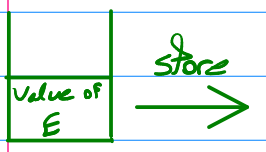
316 10-08-09

push <id>
push <const>
pop <id>
add
sub
mul
div
neg

→ evaluation of <E> id = E;

← assignment

code to evaluate E // the value of E is at the top of the stack
pop id



<E> → <term> [(+|-) <E>]

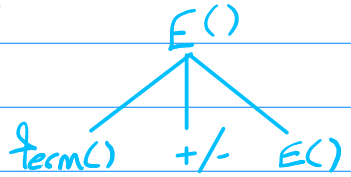
<term> → <primary> [(*|/) <term>]

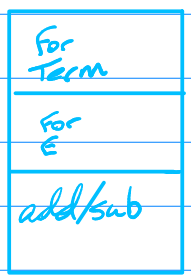
<primary> → <id> | <int> | <float> | "(" <E> ")" | - <E>

```
void primary()
{
  if (t is <id>, <int>, <float>)
  {
    emit "push t";
    getToken();
  }
  else if (t is "-")
  {
    getToken();
    E(); // will generate instructions for <E>
    emit "neg";
  }
  else if (t is "(")
  {
    same as before
  }
  ⋮
}
```

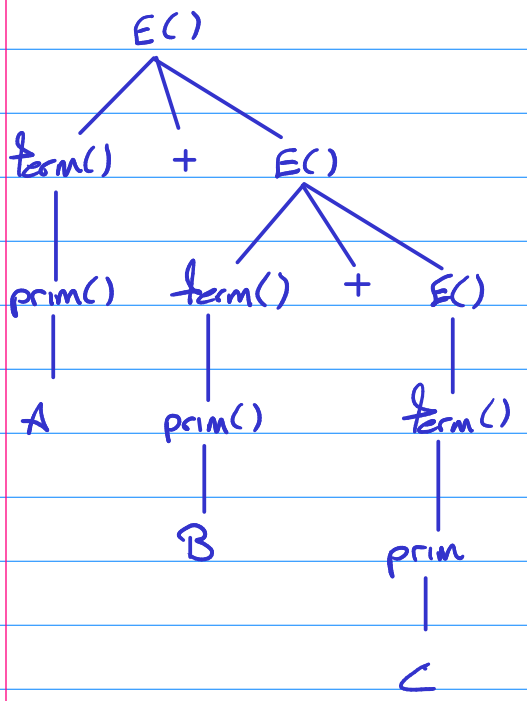
```
void E()
{
  term();
  if (t is "+" || t is "-")
  {
    saved_t = t;
    getToken();
    E(); // will generate instructions for E
    if (saved_t = "+") emit "add";
    else emit "sub";
  }
}
```

(code for term() is analogous)





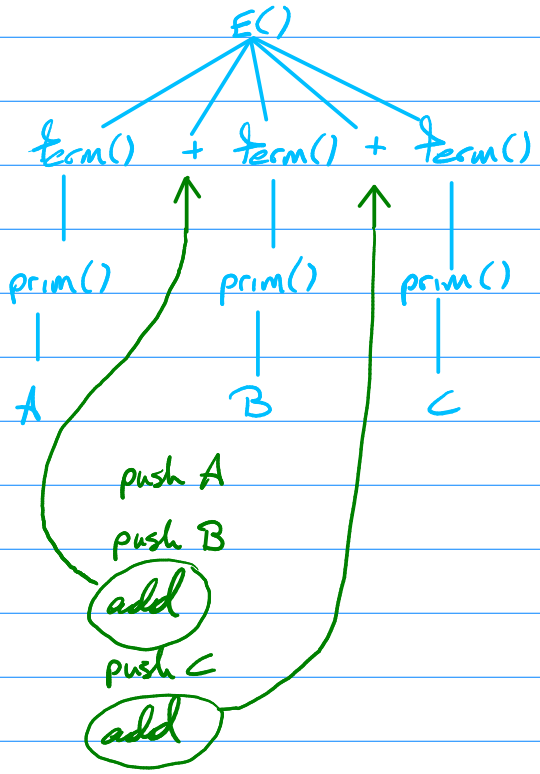
A + B + C



Iterative version

push A	<E> → <term> {(+|-) <term>}
push B	<term> → <primary> {(*|/) <primary>}
push C	<primary> → same as before
add	
add	
right associative	

```
Void E()
{ term();
  while (t is "+" || t is "-")
  {
    saved_t = t;
    getToken();
    term();
    if(saved_t = "+") emit "add";
    else  emit "sub";
  }
}
```



