编译原理实验报告-Lab4-附表

IR Type	IR Code	MIP32 Code	Comment

ASSIGN	x := y	lw $t1, -offset_1($fp)
		move $t0, $t1
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

	x := #k	li $t0, k sw $t0, -offset_1($fp)	offset_1 = findOffset(x)
	x := *y	addi $t0, $fp, -offset_1 sw $t0, -offset_2($fp)	offset_1 = findOffset(y) offset_2 = findOffset(x)

	*x := y	lw $t1, -offset_1($fp)
		lw $t0, -offset_2($fp)
		sw $t1, 0($t0)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

	*x := #k	li $t1, k
		lw $t0, -offset_1($fp)
		sw $t1, 0($t0)
			offset_1 = findOffset(x)

	*x = *y	lw $t0, -offset_1($fp)
		lw $t1, -offset_2($fp)
		lw $t1, 0($t1)
		sw $t1, 0($t0)
			offset_1 = findOffset(x)
			offset_2 = findOffset(y)

ADD	x = y + z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		add $t0, $t1, $t2
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

	x = &y + z	addi $t1, $fp, -offset_1
		lw $t2, -offset_2($fp)
		add $t0, $t0, $t2
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

	x = &y + #k	addi $t0, $fp, k sw $t0, -offset_1($fp)	offset_1 = findOffset(x)

	x = y + #k	lw $t1, -offset_1($fp)
		addi $t0, $t1, k
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

	x = #k + z	li $t1, k
		lw $t2, -offset_1($fp)
		add $t0, $t1, $t2
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(z)
			offset_2 = findOffset(x)

	x = #k_1 + #k_2	li $t1, k_1
		addi $t0, $t1, k_2
		sw $t0, -offset_1($fp)
			offset_1 = findOffset(x)

SUB	x = y - z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		sub $t0, $t1, $t2
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

	x = y - #k	lw $t1, -offset_1($fp)
		addi $t0, $t1, -k
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

Table 1 continued from previous page

IR Type	IR Code	MIP32 Code	Comment

	x = #k - z	li $t1, k
		lw $t2, -offset_1($fp)
		sub $t0, $t1, $t2
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(z)
			offset_2 = findOffset(x)

	x = #k_1 - #k_2	li $t1, k_1
		addi $t0, $t1, -k_2
		sw $t0, -offset_1($fp)
			offset_1 = findOffset(x)

MUL	x = y * z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		mul $t0, $t1, $t2
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

	x = y * #k	lw $t1, -offset_1($fp)
		li $t2, k
		mul $t0, $t1, $t2
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

	x = #k * z	li $t1, k
		lw $t2, -offset_1($fp)
		mul $t0, $t1, $t2
		sw $t0, -offset_2($fp)
			offset_1 = findOffset(z)
			offset_2 = findOffset(x)

	x = #k_1 * #k_2	li $t1, k_1
		li $t2, k_2
		mul $t0, $t1, $t2
		sw $t0, -offset_1($fp)
			offset_1 = findOffset(x)

DIV	x = y / z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		div $t1, $t2
		mflo $t0
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

	x = y / #k	lw $t1, -offset_1($fp)
		li $t2, k
		div $t1, $t2
		mflo $t0
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(x)

	x = #k / z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		div $t1, $t2
		mflo $t0
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(x)

	x = #k_1 / #k_2	lw $t1, k_1
		lw $t2, k_2
		div $t1, $t2
		mflo $t0
		sw $t0, -offset_3($fp)
			offset_1 = findOffset(y)
			offset_2 = findOffset(z)
			offset_3 = findOffset(x)

FUNCTION	FUNCTION func	func:
		addi $sp, $sp, -4
		sw $fp, 0($sp)
		move $fp, $sp
		addi $sp, $sp, stackSize
			stackSize = funcStackSpace
			(func->next)为避免函数名与MIP32指令名称冲突除main函数外所有函数添加前缀'func_'.

Table 1 continued from previous page

IR Type	IR Code	MIP32 Code	Comment

IF	IF x == y GOTO z	lw $t1, -offset_1($fp)
		lw $t2, -offset_2($fp)
		beq $t1, $t2, z
			offset_1 = findOffset(x)
			offset_2 = findOffset(y)
			若x(或y)为立即数k,那么对$t1
			(或$t2)寄存器加载值的指令修改为
			li $t1, k
			(li $t2, k)

	IF x != y GOTO z	... bne $t1, $t2, z	省略部分同上
	IF x >y GOTO z	... bgt $t1, $t2, z	省略部分同上
	IF x <y GOTO z	... blt $t1, $t2, z	省略部分同上
	IF x >= y GOTO z	... bne $t1, $t2, z	省略部分同上
	IF x <= y GOTO z	... bne $t1, $t2, z	省略部分同上
GOTO	GOTO x	j x	
PARAMETER	PARAM x	lw $t0, offset_1($fp) sw $t0, -offset_2($fp)	offset_1 = i * 4 offset_2 = findOffset(x) 其中, i为当前参数在该函数定义中的序.这是因为在调用者方面参数是按序压栈的.

CALL	x := CALL func	addi $sp, $sp, -4
		sw $ra, 0($sp)
		jal func
		lw $ra, 0($sp)
		addi $sp, $sp, 4
		sw $v0, -offset_1($fp)
		addi $sp, $sp, stackSize
			保存$ra旧值
			跳转
			恢复$ra旧值
			归还为保存$ra申请的栈空间
			取出返回值并赋值给x
			归还为传参申请的栈空间.
			offset_1 = findOffset(x)
			根据假设6:
			stackSize = 4 * argc
			其中, argc为函数func的参数个数.

ARG	ARG x	addi $sp, $sp, -4
		lw $t0, -offset_1($fp)
		sw $t0, 0($sp)
			根据假设6,每个参数在栈中分配4字节空间.
			offset_1 = findOffset(x)
			如果x是常数k, 那么lw指令修改为
			li $t0, k

READ	x := CALL READ	addi $sp, $sp, -4
		sw $ra, 0($sp)
		jal read
		lw $ra, 0($sp)
		addi $sp, $sp, 4
		sw $v0, -offset_1($fp)
			offset_1 = findOffset(x)

WRITE	WRITE x	lw $a0, -offset_1($fp)
		addi $sp, $sp, -4
		sw $ra, 0($sp)
		jal write
		lw $ra, 0($sp)
		addi $sp, $sp, 4
			offset_1 = findOffset(x)
			如果x是常数k, 那么lw指令修改为
			li $t0, k

Table 1 continued from previous page

IR Type	IR Code	MIP32 Code	Comment

RETURN	RETURN x	lw $v0, -offset_1($fp)
		addi $sp, $fp, 4
		lw $fp, 0($fp)
		jr $ra
			offset_1 = findOffset(x)
			如果x是常数k, 那么lw指令修改为
			li $t0, k