Derleyiciler

INF-400

Burak Arslan
iletisim@burakarslan.com

Galatasaray Üniversitesi

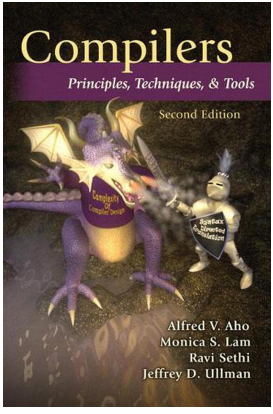
Ders I
2023-10-02

Kitap

"Mor ejderli kitap"

Compilers: Principles, Techniques, Tools

Aho, Lam, Sethi, Ullman



Kaynakça

Kitabın sitesi: suif.stanford.edu/dragonbook

Özellikle CS-143

Ders

- Ders projesi
- Vize(?)
- Final

Ders Projesi

- Kaynak dil: Kiraz(?)
- Hedef dil: WebAssembly 1.0
- Gerçekleme dili: C++

Proje

Kapsam

- Aşağıdakiler üzerinde çalışacağız:
  - Lexer (Sözcük tanıma)
  - Parser (Sözcük ilişkilendirme)
  - Anlam Kazandırma
  - WebAssembly Text Format (WAT) Generator
  - Runtime
- Aşağıdakiler üzerinde çalışmayacağız:
  - Preprocessor
  - Linker
  - Assembler
  - Optimizer

Proje

Kapsam

- Aşağıdakiler üzerinde de çalışmayacağız:
  - Linter
  - Hightlighter
  - Debug symbols / Source maps

Proje

Yöntem

- Ekip çalışması yok. Hatta ekip çalışması yasak. Herkes kendi projesini kendisi yapacak.
- Sürekli üzerine koyarak ilerleyeceğiz. Baştan işi sıkı tutun, ucu kaçmasın

Yüksek seviye programlama dilleri

Nasıl ortaya çıktı?

Programlama Dili

Bilgisayarımsılar / Programımsılar

İlk denemeler:

- Charles Babbage: Analytical Engine (teorik)
- Ada Lovelace: İlk program
- Konrad Zuse: Plankalkül (hayalî)
- John Mauchly: Short Code
- Alick Glennie: Autocode

Programlama Dili

İlk Assembler

1940'lar: Opcode'lar elle yazılıyor

```
55
48 89 e5
48 81 ec 10 02 00 00
c7 45 fc 00 00 00 00
89 7d f8
48 89 75 f0
83 7d f8 01
0f 8e 17 01 00 00
48 8b 45 f0
48 8b 78 08
48 8d 35 ac 13 bf fe
e8 09 00 ba 00
83 f8 00
0f 85 6c 00 00 00
48 8d 35 fb ed bd fe
48 8d 7d d8
48 89 bd 40 fe ff ff
```

Programlama Dili

İlk Assembler

N. Rochester, 1948: İlk assembler

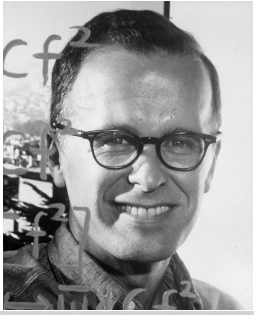
```
55                      <=  pushq %rbp
48 89 e5                <=  movq %rsp, %rbp
48 81 ec 10 02 00 00    <=  subq $0x210, %rsp # imm = 0x210
c7 45 fc 00 00 00 00    <=  movl $0x0, -0x4(%rbp)
89 7d f8                <=  movl %edi, -0x8(%rbp)
48 89 75 f0             <=  movq %rsi, -0x10(%rbp)
83 7d f8 01             <=  cmpl $0x1, -0x8(%rbp)
0f 8e 17 01 00 00       <=  jle 0x555556b37aca # <+314> at main.cpp:46:25
48 8b 45 f0             <=  movq -0x10(%rbp), %rax
48 8b 78 08             <=  movq 0x8(%rax), %rdi
48 8d 35 ac 13 bf fe    <=  leaq -0x140ec54(%rip), %rsi
e8 09 00 ba 00          <=  callq 0x5555576d79d0 # symbol stub for: strcmp
83 f8 00                <=  cmpl $0x0, %eax
0f 85 6c 00 00 00       <=  jne 0x555556b37a3c # <+172> at main.cpp:31:20
48 8d 35 fb ed bd fe    <=  leaq -0x1421205(%rip), %rsi
48 8d 7d d8             <=  leaq -0x28(%rbp), %rdi
48 89 bd 40 fe ff ff    <=  movq %rdi, -0x1c0(%rbp)
```