Iterative E	Recursive E
code for term-1	code for term-1
code for term-2	code for term-2
add/sub	code for term-3
code for term-3	⋮
add/sub	add/sub
⋮	add/sub
	⋮

<statement> → <assignment> | "{" <sList> "}"

<assignment> → <id> = <E> ;

<sList> → {<statement>}+

No instruction emission in statement() and sList()

```
void assignment()
{
  if (t is <id>)
  {
    saved_id = t;
    getToken();
    if(t is "=")
    {
      getToken();
      E(); // will generate instructions for E
      if (t is ";")
      {
        emit "pop saved_id";
        getToken();
      }
      else { same as before ..... }
    }
    else
  }
  else
}
```

Formal Description Of Dynamic Semantics

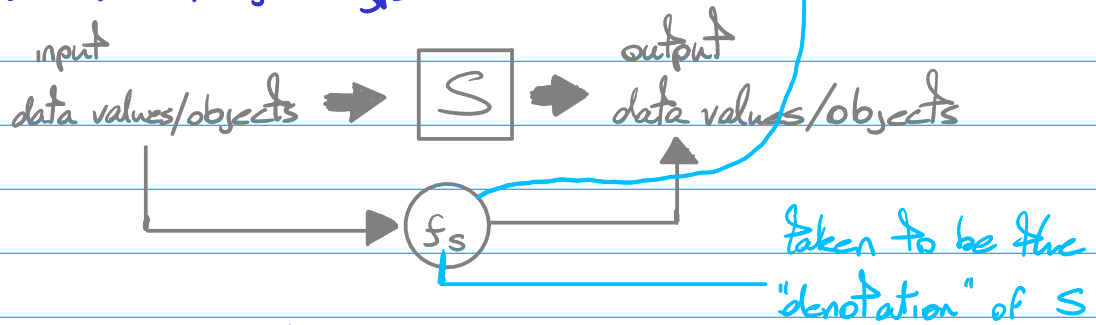
(meaning)

- Semantics Of Programming Languages:
  - Static Semantics - meaning of data values/types
    - meaning of arrays, class objects, modules, pointers/references
  - Dynamic Semantics - meaning of execution of statements
    - rigorous description of exact effect of execution of statements.
    - assignment statements
    - control structures
      - conditionals
      - loops
      - switch
      - function calls

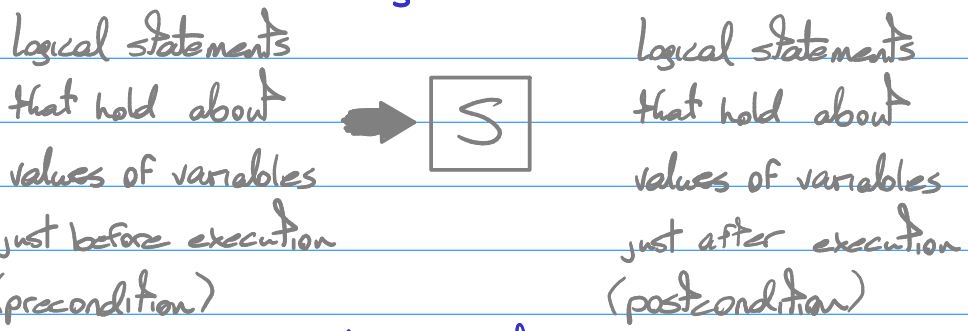
3 main methods:

① Operational Semantics
- Describe effect of statement execution by step-by-step operations

② Denotational Semantics
- Describe effect of statement execution by mathematical functions from data values/object types



③ Axiomatic Semantics (Logical)



used for program verification / correctness proof