Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 306, Fall 2019
Yale Patt, Instructor
TAs: Sabee Grewal, Arjun Ramesh, Joesph Ryan, Chirag Sakhuja, Meiling Tang, Grace Zhuang
Exam 1, October 16, 2019

Name:_____Solution_____

Problem 1 (25 points):___25___

Problem 2 (15 points):___15___

Problem 3 (15 points):___15___

Problem 4 (20 points):___20___

Problem 5 (25 points):___25___

Total (100 points):_____100_____

great job!

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

**I will not cheat on this exam.**

_____
    Signature

**GOOD LUCK!**

Name:_____

**Problem 1.** (25 points):

**Part a**. (5 points): How many of the 15 LC-3 instructions load the MAR during its instruction cycle?

<span style="color:red">everyone</span>

| 15 |
|----|

**Part b**. (5 points): Write the decimal value 23 in the following representations:

6-bit unsigned binary:

| 0 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|

6-bit 2's complement:

| 0 | 1 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|

3-digit hexadecimal:

| 0 | 1 | 7 |
|---|---|---|

3-digit base-7:

| 0 | 3 | 2 |
|---|---|---|

$$\underline{0} \times 7^2 + \underline{3} \times 7^1 + \underline{2} \times 7^0$$

**Part c**. (5 points): A computer's ALU operates on X-bit operands. When used to add a positive integer Y to the value +21, the ALU output is -20. What is the minimum number of bits (i.e. X) used to specify each operand that will produce this result? What must Y be to produce this result?

Minimum number of bits:

| 6 |
|---|

Y:

| 23 |
|----|

$$
\begin{array}{c}
\phantom{+}0\ 0\ 1\ 0\ 1\ 0\ 1 \\
+\ 1\ 0\ 1\ 0\ 1\ 1\ 1 \\
\hline
1\ 1\ 0\ 1\ 1\ 0\ 0
\end{array}
$$

$$-20 = \overline{20} + 1$$

$$
\begin{array}{c}
\phantom{+}0010100 \\
+\ 1101011 \\
\hline
1101100
\end{array}
$$

2

**Part d**. (5 points): Many ISAs have a conditional load instruction (LDC), which loads a value from memory into a register based on the condition codes. We could add that instruction to the LC-3 ISA using the unused opcode. Further we could use the BEN bit (BEN = (IR[11] AND N) OR (IR[10] AND Z) OR (IR[9] AND P)) the same way we use BEN to determine whether to take the conditional branch. The LDC instruction has three operands: DR, PC offset, and the nzp bits.

If a program contained an LDC instruction in memory location x4000, what is the largest memory address that can provide the value to be loaded into DR?
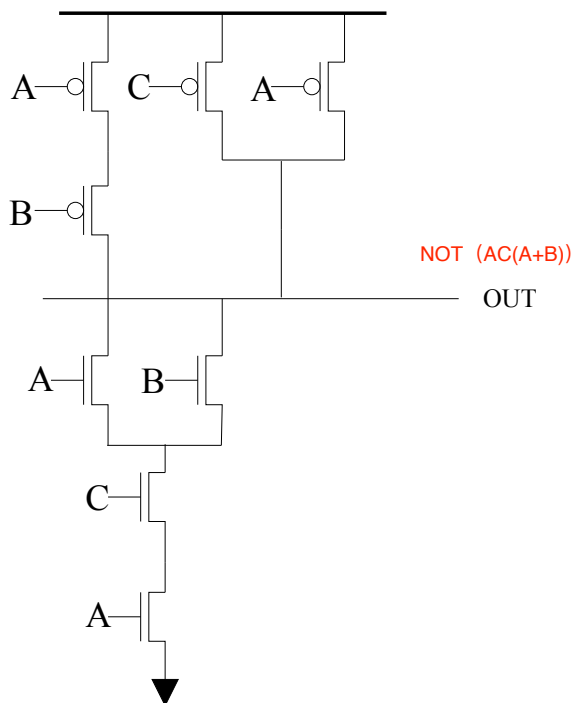
$$\begin{array}{cccc} 4 & 3 & 3 & 6 \\ \hline \text{opcode} & \text{nzp} & \text{DR} & \text{off} \end{array}$$
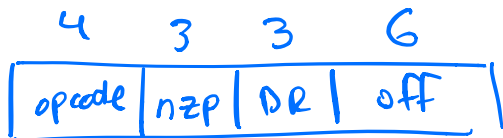
Largest memory address: $\boxed{\text{x }4020}$

