Bitne operacije:

Dec	HEX	BIN
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Pretvorba bin – dec – HEX

Pri programiranju heksadecimalna števila zapisujemo kot npr. 0xABC, tak zapis pozna večina prevajalnikov (v literaturi se jih včasih sicer označuje tudi z $ ali s h), nekateri prevajalniki pa podpirajo tudi zapis bitnih števil kot npr. 0b010110.

Izračun vrednosti hex <-> bin:
1 HEX mesto predstavlja 4 bin mesta.

$0x4B_{16}$:
gledamo vsako mesko posebej: $'4'_{16}$ ='0100'$_2$ $'B'_{16}$='1011'$_2$
$0x4B_{16}=0b01001011_2$

$0b11011_2$
gledamo po štiri mesta, spredaj dodamo enke kolikor je potrebno:
$'0001'_2$ = $'1'_{16}$, $'1011'_2$=$'B'_{16}$
$0b11011_2=0x1B_{16}$

Splošno desetiško > binarno:
Število delimo z npr. 2 (če ga hočemo pretvoriti v dvojiški sistem), ostanek je zadnji bit (mesto), s količnikom računamo naprej, dokler nam ne ostane 0...

44/2 = 22 + 0 ost.
22/2 = 11 + 0 ost
11/2 = 5 + 1 ost
5/2 = 2 + 1 ost
2/2 = 1 + 0 ost
1/2 = 0 + 1 ost

Preberemo navzgor in dobimo binarno številko $44_{10} = 101100_2$

Bin > dec
543210 (številka mesta – eksponent)
101100: 2^5 + 2^3 + 2^2 = 32+8+4 = 44
$101100_2=44_{10}$

Bin>Hex
44/16= 2 + 12 $'2'_{10}='2'_{16}$ $'12'_{10}='C'_{16}$
$44_{10}=2C_{16}$

Hex < Bin
2C: 16 * 2 + 12 = 44 $'2'_{16}='2'_{10}$ $'C'_{16}='12'_{10}$
$2C_{16}=44_{10}$

Bitne operacije:

& - AND operacija, enke v rezultatu so kjer so v obeh operandih enke
| - OR operacija, enke v rezultatu so kjer je pri katerem koli operandu enka
~ - NOT operacija, enke in ničle se zamenjajo
^ - XOR operacija,

```
spremenljivka += 5;
// je ekvivalentno
spremenljivka = spremenljivka + 5;
```

A = 0101
~A : 1010
Negirana vrednost

A = 0101
A &= 0011
A : 0001
Kjer so bile ničle v drugem operandu smo postavili mesta na nič (prvi dve mesti smo postavili na nič)

A = 0101
A &= ~1100
A : 0001
Prvi dve mesti postavimo na 0

A = 0101
A |= 0011
A : 0011
Kjer so bile enke v drugem operandu smo postavili mesta na ena (zadnji dve mesti smo postavili na 1)

A = 0101
A ^= 0011
A : 0110
Zadnji dve mesti smo negirali

Bit shift operaciji (premik mest):

<<n in >>n, deljenje oz. množenje z 2^n

A>>0 = A	0101>>0 = 0101
A>>1 = A/2	0101>>1 = 0010
A>>2 = A/4	0101>>2 = 0001
A<<1 = A*2	0101<<1 = 1010
A<<2 = A*4	0101<<2 = 10100

Premik mest uporabljamo kadar nas zanimajo neprva mesta:
VHOD=00101010 (nas zanimajo le prva stiri mesta)
VHOD>>2=1010 (mesta ki nas zanimajo)

Bit masking:
Uporabimo operator AND. Bit masking uporabljamo kadar nas zanimajo le določena mesta, druga bi pa radi odstranili:
VHOD =10101010
VHOD& =00111100
VHOD: 00101000

Seveda lahko obe funkciji, shifting in masking združimo (tako se to ponavadi tudi uporablja), masking ponavadi uporabimo za »ignoriranje« oz. odstranjevanje zgornih bitov:
VHOD =10101010
VHOD >> 2 =101010
(VHOD >> 2) & 001111 = 1010

Masking lahko uporabimo tudi za preverjanje prisotnosti enega bita
VHOD =10101010
VHOD&10000000 =10000000
VHOD&01000000 =00000000

If (VHOD&0b10000000) – stavek se bo izvedel izraz ni enak nič
If (VHOD&0b01000000) – stavek se ne bo izvedel, izraz je enak nič

Stavka ekvivalentna zgornjim:
If (VHOD&(1<<7)) – stavek se bo izvedel izraz ni enak nič
If (VHOD&(1<<6)) – stavek se ne bo izvedel, izraz je enak nič