Bit and Byte

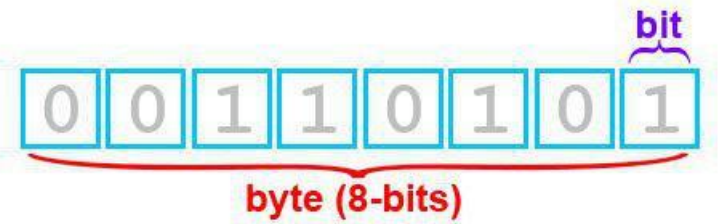
okee... sebelumnya kenapa sih kok ngangkat topik ini....?

padahal kan angka, angka ya angka... string ya string... dst...

tapi ternyata enggak loh.... dalam memory semua berupa angka....dengan satuan terkecil berupa bit yang hanya bisa menampung nilai 0 dan 1 saja...

jadi semua terdiri atas 0 dan 1..... pada bagian dasar...

tapi kalo ngoding pake binari (basis 2, bit ini) ya pusing pak....makanya bit-bit ini dikumpulkan jadi jadi per-delapan bagian untuk membentuk satu byte...



nah dalam satu byte ini bisa menampung nilai mulai dari 0 sampai 255...

kok bisa... si...? padahal kan cuman 0 dan satu doang.....

kombinasi antara angka 0 dan 1 pada satu byte ternyata memberikan nilai yang berbeda...

kalo pake 8 bit saya rasa bakal kepanjangan kalo di-list kombinasinya...

dan juga tidak sangkil dan mangkus (efektif dan efisien, bahasa baku),, jadi pake 4 bit aja...

karena kombinasi bit value-nya ya bakal itu-itu saja....

kalo yang ada source-nya bisa dibuka file dengan nama (un)signed....dan dikompilasi dan dijalanin...

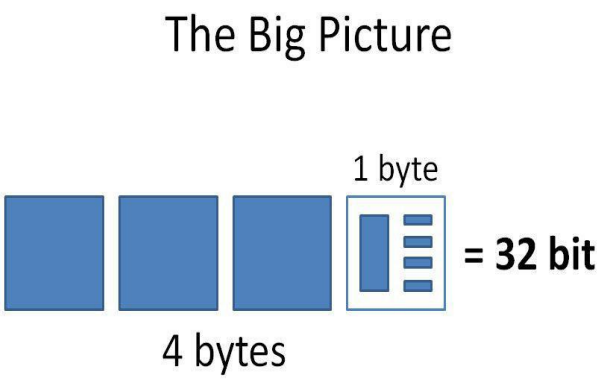
```
index  bit value unsigned signed  hexadecimal
    0.      0000        0      0          0x0
    1.      0001        1      1          0x1
    2.      0010        2      2          0x2
    3.      0011        3      3          0x3
    4.      0100        4      4          0x4
    5.      0101        5      5          0x5
    6.      0110        6      6          0x6
    7.      0111        7      7          0x7
    8.      1000        8     -8          0x8
    9.      1001        9     -7          0x9
   10.      1010       10     -6          0xA
   11.      1011       11     -5          0xB
   12.      1100       12     -4          0xC
   13.      1101       13     -3          0xD
   14.      1110       14     -2          0xE
   15.      1111       15     -1          0xF
```

nah jadi dalam 4 bit bisa menampung nilai 0-15...

