Term	Discussion
Control Word	A set of signals that executes a microoperation.
Microoperation	A register transfer or other operation that the CPU can execute in a single clock cycle.
Hardwired Control	Directly connects the control lines to the actual machine instructions.
Microprogrammed Control	Employs software consisting of microinstructions that carry out an instruction's microoperations.

4.13.2 Hardwired Control

Bus Address	Component
0	**Memory**
1	**MAR**
2	**PC**
3	**MBR**
4	**AC**
5	**InReg**
6	**OutReg**
7	**IR**

Destination				Origin		
IR			to	MAR		
7				1		
P_5	P_4	P_3		P_2	P_1	P_0
1	1	1		0	0	1

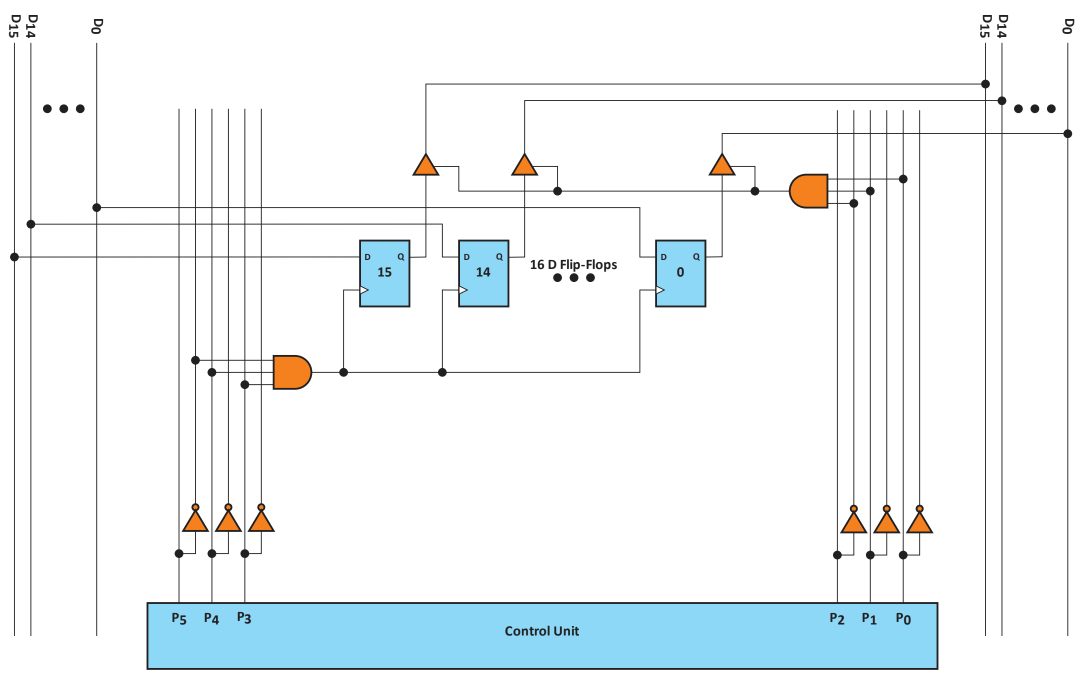
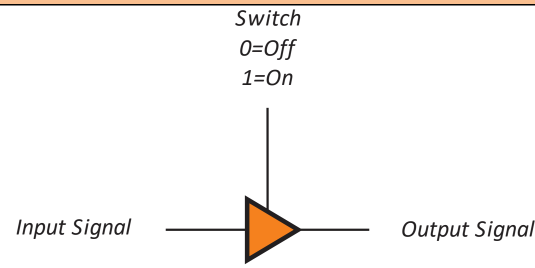
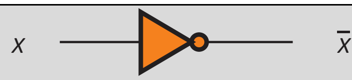


Figure 4.15 Connection of MARIE's MBR to the Datapath

Device	Operation
	Tri-state device: • If the switch is off, then the input signal is **NOT** transmitted to the output signal. • If the switch is on, then the input signal **IS** transmitted to the output signal.
	Inverter: • If $x = 0$, then $\overline{x} = 1$ • If $x = 1$, then $\overline{x} = 0$

