ASSIGNMENT NO: 6.

Title : Code generation using Register Allocation.

Problem Statement :

Write a program for Register Allocation algorithm that translates the given code into one with a fixed number of registers.

Learning Objective:

- To understand code generation phase of compilation
- To understand register allocation algorithm.

s/w package: Linux OS, 1 GB RAM, keyboard, Mouse. 16 GB HDD,

Theory :

Code generation:

Goal: Takes intermediate code representation of source program (optimized 3 address code statements) & produce equivalent target program as output.

Requirement of code generator:

- Correct target code.
- High quality code.
- Effective use of resources of target machines.
- Quick.

Simple Code generator:

- Simple code generation involves generating code for each three address statements which are divided into different basic blocks. For that it makes use of register to store operands and leaves result of computation in the register as long as possible.

At the end of a basic block the result of the computation are stored only if the register is needed for another computation or just before procedure call, jump or labelled statements because after leaving the basic block flow enters into another different basic block.

When Generating code by above mentioned strategy we need:

1. Register Descriptor.
2. Address Descriptor.

Register Descriptor:

- To keep track of what is currently in each register, register descriptor is maintained.

- Register descriptor is pointer to a list that contains information about what is currently in each of the register. Initially all the registers are empty.

Address Descriptor:

- To keep track of the locations where current value of the name can be found at run time, address descripto

is maintained. The location can be register, stack location or memory address.

- This particular info is stored in the symbol table.

Generation Algorithm:

- Input to the code generation algorithm is a sequence of 3-address statement divided into blocks.

For each 3-address statement of the form a=b op c in the basic block it performs follⁿ steps:

1. Call getreg() to obtain the location L in which computation a=b op c is performed. For that we need to pass three-address statement a=b op c as parameter to the getreg() function. Typically location L can be register or it can be memory location.

2. Obtain the current location of the operand b from its address descriptor. If the value of y is in memory location & also in register then prefer register rather than memory location. If the value of b is not in location L then generate the instruction MOV b, L to store copy of b in L.

3. The instruction OP x L is generated & the address descriptor of x is updated to indicate that x is now available in L. If L is register then update

its descriptor to indicate that it will contain the run time value of x.

4. If the current value of b and c are in register then we say that b and c are not further used. The means value of b and c are not live at the end of the basic block. So alter the register descriptor to indicate that after the execution of 3 address statement a:=b op c, those register will no longer contain b and/or c.

The function getreg()

- To perform computation specified by each of the 3 address statement, we need a location. For that function getreg()
- When a function getreg() is called, it returns a location for the computation performed by a 3 address statement. for eg: if a:=b op c is to be performed then getreg() returns a location L where the computation b op c should be performed; and if possible it returns a register.

Conclusion:

Hence, I have implemented code generation using Register Allocation.