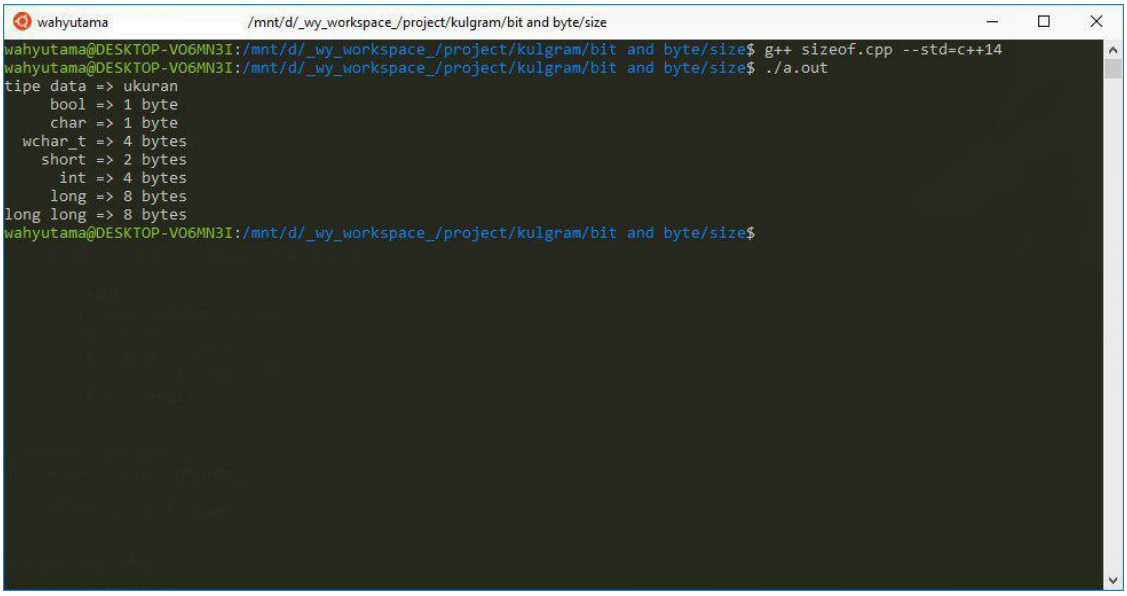
eh itu kok ada signed dan unsigned? nanti yah.... bakal dibahas juga kok

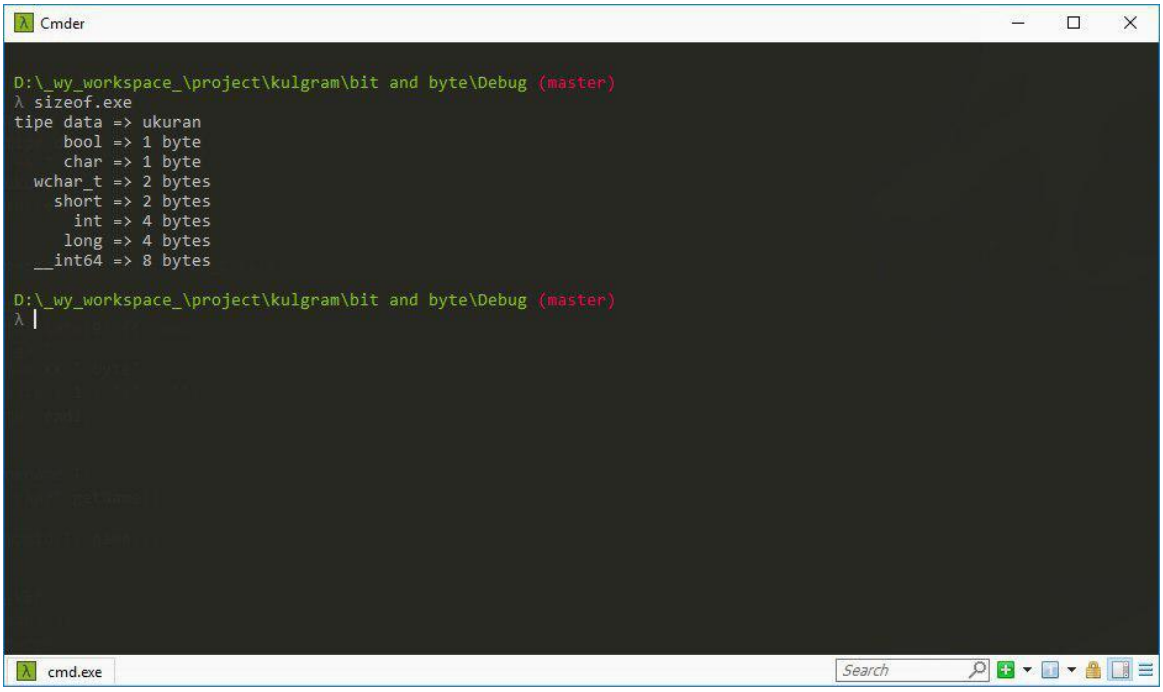
oia kelupaan, 4 bit bisa juga disebut 1 nibble...

sampai sini gimana?



kurang lebih gambarannya seperti itu.... dalam sebuah int (32 bit) jadi gimana kalo tipe data yang lainnya...? kan tipe data fundamental juga beragam, dan bisa digunakan sesuai kebutuhannya,,,

