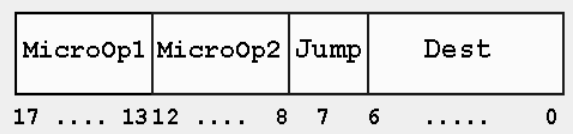
Name:_____

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




**`MicroOp1`** encodes the type of register transfer notation (RTN) to perform (e.g., AC ← 0 is $00010_2$)

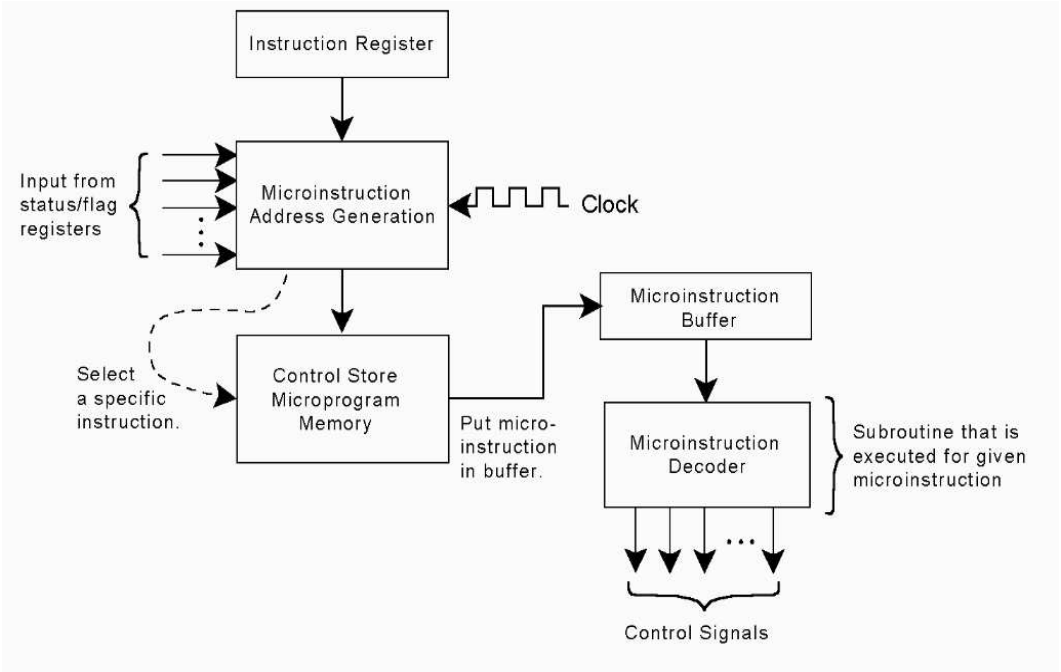
**`MicroOp2`** contains the binary codes for each instruction to allow comparison to the IR opcode.

**`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.

Table 4.8. Microoperation Codes and Corresponding MARIE RTN (p. 221)

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

Figure 4.19. Microprogrammed Control Unit



Name:_____

Notes on the Microprogrammed Control Unit:

- 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 complcated instructions and only the microprogram needs to be changed if the instruction set changes (or an error is found).

## Revised Figure 4.21 Partial Microprogram

| Part of Cycle | RTN (of MicroOp1) | μAddr | MicroOp 1 | MicroOp 2 | Jump | Dest |
|---|---|---|---|---|---|---|
| Fetch | MAR ← PC | 0 | 01001 | 0000 | 0 | 0 |
| | MBR ← M[MAR] | 1 | 01100 | 0000 | 0 | 0 |
| | IR ← MBR | 2 | 00101 | 0000 | 0 | 0 |
| | PC ← PC + 1 | 3 | 10000 | 0000 | 0 | 0 |
| Decode | If ADD, Jump | 4 | 10111 | 00110 | 1 | $17_{10}$ |
| | If LOAD, Jump | 5 | 10111 | 00010 | 1 | |
| ("Jump | If STORE, Jump | 6 | 10111 | 00100 | 1 | |
| Table") | If SKIPCOND, Jump | 7 | 10111 | 10000 | 1 | |
| | If SUBT, Jump | 8 | 10111 | 01000 | 1 | |
| | If JUMP, Jump | 9 | 10111 | 10010 | 1 | |
| | If ADDI, Jump | 10 | 10111 | 10110 | 1 | |
| | If CLEAR, Jump | 11 | 10111 | 10100 | 1 | |
| | If JNS, Jump | 12 | 10111 | 00000 | 1 | |
| | If JUMPI, Jump | 13 | 10111 | 11000 | 1 | |
| | If INPUT, Jump | 14 | 10111 | 01010 | 1 | |
| | If OUTPUT, Jump | 15 | 10111 | 01100 | 1 | |
| | If HALT, Jump | 16 | | | | |
| Execute ADD | MAR ← IR[11-0] | 17 | 00111 | 00000 | 0 | 0 |
| | MBR ← M[MAR] | 18 | 01100 | 00000 | 0 | 0 |
| | AC ← AC + MBR | 19 | 00011 | 00000 | 1 | 0 |
| Execute LOAD | MAR ← IR[11-0] | 20 | | | | |
| | MBR ← M[MAR] | 21 | | | | |
| | AC ← MBR | 22 | | | | |
| Execute STORE | MAR ← IR[11-0] | 23 | | | | |
| | MBR ← AC | 24 | | | | |
| | M[MAR] ← MBR | 25 | | | | |
| Execute SKIPCOND | | 26 | | | | |
| | | 27 | | | | |
| | | 28 | | | | |
| | | 29 | | | | |
| | | 30 | | | | |
| | | 31 | | | | |
| | | 32 | | | | |
| | | 33 | | | | |
| | | 34 | | | | |