BASIC ASSEMBLY

Introduction to Boolean Algebra

Assembly language programming
By xorpd

xorpd.net

OBJECTIVES

- We will use bits to represent the truthfulness of statements.
- We will learn about truthfulness of basic and complicated statements.
- We will study the representation of complicated statements using operators like NOT,AND,OR,XOR.
- We will learn how to calculate the truthfulness of statements in a mechanical way.

BASIC STATEMENTS

- We consider statements and their truthfulness:
  - Those statements could be for example: "3 > 2" or "There is a triangle with 4 vertices".
  - Those statements could be either True or False.

- Intuitively, we can combine different statements into new statements.
  - "3 > 2" is True.
  - "There is a triangle with 4 vertices" is False.
  - {"3 > 2" OR "There is a triangle with 4 vertices"} is True.
  - {"3 > 2" AND "There is a triangle with 4 vertices"} is False.
  - {NOT "3 > 2"} is False.

- It seems like every basic statement is either True or False.
- We can calculate the truthfulness of combined statements using the values of the basic statements.

SIMPLIFIED NOTATION

- Instead of writing the whole statement every time, we can mark it with some English letter.
- Some other shortcut representations:
  - True 1
  - False 0
  - AND $\land$
  - OR $\lor$
  - NOT $\neg$
- If we have two statements a and b, we could represent:
  - "a OR b" as $a \lor b$.
  - "a AND b" as $a \land b$.
  - "NOT a" as $\neg a$

CALCULATION RULES

- NOT ($\neg$)
  - Operates on one bit. (Unary operation).
  - "Flips" the truthfulness of a statement.
    - Only True if the original statement is **NOT** True.
  - $\neg 0 = 1$, $\neg 1 = 0$.
- AND ($\wedge$)
  - Operates on two bits (Binary operation)
  - Results in True (1) if the first argument is True **AND** the second argument is True.
- OR ($\vee$)
  - Operates on two bits (Binary operation)
  - Results in True (1) if the first argument is True **OR** the second argument is True.

TRUTH TABLES

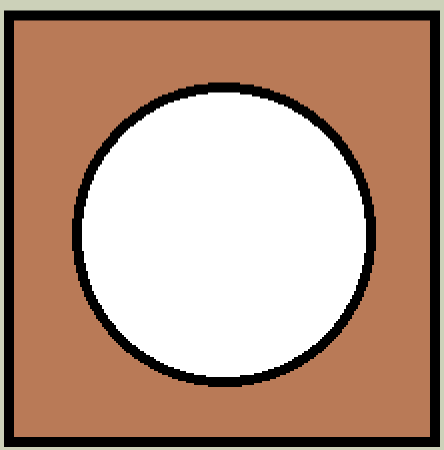
- Truth tables:

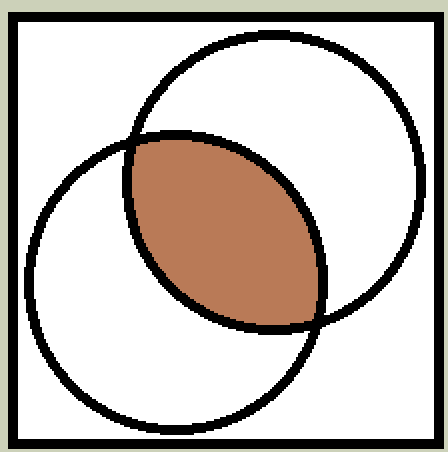
NOT	
0	1
1	0

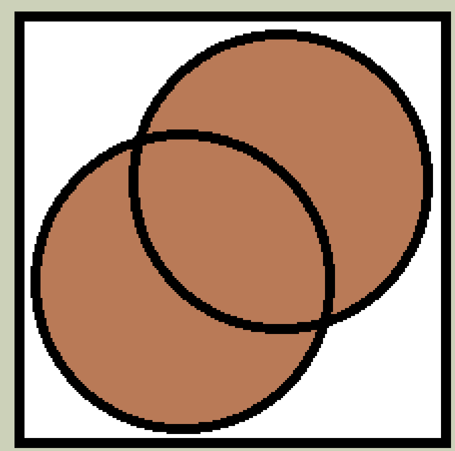
AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

- Venn Diagrams:







EXAMPLES

- Calculation examples:
  - $1 \wedge 0 = 0$
  - $\neg(1 \wedge 0) = 1$
  - $(1 \wedge 0) \vee 0 = 0$
  - $\neg\big((1 \wedge 0) \vee 0\big) = 1$
- **Creation of new operators:**
  - $f(a, b) = a \wedge (\neg b)$
  - $g(a, b, c) = (a \wedge b) \vee (a \wedge c)$

$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{f(a,b)}$
0	0	0
0	1	0
1	0	1
1	1	0

BASIC PROPERTIES

- Double Negation:
  - $\neg(\neg a) = a$
  - "Two wrongs make a right".
- Commutative laws:
  - $a \vee b = b \vee a$
  - $a \wedge b = b \wedge a$.
  - **Like** $5 \cdot 7 = 7 \cdot 5$
- Associative laws:
  - $a \vee (b \vee c) = (a \vee b) \vee c$
  - $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
  - **Like** $2 + (4 + 3) = (2 + 4) + 3$

BASIC PROPERTIES (CONT.)