- The set of signals $\{P_2, P_1, P_0\}$ are assigned to **read** values put on the data bus.
- The set of signals $\{P_5, P_4, P_3\}$ are assigned to **write** values to the data bus.
- Please note that a signal has a default value of 0. The significance of this feature is that to make the hypothetical transfer IR←MAR, we need only assign $[P_5, P_4, P_3\}$←111 (7) and $[P_2, P_1, P_0]$←001 (1). However, in practice, we only assign P_0 ←1, because the other signals P_2 and P_1 are, by default, zero (0).
- We can imagine sets of D-Flip-Flops for the other registers MAR, PC, AC, InREG, and OutReg.

Bus Address	Device
0	Main Memory
1	MAR
2	PC
3	MBR
4	AC
5	InReg
6	OutReg
7	IR

MARIE's Datapath Addresses

ALU Control Signals		ALU Response
A_1	A_0	
0	0	Do Nothing
0	1	AC←AC+MBR
1	0	AC←AC-MBR
1	1	AC←0 (Clear)

Table 4.8 ALU Control Signals and Response

L_{ALT}	Action
0	Load the AC from the Data bus Load the MBR from the Data bus
1	Load the AC from the ALU Load the MBR from the AC

M_R	M_W	Action
0	0	Do Nothing
0	1	Memory Write Enable
1	0	Memory Read Enable
1	1	(Impossible)

I_{PC}	Action
0	Do Nothing
1	PC ← PC + 1

T_0	Q_0	T_1	Q_1	T_2	Q_2	T_3	Q_3	T_4
1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0

- Text asserts seven clock cycles, $T_0 – T_6$, are needed. Find the longest instruction, JnS.

- Including the clock cycles needed to fetch an instruction, ten clock cycles are needed, $T_0 – T_9$.

Add X		
Control Signals	**RTN**	**Comment**
$T_0P_3P_1$	MAR ← PC	Fetch
$T_1P_5P_4P_3M_R$	IR ← M[MAR]	
T_2I_{PC}	PC ← PC + 1	
$T_3P_3P_2P_1P_0$	MAR ← X(IR[11 – 0])	Decode
$T_4P_4P_3M_R$	MBR ← M[MAR]	Execute
$T_5P_5A_0L_{ALT}C_R$	AC ← AC + MBR	

