Introduction to Computer Science and Engineering
Digital Lab #1

Introduction

As you start to write progressively more complex programs, you may be curious about how a computer is able to execute the code you write. One of the goals of classes like this course and CSE 2312 (Computer Organization and Assembly Language Programming) is to show a small glimpse into how computers work internally.

This lab serves as an introduction to the digital logic circuits used inside the computer that evaluate logical expressions. You will learn more about discrete-valued systems in CSE 2315 (Discrete Structures) and CSE 2441 (Digital Logic Design).

AND Logic:

To start this discussion, consider this C89 code snippet

```
int a, b, y;

...

y = a && b;

if (y)
     "do something";
```

that only does "something" if both a!=0 AND b!= 0.

If we limit the values of a and b to 1 ("true") and 0 ("false"), this allows us to create a simple truth table:

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

Similarly, consider this C99 code snippet that uses the standard bool data type:

```
#include <stdbool.h>

...

bool a, b, y;

...

y = a && b;

if (y)
     "do something";
```

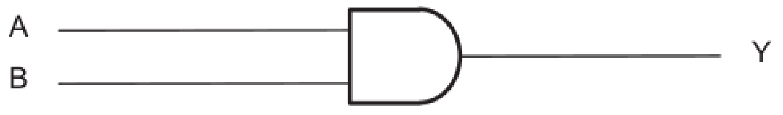
that only does "something" if both "a" AND "b" are true.

Using the bool values of "true" and "false", this allows us to create a simple truth table:

a	b	y
false	false	false
false	true	false
true	false	false
true	true	true

Regardless of whether you are using a logic expression with type int or bool variables, the values are limited to two values (true and false). The use of discrete values (binary values) is common in the computer.

Inside the computer, there are electronic logic gates that evaluate logical expressions. For the AND operation we used above, there is a unique symbol used to denote a logic gate that evaluates an AND operation, with A and B being the input pins (wires) and Y being the output (Y = A && B) as shown below:



There are many millions of logic gates like this in a modern computer.

In software, we can map the values of an integer to true and false like this:

> a !=0 → true
> a == 0 → false

In hardware, a range of voltages (V) are mapped to true and false. For instance, for an AND logic gate found in a chip called the 74HC08, we could map voltages like this:

> V >= 3.15V → true (aka High/'1')
> V <= 1.35V → false (aka Low/'0')
> 1.35V < V < 3.15V → indeterminate (unknown state)

The truth table for the 74HC08 AND logic gate might be viewed like this, where Va, Vb, and Vy are the voltages on the pins:

Va input	Vb input	Vy output
V <= 1.35V (false/low)	V <= 1.35V (false/low)	V <= 0.1V (false/low)
V <= 1.35V (false/low)	V >= 3.15V (true/high)	V <= 0.1V (false/low)
V >= 3.15V (true/high)	V <= 1.35V (false/low)	V <= 0.1V (false/low)
V >= 3.15V (true/high)	V >= 3.15V (true/high)	V >= 4.4V (true/high)

Note: outputs are intentionally much greater than the input thresholds for high signals and much lower than the input thresholds for low signals to provide a margin of error to make the logic more robust.

So, you might visualize this mapping like this:

