Eng. Ahmed Assaf — Embedded Systems — 7-12-2018

Data types

- Preemitive
  - char, int → decimal
  - float, double → Floating
  - Void → Function Argument / Function Return / void Pointer
- Derived
- User Defined

Object creation Syntax

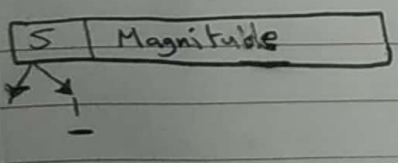
Data type object name ;

---

* char & int (default signed)

Consider char → 1 byte

① → Sign Magnitude



Problem * Zero has two values

0 000 0000

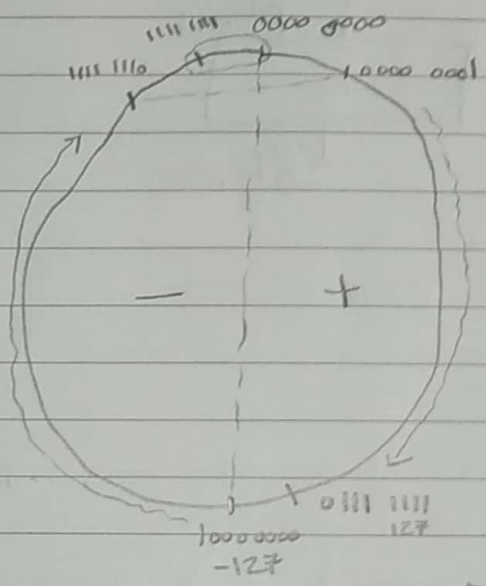
1 000 0000

0000 0001 (1)

+ 1000 0001 (-1)

1000 0010 (-2)

Number circle of 1st complement



② 1st Complement

0000 0001 (+1)

1st Comp. → 1111 1110 (-1)

→ Zero has two values

0000 0000

1st Comp. → 1111 1111

→ Math eqs.

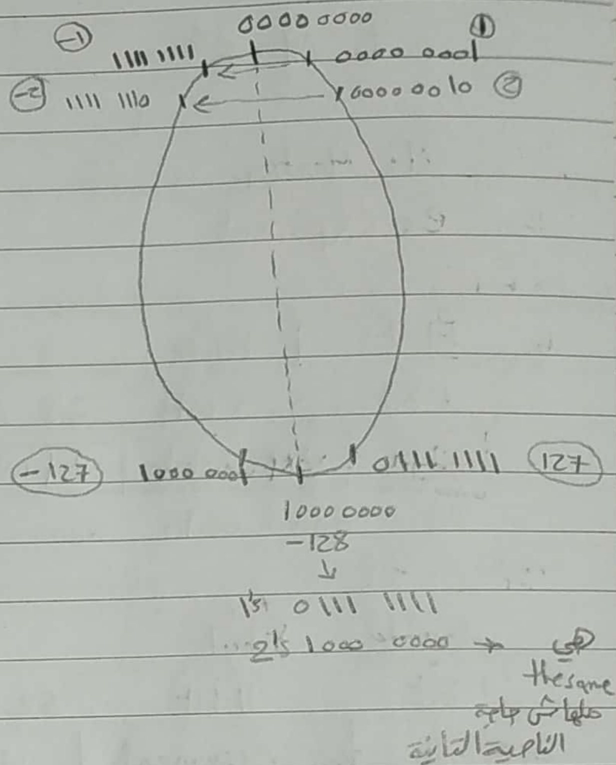
0000 0001

1111 1110

1111 1111

③ 2's Complement

Numbering Circle



- → zero has one value

  1st (0000 0000
  ↘ 1111 1111
  2's (↘ 0000 0000 بجمع واحد
  ~~1~~

- → Math eq.

```
    0000 0001   (+1)
  + 1111 1111   (-1)
  -------------
(1) 0000 0000
```

ex. 1111 1111
SM : -127
1's : 0
2's : -1

* The representation of negative number in ANSI C is compiler dependent.

