Description of Control Signals in Single Cycle Implementation of the LC4 ISA

Signal Name	# of bits	Value	Action
PCMux.CTL	3	0	Value of NZP register compared to bits I[11:9] of the current instruction if the test is satisfied then the output of TEST is 1 and NextPC = BRANCH Target, (PC+1) + SEXT(IMM9); otherwise the output of TEST is 0 and NextPC = PC + 1
		1	Next PC = PC+1
		2	Next PC = (PC+1) + SEXT(IMM11)
		3	Next PC = RS
		4	Next PC = (0x8000 | UIMM8)
		5	Next PC = (PC & 0x8000) | (IMM11 << 4)
rsMux.CTL	2	0	rs.addr = I[8:6]
		1	rs.addr = 0x07
		2	rs.addr = I[11:9]
rtMux.CTL	1	0	rt.addr = I[2:0]
		1	rt.addr = I[11:9]
rdMux.CTL	1	0	rd.addr = I[11:9]
		1	rd.addr = 0x07
regFile.WE	1	0	Register file not written
		1	Register file written: rd.addr indicates which register is updated with the value on the Write Input
regInput.Mux.CTL	2	0	Write Input = ALU output
		1	Write Input = Output of Data Memory
		2	Write Input = PC + 1
NZP.WE	1	0	NZP register not updated
		1	NZP register updated from Write Input to register file
DATA.WE	1	0	Data Memory not written
		1	Data Input written into location on Data Address lines
Privilege.CTL	2	0	PSR[15] = 0 – Clear privilege bit
		1	PSR[15] = 1 – Set privilege bit
		2	PSR[15] unchanged – no change to privilege bit
ALUInputMux.CTL	1	0	B[15:0] = RT[15:0] – B input to ALU = RT
		1	B[15:0] = I[15:0] – B input to ALU = Instruction Word

Signal Name	# of bits	Value	Action
ALU.CTL	6		
Arithmetic Ops		0	C = A + B : Addition
		1	C = A * B : Multiplication
		2	C = A - B : Subtraction
		3	C = A / B : Division
		4	C = A % B : Modulus
		5	C = A + SEXT(B[4:0])
		6	C = A + SEXT(B[5:0])
Logical Ops		8	C = A AND B : Bitwise Logical Product
		9	C = NOT A: Bitwise Negation
		10	C = A OR B: Bitwise Logical Sum
		11	C = A XOR B: Bitwise Exclusive OR
		12	C = A AND SEXT(B[4:0])
Comparator Ops		16	C = signed-CC(A-B) [-1, 0, +1]
		17	C = unsigned-CC(A-B) [-1, 0, +1]
		18	C = signed-CC(A-SEXT(B[6:0])) [-1, 0, +1]
		19	C = unsigned-CC(A-SEXT(B[6:0])) [-1, 0, +1]
Shifter Ops		24	C = A << B[3:0] : Shift Left Logical
		25	C = A >>> B[3:0] : Shift Right Arithmetic
		26	C = A >> B[3:0] : Shift Right Logical
Constant Ops		32	C = SEXT(B[8:0])
		33	C = (A & OxFF) | (B[7:0] << 8)