ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

FIGURE C.2.1 The truth table for the four ALU control bits (called Operation) as a function of the ALUOp and function code field. This table is the same as that shown in Figure 4.13.

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

a. The truth table for Operation2 = 1 (this table corresponds to the second to left bit of the Operation field in Figure C.2.1)

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

b. The truth table for Operation1 = 1

ALUOp		Function code fields					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

c. The truth table for Operation0 = 1

FIGURE C.2.2 The truth tables for three ALU control lines. Only the entries for which the output is 1 are shown. The bits in each field are numbered from right to left starting with 0; thus F5 is the most significant bit of the function field, and F0 is the least significant bit. Similarly, the names of the signals corresponding to the 4-bit operation code supplied to the ALU are Operation3, Operation2, Operation1, and Operation0 (with the last being the least significant bit). Thus the truth table above shows the input combinations for which the ALU control should be 0010, 0001, 0110, or 0111 (the other combinations are not used). The ALUOp bits are named ALUOp1 and ALUOp0. The three output values depend on the 2-bit ALUOp field and, when that field is equal to 10, the 6-bit function code in the instruction. Accordingly, when the ALUOp field is not equal to 10, we don't care about the function code value (it is represented by an X). There is no truth table for when Operation3=1 because it is always set to 0 in Figure C.2.1. See Appendix A for more background on don't cares.

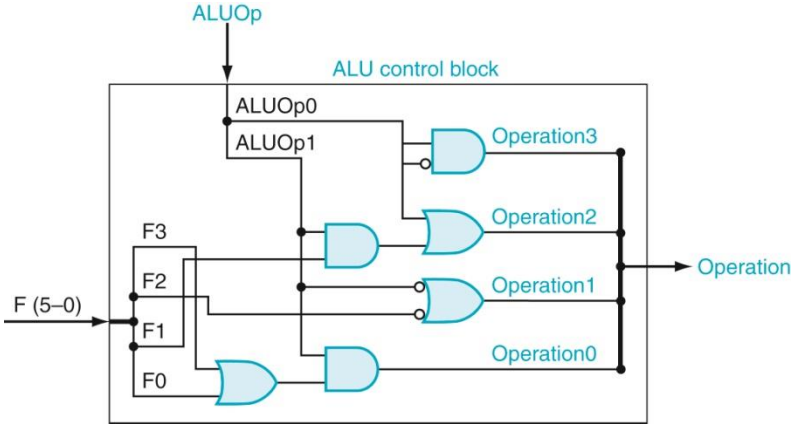


FIGURE C.2.3 The ALU control block generates the four ALU control bits, based on the function code and ALUOp bits. This logic is generated directly from the truth table in Figure C.2.2. Only 4 of the 6 bits in the function code are actually needed as inputs, since the upper 2 bits are always don't cares. Let's examine how this logic relates to the truth table of Figure C.2.2. Consider the Operation2 output, which is generated by two lines in the truth table for Operation2. The second line is the AND of two terms (F1 = 1 and ALUOp1 = 1); the top two-input AND gate corresponds to this term. The other term that causes Operation2 to be asserted is simply ALUOp0. These two terms are combined with an OR gate whose output is Operation2. The outputs Operation0 and Operation1 are derived in similar fashion from the truth table. Since Operation3 is always 0, we connect a signal and its complement as inputs to an AND gate to generate 0.

Control	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

FIGURE C.2.4 The control function for the simple one-clock implementation is completely specified by this truth table. This table is the same as that shown in Figure 4.22.

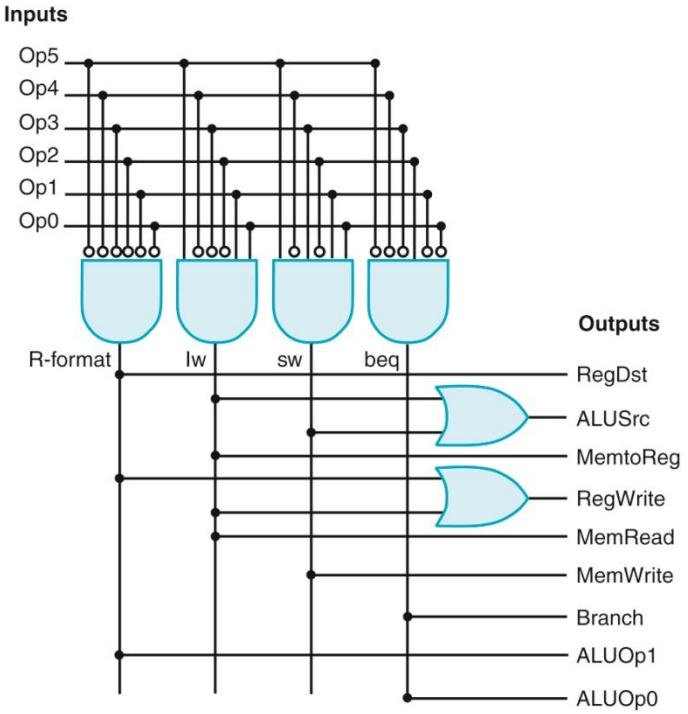


FIGURE C.2.5 The structured implementation of the control function as described by the truth table in Figure C.2.4. The structure, called a *programmable logic array (PLA), uses an array of AND gates followed by an array of OR gates. The inputs to the AND gates are the function inputs and their inverses (bubbles indicate inversion of a signal). The inputs to the OR gates are the outputs of the AND gates (or, as a degenerate case, the function inputs and inverses). The output of the OR gates is the function outputs.*