%u → هيخليني أطبع الرقم كموجب حتى لو سالب

---

double & float (has sign bit so they are always signed preemitive Data types)

```
#include <stdio.h>
void main(void)
{  float x =0.75;
   float y =0.75;
   if(x>y)
       Printf ("float is bigger");
   else if(y>x)
       Printf("double is bigger");
   else
       printf ("They are the same");
}
```

To Save Float or double

$1.M \; 2^{e}$

M: Mantissa
e: exponent

	-4		
	000	0000	0100
1s	111	1111	1011
2s	000	0000	0100

Float 4 byte

1 bit	8 bit	23 bit
Sign	exponent	Mantissa
0	0000 0010	000 ---- 1011 → 0.75

double 8 byte

1 bit	11 bit	52 bit
Sign	exponent	Mantissa
0	———— 10	———— 1011 → 0.75

0.1

	×2
0.2	0.1
0.4	0.2
0.8	0.4
1.6	0.8
1.2	0.6
	0.2

0.000110011000011 ---

1.10001100 0 ---- $* 2^{-4}$ → هكتب الـ 2's comp في الـ memory exponent

6.75

×2	
1.5	0.75
1	0.5

110.11

1.1011×2^{2} → المكتوب في الـ exponent

float 4byte

sign	8 bit exponent	23 bit Mantissa
0	-4	00011 000 1100 0 110 0011 00011 → 0.1

double 8byte

sign	exponent 11 bit	Mantissa 52 bit
0	0000 ... -4	000 11 00011 ———— → 0.1

سؤال: zero أو one

لو الـ 23 bit جوه الـ float ولسه مخلصتش هيخلي الـ 23 Bit بـ واحد فلو هي أملاها بـ zero ونكده الـ float هيكون أكبر من الـ double لو الـ 23 Bit أملاها

لكن لو كانت أصلا بواحد فكده الـ double هيكون أكبر من الـ Float

Rule

Never compare two Floating variables from different data type

Misra C:

It is not recommended to use Floating numbers. Instead we Shall use Fixed point arithmatic (can deal any floating numbers as decimal numbers).

Arithmatic

① uniry

Inc ++

Dec --

ميتفعش

Assignment

X=10

y= (x++) +1;

y= x+++++1

y = ? y=x++++ +1;

y= x++;

y=++x;

Arithmatic

② Binary

+ - * / → ينفع معاهم الـ decimal والـ Float و النتيجة Float

% → ينفع مع الـ decimals بس

① 5%5 ⇒ 0

② 10%1 ⇒ 0

③ Ⓐ 5 % -3 ⇒ 2

Ⓑ -5 % 3 ⇒ -2

Ⓒ -5 % -3 ⇒ -2

ليه

④ 0%3=0 ⇒ $d(\frac{0}{3})\times3+?=0$ → 0

$3\%0 = \text{Error} \Rightarrow d(\frac{3}{0})\times0+?=3$ → undefined

a%b

$decimal(\frac{a}{b}) * b + a\%b = a$

Ⓐ 5/-3

$d(\frac{5}{-3}) * -3 + ? = 5;$

$3 + ? = 5;$

$? = 2;$

Ⓑ $d(\frac{-5}{3}) \times 3 + ? = -5$

$-3 + ? = -5$

$? = -2$

ب zero يالكن لو كانت أصلا سواء فكده الـ double هيكون أكبر من الـ float؟

Rule

Never compare two floating variables from different data type.

Misra C:

It is not recommended to use floating numbers. Instead we shall use fixed point arithmatic (can deal any floating numbers as decimal numbers).

