Experiment No.7

Title: - Code Generation Algorithm

Aim:- Implement Code generation algorithm.

Theory:-

Next-Use Information

- If a register contains a value for a name that is no longer needed,we should re-use that register for another name
(rather than using a memory location)
- So it is useful to determine whether/when a name is used again in a block
- Definition: Given statements *i, j,* and variable x,
  - If *i* assigns a value to x, and
  - *j* has x as an operand, and
  - No intervening statement assigns a value to x,
  - Then *j uses* the value of x computed at *i*

Next-Use Information Algorithm

• Assume "liveness" info has already been computed or there is none
• Scan backwards fromend of block. At statement
i: x := y *op* z, do
1. Attatch to *i* the info currently in symbol table about next-use and liveness of x , y and z.
2. In symbol table, set x to "not live" and "no next use"
3.In symbol table, set y and z to "live" and their next uses to *i*

Next-Use Algorithm Example

Example	**Solution**
(1) t := a – b (2) u := a – c (3) v := t + u (4) d := v + u	Symbol Live Next Use d No None v Yes 4 u Yes 4
(1) t := a – b (2) u := a – c (3) v := t + u # u,v: live; next-use=4 (4) d := v + u	Symbol Live Next Use d No None v No None t Yes 3 u Yes 3

A Simple Code Generator

• *Input*: Three -address statements in a single basic bock
• *Strategy*: Remember which operands are in registers, taking advantage of that to avoid memory access
• *Assumptions*:
– Register-only ops cost half as much as memory ops
– Values can stay in register until

• register is needed for another computation, or
• right before a proc call, jump, or labeled statement –*i.e.*, must store everything in memory at end of block

Register and Address Descriptors

• Code generator uses descriptors (data structures) to track status of registers and variables:
– *Register descriptor* :what vars are in each register
– *Address descriptor*: where a variable lives (register,memory)

Choosing Registers

1. Look in register descriptor *i*
2. If it's empty, use *i*
3. Otherwise, it contains the value of a variable *v*. If *v* has no next use (inside block) and isn't live (needed after block), thenuse *i.*
4. Otherwise, keep looking for empty registers
5. If there are none
    i. "Spill" (store) the contents *w* of some register *j* to memory
    ii. Record *w*'s new location in the address and registerdescriptors
    iii.Use *j*.

Example

• Source code: d := (a – b) + (a – c) + (a – c)
• Three-address code:

t := a – b
u := a – c
v := t + u
d := v + u

– Assume d live at end of block
– Algorithm produces the following sequence

Statements	Code Generated	Register Descriptor	Address Descriptor

t := a – b	lw $t0,*a*
	lw $t1,*b*
	sub $t1,$t0,$t1
		$t0 contains a
		$t1 contans *b*
		$t1 contains t
			a in $t0
			b in $t1
			t in $t1

u := a – c	lw $t2,*c* sub $t0,$t0,$t2	$t2 c *c* $t0 c *u*	*c* i $t2 *u* i $t0
v := t + u	add $t1,$t1,$t0	$t1 c *v*	*v* i $t1
d := v + u	add $t0,$t1,$t0	$t0 c *d*	*d* i $t0
t := a – b u := a – c v := t + u d := v + u	sw $t0,*d*		*d* i $t0 and memory