RENCANA PEMBELAJARAN SEMESTER (RPS)

Berbasis Outcome-Based Education (OBE)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS LOREM IPSUM

MATA KULIAH TEKNIK KOMPILASI

1. Identitas Mata Kuliah

Nama Program Studi	: Teknik Informatika
Nama Mata Kuliah	: Teknik Kompilasi
Kode Mata Kuliah	: TIF-401
Semester	: 6 (Enam)
SKS / Bobot Kredit	: 3 SKS (2 Teori, 1 Praktikum)
Dosen Pengampu	: Tim Dosen Teknik Informatika
Tanggal Penyusunan	: 5 Februari 2026
Prasyarat	: Struktur Data, Algoritma, Pemrograman

2. Capaian Pembelajaran Lulusan (CPL)

CPL yang dibebankan pada mata kuliah ini mencakup kompetensi lulusan dalam aspek pengetahuan, keterampilan, dan sikap:

- **CPL-1 (Pengetahuan):** Menguasai konsep teoretis bidang teknik kompilasi secara mendalam, termasuk analisis leksikal, sintaksis, semantik, dan generasi kode, serta mampu memformulasikan penyelesaian masalah prosedural dalam pengembangan kompilator.
- **CPL-2 (Keterampilan Umum):** Mampu menerapkan pemikiran logis, kritis, sistematis, dan inovatif dalam konteks pengembangan atau implementasi sistem kompilator yang memperhatikan dan menerapkan nilai humaniora dan etika profesi.
- **CPL-3 (Keterampilan Khusus):** Mampu merancang, mengimplementasikan, dan mengevaluasi sistem kompilator lengkap menggunakan teknik-teknik modern dengan mempertimbangkan efisiensi, optimasi, dan kualitas kode yang dihasilkan sesuai standar industri.
- **CPL-4 (Sikap):** Menunjukkan sikap bertanggung jawab atas pekerjaan di bidang keahliannya secara mandiri dan mampu bekerja sama dalam tim dalam proyek pengembangan perangkat lunak kompilator.

3. Capaian Pembelajaran Mata Kuliah (CPMK)

Kemampuan atau kompetensi spesifik yang diharapkan mahasiswa kuasai setelah menyelesaikan mata kuliah:

- **CPMK-1:** Mahasiswa mampu menjelaskan arsitektur kompilator secara keseluruhan, termasuk fase-fase utama: analisis leksikal, parsing, analisis semantik, generasi kode intermediate, optimasi, dan generasi kode.
- **CPMK-2:** Mahasiswa mampu membangun lexer dan parser menggunakan grammar formal (misalnya regular expressions, context-free grammars), finite automata (NFA/DFA), dan teknik parsing top-down dan bottom-up.
- **CPMK-3:** Mahasiswa mampu menerapkan syntax-directed translation dan analisis semantik untuk menegakkan aturan bahasa, menyelesaikan tipe, mengelola scope, dan menangani error.
- **CPMK-4:** Mahasiswa mampu merancang dan menghasilkan representasi kode intermediate (misalnya three-address code, DAGs) dan melakukan optimasi yang tidak bergantung pada mesin (optimasi basic block, data-flow analysis).
- **CPMK-5:** Mahasiswa mampu mengimplementasikan generasi kode untuk arsitektur target, memetakan kode intermediate menjadi kode target yang efisien, mengelola struktur run-time (activation records, memory layout, symbol tables), dan menerapkan optimasi khusus mesin dasar.
- **CPMK-6:** Mahasiswa mampu mengevaluasi dan membandingkan alat kompilator (seperti parser generators) dan pendekatan optimasi, menganalisis trade-off antara waktu kompilasi, kualitas kode, dan efisiensi runtime.

4. Sub-CPMK / Indikator Pencapaian

Penjabaran CPMK menjadi indikator yang lebih terukur dan dapat diuji:

- **Sub-CPMK 1.1:** Menjelaskan perbedaan antara interpreter dan compiler
- **Sub-CPMK 1.2:** Mengidentifikasi fase-fase kompilator dalam arsitektur kompilator nyata
- **Sub-CPMK 1.3:** Menganalisis trade-off antara one-pass vs multi-pass compiler
- **Sub-CPMK 1.4:** Menjelaskan hierarki Chomsky dan hubungannya dengan automata (FA, PDA, TM)
- **Sub-CPMK 2.1:** Membuat regular expression untuk token spesifik bahasa
- **Sub-CPMK 2.2:** Mengimplementasikan NFA dan DFA untuk token recognition
- **Sub-CPMK 2.3:** Membangun recursive descent parser untuk grammar LL(1)

