PEMBUATAN KOMPILER

by Tedy Setiadi

Submission date: 02-Nov-2020 05:40AM (UTC+0700)
Submission ID: 1432988879
File name: PEMBUATAN_KOMPILER_DENGAN_METODE.docx (668.21K)
Word count: 1038
Character count: 6957

PEMBUATAN KOMPILER DENGAN METODE *RECURSIVE DESCENT PARSER*

Tedy Setiadi. Basit Adhi Prabowo
Jurusan Teknik Infomatika Unversitas Ahmad Dahlan
ledz68@yahoo.com, yi_huu@yahoo.com

Abstrak

Pembuatan kompiler merupakan pekrjaan yang tidak sederhana karena didslamnya terdapat beberapa proses yang memiki karakteristik dan masslah masing-masing. Dalam penelitian ini telah dibuat prototype kompiler dengan tahapan pembuatan scanner, pembuatan parser dengan menggunakan metode Recursive descent parser serta pembuatan object code dengan bahasa asembly yang disisipkan sawaktu pendefinisian grammar dangan notasi BNF. Prototype kompiler dikembangkan dengan bahasa Visual Basic yang mempunysi fitur dapat membaca source code, kamudian mengkornpilsainya dan menampilkan hasilnya baik kalau sukses maupun ada error serta mampu mengeksekusi program sehingga menghasilan output yang diinginkan.

Kata Kuncl: *source code, recursive descent parser, BNF, object code, Abstrak*

1. PENDAHULUAN

Pengembangan sebuah kompiler merupakan pekerjaan yang tidak sederhana. Sebuah bahasa yang terlalu kompleks akan menyulitkan pembuatan kompiler untuk bahasa tersebut. Kebanyakan pemrogram menginginkan bahasa yang sederhana, tetapi kesederhanaan dapat pula berarti kekurangan di sisi lainnya.

Sintaks adalah susunan kalimat dan grammar[3]. Sintaks dianalisis oleh suatu mesin yang disebut dengan parser. Parser. bertugas menganalisis token yang dihasilkan pada proses scan sesuai dengan grammar. Recursive Descent Parser (RDP) adalah salah satu cara untuk mengaplikasikan bahasa bebas konteks untuk melakukan analisis sintaksis suatu source code. Ciri dari RDP yang menonjol adalah secara rekursif menurunkan semua variabel dari awal sampai bertemu terminal dan tidak pemah mengambil token secara mundur (no back track). Ciri lain dari RCP adalah sangat bergantung pada algoritma scan dalam mengambil token.

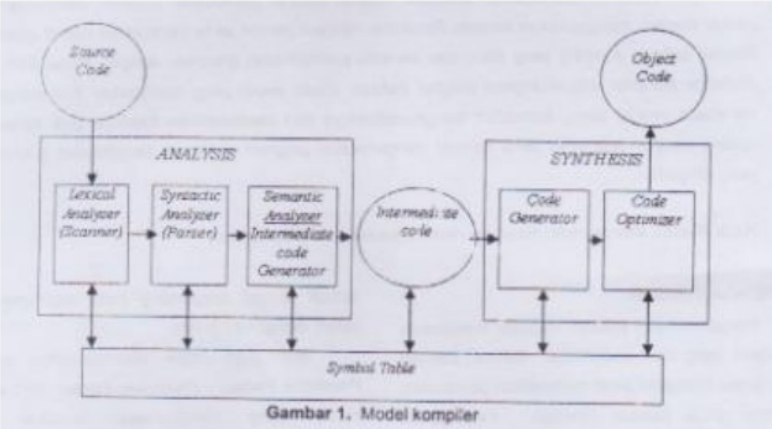
RDP juga dapat dikombinasikan dengan Predictive Parser. Predictive Parser (PP) adalah parser yang menggunakan prediksi untuk mongoptirnalkan kerja dari parser. Parser model ini juga akan mengecilkan terjadinya rekursi kiri atau salah interprestasi. Prinsip dari predictive parser adalah mengelompokkan produksi sesuai dengan pola yang ada sehingga aturan produksi tertentu sudah diprediksikan penurunannya.

