|  |  |  |  |
| --- | --- | --- | --- |
| Instructor | ***Luke Papademas*** | Due Date | **6/23** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | **1** | **2** | **3** | **4** | Total |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) 3 in your Computer Architecture and Organization textbook.

**Part 1 Glossary Terms - Boolean Algebra and Digital Logic**

Define, in detail, each of these glossary terms from the realm of computer architecture and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes

or course textbook(s) as references or the Internet by visiting Web sites such as:

[**http://www.ask.com**](http://www.ask.com) or [**http://www.webopedia.com**](http://www.webopedia.com/)

**(a) Absorption Laws**

|  |
| --- |
| The absorption law is written as x(x + y) = x in its AND form. The rules of the absortion law are that the term outside the parentheses must be the same inside, and the operators must be different. In the AND for, it can be interpreted as X or Y, and X = X. |

**(b) Boolean Identity**

|  |
| --- |
| Boolean identities are laws that will always hold true in Boolean algebra. The laws include the Identity Law, Null (or Dominance Law), Idempotent Law, Inverse Law, Commutative Law, Associative Law, Disruptive Law, Absorption Law, DeMorgan’s Law, and Double Complement Law. Identities allow us to reduce Boolean expressions to their simplest forms. |

**(c) Combinational Circuits**

|  |
| --- |
| A combinational circuit is one in which the output is always based on the given inputs. A combinational circuit may have several options. If so, each output represents a different Boolean function. Examples of combinational circuits include:   * half-adder - which can only add two bits together * full-adder – can add three bits and is composed of two half-adders and an OR gate * ripple-carry adder – a circuit that consists of 16 full-adders and feeds the carry out of one circuit into the carry in of the next circuit * multiplexer – selects binary information from one of many input lines and directs it to a single output line * parity generator – circuit that creates the necessary parity bit to add to a word * parity checker – checks to make sure proper parity is present in the word |

**(d) Digital Logic**

|  |
| --- |
| Digital logic is logic based on the values 1 and 0, referred to as True and False. This is useful in computing because it is based on the status of an electrical switch, on or off. This logic is used in Boolean algebra to determine the result of a function, commonly performed through a truth table. In addition, digital logic is seen in logic gates of digital circuits. |

**(e) Finite State Machines**

|  |
| --- |
| A finite state machine is a machine with only a fixed number of states, and the output and next state are functions of the input and present state. Fixed state machines can only be in one state at a time.  Examples include:   * Moore machine – the outputs depend only on the present state * Mealy machine – the outputs are a function of its current state and its input, and its output is a function only of its current state |

**Part 2 Exercises - Boolean Algebra and Digital Logic**

For each of the following, select the correct answer.

**(1)** If input *x* is true and if input *y* is true, then *x* · *y* is a \_\_\_\_\_\_\_\_\_\_ .

(a) true statement ( 1 ) (b) false statement ( 0 )

**(2)** If *x* has a setting of 1 and *y* has a setting of 0 , then which of these will be false ( 0 ) ?

(a) *y '* *'* (b) *y '*  (c) *x '* + *y* (d) *x* ·  *y '* (e) *x* +  *y*

**(3)** If *x* is a false statement and *y* is a true statement, then which of these will be true?

(a) *y '* (b) *x* · *y '* (c) *x* + *y '* (d) *x* ·  *y* (e) *x* +  *y*

**(4)** If *x* and *y* are true and *z* is false, find the truth value of: ( *z '* + *y* ) · *x*

(a) true statement (b) false statement

**(5)** If *x* is true and *y* and *z* are false, find the truth value of: ( *x '* · *z* ) + ( *z* · *y '* )

(a) true statement (b) false statement

**Part 3 Exercises - Boolean Algebra and Digital Logic**

Write a complete answer for each of these.

**(1)** Consider the truth table shown below. ( T represents true and F represents false )

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| *x* | *y* | *x '* | *x '* · *y* |
|  |  |  |  |
| T | T | F | F |
| T | F | F | F |
| F | T | T | T |
| F | F | T | F |

What are the missing entries in the rightmost column?

(a) T (b) F (c) F (d) F (e) F

F F F T F

F T F F T

T T T T F

**(2)** Consider the truth table shown below. ( 1 represents true and 0 represents false )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| *x* | *y* | *x '* | *y '* | *x '* + *y '* | *x '*  + ( *x '* + *y '* ) |
|  |  |  |  |  |  |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |

What are the missing entries in the rightmost column?

(a) 1 (b) 0 (c) 0 (d) 0 (e) 0

0 1 1 1 0

0 1 1 0 1

1 1 0 1 0

**(3)** The Boolean \_\_\_\_\_ function can be implemented with two switches, A and B . If a power lead is connected to switch A , and a wire connects switches A and B , then both A and B must be " on " in order for the output of the circuit to conduct electricity and provide power.

(a) NOT (b) AND (c) OR (d) NOR (e) NAND

**(4)** The Boolean \_\_\_\_\_ gate can be constructed from two switches, arranged so that if either switch is " on ", the output will also be " on ". Moreover, the output will still be on even if both switches are " on ".

(a) NOT (b) AND (c) OR (d) NOR (e) NAND

**(5)** An XOR ( exclusive OR ) logic statement is false when both of its logic conditions are true. Note: an XOR statement can be simulated as: ( *x* · *y* )*'* · ( *x* + *y* )

(a) True (b) False

**(6)** A NAND ( i.e. NOT AND ) logic statement is true when one of its logic conditions is false.

(a) True (b) False

**Part 4 Exercises - Boolean Algebra and Digital Logic**

Write a complete answer for each of these.

**(1)** Using a Web site such as [**http://www.wolframalpha.com**](http://www.wolframalpha.com) construct a truth table for the logical expression given below. As an example of using WolframAlpha to construct a truth table, try this sample command:

truth table (NOT( x ) AND y) OR (x OR NOT( y ))

*x* ( *y '*  + *z* ) + *x y z*

|  |  |  |
| --- | --- | --- |
| x | y | x(y’ + z) + xyz |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | T |

**(2)** Using Boolean identities, show that the following is a valid logical equivalence.

*x* = *x* *y* + *x* *y '*

Inverse Law x = x + x’

**(3)** If *x* is set to 1 and *y* and *z* are both set to 0 , determine the result or truth value of this Boolean function.

*F* ( *x* , *y* , *z* ) = *x* *y ' z '*  + *y z* + *x y z*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | z | y’ | z’ | xy’z’ | yz | xyz | xy’z’+yz+xyz |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

**(4)** According to De Morgan’s Principles, ( *x* · *y* )*'* is logically equivalent to *x '* + *y '* . Therefore, the logic expression *x* · *y '* is equivalent to which of these?

(a) *x '* · *y '*  (b) *x* + *y '* (c) *x '* + *y* (d) ( *x* · *y* ) (e) ( *x '* + *y* ) *'*

**(5)** According to De Morgan’s Principles, ( *x* + *y* ) *'* is logically equivalent to *x '* · *y '* . Therefore, the logic expression ( *x '* · *y* ) *'* is equivalent to which of these?

(a) *x '* + *y '*  (b) *x* · *y '* (c) *x '* · *y* (d) *x* + *y '* (e) ( *x '* · *y* ) *'*