

# Language Processors

COMS W4115

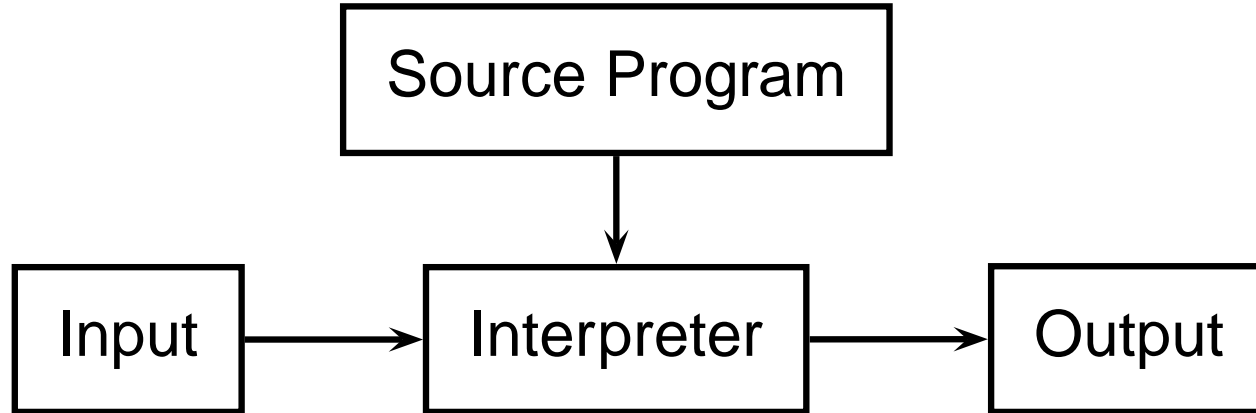
Prof. Stephen A. Edwards

Fall 2007

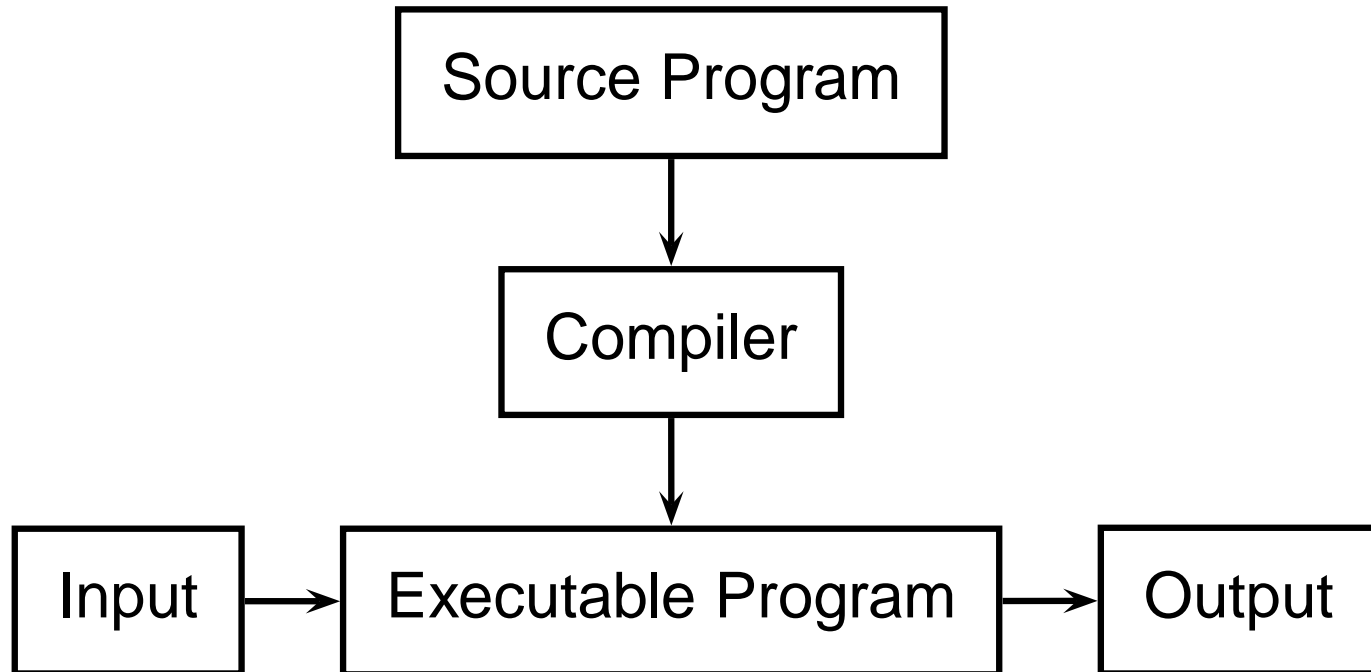
Columbia University

Department of Computer Science

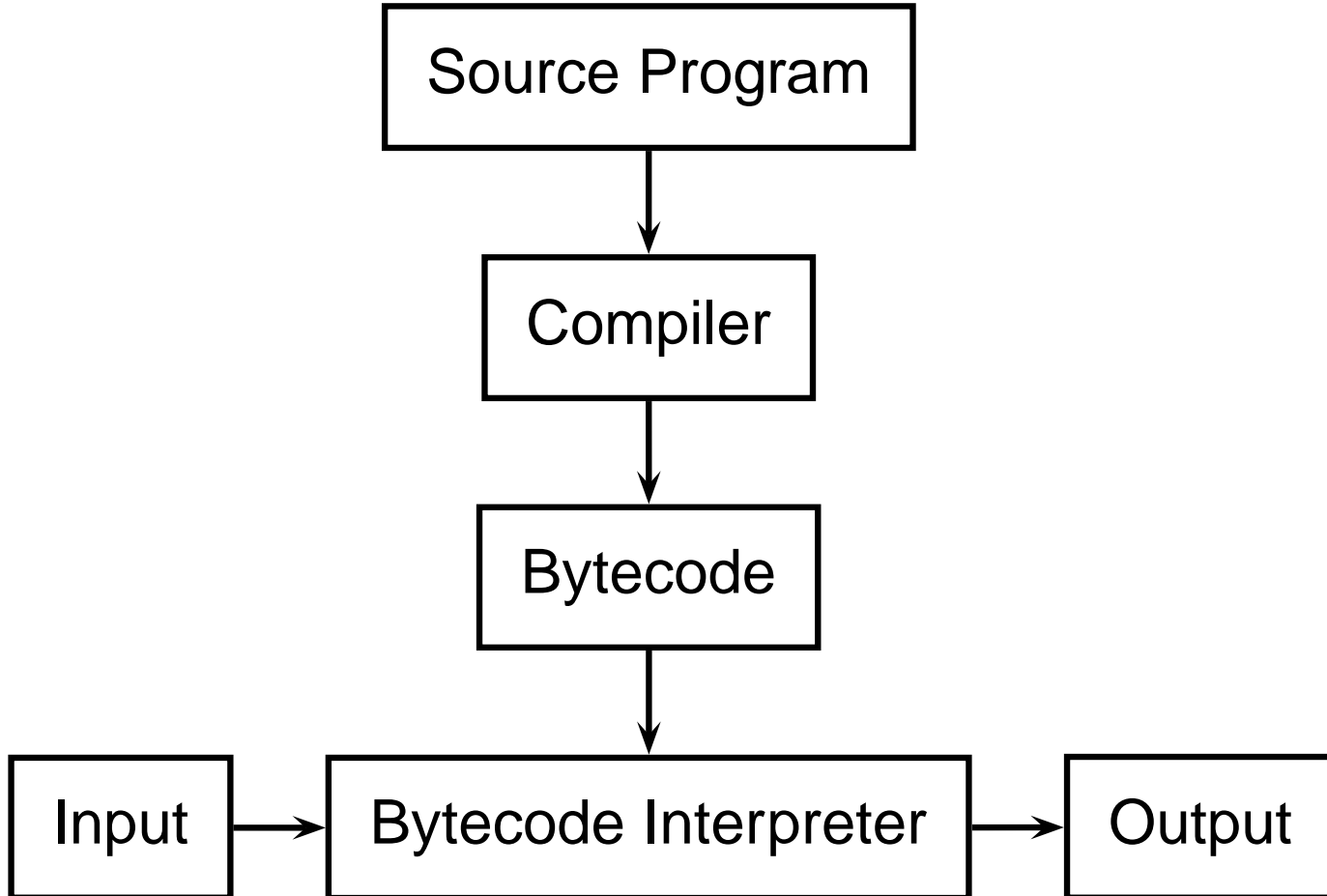
