CMSC430 Parser – Version 1
Project 2 – Due 11:59pm March 13, 2007

1. Parser Specifications

Write a parser using YACC, Bison or CUP that works with your previously written scanner for the NIP language. Make sure you modify the grammar to eliminate ambiguities and that the parser works for the modified grammar and accepts the same language. Details are:

(a) The compiler should run as a "standard" UNIX command line compiler. That is, you can include a filename on the command line, or read from STDIN. That is you can run the program as:

```
nip filename.nip
```

or

```
nip < filename.nip
```

If there are any options, include them before the filename:

```
nip –s filename.nip
```

or

```
nip –s < filename.nip
```

This program adds additional options. You can keep them separate, as in:

```
nip –s –t –q filename.nip
```

(b) Add the -q option, which causes the parser to output the program as a series of quadruples.

(c) Add the -t option which causes the parser to output the tokens in the standard output file in the order they are read by the scanner (e.g., the output you produced for the scanner project).

(d) Submit your parser, like the scanner, as a zip file to the submit server. Make sure you include the scanner as well, and also a makefile and a readme file to show how to run your compiler.

(e) Following the program listing (if -s specified), print the symbol table as a table with the following fields:

```
name  value  proc_name_declared_in  act_rec_offset  type  other
```

For this project only the *name* and *value* fields need to be filled in. *Value* is the number associated with the token for that identifier (e.g., the symbol table address). Project 3 will require a rewrite of the symbol table routines.

- As before, some test data is available in the files *~mvz/test0.nip* through *test4.nip* on the grace cluster. *test0.nip* and *test4.nip* are the most informative. These test files can be used for all phases of the compiler project.

- Projects are due March 13, 2007 before 11:59pm. Turn in a well documented source program, a make file to compile it, a readme file to explain how to run your compiler, and a file of test data that you believe tests the parser as a zip file to the submit server. Your Parser will be tested with this test data as well as other test data. There is a penalty for submitting your project late. 10% a day for up to 3 days. After that the penalty goes up even more. (Amount TBD).