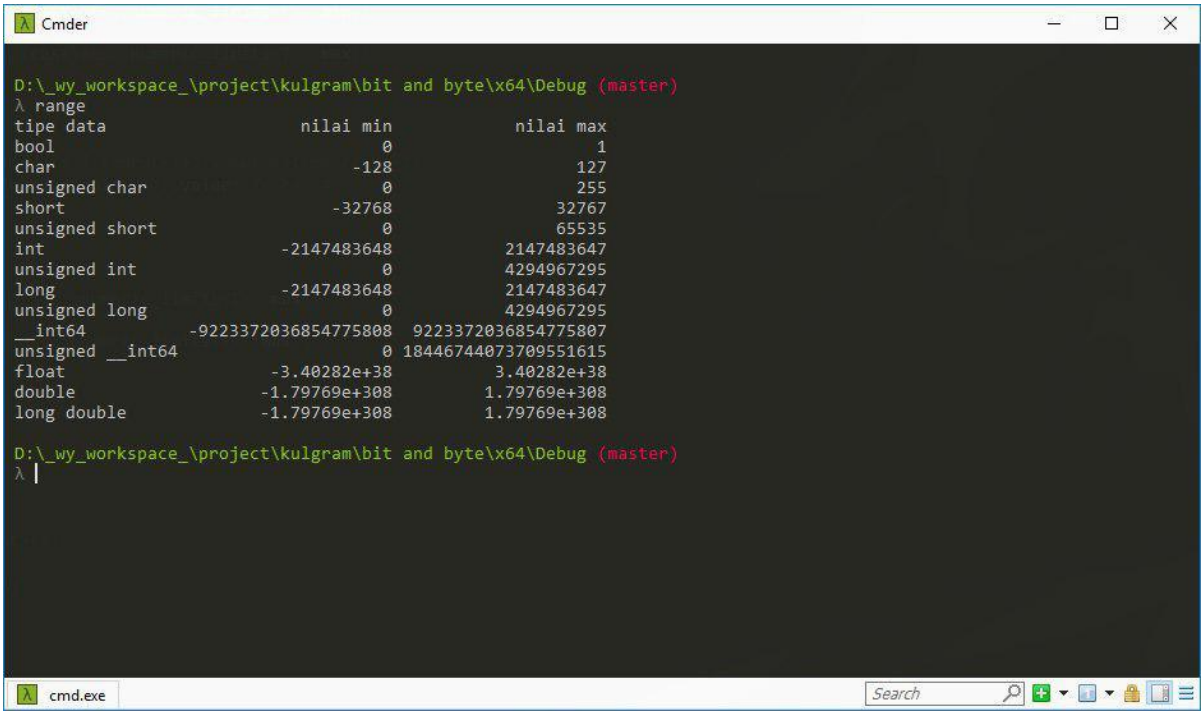




nah.. itu ada tipe data yang bisa dipake ketika ngoding... ada nilai max yang bisa ditampung... kalo kelebihan (yang ditampung) ya harap disesuaikan (cari tipe data lain)..

oia ada beda tipe data long antara msvc dengan g++...

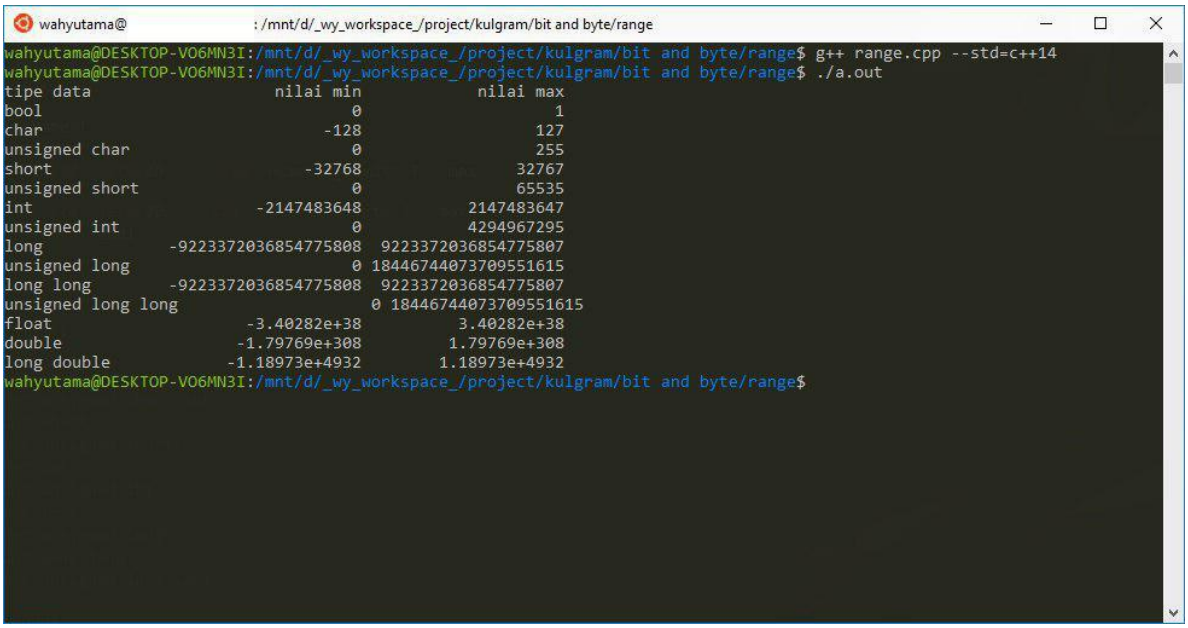
eh source code-nya yang sizeof.cpp...



