Learning Goals

1) Students will understand the concept of **program memory** and **memory addressing**
2) Students will understand what **variables** are in the C programming language, as well as how they exist in program memory.

Warm-Up

1) Use long multiplication to compute 121×34. First take 30 seconds to try to do this in your head. Then use the space provided for the computation.

Solution

121 and 34 are not nice numbers to multiply and the average person probably can't compute the product in their head. So doing long multiplication should yield:

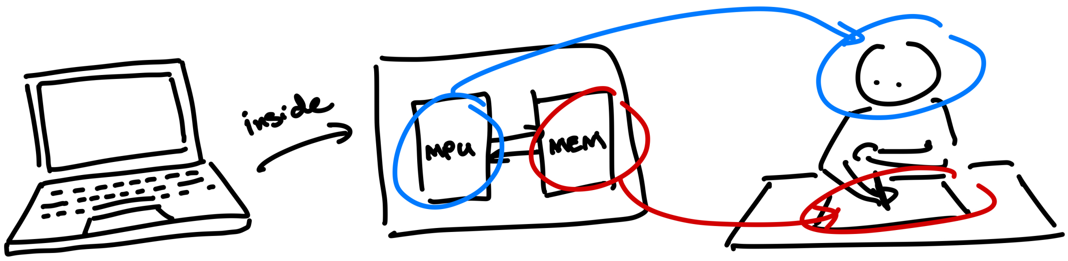
$$\begin{array}{r} 121 \\ \times\quad 34 \\ \hline 484 \\ +\,3630 \\ \hline 4114 \end{array}$$

2) What do you think this exercise illustrates about people's memory?

Solution

Since the average person can't do long division mentally, they probably need to work out the calculation on paper, because it's hard to remember all of the intermediate values you got while going through it.

If this was you, then the paper is acting as your "memory", since it's being used to save the temporary data you need for later in the calculation.



This is actually kind of analagous to a computer. A computer has 2 basic parts: the **processor** and the **memory**. In the analogy, the processor is you, since it's your brain doing the actual calculation part, and the memory is the paper—the place you store data when you're not using it. That would be the intermediate values 484 and 3630 from the exercise.

In the same way, the computer uses its memory to store information and data when it's not being actively used by the processor. Understanding the way that data is stored in memory is crucial when you want to start writing what are called "low-level"programs, which are called such because they exercise more direct control over where data is located in memory.

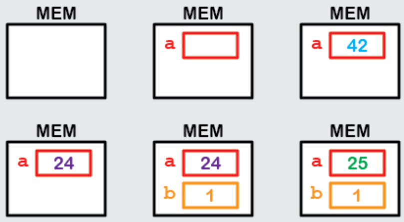
1 Variables

Some helpful definitions:

code:	the instructions given to a computer that tell it what to do.
	a very loose term ("C code", "Python code", "machine code")
variable:	a named container for a value, similar to an algebraic variable

Learn by Example

```
>>> int a;          // variable declaration
>>> a = 42;         // variable assignment
>>> a = 24;         // variable reassignment
>>> int b = 1;      // declaration + assignment
>>> a = a + b;      // operations
```



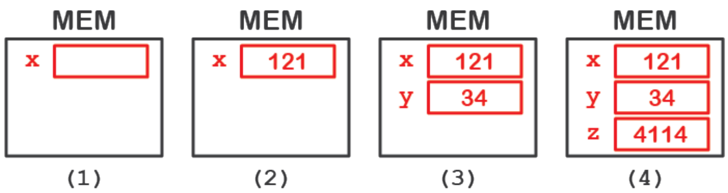
When you declare a variable, the computer sets aside some space in memory that can be used to store variables. How much space is determined by the **variable type**.

Different Data Types

```
>>> int i = 0;      // declare an integer variable i and store 0
>>> char a = 'a';   // declare a character variable a and store 'a'
>>> bool t = true;  // declare a boolean variable t and store true
```

Practice

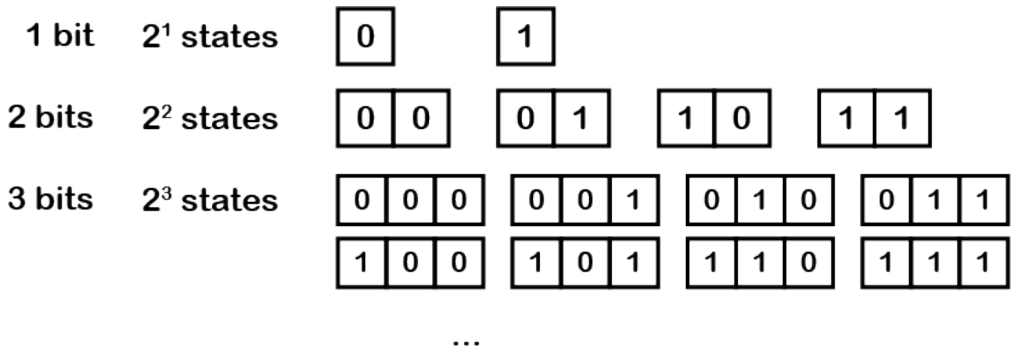
```
>>> int x;              (1)
>>> x = 121;            (2)
>>> int y = 34;         (3)
>>> int z = x * y;      (4)
```



2 Variables in Memory and Memory Organization

To understand how variables actually exist in memory, we need to understand how computer memory is organized.

Computer memory is made up of a whole bunch of units called **bits**. A bit can hold only one of 2 possible values: a *1* or a *0*. When we group bits together, we can store more values by encoding each value as a permutation of *1's* and *0's*. For example, when we have 2 bits, there are 4 possible permutations of *1's* and *0's*, so we can store 4 possible values; when there are 3 bits, there are 8 permutations, and so on and so forth.



Question: How many possible values can n bits store?

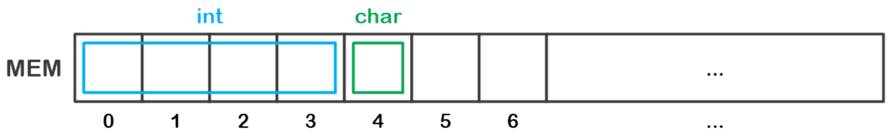
Solution

Each bit has 2 possible states, so the total number of permutations is 2^n. That means n bits can store 2^n possible values.

For historical reasons, computer memory is chunked into 8-bit units called **bytes**, so when we organize memory, we organize it by bytes.

You can think of computer memory as a very long chain of contiguous bytes, each of which are numbered with an index. This index is called the **memory address**. Programs can use these addresses to access particular bytes of data in memory.

As already mentioned, when a variable is declared, the computer sets aside some amount of memory to contain it, and the amount is determined by the variable type. Variable types have fixed memory sizes that the computer will allocate, for instance, an `int` type variable is usually 4 bytes, while a `char` variable is 1 byte. If a variable is more than 1 byte, the bytes that the computer allocates must be contiguous.



Practice

How many total bytes of memory are being allocated in following lines of code? (assume `int`'s are 4 bytes and `long`'s are 8 bytes)

```
>>> int a = 0;
>>> int b = 17;
>>> a = a + b;
>>> char c;
>>> long long x;
>>> x = a + b;
>>> short y;
```

Typical Datatype Sizes

`char`	1 byte
`short`	2 bytes
`int`	2 or 4 bytes
`long`	4 or 8 bytes
`long long`	8 bytes

Solution

Counting each variable declaration, we have 2 `int`s, 1 `char`, 1 `long long`, and 1 `short`. Using the size convertions, we get:

$$2(4) + (1) + (8) + (2) = 19 \text{ bytes}$$