Jesse Huss
001209444
CPSC 3740
Assignment 1

CPSC 3740 Assignment 1 – Report

In Chapter 1 of this course we covered a topic regarding features v/s evaluation criteria for different programming languages. This report will compare and contrast these concepts against RML (Register Machine Language) and programs created in this assignment.

The features we covered in Chapter 1 include: Simplicity, Orthogonality, Data Types, Syntax, Abstraction, Expressive Power, Type Checking, Exception Handling, and Restricted Aliasing. From working with RML for such a short period of time I can tell that it has the potential to be quite efficient, however it is a very feature poor language as there are only three instructions: HALT, INC, DEB.

HALT is the instruction that is used to terminate the machine.

INC r j, increments the contents of register r, and moves to instruction number j.

DEB r i j, If contents of register r is > 0 then decrement the register and move to instruction i, else move to instruction j.

With these instructions it is more than possible to do anything and everything we need to do in the world of programming, however we would be sacrificing readability and writability if we chose to write complex programs in RML.

Regarding features that would be present in RML, I would argue that the only thing it has is simple syntax. The syntax used in RML is extremely basic and easy to use seeing as there are only three instructions. This benefits users who have a strong understanding of programming concepts and register management because with only three instructions it would be fairly easy for an experienced computer scientist to sit down and create a basic program in RML without much practice.

As for the absence of features in RML, basically everything but syntax is missing from this language. We could argue that RML has simplicity for very basic programs, however in the grand scheme of things this is not a practical argument. We don't have the luxury of data types, or abstraction, type checking, or exception handling. RML is incredibly basic and could definitely be used to build out the features we talked about in Chapter 1 but would be quite the task.

We also covered: Readability, Writability, and Maintainability in Chapter 1. As I mentioned previously, for very basic programs such as multiplication, division, addition and subtraction, we have perfectly fine readability, writability, and maintainability. The reason we have these things on a basic level is because at most we should be looking at about 10-15 lines of instructions for each program. This means that it should be fairly easy for someone who is familiar with RML to be able to read the program and understand it. This also means it should be fairly straight forward to write since we have a max of 15 or so lines. And then finally, a program this basic will most likely not need to be maintained. All things considered, readability, writability, and maintainability go out the window the second you start building more complex programs.

Overall I think that RML has its uses and the potential to be very powerful, however with all the modern languages around it does not seem overly practical.