# Interpreter



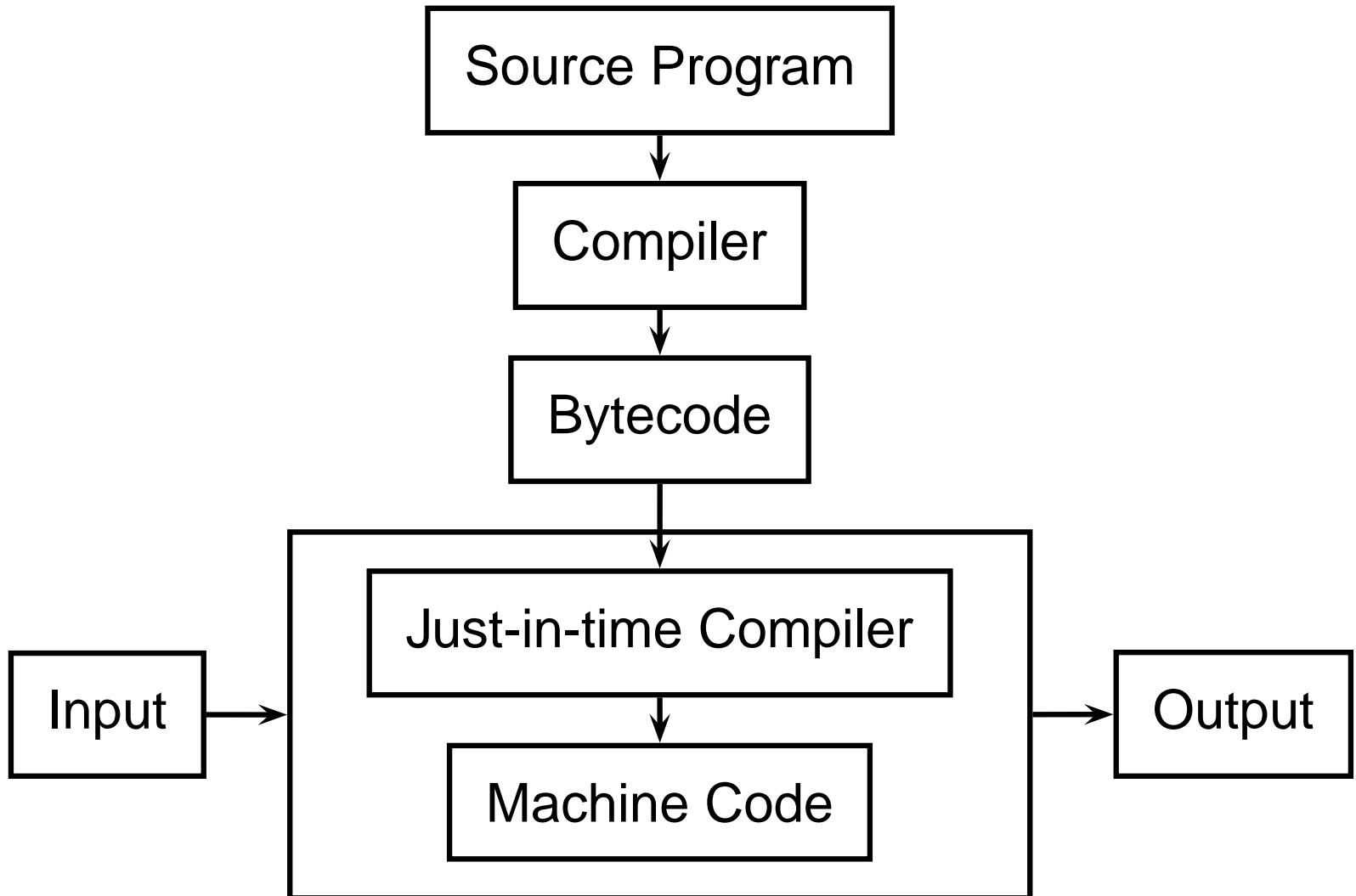
# Compiler



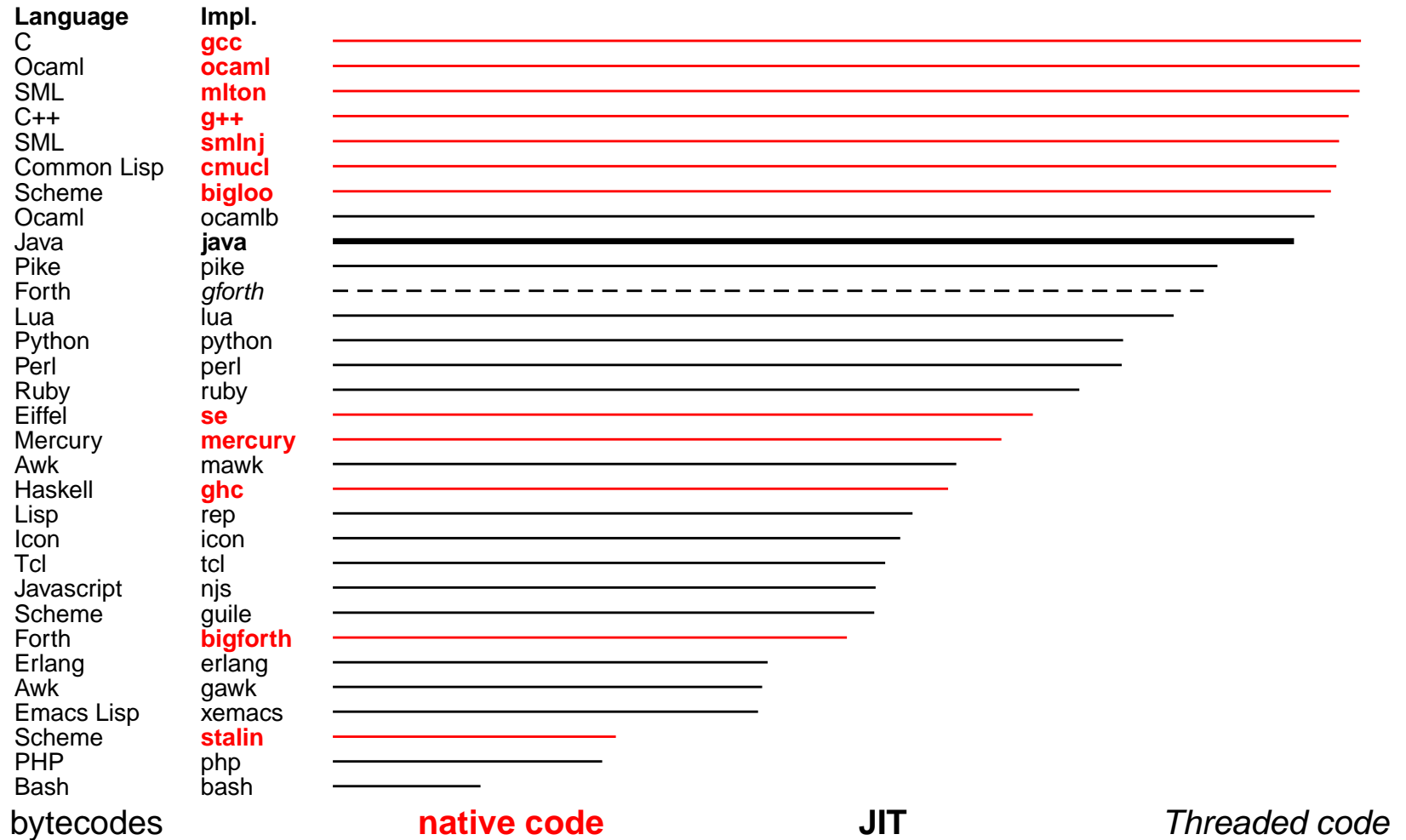
# Bytecode Interpreter



# Just-in-time Compiler

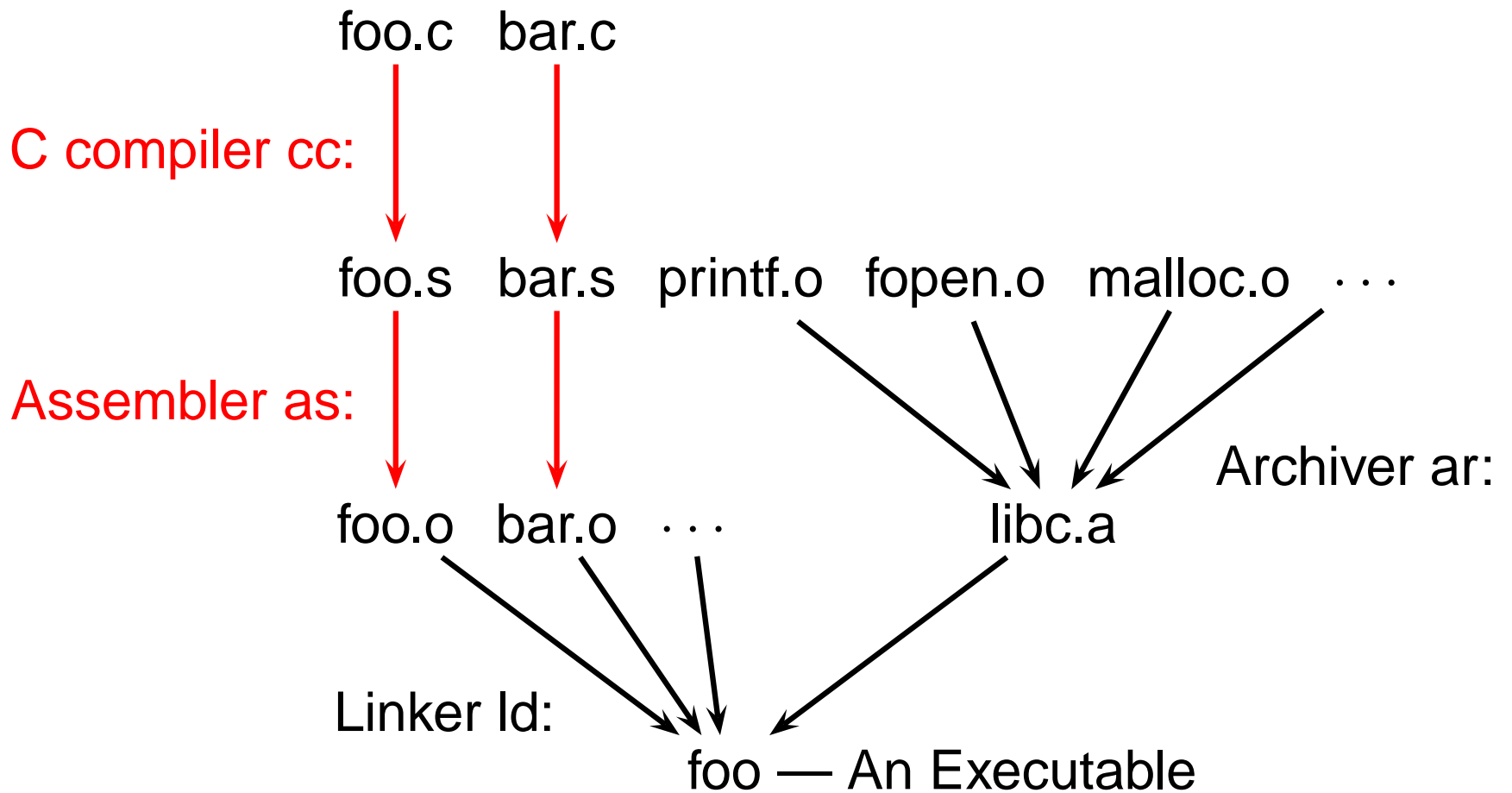


# Language Speeds Compared



<http://www.bagley.org/~doug/shootout/>

# Separate Compilation



# Preprocessor

“Messages” the input before the compiler sees it.

- Macro expansion
- File inclusion
- Conditional compilation



# The C Preprocessor

```
#include <stdio.h>
#define min(x, y) \
    ((x)<(y))?(x):(y)
#ifdef DEFINE_BAZ
int baz();
#endif
void foo()
{
    int a = 1;
    int b = 2;
    int c;
    c = min(a,b);
}
```

`cc -E example.c` gives

```
extern int
printf(char*,...);
... many more declarations
from stdio.h

void foo()
{
    int a = 1;
    int b = 2;
    int c;
    c = ((a)<(b))?(a):(b);
}
```

# Compiling a Simple Program

```
int gcd(int a, int b)
{
    while (a != b) {
        if (a > b) a -= b;
        else b -= a;
    }
    return a;
}
```

