Liveness Analysis

Lecture 10

Liveness

In our intermediate representation (IR), we assume we have an unlimited number of *temporaries* (virtual registers).

In the PowerPC architecture, we have 32 general purpose registers (at most 30 available).

How do we fit all the temporaries into the limited set of real registers?

Question: how many temps are really “in use” at the same time?

If the maximum number of temps in use at any time is <= 30, then they can fit into the registers.

When the maximum number exceeds 30, the excess has to be *spilled* to memory (space reserved in the stack frame). Which temps should be spilled?

Liveness

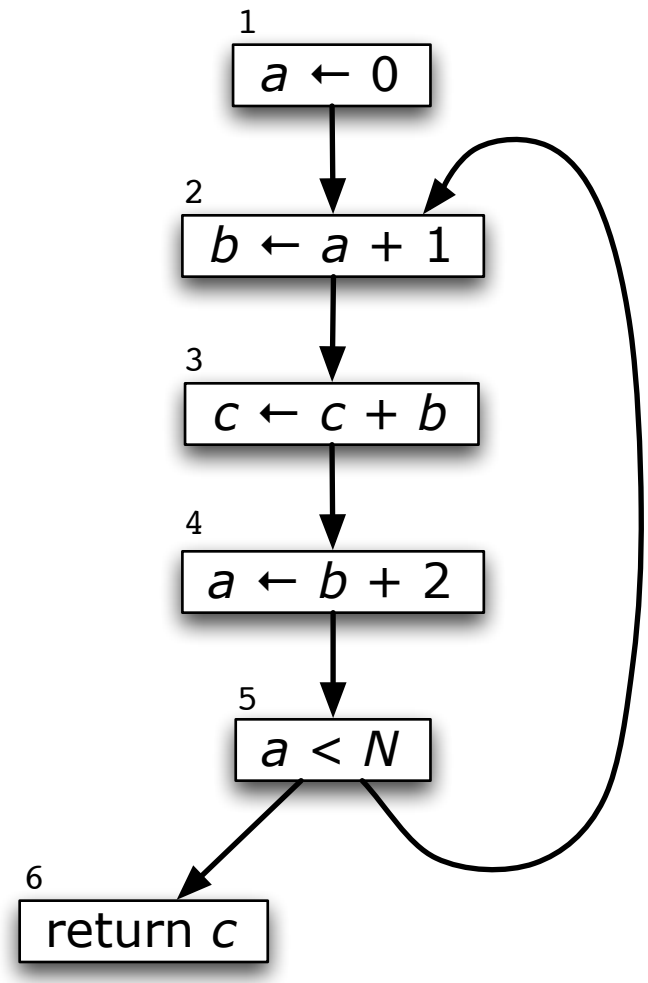
Definition: A variable is *live* if it holds a value that will/might be used in the future.

The representation of the program that we use for liveness analysis (determining which variables are live at each point in a program), is a *control flow graph*.

The nodes in a control flow graph are basic statements (instructions). There is an edge from statement x to statement y if x can be immediately followed by y (control *flows* from x to y).

