* Compilers
* Interpreters
* Objects
* Binding
* Binding Time.

Sept. 10

The gcc Compilation Process

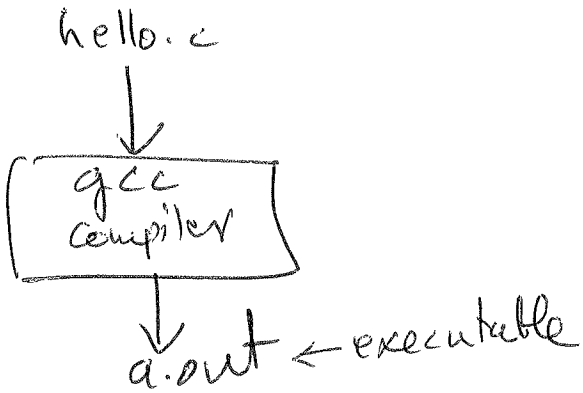
Program: hello.c

```
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

1. Compile and run.

```
gcc hello.c
./a.out
```

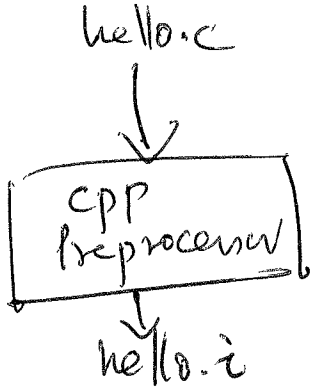


2. Compile and run + name executable

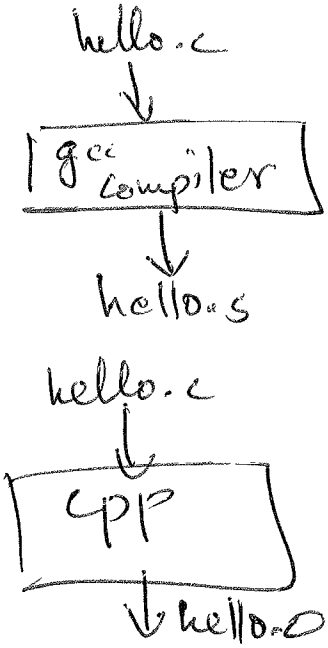
```
gcc hello.c -o hello
./hello
```

3. The Pre-processor (cpp)
   - collects headers of all #includes
   - process macros (#define N 5)

