Project 3
CS 5130
Robbie Watling

The goal of this project was to detect spatial reuse and report the enclosing loop and the associated stride. This was implemented by extending DependenceAnalysis.cpp in LLVM. First, I took advantage of the dump example dependence function which examines all pairs of instructions and calls the depends function. In the depends function, the pair of instructions is first determined if they are eligible memory instructions for dependence analysis which is also necessary for our purposes. Then GEP operators are used to get subscript information. From there, I consider only the least significant array subscript for the stride. The Scalar Evolution object of the subscript (SCEV) is obtained and converted into a Scalar Evolution object of “AddRecExpr” where the coefficient corresponds to the stride. The stride is then unwrapped from the AddRecExpr to a constant SCEV then to a Value object. This Value is then converted into a usable integer. Thus the stride is now in a mutable integer form. I also obtain the array element type (integer or double) from the GEP operator using getResultElementType() which allows the determination of the result element size. The stride obtained is divided by the result element size (in words). This results in the stride being divided by 1 or 2 with respect to the integer and double element types. The cache line size is divided by the number of elements it can hold which is either 16 integers or 8 doubles. The cache line size and word size are macros set at the top of the program. The level is determined by the common nesting loops already in the depends function. The instruction itself was passed into the depends function. Then if the stride is smaller than the number of elements in the cache line then the instruction, level, and stride are reported. This effectively produces the output described in Project 3 to the best of my knowledge.

Future improvements for the project would be using relevant LLVM structures. First would be either detecting the cache line size as an option or obtaining it within the program. I did find one example of finding the loop cost which includes obtaining the cache line size in LoopCacheAnalysis.cpp which may be relevant for future projects. The second improvement would be utilizing some of the arithmetic functions for SCEV instead of unwrapping the SCEV’s to calculate the stride. This would likely be a more flexible implementation, but this project’s implementation is sufficient.