

Bab 18

Kumpulan Quiz Pilihan Ganda

18.1 Soal

18.1.1 Quiz Bab 1: Pengenalan Kompilator dan Fase-Fase Kompilasi

1. Apa perbedaan utama antara kompilator dan interpreter?
 - a.) Kompilator mengeksekusi baris per baris; interpreter menerjemahkan seluruh program sekaligus
 - b.) Kompilator menerjemahkan seluruh program sekaligus sebelum dieksekusi; interpreter menerjemahkan dan mengeksekusi baris demi baris
 - c.) Kompilator menghasilkan bytecode; interpreter menghasilkan machine code
 - d.) Tidak ada perbedaan yang signifikan
2. Fase kompilasi manakah yang bertanggung jawab menghapus whitespace dan komentar?
 - a.) Syntax analysis
 - b.) Semantic analysis
 - c.) Lexical analysis
 - d.) Code generation
3. Manakah yang termasuk bahasa yang menggunakan pendekatan interpreter?
 - a.) C dan C++
 - b.) Python dan JavaScript
 - c.) Rust dan Go
 - d.) Pascal dan Fortran
4. Fase manakah yang mengonversi ekspresi $a + b * c$ menjadi three-address code?

- a.) Lexical analysis
 - b.) Syntax analysis
 - c.) Semantic analysis
 - d.) Intermediate code generation
5. Program yang menerjemahkan assembly code ke machine code disebut...
- a.) Kompilator
 - b.) Interpreter
 - c.) Assembler
 - d.) Linker

18.1.2 Quiz Bab 2: Regular Expression dan Finite Automata

1. Algoritma Thompson digunakan untuk...
- a.) Mengkonversi NFA ke DFA
 - b.) Mengkonversi regular expression ke NFA
 - c.) Minimisasi DFA
 - d.) Mensimulasikan DFA
2. Mengapa DFA lebih efisien untuk simulasi dibanding NFA?
- a.) DFA lebih mudah dikonstruksi
 - b.) DFA tidak memiliki ϵ -transition; setiap state tepat satu transition per input symbol
 - c.) NFA tidak dapat mengenali regular language
 - d.) DFA menggunakan lebih sedikit memori
3. Operasi Kleene star (*) dalam regular expression menyatakan...
- a.) Satu atau lebih pengulangan
 - b.) Nol atau lebih pengulangan
 - c.) Tepat satu kemunculan
 - d.) Pilihan antara dua pattern
4. Subset construction digunakan untuk...
- a.) Mengkonversi regular expression ke NFA

- b.) Mengkonversi NFA ke DFA
 - c.) Minimisasi DFA
 - d.) Membangun lexical analyzer
5. Pattern `[0-9]+` dalam regular expression cocok untuk...
- a.) Identifier
 - b.) Integer literal
 - c.) String literal
 - d.) Keyword

18.1.3 Quiz Bab 3: Implementasi Lexer (Hand-Written)

1. Keuntungan hand-written lexer dibanding lexer generator adalah...
 - a.) Lebih cepat dalam development
 - b.) Kontrol penuh dan pemahaman mendalam terhadap proses tokenization
 - c.) Kode yang dihasilkan lebih pendek
 - d.) Tidak perlu menangani edge case
2. Proses utama yang dilakukan lexer adalah...
 - a.) Parsing struktur program
 - b.) Tokenization atau pengenalan token dari source code
 - c.) Type checking
 - d.) Code generation
3. Komentar multi-line dalam C menggunakan...
 - a.) `//`
 - b.) `#`
 - c.) `/* */`
 - d.) `-`
4. Escape sequence `\n` menyatakan...
 - a.) Karakter backslash
 - b.) Karakter newline
 - c.) Karakter tab

- d.) Karakter null
- 5. Output dari lexer adalah. . .
 - a.) Parse tree
 - b.) Stream of tokens
 - c.) Symbol table
 - d.) Machine code