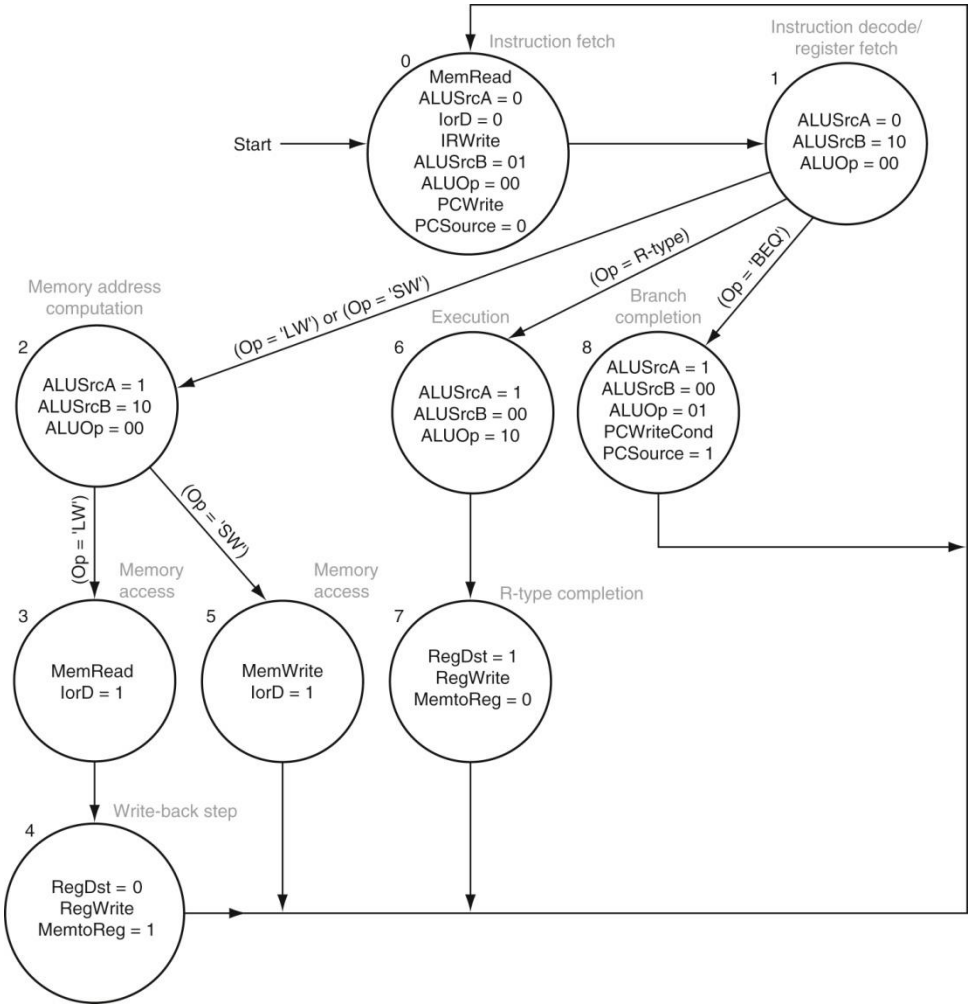


FIGURE C.3.1 The finite-state diagram for multicycle control.

