Author:

James F. Plusquellic

Project Description:

XXX

Opcodes:
5 bit opcode; 8 general purpose registers (3 bits each to specify Rd, Rs1, Rs2); Immediates are restricted in length to the number of bit positions available in a 16 bit instruction word. ALL instructions are 16 bits in length. Shift immediates are restricted to 2 bits.

All PC-Relative addresses are sign extended. All addresses are word aligned. PC is incremented by 2 after each instruction.

Table 1: OPCODES

Name	Opcode	Format	Notes
SEQ (SetIfEq)	00000	SEQ Rd Rs1 Rs2	Set register Rd to 1 if the values in Rs1 and Rs2 are equal, else set to 0.
ADD	00001	ADD Rd Rs1 Rs2	Add registers Rs1 and Rs2 and store the result in Rd.
SGT (SetIfGtr)	00010	SGT Rd Rs1 Rs2	Set register Rd to 1 if Rs1 > Rs2, else set to 0.
ADDI	00011	ADDI Rd Rs #	Store in register Rd the value Rs + the 5-bit **sign-extended** immediate.
SUB	00100	SUB Rd Rs1 Rs2	Compute Rs1 - Rs2 and store the result in Rd.
SLT (SetIfLes)	00101	SLT Rd Rs1 Rs2	Set register Rd to 1 if Rs1 < Rs2, else set to 0.
Unused	00110		
OR	00111	OR Rd Rs1 Rs2	Store in register Rd the bitwise OR of Rs1 and Rs2.
ORI	01000	ORI Rd Rs #	Store in register Rd the bitwise OR of Rs and the 5-bit **zero-extended** immediate.
AND	01001	AND Rd Rs1 Rs2	Store in register Rd the bitwise AND of Rs1 and Rs2.
ANDI	01010	ANDI Rd Rs #	Store in register Rd the bitwise AND of Rs and the 5-bit **zero-extended** immediate.

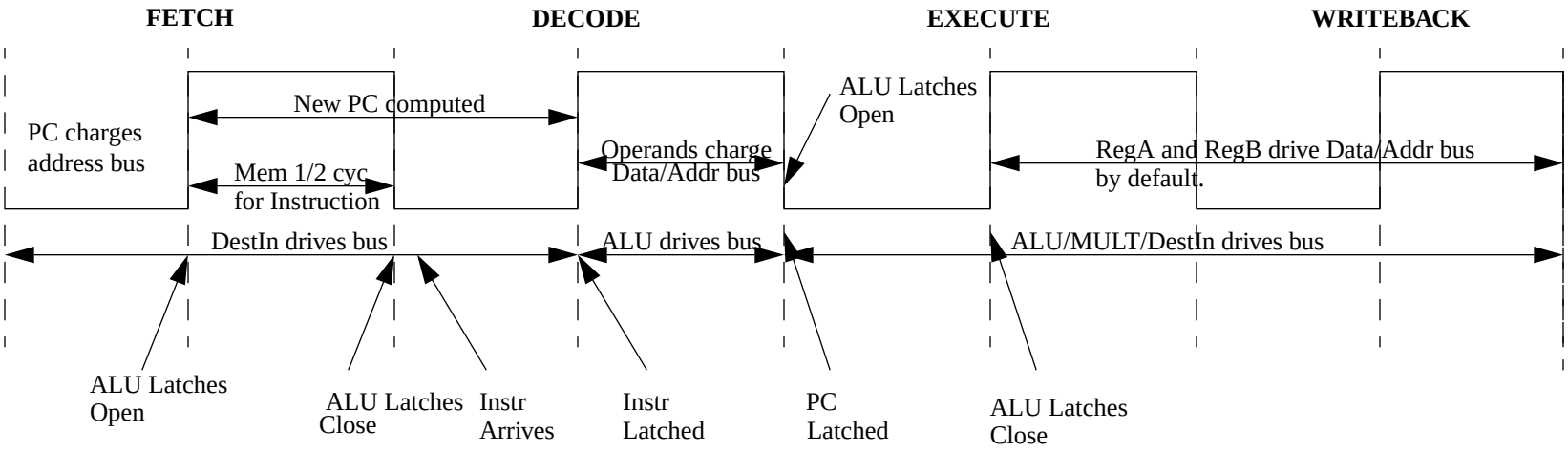
Table 1: OPCODES

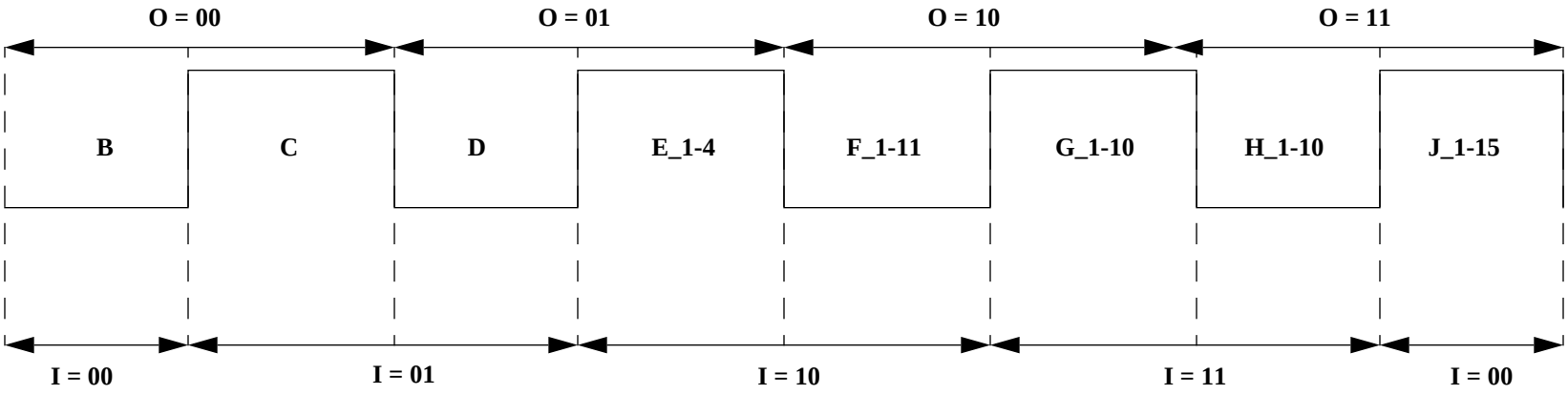
Name	Opcode	Format	Notes
XOR	01011	XOR Rd Rs1 Rs2	Store in register Rd the bitwise XOR of Rs1 and Rs2.
XNOR	01100	XNOR Rd Rs1 Rs2	Store in register Rd the bitwise XNOR of Rs1 and Rs2.
NOT	01101	NOT Rd Rs	Store in register Rd the bitwise complement Rs.
SRA (ShftRgtArith)	01110	SRA Rd Rs #	Store in register Rd the **sign-extended** value of Rs shifted to the right by the 2-bit immediate. (An immediate of 0 does not shift the operand).
SRL (ShftRghtLogic)	01111	SRL Rd Rs #	Store in register Rd the **zero-extended** value of Rs shifted to the right by the 2-bit immediate.
SLL (ShftLeftLogic)	10000	RLL Rd Rs #	Store in register Rd the **zero-extended** value of Rs shifted to the left by the 2-bit immediate.
SW (StoreWord)	10001	SW 0 Rs Raddr	Store to memory at address Raddr the value in Rs. (You can assume that address is an even number and the compiler inserts 3 zero bits for Rd).
MUL (Multiply)	10010	MUL Rd Rs1 Rs2	Multiply the lower 8 bits of Rs1 and Rs2 and store the result in Rd.
Unused	10011		
LW (LoadWord)	10100	LR Rd 0 Raddr	Load the word value at memory address Raddr into Rd. (You can assume that address is an even number and the compiler inserts 3 zero bits for Rs).
LBI (LoadBytImmed)	10101	LBI Rd #	Store the 8-bit **sign-extended** immediate into register Rd after sign extending it.
LBIU (LoadBytIUnsign)	10110	LBIU Rd #	Write the 8-bit **zero-extended** immediate into register Rd.

Table 1: OPCODES

Name	Opcode	Format	Notes
LHI (LoadHighImmed)	10111	LHI Rd #	Write the 8-bit immediate into the upper 8 bits of register Rd and optionally clear the low order 8 bits.
BEQZ (BrIfEqZero)	11000	BEQZ Rs #	Set PC to PC + 8-bit **sign-extended** immediate if the value in register Rs is zero.
BNEZ (BrIfNotZero)	11001	BNEZ Rs #	Set PC to PC + 8-bit **sign-extended** immediate if the value in register Rs is non-zero.
BC (BrIfCarrySet)	11010	BC #	Set PC to PC + 11-bit **sign-extended** immediate if the carry out is set.
BO (BrIfOvrflowSet)	11011	BO #	Set PC to PC + 11-bit **sign-extended** immediate if the overflowis set.
NOP (NoOperation)	11100	NOP	Do nothing.
J (Jump)	11101	J #	Set the PC to PC + 11-bit **sign-extended** immediate.
JR (JumpToRegister)	11110	JR 0 Rs	Set the PC to the value in register Rs.
JALR (JumpAndLink)	11111	JALR Rd #	Rd=PC, PC=PC+#. Save the current value of PC (which points to the NEXT instruction) to register Rd and set PC to PC + 8-bit **sign-extended** immediate.

NOTES: All PC-relative addresses are sign-extended to allow both forward and backward branches.
'Unused' opcodes may be optionally encoded for other instructions, e.g. LLI (LoadLowImmed preserving the high order bits of destination register).
You can assume all addresses (whether for instruction or data) are word aligned.
You must increment PC by 2 by the end of the 2nd clock cycle so it is available for PC-relative branch additions such as BEQZ.


