Optimal Spilling for CISC Machines with Few Registers

Andrew Appel
Princeton University

Lal George
Bell Labs

June 20, 2001

The Problem

Few Registers $\Rightarrow$ Spilling!

Total Instructions	Static number of Spill instructions	
	K = 32	K = 8
163,355	84	22,123 14%

CISC Machines

Allow memory operands

Can free the register t in the instruction:

$$t \leftarrow t \oplus s$$

by generating:

$$mem_t \leftarrow mem_t \oplus s$$

Traditional register allocators:

$$\begin{aligned} r &\leftarrow mem_t \\ r &\leftarrow r \oplus s \\ mem_t &\leftarrow r \end{aligned}$$

Register allocation using integer linear programming

Ken Wilkens, T. Kong, D.Goodwin ['96, '98]

Solve the allocation problem in its entirety!

- Can exhibit long solve times for small programs.
- Published literature is difficult and complex.
- "The performance results are all in the details"

Our Approach

Phase I: Optimal Spilling

At a program point, should each variable be in a register or memory?

- **At most K live variables at any point**

Phase II: Register Assignment

Which register?

- **Guaranteed not to insert spills.**

Phase I

Optimal Spilling

Register or memory?

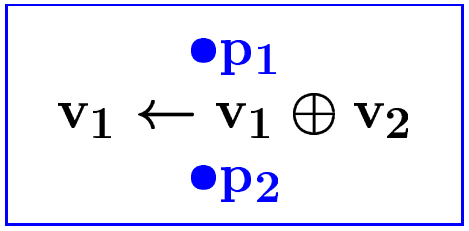
- **0-1 integer linear program.**
- **Solved optimally and quickly .**
- **Modelled in AMPL**
  - **Extremely simple**
  - **No special tuning required**
- **At most K live variables at any program point.**

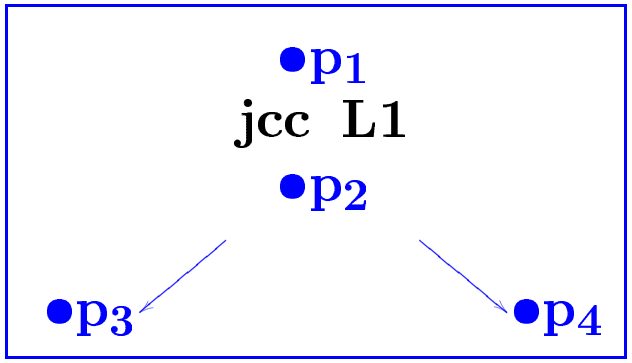
Phase I — in more detail

Pseudo registers and input flowgraph

Set V of Pseudo registers = $\{ v_1\ v_2 \cdots \}$

Set P of Program points = $\{p_1\ p_2 \cdots \}$





Linear programming variables

Just 4 kinds of LP variables

For all live variables $\mathbf{v}$ at each program point $\mathbf{p}$:

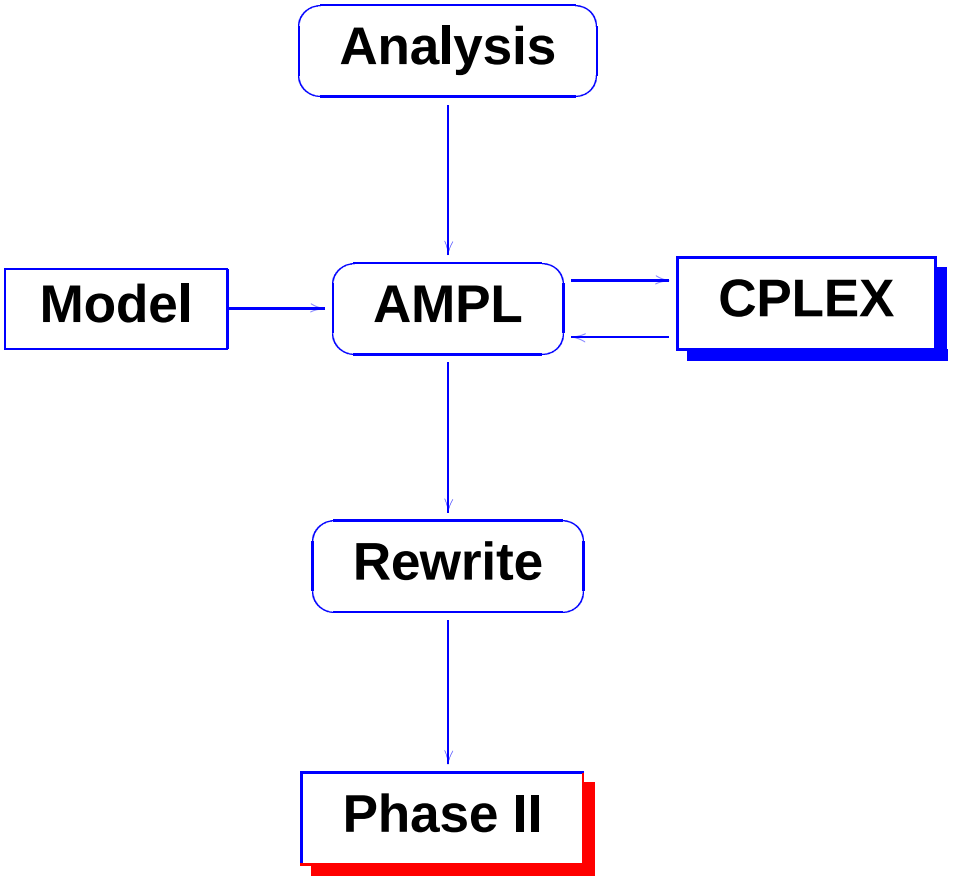
Variable	Description
$\mathbf{load_p^v}$	at $\mathbf{p}$, $\mathbf{v}$ must be loaded from memory
$\mathbf{store_p^v}$	at $\mathbf{p}$, $\mathbf{v}$ must be stored to memory
$\mathbf{inReg_p^v}$	at $\mathbf{p}$, $\mathbf{v}$ continues in a register
$\mathbf{inMem_p^v}$	at $\mathbf{p}$, $\mathbf{v}$ continues in memory

Liveness constraint: $\mathbf{load_p^v + store_p^v + inReg_p^v + inMem_p^v = 1}$

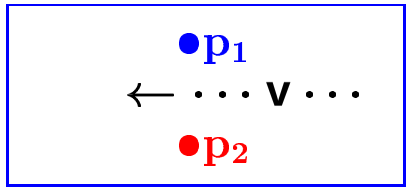
Using Linear Programming Results

Program	LP Results	Final Program
$\bullet p_j$	$\mathrm{load}^{v}_{p_j} = 1$	$\bullet\ p_j \quad v \leftarrow \mathrm{mem}_v$
$\bullet p_k$ $v \leftarrow v \oplus w$	$\mathrm{inMem}^{w}_{p_k} = 1$	$\bullet p_k$ $v \leftarrow v \oplus \mathrm{mem}_w$
$\bullet p_l$ $v \leftarrow w$ $\bullet\ p_m$	$\mathrm{store}^{v}_{p_m} = 1$	$\bullet p_l$ $\mathrm{mem}_v \leftarrow w$ $\bullet\ p_m$

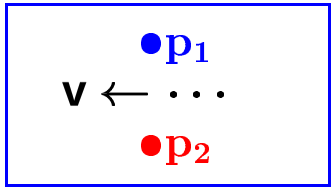
Compiler Organization



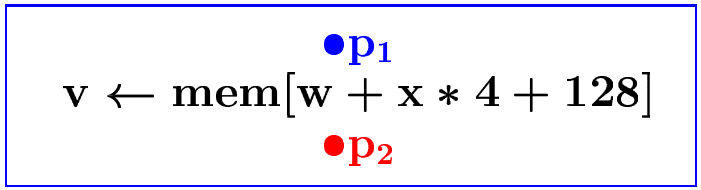
Constraints



$$\mathrm{inReg}^{v}_{p_1} + \mathrm{load}^{v}_{p_1} = 1$$



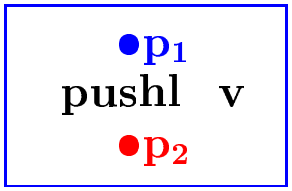
$$\mathrm{inReg}^{v}_{p_2} + \mathrm{store}^{v}_{p_2} = 1$$



