**F.1 Chapter 1 Solutions**

1.1 Every computer can do the same thing as every other computer. A smaller or slower computer will just take longer.

1.3 It is hard to increase the accuracy of analog machines.

* 1. (a) inputs to first (x) box are *a* and *x*

output of first (x) box is *ax*

inputs to second (+) box are *ax* and *b*

output of second (+) box is *ax* + *b*

* + 1. inputs to first (+) box are *w* and *x* output of first (+) box is *w* + *x* inputs to second (+) box are *y* and *z* output of second (+) box is *y* + *z*

inputs to third (+) box are (*w* + *x*) and (*y* + *z*)

output of third (+) box is *w* + *x* + *y* + *z*

inputs to fourth (x) box are (*w* + *x* + *y* + *z*) and .25

output of fourth (x) box is 0*.*25(*w* + *x* + *y* + *z*), which is the average

* + 1. The key is to factor *a*2 + 2*ab* + *b*2 = (*a* + *b*)2

inputs to first (+) box are *a* and *b*

output of first (+) box is *a* + *b*

inputs to second (x) box are (*a* + *b*) and (*a* + *b*)

output of second (x) box is (*a* + *b*)2 = *a*2 + 2*ab* + *b*2

1.7 If the taxi driver is honorable, he/she asks you whether time or money is more important to you, and then gets you to the airport as quickly or as cheaply as possible. You are freed from knowing anything about the various ways one can get to the airport. If the taxi driver is dishonorable, you get to the airport late enough to miss your flight and/or at a taxi fare far in excess of what it should have been, as the taxi driver takes a very circuitous route.

1

1. APPENDIX F. SELECTED SOLUTIONS

1.9 Yes, if phrased in a way that is definite and lacks ambiguity.

1.11 (a) Lacks definiteness: Go south on Main St. for a mile or so.

1. Lacks effective computability: Find the integer that is the square root of 14.
2. Lacks finiteness: Do something. Repeat forever.

1.13 Both computers, A and B, are capable of solving the same problems. Computer B can perform subtraction by taking the negative of the second number and adding it to the first one. As A and B are otherwise identical, they are capable of solving the same problems.

1.15 Advantages of a higher level language: Fewer instructions are required to do the same amount of work. This usually means it takes less time for a programmer to write a program to solve a problem. High level language programs are generally easier to read and therefore know what is going on. Disadvantages of a higher level language: Each instruction has less control over the underlying hardware that actually performs the computation that the program frequently executes less fficiently.

NOTE: this problem is beyond the scope of Chapter 1 or most students.

1.17 An ISA describes the interface to the computer from the perspective of the 0s and 1s of the program. For example, it describes the operations, data types, and addressing modes a programmer can use on that particular computer. It doesn’t specify the actual physical implementation. The microarchitecture does that. Using the car analogy, the ISA is what the driver sees, and the microarchitecture is what goes on under the hood.

1.19 (a) Problem: For example, what is the sum of the ten smallest positive integers.

1. Algorithm: Any procedure is fine as long as it has definiteness, effective computability, and finiteness.
2. Language: For example, C, C++, Fortran, IA-32 Assembly Language.
3. ISA: For example, IA-32, PowerPC, Alpha, SPARC.
4. Microarchitecture: For example, Pentium III, Compaq 21064.
5. Circuits: For example, a circuit to add two numbers together.
6. Devices: For example, CMOS, NMOS, gallium arsenide.

1.21 It is in the ISA of the computer that will run it. We know this because if the word procesing software were in a high- or low-level programming language, then the user would need to compile it or assemble it before using it. This never happens. The user just needs to copy the files to run the program, so it must already be in the correct machine language, or ISA.

1.23 ISA’s don’t change much between successive generations, because of the need for backward compatibility. You’d like your new computer to still run all your old software.

**F.2 Chapter 2 Solutions**

2.1 The answer is 2n

2.3 (a) For 400 students, we need at least 9 bits.

1. 29 = 512, so 112 more students could enter.

2.5 If each number is represented with 5 bits,

7 = 00111 in all three systems

-7 = 11000 (1's complement)

* 10111 (signed magnitude)
* 11001 (2's complement)

2.7 Refer to the following table:

|  |  |
| --- | --- |
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | -8 |
| 1001 | -7 |
| 1010 | -6 |
| 1011 | -5 |
| 1100 | -4 |
| 1101 | -3 |
| 1110 | -2 |
| 1111 | -1 |

2.9 Avogadro's number (6.02 x 1023) requires 80 bits to be represented in two's complement binary representation.

1

2 *APPENDIX F. SELECTED SOLUTIONS*

2.11 (a) 01100110

1. 01000000
2. 00100001
3. 10000000
4. 01111111

2.13 (a) 11111010

1. 00011001
2. 11111000
3. 00000001

2.15 Dividing the number by two.

2.17 (a) 1100 (binary) or -4 (decimal)

1. 01010100 (binary) or 84 (decimal)
2. 0011 (binary) or 3 (decimal)
3. 11 (binary) or -1 (decimal)

2.19 11100101, 1111111111100101, 11111111111111111111111111100101. Sign extension does not affect the value represented.

2.21 Overflow has occurred if both operands are positive and the result is negative, or if both operands are negative and the result is positive.

2.23 Overflow has occurred in an unsigned addition when you get a carry out of the leftmost bits.

2.25 Because their sum will be a number which if positive, will have a lower magnitude (less positive) than the original positive number (because a negative number is being added to it), and vice versa.

