Dragon book

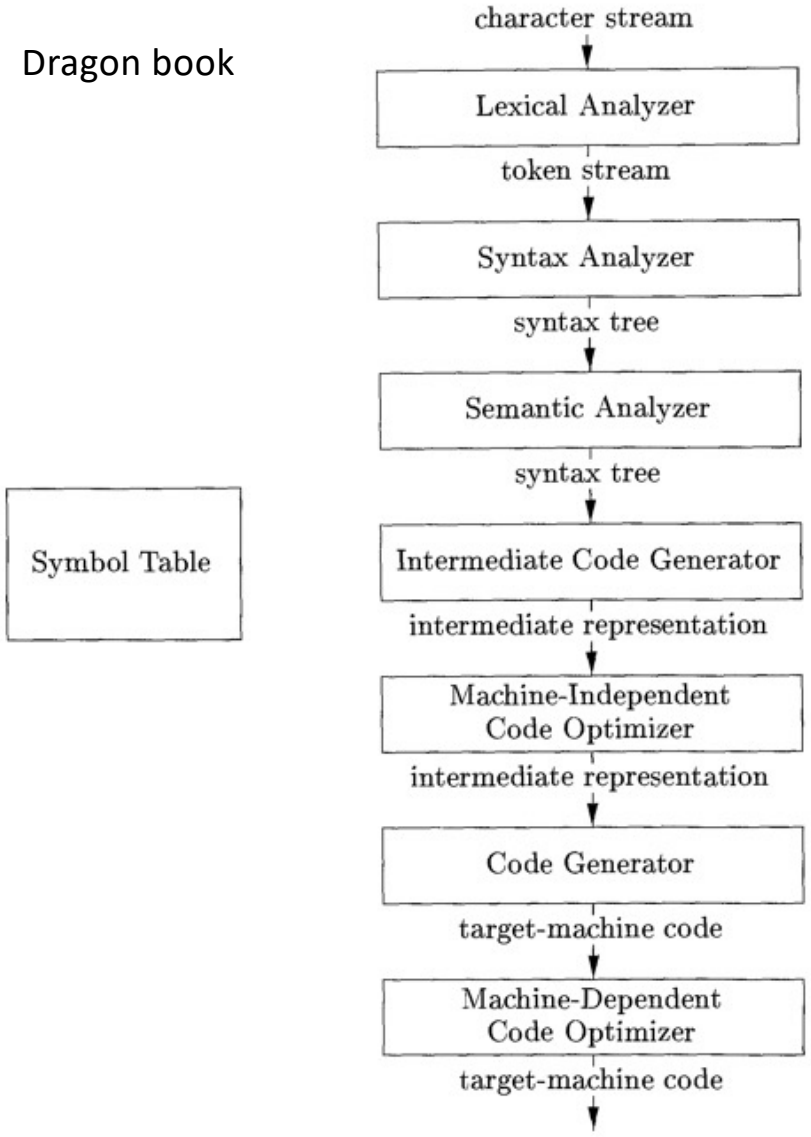