# What the Compiler Sees

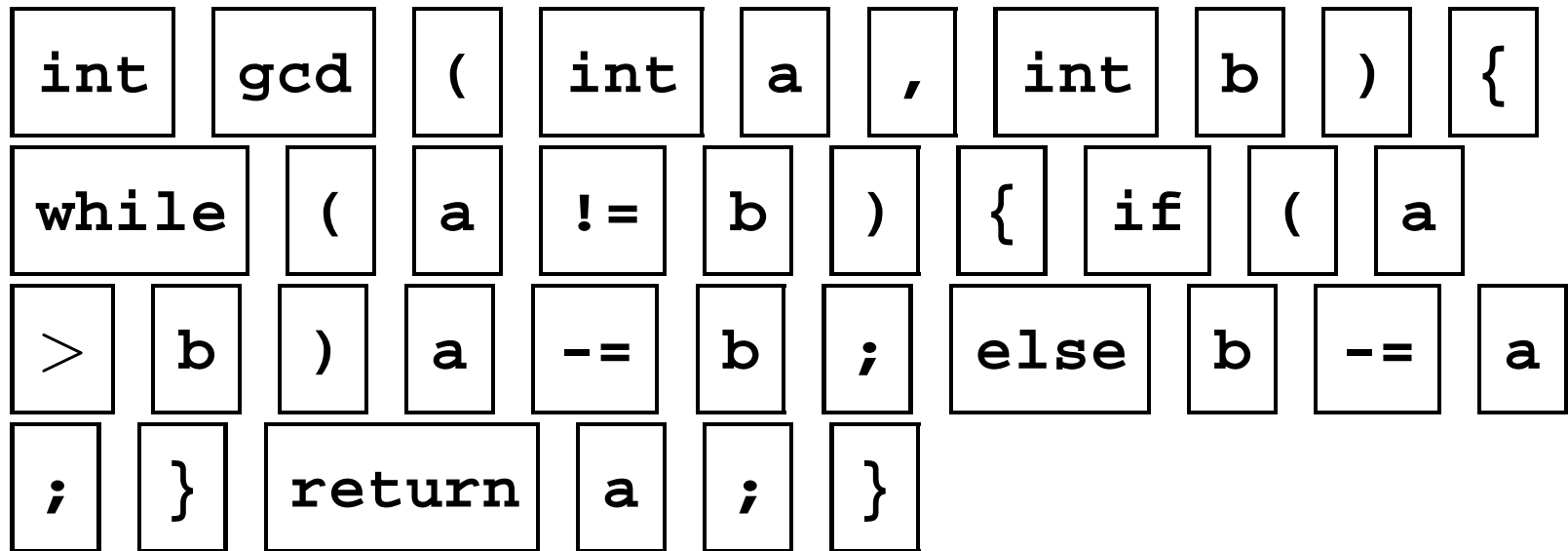
```
int gcd(int a, int b)
{
    while (a != b) {
        if (a > b) a -= b;
        else b -= a;
    }
    return a;
}
```

```
i n t   s p   g c d   (   i n t   s p   a   ,   s p   i
n t   s p   b   )   n l   {   n l   s p   s p   w h i l e   s p
(   a   s p   !   =   s p   b   )   s p   {   n l   s p   s p   s p   s p   i
f   s p   (   a   s p   >   s p   b   )   s p   a   s p   -   =   s p   b
;   n l   s p   s p   s p   s p   e l s e   s p   b   s p   -   =   s p
a   ;   n l   s p   s p   }   n l   s p   s p   r e t u r n   s p
a   ;   n l   }   n l
```