FETCH
DECODE
EXECUTE
WRITEBACK
PC charges address bus
New PC computed
Mem 1/2 cyc for Instruction
Operands charge Data/Addr bus
ALU Latches Open
RegA and RegB drive Data/Addr bus by default.
DestIn drives bus
ALU drives bus
ALU/MULT/DestIn drives bus
ALU Latches Open
ALU Latches Close
Instr Arrives
Instr Latched
PC Latched
ALU Latches Close


O = 00
O = 01
O = 10
O = 11
B
C
D
E_1-4
F_1-11
G_1-10
H_1-10
J_1-15
I = 00
I = 01
I = 10
I = 11
I = 00

Input Signals:

- OC_4
- OC_3
- OC_2
- OC_1
- OC_0
- O_0
- O_1
- ALU_overflow
- ALU_carry
- REG_zero
- RESET
- Clk

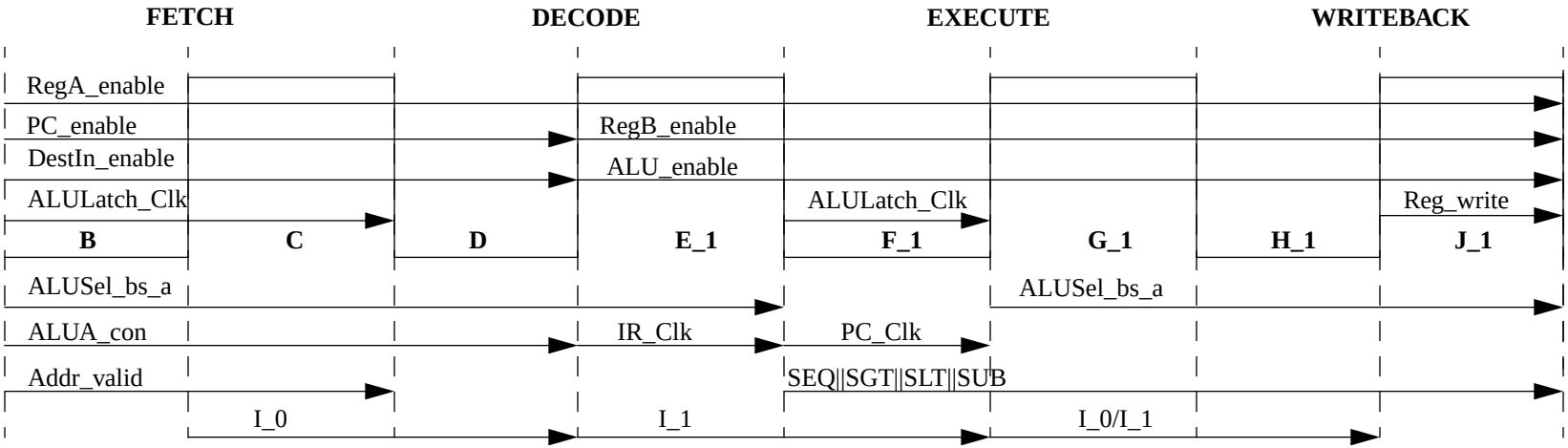
Output Signals:

- PC_Clk (0: master follows D)
- IR_Clk (0: master follows D)
- ALULatch_Clk (0: off; 1: transparent)
- ALUOC_Clk (0: off; 1: transparent)
- Addr_valid (Output PAD: 1 for valid addresses)
- Read_write (Output PAD: 0 for read; 1 for write)
- Immed_enable (0: high imp; 1 drive)
- RegA_enable (0: high imp; 1 drive)
- PC_enable (0: high imp; 1 drive)
- RegB_enable (0: high imp; 1 drive)
- Mult_enable (0: high imp; 1 drive)
- DestIn_enable (0: high imp; 1 drive)
- ALU_enable (0: high imp; 1 drive)
- ALUSel_bs (0: BFU; 1: Shifter)
- ALUSel_bs_a (0: BFU&Shifter; 1: Adder&Constant)
- ALUShift_la (0: logic; 1: arithmetic)
- ALUShift_lr (0: left; 1: right)
- ALUShift_0/1 (00: no shift; 01,10, 11)
- ALUA_con (0: Databus; 1: Constant)
- ALUBFU_0/1/2/3 (_0 is input 00 in truth table, etc)
- Rs1_is_Rs2 (0: no; 1: yes - Rs1 OC bits select Rs2 reg)
- Rd_is_Rs1 (0: no; 1: yes - Rd OC bits select Rs1 reg)
- Reg_write (0: all zeros; 1: select reg file write decoder)
- I_0/1 (00, 01, 10, 11: Selects next state)
- ALUShift_force (0; use OC bits 0 and 1; 1: force 0)

Individual Instruction Decoding:

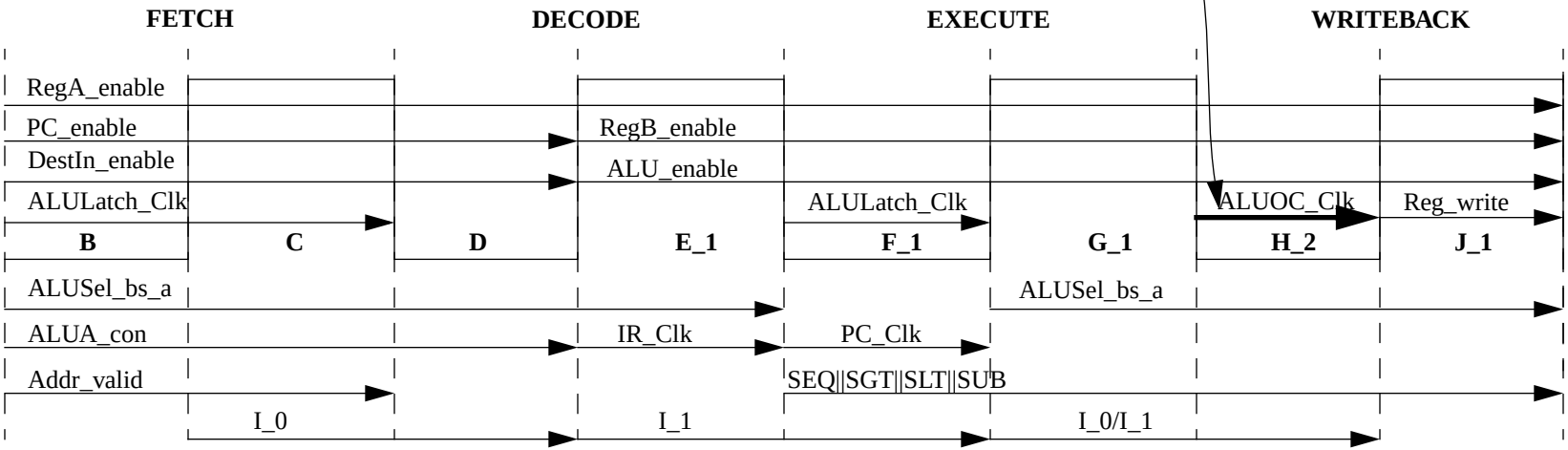
- JALR
- LBI
- BEQZ
- BNEQ
- LBIU
- ORI
- ANDI
- NOT
- ADDI
- J
- BO
- BC
- LHI
- SEQ
- SGT
- SLT
- SUB

