Simple liveness analysis and register allocation example

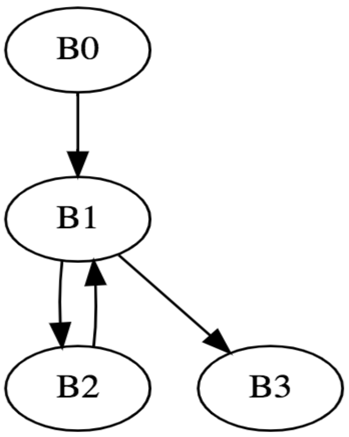
Sunday, 19 April 2020 14:24

```
#start_function main
  void main()
  int-list: total_s1, n_s1, index_s1, _t1_s0, _t2_s0, _t3_s0,
_t4_s0
  float-list:
  assign, n_s1, 10,
  main:
  assign, total_s1, 0,
  assign, index_s1, 1,
  loop_label_0:
  brgt, index_s1, n_s1, loop_label_1
  add, total_s1, index_s1, _t1_s0
  assign, total_s1, _t1_s0,
  add, index_s1, 1, index_s1
  goto, loop_label_0, ,
  loop_label_1:
  call, printi, total_s1
  add, n_s1, 1, _t2_s0
  mul, n_s1, _t2_s0, _t3_s0
  div, _t3_s0, 2, _t4_s0
  call, printi, _t4_s0
  return, , ,
#end_function main
```

Basic blocks

Block	Code
B0	0: assign, n_s1, 10, 1: main: 2: assign, total_s1, 0 3: assign, index_s1, 1
B1	4: loop_label_0: 5: brgt, index_s1, n_s1, loop_label_1
B2	6: add, total_s1, index_s1, _t1_s0 7: assign, total_s1, _t1_s0 8: add, index_s1, 1, index_s1 9: goto, loop_label_0
B3	10: loop_label_1: 11: call, printi, total_s1 12: add, n_s1, 1, _t2_s0 13: mul, n_s1, _t2_s0, _t3_s0 14: div, _t3_s0, 2, _t4_s0 15: call, printi, _t4_s0 16: return, , ,

Control Flow Graph



Live range analysis

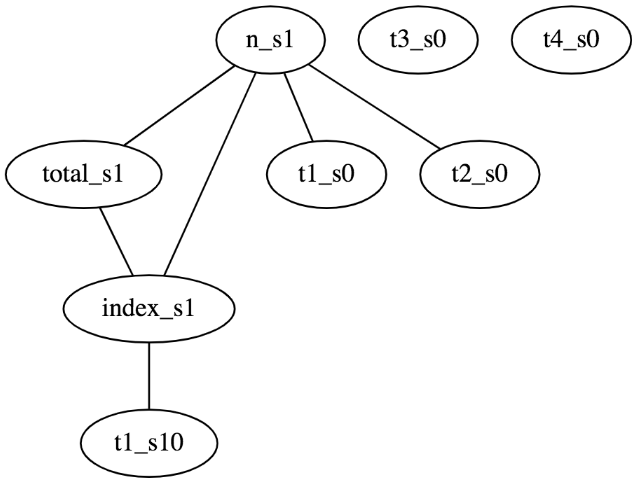
Block	Instruction	In	Out	Def	Use
B0	0: assign, n_s1, 10,		n_s1	n_s1	
	1: main:	n_s1	n_s1		
	2: assign, total_s1, 0	n_s1	n_s1, total_s1	total_s1	
	3: assign, index_s1, 1	n_s1, total_s1	n_s1, total_s1, index_s1	index_s1	
B1	4: loop_label_0:	n_s1, total_s1, index_s1	n_s1, total_s1, index_s1		
	5: brgt, index_s1, n_s1, loop_label_1	n_s1, total_s1, index_s1	n_s1, total_s1, index_s1		index_s1, n_s1
B2	6: add, total_s1, index_s1, _t1_s0	n_s1, total_s1, index_s1	n_s1, index_s1, _t1_s0	_t1_s0	total_s1, index_s1
	7: assign, total_s1, _t1_s0	n_s1, index_s1, _t1_s0	n_s1, total_s1, index_s1	total_s1	_t1_s0
	8: add, index_s1, 1, index_s1	n_s1, total_s1, index_s1	n_s1, total_s1, index_s1	index_s1	index_s1
	9: goto, loop_label_0	n_s1, total_s1, index_s1	n_s1, total_s1, index_s1		
B3	10: loop_label_1:	n_s1, total_s1	n_s1, total_s1		
	11: call, printi, total_s1	n_s1, total_s1	n_s1		total_s1
	12: add, n_s1, 1, _t2_s0	n_s1	n_s1, _t2_s0	_t2_s0	n_s1
	13: mul, n_s1, _t2_s0, _t3_s0	n_s1, _t2_s0	_t3_s0	_t3_s0	n_s1, _t2_s0
	14: div, _t3_s0, 2, _t4_s0	_t3_s0	_t4_s0	_t4_s0	_t3_s0
	15: call, printi, _t4_s0	_t4_s0			_t4_s0
	16: return, , ,				