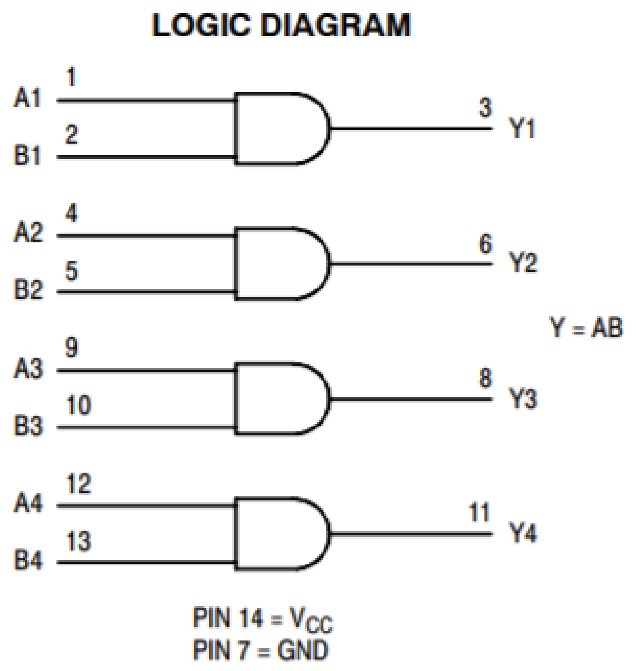
C89 int	C99 bool	Hardware
0	false	V <= 1.35V (Low)
1	true	V >= 3.15V (High)

The 74HC08 logic chip actually contains 4 AND gates so that we could evaluate up to 4 AND operations in one chip. The chip has 14 pins. Two of the pins provide power to the chip as follows:

Pin	Connection
Vcc	Connect to a +5V power supply (red wires)
GND	Connect to 0V (ground) (black wires)

The remaining 14 – 2 = 12 pins are used to create 4 AND gates (4 gates x 3 pins/gate = 12 pins). All the AND gates can be used interchangeably.

A small portion of the datasheet is shown below:



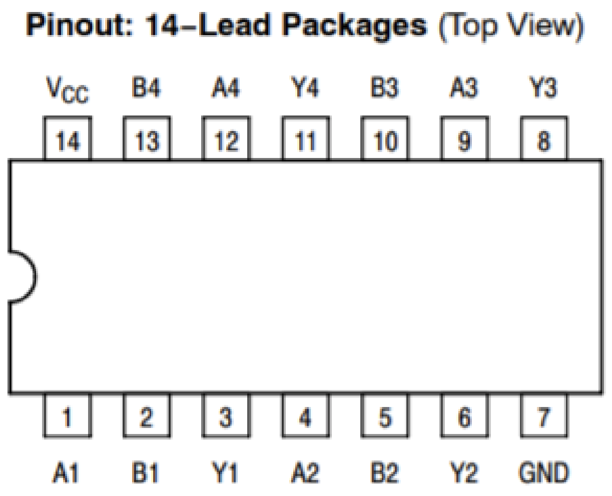


Figure: 74HC08 Logic Chip Containing 4 AND gates, © On Semiconductor

The logic gate chip will simply plug into a “breadboard” to allow us to use it. More on this later.

OR Logic:

Now let’s look at the OR operation.

In C89, you might write

```
int a, b, y;

...

y = a || b;

if (y)
      “do something”;
```

that only does “something” if either a!=0 OR b!= 0.

As in the case of the AND operation, if we limit the values of a and b to 1 (“true”) and 0 (“false”), this allows us to create a simple truth table:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

And the C99 code snippet that uses the standard bool data type might look like:

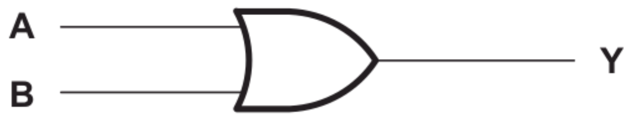
```
#include <stdbool.h>

...

bool a, b, y;

...

y = a || b;

if (y)
      “do something”;
```

that only does “something” if either “a” OR “b” are true.

