Compilation of a CSP-like language

OCHAN JULIUS JOSHUA

April 18, 2017

1 Introduction

Csp-like language is the communicating sequential of processes, its a process for describing concurrent processes and their interaction with other compiler languages. The parallel programming language Occam is essentially an implementable sublanguage of CSP.

2 Background to the Problem

Most of the programming languages in the computing world have a very heavy syntax and wide grammar, lack parallelism and more so not portable. These programs run on a few clusters, servers and embedded systems.

More so, they lack an extension to Occam which permits recursion and still lack virtual machines which are fully optimized for displaying a simulation, and translating the virtual machine code into native code for a real machine.

Csp-like language is used for describing some interesting and complex system that are ready implemented directly and others that we cannot yet implement directly, during implementation of such interesting and complex systems we can use tools such as JCsp, c++csp,CTJ and Kroc/Occam-pi

The aims of the project is to implement an extension to Occam which permits recursion; more ambitious projects might implement a distributed implementation with several communicating copies of the virtual machine. Other possibilities are to produce separate virtual machines, optimized for displaying a simulation, or for efficiency of implementation, or translating the virtual machine code into native code for a real machine.

3 Problem Statement

The problem this project will address is to produce a small portable implementation of a subset of Occam, to implement a virtual machine based on the inmos transputer, and a compiler which will target the language.

The proposed system will implement an extension to Occam which permits recursion and separate virtual machines which are fully optimized for displaying a simulation.

The exiting compiler could not permit recursion and its difficulty to use and maintain, therefore because of that we came up with a project to implement an extension to Occam-compiler that would support and permit recursion that would produce a separate virtual machines which are optimized for displaying simulation, for efficiency of implementation and also translating the virtual machine code into native code for a real machine.

4 Objectives

4.1 Main Objective

The aim of this project is to produce a small portable implementation of a subset of Occam; the proposed technique is to implement a virtual machine based on the inmos transputer, and a compiler which targets that language.

To implement an extension in Occam-compiler that would permit or support recursion and also to produce a separate virtual machine which are optimized for displaying simulation ,efficiency of implementation and translating the virtual machine code into native code for real machines.

4.2 Other Objectives

- To gather and analyze requirements that will be used in the design and development of a virtual machine based on the in mos Transputer, and a compiler which targets that language.
- To implement the designed system.
- To test and validate the designed system.

5 The scope

The design and its implementation will only be on the Occam-compiler by creating an extension on it that will permit and support recursion and thus producing separable virtual machine that are optimized for displaying simulation, efficiency of implementation and translating the virtual machine code into native code for real machine.

6 Methodology

In order to implement a virtual machine based on the in mos Transputer, and a compiler which targets that language, we need to collect user requirements using various tools and techniques to achieve our objectives for example requirements analysis, interviews and literature review of existing systems.

We will analyze our data collected using various methods such as process modeling specification which will include data flow diagrams and context diagrams. We will also use process specifica-

tion methods during analysis of requirements for example action diagrams.

We will use c-programming language to create an extension in Occam-compiler that will permit recursion.

7 Outcomes/ significance

- Occam language support parallelism then other compiler languages.
- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources since its implemented with separable virtual machines.
- System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an exact duplicate to the underlying machine since Occam-compiler enables implementation of separate virtual machines.

8 References

- Compiling csp- slide by Fred Barnes, September 2006.
- http://tack.sourceforge.net/olddocs/occam.pdf.