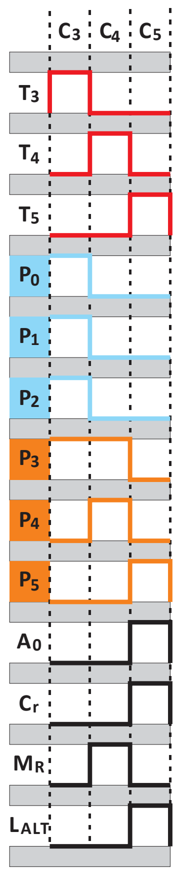


Figure 4.16 Timing Diagram for the Microoperations of MARIE's Add Instruction

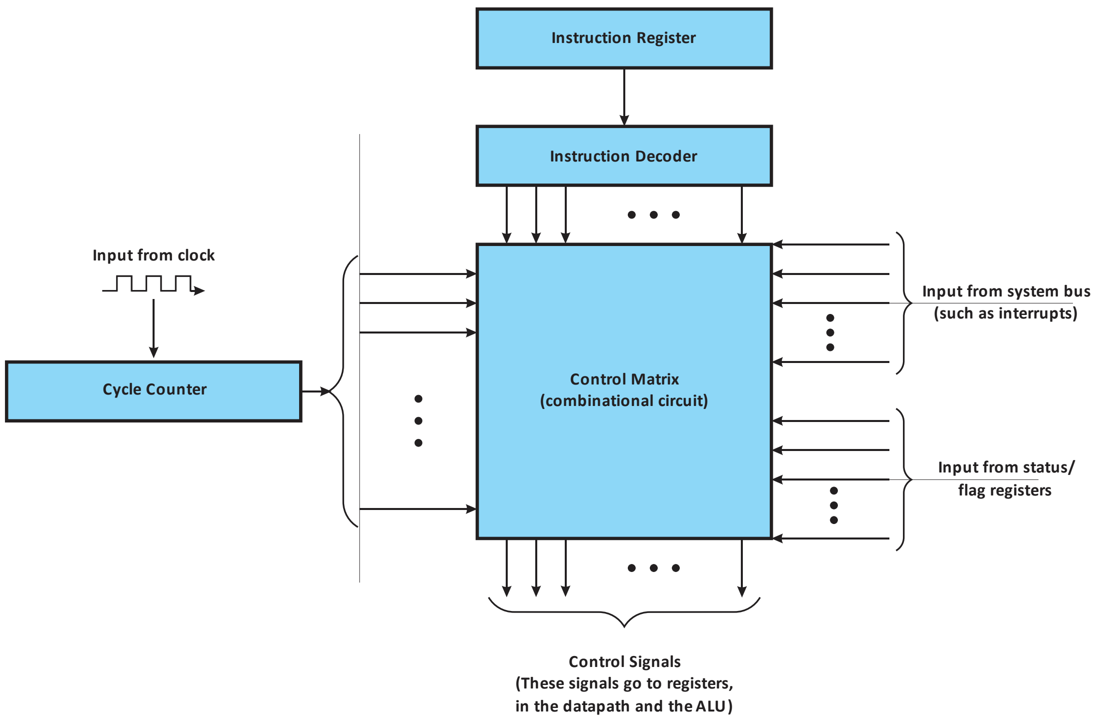


Figure 4.17 Hardwired Control Loop

Figure 4.18 Partial Instruction Decoder for MARIE's Instruction Set

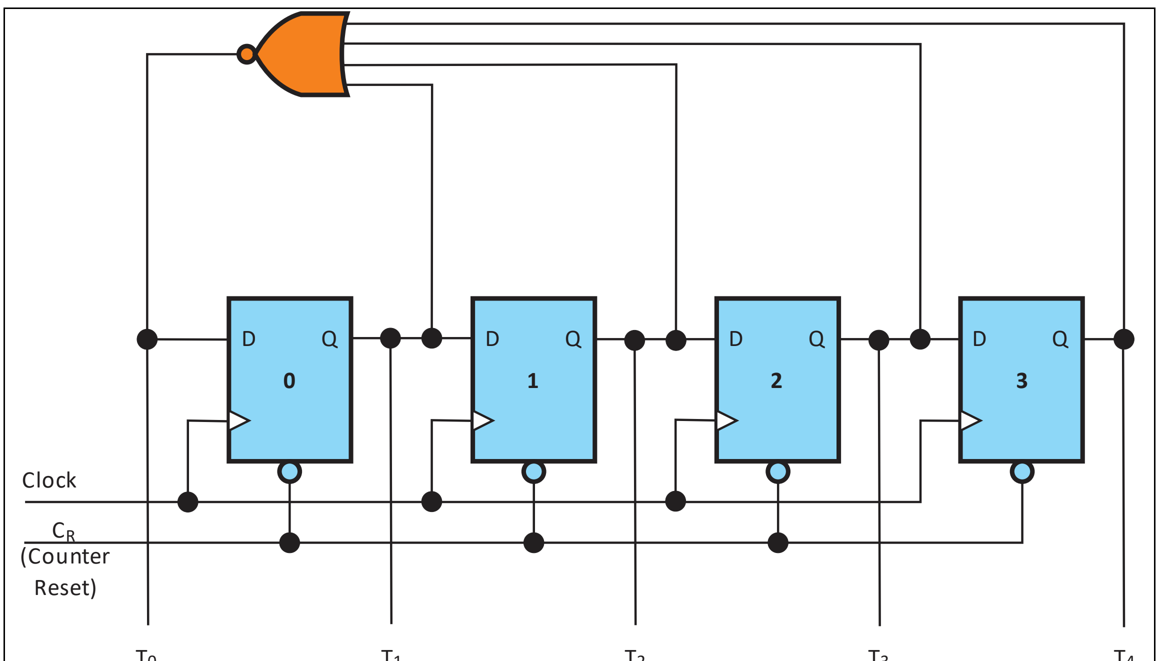


