**Changes needed are highlighted**

**Change 4 to 32 and 31 to 255**

/\*

Super Integer!

When 64 bits just isn't enough!

Implements IO, string conversion, and

addition for a 256bit (or bigger) integer.

Dan Ross Fall 2013

http://stackoverflow.com/questions/5246976/binary-string-to-decimal-string?rq=1

http://stackoverflow.com/questions/11006844/convert-a-number-from-decimal-string-to-binary-representation?rq=1

\*/

#include "stdio.h"

#include "conio.h"

#include "string.h"

// a 256 bit (8 byte) integer

struct superint {

char byte[4]; // 32 bit for now

};

/\*

Takes 2 superints and adds them

returns -1 if overflow occurs,

returns 0 otherwise

uses the hugeint addition algorithm

and setbit helper function

\*/

int add\_si(struct superint si1, struct superint si2);

/\*

Sets the bit at position 'index' to the value 'v'.

Calculates the bit and byte position,

generates bitmasks, and sets the bit using the mask.

\*/

void setbit(struct superint \* si, int index, int v)

{

int bytenum = index / 8;

int bitnum = index % 8;

unsigned char mask = 128;

int i;

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

mask = mask >> 1;

if(v)

si->byte[bytenum] = si->byte[bytenum] | mask;

else

{

mask = ~mask;

si->byte[bytenum] = si->byte[bytenum] & mask;

}

}

/\*

Gets the bit value at position 'index'.

Calculates the bit and byte position,

generates a bitmask, gets the bit using the mask,

returns the character '1' or '0'

\*/

char getbit(struct superint si, int index)

{

int bytenum = index / 8;

int bitnum = index % 8;

unsigned char mask = 128;

int i;

int b;

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

mask = mask >> 1;

b = si.byte[bytenum] & mask;

if(b)

return 1;

else

return 0;

}

/\*

Sets a superint to zero

\*/

void setzero(struct superint \* si)

{

int B;

for(B = 0; B < 4; B++)

si->byte[B] = 0;

}

/\*

Returns -1 (true) if the superint is 0

otherwise returns 0 (false)

\*/

int iszero(struct superint si)

{

int b;

// assume superint is zero

// try to prove otherwise

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

{

if(getbit(si, b))

return 0;

}

// if we got here, superint must be zero

return -1;

}

/\*

Returns -1 (true) if the string is 0

otherwise returns 0 (false)

0 means filled with all '0'

\*/

int iszeroStr(char s[])

{

int c;

// assume string is zero

// try to prove otherwise

for(c = 0; s[c]; c++)

{

if(s[c] != '0')

return 0;

}

// if we got here, superint must be zero

return -1;

}

/\*

returns 1 (true) if odd

returns 0 (false) if even

\*/

int isOdd(char s[])

{

return (s[(strlen(s) - 1)] - '0') % 2;

}

/\*

Receives a superint pointer and a string representation of a

decimal number.

Converts the string to binary and fills a superint with the bits.

Returns 1 if the binary bits overflow,

otherwise returns 0

\*/

int dec\_to\_si(struct superint \* si, char s[])

{

char buf[80];

int c = 0;

int b = 0;

int nxtadd;

int add;

// make a copy

strcpy(buf, s);

while(!iszeroStr(buf))

{

// check for binary overflow

if(b > 31)

{

return 1;

}

if(isOdd(buf)) // is it odd

setbit(si, b, 1);

else

setbit(si, b, 0);

// Begin divide by 2 algorithm

nxtadd = 0;

for(c = 0; buf[c]; c++)

{

add = nxtadd;

if((buf[c] - '0') % 2)

nxtadd = 5;

else

nxtadd = 0;

buf[c] = ((buf[c] - '0') / 2) + add + '0';

}

// end divide by two algorithm

b++;

}

return 0;

}

/\*

Receives a string buffer and a superint.

Converts the superint to a decimal string and fills the

buffer with the digits.

Example:

X = 356

0th iteration... 356/10 = 35R6

b-------->

X 101100100 = 356

Y 000100011 = 35

R 0 1 2 5 11 1 2 4 9 18 8 16 6

1st iteration... 35/10 = 3R5

b-------->

X 100011 = 35

Y 000011 = 3

R 0 1 2 4 8 17 7 15 5

2nd iteration... 3/10 = 0R3

b-------->

X 000011 = 3

Y 000000 = 0

R 0 0 0 0 1 3

\*/

void si\_to\_dec(struct superint X, char s[])

{

int b;

int R = 0;

struct superint Y = {0,0,0,0};

int c1 = 0;

int c2 = 0;

char buf[80];

while(!iszero(X))

{

// begin division by ten algorithm

R = 0;

setzero(&Y);

for(b = 31; b >= 0; b--) // msb to lsb

{

R = 2 \* R + getbit(X, b); // build remainder

if(R >= 10)

{

setbit(&Y, b, 1); // Y += "1"

R -= 10;

}

else

setbit(&Y, b, 0); // Y += "0"

}

// end division by ten algorithm

// output remainder to next buffer position

buf[c1] = R + '0';

c1++;

// copy Y to X for next division

X = Y;

}

// reverse the decimal string

c1--;

while(c1 >= 0)

s[c2++] = buf[c1--];

s[c2] = 0; // null terminator

// check for null string and insert one '0'

if(!s[0])

{

s[0] = '0'; s[1] = 0;

}

}

/\*

Receives a superint and prints it to the console.

Uses the value of format to determine output format.

"%b" binary format

"%d" decimal format

\*/

void si\_printf(char format[], struct superint si)

{

char buf[80];

int i;

if(!strcmp(format, "%b"))

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

putch(getbit(si, i) + '0');

if(!strcmp(format, "%d"))

{

si\_to\_dec(si, buf);

printf(buf);

}

}

void main()

{

char buf[80];

struct superint si;

int x;

strcpy(buf, "9");

setzero(&si);

dec\_to\_si(&si, buf);

si\_printf("%b", si);

printf("\n");

si\_printf("%d", si);

printf("\n");

strcpy(buf, "356");

setzero(&si);

dec\_to\_si(&si, buf);

si\_printf("%b", si);

printf("\n");

si\_printf("%d", si);

printf("\n");

strcpy(buf, "4294967295");

setzero(&si);

dec\_to\_si(&si, buf);

si\_printf("%b", si);

printf("\n");

si\_printf("%d", si);

printf("\n");

strcpy(buf, "4294967296");

setzero(&si);

dec\_to\_si(&si, buf);

si\_printf("%b", si);

printf("\n");

si\_printf("%d", si);

printf("\n");

}