$$\text{x}4000 + 1 + 011111_2 = \text{x }4001 + \text{x }1F = \text{x }4020$$

**Part e**. (5 points): Construct the truth table for the function OUT produced by the transistor circuit shown.
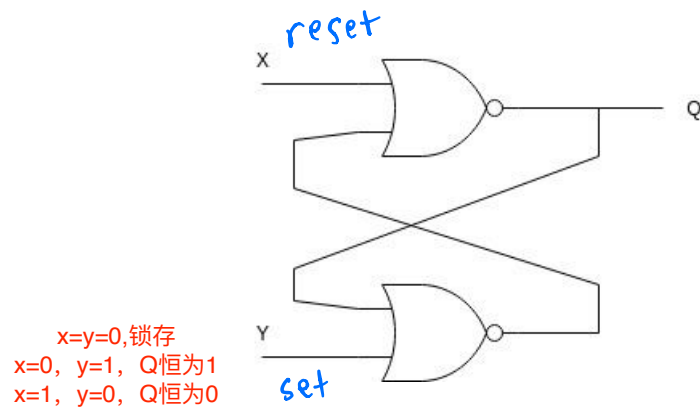


NOT (AC(A+B))

OUT

| A | B | C | OUT |
|---|---|---|-----|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

3

Name:_____

**Problem 2.** (15 points):

**Part a**. (5 points): In class we implemented latches with two NAND gates. We can also do it with two NOR gates, as shown below.



x=y=0,锁存
x=0，y=1，Q恒为1
x=1，y=0，Q恒为0

For what values of X and Y will the latch be in its 'quiescent state' (i.e. the latch will retain whatever value was previously stored in it)?

X: $0$    Y: $0$

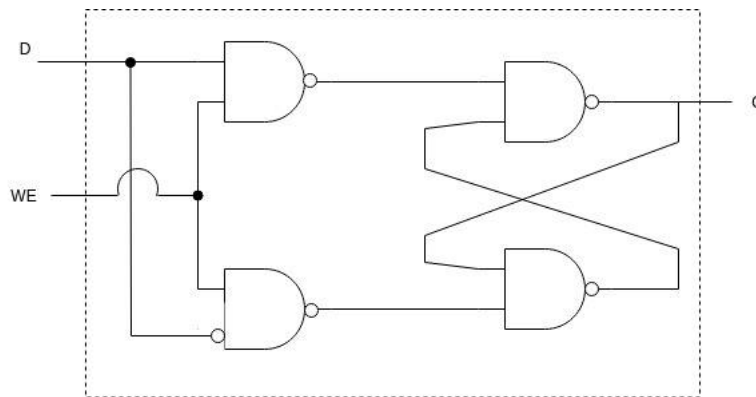What must be done to X and Y in order to store a 1 in the latch?

X: $0$    Y: $1$

What must be done to X and Y in order to store a 0 in the latch?