```
cpp hello.c > hello.i
```



4. C program to Assembly (Compiling)



gcc -S hello.s

5. Create relocatable object code

gcc -c hello.c

look at the names in object code

nm hello.o

6. Create executable

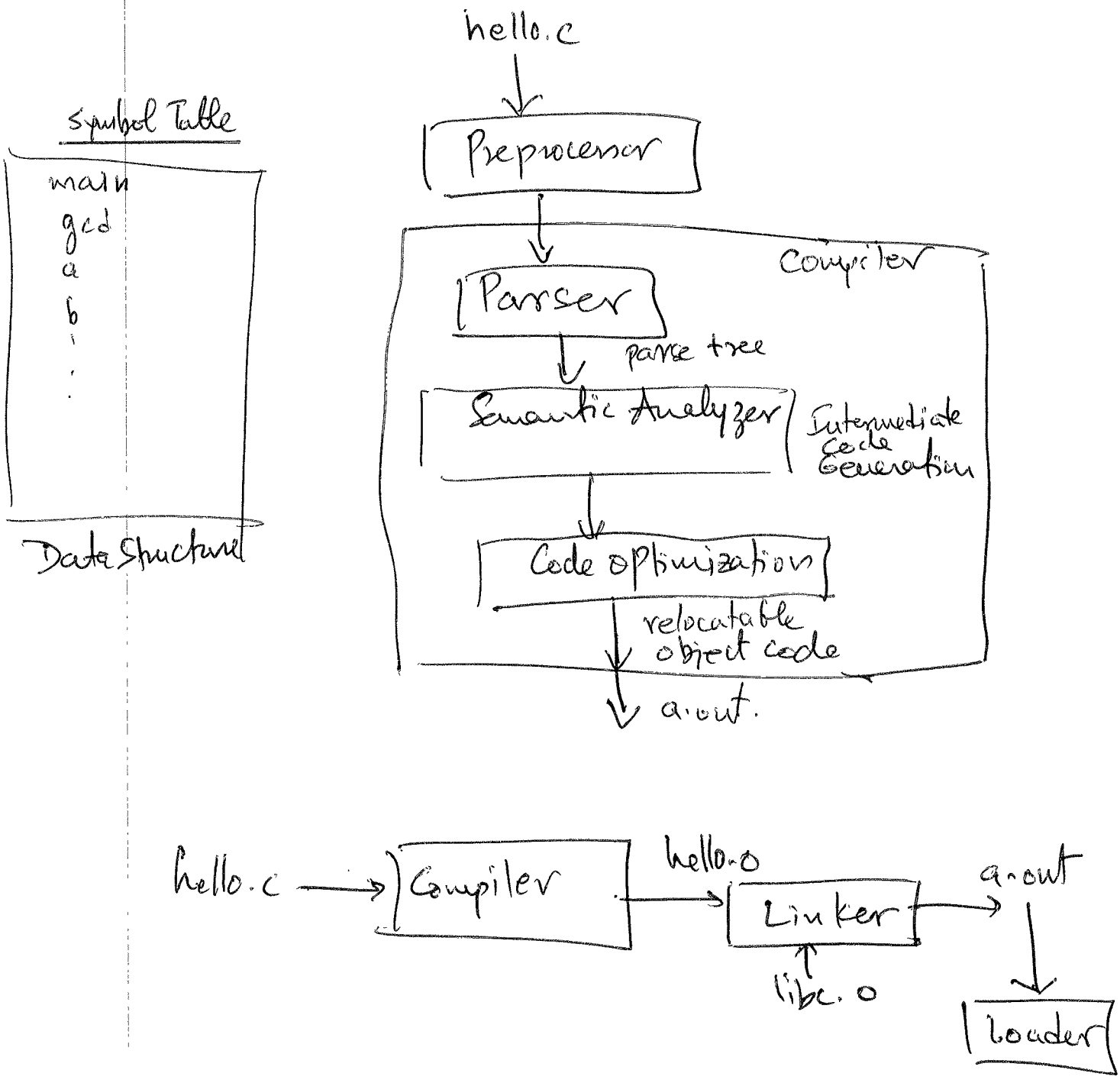
gcc hello.c
gcc hello.c -o hello

| Linker links .o with .o of all libraries

7. Run program

./a.out
./hello

| loader loads program in memory + points PC to it.


hello.c
Preprocessor
Compiler
Parser
parse tree
Semantic Analyzer
Intermediate Code Generation
Code optimization
relocatable object code
a.out.
Symbol Table
main
gcd
a
b
Data Structure
hello.c
Compiler
hello.o
Linker
a.out
libc.o
loader

Objects in a Program

JAVA

```
public class GCD {
    public static int gcd(int a, int b) {
        while (a != b) {
            if (a > b)
                a = a - b;
            else
                b = b - a;
        }
        return a;
    } //gcd()
    public static void main(String[] args) {
        int x, y;
        //Input x + y
        int g = gcd(x, y);
        //output g
    } //main()
} //class GCD
```

C

```
#include <stdio.h>

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

int main() {
    int x, y;
    //input x & y
    int g = gcd(x, y);
    //output g
    return 0;
} //main()
```

Find all names/objects

	Java	C
keywords	public, static class, int, while, if, else return	int while if, else return

names

GCD	gcd
gcd	a
a	b
b	main
main	x
x	y
y	g
g	

Binding: associating a name to the thing it represents.

e.g. gcd : function, code
a, b : int, parameters of gcd()
main : function, code
String : Type
args : array of strip
x : int
y : int
g : int

Binding Time

Time at which a binding is created.

Binding Times

There are many different times when bindings happen

Language Design Time

bindings chosen when a language is designed.

e.g. built-in types (int, float, etc)
statements (if, for, while, etc)
other (main, stdio, etc)
all keywords

Language Implementation Times

Decisions / bindings made by language implementors

e.g. # bits for int, float values
how I/O is done on the specific operating system
memory sizes allocated for programs

Program Writing Time

Programmers choose variable names, functions, types, etc

Compile Time

How language constructs map to machine code
Where + how data is allocated in memory

Link Time

linking library names and other program units

Load Time

: when o/s loads program into memory
binds virtual/physical addresses

Runtime

Binding of values to variables, function call management

Static Binding

when binding can be done before runtime

Dynamic Binding

binding is done during runtime