Chapter 7

Microarchitecture

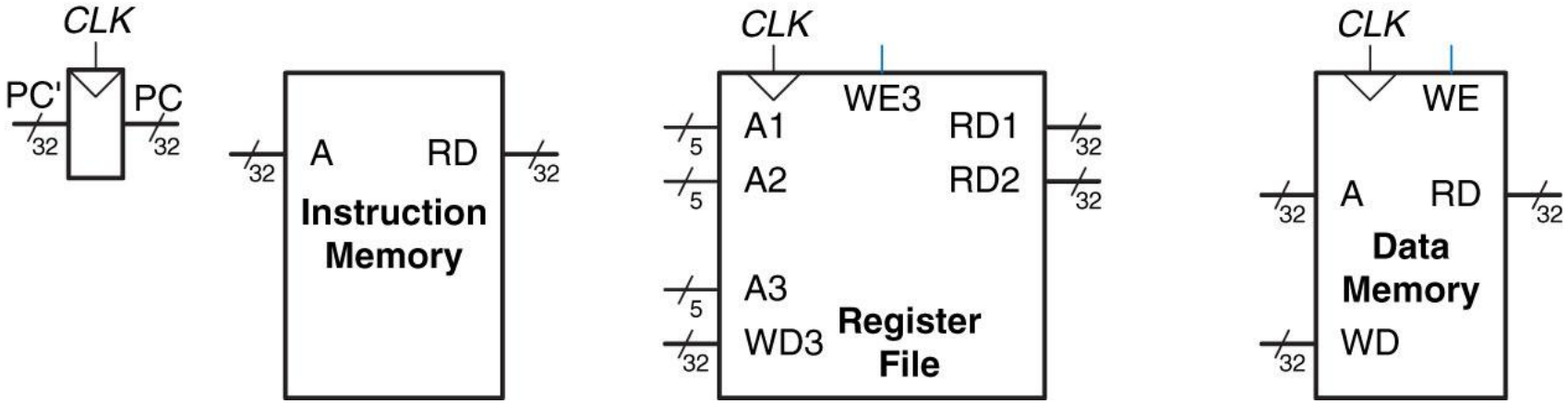


Figure 7.1 State elements of MIPS processor

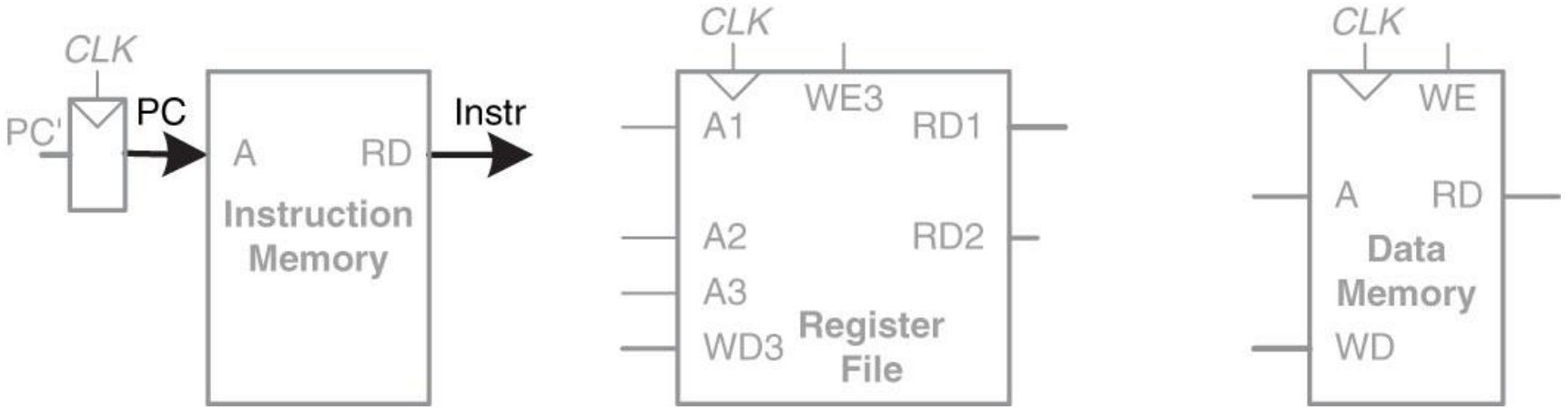


Figure 7.2 Fetch instruction from memory

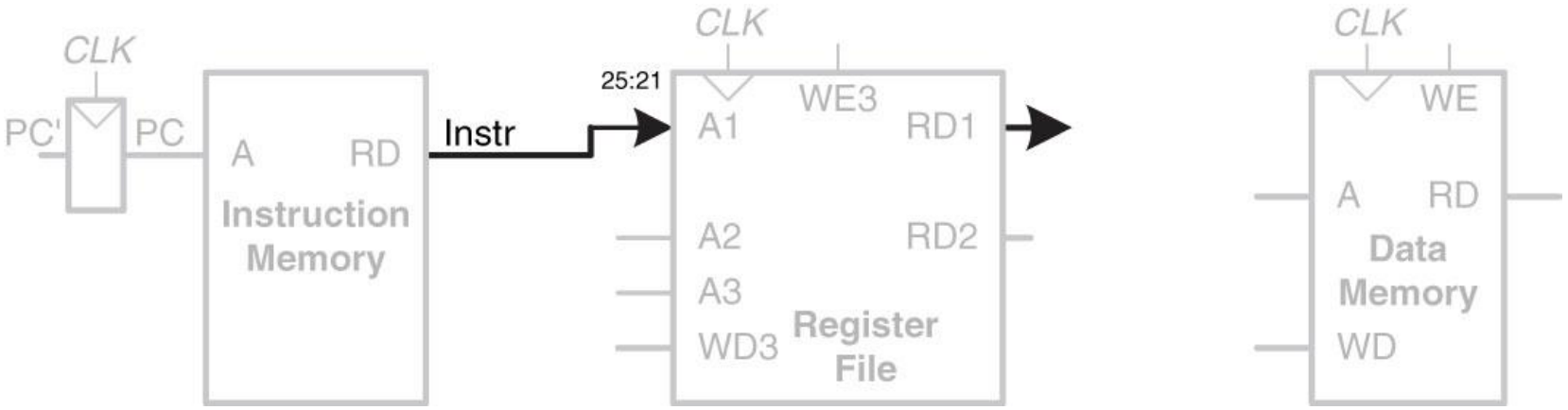


Figure 7.3 Read source operand from register file

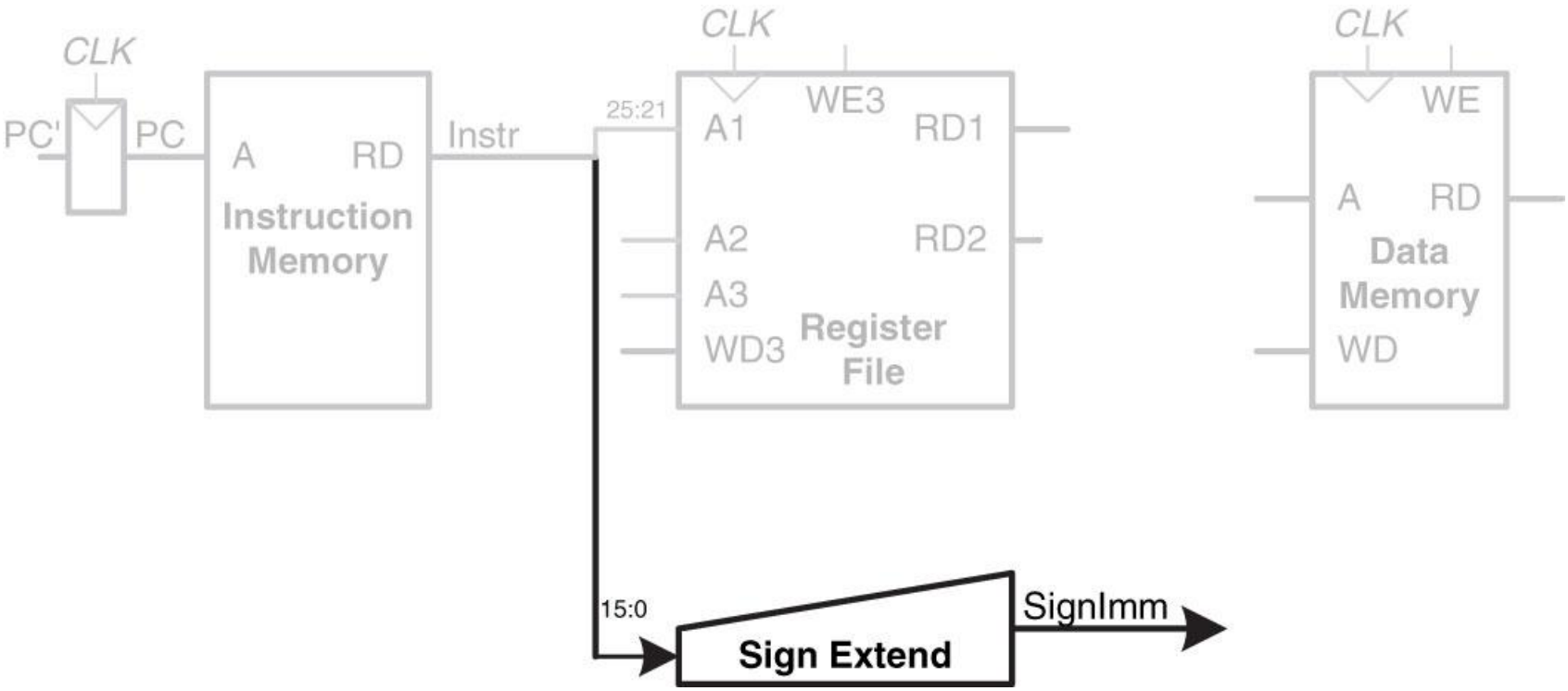


Figure 7.4 Sign-extend the immediate

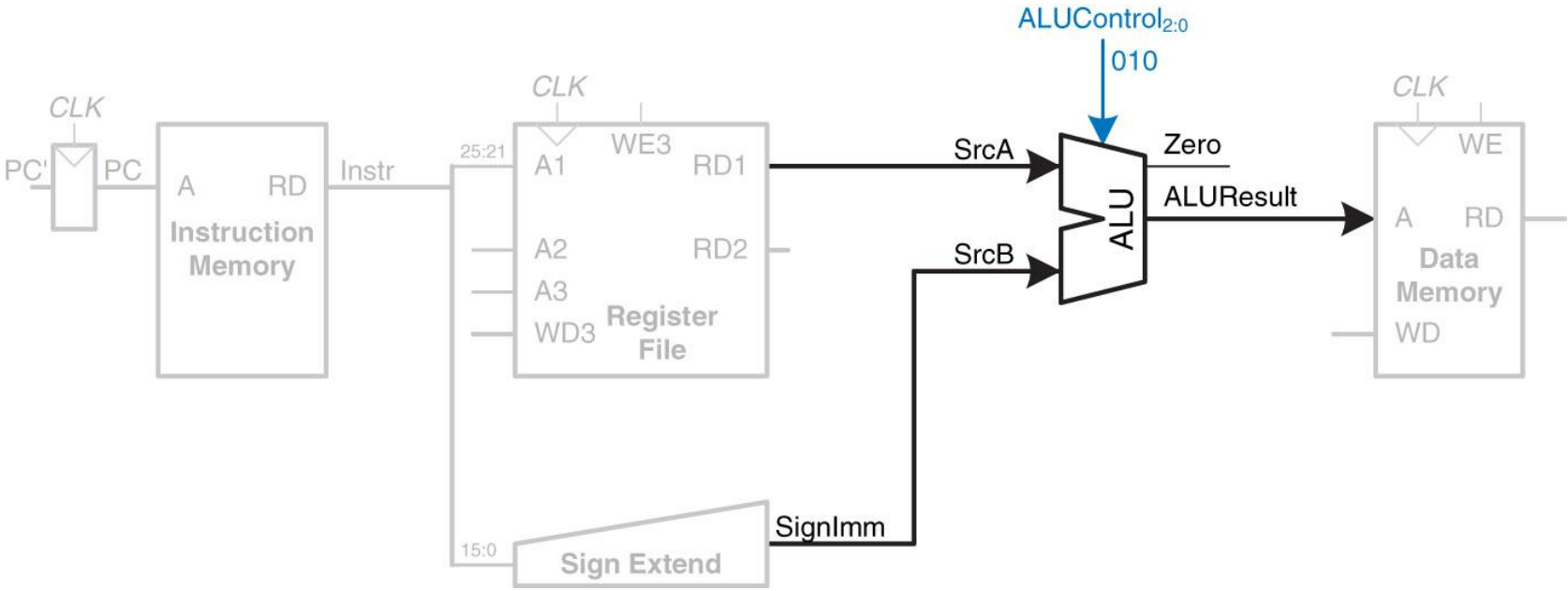


Figure 7.5 Compute memory address

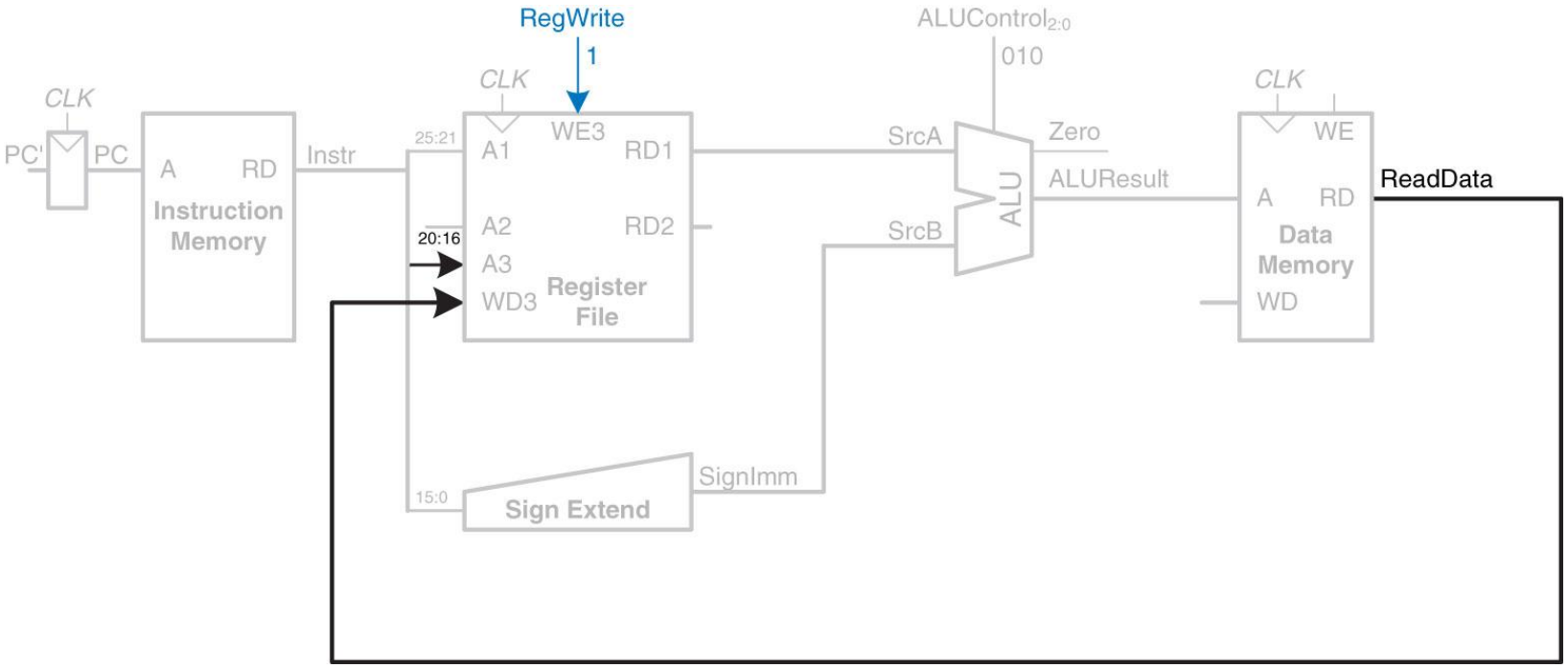


Figure 7.6 Write data back to register file

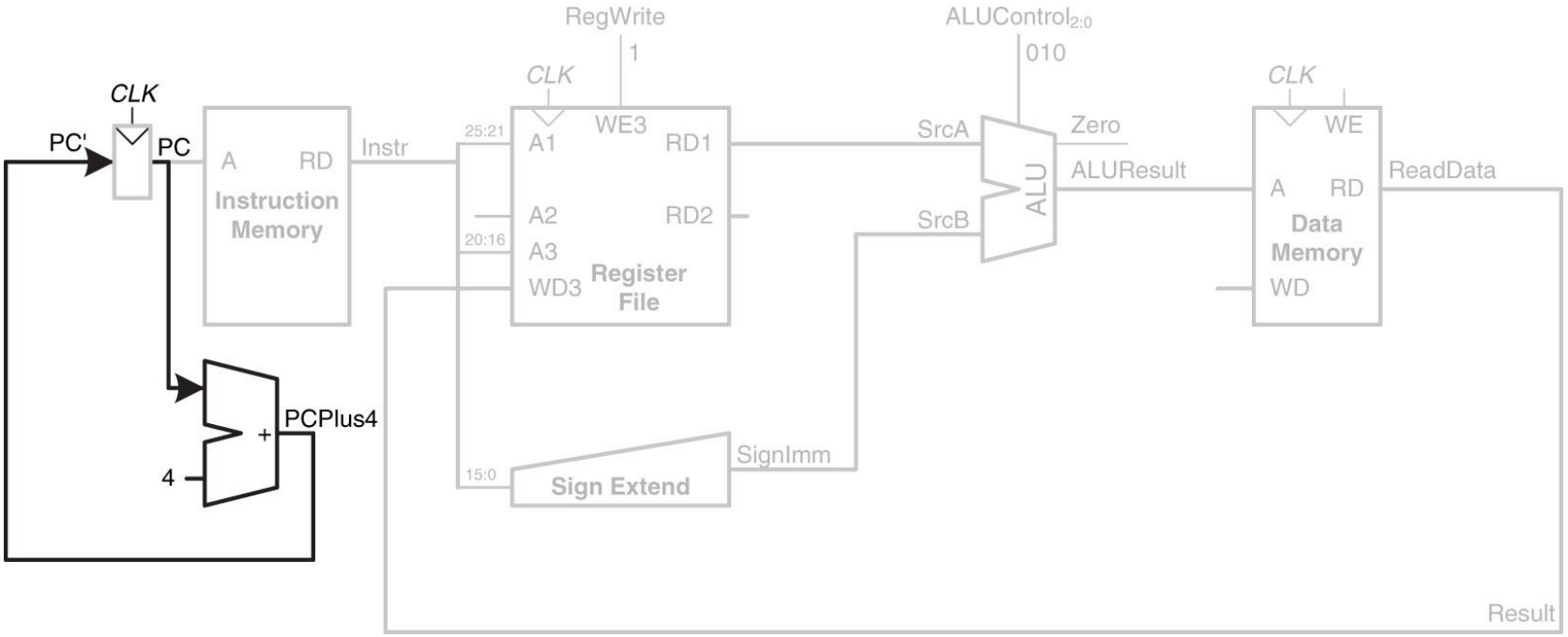


Figure 7.7 Determine address of next instruction for PC

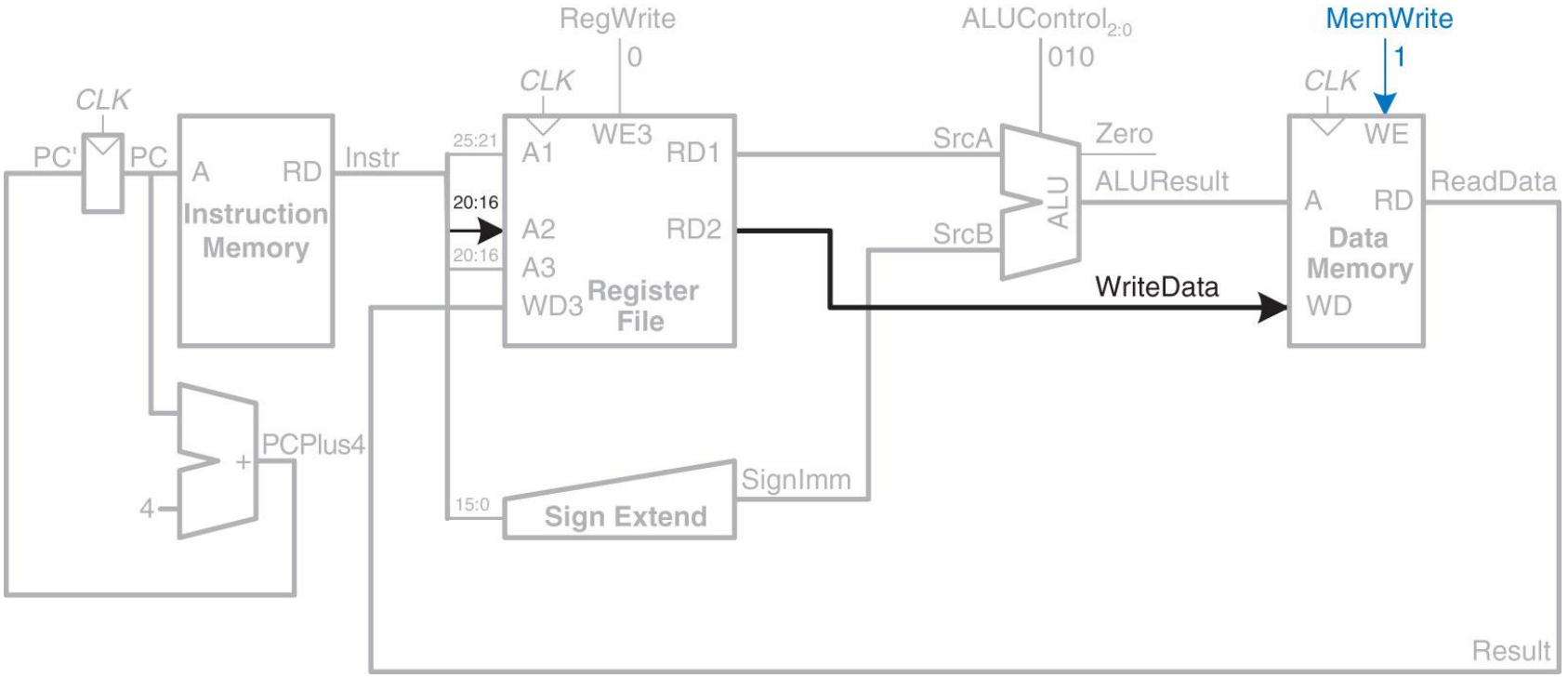


Figure 7.8 Write data to memory for sw instruction

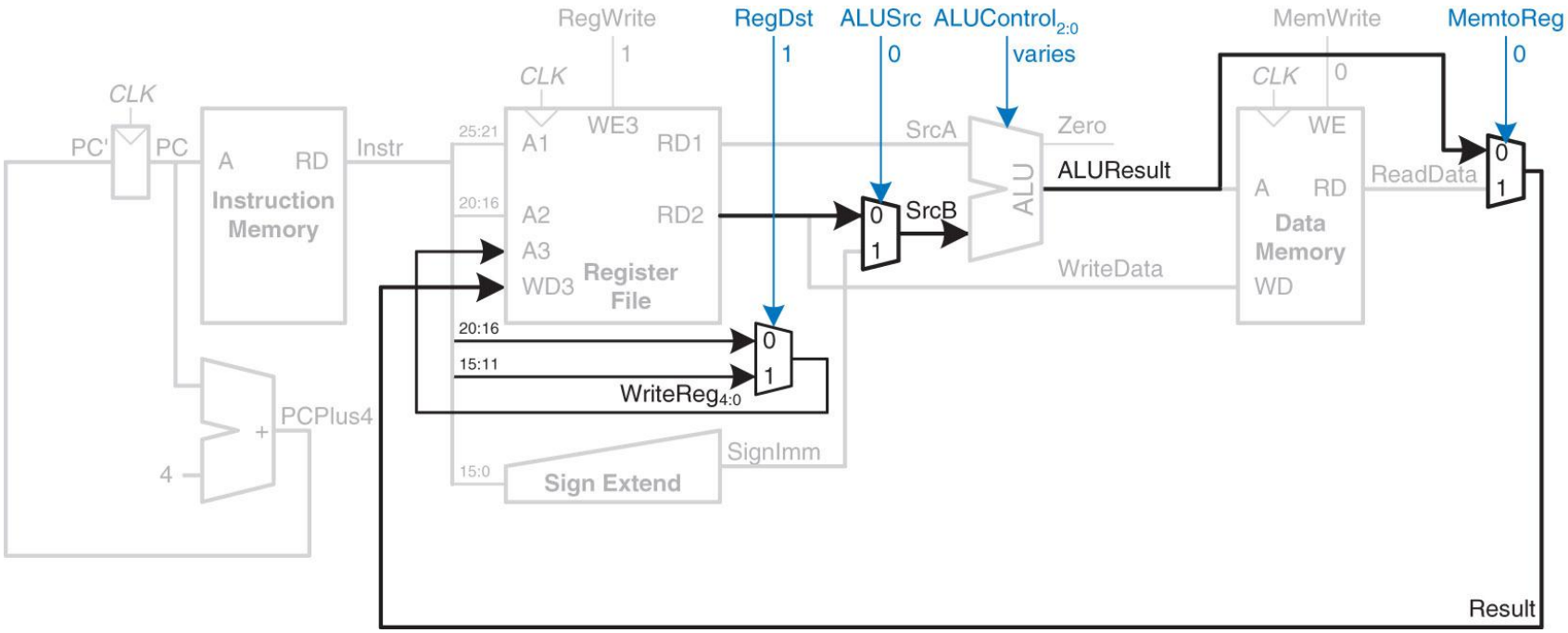


Figure 7.9 Datapath enhancements for R-type instruction

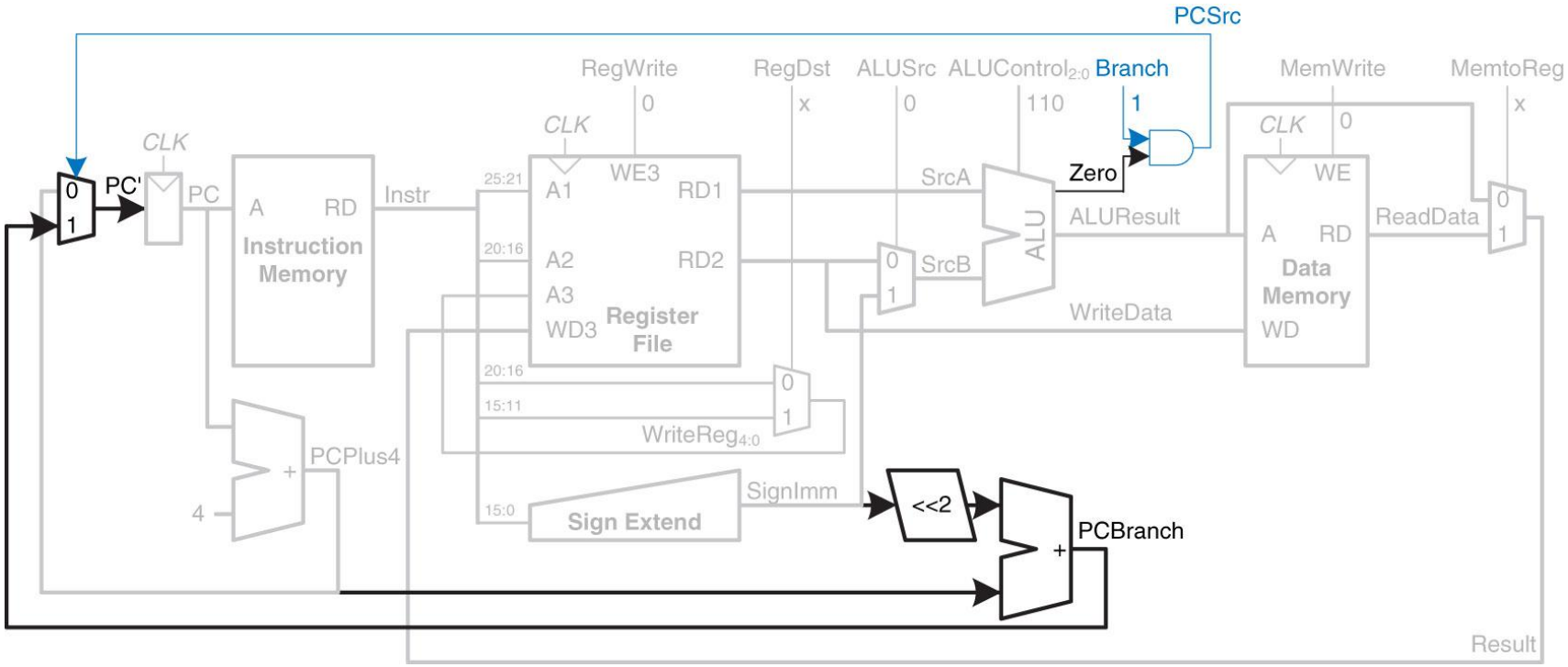


Figure 7.10 Datapath enhancements for beq instruction

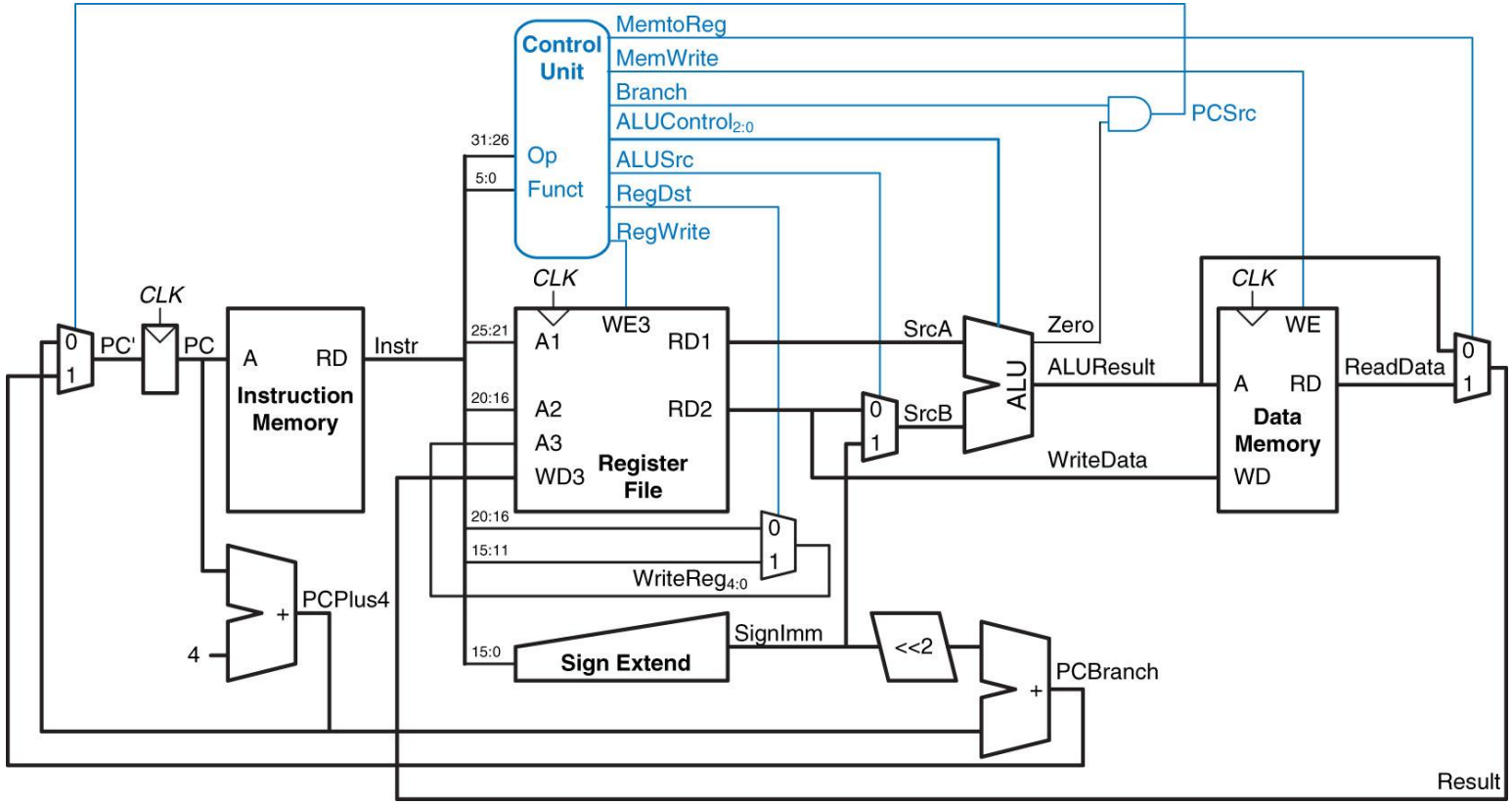


Figure 7.11 Complete single-cycle MIPS processor

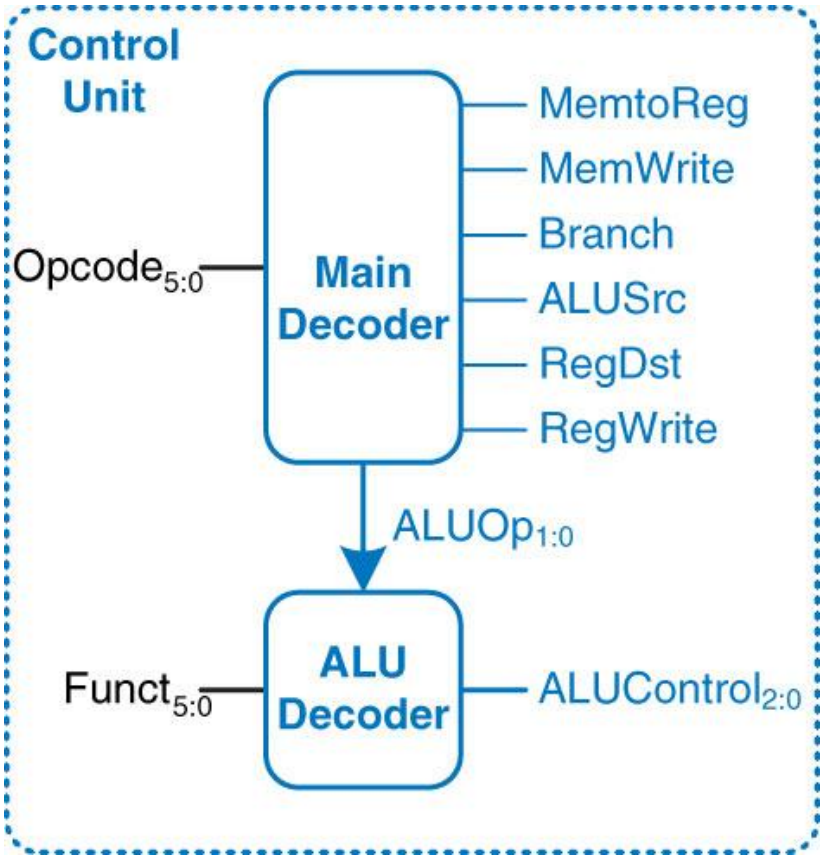


Figure 7.12 Control unit internal structure

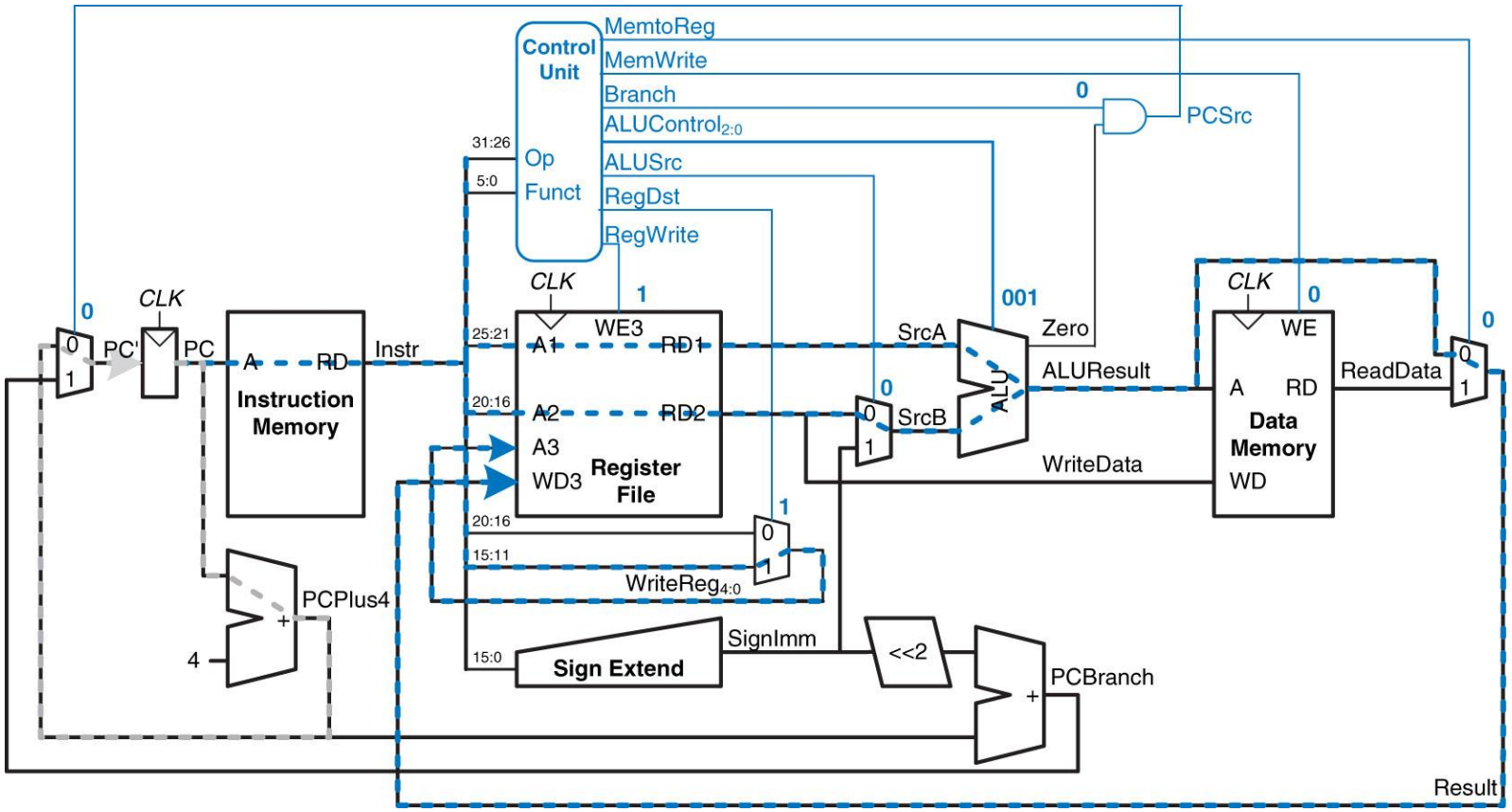