- **Sub-CPMK 2.4:** Menggunakan parser generator (Flex/Bison) untuk bahasa sederhana
- **Sub-CPMK 2.5:** Menjelaskan ekivalensi NFA–DFA dan konsep minimalisasi DFA
- **Sub-CPMK 2.6:** Menganalisis ambiguitas grammar, left recursion, left factoring; menjelaskan sifat grammar LL(1) dan LR(1) serta konflik pada parsing
- **Sub-CPMK 3.1:** Mengimplementasikan symbol table dengan nested scopes
- **Sub-CPMK 3.2:** Melakukan type checking untuk ekspresi kompleks
- **Sub-CPMK 3.3:** Menangani semantic error dengan pesan yang informatif
- **Sub-CPMK 3.4:** Menerapkan attribute grammar (synthesized/inherited) dan syntax-directed definition (SDD) atau translation schemes
- **Sub-CPMK 4.1:** Merancang three-address code representation
- **Sub-CPMK 4.2:** Mengimplementasikan basic block identification
- **Sub-CPMK 4.3:** Melakukan optimasi local (constant folding, dead code elimination)
- **Sub-CPMK 4.4:** Menjelaskan konsep data-flow analysis (lattice, fixed point) untuk optimasi
- **Sub-CPMK 5.1:** Memetakan intermediate code ke assembly target
- **Sub-CPMK 5.2:** Mengimplementasikan register allocation sederhana
- **Sub-CPMK 5.3:** Mengelola activation records dan calling conventions
- **Sub-CPMK 6.1:** Membandingkan performa hand-written vs generated compiler
- **Sub-CPMK 6.2:** Menganalisis trade-off compilation time vs runtime efficiency

5. Materi Pembelajaran (Bahan Kajian)

Daftar topik materi yang relevan dengan Sub-CPMK dan CPMK:

1. Pengenalan Kompilator dan Arsitektur Kompilator
2. Bahasa Pemrograman (tingkat mesin, rendah, menengah, tinggi)
3. Translator: definisi; Assembler, Interpreter, Compiler; perbedaan compiler dan interpreter
4. Tahapan Kompilasi: Front End (analisis leksikal, sintaksis, semantik), Back End (intermediate code, optimizer, code generator), Symbol Table Management, Error Handling
5. Runtime Environment (static, stack, heap allocation)

6. Teori bahasa formal dan hierarki Chomsky (Type 3–0); hubungan dengan automata (FA, PDA, TM)
7. Token, Lexeme, dan Pattern
8. Ekspresi reguler: definisi, operasi, dan notasi
9. Finite automata: NFA, DFA; konversi NFA ke DFA; minimalisasi DFA
10. Implementasi lexical analyzer: struktur data dan algoritma scanning; lexer hand-written dan generator-based
11. Context-Free Grammar: definisi, notasi BNF dan EBNF
12. Derivasi dan parse tree; ambiguitas dalam grammar
13. Sifat grammar: left recursion, left factoring; himpunan FIRST dan FOLLOW
14. Top-Down parsing: recursive descent, predictive parser, LL(1)
15. Bottom-Up parsing: shift-reduce parser, operator precedence parser
16. LR parser: Canonical LR, SLR, LALR; konflik shift-reduce dan reduce-reduce; kelas grammar LL(k) dan LR(k)
17. Parser generator (Flex/Bison) dan semantic actions
18. Abstract Syntax Tree (AST) dan struktur data
19. Pengertian analisis semantik; attribute grammar (synthesized, inherited)
20. Syntax-directed definition (SDD) dan translation schemes
21. Symbol table dan scope management
22. Type checking: type system, type expression, ekspresi dan statement; type conversion dan type coercion
23. Jenis dan penanganan semantic error
24. Intermediate code: three-address code, quadruples, triples, postfix notation, syntax tree
25. Optimasi lokal: constant folding, dead code elimination, strength reduction; basic block
26. Optimasi global: loop optimization, common subexpression elimination
27. Data-flow analysis: reaching definition, live variable, available expression; lattice, fixed point, monotonicity
28. Runtime environment dan memory management (activation records, calling conventions)
29. Code generation: target language, register allocation, instruction selection
30. Peephole optimization