Text file is a sequence of characters

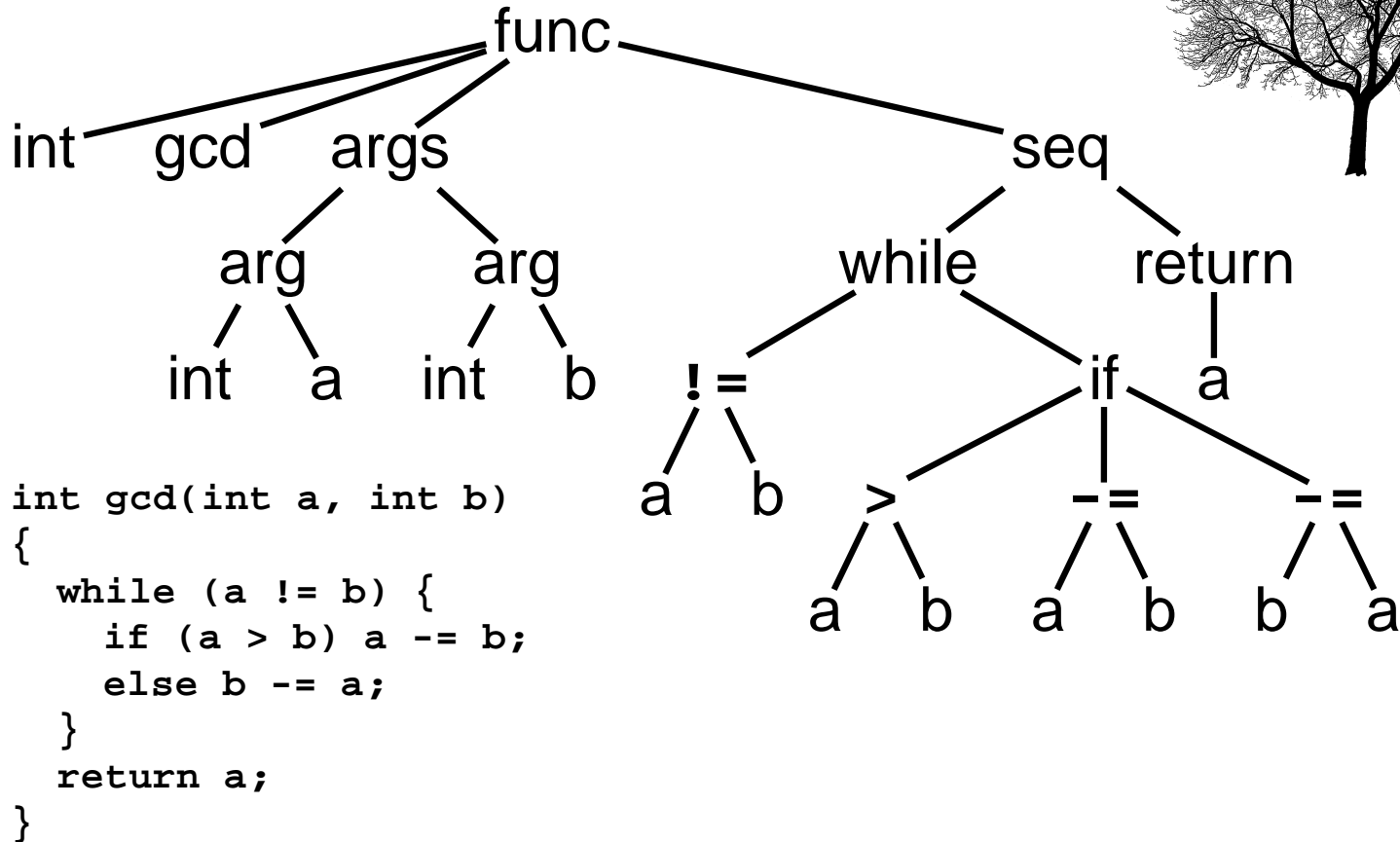
# Lexical Analysis Gives Tokens

```
int gcd(int a, int b)
{
  while (a != b) {
    if (a > b) a -= b;
    else b -= a;
  }
  return a;
}
```