Using the bool values of “true” and “false”, this allows us to create a simple truth table:

a	b	y
false	false	false
false	true	true
true	false	true
true	true	true

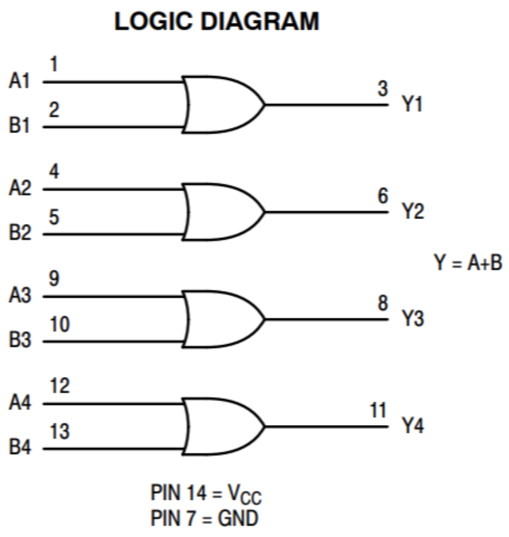
In hardware, the symbol for a logical OR gate is:



The truth table for the OR logic gate found in a chip called 74HC32 might be viewed like this, where Va, Vb, and Vy are the voltages on the pins:

Va input	Vb input	Vy output
V <= 1.35V (false/low)	V <= 1.35V (false/low)	V <= 0.1V (false/low)
V <= 1.35V (false/low)	V >= 3.15V (true/high)	V >= 4.4V (true/high)
V >= 3.15V (true/high)	V <= 1.35V (false/low)	V >= 4.4V (true/high)
V >= 3.15V (true/high)	V >= 3.15V (true/high)	V >= 4.4V (true/high)

A small portion of the datasheet is shown below:



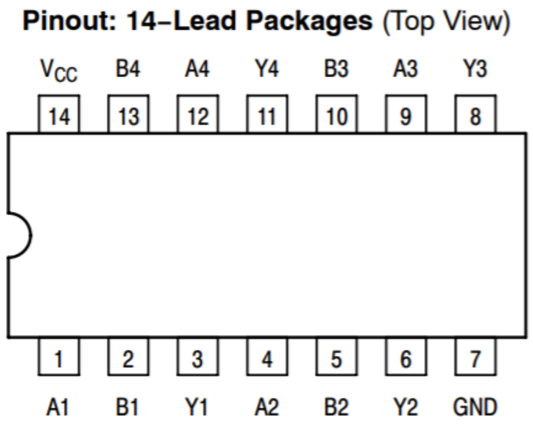


Figure: 74HC32 Logic Chip Containing 4 OR gates, © On Semiconductor

<u>NOT Logic:</u>

The final logic operation we will study is NOT (also called one's complement in software).

In C89, you might write

```
int a, y;

...

y = !a;

if (y)
     "do something";
```

that only does "something" if a==0.

If we limit the values of a to 1 ("true") and 0 ("false"), this allows us to create a simple truth table:

a	y
0	1
1	0

And the C99 code snippet that uses the standard bool data type might look like:

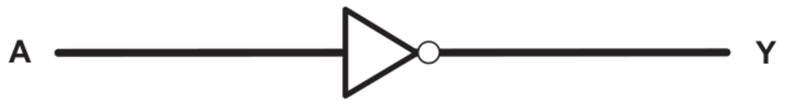
```
#include <stdbool.h>

...

bool a, y;

...

y = !a;

if (y)
     "do something";
```

that only does "something" if "a" is false.

Using the bool values of "true" and "false", this allows us to create a simple truth table:

a	y
false	true
true	false

In hardware, the symbol for a logical OR gate is:



(c) Texas Instruments

The truth table for the NOT logic gate (called an inverter) found in a chip called 74HC04 might be viewed like this, where Va and Vy are the voltages on the pins:

Va input	Vy output
V <= 1.35V (false/low)	V >= 4.4V (true/high)
V >= 3.15V (true/high)	V <= 0.1V (false/low)

Since NOT gates (inverters) only have one input and one output, you can fit 6 of them into a 14 pin chip assuming two power pins. A small portion of the datasheet is shown below:

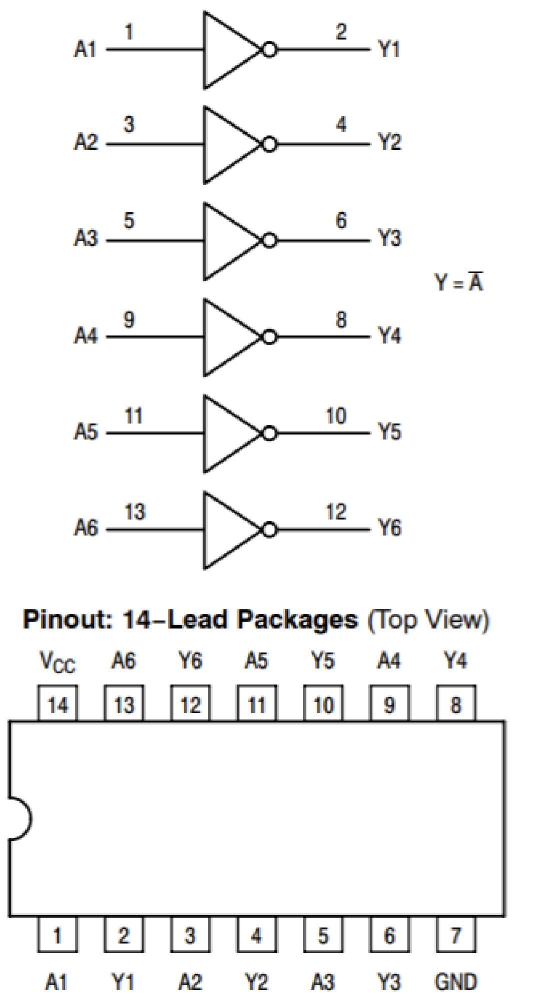
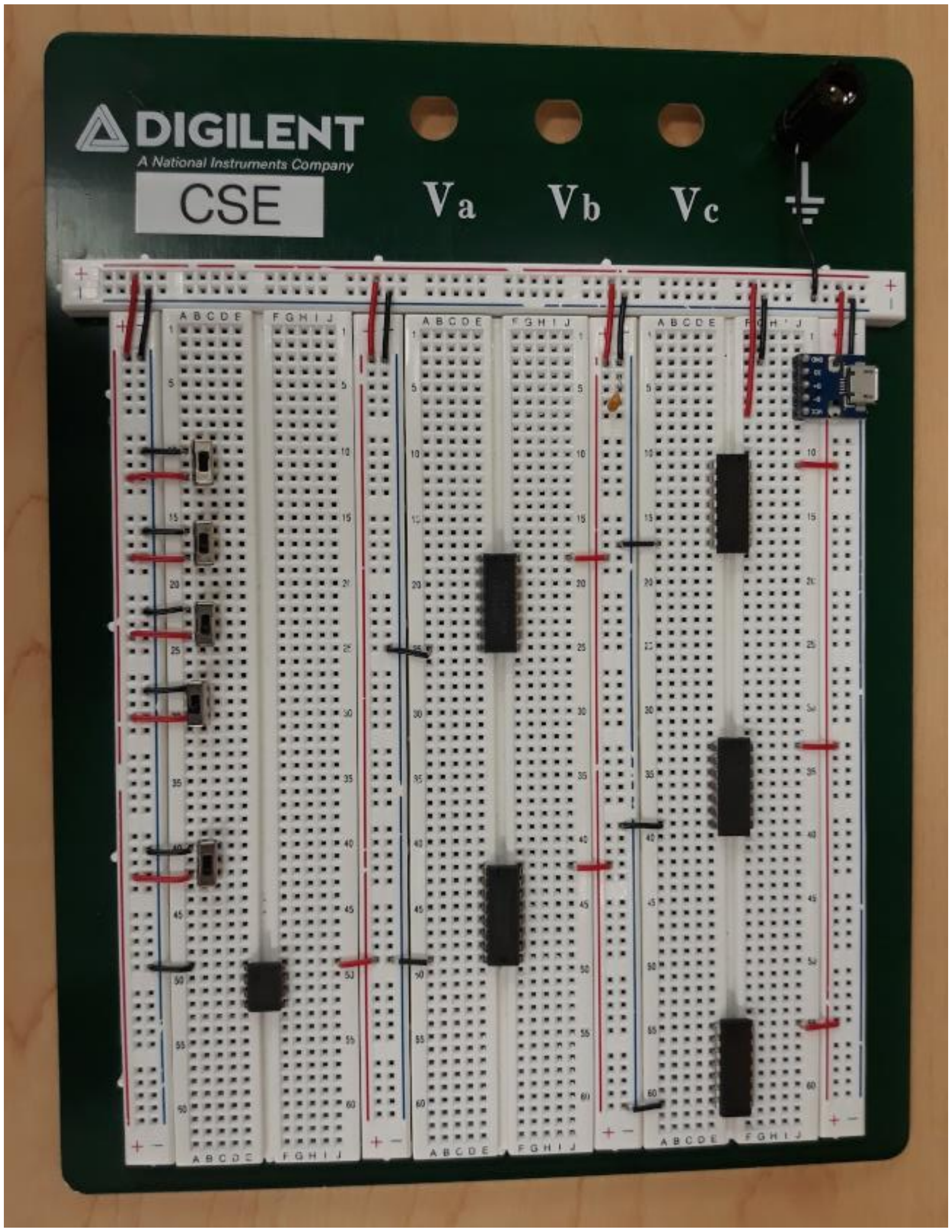


