Bab 15

Compiler Tools Analysis

Sub-CPMK yang Dicakup dalam Bab Ini:

- **Sub-CPMK 6.1:** Menganalisis dan menggunakan compiler tools modern

15.1 Pengenalan Compiler Tools

15.1.1 Kategori Compiler Tools

Compiler Tools adalah software yang membantu pembuatan compiler. Ekosistem pengembangan kompilator modern melibatkan berbagai alat dan standar industri yang canggih untuk memastikan kualitas dan efisiensi kode yang dihasilkan [1].

- **Lexer Generators**: Flex, re2c
- **Parser Generators**: Bison, ANTLR
- **Optimization Frameworks**: LLVM, GCC
- **Build Systems**: Make, CMake
- **Debugging Tools**: GDB, Valgrind

15.1.2 Ekosistem Compiler Development

15.2 Lexer Generators

15.2.1 Flex (Fast Lexical Analyzer)

Flex adalah lexer generator yang populer:

Tool	Tipe	Platform
Flex	Lexer Generator	Cross-platform
Bison	Parser Generator	Cross-platform
ANTLR	Parser Generator	Java-based
LLVM	Compiler Framework	Cross-platform
GCC	Compiler Suite	Cross-platform

Tabel 15.1: Popular Compiler Tools

```
/* Example Flex specification */
%{
#include "y.tab.h"
%}

digit        [0-9]
letter       [a-zA-Z]
identifier   {letter}({letter}|{digit})*

%%

{digit}+         { yylval.ival = atoi(yytext); return NUMBER; }
{identifier}     { yylval.sval = strdup(yytext); return IDENTIFIER; }
"+"              { return PLUS; }
"*"              { return TIMES; }
[ \t\n]          { /* skip whitespace */ }
.                { fprintf(stderr, "Unknown character: %s\n", yytext); }

%%

int yywrap(void) {
    return 1;
}
```

15.2.2 re2c

re2c adalah alternatif yang lebih sederhana:

```
/* re2c specification */
/*!re2c
re2c:define:YYCTYPE        unsigned char
re2c:define:YYCURSOR       cursor
re2c:define:YYLIMIT         limit
re2c:define:YYMARKER        marker
re2c:define:YYFILL(n)      { memset(cursor, 0, n); cursor += n; }

NUMBER = [0-9]+;
IDENTIFIER = [a-zA-Z][a-zA-Z0-9]*;
*/

void scan(char *cursor, char *limit) {
    char *marker = cursor;

```

```
    /*!re2c
    NUMBER { return NUMBER; }
    IDENTIFIER { return IDENTIFIER; }
    "+" { return PLUS; }
    "*" { return TIMES; }
    [ \t\r\n]+ { continue; }
    . { return UNKNOWN; }
    */
}
```

15.3 Parser Generators

15.3.1 Bison (GNU Yacc)

Bison adalah LALR(1) parser generator:

```
/* Bison specification */
%{
#include <stdio.h>
#include <stdlib.h>
int yylex(void);
void yyerror(const char *s);
%}

%token NUMBER IDENTIFIER PLUS TIMES

%left PLUS
%left TIMES

%%
input:
    /* empty */
  | input line
  ;

line:
    expr '\n'          { printf("= %d\n", $1); }
  | '\n'               { /* empty line */ }
  ;

expr:
    expr PLUS expr     { $$ = $1 + $3; }
  | expr TIMES expr    { $$ = $1 * $3; }
  | NUMBER             { $$ = $1; }
  | IDENTIFIER         { $$ = symbol_lookup($1); }
  ;
%%

void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}
```

15.3.2 ANTLR

ANTLR adalah parser generator modern:

```
// ANTLR grammar for arithmetic expressions
grammar Expr;

prog: stat+ ;

stat: expr NEWLINE
    | ID '=' expr NEWLINE { symbols.put($ID.text, $expr.v); }
    ;

expr: expr op=('*'|'/') expr      { $v = new BinOp($op.text, $expr.v,
↪ $expr2.v); }
    | expr op=('+'|'-') expr      { $v = new BinOp($op.text, $expr.v,
↪ $expr2.v); }
    | INT                         { $v = new Int($INT.text); }
    | ID                          { $v = symbols.get($ID.text); }
    | '(' expr ')'                { $v = $expr.v; }
    ;
```

15.4 Compiler Frameworks

15.4.1 LLVM (Low Level Virtual Machine)

LLVM adalah compiler framework modern:

```
// LLVM IR example
define i32 @add(i32 %a, i32 %b) {
entry:
  %sum = add i32 %a, %b
  ret i32 %sum
}

// Using LLVM C++ API
#include "llvm/IR/IRBuilder.h"

llvm::Value* createAdd(llvm::IRBuilder<> &builder,
                       llvm::Value *a, llvm::Value *b) {
    return builder.CreateAdd(a, b, "sum");
}
```

15.4.2 GCC (GNU Compiler Collection)

GCC adalah compiler suite lengkap:

- Frontends: C, C++, Objective-C, Fortran, Ada, Go
- Backends: x86, ARM, MIPS, PowerPC, RISC-V
- Middle-end: GIMPLE, RTL, Tree SSA

