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

Report: hw6

Author: F74045018 廖其忻<cayon.1318.96@hotmail.com>

Class: 甲班

Description:

I starting doing this homework 1 week before but finished at the last time. This homework is doing the work of converting between floating point and bit pattern. This homework is divided into 2 parts by applying different function.hw6\_1 is applied with integer pointer and hw6\_2 is applied with union and bit pattern.

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

[hw6\_1]

#include<stdio.h>

#include<limits.h> //for CHAR\_BIT

void Float\_display\_Bits(unsigned value) //the function of changing FLOAT to 32-BITS PATTERN

{

int c;

unsigned displaymask = 0x80000000;

int C\_B= CHAR\_BIT\*sizeof(unsigned);

printf("The bit pattern of Float=\n");

for(c=0;c<C\_B;c++)

{

putchar(value&displaymask?'1':'0'); //AND case

value<<=1; //shifting

}

putchar('\n');

}

void Double\_display\_Bits(unsigned long long value) //the function of changing DOUBLE to 64-BITS PATTERN

{

int c;

unsigned long long displaymask = 0x8000000000000000;

int C\_B= CHAR\_BIT\*sizeof(unsigned long long);

printf("The bit pattern of Double=\n");

for(c=0;c<C\_B;c++)

{

putchar(value&displaymask?'1':'0'); //AND case

value<<=1; //shifting

}

putchar('\n');

}

void Bits\_display\_Float(int b[]) //the function of changing 32-BITS PATTERN to FLOAT

{

int i,e;

int t=0;

float two=1;

int Exponent=0;

float Value;

float Mantissa=0.0;

int power1=1;

float power2=0.5;

for(i=8;i>=1;i--)

{

Exponent=Exponent+(b[i])\*power1;

power1=power1\*2;

}

e=Exponent-127;

while(e>=0)

{

two=two\*2;

t++;

if(t==e)break;

}

while(e<0)

{

two=two\*0.5;

t--;

if(t==e)break;

}

for(i=9;i<=31;i++)

{

Mantissa=Mantissa+(b[i])\*power2;

power2=power2\*(0.5);

}

Value=(1+Mantissa)\*two;

if(b[0]==0)

Value=Value\*1;

if(b[0]==1)

Value=Value\*-1;

printf("Exponent:%f\n",Exponent);

printf("e:%f\n",e);

printf("Mantissa:%f\n",Mantissa);

printf("The Standard form of the 32-bits pattern:\n");

printf("%e\n",Value);

}

void Bits\_display\_Double(int b[]) //the function of changing 64-BITS PATTERN to DOUBLE

{

int i;

int Exponent,e;

double Value;

Exponent=0;

double Mantissa=0.0;

int power1=1;

float power2=0.5;

for(i=11;i>=1;i--)

{

Exponent=Exponent+(b[i])\*power1;

power1=power1\*2;

}

e=Exponent-1023;

int t=0;

float two=1.0;

while(e>=0)

{

two=two\*2;

t++;

if(t==e)break;

}

while(e<0)

{

two=two\*0.5;

t--;

if(t==e)break;

}

for(i=12;i<=63;i++)

{

Mantissa=Mantissa+(b[i])\*power2;

power2=power2\*(0.5);

}

Value=(1+Mantissa)\*two;

if(b[0]==0)

Value=Value\*1;

if(b[0]==1)

Value=Value\*-1;

printf("The Standard form of the 64-bits pattern:\n");

printf("%e\n",Value);

}

int main()

{

float f;

double d;

int b,B;

int C\_B\_F= CHAR\_BIT\*sizeof(unsigned);

int C\_B\_D= CHAR\_BIT\*sizeof(unsigned long long);

int i;

int y=1;

while(y==1)

{

printf("Enter the Floating number=\n");

scanf("%f",&f);

unsigned u = \*(unsigned\*)&f; //not casting, the way of putting the contents of the memory of f into u

Float\_display\_Bits(u);

char bb[32];

int BB[32];

printf("Please input a bit pattern in 32 bits:\n");

scanf("%s",&bb); //input with the type of string into the charater array

for(i=31;i>=0;i--)

BB[i]=(int)bb[i]-48; //casting

Bits\_display\_Float(BB);

printf("Enter the Double number=\n");

scanf("%lf",&d);

unsigned long long h = \*(unsigned long long\*)&d; //not casting, the way of putting the contents of the memory of h into d

Double\_display\_Bits(h);

char cc[64];

int CC[64];

printf("Please input a bit pattern in 64 bits:\n");

scanf("%s",&cc); //input with the type of string into the charater array

for(i=63;i>=0;i--)

CC[i]=(int)cc[i]-48; //casting

Bits\_display\_Double(CC);

break;

}

return 0;

}

[hw6\_2]

/\*\*\*\*\*\*\*\*\*\*

Student Number: F74045018

Name: Cayon Liow Keei Yann

Way of Compiling: gcc -o hw6\_2 hw6\_2.c

Way of Executing: ./hw6\_2

Function of the Program:A program designed to convert between float or double and bit pattern by applying the method of union.

Updated Date: 2015.12.20

\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include<limits.h>

typedef union Float\_to\_Bits //the application of bit pattern and union

{

struct {

unsigned int mantissa : 23;

unsigned int exponment : 8;

unsigned int sign : 1;

};

float s;

}FtB;

void Double\_display\_Bits(unsigned long long value) //the function of changing DOUBLE to 64-BITS PATTERN

{

int c;

unsigned long long displaymask = 0x8000000000000000;

int C\_B= CHAR\_BIT\*sizeof(unsigned long long);

printf("The bit pattern of Double=\n");

for(c=0;c<C\_B;c++)

{

putchar(value&displaymask?'1':'0'); //AND case

value<<=1; //shifting

}

putchar('\n');

}

void Bits\_display\_Float(int b[]) //the function of changing 32-BITS PATTERN to FLOAT

{

int i,e;

int t=0;

float two=1;

int Exponent=0;

float Value;

float Mantissa=0.0;

int power1=1;

float power2=0.5;

for(i=8;i>=1;i--)

{

Exponent=Exponent+(b[i])\*power1;

power1=power1\*2;

}

e=Exponent-127;

while(e>=0)

{

two=two\*2;

t++;

if(t==e)break;

}

while(e<0)

{

two=two\*0.5;

t--;

if(t==e)break;

}

for(i=9;i<=31;i++)

{

Mantissa=Mantissa+(b[i])\*power2;

power2=power2\*(0.5);

}

Value=(1+Mantissa)\*two;

if(b[0]==0)

Value=Value\*1;

if(b[0]==1)

Value=Value\*-1;

printf("The Standard form of the 32-bits pattern:\n");

printf("%e\n",Value);

}

void Bits\_display\_Double(int b[]) //the function of changing 64-BITS PATTERN to DOUBLE

{

int i;

int Exponent,e;

double Value;

Exponent=0;

double Mantissa=0.0;

int power1=1;

float power2=0.5;

for(i=11;i>=1;i--)

{

Exponent=Exponent+(b[i])\*power1;

power1=power1\*2;

}

e=Exponent-1023;

int t=0;

float two=1.0;

while(e>=0)

{

two=two\*2;

t++;

if(t==e)break;

}

while(e<0)

{

two=two\*0.5;

t--;

if(t==e)break;

}

for(i=12;i<=63;i++)

{

Mantissa=Mantissa+(b[i])\*power2;

power2=power2\*(0.5);

}

Value=(1+Mantissa)\*two;

if(b[0]==0)

Value=Value\*1;

if(b[0]==1)

Value=Value\*-1;

printf("The Standard form of the 64-bits pattern:\n");

printf("%e\n",Value);

}

int main()

{

int i;

int y=1;

int t=0;

float v;

int b[23];

FtB s;

printf("Please input a floating number:\n");

scanf("%f", &s);

printf("The bit pattern of the floating number:\n");

printf("%d", s.sign);

unsigned ddisplaymask = 1<<7;