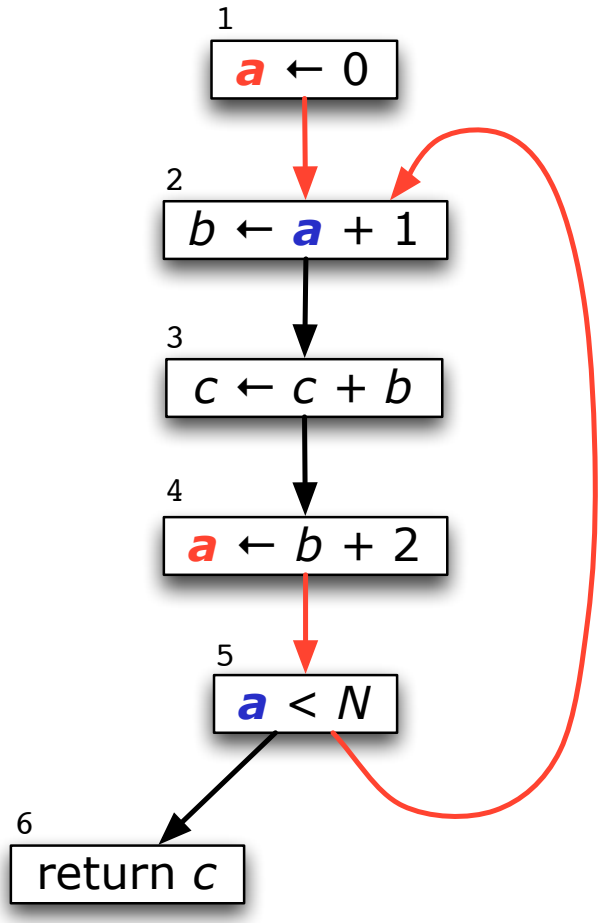
Control Flow Graph

$a \leftarrow 0$
L_1: $b \leftarrow a + 1$
$c \leftarrow c + b$
$a \leftarrow b + 2$
if $a < N$ goto L_1
return c

