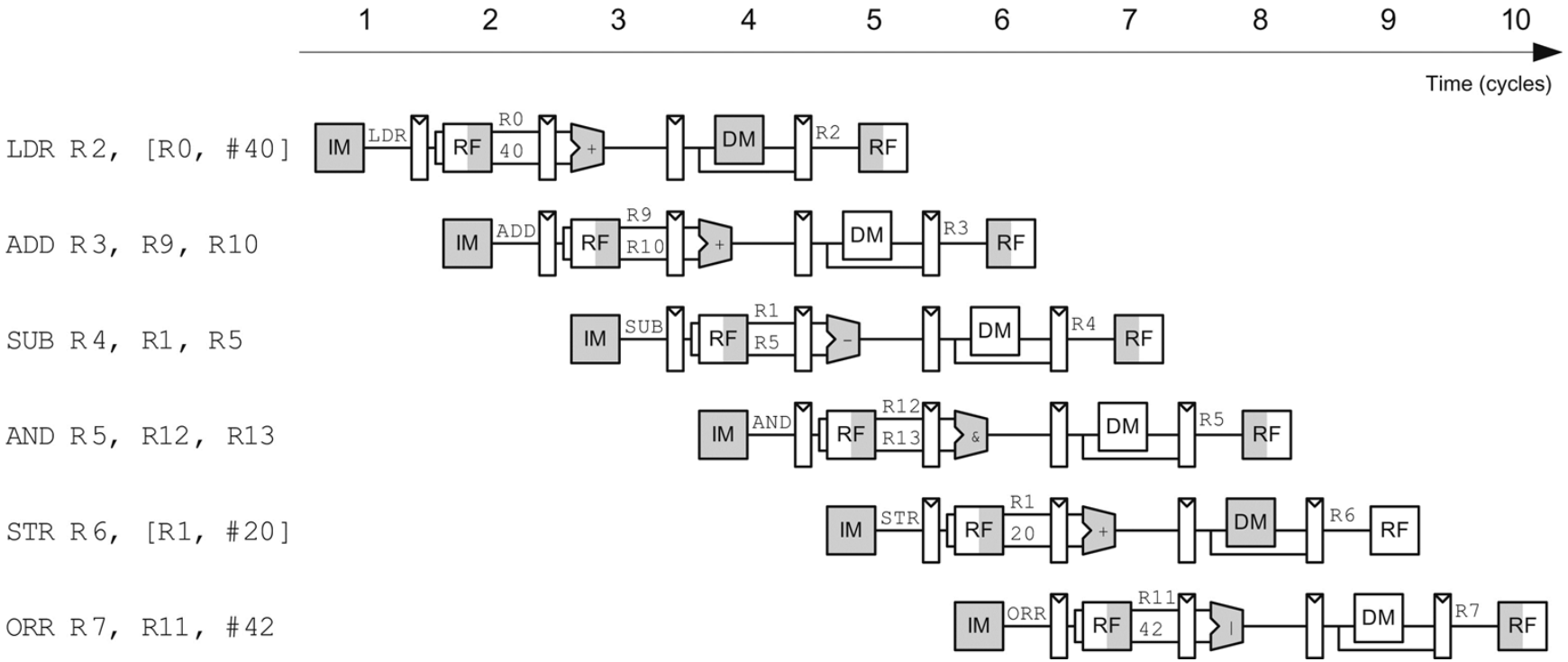


Figure 7.44 Datapaths: (a) single-cycle and (b) pipelined

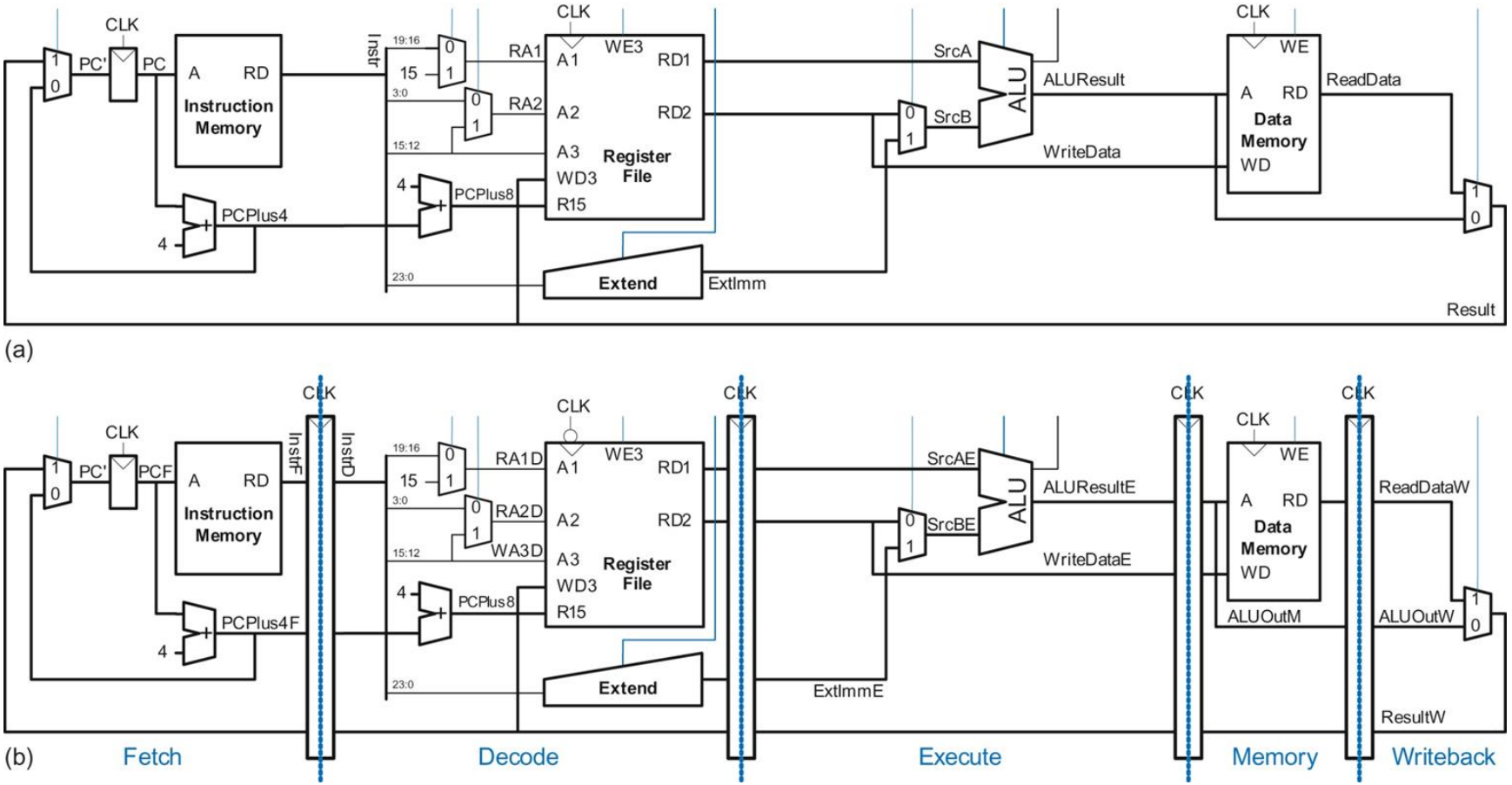


Figure 7.45 Corrected pipelined datapath

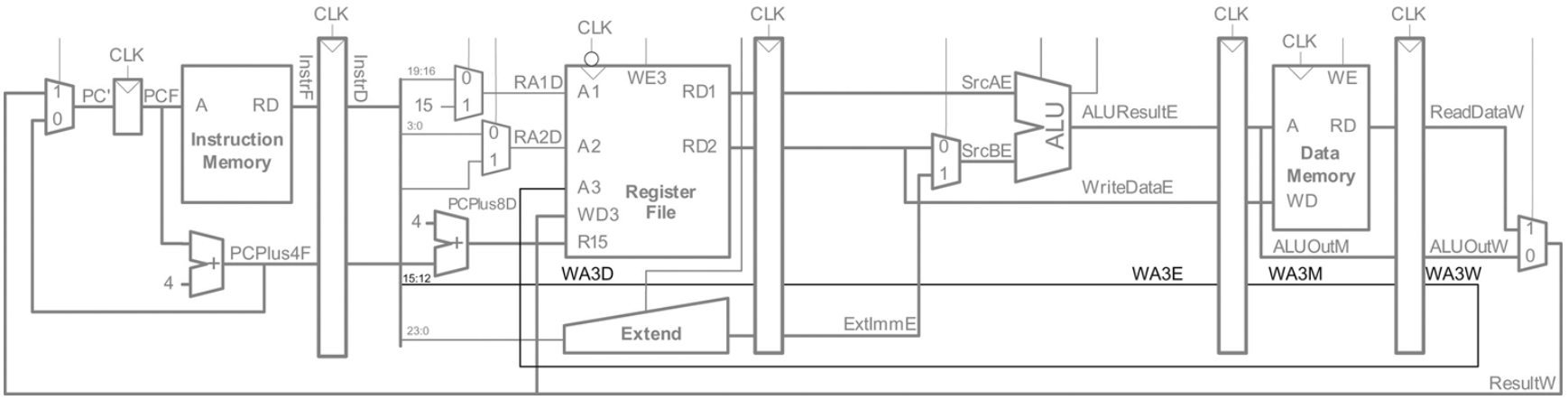


Figure 7.46 Optimized PC logic eliminating a register and adder

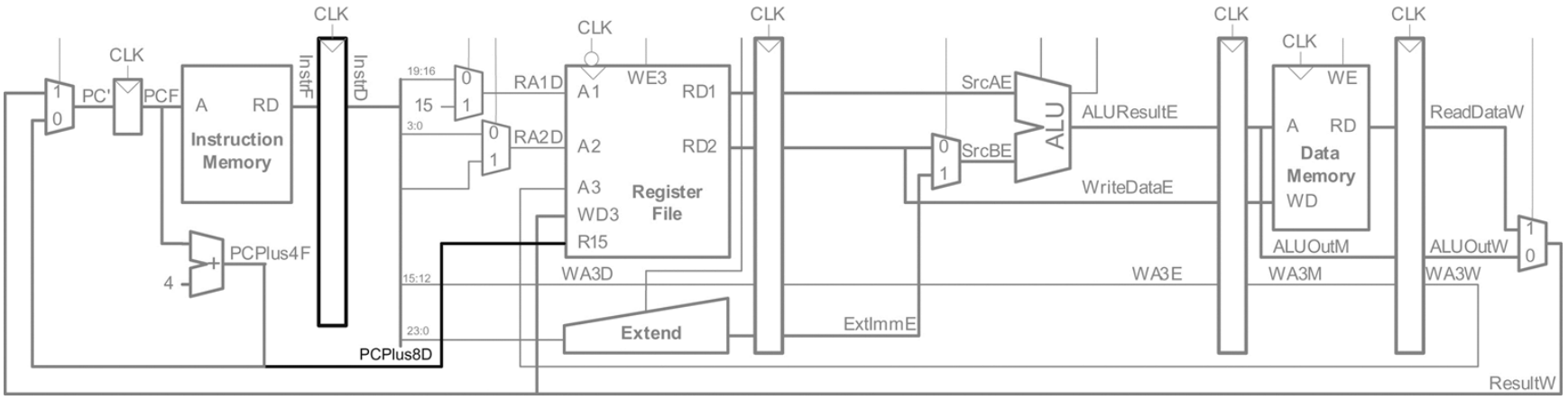


Figure 7.47 Pipelined processor with control

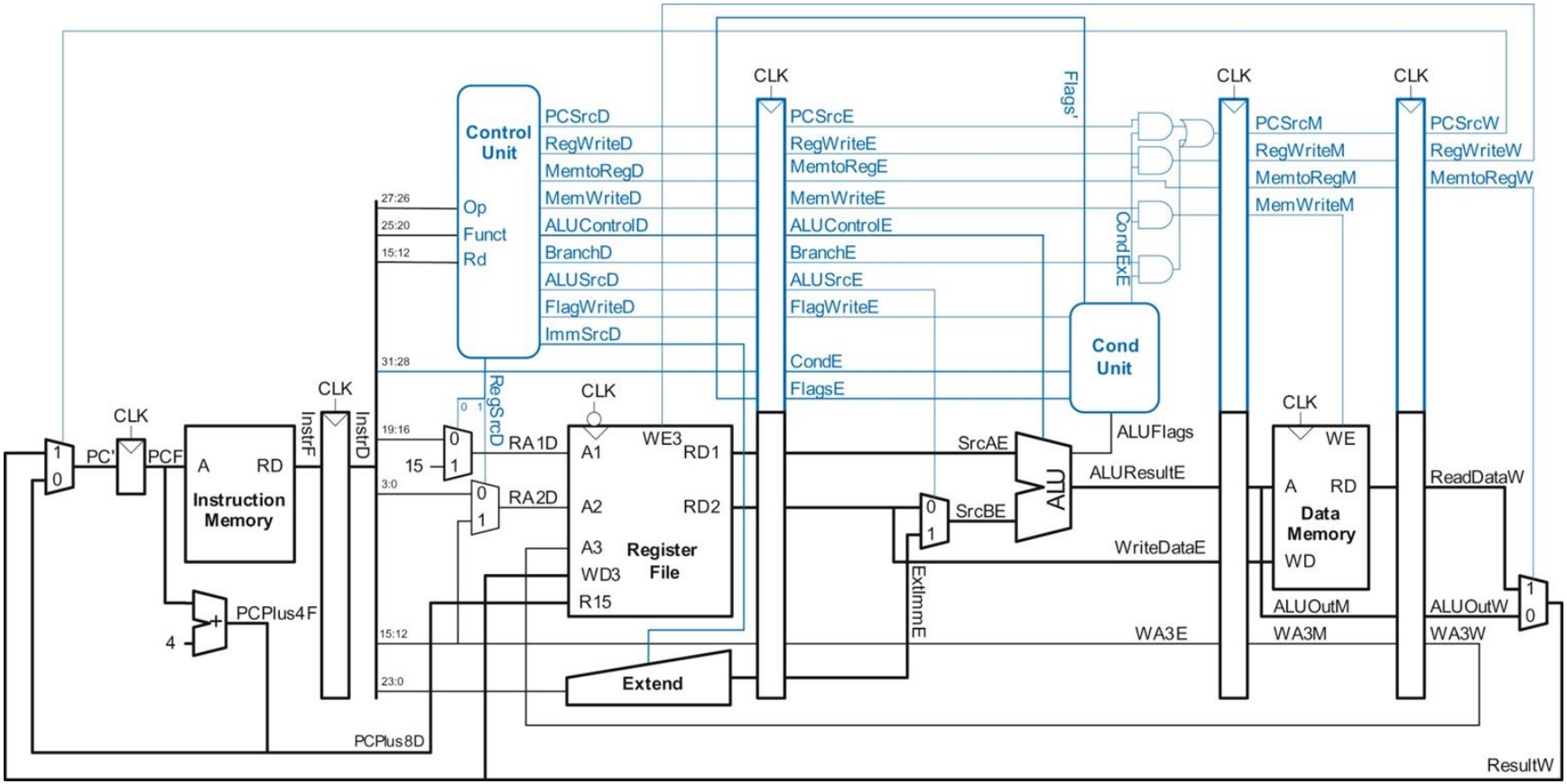