18.1.4 Quiz Bab 4: Lexer Generator (Flex/re2c)

1. File specification Flex memiliki ekstensi. . .
 - a.) `.y`
 - b.) `.l`
 - c.) `.c`
 - d.) `.flex`
2. Bagian Rules dalam file Flex berisi. . .
 - a.) Definisi makro dan C code
 - b.) Pasangan pattern–action (regular expression dan kode C)
 - c.) Fungsi `main()` dan `yywrap()`
 - d.) Deklarasi token
3. Ketika beberapa pattern cocok dengan input, Flex memilih. . .
 - a.) Pattern yang pertama didefinisikan
 - b.) Longest match
 - c.) Shortest match
 - d.) Pattern yang terakhir didefinisikan
4. Tool re2c adalah. . .
 - a.) Parser generator
 - b.) Lexer generator untuk C/C++
 - c.) Optimizer
 - d.) Linker
5. Keuntungan lexer generator dibanding hand-written lexer adalah. . .

- a.) Kontrol lebih penuh atas implementasi
- b.) Produktivitas lebih tinggi dan specification lebih mudah di-maintain
- c.) Lebih cepat saat runtime
- d.) Tidak memerlukan dependency eksternal

18.1.5 Quiz Bab 5: Context-Free Grammar dan Parsing

1. CFG $G = (V, \Sigma, R, S)$: S menyatakan...
 - a.) Himpunan terminal
 - b.) Himpunan nonterminal
 - c.) Start symbol
 - d.) Himpunan production
2. Grammar disebut ambiguous jika...
 - a.) Memiliki left recursion
 - b.) Memiliki lebih dari satu production untuk satu nonterminal
 - c.) Ada string yang dapat di-parse dengan lebih dari satu parse tree
 - d.) Memiliki ϵ -production
3. Eliminasi left recursion diperlukan agar grammar...
 - a.) Dapat di-parse dengan bottom-up parser
 - b.) Dapat di-parse dengan recursive descent (top-down) parser
 - c.) Menjadi unambiguous
 - d.) Mendukung semua fitur bahasa
4. Notasi BNF adalah singkatan dari...
 - a.) Binary Normal Form
 - b.) Backus-Naur Form
 - c.) Boolean Notation Format
 - d.) Base Notation Form
5. Fase kompilasi yang menggunakan CFG adalah...
 - a.) Lexical analysis
 - b.) Syntax analysis
 - c.) Semantic analysis
 - d.) Code optimization

18.1.6 Quiz Bab 6: Top-Down Parsing dan Recursive Descent

1. Top-down parsing membangun parse tree. . .
 - a.) Dari leaves ke root
 - b.) Dari root (start symbol) ke leaves
 - c.) Secara paralel
 - d.) Dari tengah ke kedua ujung
2. Recursive descent parser cocok untuk grammar. . .
 - a.) Dengan left recursion
 - b.) LL(1) yang unambiguous
 - c.) LR(1) saja
 - d.) Yang ambiguous
3. Dalam recursive descent, setiap nonterminal biasanya diimplementasikan sebagai. . .
 - a.) Struct atau class
 - b.) Fungsi atau prosedur
 - c.) Macro
 - d.) Global variable
4. Precedence operator dalam recursive descent untuk ekspresi aritmatika ditangani dengan. . .
 - a.) Urutan aturan grammar (layering: $\text{expression} \rightarrow \text{term} \rightarrow \text{factor}$)
 - b.) Tabel lookup
 - c.) Register khusus
 - d.) Stack terpisah
5. Top-down parsing menggunakan. . .
 - a.) Rightmost derivation
 - b.) Leftmost derivation
 - c.) Bottom-up reduction
 - d.) Reverse derivation

