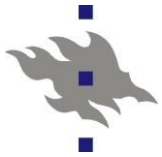
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

C-ohjelmointi: Bittioperaatiot

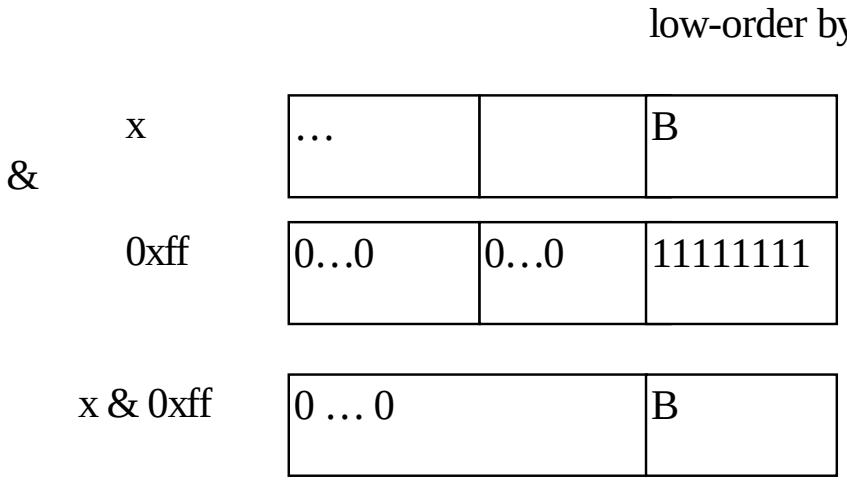
Viikko 4

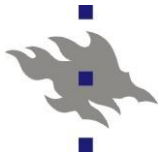


Bittioperaatiot

- & | ^ vertaavat alkioita bitti kerrallaan ja palauttavat tämän tuloksen
- << >> siirtävät bittejä sanan sisällä
- ~ yhden komplementti 0-> 1 ja 1-> 0

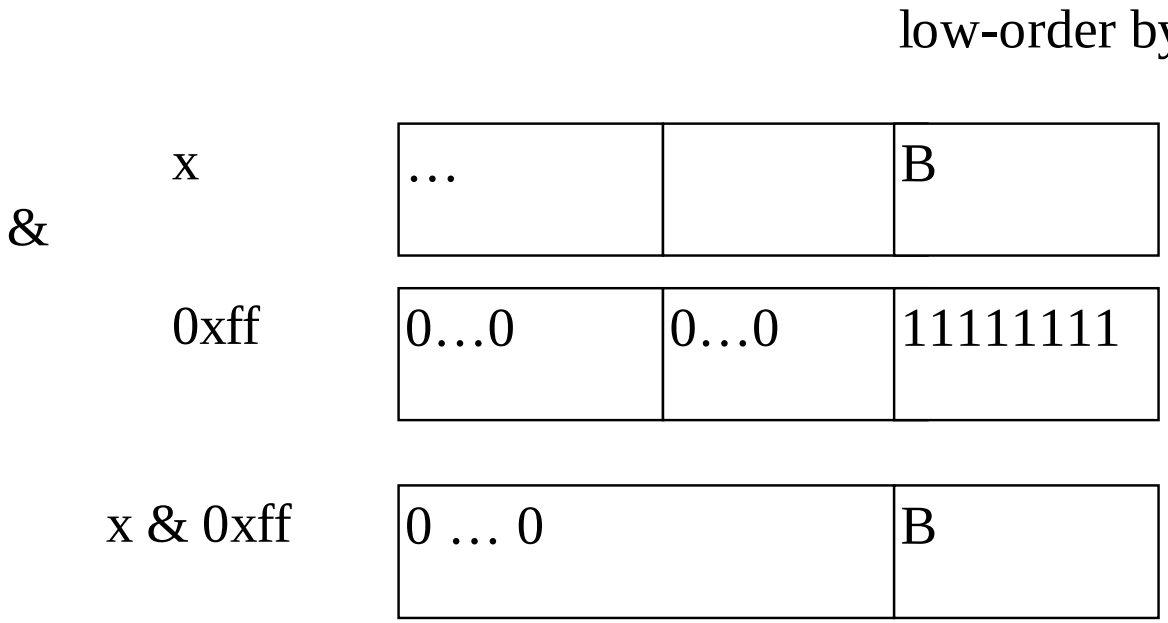
&	bitwise and
|	bitwise or
^	bitwise xor (exclusive or)
<<	left shift
>>	right shift
~	one's complement

