Logic Gates and Truth Tables

We can begin to define the building blocks of the CPU by basic instructions with input bits and output bits through **logical gates**.

- By convention, you will see that the input bits are labeled **A** and **B** by default.

Logic Gate #1:

Logic Gate #2:

Logic Gate #3:

Logic Gate Challenge: ***A XOR B***

We can also express this in a table known as a **truth table**:

Op.	Binary	Math	Example Values		
	A	x	1100	110011	101
	B	y	1010	11	010
AND	A & B	xy			
OR	A | B	x + y			
XOR	A ^ B	x XOR y			
NOT	!A	x'			

Truth Table: Half Adder

A	B	A + B	SUM	CARRY

Truth Table for a Half Adder

Circuit Diagram for a "Half Adder":

Full Adder:

A	B	$CARRY_{in}$			SUM	$CARRY_{out}$

Truth Table for a Full Adder

Circuit Diagram for a "Full Adder":

Chaining Circuits Together: _____

Disadvantages:

Instruction Set Architecture (ISA)
Every CPU has a set of commands it understands known as its Instruction Set Architecture or ISA. Two ISAs are **very** common:

1.

2.

An ISA defines the function of the hardware in the CPU.

CPU Registers
Each CPU core has an extremely limited number of _____ that are used for general purpose CPU operations:

- x64: 16 registers of 64 bits
- ARMv8: 31 registers of 64 bits

With very few exceptions _____.

Instruction Sets
Every ISA defines a set of instructions that a CPU can execute:

Move:	`MOV, XCHG, PUSH, POP, ...`
Arithmetic (int):	`ADD, SUB, MUL, DIV, NEG, CMP, ...`
Logic:	`AND, OR, XOR, SHR, SHL, ...`
Control Flow:	`JMP, LOOP, CALL, RET, ...`
Synchronization:	`LOCK, ...`
Floating Point:	`FADD, FSUB, FMUL, FDIV, FABS, ...`

ARM processors have significantly fewer instructions and are known as _____ while x64 processors have a greater set of instructions and known as _____.

Q: Advantages of RISC / CISC?

CPU Instruction in a Real Program

	04.c	gcc 04.c objdump -d ./a.out	

3	`int main() {`	`f3 0f 1e fa`
		`55`
		`48 89 e5`
		`48 83 ec 10`
			`endbr64`
			`push %rbp`
			`mov %rsp,%rbp`
			`sub $0x10,%rsp`

4	`int a = 0;`	`c7 45 fc 00 00 00 00`	`movl $0x0,-0x4(%rbp)`
5	`a = a + 3;`	`83 45 fc 03`	`addl $0x3,-0x4(%rbp)`
6	`a = a - 2;`	`83 6d fc 02`	`subl $0x2,-0x4(%rbp)`
7	`a = a * 4;`	`c1 65 fc 02`	`shll $0x2,-0x4(%rbp)`

8	`a = a / 2;`	`8b 45 fc`
		`89 c2`
		`c1 ea 1f`
		`01 d0`
		`d1 f8`
		`89 45 fc`
			`mov -0x4(%rbp),%eax`
			`mov %eax,%edx`
			`shr $0x1f,%edx`
			`add %edx,%eax`
			`sar %eax`
			`mov %eax,-0x4(%rbp)`

9	`a = a * 5;`	`8b 55 fc`
		`89 d0`
		`c1 e0 02`
		`01 d0`
		`89 45 fc`
			`mov -0x4(%rbp),%edx`
			`mov %edx,%eax`
			`shl $0x2,%eax`
			`add %edx,%eax`
			`mov %eax,-0x4(%rbp)`

10	`printf("Hi");`	`48 8d 3d f0 0d 00 00`
		`b8 00 00 00 00`
		`e8 42 fe ff ff`
			`lea 0xdf0(%rip),%rdi`
			`# 2004 <_IO_stdin_used+0x4>`
			`mov $0x0,%eax`
			`callq 1060<printf@plt>`

11	`a = a * 479;`	`8b 45 fc`
		`69 c0 df 01 00 00`
		`89 45 fc`
			`mov -0x4(%rbp),%eax`
			`imul $0x1df,%eax,%eax`
			`mov %eax,-0x4(%rbp)`

Program Counter (PC):

Operation Timings: