Instructor: Phil Nelson

Time: MTWF 11am

Place: CF 314

Office: CF 471 (650-3035, phil.nelson@wwu.edu)

Office Hours: MWF 2:00-3:00, others by appointment.


Graded Work: 2 Tests, 3 Assignments and a Project.

Test Schedule: Test 1 is scheduled for Wednesday, October 31 and the final is scheduled for Wednesday, December 12, 8-10 AM.

Assignments: Three assignments during the first 3 weeks of the class.

Project: Write a working ATL/1 compiler. See the ATL/1 documentation. It will be due on the last class of dead week, December 7. You must schedule an evaluation session with the instructor during the first 2 days of test week.

Grading: Test 1 is worth 20% of your final grade and Test 2 is worth 20% of your final grade. Assignments are worth 10% of your final grade and the Project is worth 50% of your final grade. (You must earn at least 50% of the test points to pass this course.)

Course Outcomes: The course outcomes for this class are:

- Thorough understanding of the phases of a compiler
- Thorough understanding of lexical analysis and the use of scanner generators
- Thorough understand of predictive and LR parsing and the use of parser generators
- Basic understanding of Abstract Syntax Trees
- Thorough understanding of Semantic Analysis and the use of symbol tables
- Thorough understanding of activation records
- Basic understanding if Intermediate Code generation and intermediate representation trees
- Basic understanding of compilation issues for object-oriented languages
• Basic understanding of optimization issues and techniques
• Ability to construct a functioning compiler and run time environment for a block
  structured language with nested procedures and at least two parameter passing meth-
  ods

**Example Grades:** Percentages of the top score:

- A 90 - 100 %
- B 80 - 89 %
- C 65 - 79 %
- D 50 - 64 %

**Collaboration:** Each student *MUST* do their own programming. Original work is required. You should not see the source code of any other student, current or past, on this project. You may discuss problems using diagrams on scratch paper, but you should not see source code. Even helping a fellow student debug their program so that source code is seen should be avoided. Students having problems should e-mail me or visit me in my office.

**Cheating:** Is (obviously) not allowed. If you do cheat and are caught you will receive an F as your grade for the class. This includes **ALL** students knowingly involved in any cheating event. Not properly protecting your source code may be considered knowingly involved. If you give your password to your friend or allow access to your files or a machine on which your sources are stored, this can be considered knowingly involved. I use mechanical means to compare student programs, not only all students this quarter, but from students who took this class in the past. These comparisons are used to raise the possibility of cheating, but all decisions about cheating will be made by me after inspecting the programs of all students involved. **NOTE:** Making your source files for this class available for public, unprotected access will be considered cheating and may even get you an F for this class **AFTER** a passing grade has been reported and you have graduated.

**Late Work:** Work is due *at the beginning of class on the day due*. Work will be accepted up to TWO meetings of the class late and will be worth 75% of the original value. (For example, if the assignment is due on Wednesday, the second meeting of the class would be the following Monday.) Work later than two class periods will be worth nothing. A late project will be worth 75% of its original value if turned by the start of the final.

**Topics:** Topics to be covered in the approximate order.

1. Introduction to Compilers (Chapt 1)
2. A simple compiler (ATL/0) (Chapt 2)
3. hc - a hypothetical computer
4. ATL/1 - A Teaching Language
5. Scanners (Chapt 3)
6. flex - a scanner generator
7. Grammars (Chapt 4)
8. Parsers
9. Grammar Analysis
10. LL(1) Grammars and Parsers (Chapt 5)
11. LR(1) Grammars and Parsers (Chapt 6)
12. Syntax-directed Translation (Chapt 7)
13. Semantic Processing Techniques
14. Intermediate code
15. Symbol tables (Chapt 8)
16. Block structured symbol tables
17. Static Run-time Storage Organization (Chapt 12)
18. Building a symbol table, processing declarations (Chapt 8)
19. Processing yacc rules, building an AST (Chapt 7)
20. Semantic Analysis (Chapt 9)
21. Translating Control Structures (Chapt 11)
22. Translating Procedure and Functions
23. Target Code Generation (Chapt 13)
24. Optimization
25. other topics as time permits