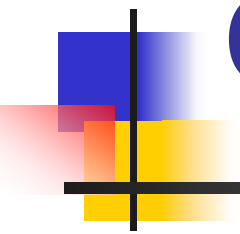
MK. PEMROGRAMAN SISTEM
Semester/SKS : 6/3



COMPILER

Jurusan Sistem Komputer – S1
Universitas Gunadarma

Aspek Kompilasi

- Programmer dengan High Level Languange (HLL) vs. Komputer dengan Machine Language
- Untuk mengeksekusi perlu translasi dari HLL ke Machine Language pada Komputer
- Fungsi Compiler :
  - Melakukan translasi dari HLL program (source program) ke bentuk yang equivalent dengan machine language program
  - Memberikan pesan sebagai diagnostik bila terdapat kesalahan pada source program di HLL yang dibuat programmer

Assembler vs. Compiler

Assembler

- Translator bagi Low Level Languange
- Machine dependent
- Mewariskan arsitektur yang dimilikinya

Compiler

- Translator bagi High Level Languange
- Machine independent
- Membangun feature arsitekturnya

Tipe Data

- HLL melindungi integritas tipe data
- Kompatibilitas operator dan operand di-cek oleh compiler
- Bila perlu compiler dapat melakukan pemaksaan untuk mencapai kompatibilitas operasi
- Contoh :

```
x, y        : real ;
i, j        : integer ;
y := 10 ;
x := y + I ;
```

Diperlukan representasi integer menjadi floating point

Operasi dilakukan pada operand dengan tipe data berbeda, diperlukan konversi tipe data pada salah satu operand

Struktur Data

- HLL mengijinkan programmer mendeklarasikan struktur data : array, stack, table, record, list, dll
- Compiler membangun dan menggunakan storage mapping untuk mengakses actual storage yang dialokasikan untuk beragam elemen dari struktur data
- Contoh :

```
type
    employee = record
        name : array [1..10] of character;
        sex : character;
        id : integer;
    end;
    weekday = (Monday,Tuesday,Wednesday,Thursday, Friday);
var
    info  : array[1..500] of employee;
    today  : weekday;
    i,j  : integer;
begin
    today  := Monday;
    info[I].id  := j;
    if today = Monday then …..
end.
```

Lingkup Aturan (scope rule)

- Dalam HLL, lingkup data merupakan block structure dari languange
- Dua aspek dalam scope rule :
  1. Kesesuaian dalam menetapkan waktu kompilasi
  2. Kesesuaian selama eksekusi
- Contoh :

```
Procedure abc(m,n);
Var
Z,w  : real;
     Procedure P(g : real);
     Var
     Z,s   : real;
     Begin
               Q(z,g);
     End;
     Procedure q(a,b  : real);
     Var
     C,d   : real;
     Begin
               c := w;
               d := a;
     End;
P(w);
End.
```

Struktur Kendali

- Struktur kendali adalah kumpulan fasilitas bahasa untuk mengurutkan kendali pada program yang berguna untuk memecahkan masalah

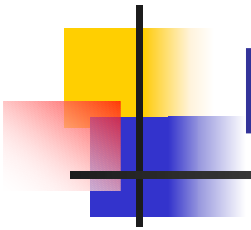
- Contoh struktur kendali :
  - If…. Then…. Else….
  - Do
  - Do while….