2. DASAR TEORI

Sebuah kompiler umumnya memilki dua tugas pokok, yaitu fungsi analis dan fungsi sintesis. Tugas dari ungsi analisis adalah melakukan dekomposisi program sumber menjadi bagian-bagian dasarnya, sedangkan fungsi sintesis memiliki tugas melakukan pembangkitan dan optimasi program objek. Model sebuah kompiler dapat dilihat pada gambar 1.

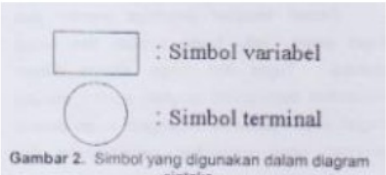
Program sumber merupakan deretan simbol-simbol yag memuat konstruksi bahasa optimasi program objek. Model sebuah kompiler dapat dilihat pada gambar 1.

Program sumber merupakan deretan simbol-simbol yang memuat konstruksi bahasa yang mendasar seperti nama variabel, label, konstanta, keyword, dan operator. *Sacanner* membaca setiap karakter kode sumber dan memisahkan tekas ke dalambeberapa token. Token digunakan oleh parser untuk menganalisis sintesis sesuai dengan grammar yang telah dirancang. Selanutnya, semantic analyzer menentukan maksud dari progam sumber dan menentukan aksi yng harus dilakukan. Analisis semantik bisa menghasilkan intermediate from dari dari program sumber yang berguna untuk memudahkan dalam pembuatan *code generator* dan *code opmizer*. Keluaran dari semantic analyzer diberikan ke code genarator untuk ditranslitasikan ke dalam bahasa assembly atau ke dalam bahasa mesin. Keluarn dari code generaator diberikan ke code optimazer. Proses ini bertujuan untuk menghasilkan progrm yang lebih efisien.



Gambar 1. Model kompiler

Didalam perancangn sebuah parser dikenal dua buah notasi, yaitu BNF dan diagram sintaks. BNF adalh notasi yang dapat digunakan untuk memberikan definisi formal dari bahasa pemrograman. Diagram sintaks memberikan gambaran yang jelas mengneal BNF yang telah dirancang dalambenruk grafis. Diagram sintaks menggunkan simbol pada gambar 2.



Gambar 2. Simbol yang digunakan dalam diagram sintaks

Di dalm proses kompilsi, analisis sintaksis menggunkan parser merupakan bagian front end dari model kompilasi. Karakteristik dari parser top-down hanya membutuhkan beberapa baris untuk mengakomodsi bahas yang telah dirancang

dan sangat cocok dengan BNF,simbol variabel direpresentsikam dengan sebuah prosedur, misalnya

```
<exp> ::= <bool_term> { T_OR
          <bool_term> }
<bool_term> ::= <bool_not_fact> { T_AND
                <bool_not_fact> }
```

Dapat diubah dalan algotiman:

```
procedure exp()
   bool_term()
   while CToken = T_OR do
      scan()
      bool_term()
   endwhile
endprocedure

procedure bool_term()
   bool_not_fact()
   while CToken = T_AND do
      scan()
      bool_not_fact()
   endwhile
endprocedure
```

Penurunan seara rekursif dapt terlihat dari kedua lgoritma. Prosedur boot_term dipanggi oleh prosedu exp. Di dalam prosedur boot_term juga pemanggilan prosedur, yaitu prosedu boo_not_fact. Pemanggilan prosedur tersebut terjadi beruang-ulang (rekursif) dan terjadi penurunan (*descent*). Rekursif (penurunan) terjadi samapai menemui simbol terminl (*token*). Masalah utama dalam penggunaan rekursi adalah rekursi yang tidak berhenti. Oleh karena itu, diperlukan kehati-hatian dalam pemakaian rekurs. *Predictive parser* diperlukan untuk menekan atau menghilangkan kemungkinn terjadinya hal tersebut.