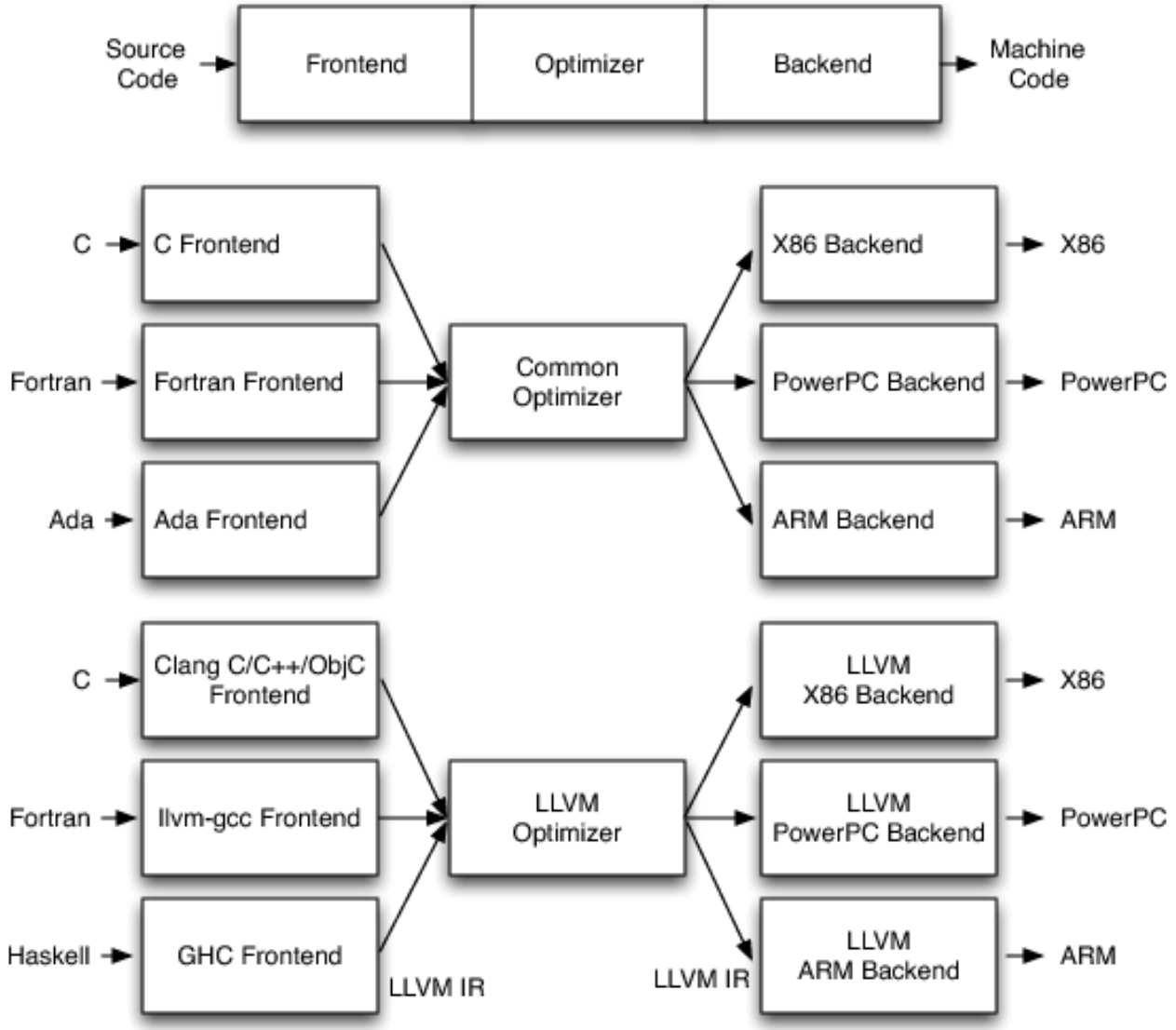


