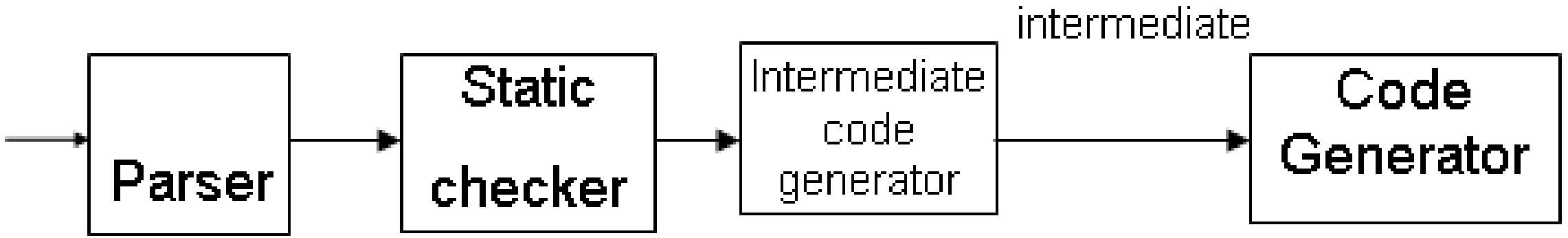
INTERMEDIATE
CODE GENERATION

Dimana ICG


Parser
Static checker
Intermediate code generator
intermediate
Code Generator

INTERMEDIATE CODE GENERATION

- Kegunaan intermediate code :
  - Untuk memperkecil usaha dalam membangun kompilator dari sejumlah bahasa ke sejumlah mesin
  - Proses optimasi lebih mudah
    - (dibandingkan pada program sumber atau kode assembly dan kode mesin)
  - Bisa melihat program internal yang gampang dimengerti.
- 2 macam intermediate code yang biasa digunakan adalah
  - Notasi Postfix dan N-*Tuple*

INTERMEDIATE LANGUAGES

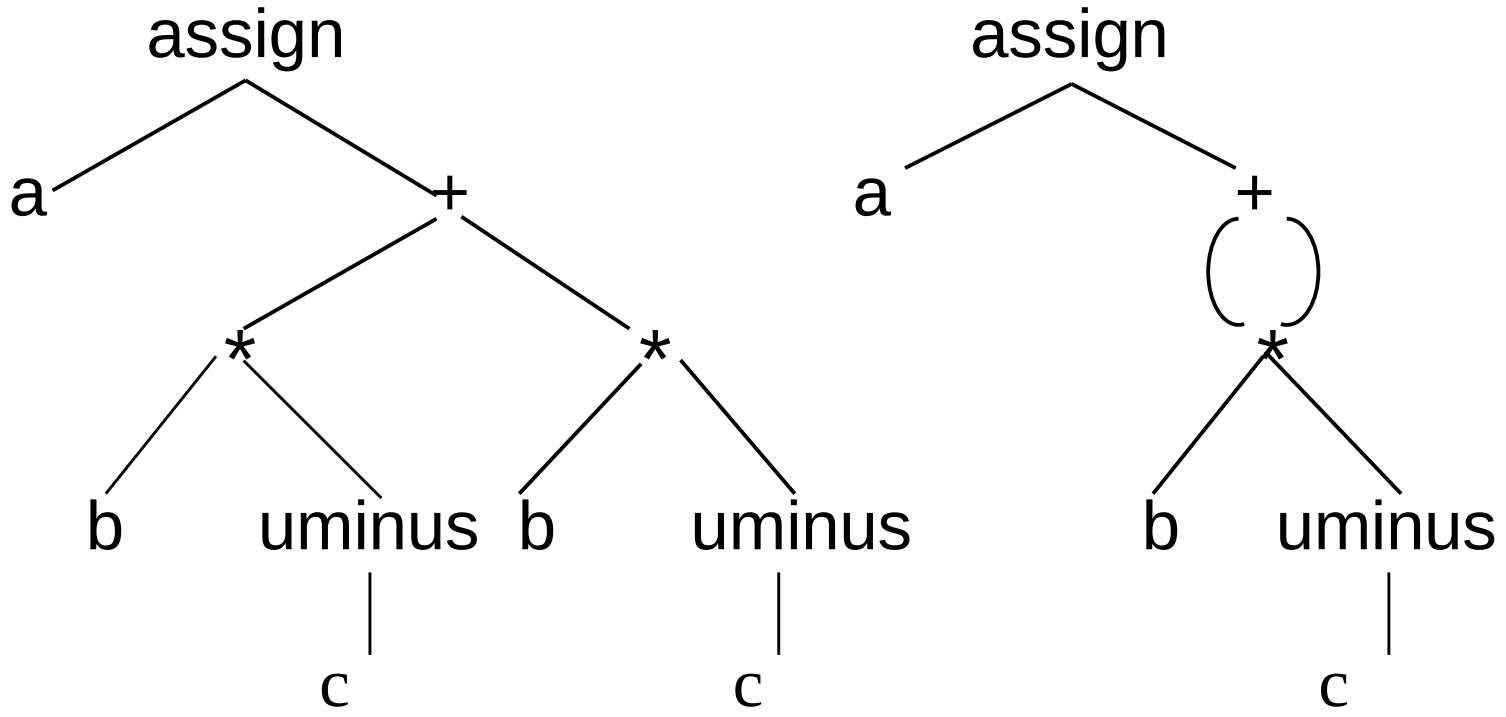
- Syntax Tree
- Postfix Notation
- Three-Address Code