3. PEMBAHASAN

Tahap-tahap penelitian yang dilakukan dalam penelitian ini adalah:

1. Pendefinisian Kebutuhan

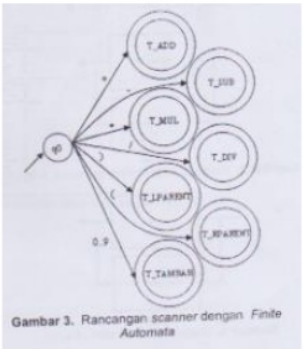
Pendefinisia kebutuhn meliputi penentuan perngkat lunak, penentuan perngkat keras yang seuai dngan perangkat lunak dan aplikasi yang hendk dibuat serta data-data yang diperlukan untuk membuat aplikasi. Aplikasi ini hanya membutuhkan 3 elemn dari elemen-eleman kompiler yang ada, yaitu *scanner,parer,* dan *code generator*. Hal ini bisa dilakukan karena bahasa yang dirancang tidak memerlukan deklarasi variabl (memerlukan tabel informasi) dan optimasi kode (memerlukan kode antara), sedangkan analisis semantik sudah aa secara implisit di dalam *parser*.

2. Perancangan

Perancangan merupakan tahap awal dari suatu alikasi program untuk meghasikan sistem yang baik. Perancangan dibuat sebagai landasan implementasi untuk mempermudah pembuatan aplikasi. Perancangan meliputi peranvangan *scanner* (FA), perancangan *parser* (BNF dan diagram sintaks), dan desain aplikasi.

a. Perancangan *Scanner* menggunakan Finite Automata

Dalam memerksa sintaks dari kode sumer, *parser* membuthukan bantuan *scanner* untuk memberikan token dari sintaks. Berdasarkan BNF dan diagram sintaks yang telah ibuat, maka dapat dirancang sebuah *scanner* dengan rancangan seperti gambar 3.



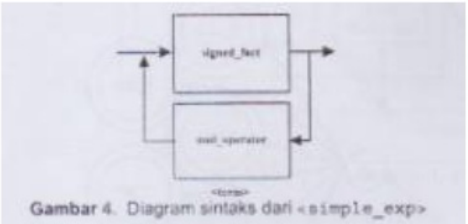
Gambar 3. Rancangan scanner dengan *Finite Automata*

b. Perancangan *Parser* menggunkan BNF dan Diagram Sitaks

Kode sumber pada kompiler memerlukan BNF dan diagram sinaks agar pembuat progam mudah dalam membuat program. Adapun beberapa program rancangan BNF dari kompiler yang dibuat adalah sebagai berikut:

```
<simple_exp>    ::= <term> { <add_operator>
                    <term> }
<add_operator>  ::= T_ADD | T_SUB
<term>          ::= <signed_fact> {
                    <mul_operator>
                    <signed_fact> }
<signed fact>   ::= <add_operator> <fact> |
                    <fact>
<mul_operator>  ::= T_MUL | T_DIV
<fact>          ::= T_LPARENT <exp>
                    T_RPARENT | T_NUMERIC
<simple_exp>    ::= <term> { <add_operator>
                    <term> }
<add_operator>  ::= T_ADD | T_SUB
<term>          ::= <signed_fact> {
                    <mul_operator>
                    <signed_fact> }
<signed fact>   ::= <add_operator> <fact> |
                    <fact>
<mul_operator>  ::= T_MUL | T_DIV
<fact>          ::= T_LPARENT <exp>
                    T_RPARENT | T_NUMERIC
```

Sedangkan beberapa rancangan berupa diagram sintaks dari kompiler yang dibuat dapat dilihat pada gambar 4 dan 5.