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

{

putchar(s.exponment&ddisplaymask?'1':'0'); //AND case

ddisplaymask>>=1;

}

unsigned displaymask = 1<<22;

for(i=0;i<23;i++)

{

b[i]=(s.mantissa&displaymask?'1':'0')-48; //AND case

displaymask>>=1; //shifting

printf("%d",b[i]);

}

printf("\n");

double d;

char bb[32];

int BB[32];

printf("Please input a bit pattern in 32 bits:\n");

scanf("%s",&bb); //input with the type of string into the charater array

for(i=31;i>=0;i--)

BB[i]=(int)bb[i]-48; //casting

Bits\_display\_Float(BB);

;

printf("Enter the Double number=\n");

scanf("%lf",&d);

unsigned long long h = \*(unsigned long long\*)&d; //not casting, the way of putting the contents of the memory of h into d

Double\_display\_Bits(h);

char cc[64];

int CC[64];

printf("Please input a bit pattern in 64 bits:\n");

scanf("%s",&cc); //input with the type of string into the charater array

for(i=63;i>=0;i--)

CC[i]=(int)cc[i]-48; //casting

Bits\_display\_Double(CC);

return 0;

}

Compilation:

[hw6\_1]gcc -o hw6\_1 hw6\_1.c

[hw6\_2]gcc -o hw6\_2 hw6\_2.c

Execution:

[hw6\_1]./ hw6\_1

[hw6\_2]./ hw6\_2

Output:

[hw6\_1]

Enter the Floating number=

-4.5

The bit pattern of Float=

11000000100100000000000000000000

Please input a bit pattern in 32 bits:

11000000100100000000000000000000

The Standard form of the 32-bits pattern:

-4.500000e+00

Enter the Double number=

-4.5

The bit pattern of Double=

1100000000010010000000000000000000000000000000000000000000000000

Please input a bit pattern in 64 bits:

1100000000010010000000000000000000000000000000000000000000000000

The Standard form of the 64-bits pattern:

-4.500000e+00

[hw6\_2]

Please input a floating number:

5.6

The bit pattern of the floating number:

01000000101100110011001100110011

Please input a bit pattern in 32 bits:

01000000101100110011001100110011

The Standard form of the 32-bits pattern:

5.600000e+00

Enter the Double number=

5.6

The bit pattern of Double=

0100000000010110011001100110011001100110011001100110011001100110

Please input a bit pattern in 64 bits:

0100000000010110011001100110011001100110011001100110011001100110

The Standard form of the 64-bits pattern:

5.600000e+00

Error message:

hw6\_1.c: In function ‘Float\_display\_Bits’:

hw6\_1.c:23:16: error: invalid operands to binary & (have ‘unsigned int’ and ‘double’)

putchar(value&displaymask?'1':'0'); //AND case

hw6\_1.c: In function ‘Float\_display\_Bits’:

hw6\_1.c:18:11: error: ‘CHAR\_BIT’ undeclared (first use in this function)

int C\_B= CHAR\_BIT\*sizeof(unsigned);

^

hw6\_1.c:18:11: note: each undeclared identifier is reported only once for each function it appears in

hw6\_1.c: In function ‘Double\_display\_Bits’:

hw6\_1.c:34:11: error: ‘CHAR\_BIT’ undeclared (first use in this function)

int C\_B= CHAR\_BIT\*sizeof(unsigned long long);

^

hw6\_1.c: In function ‘main’:

hw6\_1.c:149:13: error: ‘CHAR\_BIT’ undeclared (first use in this function)

int C\_B\_F= CHAR\_BIT\*sizeof(unsigned);

^

Answer:

2-1

Yes, it is.

2-2

The bit pattern of the floating number:

00000000000000000000000000000000

2-3

The result: 1.175494e-38 =1.175494e-38

f1 is equal to f2. It is because the smallest value is 1.175494e-38 so when is slightly larger the smallest value the remainder will automatically be ignored.