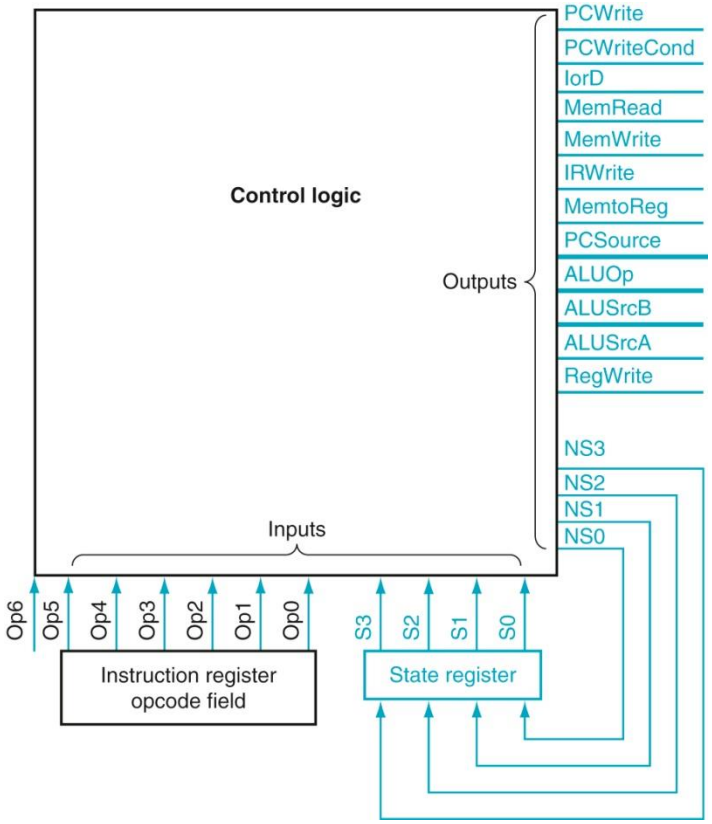
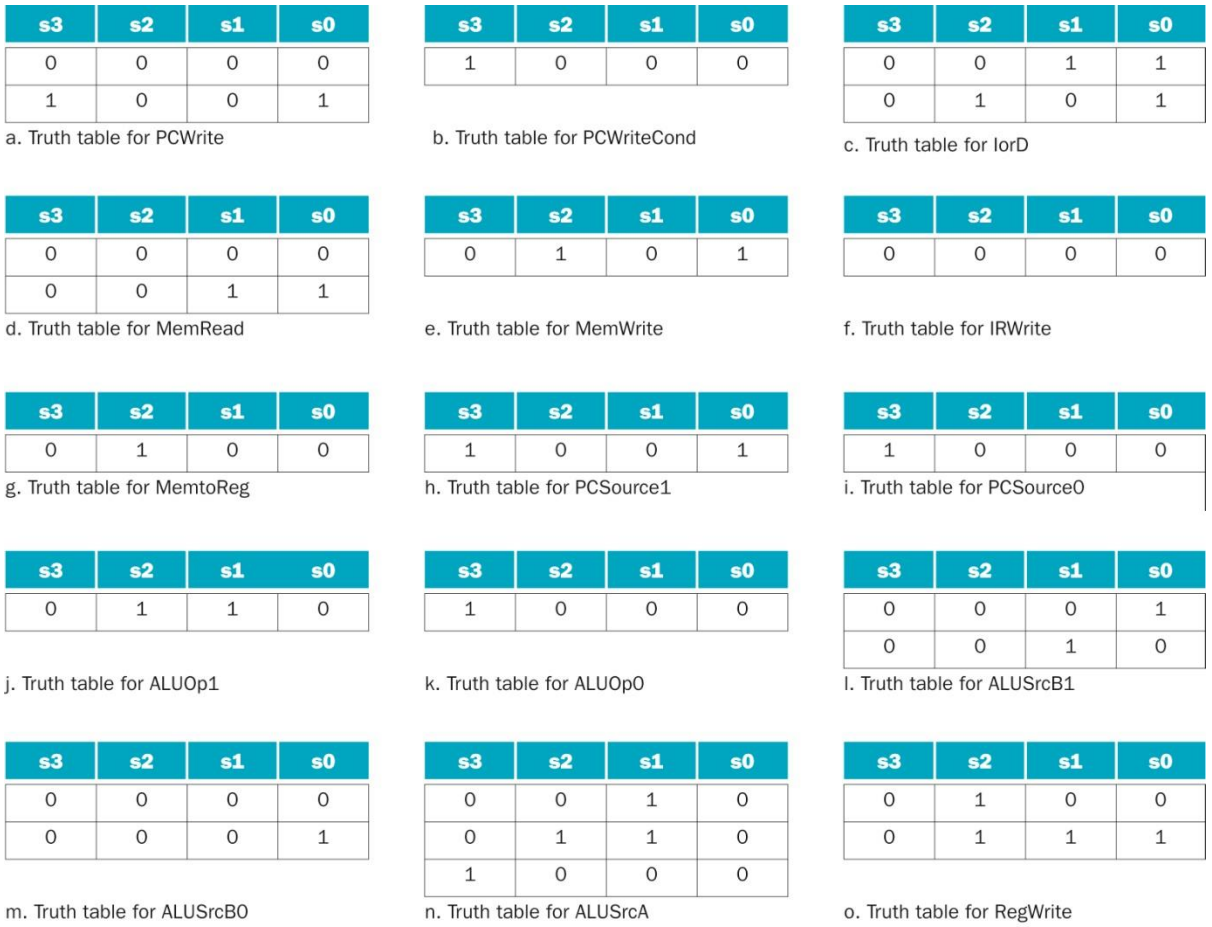


FIGURE C.3.2 The control unit for RISC-V will consist of some control logic and a register to hold the state. The state register is written at the active clock edge and is stable during the clock cycle.