Figure 7.13 Control signals and data flow while executing or instruction

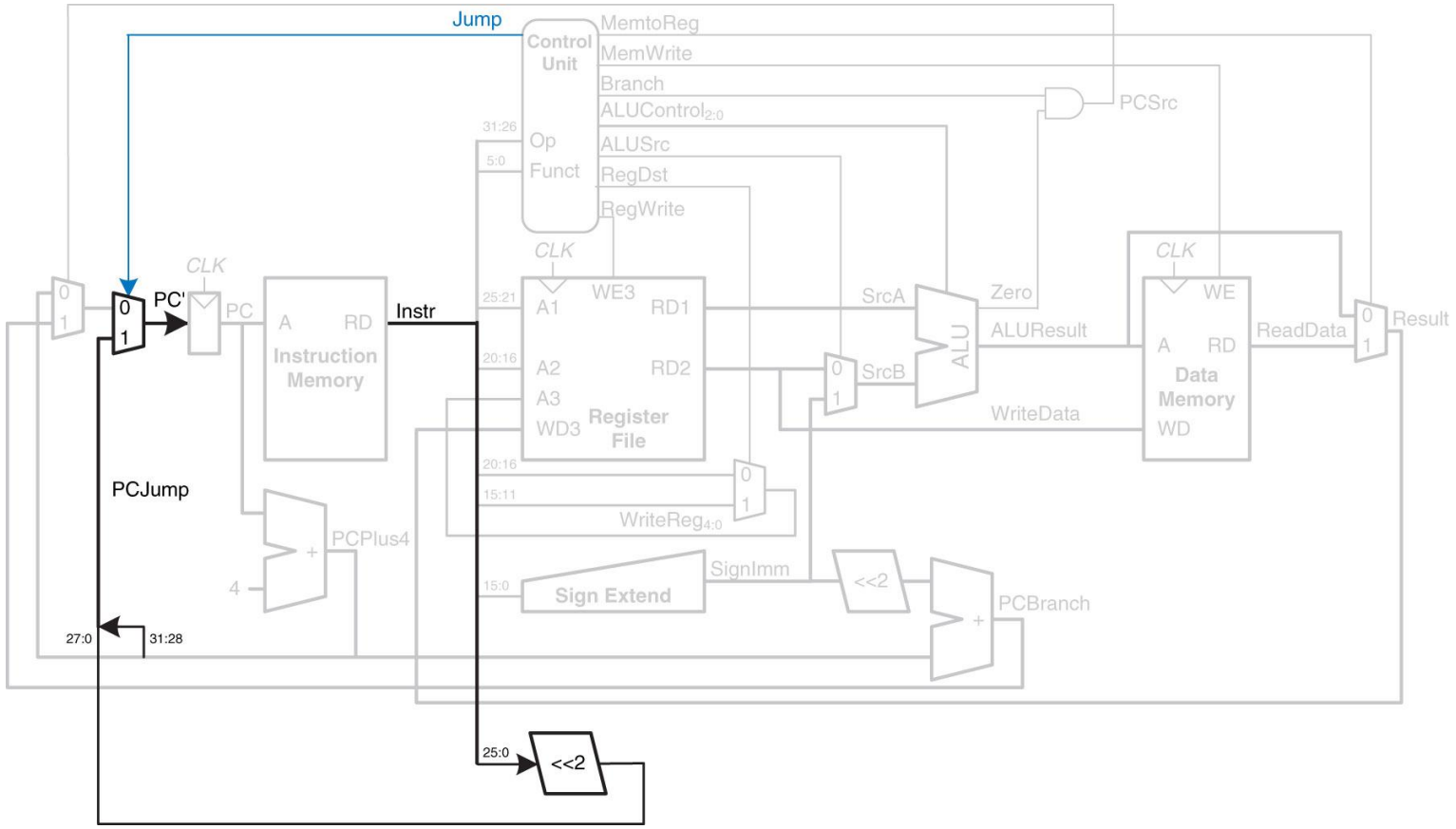


Figure 7.14 Single-cycle MIPS datapath enhanced to support the j instruction

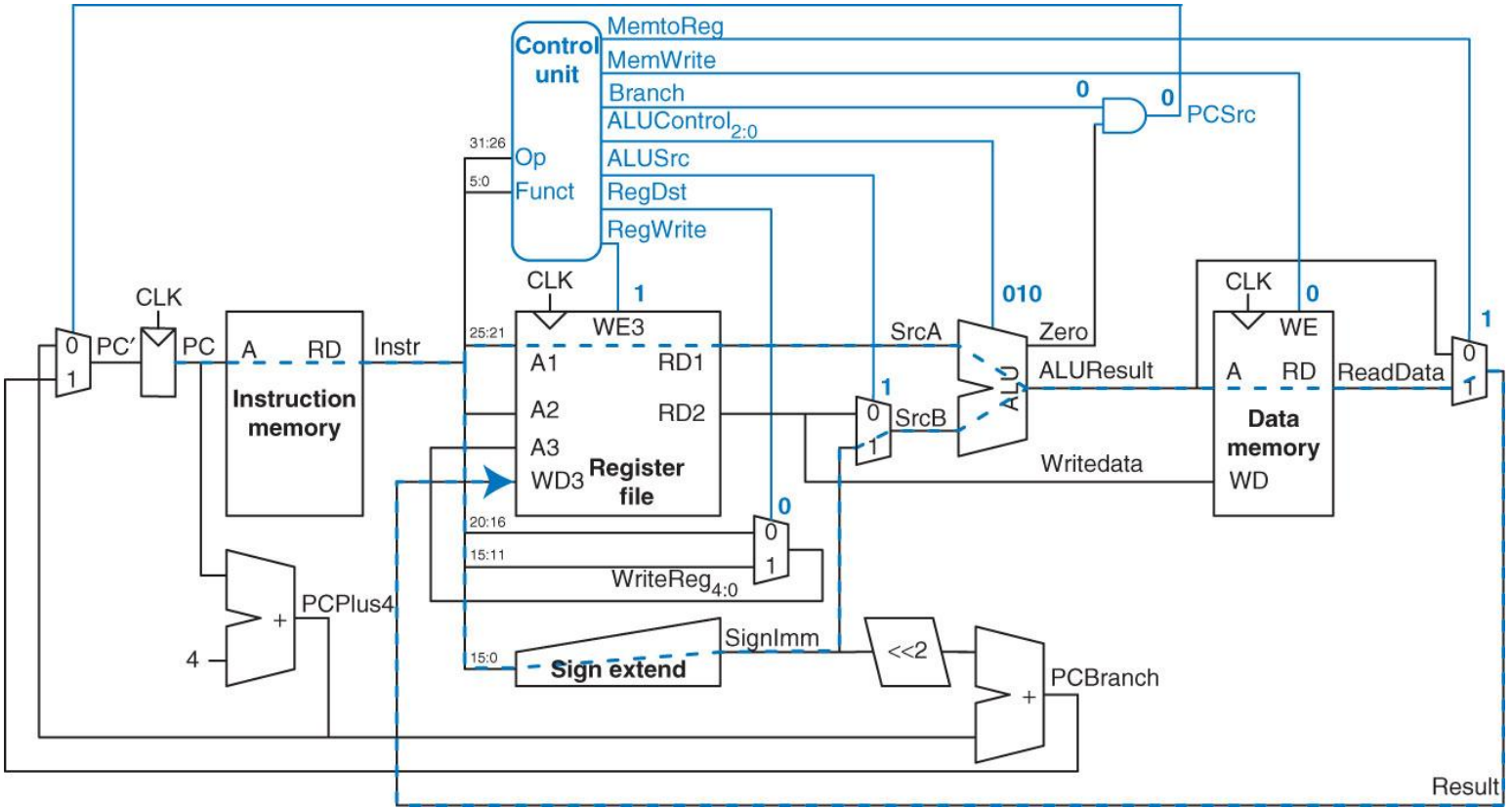


Figure 7.15 Critical path for lw instruction

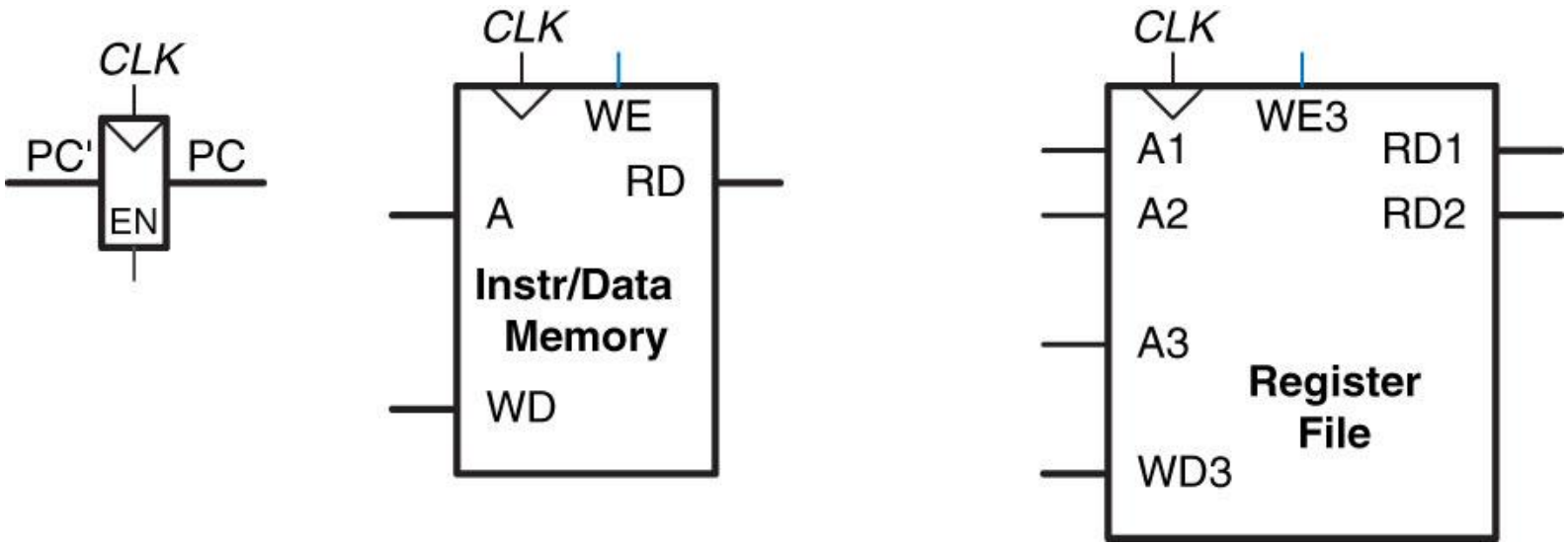


Figure 7.16 State elements with unified instruction/data memory

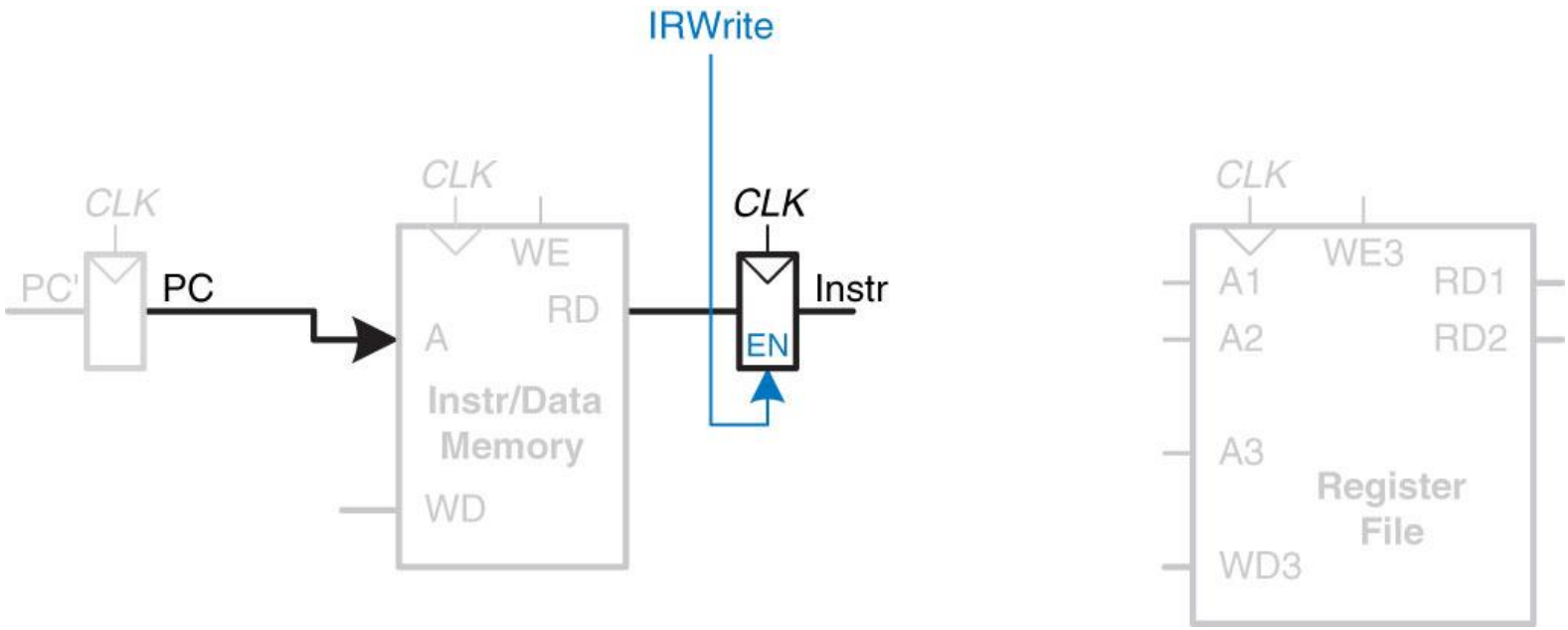


Figure 7.17 Fetch instruction from memory

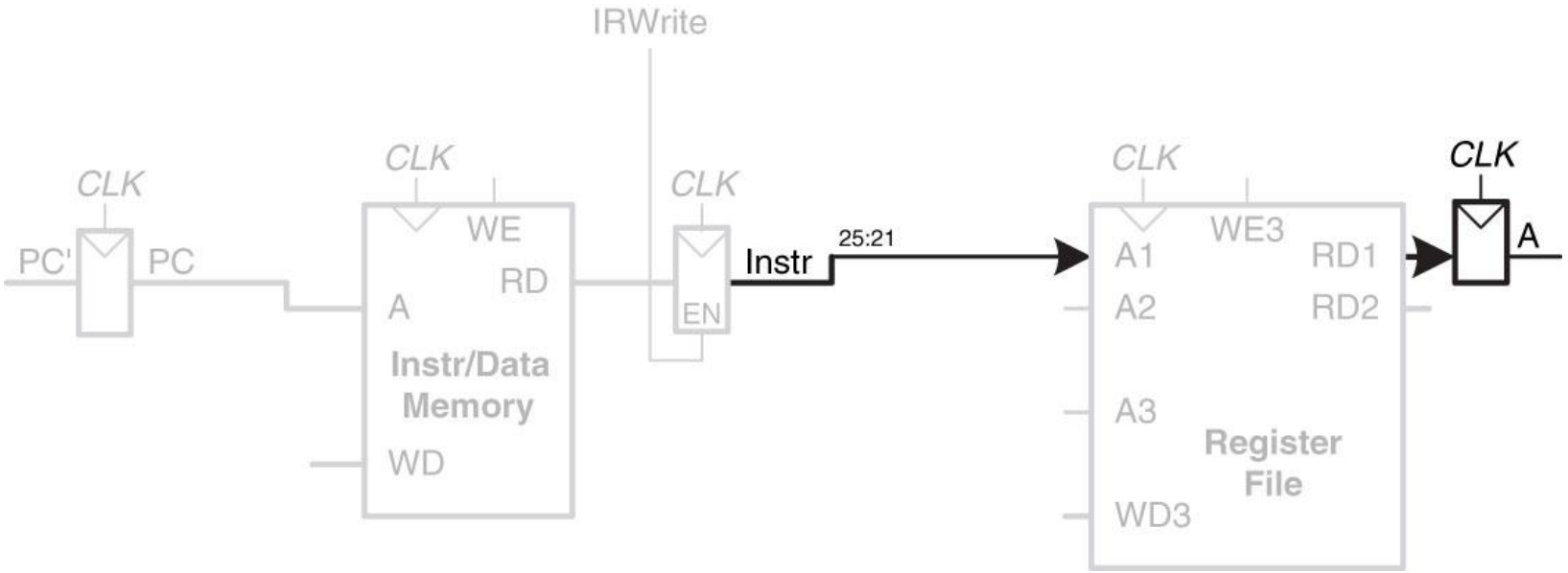


Figure 7.18 Read source operand from register file

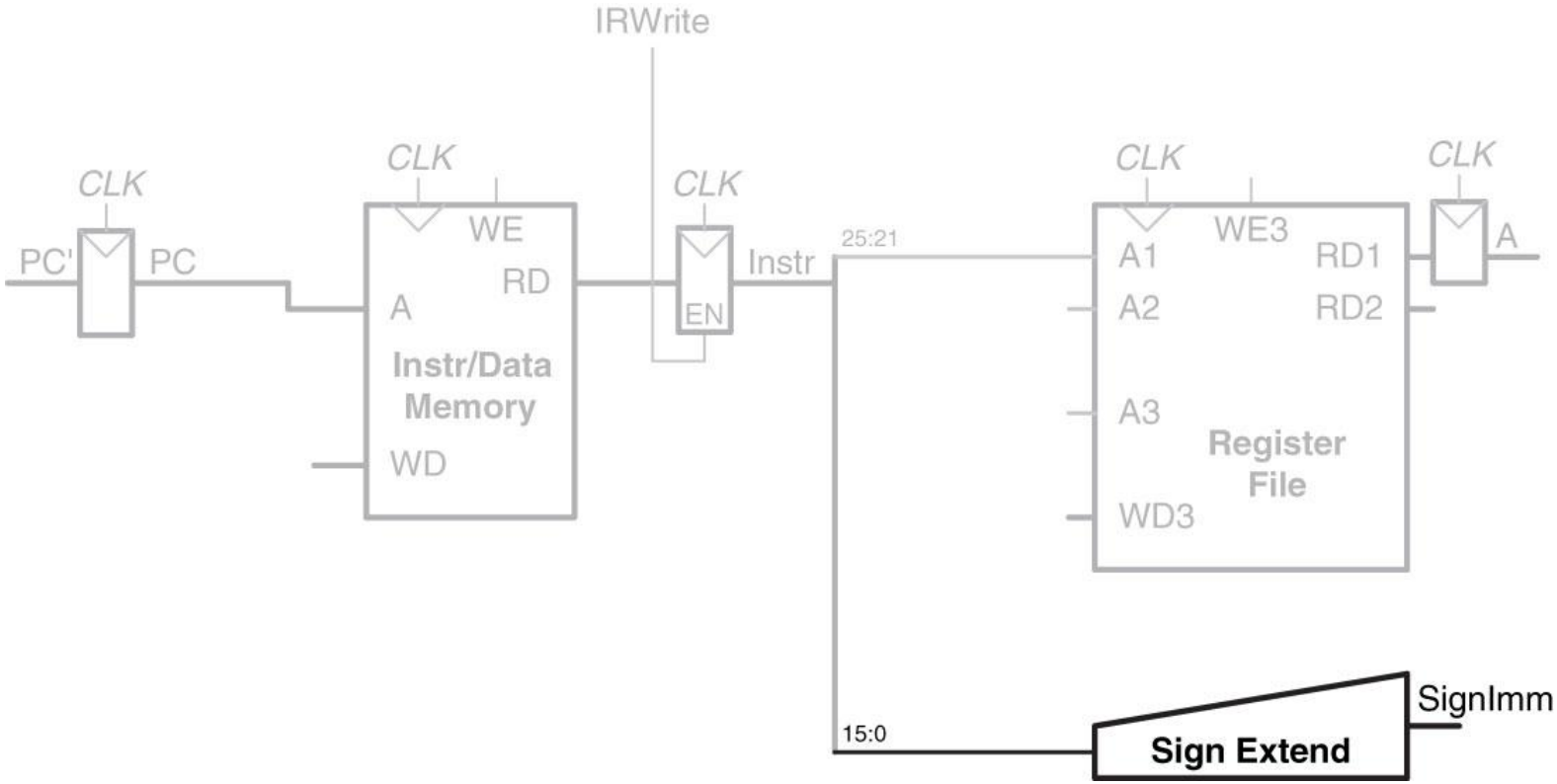


Figure 7.19 Sign-extend the immediate

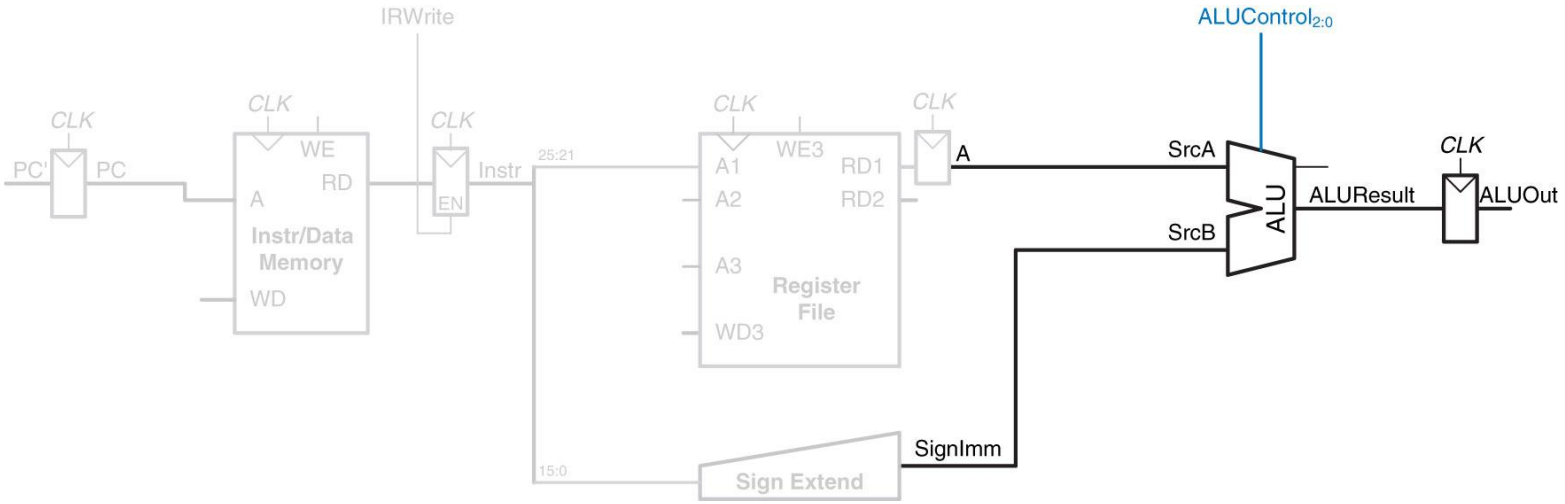


Figure 7.20 Add base address to offset

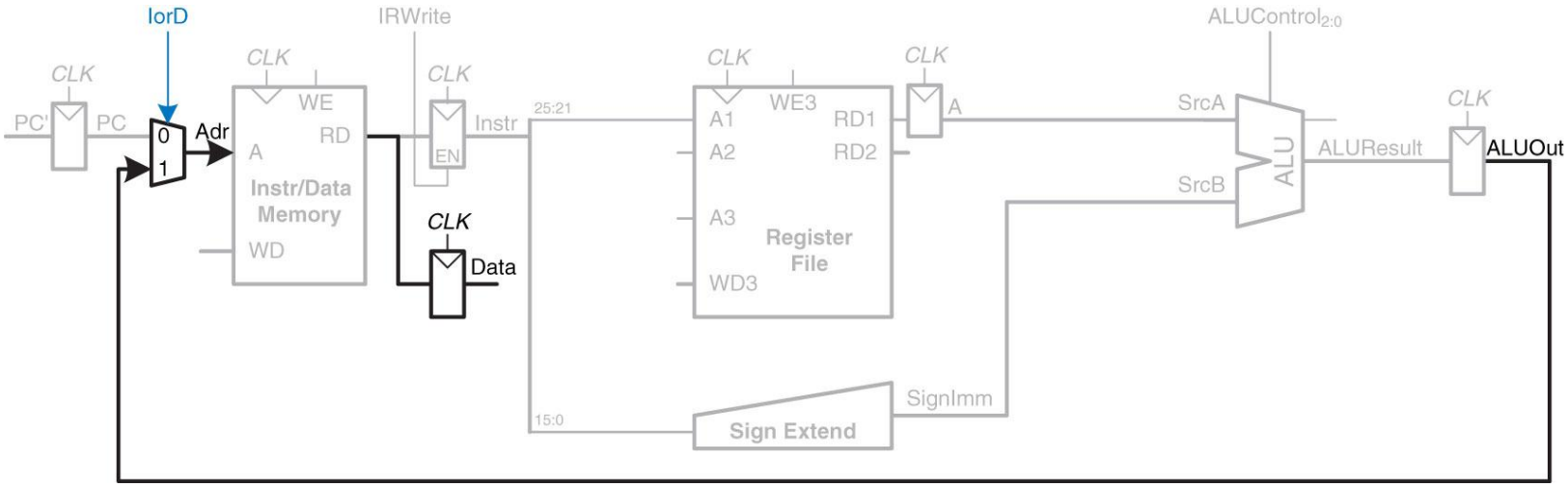


Figure 7.21 Load data from memory

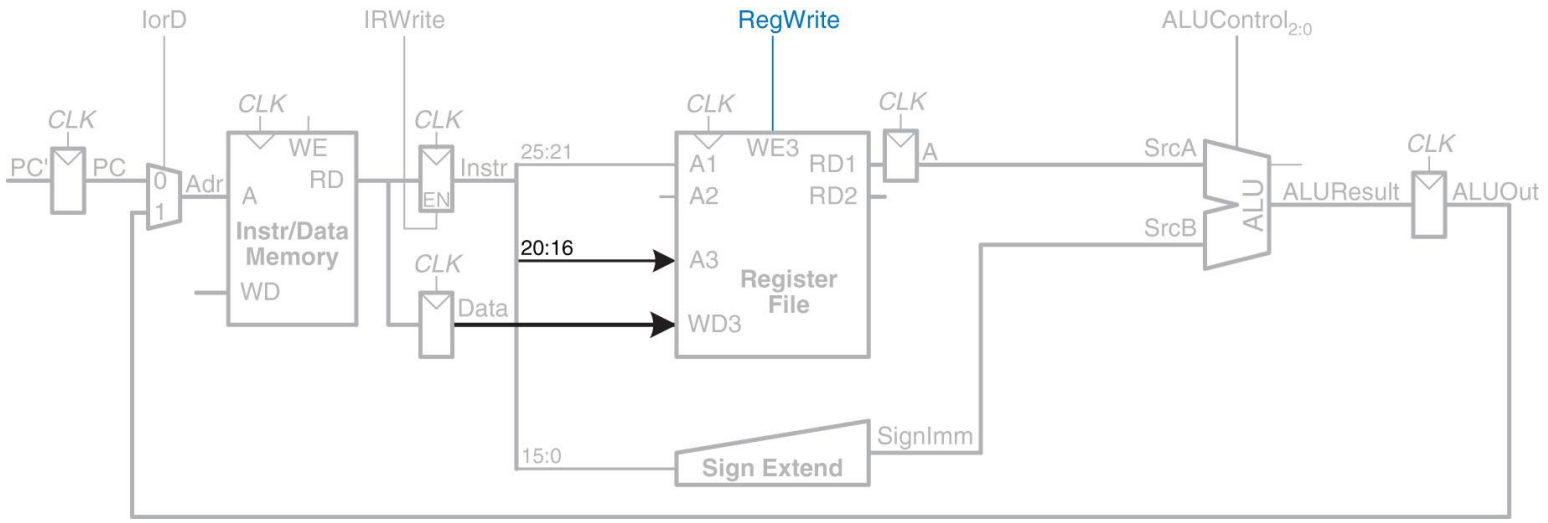


Figure 7.22 Write data back to register file

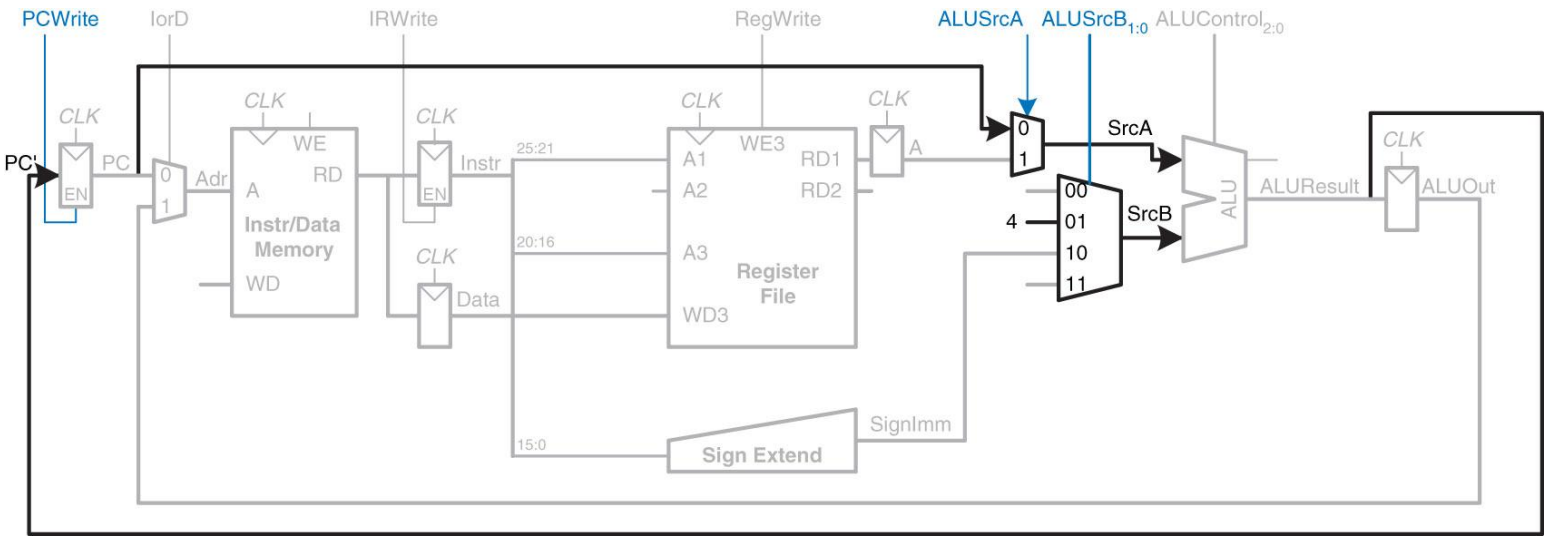


Figure 7.23 Increment PC by 4

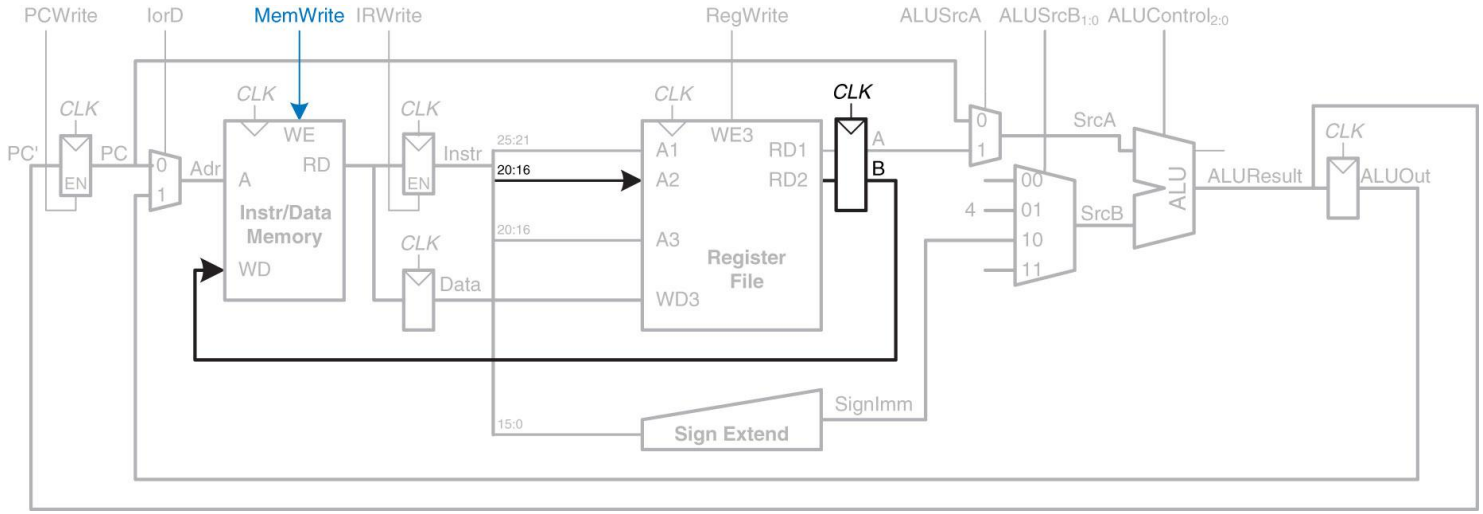


Figure 7.24 Enhanced datapath for sw instruction

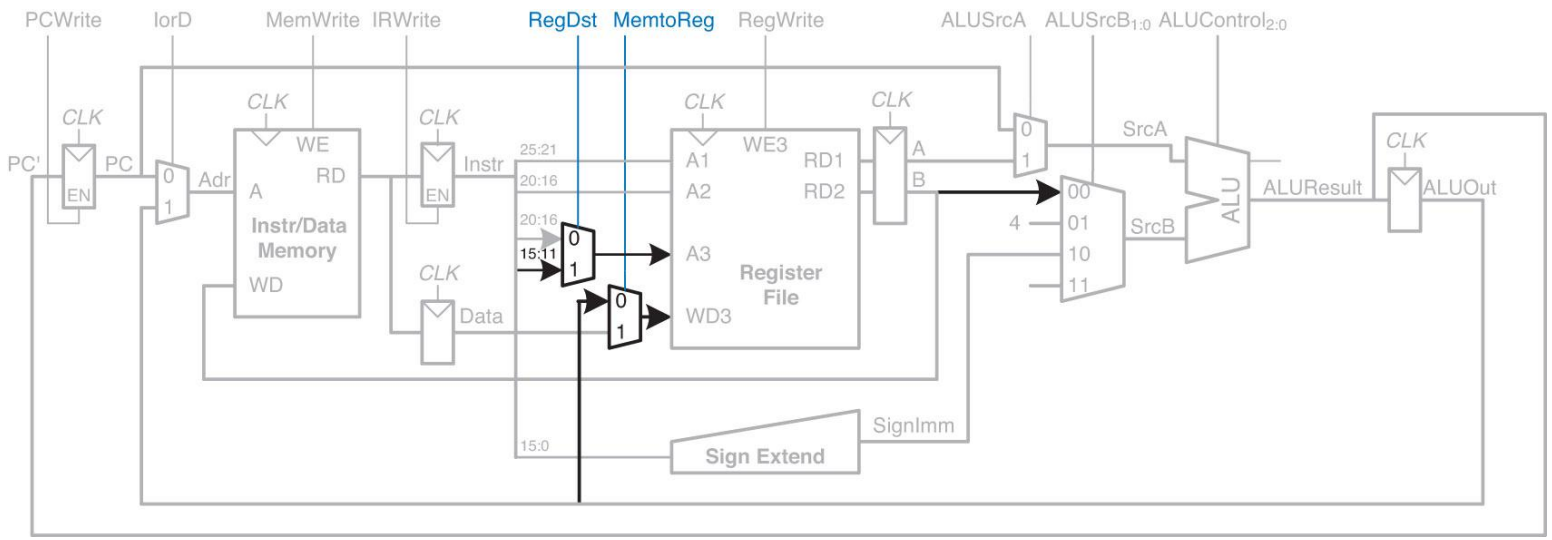


Figure 7.25 Enhanced datapath for R-type instructions

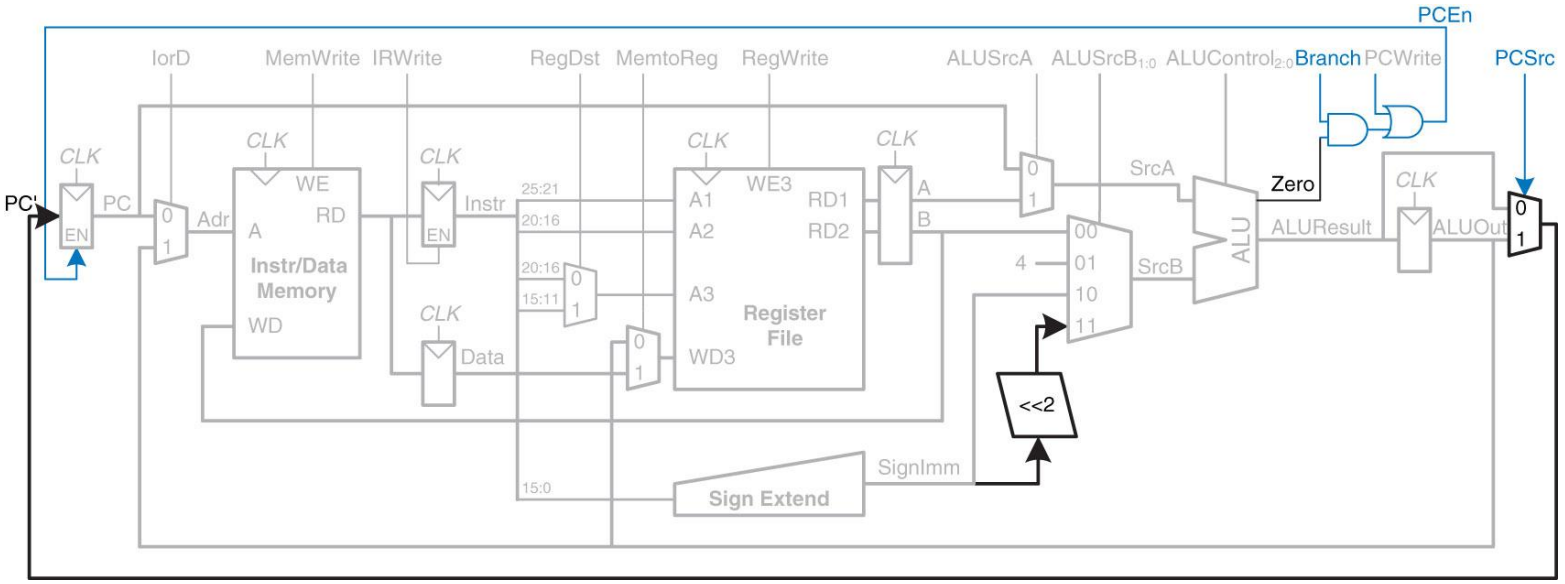


