COMP 621 Program Analysis and Transformations
Assignment #0

Getting Ready

Due: Tuesday, September 13, 2015 – beginning of class

Overview:

The purpose of this assignment is to get you engaged in the course and ready to do assignment #1. Assignment #0 is not worth any marks, but I do want a report from you handed in, and I will be giving them a grade of Unsatisfactory/Satisfactory/Excellent. I would like you take this assignment seriously. For each question, just follow the directions about what to do and what to report. If you had trouble with the question, please describe what the difficulty was.

Question 0: *Check out the MyCourses page and the course web page*

Visit the course web site at `http://www.sable.mcgill.ca/~hendren/621/index.html` and the My Courses for COMP 621. On the MyCourses site, introduce yourself on the discussion list entitled "Introduce yourself".

Question 1: *Background - Compilers*

If you have already taken an introductory compiler course, then the answer for this question is just the name, and a short description of the course.

If you have not taken an introductory compiler course, then you can fill in the background you need for this course by doing some reading from any standard compiler textbook. To answer this question say what reading you did to prepare.

Question 2: *Background - Object-Oriented Programming*

In this course you will be expected to be able to understand large frameworks written in Java, and if you plan to do a project with the VM implementation, you should be comfortable with C++. To answer this question, please summarize your experience with Java and C++.

If you don't have significant experience, especially with Java, then please do some background reading on Java, and write some practice programs in Java. If this is your situation, please summarize how you prepared for Java.

Question 3: *Background in MATLAB*

MATLAB is a very popular language that is challenging for compiler optimizations. MATLAB was made to be easy for Scientists/Engineers to use, but it is actually very hard to analyze and optimize. In this course we will use the McLAB analysis framework for MATLAB to practice with designing new compiler analyses. However, before you can understand how to build the analyses you need to understand the foundations of the MATLAB language.

Both Mathworks' MATLAB and an open source version, Octave, are installed on the SOCS machines in Trottier. You may also find them on the research machines in your graduate labs (if they are not there, please ask your research groups to install them, McGill has an institutional license to install on all university-owned computers).

If you are not familiar with MATLAB, you can refer to the following documents:

- `http://www.mathworks.com/help/matlab/`, for the documentation on Matlab basics.
- `http://ta.twi.tudelft.nl/nw/users/domenico/intro_fem/afternoon_lab_sessions/Matlab.manual.pdf`
- `http://ubcmatlabguide.github.io`
- `http://www.gnu.org/software/octave/` for documentation for Octave.

To experiment with MATLAB and to start to understand performance issues with MATLAB, you should write a short numerical program, one version using MATLAB vector notation and one version using explicit loops. For the explicit loops version, be sure to "pre-allocate" any arrays you are defining in the body of the loop using the built-in `zeros` function. This is to prevent MATLAB from growing the array on each loop iteration.

You can use code you find online or in a book, but make sure you say where you got the code from, and make sure you understand the code thoroughly. You can also write your own code.

Check that your vector version and your loop version compute the same results.

Time the two versions for various problem sizes. Hint: you can use the commands `tic` and `toc`.

Each test that you time should be in its own function, and the calls to `tic` and `toc` should be outside of those functions. This will give the JIT compiler the best opportunity to optimize the code.

To answer this question, provide the two versions of your program, and briefly describe the results of your timing, and your interpretation of those results. You might have fun searching the web for various advice about how to use vectors to speed up MATLAB, and seeing if that advice is true or not. One relevant research paper can be found at:
`http://dl.acm.org/citation.cfm?id=331972` .[1]

[1]Note that any paper in the ACM digital library can be freely download from a McGill IP address. If you are working from home, use a McGill VPN.

Hint for this question: MATLAB has a "JIT-accelerator" now, whereas before it was only an interpreter.

If you feel even more ambitious, try running MATLAB in interpreter-only mode and repeat the experiment. You can turn the JIT on and off using the MATLAB commands `feature accel on` and `feature accel off`. This will only work for older versions of MATLAB, in the most recent versions the JIT cannot be turned off. What did you find?

To be even more ambitious, try it also on Octave, which is an open source interpreter. What did you find?

Recommended: using the Wu-Wei benchmarking toolkit

You may use the freshly baked Wu-Wei-Benchmarking Toolkit written by your TA to automate the testing of your benchmark on various versions of MATLAB and Octave and make your life easier for the last parts of the question. You will be invited to use the same tools in the next assignment so learning the tools now will give you a head start.

In order to use Wu-Wei for the task, log into any of the SOCS computer and follow the installation instructions:

`https://github.com/Sable/wu-wei-handbook#installing-the-tools`

Then execute the following commands in the linux prompt:

```
mkdir a0-q4
cd a0-q4
wu init   # create a repository for benchmarking
wu install https://github.com/Sable/COMP621-a0.git
wu install https://github.com/Sable/matlab-implementation-template.git \
   --destination benchmarks/template/implementations/matlab-vector
wu install https://github.com/Sable/matlab-implementation-template.git \
   --destination benchmarks/template/implementations/matlab-loop
```

Test the installation by running the implementations against all versions of MATLAB and Octave and generate a report:

```
wu run -v # Enter a short-name when prompted (ex: lab2-2)
wu report
```

Make sure the run executes correctly by checking by hand that the output is the same for all executions. Then check that the report is displayed, you should see something like:


```
### Invariants (configuration parameters that are the same for all runs) ###

| category   | short-name    |
| ---------- | ------------- |
| benchmark  | template      |
| compiler   | matlab-concat |
| platform   | sableintel    |
| input-size | medium        |

### Results ###

| implementation | environment         | mean    | ...
| -------------- | ------------------- | ------- | ...
| matlab-loop    | matlab-2015b-jit    | ...
| matlab-loop    | matlab-vm-2013a-jit |
| matlab-loop    | matlab-2016a-jit    |
| matlab-loop    | matlab-vm-2014b-jit |
| matlab-loop    | octave-4.0          |
| matlab-vector  | matlab-2015b-jit    |
| matlab-vector  | matlab-vm-2013a-jit |
| matlab-vector  | matlab-2016a-jit    |
| matlab-vector  | matlab-vm-2014b-jit |
| matlab-vector  | octave-4.0          |
```

If something goes wrong, send an email to the TA at erick.lavoie@mail.mcgill.ca for help.

Then modify the template implementation to use your benchmark vector code by changing the runner.m file under `./benchmarks/template/implementations/matlab-vector`. (By convention, the vector-specific code should go in the kernel.m file. However, you may put all your code in the runner.m file if it is simpler for you. If you rely on functions in other files, you may copy them in the runner.m file under the runner function.)

Do the same for the loop version by modifying the same files under `./benchmarks/template/implementations/matlab-loop`.

Make sure they execute without errors on the latest version of MATLAB by doing:

```
wu run matlab-2016a-jit -v
```

If you used the verify function to compute a checksum, make sure both the vector and loop versions get the same checksum as output. If you run into troubles with the tools, this handbook should help you understand better how the tools work and how to use them: `https://github.com/Sable/wu-wei-handbook`.

Once both versions run correctly, compare the performance of the two versions on all versions of MATLAB using 5 iterations by doing:

```
wu run --clean # Erase the previous performance results
wu run -n 5 -v
```

Then generate a report with:

```
wu report
```

Compare the results you obtained on the different versions of MATLAB and Octave. What did you find? If you got problems using the tools, what happened? How convenient were the tools compared to the manual approach? What made them difficult to learn or use?

Question 4: *Your research interests*

Briefly describe your research interests and how COMP 621 fits with those interests.