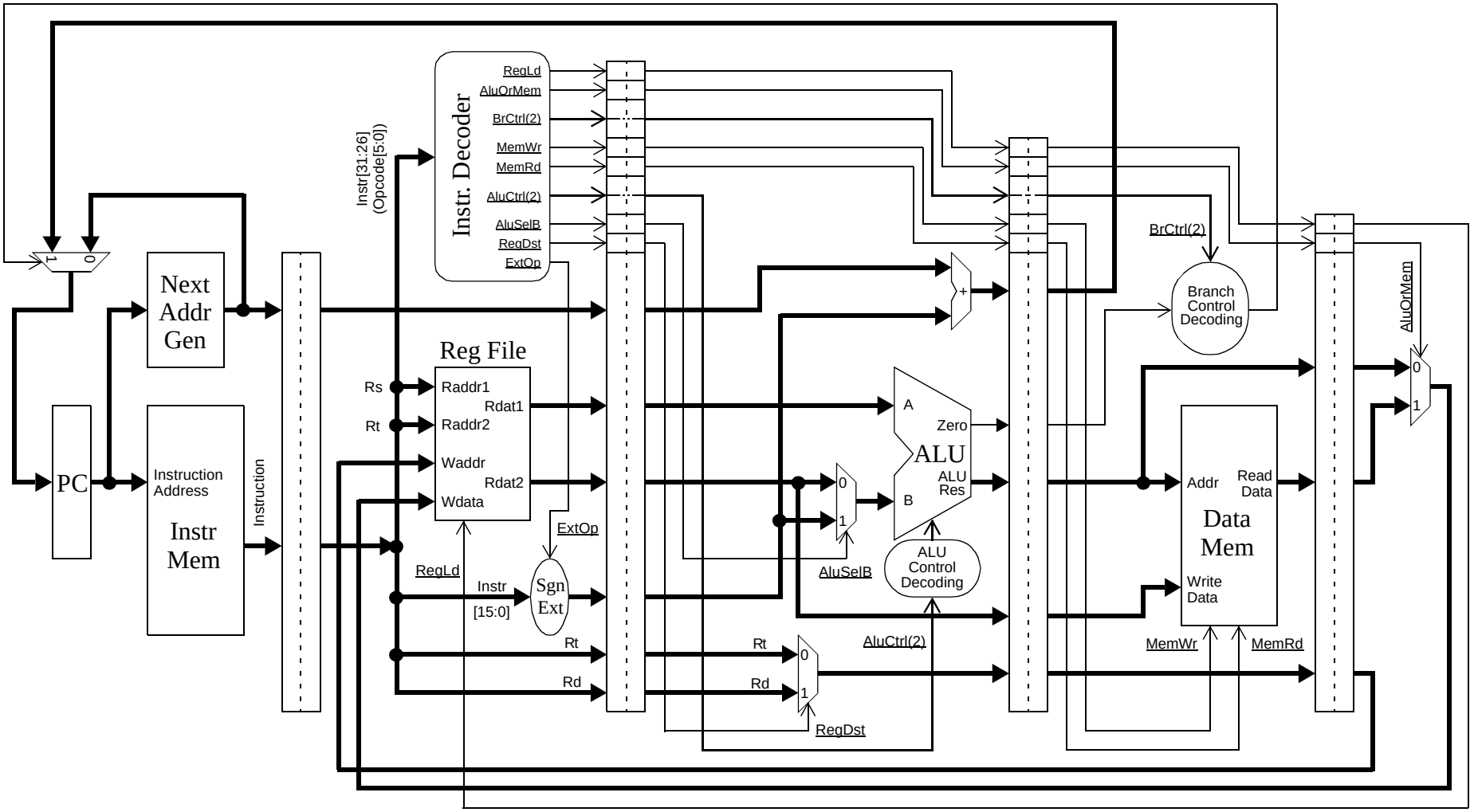
Pipelined MIPS Datapath with Control Signals



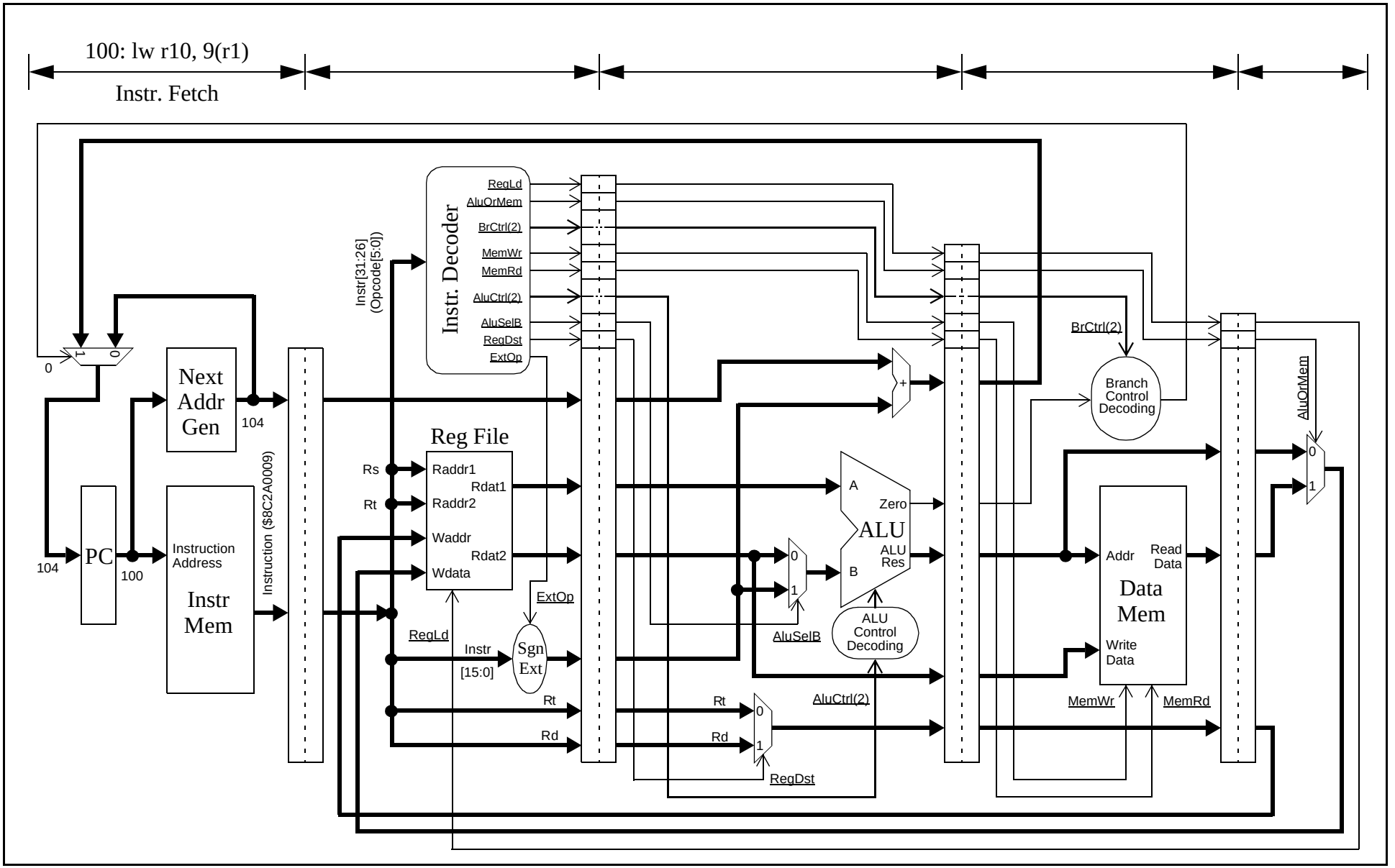
Five instruction sequence to be processed by pipeline:

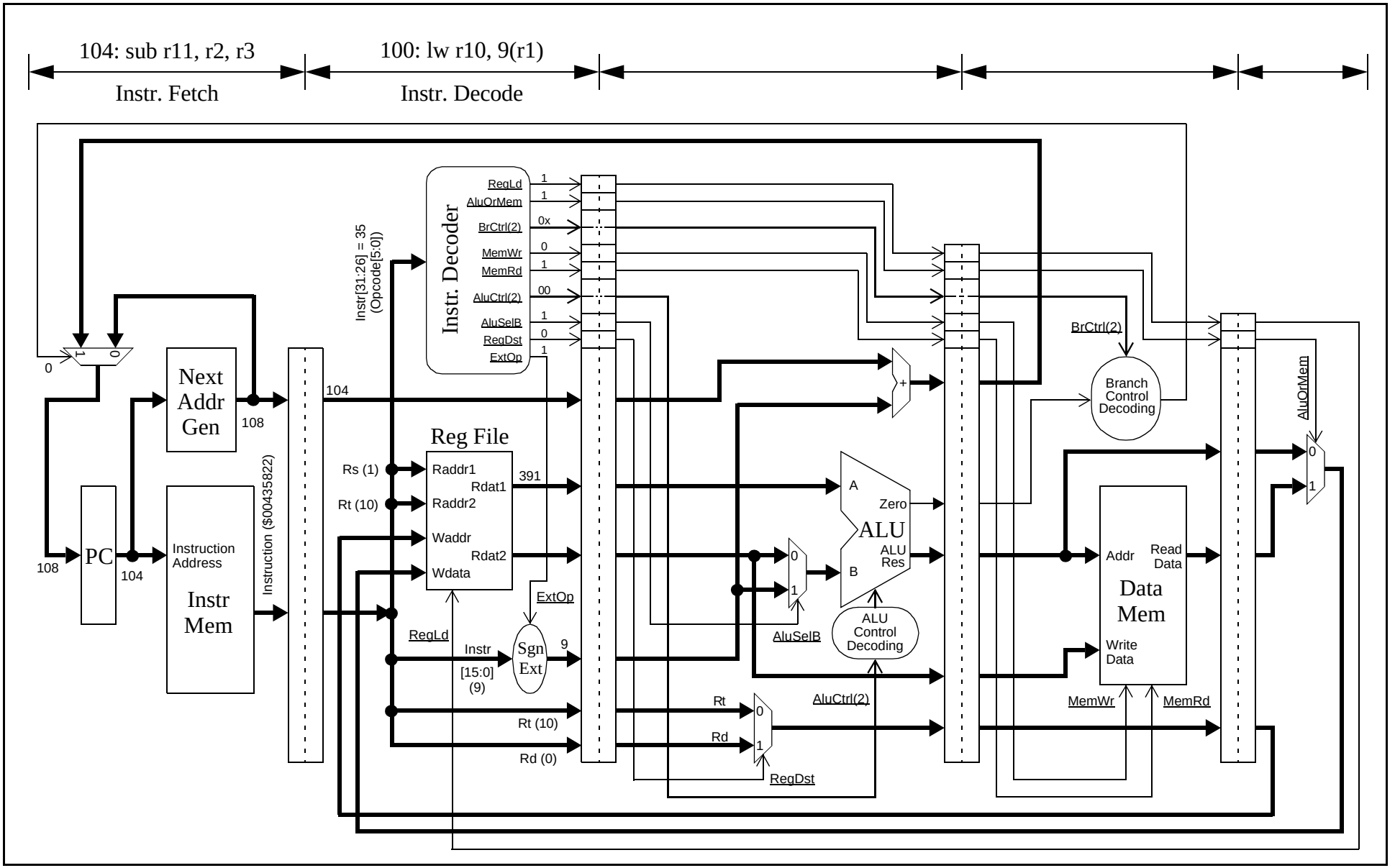
			op [31:26]	rs [25:21]	rt [20:16]	rd [15:10]	funct [5:0]	immed [15:0]
100:	lw	r10, 9(r1)	35	01	10	n/a	n/a	9
104:	sub	r11, r2, r3	00	02	03	11	34	n/a
108:	and	r12, r4, r5	00	04	05	12	36	n/a
112:	or	r3, r6, r7	00	06	07	03	37	n/a
116:	add	r14, r8, r9	00	08	09	14	32	n/a

