\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW4

Author: E34071061 謝沅承 <andy420811@gmail.com>

Class: 化工 (甲班)

Description:

引用math.h資料庫，並用switch轉跳到參數指定的功能

case(1)or case(2)將輸入的數字給予num\_f/num\_lf變數且將位址存入指標p中，利用for迴圈以及位元運算子&把num\_f/num\_lf的bit pattern存入陣列中，並印出翻轉後的陣列以及對應的空格。

case(3)or case(4)將輸入參數的對應指數部分字元以if將內容與字元1比較轉換為次方存入exp變數中，並減去偏差值127，再利用math.h的pow函數以for及if把fraction部分乘上對應指數並印出結果。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int main(int argc,char \*argv[]){

int mode=atol(argv[1]);

int \*p;

long long int \*p2;

long long int a=1;

float num\_f;

double num\_lf;

int exp=0,j=0;

int bit\_32[32];

int bit\_64[64];

switch(mode){

case 1:

num\_f=atof(argv[2]);

p=&num\_f;

for(int i=0;i<32;i++){

bit\_32[i]=\*p&a<<i?1:0;

}

for(int i=0;i<32;i++){

printf("%d",bit\_32[31-i]);

if(i==0)printf(" ");

if(i==8)printf(" ");

}

printf("\n");

break;

case 2:

num\_lf=atof(argv[2]);

p2=&num\_lf;

for(int i=0;i<64;i++){

bit\_64[i]=\*p2&a<<i?1:0;

}

for(int i=0;i<64;i++){

printf("%d",bit\_64[63-i]);

if(i==0)printf(" ");

if(i==11)printf(" ");

}

printf("\n");

break;

case 3:

for(int i=0;i<8;i++){

if(argv[3][i]=='1')

exp+=pow(2,7-i);

}

exp-=127;

num\_f=pow(2,exp);

for(int i=exp;i>exp-23;i--){

if(argv[4][j]=='1')

num\_f+=pow(2,i-1);

j++;

}

printf("%f\n",num\_f);

break;

case 4:

for(int i=0;i<11;i++){

if(argv[3][i]=='1')

exp+=pow(2,7-i);

}

exp-=127;

num\_f=pow(2,exp);

for(int i=exp;i>exp-23;i--){

if(argv[4][j]=='1')

num\_f+=pow(2,i-1);

j++;

}

printf("%f\n",num\_f);

break;

}

return 0;

}

Compilation:

gcc -o hw4 hw4.c -lm

Execution:

./hw4 1 85.125

./hw4 2 85.125

./hw4 3 0 10000101 01010100100000000000000

./hw4 4 0 10000000101 0101010010000000000000000000000000000000000000000000

Output:

0 10000101 01010100100000000000000

0 10000000101 0101010010000000000000000000000000000000000000000000

85.125000

85.125000

2

2-1 (no), If not, what is the smallest floating point number?

5.877471754111437539843682686111228389093327783860437607543 7585313920862972736358642578125e-39

2-2 (bit pattern) 0 00000000 00000000000000000000000

2-3 explanation 兩個float的數字在電腦儲存的型態皆是相同的 1.1754943508222875079687365372222456778186655567720875215087517062784172594547271728515625000000000000e-38 = 1.1754943508222875079687365372222456778186655567720875215087517062784172594547271728515625000000000000e-38

2-4 (bit pattern,選其中一個即可)

0 11111111 00000000000000000000000

2-5 i.number 3.141593

ii.number 0.333333