Plano Analítico: Compiladores

1. Identificação da Unidade Curricular

- **Instituição:** Instituto Superior Politécnico de Ciências e Tecnologia (INSUTEC)
- **Curso:** Engenharia de Informática e Sistemas de Informação (EISI)
- **Classificação:** Disciplina Específica (Nuclear)
- **Ano:** 4º | **Semestre:** 1º (7º Semestre)
- **Créditos:** 6.0 UC
- **Carga Horária Total:** 90 Horas (60h de Contacto | 30h de Trabalho Complementar)

2. Apresentação e Justificação

A disciplina de Compiladores é fundamental para compreender como as linguagens de alto nível são transformadas em código executável pela máquina. O curso aborda a análise léxica, sintática e semântica, bem como a geração e otimização de código. Justifica-se pela necessidade de o engenheiro entender a eficiência das linguagens de programação e ser capaz de desenvolver tradutores ou interpretadores específicos, em conformidade com o **Decreto Presidencial 193/18**.

3. Competências a Desenvolver (Decreto 193/18)

3.1 Competências Instrumentais (Saber)

- Compreender a estrutura de um compilador e as suas diversas fases.
- Dominar a teoria das gramáticas formais e autómatos (Hierarquia de Chomsky).
- Entender os mecanismos de análise sintática descendente (LL) e ascendente (LR).

3.2 Competências Técnicas e Operacionais (Saber Fazer)

- **Desenvolvimento de Analisadores:** Implementar analisadores léxicos utilizando expressões regulares e autómatos finitos.
- **Construção de Parsers:** Utilizar ferramentas de geração de compiladores (ex: Lex/Yacc, Flex/Bison ou ANTLR).
- **Tratamento Semântico:** Implementar tabelas de símbolos e verificações de tipos.

3.3 Competências Atitudinais (Saber Ser/Estar)

- Demonstrar precisão e rigor na definição de linguagens e gramáticas.
- Valorizar a otimização de recursos computacionais através da eficiência do código gerado.

4. Conteúdo Temático (Estrutura de 90 Horas)

1. **Introdução à Compilação:** Tradutores, interpretadores e a anatomia de um compilador.
2. **Análise Léxica:** Expressões regulares, Autómatos Finitos (DFA/NFA) e geração de tokens.
3. **Análise Sintática:** Gramáticas Independentes de Contexto (GIC), derivações e árvores de derivação.
4. **Técnicas de Parsing:** Algoritmos preditivos LL(1) e algoritmos de deslocação-redução (SLR, LALR).
5. **Análise Semântica:** Verificação de tipos, âmbito de variáveis e gestão da tabela de símbolos.
6. **Geração de Código Intermédio:** Notação de três endereços e árvores sintáticas abstratas (AST).
7. **Otimização e Geração de Código Final:** Gestão de registos e tradução para código de máquina ou assembly.

5. Regime de Avaliação (Disciplina Específica)

- **Avaliação Contínua (40%):**
    - 1ª Frequência (Análise Léxica e Sintática): 13%
    - 2ª Frequência (Semântica e Geração de Código): 14%
    - **Projeto Prático:** Desenvolvimento de um compilador para uma linguagem simplificada: 13%
- **Exame Normal (60%):** Prova global teórica e prática.

6. Referências Bibliográficas (APA 7ª Ed.)

- Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). *Compilers: Principles, techniques, and tools* (2nd ed.). Pearson ("The Dragon Book").
- Louden, K. C. (2004). *Compiladores: Princípios e práticas*. Thomson Learning.

- Cooper, K., & Torczon, L. (2011). *Engineering a compiler* (2nd ed.). Morgan Kaufmann.

- Appel, A. W. (2002). *Modern compiler implementation in C*. Cambridge University Press.