X: $1$    Y: $0$

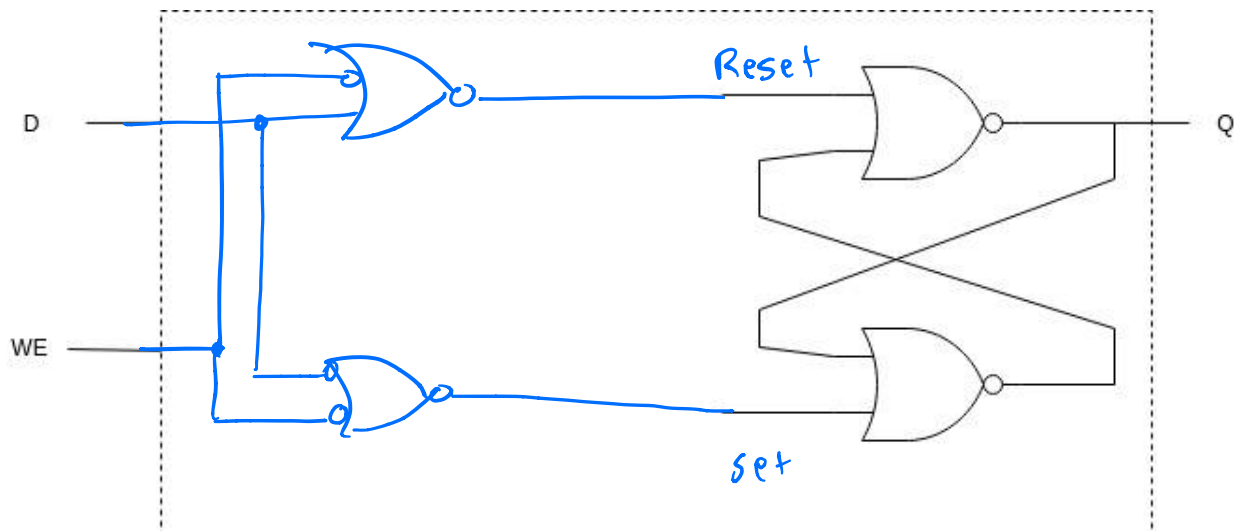**Part b**. (10 points): Below is the gated D latch we discussed in class.



As you can see, this gated D latch is implemented using only NAND and NOT gates. The inputs are D and WE, and the output is Q.

**Your job:** Implement a gated D latch, with the same functionality as the gated D latch shown above, using only NOR and NOT gates. Part of it has been completed for you.



| WE | D | Reset | Set |
|----|---|-------|-----|
| 0 | 0 | D | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$Reset = WE \cdot \bar{D} \Rightarrow \overline{\overline{WE} + D}$

$Set = WE \cdot D \Rightarrow \overline{\overline{WE} + \bar{D}}$

5

**Problem 3.** (15 points):

We want to design a synchronous finite state machine with a single input and a single output. **The output is 1 if the most recent three inputs are the same.**

Recall, outputs are determined solely by the state. Since the state is latched at the end of the clock cycle, the output due to the input in clock cycle $n$ will be present in clock cycle $n + 1$.

Here is an example sequence of inputs and the outputs the sequence causes:

| Cycle | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| Input  | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0  | 1  | 0  | 0  | 0  | 1  | 1  | –  |
| Output | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  |

**Your job:** Complete the synchronous finite state machine. That is, show the output (0 or 1) for every state, and show the input (0 or 1) that takes the machine from its current state to its next state.

We have provided twelve states. You will not need all of them. Use what you need. We have also provided the initial state, shown in bold, where the sequence begins.