2.27 The problem here is that overflow has occurred as adding 2 positive numbers has resulted in a negative number.

2.29 Refer to the following table:

|  |  |  |
| --- | --- | --- |
| X | Y | XANDY |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

2.31 When at least one of the inputs is 1.

|  |  |
| --- | --- |
| *F.2. CHAPTER 2 SOLUTIONS* | 3 |

2.33 (a) 11010111

1. 111
2. 11110100
3. 10111111
4. 1101
5. 1101

2.35 The masks are used to set bits (by ORing a 1) and to clear bits (by ANDing a 0).

2.37 [(n AND m AND (NOT s)) OR ((NOT n) AND (NOT m) AND s)] AND 1000

2.39 (a) 0 10000000 11100000000000000000000

1. 1 10000100 10111010111000000000000
2. 0 10000000 10010010000111111011011
3. 0 10001110 11110100000000000000000

2.41 (a) 127

(b) -126

2.43 (a) Hello!

1. hELLO!
2. Computers!
3. LC-2

2.45 (a) xD1AF

1. x1F
2. x1
3. xEDB2

2.47 (a) -16

1. 2047
2. 22
3. -32768

2.49 (a) x2939

1. x6E36
2. x46F4
3. xF1A8
4. The results must be wrong. In (3), the sum of two negative numbers produced a positive result. In (4), the sum of two positive numbers produced a negative result. We call such additions OVERFLOW.

4 *APPENDIX F. SELECTED SOLUTIONS*

2.51 (a) x644B

1. x4428E800
2. x48656C6C6F

2.53 Refer to the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Q1 | Q2 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

Q2=AORB

2.55 (a) 63

1. 4n – 1
2. 310
3. 222
4. 11011.11
5. 0100 0001 1101 1110 0000 0000 0000 0000
6. 4(4^m)

**F.3 Chapter 3 Solutions**

3.1

|  |  |  |
| --- | --- | --- |
|  | N-Type | P-Type |
| Gate=1 | closed | open |
| Gate=0 | open | closed |

3.3 There can be 16 different two input logic functions.

3.5

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

3.7 There is short circuit (path from Power to Ground) when either A = 1 and B = 0 or A = 0 and B = 1.

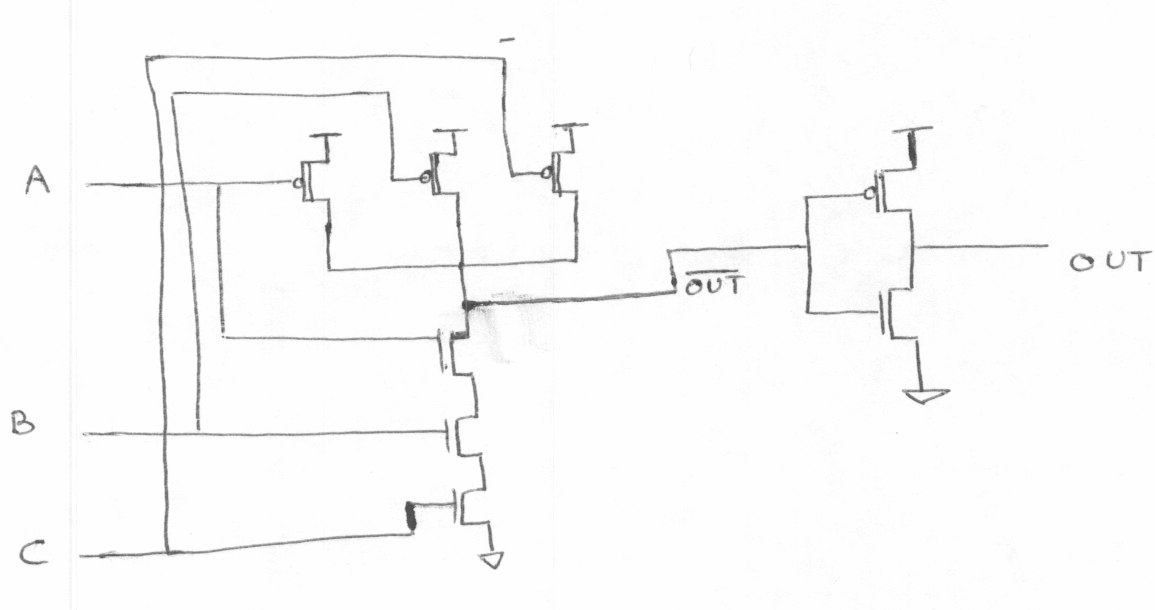
* 1. The circuit is floating when A = 0 and creates a short-circuit when A = 1, which is guaranteed to burn out the power supply.
  2. When A = 0, Out = 3.3 V  
      When A = 1, Out is floating, but power and ground are shorted, which will result in very high current and possible breakdown.
  3. \***NOTE**: This problem was mistakenly included in Chapter 3 when it should belong in Chapter 5. See ERRATA for more information\*  
     OUT = 1 signifies that the instruction is a branch instruction, and the branch will be taken. OUT = 0 means otherwise.