Figure 7.48 Abstract pipeline diagram illustrating hazards

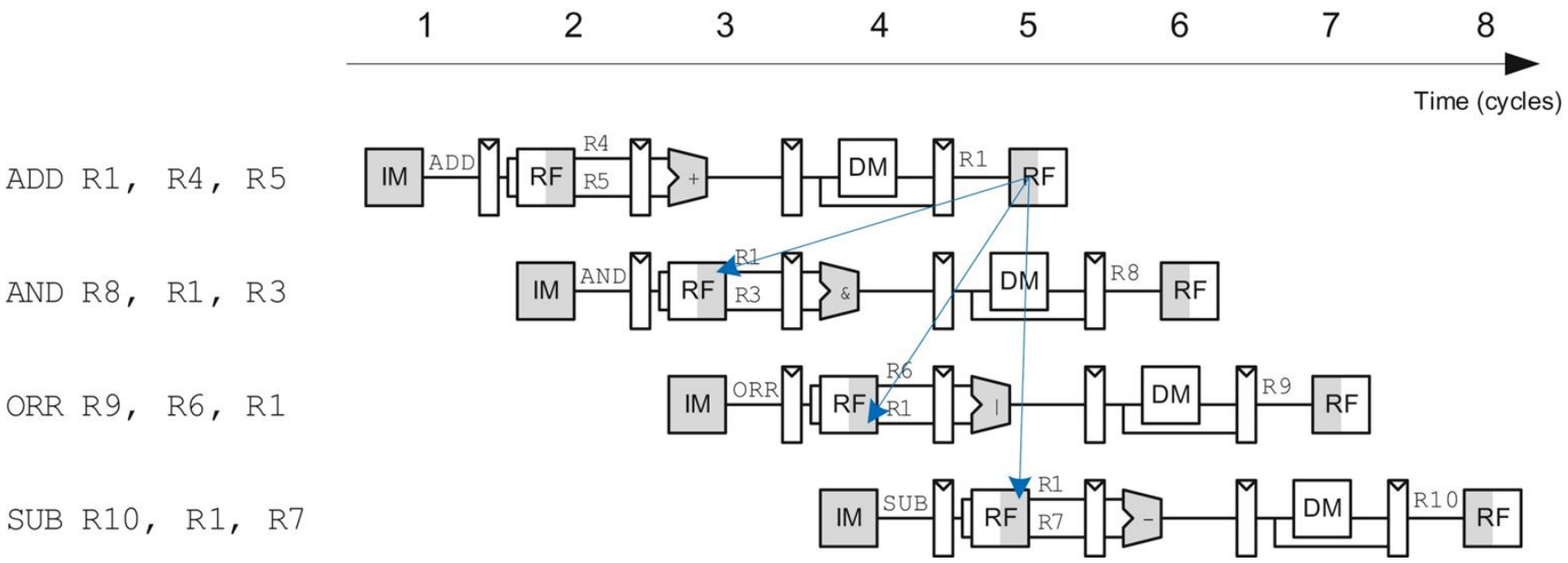


Figure 7.49 Solving data hazard with NOP

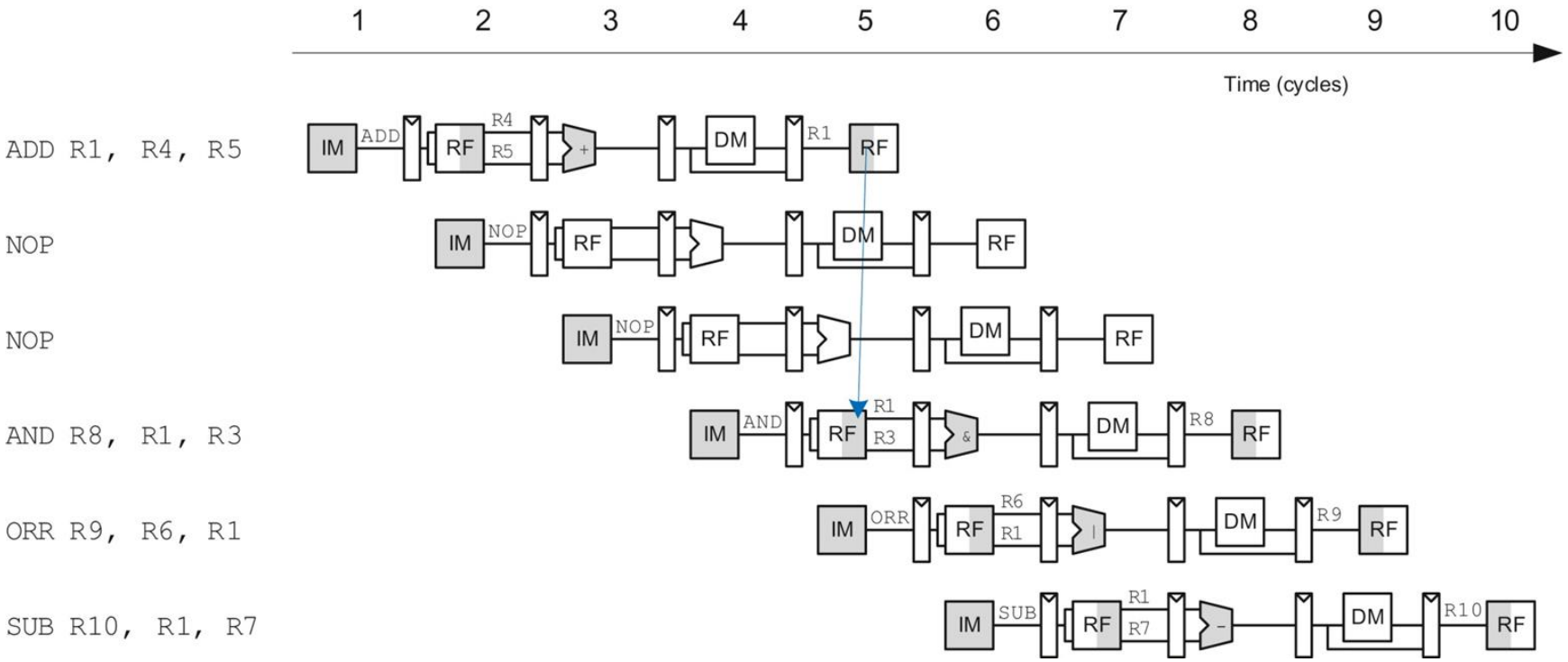


Figure 7.50 Abstract pipeline diagram illustrating forwarding

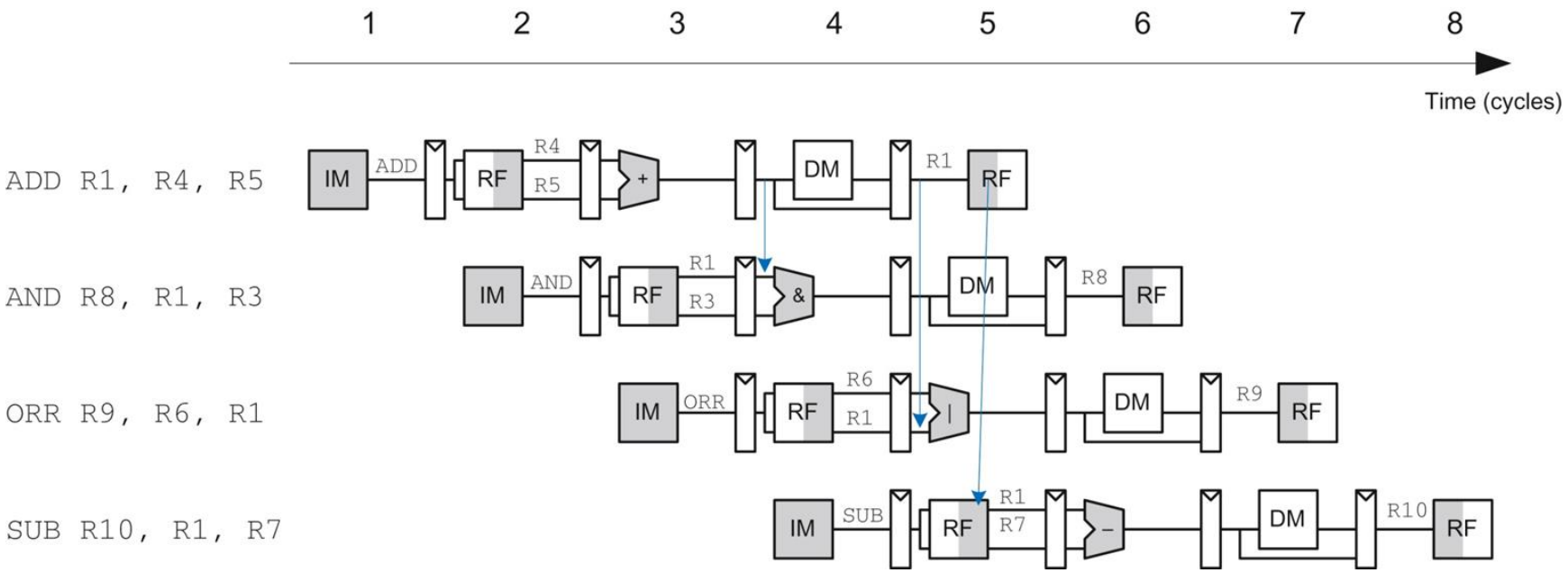


Figure 7.51 Pipelined processor with forwarding to solve hazards

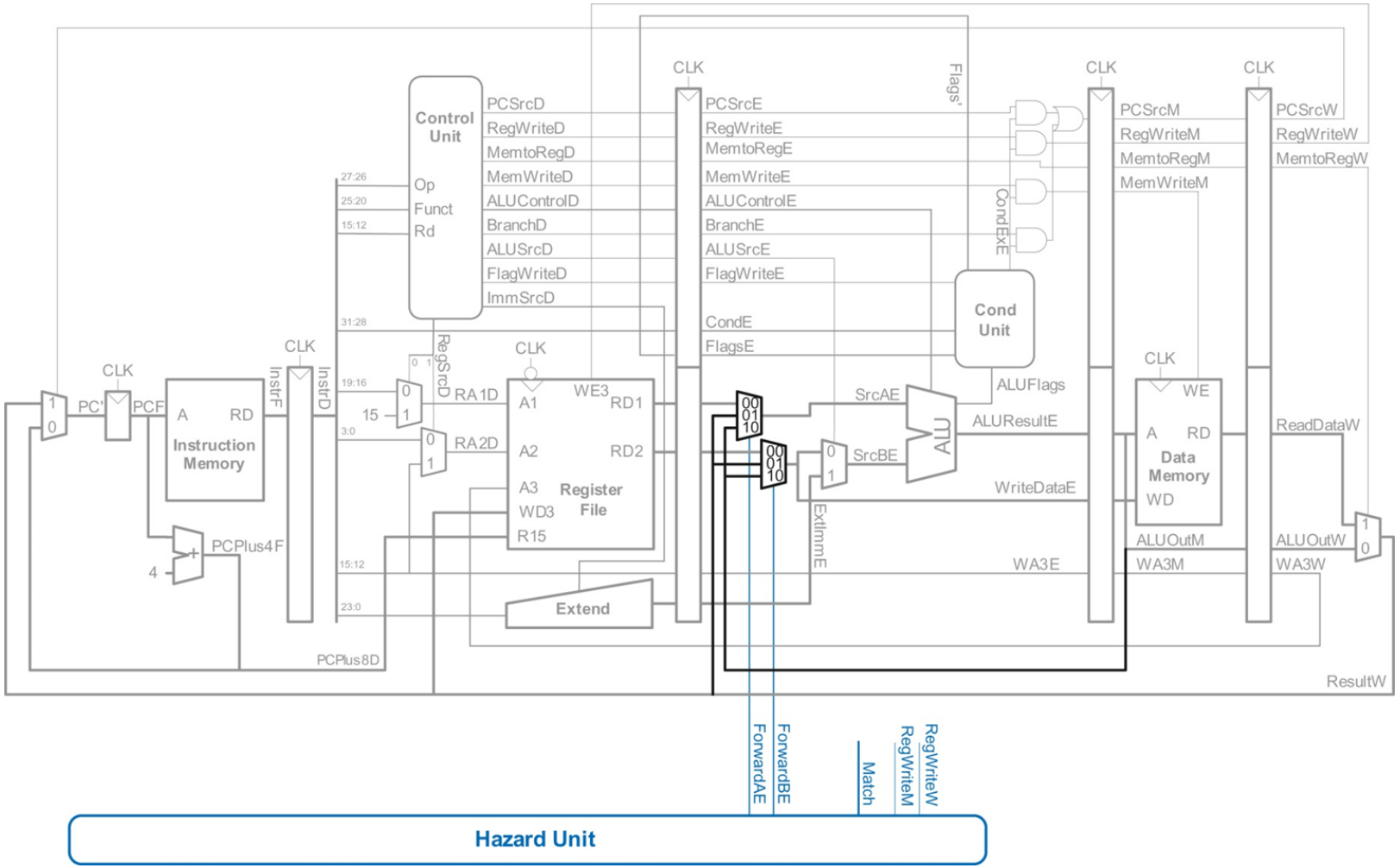