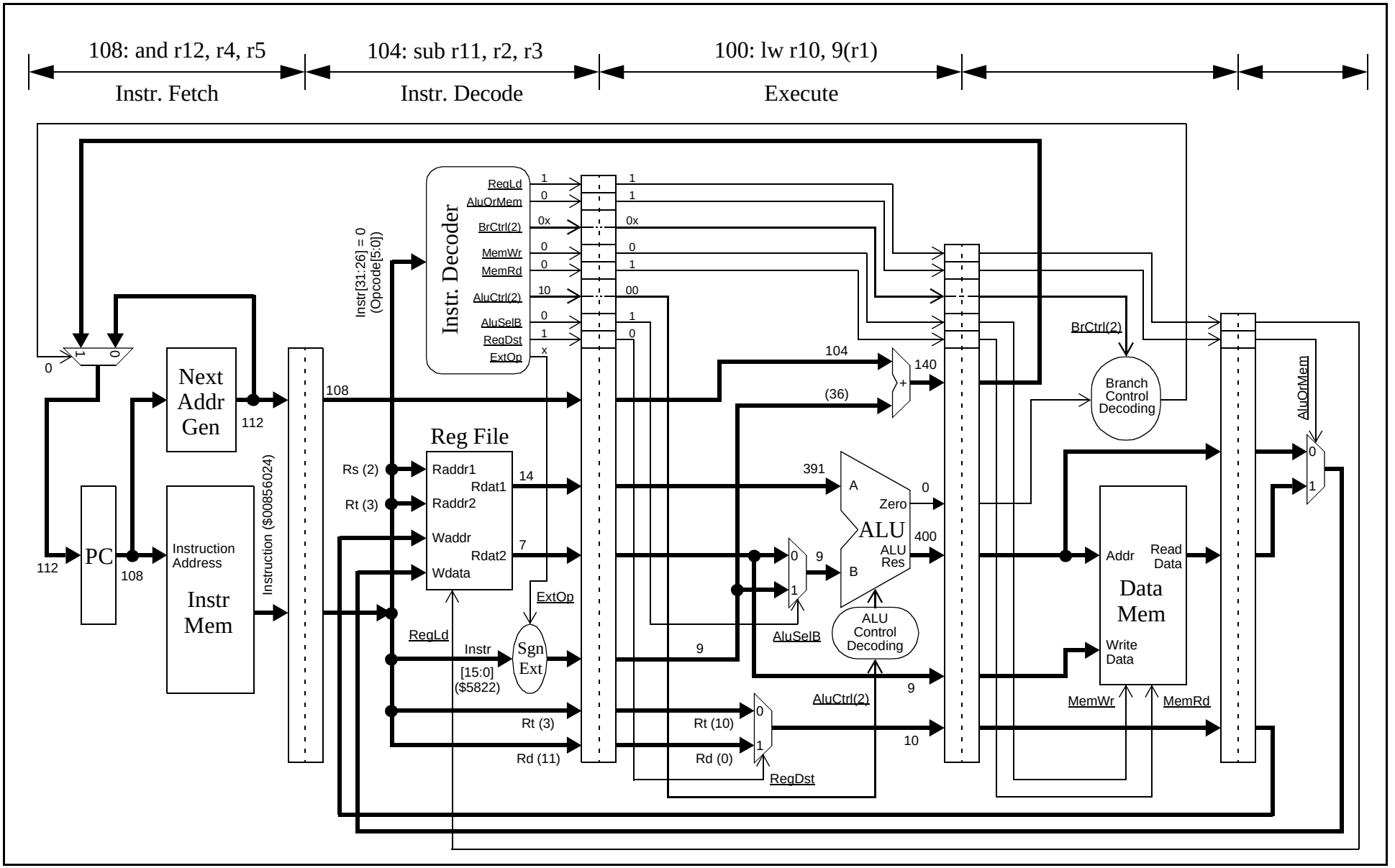
Control decoding for each of these instructions:

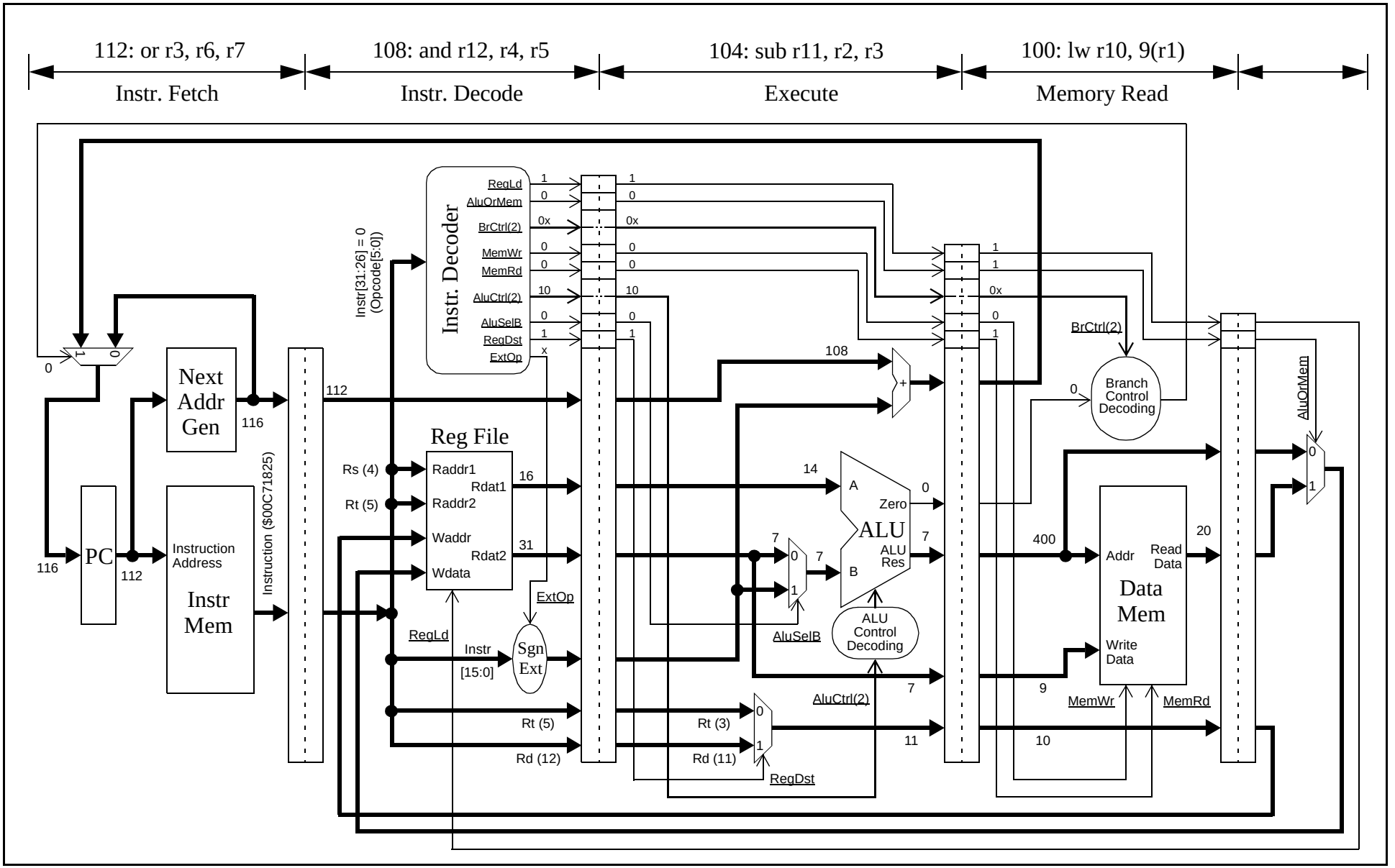
	ExtOp	RegDst	AluSelB	AluCtrl	MemRd	MemWr	BrCtrl	AluOrMem	RegLd
lw	1	0	1	00	1	0	0x	1	1
sub	x	1	0	10	0	0	0x	0	1
and	x	1	0	10	0	0	0x	0	1
or	x	1	0	10	0	0	0x	0	1
add	x	1	0	10	0	0	0x	0	1

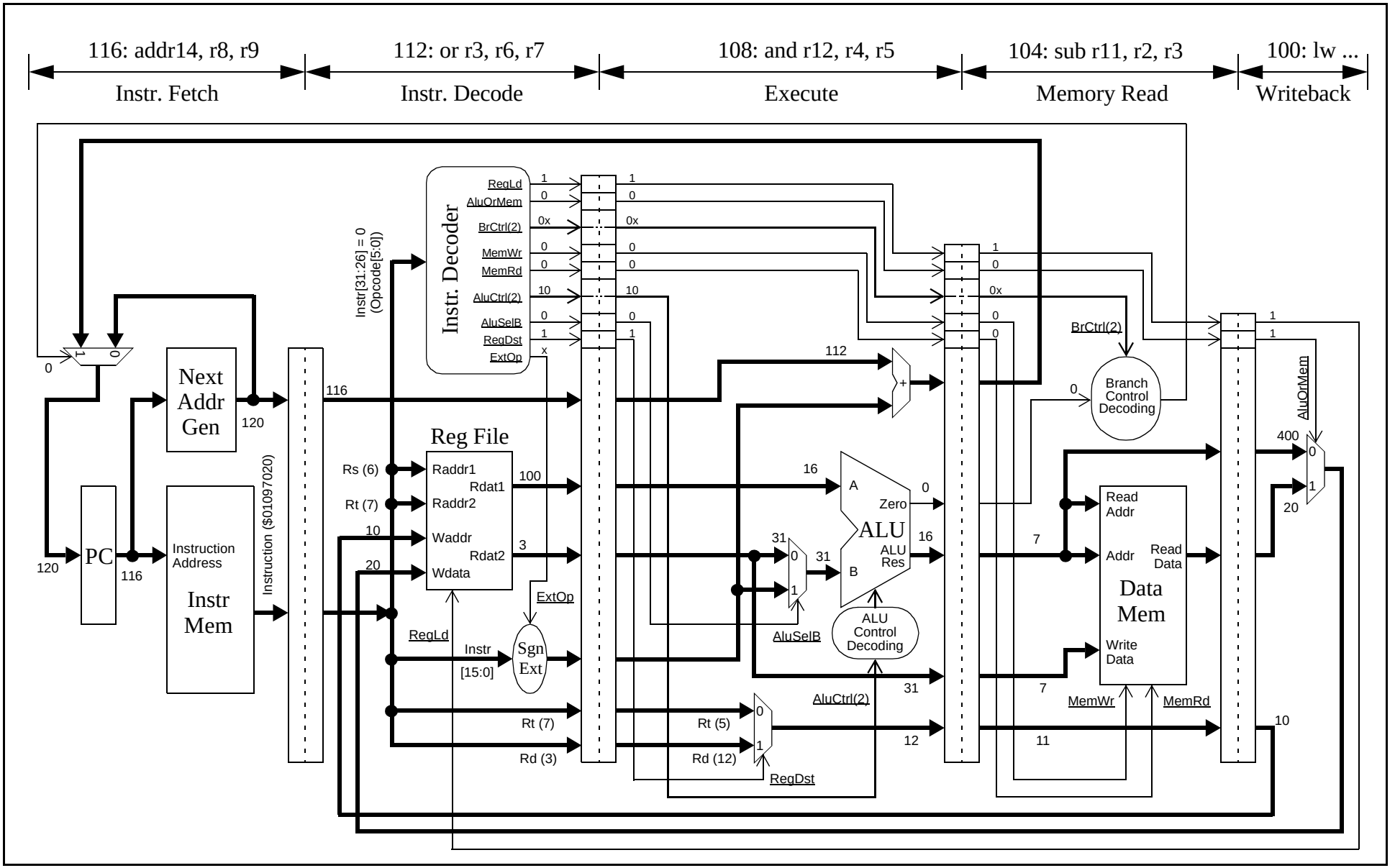
(N.B., ALU control for all R-type instructions is generated by the funct field, and AluCtrl in this case serves to allow local decoding of this field). Assume that, initially, r1 = 391, r2 = 14, r3 = 7, r4 = 16, r5 = 31, r6 =100, r7 = 3, r8 = 10, and r9 =20. Memory location 400 is assumed to contain the value 20.

