University of Puerto Rico
Mayagüez Campus
College of Engineering
Department of Electrical and Computer Engineering

ICOM4029 – Compilers
Professor: Bienvenido Vélez

Lab 1 – Introduction to Cool

Cool, or Class Object-Oriented Language, is a simple language specifically designed for use in a compilers course. It contains enough necessary properties and features for learning how a compiler works and to be used as a basis for developing your own compiler.
The following steps will guide you on creating your first *Cool* program, compiling it, and running it using the *spim* MIPS emulator.

Side note: Linux Access
To complete this lab you will need access to a Linux system. You may install a Linux VM on your laptop or PC or get an account at the Amadeus computing lab. To request an account send an email to Bienvenido.Velez@upr.edu.

1. Preparation

Step 1. Get access to a linux distribution
An easy way is to get an account for the Amadeus lab PCs or to install a Linux distribution as a VM on you PC.

Step 2. Download and extract the Cool language support code
You may download the distribution from the course website using a browser running on your Linux machine.

Step 3. Install emacs and spim
Install the following tools in your linux environment if they do not already exist: emacs, spim. In Fedora you can use yum to install packages easily as follows:

```
yum –i install emacs
yum –i install spim
```

You must be connected to the Internet for yum to be able to download these packages.

Step 4. Login and setup your environment variables
After logging on, open a terminal window and enter the following commands in the order they appear:

```
export PATH=$PATH:~<userid>/cool/bin:~
```

You should add this line to the `.bashrc_profile` file in your home directory so that you can use Cool executables from any folder.

2. Writing the Program

Now, we are going to write a simple Cool program that displays "Hello World!". Open up emacs or any other text editor and write the following code:

```
class Main {
   out : IO <- new IO;

   main(): Object {
      out.out_string("Hello World!\n")
   };
};
```

Save your file as *hello.cl* when finished.

3. Compiling it

To compile your program, go to the folder where you saved it and enter:

```
coolc –o hello.s hello.cl
```

(the "`–o hello.s`" can be omitted). This will create a file named *hello.s* which is the MIPS assembly code that resulted from the compilation.

4. Running it

To actually run the program and see its output we are going to use a MIPS emulator called *spim* since the lab's computers have a different architecture (*x86*).

To run your compiled Hello World program (*hello.s*), enter the following:

```
spim –trap_file ~<userid>/lib/trap.handler –file hello.s
```

The screen will display *spim*'s initialization messages and then run the program, which will output

"`Hello World!`"

5. Sample Program 2

Write the following *Cool* program (*stat.cl*):

```
class Main inherits IO {
  i : Int <- 0;
  number : Int;
  max : Int <- 0;
  sum : Int <- 0;
  maxStr : String;
  avgStr : String;
  conv : A2I <- new A2I;

  main() : Object {
  {
    while (i < 4) loop {
      out_string("Enter an integer: ");
      number <- in_int();
      if (max < number)
      then max <- number
      else 0
      fi;
      i <- i + 1;
      sum <- sum + number;
    } pool;
    maxStr <- conv.i2a(max);
    out_string(("The greatest # was: ".concat(maxStr)).concat("\n") );
    avgStr <- conv.i2a(sum / 4);
    out_string((("The average is: ").concat(avgStr)).concat("\n"));
  }
  };
};
```

Copy the *atoi.cl* sample program from the *Cool* `examples` directory:

```
cp ~<userid>/cool/examples/atoi.cl .
```

Compile the program:

```
coolc –o stat.s atoi.cl stat.cl
```

Run it:

```
spim –trap_file ~<userid>/lib/trap.handler –file stat.s
```

VI. Closing Notes

There are some sample cool programs at `~<userid>/cool/examples`. Your first programming assignment (PA1) will have you writing a stack machine in cool so you should take a look at the examples, read the cool manual at least up to Section 11 and get familiar with cool by writing some sample programs of your own.