6
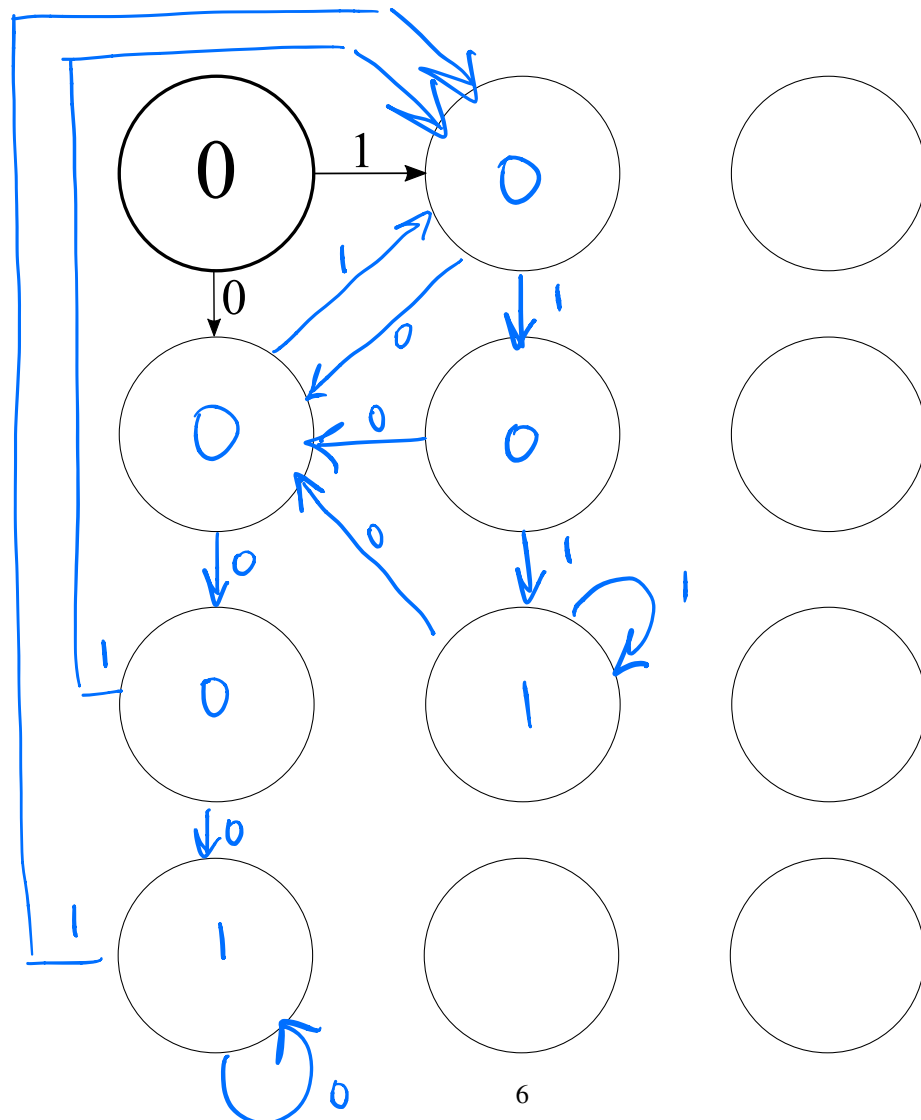
**Problem 4.** (20 points):

The incomplete program shown below starts executing at location x3000.

| Address | Value |
|---------|-------|
| x3000 | 0101 000 000 1 *00000* |
| x3001 | *0001 000 000 1 00101* |
| x3002 | 0001 000 000 1 11111 |
| x3003 | *0000 011* *11111110* |
| x3004 | 1111 0000 0010 0101 |

During the execution of the program, each time an instruction sets condition codes we record the values of those condition codes in the table below. That is, the first row shows the condition codes set by the first instruction in the program that sets condition codes (i.e., the instruction in location x3000). The second row shows the condition codes set by the second instruction in the program that sets condition codes, and so on. If an instruction does not set condition codes, nothing is recorded. The table records the condition codes set by all instructions up to the point just before the instruction in memory location x3004 executes.

*due to*

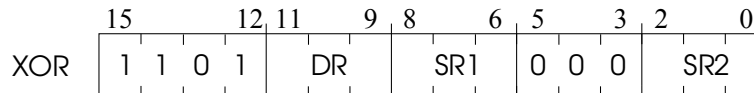| | N | Z | P | |
|-------|---|---|---|------|
| x3000 | 0 | 1 | 0 | |
| x3001 | 0 | 0 | 1 | R0=5 |
| x3002 | 0 | 0 | 1 | R0=4 |
| x3002 | 0 | 0 | 1 | R0=3 |
| x3002 | 0 | 0 | 1 | R0=2 |
| x3002 | 0 | 0 | 1 | R0=1 |
| x3002 | 0 | 1 | 0 | R0=0 ← figure this one out first |
| x3002 | 1 | 0 | 0 | R0=-1 |

**Your job:** Complete the program by filling in the blanks so that the resulting program produces the condition codes shown in the table.

**Problem 5.** (25 points):

The *Hamming distance* of two bit vectors of equal length is the number of bits in which the two bit vectors differ. For example, the Hamming distance of 0110 and 0111 is 1 because they differ in only one bit (the right most bit). The Hamming distance of 11110000 and 10010010 is 3.