18.1.7 Quiz Bab 7: Bottom-Up Parsing dan Parser Generator

1. Bottom-up parsing membangun parse tree...
 - a.) Dari root ke leaves
 - b.) Dari leaves (input tokens) ke root (start symbol)
 - c.) Hanya untuk ekspresi
 - d.) Menggunakan recursive calls
2. Operasi **reduce** dalam shift-reduce parsing berarti...
 - a.) Memasukkan token ke stack
 - b.) Mengganti handle di stack dengan LHS production
 - c.) Mengeluarkan token dari stack
 - d.) Memindahkan pointer input
3. LALR(1) parser merupakan pilihan populer karena...
 - a.) Lebih powerful dari CLR(1)
 - b.) Tabel parsing lebih kecil daripada CLR(1) tapi tetap cukup powerful
 - c.) Tidak memerlukan lookahead
 - d.) Dapat menangani grammar ambiguous
4. Bison adalah...
 - a.) Lexer generator
 - b.) Parser generator (implementasi GNU dari Yacc)
 - c.) Optimizer
 - d.) Assembler
5. Handle dalam bottom-up parsing adalah...
 - a.) Simbol paling kiri yang akan di-expand
 - b.) Substring yang cocok dengan RHS production dan reduced menuju start symbol
 - c.) Lookahead token
 - d.) State awal parser

18.1.8 Quiz Bab 8: Parser Generator (Bison/Yacc) dan Praktikum

1. File grammar Bison biasanya berekstensi. . .
 - a.) .l
 - b.) .y
 - c.) .lex
 - d.) .bnf
2. Semantic action dalam Bison berfungsi untuk. . .
 - a.) Mendefinisikan token
 - b.) Membangun AST atau melakukan aksi saat production di-reduce
 - c.) Menangani error saja
 - d.) Mengatur precedence
3. Integrasi Flex dan Bison: Flex menghasilkan. . .
 - a.) Parser; Bison menghasilkan lexer
 - b.) Lexer; Bison menghasilkan parser
 - c.) Keduanya menghasilkan parser
 - d.) Keduanya menghasilkan lexer
4. %token dalam file Bison digunakan untuk. . .
 - a.) Mendefinisikan nonterminal
 - b.) Mendeklarasikan token yang dikirim lexer
 - c.) Mendefinisikan semantic value
 - d.) Menentukan start symbol
5. Bison default menggunakan. . .
 - a.) LR(0) parsing
 - b.) LALR(1) parsing
 - c.) SLR(1) parsing
 - d.) GLR parsing

18.1.9 Quiz Bab 9: Abstract Syntax Tree (AST)

1. Perbedaan utama AST dengan parse tree adalah...
 - a.) AST lebih detail daripada parse tree
 - b.) AST menghilangkan detail sintaksis yang tidak relevan; lebih kompak
 - c.) Parse tree tidak memiliki root
 - d.) AST hanya untuk ekspresi
2. Traversal **in-order** pada AST ekspresi $a + b$ mengunjungi node dalam urutan...
 - a.) +, a, b
 - b.) a, +, b
 - c.) a, b, +
 - d.) +, b, a
3. Visitor pattern biasanya digunakan untuk...
 - a.) Mem-parse token
 - b.) Traverse dan memproses node AST (misalnya pretty-print, analisis)
 - c.) Generate lexer
 - d.) Allokasi register
4. Constant folding pada AST mengubah...
 - a.) Variabel menjadi konstanta
 - b.) Ekspresi konstanta (mis. $3+5$) menjadi satu nilai (mis. 8)
 - c.) Semua node menjadi leaf
 - d.) AST menjadi parse tree
5. AST digunakan sebagai input oleh...
 - a.) Hanya lexical analyzer
 - b.) Semantic analysis dan code generation
 - c.) Hanya parser
 - d.) Hanya optimizer

