Linear Scan Register Allocation

on Static Single Assignment Form

Christian Wimmer
cwimmer@uci.edu
www.christianwimmer.at

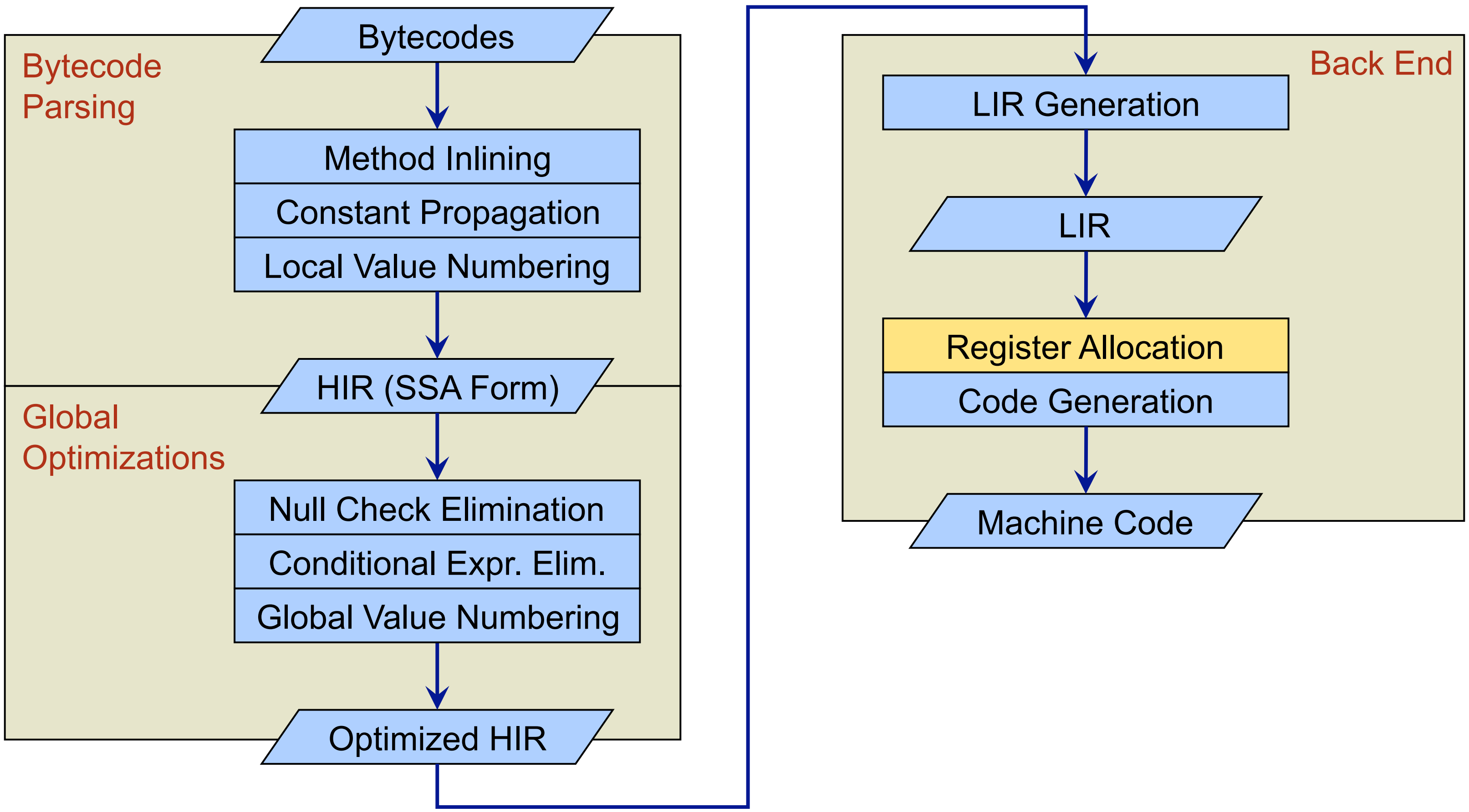
April 2010

Department of Computer Science
University of California, Irvine

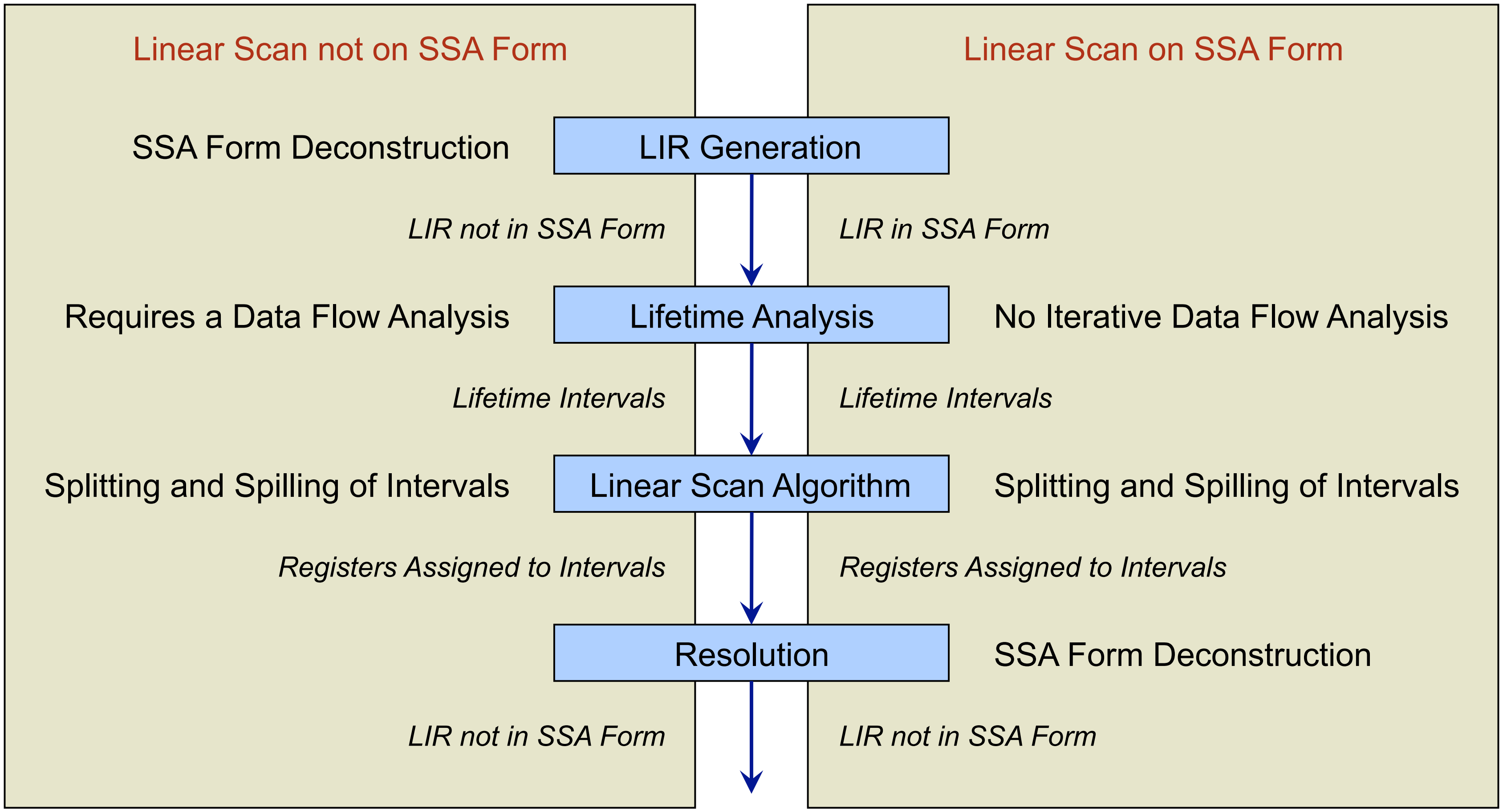
Introduction

- Register allocation
  - Graph coloring algorithm
  - Linear scan algorithm
- Static single assignment (SSA) form
  - One definition per variable that dominates all uses
  - Variables that interfere somewhere also interfere at one definition
  - Interference graph is chordal
  - Graph coloring in polynomial time
- Linear scan algorithm on SSA form
  - Liveness analysis without iterative data flow analysis
  - Use SSA properties during register allocation
  - SSA deconstruction integrated with resolution phase of linear scan

