Floating Point Conversion

By your friendly neighborhood Bilal.

https://cs.brown.edu/courses/cs033/docs/guides/x64_cheatsheet.pdf

https://www.cs.uaf.edu/2017/fall/cs301/reference/x86_64.html

isinteger() (validates a string of numbers if its an integer)

atol() (converts a string of numbers into a long integer)

Now we all (well most of us) completed assignment 2 and have used both of these.

- We use isinteger() to verify that our input was an integer.
- We use atol() (or atolong) to convert the ascii characters of the string into a long integer.
- https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/
- Calculator: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Here is how it Worked.

- String a = “355”;
- String b = “123fdgh”;

- Bool a_valid = isinteger(a); - returns true because string a is an int.
- Bool b_valid = isinteger(b); - returns false because string b is not an int.

Once a string is validated as an int we can call the atol function to convert it into a long integer!

- long int num = atol(a);
  cout << num << endl;
  output: 355 as int not as string.

Comment for Slide 3

- Source: https://www.wikihow.com/Convert-a-Number-from-Decimal-to-IEEE-754-Floating-Point-Representation
- Single precision has 32 bits total that are divided into 3 different subjects. These subjects consist of a sign (1 bit), an exponent (8 bits), and a mantissa or fraction (23 bits).
- Double precision, on the other hand, has the same setup and same 3 parts as single precision; the only difference is that it will be larger and more precise number. In this case, the sign will have 1 bit, the exponent will have 11 bits and the mantissa will have 52 bits

Now what if the string was a floating point number instead of an integer?

**Surprise it is!!

Note: The exponent and mantissa on their own have no meaning, but together they create a floating point number.

We Use Different Functions For Floats.

isfloat() (validates a string of numbers if its a floating point number)

atof() (converts a string of numbers into a floating point number)

- We use isfloat() to verify our input was a floating point .
- We use atof() to convert the ascii characters of the string into a floating point number.
- *Note: scientific notation but in binary*

So What Is The Problem?

These exist as library functions:

- Isinteger()
  isdigit()

⇐ by Professor Holliday library function.

- Atol()
- Atof()

This does not exist as a library function:

Isfloat()

Note: scientific notation but in binary

Solution

- Take Professor Holliday’s isinteger() function (or create your own) and include the only difference between integers and floats.

A DECIMAL!

- Therefore instead of validating the input of just a digit (‘0’, ’1’, ’2’, ’3’, ’5’, ’6’ ,’7’ , ’8’, ’9’), as in isdigit() and isinteger(), include the input of a decimal (‘.’)!

Note: scientific notation but in binary

From isinteger() to isfloat()

```
bool isinteger(char w[])
{
    bool result = true;
    int start = 0;
    if (w[0] == '-' || w[0] == '+') start = 1;
    unsigned long int k = start;
    while( !(w[k]=='\0') && result )
    {
        result = result && isdigit(w[k]);
        k++;
    }
    return result;
}
```

Note: scientific notation but in binary

From isinteger() to isfloat()

```
bool isinteger(char w[])
{
   bool result = true;
   int start = 0;
   if (w[0] == '-' || w[0] == '+') start = 1;
   unsigned long int k = start;
   while( !(w[k]=='\0') && result )
   {
       result = result && isdigit(w[k])
           || result && (w[k] == ‘.’);
       k++;
   }
   return result;
}
```

Note: scientific notation but in binary

From isinteger() to isfloat()

```
bool isfloat(char w[])
{
    bool result = true;
    bool onedecimal = false;
    int start = 0;
    if (w[0] == '-' || w[0] == '+') start = 1;
    unsigned long int k = start;
    while( !(w[k]=='\0') && result )
    {
        if (w[k] == ‘.’ && !onedecimal) onedecimal = true;
        else {
        result = result && isdigit(w[k])
           ||  result && (w[k] == ‘.’); }
        k++;
    }
    return result && onedecimal;}
```

Note: scientific notation but in binary

Atof()

- Exactly the same way you call atol()

  -
    ```
    mov rax, 0
    mov rdi, rsp
    call atol
    ```
- Only difference is passing a 1 to rax instead of 0 to indicate the passing of a float number and no need to copy rsp to rdi because atof will return its value into xmm0 register

  -
    ```
    mov rax, 1
    call atof
    ```

Note: scientific notation but in binary

Wait so you mean to tell
me asssembly is easy and fun?
ALWAYS HAS BEEN
imgflip.com

I Believe In You All!

But nonetheless any questions?