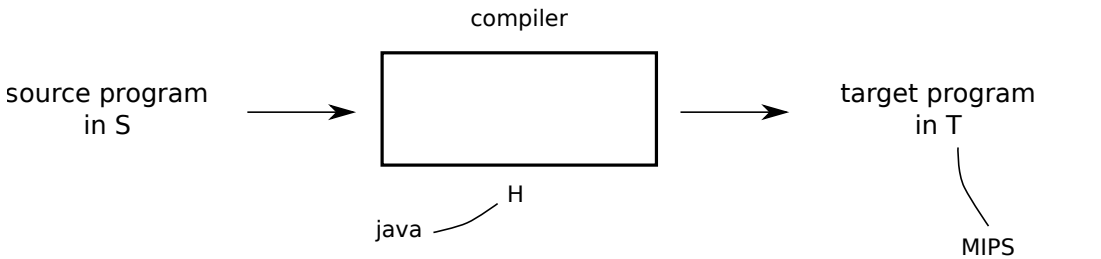
Welcome to
CS 536

Course website:
pages.cs.wisc.edu/~cs536-1

Email:
davidson@cs.wisc.edu



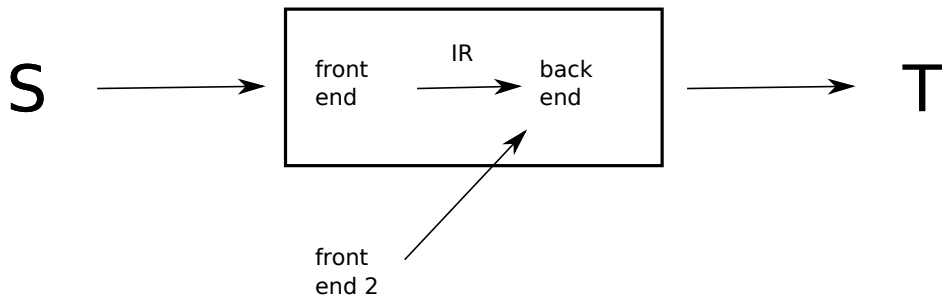
A compiler is...
- a recognizer of language S
- a translator from S to T
- a program in language H

what will we name S?