Syntax tree vs DAG

- Syntax tree:
  - Menggambarkan struktur hirarki dari source program.
- DAG (Directed acyclic graph):
  - memberi informasi yang sama tetapi lebih ringkas karena sub-ekspresi yang sama dapat diketahui.

Contoh Syntax tree vs DAG

- a := b * -c + b * -c

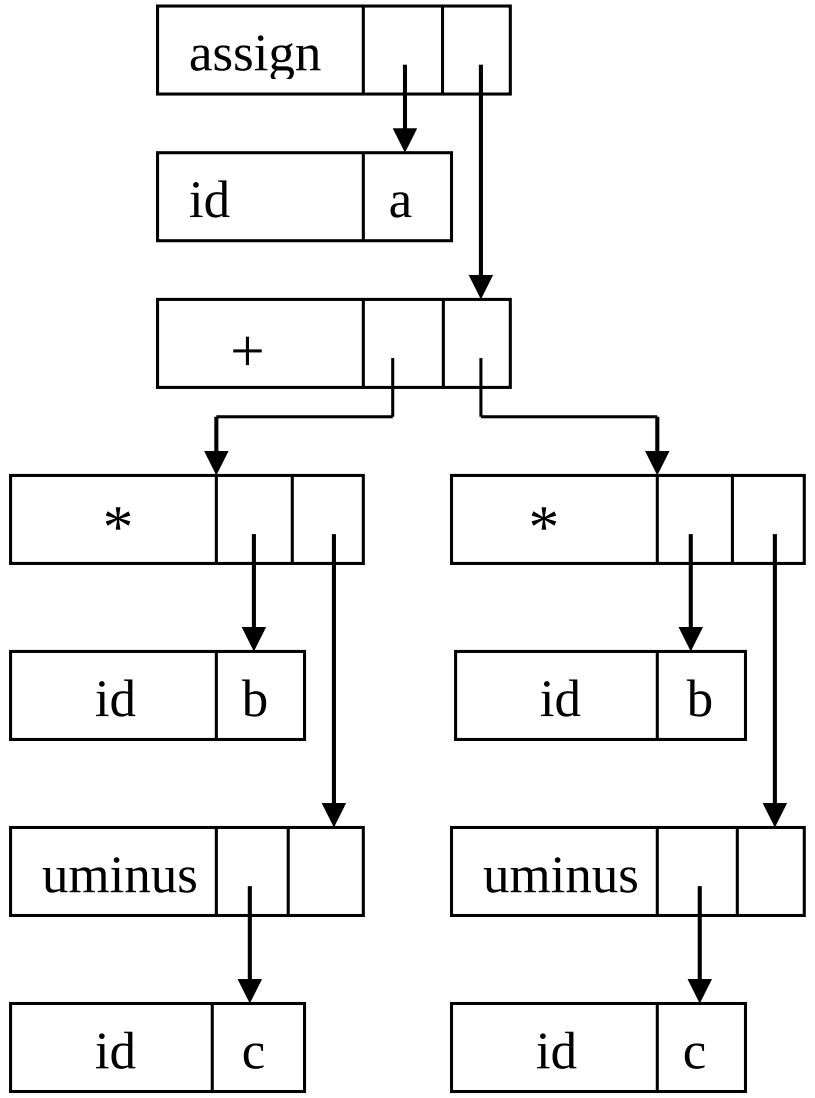


Syntax-directed definition

Production	Semantic Rule
$S \rightarrow$ id := E	S.nptr :=mknode('assign', mkleaf(id,id.place),
$E \rightarrow E_1 + E_2$	E.nptr)
$E \rightarrow E_1 * E_2$	E.nptr := mknode ('+', E_1.nptr, E_2.nptr)
$E \rightarrow -E_1$	E.nptr := mknode ('*', E_1.nptr, E_2.nptr)
$E \rightarrow (E_1)$	E.nptr := mknode ('uminus', E_1.nptr)
$E \rightarrow$ id	E.nptr := E_1.nptr
	E.nptr := mkleaf (id, id.place)