We decided to write a program that computes the Hamming distance of two bit vectors. To make life easier for us, we decided to use our unused LC-3 opcode 1101 to form the exclusive-OR (XOR) of two bit vectors. The format of this instruction is shown below.

| | 15 | | 12 | 11 | | 9 | 8 | | 6 | 5 | | 3 | 2 | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| XOR | 1 | 1 | 0 1 | | DR | | | SR1 | | 0 | 0 0 | | SR2 | |

That is, bit $n$ of DR is 1 if bit $n$ of SR1 and bit $n$ of SR2 are not the same.

The program uses the contents of memory locations x3100 and x3101 as the two 16-bit bit vectors, computes their Hamming distance, and stores that Hamming distance in memory location x3055. You will note that the program we wrote is incomplete.

**Your job:** Complete the program by filling in the blanks in the instructions so that the resulting program correctly computes the Hamming distance of the two bit vectors and stores the result in memory location x3055.

| Address | Value | Comments |
|---------|-------|----------|
| x3000 | 0010 010 011111111 | ; R2 ← M[x3100] |
| x3001 | 0010 011 011111111 | ; R3 ← M[x3101] |
| x3002 | 1101 100 010 000 011 | ; XOR |
| x3003 | 0101 000 000 1 00000 | ; R0 ← 0 |
| x3004 | 0101 001 001 1 00000 | ; R1 ← 0 |
| x3005 | 0001 001 001 1 01111 | /////////////////////// |
| x3006 | 0001 100 100 1 00000 | ; R4 ← R4 + 0 |
| x3007 | 0000 011 000000001 | ; Branch to x3009 if Z or P is set |
| x3008 | 0001 000 000 1 00001 | /////////////////////// |
| x3009 | 0001 100 100 000 100 | ; R4 ← R4 + R4 |
| x300A | 0001 001 001 1 11111 | ; R1 ← R1 - 1 |
| x300B | 0000 001 111111010 | /////////////////////// |
| x300C | 0011 000 001001000 | /////////////////////// |
| x300D | 1111 0000 0010 0101 | ; HALT |