SEQ, SGT, SLT

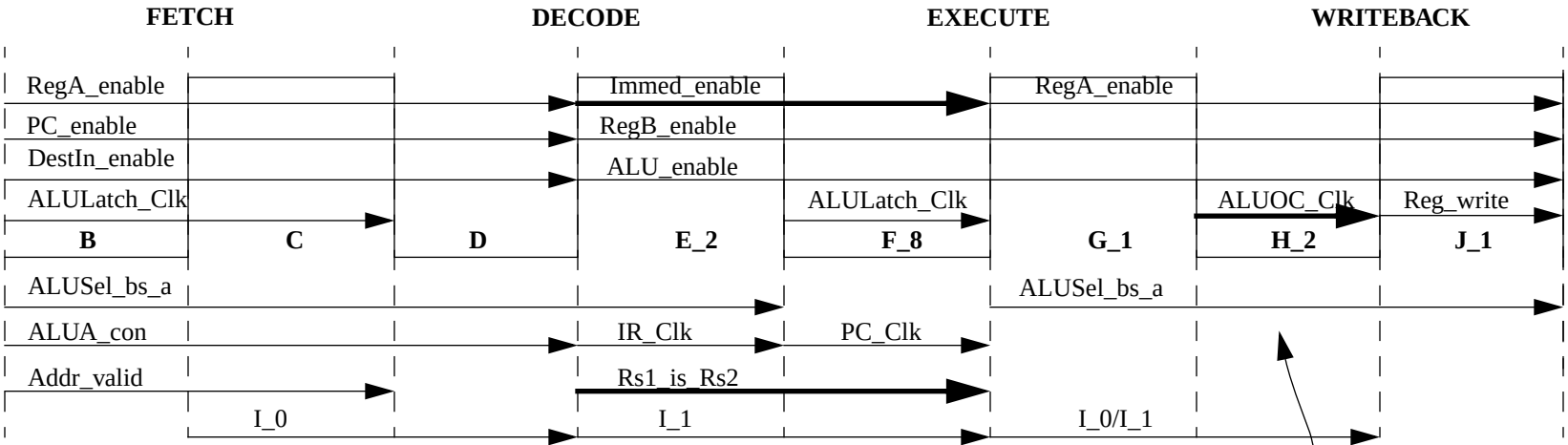


Only difference

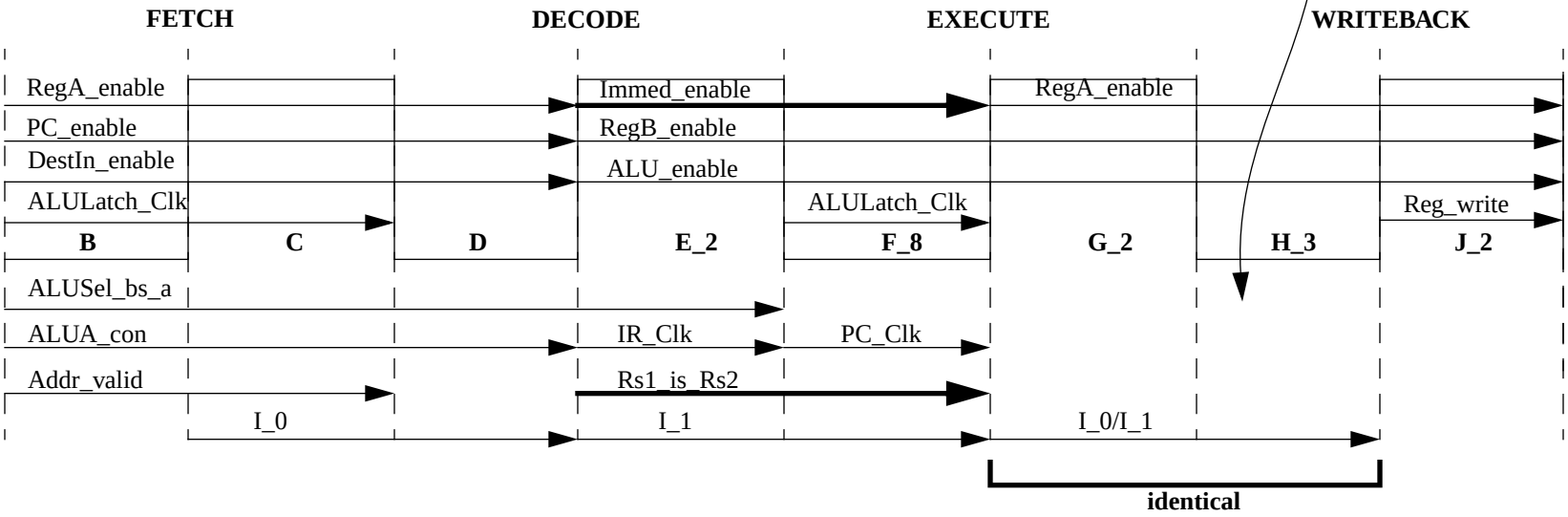
ADD, SUB



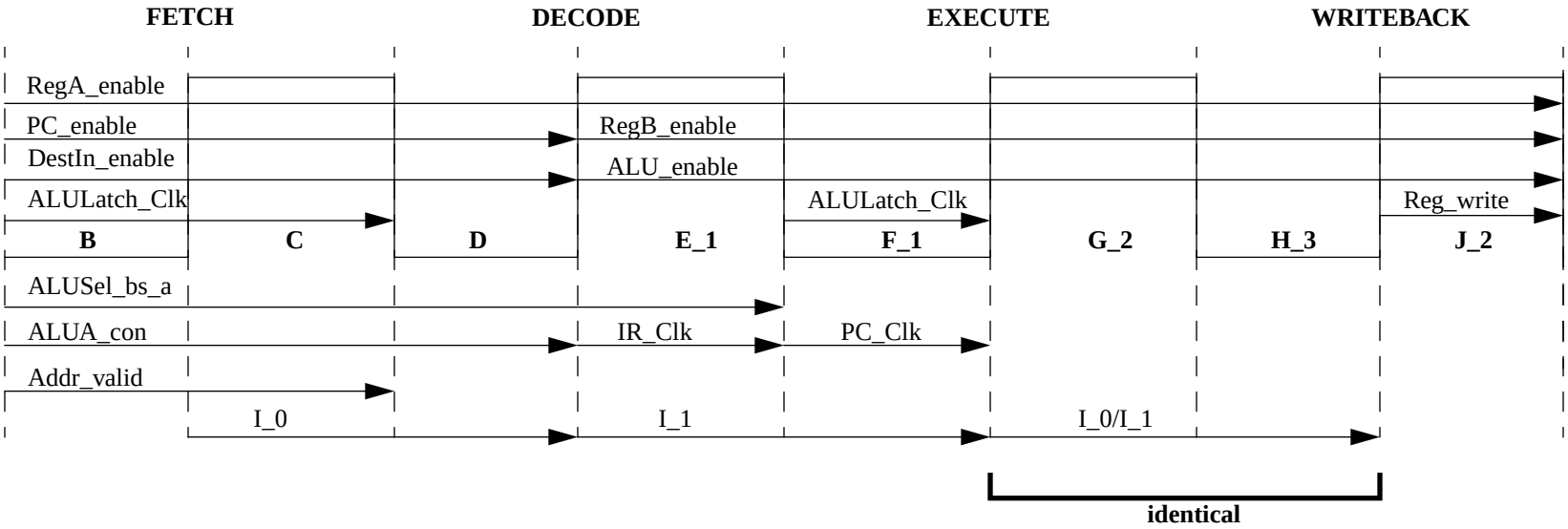
ADDI



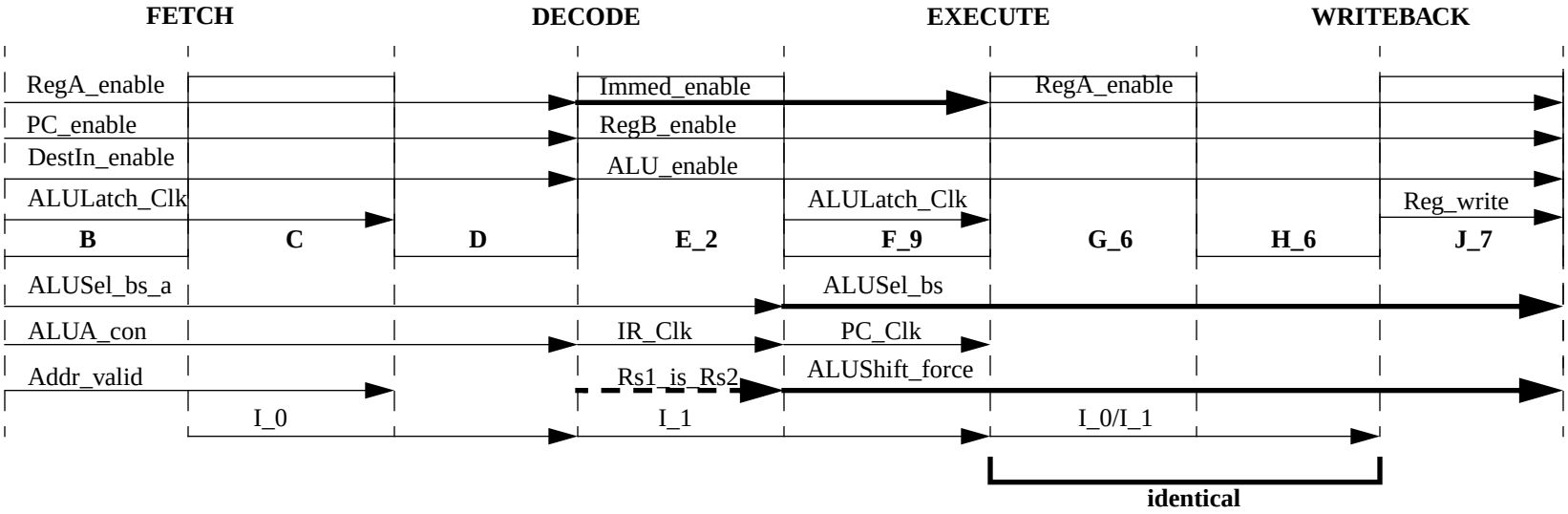
ORI, ANDI, NOT



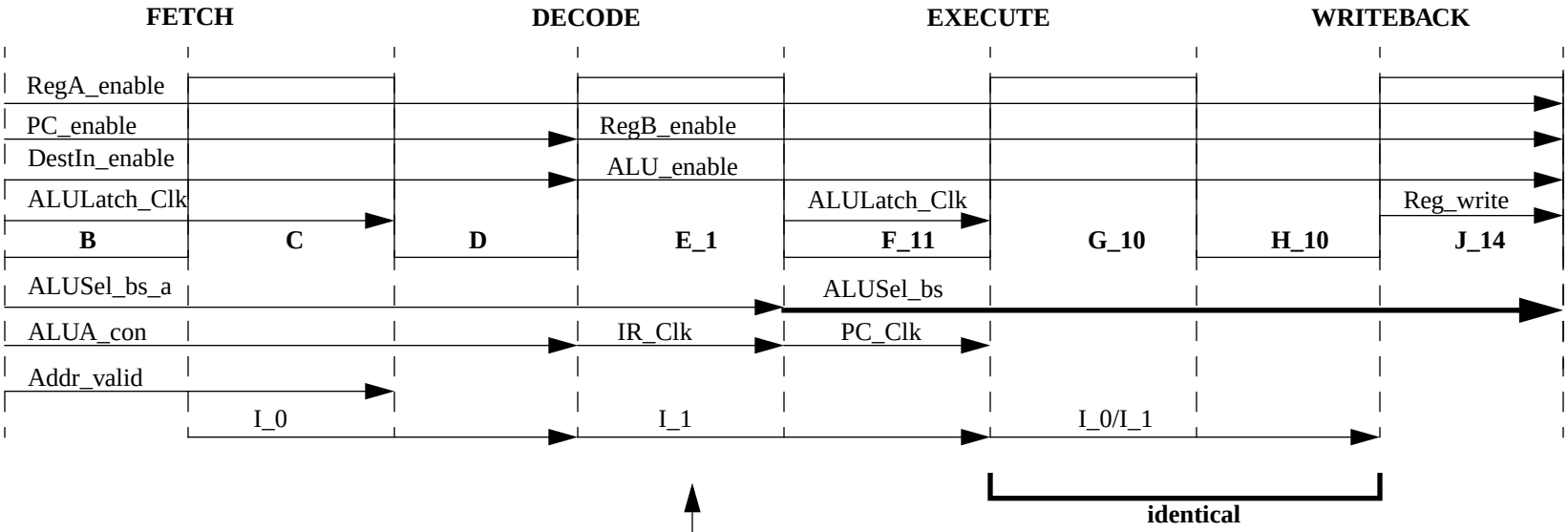
OR, AND, XOR, XNOR



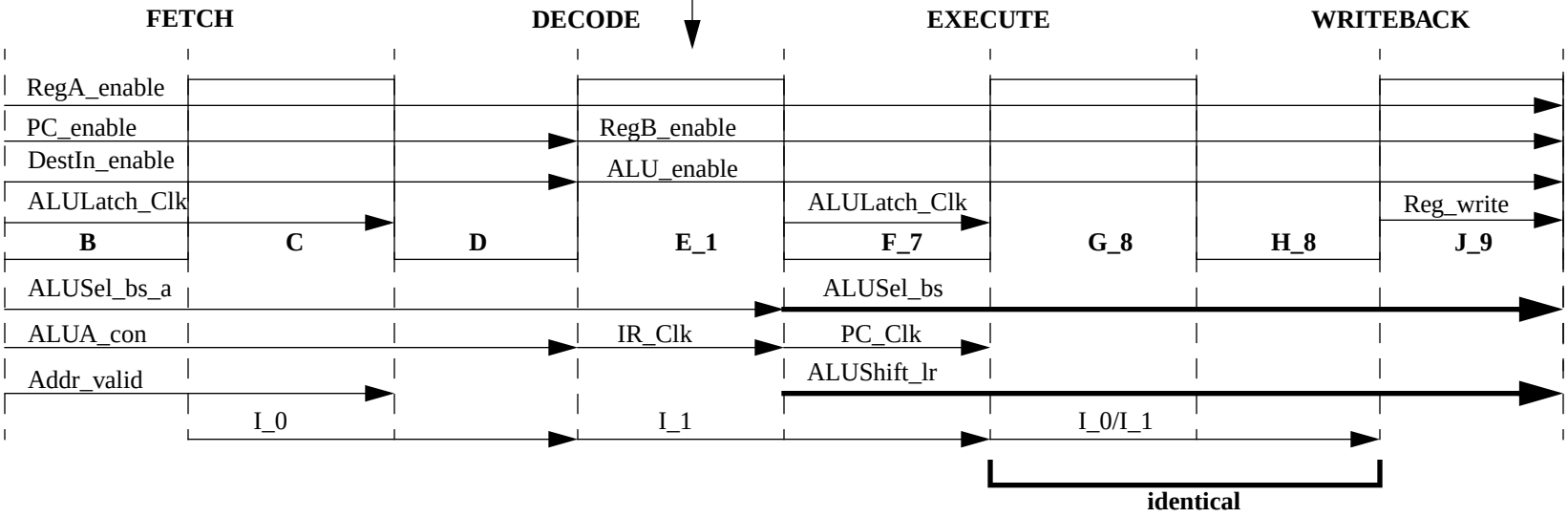
LBI, LBIU, LHI



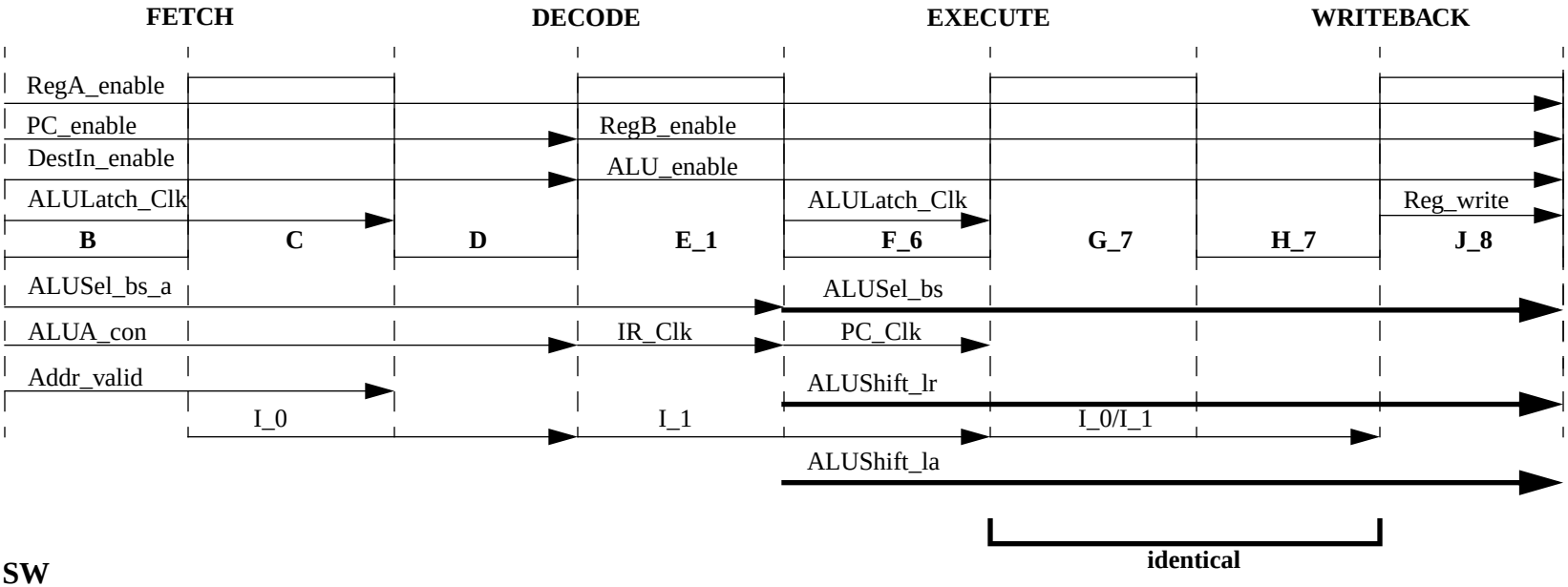
SLL


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
ALULatch_Clk
Reg_write
B
C
D
E_1
F_11
G_10
H_10
J_14
ALUSel_bs_a
ALUSel_bs
ALUA_con
IR_Clk
PC_Clk
Addr_valid
I_0
I_1
I_0/I_1
identical

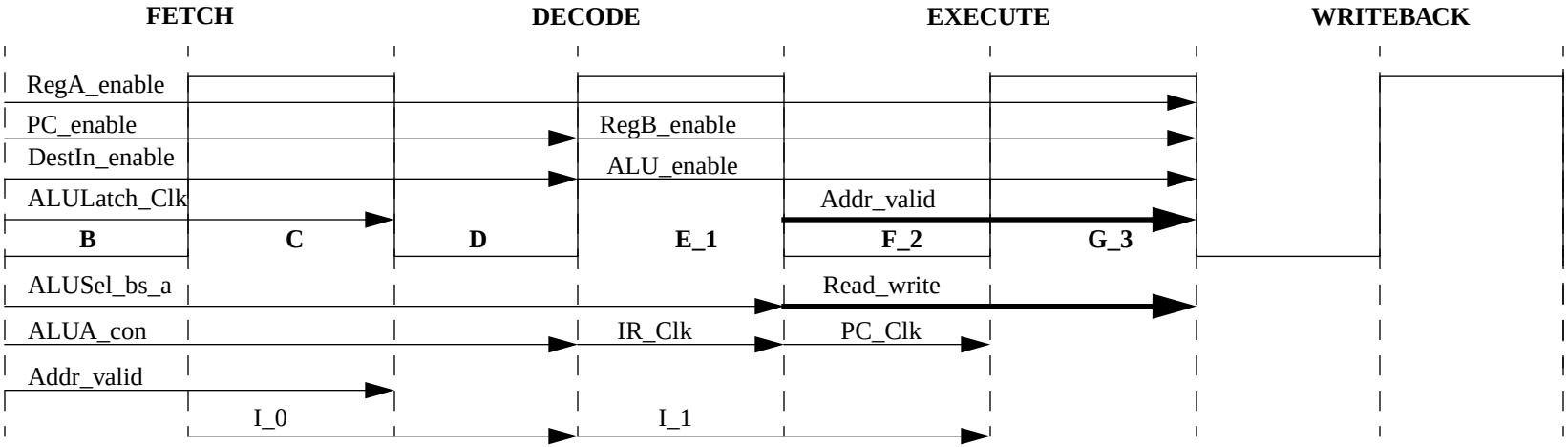
SRL