18.1.10 Quiz Bab 10: Symbol Table dan Scope Management

1. Symbol table menyimpan informasi tentang...
 - a.) Hanya keywords
 - b.) Identifier (variabel, fungsi, tipe) beserta atribut
 - c.) Hanya variabel global
 - d.) Hanya nama fungsi
2. Nested scope biasanya diimplementasikan dengan...
 - a.) Array linear
 - b.) Stack of hash tables (atau struktur serupa)
 - c.) Single global table
 - d.) Linked list tanpa hierarchy
3. Shadowing terjadi ketika...
 - a.) Variabel tidak dideklarasikan
 - b.) Identifier di inner scope menutupi identifier sama di outer scope
 - c.) Ada dua variabel dengan tipe berbeda
 - d.) Fungsi dipanggil dengan argumen salah
4. Operasi **lookup** pada symbol table berarti...
 - a.) Menambah entry baru
 - b.) Mencari identifier dan mengembalikan atributnya (jika ada)
 - c.) Menghapus scope
 - d.) Mendekomposisi tabel
5. Symbol table digunakan oleh...
 - a.) Hanya lexical analyzer
 - b.) Semantic analysis, type checker, dan code generator
 - c.) Hanya parser
 - d.) Hanya optimizer

18.1.11 Quiz Bab 11: Type Checking dan Semantic Analysis

1. Static type checking dilakukan...
 - a.) Saat runtime
 - b.) Saat compile time
 - c.) Hanya untuk fungsi
 - d.) Hanya untuk variabel global
2. Type promotion `int` \rightarrow `float` berarti...
 - a.) float bisa di-assign ke int tanpa cast
 - b.) int bisa di-assign ke float (implicit conversion)
 - c.) int dan float tidak kompatibel
 - d.) Hanya explicit cast yang diperbolehkan
3. Output semantic analysis biasanya berupa...
 - a.) Token stream
 - b.) Annotated AST (dengan informasi tipe) dan daftar error semantik
 - c.) Machine code
 - d.) Parse tree mentah
4. Contoh semantic error adalah...
 - a.) Token tidak valid
 - b.) Variabel tidak dideklarasikan, type mismatch, wrong number of arguments
 - c.) Kurung tidak seimbang
 - d.) Keyword salah eja
5. Dynamic type checking biasanya digunakan oleh bahasa...
 - a.) C dan C++
 - b.) Python dan JavaScript (tanpa static typing)
 - c.) Rust
 - d.) Java

18.1.12 Quiz Bab 12: Intermediate Code Generation

1. Three-address code biasanya berbentuk...
 - a.) Satu operasi per instruksi (mis. $t = a \text{ op } b$)
 - b.) Satu operand per instruksi
 - c.) Tree structure
 - d.) Bytecode stack-based
2. IR (Intermediate Representation) bersifat machine-independent artinya...
 - a.) Hanya untuk satu arsitektur
 - b.) Tidak tergantung arsitektur target; memungkinkan portabilitas
 - c.) Harus dioptimasi manual
 - d.) Tidak bisa diubah
3. Common subexpression elimination adalah...
 - a.) Menghapus semua ekspresi
 - b.) Menggunakan hasil komputasi yang sama untuk ekspresi yang muncul berulang
 - c.) Mengganti variabel dengan konstanta
 - d.) Menghapus dead code
4. Quadruples merepresentasikan instruksi dalam bentuk...
 - a.) (op, arg1, arg2, result)
 - b.) Hanya (op, result)
 - c.) Tree
 - d.) Daftar token
5. Input generator TAC (three-address code) adalah...
 - a.) Token stream
 - b.) AST (biasanya hasil semantic analysis)
 - c.) Source code mentah
 - d.) Machine code

18.1.13 Quiz Bab 13: Runtime Environment dan Memory Management

1. Activation record (stack frame) digunakan untuk...
 - a.) Menyimpan kode program
 - b.) Setiap pemanggilan fungsi: parameter, return address, variabel lokal
 - c.) Hanya variabel global
 - d.) Hanya konstanta
2. Stack dalam memory layout biasanya tumbuh...
 - a.) Ke atas (low \rightarrow high address)
 - b.) Ke bawah (high \rightarrow low address)
 - c.) Secara acak
 - d.) Hanya untuk heap
3. Variabel global dan static biasanya dialokasikan di...
 - a.) Stack
 - b.) Heap
 - c.) Static/global data region
 - d.) Register saja
4. Garbage collection umumnya mengelola...
 - a.) Stack frame
 - b.) Objek di heap yang tidak lagi direferensi
 - c.) Variabel lokal
 - d.) Kode program
5. Calling convention mengatur...
 - a.) Hanya urutan parameter
 - b.) Cara mem-pass parameter, return value, siapa membersihkan stack
 - c.) Hanya return value
 - d.) Hanya alokasi heap

