

Grade 5/6

Micro:Bits

Any questions?
Reach out
svcamp@engr.uvic.ca

Micro:Bits Binary



Learn the binary number system, boolean logic, and how computers process bits and bytes. Understand what bits and bytes are and how they relate to computers and the way information is processed and stored. Learn to count in Base-2 (binary) and translate numbers from Base-10 (decimal) to binary and decimal. Apply the above knowledge and skills to create a unique program that uses binary counting as an integral part of the program.

Key Words

A **Binary Number** is a number expressed in base-2, a number system that only uses the symbols “0” and “1”.

A **Bit** is a binary digit with two possible values, zero or one

A **Byte** is a sequence of 8 bits and has 256 possible values from 00000000 through 11111111

A **kilobyte (kB)** is 1,024 bytes or 2^{10} bytes

A **Megabyte (MB)** is 1,048, 576 bytes or 2^{20} bytes

A **Gigabyte (GB)** is 1,073,741,824 bytes or 2^{30} bytes

A **Terabyte (TB)** is 1,099,511,627,776 bytes or 2^{40} bytes

Materials

- Micro:bit kit
 - 1 Micro:bit
 - 1 Battery pack
 - 1 Micro USB cable
- Optional: 1 Dongle adapter if using Mac

Micro:Bits Binary



Have any of you bought anything in the last 24 hours?

Did any of you use cash? What bills or coins did you use?

What are the core denominations of money in Canada?

The number system we use for our money in Canada is in base-10, but what does that mean?

- 1 penny
- 1 dime = 10 pennies
- 1 loonie = 10 dimes (or 100 pennies)
- 1 ten dollar bill = 10 loonies
- 1 hundred dollar bill = 10 ten dollar bills

Our money system is based on our number system, the decimal system. The deci- prefix means 'one tenth'. Each place value in the decimal is one tenth of the place value to its left. Each digit has 10 possibilities from 0-9, we call this base-10.

But what about number systems that do not have 10 possibilities per digit?

Activity 1

Students will explore the concept of binary numbers by experimenting with a very odd vending machine that only accepts Base-2 coins and doesn't give change! In the process, students will become familiar with an alternate numbering system, in this case binary (Base-2). Students will learn how binary relates to decimal, and will be able to convert between the two systems.

Activity 1 Continued

Our vending machine (see next page) lists its prices in as decimal values (base-10). Choose one item from the vending machine and decide which binary coins you must use to match the decimal price tag

Remember

- Exact change only
- You can only use one of each coin
- The four binary coins are worth 1, 2, 4 and 8

Example:

I want to purchase the camera for \$10. How can I use my binary coins to match that price?

- In this example, the '8' coin and the '2' coin will add to 10

Now I'm going to mark which coins I've used in binary.

8 value binary coin	4 value binary coin	2 value binary coin	1 value Binary Coin
1	0	1	0

I have marked each coin I used with a '1' and each coin I did not use with a '0'




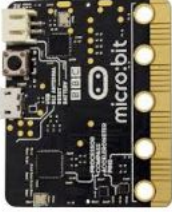



From this table I created, I can now see how to represent the number ten in binary, 1010.

Your turn! Try and represent other base-10 numbers in binary.

Micro:Bits Binary



									
15	14	13	12	11	10	9	8	7	6
Decimal Price \$:									
Binary Payment:									

						
5	4	3	2	1	0	
Decimal Price \$:						
Binary Payment:						

Micro:Bits Binary



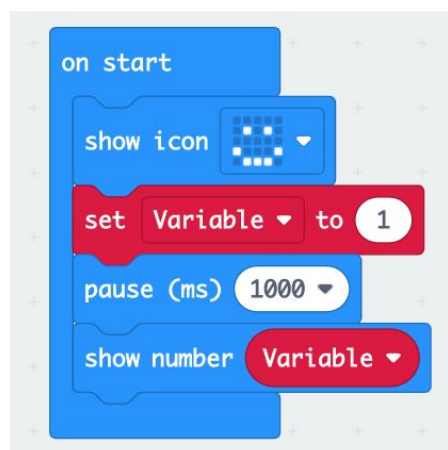
Activity 2

Doubler / Binary Calculator

Create a program which doubles values, just like the order of binary numbers. What do you notice about every number we can generate with this program? All numbers are even and represent a 1 bit with 0s following. Extension: Press B to add numbers to our sum and A+B to show sum and reset the binary counter. Try to only use each bit once and see if you can generate any number you can think of with a unique combination of binary values.

Step-by-Step

- We can start our program by displaying a unique image 'on start'. This is a sign to tell us that our program has started fresh.
- We then want to create a variable so we have a value to perform our calculations. You can name your variable whatever you like, in this case I have kept it listed as 'Variable'.
- Set the variable to 1 to begin, as this is the lowest binary digit containing 1.
- Add a pause so our initial image can be displayed briefly, and then add a 'Show number' block and connect a Variable value block to display our starting variable.
- Now our program has begun! The variable of 1 will display until we introduce our multiplication function.



Micro:Bits Binary

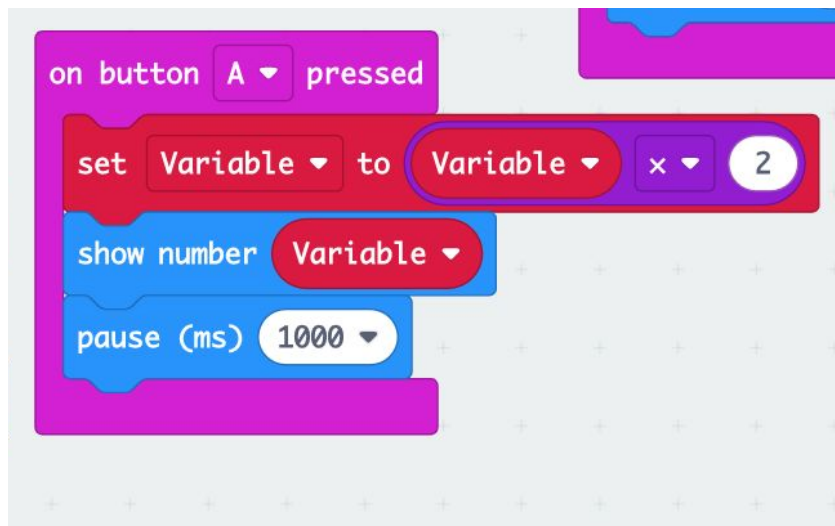


Activity 2 Continued

Multiplication function

- Now we want to implement a way to multiply our binary digit. To begin we use an input block to start our multiplication function. In this case I have used 'On button A pressed'.
- To multiply, we can set our variable to a new value. Take the 'set variable' block from the variables category.
- Look in the Math category to find a mathematical operation value block to attach to 'set variable'. Set the middle dropdown to 'x' multiplication. Type 2 in one half of the operation, and add the variable value block to the other space in the operation.
- Add a 'show number' block to display our new variable value.
- Add a pause as a good coding convention, to limit accidental presses of the A button in quick succession.

Now when we press A, our binary digit is multiplied by 2 and shifted to the left by 1 binary digit. Our decimal value is then displayed on the Micro:bit.



Micro:Bits Binary

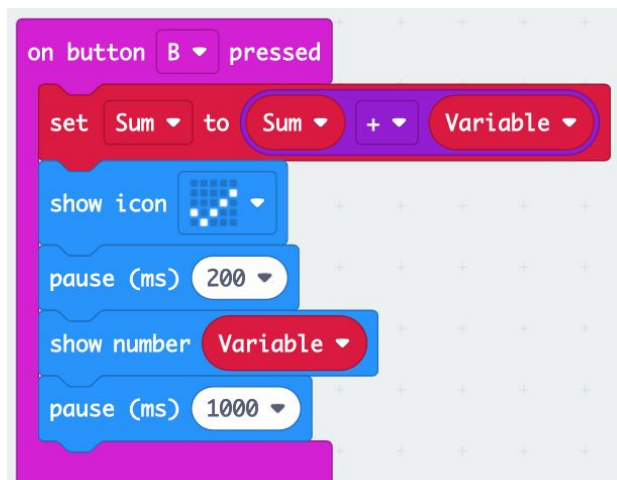


Activity 2 Continued

Sum different binary digit values to create a binary calculator!

- Use another input block to activate our summing function. In this case I have used 'on Button B pressed'.
- Now we create a new variable to sum our values. I have named this variable 'Sum'. Attach a 'set variable' block to our input frame. Be sure that this block is changing our Sum value by selecting 'Sum' in the leftmost dropdown menu.
- Just like in our previous function, we are now going to use a math block to add values. Change the middle operator to a +. Use variable value blocks to place 'Sum' in one half of the operation and 'Variable' in the other.
- We can place a unique icon and short delay after our summing operation to show that we have added to our global sum.
- Finish the function by displaying our current variable just as we have done in the previous functions.

Now when we press B, our Sum is increased by the current value of our variable and we return to the default program state to continue multiplying or adding to our sum.



Activity 2 Continued

Reset / Finish calculation:

- Now we will implement one more input function to show our final sum and reset the program as if we had just started.
- To continue with the button inputs, I have chosen A+B to reset our program. You can use pin inputs if campers wish to try inputs other than buttons.
- Start the reset function by clearing the screen and stopping animations. You can search for these blocks in the search window on top of the block categories. Blocks are titled 'Clear screen' and 'Stop animation'.
- Again we will use an icon to indicate we have begun to reset. I have chosen the same image as my 'on start' to represent a fresh program restart. Add a pause to see the icon displayed for as long as you like.
- Use a 'show string' block to add some text before you reveal your Sum. I wrote "Sum = " to introduce our sum.
- Then show number to display our Sum variable. Use a longer delay to allow your number to show for a while, as the number may become quite large if you have added many binary digits into the sum.
- Now we need 2 'set variable' blocks to reset our variables to their default state. Set "Variable" to 1 and "Sum" to 0.
- Finally, use the 'show number' block one more time to display our starting variable value of 1.
- Now A+B will show our Sum, reset our variables, and start our program over

Micro:Bits Binary



Activity 2 Continued

From previous page's steps

```
on button A+B pressed
  clear screen
  stop animation
  show icon [grid icon]
  pause (ms) 500
  show string "Sum ="
  show number Sum
  pause (ms) 5000
  set Variable to 1
  set Sum to 0
  show number Variable
```

Complete Program

```
on button A+B pressed
  clear screen
  stop animation
  show icon [grid icon]
  pause (ms) 500
  show string "Sum ="
  show number Sum
  pause (ms) 5000
  set Variable to 1
  set Sum to 0
  show number Variable

on start
  show icon [grid icon]
  set Variable to 1
  pause (ms) 1000
  show number Variable

on button B pressed
  set Sum to Sum + Variable
  show icon [grid icon]
  pause (ms) 200
  show number Variable
  pause (ms) 1000

on button A pressed
  set Variable to Variable x 2
  show number Variable
  pause (ms) 1000
```

Micro:Bits Binary



Additional Activity

Binary to decimal transmogriber

<https://makecode.microbit.org/courses/csintro/binary/activity>

The following activity will walk you through the steps to create a binary transmogriber with the micro:bit. The user will be able to use the buttons to enter binary 0s and 1s and will be able to press A+B at any time to display the decimal equivalent of the number that has been entered.

Additional Information

Most everyone who uses a computer has heard the terms, kilobyte (kB), Megabyte (MB), Gigabyte (GB) and even Terabyte (TB), usually when referring to the size of computer files and hard drives as well as download speeds. Bandwidth or connection rates are measured in bits/second. But what is a bit and what is a byte and what do they have to do with computers?

Picture a basic room light. The light is either on or it is off. You control the current state of the light by flipping a switch that has only two settings, down (light off) and up (light on). The earliest computers used a series of mechanical switches to control the flow of electricity through their circuits, turning each one on or off. The on/off states of the circuits was used to represent and even store information. The smallest unit of information, representing the state of one switch, is known as a bit.



Micro:Bits Binary



Additional Information

A bit is a binary digit and has only two possible values, zero or one. The value of the bit represents the current state of a single switch. If the switch is off, then the bit has the value zero. If the switch is on, then the bit has the value one.

A bit can only represent two different values, zero or one. To represent larger pieces of information, bits are strung together in sequences of 8 called bytes.

A byte is a sequence of binary digits made up of 8 bits.

A byte can represent any value from 00000000 through 11111111, for a total of 256 different possible values. Each digit in a byte can be thought of as representing an individual switch that is either off (zero) or on (one).

Modern computers rely on transistors, which pack millions of tiny switches into a chip smaller than your thumb, but information is still represented in essentially the same way: as a series of ones and zeros. By using binary, computers can represent information simply and efficiently using a system that is very effectively modeled in digital circuitry.