Figure 7.26 Enhanced datapath for beq instruction

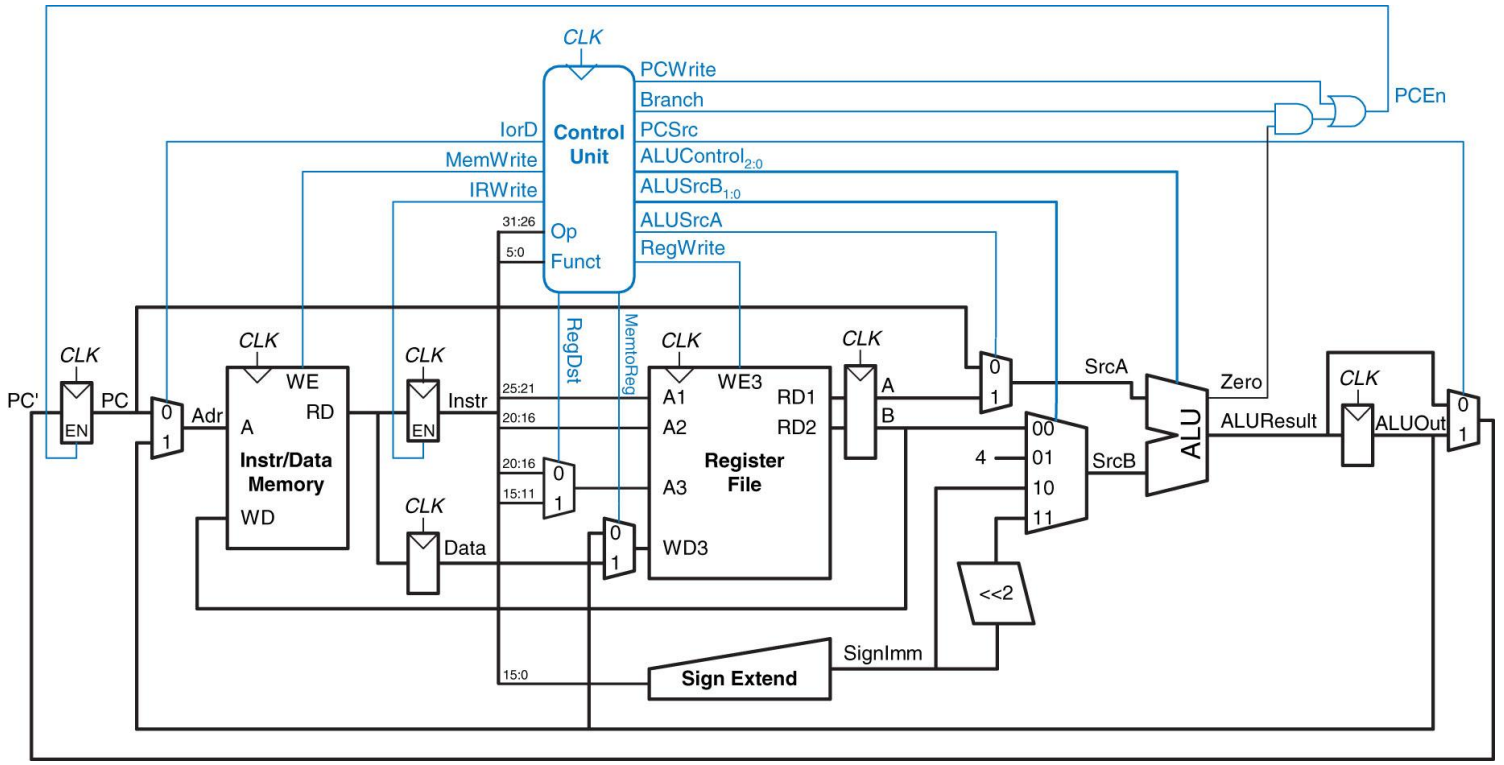


Figure 7.27 Complete multicycle MIPS processor

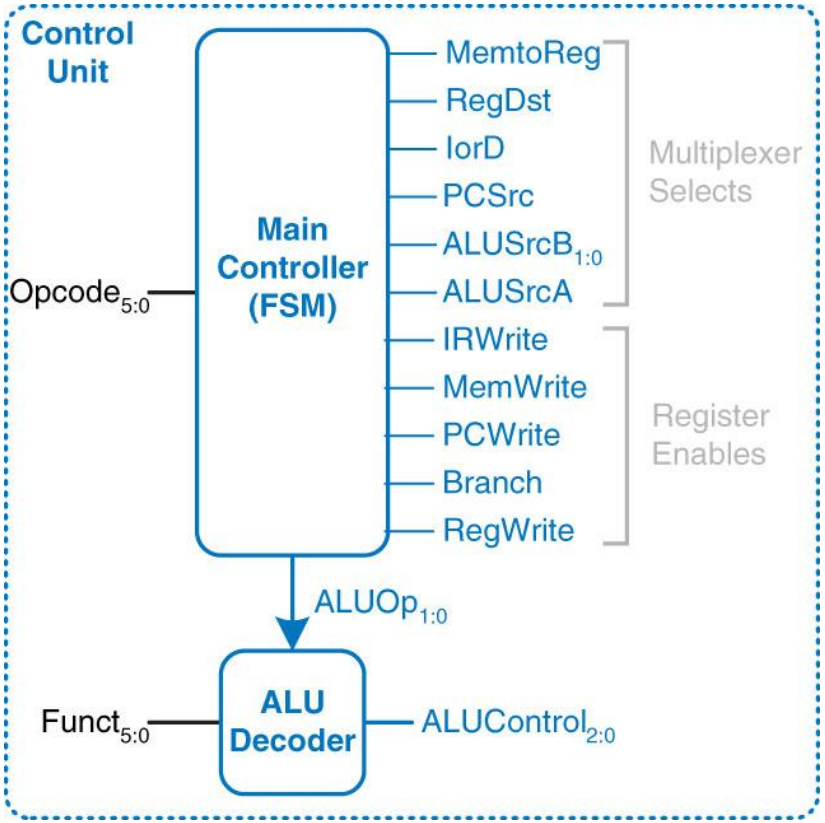


Figure 7.28 Control unit internal structure

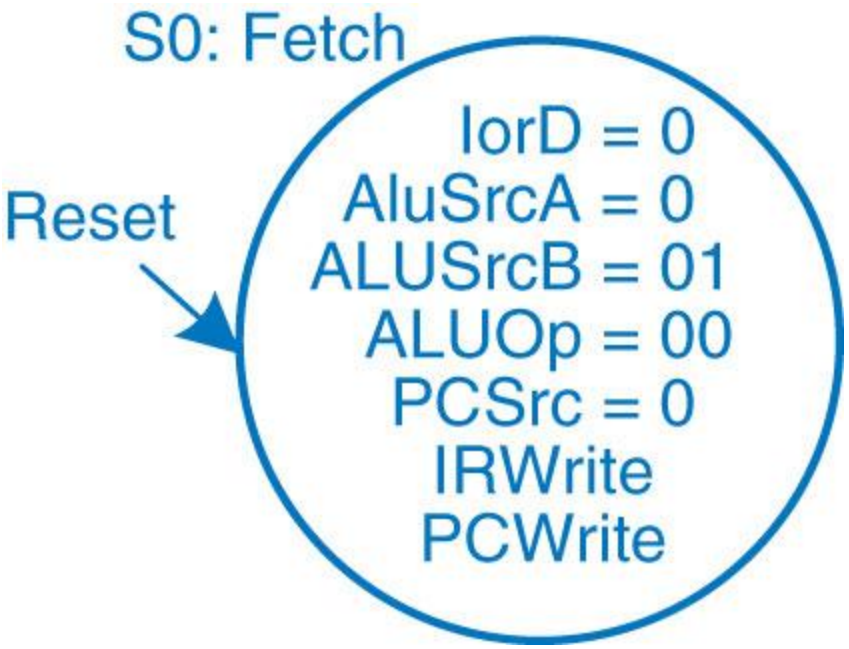


Figure 7.29 Fetch

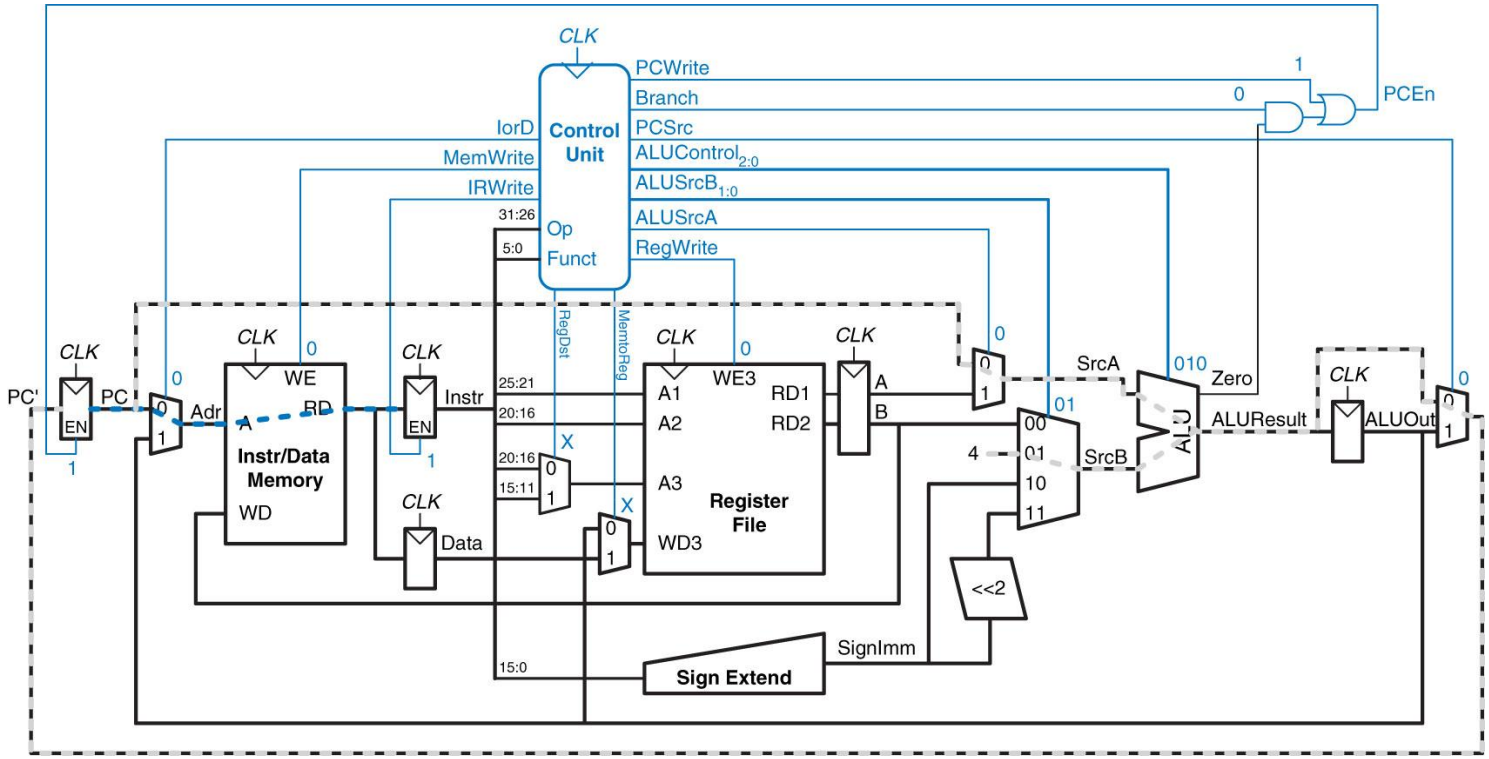


Figure 7.30 Data flow during the fetch step

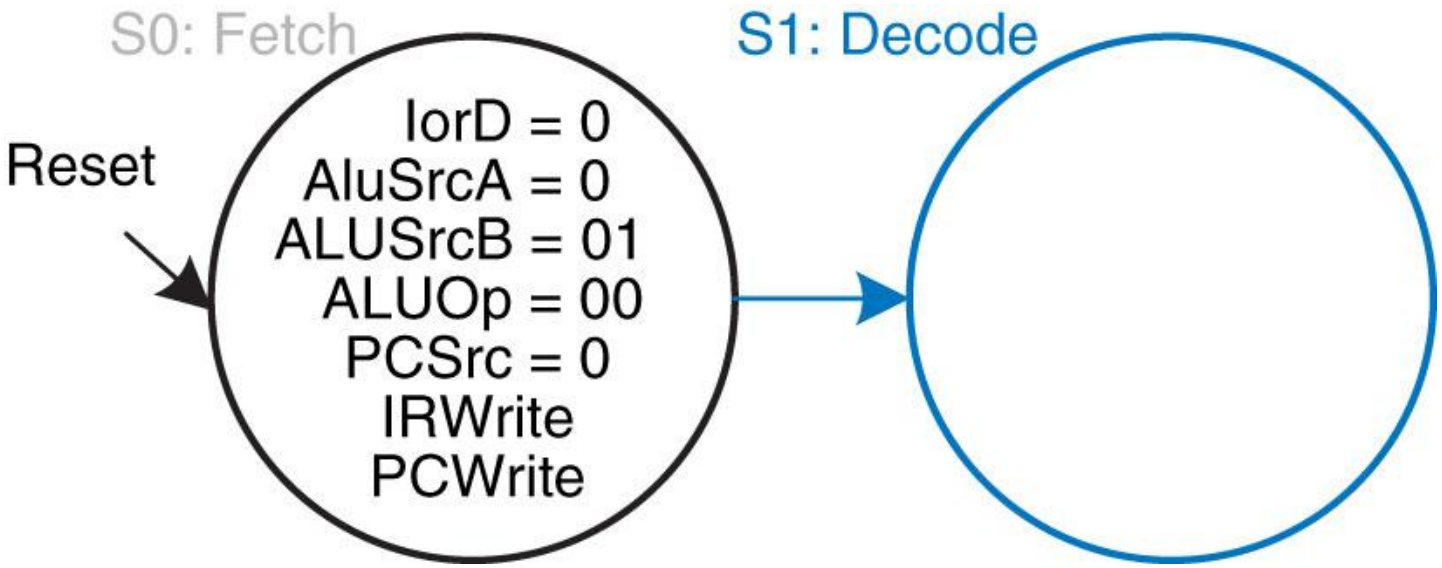


Figure 7.31 Decode

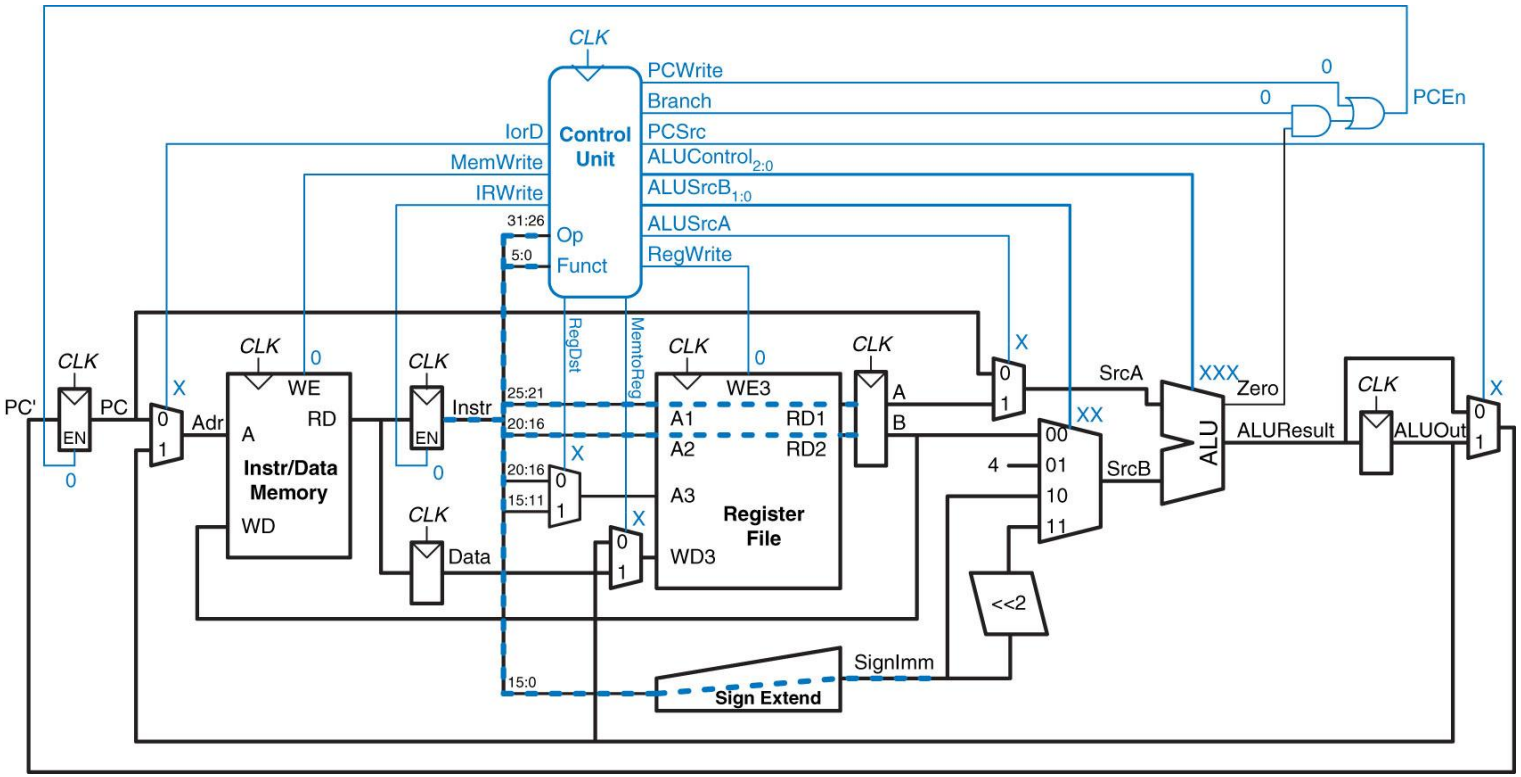


Figure 7.32 Data flow during the decode step

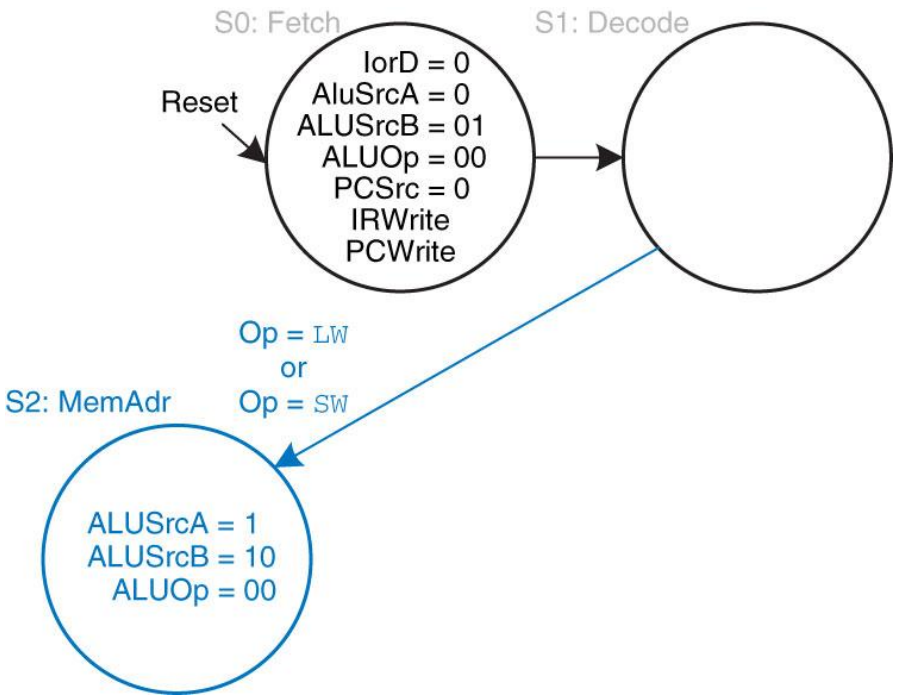


Figure 7.33 Memory address computation

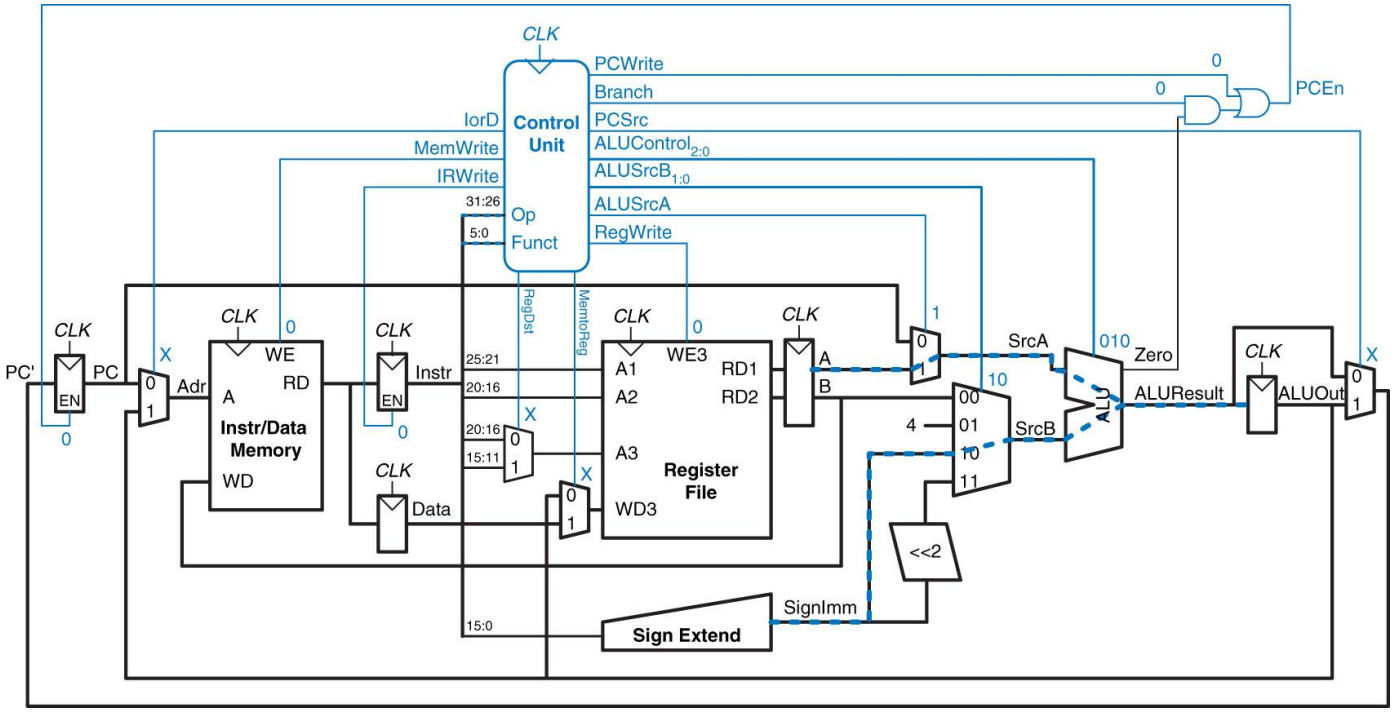


Figure 7.34 Data flow during memory address computation

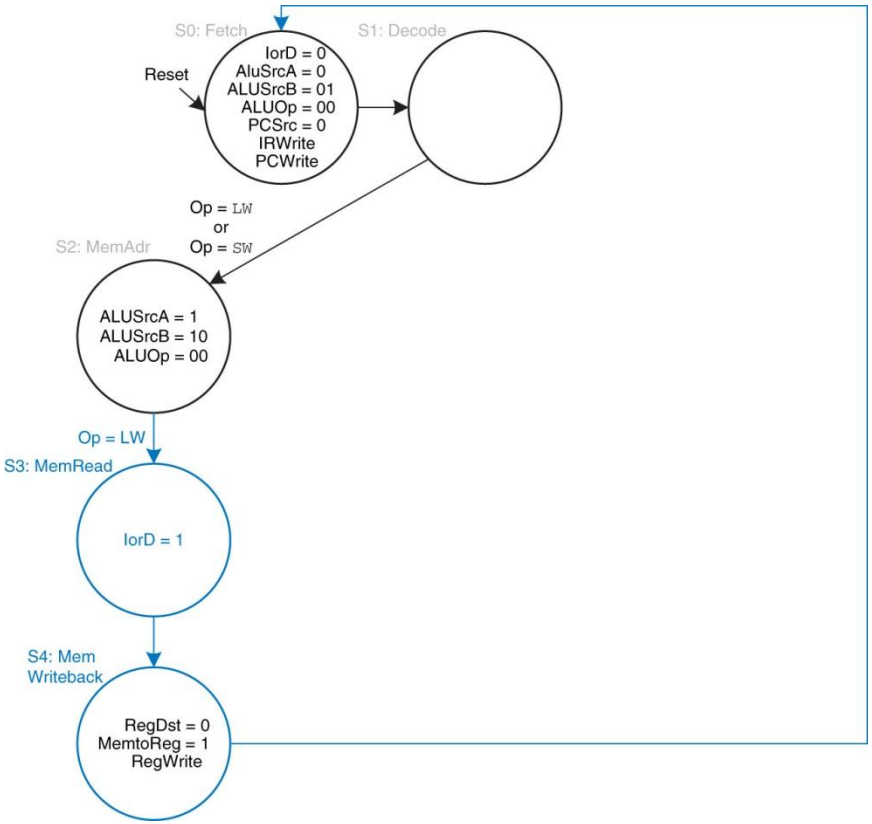


Figure 7.35 Memory read

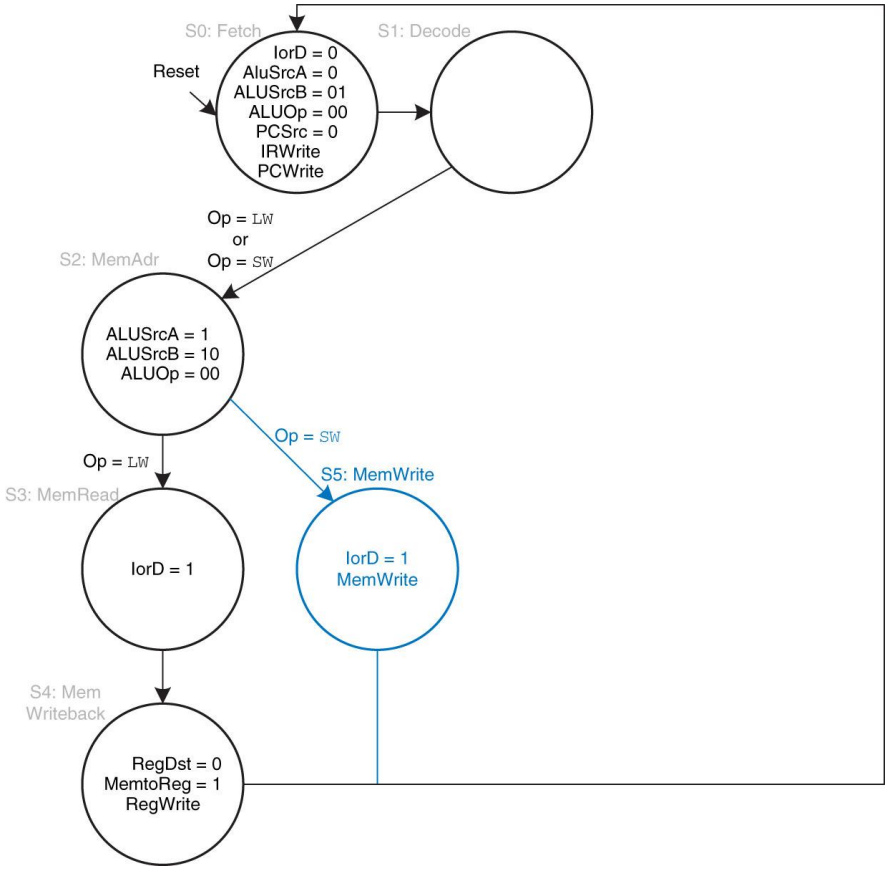


Figure 7.36 Memory write

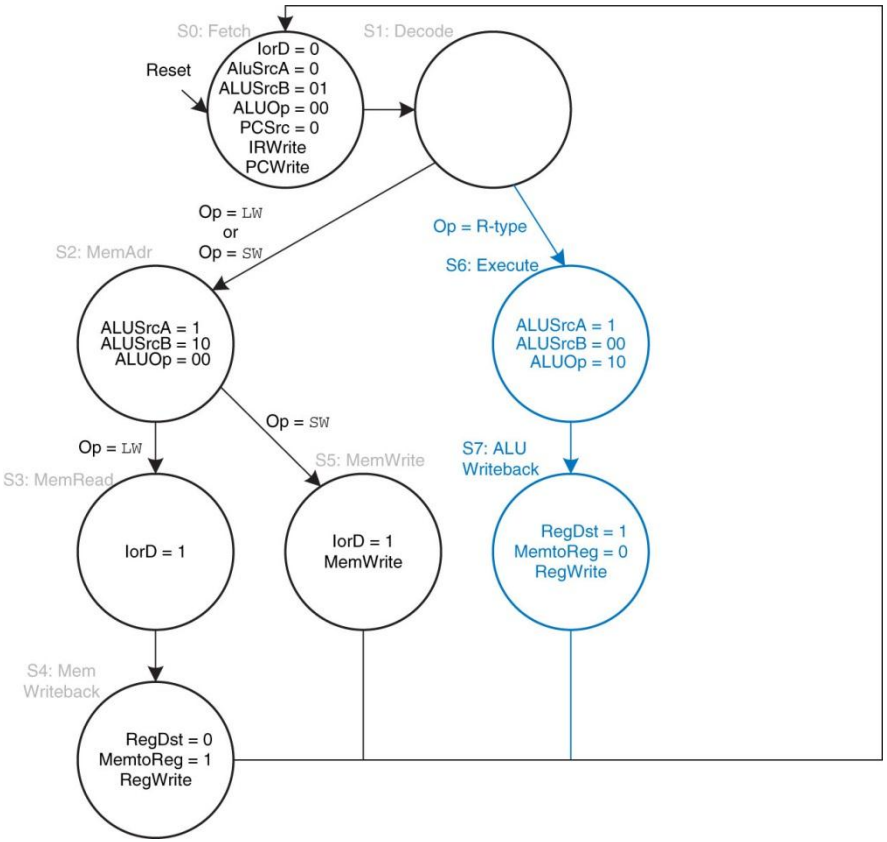


Figure 7.37 Execute R-type operation

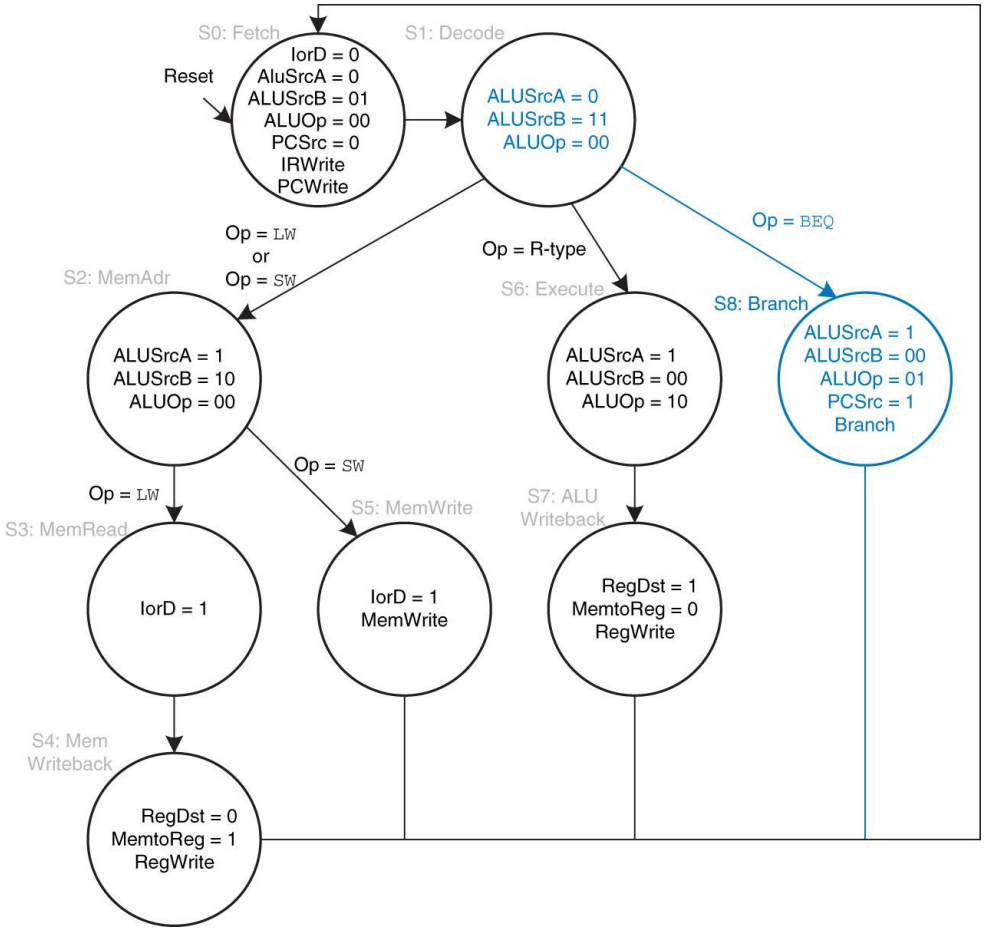


Figure 7.38 Branch

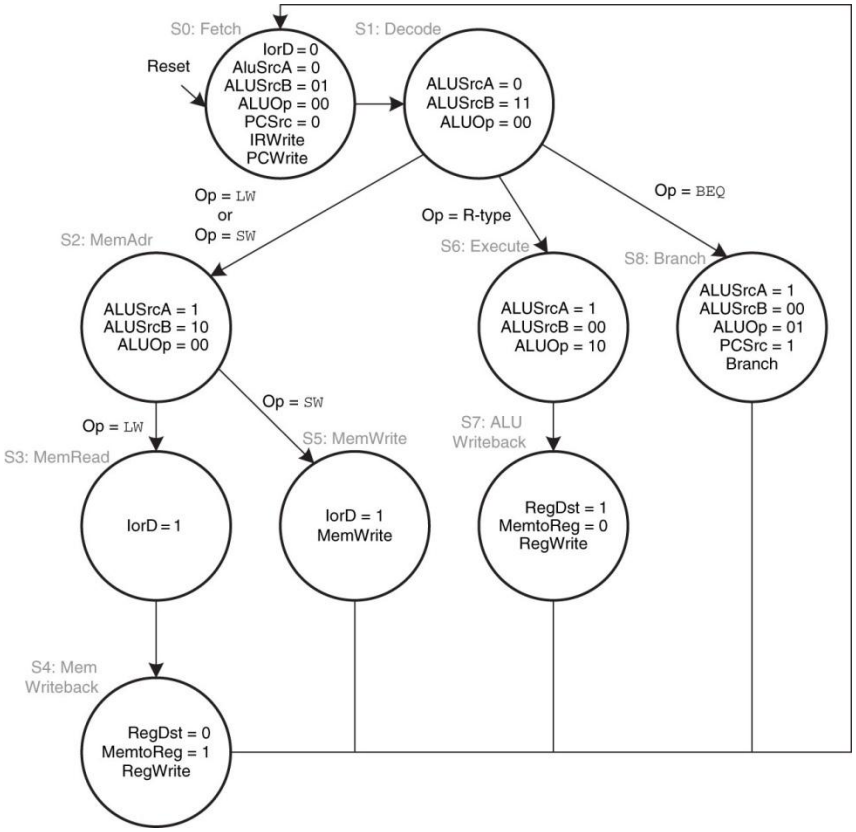


Figure 7.39 Complete multicycle control FSM

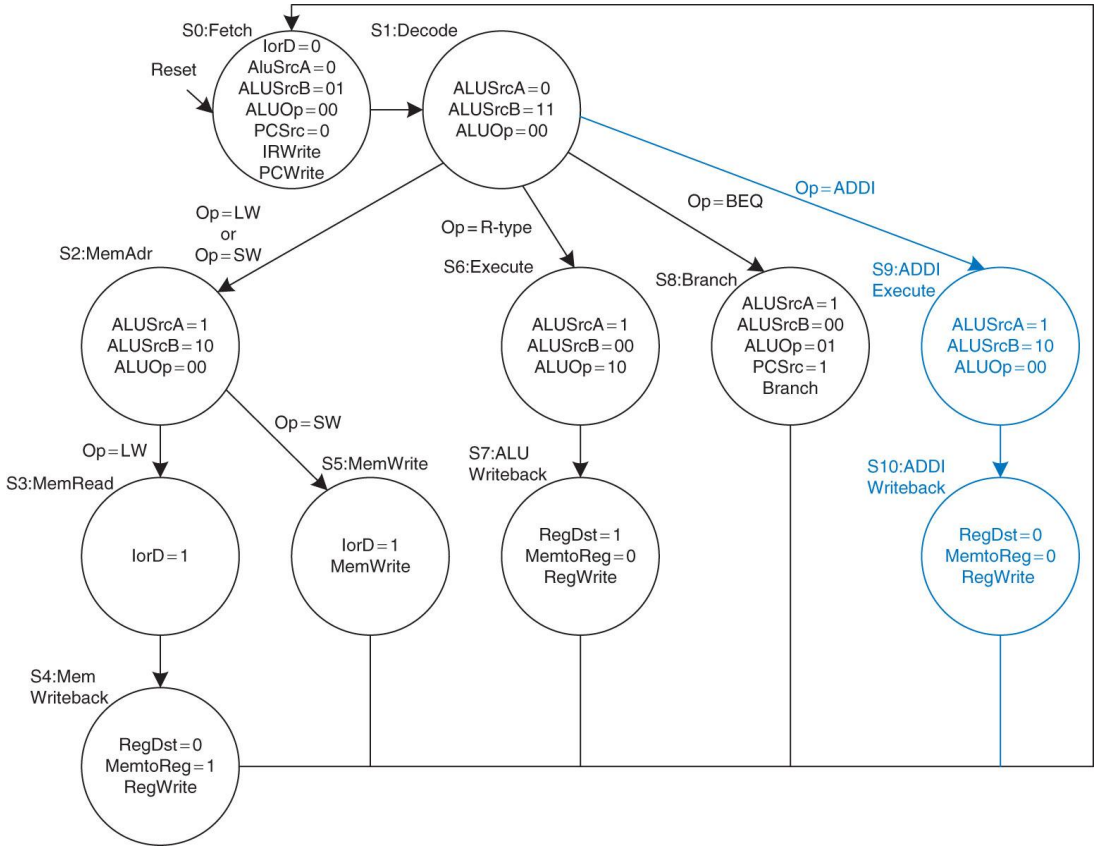


Figure 7.40 Main controller states for addi