Variable	Live ranges
n_s1	B0[0 - 3 defines] B1[4 - 5] B2[6 - 9] B3[10 - 13]
total_s1	B0[2 - 3 defines] B1[4 - 5] B2[6-6] B2[7 - 9, defines] B3[10-11]
index_s1	B0[3-3 defines] B1[4-5] B2[6-8] B2[8-9 defines]
_t1_s0	B2[6-7 defines]
_t2_s0	B3[12-13 defines]
_t3_s0	B3[13-14 defines]
_t4_s0	B3[14-15 defines]

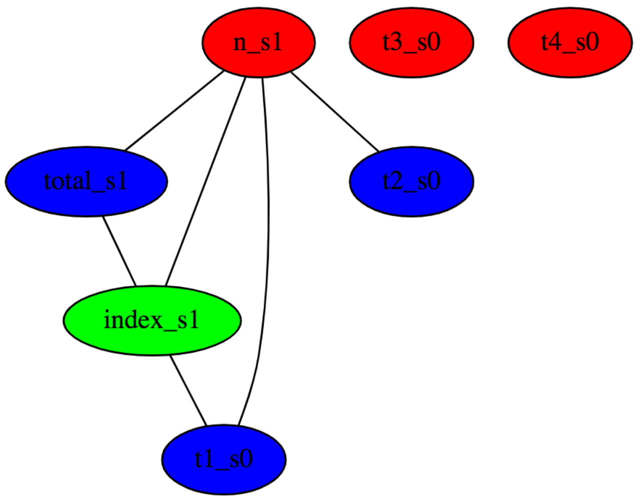
In this example, all live ranges of the same variables happen to be on the same web

Interference graph and graph coloring

n_s1 -> total_s1, index_s1, t1_s0, t2_s0
total_s1 -> n_s1, index_s1
index_s1 -> total_s1, n_s1, t1_s0
t1_s0 -> n_s1, index_s1
t2_s0 -> n_s1
t3_s0 ->
t4_s0 ->



Sample coloring



Each color represents a register allocated to the live range specified by the given node. 3 registers are sufficient in this case. Live ranges that interfere may not get the same register.

Estimating spill costs

Assume store cost = load cost = 1
Assume block is a loop if it ends with a branch (not robust, but me be a sufficient hack for time being). Assume loops have weight of 10. Not checking for nested loops

web	range	stores	loads	block weight	cost	web total cost
n_s1	B0[0-3]	1	0	1	1	1
	B1[4-5]	0	1	10	10	11
	B2[6-9]	0	0	10	0	11
	B3[10-13]	0	2	1	2	**13**
total_s1	B0[2-3]	1	0	1	1	1
	B1[4-5]	0	0	1	0	1
	B2[6-6]	0	1	10	10	11
	B2[7-9]	1	0	10	10	21
	B3[10-11]	0	1	1	1	**22**
index_s1	B0[3-3]	1	0	1	1	1
	B1[4-5]	0	1	10	10	11
	B2[6-8]	0	2	10	20	31
	B2[8-9]	0	1	10	10	**41**
_t1_s0	B2[6-7]	1	1	10	20	**20**
_t2_s0	B3[12-13]	1	1	1	2	**2**
_t3_s0	B3[13-14]	1	1	1	2	**2**
_t4_s0	B3[14-15]	1	1	1	2	**2**