3.15

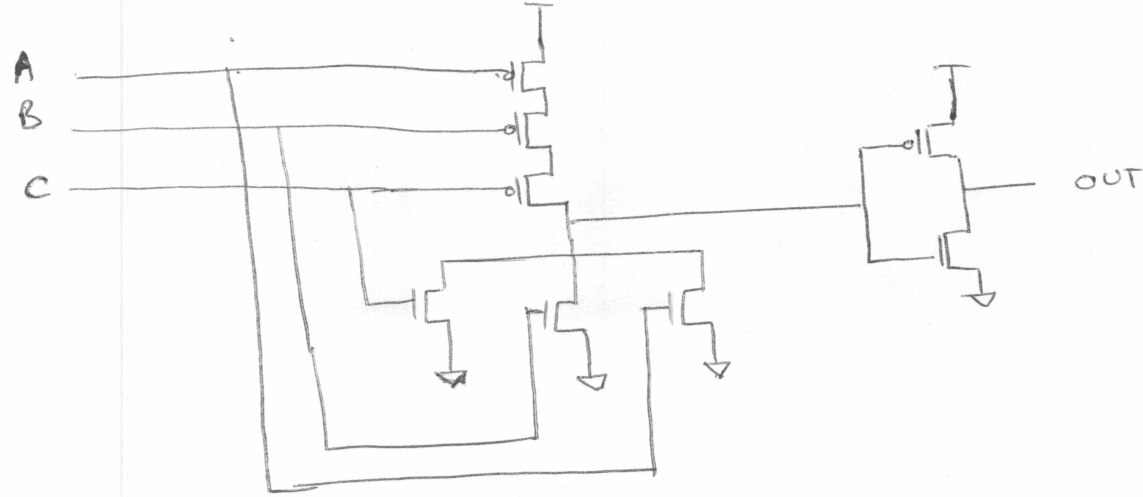
|  |  |  |  |
| --- | --- | --- | --- |
| A | B |  | NOT(NOT(A) OR NOT(B)) |
| 0 | 0 |  | 0 |
| 0 | 1 |  | 0 |
| 1 | 0 |  | 0 |
| 1 | 1 |  | 1 |

AND gate has the same truth table.

3.17 a. Three input And-Gate

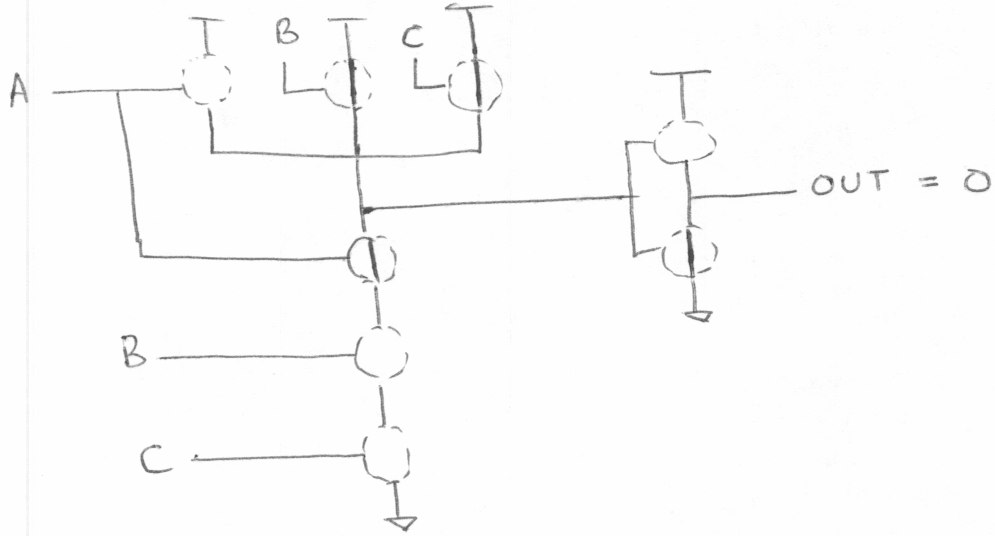


Three input OR-Gate

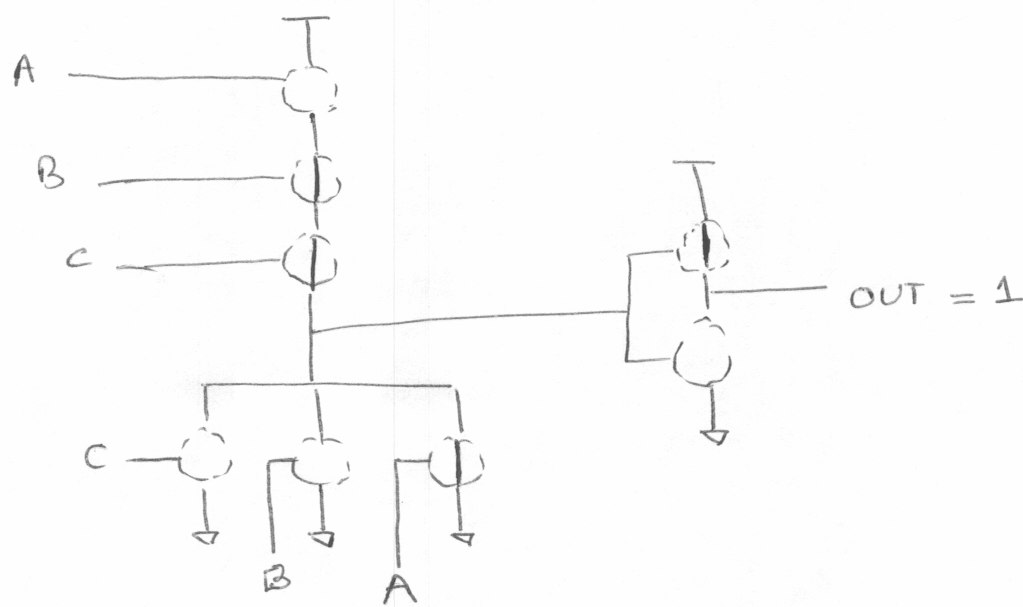


b. (1) A = 1, B = 0, C = 0.