Proses Kompilasi

Proses Kompilasi

$$\left\{\begin{array}{c}\text{Analysis of}\\ \text{source text}\end{array}\right\} + \left\{\begin{array}{c}\text{Synthesis of}\\ \text{target text}\end{array}\right\} = \left\{\begin{array}{c}\text{Translation from Source Text}\\ \text{to Target Text}\end{array}\right\}$$

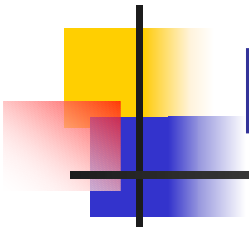
- Fase Analisis melakukan proses sintaks analisis dan semantic analisis
- Fase Sintesis, compiler membuat code dan menentukan pengalamatan



Fase Analisis

Fase Analisis meliputi :

- Lexical analisis
- Syntax analisis
- Semantic analisis



Lexical Analisis

- Lexical analisis menjalankan micro level examination input teks untuk mengidentifikasikan lexical unit (identifier/konstanta) dan menentukan kategori sintaks
- Lexical unit diisolasi untuk mencari set simbol spesial yang disebut delimiter, biasanya berupa operator dan keyword languange. Proses ini disebut **SCANNING**
- Setelah diisolasi, lexical contituent diklasifikasikan menurut kategori sintaks dan dilakukan pemeriksaan validitasnya untuk mendeteksi dan indikasi kesalahan lexical (**lexical error**), seperti invalid variable name, invalid contanta
- **Descriptor** (**token**) dibangun untuk lexical unit, sebagai pembeda diantara kategori sintaks

Lexical Analisis

Contoh : a := b + I

String descriptor :

Id #5	'='	Id#2	'+'	Id#11

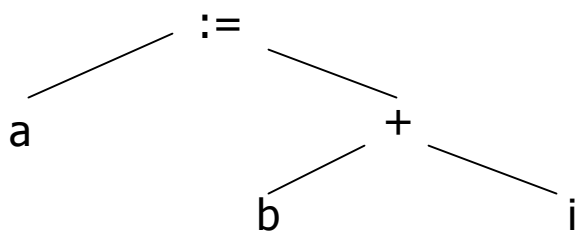
Symbol table :

	Id	Other info
#2	b	
#5	a	
#11	i	

- Hal penting dalam analisis lexical adalah membuat tabel yang sesuai untuk identifier, konstanta, dsb untuk keperluan analisis sintaks berikutnya.

Syntaks Analisis

- Proses syntaks analisis dilakukan terhadap descriptor dari lexical analisis untuk menentukan struktur sintaks
- Prosesnya disebut **PARSING**
- Output dari Parsing adalah representasi dari struktur sintaks suatu statement.
- Representasi sintaks sering dibuat dalam bentuk **Sintax Tree**
- Contoh :

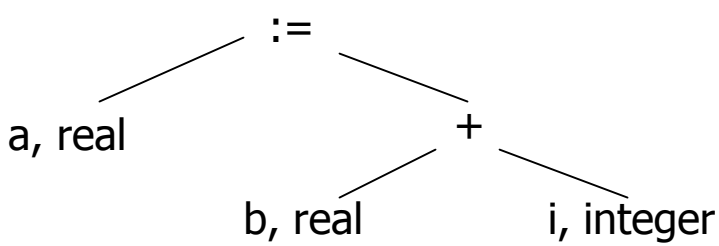


Semantic Analisis

Proses Semantic Analisis :