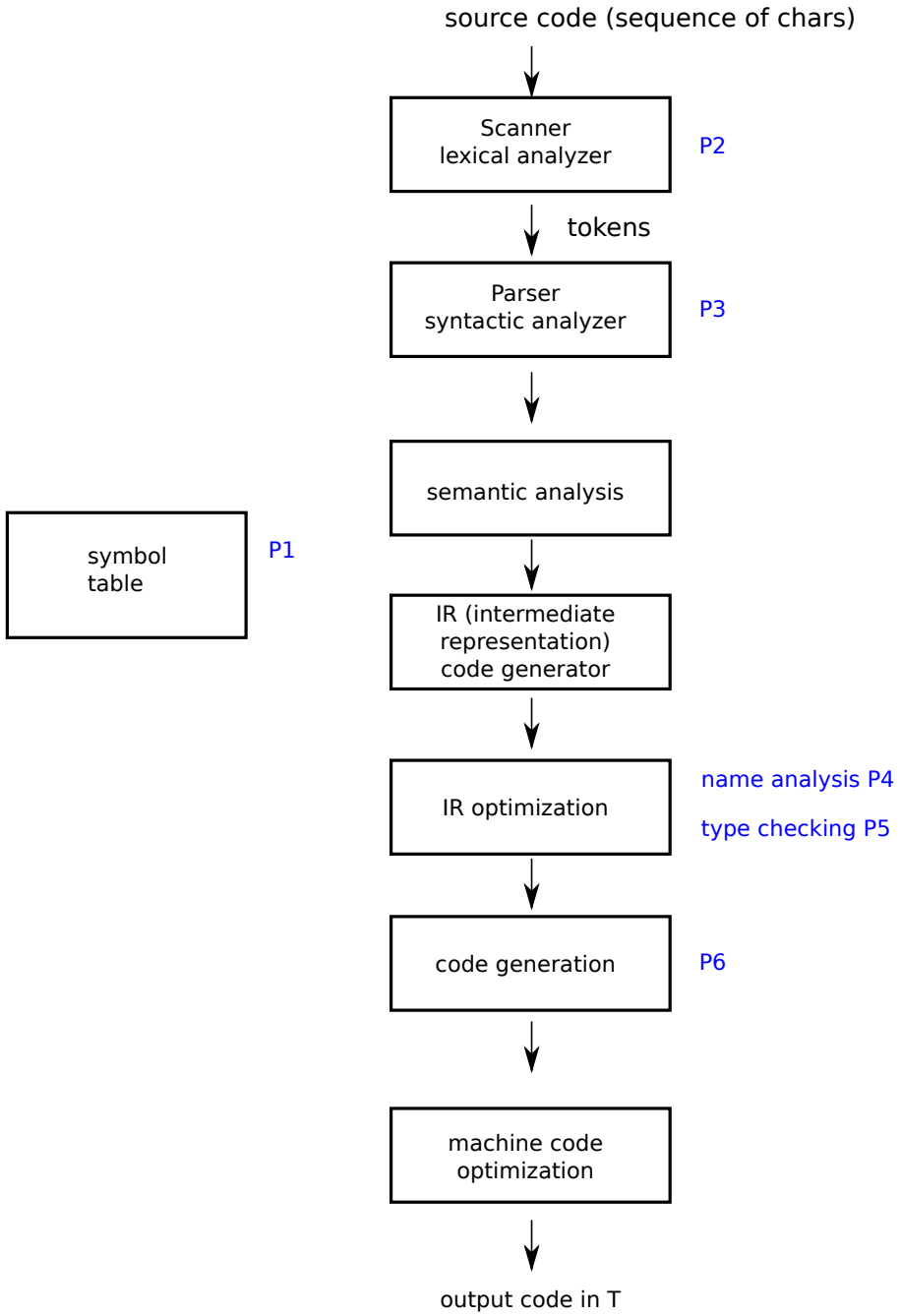
~~C-min~~ C minor

BuckC C flat

WIS-C x43



LLVM


source code (sequence of chars)
Scanner
lexical analyzer
P2
tokens
Parser
syntactic analyzer
P3
semantic analysis
symbol
table
P1
IR (intermediate
representation)
code generator
IR optimization
name analysis P4
type checking P5
code generation
P6
machine code
optimization
output code in T

Scanner

input: characters from source program

output: sequence of tokens

Group characters into lexemes
identify and ignore white space, comments, etc.

Error Checking:
"bad" characters such as ^
unterminated strings "Hello
int literals that are too large

Parser

input: sequence of tokens from the scanner

output: AST (Abstract Syntax Tree)

Error Checking:

syntax errors x = y * = 5;

(possibly) "static semantic" errors such as the use of an undeclared variable

Semantic Analysis

input: AST

output: annotated AST

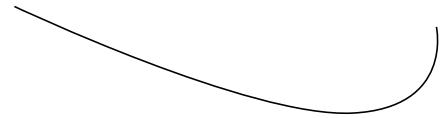
Name analysis:
- process declarations & uses of variables
- enforces scope

type checking:
- checks types
- augments AST w types

scope example

```
{
int i = 4;
i = i + 2;
}
i = 5;
```

i is out of scope here



Example

source

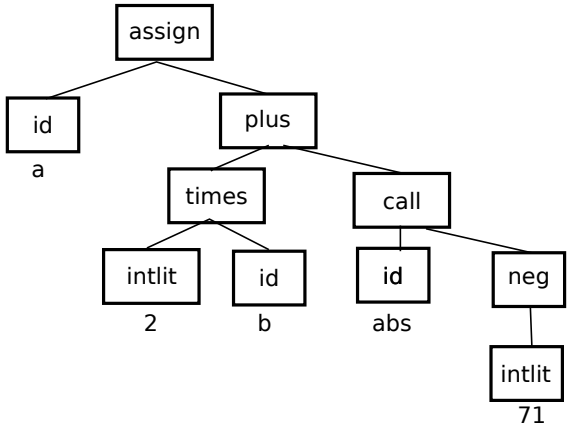
```
a = 2 * b + abs(-71)
```

scanner

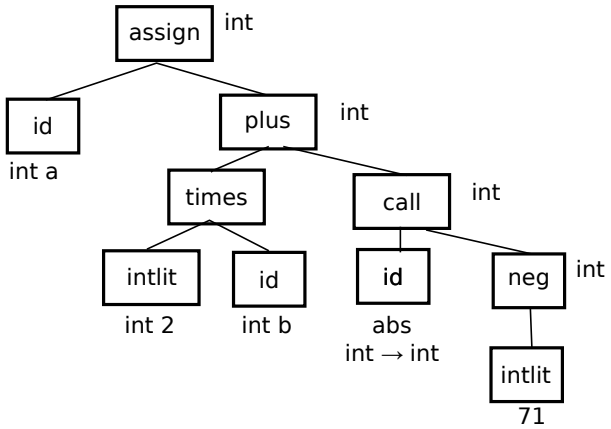
ident	assign	int lit	times	ident	plus	ident	lparens	minus	int lit	rparens
(a)		(2)		(b)		(abs)			(71)	

parser

the AST



semantic analysis

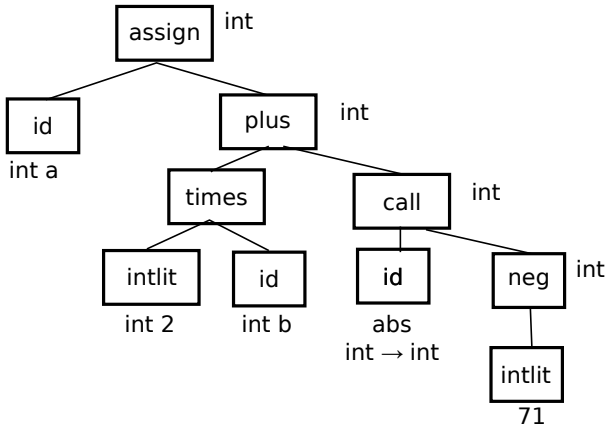


symbol table

a	var	int
b	var	int
abs	fun	int → int

Example (2)

semantic analysis



symbol table

a	var	int
b	var	int
abs	fun	int → int

code generation

```
temp1 = 0 - 71
move temp1 param1
call abs
move ret1 temp2
temp3 = 2 * b
temp4 = temp3 + temp2
a = temp4
```