AND Gate

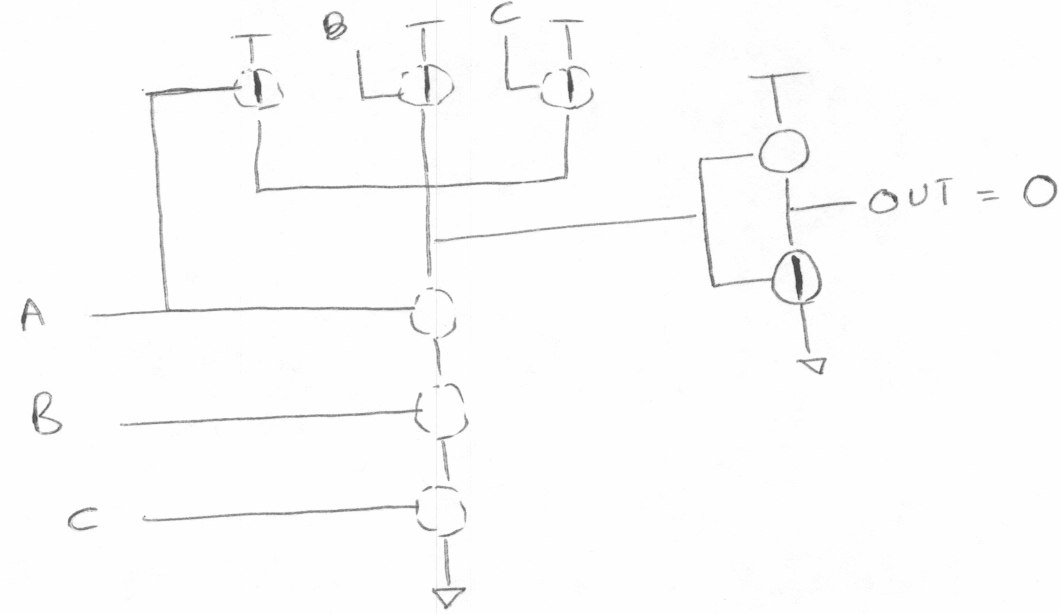


OR Gate

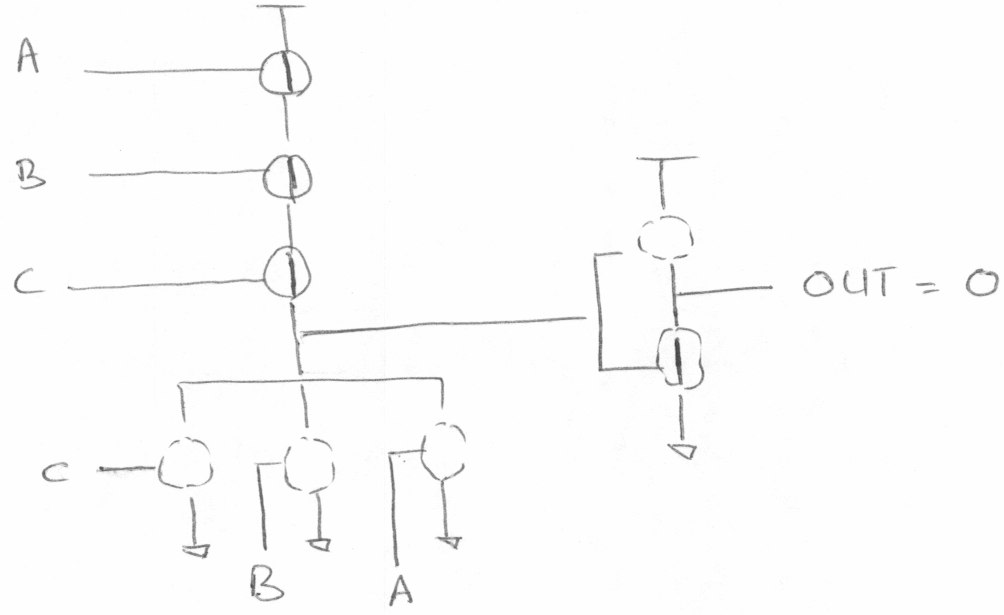


b. (2) A = 0, B = 0, C = 0

AND Gate

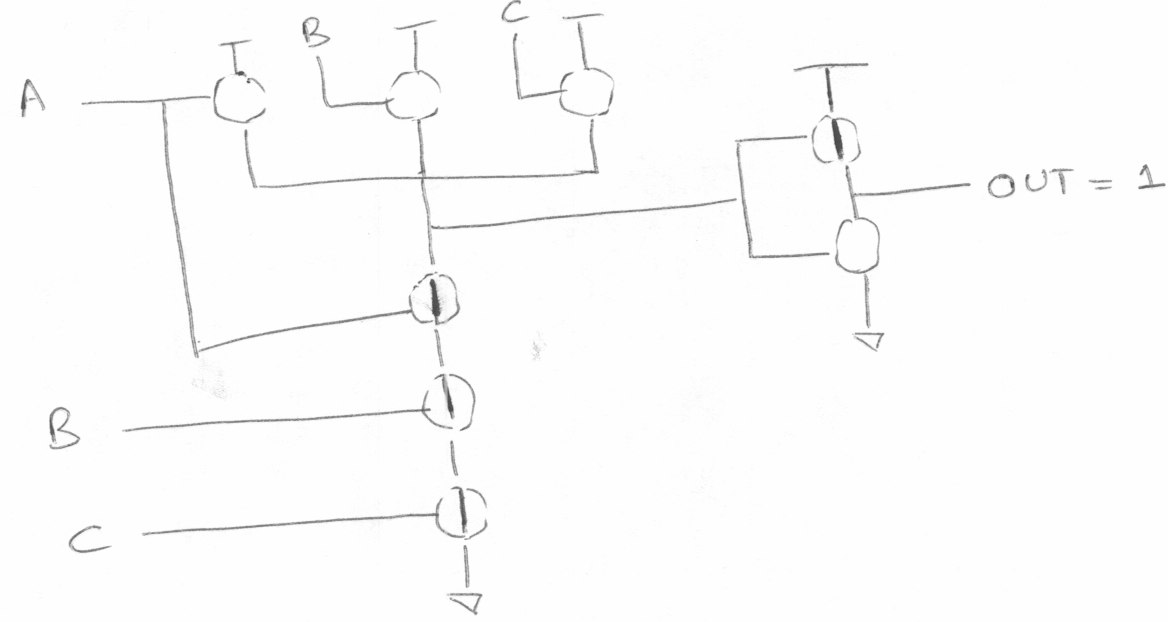


OR Gate

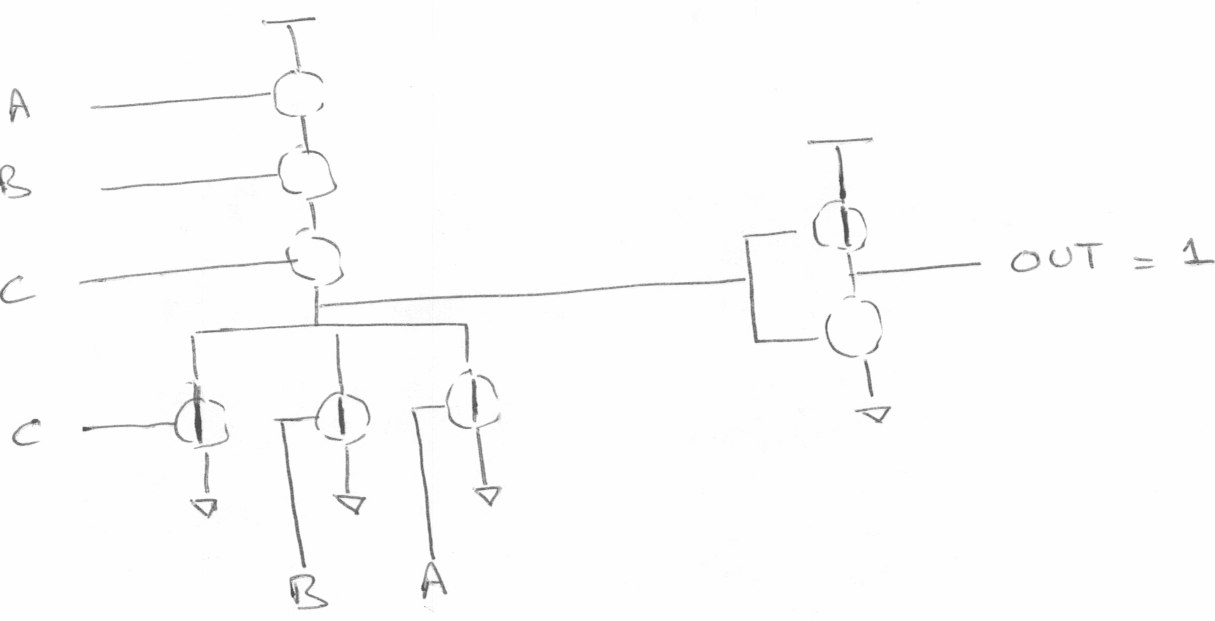


b. (3) A = 1, B = 1, C = 1

AND Gate



OR Gate



3.19 A five input decoder will have 32 output lines.

3.21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cin | 1 | 1 | 1 | 0 |
| A | 0 | 1 | 1 | 1 |
| B | 1 | 0 | 1 | 1 |
| S | 0 | 0 | 1 | 0 |
| Cout | 1 | 1 | 1 | 1 |

A = 7, B = 11, A + B = 18.

In the above calculation, the result (S) is 2 !! This is because 18 is too large a number to be represented in 4 bits. Hence there is an overflow - Cout[3] = 1.

3.23 (a) The truth table will have 16 rows. Here is the truth table for Z = XOR (A, B, C, D). Any circuit with at least seven input combinations generating 1s at the output will work.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D |  | Z |