Representasi dari syntax tree

0	Id	b	
1	Id	c	
2	Uminus	1	
3	*	0	2
4	Id	b	
5	Id	c	
6	Uminus	5	
7	*	4	6
8	+	3	7
9	Id	a	
10	Assign	9	8
11	...		



POSTFIX NOTATION

- Representasi linear dari syntax tree yaitu list dari nodes dari tree dimana suatu node muncul segera setelah childrennya
- Pada notasi Postfix operator diletakkan paling akhir
- Sintaks Notasi Postfix:
  - < operan><operan><operator>

Contoh Postfix

- Contoh dari soal:
  - (a+b)*(c+d)
- Notasi Postfix :
  - ab+cd+*

- Contoh dari soal:
  - a := b * -c + b * -c
- Postfix notation:
  - a b c uminus * b c uminus * + assign

INTERMEDIATE CODE GENERATION

Kontrol program yang ada dapat diubah kedalam bentuk notasi postfix, misalnya:

IF <exp> THEN <stmt1> ELSE <stmt2>

Diubah kedalam Notasi Postfix :

<exp> <label1> BZ <stmt1> <label2> BR <stmt2>

Keterangan :

- BZ : branch if zero (zero = salah) {bercabang jika kondisi yang dites salah}
- BR : branch {bercabang tanpa ada kondisi yang dites}

Arti dari notasi Postfix diatas adalah :

“Jika kondisi ekspresi salah, maka instruksi akan meloncat ke Label1 dan menjalankan statement2. Bila kondisi ekspresi benar, maka statement1 akan dijalankan lalu meloncat ke Label2. Label1 dan Label2 sendiri menunjukkan posisi tujuan loncatan, untuk Label1 posisinya tepat sebelum statement2 dan Label2 setelah statement2.”

INTERMEDIATE CODE GENERATION

Contoh lain :

WHILE <exp> DO <stat>

Diubah ke postfix :

<exp><label1>BZ<stat><label2>BR

label1 label2

THREE-ADDRESS CODE

- Merupakan sekuens dari statement dengan bentuk :
  - x := y op z
- dimana :
  - x, y, z = nama, constant atau compiler generated temporaries
  - *op* = sembarang operator seperti fixed point atau floating-point operator aritmetika atau logical operator untuk data boolean

a := b * -c + b * -c

Syntax tree

- t_1 := -c
- t_2 := b * t_1
- t_3 := -c
- t_4 := b * t_3
- t_5 := t_2 + t_4
- a := t_5

DAG

- t_1 := -c
- t_2 := b * t_1
- t_5 := t_2 + t_2
- a := t_5

IMPLEMENTASI THREE-ADDRESS STATEMENT

- Quadruple
  - Struktur record dengan 4 field yaitu : **OP, arg1, arg2** dan **result**
- Triple
  - Struktur record dengan direpresentasi 3 field yaitu : **OP, arg1, arg2**
- Indirect triple
  - Merupakan pengembangan dari triple, dengan menambahkan “tabel index”

Quadruple

$$a := b * -c + b * -c$$

	op	Arg1	arg2	result
(0)	uminus	c		t_1
(1)	*	b	t_1	t_2
(2)	uminus	c		t_3
(3)	*	b	t_3	t_4
(4)	+	t_2	t_4	t_5
(5)	:=	t_5		a

Quadruple

Quadruples Notation

Format instruksi *Quadruples*

<operator><operan><operan><hasil>

- hasil adalah temporary yang bisa ditempatkan pada *memory* atau *register*

contoh instruksi:

A:=D*C+B/E

Bila dibuat dalam Kode Antara :

1. *,D,C,T1
2. /,B,E,T2
3. +,T1,T2,A

Triple

a := b * -c + b * -c

	OP	arg1	arg2
(0)	uminus	c	
(1)	*	b	(0)
(2)	uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	assign	a	(4)

Triple

Triples Notation

Memiliki format

<operator><operand><operand>

contoh, instruksi :

A:=D*C+B/E

Bila dibuat Kode Antara *tripel*:

1. *,D,C
2. /,B,E
3. +,(1),(2)
4. :=,A,(3)

Indirect triple

a := b * -c + b * -c

	Index			OP	arg1	arg2
(0)	0		(0)	uminus	c	
(1)	1		(1)	*	b	(0)
(2)	2		(2)	uminus	c	
(3)	3		(3)	*	b	(2)
(4)	4		(4)	+	(1)	(3)
(5)	5		(5)	assign	a	(4)

Indirect triple

Kekurangan dari notasi *tripel* adalah sulit pada saat melakukan optimasi, maka dikembangkan *Indirect Triples* yang memiliki dua *list* (senarai), yaitu *list* instruksi dan *list* eksekusi. *List* instruksi berisi notasi *tripel*, sedangkan *list* eksekusi mengatur urutan eksekusinya. Misalnya terdapat urutan instruksi :

A := B+C*D/E
F := C*D

List Instruksi :

1. *,C,D
2. /, (1), E
3. +, B, (2)
4. :=, A, (3)
5. :=, F, (1)

List Eksekusi

1. 1
2. 2
3. 3
4. 4
5. 1
6. 5

Terima Kasih