31. Teknik modern: LLVM IR, JIT compilation, optimasi levels

32. Evaluasi dan perbandingan compiler tools

6. Metode Pembelajaran

Strategi atau pendekatan pembelajaran yang dipilih sesuai OBE yang menekankan aktivitas mahasiswa:

- **Ceramah Interaktif:** Penjelasan konsep kompilator dengan diskusi tanya jawab
- **Problem-Based Learning (PBL):** Mahasiswa menyelesaikan permasalahan kompilasi nyata
- **Project-Based Learning:** Pengembangan compiler lengkap sebagai proyek bertahap
- **Praktikum Terbimbing:** Implementasi komponen kompilator dengan bimbingan
- **Peer Review:** Mahasiswa melakukan code review terhadap implementasi compiler
- **Flipped Classroom:** Mahasiswa mempelajari materi teori sebelum kelas, praktik di kelas
- **Studi Kasus:** Analisis compiler nyata (GCC, Clang, V8 JavaScript Engine)

7. Pengalaman Belajar Mahasiswa

Deskripsi tugas, aktivitas, atau pengalaman belajar yang mendukung pencapaian Sub-CPMK:

- Mengidentifikasi token, lexeme, dan pattern pada contoh bahasa sederhana
- Membuat derivasi dan parse tree untuk kalimat dari grammar yang diberikan
- Menghitung himpunan FIRST dan FOLLOW untuk grammar
- Menganalisis konflik shift-reduce dan reduce-reduce pada tabel LR
- Menerapkan attribute grammar (synthesized/inherited) pada pohon syntax
- Menghitung reaching definition, live variable, atau available expression pada basic block sederhana
- Membaca dan merangkum konsep dari buku/referensi terbuka (Thain, Bergmann, Wikibooks Formal Languages)
- Mengerjakan soal formal: desain grammar untuk bahasa sederhana, analisis ambiguitas, konstruksi NFA/DFA dari RE, penjelasan konflik parsing

- Presentasi singkat konsep teori (hierarki Chomsky, NFA vs DFA, LL vs LR, atau data-flow analysis)
- Menganalisis trade-off teori (expressive power vs complexity) untuk pilihan desain kompilator
- Menganalisis arsitektur compiler sederhana (TinyCC, TCC)
- Mengimplementasikan lexer hand-written untuk subset bahasa C
- Membangun parser recursive descent untuk ekspresi aritmatika
- Menggunakan Flex dan Bison untuk parser generation
- Mengimplementasikan symbol table dengan nested scopes
- Melakukan type checking dan semantic analysis
- Menghasilkan three-address code dari AST
- Memetakan intermediate code ke assembly target
- Mengimplementasikan optimasi compiler dasar
- Mengeksplorasi LLVM infrastructure dan JIT compilation
- Berkolaborasi dalam tim untuk mengembangkan compiler lengkap
- Mempresentasikan hasil proyek compiler
- Melakukan benchmarking dan analisis performa

8. Kriteria, Indikator, dan Bobot Penilaian

Teknik/alat asesmen dipetakan ke Sub-CPMK/CPMK dengan bobot yang jelas:

Komponen	**Teknik Asesmen**	**Indikator/CPMK**	**Bobot (%)**
Kuis	Multiple Choice & Essay	Sub-CPMK 1.1, 1.2, 1.3	5
Tugas Teori / Problem Set	Soal grammar, automata, parsing, data-flow	Sub-CPMK 1.4, 2.5, 2.6, 4.4	8
Tugas Praktikum 1	Lexer Implementation	Sub-CPMK 2.1, 2.2, 2.4	8
Tugas Praktikum 2	Parser Implementation	Sub-CPMK 2.3, 2.4	7
Tugas Praktikum 3	Symbol Table	Sub-CPMK 3.1, 3.2, 3.3	8
Tugas Praktikum 4	Semantic Analysis + TAC	Sub-CPMK 3.3, 4.1, 4.2	8

Komponen	Teknik Asesmen	Indikator/CPMK	Bobot (%)
Tugas Praktikum 5	Code Generation	Sub-CPMK 5.1, 5.2, 5.3	6
UTS	Written Exam & Coding (konsep teori: bahasa formal, automata, sifat grammar, parsing)	CPMK-1, CPMK-2	20
Project Final	Complete Compiler	CPMK-2, CPMK-3, CPMK-4, CPMK-5	25
UAS	Comprehensive Exam (konsep teori & arsitektur kompilator, optimasi)	CPMK-1, CPMK-2, CPMK-3, CPMK-4, CPMK-5, CPMK-6	15
Total			**100**

