Register Allocation

66.648 Compiler Design Lecture (04/06/97)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Optimized Register Allocation
- Examples
- Administration

Motivation

Improve utilization of registers compared to a simple register allocation:

1. Pay careful attention to assigning registers to variables that are more profitable.
2. Use the same register for multiple variables when legal to do so.

Notation

Live range of variable V = set of basic blocks that V is live in.

Register allocation and register assignment mean different.

Register Allocation -- identifying a subset of variables that should be allocated to registers (at each program point)

Register Assignment -- assigning a physical register to each variable selected for register allocation.

Example

```
Public class test {
public static void main(String argsv[]) {
int I,j,k,counter;
 I = 78;
j = 89;
k = j;
for (counter=1;counter<I;counter++) k=k+j;
}
```

Example

```
I = 78
j = 89
k = j
counter=1
loop: if (counter>I) goto  out
k=k+j
counter=counter+1
goto Loop
out:
```

Desired Register Allocation

Assume that there are only three non-reserved register variables available for allocation. A desired register allocation for the above example is as follows:

Variable	Register
k	
I	
j	
counter	

Register-Interference Graph

The register-interference graph is an undirected graph in which

- A node is a variable/temporary that is a candidate for register allocation (exceptions are volatile and aliased variables)
- An edge connects nodes V1 and V2 if there is some point in the program where variables V1 and V2 are simultaneously live i.e. they interfere.

Example

Register Allocation by Graph Coloring

Defn: A graph is said to be k-colored if each node has been assigned to one of k colors in such a way that no two adjacent nodes have the same color.

Idea: Let k be the number of non-reserved registers available. A k-coloring of the interference graph can be directly mapped to a legal register allocation by mapping each color to a distinct register.

Register Profitability

By keeping a variable in a register, we save load and store instructions. However, by keeping a variable in a register, we incur a cost in storing the value of the register in memory..

Optimal Order

If our expression is a tree, the following labeling algorithm will give us an optimal order.

Label(v) = 0 if v is a leaf node.

Label(w) = max {label(t), t are the children of w}

if not all labels are equal.

= label(t) + 1 if all are equal.

Examples

Dags

The above algorithm wll not work in the case of dags (directed acyclic graphs). One can employ a dynamic programming algorithm to handle this.

Please read chapter .

Loops in CFG

Basic Idea: Identify regions/subgraphs that correspond to program loops. Loops can be constructed using for while or goto statements.

Motivation: Programs spend more time in loops, so there is a larger payoff from optimizations that exploit loop structure.

E.g., loop invariant code motion.

Profilers are important tools (compiler options) to measure which codes are being executed more.

Dominators

Node (basic block) D in a CFG dominated node N, if every path from the source node to N passes

through node D. (critical term is “every”).

We also say that node D dominates N.

dom(N) = set of nodes that dominate node N.

By definition N is in dom(N).

Dominators and Loops

One can use dominators to identify loops.

In the control flow graph an edge a-->b is a back edge, if b dominates a.

Comments and Feedback

Project 3 is out. Please start working. PLEASE do not wait for the due date to come.

We are in chapters 9 and 10. Please read that chapter and read the relevant portion of Java. Please keep studying this material and work exercises.