Figure 7.52 Abstract pipeline diagram illustrating trouble forwarding from LDR

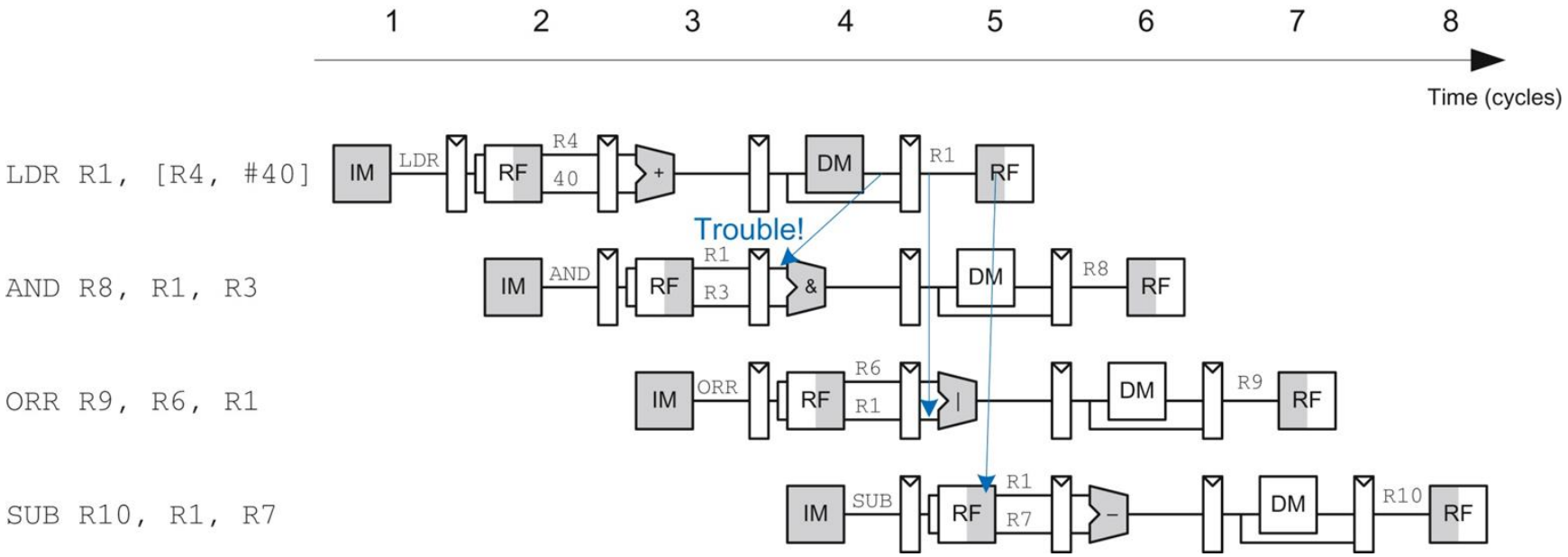


Figure 7.53 Abstract pipeline diagram illustrating stall to solve hazards

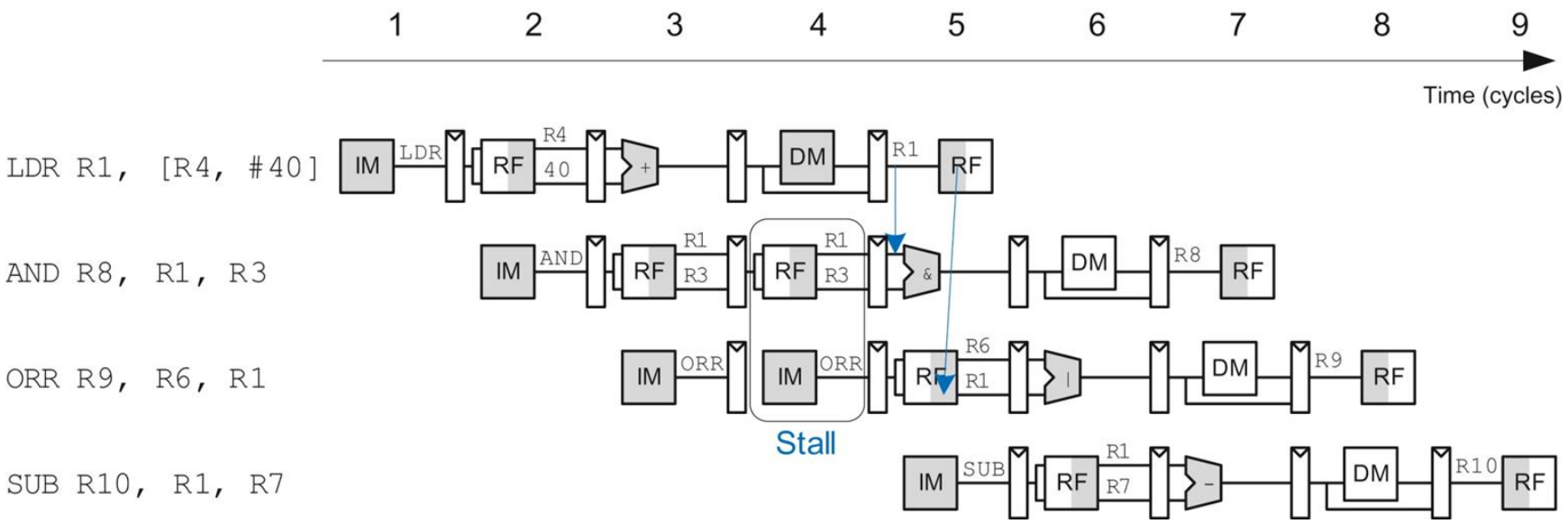


Figure 7.54 Pipelined processor with stalls to solve **LDR** data hazard

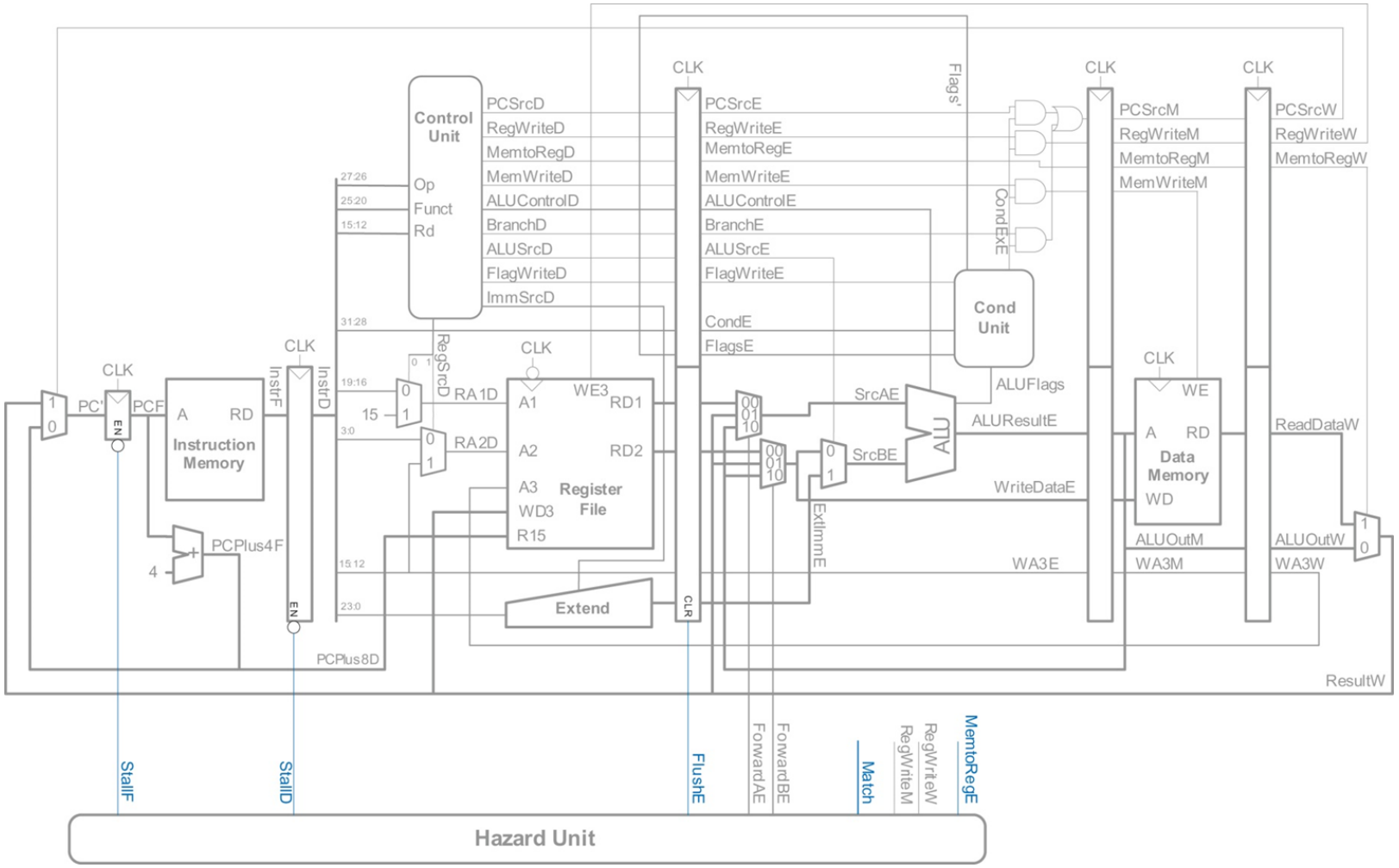


Figure 7.55 Abstract pipeline diagram illustrating flushing when a branch is taken