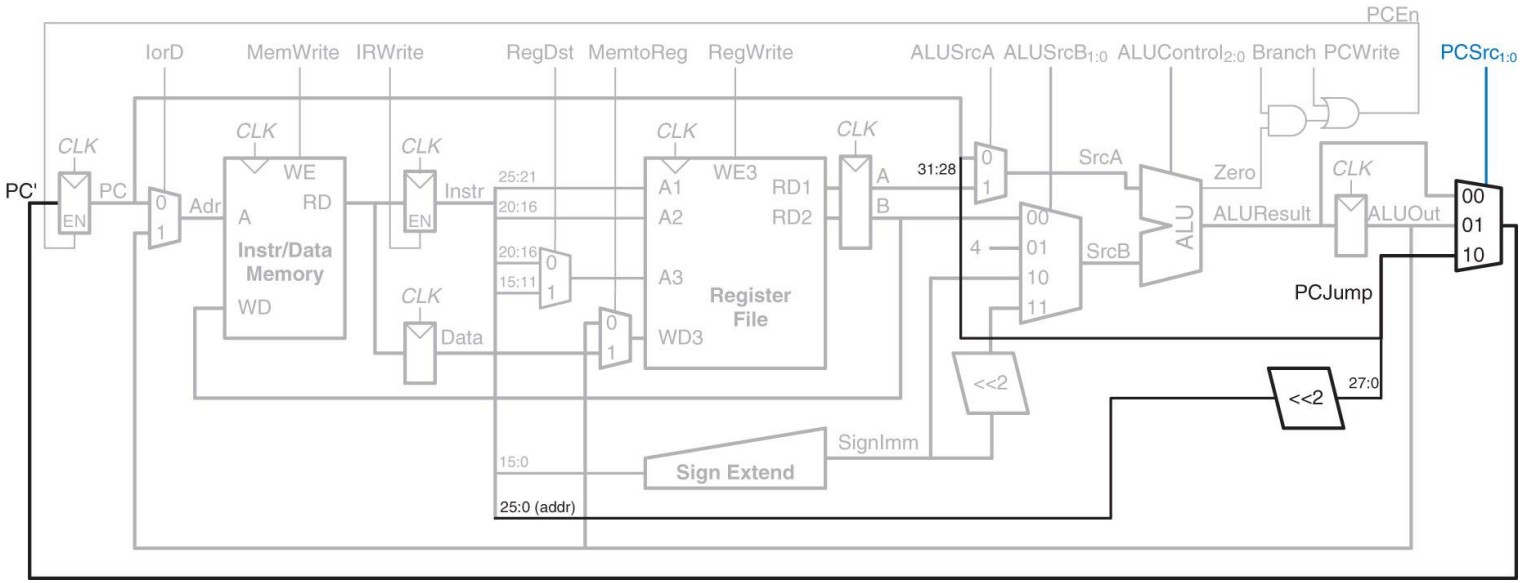


Figure 7.41 Multicycle MIPS datapath enhanced to support the j instruction

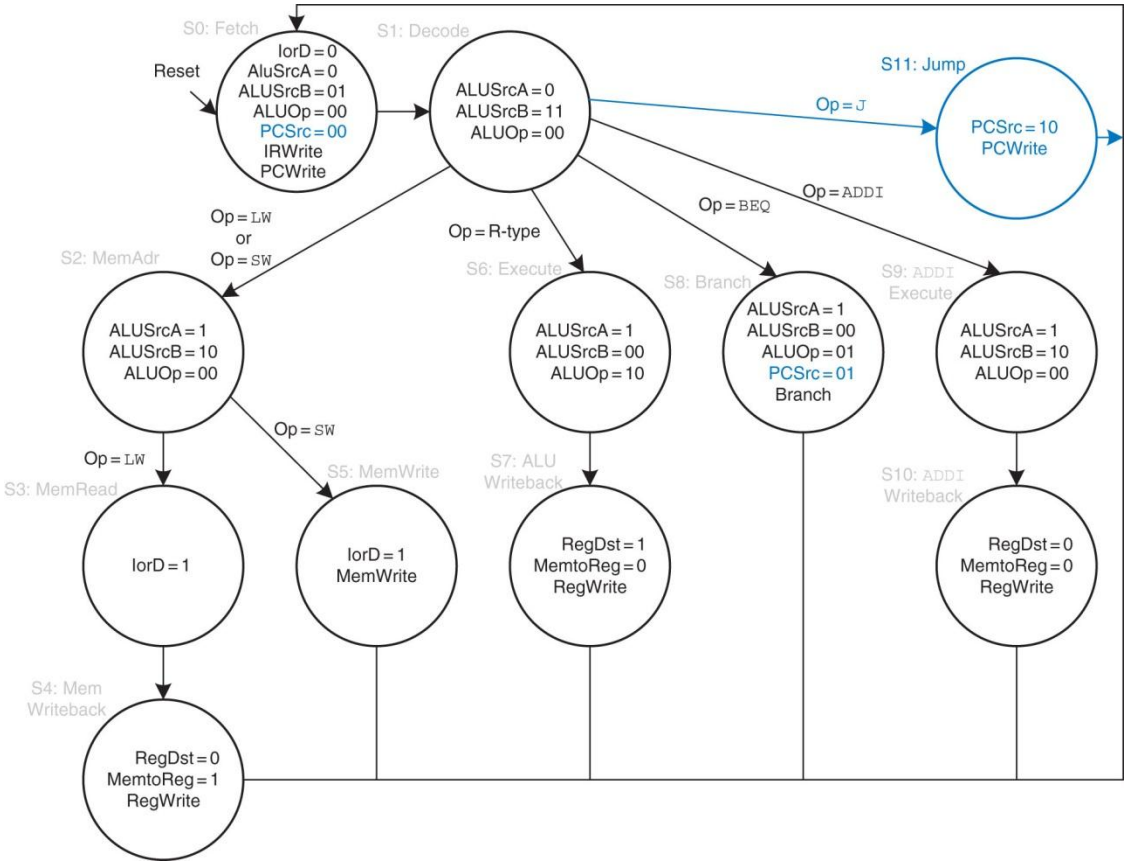


Figure 7.42 Main controller state for j

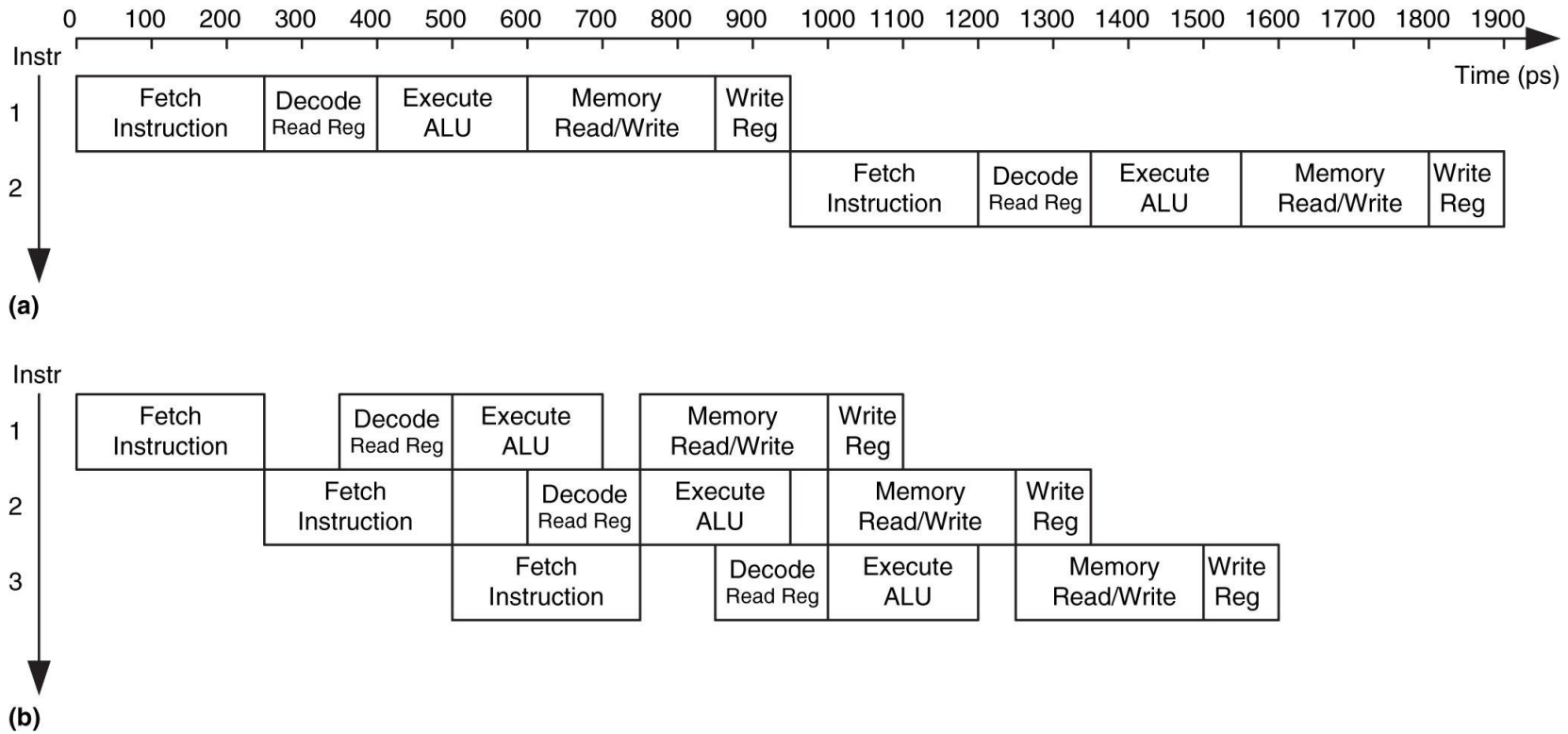


Figure 7.43 Timing diagrams: (a) single-cycle processor, (b) pipelined processor

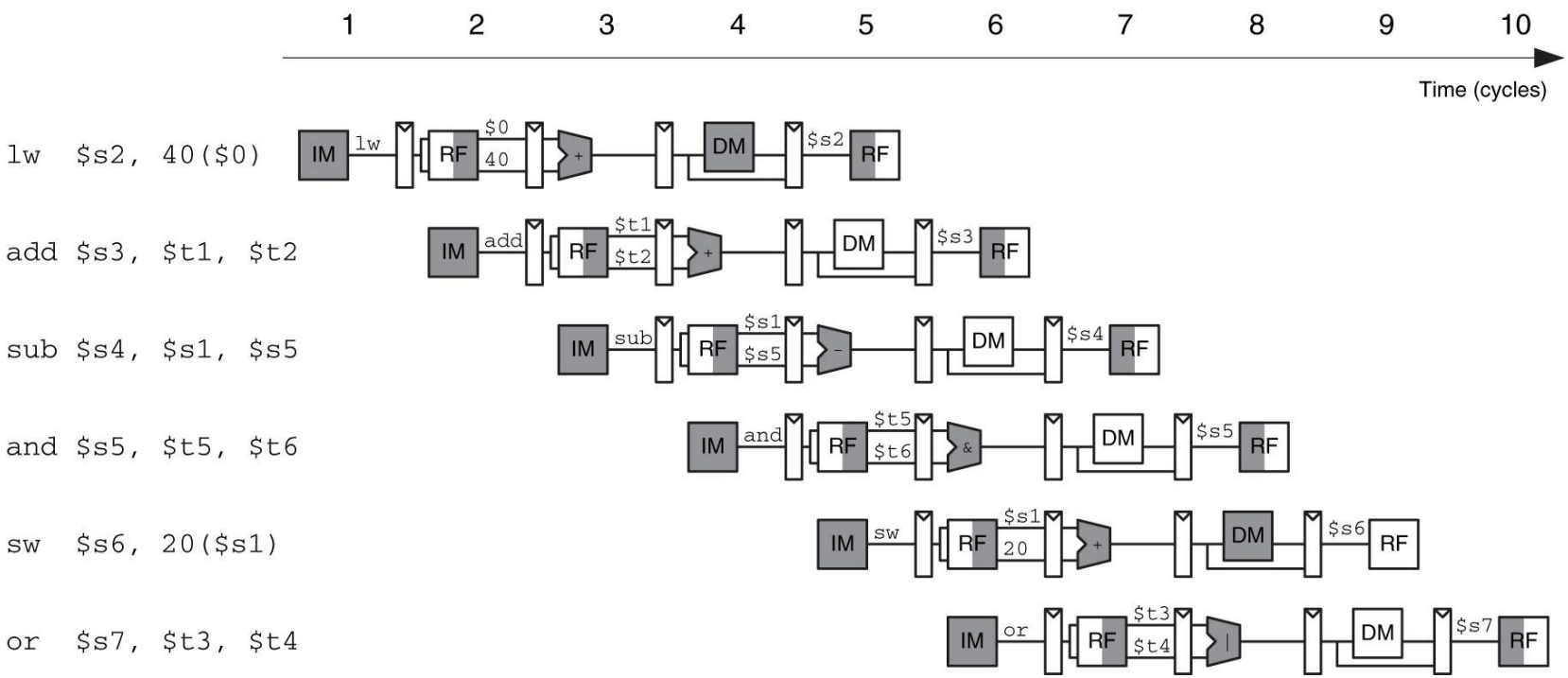


Figure 7.44 Abstract view of pipeline in operation

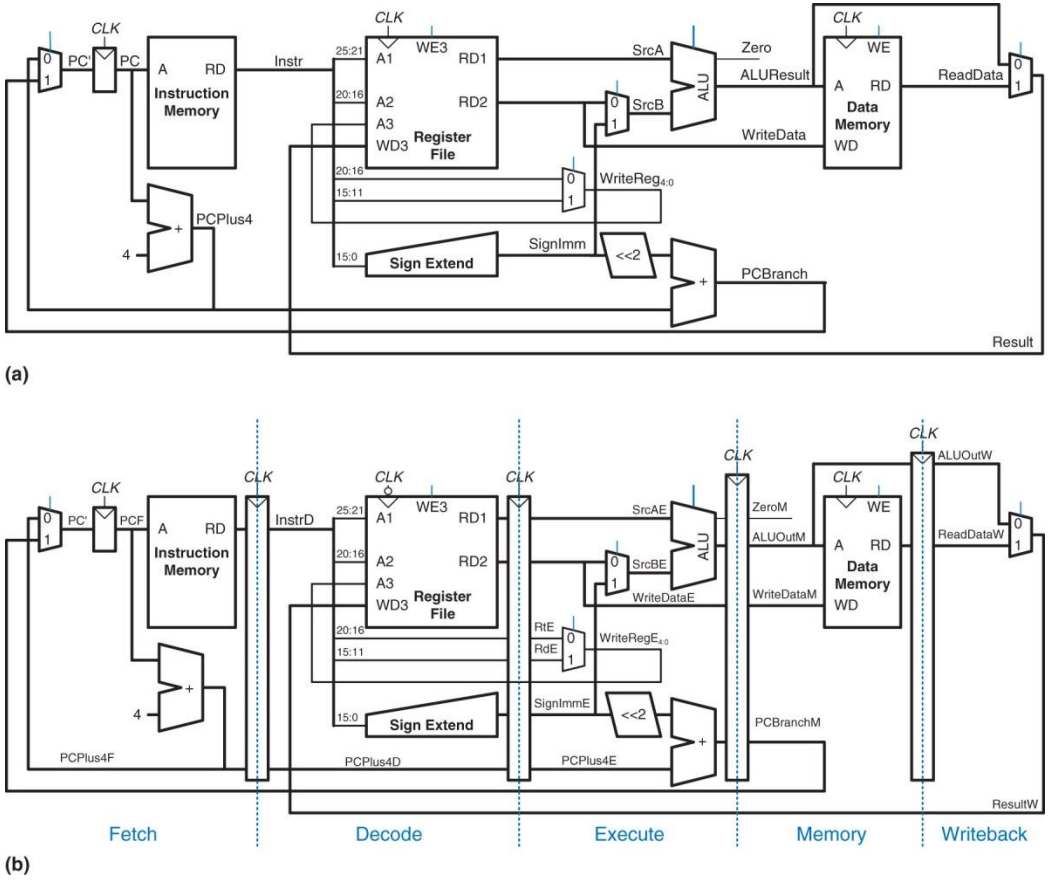


Figure 7.45 Single-cycle and pipelined datapaths

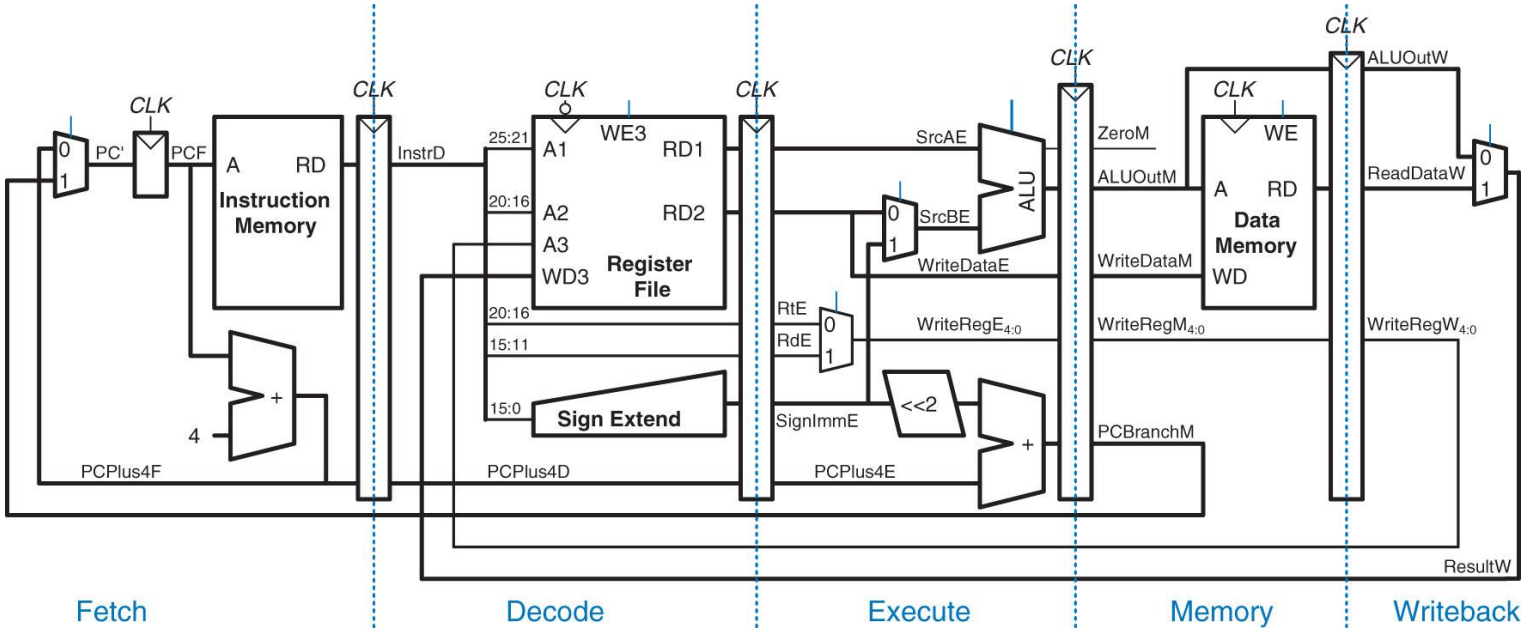


Figure 7.46 Corrected pipelined datapath

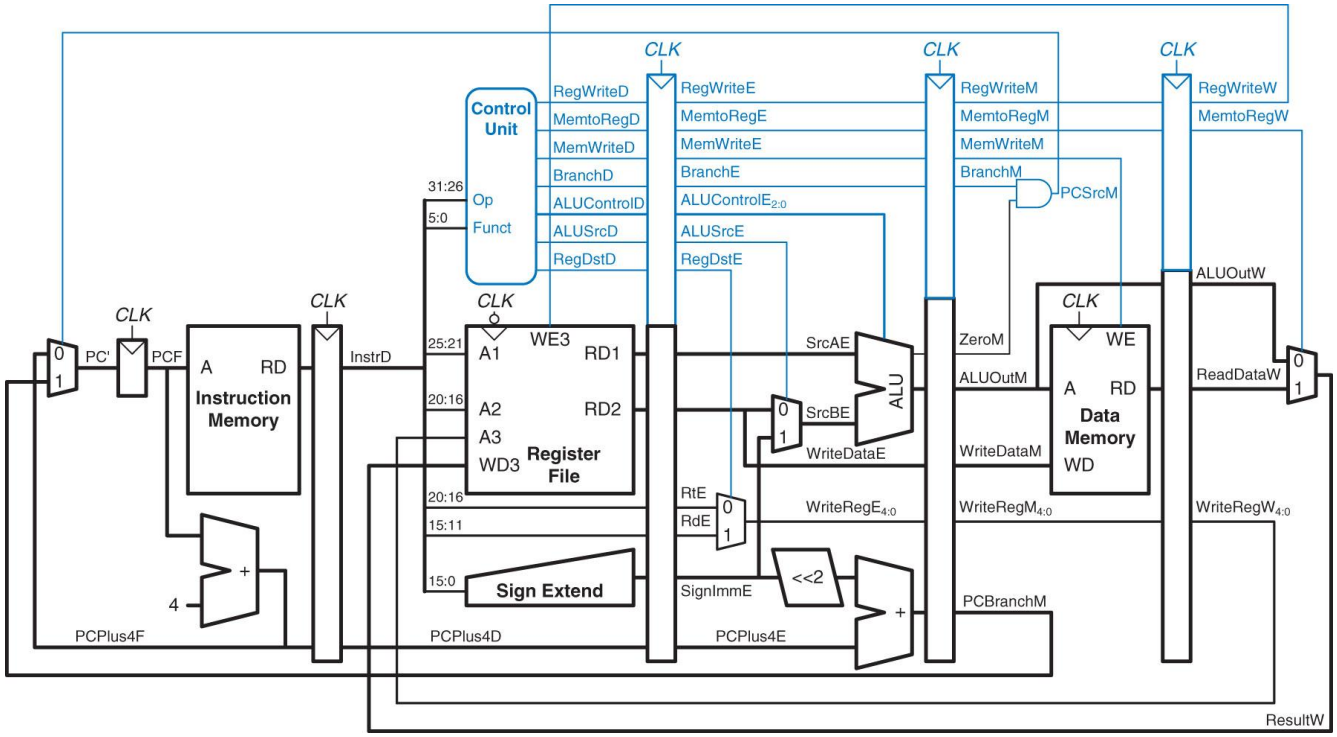


Figure 7.47 Pipelined processor with control

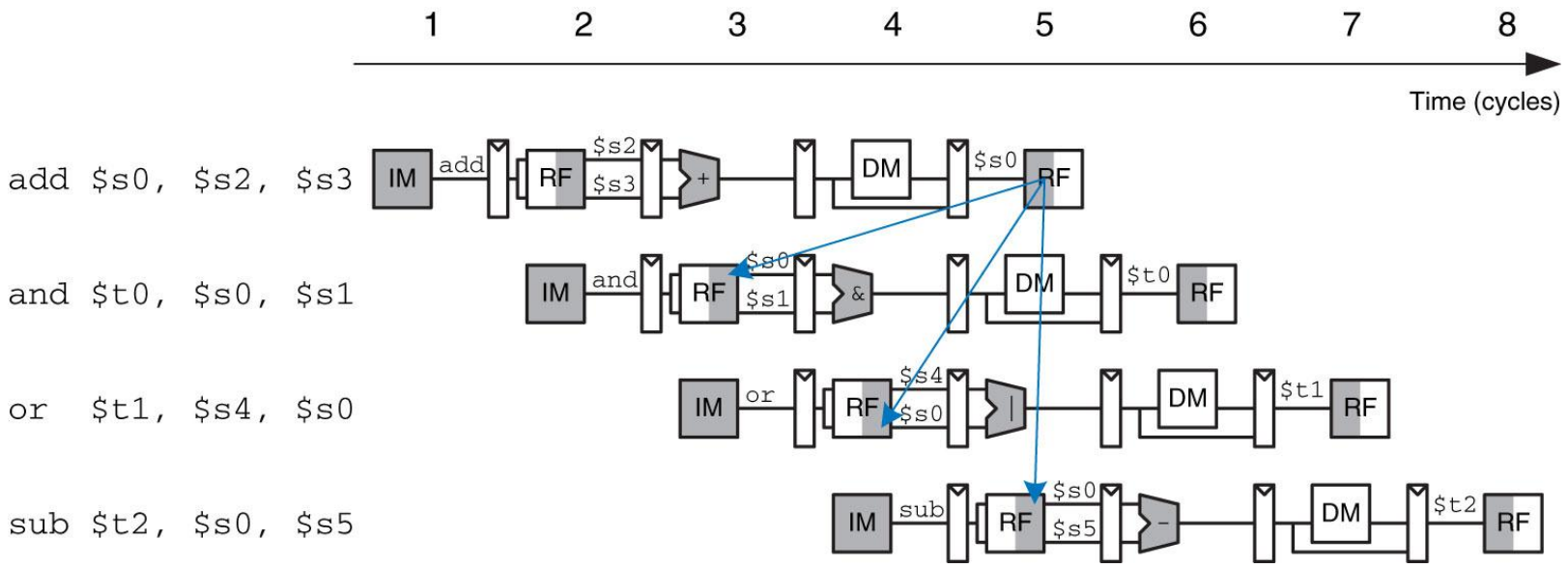


Figure 7.48 Abstract pipeline diagram illustrating hazards

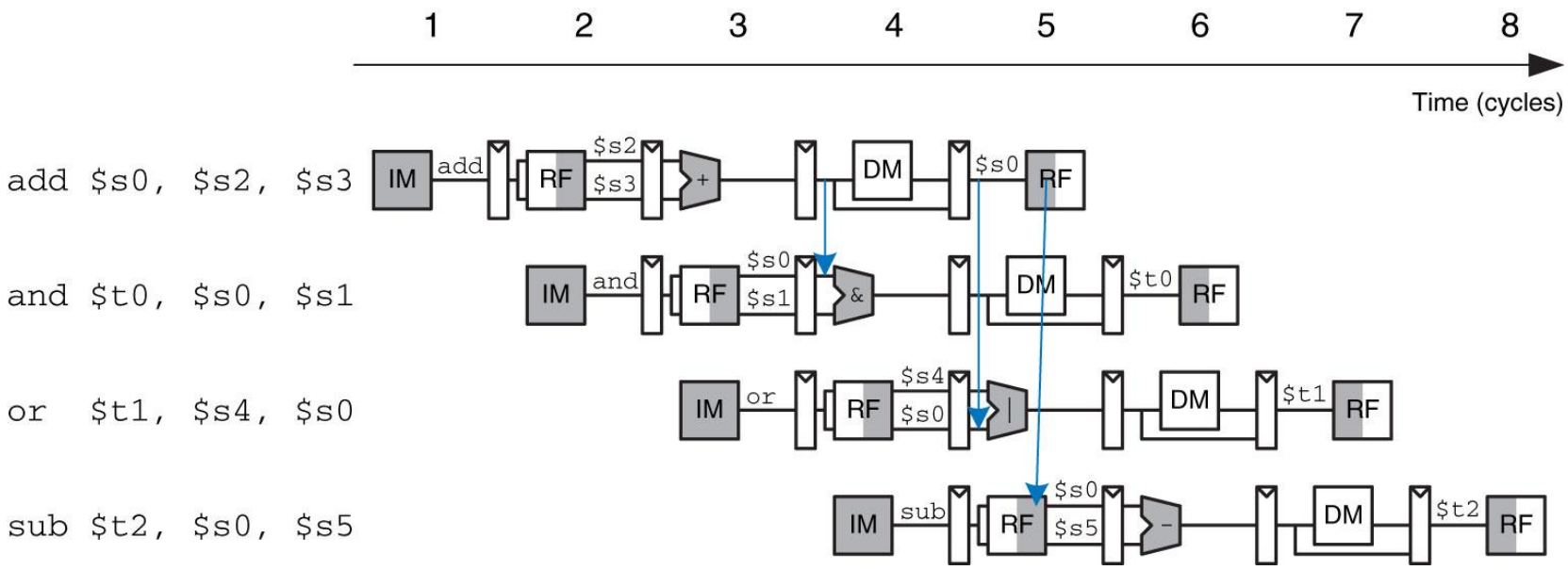


Figure 7.49 Abstract pipeline diagram illustrating forwarding

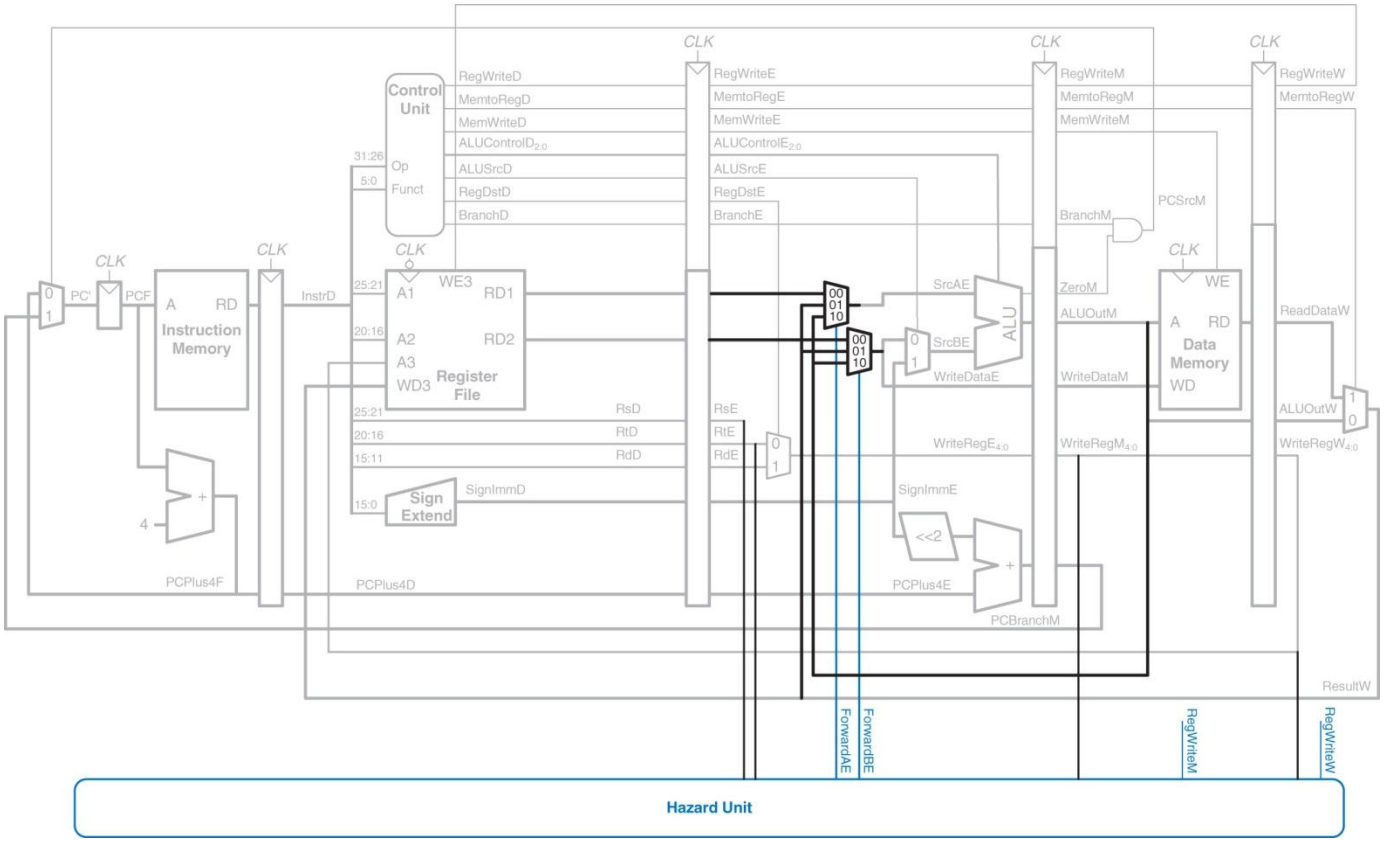


Figure 7.50 Pipelined processor with forwarding to solve hazards

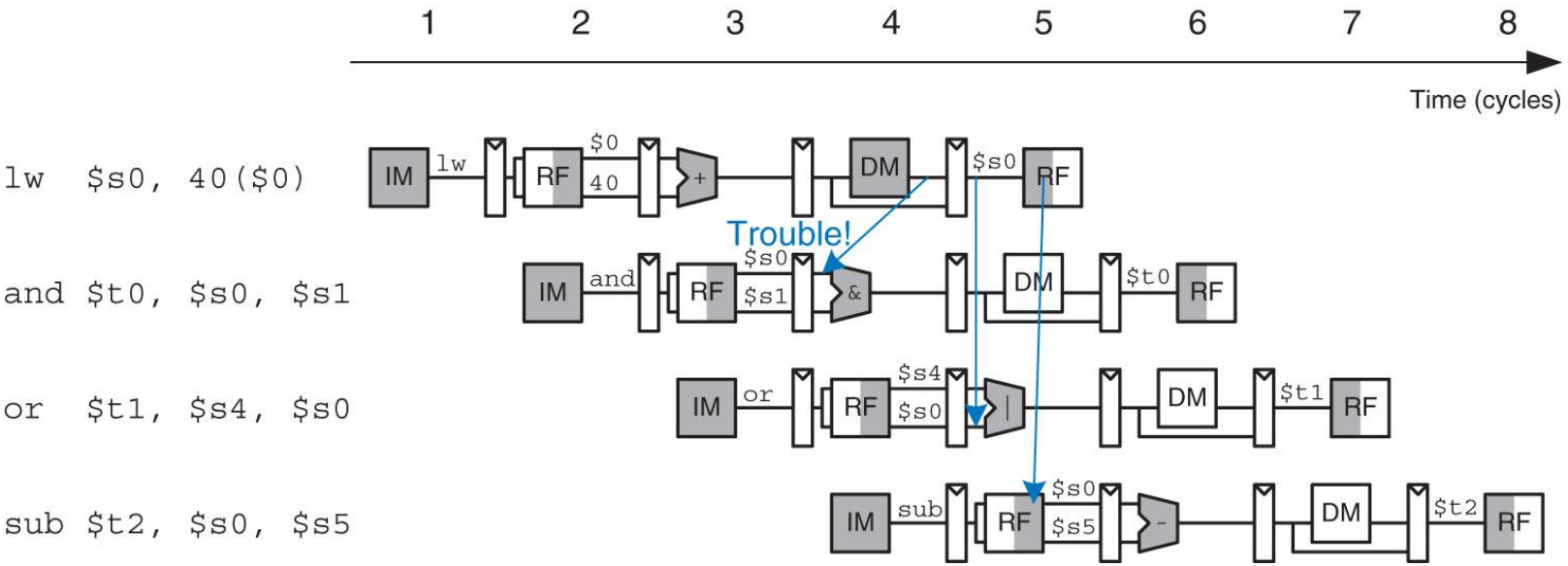


Figure 7.51 Abstract pipeline diagram illustrating trouble forwarding from lw

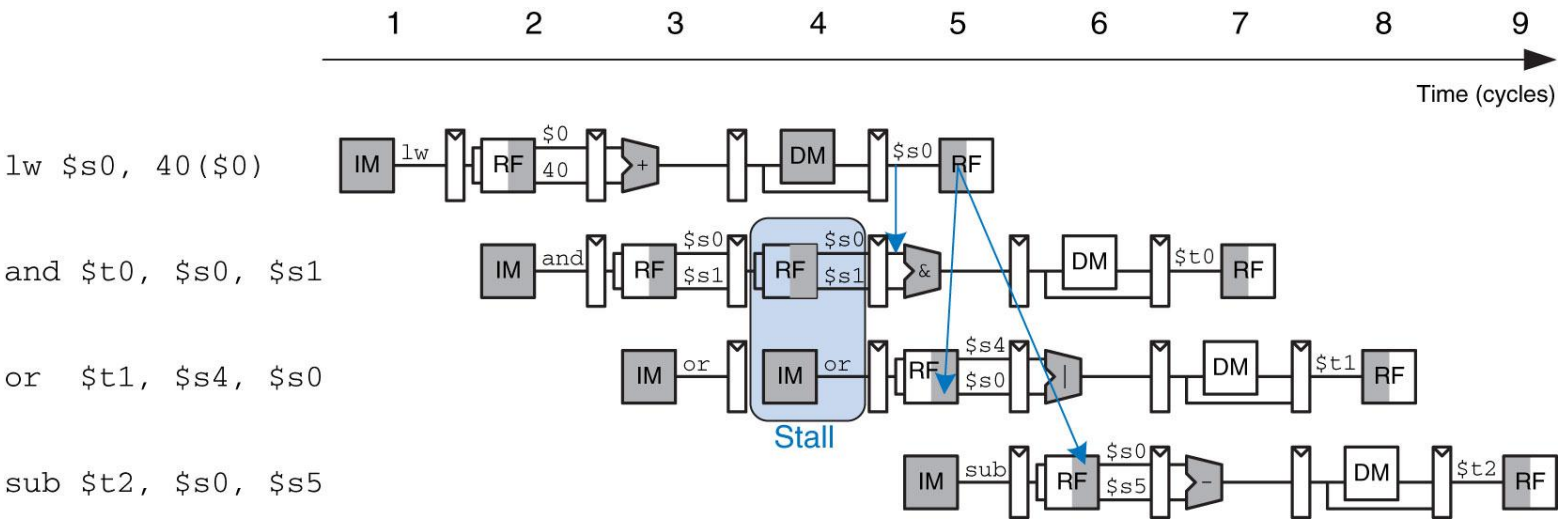


