Binary and Hexadecimal System

Decimal System
237 = 2 * 10^2 + 3 * 10^1 + 7 * 10^0
Conversion: Base 2 → Base 10
11010010 =
1*2^7 + 1*2^6 + 0*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 0*2^0
= 210
Conversion: Base 10 → Base 2
145 | → divide 145:2=72 with remainder of 1...
72 | 1 (least significative digit)
36 | 0
18 | 0
9 | 0
4 | 1
2 | 0
1 | 0
0 | 1 (most significative digit)

145 in base 10 corresponds to 10010001 in base 2
with N bits we can represent all integer numbers
between 0 and 2^N-1

DEC 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
HEX 0 1 2 3 4 5 6 7 8 9 A B C D E F
Hex Bin

0 0000
1 0001
2 0010
3 0011
4 0100
5 0101
6 0110
7 0111
8 1000
9 1001
A 1010
B 1011
C 1100
D 1101
E 1110
F 1111

Conversion: Base 16 → Base 2:
Hexadecimal number 3F5 corresponds
to 0011 1111 0101

Conversion Base 2 → Base 16:

10 0101 1101 0101 0010

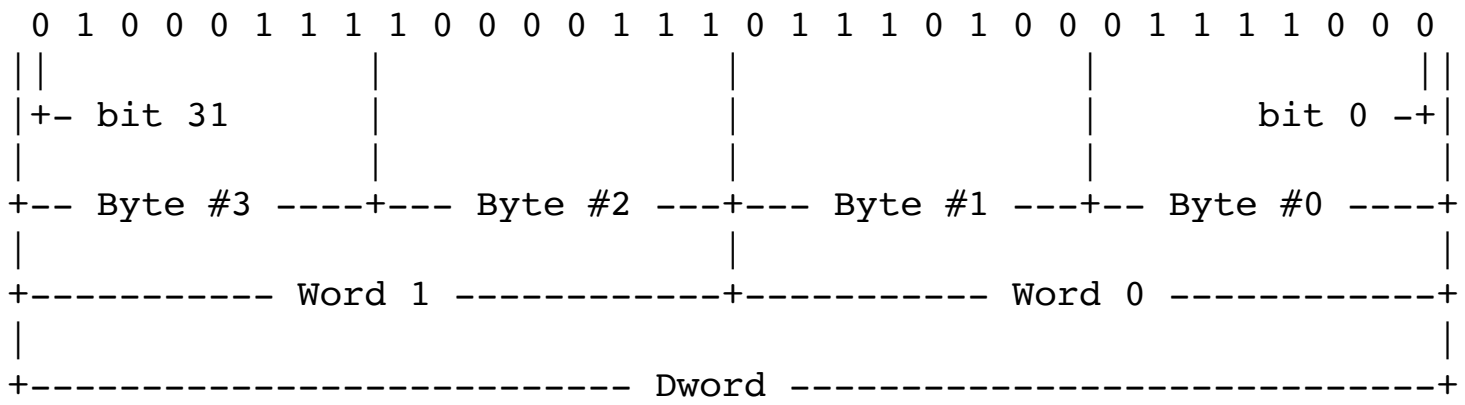
0010 → 2
0101 → 5
1101 → D
10 → 2

0010 0101 1101 0101 0010 corresponds to
2 5 D 5 2

Bitwise Operators

Bitwise operators work on binary data

1 Byte = 8 bits; 1 Word = 2 Bytes = 16 bits; Dword = 2 Words = 32 bits



These are the 6 most important Bitwise operators:

& **AND operator**
| **OR operator**
^ **XOR operator**
~ **Ones Complement (Bitwise Not)**
>> **Right Shift operator**
<< **Left Shift operator**

Bitwise Operators

```
AND
 1   &   1   ==   1
 1   &   0   ==   0
 0   &   1   ==   0
 0   &   0   ==   0

  00110010
& 00010000
 ----------
  00010000
```

```
OR
 1   |   1   ==   1
 1   |   0   ==   1
 0   |   1   ==   1
 0   |   0   ==   0

  00110010
| 00000100
 ----------
  00110110
```

```
XOR
 1   ^   1   ==   0
 1   ^   0   ==   1
 0   ^   1   ==   1
 0   ^   0   ==   0

  00110010
^ 00011000
 ----------
  00101010
```

```
Bitwise NOT
~ 1  ==   0
~ 0  ==   1

~ 00000011
 ----------
  11111100
```

```
LEFT SHIFT

 00001100 << 2 (decimal)
----------
 00110000
```

```
RIGHT SHIFT
 00001100 >> 2 (decimal)
----------
 00000011
```

```
RIGHT AND LEFT SHIFT
(be careful...)
 00001111 >> 2 (decimal)
----------
 00000011 << 2 (decimal)
----------
 00001100
```