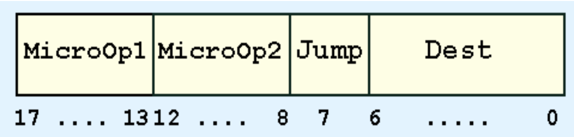
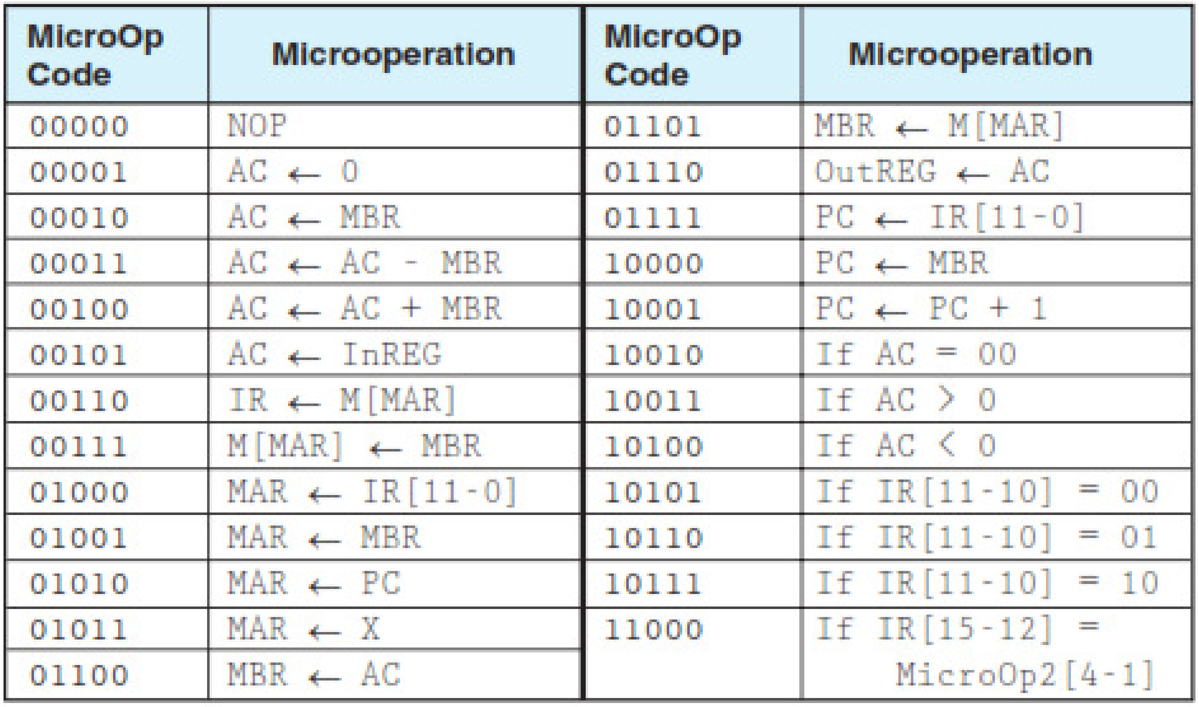
The microprogrammed version of MARIE executes a fixed microprogam to perform the fetch-decode-execute cycle. The instruction format for the microinstructions look like:



MicroOp1 encodes the type of register transfer notation (RTN) to perform (e.g., AC ← MBR is 00010_2)

MicroOp2 is used only when decoding the instruction. It contains the binary codes for each instruction to allow comparison to the IR opcode. (Since the MARIE opcodes are only 4-bits long, only bits 12 - 9 are used and bit 8 is unused.

Jump is a single bit indicating that the value in the **Dest** field is a valid micro-address and should be placed in the microsequencer; if **Jump** is "FALSE" (0), then increment to the next microinstruction.



MicroOp Code	Microoperation	MicroOp 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 = 00
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

We need to augment this table to include a few omitted microoperations and because we modified Figure 4.9 to remove the Memory from direct connection to the datapath. The following additional microoperations are used.

MicroOp Code	Microoperation
00110*	IR ← MBR

* This microop code is being reused.

a) Write the microprogram for the LOAD and STORE instructions.

b) Write the microoperations/RTN (register transfer notation) and the microprogram for the SKIPCOND instruction.

Revised Figure 4.23 Partial Microprogram

Part of Cycle	RTN (of MicroOp1)	μAddr	MicroOp1	MicroOp2	Jump	Dest
Fetch	MAR ← PC	0	01010	00000	0	0
	MBR ← M[MAR]	1	01101	00000	0	0
	IR ← MBR	2	00110	00000	0	0
	PC ← PC + 1	3	10001	00000	0	0
Decode	If ADD, Jump	4	11000	00110	1	19_{10}
("Jump Table")	If LOAD, Jump	5	11000	00010	1	
	If STORE, Jump	6	11000	00100	1	
	If SKIPCOND, Jump	7	11000	10000	1	
	If SUBT, Jump	8	11000	01000	1	
	If JUMP, Jump	9	11000	10010	1	
	If ADDI, Jump	10	11000	10110	1	
	If CLEAR, Jump	11	11000	10100	1	
	If JNS, Jump	12	11000	00000	1	
	If JUMPI, Jump	13	11000	11000	1	
	If INPUT, Jump	14	11000	01010	1	
	If OUTPUT, Jump	15	11000	01100	1	
	If LOADI, Jump	16	11000	11010	1	
	If STOREI, Jump	17	11000	11100	1	
	If HALT, Jump	18	11000	01110	1	0
Execute ADD	MAR ← IR[11-0]	19	01000	00000	0	0
	MBR ← M[MAR]	20	01101	00000	0	0
	AC ← AC + MBR	21	00100	00000	1	0
Execute LOAD	MAR ← IR[11-0]	22				
	MBR ← M[MAR]	23				
	AC ← MBR	24				
Execute STORE	MAR ← IR[11-0]	25				
	MBR ← AC	26				
	M[MAR] ← MBR	27				
Execute SKIPCOND		28				
		29				
(here only if		30				
IR[11-10]=10)		31				
		32				
		33				
		34				
		35				
		36				
		37				
		38				
		39				