| 0 | 0 | 0 | 0 |  | 0 |
| 0 | 0 | 0 | 1 |  | 1 |
| 0 | 0 | 1 | 0 |  | 1 |
| 0 | 0 | 1 | 1 |  | 0 |
| 0 | 1 | 0 | 0 |  | 1 |
| 0 | 1 | 0 | 1 |  | 0 |
| 0 | 1 | 1 | 0 |  | 0 |
| 0 | 1 | 1 | 1 |  | 1 |
| 1 | 0 | 0 | 0 |  | 1 |
| 1 | 0 | 0 | 1 |  | 0 |
| 1 | 0 | 1 | 0 |  | 0 |
| 1 | 0 | 1 | 1 |  | 1 |
| 1 | 1 | 0 | 0 |  | 0 |
| 1 | 1 | 0 | 1 |  | 1 |
| 1 | 1 | 1 | 0 |  | 1 |
| 1 | 1 | 1 | 1 |  | 0 |

Z = XOR (A,B,C,D)

(b)

Z

A B C D

3.25 Figure 3.36 is a simple combinational circuit. The output value depends ONLY on the input values as they currently exist. Figure 3.37 is an R-S Latch. This is an example of a logic circuit that can store information. That is, if A, B are both 1, the value of D depends on which of the two (A or B) was 0 most recently.

