Compiler Design Assignment

Name- Adhyyan Tripathi

Sec/Dept. – C/CSE

Reg. no. – 201700403

Peephole Optimization

Peephole optimization is a type of Code Optimization performed on a small part of the code. It is performed on the very small set of instructions in a segment of code.

It basically works on the theory of replacement in which a part of code is replaced by shorter and faster code without change in output.

Peephole is the machine dependent optimization.

Objectives of Peephole Optimization:

The objective of peephole optimization is:

- To improve performance
- To reduce memory footprint
- To reduce code size

Peephole Optimization Techniques:

1. Redundant Instruction Elimination:

At compilation level, the compiler searches for instructions redundant in nature. Multiple loading and storing of instructions may carry the same meaning even if some of them are removed. For example:

MOV x, R0

MOV R0, R1

We can delete the first instruction and re-write the sentence as:

MOV x, R1

2. Removal of unreachable code:

Inaccessible code is a piece of the program code that is never gotten to considering programming builds. Software engineers may have accidently composed a bit of code that can never be reached.

3. Flow of control optimization:

There are instances in a code where the program control jumps back and forth without performing any significant task. These jumps can be removed.

4. Algebraic Simplification:

There are occasions where algebraic expressions can be made simple. For example, the expression a = a + 0 can be replaced by itself and the expression a = a + 1 can simply be replaced by INC a.

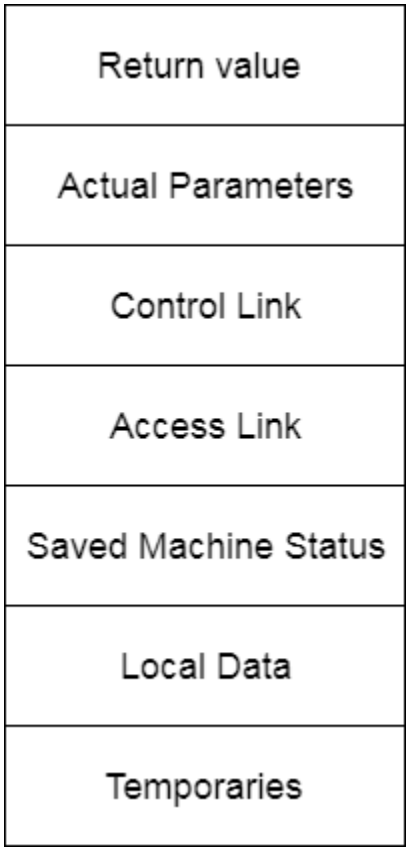
5. Machine Idioms:

The objective machine can send more advanced guidelines, which can have the ability to perform explicit activities much productively. On the off chance that the objective code can oblige those guidelines straightforwardly, that will not just improve the nature of code, yet additionally yield more productive outcomes.

Activation Record

- Control stack is a run time stack which is used to keep track of the live procedure activations i.e. it is used to find out the procedures whose execution have not been completed.
- When it is called (activation begins) then the procedure name will push on to the stack and when it returns (activation ends) then it will popped.
- Activation record is used to manage the information needed by a single execution of a procedure.
- An activation record is pushed into the stack when a procedure is called and it is popped when the control returns to the caller function.

The diagram below shows the contents of activation records:



Return Value: It is used by calling procedure to return a value to calling procedure.

Actual Parameter: It is used by calling procedures to supply parameters to the called procedures.

Control Link: It points to activation record of the caller.

Access Link: It is used to refer to non-local data held in other activation records.

Saved Machine Status: It holds the information about status of machine before the procedure is called.

Local Data: It holds the data that is local to the execution of the procedure.

Temporaries: It stores the value that arises in the evaluation of an expression.

DAG representation for basic blocks

A DAG for basic block is a directed acyclic graph with the following labels on nodes:

1. The leaves of graph are labeled by unique identifier and that identifier can be variable names or constants.
2. Interior nodes of the graph is labeled by an operator symbol.
3. Nodes are also given a sequence of identifiers for labels to store the computed value.

- DAGs are a type of data structure. It is used to implement transformations on basic blocks.
- DAG provides a good way to determine the common sub-expression.
- It gives a picture representation of how the value computed by the statement is used in subsequent statements.

Algorithm for construction of DAG

Input: It contains a basic block

Output: It contains the following information:

- Each node contains a label. For leaves, the label is an identifier.
- Each node contains a list of attached identifiers to hold the computed values.

Step 1:

If y operand is undefined then create node(y).

If z operand is undefined then for case(i) create node(z).

Step 2:

For case(i), create node(OP) whose right child is node(z) and left child is node(y).

For case(ii), check whether there is node(OP) with one child node(y).

For case(iii), node n will be node(y).