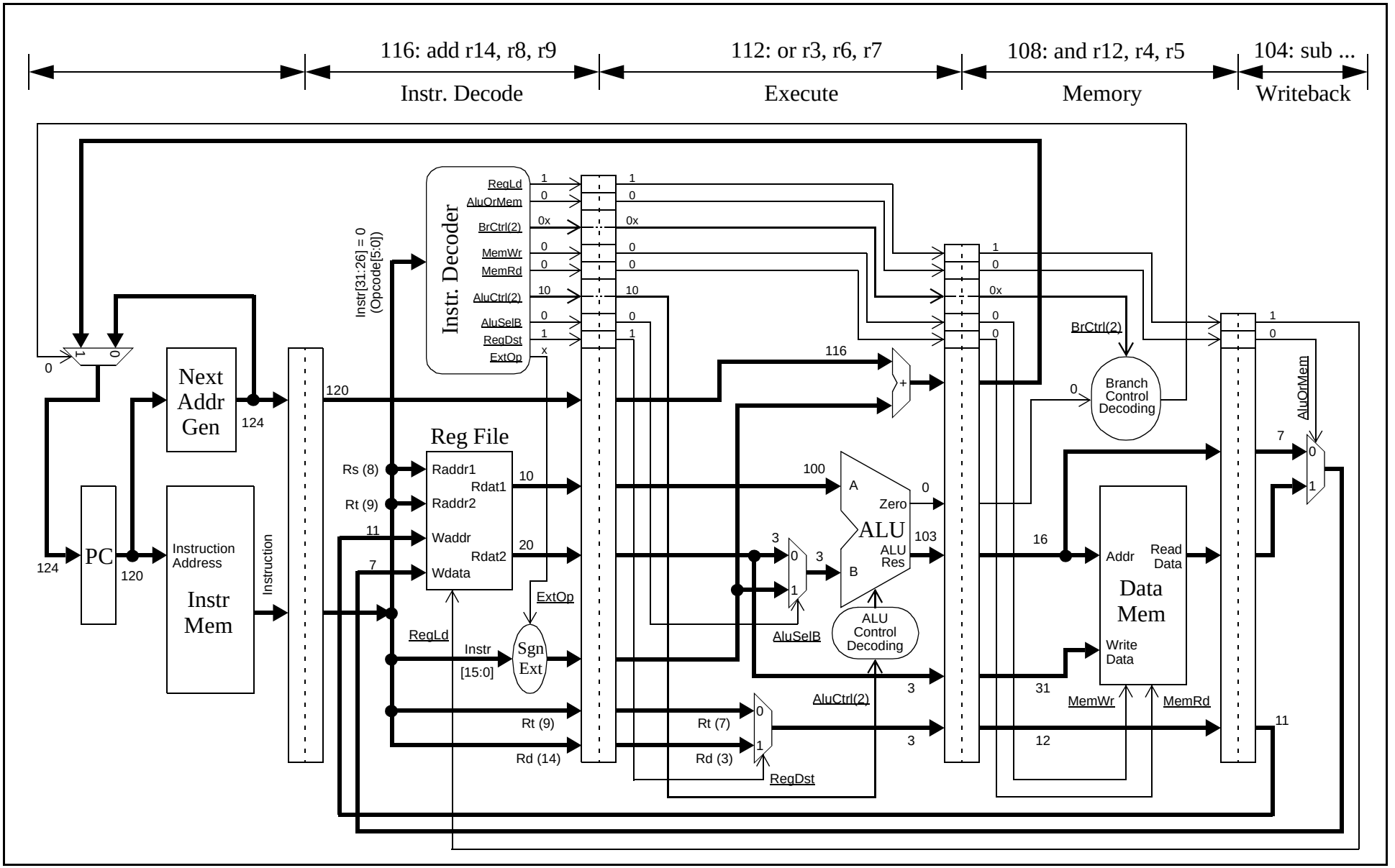

100: lw r10, 9(r1)
Instr. Fetch
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26]
(Opcode[5:0])
Next
Addr
Gen
104
0
1
0
PC
104
100
Instruction
Address
Instr
Mem
Instruction ($8C2A0009)
Reg File
Rs
Rt
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
RegLd
ExtOp
Instr
[15:0]
Sgn
Ext
Rt
Rd
0
1
RegDst
A
Zero
ALU
ALU
Res
B
AluSelB
ALU
Control
Decoding
AluCtrl(2)
+
BrCtrl(2)
Branch
Control
Decoding
Addr
Read
Data
Data
Mem
Write
Data
MemWr
MemRd
AluOrMem