- Optimizations: -O0, -O1, -O2, -O3, -Os

15.5 Build Systems

15.5.1 Make

Make adalah build system tradisional:

```
# Makefile for simple compiler
CC = gcc
CFLAGS = -Wall -g

compiler: lexer.o parser.o main.o
  $(CC) $(CFLAGS) -o $@ $^

lexer.o: lexer.l
  flex lexer.l
  $(CC) $(CFLAGS) -c lex.yy.c -o $@

parser.o: parser.y
  bison -d parser.y
  $(CC) $(CFLAGS) -c parser.tab.c -o $@

clean:
  rm -f *.o compiler lex.yy.c parser.tab.c parser.tab.h
```

15.5.2 CMake

CMake adalah build system modern:

```
# CMakeLists.txt
cmake_minimum_required(VERSION 3.10)
project(MyCompiler)

set(CMAKE_C_STANDARD 11)

# Find required packages
find_package(FLEX REQUIRED)
find_package(BISON REQUIRED)

# Generate lexer and parser
flex_target(lexer lexer.l)
bison_target(parser parser.y parser.tab.h parser.tab.c)

# Create executable
add_executable(mycompiler
    main.c
    ${FLEX_lexer_OUTPUTS}
    ${BISON_parser_OUTPUTS}
)
```

15.6 Debugging Tools

15.6.1 GDB (GNU Debugger)

GDB untuk debugging compiler:

```
# Debugging compiler with GDB
gdb ./mycompiler
(gdb) break main
(gdb) run input.c
(gdb) step
(gdb) print token
(gdb) backtrace
(gdb) quit
```

15.6.2 Valgrind

Valgrind untuk memory debugging:

```
# Check for memory leaks
valgrind --leak-check=full ./mycompiler input.c

# Check for memory errors
valgrind --tool=memcheck ./mycompiler input.c

# Profile memory usage
valgrind --tool=massif ./mycompiler input.c
```

15.7 Performance Analysis

15.7.1 Profiling Tools

Tool	Fungsi
gprof	Function profiling
perf	System-wide profiling
valgrind/callgrind	Call graph profiling
time	Execution time measurement

Tabel 15.2: Profiling Tools

15.7.2 Benchmarking

```
// Simple benchmark framework
#include <time.h>
#include <stdio.h>

void benchmark_compiler(char *input_file) {
    clock_t start = clock();
```

```
    // Run compiler
    compile_file(input_file);

    clock_t end = clock();
    double elapsed = ((double)(end - start)) / CLOCKS_PER_SEC;

    printf("Compilation time: %.3f seconds\n", elapsed);
}
```

Aktivitas Pembelajaran

1. **Flex/Bison**: Buat simple calculator menggunakan Flex dan Bison.
2. **ANTLR**: Implementasikan parser untuk bahasa ekspresi dengan ANTLR.
3. **LLVM**: Buat LLVM pass untuk optimasi sederhana.
4. **Build Systems**: Konversi Makefile ke CMake.
5. **Debugging**: Debug compiler crash dengan GDB.

Latihan dan Refleksi

1. Bandingkan Flex dan re2c untuk lexer generation!
2. Implementasikan parser untuk bahasa dengan control structures menggunakan Bison!
3. Analisis LLVM IR untuk program sederhana!
4. Konversi build system dari Make ke CMake untuk project compiler!
5. Debug memory leak dalam parser generator!
6. **Refleksi**: Bagaimana compiler tools mempengaruhi produktivitas pengembangan compiler?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 6.1

A. Pilihan Ganda

1. Flex adalah:
   (a) Parser generator
   (b) Lexer generator
   (c) Build system
   (d) Debugger
2. Bison menggunakan parsing algorithm:
   (a) LL(1)
   (b) LR(1)
   (c) LALR(1)
   (d) Recursive descent
3. LLVM adalah:
   (a) Compiler framework
   (b) Build system
   (c) Debugger
   (d) Lexer generator

B. Essay

1. Jelaskan ekosistem compiler tools dan peran masing-masing tool!
2. Implementasikan mini compiler menggunakan Flex, Bison, dan build system modern!

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menganalisis dan menggunakan compiler tools modern

☐ Saya dapat menggunakan lexer generators (Flex, re2c)

☐ Saya dapat menggunakan parser generators (Bison, ANTLR)

☐ Saya memahami compiler frameworks (LLVM, GCC)

☐ Saya dapat menggunakan build systems (Make, CMake)

□ Saya dapat menggunakan debugging tools untuk compiler development

Rangkuman

Bab ini membahas compiler tools analysis, termasuk lexer generators, parser generators, compiler frameworks, build systems, dan debugging tools. Mahasiswa belajar menggunakan tools modern untuk pengembangan compiler yang efisien.

Poin Kunci:

- Compiler tools menyederhanakan pengembangan compiler
- Lexer generators otomatisasi token recognition
- Parser generators menghasilkan efficient parsers
- Compiler frameworks menyediakan infrastructure lengkap
- Build systems mengelola compilation dependencies
- Debugging tools membantu identifikasi dan perbaikan bugs

Kata Kunci: ***Compiler Tools***, ***Flex***, ***Bison***, ***ANTLR***, ***LLVM***, ***GCC***, ***Make***, ***CMake***, ***GDB***, ***Valgrind***

Daftar Pustaka

[1] University of Oxford. *Compilers Course*. Course materials, Michaelmas Term 2024, taught by Matty Hoban. 2024. URL: https://www.cs.ox.ac.uk/teaching/courses/2024-2025/com/.