Figure 7.52 Abstract pipeline diagram illustrating stall to solve hazards

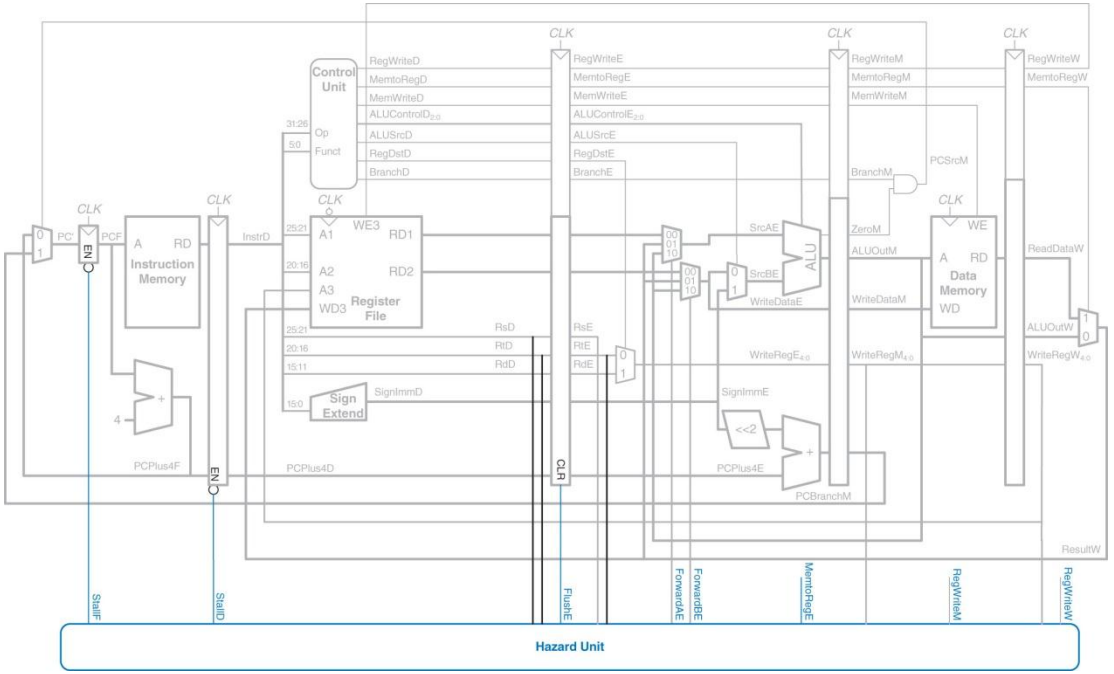


Figure 7.53 Pipelined processor with stalls to solve lw data hazard

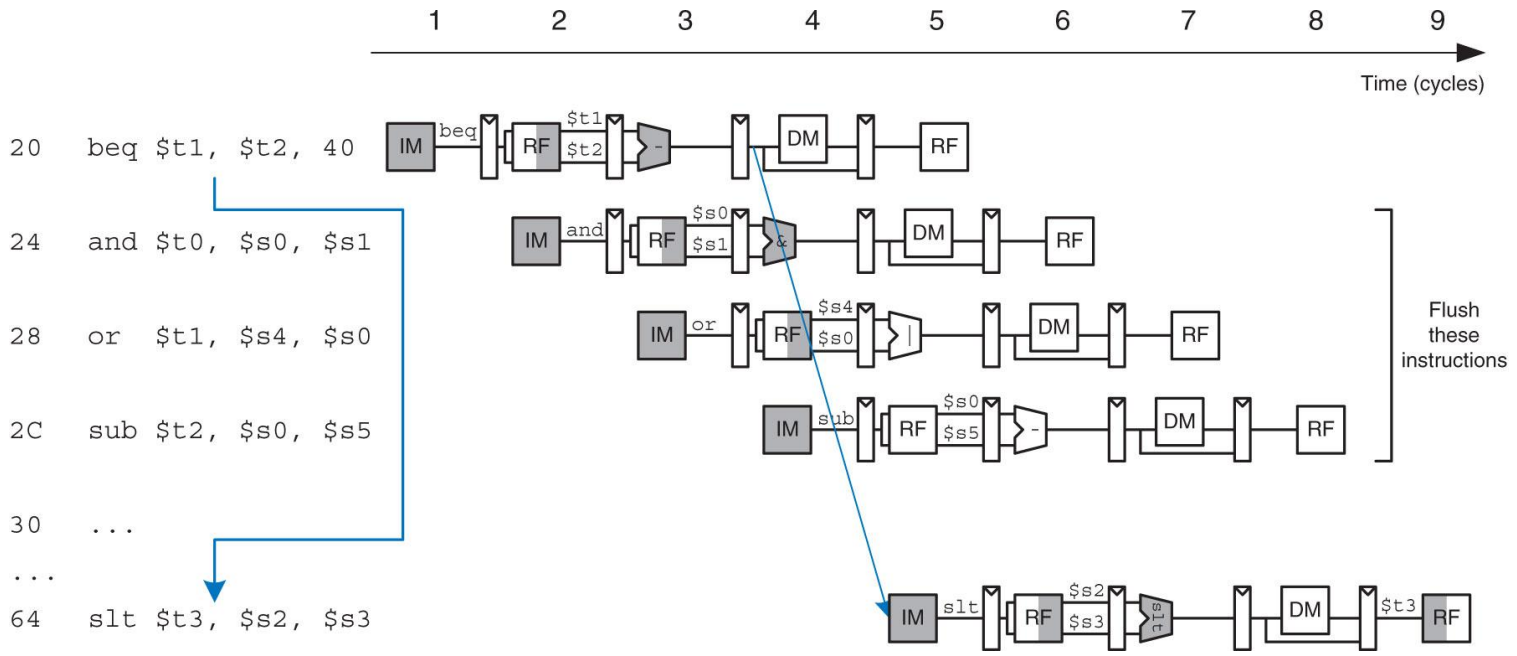


Figure 7.54 Abstract pipeline diagram illustrating flushing when a branch is taken

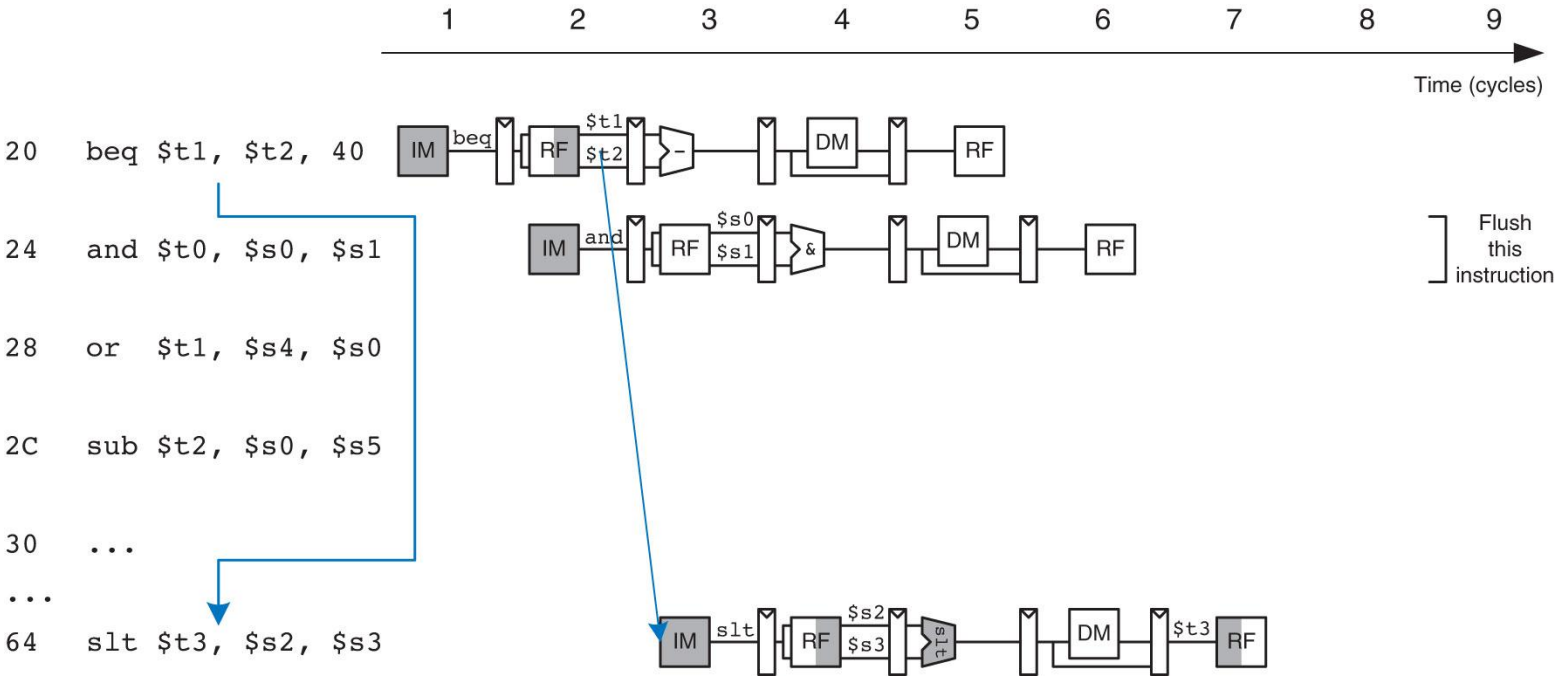


Figure 7.55 Abstract pipeline diagram illustrating earlier branch decision

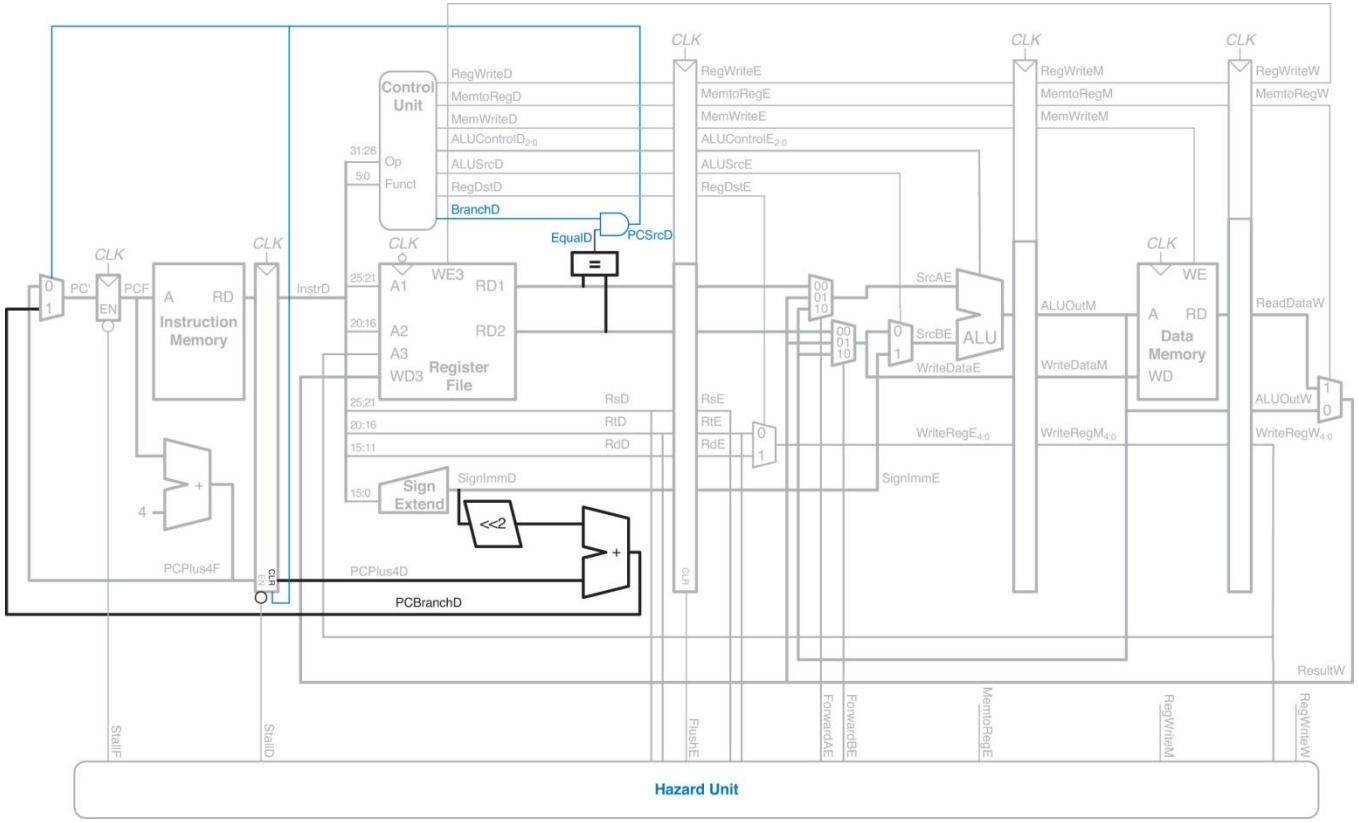


Figure 7.56 Pipelined processor handling branch control hazard

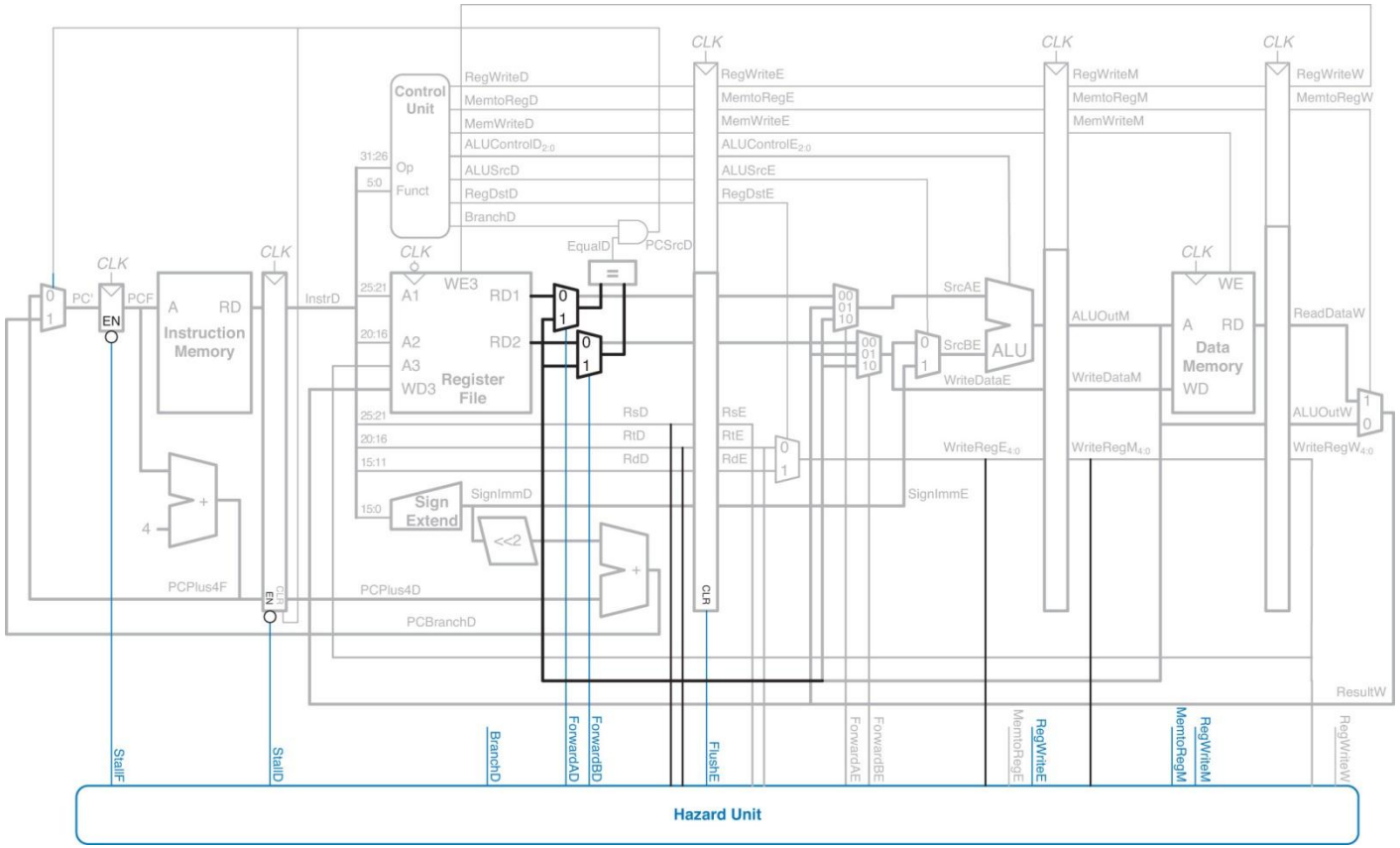


Figure 7.57 Pipelined processor handling data dependencies for branch instructions

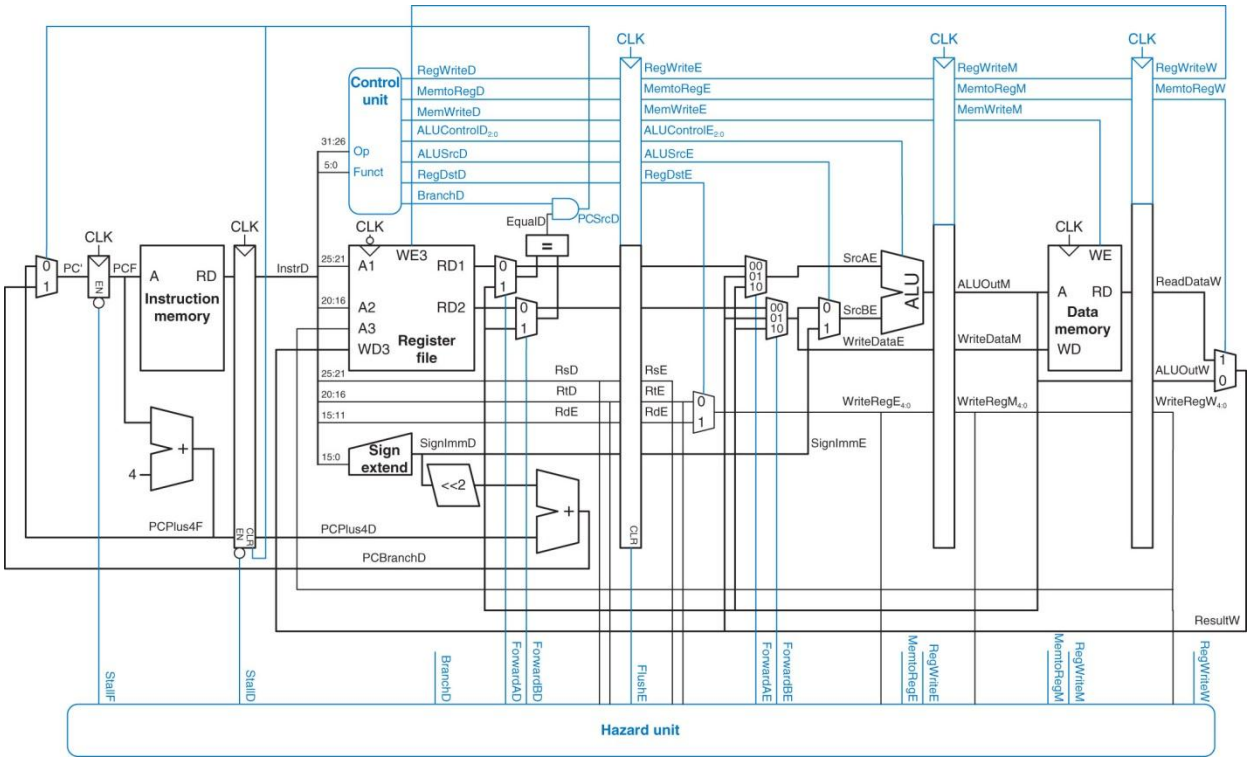


Figure 7.58 Pipelined processor with full hazard handling

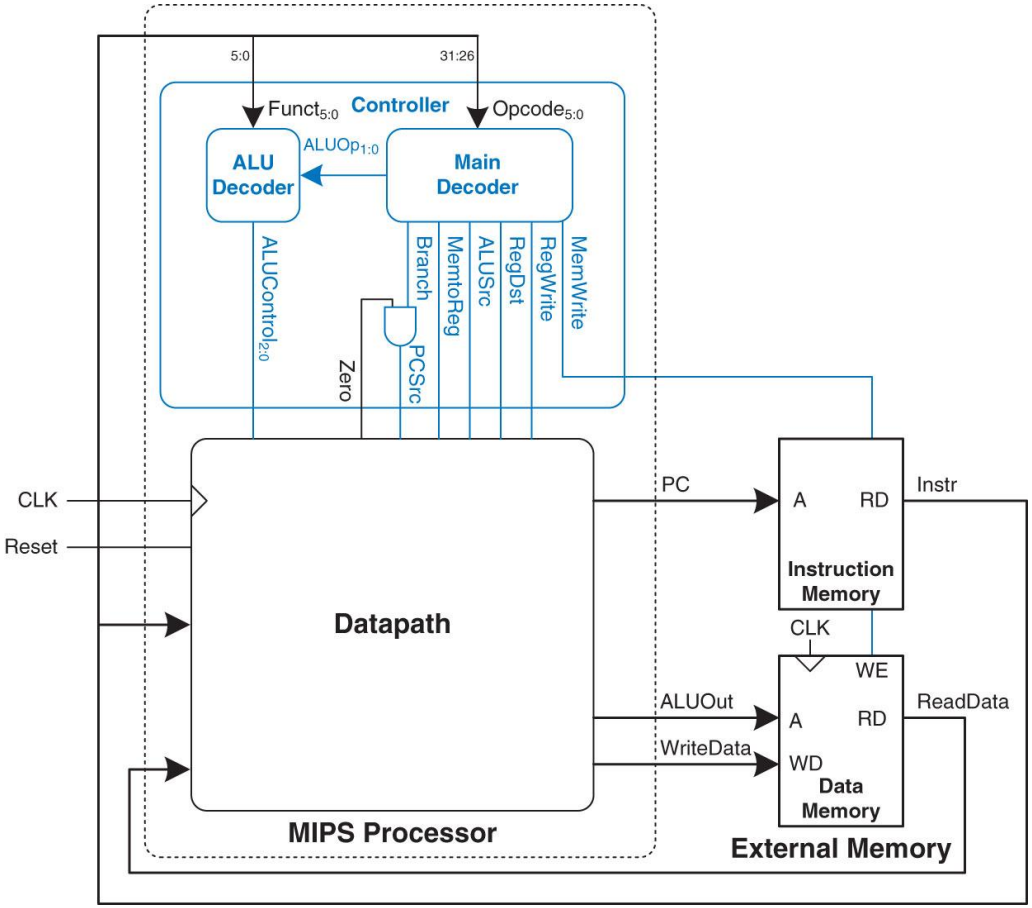


Figure 7.59 MIPS single-cycle processor interfaced to external memory

```
#          Assembly              Description                Address    Machine
main:      addi $2, $0, 5        # initialize $2 = 5        0          20020005
           addi $3, $0, 12       # initialize $3 = 12       4          2003000c
           addi $7, $3, -9       # initialize $7 = 3        8          2067fff7
           or   $4, $7, $2       # $4 = (3 OR 5) = 7        c          00e22025
           and  $5, $3, $4       # $5 = (12 AND 7) = 4      10         00642824
           add  $5, $5, $4       # $5 = 4 + 7 = 11          14         00a42820
           beq  $5, $7, end      # shouldn't be taken       18         10a7000a
           slt  $4, $3, $4       # $4 = 12 < 7 = 0          1c         0064202a
           beq  $4, $0, around   # should be taken          20         10800001
           addi $5, $0, 0        # shouldn't happen         24         20050000
around:    slt  $4, $7, $2       # $4 = 3 < 5 = 1           28         00e2202a
           add  $7, $4, $5       # $7 = 1 + 11 = 12         2c         00853820
           sub  $7, $7, $2       # $7 = 12 - 5 = 7          30         00e23822
           sw   $7, 68($3)       # [80] = 7                 34         ac670044
           lw   $2, 80($0)       # $2 = [80] = 7            38         8c020050
           j    end              # should be taken          3c         08000011
           addi $2, $0, 1        # shouldn't happen         40         20020001
end:       sw   $2, 84($0)       # write mem[84] = 7        44         ac020054
```

