

# Optimal Bitwise Register Allocation using Integer Linear Programming

Rajkishore Barik<sup>†</sup>, Christian Grothoff\*,  
Rahul Gupta<sup>†</sup> and Vinayaka Pandit<sup>†</sup>

<sup>†</sup> IBM India Research Lab

\* University of Denver, Colorado

# Optimal (Bitwise) Reg. Allocation Matters

- Evaluate register allocation algorithms
- Compilation time increasingly irrelevant
  - especially for embedded devices
- Helps predict architectural trade-offs

# Goals

- Give reference point for “optimal” register allocation and simple but **general** framework for comparison
- Model should support real-world architectures
- Practical for medium-size problems for critical pieces of code

# Non-goals

Our work is:

- not a replacement for general purpose allocators
- not a precise model for modern high-end processors
- not architecture specific

# Background: Integer Linear Programming

Linear problems are of the form:

$$\begin{array}{ll} \text{minimize} & \sum_{i \in I} a_i x_i \\ \text{such that} & \sum_{i \in I} b_i x_i \leq c_i \end{array}$$

For integer linear problems, constraints such as  $x_i \in \mathbb{N}$  maybe added.

## Related work: ILP in Reg. Allocation (1/2)

- Nandivada and Palsberg use ILP for maximizing opportunities for parallel loads
- Parallel load reads two 32-bit values on 64-bit bus
- Need to compute optimal stack co-locations
- Problem is NP-complete

## Related work: ILP in Reg. Allocation (2/2)

- Fu and Wilken (MICRO 35) used 0-1-ILP for Optimal Register Allocation
- ILP decides at each program point: to spill or not to spill
- Improved ILP runtime using various reduction techniques

## Related Work: Bitwise Reg. Alloc. (1/2)

- Tallam and Gupta (POPL 2003) proposed bitwidth aware global register allocation
- Iterative algorithm used to determine leading and trailing dead bit sections.
- Bitwidth used for iterative coalescing of interfering nodes

## Related Work: Bitwise Reg. Alloc. (2/2)

- Li, Zhang and Gupta (LCPC 2004) proposed speculative subword register allocation
- Replace static bitwidth estimation algorithms with profiling data
- Use hardware support to revert to less-efficient variant if profiling data is wrong

# Our Approach

- Produce simple and general ILP model
  - Feasible  $\equiv$  correct
  - No fixed binding of temporary to register
  - Consider control-flow and execution counts
  - Support RISC and CISC architectures
  - Support bitwise or subword allocations
- Given general formulation, find ways to reduce search space

# Basic Formulation

$$\min \sum_{i \in V} S_i \cdot x_{i,\sigma} \quad (1)$$

such that

$$\bigwedge_{\substack{r \in R - \{\sigma\} \\ n \in N}} \sum_{i \in V} x_{i,r} \cdot w_{i,n} \leq b \quad (2)$$

$$\bigwedge_{i \in V} \sum_{r \in R} x_{i,r} = 1 \quad (3)$$

$$\bigwedge_{\substack{i \in V \\ r \in R}} x_{i,r} \in \{0, 1\} \quad (4)$$

# Extension: Control-Flow Graph

$$\min \sum_{\substack{i \in V \\ n \in N}} S_{i,n} \cdot x_{i,\sigma,n} + \sum_{\substack{i \in V \\ r \in R, n \in N}} \mu_{r,n} \cdot C_{i,r,n} \quad (5)$$

such that

$$\bigwedge_{\substack{r \in R - \{\sigma\} \\ n \in N}} \sum_{i \in V} x_{i,r,n} \cdot w_{i,n} \leq b \quad (6)$$

$$\bigwedge_{\substack{i \in V, r \in R \\ (p,n) \in E}} \begin{aligned} (x_{i,r,n} - x_{i,r,p}) + C_{i,r,n} &\geq 0 \\ (x_{i,r,p} - x_{i,r,n}) + C_{i,r,n} &\geq 0 \end{aligned} \quad (7)$$

$$\bigwedge_{\substack{i \in V, r \in R \\ n \in N}} \begin{aligned} x_{i,r,n} &\in \{0, 1\} \\ C_{i,r,n} &\in \{0, 1\} \end{aligned} \quad \bigwedge_{\substack{i \in V \\ n \in N}} \sum_{r \in R} x_{i,r,n} \geq 1 \quad (8)$$

## Extension: Spill Cost

Support rematerialization:

$$\min \sum_{\substack{i \in V \\ n \in N}} S_{i,n} \cdot x_{i,\sigma,n} + \sum_{\substack{i \in V \\ r \in R, n \in N}} \mu_{i,r,n} \cdot c_{i,r,n} \quad (9)$$

Reserve registers for accessing spilled variables:

$$\bigwedge_{n \in N} \sum_{i \in V} a_{i,n} \cdot x_{i,\sigma,n} + \sum_{r \in R - \{\sigma\}} b_{r,n} \leq |R - \{\sigma\}| \quad (10)$$

$$\bigwedge_{\substack{i \in V, r \in R \\ n \in N}} b_{r,n} \cdot w_{i,n} \geq x_{i,r,n} \cdot w_{i,n} \quad \bigwedge_{\substack{r \in R \\ n \in N}} b_{r,n} \in \{0, 1\} \quad (11)$$

# Formulation in AMPL

```

set V; set N; set E within (N cross N); set R; set SIG;
param mu {R,N}; param w {V,N}; param S {V,N}; param b;
var x { i in V, r in R, n in N } binary;
var c { i in V, r in R, n in N } binary;
minimize Cost:
    sum {i in V, n in N, s in SIG} S[i,n] * x[i,s,n]
  + sum {i in V, r in R, n in N} mu[r,n] * c[i,r,n];
subject to eq6 {r in R diff SIG, n in N}:
    sum { i in V } x[i,r,n] * w[i,n] <= b;
subject to eq7a {i in V, r in R, (p,n) in E}:
    (x[i,r,n] - x[i,r,p]) + c[i,r,n] >= 0;
...