- Proses declarative statement
- Proses eksekusi dari imperative statement

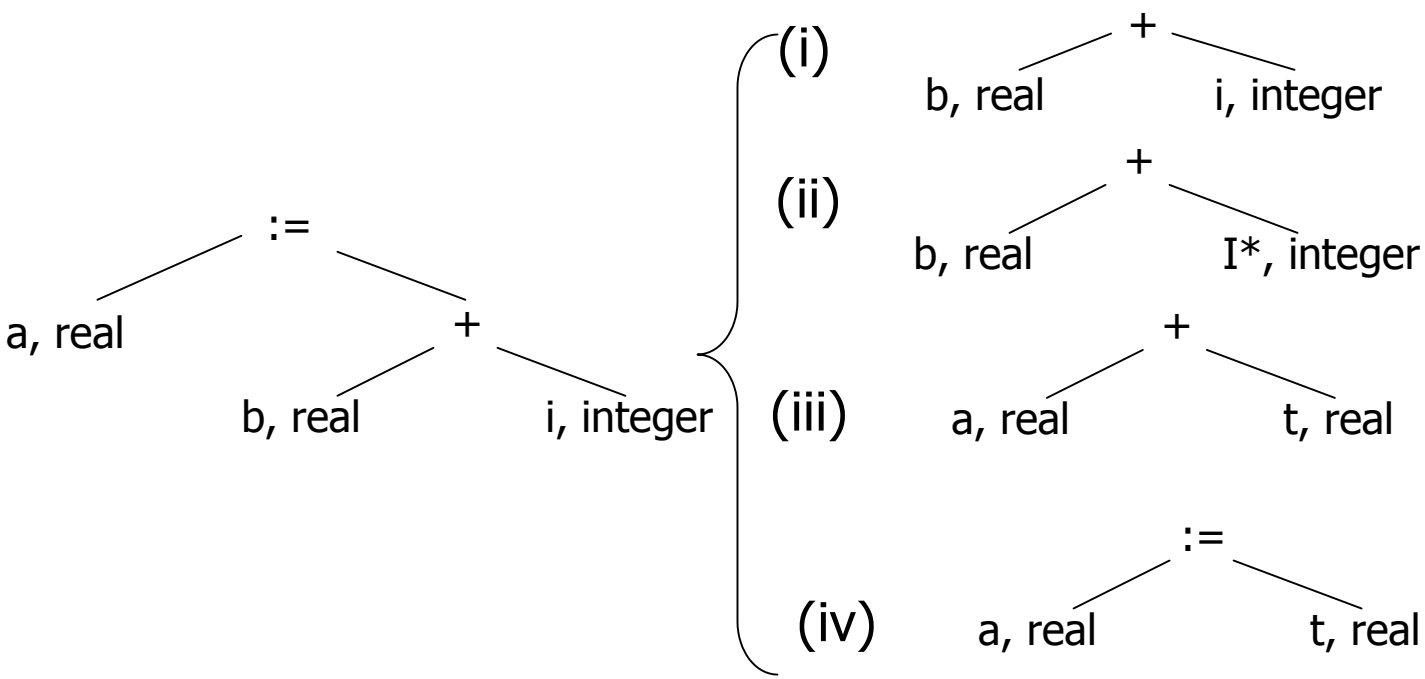
- Proses Declarative statement, memasukkan informasi tambahan dalam lexical table (seperti tipe, panjang dan dimensi variable)