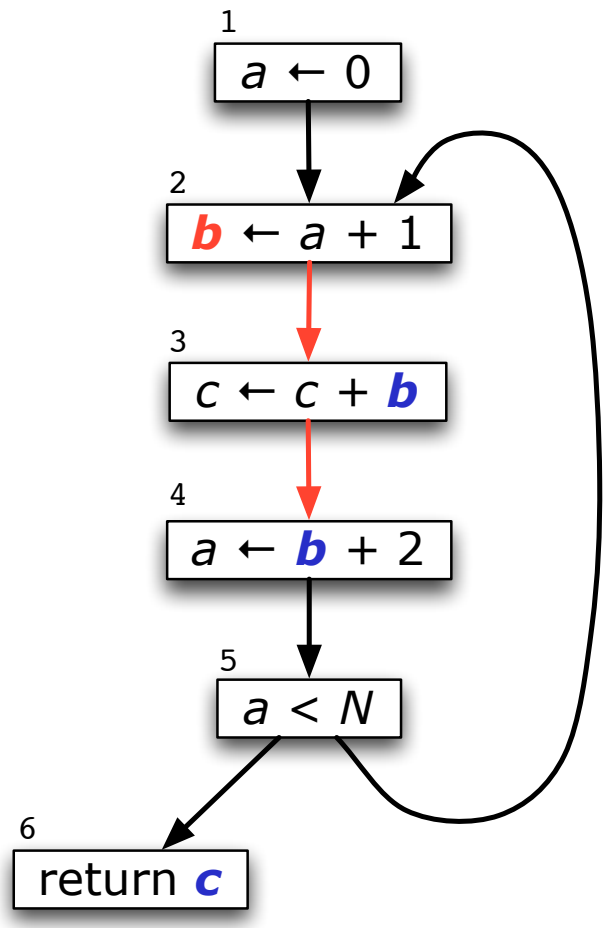


Control flow graph of a program

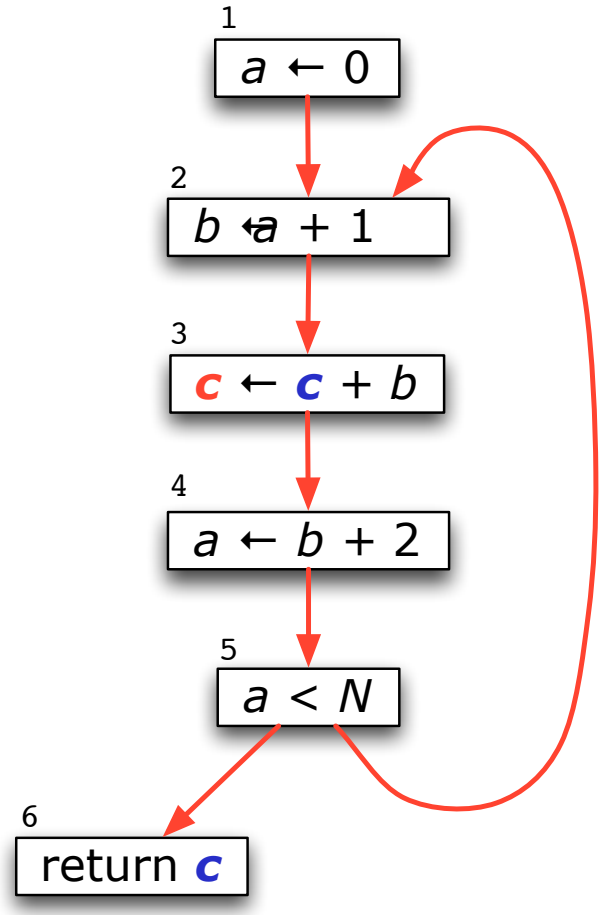
Live Ranges



Live ranges of a

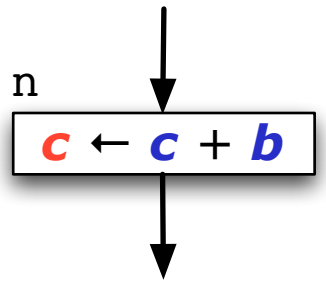


Live ranges of b



Live ranges of c

Use/Def



use [*n*] = { *c*, *b* }

def [*n*] = { *c* }

Live In - Live Out

Definition: a variable (temp) *a* is ***live-in*** at node *n* if it is used at n (*a* ∈ *use*[*n*]), or if there is a path from n to a node that uses *a* that does not contain a definition of (assignment to) *a*. We write *a* ∈ *in*[*n*].

Definition: a variable *a* is ***live-out*** at node *n* if it is live-in at one of the successors of n. We write *a* ∈ *out*[*n*].

The sets *in*[*n*] and *out*[*n*] satisfy the equations

$$in[n] = use[n] \cup (out[n] - def[n])$$

$$out[n] = \bigcup \{ in[s] \mid s \in succ[n] \}$$

Calculation of in[n], out[n]

Fixpoint calculation for two families of sets: *in*[*n*] and *out*[*n*]

```
for each n

      in[n] := {}; out[n] = {}

repeat

      for each n
            in'[n] := in[n]; out'[n] := out[n]
            in[n] := use[n] ∪ (out[n] - def[n])
            out[n] := ∪ {in[s] | s ϵ succ[n]}

until in'[n] = in[n] and out'[n] = out[n]
for all n
```

Calculation of in[n], out[n]

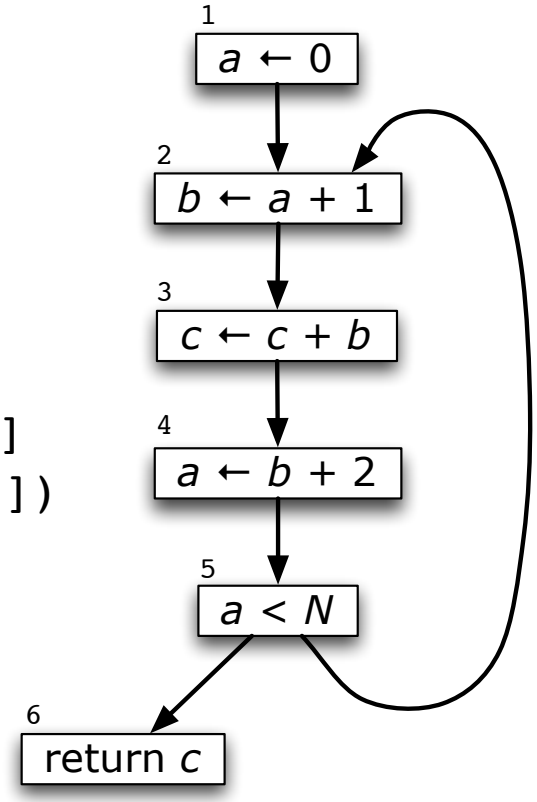
```
for each n

      in[n] := {}; out[n] = {}

repeat

      for each n
            in'[n] := in[n]; out'[n] := out[n]
            in[n] := use[n] ∪ (out[n] - def[n])
            out[n] := ∪ {in[s] | s ϵ succ[n]}

until in'[n] - in[n] and out'[n] = out[n]
for all n
```