Output	Current states	Op
PCWrite	state0 + state9	
PCWriteCond	state8	
IorD	state3 + state5	
MemRead	state0 + state3	
MemWrite	state5	
IRWrite	state0	
MemtoReg	state4	
PCSource1	state9	
PCSource0	state8	
ALUOp1	state6	
ALUOp0	state8	
ALUSrcB1	state1 +state2	
ALUSrcB0	state0 + state1	
ALUSrcA	state2 + state6 + state8	
RegWrite	state4 + state7	
NextState0	state4 + state5 + state7 + state8 + state9	
NextState1	state0	
NextState2	state1	(Op = 'lw') + (Op = 'sw')
NextState3	state2	(Op = 'lw')
NextState4	state3	
NextState5	state2	(Op = 'sw')
NextState6	state1	(Op = 'R-type')
NextState7	state6	
NextState8	state1	(Op = 'beq')

FIGURE C.3.3 The logic equations for the control unit shown in a shorthand form. Remember that "+" stands for OR in logic equations. The state inputs and NextState outputs must be expanded by using the state encoding. Any blank entry is a don't care.



s3	s2	s1	s0
0	0	0	0
1	0	0	1

a. Truth table for PCWrite

s3	s2	s1	s0
1	0	0	0

b. Truth table for PCWriteCond

s3	s2	s1	s0
0	0	1	1
0	1	0	1

c. Truth table for IorD

s3	s2	s1	s0
0	0	0	0
0	0	1	1

d. Truth table for MemRead

s3	s2	s1	s0
0	1	0	1

e. Truth table for MemWrite

s3	s2	s1	s0
0	0	0	0

f. Truth table for IRWrite

s3	s2	s1	s0
0	1	0	0

g. Truth table for MemtoReg

s3	s2	s1	s0
1	0	0	1

h. Truth table for PCSource1

s3	s2	s1	s0
1	0	0	0

i. Truth table for PCSource0

s3	s2	s1	s0
0	1	1	0

j. Truth table for ALUOp1

s3	s2	s1	s0
1	0	0	0

k. Truth table for ALUOp0

s3	s2	s1	s0
0	0	0	1
0	0	1	0

l. Truth table for ALUSrcB1

s3	s2	s1	s0
0	0	0	0
0	0	0	1

m. Truth table for ALUSrcB0

s3	s2	s1	s0
0	0	1	0
0	1	1	0
1	0	0	0

n. Truth table for ALUSrcA

s3	s2	s1	s0
0	1	0	0
0	1	1	1

o. Truth table for RegWrite

FIGURE C.3.4 The truth tables are shown for the 15 datapath control signals that depend only on the current-state input bits, which are shown for each table. Each truth table row corresponds to 64 entries: one for each possible value of the six Op bits. Notice that some of the outputs are active under nearly the same circumstances. For example, in the case of PCWriteCond, PCSource0, and ALUOp0, these signals are active only in state 8 (see b, i, and k). These three signals could be replaced by one signal. There are other opportunities for reducing the logic needed to implement the control function by taking advantage of further similarities in the truth tables.

Op5	Op4	Op3	Op2	Op1	Op0	S3	S2	S1	S0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	0	1

a. The truth table for the NS3 output, active when the next state is 8 or 9. This signal is activated when the current state is 1.

Op5	Op4	Op3	Op2	Op1	Op0	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	1
1	0	1	0	1	1	0	0	1	0
X	X	X	X	X	X	0	0	1	1
X	X	X	X	X	X	0	1	1	0

b. The truth table for the NS2 output, which is active when the next state is 4, 5, 6, or 7. This situation occurs when the current state is one of 1, 2, 3, or 6.

Op5	Op4	Op3	Op2	Op1	Op0	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	0	1
1	0	1	0	1	1	0	0	0	1
1	0	0	0	1	1	0	0	1	0
X	X	X	X	X	X	0	1	1	0

c. The truth table for the NS1 output, which is active when the next state is 2, 3, 6, or 7. The next state is one of 2, 3, 6, or 7 only if the current state is one of 1, 2, or 6.

Op5	Op4	Op3	Op2	Op1	Op0	S3	S2	S1	S0
X	X	X	X	X	X	0	0	0	0
1	0	0	0	1	1	0	0	1	0
1	0	1	0	1	1	0	0	1	0
X	X	X	X	X	X	0	1	1	0
0	0	0	0	1	0	0	0	0	1

d. The truth table for the NS0 output, which is active when the next state is 1, 3, 5, 7, or 9. This happens only if the current state is one of 0, 1, 2, or 6.

