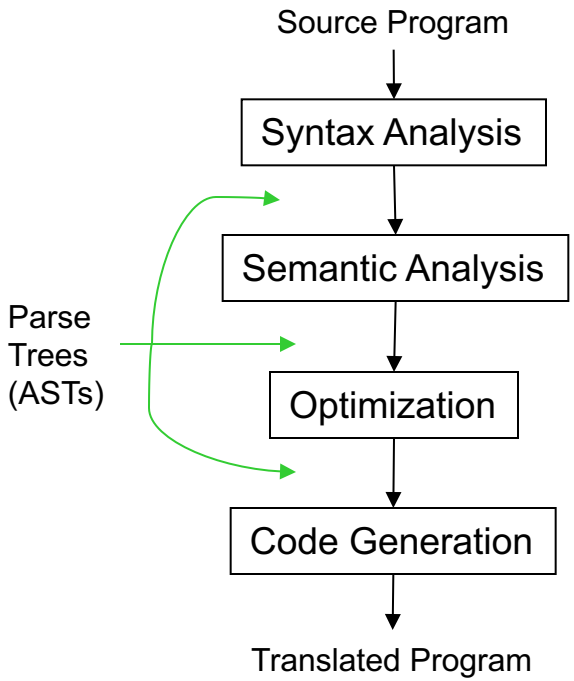
The Anatomy of a Compiler



Recognize the structure of a source program, generate parse tree

Recognize/validate the meaning of a source program

Recognition Phases

Reorganize the parse tree/AST to make computations more efficient

Translate parse tree/AST into low-level language

Code Generation Phases

Observations:

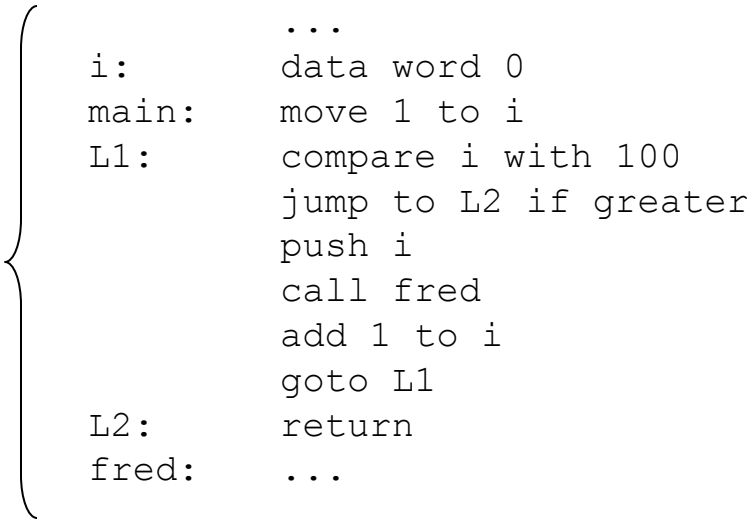
-Language definitions have two parts: syntax and semantics

-Compilers have two phases which deal with each of these language definition components: syntax analysis, semantic analysis.

Compilation Example

Translating a C-like language to assembly language

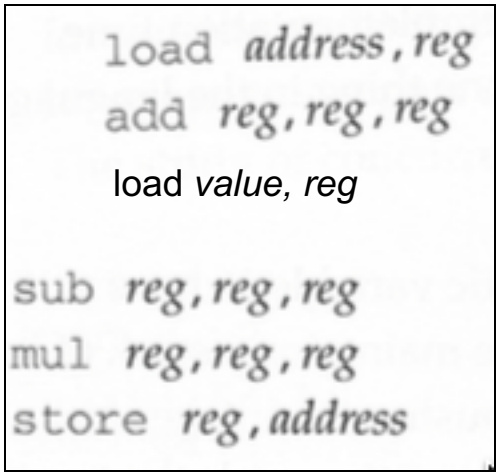
```
int i;

void main () {
   for (i = 1; i <= 100; i++)
       fred(i);
}
```



```
            ...
i:          data word 0
main:       move 1 to i
L1:         compare i with 100
            jump to L2 if greater
            push i
            call fred
            add 1 to i
            goto L1
L2:         return
fred:       ...
```

Compilation Example

Assembly Language



Three registers: *r1, r2, r3*

consider: 3*2+5

Assembly Code:

load 3,r1
load 2,r2
mul r1,r2,r1
load 5,r2
add r1,r2,r1

Assignments

- Read chap 4
- Exercise #3 – see the website