3.27 2 \* 214 = 215 = 32768 nibbles

3.29

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C |  | Z |
| 0 | 0 | 0 |  | 0 |
| 0 | 0 | 1 |  | 0 |
| 0 | 1 | 0 |  | 0 |
| 0 | 1 | 1 |  | 0 |
| 1 | 0 | 0 |  | 0 |
| 1 | 0 | 1 |  | 0 |
| 1 | 1 | 0 |  | 0 |
| 1 | 1 | 1 |  | 0 |

3.31 (a) 3 gate delays

3.31 (b) 3 gate delays

3.31 (c) 3\*4 = 12 gate delays

3.31 (d) 3\*32 = 96 gate delays

3.33(a) When S=0, Z = A

3.33(b) When S=1, Z retains its previous value. 3.33(c) Yes; the circuit is a storage element.

3.35 No. The original value cannot be recovered once a new value is written into a register.

3.37. 8 \* (2^3) = 64 bytes

3.39.(a) To read the 4th memory location, A[1,0] = 11, WE = 0

3.39.(b) A total of 6 address lines are required for a memory with 60 locations. The addressability of the memory will remain unchanged.

3.39.(c) A program counter of width 6 can address 2^6 = 64 locations. So without changing the width of the program counter, 64-60 = 4 more locations can be added to the memory.

3.41 Total bits of storage = 2^22 \* 3 = 12582912

3.43 There are a total of four possible states in this lock. Any other state can be expressed as one of states A, B, C or D. For example, the state performed one correct followed by one incorrect operation is nothing but state A as the incorrect operation reset the lock.

* 1. No. An arc is needed between the two states.
     1. Game in Progress:

Texas \* Oklahoma

Fouls:4 Fouls: 4

73 68

First Half 7:38

Shot Clock : 14

* + 1. Texas Win:

Texas \* Oklahoma

Fouls:10 Fouls: 10

85 70

Second Half 0:00

Shot Clock : 0

* + 1. Oklahoma Win:

Texas \* Oklahoma

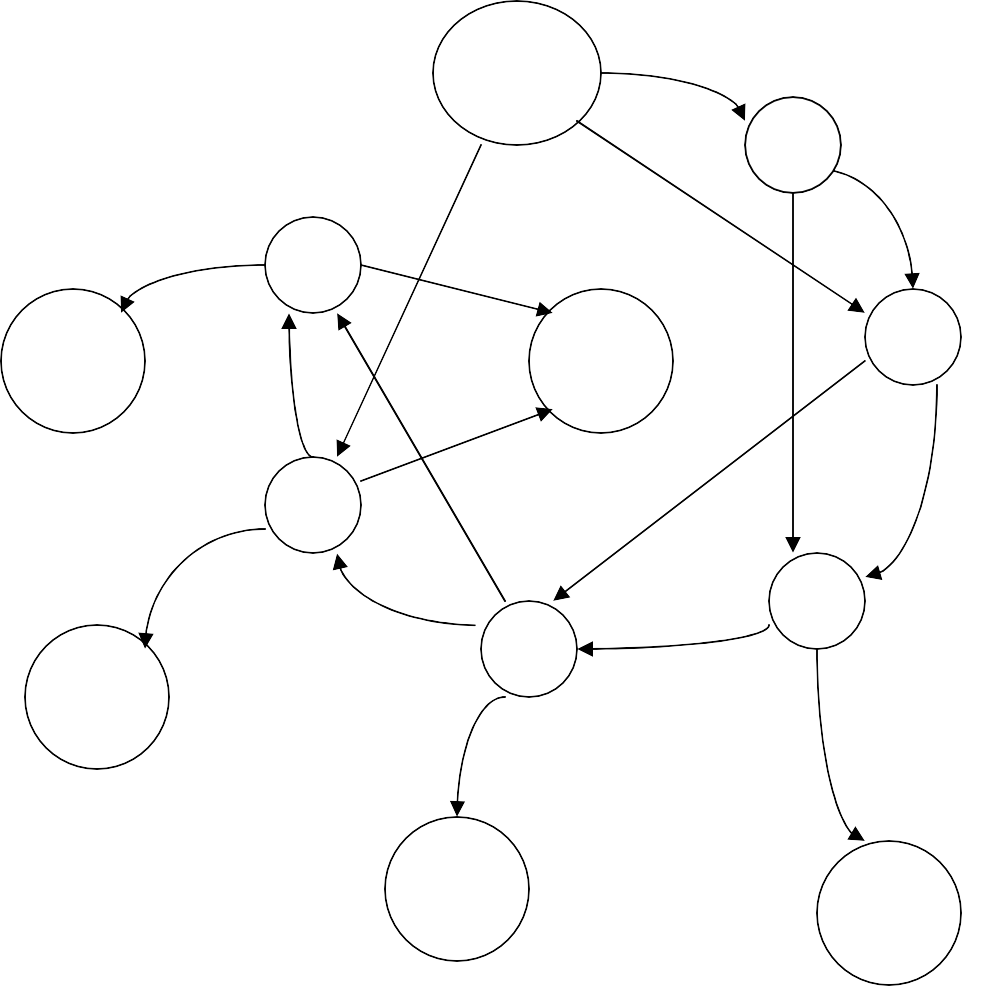
Fouls:10 Fouls: 10

81 90

First Half 7:38

Shot Clock : 0

3.47



No coins

N

Q

5

D/Q

Q

N

30

N

D

D

35+

ch

N

D

35

10

Q

D

Q

25

N

D

D

15

35+

ch

N

20

N

Q Q

35+

ch

35+

ch

3.49

a)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S1 | S0 | X |  | D1 | D0 | Z |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 0 | 1 |  | 0 | 0 | 0 |
| 0 | 1 | 0 |  | 0 | 0 | 1 |
| 0 | 1 | 1 |  | 1 | 0 | 1 |
| 1 | 0 | 0 |  | 1 | 1 | 1 |
| 1 | 0 | 1 |  | 1 | 1 | 1 |
| 1 | 1 | 0 |  | 1 | 0 | 1 |
| 1 | 1 | 1 |  | 1 | 0 | 1 |

b)

0/1

0

00

01

0/1

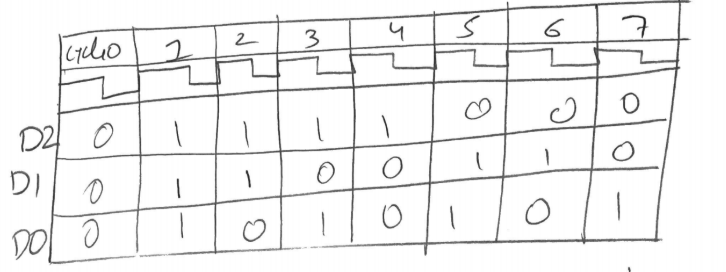
1

11

10

0/1

3.51 Flip-Flop

3.53   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 The circuit is a decrementing 3-bit counter.

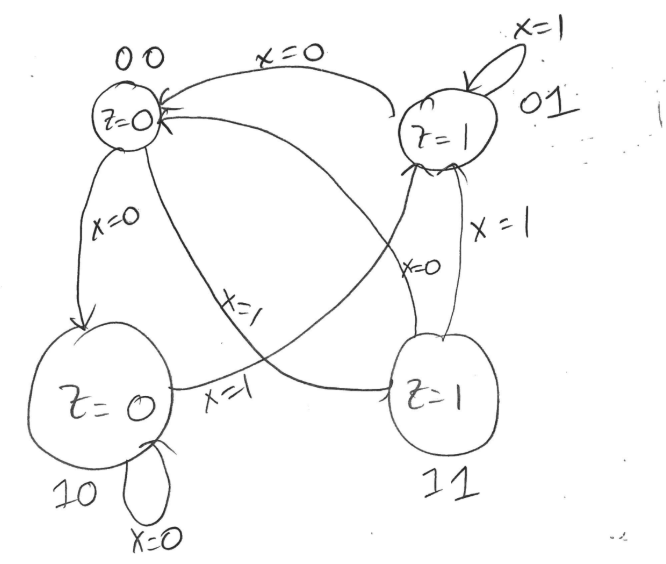
A close up of text on a whiteboard

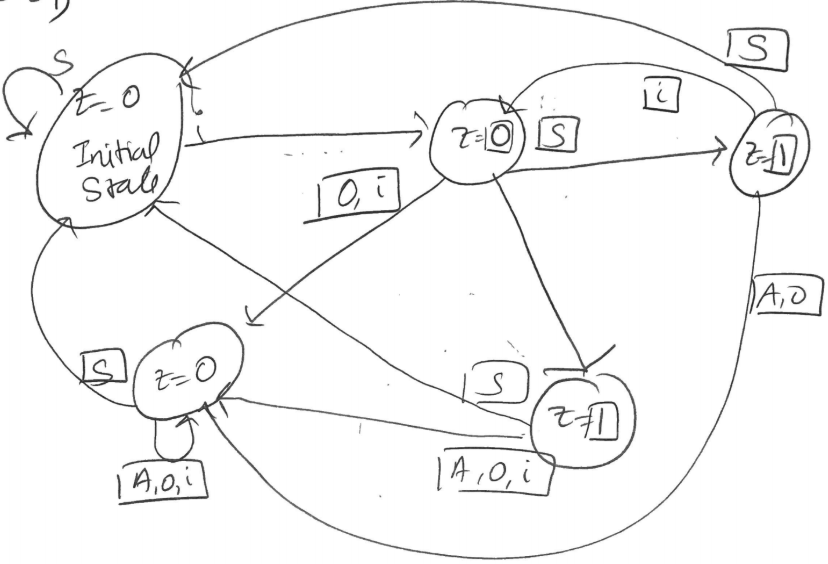
Description automatically generated3.55

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c |  | Y | Z |
| 0 | 0 | 0 |  | 0 | 0 |
| 0 | 0 | 1 |  | 1 | 0 |
| 0 | 1 | 0 |  | 1 | 1 |
| 0 | 1 | 1 |  | 0 | 1 |
| 1 | 0 | 0 |  | 1 | 1 |
| 1 | 0 | 1 |  | 0 | 1 |
| 1 | 1 | 0 |  | 0 | 0 |
| 1 | 1 | 1 |  | 1 | 0 |