FIGURE C.3.5 The four truth tables for the four next-state output bits (NS[3–0]). The next-state outputs depend on the value of Op[5-0], which is the opcode field, and the current state, given by S[3–0]. The entries with X are don't-care terms. Each entry with a don't-care term corresponds to two entries, one with that input at 0 and one with that input at 1. Thus an entry with *n don't-care terms actually corresponds to 2n truth table entries.*

Outputs	Input values (S[3–0])									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
PCWrite	1	0	0	0	0	0	0	0	0	1
PCWriteCond	0	0	0	0	0	0	0	0	1	0
IorD	0	0	0	1	0	1	0	0	0	0
MemRead	1	0	0	1	0	0	0	0	0	0
MemWrite	0	0	0	0	0	1	0	0	0	0
IRWrite	1	0	0	0	0	0	0	0	0	0
MemtoReg	0	0	0	0	1	0	0	0	0	0
PCSource1	0	0	0	0	0	0	0	0	0	1
PCSource0	0	0	0	0	0	0	0	0	1	0
ALUOp1	0	0	0	0	0	0	1	0	0	0
ALUOp0	0	0	0	0	0	0	0	0	1	0
ALUSrcB1	0	1	1	0	0	0	0	0	0	0
ALUSrcB0	1	1	0	0	0	0	0	0	0	0
ALUSrcA	0	0	1	0	0	0	1	0	1	0
RegWrite	0	0	0	0	1	0	0	1	0	0

FIGURE C.3.6 The truth table for the 16 datapath control outputs, which depend only on the state inputs. The values are determined from Figure C.3.4. Although there are 16 possible values for the 4-bit state field, only 10 of these are used and are shown here. The 10 possible values are shown at the top; each column shows the setting of the datapath control outputs for the state input value that appears at the top of the column. For example, when the state inputs are 0011 (state 3), the active datapath control outputs are IorD or MemRead.

Lower 4 bits of the address	Bits 19–4 of the word
0000	1001010000001000
0001	0000000000011000
0010	0000000000010100
0011	0011000000000000
0100	0000001000000010
0101	0010100000000000
0110	0000000001000100
0111	0000000000000011
1000	0100000010100100
1001	1000000100000000

FIGURE C.3.7 The contents of the upper 16 bits of the ROM depend only on the state inputs. These values are the same as those in Figure C.3.6, simply rotated 90°. This set of control words would be duplicated 64 times for every possible value of the upper six bits of the address.

	Op [5–0]				
Current state S[3–0]	000000 (R-format)	000100 (beq)	100011 (lw)	101011 (sw)	Any other value
0000	0001	0001	0001	0001	0001
0001	0110	1000	0010	0010	Illegal
0010	XXXX	XXXX	0011	0101	Illegal
0011	0100	0100	0100	0100	Illegal
0100	0000	0000	0000	0000	Illegal
0101	0000	0000	0000	0000	Illegal
0110	0111	0111	0111	0111	Illegal
0111	0000	0000	0000	0000	Illegal
1000	0000	0000	0000	0000	Illegal
1001	0000	0000	0000	0000	Illegal

FIGURE C.3.8 This table contains the lower 4 bits of the control word (the NS outputs), which depend on both the state inputs, S[3–0], and the opcode, Op[5–0], which correspond to the instruction opcode. These values can be determined from Figure C.3.5. The opcode name is shown under the encoding in the heading. The four bits of the control word whose address is given by the current-state bits and Op bits are shown in each entry. For example, when the state input bits are 0000, the output is always 0001, independent of the other inputs; when the state is two, the next state is don't care for three of the inputs, three for lw, and five for sw. Together with the entries in Figure C.3.7, this table specifies the contents of the control unit ROM. For example, the word at address 1000110001 is obtained by finding the upper 16 bits in the table in Figure C.3.7 using only the state input bits (0001) and concatenating the lower four bits found by using the entire address (0001 to find the row and 100011 to find the column). The entry from Figure C.3.7 yields 0000000000011000, while the appropriate entry in the table immediately above is 0010. Thus the control word at address 1000110001 is 00000000000110000010. The column labeled "Any other value" applies only when the Op bits do not match one of the specified opcodes.