Figure 7.60 Assembly and machine code for MIPS test program

```
20020005
2003000c
2067fff7
00e22025
00642824
00a42820
10a7000a
0064202a
10800001
20050000
00e2202a
00853820
00e23822
ac670044
8c020050
08000011
20020001
ac020054
```

Figure 7.61 Contents of memfile.dat

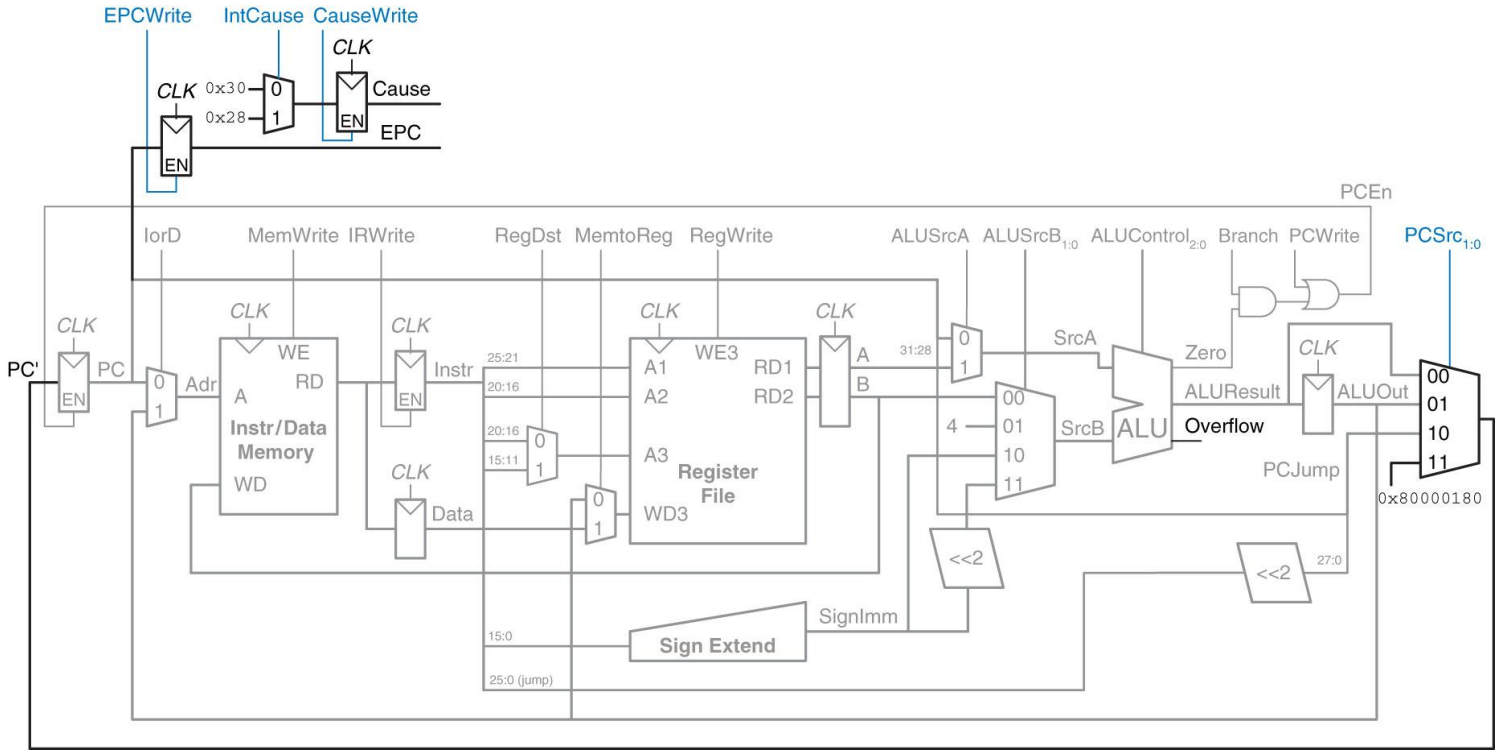


Figure 7.62 Datapath supporting overflow and undefined instruction exceptions

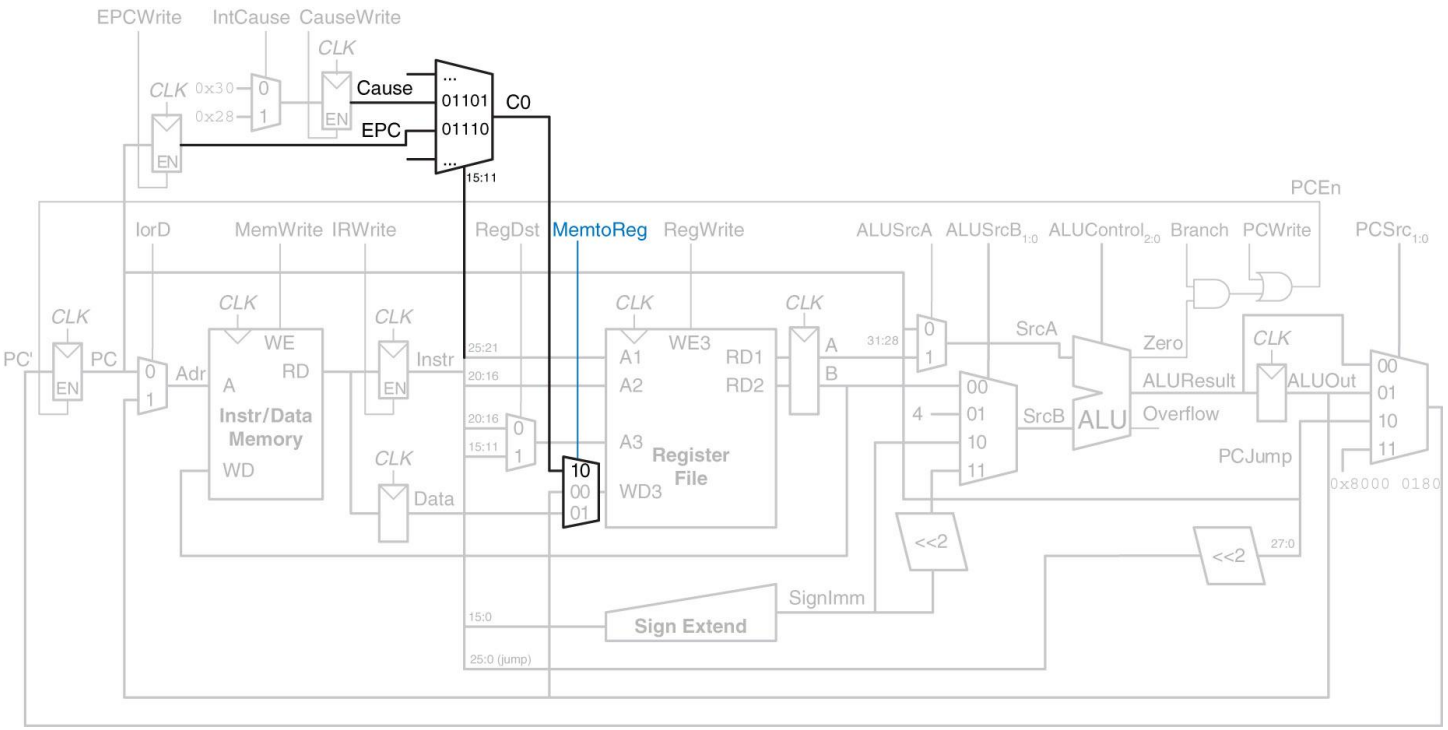


Figure 7.63 Datapath supporting mfcO

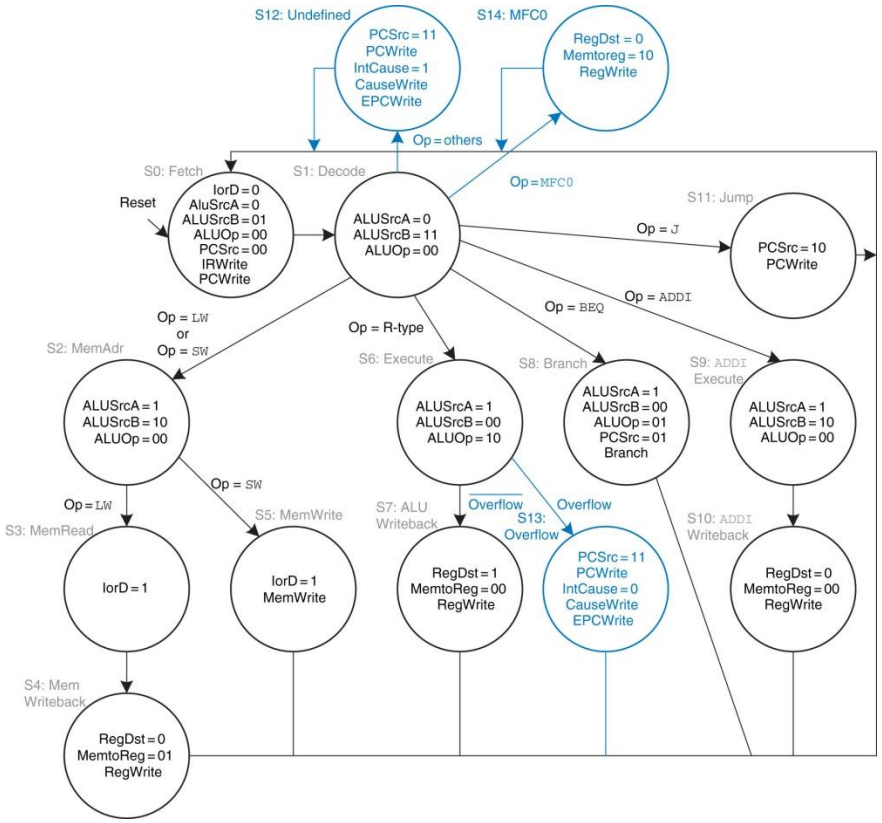


Figure 7.64 Controller supporting exceptions and mfc0

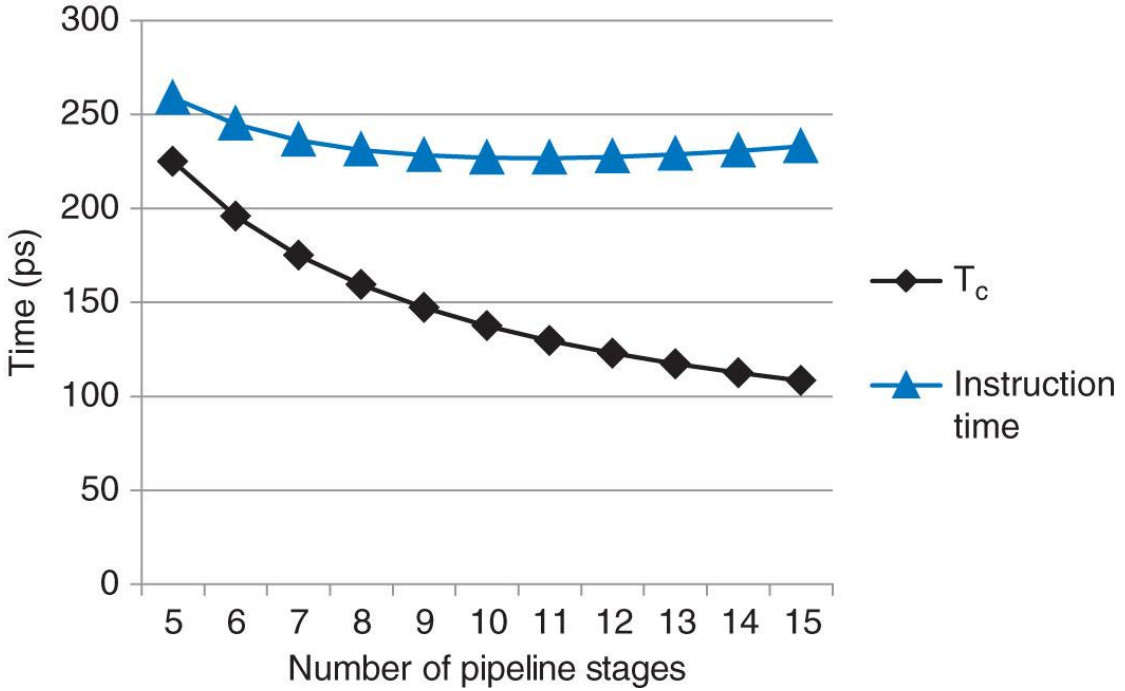


Figure 7.65 Cycle time and instruction time versus the number of pipeline stages

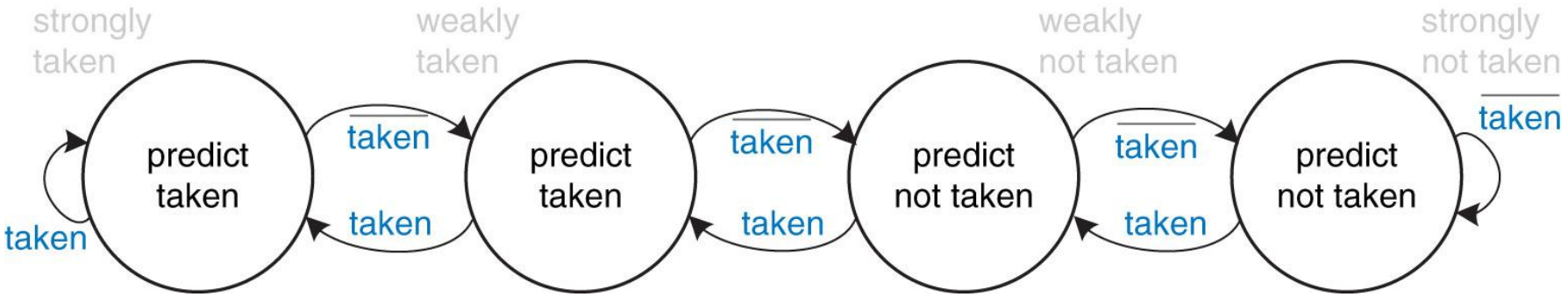


Figure 7.66 2-bit branch predictor state transition diagram

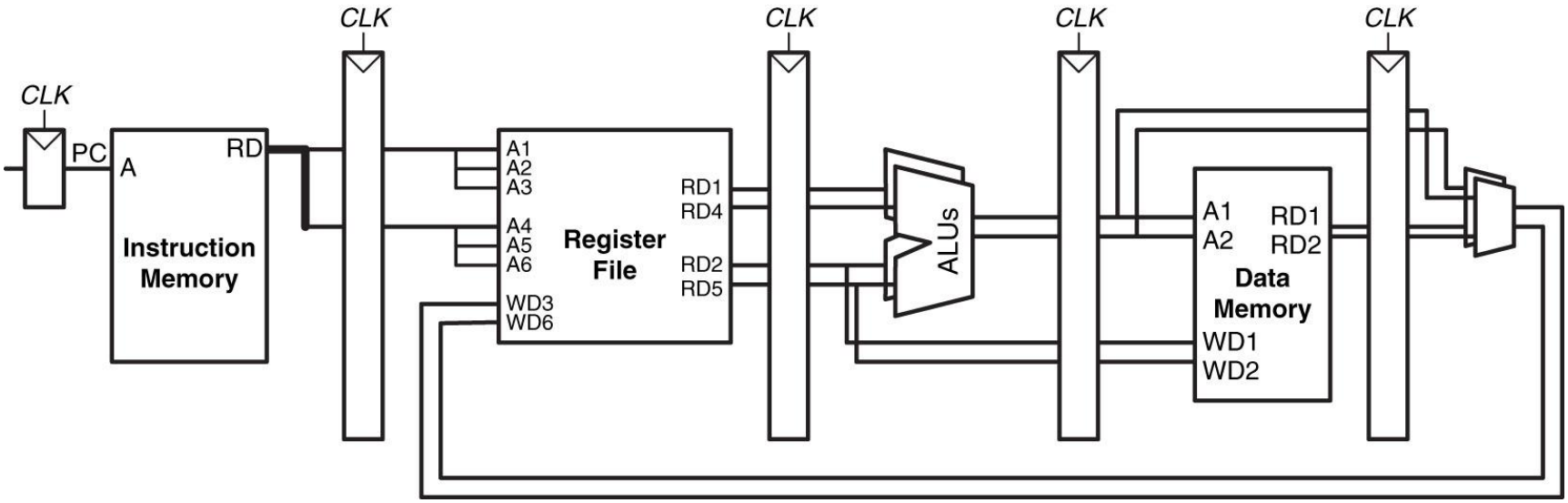


Figure 7.67 Superscalar datapath

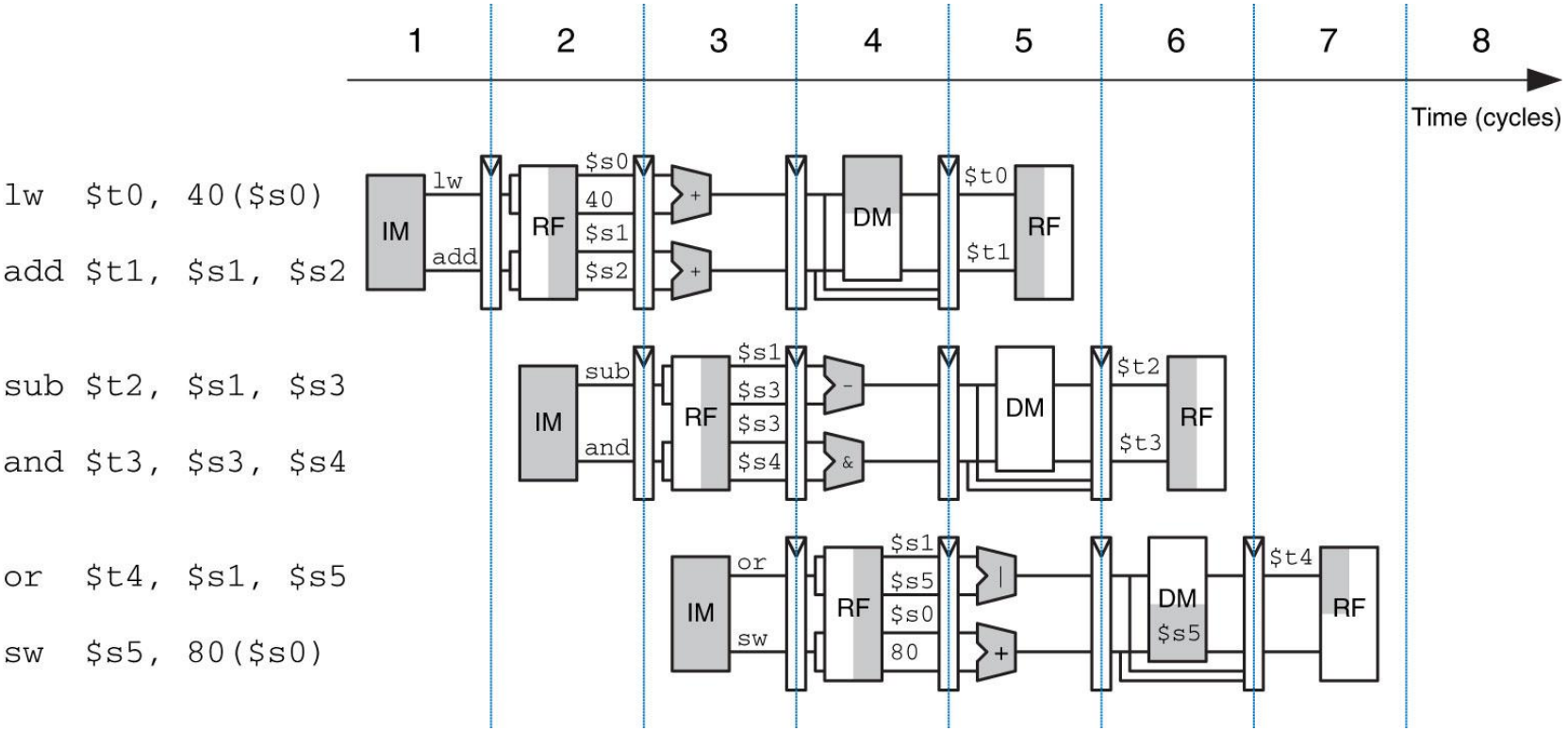


Figure 7.68 Abstract view of a superscalar pipeline in operation

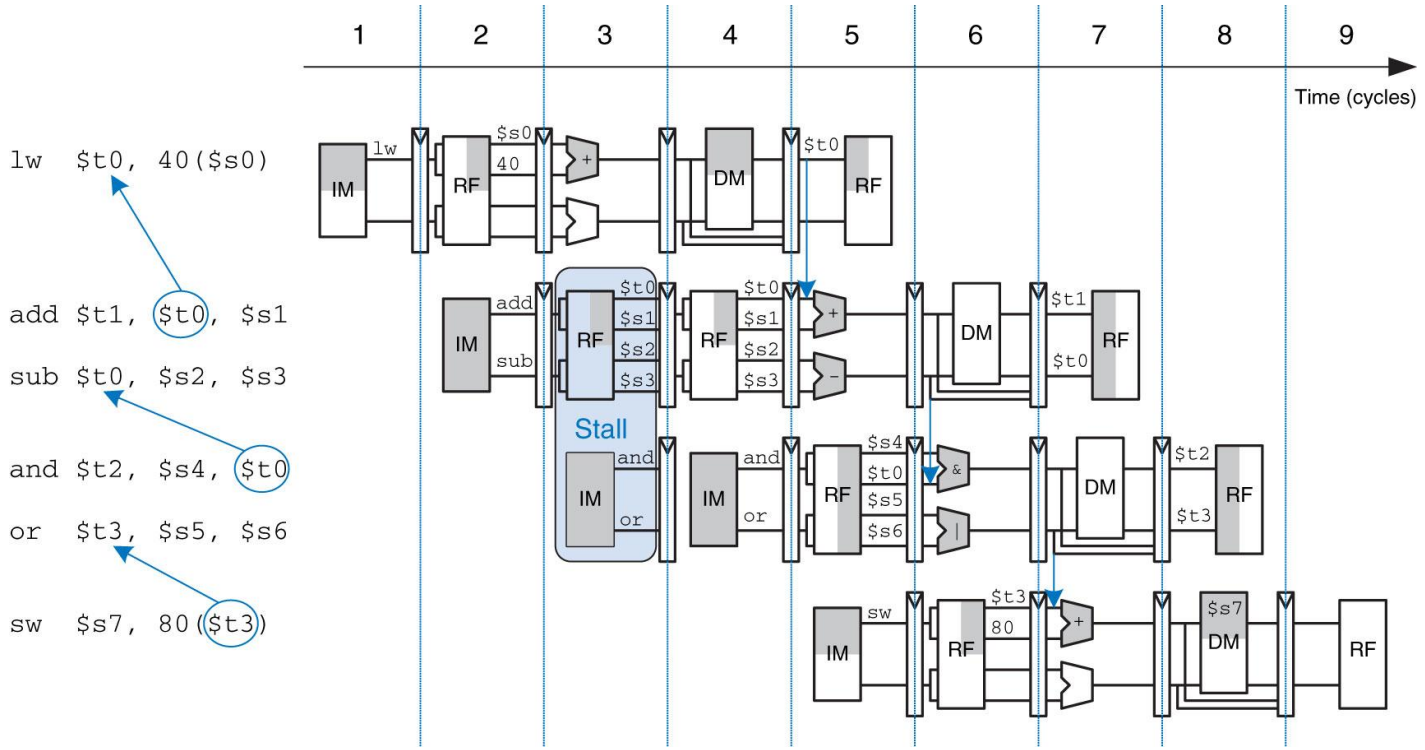


Figure 7.69 Program with data dependencies

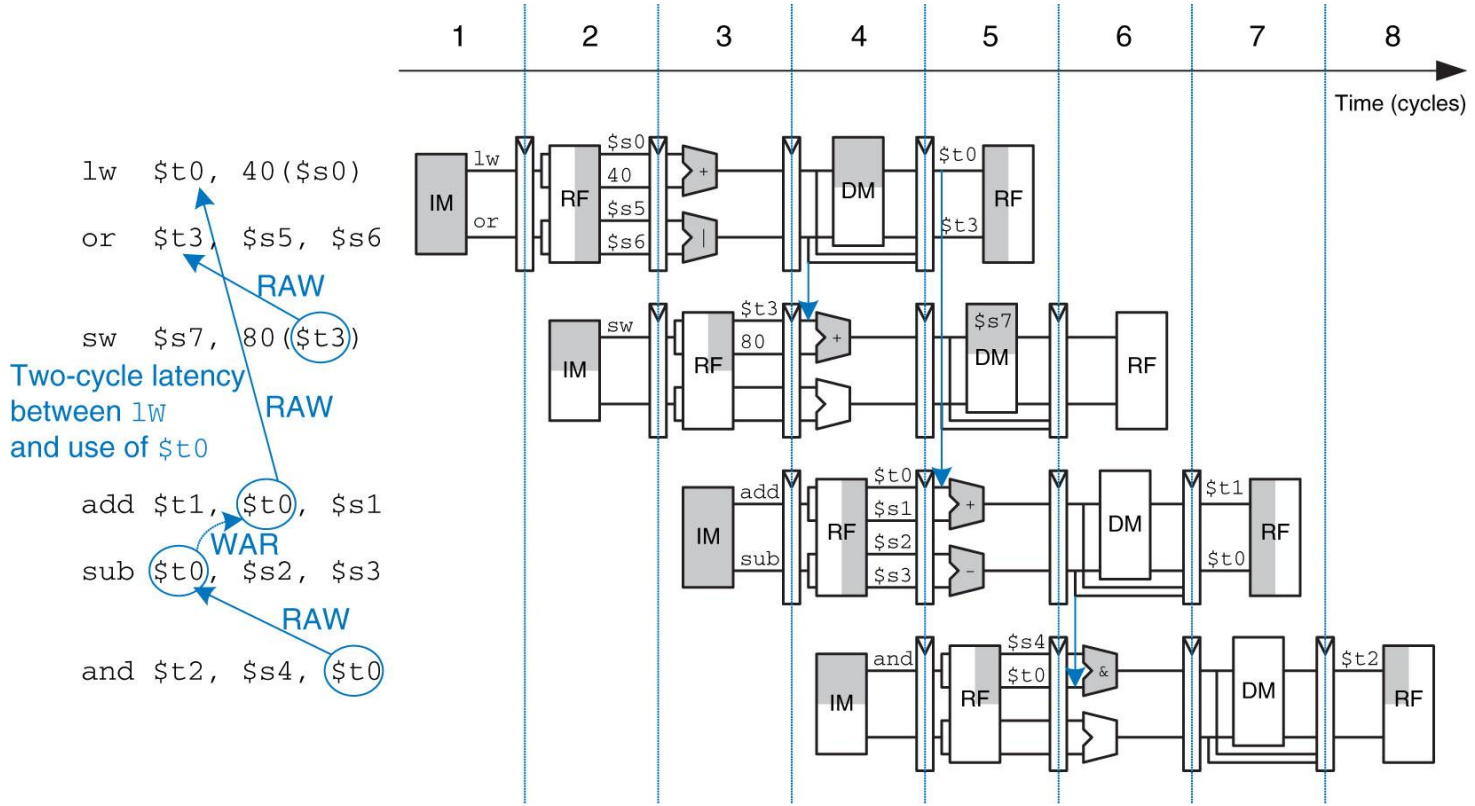