Kriteria Penilaian:

- A (85-100): Menguasai semua CPMK dengan sangat baik, mampu menerapkan dalam kasus kompleks
- B (70-84): Menguasai sebagian besar CPMK dengan baik
- C (60-69): Menguasai CPMK dasar dengan cukup
- D (50-59): Menguasai sebagian kecil CPMK
- E (<50): Belum menguasai CPMK yang ditetapkan

Rubrik Penilaian Tugas Praktikum:

- **Correctness (40%):** Program berjalan dengan benar, mengatasi edge cases
- **Code Quality (30%):** Struktur kode, readability, dokumentasi
- **Completeness (20%):** Semua requirement terpenuhi
- **Testing (10%):** Test cases yang komprehensif

9. Evaluasi dan Refleksi Pembelajaran

Penilaian sumatif/formatif untuk memantau ketercapaian outcome secara menyeluruh:

- **Evaluasi Formatif:** Kuis mingguan, progress review proyek compiler, peer review untuk memberikan feedback berkelanjutan
- **Evaluasi Sumatif:** UTS dan UAS untuk mengukur pencapaian CPMK secara komprehensif
- **Refleksi Mahasiswa:** Jurnal belajar mingguan untuk refleksi diri terhadap pemahaman materi kompilator

- **Evaluasi Dosen:** Survey kepuasan mahasiswa di tengah dan akhir semester
- **Continuous Improvement:** Analisis hasil penilaian untuk perbaikan RPS di semester berikutnya

10. Daftar Referensi

Sumber belajar utama yang digunakan dalam penyusunan materi dan asesmen:

1. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Pearson Education. (Dragon Book)
2. Cooper, K. D., & Torczon, L. (2024). *Engineering a Compiler* (3rd ed.). Morgan Kaufmann. (TAA Textbook Excellence Award Winner 2024)
3. Grune, D., van Reeuwijk, K., Bal, H. E., Jacobs, C. J. H., & Langendoen, K. (2012). *Modern Compiler Design* (2nd ed.). Springer.
4. Levine, J. R. (2009). *flex & bison: Text Processing Tools*. O'Reilly Media.
5. Appel, A. W., & Palsberg, J. (2022). *Modern Compiler Implementation in Java* (2nd ed.). Cambridge University Press.
6. LLVM Project. (2024). *LLVM Language Reference Manual*. Retrieved from `https://llvm.org/docs/LangRef.html`
7. Thain, D. (2024). *Introduction to Compilers and Language Design*. Retrieved from `https://dthain.github.io/books/compiler/`
8. Bergmann, S. D. *Compiler Design: Theory, Tools, and Examples* (Java/C++ edition). Rowan University, open access (CC BY-NC-ND). Retrieved from `https://onlinebooks.library.upenn.edu/webbin/book/lookupid?key=olbp64271`
9. Wikibooks. *Theory of Formal Languages, Automata, and Computation* (Grammars and the Chomsky Hierarchy). Retrieved from `https://en.wikibooks.org/wiki/Theory_of_Formal_Languages,_Automata,_and_Computation`
10. Critchlow, C., & Eck, D. *Foundations of Computation*. Open Textbook Library. Retrieved from `https://open.umn.edu/opentextbooks/textbooks/166`
11. MIT OpenCourseWare. (2010). *6.035 Computer Language Engineering* (syllabus, lecture notes, exams). Retrieved from `https://ocw.mit.edu/courses/6-035-computer-language-engineering-spring-2010/`
12. ANTLR Project. (2024). *ANTLR 4 Documentation*. Retrieved from `https://www.antlr.org/`
13. UC San Diego CSE 231: Compiler Construction. (2024). Retrieved from `https://ucsd-cse231.github.io/sp24/`

14. Stanford University CS143: Compilers. (2024). Termasuk written assignments (WA1–WA4) untuk aspek teori dan problem sets. Retrieved from `https://web.stanford.edu/class/cs143/`

Disusun oleh,

Tim Dosen Teknik Informatika
Program Studi S1 Teknik Informatika
Fakultas Teknik

5 Februari 2026