8
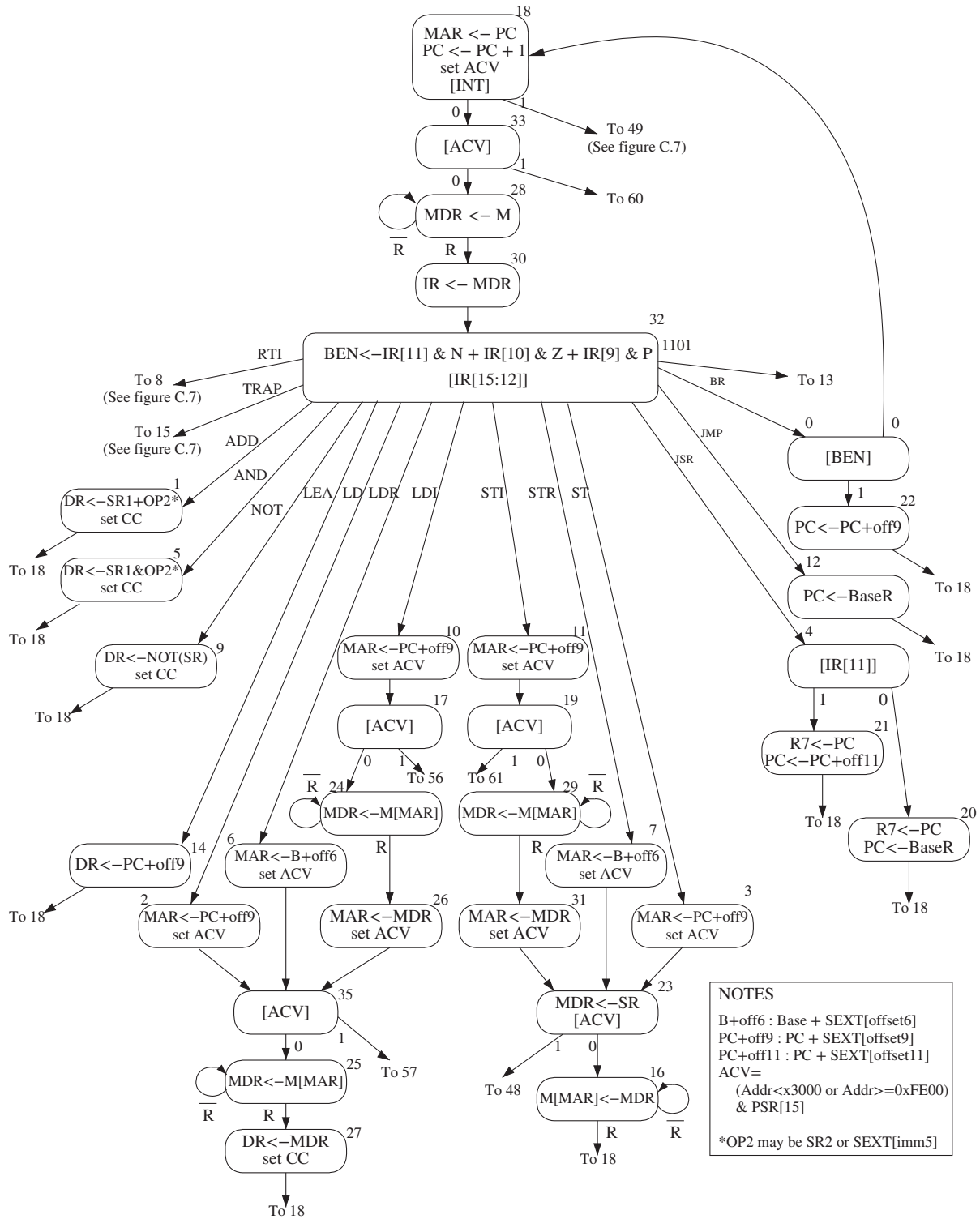
**Figure C.2**   **A state machine for the LC-3.**
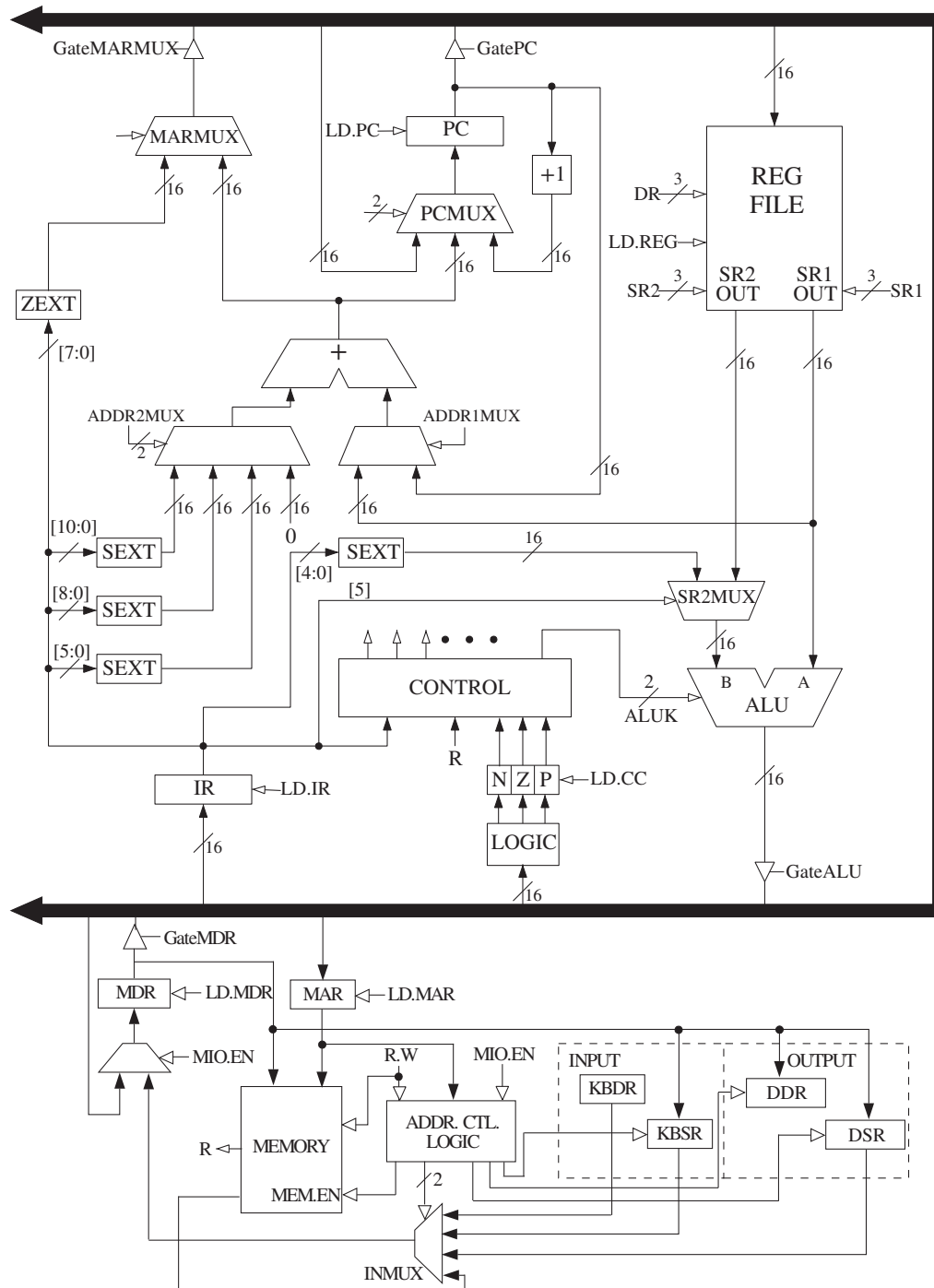
**Figure C.3    The LC-3 data path.**