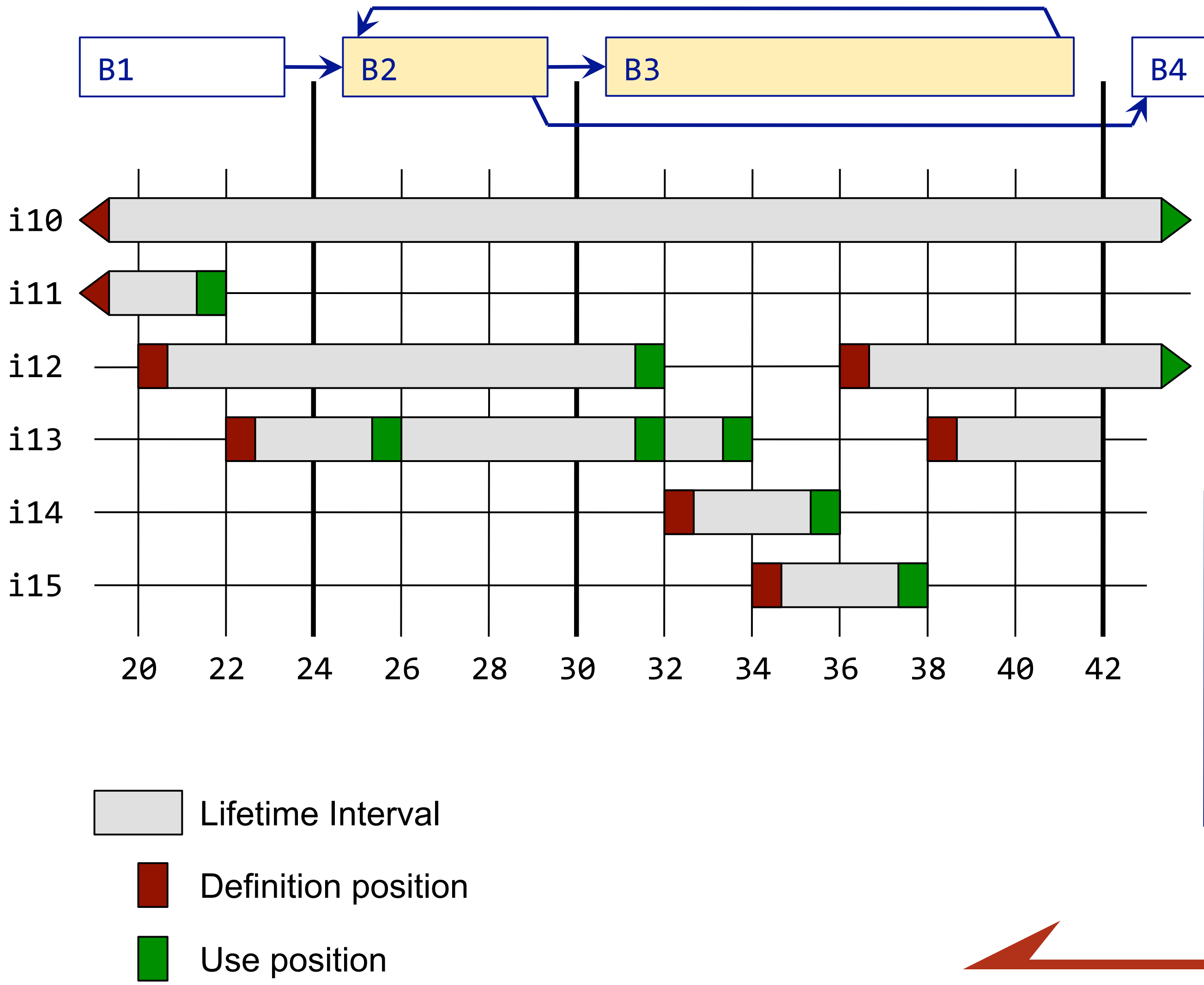
Java HotSpot™ Client Compiler



Phases of Linear Scan Algorithm



Lifetime Intervals Without SSA Form



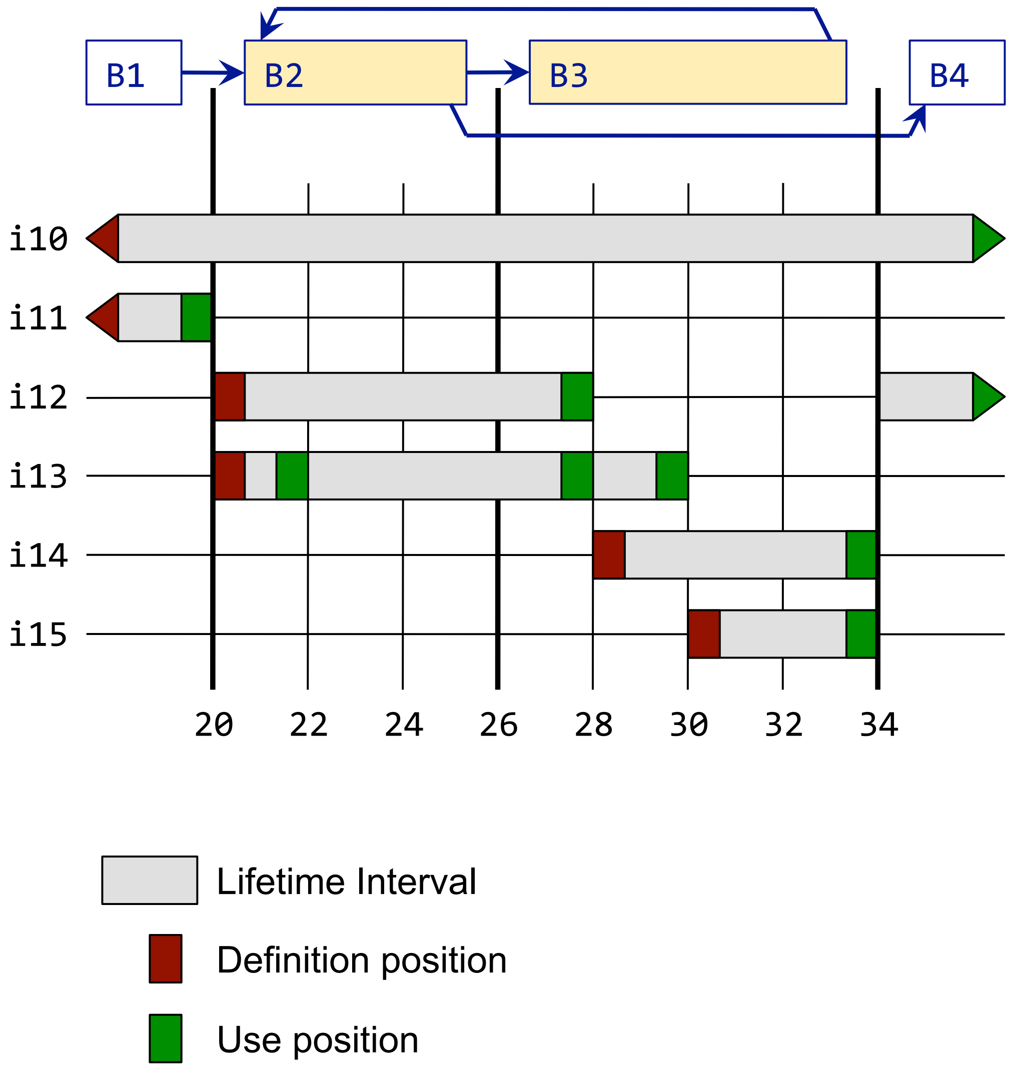

```
    define R10, R11
20: move 1 -> R12
22: move R11 -> R13

24: label B2
26: cmp R13, 1
28: branch lessThan B4

30: label B3
32: mul R12,R13 -> R14
34: sub R13, 1 -> R15
36: move R14 -> R12