Figure 7.70 Out-of-order execution of a program with dependencies

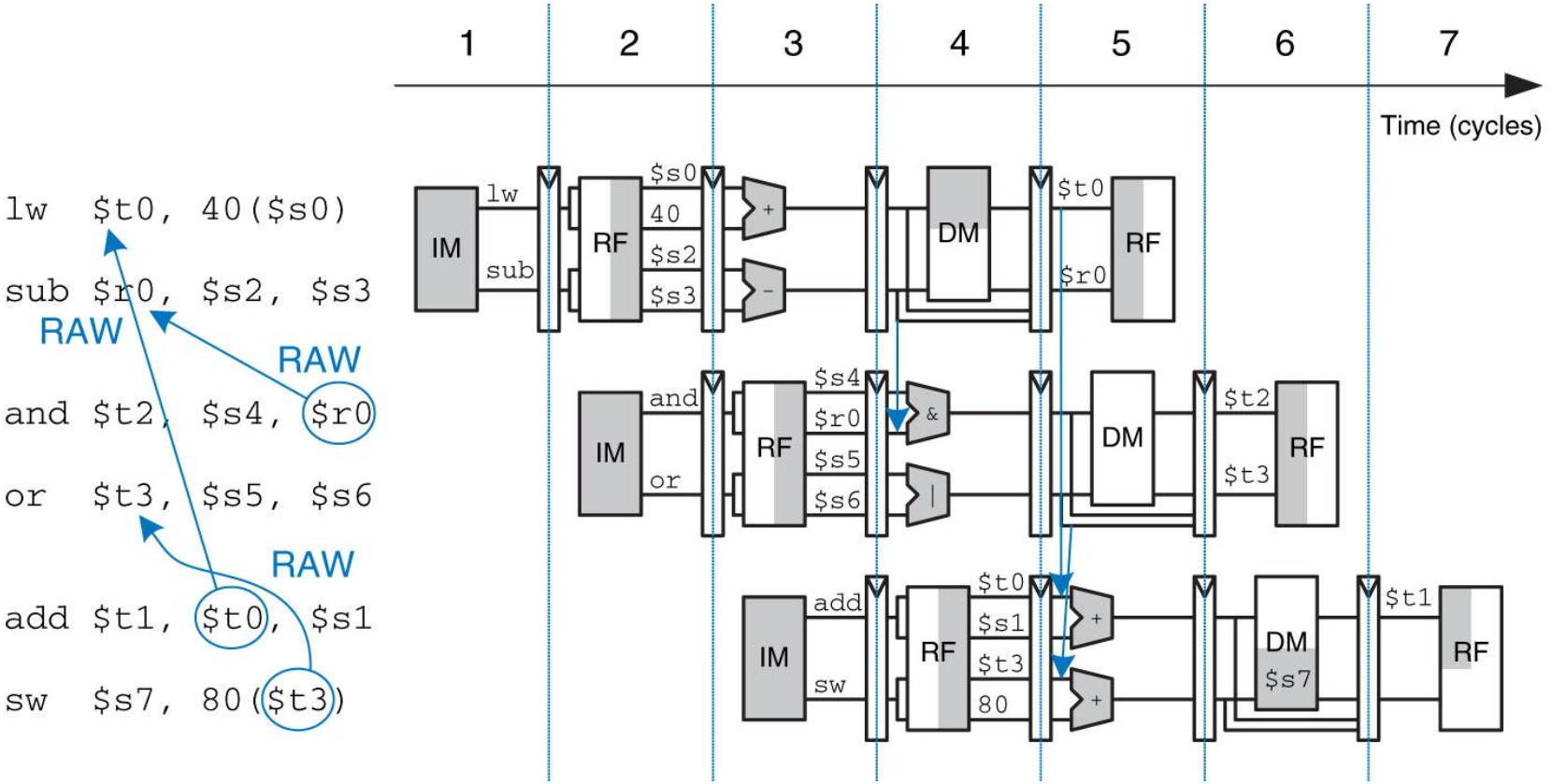


Figure 7.71 Out-of-order execution of a program using register renaming

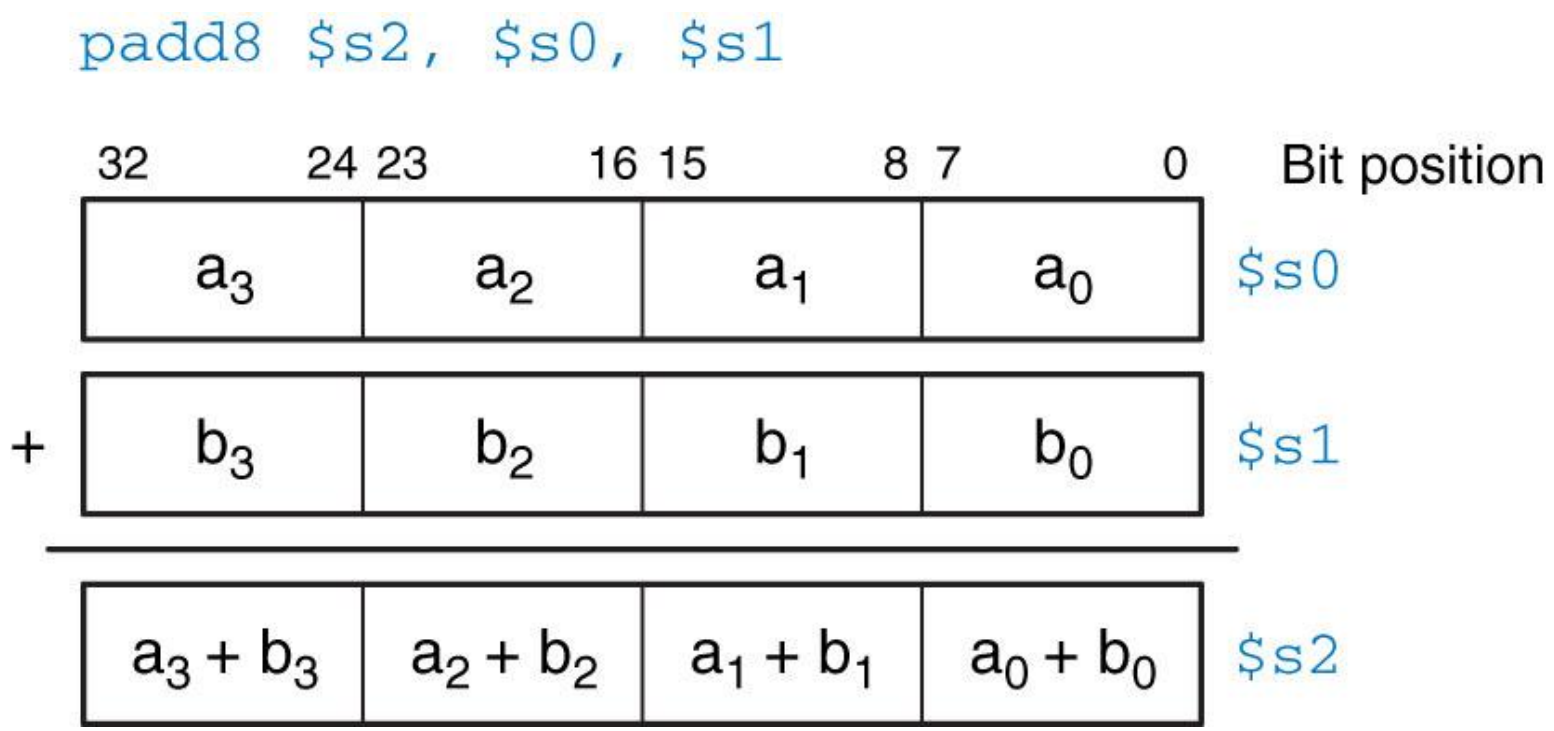


Figure 7.72 Packed arithmetic: four simultaneous 8-bit additions

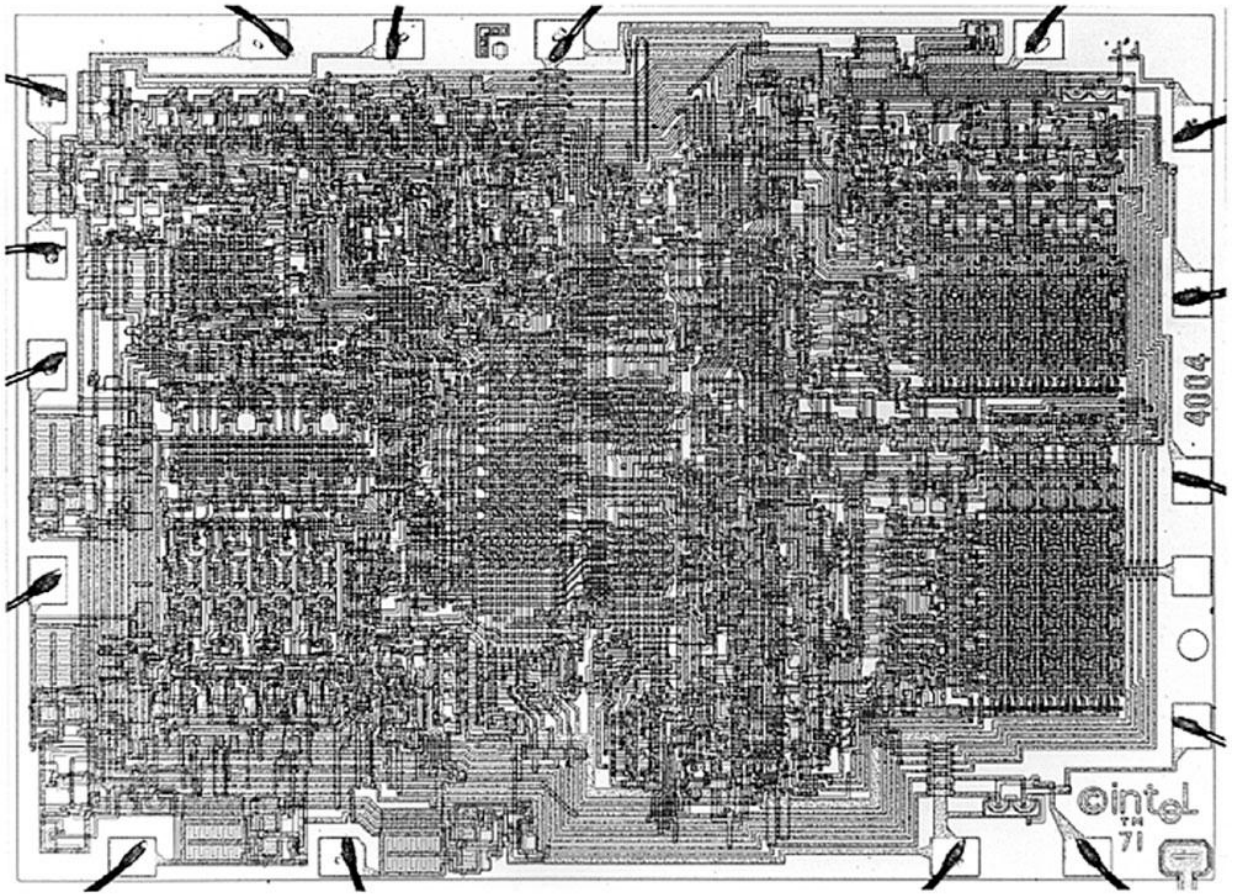


Figure 7.73 4004 microprocessor chip

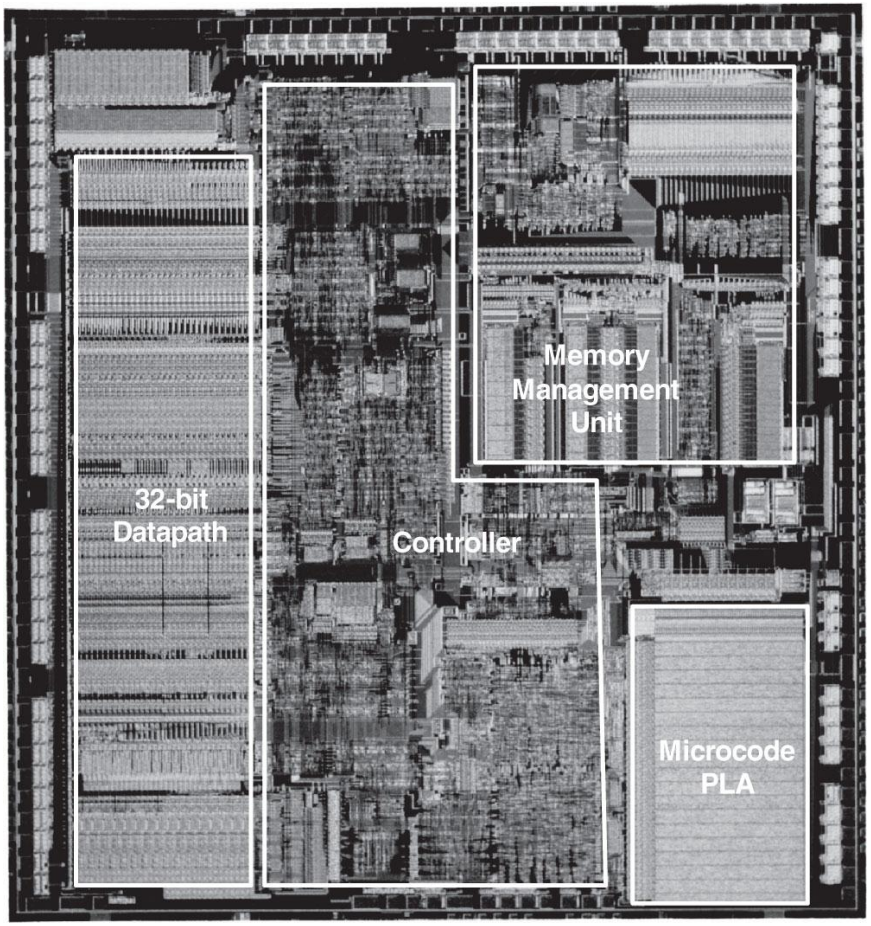


Figure 7.74 80386 microprocessor chip

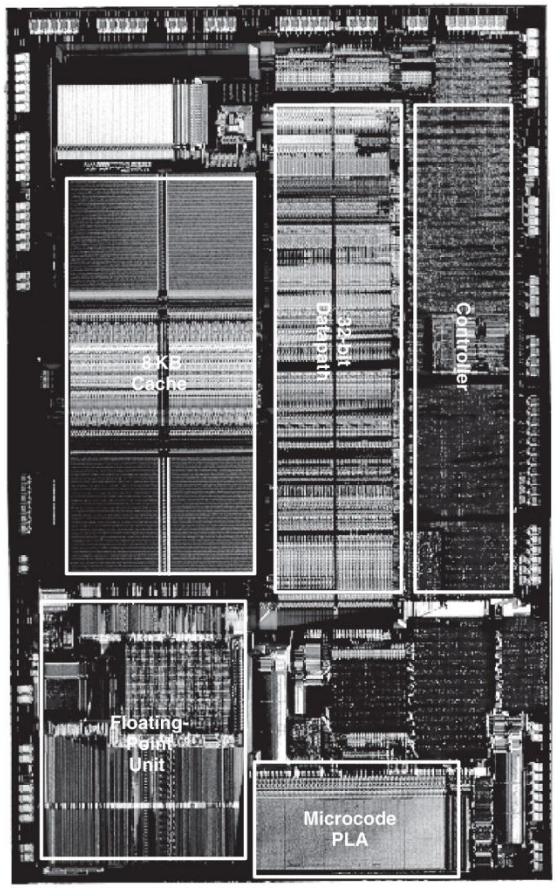


Figure 7.75 80486 microprocessor chip

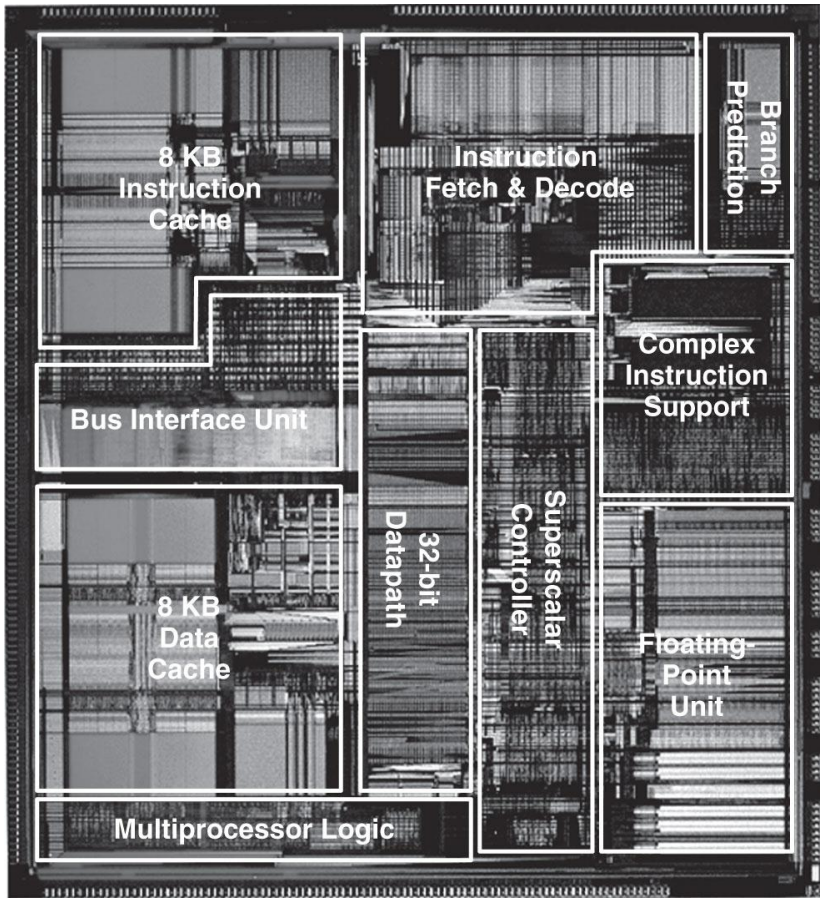


Figure 7.76 Pentium microprocessor chip

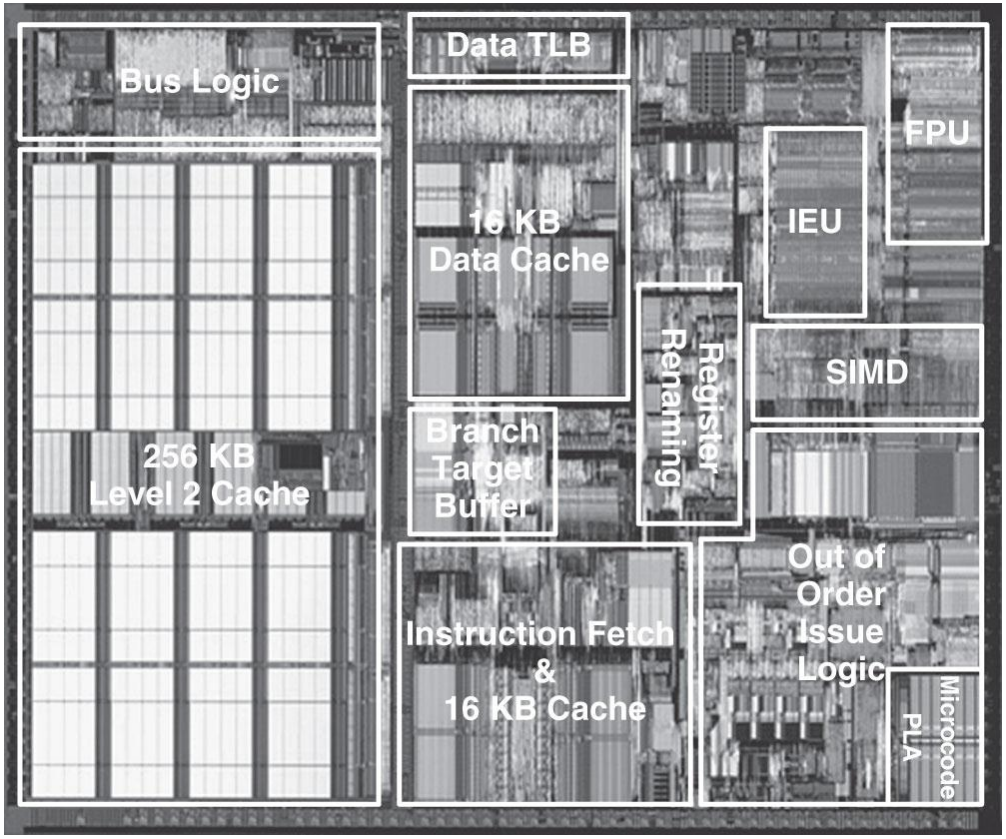


Figure 7.77 Pentium III microprocessor chip

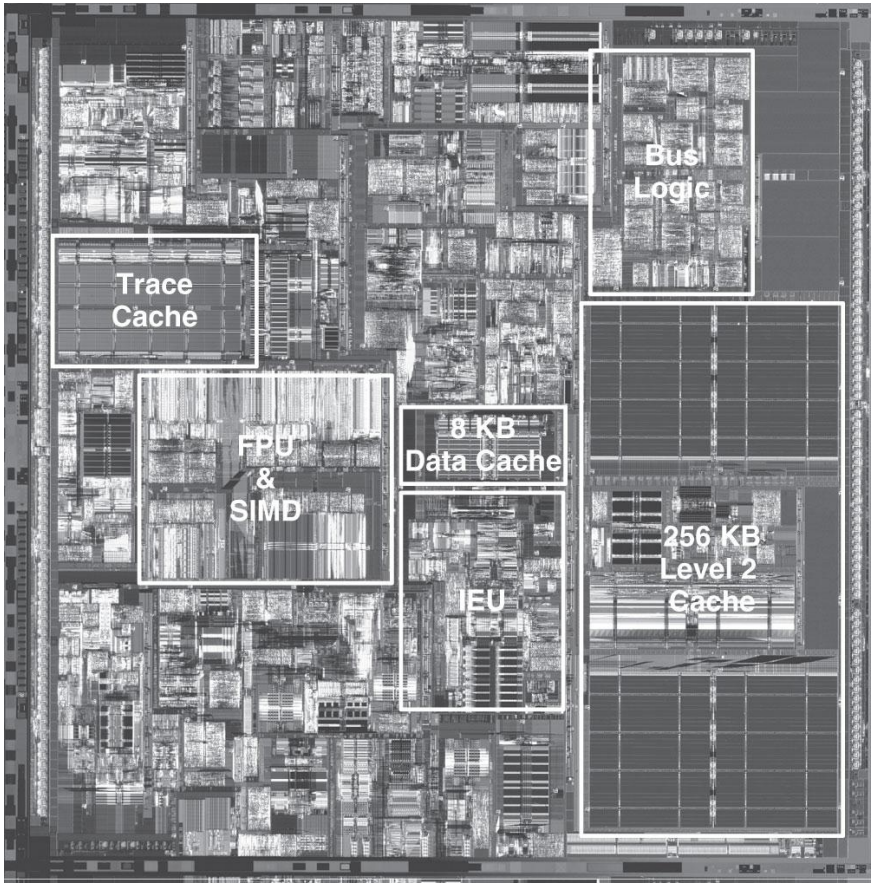


Figure 7.78 Pentium 4 microprocessor chip

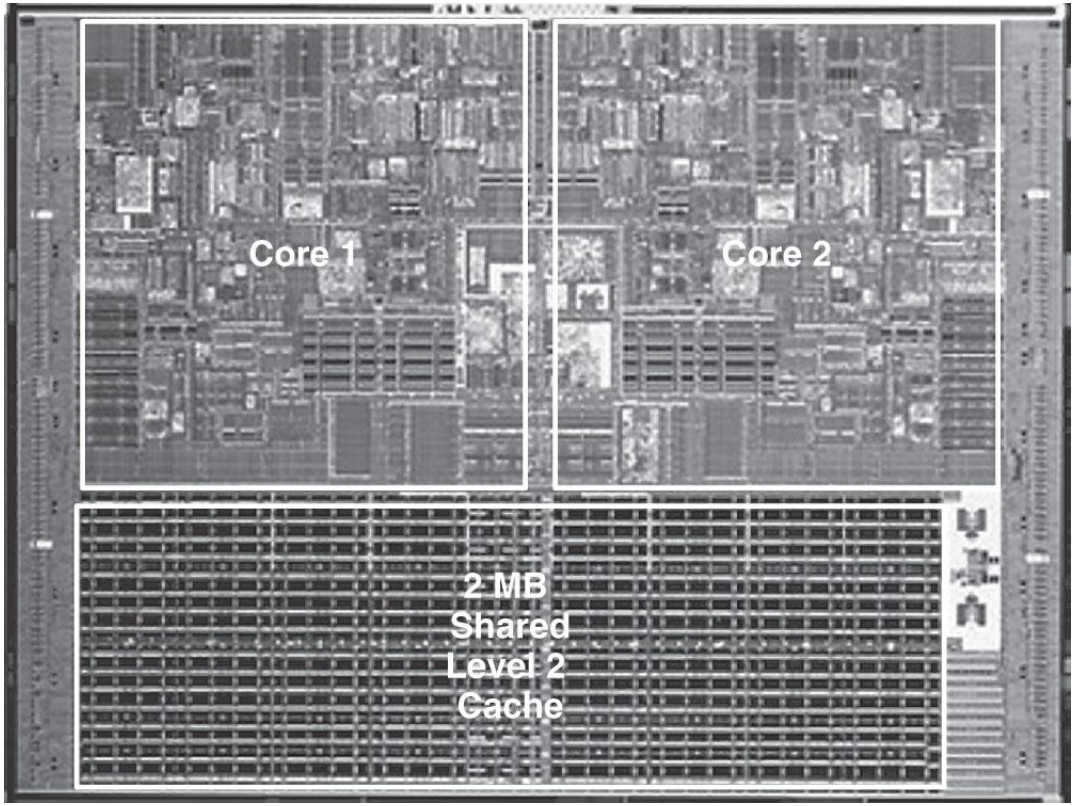


Figure 7.79 Core Duo microprocessor chip

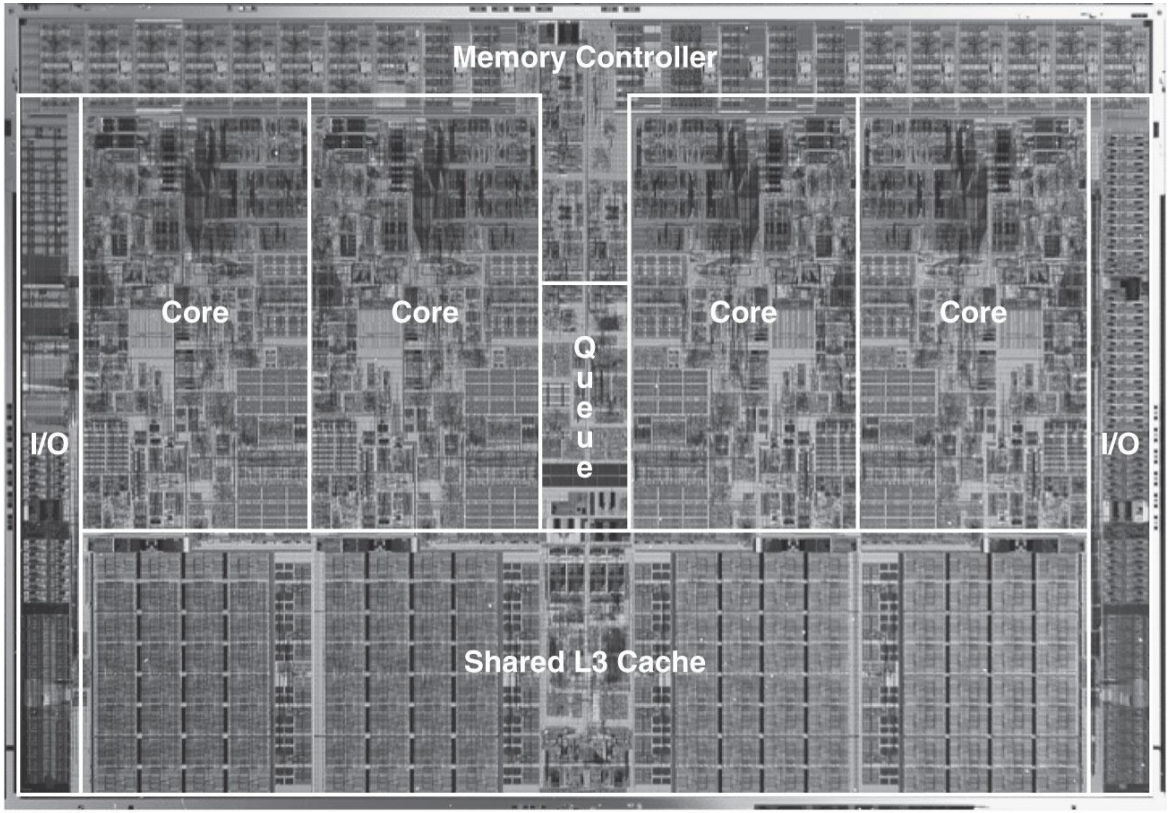
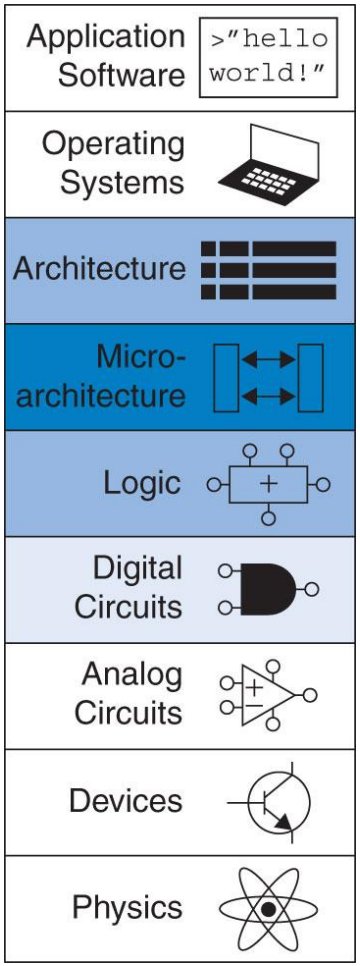


Figure 7.80 Core i7 microprocessor chip
(Source: http://www.intel.com/pressroom/archive/releases/2008/20081117comp_sm.htm. Courtesy Intel)



UNN Figure 1