Figure 1.6: Phases of a compiler



LLVM, Lattner

A new language…

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;
contract SimpleStorage {
  uint storedData;
// 256 bit

  function set(uint x) public { storedData = x; }
// anyone can set any value any number of times
// but all  the changes in blockchain for all time!

  function get() public view returns (uint) { return storedData; }
}
```

Add access control? Etc…

wiki

ANTLR LL(k) Parser

```
// Common options, for example, the target language
options { language = "CSharp"; }
// Followed by the parser
class SumParser extends Parser;
options { k = 1; // Parser Lookahead: 1 Token }
// Definition of an expression
statement: INTEGER (PLUS^ INTEGER)*;
// Here is the Lexer class SumLexer extends Lexer;
options { k = 1; // Lexer Lookahead: 1 character }
PLUS: '+’;
DIGIT: ('0'..'9’);
INTEGER: (DIGIT)+;
```

```
TextReader reader;
// (...) Fill TextReader with character
SumLexer lexer = new SumLexer(reader);
SumParser parser = new SumParser(lexer);
parser.statement();
```

Bootstrapping

"Bootstrapping a simple compiler from nothing" (G. Evans02)

- Starts from HEX1: the boot loader ([0-9a-f]{2})*
  - Uses ELF (loading addr of code, start execution addr, etc.), no error checks
- HEX2: one-character labels ([0-9a-f]{2}|\.L|L)*
  - Single pass, so only backward refs, no error checks
- HEX3: 4-char labels and a lot of calls ([0-9a-f]{2}|:....|\.....)*
  - Single pass still, error checks (syntax, undef refs), brk syscall for mem)
  - Now add push const or addr, call func, uncond jump, cond br
- HEX4: any-length labels and implicit calls
  - Stack ops, arith, compare, bitlogic, memaccess, br/call rel/abs, funcs, dyn mem, limited syscalls (exit, in/out) + a few more
- HEX5: structured programming, now a CFL (ifthenelse, loops, break, continue, vars, globals, procs/args, numbers, strings, …)
  but uses reg expr for lex
- HEX6?

ChatGPT

- Seems to be good at regexps. See
  - “Trying and failing to use ChatGPT to write python to parse a PDF file!”
  - https://www.youtube.com/watch?v=NXXuVqZun48 (Dec’22)
- How about CFL?

- More generally: see
  - https://www.newyorker.com/tech/annals-of-technology/chatgpt-is-a-blurry-jpeg-of-the-web

Problems with chatGPT?

- Cannot do simple arithmetic (as of now)
  - Eg. 194050*1202423 gives incorrect 232278999550 (correct: 233330183150)
    - Need multiple competencies? Rather than just thru data?
  - Comes up with bogus relationships
    - Work = (#workers*#hours/worker/day)/(Total Time taken)!
- Will it be able to parse C or C++ correctly like a good compiler can???
  - Give it switch{if stmt}?
  - Find examples of legal C grammar (K&R) that it refuses?

The C Programming Language

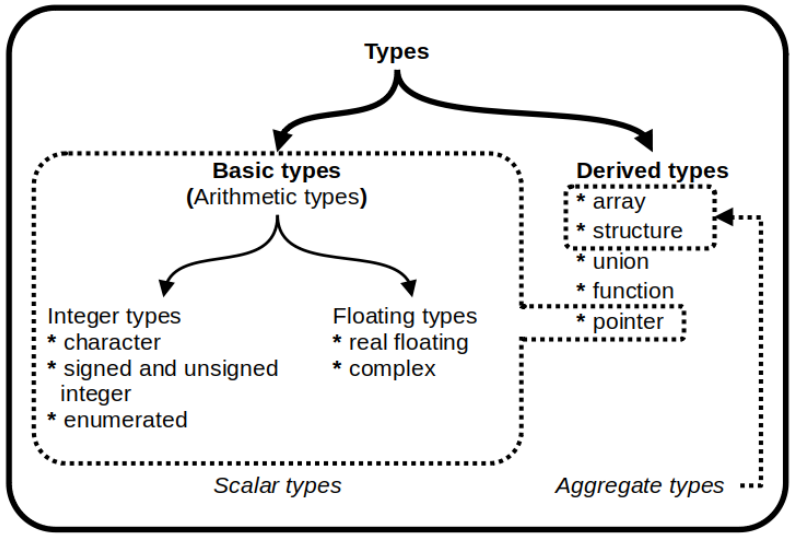
Concepts

Scopes of identifiers
* function
* file
* block
* function prototype

Linkages of identifiers
* external
* internal
* none

Name spaces of identifiers
* label names
* the tags of structures, unions and enumerations
* the members of structures or unions
* ordinary identifiers

Storage durations of objects
* static
* automatic
* allocated



wiki

Block scope: within {} or parameter list of a function def
File scope: from where declared to end of file but not
(within {} or parameter list of a function def}
Function scope: only labels
Function prototype: within fwd declarations only

External linkage thru linker
Internal linkage thru compiler (static): access by anyone in translation unit
Access to outside only thru #include

Tags of structures: name equiv vs structural equiv
Tags of enum: enum week {Mon, Tue, Wed, Thur, Fri, Sat, Sun};
enum week day;
Tags of unions

```
#include<stdio.h>
union tag1 {
    int x;
    char y;
} p;
int main() {
 p.x=47;
 printf("char %c\n", p.y);
}
```

prints "char /"

```
%cat name.c
struct st1 {
    int x;
    char c;
} x;

struct {
    int x;
    char c;
} y;

int main() {
 y=x;
}

%cc name.c
name.c:12:3: error: assigning to 'struct
(unnamed struct at name.c:6:1)' from
incompatible type 'struct st1'
 y=x;
  ^~
1 error generated.
```

```
%cat enum.c
enum week {Mon, Tue, Wed, Thur, Fri, Sat, Sun};
enum week day; //without enum, compiler error
%cc enum.c
Undefined symbols for architecture arm64:
  "_main", referenced from:
     implicit entry/start for main executable
ld: symbol(s) not found for architecture arm64
clang: error: linker command failed with exit
code 1 (use -v to see invocation)
```