

Bit and Byte

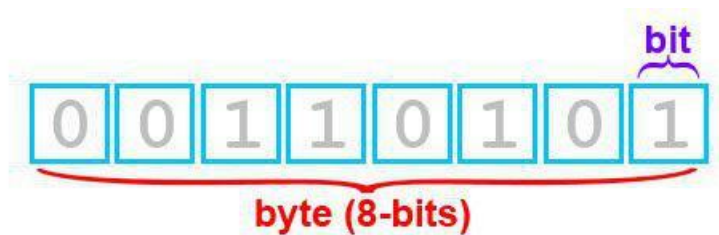
okee... sebelumnya kenapa sih kok ngangkat topik ini....?

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 dijalankan...

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

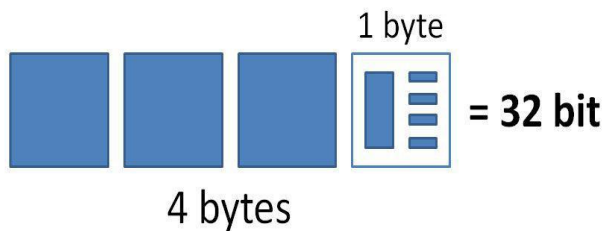
nah jadi dalam 4 bit bisa menampung nilai 0-15...

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?

The Big Picture



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,,,

```
wahyutama /mnt/d/_wy_workspace_/project/kulgram/bit and byte/size
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/size$ g++ sizeof.cpp --std=c++14
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/size$ ./a.out
tipe data => ukuran
bool => 1 byte
char => 1 byte
wchar_t => 4 bytes
short => 2 bytes
int => 4 bytes
long => 8 bytes
long long => 8 bytes
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/size$
```

```
Cmder

D:\_wy_workspace_\project\kulgram\bit and byte\Debug (master)
λ sizeof.exe
tipe data => ukuran
bool => 1 byte
char => 1 byte
wchar_t => 2 bytes
short => 2 bytes
int => 4 bytes
long => 4 bytes
__int64 => 8 bytes

D:\_wy_workspace_\project\kulgram\bit and byte\Debug (master)
λ
```

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...

```
Cmder

D:\_wy_workspace_\project\kulgram\bit and byte\x64\Debug (master)
λ range
tipe data          nilai min          nilai max
bool               0                  1
char               -128                 127
unsigned char      0                  255
short              -32768                32767
unsigned short     0                  65535
int                -2147483648             2147483647
unsigned int       0                  4294967295
long               -2147483648             2147483647
unsigned long      0                  4294967295
__int64            -9223372036854775808      9223372036854775807
unsigned __int64   0                  18446744073709551615
float              -3.40282e+38             3.40282e+38
double             -1.79769e+308            1.79769e+308
long double        -1.79769e+308            1.79769e+308

D:\_wy_workspace_\project\kulgram\bit and byte\x64\Debug (master)
λ
```

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

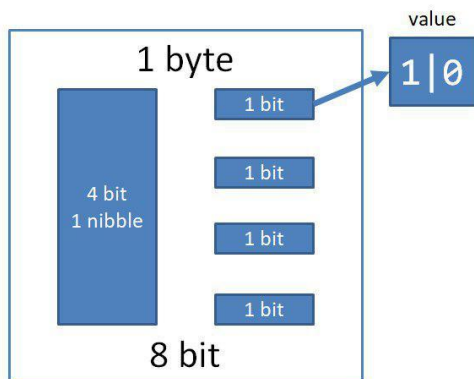
*catatan msvc: `__int64 == long long`

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

```
wahyutama@ :/mnt/d/_wy_workspace_/project/kulgram/bit and byte/range
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/range$ g++ range.cpp --std=c++14
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/range$ ./a.out
tipe data      nilai min      nilai max
bool           0              1
char          -128           127
unsigned char   0             255
short         -32768          32767
unsigned short  0             65535
int           -2147483648       2147483647
unsigned int    0             4294967295
long          -9223372036854775808  9223372036854775807
unsigned long   0             18446744073709551615
long long      -9223372036854775808  9223372036854775807
unsigned long long  0             18446744073709551615
float          -3.40282e+38       3.40282e+38
double        -1.79769e+308    1.79769e+308
long double    -1.18973e+4932     1.18973e+4932
wahyutama@DESKTOP-VO6MN3I:/mnt/d/_wy_workspace_/project/kulgram/bit and byte/range$
```

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

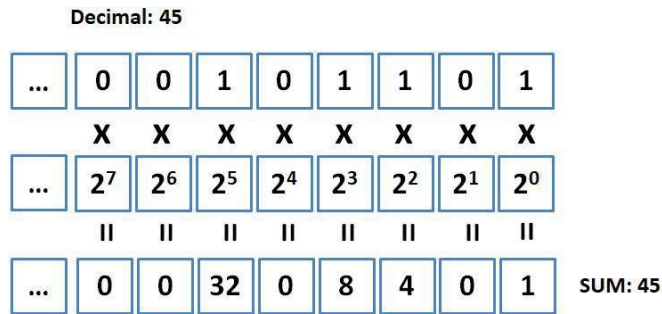
The Big Picture



sekarang gimana si cara nyimpennya...