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
ALULatch_Clk
Reg_write
B
C
D
E_1
F_7
G_8
H_8
J_9
ALUSel_bs_a
ALUSel_bs
ALUA_con
IR_Clk
PC_Clk
Addr_valid
ALUShift_lr
I_0
I_1
I_0/I_1
identical

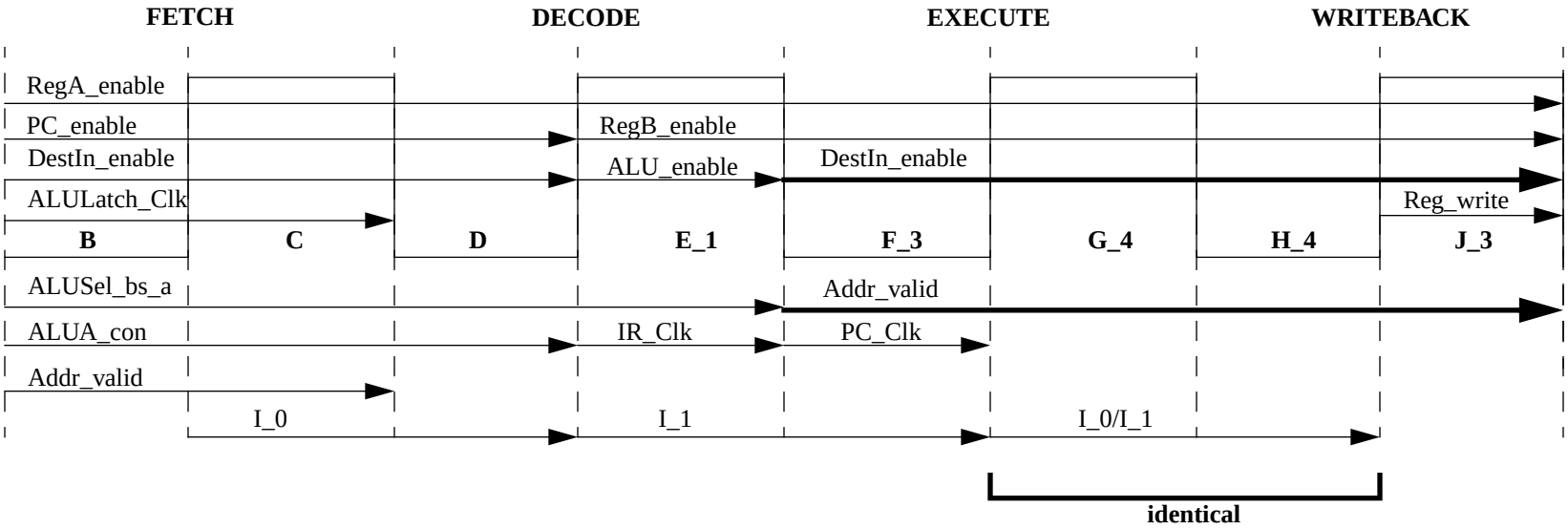
SRA


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
ALULatch_Clk
Reg_write
B
C
D
E_1
F_6
G_7
H_7
J_8
ALUSel_bs_a
ALUSel_bs
ALUA_con
IR_Clk
PC_Clk
Addr_valid
ALUShift_lr
I_0
I_1
I_0/I_1
ALUShift_la
identical

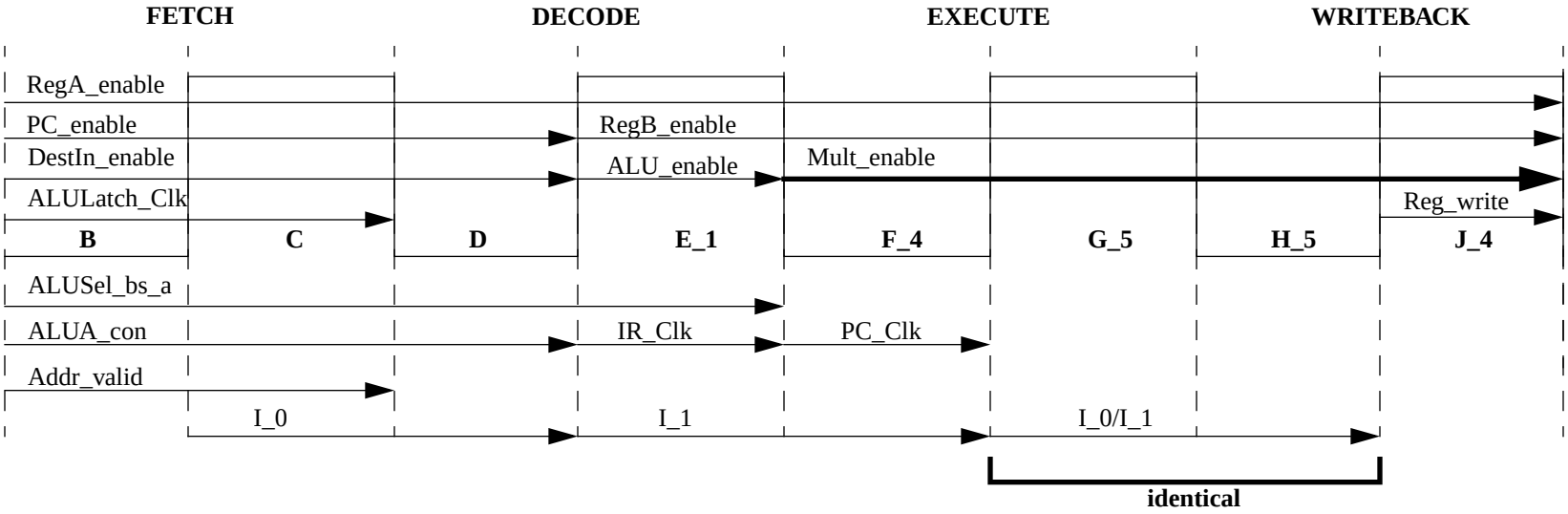
SW


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
Addr_valid
B
C
D
E_1
F_2
G_3
ALUSel_bs_a
Read_write
ALUA_con
IR_Clk
PC_Clk
Addr_valid
I_0
I_1

