ELEC1601 Week 3

Introduction to Computer Systems

Outline:

Lab Discussion

Encoding Information

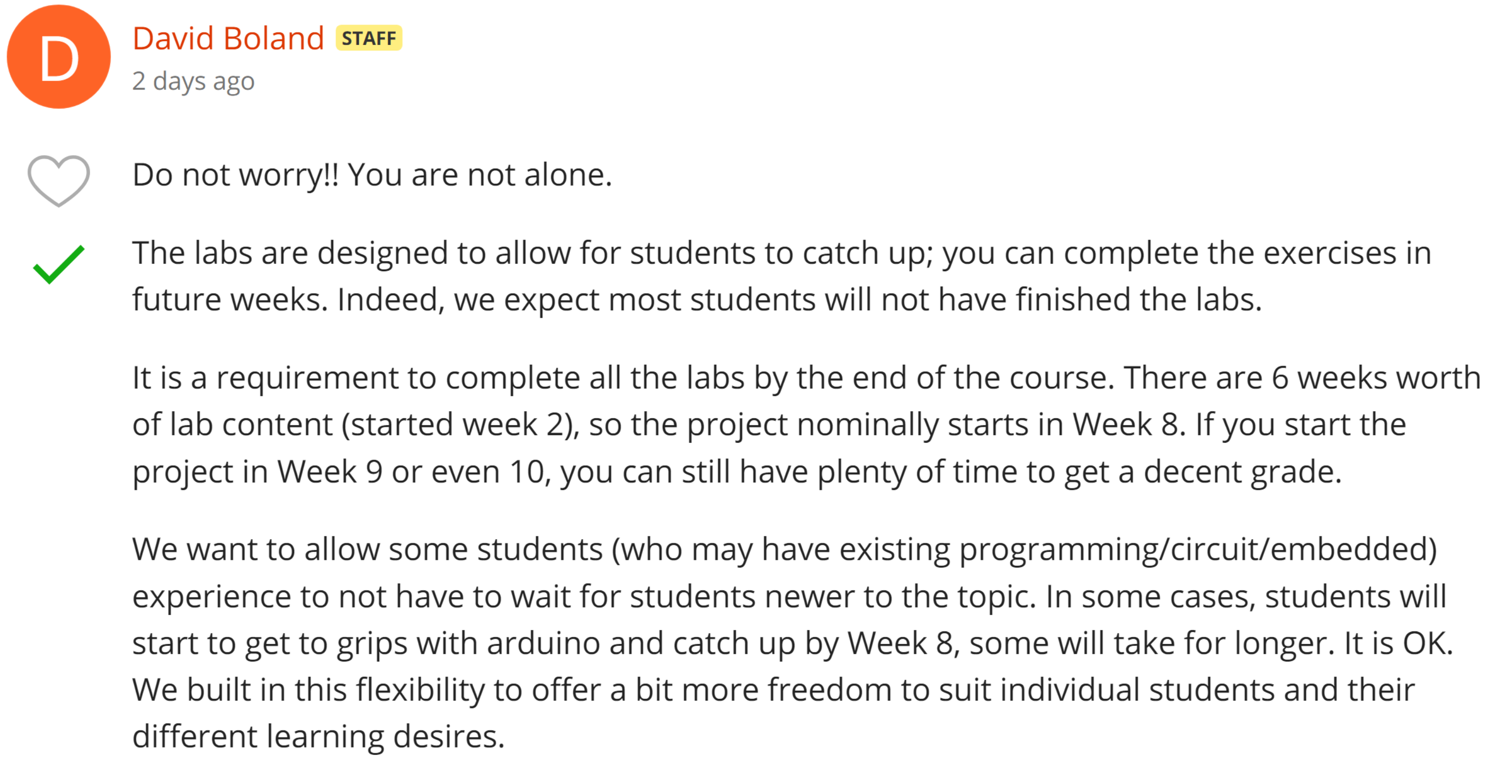
- Introduction
- Encoding numbers
- General Encoding
- Floating-point

Lab 1 Discussion

- If you're still on Lab 1…
- Debouncing
- Debugging with print statements

If you're still on Lab 1…

– https://edstem.org/au/courses/16814/discussion/2139397



David Boland STAFF
2 days ago

Do not worry!! You are not alone.

