Background

A data dependency in computer science is a situation in which a program statement (instruction) refers to the data of a preceding statement [1].

In compiler theory, the technique used to discover data dependencies among statements (or instructions) is called dependence analysis [1].

Dependence analysis determines whether it is safe to reorder or parallelize statements.

As the number of software grows and programs become huge and complex, dependence analysis plays important role for their optimization and ensuring that the program does what it is supposed to do.

Motivation

…

Objectives

The objectives of this project is to apply static techniques to extract data dependencies from programming code. The second objective is to convert results to DiscoPoP format. Thus, we can compare these techniques with two other data dependencies extracting techniques, which the third objective.

First of two other techniques is dynamic technique DiscoPoP and second is static technique PLUTO. The results of applying these two techniques are readily available.

Proposed solution including software design.

A suitable tool for our purpose is LLVM [2]. It is a powerful instrument that is used to construct, optimize and produce intermediate and/or binary machine code.

For our aim we can use it to extract data dependencies. The techniques we propose to apply are, namely, Use-Def Chain analysis, Alias analysis and Inter-procedural analysis.

A Use-Definition Chain [3] (UD Chain) is a data structure that consists of a use, U, of a variable, and all the definitions, D, of that variable that can reach that use without any other intervening definitions. A definition can have many forms, but is generally taken to mean the assignment of some value to a variable (which is different from the use of the term that refers to the language construct involving a data type and allocating storage). Making the use-define chains is a step in liveness analysis, so that logical representations of all the variables can be identified and tracked through the code.

Alias analysis [4] is a technique in compiler theory, used to determine if a storage location may be accessed in more than one way. Two pointers are said to be aliased if they point to the same location. In general, alias analysis determines whether or not separate memory references point to the same area of memory. This allows the compiler to determine what variables in the program will be affected by a statement.

Inter-procedural analysis [5] uses calling relationships among procedures to analyze call-return and parameter passing mechanisms, local variable of the function and function recursion.

In this project we will use LLVM to execute these techniques on benchmark suits, which are exactly Polybench, NPB and BOTS.

As the next step we will apply available from previous research python-script to convert our results to DiscoPoP format.

Since dynamic data dependencies extracting technique DiscoPoP and static data dependencies extracting technique PLUTO were already executed on mentioned above benchmark suits and their results are represented in DiscoPoP format, we will be able to compare them with the results of our experiment.

Expected results

…

…

Work plan

…

References

[1] Data dependency  
<https://en.wikipedia.org/wiki/Data_dependency>

[2] LLVM overview  
<http://llvm.org/docs/>

[3] Use-Def Chain   
<https://en.wikipedia.org/wiki/Use-define_chain>

[4] Alias analysis  
<https://en.wikipedia.org/wiki/Alias_analysis>

[5] Inter-procedural optimization  
<https://en.wikipedia.org/wiki/Interprocedural_optimization>