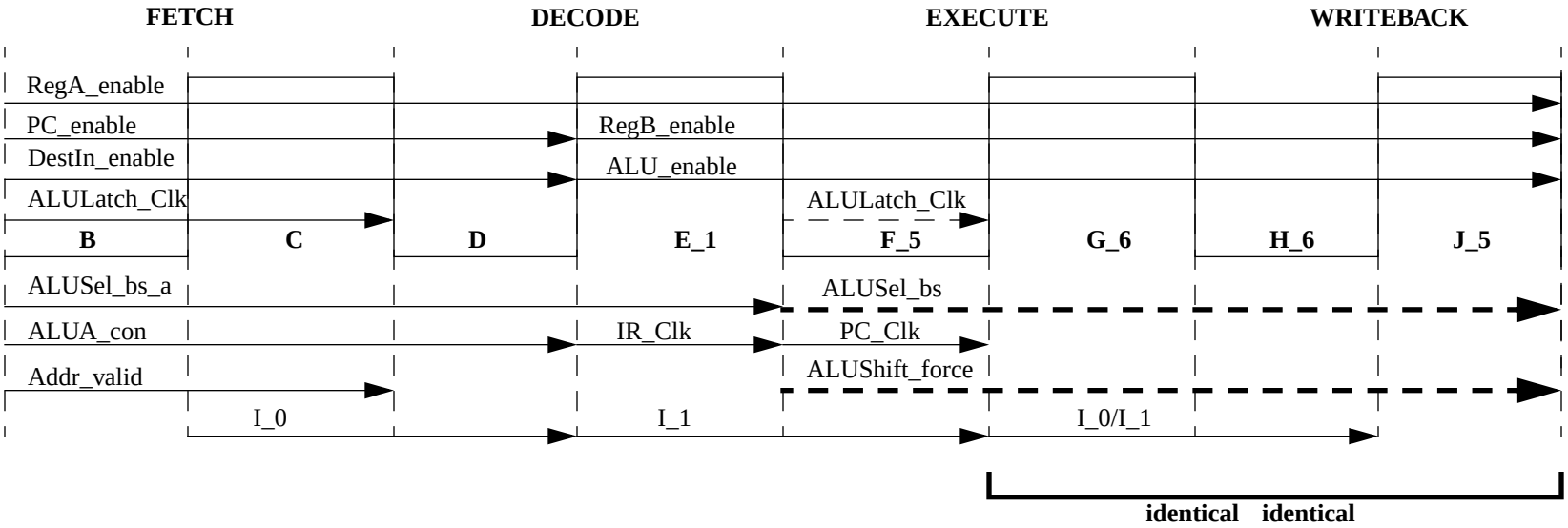
LW


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
RegB_enable
ALU_enable
DestIn_enable
Reg_write
B
C
D
E_1
F_3
G_4
H_4
J_3
ALUSel_bs_a
Addr_valid
ALUA_con
IR_Clk
PC_Clk
Addr_valid
I_0
I_1
I_0/I_1
identical

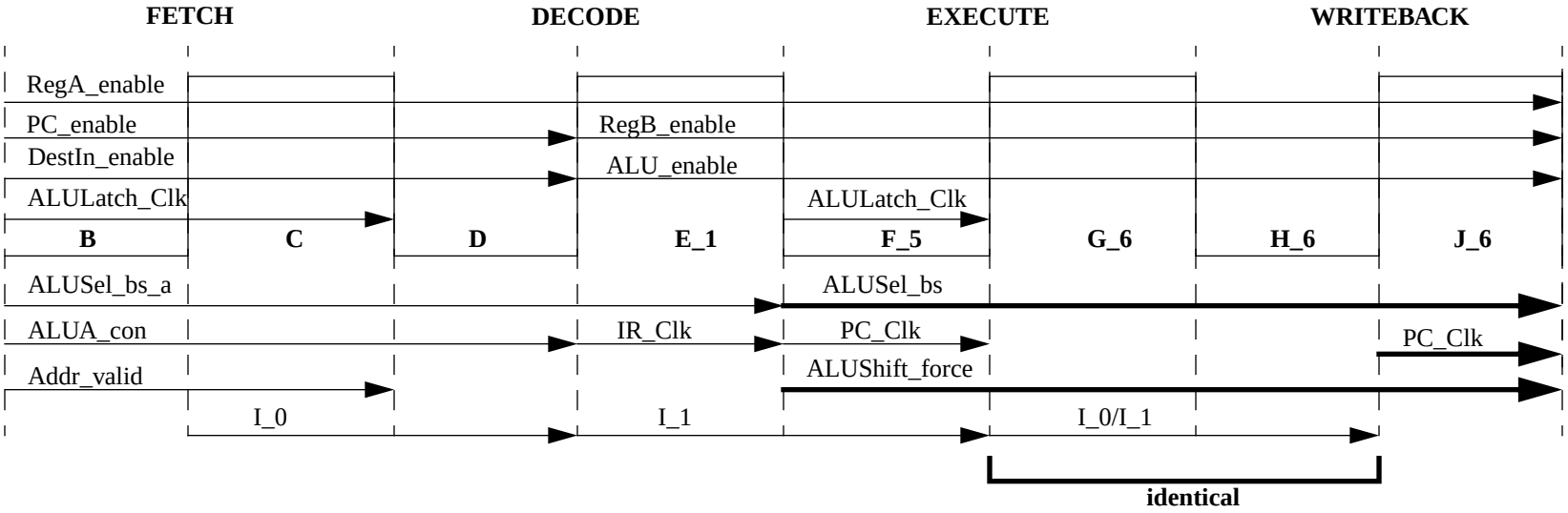
MULT


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
RegB_enable
ALU_enable
Mult_enable
Reg_write
B
C
D
E_1
F_4
G_5
H_5
J_4
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
Addr_valid
I_0
I_1
I_0/I_1
identical

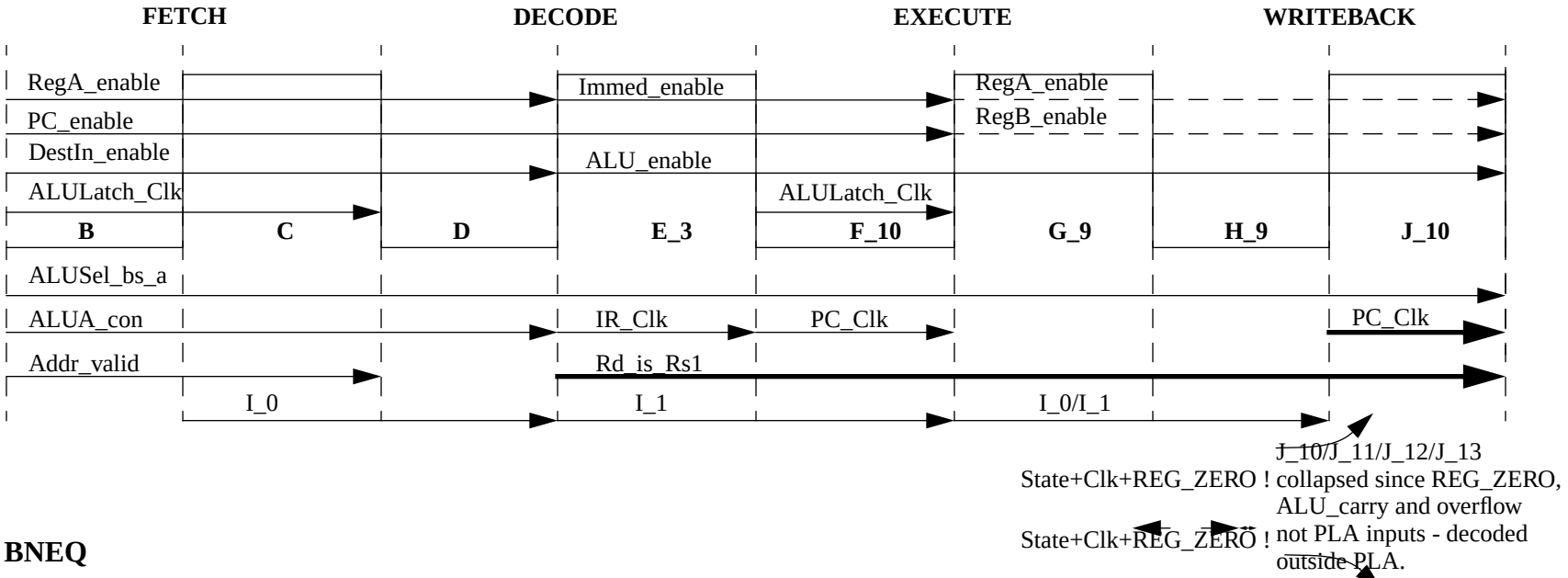
NOP


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
ALULatch_Clk
B
C
D
E_1
F_5
G_6
H_6
J_5
ALUSel_bs_a
ALUSel_bs
ALUA_con
IR_Clk
PC_Clk
Addr_valid
ALUShift_force
I_0
I_1
I_0/I_1
identical
identical

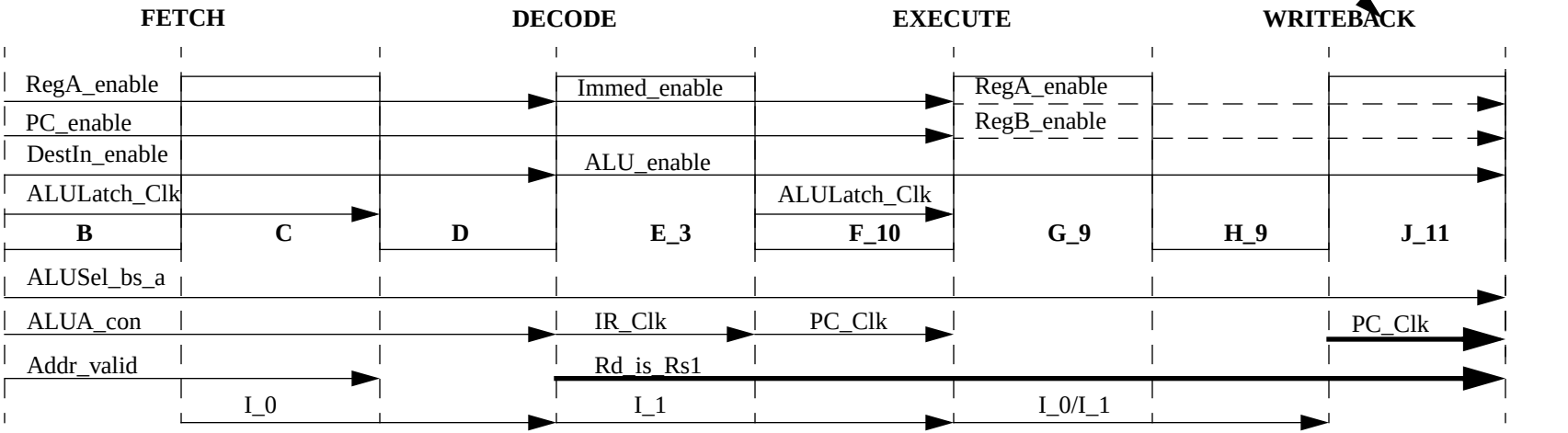
JR


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
RegB_enable
DestIn_enable
ALU_enable
ALULatch_Clk
ALULatch_Clk
B
C
D
E_1
F_5
G_6
H_6
J_6
ALUSel_bs_a
ALUSel_bs
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
ALUShift_force
I_0
I_1
I_0/I_1
identical