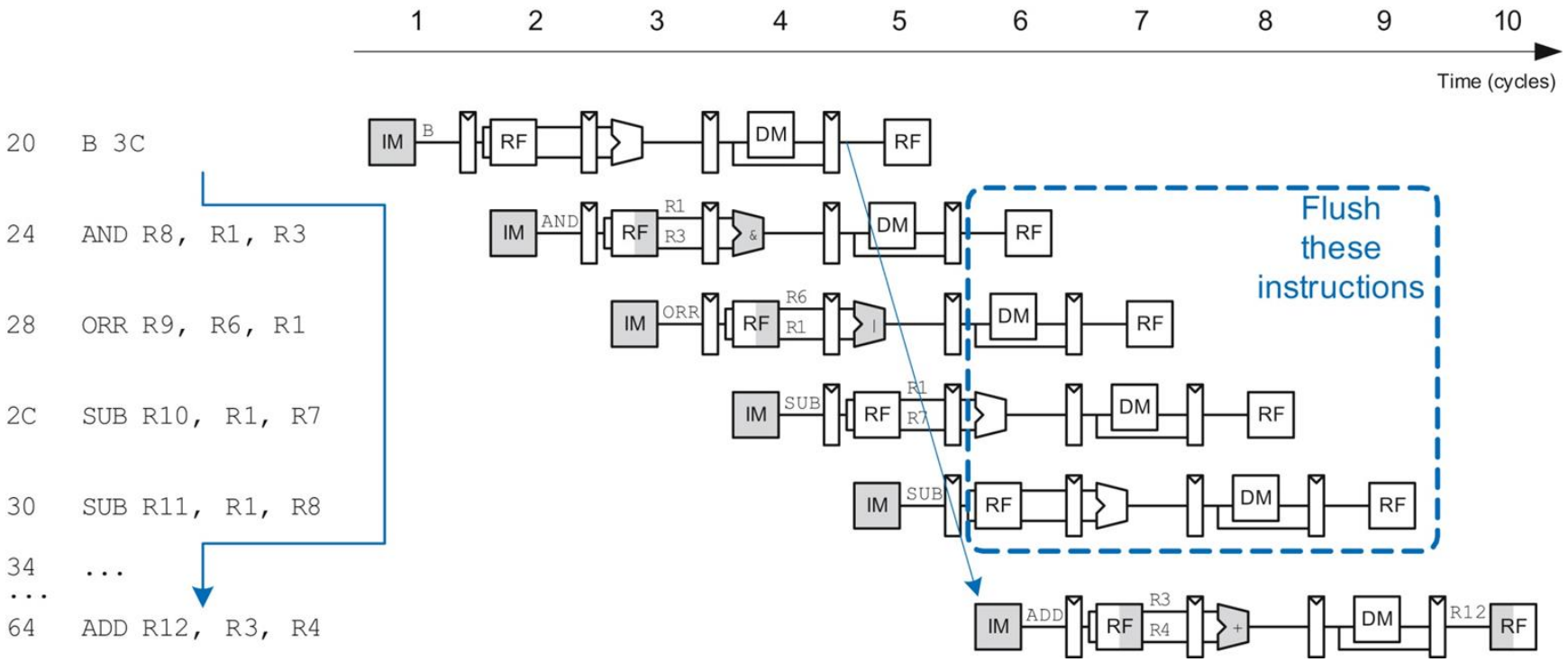


Figure 7.56 Abstract pipeline diagram illustrating earlier branch decision

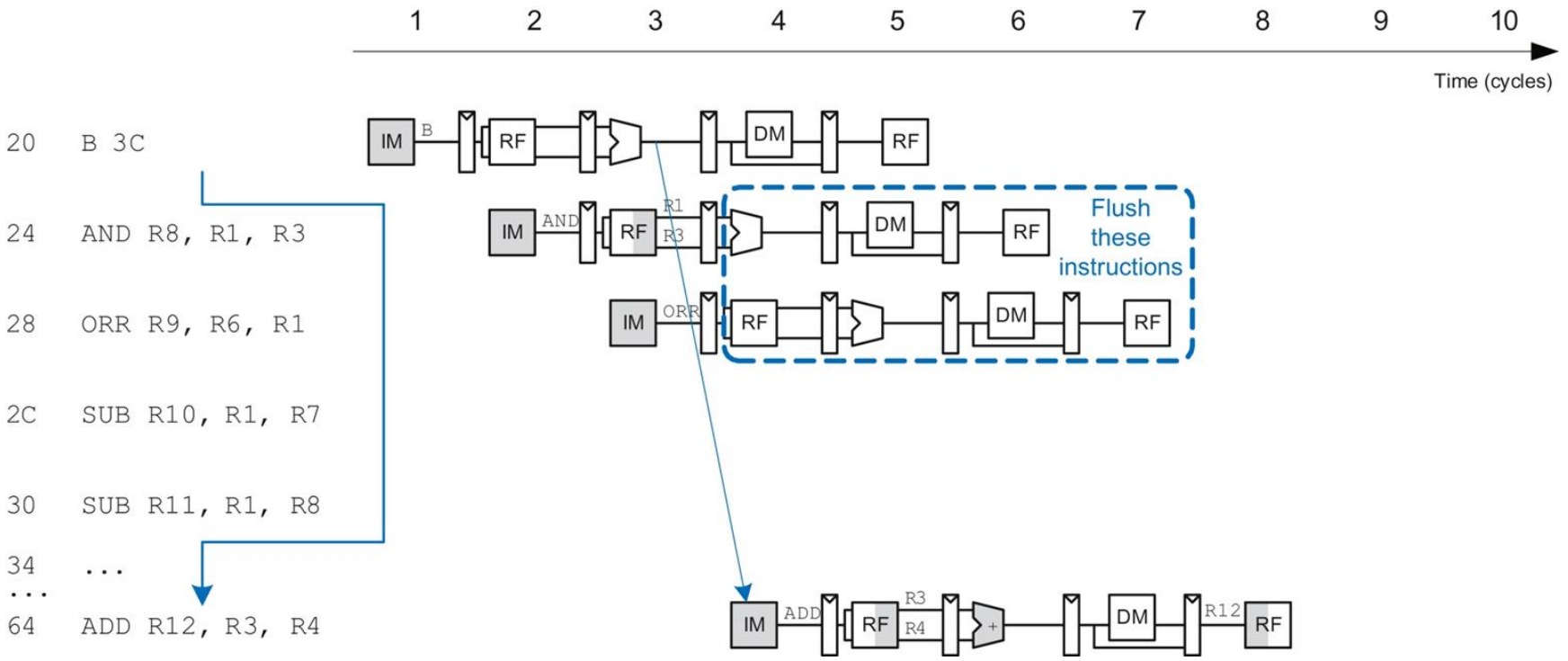


Figure 7.57 Pipelined processor handling branch control hazard

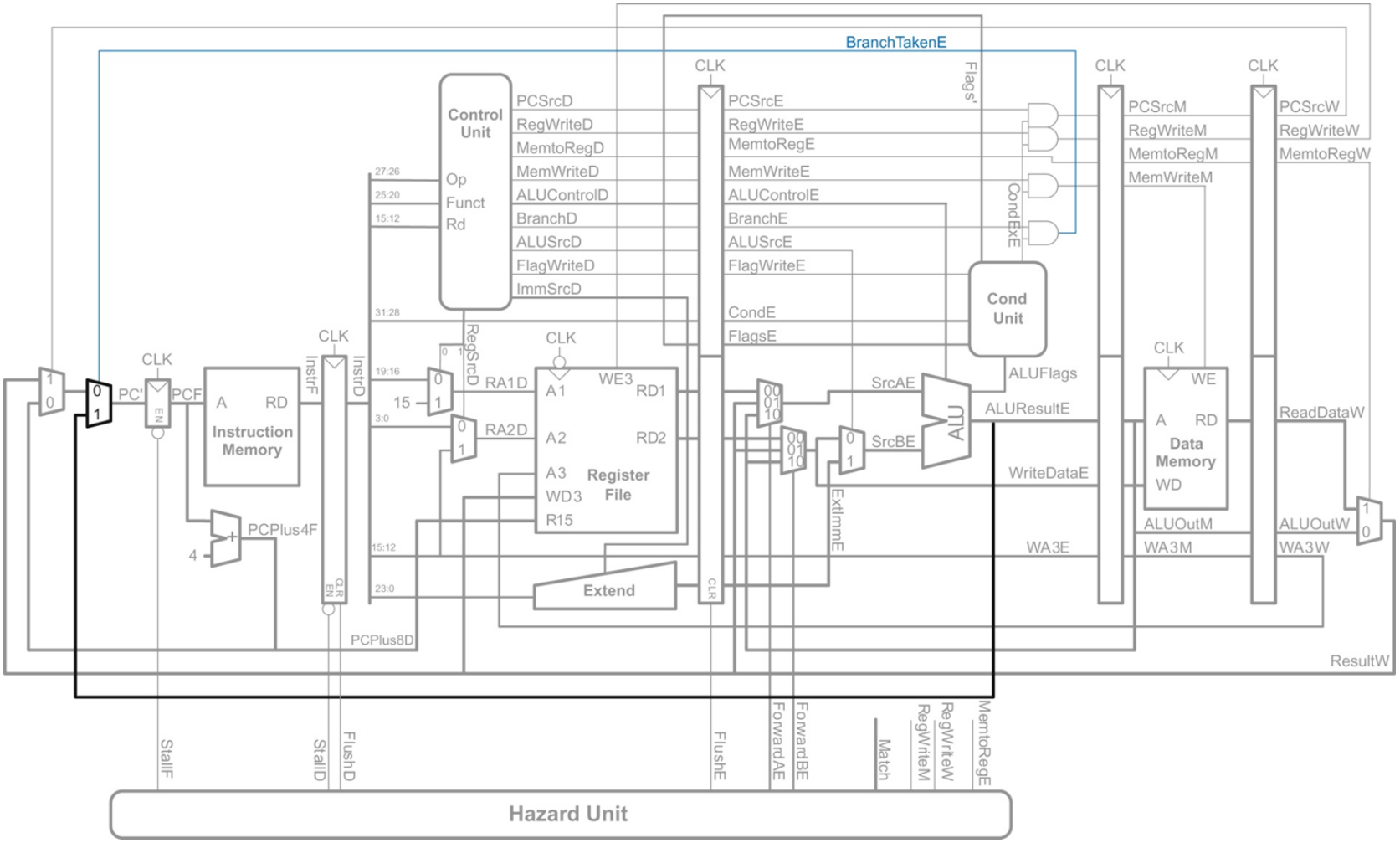


Figure 7.58 Pipelined processor with full hazard handling

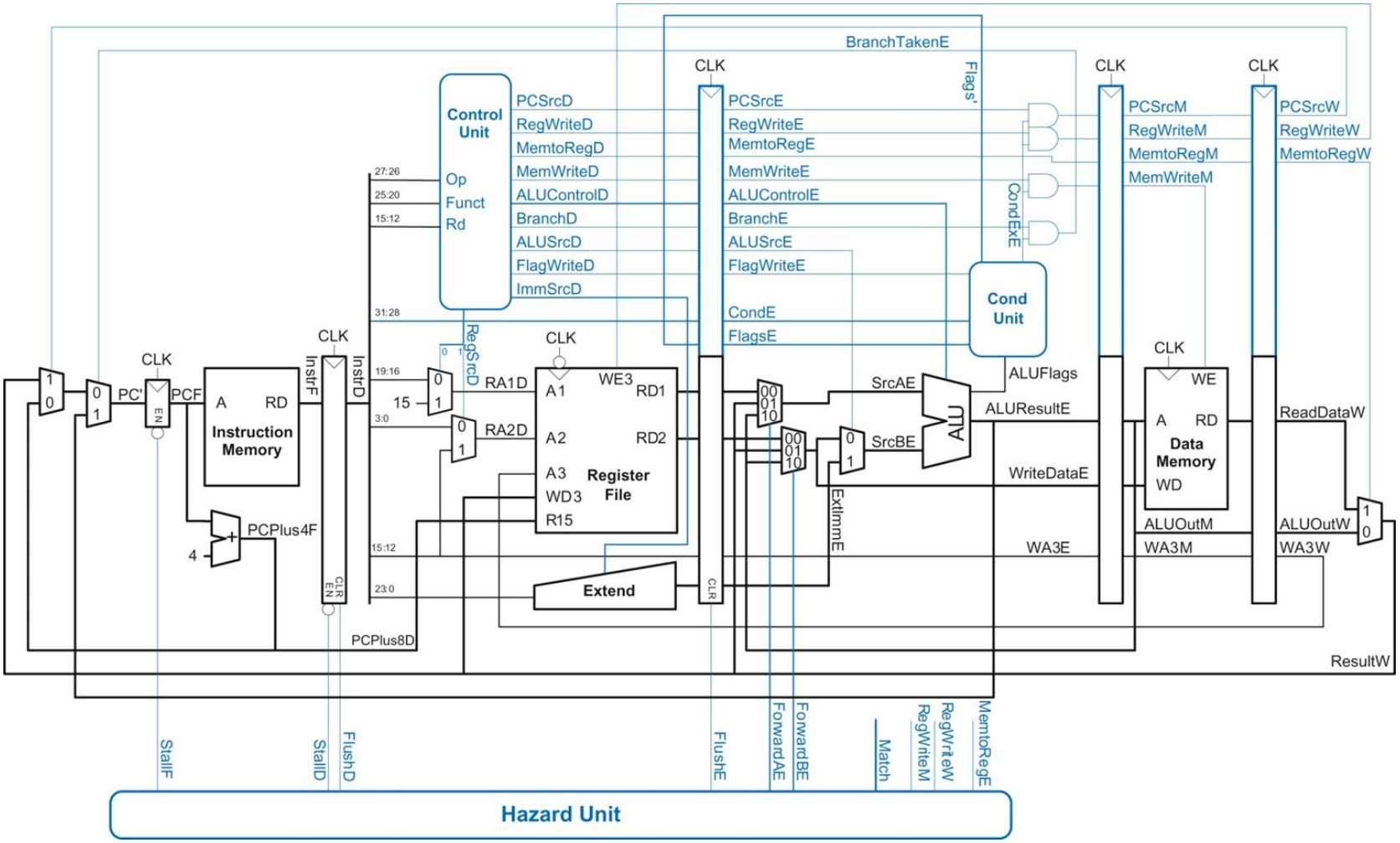


Figure 7.59 Single-cycle processor interfaced to external memory

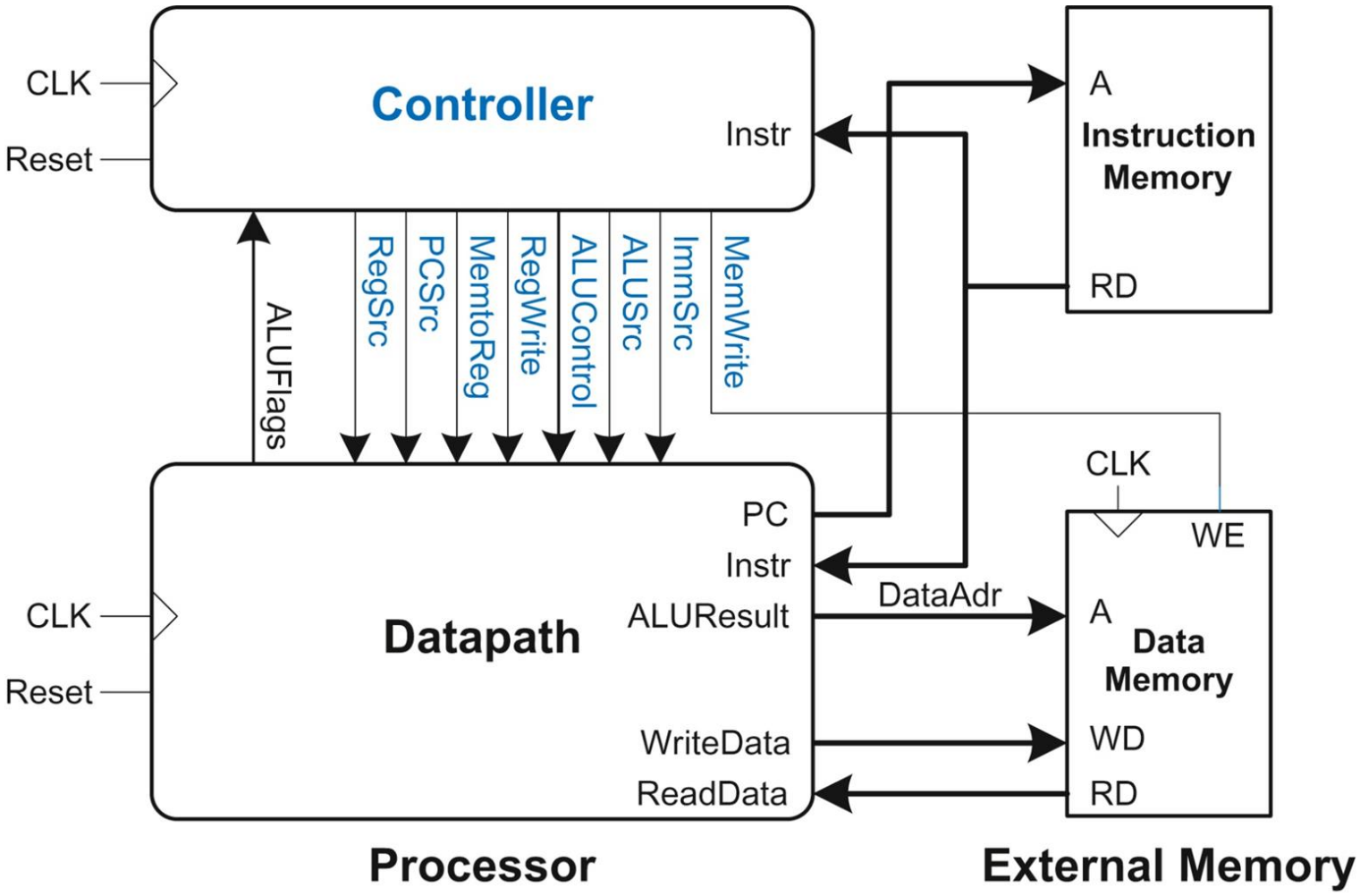


Figure 7.60 Assembly and machine code for test program

ADDR	PROGRAM	; COMMENTS	BINARY MACHINE CODE	HEX CODE
00 MAIN	SUB R0, R15, R15	; R0 = 0	1110 000 0010 0 1111 0000 0000 0000 1111	E04F000F
04	ADD R2, R0, #5	; R2 = 5	1110 001 0100 0 0000 0010 0000 0000 0101	E2802005
08	ADD R3, R0, #12	; R3 = 12	1110 001 0100 0 0000 0011 0000 0000 1100	E280300C
0C	SUB R7, R3, #9	; R7 = 3	1110 001 0010 0 0011 0111 0000 0000 1001	E2437009
10	ORR R4, R7, R2	; R4 = 3 OR 5 = 7	1110 000 1100 0 0111 0100 0000 0000 0010	E1874002
14	AND R5, R3, R4	; R5 = 12 AND 7 = 4	1110 000 0000 0 0011 0101 0000 0000 0100	E0035004
18	ADD R5, R5, R4	; R5 = 4 + 7 = 11	1110 000 0100 0 0101 0101 0000 0000 0100	E0855004
1C	SUBS R8, R5, R7	; R8 = 11 - 3 = 8, set Flags	1110 000 0010 1 0101 1000 0000 0000 0111	E0558007
