A New Technique for Copy Propagation And Dead Code Elimination Using Hash Based Value Numbering

Dr. K.V.N.Sunitha,
Professor, & Head,
CSE Dept.
G.Narayanamma Inst.of Tech. & Science,
Shaikpet, Hyderabad, India.
sunitha_kvn@rediffmail.com

Dr V. Vijaya Kumar
Professor, & Head,
CSE Dept.
R.G.M.I.T,
Nandyal, Kurnool, India
vijayvakula@yahoo.com

Abstract

A novel technique for copy propagation and dead code elimination using hash based value numbering is discussed in this paper. This technique uses less data structures and is more efficient than earlier methods. Copy propagation generally creates dead code that can then be eliminated. Eliminating dead code improves efficiency of the program by avoiding the execution of unnecessary statements at run time. Hence in this paper, the two techniques are combined using new algorithm called hash based value numbering. This algorithm is optimal, uses less data structures and is more efficient than earlier methods. The results clearly indicate aggressive optimization.

I. INTRODUCTION

Copy Propagation is a transformation that, given an assignment y ← x for some variables x and y, replaces later uses of y with uses of x as long as intervening instructions have not changed the value of either y or x. A variable is dead if it is not used on any path from the location in the code where it is defined to the exit point of the routine in question. An instruction is dead if it computes only values that are not used on any executable path leading from the instruction. If a dead variable's value is assigned to a local variable, the variable and the instruction that assigns to it are dead if the variable is not used on any executable path to the procedure's exit [1, 2]. Strength reduction is an example of an optimization that produces dead code. The algorithm [3] uses many data structure for identifying copy statements, to propagate copy statements, for identifying dead code and to eliminate dead code. The present paper discusses a technique that uses hash based value numbering for optimization using copy propagation and dead code elimination [4, 5].

II. METHODOLOGY

To perform global copy propagation, a data-flow analysis is performed to determine which copy assignments reach uses of their left-hand variables unimpaired, i.e., without having either variable redefined in between.

The present method uses hash table HT[B] for value numbering, CP_Table[i] for storing i^{th} copy statement u = v in basic block b as a triple {u,v,b}, Mark[i] for marking an i^{th} statement as 0 or 1 depending on whether it is useful or not.

III. ALGORITHM OF A NEW TECHNIQUE FOR COPY PROPAGATION AND DEAD CODE ELIMINATION USING HASH BASED VALUE NUMBERING

The algorithm consists of two passes. In pass1, copy propagation and value numbering is performed. The required information to identify dead code is also collected in hash table and Mark array. In pass2, dead code elimination is performed by using Mark array which has been created in pass1.

Pass 1:

1. Initialize the value number of all variables as 0 in hash table.
2. For each arithmetic statement, since it involves redefinition or use of a variable, all its references are replaced by its value numbers as follows.
3. First assign value number to variables on right hand side of expression based on values in hash table.
4. Assign a new value number to variable on the left hand side of expression as current value number in hash table + 1.
5. Update the value numbers in hash table.
6. If the statement is a copy statement then store it in CP_Table [ ] along with block number.
7. If variable in use is found in CP_Table, replace it with its assignment.
8. Repeat steps 2 to 7 for all statements within a block i.
9. Repeat steps 2 to 8 for all basic blocks.
10. On reaching the last statement, perform backward traversal from the current statement, update Mark [ ] array to indicate useful statements which define values for variables used in current statement.
11. If statement is a conditional or return or Input or output statement or declaration statement then update in Mark [ ] array that it as useful statement.
12. Repeat 10 and 11 for all marked statements.

1-4244-0716-8/06/$20.00 ©2006 IEEE.

Pass 2:

I. If the statement is a marked statement then replace the value numbers by the true variables.
II. If statement is unmarked then delete it as it is identified to be dead code.
III. Repeat above steps for all statements.

The above algorithm is applied on a code segment shown in Fig. 1. The results after pass1 and pass2 are shown in Fig. .2.