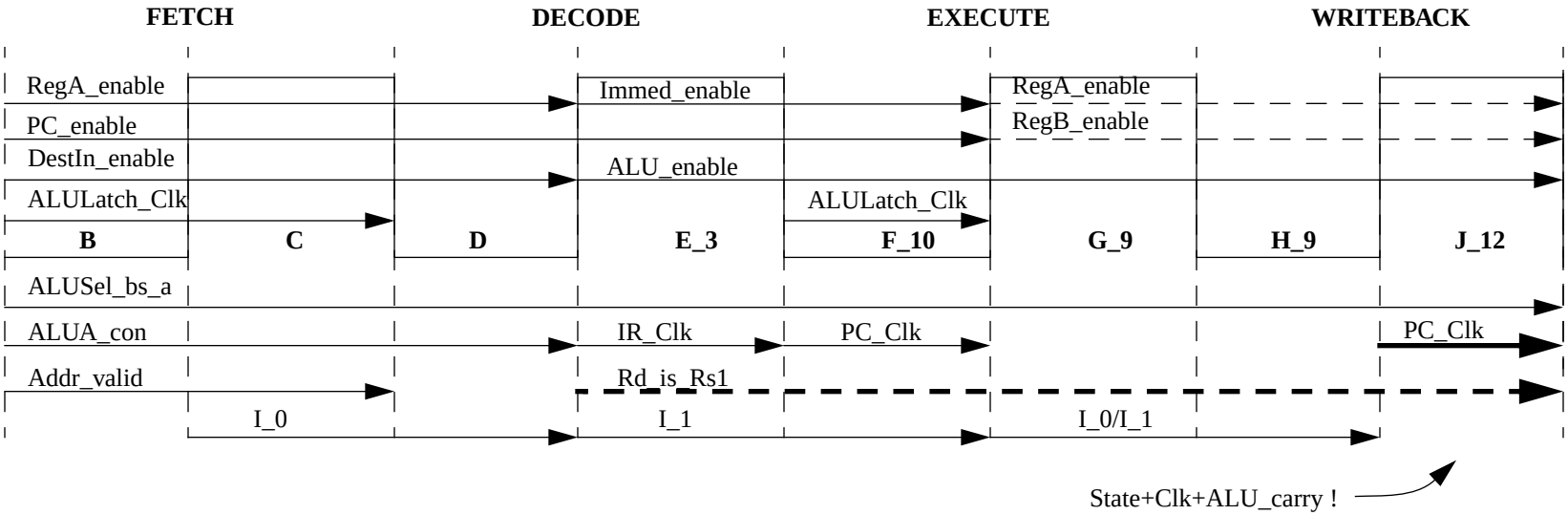
BEQZ


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
ALULatch_Clk
RegA_enable
RegB_enable
B
C
D
E_3
F_10
G_9
H_9
J_10
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1
J_10/J_11/J_12/J_13
State+Clk+REG_ZERO !
collapsed since REG_ZERO,
ALU_carry and overflow
not PLA inputs - decoded
outside PLA.
State+Clk+REG_ZERO !

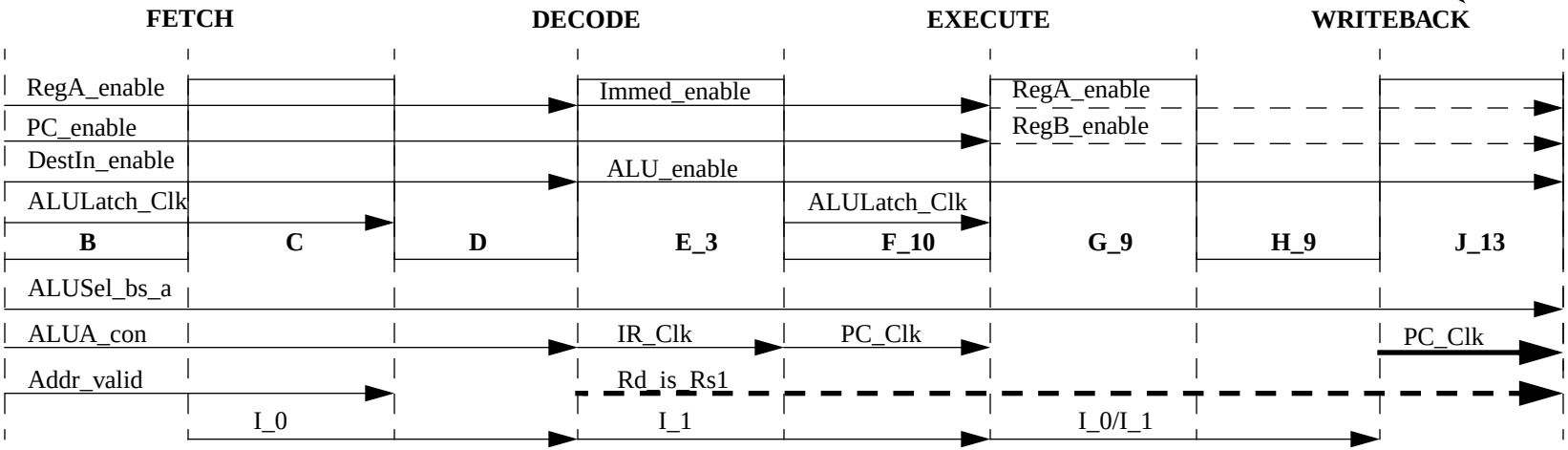
BNEQ


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
ALULatch_Clk
RegA_enable
RegB_enable
B
C
D
E_3
F_10
G_9
H_9
J_11
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1

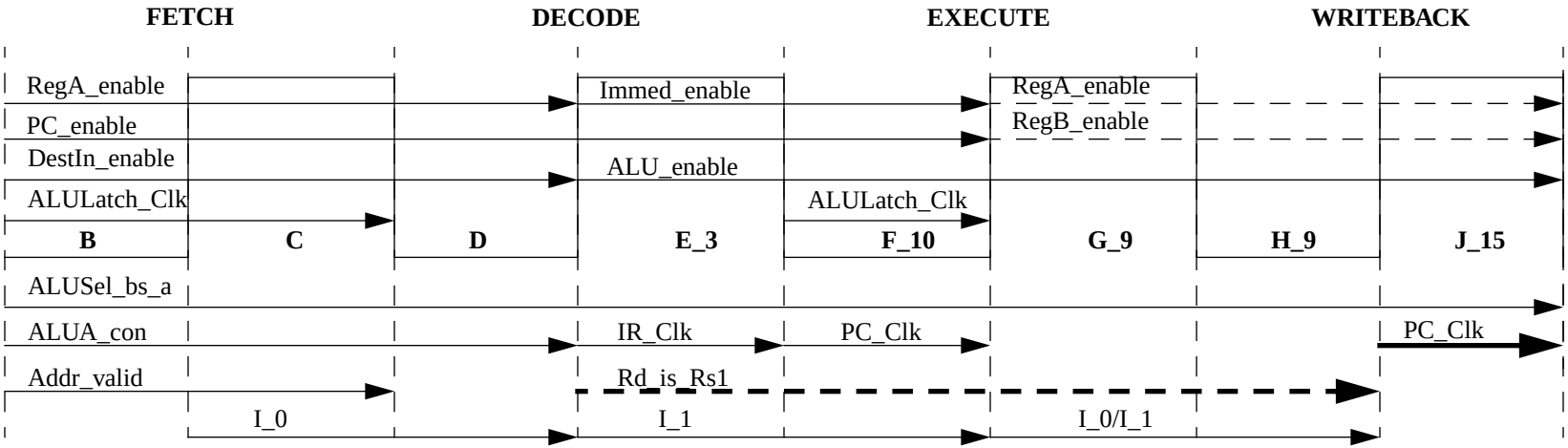
BC


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
ALULatch_Clk
RegA_enable
RegB_enable
B
C
D
E_3
F_10
G_9
H_9
J_12
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1
State+Clk+ALU_carry !
State+Clk+ALU_overflow !

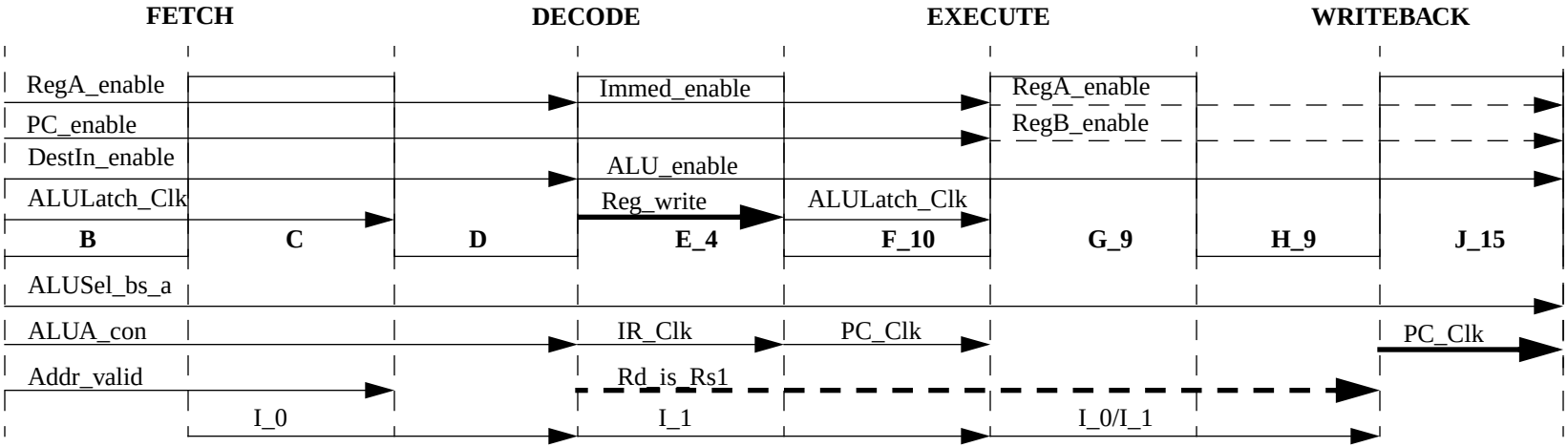
BO


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
ALULatch_Clk
RegA_enable
RegB_enable
B
C
D
E_3
F_10
G_9
H_9
J_13
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1

J


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
RegA_enable
RegB_enable
ALULatch_Clk
B
C
D
E_3
F_10
G_9
H_9
J_15
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1

JALR


FETCH
DECODE
EXECUTE
WRITEBACK
RegA_enable
PC_enable
DestIn_enable
ALULatch_Clk
Immed_enable
ALU_enable
Reg_write
RegA_enable
RegB_enable
ALULatch_Clk
B
C
D
E_4
F_10
G_9
H_9
J_15
ALUSel_bs_a
ALUA_con
IR_Clk
PC_Clk
PC_Clk
Addr_valid
Rd_is_Rs1
I_0
I_1
I_0/I_1