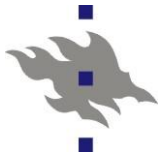




`&` (bitti and)

- jos $(x)_i$ == 1 ja $(y)_i$ == 1 niin $(x \& y)_i$ == 1
- muuten $(x \& y)_i$ == 0
- Tätä käytetään usein bittien nollaukseen, esim:
- `x & 0xff` nollaa sanan kaikki muut bitit paitsi vähiten merkitsevän tavun (low-order byte):



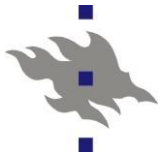


| (bitti or)

- jos $(x)_i$ == 1 *tai* $(y)_i$ == 1 niin $(x \mid y)_i$ == 1
- muuten $(x \mid y)_i$ == 0
- Tällä usein asetetaan tiettyjä bittejä ykkösiksi.
- Asetetaan x:n alimman tavun ylin bitti ykköseksi:
  `x | 0x80`,

low-order byte

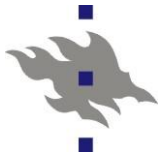
x	C	$B_1B_2B_3\ldots$
|		
0x80	0…0	10000000
x | 0x80	C	$1B_2B_3\ldots$



^ (bitti xor)

- jos $(x)_i$ == $(y)_i$ niin $(x \hat{} y)_i == 0$
- muuten $(x \hat{} y)_i == 1$

- Bittioperaatio ^ (*xor,* poissulkeva or) nollaa ne bittipositiot, joiden arvo on sama ,
- ja asettaa ykköseksi ne bittipositiot, jotka olivat eriarvoiset.
- Yksinkertainen testi sanojen samuudelle: **`x ^ y`**
- palauttaa 0 jos **`x`** ja **`y`** samat (eli kaikki bitit vastasivat toisiaan).

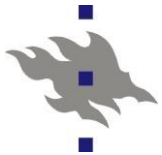


~ (yhden komplementti)

- **Yhden komplementti**: **~x**

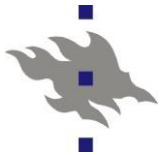
  $(\sim x)_i == 1 - (x)_i$

- Bitit vaihtuvat vastakkaisiksi.
- Esimerkiksi. Halutaan vain nollata sanan **x** 3 alinta bittiä ja jättää loput ennalleen: **x & ~7**
  (Niin miten tuo oikein toimii?)
- Vastaavasti, kun halutaan asettaa kaikki bitit ykkösiksi **~0**



<< Siirto vasemmalle

- **Left shift**: `i << j`
- Tuloksena on sana, jossa bitit ovat siirtyneet vasemmalle `j` paikkaa
- Sana on täydennetty oikealta tarvittavalla määrällä nollia.
- Tämä on näppärä kahden kertolaskuissa:
- `x <<= 1` on sama kuin `x *= 2`
- `x <<= 3` on sama kuin `x *= 8`.



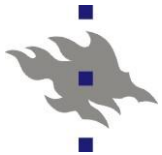
>> Siirto oikealle

- **Right shift**: `i >> j`

- Tuloksena on sana, jossa bitit ovat siirtyneet `j` paikkaa oikealle.
- Etumerkittömillä tyypeillä (ja positiivisilla luvuilla) oikealta täydennetään aina nollilla.
- Etumerkillisillä ja negatiivisilla luvuilla täydennyksenä voi olla 0 tai 1 toteutuksesta riippuen. (MIKSI?)

- `x >>= 1` is equivalent to `x /= 2`

- `x >>= 2` is equivalent to `x /= 4`.



Esimerkki:

```
#include <stdio.h>
/* Bittipeliä*/
int main(void)
{
  enum {LL = 011 };
  int i, j;

  i = 0;
  j = i | LL;
  printf(”i: %d, LL (okt):%o, i|LL: %d, oktaalina %o\n”,
            i, LL, j, j);

  printf(”1 & 6: %d, 1 && 6: %d\n”,
            1 & 6, 1 && 6);

  printf(”1<<3: %d, 8>>3: %d\n”,
            1<<3, 8>>3);
return 0;
}
```

Tulostaa:
i: 0, LL (okt):11, i|LL: 9, oktaalina 11
1 & 6: 0, 1 && 6: 1
1<<3: 8, 8>>3: 1