20	BEQ END	; shouldn't be taken	0000 1010 0000 0000 0000 0000 0000 1100	0A00000C
24	SUBS R8, R3, R4	; R8 = 12 - 7 = 5	1110 000 0010 1 0011 1000 0000 0000 0100	E0538004
28	BGE AROUND	; should be taken	1010 1010 0000 0000 0000 0000 0000 0000	AA000000
2C	ADD R5, R0, #0	; should be skipped	1110 001 0100 0 0000 0101 0000 0000 0000	E2805000
30 AROUND	SUBS R8, R7, R2	; R8 = 3 - 5 = -2, set Flags	1110 000 0010 1 0111 1000 0000 0000 0010	E0578002
34	ADDLT R7, R5, #1	; R7 = 11 + 1 = 12	1011 001 0100 0 0101 0111 0000 0000 0001	B2857001
38	SUB R7, R7, R2	; R7 = 12 - 5 = 7	1110 000 0010 0 0111 0111 0000 0000 0010	E0477002
3C	STR R7, [R3, #84]	; mem[12+84] = 7	1110 010 1100 0 0011 0111 0000 0101 0100	E5837054
40	LDR R2, [R0, #96]	; R2 = mem[96] = 7	1110 010 1100 1 0000 0010 0000 0110 0000	E5902060
44	ADD R15, R15, R0	; PC = PC+8 (skips next)	1110 000 0100 0 1111 1111 0000 0000 0000	E08FF000
48	ADD R2, R0, #14	; shouldn't happen	1110 001 0100 0 0000 0010 0000 0000 0001	E280200E
4C	B END	; always taken	1110 1010 0000 0000 0000 0000 0000 0001	EA000001
50	ADD R2, R0, #13	; shouldn't happen	1110 001 0100 0 0000 0010 0000 0000 0001	E280200D
54	ADD R2, R0, #10	; shouldn't happen	1110 001 0100 0 0000 0010 0000 0000 0001	E280200A
58 END	STR R2, [R0, #100]	; mem[100] = 7	1110 010 1100 0 0000 0010 0000 0101 0100	E5802064