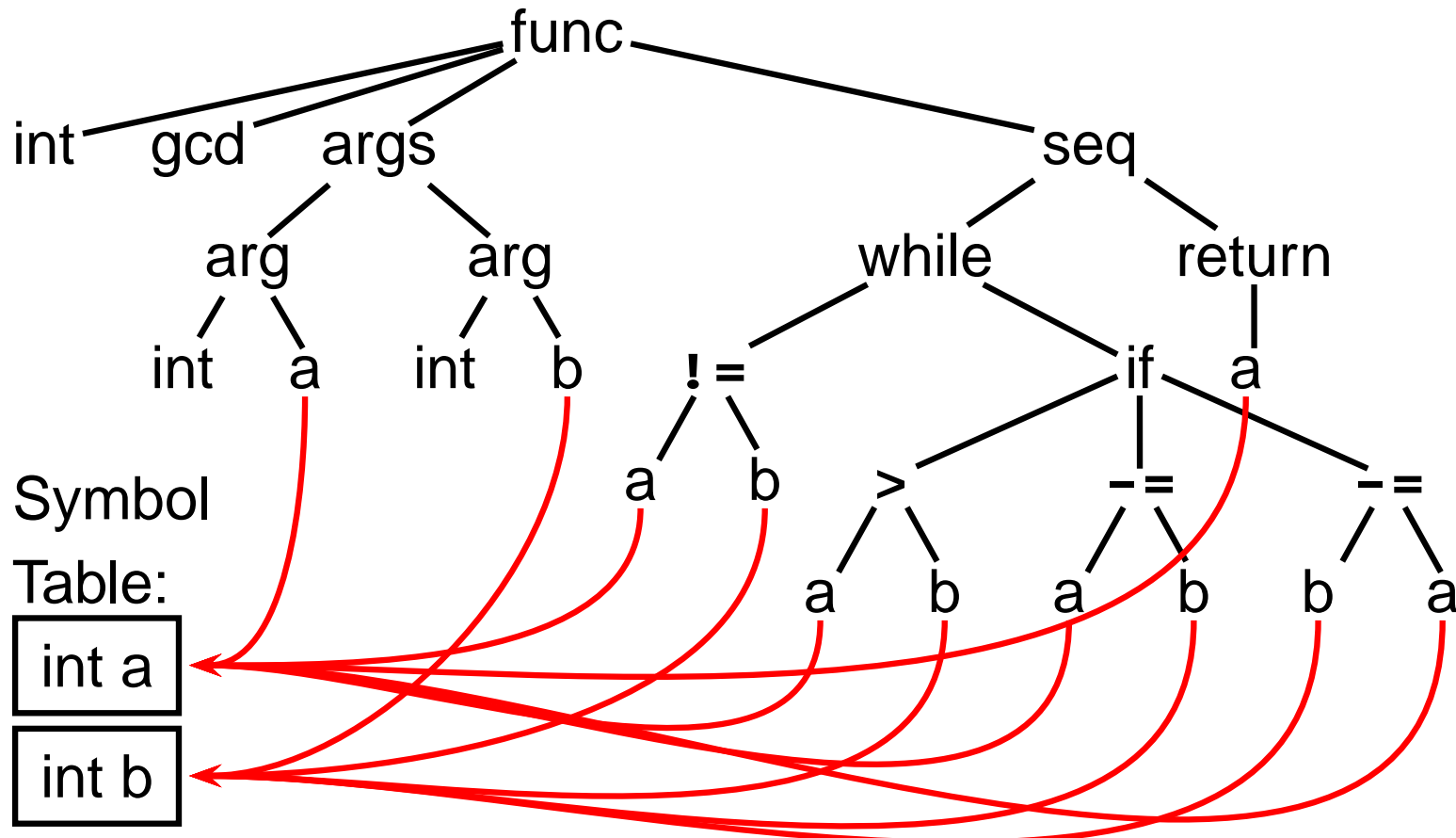
A stream of tokens. Whitespace, comments removed.

# Parsing Gives an AST



Abstract syntax tree built from parsing rules.

# Semantic Analysis Resolves Symbols



Types checked; references to symbols resolved

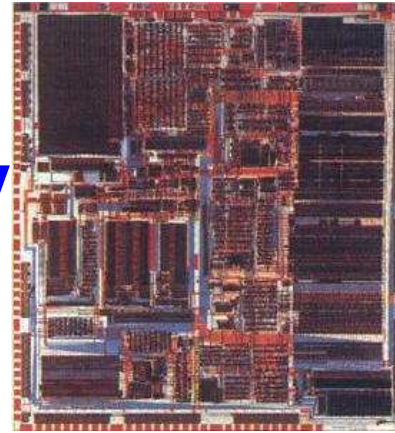
# Translation into 3-Address Code

```
L0: sne    $1,  a,  b
      seq    $0,  $1,  0
      btrue  $0,  L1    % while (a != b)
      sl     $3,  b,  a
      seq    $2,  $3,  0
      btrue  $2,  L4    % if (a < b)
      sub    a,   a,  b % a -= b
      jmp    L5
L4: sub    b,   b,  a % b -= a
L5: jmp    L0
L1: ret    a
```

```
int gcd(int a, int b)
{
    while (a != b) {
        if (a > b) a -= b;
        else b -= a;
    }
    return a;
}
```

Idealized assembly language w/ infinite registers

# Generation of 80386 Assembly



```
gcd:   pushl   %ebp                % Save FP
       movl   %esp, %ebp
       movl   8(%ebp), %eax       % Load a from stack
       movl   12(%ebp), %edx     % Load b from stack
.L8:   cmpl   %edx, %eax
       je     .L3                % while (a != b)
       jle   .L5                % if (a < b)
       subl  %edx, %eax         % a -= b
       jmp   .L8
.L5:   subl  %eax, %edx         % b -= a
       jmp   .L8
.L3:   leave                % Restore SP, BP
       ret
```