38: move R15 -> R13
40: jump B2

42: label B4
    use R10, R12
```

Lifetime Intervals With SSA Form



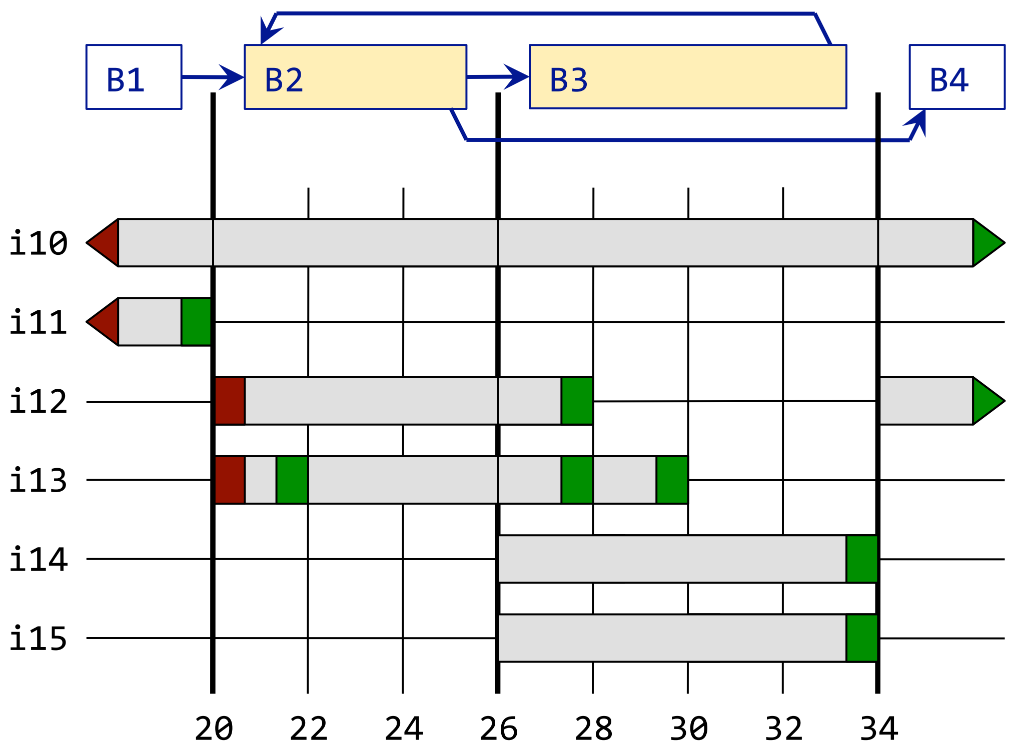

```
    define R10, R11

20: label B2
    phi [1, R14] -> R12
    phi [R11, R15] -> R13
22: cmp R13, 1
24: branch lessThan B4

26: label B3
28: mul R12, R13 -> R14
30: sub R13, 1 -> R15
32: jump B2

34: label B4
    use R10, R12
```