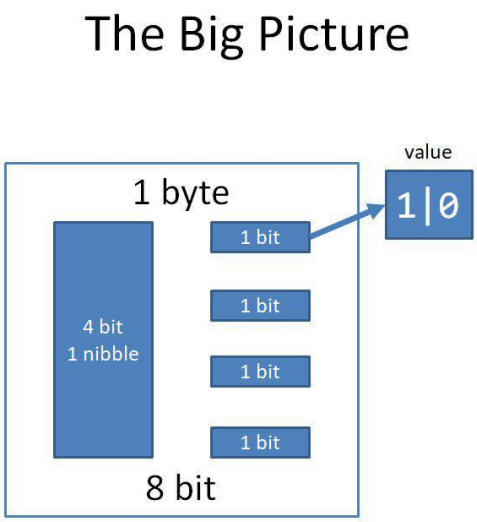
nah kalo gambar ini, nunjukin berapa aja nilai yang bisa masuk dalam sebuah tipe data...

*catatan msvc: __in64 == long long

*dan long double-nya sama dengan double... beda dengan gcc....

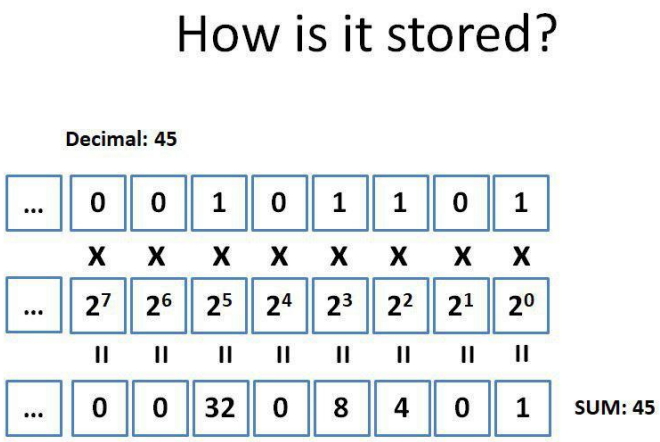


oke balik lagi ke 4 bit yak.... gambarannya kayak gini kalo diilustrasikan...



sekarang gimana si cara nyimpennya...

saya ga akan bahas cara ngitungnya yang njelimet....



jadi misalnya angka 45, dia bakal kesimpen dengan sequence yang seperti itu... compilernya udah cukup canggih kok.... kasih aja desimal 45, udah disimpen sendiri....

Oke.. terus apa coba manfaatnya kalo kompiler udah "doing the work for us" dan orang yang make malah melajarin yang beginian....?

pernah tau kan yang namanya ada flag dalam file header bawaan...

misalnya:

#define BAPER 0x01

#define CAPER 0x02

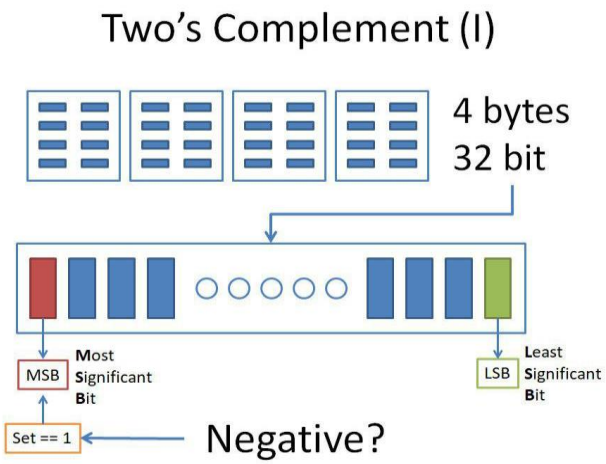
... dst...

nah salah satunya disitu....

kita ngeset nlai dari bit yang ada dalam sebuah tipe data.... jadi alih-alih make banyak variabel dan bikin mess labih baik pake satu aja.... tapi bisa menampung sejumlah bit yang ada dalam satu tipe data... misalnya int bisa nampung 32 bit (artinya 32 nilai) dst...

