Future of EASM

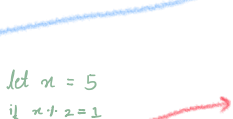
Styles

```
main
    Int 8
    [ 2 ] "hello world"
    Str_len
    Add
    Print
end
```

[2] - accessing twice will make it disappear

```
main {
once Str "hello"
    Str_len
    Print
}
```

```
let x = 5
if x % 2 = 1
    Print "Odd!"
else
    Print "Even"
```



```
main
    Int 5
    Mov 0
    Load 0
    Int 2
    Mod
    Int 1
    Int_Cmp
End
```

```
odd_then
    "Odd"
    Print
end
```

```
even_else
    "Even"
    Print
end
```

For every Branches or scopes, the compiler will generate a new section for it

Other Style?

```
Int_Cmp
Mod
Load x
Int 2
Int 1
```

```
5
mov 0
Load 0
2
Mod
1
=
```

fuels empty

```
= (-1 (load 0; 2); 1)
equ (mod (load 0; 2); 1)
(= (mod x 2) 0)
```

maybe we gotta do some loops?

Not good

* Maybe Try to implement Lua like syntax
That is, functional Programming.
That format is much easier to parse
and convert to EASM like code.

* OK, so traditionally lisp langs are interpreted, but
maybe Bringing it down to Bytecode could be
efficient.

Popular lisP Impls = Jamt,
Schume

* Maybe Hybrid lisp?

```
(fn fib [n]
  (if (< n 2)
    n
    (+ (fib (- n 1)) (fib (- n 2)))))
```

can we further simplify it?

```
fn fib [n](
  if (< n 2)
    n
    + (fib (- n 1)) (fib (- n 2))
)
Print (fib 10)
```

Or what if for function Body
you replace Braces with Curlies

yah so in this case we skipped Braces at a few Places

```
fn fib [n] {
    if (< n 2)
      n
      + fib (- n 1) fib (- n 2)
}
```

So what made lisp languages use curve Braces in place of curly Ones?