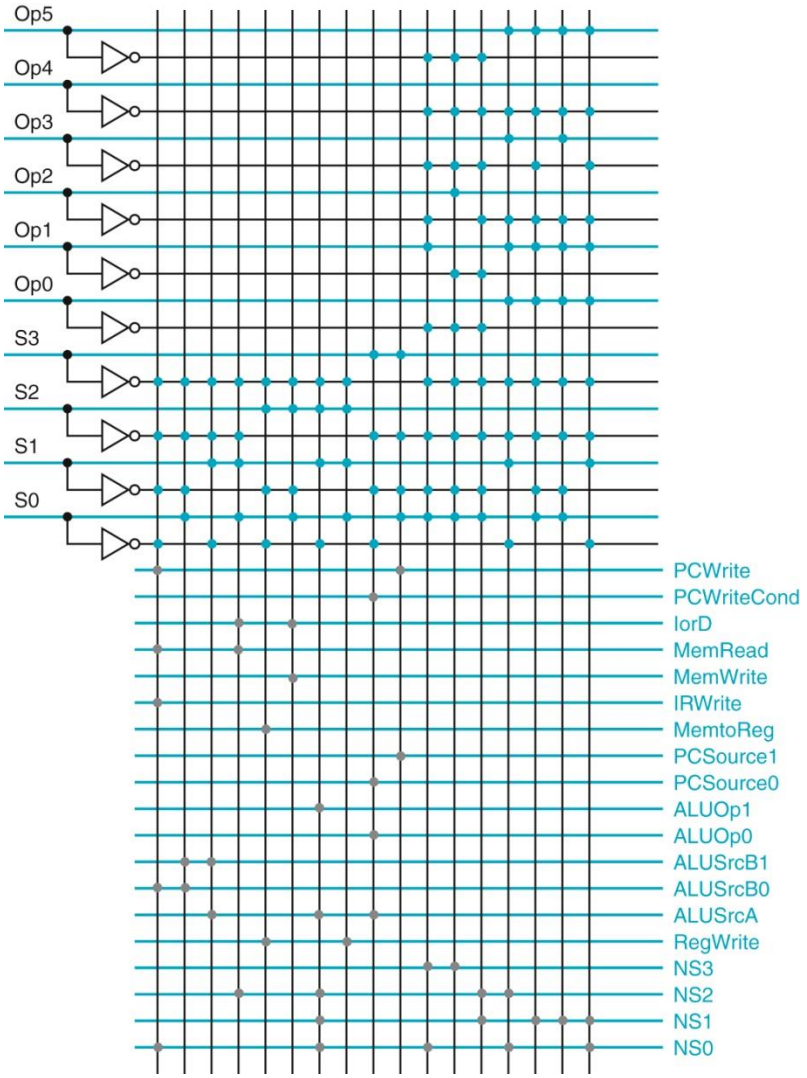


FIGURE C.3.9 This PLA implements the control function logic for the multicycle implementation. The inputs to the control appear on the left and the outputs on the right. The top half of the figure is the AND plane that computes all the minterms. The minterms are carried to the OR plane on the vertical lines. Each colored dot corresponds to a signal that makes up the minterm carried on that line. The sum terms are computed from these minterms, with each gray dot representing the presence of the intersecting minterm in that sum term. Each output consists of a single sum term.

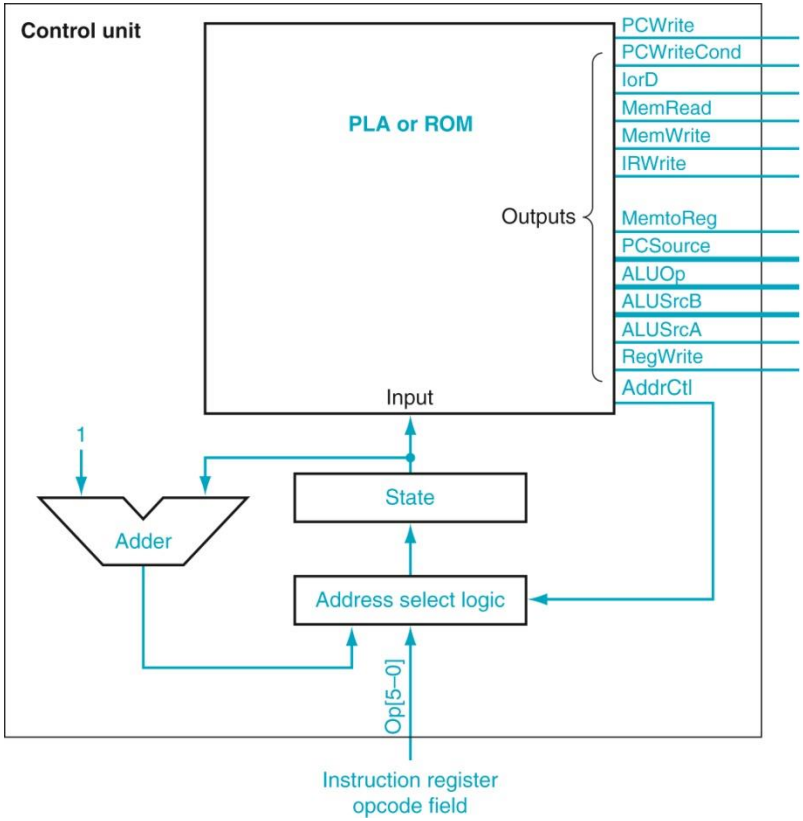


FIGURE C.4.1 The control unit using an explicit counter to compute the next state. In this control unit, the next state is computed using a counter (at least in some states). By comparison, Figure C.3.2 encodes the next state in the control logic for every state. In this control unit, the signals labeled *AddrCtl control how the next state is determined.*