Gambar 4. Diagram sintaks dari <simple_exp>

Gambar 5. Diagram sintaks dari <term>

c. Pembuatan *object code*

Pembuatan *object code* dalam bentuk bahasa *assembly* disisipkan ke dalam implementasi notasi BNF untuk mempermudah pembuatan program. *Object code* yang disisipkan ditandai dengan pemanggilan sub "writeAssemblyCode". Berikut ini adalah beberapa contok implementasi dari BNF yang telah dibuat dan pembuatan *object code* Option Explicit

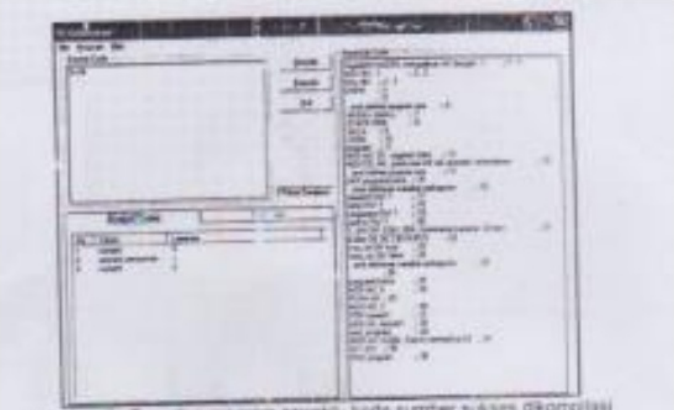
```
Sub simple_exp()
'<simple_exp> ::= <term> {
<add_operator> <term> }
 term
 While add_operator
  writeAssemblyCode "PUSH AX"
  Select Case CToken
   Case T_ADD
    Scan
    term
    writeAssemblyCode "POP tempAX" &
gantiBaris & "ADD AX, tempAX"
   Case T_SUB
    Scan
    term
    writeAssemblyCode "POP tempAX" &
gantiBaris & "haliAX2tempAX" &
gantiBaris & "SUB AX, tempAX"
  End Select
 Wend
End Sub
```

d. Tampilan Program

Pada progrm ini terdapat 3 bagian yag berisi teks, yaitu *source code* (tempat untuk mengetikkan baris program), tabel informasi dan *assembly code*. Program ini akan meghasilkan sebuah file aplikasi (*.exe) jika baris program sukses dikompilasi seperti pada gambar 6 dan program ini akan menghasilkan pesan kesalahan jika baris program tidak sukses dikompilasi seperti pada gambar 7.

4. KESIMPULAN

1. Pembuatan kompiler merupakan masalah yang kompleks karena di dalam kompiler sendiri terdapat beberapa proses yang memiliki karakteristik dan masalah masing-masing.
2. *Recursive descent parser* mudah untuk diimplementasikan tetapi memerlukan kehati-hatian dalam perencanaaanya.



Gambar 6. Tampilan program sewaktu kode sumber sukses dikompilasi

Gambar 7. Tampilan program sewaktu kode sumber tidak sukses dikompilasi

5. DAFTAR PUSTAKA

[1] Crenshaw, Jack W., "Lets Build a Compiler", http://www.etekchalmers.se/-e8johan/compiler/, 1988

[2] V. Aho, Alfred, Ravi Sethi, and Jefrey D. Ullman, "Compiler Principles, Techniques, and Tools", Addison-Wsley, 1986.

[3] Utdirartatmo, Firrar, "Teknik Kompilasi", Graha Ilmu, Yogyakarta, 2005.

[4] Wiryana, I Made, "Compiler Construction". http://wiryana.pandu.org/

PEMBUATAN KOMPILER

ORIGINALITY REPORT

10%	10%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	edoc.site Internet Source	4%
2	www.scribd.com Internet Source	4%
3	digilib.uinsby.ac.id Internet Source	1%
4	mafiadoc.com Internet Source	1%
5	D. Lee, M. Yannakakis. "Principles and methods of testing finite state machines-a survey", Proceedings of the IEEE, 1996 Publication	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		