Programlama Dili

İlk Yorumlayıcı

John Backus, 1953:

- İlk yüksek seviye programlama dili: **Speedcoding**
- Yorumlayıcısıyla birlikte 1953'te yayımlandı
- 10-20x yavaş



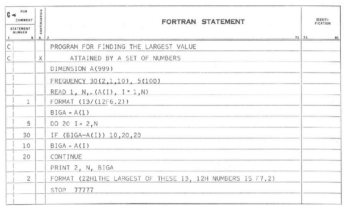
LOC	OP_1	R	A	B	C	OP_2	D
0379	MPY	0	$L(x)$	$L(x)$	0400	STCH	0
0380	MPY	0	$L(y)$	$L(y)$	0401	——	——
0381	ADD	0	0400	0401	0402	——	——
0382	SQRT	0	0402	0000	0403	ECHTR	0379
0383	NOOP	0	0000	0000	0000	STOP	0379

Programlama Dili

İlk Derleyici

John Backus, 1954-1957:

- İlk derlenen dil Fortran I (ve derleyicisi)
- Performans kaybı çok az, geliştirmesi hızlı



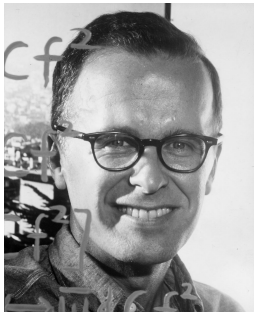
C	STATEMENT NUMBER	CONTINUATION	FORTRAN STATEMENT	IDENTIFICATION
C			PROGRAM FOR FINDING THE LARGEST VALUE	
C		X	ATTAINED BY A SET OF NUMBERS	
			DIMENSION A(999)	
			FREQUENCY 30(2,1,10), 5(100)	
			READ 1, N,.(A(I), I=1,N)	
	1		FORMAT (I3/(12F6.2))	
			BIGA = A(1)	
	5		DO 20 I=2,N	
	30		IF (BIGA-A(I)) 10,20,20	
	10		BIGA = A(I)	
	20		CONTINUE	
			PRINT 2, N, BIGA	
	2		FORMAT (22HTHE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)	
			STOP 77777	

Programlama Dili

Fortran I

John Backus,

- Çığır açtı, tarihe geçti, literatüre geçti. (Backus-Naur Form)
- Boş hayal, çalışması imkansız vb. zırva lara katlandı
- Bugünkü derleyiciler ilk derleyicisine oldukça benziyor



Yüksek seviye programlama dilleri

Nasıl gerçeklenir?

Programlama Dili

Nasıl gerçeklenir?

İki temel strateji:

Derleyici
(compiler)

Yorumlayıcı
(interpreter)

Programlama Dili

Nasıl gerçeklenir?

