Future of EASM

Styles

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

[2] - accesing 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                    odd_then          even_else
    Int 5                   "Odd"             "Even"
    Mov 0                   Print             Print
    Load 0              end               end
    Int 2
    Mod
    Int 1
    Int_Cmp
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
```

not goo

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

feels empty

```
= (% (load 0; 2); 1)

equ (mod (load 0; 2); 1)

(= (mod x 2) 0)
```

maybe we gotta do some lisp?

* Maybe Try to implement Lua like syntax
  That is, functional Programming.
  That format is much easier to parse
  and convert to E4SM like code.

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

Popular lisP Impls = Janet,
                     Scheme

* 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

yeah 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?