Terus pasti muncul pertanyaan... lha kalo nilai negatif gimana coba...?

jawabannya ada di Two's complement....



nah disitu ada MSB dan LSB... pernah tau kan... ? atau pernah denger kan...? (lha ini tau...)

jadi nilai negatif itu bila bagian MSB (most significant bit) bernilai satu...

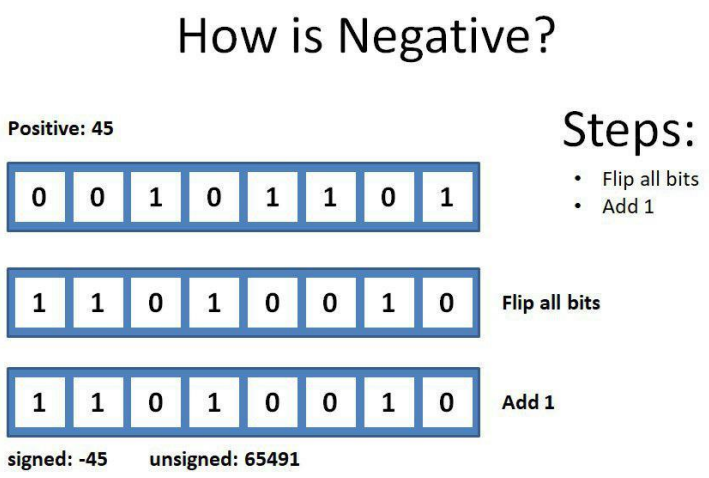
balik kesini...

index	bit value	unsigned	signed	hexadecimal
0.	0000	0	0	0x0
1.	0001	1	1	0x1
2.	0010	2	2	0x2
3.	0011	3	3	0x3
4.	0100	4	4	0x4
5.	0101	5	5	0x5
6.	0110	6	6	0x6
7.	0111	7	7	0x7
8.	1000	8	-8	0x8
9.	1001	9	-7	0x9
10.	1010	10	-6	0xA
11.	1011	11	-5	0xB
12.	1100	12	-4	0xC
13.	1101	13	-3	0xD
14.	1110	14	-2	0xE
15.	1111	15	-1	0xF

dengan adanya two's complenent seorang programmer akhirnya bisa nyimpen nilai negatif... Horeeeeee....

sebelumnya ada juga sistem one comlement... (tapi ada kekurangannya... nilai 0 ada dua..)

kalo cara nyarinya...?



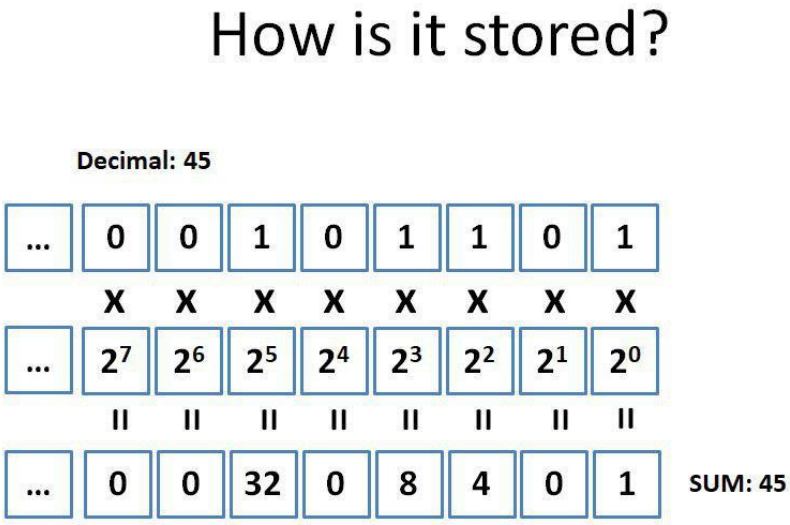
1. bit-nya diflip

2. ditambah satu....

nah sekarang tergantung programmernya mau pake tipe data unsigned atau signed... unsigned itu artinya enggak ada nilai negatifnya.... unsigned itu artinya enggak ada nilai negatifnya....

kalo pake tipe data signed maka programmer dapat nilai negatif... tapi rentang nilai postifnya kepotong... (trade-off lah...) bisa juga kalo mau di-cast antara unsigned dan signed....

sampe sini...?