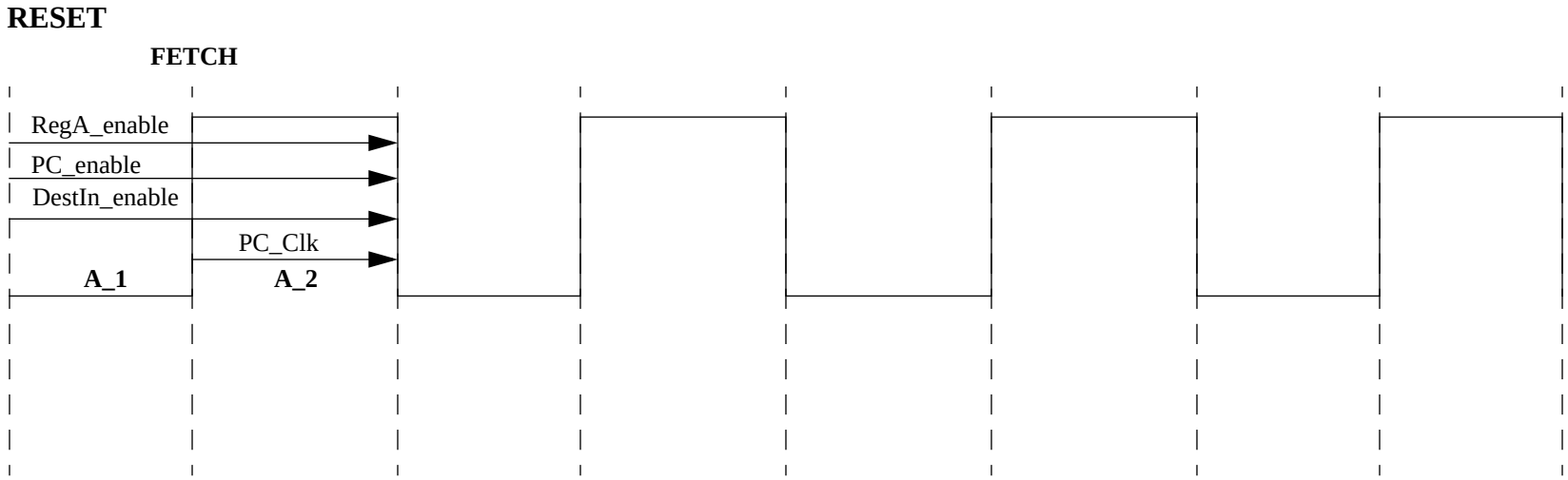

RESET
FETCH
RegA_enable
PC_enable
DestIn_enable
PC_Clk
A_1
A_2

Table 2: Code to Instruction map

Name (S1/S0/Clk)	
A_1 (000)	RESET=1: All Instructions
A_2 (001)	RESET=1: All Instructions
B (000)	RESET=0: FETCH: All Instructions
C (001)	RESET=0: FETCH: All Instructions
D (010)	DECODE: All Instructions
E_1 (011)	DECODE: SEQ + ADD + SGT + SUB + SLT + OR + AND + XOR + XNOR + SW + MULT + LW + NOP + JR + SRA + SRL + SLL
E_2 (011)	DECODE: ADDI + ORI + ANDI + NOT + LBI + LBIU + LHI
E_3 (011)	DECODE: BEQZ + BNEZ + BC + BO + J
E_4 (011)	DECODE: JALR
F_1 (100)	EXECUTE: SEQ + SGT + SLT + ADD + SUB + OR + AND + XOR + XNOR
F_2 (100)	EXECUTE: SW
F_3 (100)	EXECUTE: LW
F_4 (100)	EXECUTE: MULT
F_5 (100)	EXECUTE: NOP + JR
F_6 (100)	EXECUTE: SRA
F_7 (100)	EXECUTE: SRL
F_8 (100)	EXECUTE: ADDI + ORI + ANDI + NOT
F_9 (100)	EXECUTE: LBI + LBIU + LHI
F_10 (100)	EXECUTE: BEQZ + BNEZ + BC + BO + J + JALR
F_11 (100)	EXECUTE: SLL
G_1 (101)	EXECUTE: SEQ + SGT + SLT + ADD + SUB + ADDI
G_2 (101)	EXECUTE: OR + AND + XOR + XNOR + ORI + ANDI + NOT
G_3 (101)	EXECUTE: SW
G_4 (101)	EXECUTE: LW
G_5 (101)	EXECUTE: MULT
G_6 (101)	EXECUTE: NOP + JR + LBI + LBIU + LHI
G_7 (101)	EXECUTE: SRA
G_8 (101)	EXECUTE: SRL
G_9 (101)	EXECUTE: BEQZ + BNEZ + BC + BO + J + JALR
G_10 (101)	EXECUTE: SLL
H_1 (110)	WRITEBACK: SEQ + SGT + SLT

Table 2: Code to Instruction map

Name (S1/S0/Clk)	
H_2 (110)	WRITEBACK: ADD + SUB + ADDI
H_3 (110)	WRITEBACK: OR + AND + XOR + XNOR + ORI + ANDI + NOT
H_4 (110)	WRITEBACK: LW
H_5 (110)	WRITEBACK: MULT
H_6 (110)	WRITEBACK: NOP + JR + LBI + LBIU + LHI
H_7 (110)	WRITEBACK: SRA
H_8 (110)	WRITEBACK: SRL
H_9 (110)	WRITEBACK: BEQZ + BNEZ + BC + BO + J + JALR
H_10 (110)	WRITEBACK: SLL
J_1 (111)	WRITEBACK: SEQ + SGT + SLT + ADD + SUB + ADDI
J_2 (111)	WRITEBACK: OR + AND + XOR + XNOR + ORI + ANDI + NOT
J_3 (111)	WRITEBACK: LW
J_4 (111)	WRITEBACK: MULT
J_5 (111)	WRITEBACK: NOP
J_6 (111)	WRITEBACK: JR
J_7 (111)	WRITEBACK: LBI + LBIU + LHI
J_8 (111)	WRITEBACK: SRA
J_9 (111)	WRITEBACK: SRL
J_10 (111)	REG_ZERO = 1: WRITEBACK: BEQZ
J_11 (111)	REG_ZERO = 0: WRITEBACK: BNEZ
J_12 (111)	ALU_CARRY = 1: WRITEBACK: BC
J_13 (111)	ALU_OVERFLOW = 1: WRITEBACK: BO
J_14 (111)	WRITEBACK: SLL
J_15 (111)	WRITEBACK: J + JALR

Table 3: PLA Control lines (original order)

Output line (PLA row #)	A1	A2	B	C	D	E1	E2	E3	E4	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15
PC_Clk (4)		1								1	1	1	1	1	1	1	1	1	1	1																										1				1	1	1	1		1
IR_Clk (1)						1	1	1	1																																														
ALULatch_Clk (7)			1	1						1				1	1	1	1	1	1	1																																			
ALUOC_Clk (16)																																1																							
Addr_valid (2)			1	1							1	1											1	1										1									1												
Read_write (3)											1												1																																
Immed_enable (6)							1	1	1								1	1	1																																				
RegA_enable (12)	1	1	1	1	1	1				1	1	1	1	1	1	1				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
PC_enable (5)	1	1	1	1	1			1	1										1																																				
RegB_enable (13)						1	1			1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Mult_enable (19)													1												1										1									1											
DestIn_enable (18)	1	1	1	1	1							1												1										1									1												
ALU_enable (17)						1	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1			1	1	1	1	1	1	1	1			1	1	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1	1
ALUSel_bs (15)														1	1	1		1		1						1	1	1		1						1	1	1		1					1	1	1	1	1					1	
ALUSel_bs_a (0)			1	1	1	1	1	1	1										1		1								1		1	1							1		1									1	1	1	1		1
ALUShift_la (11)															1												1										1											1							
ALUShift_lr (9)															1	1											1	1									1	1										1	1						
ALUA_con (8)			1	1	1																																																		
Rs1_is_Rs2 (21)							1										1																																						
Rd_is_Rs1 (20)								1	1										1										1										1											1	1	1	1		
Reg_write (14)									1																																1	1	1	1			1	1	1					1	
I_0 (22)				1	1																1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1															
I_1 (23)						1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1															
ALUShift_force (10)														1				1								1										1									1	1	1								

24 control lines here + 17 more for individual instructions = 41.

Table 4: PLA Control lines (Same table reordered)

Output line (PLA row #)	A1	A2	B	C	D	E1	E2	E3	E4	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15
ALUSel_bs_a (0)			1	1	1	1	1	1	1										1		1								1		1	1							1		1									1	1	1	1		1
IR_Clk (1)						1	1	1	1																																														
Addr_valid (2)			1	1							1	1											1	1										1									1												
Read_write (3)											1												1																																
PC_Clk (4)		1								1	1	1	1	1	1	1	1	1	1	1																										1				1	1	1	1		1
PC_enable (5)	1	1	1	1	1			1	1										1																																				
Immed_enable (6)							1	1	1								1	1	1																																				
ALULatch_Clk (7)			1	1						1				1	1	1	1	1	1	1																																			
ALUA_con (8)			1	1	1																																																		
ALUShift_lr (9)															1	1											1	1									1	1										1	1						
ALUShift_force (10)														1				1								1										1									1	1	1								
ALUShift_la (11)															1												1										1											1							
RegA_enable (12)	1	1	1	1	1	1				1	1	1	1	1	1	1				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RegB_enable (13)						1	1			1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reg_write (14)									1																																1	1	1	1			1	1	1					1	
ALUSel_bs (15)														1	1	1		1		1						1	1	1		1						1	1	1		1					1	1	1	1	1					1	
ALUOC_Clk (16)																																1																							
ALU_enable (17)						1	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1			1	1	1	1	1	1	1	1			1	1	1	1	1	1	1			1	1	1	1	1	1	1	1	1	1	1
DestIn_enable (18)	1	1	1	1	1							1												1										1									1												
Mult_enable (19)													1												1										1									1											
Rd_is_Rs1 (20)								1	1										1										1										1											1	1	1	1		
Rs1_is_Rs2 (21)							1										1																																						
I_0 (22)				1	1																1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1															
I_1 (23)						1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1															

