BÁO CÁO THỰC HÀNH

Môn: Lập trình hệ thống

Buổi báo cáo: Lab 01

Lớp: NT209.O22.ANTT.2

GVHD: Đỗ Thị Hương Lan

Ngày thực hiện: 13/3/2024

THÔNG TIN CHUNG

STT	Họ và Tên	MSSV	Lớp
1	Trần Tuấn Anh	22520080	ATTT2022.1
2	Nguyễn Khắc Hậu	22520410	ATTT2022.1

Báo Cáo Chi Tiết

Câu 1.3

Chỉnh sửa:

```
int getHexcha(int x, int n)
{
    int a = 0xf;
    //lấy vị trí hex dùng mask 0...1111..0
    //dịch a đến vị trí n sẽ dịch trái mỗi 4 * n bit
    //4 * n = n << 2
    //sau khi and với mask dịch phải lại đúng 4*n bit để lấy giá trị hexchar đó
    return (x & (a << (n << 2))) >> (n << 2);
}
```

Thành:

```
// 1.3
int getHexcha(int x, int n)
{
    //đưa Hexchar cần về 4 bit cuối --> x >> (n << 2)
    //lấy 4 bit dùng mask 0xf --> & 0xf
    return (x >> (n << 2)) & 0xf;
}
```

➔Giảm số lượng toán tử cần dùng

Câu 1.5

Chỉnh sửa:

```
// 1.5
int divpw2(int x, int n)
{
    //kiểm tra số n dương hay âm
    //(n >> 31) --> bit dấu của n
    //!n kiểm tra n có = 0 hay không
    int test = !(n >> 31) ^ !n;
    //dịch test đến vị trí bits dấu
    //dịch phải bit dấu sẽ lấp chỗ vị trí bit dấu trước đó
    test = ((test) << 31) >> 31;
    //n dương thì x/2^n sẽ là x >> n
    //n âm thì x/2^n sẽ là x * 2^(-n)
    //-n = ~n+1
    //x * 2^(-n) = x << (~n+1)
    //n dương nên test có 32 bit 1 and với x >> n sẽ giữ nguyên giá trị còn x << (~n+1) and với ~test sẽ bằng 0 và or 2 kết quả lại sẽ lấy x >> n
    //n âm thì test có 32 bit 0 and với x >> n sẽ bằng 0 còn x << (~n+1) and với ~test giữ nguyên giá trị và or 2 kết quả lại sẽ lấy x << (~n+1)
    return ((test) & (x >> n)) | ((~test) & ( x << (~n+1)));
}
```

Thành:

```
// 1.5
int divpw2(int x, int n)
{
    //n dương thì x/2^n hay x >> n
    //n âm thì x/2^n hay x * 2^(-n) hay x << (~n+1)
    //n dương --> n >> 31 sẽ có 32 bit 0 --> ((n >> 31) & ( x << (~n+1)) sẽ bằng 0) or (~(n >> 31) & (x >> n) sẽ bằng x >> n) --> lấy x >> n
    //n âm --> n >> 31 sẽ có 32 bit 1 --> ((n >> 31) & (x << (~n+1)) sẽ bằng (x << (~n+1))) or (~(n >> 31) & (x >> n) sẽ bằng 0)
    //--> lấy (x << (~n+1))
    return (~(n >> 31) & (x >> n)) | ((n >> 31) & ( x << (~n+1)));
}
```

➔Giảm số lượng toán tử cần dùng, bỏ bước xét dấu của n

Câu 2.2:

Sửa

```c
// 2.2
int is16x(int x)
{
    int a = 0xf;
    x = x & a;
    return !(x ^ 0x0);
}
```

Thành:

```c
// 2.2
int is16x(int x)
{
    // những số chia hết cho 16 sẽ có 4 bit cuối đều bằng 0
    //lấy 4 bit cuối x & 0xf --> và kiểm tra có bằng 0 hay không
    return !(x & 0xf);
}
```

➔Bỏ toán tử thừa

Hàm max(int x, int y)

Ý tưởng : Lấy x – y và xét bit dấu của hiệu :

- Nếu bit dấu là 0 → hiệu lớn hơn hoặc bằng 0 →trả về x
- Nếu bit dấu là 1 → hiệu nhỏ hơn 0 →trả về y

Source code :

```c
//hàm max(int x, int y)
int max(int x, int y) {
    return (~((x + (~y + 1)) >> 31) & x) | (((x + (~y + 1)) >> 31) & y);
}
```

Minh chứng kết quả:

```
Your evaluation result:
1.1 bitOr       Pass.
1.2 negative    Pass.
1.3 getHexcha   Pass.
1.4 flipByte    Pass.
1.5 divpw2      Advanced Pass.
2.1 isEqual     Pass.
2.2 is16x       Pass.
2.3 isPositive  Pass.
2.4 isGE2n      Pass.
--- FINAL RESULT ---
Score: 10.5
Excellent. We found a master in bit-wise operations :D
----Function max(x, y)----
max(5, 6): 6
max(5, 5): 5
max(10, 6): 10
```