Figure 4.19 Ring Counter Using D Flip-Flops

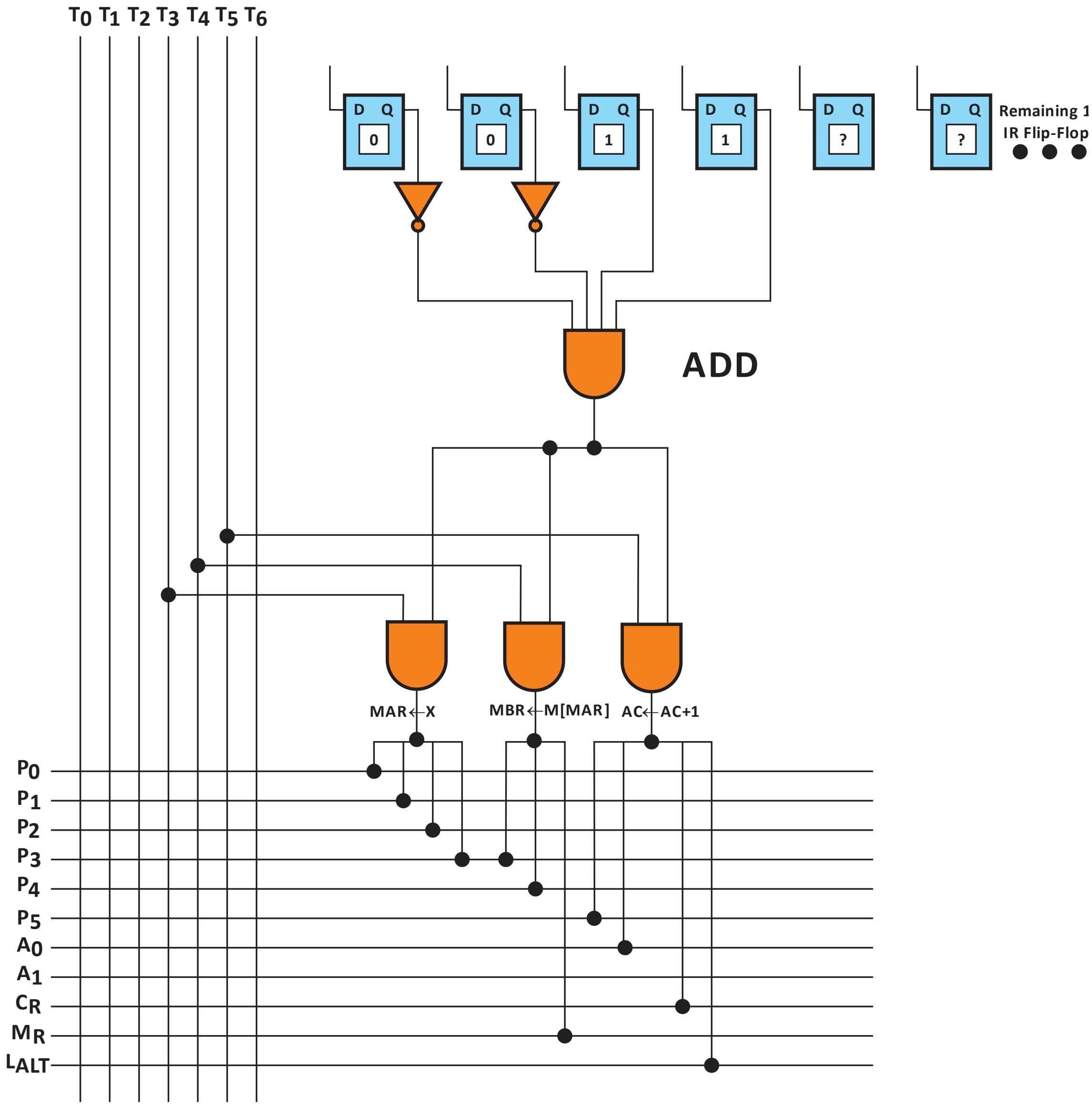


Figure 4.20 Combinational Logic for Signal Controls of MARIE's Add Instruction

4.13.3 Microprogrammed Control

- In microprogrammed control, instruction **microcode** produces the necessary control signals.