Figure 7.61 Cycle time and instruction time vs. the number of pipeline stages

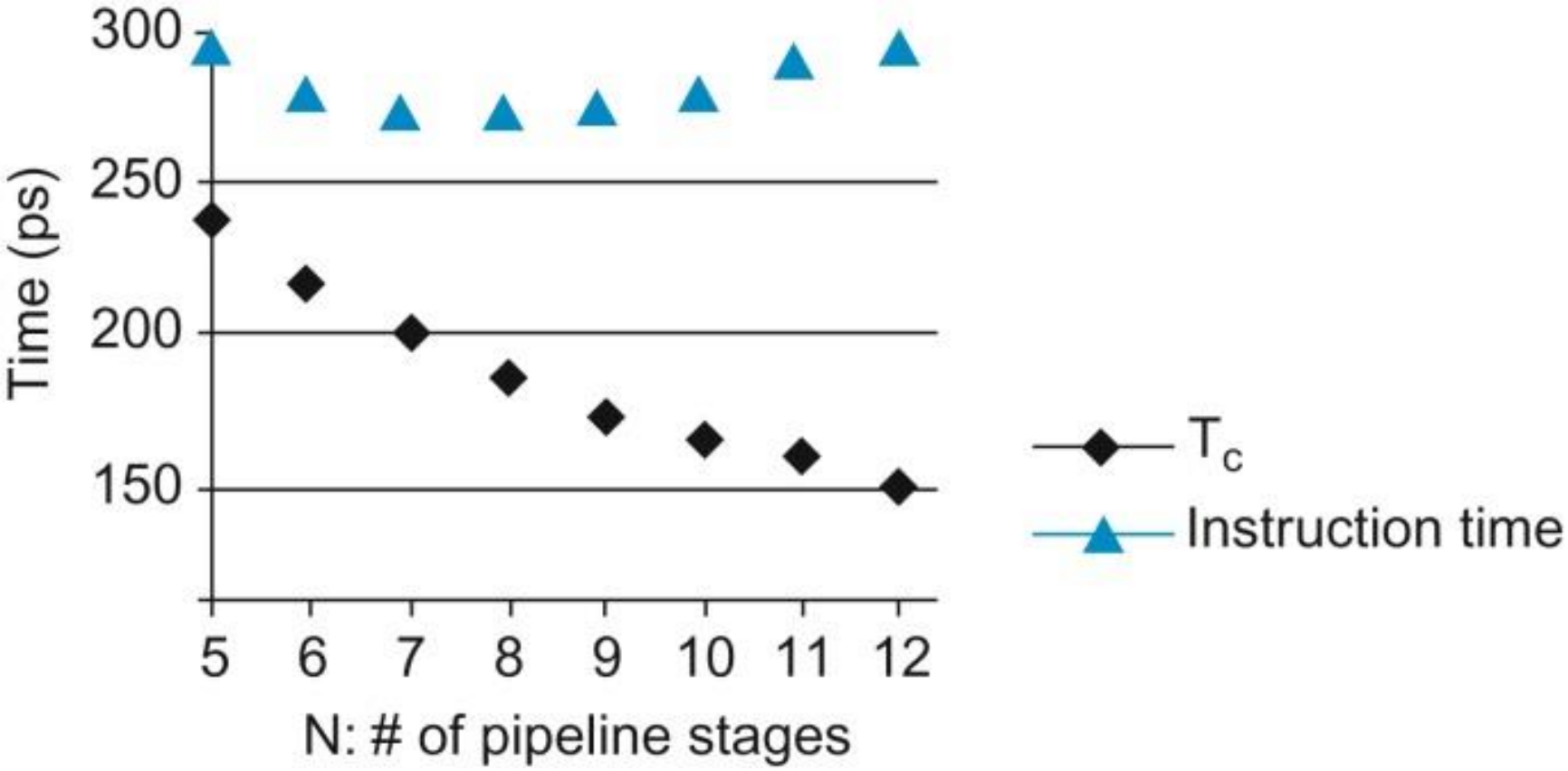


Figure 7.62 Two-bit branch predictor state transition diagram

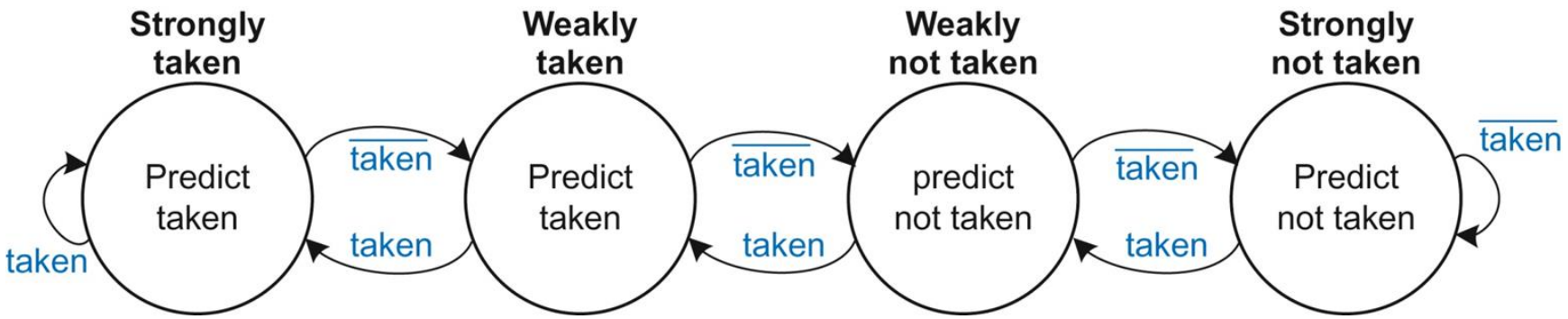


Figure 7.63 Superscalar datapath

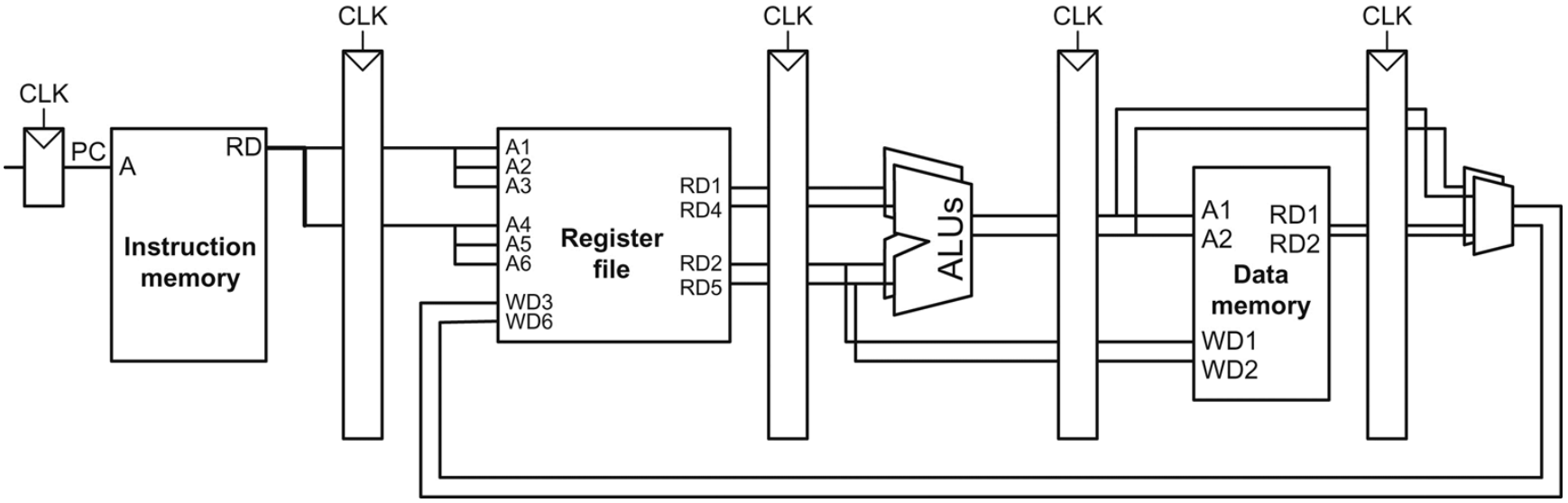


Figure 7.64 Abstract view of a superscalar pipeline in operation

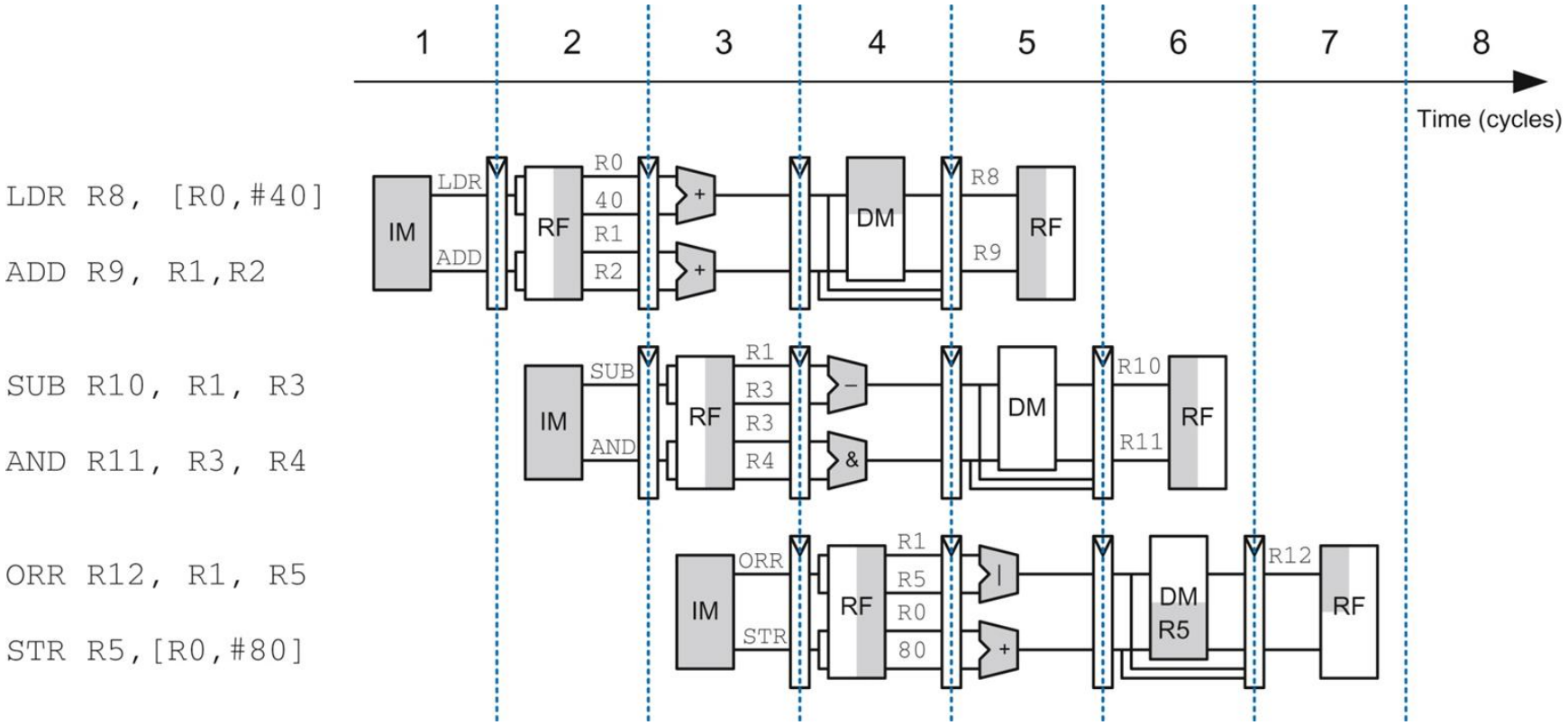


Figure 7.65 Program with data dependencies

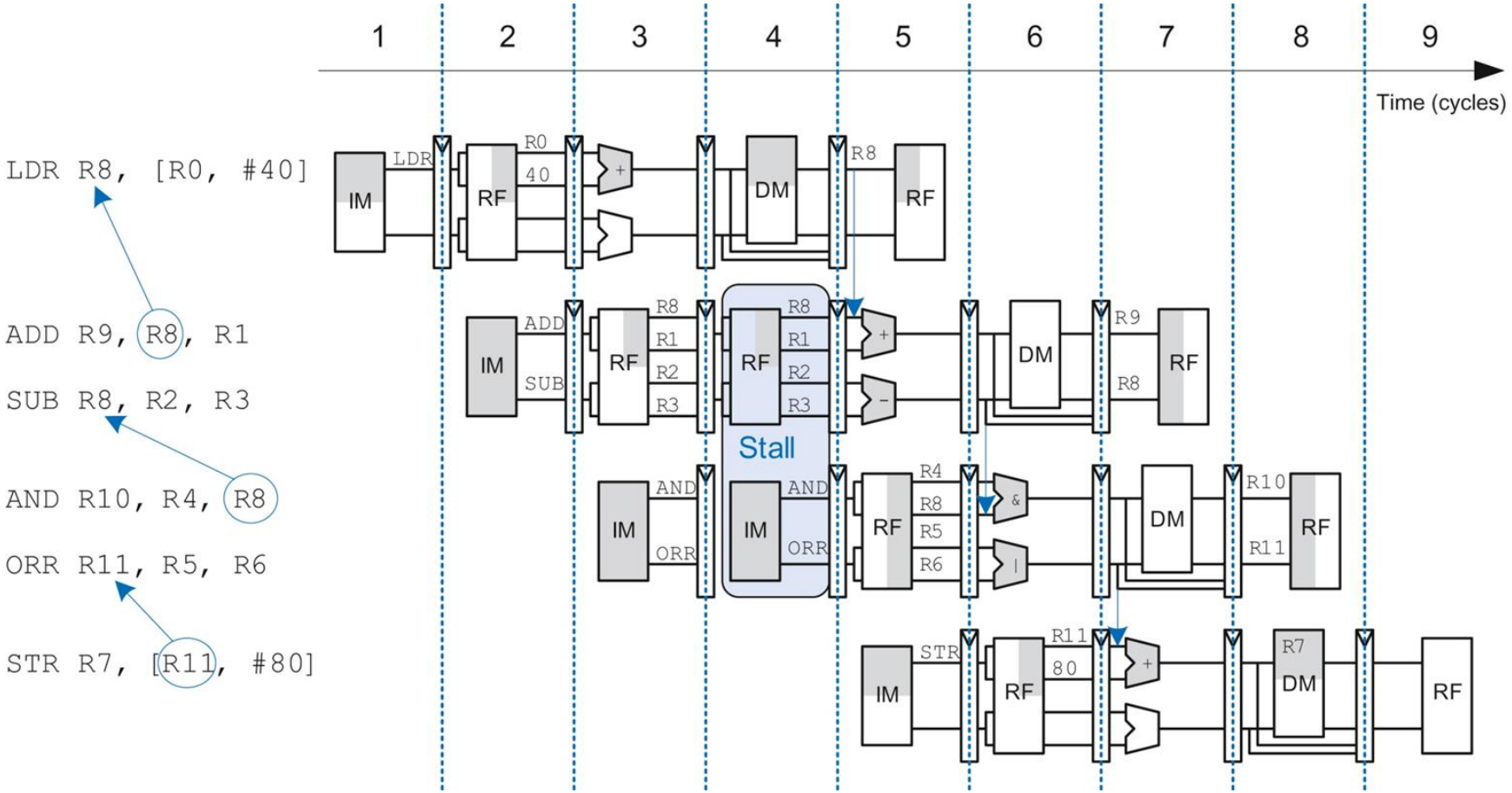


Figure 7.66 Out-of-order execution of a program with dependencies