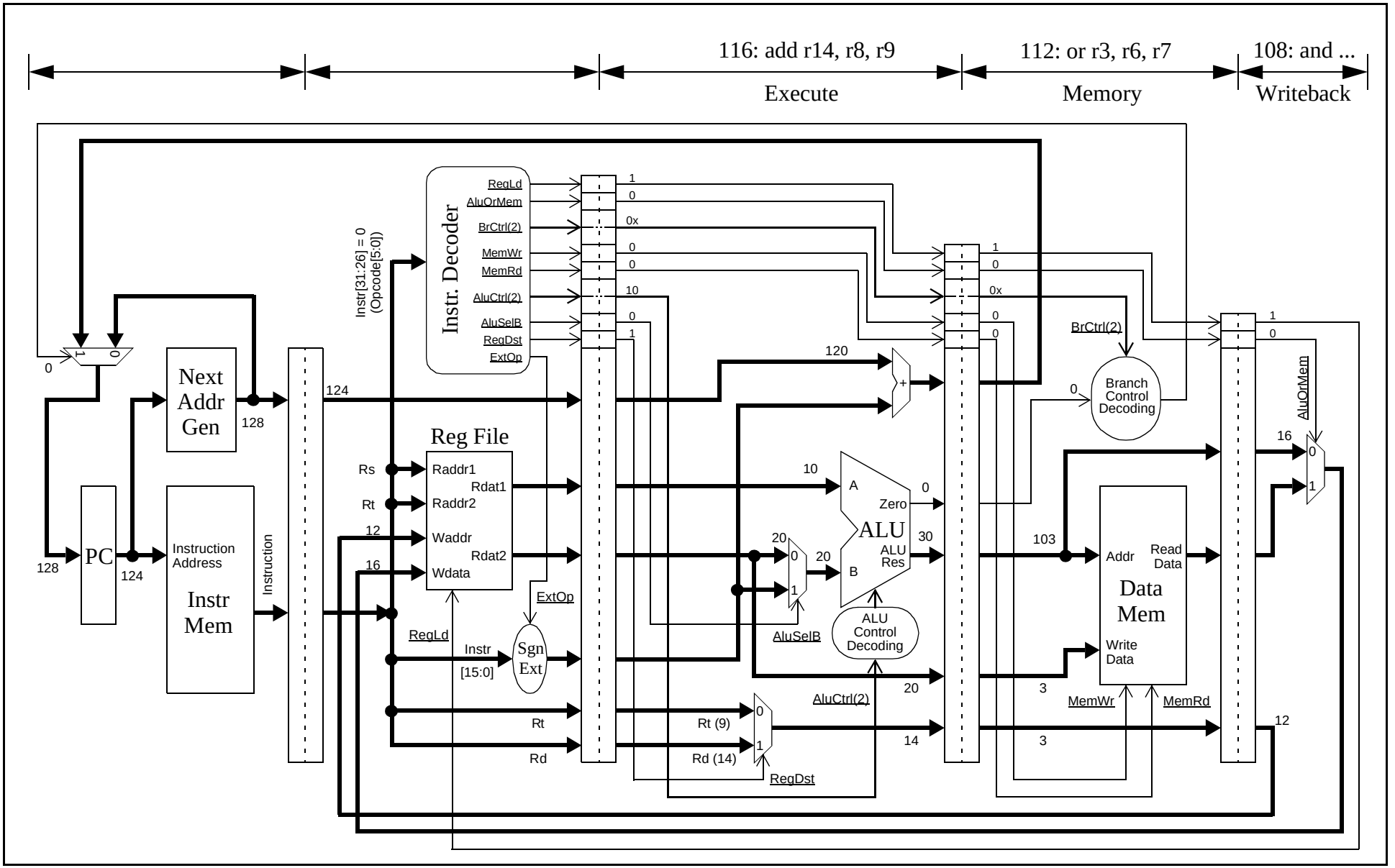

104: sub r11, r2, r3
100: lw r10, 9(r1)
Instr. Fetch
Instr. Decode
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 35
(Opcode[5:0])
Next Addr Gen
108
104
PC
Instruction Address
Instr Mem
Instruction ($00435822)
Reg File
Rs (1)
Rt (10)
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
391
RegLd
Instr [15:0] (9)
Sgn Ext
9
ExtOp
Rt
Rd
Rt (10)
Rd (0)
RegDst
AluSelB
A
B
ALU
Zero
ALU Res
ALU Control Decoding
AluCtrl(2)
BrCtrl(2)
Branch Control Decoding
AluOrMem
Addr
Read Data
Data Mem
Write Data
MemWr
MemRd