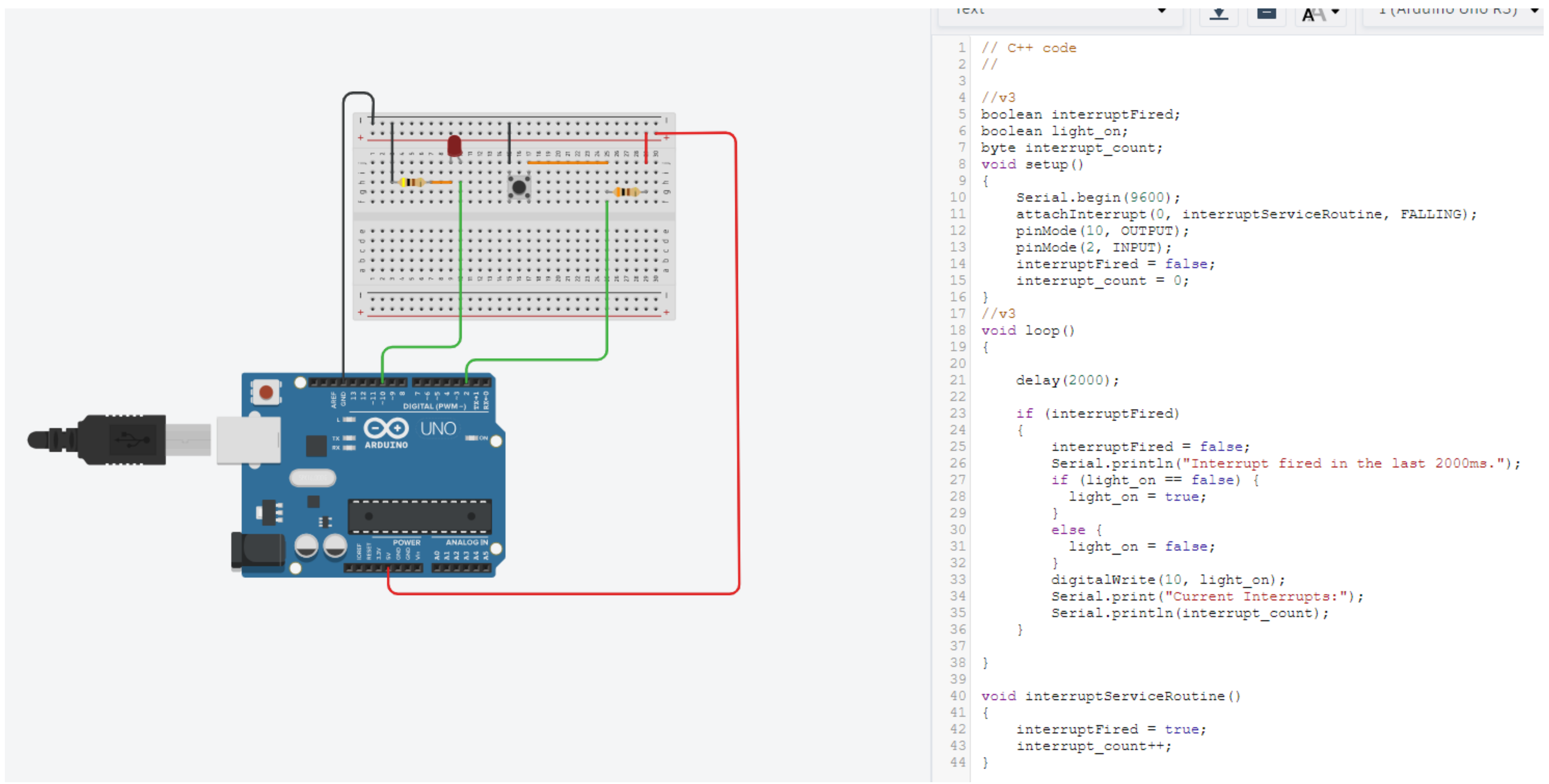
The labs are designed to allow for students to catch up; you can complete the exercises in future weeks. Indeed, we expect most students will not have finished the labs.

It is a requirement to complete all the labs by the end of the course. There are 6 weeks worth of lab content (started week 2), so the project nominally starts in Week 8. If you start the project in Week 9 or even 10, you can still have plenty of time to get a decent grade.

We want to allow some students (who may have existing programming/circuit/embedded) experience to not have to wait for students newer to the topic. In some cases, students will start to get to grips with arduino and catch up by Week 8, some will take for longer. It is OK. We built in this flexibility to offer a bit more freedom to suit individual students and their different learning desires.

Debouncing

- When you press or release a button, it can voltage can bounce many times as contacts are briefly made/disconnected



```
// C++ code
//

//v3
boolean interruptFired;
boolean light_on;
byte interrupt_count;
void setup()
{
    Serial.begin(9600);
    attachInterrupt(0, interruptServiceRoutine, FALLING);
    pinMode(10, OUTPUT);
    pinMode(2, INPUT);
    interruptFired = false;
    interrupt_count = 0;
}
//v3
void loop()
{

    delay(2000);

    if (interruptFired)
    {
        interruptFired = false;
        Serial.println("Interrupt fired in the last 2000ms.");
        if (light_on == false) {
          light_on = true;
        }
        else {
          light_on = false;
        }
        digitalWrite(10, light_on);
        Serial.print("Current Interrupts:");
        Serial.println(interrupt_count);
    }

}

void interruptServiceRoutine()
{
    interruptFired = true;
    interrupt_count++;
}
```

Debugging with print statements

- Your code (as it gets more complicated) is almost never correct first time.
- You need a way of identifying the problem.
- Print statements are the easiest way

Lab 2 Intro

- Const or #define
- 7 segment display
- arrays
- The switch…case statement
- Servos

Const or #define

- Can be used to make code easier to read

```
boolean interruptFired;
boolean light_on;
int counter;
int var;
#define PIN_A 13
const int  PIN_B = 12, PIN_C = 11;
const int PIN_D = 10, PIN_E = 9, PIN_F = 8;
const int PIN_G = 7;
Servo myservo, myservo2;

void setup()
{
    Serial.begin(9600);
    attachInterrupt(0, interruptServiceRoutine, FALLING);
    pinMode(PIN_G, OUTPUT);
    pinMode(8, OUTPUT); //F
```