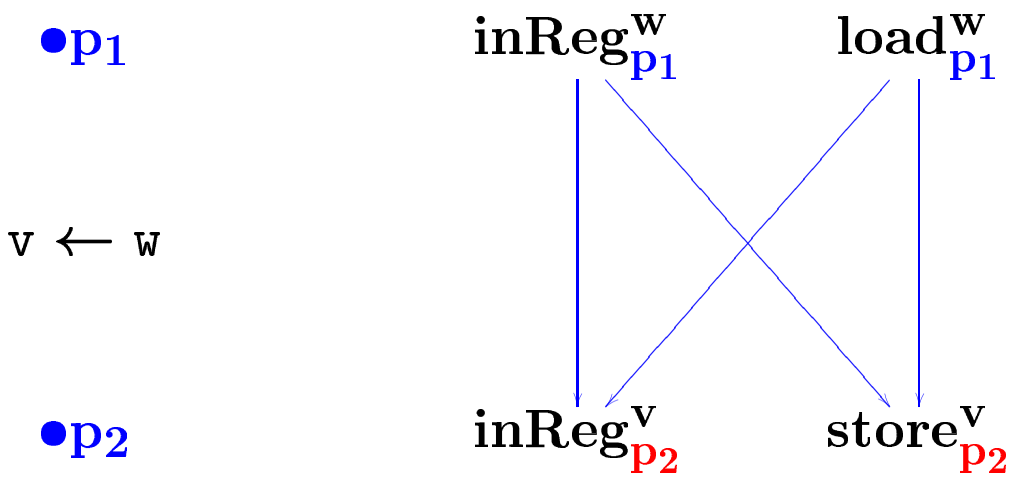
$$\mathrm{inReg}^{w}_{p_1} + \mathrm{load}^{w}_{p_1} = 1$$

$$\mathrm{inReg}^{x}_{p_1} + \mathrm{load}^{x}_{p_1} = 1$$

$$\mathrm{inReg}^{v}_{p_2} + \mathrm{store}^{v}_{p_2} = 1$$



Splits



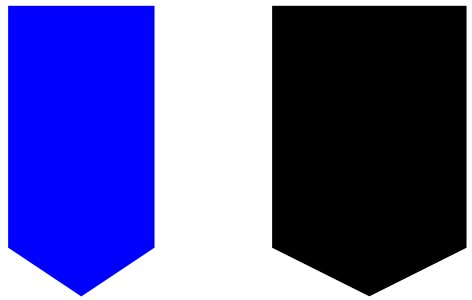
$$\mathrm{load}^{w}_{p_1} + \mathrm{inReg}^{w}_{p_1} = \mathrm{store}^{v}_{p_2} + \mathrm{inReg}^{v}_{p_2}$$

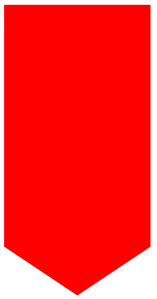
$$\mathbf{inMem}^{\mathbf{w}}_{\mathbf{p_1}} + \mathbf{store}^{\mathbf{w}}_{\mathbf{p_1}}$$

$$\mathbf{inMem}^{\mathbf{v}}_{\mathbf{p_2}} + \mathbf{load}^{\mathbf{v}}_{\mathbf{p_2}}$$

Coloring constraint

$$\sum_{v} store_{p}^{v} + \sum_{v} inReg_{p}^{v} <= K$$





$$\sum_{v} inReg_{p}^{v} + \sum_{v} load_{p}^{v} <= K$$

Objective cost function

Minimize the weighted cost of:

- **inserted loads and stores from $\mathrm{load}_{\mathrm{p}}^{\mathrm{v}}$ and $\mathrm{store}_{\mathrm{p}}^{\mathrm{v}}$,**
- **Penalty for using memory operands**

where

$$\mathtt{t} \leftarrow \mathtt{t} \oplus \mathtt{s}$$

is rewritten to

$$\mathtt{mem_t} \leftarrow \mathtt{mem_t} \oplus \mathtt{s}$$

Phase II: In more detail

Which register?

Should be done without further spilling!

However · · ·

no more than K live variables at any point!

Register Assignment

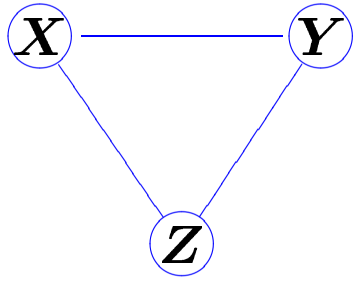
Aggregate register pressure is insufficient!!

