Operating System

Friday, April 8, 2022 11:08 PM

"Civilization progresses by extending the number of operations that we can perform without thinking about them."

- *Alfred North Whitehead, Introduction to Mathematics (1911)*

12.1 Background

let n = keyboard.readInt("Enter a number: ")

First we display the "Enter a number: " prompt

- *Initializing a char[] to 'E', 'n', 't', 'e',..., ':', ' '*
- *Next we must realize each of these characters onto the screen while also moving the cursor forward.*

After this we wait for user input. Taking in one keyboard entry (hopefully it is a digit key!)

- *Capture the keystroke*
- *Get a single character input*
- *Append these characters to a string*
- *Convert string into integer.*

12.1.1 Mathematical Operations

- *add*
- *subtract*
- *multiply*
- *divide*

Efficiency First

*If you naively implement multiplication of x * y as an iterative method*

for 1...x (sum = sum + y)

Then, on a 64 bit machine

x = 0111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111

will run for a long time.

We discuss these issues throughout the chapter. The battle between efficiency and readability will be a constant battle (although if an OS is then compiled down into a lower level language, readability doesn't exactly matter as much as efficiency)

Multiplication

*356 * 73 = 356 * 3 + 356 * 70 =*

Is the method we will use to multiply as it is only iterative over n bits where n is the number of bits used to represent each number

See book for diagram

```
multiply(x, y):
    sum = 0
    shiftedx = x
    for i = 0…n-1 do:
        if ( ( i-th bit of y ) == 1 )
            sum = sum + shiftedx
        shiftdex = 2 * shiftedx
    return sum
```

O(n) complexity where n = the number of bits necessary to represent an integer

Division

Division will work the same as normal long division except instead of accelerating the subtractions by powers of ten, it will do so by powers of two

```
divide(x, y):
    if (y > x) return 0;
    q = divide(x, 2 * y)
    if ( (x - 2 * q * y) < y )
        return 2 * q
```

```
    else
        return 2 * q + 1
```

Square root

Taking square roots (to no surprise) will be done by a simple binary search. As there are not decimal numbers in the jack language, $\sqrt{x} = n$ or $n + 1$ There is no in between

```
sqrt(x):
    y=0
    for j = ( n / 2 - 1 )…0 do
        if ( y + 2^j ) <= x then y = y + 2^j
    return y
```

12.1.2 String

Strings are implemented as a char[]

12.1.3 Memory

heapBase = 2048 (after the stack ends)

Memory Alloc

```
init():
    freeList = heapBase
    freeList.size = heapSize
    freeList.next = 0

alloc(size):
    search freeList using best-fit or first-fit heuristics to
    obtain a
        segment with segment.size >= size + 2
    if no such segment is found return failure
    block = base address of the found space
    update the freeList and the fields of block to account
    the
        allocation
    return block

deAlloc(object):
    append object to the end of the freeList
```

Peek and Poke

Memory.peek(address)
returns the value at address in RAM

Memory.poke(address, value)
sets RAM[address] = value

12.1.4 Graphical Output

Pixel Drawing

```
drawPixel(x, y):
    using x and y compute the RAM address where the pixel is
        represented
    Using Memory.peek get the 16-bit value of this address
    using some bitwise operation, set (only) the bit that
    corresponds
        to the pixel to the current color.
    Using Memory.poke, write the modified 16-bit value back
    to the RAM
        address
```

Line Drawing

```
drawline(x1, x2, y1, y2):
    set x = x1, y = y1 compute dx dy
    // a and b track how many times we went right and up so
    far
    a = 0, b = 0
    diff = 0
    while ( ( a <= dx ) and ( b <= dy ) )
        drawPixel(x + a, y + b)
        // Goes right or up
        if ( diff < 0 ) { a = a + 1, diff = diff + dy }
        else            { b = b + 1, diff = diff - dx }

drawCircle(x, y, r):
    for each dy = -r to r do
        drawLine(x − √(r² − dy²), y + dy, x − √(r² − dy²), y + dy)
```

where $(x - \sqrt{r^2 - dy^2}, y + dy)$ is a point

12.1.5 Character Output

Fonts

Made from an 8 x 11 pixel grid. 95 characters that are mapped Which includes symbols, characters, and 0-9 digits

The specification is within the appendix of the book

Cursor

keep track of where a character is to be printed on the screen.

if no new line then you will simply just be incrementing the cursor otherwise you will have to enact a new line

which will move the cursor one line down

12.1.6 Keyboard Input

- detect which key is being pressed
- capturing single character input
- capturing a multicharacter input

Detecting Keyboard Input

```
keyPressed() {
    // use memory peek to check if anything is being
    // pressed. It returns a Boolean
    Memory.Peek(address of KBD) == 0
}

readChar() {
    display the cursor
    // wait for keyboard input
    while( keyPressed() == 0 ) {
        do nothing
    }
    c = code the currently pressed key
    while( keyPressed() != 0 ) {
```

```
        do nothing
    }
    display c at the current cursor position
    advance the cursor
    return c
}

readLine(message) {
    display the message
    str = empty string
    while ( 1 ) {
        if (c == newline ) {
            display newLine
            return str
        } else if (c == backSpace ) {
            remove the last character from str
            move the cursor back
        } else {
            str.appendChar(c)
        }
    }
    return str
}
```