Benchmarking Memory and Computational Efficiency of Various Languages

COS 284, Regan Koopmans

July 24, 2016

Testing Methods

In order to test the time taken when running a program, I used the '`perf`' bash command, which can be used for a variety of performance testing applications. This was used over the `time` command, which proved to be too inaccurate for valuable results. In order to record the averages and control the test sequences, I constructed the following short bash script:

```
#!/bin/bash
for i in {1..100}
do
  echo $(perf stat -r 1000 ./HelloWorld 2>&1 >/dev/null
  | tail -n 2 | sed 's/ \+//' | sed 's/ /,/'
  | sed 's/[^0-9.]//g')
done
```

This was run on each program, dependent on its specific running requirements (such as invoking Java or Lisp). The results were then sorted and stored in a text file for later comparison. An example of the output can be seen below (results are recorded in seconds) :

```
0.0007464720
0.0007465560
0.0007470370
0.0007487890
0.0007489340
...
```

Results

Memory

Language	*Space Occupied on Disk*
Assembly	4 KB
C++	12 KB
COBOL	16 KB
Fortran	12 KB
Lisp	4 KB (clisp binary is 9.5 MB)
Java	4 KB (JVM is approx. 150 MB)

Computation Time

Language	*Best Average Run Time*
Assembly	0.1489200 ms
C++	0.8101980 ms
COBOL	0.7464720 ms
Fortran	0.5536200 ms
Lisp	6.9800960 ms
Java	48.8928690 ms

Conclusion

It is obvious that the Assembly implementation is the most efficient in both excecution time and binary size, which concurs with expectation. The compiled languages all complete within less than a millisecond, and the interpreted languages run dramatically slower (almost 50 times slower in Java's case). This is however only a limited test, and languages may be faster or slower than one another in certain environments and tasks.