Construction of Lifetime Intervals




```
define R10, R11
```

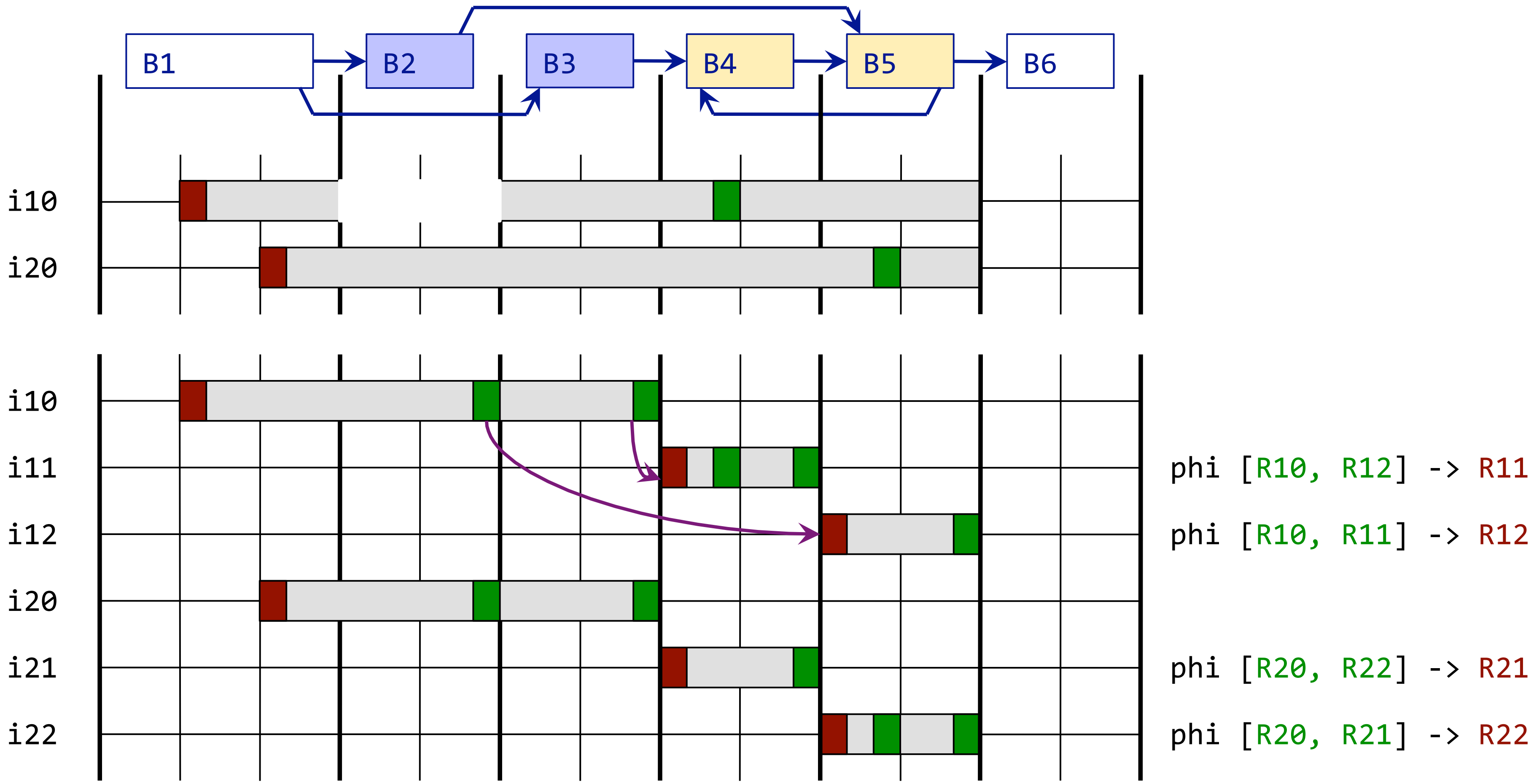
```
20: label B2
    phi [1, R14] -> R12
    phi [R11, R15] -> R13
22: cmp R13, 1
24: branch lessThan B4
```

```
26: label B3
28: mul R12, R13 -> R14
30: sub R13, 1 -> R15
32: jump B2
```

```
34: label B4
    use R10, R12
```