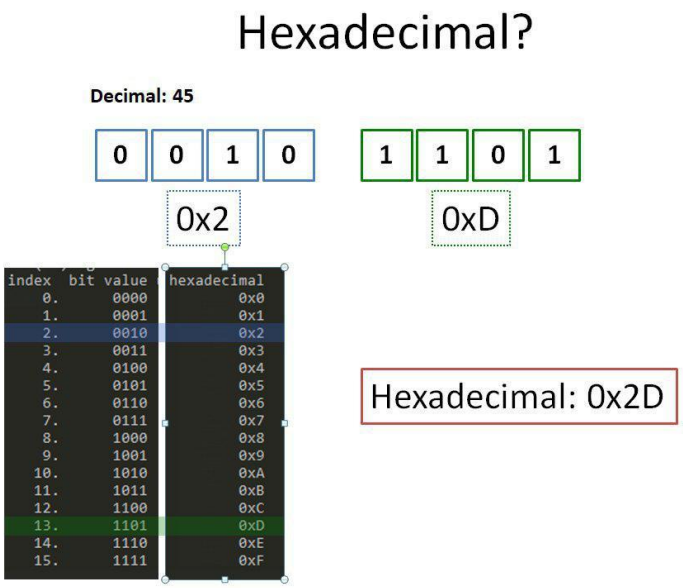
pasti ribet lah ya ngonversi dari desimal ke bit kalo pas butuh.... hexadesimal to the resque...

nah apa pula itu...?

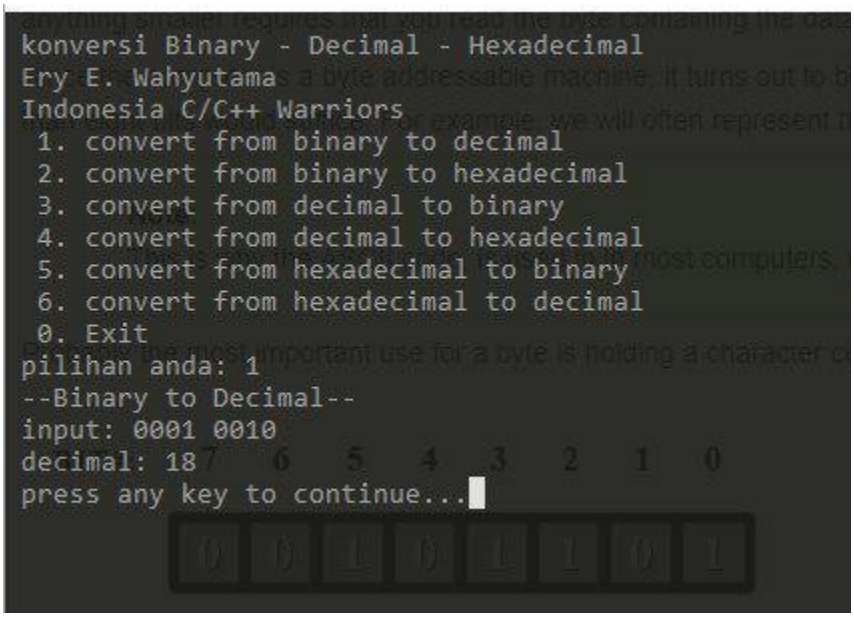
sistem dengan 16 bilangan 0-9 kemudian dilanjut A-F eh apa malah ga ribet...? yang desimal aja udah ribet.....? ternyata enggak loh...

sering kan liat dalam header2 standar ada nilai yang diawali dengan 0x.... nah itu dia hexadesimal....

tar... mouse-nya error....



konversinya cukup mudah malah kalo dari heksa desimal ke binari.... nggak perlu perkalian cukup di-look-up aja.... kalo mau nyoba-nyoba konversi bisa dibuka dan dikompilasi source yang conversion....



misalnya kayak gitu,,, yang dari hexa bisa dikasih prefix 0x lo.... atau malah bisa pake calculator bawaan windows kalo di sistem Windows....nah... ini mousenya error... saya pake pc,,,

jadi saya cukupin sampe sini yak.... maap yak kalo enggak sempurna....