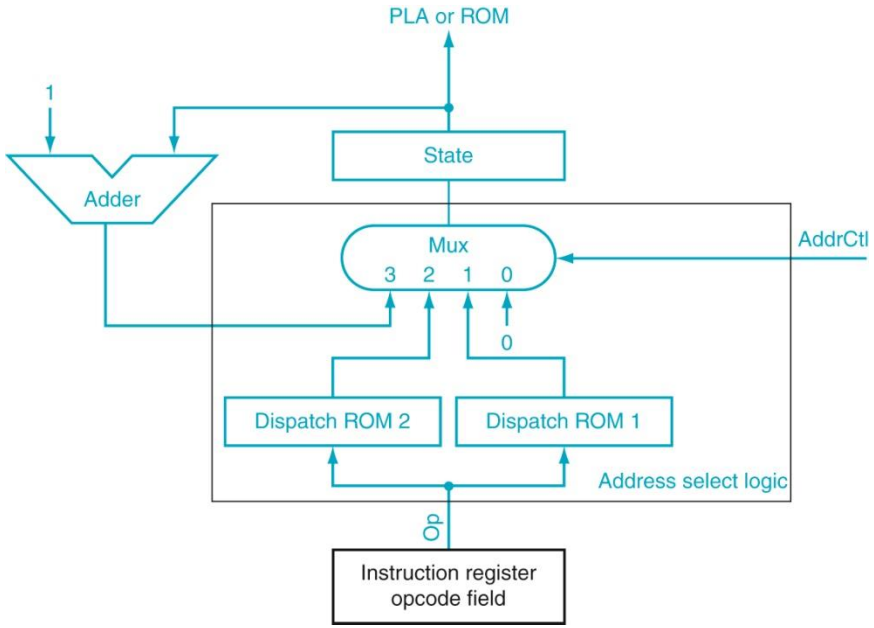


FIGURE C.4.2 This is the address select logic for the control unit of Figure C.4.1.

Dispatch ROM 1		
Op	**Opcode name**	**Value**
000000	R-format	0110
000100	beq	1000
100011	lw	0010
101011	sw	0010

Dispatch ROM 2		
Op	**Opcode name**	**Value**
100011	lw	0011
101011	sw	0101

FIGURE C.4.3 The dispatch ROMs each have 26 = 64 entries that are 4 bits wide, since that is the number of bits in the state encoding. This figure only shows the entries in the ROM that are of interest for this subset. The first column in each table indicates the value of Op, which is the address used to access the dispatch ROM. The second column shows the symbolic name of the opcode. The third column indicates the value at that address in the ROM.

State number	Address-control action	Value of AddrCtl
0	Use incremented state	3
1	Use dispatch ROM 1	1
2	Use dispatch ROM 2	2
3	Use incremented state	3
4	Replace state number by 0	0
5	Replace state number by 0	0
6	Use incremented state	3
7	Replace state number by 0	0
8	Replace state number by 0	0
9	Replace state number by 0	0

FIGURE C.4.4 The values of the address-control lines are set in the control word that corresponds to each state.

State number	Control word bits 17–2	Control word bits 1–0
0	1001010000001000	11
1	0000000000011000	01
2	0000000000010100	10
3	0011000000000000	11
4	0000001000000010	00
5	0010100000000000	00
6	0000000001000100	11
7	0000000000000011	00
8	0100000010100100	00
9	1000000100000000	00

FIGURE C.4.5 The contents of the control memory for an implementation using an explicit counter. The first column shows the state, while the second shows the datapath control bits, and the last column shows the address-control bits in each control word. Bits 17–2 are identical to those in Figure C.3.7.