| | 15 14 13 12 | 11 10 9 | 8 7 6 | 5 | 4 3 | 2 1 0 |
|---|---|---|---|---|---|---|
| ADD+ | 0001 | DR | SR1 | 0 | 00 | SR2 |
| ADD+ | 0001 | DR | SR1 | 1 | imm5 | |
| AND+ | 0101 | DR | SR1 | 0 | 00 | SR2 |
| AND+ | 0101 | DR | SR1 | 1 | imm5 | |
| BR | 0000 | n z p | PCoffset9 | | | |
| JMP | 1100 | 000 | BaseR | 000000 | | |
| JSR | 0100 | 1 | PCoffset11 | | | |
| JSRR | 0100 | 0 00 | BaseR | 000000 | | |
| LD+ | 0010 | DR | PCoffset9 | | | |
| LDI+ | 1010 | DR | PCoffset9 | | | |
| LDR+ | 0110 | DR | BaseR | offset6 | | |
| LEA | 1110 | DR | PCoffset9 | | | |
| NOT+ | 1001 | DR | SR | 111111 | | |
| RET | 1100 | 000 | 111 | 000000 | | |
| RTI | 1000 | 000000000000 | | | | |
| ST | 0011 | SR | PCoffset9 | | | |
| STI | 1011 | SR | PCoffset9 | | | |
| STR | 0111 | SR | BaseR | offset6 | | |
| TRAP | 1111 | 0000 | trapvect8 | | | |
| reserved | 1101 | | | | | |

**Figure A.2**    Format of the entire LC-3 instruction set. *Note:* + indicates instructions that modify condition codes