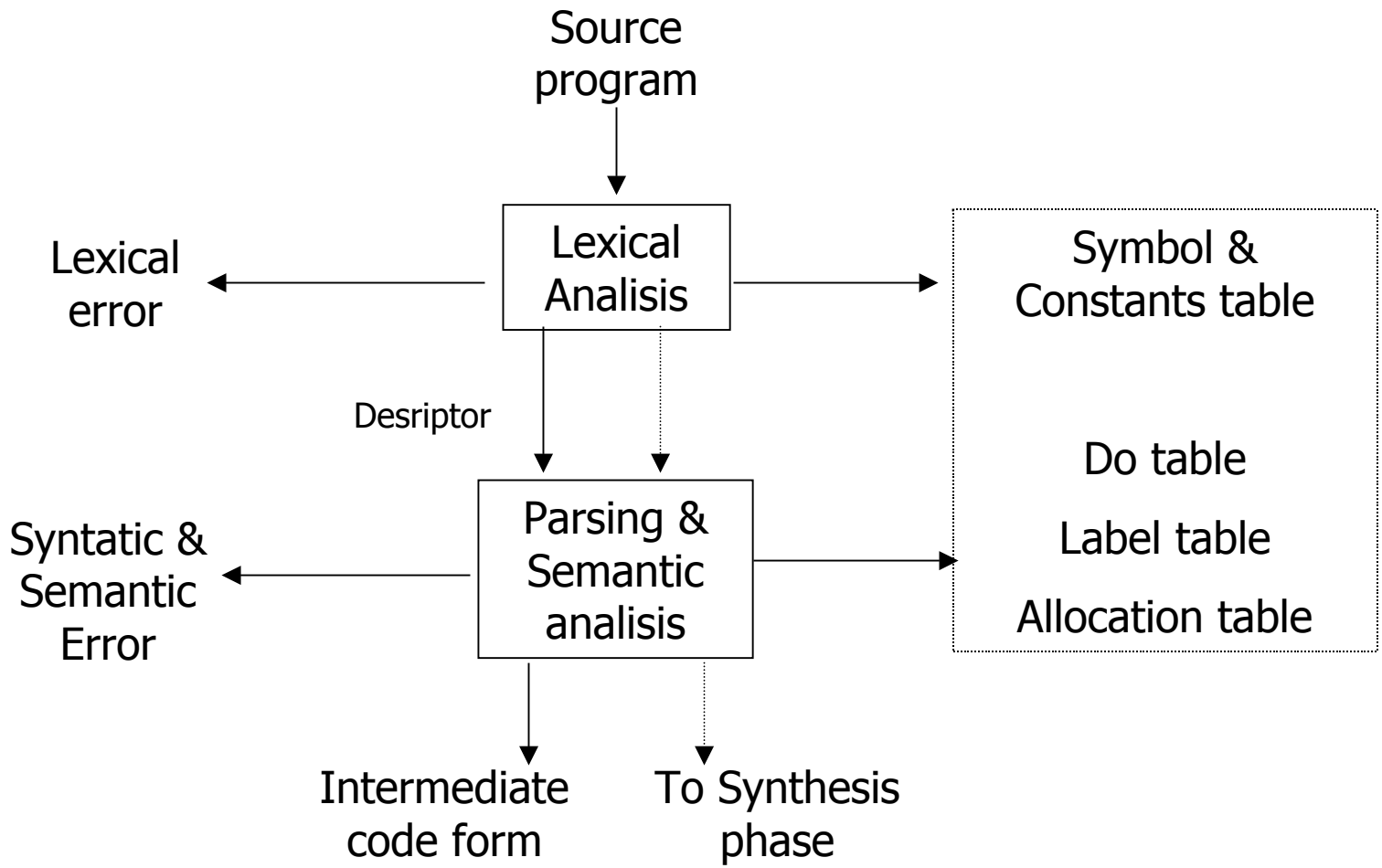
- Menganalisa sintaks tree untuk mengidentifikasi elemental evaluation dan melihat validitasnya

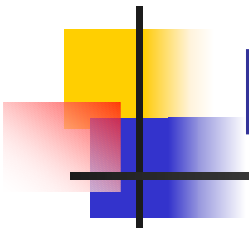
Semantic Analisis

- Menganalisa sintaks tree untuk mengidentifikasi elemental evaluation dan melihat validitasnya



Skema Fase Analisis Compiler





Fase Sintesis

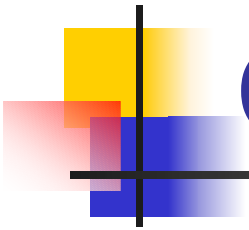
Fase Sintesis meliputi :

- Storage Allocation
- Code Generation

Storage Allocation

- Kebutuhan storage dihitung dari beragam informasi, seperti panjang dan dimensi penyimpanan dalam tabel dan unsur yang dialokasikan dalam machine address
- Alamat yang dialokasikan dimasukkan ke dalam field tabel mengacu pada intermediate code yang kemudian dikonversikan dalam target code

	Id	Type	Address
#2	b	Real	2000
#5	a	Real	2001
#11	i	int	2002



Code Generation

- Code generation ditentukanoleh aspek pragmatis dari compiler
- Pengetahuan tentang arsitektur komputer menjadi dasar untuk men'generate target program instruction

Skema Fase Sintesis Compiler