I used this as input to espresso, which minimized the logic to generate the programming information I needed for the PLA.

```
# Format of input:
# s1 s0 clk RESET oc_4 oc_3 oc_2 oc_1 oc_0

# Format of output:
# PC_Clk IR_Clk ALULatch_Clk ALUOC_Clk Addr_valid
# Read_write Immed_enable RegA_enable PC_enable RegB_enable
# Mult_enable DestIn_enable ALU_enable ALUSel_bs ALUSel_bs_a
# ALUShift_la ALUShift_lr ALUA_con Rs1_is_Rs2 Rd_is_Rs1
# Reg_write I_0 I_1 ALUShift_force

.i 9
.o 24

# A_1 (000) RESET=1: All Instructions
0001----- 000000011001000000000000

# A_2 (001) RESET=1: All Instructions
0011----- 100000011001000000000000

# B (000) RESET=0: All Instructions
0000----- 001010011001001001000000

# C (001) RESET=0: All Instructions
0010----- 001010011001001001000100

# D (010) DECODE: All Instructions
010------ 000000011001001001000100

# E_1 (011) DECODE: SEQ + ADD + SGT + SUB + SLT + OR + AND + XOR + XNOR + SW + MULT + LW +
NOP + JR + SRA + SRL + SLL
011-00000 010000010100101000000010
```

```
011-00001 01000001010010100000010
011-00010 01000001010010100000010
011-00100 01000001010010100000010
011-00101 01000001010010100000010
011-00111 01000001010010100000010
011-01001 01000001010010100000010
011-01011 01000001010010100000010
011-01100 01000001010010100000010
011-10001 01000001010010100000010
011-10010 01000001010010100000010
011-10100 01000001010010100000010
011-11100 01000001010010100000010
011-11110 01000001010010100000010
011-01110 01000001010010100000010
011-01111 01000001010010100000010
011-10000 01000001010010100000010

# E_2 (011) ADDI + ORI + ANDI + NOT + LBI + LBIU + LHI
011-00011 01000010010010100100010
011-01000 01000010010010100100010
011-01010 01000010010010100100010
011-01101 01000010010010100100010
011-10101 01000010010010100100010
011-10110 01000010010010100100010
011-10111 01000010010010100100010

# E_3 (011) BEQZ + BNEZ + BC + BO + J
011-11000 01000010100010100010010
011-11001 01000010100010100010010
011-11010 01000010100010100010010
011-11011 01000010100010100010010
011-11101 01000010100010100010010
```

```
# E_4 (011) JALR
011-11111 01000010100010100011010

# F_1 (100) SEQ + SGT + SLT + ADD + SUB + OR + AND + XOR + XNOR
100-00000 10100001010010000000010
100-00010 10100001010010000000010
100-00101 10100001010010000000010
100-00001 10100001010010000000010
100-00100 10100001010010000000010
100-00111 10100001010010000000010
100-01001 10100001010010000000010
100-01011 10100001010010000000010
100-01100 10100001010010000000010

# F_2 (100) SW
100-10001 10001101010010000000010

# F_3 (100) LW
100-10100 10001001010100000000010

# F_4 (100) MULT
100-10010 10000001011000000000010

# F_5 (100) NOP + JR
100-11100 10100001010011000000011
100-11110 10100001010011000000011

# F_6 (100) SRA
100-01110 10100001010011010000010

# F_7 (100) SRL
100-01111 10100001010011001000010
```

```
# F_8 (100) ADDI + ORI + ANDI + NOT
100-00011 10100010010010000010010
100-01000 10100010010010000010010
100-01010 10100010010010000010010
100-01101 10100010010010000010010

# F_9 (100) LBI + LBIU + LHI
100-10101 10100010010011000000011
100-10110 10100010010011000000011
100-10111 10100010010011000000011

# F_10 (100) BEQZ + BNEZ + BC + BO + J + JALR
100-11000 10100010100010100010010
100-11001 10100010100010100010010
100-11010 10100010100010100010010
100-11011 10100010100010100010010
100-11101 10100010100010100010010
100-11111 10100010100010100010010

# F_11 (100) SLL
100-10000 10100001010011000000010

# G_1 (101) SEQ + SGT + SLT + ADD + SUB + ADDI
101-00000 00000001010010100000110
101-00010 00000001010010100000110
101-00101 00000001010010100000110
101-00001 00000001010010100000110
101-00100 00000001010010100000110
101-00011 00000001010010100000110

# G_2 (101) OR + AND + XOR + XNOR + ORI + ANDI + NOT
101-00111 00000001010010000000110
101-01001 00000001010010000000110
```

```
101-01011 000000010100100000000110
101-01100 000000010100100000000110
101-01000 000000010100100000000110
101-01010 000000010100100000000110
101-01101 000000010100100000000110

# G_3 (101) SW
101-10001 000011010100100000000000

# G_4 (101) LW
101-10100 000010010101000000000110

# G_5 (101) MULT
101-10010 000000010110000000000110

# G_6 (101) NOP + JR + LBI + LBIU + LHI
101-11100 000000010100110000000111
101-11110 000000010100110000000111
101-10101 000000010100110000000111
101-10110 000000010100110000000111
101-10111 000000010100110000000111

# G_7 (101) SRA
101-01110 000000010100110110000110

# G_8 (101) SRL
101-01111 000000010100110010000110

# G_9 (101) BEQZ + BNEZ + BC + BO + J + JALR
101-11000 000000010100101000010110
101-11001 000000010100101000010110
101-11010 000000010100101000010110
101-11011 000000010100101000010110
```

```
101-11101 00000001010010100010110
101-11111 00000001010010100010110

# G_10 (101) SLL
101-10000 00000001010011000000110

# H_1 (110) SEQ + SGT + SLT
110-00000 00000001010010100000110
110-00010 00000001010010100000110
110-00101 00000001010010100000110

# H_2 (110) ADD + SUB + ADDI
110-00001 00010001010010100000110
110-00100 00010001010010100000110
110-00011 00010001010010100000110

# H_3 (110) OR + AND + XOR + XNOR + ORI + ANDI + NOT
110-00111 00000001010010000000110
110-01001 00000001010010000000110
110-01011 00000001010010000000110
110-01100 00000001010010000000110
110-01000 00000001010010000000110
110-01010 00000001010010000000110
110-01101 00000001010010000000110

# H_4 (110) LW
110-10100 00001001010100000000110

# H_5 (110) MULT
110-10010 00000001011000000000110

# H_6 (110) NOP + JR + LBI + LBIU + LHI
110-11100 00000001010011000000111
```

```
110-11110 00000001010011000000111
110-10101 00000001010011000000111
110-10110 00000001010011000000111
110-10111 00000001010011000000111

# H_7 (110) SRA
110-01110 00000001010011011000110

# H_8 (110) SRL
110-01111 00000001010011001000110

# H_9 (110) BEQZ + BNEZ + BC + BO + J + JALR
110-11000 00000001010010100010110
110-11001 00000001010010100010110
110-11010 00000001010010100010110
110-11011 00000001010010100010110
110-11101 00000001010010100010110
110-11111 00000001010010100010110

# H_10 (110) SLL
110-10000 00000001010011000000110

# J_1 (111) SEQ + SGT + SLT + ADD + SUB + ADDI
111-00000 00000001010010100001000
111-00010 00000001010010100001000
111-00101 00000001010010100001000
111-00001 00000001010010100001000
111-00100 00000001010010100001000
111-00011 00000001010010100001000

# J_2 (111) OR + AND + XOR + XNOR + ORI + ANDI + NOT
111-00111 00000001010010000001000
111-01001 00000001010010000001000
```

```
111-01011 00000001010010000001000
111-01100 00000001010010000001000
111-01000 00000001010010000001000
111-01010 00000001010010000001000
111-01101 00000001010010000001000

# J_3 (111) LW
111-10100 00001001010100000001000

# J_4 (111) MULT
111-10010 00000001011000000001000

# J_5 (111) NOP
111-11100 00000001010011000000001

# J_6 (111) JR
111-11110 10000001010011000000001

# J_7 (111) LBI + LBIU + LHI
111-10101 00000001010011000001001
111-10110 00000001010011000001001
111-10111 00000001010011000001001

# J_8 (111) SRA
111-01110 00000001010011011001000

# J_9 (111) SRL
111-01111 00000001010011001001000

# J_10 (111) REG_ZERO = 1: BEQZ
111-11000 10000001010010100010000

# J_11 (111) REG_ZERO = 0: BNEZ
```

```
111-11001 10000001010010100010000

# J_12 (111) ALU_CARRY = 1: BC
111-11010 10000001010010100010000

# J_13 (111) ALU_OVERFLOW = 1: BO
111-11011 10000001010010100010000

# J_14 (111) SLL
111-10000 00000001010011000001000

# J_15 (111) J + JALR
111-11101 10000001010010100000000
111-11111 10000001010010100000000
```

Table 5: CD PLA Specification

	S1/S0/Clk/Reset/OC_4-0	PC_Clk IR_Clk ALULatch_Clk ALUOC_Clk Addr_valid Read_write Immed_enable RegA_enable PC_enable RegB_enable Mult_enable DestIn_enable ALU_enable ALUSel_bs ALUSel_bs_a ALUShift_la ALUShift_lr ALUA_con Rs1_is_Rs2 Rd_is_Rs1 Reg_write I_0 I_1 ALUShift_force
0	011-11111	000000000000000000001000
1	011-00011	000000100000000000100000
2	011-01101	000000100000000000100000
3	100-00011	000000100000000000100000
4	100-01101	000000100000000000100000
5	110-000-1	000100000000000000000000
6	100-010-0	000000100000000000100000
7	1---01110	000000000000000100000000
8	100---000	001000000000000000000000
9	100-1011-	101000100000000000000010
10	110-00100	000100000100100000000110
11	111-11-1-	100000000000000000000000
12	1---10010	000000000110000000000000
13	100-1-1-1	001000100000000000000000
14	1---0111-	000000000000010010000000
15	111-10--0	000000010000000000001000
16	10--10001	000011010100100000000000
17	0010-----	000000000000000000000100
18	1---10100	000010010101000000000000
19	1---10000	000000000100110000000000
20	100--00-0	100000010000000000000010
21	100-00-0-	001000010000100000000000
22	1-0-11--1	000000000000001000010000
23	100-110--	101000101000000000000010
24	10--11--1	000000000000001000010000

Table 5: CD PLA Specification

	S1/S0/Clk/Reset/OC_4-0	PC_Clk IR_Clk ALULatch_Clk ALUOC_Clk Addr_valid Read_write Immed_enable RegA_enable PC_enable RegB_enable Mult_enable DestIn_enable ALU_enable ALUSel_bs ALUSel_bs_a ALUShift_la ALUShift_lr ALUA_con Rs1_is_Rs2 Rd_is_Rs1 Reg_write I_0 I_1 ALUShift_force
25	011-1011-	010000100100101000100010
26	----0-111	000000010000000000000000
27	011-010-0	010000100100101000100010
28	100-11--1	100000001000100000000010
29	1---101-1	000000000000110000000001
30	111-110--	100000010100000000000000
31	011-101-1	010000100100101000100010
32	----010-1	000000010000000000000000
33	0011-----	100000011001000000000000
34	011-110--	010000101000101000010010
35	011-11--1	010000101000101000010010
36	1---1011-	000000000100110000000001
37	100--0-0-	100000000100000000000010
38	111--01-1	000000010100100000001000
39	-1--00-0-	000000010000001000000000
40	110-1---0	000000010000000000000110
41	101-1---0	000000010000000000000110
42	111-0--0-	000000000100100000001000
43	011---100	010000010100101000000010
44	011--000-	010000010100101000000010
45	011--11-0	010000010100101000000010
46	011--00-0	010000010100101000000010
47	111-000--	000000010100101000001000
48	100--11-0	101000010100100000000010
49	111-11--1	100000010100101000000000
50	010------	000000000000001001000100
51	1---111-0	000000010100110000000001

Table 5: CD PLA Specification

	S1/S0/Clk/Reset/OC_4-0	PC_Clk IR_Clk ALULatch_Clk ALUOC_Clk Addr_valid Read_write Immed_enable RegA_enable PC_enable RegB_enable Mult_enable DestIn_enable ALU_enable ALUSel_bs ALUSel_bs_a ALUShift_la ALUShift_lr ALUA_con Rs1_is_Rs2 Rd_is_Rs1 Reg_write I_0 I_1 ALUShift_force
52	1---110--	000000000000101000010000
53	011-0---1	010000000100101000000010
54	100-0-0--	101000000100100000000010
55	101-00-0-	000000010100101000000110
56	111-01---	000000010100100000001000
57	100-0---1	101000000100100000000010
58	00-0-----	001010011001001001000000
59	110-000--	000000010100101000000110
60	101-000--	000000010100101000000110
61	110---1-1	000000010100100000000110
62	101---1-1	000000010100100000000110
63	0-0------	000000011001000000000000
64	110--1---	000000010100100000000110
65	101--1---	000000010100100000000110

Table 6:

S1/S0/Clk/ OC_4-0	ADDI ORI ANDI NOT LBI LBIU LHI BEQZ BNEZ BC BO J JALR SEQ SGT SLT SUB
---0--11	10000000000000000
---0--00	01000000000000000
---0-010	00100000000000000
---0--01	00010000000000000

Table 6:

S1/S0/Clk/ OC_4-0	ADDI ORI ANDI NOT LBI LBIU LHI BEQZ BNEZ BC BO J JALR SEQ SGT SLT SUB
----010-	0000100000000000
---1-1-0	0000010000000000
----0111	0000001000000000
---1--00	0000000100000000
----1001	0000000010000000
---1-010	0000000001000000
----1011	0000000000100000
---1110-	0000000000010000
----111-	0000000000001000
1--0-000	0000000000000100
1--0-010	0000000000000010
1---01-1	0000000000000001
1----100	0000000000000001