Suppose K = 2

Live variables

Point	Live
p_i	{X, Y}
p_j	{Y, Z}
p_k	{Z, X}

Interference graph

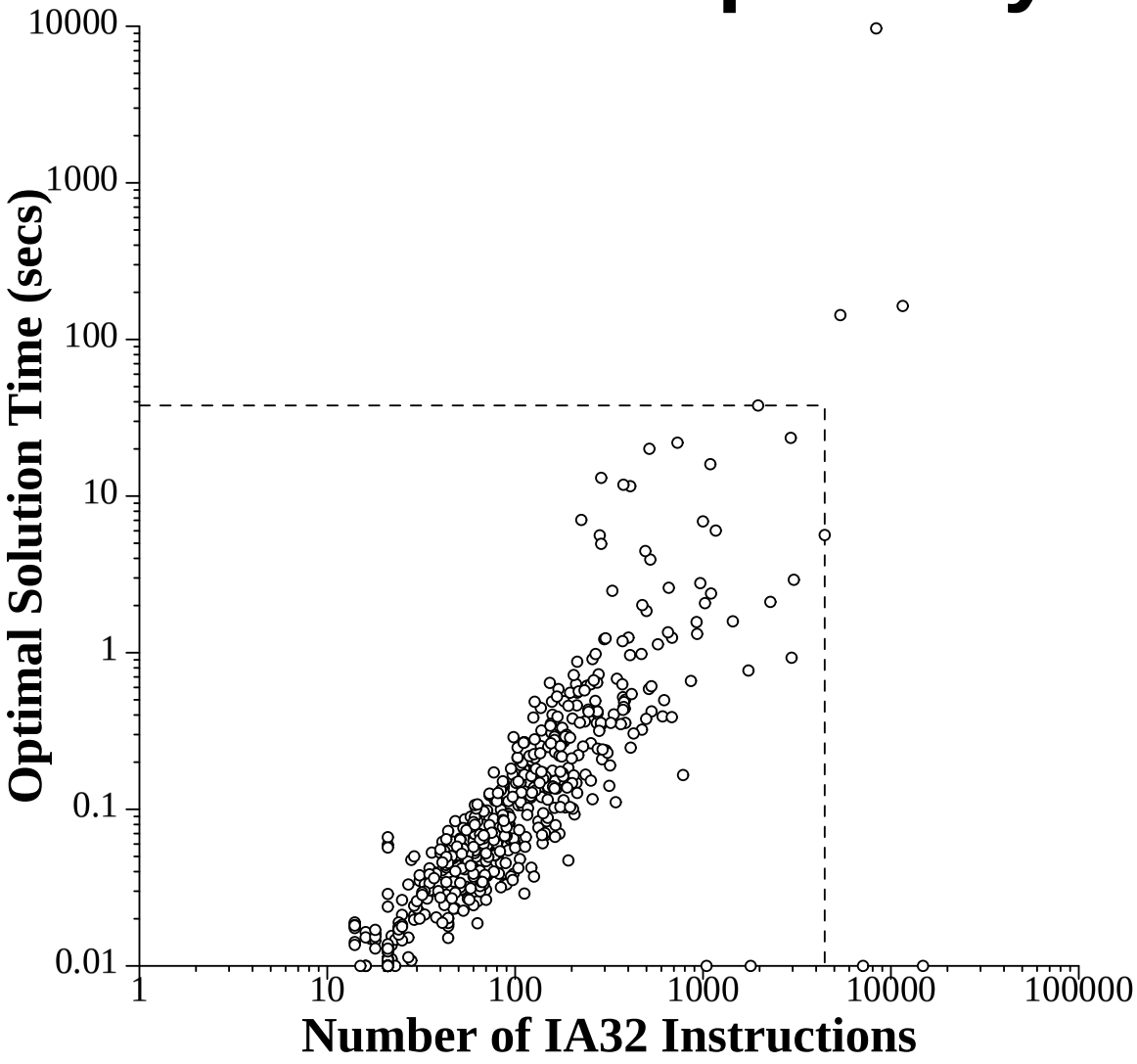


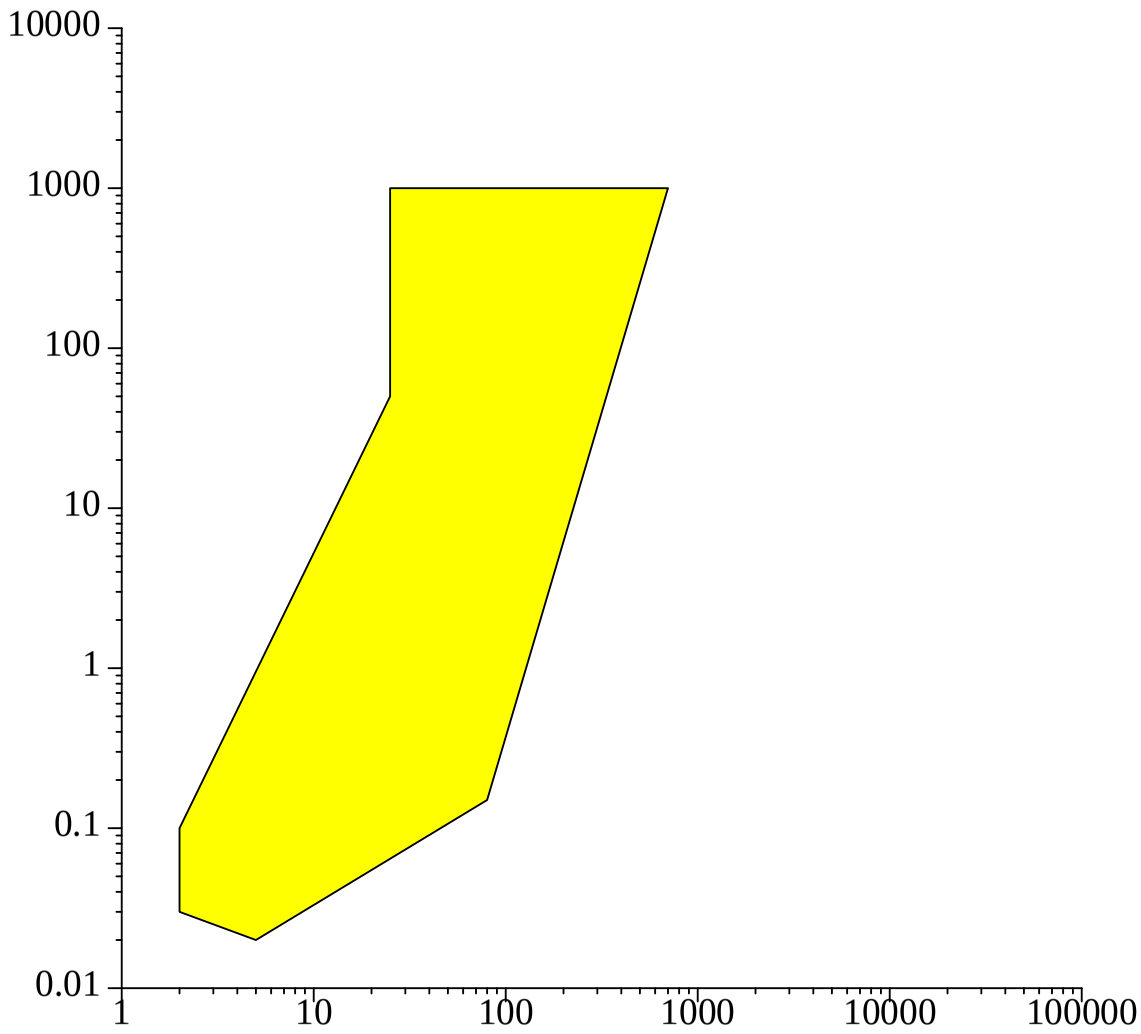
Optimistic coalescing

- Insert parallel copies before every instruction.
- Resulting interference graph guaranteed to be colorable.
- Paper defines the Optimal Coalescing Problem.
- Our solution based on the Park and Moon optimistic coalescing.
  - ▷ Details in the paper