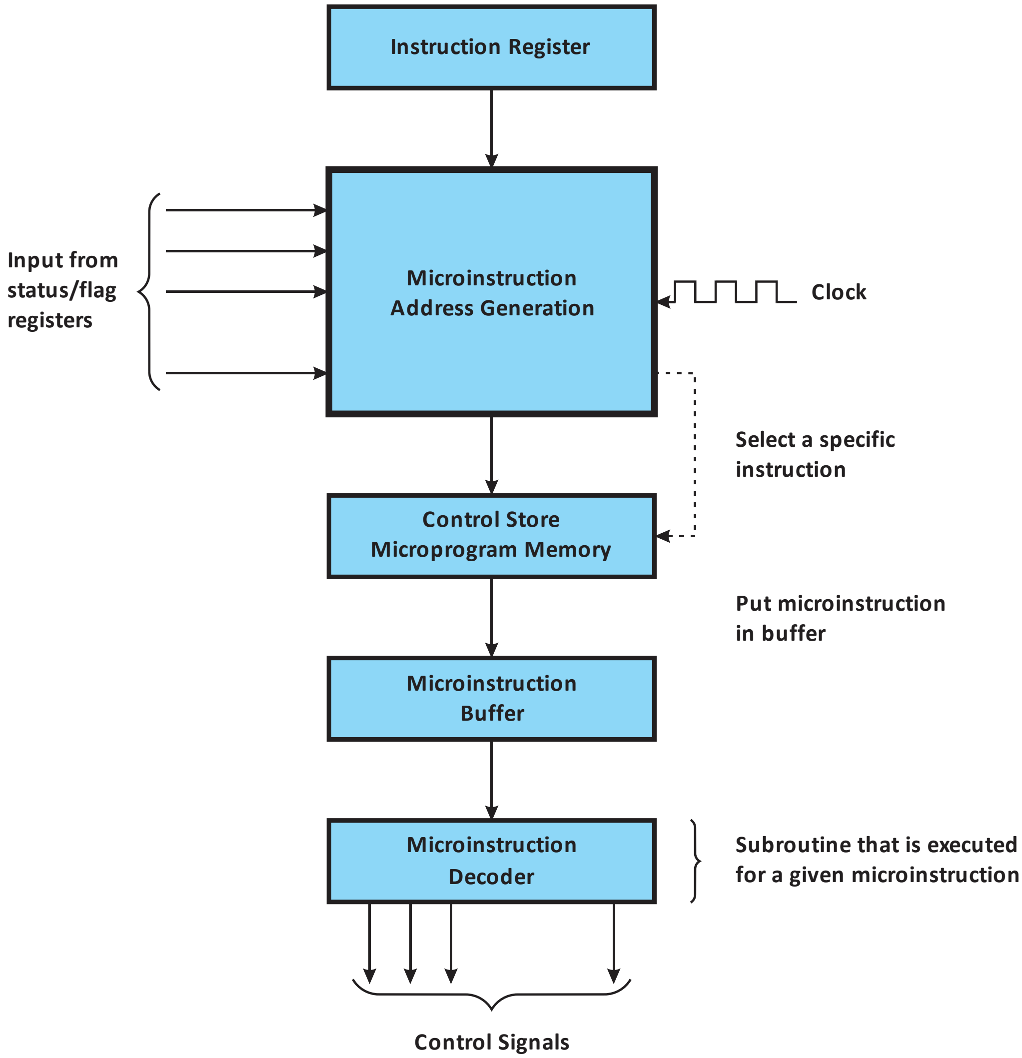


Figure 4.21 Microprogrammed Control Unit

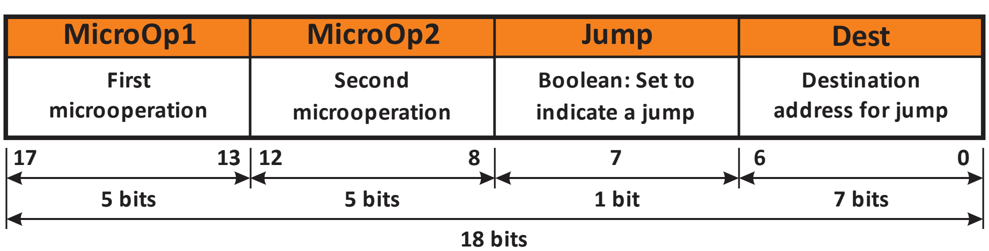


Figure 4.22 MARIE's Microinstruction Format

- All machine instructions are input into a special program, the **microprogram,** that converts machine instructions of 0s and 1s into control signals.

