OCHAN JULIUS JOSHUA

COMPILATION OF CSP-LIKE LANGUAGES.

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 of the csp-like languages.

The csp is a communicating sequential of processes, it a process algebra for describing concurrent processes and their interaction with each other. Csp itself is used primarily for formal modelling and it uses tools such as FBR and ProBE.

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 Occam compiler is the new implemented compiler that was designed to replace Kroc complier, Kroc compiler was becoming increasingly difficult to use and maintain basing on its fairly old code base and most it was designed to run on 2MB of memory, therefore the Occam compiler was implemented to carter for the dynamic system.

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 exiting compiler could not permit recursion and its difficulty to use and maintain, therefore because that we came up with a project

to implement an extension to Occam-complier 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 The main objective

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.

4.1 Specific objectives.

Review the existing literature on the compilation of the csp-liked languages and how they have been operating. Design a new extension on the Occam-compiler that would intercorporate the new and the old Occam-compiler in order to test recursion and optimization of the system using separate virtual machine. Implement the new system in order to intercorporate the new features. Test the system to see whether its meets its requirements.

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 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.

7 Methodology.

This section comprises of research/project design which describes the tools, instruments, approaches, processes and techniques, major algorithms and data structures to be employed in the research study, data collection, analysis, synthesis, design, logical flow, implementation, testing, and validation

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

8 References

Compiling csp- slide by Fred Barnes, September 2006