Table 2.2   ASCII character set

| Oct | Dec | Hex | Char | Oct | Dec | Hex | Char | Oct | Dec | Hex | Char | Oct | Dec | Hex | Char |
|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|
| 000 | 0 | 00 | NUL | 040 | 32 | 20 | SPACE | 100 | 64 | 40 | @ | 140 | 96 | 60 | ` |
| 001 | 1 | 01 | SOH | 041 | 33 | 21 | ! | 101 | 65 | 41 | A | 141 | 97 | 61 | a |
| 002 | 2 | 02 | STX | 042 | 34 | 22 | " | 102 | 66 | 42 | B | 142 | 98 | 62 | b |
| 003 | 3 | 03 | ETX | 043 | 35 | 23 | # | 103 | 67 | 43 | C | 143 | 99 | 63 | c |
| 004 | 4 | 04 | EOT | 044 | 36 | 24 | $ | 104 | 68 | 44 | D | 144 | 100 | 64 | d |
| 005 | 5 | 05 | ENQ | 045 | 37 | 25 | % | 105 | 69 | 45 | E | 145 | 101 | 65 | e |
| 006 | 6 | 06 | ACK | 046 | 38 | 26 | & | 106 | 70 | 46 | F | 146 | 102 | 66 | f |
| 007 | 7 | 07 | BEL | 047 | 39 | 27 | ' | 107 | 71 | 47 | G | 147 | 103 | 67 | g |
| 010 | 8 | 08 | BS | 050 | 40 | 28 | ( | 110 | 72 | 48 | H | 150 | 104 | 68 | h |
| 011 | 9 | 09 | HT | 051 | 41 | 29 | ) | 111 | 73 | 49 | I | 151 | 105 | 69 | i |
| 012 | 10 | 0A | LF | 052 | 42 | 2A | * | 112 | 74 | 4A | J | 152 | 106 | 6A | j |
| 013 | 11 | 0B | VT | 053 | 43 | 2B | + | 113 | 75 | 4B | K | 153 | 107 | 6B | k |
| 014 | 12 | 0C | FF | 054 | 44 | 2C | , | 114 | 76 | 4C | L | 154 | 108 | 6C | l |
| 015 | 13 | 0D | CR | 055 | 45 | 2D | - | 115 | 77 | 4D | M | 155 | 109 | 6D | m |
| 016 | 14 | 0E | SO | 056 | 46 | 2E | . | 116 | 78 | 4E | N | 156 | 110 | 6E | n |
| 017 | 15 | 0F | SI | 057 | 47 | 2F | / | 117 | 79 | 4F | O | 157 | 111 | 6F | o |
| 020 | 16 | 10 | DLE | 060 | 48 | 30 | 0 | 120 | 80 | 50 | P | 160 | 112 | 70 | p |
| 021 | 17 | 11 | DC1 | 061 | 49 | 31 | 1 | 121 | 81 | 51 | Q | 161 | 113 | 71 | q |
| 022 | 18 | 12 | DC2 | 062 | 50 | 32 | 2 | 122 | 82 | 52 | R | 162 | 114 | 72 | r |
| 023 | 19 | 13 | DC3 | 063 | 51 | 33 | 3 | 123 | 83 | 53 | S | 163 | 115 | 73 | s |
| 024 | 20 | 14 | DC4 | 064 | 52 | 34 | 4 | 124 | 84 | 54 | T | 164 | 116 | 74 | t |
| 025 | 21 | 15 | NAK | 065 | 53 | 35 | 5 | 125 | 85 | 55 | U | 165 | 117 | 75 | u |
| 026 | 22 | 16 | SYN | 066 | 54 | 36 | 6 | 126 | 86 | 56 | V | 166 | 118 | 76 | v |
| 027 | 23 | 17 | ETB | 067 | 55 | 37 | 7 | 127 | 87 | 57 | W | 167 | 119 | 77 | w |
| 030 | 24 | 18 | CAN | 070 | 56 | 38 | 8 | 130 | 88 | 58 | X | 170 | 120 | 78 | x |
| 031 | 25 | 19 | EM | 071 | 57 | 39 | 9 | 131 | 89 | 59 | Y | 171 | 121 | 79 | y |
| 032 | 26 | 1A | SUB | 072 | 58 | 3A | : | 132 | 90 | 5A | Z | 172 | 122 | 7A | z |
| 033 | 27 | 1B | ESC | 073 | 59 | 3B | ; | 133 | 91 | 5B | [ | 173 | 123 | 7B | { |
| 034 | 28 | 1C | FS | 074 | 60 | 3C | < | 134 | 92 | 5C | \ | 174 | 124 | 7C | \| |
| 035 | 29 | 1D | GS | 075 | 61 | 3D | = | 135 | 93 | 5D | ] | 175 | 125 | 7D | } |
| 036 | 30 | 1E | RS | 076 | 62 | 3E | > | 136 | 94 | 5E | ^ | 176 | 126 | 7E | ~ |
| 037 | 31 | 1F | US | 077 | 63 | 3F | ? | 137 | 95 | 5F | _ | 177 | 127 | 7F | DEL |