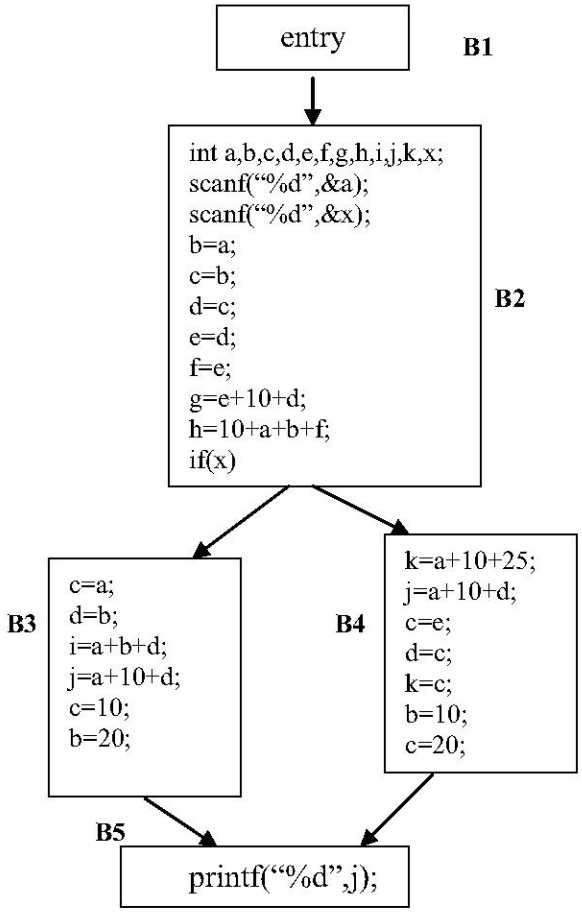

Figure. 1: Code Fragment for testing the technique.

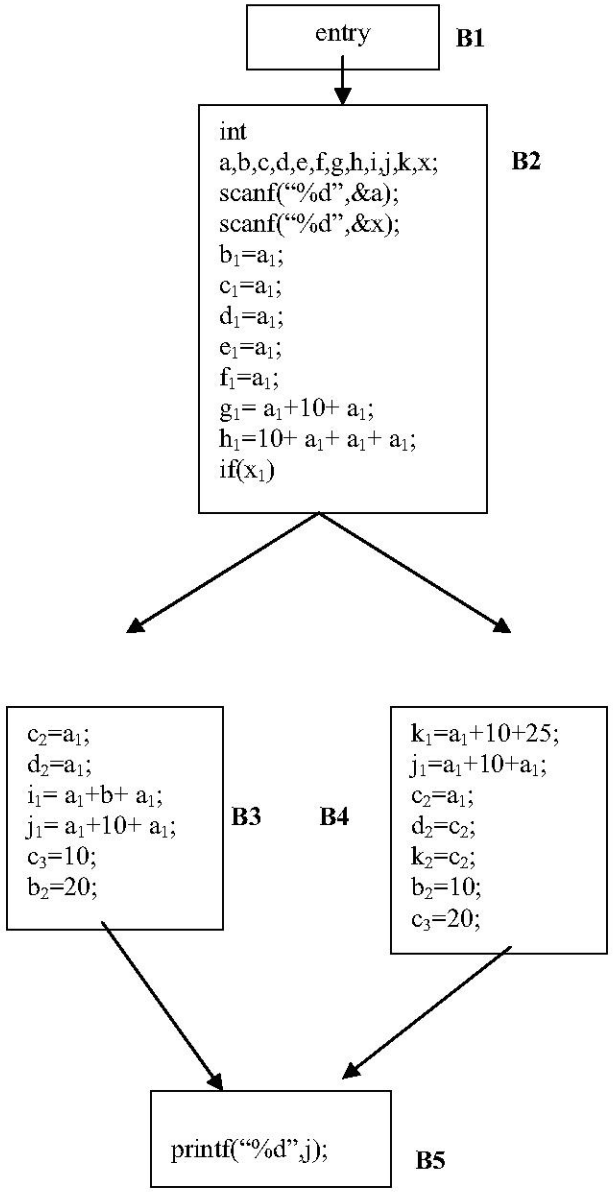

Figure.2: Code Fragment for after applying pass1 of the algorithm.

IV. EXPERIMENTAL RESULTS

This technique is tested on different programs using IBM Rational Quantify tool on Pentium IV 2.5 GHz Processor The results are evaluated and compared not only in terms of execution times but also in terms of machine cycles and number of assembly instructions. The present technique 'A New Technique for copy propagation and dead code elimination using hash based value numbering' is applied on code segment shown in Fig.1.