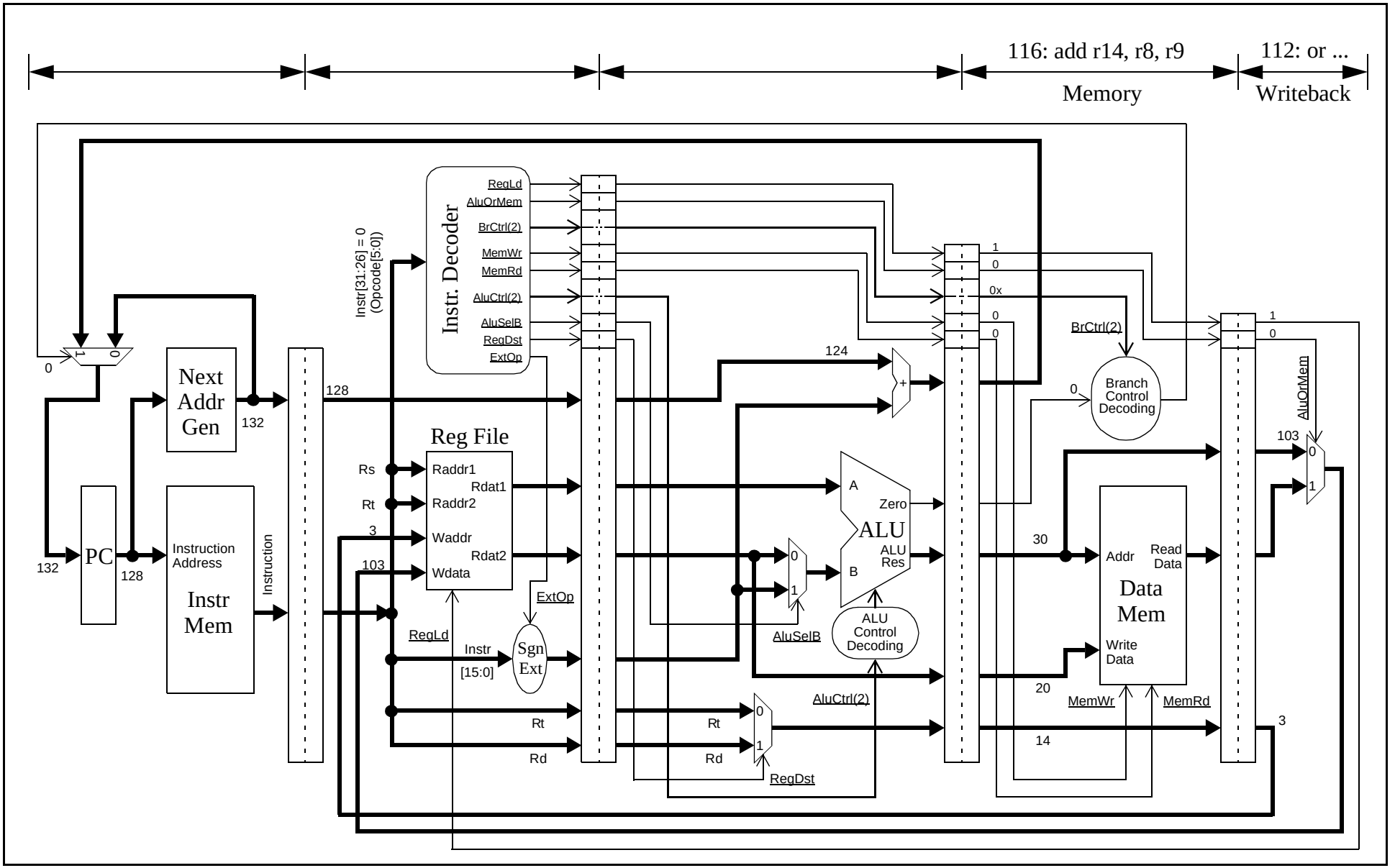

108: and r12, r4, r5
104: sub r11, r2, r3
100: lw r10, 9(r1)
Instr. Fetch
Instr. Decode
Execute
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
112
108
PC
Instruction Address
Instr Mem
Instruction ($00856024)
Reg File
Rs (2)
Rt (3)
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
14
7
ExtOp
RegLd
Instr [15:0] ($5822)
Sgn Ext
Rt (3)
Rd (11)
Rt (10)
Rd (0)
RegDst
104
(36)
140
391
A
B
ALU
Zero
ALU Res
0
400
9
AluSelB
ALU Control Decoding
AluCtrl(2)
10
BrCtrl(2)
Branch Control Decoding
Addr
Read Data
Data Mem
Write Data
MemWr
MemRd
AluOrMem