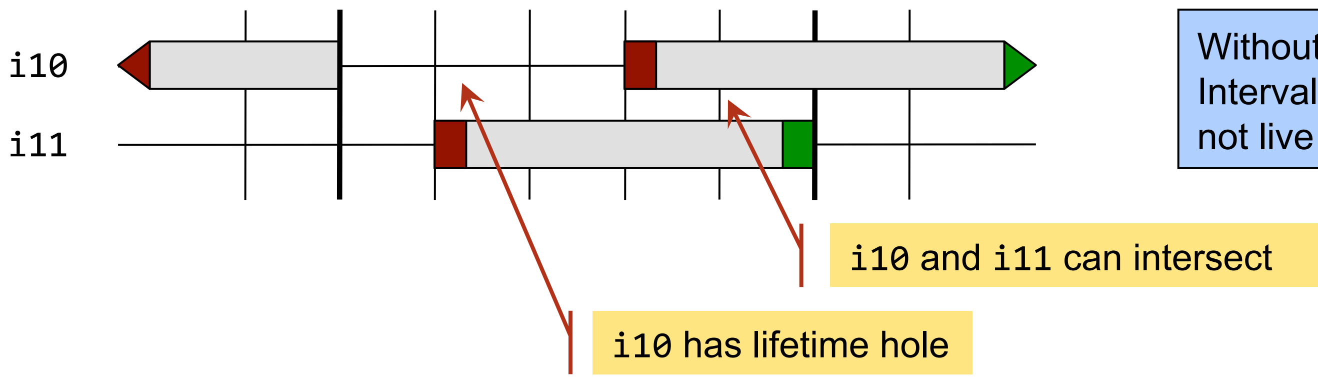
Initial Live Set from Successors
Add Input Operands of Successors' Phis
Process Operations in Reverse Order
Remove Phi Functions from Live Set
Extend Live Ranges of Loop Variables