Phase I

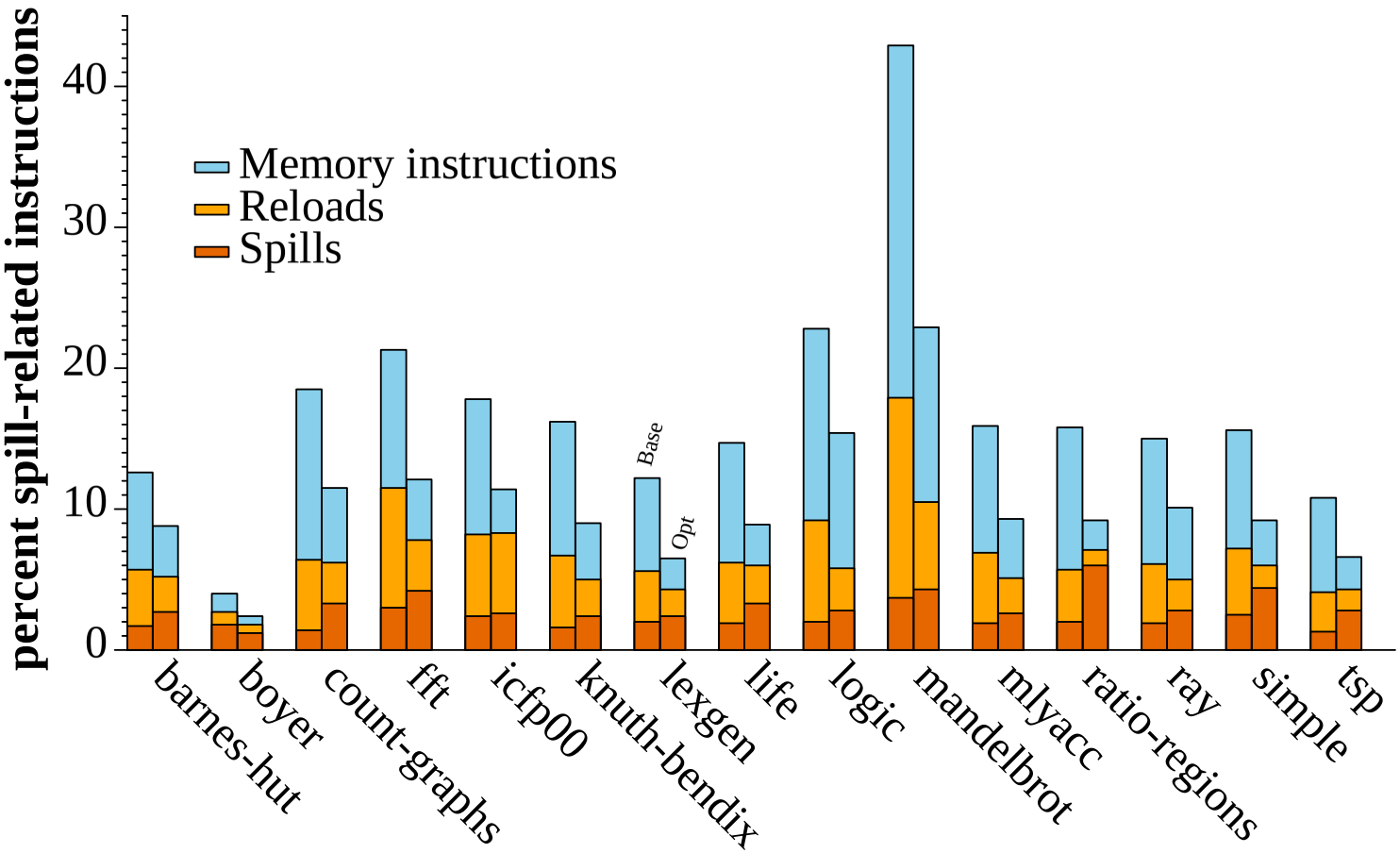
Time to solve optimally




10000
1000
100
10
1
0.1
0.01
1
10
100
1000
10000
100000

Static spill statistics

	Spills		Reloads		Memory Instructions	
	Base	Opt	Base	Opt	Base	Opt
Total	3040	4310	6771	3804	12312	5009



Execution Speed

Benchmark	Base	Opt	Speedup %
barnes-hut	2.92s	2.92s	0.0
boyer	12.57	12.49	0.0
mlyacc	9.14	9.11	0.0
tsp	6.92	6.77	2.2
lexgen	9.08	8.84	2.7
count-graphs	24.07	22.15	8.7
icfp00	109.29	99.72	9.6
fft	8.58	7.80	10.0
logic	5.10	4.61	10.6
knuth-bendix	8.08	7.22	11.9
mandelbrot	27.92	23.21	20.3
life	19.03	15.24	24.9
simple	31.53	25.12	25.5

Contributions

- A two phase approach
- An integer linear program that is:
  - Extremely simple
  - Requires no fine tunning
  - Solved quickly
- Effective!
  - Production quality compilers.
  - Highly dependent on good integer IL solvers.