- **Arithmatic**
  - ① Unary
    - Inc ++
    - Dec --

ميتفش

Assignment
x = 10

y = x++ +1;
y = x++ ++1
y = 1   y = x++ +1;

y = x++;
y = ++x;

Arith
② Binary

+ - * /	→	يتعامل مع الـ decimal والـ float والنتيجة float
%	→	يتعامل مع الـ decimal بس

① 5 % 5 ⇒ 0
② 10 % 1 ⇒ 0
③ 5 % -3 ⇒ 2
④ -5 % 3 ⇒ -2
⑤ -5 % -3 ⇒ -2

a % b

$$\text{decimal}\left(\frac{a}{b}\right) * b + a \% b = a$$

③ 5 / -3
d(5/-3) * -3 + ? = 5;
3 + ? = 5;
? = 2;

④ d(-5/3) * 3 + ? = -5
-5 + ? = -5
? = -2

④ 0 % 3 = 0 ⇒ d(0/3) * 3 + ? = 0

3 % 0 = Error ⇒ d(3/0) * 0 + ? = 3
undefined

Relational

> < ==

>= <= !=

```
if(x>3)
{
   ==
}
else
{
   ==
}
```

Notes

①
```
if(3==x)
{
   ==
}
```

② `y=(x==3);`

قاعدة

ال False هو ال zero وأي حاجة غير كده هيكون true
ومن هنا ال compiler يختار ال true بتاعته

x	y
False: 0000 0000	0000 0000
true: 0000 0001	1111 1111

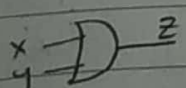
gcc compiler

ده Compiler أفضل علشان لو حصل interference فإني أغير من false لـ true
والعكس غالبًا هيكون أصعب هنا من أول case

logical

&& 

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

|| 

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

!

```
int x = 5;
int z = 10;
int K = 0;
int y = x && z;
        ↑ True
int y = K && z;
        ↑ False
int y = K || z;
        ↑ True
int y = !K;
        ↑ True
```

Bitwise

& | ~

^ >> <<

x + 3 >> 4 %

```
int x=5;
int y=10;
```

int z = x & y

0000 0101
0000 1010
0000 0000

→ False

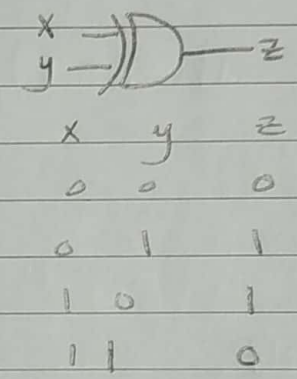
int k = x && y;

↑ True

int L = 7 ^ 3;

↑ 4

0000 0111
0000 0011
0000 0100 (4)



x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

```
int x=11;
y= x>>3;
```

0000 1011 >>3

↓

0000 0001

Assignment operator

- x = 3 ✓
- y = x = 3 ✓

- if(x=3) always true

unless if(x=0) → Assignment operator will return zero

So it will be false.

* += -= *= /= %=

&= |= ~= ^= >>= <<=

Assignment 6

```
circularFunc(operand, num of shifts)
{

}
```

1011 0101
1 0110

y = ((x >> 3) + (x << 3))

* X++; → (Inc) 1 cycle in Assembly

X=X+1;
X+=1; } → the same with different notation

1. Read
2. Add
3. Store } → 3 Cycle

Other
↳ Ternary

Rule: Never use Tenary operator because it not good for Readibility wise

Condition ? True Action : False Action

(ex)

```
X==3 ? printf("Ahmed"); printf("Ali");
```

Assignment 7
- Bit_calc.h
  - → Set_bit (var, bit No)
  - → CLR_bit (var, bit No)
  - → Tog_bit (Var, bit No)
  - → Get_bit (Var, bit No)
  - → Assign_bit (var, bit No, value)
    - 0 او 1

Assignment 8
↳ Print Binary Decimal ()
{

}

Assignment 9
Print Binary float ()
{

}
101.110

Mantissa = ✓
Exponent = ✓

Assignment 10
Print Real value ()

+Ve

-ve
- LS.M →
- 1's ⇒
- 2's ⇒

بيعمل Scan
لـ الرقم كـ binary
1's & 0's