Irreducible Control Flow



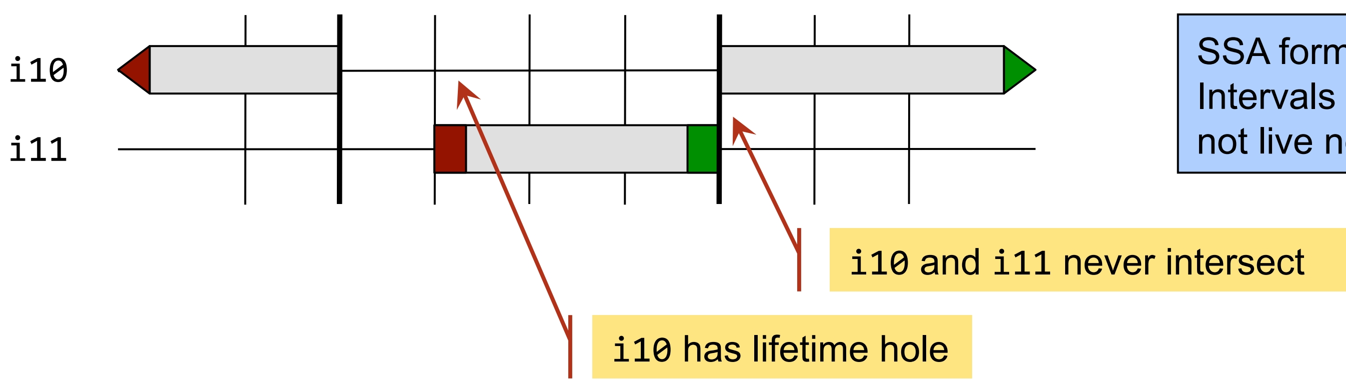
Changes to Linear Scan Algorithm

Linear scan not on SSA form



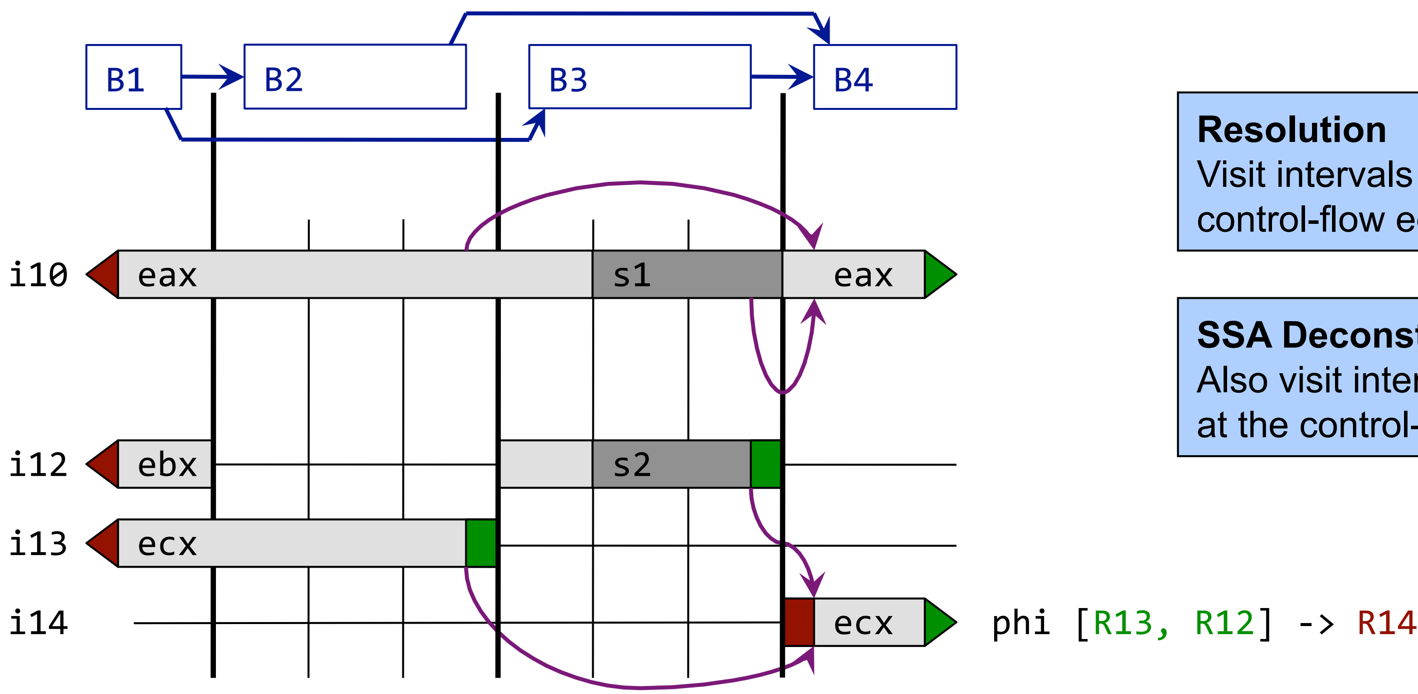
Without SSA form:
Intervals that are currently not live can block registers