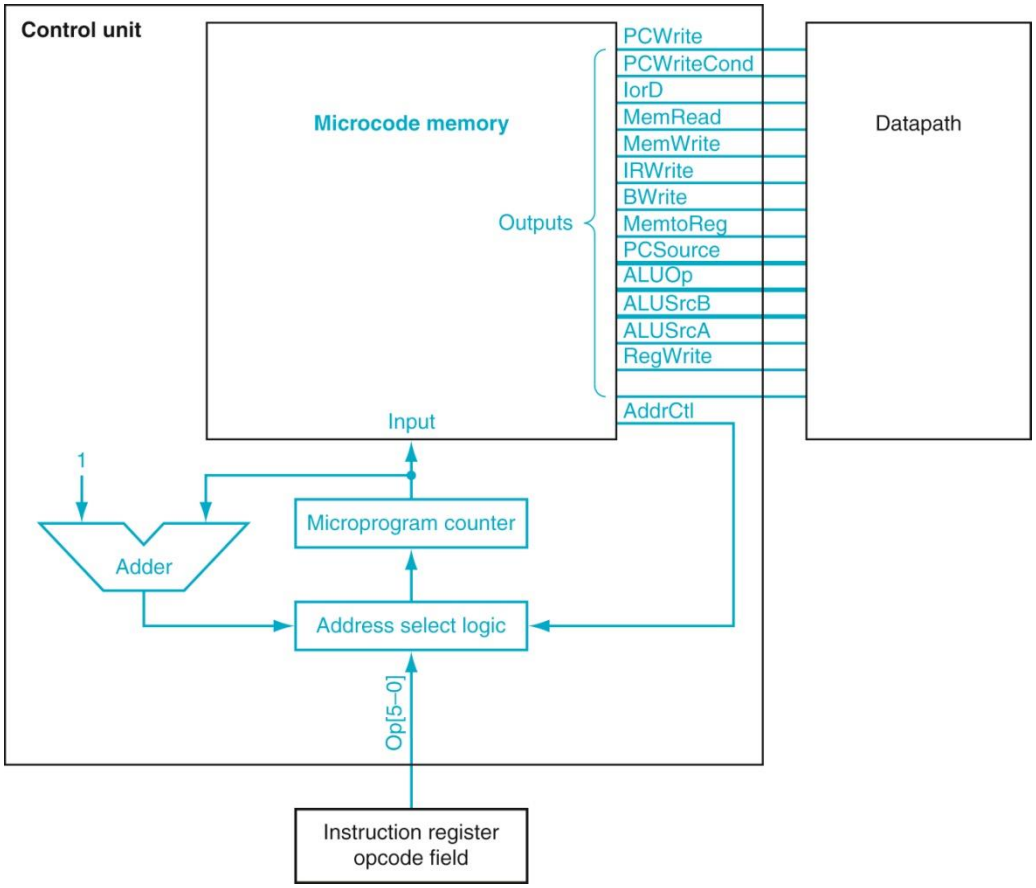


FIGURE C.4.6 The control unit as a microcode. The use of the word “micro” serves to distinguish between the program counter in the datapath and the microprogram counter, and between the microcode memory and the instruction memory.

Field name	Value	Signals active	Comment
ALU control	Add	ALUOp = 00	Cause the ALU to add.
	Subt	ALUOp = 01	Cause the ALU to subtract; this implements the compare for branches.
	Func code	ALUOp = 10	Use the instruction's function code to determine ALU control.
SRC1	PC	ALUSrcA = 0	Use the PC as the first ALU input.
	A	ALUSrcA = 1	Register A is the first ALU input.
SRC2	B	ALUSrcB = 00	Register B is the second ALU input.
	4	ALUSrcB = 01	Use 4 as the second ALU input.
	Extend	ALUSrcB = 10	Use output of the sign extension unit as the second ALU input.
	Extshft	ALUSrcB = 11	Use the output of the shift-by-two unit as the second ALU input.
Register control	Read		Read two registers using the rs and rt fields of the IR as the register numbers and putting the data into registers A and B.
	Write ALU	RegWrite, MemtoReg = 0	Write a register using the rd field of the IR as the register number and the contents of ALUOut as the data.
	Write MDR	RegWrite, MemtoReg = 1	Write a register using the rt field of the IR as the register number and the contents of the MDR as the data.
Memory	Read PC	MemRead, IorD = 0, IRWrite	Read memory using the PC as address; write result into IR (and the MDR).
	Read ALU	MemRead, IorD = 1	Read memory using ALUOut as address; write result into MDR.
	Write ALU	MemWrite, IorD = 1	Write memory using the ALUOut as address, contents of B as the data.
PC write control	ALU	PCSource = 00, PCWrite	Write the output of the ALU into the PC.
	ALUOut-cond	PCSource = 01, PCWriteCond	If the Zero output of the ALU is active, write the PC with the contents of the register ALUOut.
	Jump address	PCSource = 10, PCWrite	Write the PC with the jump address from the instruction.
Sequencing	Seq	AddrCtl = 11	Choose the next microinstruction sequentially.
	Fetch	AddrCtl = 00	Go to the first microinstruction to begin a new instruction.
	Dispatch 1	AddrCtl = 01	Dispatch using the ROM 1.
	Dispatch 2	AddrCtl = 10	Dispatch using the ROM 2.

FIGURE C.5.1 Each microcode field translates to a set of control signals to be set. These 22 different values of the fields specify all the required combinations of the 18 control lines. Control lines that are not set, which correspond to actions, are 0 by default. Multiplexor control lines are set to 0 if the output matters. If a multiplexor control line is not explicitly set, its output is a don't care and is not used.

dispatch table 1		
Opcode field	**Opcode name**	**Value**
000000	R-format	Rformat1
000100	beq	BEQ1
100011	lw	Mem1
101011	sw	Mem1

Microcode dispatch table 2		
Opcode field	**Opcode name**	**Value**
100011	lw	LW2
101011	sw	SW2

FIGURE C.5.2 The two microcode dispatch ROMs showing the contents in symbolic form and using the labels in the microprogram.

AddrCtl value	Action
0	Set state to 0
1	Dispatch with ROM 1
2	Dispatch with ROM 2
3	Use the incremented state