INSTRUCTION LEVEL LOOP DE-OPTIMIZATION

LOOP REROLLING AND SOFTWARE DE-PIPELINING

Erh-Wen Hu and Bogong Su, Dept. of Computer Science, William Paterson University Wayne NJ USA
Jian Wang, Mobile Broadband Software Design, Ericsson, Ottawa, ON, Canada

DECOMPILATION
Important issue in software reverse engineering
- Porting legacy software
- Re-optimizing assembly code
- Detecting bugs
- Detecting malware
- Decompilation for modern high performance processors needs instruction level de-compilation

COMPILATION/DECOMPILATION for Instruction Level Parallel Assembly Code
Source Code
Sequential Assembly Code
Instruction level optimization
Instruction Level Partial Assembly Code
COMPILATION
Source Code
Sequential Assembly Code
Instruction level optimization
Instruction Level Parallel Assembly Code
DECOMPILATION

SOFTWARE PIPELINING
A popular loop optimization technique to exploit instructions in level parallelism

LOOP UNROLLING
Another popular loop optimization technique
- Reduce loop overhead
- Loop unrolling provides more opportunities to increase instruction level parallelism

ORIGINAL LOOP

Unrolled Loop

Unrolled Loop with Data Dependent Looping

DATA DEPENDENCY

OPERATIONraphs

COST REDUCTION

Unrolled Loop with Data Dependent Looping

Software Pipelined result after unrolling

CATEGORY OF SUBDDGS
- SubDDGs are all isomorphic: use the same index register and have the same operations on their corresponding nodes.
- Contains some instruction with instructions using an additional index register for accessing the same array due to initial of instruction format when unrolling too many times.
- Our approach can be a useful technique for the difficult tasks of loop rerolling and software de-pipelining.

METHODOLOGIES
1. Perform software de-pipelining first, then perform rerolling if the loop has been software pipelined after unrolling
2. Build data dependence graphs of subDDG based on the analysis of innermost loops in assembly code. The process begins from the first instructions to reduce the search space
3. Find clusters of potential unrolled loops including all isomorphic subDDGs and isomorphic subDDGs.
4. Convert all isomorphic subDDGs to isomorphic subDDGs using symbolic calculation, instruction replacing, de-peephole optimization, and other techniques
5. Use single loop to represent all isomorphic subDDGs, which is the rerolled loop

FLOW CHART OF LOOP DE-OPTIMIZATION TECHNIQUE

SUMMARY
- Our instruction level loop de-optimization algorithms involve software de-pipelining and loop rerolling
- Instruction level loop de-optimization can be very complicated, particularly when the assembly code after loop unrolling is combined with software de-pipelining and other optimizations
- Although different compilers may generate different optimized assembly code, our approach can be a useful technique for the difficult task of loop rerolling and software de-pipelining necessary to decompile loops at instruction level
- In this paper, we consider only loop-independent dependency and plan to extend it to handle loop carried dependency in the future