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

How is it stored?



jadi misalnya angka 45, dia bakal kesimpn dengan sequence yang seperti itu... compilernya udah cukup canggih kok.... kasih aja desimal 45, udah disimpn 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...
```

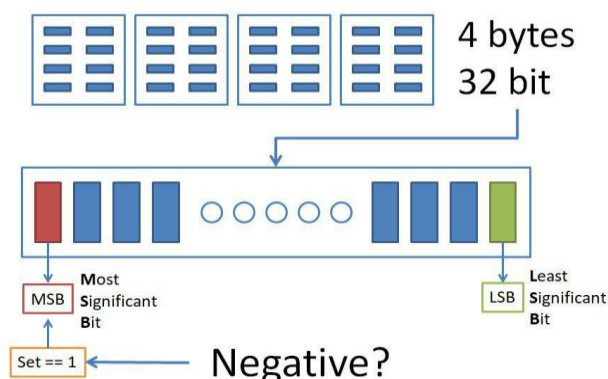
nah salah satunya disitu....

kita ngeset nilai dari bit yang ada dalam sebuah tipe data.... jadi alih-alih make banyak variabel dan bikin mess lebih 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....

Two's Complement (I)



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

dengan adanya two's complement seorang programmer akhirnya bisa nyimpen nilai negatif...
Horeeeeeee....

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

kalo cara nyarinya...?

How is Negative?

Positive: 45

0	0	1	0	1	1	0	1
---	---	---	---	---	---	---	---

Steps:

- Flip all bits
- Add 1

1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---

Flip all bits

1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---

Add 1

signed: -45 unsigned: 65491

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 positifnya kepotong... (trade-off lah...) bisa juga kalo mau di-cast antara unsigned dan signed....

sampe sini...?

How is it stored?

Decimal: 45								
...	0	0	1	0	1	1	0	1
	X	X	X	X	X	X	X	X
...	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	II	II	II	II	II	II	II	II
...	0	0	32	0	8	4	0	1
SUM: 45								

pasti ribet lah ya ngonversi dari desimal ke bit kalo pas butuh.... hexadecimal to the rescue...

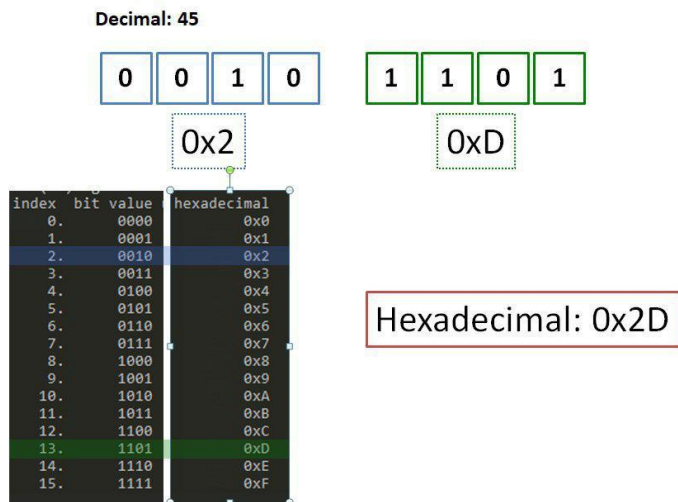
nah apa pula itu...?

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 hexadecimal....

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

Hexadecimal?



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....

```
konversi Binary - Decimal - Hexadecimal
Ery E. Wahyutama a byte addressable machine. it turns out to b
Indonesia C/C++ Warriors example, we will often represent t
1. convert from binary to decimal
2. convert from binary to hexadecimal
3. convert from decimal to binary
4. convert from decimal to hexadecimal
5. convert from hexadecimal to binary
6. convert from hexadecimal to decimal
0. Exit
pilihan anda: 1
--Binary to Decimal--
input: 0001 0010
decimal: 18 7 6 5 4 3 2 1 0
press any key to continue...
```

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.... maaf yak kalo enggak sempurna....