Arrays

- Store a list of numbers of the same type

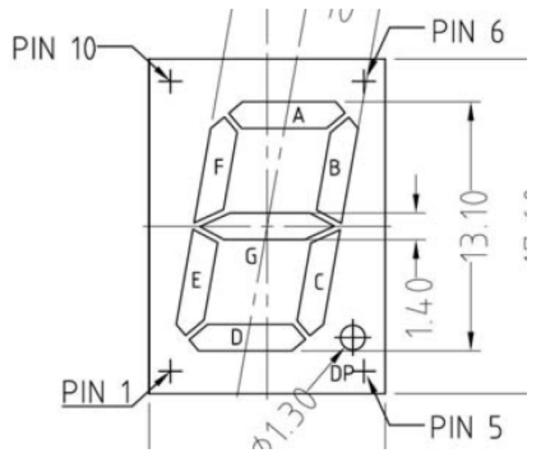
```
const int a[10] = {45, 55, 65,
```

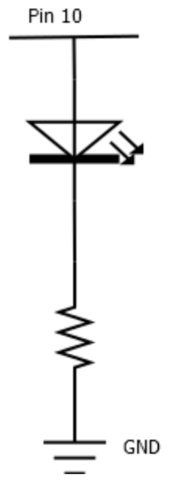
- Useful to iterate over a list with a variable

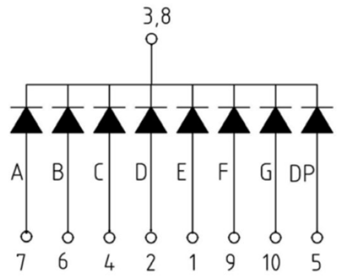
```
int index=0;
Serial.println(a[index]);
index++;
Serial.println(a[index]);
```

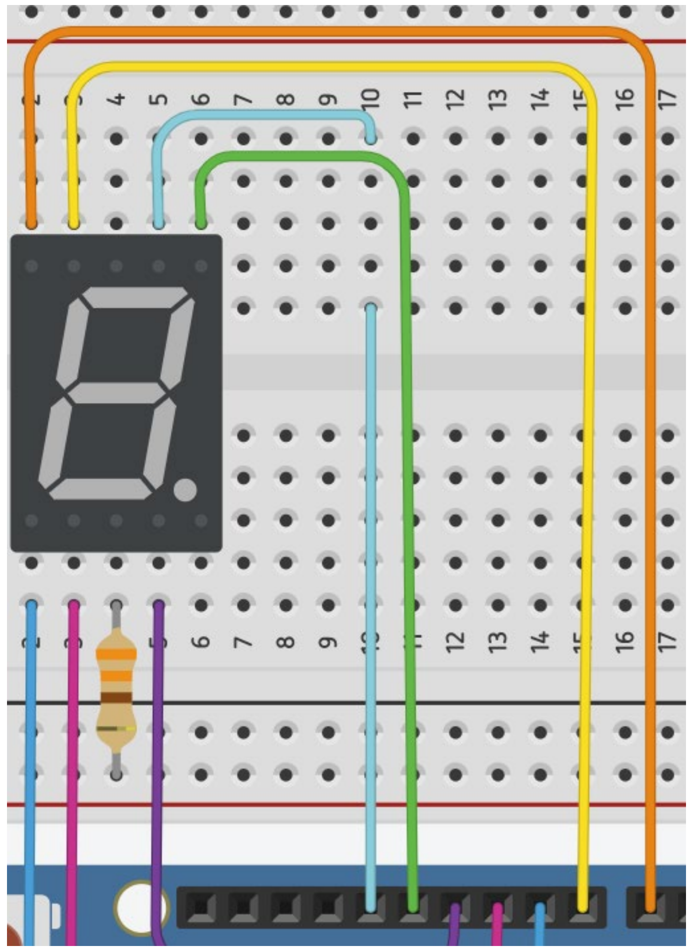
7 segment display

– **Just like many LEDs**



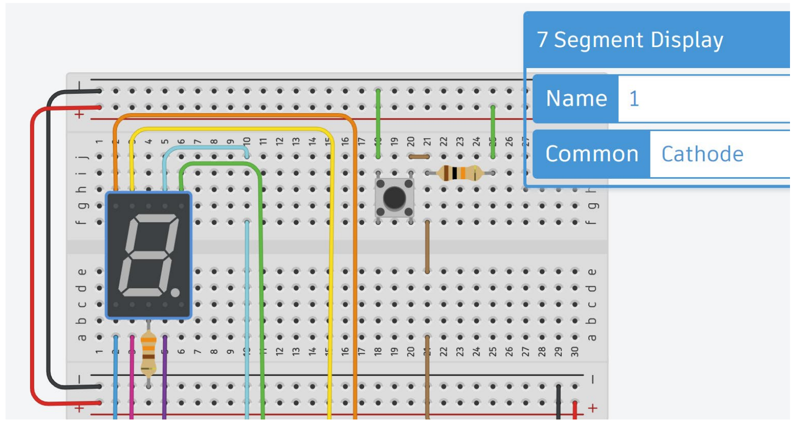






7 segment display

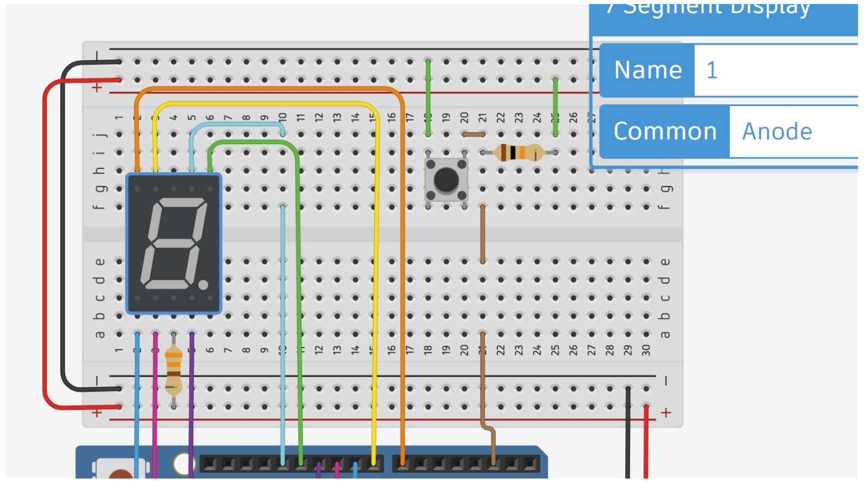
- **Just like many LEDs**
  - Be aware common cathode vs common anode



7 Segment Display

Name 1

Common Cathode



7 Segment Display

Name 1

Common Anode

The switch…case statement

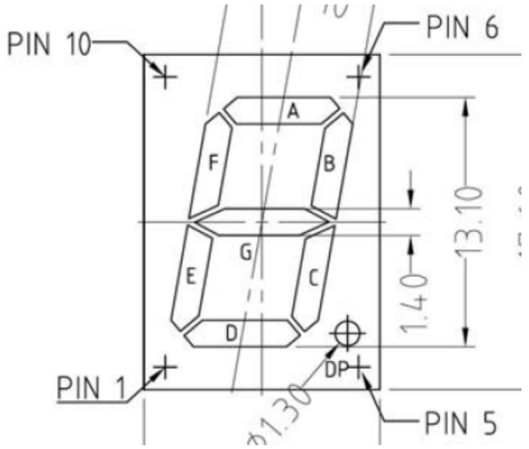
- Like a lot of if…else statements

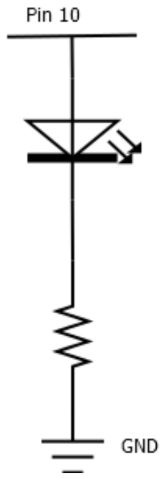
```
if (count == 1) {
  <<do something>>
}
else if (count ==2) {
  <<do something else1>>
}
else if (count ==3) {
  <<do something else2>>
}
else {
  <<do something else3>>
}
```

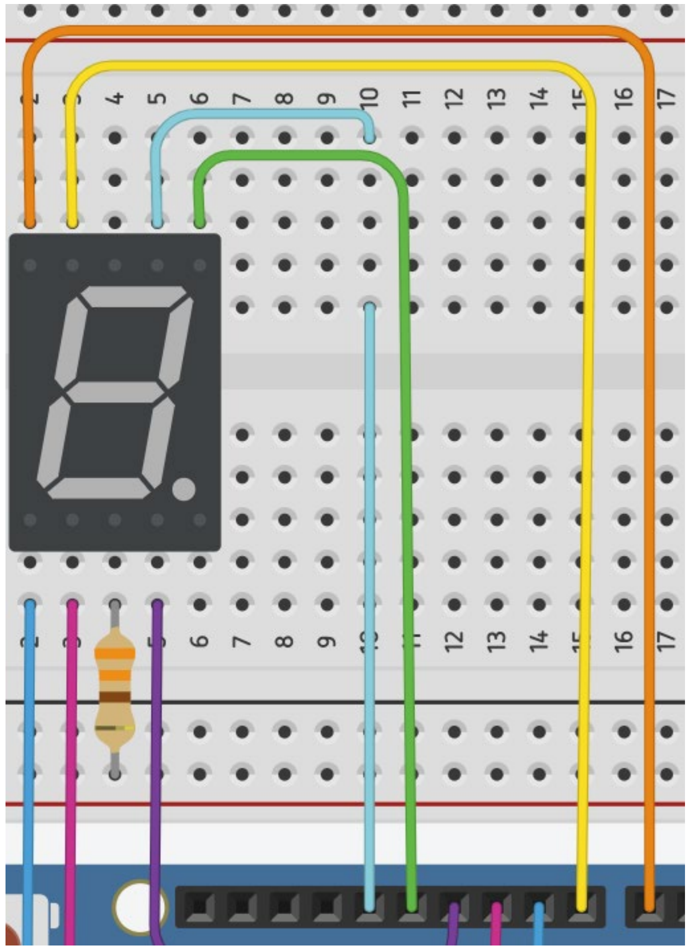
```
switch (count) {
case 1:
  <<do something>>
  break;
case 2:
  <<do something else1 >>
  break;
case 3:
  <<do something else2 >>
  break;
default:
  <<do something else3 >>
  break;
}
```

7 segment display

– **Just like many LEDs**







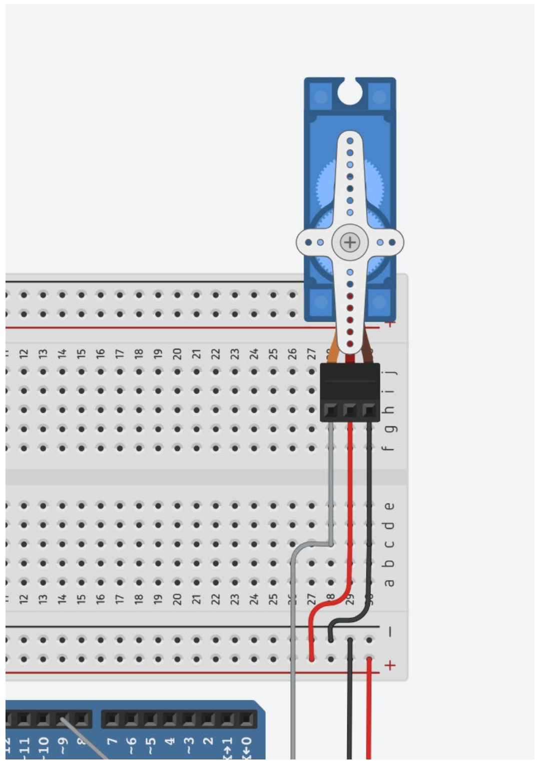
```
switch (          ) {
   case 0: A=0; B=0; C=0; D=0; E=0; F=0; G=1;break;
   case 1: A=1; B=0; C=0; D=1; E=1; F=1; G=1; break;
```

Servos

```
#include <Servo.h>
boolean interruptFired;
boolean direction;
int counter;
const int PIN_SERVO1=9;
Servo myservo;
int var;

void setup()
{
    Serial.begin(9600);
    myservo.attach(PIN SERVO1);
```

```
myservo.write(a[servo1_index]);
```



Encoding

Encoding Information

- **What do you need to know?**
  - Binary Numbers
  - Two’s complement numbers
    - Sign Extension
  - Hexadecimal numbers
    - Useful properties of binary/hexadecimal numbers
  - Fixed-point
    - How Bias/Scaling can modify a fixed-point number system
    - Errors vs Precision for number systems
  - General encoding
  - Floating-point

Recap: Binary numbers

– Convert 827_{10} to binary?

Recap: Two's complement numbers

– **Convert -827$_{10}$ to binary?**

Two’s complement numbers – How many bits

– **How do I represent 827 in two’s complement**

Sign Extension: Two’s complement

- What is 18 in 5-bits, 6-bits, 7-bits?
- What is -18 in 5-bits, 6-bits, 7-bits?

Recap: Two’s complement – example addition

- Compute 11+18 in two’s complement

Recap: Two's complement – example subtraction

- Compute 18-11 in two's complement

What are hexadecimal numbers

- 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Hexadecimal numbers

- **Convert 827_{10} to hex?**

Encoding Information

- **What do you need to know?**
  - Binary Numbers
  - Two's complement numbers
  - Hexadecimal numbers
    - Useful properties of binary/hexadecimal numbers
  - Fixed-point
    - How Bias/Scaling can modify a fixed-point number system
    - Errors vs Precision for number systems
  - Floating-point
  - General encoding

Extension: How many different numbers can you represent?

- 10 digits?
- 10 octal values?
- 10 hex values?
- 10 bits?

Extension: Useful properties of base 10

- Is 1345 a multiple of 10? Is it a multiple of 100?
- Is 13450 a multiple of 10? Is it a multiple of 100?
- Is 10400 a multiple of 10? Is it a multiple of 100?

- Is this hard?

Extension: Useful properties of base 10

- How do you multiply 1274 by 10?
- How do you multiply 1830 by 100?

- Is this hard?

Extension: Useful properties of base 10

- **How do you divide 10400 by 10?**
- **How do you divide 10400 by 100?**

- **Is this hard?**

Extension: Useful properties of base 10

- Is 1345 a multiple of 7? Is it a multiple of 77?
- Is 13450 a multiple of 70? Is it a multiple of 777?
- Is 10400 a multiple of 70? Is it a multiple of 777?
- How do you multiply 1274 by 7?
- How do you multiply 1830 by 77?
- How do you divide 10400 by 77?
- How do you divide 10400 by 777?

- Is this hard?

Extension: Useful properties of base 2

- Is 101100 a multiple of 2? Is it a multiple of 4?

- How do you multiply 1010010 by 2?
- How do you multiply 1011110 by 4?

- How do you divide 10101010 by 2?
- How do you divide 10111100 by 4?

- Is this hard?

Extension: Useful properties of base 8, base 16

- What could you say about a hexadecimal number terminated by 3 zeros?

Encoding Information

- **What do you need to know?**
  - Binary Numbers
  - Two’s complement numbers
  - Hexadecimal numbers
    - Useful properties of binary/hexadecimal numbers
  - Fixed-point
    - How Bias/Scaling can modify a fixed-point number system
    - Errors vs Precision for number systems
  - Floating-point
  - General encoding

Fixed-point: Before we begin:

- You have 8 bits. What range of numbers can you represent?

What is fixed point?

- A number system where you choose where the point that separates integers from fractions is fixed

How do we interpret fixed-point in radix-10?

- Consider the number 429.86
- How do we break it down?

How do we interpret fixed-point in radix-2?

- Consider the number 01011.11_2
- How do we break it down?

Encoding Information

- **What do you need to know?**
  - Binary Numbers
  - Two’s complement numbers
  - Hexadecimal numbers
    - Useful properties of binary/hexadecimal numbers
  - Fixed-point
    - How Bias/Scaling can modify a fixed-point number system
    - Errors vs Precision for number systems
  - Floating-point
  - General encoding

Extension: What is the worst-case rounding error in base-10?

- Consider the number 429.27
- We know we are representing the number to two decimal places. What is the worst-case rounding error?

Extension: What is the worst-case rounding error in base-2?

- Consider the number 01011.111
- We know we are representing the number to two decimal places. What is the worst-case rounding error?

Extension: Scaling in base-10

- Suppose we wish to represent numbers in the range [1000 10000] to the nearest 1000.
- How many digits do we need to store?

Extension: Scaling in base-2

- Suppose we wish to represent numbers in the range [1024 to 8192] to the nearest kbyte.
- How many bits do we need to store?

Extension: The relationship between bits, range, precision

- You have 8 bits. What range of numbers can you represent?

Extension: Bias in base-10

– Suppose we wish to represent numbers in the range [50 to 130]. How many digits do we need?

Extension: Bias in base-2

– Suppose we wish to represent numbers in the range [50 to 130]. How many bits do we need?

Extension: Advantages/Disadvantages of bias?

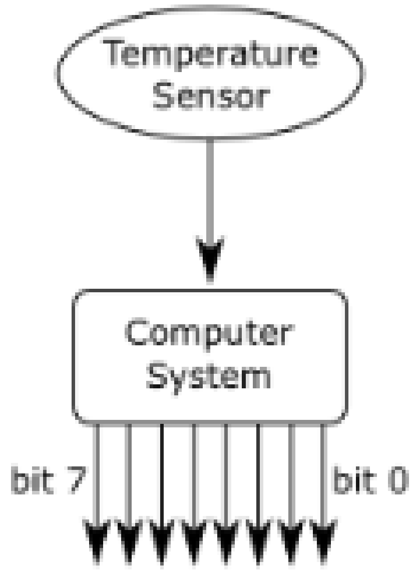
- Why do we want fewer bits when encoding?
- How does this impact addition/subtraction?
- How does this impact memory use?
- How does this impact multiplication?
- How does it impact data transfer (e.g. from sensor to CPU/from off-chip to on-chip memory)?

Some practice examples for fixed-point, scaling, bias and roundoff error

THE UNIVERSITY OF
SYDNEY

Custom fixed-point representation. Example 1

- What temperature can you encode?



Custom fixed-point representation. Example 2

- **Suppose I represent an 8-bit number that lies in the range [-2, 2] in two's complement fixed-point. What is the worst case round-off error?**

Custom fixed-point representation. Example 3

– Now suppose we wish to represent real numbers in the range [-10 10] with a worst-case error of 0.0625 using two’s complement fixed-point. How many bits do we need?

Custom fixed-point representation. Example 4

– Suppose we wish to represent real numbers in the range [50 to 130] with a worst-case error of 0.0625. How many bits do we need?

Bias/Scaling. Example 5

- **Calibrating an accelerometer**
  - Accelerometer Datasheet
  - analogRead() - Arduino Reference

- **Accelerometer returns 3 voltages in the x, y and z directions**
  - Analogue read represents a voltage of between 0 and 5V with 10 bits.
  - What voltage reading does {01 0101 0111, 00 1101 1011 10 0001 1000} represent?

Bias/Scaling Example

- Accelerometer
- analogRead() - Arduino Reference

- The voltage on a g scale has a sensitivity of 800mV/g with an offset (bias) of 1.65.
- What does {01 0101 0111, 00 1101 1011 10 0001 1000} represent in terms of g?

Encoding Information

- **What do you need to know?**
  - Binary Numbers
  - Two's complement numbers
  - Hexadecimal numbers
    - Useful properties of binary/hexadecimal numbers
  - Fixed-point
    - How Bias/Scaling can modify a fixed-point number system
    - Errors vs Precision for number systems
  - General encoding
  - Floating-point

Encoding symbols

- **Why?**
  - Computers do not simply work with numbers

- **Need to make a link between 1's and 0's and a symbol**

Example 1: ASCII

- **Came up with 100 characters initially. Leave some room for redundancy (future bits).**
  - How many bits to use?

- **Problems?**
  - Many other symbols desired:
    - E.g. unicode (currently 144,697 characters)

ASCII vs Unicode

- Which is better?

RGB coding

- Suppose you want to encode Red, Green and Blue. Each number lies in the range [0 255].
- How many bits do you require

- What is 1110 0010 1101 0100 0000 0101?

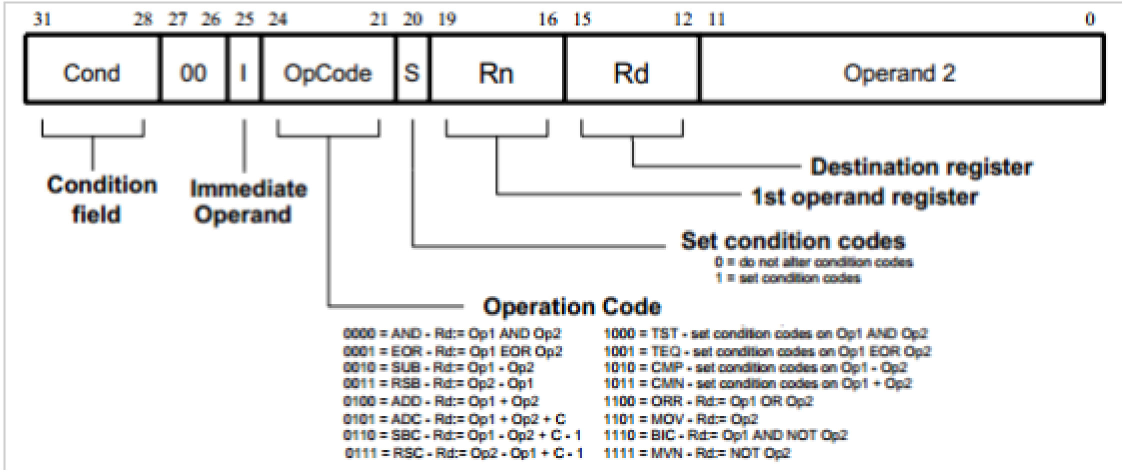
HSL coding

- Suppose you want to encode Hue [0 360], Saturation (percentage) and Lightness (percentage).
- How many symbols are required?
- What is a good encoding

Machine code – we'll revisit this later!

- **Machine Code**

```
            Cond    I OpCd S  Rn Rd Op2
ADD    R0,  R1, R2     @ 1110|00|0|0100|0|0001|0000|000000000010
```



0000 = AND - Rd:= Op1 AND Op2
0001 = EOR - Rd:= Op1 EOR Op2
0010 = SUB - Rd:= Op1 - Op2
0011 = RSB - Rd:= Op2 - Op1
0100 = ADD - Rd:= Op1 + Op2
0101 = ADC - Rd:= Op1 + Op2 + C
0110 = SBC - Rd:= Op1 - Op2 + C - 1
0111 = RSC - Rd:= Op2 - Op1 + C - 1
1000 = TST - set condition codes on Op1 AND Op2
1001 = TEQ - set condition codes on Op1 EOR Op2
1010 = CMP - set condition codes on Op1 - Op2
1011 = CMN - set condition codes on Op1 + Op2
1100 = ORR - Rd:= Op1 OR Op2
1101 = MOV - Rd:= Op2
1110 = BIC - Rd:= Op1 AND NOT Op2
1111 = MVN - Rd:= NOT Op2

Why Floating-point?

- **You want to represent a wide range with a fixed number of bits?**

- **Reason for floating-point: You want to represent a *variable* range with a fixed number of bits**

Example:

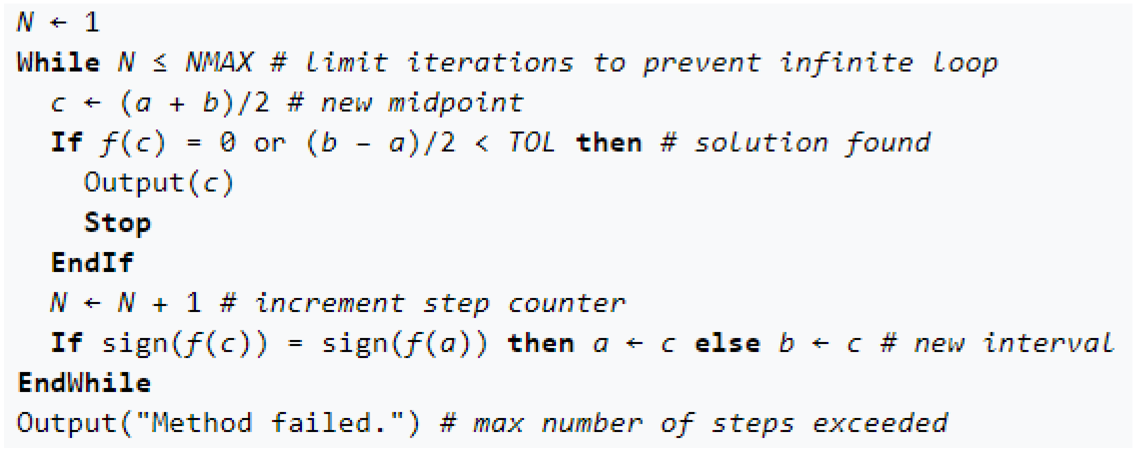
- You want to represent a wide range with a fixed number of bits?

- Reason for floating-point: You want to represent a *variable* range with a fixed number of bits

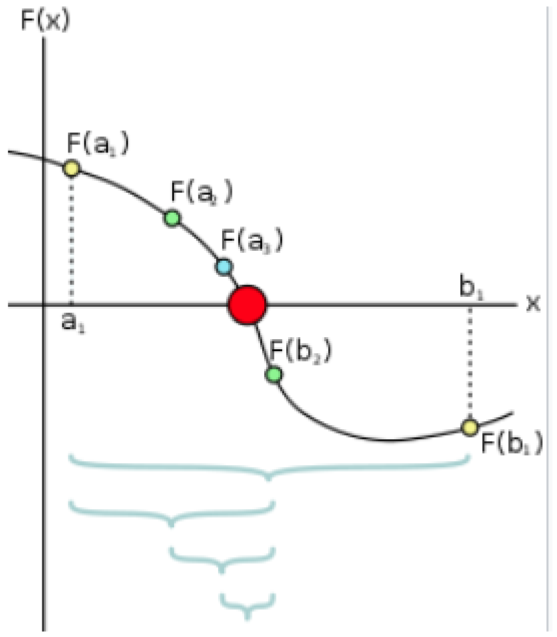
- Your memory can store 16 bits.
- Your algorithm involves some calculations with many variables
- Variables include numbers as large as 30000.
- Your algorithm has an exit condition var1-var2>1e-6
- You need more fixed-point bits than you can handle

Example: Bisection algorithm

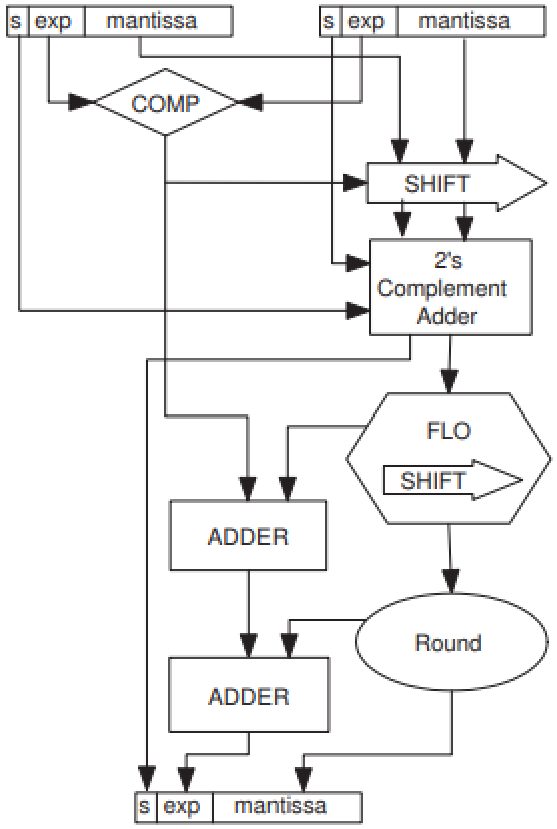
– **The result of f(a) may be large or small – need floating point**



```
N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
  c ← (a + b)/2 # new midpoint
  If f(c) = 0 or (b – a)/2 < TOL then # solution found
    Output(c)
    Stop
  EndIf
  N ← N + 1 # increment step counter
  If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```

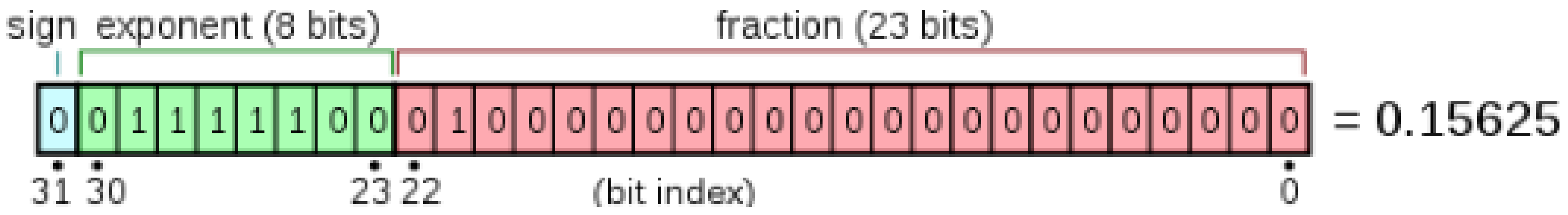


Is floating-point good?

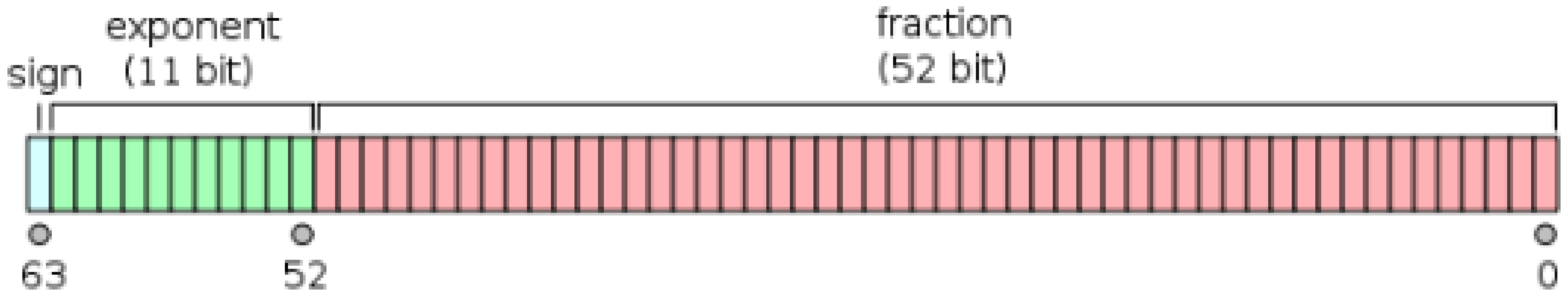


Notes vs IEEE 754 Standard Floating Point

– **Single precision:**

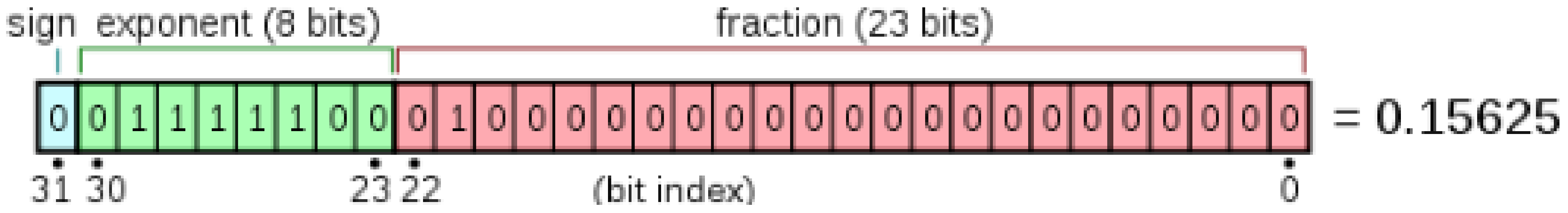


– **Double precision**

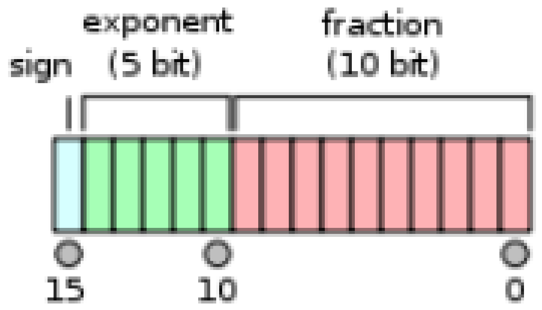


Notes vs IEEE 754 Standard Floating Point

– **Single precision:**



– **Half precision**



Notes vs IEEE 754 Standard Floating Point

- **IEEE standard**
  - Implicit leading 1
  - Special codes:
    - NaN (0/0, sqrt(-10))
    - Infinity (100/0, -100/0)
  - Exponent subtract a bias (127 for single gives range -126 to +127)