```

## Optimizations: No-Move

$$L_n := \left\{ i \in V \mid w_{i,n} \in \{0, b\} \wedge \bigwedge_{d \in \text{pred}(n) \cup \{n\}} S_{i,d} = 0 \right\} \quad (12)$$

$$M := \left\{ n \in N \mid \bigwedge_{p \in \text{pred}(n)} \bigwedge_{\substack{s \in \text{succ}(n) \\ r \in R}} \mu_{r,p} = \mu_{r,n} = \mu_{r,s} \right\} \quad (13)$$

The optimality of the solution computed by the ILP solver is preserved if the constraint

$$\bigwedge_{\substack{r \in R \\ n \in M}} \bigwedge_{\substack{i \in L_n \\ p \in \text{pred}(n)}} x_{i,r,n} = x_{i,r,p} \quad (14)$$

is added.

# Optimizations: Symmetry

Let  $n_S \in N$  be a node where the number of live variables is maximized. Let  $W \subseteq V$  be the set of variables at node  $n_S$  for which  $w_{i,n_S} = b$ . Let  $Q \subseteq R$  be a subset of equivalent registers (that is,  $S_{i,n} = S_{j,n}$  and  $\mu_{i,n} = \mu_{r,n}$  for all  $i, j \in Q$  and  $n \in N$ ). Let  $<_Q$  be a total ordering of the registers and  $<_W$  be a total ordering of the variables in  $W$ . Then, adding the constraint

$$\bigwedge_{\substack{r_1, r_2 \in Q, i_1, i_2 \in W \\ r_1 <_Q r_2, i_1 <_W i_2}} x_{i_1, r_1, n_S} + x_{i_2, r_2, n_S} \leq 1 \quad (15)$$

does not change the value of the optimal solution for the ILP.

# Results: Cost

	Reg.	adpcm	median	mpegcorr	NewLife	MT	Hist
Coloring	4	<b>1225415</b>	91280	92400	2236190	4690	7515
Linear	4	<b>1450425</b>	131217	127913	1752698	7060	106605
Tallam	4	<b>800330</b>	91280	92400	2136180	4690	5160
ILP GCF	4	<b>490124</b>	44710	73850	599642	1919	3773
ILP GCFB	4	<b>330071</b>	44710	73850	599642	1916	2837
Coloring	6	750315	34575	34835	<b>531305</b>	260	1990
Linear	6	1025311	82283	67444	<b>743840</b>	4560	4310
Tallam	6	325230	34575	34835	<b>531305</b>	260	1195
ILP GCF	6	270084	<b>17795</b>	28550	<b>251428</b>	105	794
ILP GCFB	6	120045	<b>17795</b>	28550	<b>251428</b>	105	6
Coloring	8	275215	<b>17870</b>	8055	27915	0	0
Linear	8	575214	72248	38415	218790	0	0
Tallam	8	130	<b>17870</b>	8055	27915	0	0
ILP GCF	8	120054	6452	1062	11404	0	0
ILP GCFB	8	42	6452	1062	11404	0	0

## Results: Time

Bitwise	adpcm	median	mpegcorr	NewLife	MT	Hist
4 registers	53257s	<b>73s</b>	10s	<b>57s</b>	2s	<b>6s</b>
6 registers	$\geq 10^5s$	<b>44s</b>	35s	<b>163s</b>	3s	<b>11s</b>
8 registers	454s	<b>80s</b>	46s	<b>312s</b>	1s	<b>6s</b>
Word-wise	adpcm	median	mpegcorr	NewLife	MT	Hist
4 registers	$\geq 10^5s$	<b>23s</b>	10s	42s	2s	5s
6 registers	<b>331s</b>	<b>39s</b>	27s	168s	3s	11s
8 registers	4162s	<b>80s</b>	43s	286s	1s	6s

# Results: Scalability

Benchmark	$ V $	$ E $	$ N $	time (GCFB)
mpegcorr	31	185	178	10s
levdurb	37	206	199	24s
NewLife	<b>61</b>	<b>312</b>	<b>302</b>	57s
median	34	190	184	73s
adpcm	29	228	218	53257s
bilint	54	200	200	$> 10^6$ s

# Future Work

- Full implementation for real architecture
- Need to understand characteristics that cause slow-down!
- Heuristic simplifications of ILP input
- Better starting solutions
- Combination with instruction selection
- Specialized ILP solver

# Conclusion

- ILP allows for clean formulation of problem  
⇒ applications in certified compilation
- Huge theoretical gap in performance between other algorithms and optimal solution
- Subword access savings  $\approx$  0-2 registers
- ILP run-time so far unpredictable
- Feasible, especially for improving solutions

# RTFL

Copyright (C) 2006 Christian Grothoff

Verbatim copying and distribution of this entire article is permitted in any medium, provided this notice is preserved.