Iterative Calculation of in, out

iterations:			0		1		2		3	
node	use	def	out	in	out	in	out	in	out	in
6	c	∅	∅	∅	∅	c	∅	c	∅	c
5	a	∅	∅	∅	c	ac	ac	ac	ac	ac
4	b	a	∅	∅	ac	bc	ac	bc	ac	bc
3	bc	c	∅	∅	bc	bc	bc	bc	bc	bc
2	a	b	∅	∅	bc	ac	bc	ac	bc	ac
1	∅	a	∅	∅	ac	c	ac	c	ac	c

Iterative Calculation of in, out

iterations:			0		1		2		3	
node	use	def	out	in	out	in	out	in	out	in
6	c	∅	∅	∅	∅	c	∅	c	∅	c
5	a	∅	∅	∅	c	ac	ac	ac	ac	ac
4	b	a	∅	∅	ac	bc	ac	bc	ac	bc
3	bc	c	∅	∅	bc	bc	bc	bc	bc	bc
2	a	b	∅	∅	bc	ac	bc	ac	bc	ac
1	∅	a	∅	∅	ac	c	ac	c	ac	c

$$in(6) = use(6) \cup (out(6) - def(6))$$
$$= \{c\} \cup (\varnothing - \varnothing) = \{c\}$$

Iterative Calculation of in, out

iterations:			0		1		2		3	
node	use	def	out	in	out	in	out	in	out	in
6	c	∅	∅	∅	∅	c	∅	c	∅	c
5	a	∅	∅	∅	c	ac	ac	ac	ac	ac
4	b	a	∅	∅	ac	bc	ac	bc	ac	bc
3	bc	c	∅	∅	bc	bc	bc	bc	bc	bc
2	a	b	∅	∅	bc	ac	bc	ac	bc	ac
1	∅	a	∅	∅	ac	c	ac	c	ac	c

succ(5) = {2,6}

out(5) = in(2) ∪ in(6)

= ∅ ∪ {c} = {c}

Iterative Calculation of in, out

iterations:			0		1		2		3	
node	use	def	out	in	out	in	out	in	out	in
6	c	∅	∅	∅	∅	c	∅	c	∅	c
5	a	∅	∅	∅	c	ac	ac	ac	ac	ac
4	b	a	∅	∅	ac	bc	ac	bc	ac	bc
3	bc	c	∅	∅	bc	bc	bc	bc	bc	bc
2	a	b	∅	∅	bc	ac	bc	ac	bc	ac
1	∅	a	∅	∅	ac	c	ac	c	ac	c

succ(5) = {2,6}

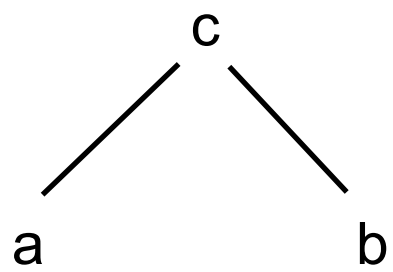
in(5) = use(5) ∪ (out(5) - def(5))

= {a} ∪ ({c} - ∅) = {a,c}

Interference

Two temps *interfere* if they are live at the same time. Interfering temps cannot be assigned to the same register.

The interference relation can be represented by an interference graph in which nodes correspond to temps and edges connect interfering temps.



The interference graph
for our example.