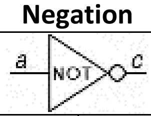
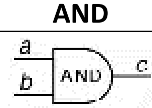
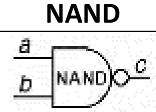
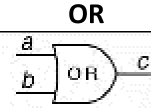
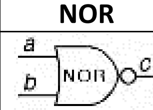
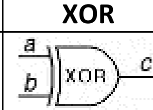
1. Basic definitions
    a. Computer architecture
        i. "Hardware-software interface"
        ii. Attributes of a system visible to the programmer (instruction set, I/O mechanisms…)
        iii. Attributes which impact the logical execution of a program
    b. Computer organization/microarchitecture
        i. Operational units of a machine transparent to programmer (control signals, memory technology)
        ii. Interconnections between them that realize the architectural specifications
    c. Tasks of a computer (according to Stallings)
        i. Data processing
            1. Keyboard to monitor, microphone to speaker, so on
        ii. Data storage
            1. Network to memory when downloading a webpage
            2. Longer-term storage on hard drive or SSD
        iii. Data movement
            1. Internal or external source and destination
        iv. Control
            1. Manage resources and control its functional parts
    d. Parts of a computer
        i. I/O
            1. Mouse, keyboard main examples we think of
            2. Other peripherals count too (like speakers)
        ii. Main memory
            1. RAM, caches
        iii. CPU
            1. Registers, store values
            2. ALU, perform operations
            3. Internal bus, transfer data
            4. Control unit, the brain
                a. Sequencing logic, where to go next
                b. Control unit registers, decoders
                c. Control memory
        iv. System interconnection/bus
            1. Move data around other parts

2. Analog vs. digital
    a. Analog – represent values by a continuously variable physical quantity
        i. For our uses, voltage
        ii. Key word is *continuous*
            1. Suited to amplification of real world phenomena (sound)
            2. Suited to calculating continuous function values (integrals)
            3. Subject to noise, difficult to debug
    b. Digital – use discrete (discontinuous) values to represent data
        i. Suited to discrete mathematics (like accounting)
        ii. Needs to sample continuous data
            1. Will miss data that fluctuates faster than sampling rate
        iii. Fixed 0 and 1, low and high, false and true
            1. Far more resistant to noise, easier to debug

3. Boolean algebra
    a. Algebra of truth values 0 and 1, along with conjunction (AND), disjunction (OR), and negation
        i. George Boole, 1854
        ii. Claude Shannon, 1938, uses it to solve circuit design problems
    b. * = AND, + = OR, ~ = negation, ⊕ = XOR, variables A, B…
    c. Duality principle – swap all signs (+, *, 0, 1) and the underlying logic is still the same
    d. Operator precedence: NOT, AND, OR
    e. Logic types
        i. Combinational – output based solely on current input
        ii. Sequential logic – output based on input and previous stored values (memory)

Truth Tables for Digital Design Gates								
Operation:		**Negation**		**AND**	**NAND**	**OR**	**NOR**	**XOR**
Gates:								
A	B	~A	~B	A * B	~(A * B)	A + B	~(A + B)	A ⊕ B
0	0	1	1	0	1	0	1	0
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	1	0	0	1	0	1	0	0

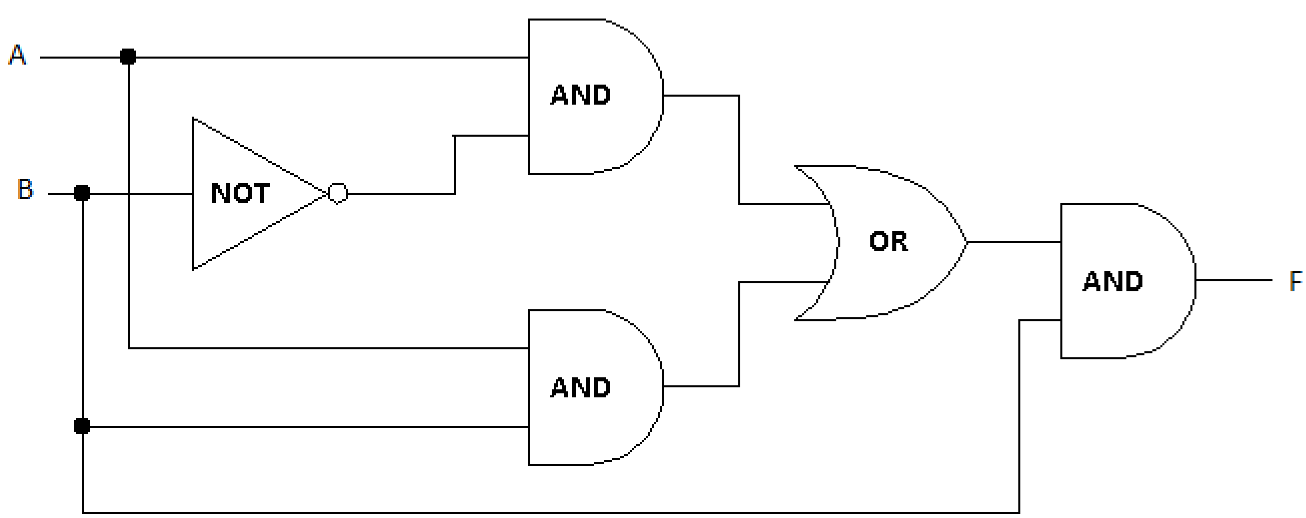
4. Logical equivalence
   a. May have seen these before if you took ECS 20 or another discrete mathematics class

Laws of Logical Equivalence		
Name	**OR version**	**AND version**
Commutative	A + B = B + A	A * B = B * A
Associative	(A + B) + C = A + (B + C)	(A * B) * C = A * (B * C)
Distributive	A + (B * C) = (A + B) * (A + C)	A * (B + C) = (A * B) + (A * C)
Idempotent	A + A = A	A * A = A
Identity	A + 0 = A	A * 1 = A
	A + 1 = 1	A * 0 = 0
Complement	A +~A = 1	A * ~A = 0
	~1 = 0	~0 = 1
Double Negative	~(~A) = A	
De Morgan's	~(A + B) = ~A * ~B	~(A * B) = ~A + ~B
Absorption	A + (A * B) = A	A * (A + B) = A

5. Examples
   a. A + ~A * B = A + B. Why?

Assertion	Reason
A + ~A * B	Initial function
= (A + ~A) * (A + B)	Distributive Law for OR
= 1 * (A + B)	Complement Law for OR
= (A + B) * 1	Commutative Law for AND (won't bother with this from now on)
= A + B	Identity Law for AND

   b. Write down the function for the following digital logic circuit.



      i. f = ((A * ~B) + (A * B)) * B