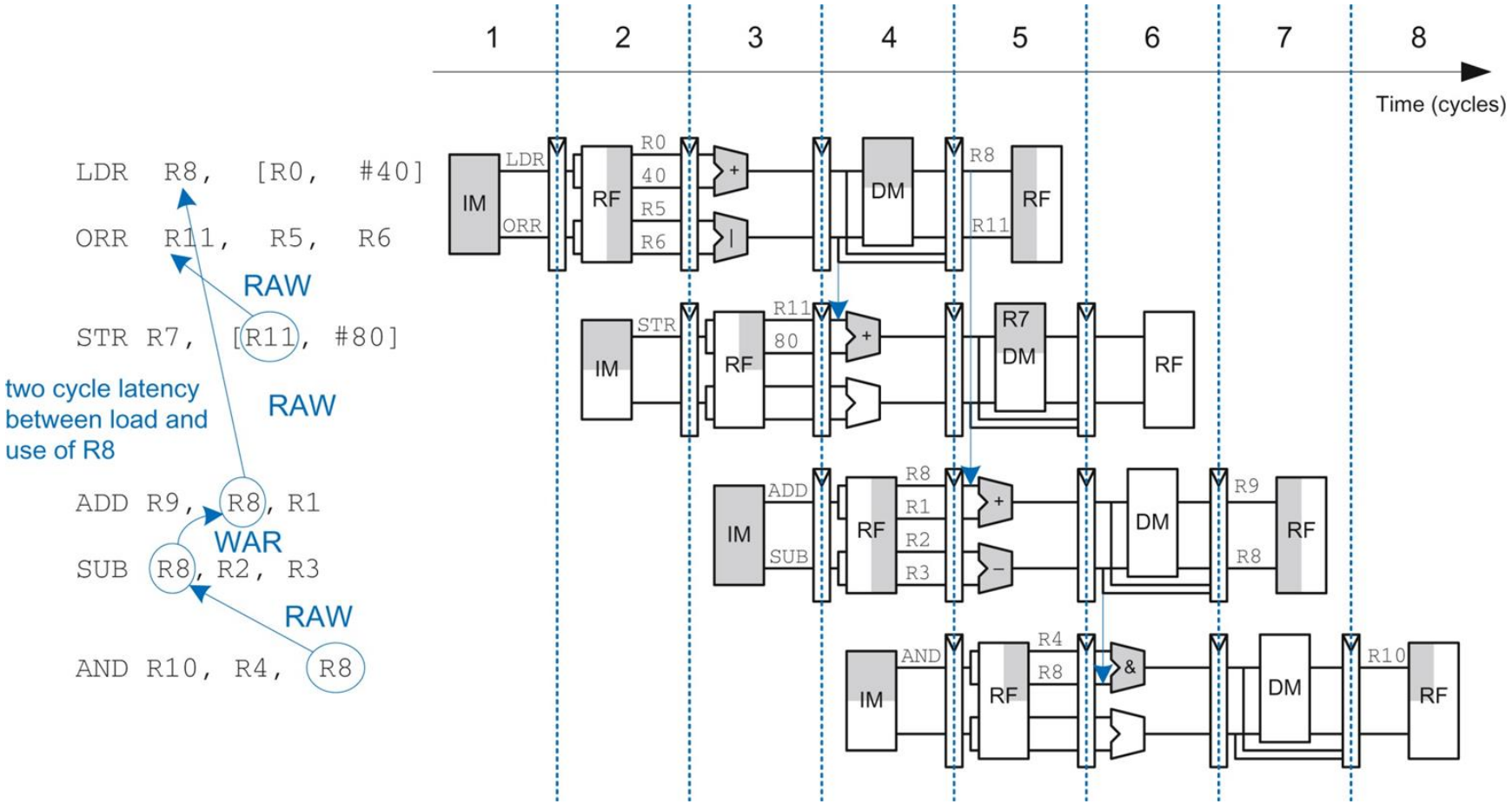


Figure 7.67 Out-of-order execution of a program using register renaming

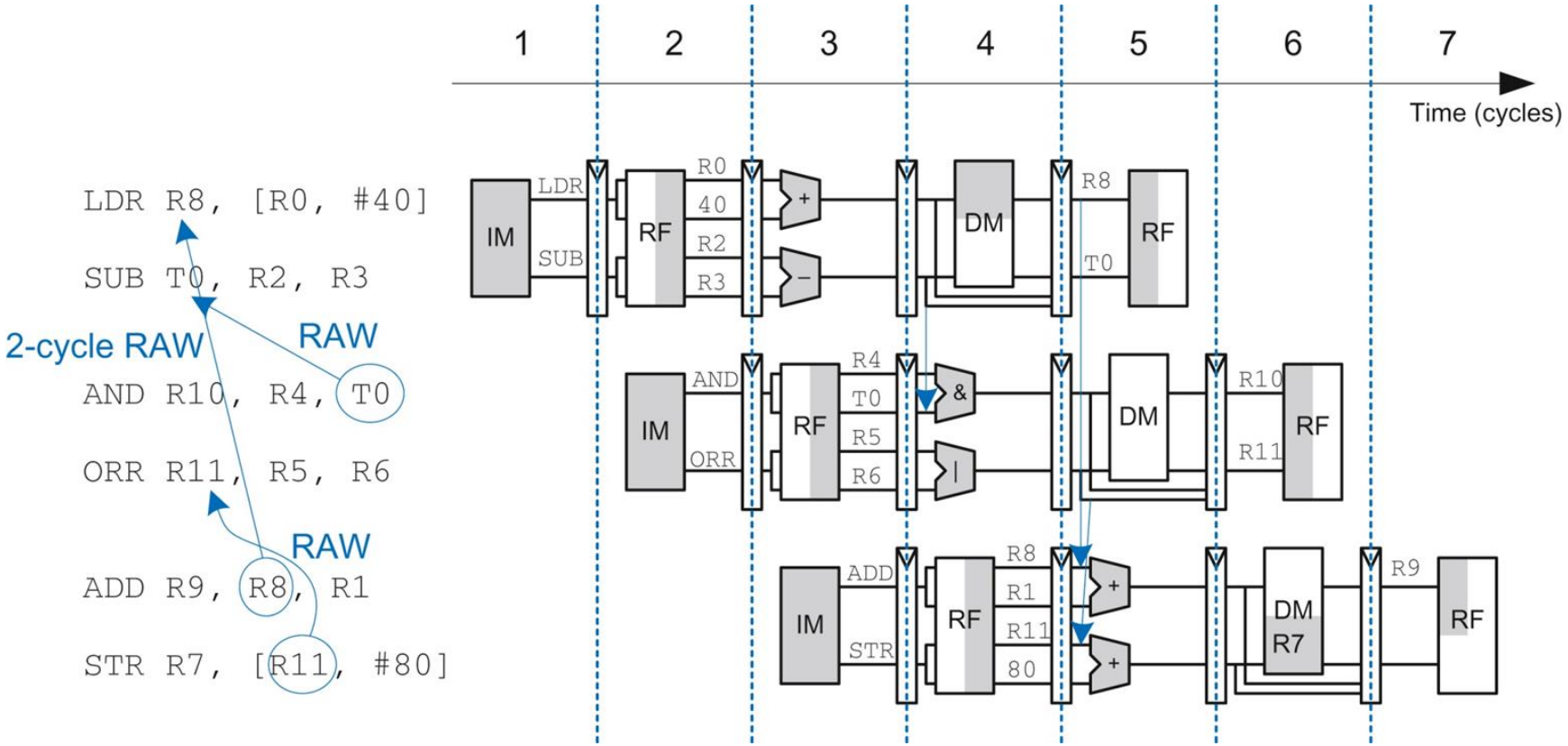


Figure 7.68 ARM1 die photograph. Reproduced with permission from ARM. © 1985 ARM Ltd.

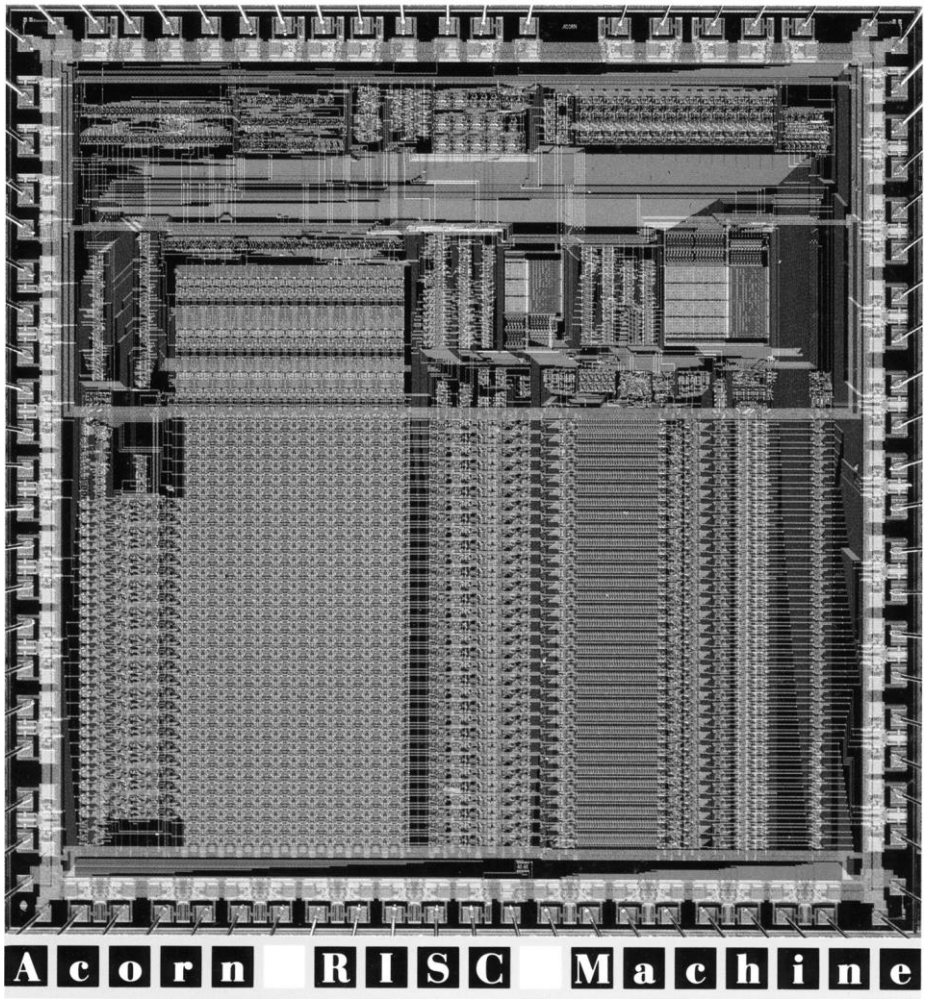


Figure 7.69 ARM7 block diagram. Reproduced with permission from ARM. © 1998 ARM Ltd.

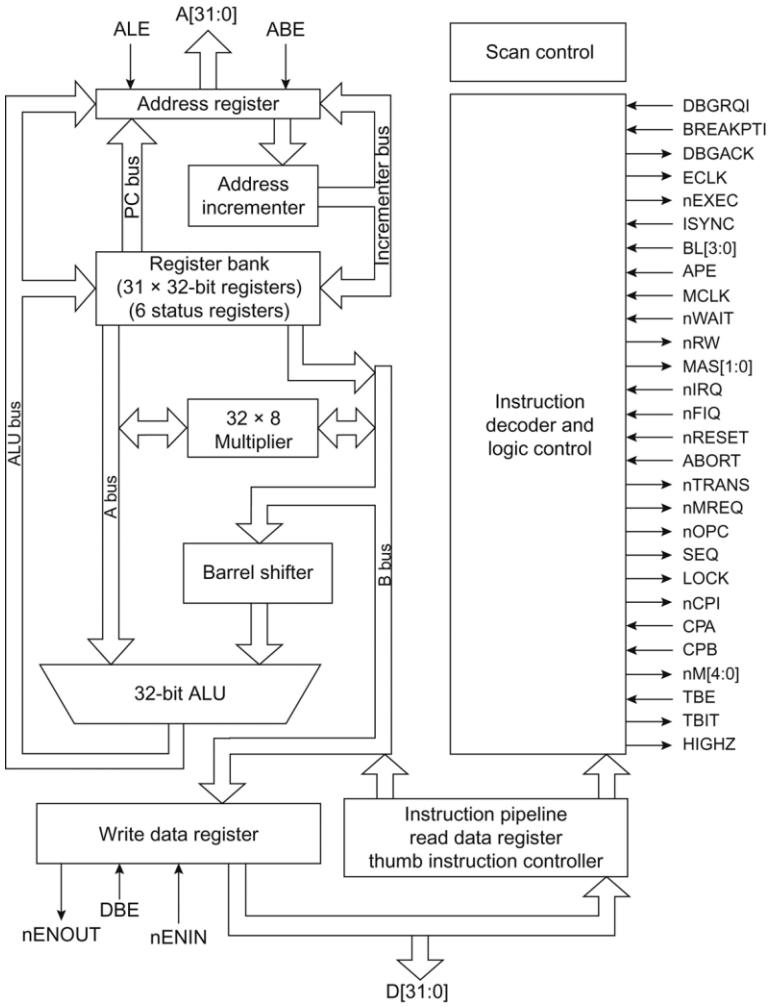


Figure 7.70 ARM9 block diagram. Reproduced with permission from the ARM9TDMI Technical Reference Manual. © 1999 ARM Ltd.

