Relational Operators Also Depend on Data Type

Declare:
```
int A = -120;/* 0xFFFFFF88 */
int B =  256;/* 0x00000100 */
```

Is **A < B**?

- Yes, -120 < 256.
- But if the same bit patterns were interpreted using the **unsigned** representation,

```
0xFFFFFF88 > 0x00000100
```

As with shifts, a **C** compiler **uses the data type to perform the correct comparison**.

The Assignment Operator Can Change a Variable's Value

The **C** language uses **=** as the **assignment operator**. For example,

```
A = 42
```

changes the bits of variable **A** to represent the number **42**.

One can write **any expression on the right-hand side of assignment**. So

```
A = A + 1
```

increments the value of variable **A** by **1**.

Assignment Calculates an Expression, then Writes Bits

The code for an assignment

1. **calculates the expression**, then
2. **writes the result to the address** for the left-hand side.

For example, given

```
A = B + C
```

A compiler produces something akin to this code.

```
LD R0,B
LD R1,C
ADD R0,R0,R1
ST R0,A
```


Assignments Write to Memory Addresses

A **C** compiler can not solve equations.

For example,

```
A + B = 42
```

results in a compilation error (the compiler cannot produce instructions for you).

The left-hand side of an assignment must have an address.

An expression with an address is called an **l-value**. **Variables are l-values.**