- The microprogram is, essentially, an interpreter, written in microcode, that is stored in **firmware** (ROM, PROM, or EPROM), which is often referred to as the control store.
- The **microsequencer** is like the program counter that selects the next microinstruction to execute.
- MicroOp1 and MicroOp2 are binary codes for each unique microoperation specified in the RTN for MARIE's instruction set.
- So far, we have defined 22 unique microoperations.
- We need a NOP – no operation microcode.
- We need a comparison microoperation that compares the bit pattern in the first 4 bits of the instruction register (IR[14-12]) to a literal value that is in the first 4 bits of the MicoOp2 field.
- MARIE's entire microprogram consists of fewer than 128 statements, so each statement can be uniquely identified by seven (7) bits. Each microinstruction has a seven-bit address.

MicroOp Code	Microoperation	MicoOp Code	Microoperation
00000	NOP	01101	MBR ← M[MAR]
00001	AC ← 0	01110	OutREG ← AC
00010	AC ← MBR	01111	PC ← IR[11 – 0]
00011	AC ← AC - MBR	10000	PC ← MBR
00100	AC ← AC + MBR	10001	PC ← PC + 1
00101	AC ← InREG	10010	If AC = 0
00110	IR ← M[MAR]	10011	If AC > 0
00111	M[MAR] ← MBR	10100	If AC < 0
01000	MAR ← IR[11 - 0]	10101	If IR[11 – 10] = 00
01001	MAR ← MBR	10110	If IR[11 – 10] = 01
01010	MAR ← PC	10111	If IR[11 – 10] = 10
01011	MAR ← X	11000	If IR[15 – 12] = MicroOp2[4 – 1]
01100	MBR ← AC		

Table 4.9 Microoperation Codes and Corresponding MARIE RTL

Address	MicroOp1	MicroOp2	Jump	Dest
000 0000	MAR ← PC	NOP	0	000 0000
000 0001	IR ← M[MAR]	NOP	0	000 0000
000 0010	PC ← PC + 1	NOP	0	000 0000
000 0011	MAR ← IR[11 - 0]	NOP	0	000 0000
000 0100	If IR[15 – 12] = MicroOp2[4 – 1]	00000 JnS X	1	010 000
000 0101	If IR[15 – 12] = MicroOp2[4 – 1]	00010 Load X	1	010 0111
000 0110	If IR[15 – 12] = MicroOp2[4 – 1]	00100 Store X	1	010 1010
000 0111	If IR[15 – 12] = MicroOp2[4 – 1]	00110 Add X	1	010 1100
000 1000	If IR[15 – 12] = MicroOp2[4 – 1]	01000 Subt X	1	010 1111
. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .
010 1010	MAR ← X	MBR ← AC	0	000 0000
010 1011	M[MAR] ← MBR	NOP	1	000 0000
010 1100	MAR ← X	NOP	0	000 0000
010 1101	M[MAR] ← MBR	NOP	0	000 0000
010 1110	AC ← AC + MBR	NOP	1	000 0000
010 1111	MAR ← X	NOP	0	000 0000
011 0000	M[MAR] ← MBR	NOP	0	000 0000
011 0001	AC ← AC - MBR	NOP	1	000 0000
. . .	. . .	. . .	. . .	. . .

FIGURE 4.23 Selected Statements in MARIE's Microprogram

- It's important to remember that a microprogrammed control unit works like a system-in-miniature.
- Microinstructions are fetched, decoded, and executed in the same manner as regular instructions.
- This extra level of instruction interpretation is what makes microprogrammed control slower than hardwired control.
- The advantages of microprogrammed control are that it can support very complicated instructions and only the microprogram needs to be changed if the instruction set changes (or an error is found).