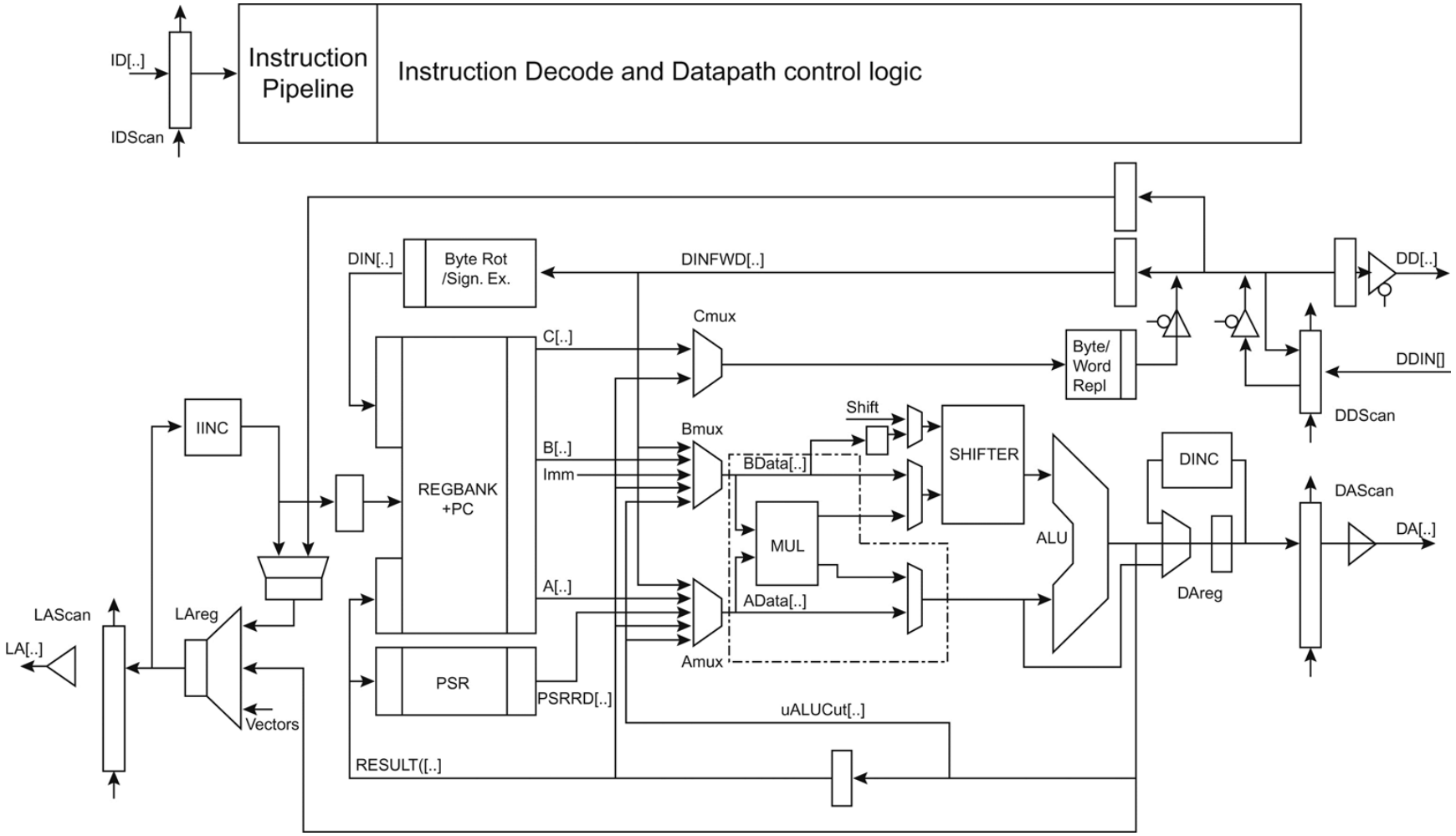


Figure 7.71 Cortex-A9 block diagram. This image has been sourced by the authors and does not imply ARM endorsement.

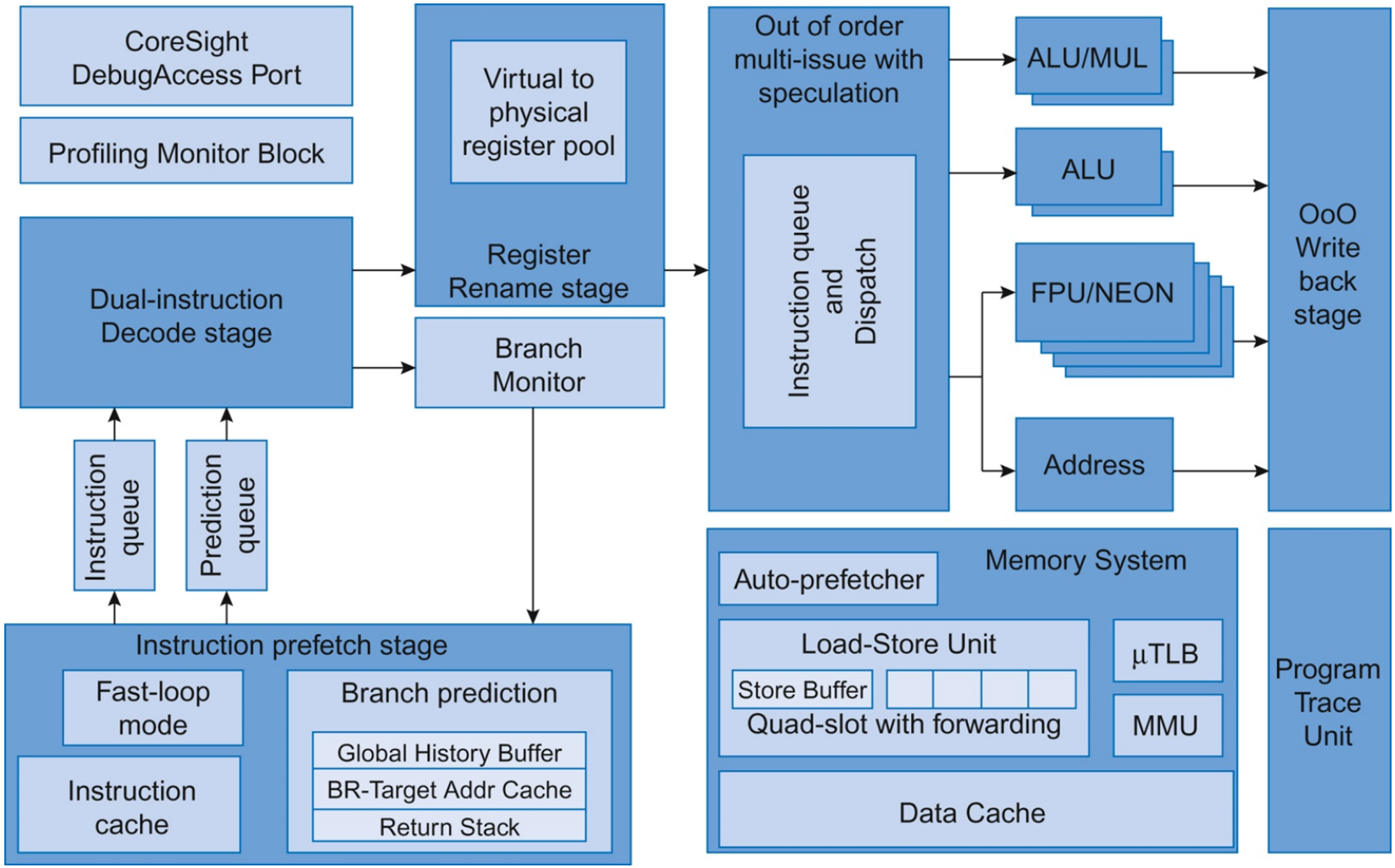


Figure 7.72 Cortex-A7 and -A15 block diagrams. This image has been sourced by the authors and does not imply ARM endorsement.

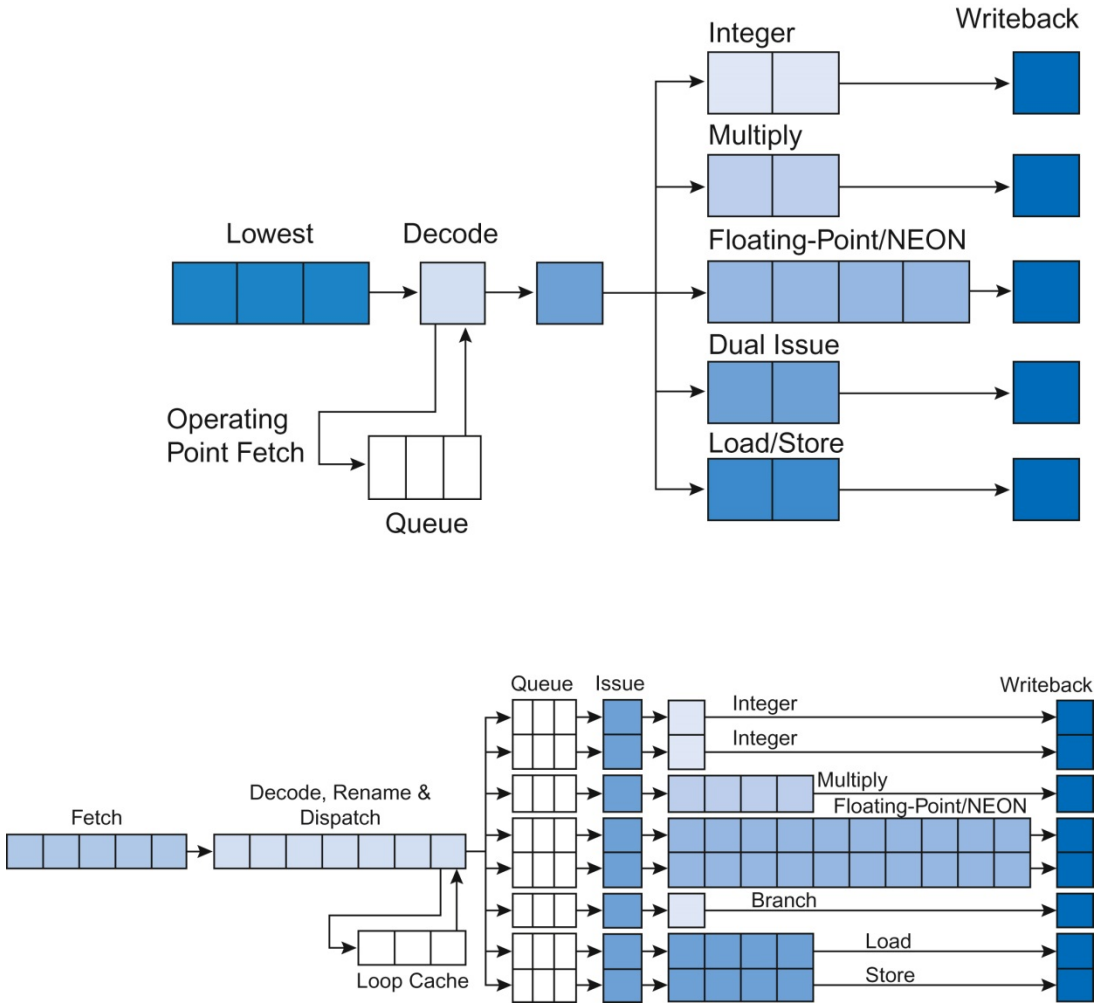


Figure u07-01