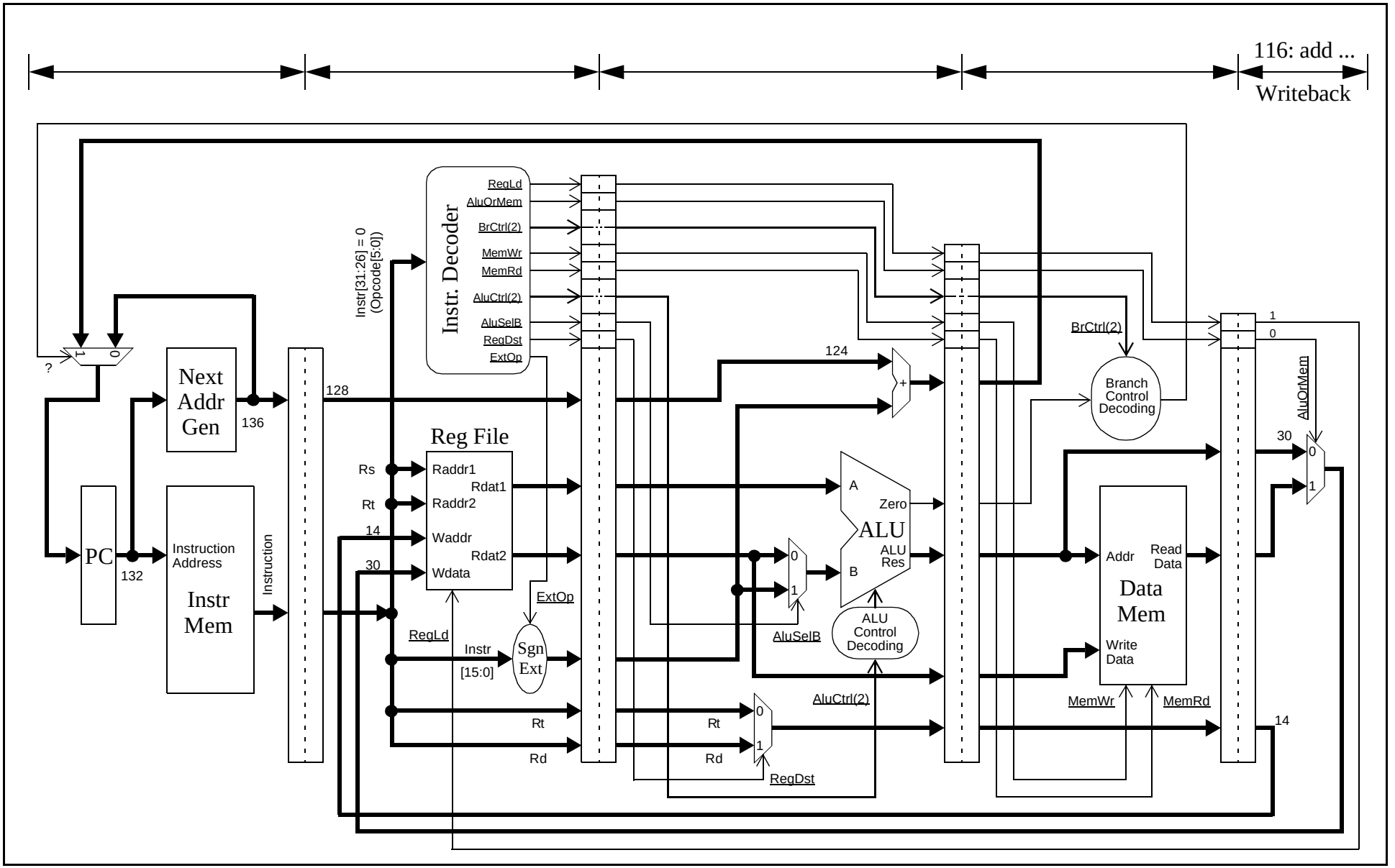

112: or r3, r6, r7
108: and r12, r4, r5
104: sub r11, r2, r3
100: lw r10, 9(r1)
Instr. Fetch
Instr. Decode
Execute
Memory Read
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
116
112
PC
Instruction Address
Instr Mem
Instruction ($00C71825)
Reg File
Rs (4)
Rt (5)
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
16
31
ExtOp
RegLd
Instr [15:0]
Sgn Ext
Rt (5)
Rd (12)
Rt (3)
Rd (11)
RegDst
108
14
A
B
ALU
Zero
ALU Res
ALU Control Decoding
AluSelB
AluCtrl(2)
Branch Control Decoding
BrCtrl(2)
400
Addr
Read Data
Data Mem
Write Data
MemWr
MemRd
20
9
10
11
AluOrMem


116: addr14, r8, r9
112: or r3, r6, r7
108: and r12, r4, r5
104: sub r11, r2, r3
100: lw ...
Instr. Fetch
Instr. Decode
Execute
Memory Read
Writeback
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
PC
Instruction Address
Instr Mem
Instruction ($01097020)
Reg File
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
Rs (6)
Rt (7)
Rd (3)
Rt (5)
Rd (12)
RegDst
Instr [15:0]
Sgn Ext
ALU
A
B
Zero
ALU Res
ALU Control Decoding
AluSelB
AluCtrl(2)
Branch Control Decoding
BrCtrl(2)
Data Mem
Read Addr
Addr
Read Data
Write Data
MemWr
MemRd
AluOrMem
120
116
112
100
400
20
10
7
11
12
16
31
3


116: add r14, r8, r9
112: or r3, r6, r7
108: and r12, r4, r5
104: sub ...
Instr. Decode
Execute
Memory
Writeback
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
PC
Instruction Address
Instr Mem
Instruction
Reg File
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
Rs (8)
Rt (9)
Rd (14)
Rt (7)
Rd (3)
RegDst
Instr [15:0]
Sgn Ext
ALU
A
B
Zero
ALU Res
ALU Control Decoding
AluSelB
AluCtrl(2)
Branch Control Decoding
BrCtrl(2)
Data Mem
Addr
Read Data
Write Data
MemWr
MemRd
AluOrMem


116: add r14, r8, r9
112: or r3, r6, r7
108: and ...
Execute
Memory
Writeback
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
PC
Instruction Address
Instr Mem
Instruction
Reg File
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
Rs
Rt
Rd
Sgn Ext
Instr [15:0]
ALU
A
B
Zero
ALU Res
ALU Control Decoding
Branch Control Decoding
Data Mem
Addr
Read Data
Write Data
MemWr
MemRd
AluSelB
AluCtrl(2)
RegDst
Rt (9)
Rd (14)


116: add r14, r8, r9
112: or ...
Memory
Writeback
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next Addr Gen
132
128
PC
Instruction Address
Instr Mem
Instruction
Reg File
Rs
Rt
3
103
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
RegLd
Instr [15:0]
Sgn Ext
ExtOp
Rt
Rd
RegDst
124
A
Zero
ALU
ALU Res
B
AluSelB
ALU Control Decoding
AluCtrl(2)
BrCtrl(2)
Branch Control Decoding
AluOrMem
30
Addr
Read Data
Data Mem
Write Data
MemWr
MemRd
20
14
3
1
0
0x


116: add ...
Writeback
Instr. Decoder
RegLd
AluOrMem
BrCtrl(2)
MemWr
MemRd
AluCtrl(2)
AluSelB
RegDst
ExtOp
Instr[31:26] = 0
(Opcode[5:0])
Next
Addr
Gen
136
128
PC
132
Instruction
Address
Instr
Mem
Instruction
Reg File
Rs
Rt
14
30
Raddr1
Rdat1
Raddr2
Waddr
Rdat2
Wdata
RegLd
Instr
[15:0]
Sgn
Ext
ExtOp
Rt
Rd
124
A
Zero
ALU
ALU
Res
B
AluSelB
ALU
Control
Decoding
AluCtrl(2)
RegDst
BrCtrl(2)
Branch
Control
Decoding
Addr
Read
Data
Data
Mem
Write
Data
MemWr
MemRd
AluOrMem
30
14
?

Important points about the pipeline operation:

1 Although the pipeline allows one instruction to complete in every clock cycle (when operating at maximum efficiency), each individual instruction still takes five clock cycles to complete.

2 It takes four clock cycles before the pipeline is operating a maximum efficiency (startup time).

3 Register writeback data and controls (RegLd, Waddr and Wdata) are provided by the WB stage, not the ID stage. This can be a pitfall because the pipeline diagram shows the register file as being part of the ID stage. Don’t forget the feedback.

4 Register reads from instruction *i* happen in parallel with register writes from instruction *i* - 3.

5 When a stage is inactive, any lines which control memory or register accesses are deasserted (in active high logic, set to 0).

6 Notice that the sequencing of the pipeline is controlled by the instruction currently in the MEM stage of execution, i.e., the decision about whether to sequence or branch is made by the instruction which entered the pipeline three cycles ago.