Figure: 74HC04 Logic Chip Containing 6 NOT gates, © On Semiconductor

Circuit Components

In the lab, we will use a selection of integrated circuits (chips), LEDs, switches, resistors, and wire leads to build circuits and compare them to the function of a corresponding C program.

To make the lab easier, several ICs, a USB adapter, slide switches, and red and black wires have been populated in the board. Please do not remove or change these wires, switches, or remove the USB adapter. The board is shown below:



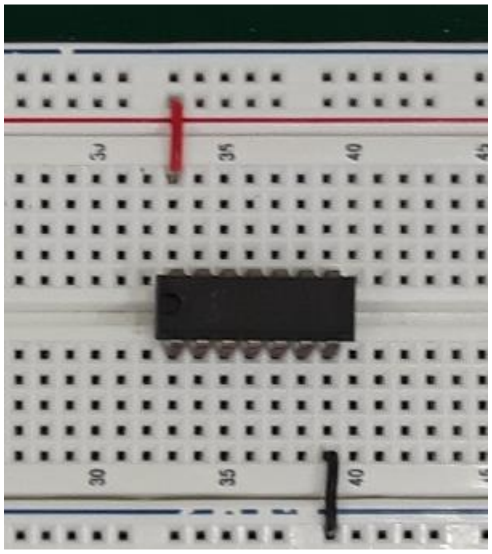
A Few Wiring Hints

1. Red wires should only be used when connecting 5V to a component.

2. Black wires should only be used when connecting 0V (ground) to a component.

3. To assist in debugging, please do not use red or black wires for any other purpose.

4. While the chips on the board are already inserted, note that they straddle the divider between two 5 contact rows on the board as shown below for a typical AND, OR, or NOT gate ICs with 14 pins:



Note that pin 14 is connected to 5V and pin 7 is connected to 0V so that the chip is powered.

5. Please use the LED colors mentioned in the worksheet.

Lab 2 Worksheet

Name Servando Olvera

Course/Section 010

Please complete the following steps to complete Lab 2:

1. Take the blue LED and a 220 ohm resistor (red-red-brown) in series as shown in the circuit above and connect them across the red and blue distribution strip at a point by the USB connector so it is out of the way for the next steps. It should glow brightly.

2. Locate a 74HC08 (AND gate) chip on the breadboard. It should already be powered through red and a black wires connected from pins 14 and 7 of the chip to the 5V and 0V (red and blue) distribution strips. Again, red is used to 5V and black is used for 0V.