İki temel strateji:

Derleyici
(compiler)

Yorumlayıcı
(interpreter)

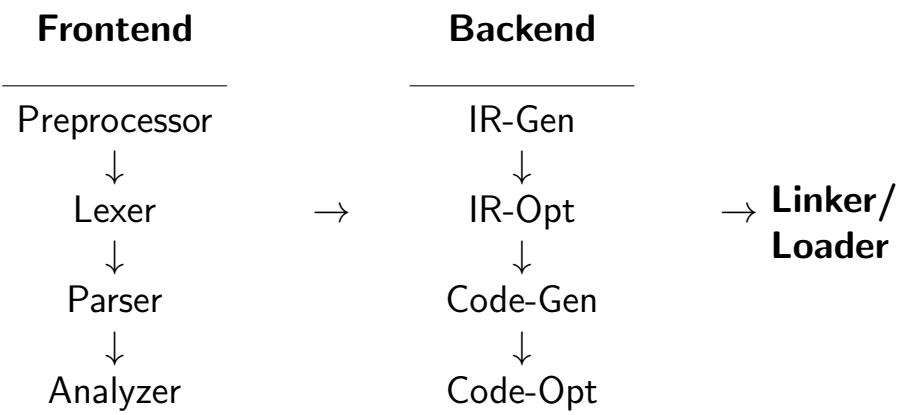
Programlama Dili

Modern Bir Derleyici

Frontend $\rightarrow$ **Backend** $\rightarrow$ **Linker/ Loader**

Programlama Dili

Modern Bir Derleyici



Derleme aşamaları

(Compilation Pipeline)

Genel kültür

Derleme aşamaları

Lexing

Lexing veya **Tokenization**, tipik bir AOT derleyicinin ilk aşamasıdır.

```
import io

func main() -> int32 {
    io.print("Hello world!\n");
    return 0;
}
```

Derleme aşamaları

Lexing

- Yani:

```
00000000  0a 69 6d 70 6f 72 74 20  69 6f 0a 0a 66 75 6e 63  .import io..func
00000010  20 6d 61 69 6e 28 29 20  2d 3e 20 69 6e 74 33 32   main() -> int32
00000020  20 7b 0a 20 20 20 20 69  6f 2e 70 72 69 6e 74 28   {.    io.print(
00000030  22 48 65 6c 6c 6f 20 77  6f 72 6c 64 21 5c 6e 22  "Hello world!\n"
00000040  29 3b 0a 20 20 20 20 72  65 74 75 72 6e 20 30 3b  );.    return 0;
00000050  0a 7d 0a                                          .}.
00000053
```

Derleme aşamaları

Lexing

- Lexer çıktısı bir token array oluyor:

```
KW_IMPORT VAR(io) OP_NEWLINE
KW_FUNC VAR(main) OP_RETURNS VAR(int32) OP_SCOPE_BEGIN OP_NEWLINE
    VAR(io) OP_DOT VAR(print) OP_PAREN_BEGIN STR(Hello World!\n) OP_PAREN_END
                                                                  OP_SCOL OP_NEWLINE
    KW_RETURN INT(0) OP_SCOL OP_NEWLINE
OP_SCOPE_END OP_NEWLINE
```

Derleme aşamaları

Lexing

Lexing adımı;

- Teorik olarak gerekli bir adım değil
- Amaç parser'ı basitleştirmek ve kaynak ihtiyacını azaltmak
- Örnek:
  - Kaynak kodda uzun bir string varsa, Lexer'sız bir derleyicinin Parser'ı her karakterle ilgilenmeli
  - Ancak karakterler değil lexeme'ler (token'lar) üzerinde çalışan bir parser için bir string $==$ bir lexeme

Derleme aşamaları

Lexing

- Lexer yakalar mı?

```
import io

func main() -> int32 {
    io.print("Hello world!\n") . ;
    return 0;
}
```

Derleme aşamaları

Lexing

- Lexer yakalar mı?

```
import io

func main() -> int32 {
    io.print("Hello world!\n")
    return 0;
}
```

Derleme aşamaları

Lexing

- Lexer yakalar mı?