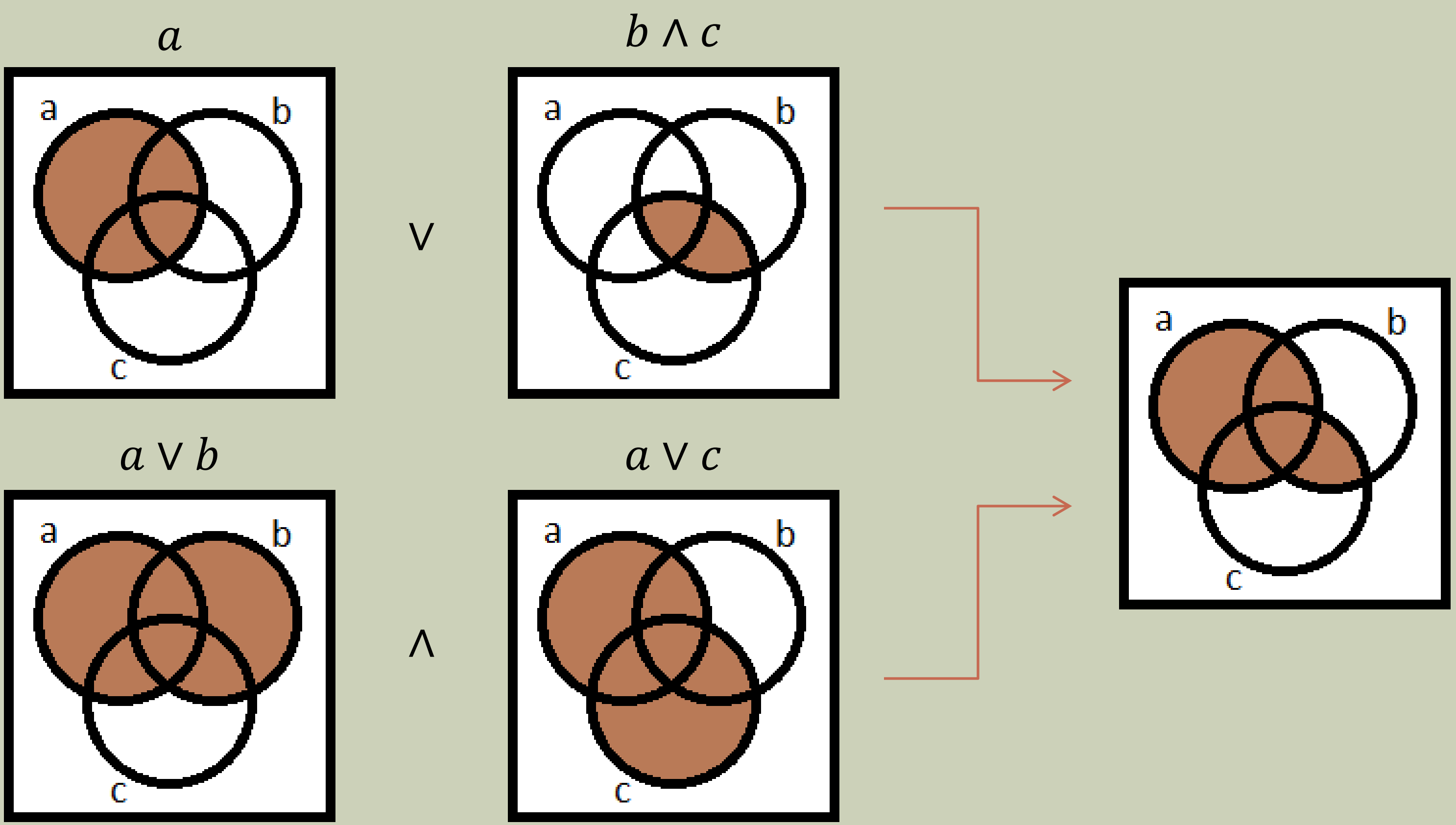
- Distributive laws:
  - $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
  - $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
  - **Like** $3 \cdot (2 + 5) = (3 \cdot 2) + (3 \cdot 5)$

Truth table style proof:

a	b	c	$a \vee (b \wedge c)$	$(a \vee b) \wedge (a \vee c)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

BASIC PROPERTIES (CONT.)

- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
- **Demonstration with Venn Diagrams:**



BREAK

- Take a break.
- Come back when you are ready to learn about De Morgan Laws :)

DE MORGAN'S LAWS

- Laws about the duality of AND and OR.
- Allows to represent some operator using other operators:
- Representing $\wedge$ using $\neg$ and $\vee$:
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$
- Representing $\vee$ using $\neg$ and $\wedge$:
  - $a \vee b = \neg\big((\neg a) \wedge (\neg b)\big)$

a	**b**	$\boldsymbol{a \wedge b}$	$\neg((\neg a) \vee (\neg b))$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

DE MORGAN'S LAWS (CONT.)

- AND and OR are dual.
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$

Regular world

a, b

Inverted bits world

DE MORGAN'S LAWS (CONT.)

- AND and OR are dual.
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$

DE MORGAN'S LAWS (CONT.)

- AND and OR are dual.
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$

Regular world

$$\begin{tikzcd}
a, b \arrow[d, "\neg"'] & \\
\neg a, \neg b \arrow[r, "\vee"'] & (\neg a) \vee (\neg b)
\end{tikzcd}$$

Inverted bits world

DE MORGAN'S LAWS (CONT.)

- AND and OR are dual.
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$

Regular world

$$\begin{tikzcd}
a, b \arrow[d, "\neg"'] & & \neg\big((\neg a) \vee (\neg b)\big) = a \wedge b \\
\neg a, \neg b \arrow[rr, "\vee"'] & & (\neg a) \vee (\neg b) \arrow[u, "\neg"']
\end{tikzcd}$$

Inverted bits world

DE MORGAN'S LAWS (CONT.)

- AND and OR are dual.
  - $a \wedge b = \neg((\neg a) \vee (\neg b))$

Regular world

$$\begin{tikzcd}
a, b \arrow[r, "\wedge"] \arrow[d, "\neg"'] & \neg\big((\neg a) \vee (\neg b)\big) = a \wedge b \\
\neg a, \neg b \arrow[r, "\vee"'] & (\neg a) \vee (\neg b) \arrow[u, "\neg"']
\end{tikzcd}$$

Inverted bits world

SIMPLIFYING STATEMENTS

- **Could we write the following in a simpler form?**

- $(a \wedge b) \vee b = ?$
  - **If** $b = 0$**, the result is** 0**.**
  - **If** $b = 1$**, the result is** 1**.**
  - $(a \wedge b) \vee b = b$

- $(a \wedge b) \vee \big(a \wedge (\neg b)\big) = ?$
  - $(a \wedge b) \vee \big(a \wedge (\neg b)\big) =^{dist} a \wedge \big(b \vee (\neg b)\big) = a \wedge 1 = a$

XOR OPERATOR

- XOR – Exclusive OR.
  - {a XOR b}=1 if a=1 **OR** b=1 but **not both.**
- Truth tables:

XOR	0	1
0	0	1
1	1	0

OR	0	1
0	0	1
1	1	1

- Marked by $a \oplus b$.
- $a \oplus b = (a \vee b) \wedge (\neg(a \wedge b))$
  - Equals 1 only if $a \vee b$ and not $a \wedge b$.

XOR PROPERTIES

- Equivalent to addition modulo 2.
  - (Even + Odd = Odd; Odd + Odd = Even, etc.)
- Bit Flipping:
  - $a \oplus 0 = a$
  - $a \oplus 1 = \neg a$
- Self Xoring:
  - $a \oplus a = 0$
- Commutative:
  - $a \oplus b = b \oplus a$
- Associative:
  - $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

XOR	0	1
0	0	1
1	1	0

XOR PROPERTIES (CONT.)

XOR	0	1
0	0	1
1	1	0

AND	0	1
0	0	0
1	0	1

- Distributive with AND:
  - $a \wedge (b \oplus c) = (a \wedge b) \oplus (a \wedge c)$
  - AND behaves like multiplication, and XOR like addition, modulo 2.
  - $a \cdot (b + c) = a \cdot b + a \cdot c$

ONLY AN INTRODUCTION

- This is only a very short introduction.
- There is so much more to learn about Boolean Algebra and about bits.
- Further subjects to research:
  - Mathematical Logic.
  - Circuit complexity.
  - Coding Theory.

SUMMARY

- Basic statements are either True or False.
- NOT($\neg$), AND($\wedge$), OR($\vee$) and XOR ($\oplus$) are Boolean operators.
  - Could be used to combine basic statements into complex statements.
  - Different representations: Truth tables, Venn Diagrams.

- We have seen some properties of NOT,AND,OR and XOR.
  - AND and OR distribute over each other.
  - AND and OR are dual (With respect to the NOT transformation).
  - XOR and AND behave like addition and multiplication.

- We can sometimes use the properties of the Boolean operators to simplify complex statements.

EXERCISES

- Calculating expressions.
- Building truth tables and Venn diagrams.
- Proving basic properties of AND, OR, NOT, XOR.
- Simplifying expressions.