Task_4 Report by: Zyad Hesham AbdelWahab

Data Modifiers

Sign, Size, Storage, Constant, Volatile

Sign
- signed (%d): $[-2^{n-1}, 2^{n-1}-1]$: 31 bit & 1 bit for sign
- unsigned (%u): +ve → $[0, 2^{n-1}]$: 32 bit

Constant: makes the var. Read only (but u can trick the system with pointer deReference)

Volatile: Prevents the System from optimizing the variable

differ from system to another

types of storing -ve int

Sign magnitude: MSB (Most significant bit) — Convert [struck out] bit : 0 +ve, 1 -ve

Cons: two values of Zero.

Some mathematical operations errors like +1-1 → -2 not 0.

ex: 0101 → 1101 (changed the MSB)

1's Complement:

Convert every bit from 0 to 1 and vice versa (0 ⇄ 1) & the MSB represents the sign

Ex

101 → 010

Cons: two values of Zero

2's Complement: Widely used in modern Computers

Convert every bit (0 ⇄ 1) and add 1. & the MSB represents the sign.

EX

101 → 011

Used for integral only

Floating Point representation: MSB | n+bias: 8 bits (exponent) | 23 bits (mantissa) — 32 bits, bias 127

MSB | n+bias: 11 bits (exp..) | 52 bits (mantissa) — 64 bits, bias 1023

Normalization for Floating Point: $(4.25)_{10} \rightarrow (100.01)_2$ → after normalization

1.0001×2^2

$1.\_\_ \times 2^{\square}$

---

there is also long double

Size

short
Guarantee 2 byte int
≡
short int

long
Guarantee 4 byte int
≡
long int

long long
8 byte int

---

storage

auto
↙
local var.
int x; ≡ auto int x;
scope within block {..}
lifetime as long as func. running
initial value: garbage
memory segments: stack

extern
used to declare a variable that is defined in another file in the program

static
at local variable the variable keeps it's value between func. calls.
ex. void func(){
static int counter=0;
counter ++;
} ----
after func. ends if called again the value would stay 1 not zero
at Global variable
the variable will be invisible for other .C files
(limits the variable's scope to it's file)

register
suggests the compiler to store the var. in CPU registers instead of RAM.
※ you can't get the address of the variable in a register using &

Pointers

Syntax.

Pointee type * Pointer-Name

ex.

```
int * ptr;      Ptr = &(int-Name)
```

it stores the address of a variable

the size of the pointer differ due to system type

16-bit system : Pointer size 2 bytes
32-bit " : " 4 bytes
64-bit " : " 8 bytes

the Pointee type affects how many data addresses the Pointer will hold

ex

```
int x = INT_MAX;
int *ptr = &x;
printf("%d", x);
Printf ("%d", *Ptr);
Printf ("%p", Ptr);
```

Output:
2107483647
"
0X0Feeo1 for example

if we used char *ptr instead of int *ptr it will read only the first byte of the int.