18.1.14 Quiz Bab 14: Code Generation

1. Instruction selection adalah. . .
 - a.) Memilih instruksi mesin untuk setiap operasi IR
 - b.) Memilih register saja
 - c.) Memilih optimasi
 - d.) Memilih target OS
2. Register allocation bertujuan. . .
 - a.) Mengurangi jumlah instruksi
 - b.) Memetakan variabel/temporary ke register (atau memory jika perlu)
 - c.) Hanya menggunakan stack
 - d.) Menghapus label
3. RISC-V adalah contoh. . .
 - a.) Lexer generator
 - b.) Instruction set architecture (ISA) / arsitektur target
 - c.) Format IR
 - d.) Optimizer
4. Input code generator biasanya. . .
 - a.) Token stream
 - b.) IR (mis. TAC), symbol table, spesifikasi target
 - c.) Hanya source code
 - d.) Hanya AST tanpa IR
5. Output code generator dapat berupa. . .
 - a.) Hanya binary
 - b.) Assembly code atau machine code
 - c.) Hanya AST
 - d.) Hanya symbol table

18.1.15 Quiz Bab 15: Optimasi Kompilator Dasar

1. Optimasi kompilator tidak boleh...
 - a.) Mengubah semantik program
 - b.) Mengurangi ukuran kode
 - c.) Meningkatkan kecepatan eksekusi
 - d.) Melakukan constant folding
2. Constant folding mengubah $x = 3 + 5$ menjadi...
 - a.) $x = 3 + 5$
 - b.) $x = 8$
 - c.) Menghapus seluruh baris
 - d.) $x = 35$
3. Dead code elimination menghapus...
 - a.) Semua konstanta
 - b.) Kode yang tidak pernah dieksekusi atau hasilnya tidak dipakai
 - c.) Semua komentar
 - d.) Semua label
4. Basic block adalah...
 - a.) Seluruh program
 - b.) Urutan instruksi dengan satu entry, satu exit, tidak ada branch di tengah
 - c.) Hanya satu instruksi
 - d.) Hanya loop
5. Constant propagation mengganti...
 - a.) Semua variabel dengan konstanta
 - b.) Penggunaan variabel yang nilainya diketahui konstan dengan nilai tersebut
 - c.) Semua operator
 - d.) Semua label

18.2 Kunci Jawaban

Cara membaca: Untuk Quiz Bab 1, jawaban benar soal 1 = **b**, soal 2 = **c**, soal 3 = **b**, soal 4 = **d**, soal 5 = **c**. Demikian seterusnya untuk bab lainnya.

Bab	Materi	1	2	3	4	5
1	Pengenalan Kompilator	b	c	b	d	c
2	Regular Expression dan FA	b	b	b	b	b
3	Implementasi Lexer	b	b	c	b	b
4	Lexer Generator	b	b	b	b	b
5	CFG dan Parsing	c	c	b	b	b
6	Top-Down Parsing	b	b	b	a	b
7	Bottom-Up Parsing	b	b	b	b	b
8	Parser Generator (Bison)	b	b	b	b	b
9	AST	b	b	b	b	b
10	Symbol Table	b	b	b	b	b
11	Type Checking	b	b	b	b	b
12	Intermediate Code	a	b	b	a	b
13	Runtime Environment	b	b	c	b	b
14	Code Generation	a	b	b	b	b
15	Optimasi	a	b	b	b	b

Tabel 18.1: Kunci jawaban quiz Bab 1–15. Kolom 1–5 = nomor soal; isi = pilihan benar (a, b, c, atau d).