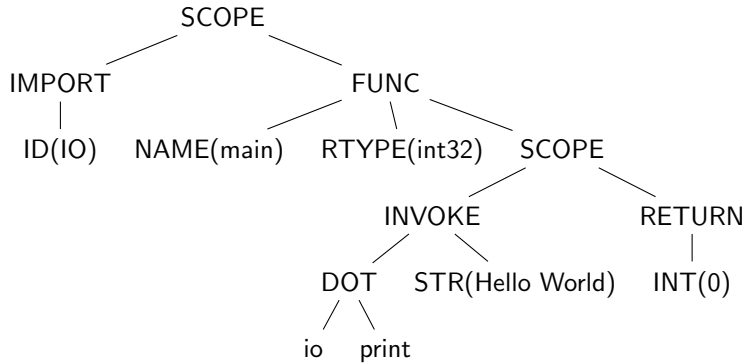
```
import io

func main() -> int32 {
    io.print("Hello world!\n)
    return 0;
}
```

Derleme aşamaları

Parsing

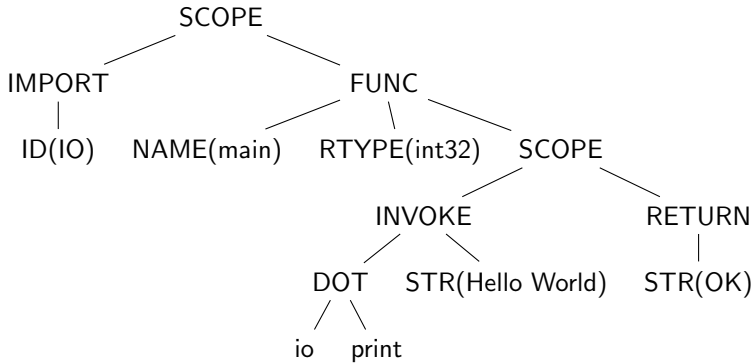
- Parser çıktısı ise bir Abstract Syntax Tree oluyor



Derleme aşamaları

Parsing

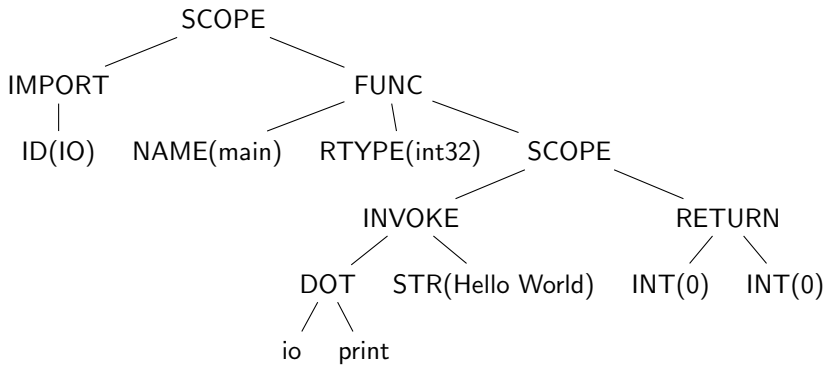
- Parser yakalar mı?



Derleme aşamaları

Parsing

- Parser yakalar mı?



Derleme aşamaları

Anlam Kazandırma

Burada anlam == tutarlı eylemler dizisi.

- **İşin kalbi**: programlama dili gerçekleme çilesine buradaki cambazlıkları yapabilmek için katlanıyoruz
- *Asıl* kod doğrulama işlemleri burada yapılıyor
- Düşük seviye dil burada üretiliyor

Derleme aşamaları

Ara form

Intermediate Representation:

- Derleme aşamaları sadeleşsin
- Platform-bağımsız optimizasyon yapılabilsin

Derleme aşamaları

Hedef Kodun Üretilmesi

- Seviye giderek düşüyor
- Platform-bağımlı optimizasyonlar
- Hedef dile özgü diğer işlemler

Günümüzde Derleyiciler

Bilin bakalım nerelerde var?

Future Work

You need to be comfortable around;

→ **GNU/Linux:** It's our workbench.

→ **C++:** OOP concepts like encapsulation, interitance and polymorphism, as well as practical stuff like IO, smart pointers and parsing command line arguments.

→ **Regular expressions:** Bien évidemment ☺

→ **Lexer/Parser:** We will use lexer/parser generators like FLEX, Bison or ANTLR4

→ **WebAssembly:** You will write a WASM emitter as part of this course.