The program contains 27 statements. On this code fragment when only copy propagation is applied, there was no reduction in number of instructions but copy propagation creates candidates for dead code. When dead code elimination alone was applied on code segment shown in fig 1, it reduced the number of instructions to 14. When the proposed technique which is combined approach of copy propagation and dead code elimination using hash based value numbering is applied it resulted in 10 instructions. The results of optimization in terms of number of assembly instructions, number of machine cycles and execution times are shown as bar charts in Fig. 3 Fig. 4 and Fig. 5 respectively. In the graphs, **Unopt** represents Un optimized code , **Copy Prop** indicates applying Copy propagation, **Dead Code** indicates applying Dead code elimination and **Combined** indicates applying Copy propagation and dead code elimination using hash based value numbering.

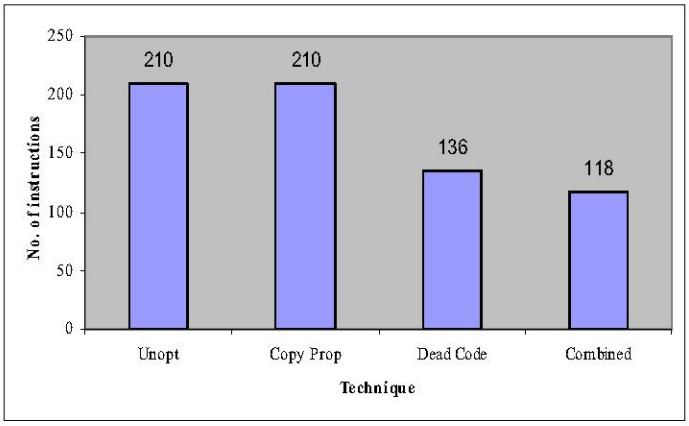

Figure.3: optimization in terms of number of assembly instructions.

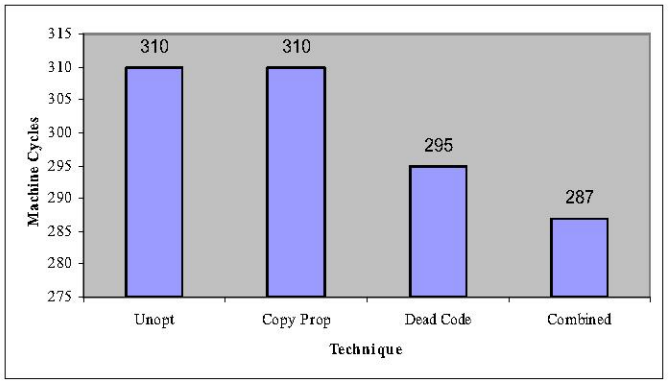

Figure.4: optimization in terms of number of Machine cycles.

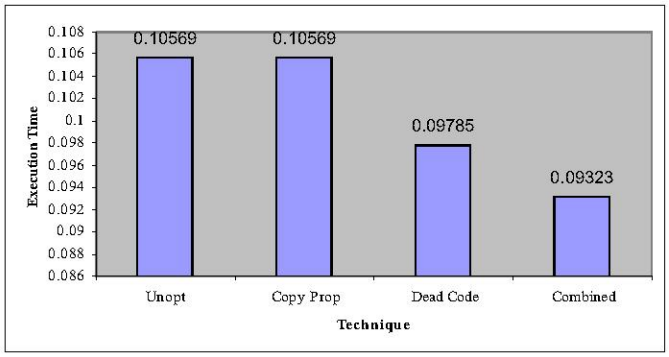

Figure.5: optimization in terms of number of execution time in microseconds.

V. CONCLUSION

To achieve the goal of combining copy propagation and dead code elimination an innovative method called hash based value numbering has been introduced and applied successfully. This algorithm is optimal, uses less data structures and is more efficient than earlier methods. The results clearly indicate aggressive optimization. The other advantage of the present algorithm is that it is performed in two passes which reduces memory. Thus it bridges gap of high memory requirement of hash based schemes.

VI REFERENCES

[1]. J. Knoop, Oliver Ruthing and Bernhard Steffen, Partial dead code elimination. ACM SIGPLAN notices, 1994.

[2]. L. T. Kou, On live-dead analysis for global data flow problems. Journal of the ACM, 24(3):473 - 483, July 1977.

[3]. Steven Muchnick . S. , Advanced Compiler Design and Implementation. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[4]. K.V.N. Sunitha ..and V. Vijaya Kumar., CARD –an optimizing tool, published in the International Journal-WESAS transactions on computers, issue 11, vol 4, Nov 2005, ISSN 1109-2750, P.No 1627-1732.

[5]. K.V.N .Sunitha .and V.Vijay Kumar., Redundancy Elimination using hash based value numbering. National Journal-Institution of Engineers, 2006.