Declare Variables Based on Our Approach

First, we need a couple of variables…

```
uint32_t reverse_bits (uint32_t arg)
{
    int32_t i; // iteration counter
    uint32_t result = 0;
    ...
```

What about the "original" value?

Just use `arg`!

One `for` loop Covers the Reversal Algorithm

Now for the loop…

```
    for (i = 0; 32 > i; i++) {
        result = ((result << 1) |
                  (arg & 1));
        arg >>= 1;
    }
```

(By adding two shifts, we avoid writing a second bit copy operation outside the loop.)

`result` Has the Reverse Bits

All that remains is to return the reversed bits.

```
    ...
    return result;
}
```


What Happens to `reverse_bits`' Changes to `arg`?

Let's write a call…

```
uint32_t x = 42;
uint32_t x_rev;
x_rev = reverse_bits (x);
```

`reverse_bits` changes the value of `arg`!

So does x change value?

No. C uses call by value.