Linear scan on SSA form



SSA form guarantees:
Intervals that are currently not live never block registers

SSA Deconstruction during Resolution



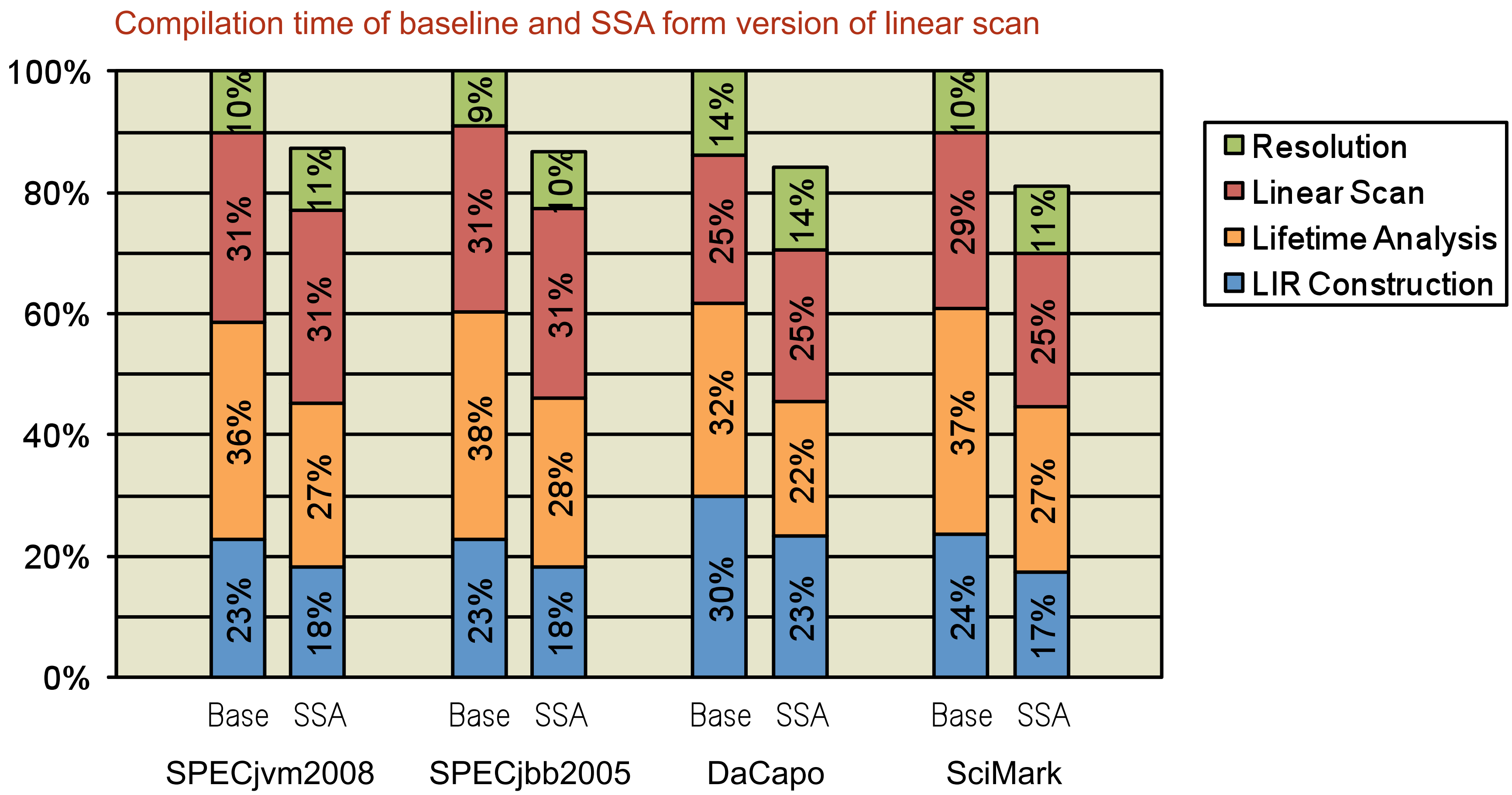
Resolution
Visit intervals live across control-flow edges

SSA Deconstruction
Also visit intervals starting at the control-flow edge

```
B2 – B4:
No move necessary
```

```
B3 – B4:
move s1 -> eax
move s2 -> ecx
```

Compilation Time



2 * Intel Xeon X5140, 2.33 GHz, 4 cores, 32 GByte memory
Ubuntu Linux, kernel version 2.6.28
SPECjvm2008: Lagom w/o SciMark

Phi Functions and Move Instructions

	DaCapo			SciMark		
	Baseline	SSA Form		Baseline	SSA Form	
Before Register Allocation						
Moves	402,678	355,936	-12%	908	593	-35%
Phi Functions	0	20,542		0	168	
After Register Allocation						
Moves Register to Register	127,318	124,351	-2%	193	177	-8%
Moves Constant to Register	71,967	70,663	-2%	99	98	-1%
Moves Stack to Register	3,718	3,722	+0%	12	12	0%
Moves Register to Stack	65,973	56,639	-14%	166	158	-5%
Moves Constant to Stack	0	1,386		0	1	
Moves Stack to Stack	0	647		0	0	

Summary

- Linear scan algorithm on SSA form
  - Liveness analysis without iterative data flow analysis
  - Use SSA properties during register allocation
  - SSA deconstruction integrated with resolution phase of linear scan
- Benefits
  - Faster, especially liveness analysis
  - Simpler compiler code
  - Equally good (or slightly better) machine code
  - Eliminates SSA deconstruction phase
- Do register allocation on SSA form!
  - No matter what algorithm you use