3. Connect the center connection of two slide switches to the A and B inputs of the AND gate (use any of the 4 AND gates in the chip). Now, connect two pairs of red LEDs and resistors in series and connect to the center connection of the two switches. You should be slide the switch from left to right and see the LED turn on and off for the corresponding switch. You are now sending '1'/true and '0'/false signals to the two inputs of an AND gate.

4. Measure the voltage of the two slide switches which provide the AND gate inputs and record below the voltages and LED status below:

Switch A	Voltage at A input of gate	Red LED on?
'0' (connected to the black wire)	0 V	No
'1' (connected to the red wire)	4.931 V	Yes

Switch B	Voltage at B input of gate	Red LED on?
'0' (connected to the black wire)	0 V	No
'1' (connected to the red wire)	4.926 V	Yes

5. Connect the output of the AND gate to an orange LED and resistor in series. Now, slide the two switches to all four combinations and record the observations below.

Switch A	Switch B	Voltage at Y output	Orange LED on?
0	0	0.002 V	No
0	1	0 V	No
1	0	0 V	No
1	1	4.557 V	Yes

An LED on indicates the output of the gate is true/'1'. Verify that this matches the table shown in the introduction. The orange LED shows the evaluation of A && B.

6. Locate a 74HC32 (OR gate) chip on the breadboard. It should already be powered through red and a black wires connected from pins 14 and 7 of the chip to the 5V and 0V (red and blue) distribution strips. Connect the A and B inputs of the AND gate to the A and B inputs of the OR gate.

7. Connect the output of the OR gate to a yellow LED and resistor in series. Now, slide the two switches to all four combinations and record the observations below.

Switch A	Switch B	Voltage at Y output	Yellow LED on?
0	0	0 V	No
0	1	4.543 V	Yes
1	0	4.538 V	Yes
1	1	4.522 V	Yes

An LED on indicates the output of the gate is true/'1'. Verify that this matches the table shown in the introduction. The yellow LED shows the evaluation of A || B.

8. Locate a 74HC04 chip on the breadboard. It should already be powered through red and a black wires connected from pins 14 and 7 of the chip to the 5V and 0V (red and blue) distribution strips. Connect the A input of a NOT gate to output of the OR gate.

9. Connect the output of the NOT gate to a green LED and resistor in series. Now, slide the two switches to all four combinations and record the observations below.

Switch A	Switch B	Voltage at Y output	Green LED on?
0	0	4.502 V	Yes
0	1	4.479 V	Yes
1	0	0 V	No
1	1	0 V	No

An LED on indicates the output of the gate is true/'1'. Verify that the output of the yellow and green are the inverted versions of each other. The green LED shows the evaluation of !(A || B).

10. Write a short C program compiled with GCC on your Raspberry Pi that:

a. Accepts two integers (A and B) from the user.

b. Displays the value of A && B, A || B, and !(A || B) on the screen.

c. Verify that these outputs match the values in steps 5, 7, and 9.

11. Lab checkout steps:

a. Show your working circuit and program to the grader.

b. Create a file, lastname_netid_lab2.zip, that includes the following files:

- A JPEG image of your working circuit

- The source code of your program

- The output of your program

- This completed lab worksheet with all data completed

- Note: images do not need specific file names

c. Upload the zip file to Canvas.

d. After you have your picture and send the zip file, dismantle your circuit and return everything to the box on the shelf. Do not remove the USB adapter, ICs, slide switches, or the pre-loaded red and black wires.

Thank you for attending the lab. We hope some of this material was new and interesting to you.

Drs. Losh and Eary

Lab 2 Rubric

Prerequisite:
-40: Did not complete familiarization quiz on time

Assignment:
-40: No completed worksheet
-10: No image of working circuit
-20: No source code for program
-20: No output of source code
-10: Wrong file name