## Shift and Copy One Byte at a Time

```
int32_t ntohl (int32_t arg)
{
    int32_t res;
    res = (arg << 24);
    res |= ((arg & 0xFF00) << 8);
    res |= ((arg >> 8) & 0xFF00);
    res |= (arg >> 24);
    return res;
}
```

arg: 0x80202425

res: 0x25242000

So far, so good!

  

29

## Shift and Copy One Byte at a Time

```
int32_t ntohl (int32_t arg)
{
    int32_t res;
    res = (arg << 24);
    res |= ((arg & 0xFF00) << 8);
    res |= ((arg >> 8) & 0xFF00);
    res |= (arg >> 24);
    return res;
}
```

arg: 0x80202425

res: 0xFFFFFF80

Er … is that right?

  

30

## Shift and Copy One Byte at a Time

```
int32_t ntohl (int32_t arg)
{
    int32_t res;
    res = (arg << 24);
    res |= ((arg & 0xFF00) << 8);
    res |= ((arg >> 8) & 0xFF00);
    res |= (arg >> 24);
    return res;
}
```

We told the compiler that we wanted an arithmetic right shift.

  

31

## A Correct Variant of `ntohl`

```
uint32_t ntohl (uint32_t arg)
{
    uint32_t res;
    res = (arg << 24);
    res |= ((arg & 0xFF00) << 8);
    res |= ((arg >> 8) & 0xFF00);
    res |= (arg >> 24);
    return res;
}
```

Note use of `uint32_t`.

  

32