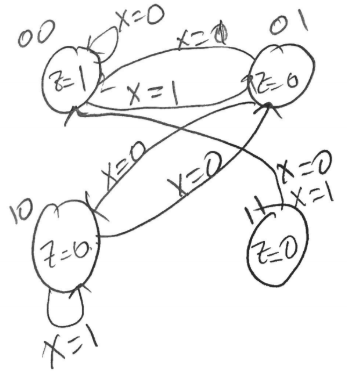
3.57





3.59

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S1 | S0 | X |  | Z | S1’ | S0’ |
| 0 | 0 | 0 |  | 1 | 0 | 0 |
| 0 | 0 | 1 |  | 1 | 0 | 1 |
| 0 | 1 | 0 |  | 0 | 1 | 0 |
| 0 | 1 | 1 |  | 0 | 0 | 0 |
| 1 | 0 | 0 |  | 0 | 0 | 1 |
| 1 | 0 | 1 |  | 0 | 1 | 0 |
| 1 | 1 | 0 |  | 0 | 0 | 0 |
| 1 | 1 | 1 |  | 0 | 0 | 0 |

3.61

**F.4 Chapter 4 Solutions**

* 1. Components of the Von Neumann Model:
     1. Memory: Storage of information (data/program)
     2. Processing Unit: Computation/Processing of Information
     3. Input: Means of getting information into the computer. e.g. keyboard, mouse
     4. Output: Means of getting information out of the computer. e.g. printer, monitor
     5. Control Unit: Makes sure that all the other parts perform their tasks correctly and at the correct time.

4.3 The program counter does not maintain a count of any sort. The value stored in the program counter is the address of the next instruction to be processed. Hence the name ’Instruction Pointer’is more appropriate for it.

4.5 (a) Location 3 contains 0000 0000 0000 0000

Location 6 contains 1111 1110 1101 0011

1. i. Two’s Complement -

Location 0: 0001 1110 0100 0011 = 7747

Location 1: 1111 0000 0010 0101 = -4059

ii. ASCII - Location 4: 0000 0000 0110 0101 = 101 = ’e’

* 1. Floating Point -

Locations 6 and 7: 0000 0110 1101 1001 1111 1110 1101 0011

Number represented is 1.10110011111111011010011 x 2−114

* 1. Unsigned -

Location 0: 0001 1110 0100 0011 = 7747

Location 1: 1111 0000 0010 0101 = 61477

1. Instruction - Location 0: 0001 1110 0100 0011 = Add R7 R1 R3
2. Memory Address - Location 5: 0000 0000 0000 0110 Refers to location 6. Value stored in location 6 is 1111 1110 1101 0011

4.7 60 opcodes = 6 bits 32 registers = 5 bits

So number of bits required for IMM = 32 - 6 - 5 - 5 = 16

Since IMM is a 2’s complement value, its range is -215 ... (215 -1) = -32768 .. 32767.

4.9 The second important operation performed during the FETCH phase is the loading of the address of the next instruction into the program counter.

* 1. The phases of the instruction cycle are:
     1. Fetch: Get instruction from memory. Load address of next instruction in the Program Counter.
     2. Decode: Find out what the instruction does.
     3. Evaluate Address: Calculate address of the memory location that is needed to process the instruction.
     4. Fetch Operands: Get the source operands (either from memory or register file).
     5. Execute: Perform the execution of the instruction.
     6. Store Result: Store the result of the execution to the specified destination.

4.13 F D EA FO E SR

x86: ADD [eax] edx 100 1 1 100 1 100 = 303

LC3: ADD R6, R2, R6 100 1 - 1 1 1 = 104

4.15 Once the RUN latch is cleared, the clock stops, so no instructions can be processed. Thus, no instruction can be used to set the RUN latch. In order to re-initiate the instruction cycle, an external input must be applied. This can be in the form of an interrupt signal or a front panel switch, for example.

4.17

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R/W** | **MAR** | **MDR** |
| **Operation 1** | W | x4000 | 1 1 1 1 0 |
| **Operation 2** | R | x4003 | 1 0 1 1 0 |
| **Operation 3** | W | x4001 | 1 0 1 1 0 |
| **Operation 4** | R | x4002 | 0 1 1 0 1 |
| **Operation 5** | W | x4003 | 0 1 1 0 1 |

Before Access 1 After Access 3

|  |  |
| --- | --- |
| X4000 | 0 1 1 0 1 |
| X4001 | 1 1 0 1 0 |
| X4002 | 0 1 1 0 1 |
| X4003 | 1 0 1 1 0 |
| X4004 | 1 1 1 1 0 |

|  |  |
| --- | --- |
| X4000 | 1 1 1 1 0 |
| X4001 | 1 0 1 1 0 |
| X4002 | 0 1 1 0 1 |
| X4003 | 1 0 1 1 0 |
| X4004 | 1 1 1 1 0 |

After Access 5

|  |  |
| --- | --- |
| X4000 | 0 1 1 0 1 |
| X4001 | 1 1 0 1 0 |
| X4002 | 0 1 1 0 1 |
| X4003 | 0 1 1 0 1 |
| X4004 | 1 1 1 1 0 |

4.19 (a) MAR: x2 MDR: 01010000  
 (b) MDR: 00111001