7th International Workshop on Software and Compilers for
Embedded Systems - SCOPES 2003

Improving Offset Assignment through Simultaneous Variable Coalescing

Desiree Ottoni, Guido Araujo
{desiree.ottoni | guido}@ic.unicamp.br
Guilherme Ottoni
ottoni@cs.princeton.edu
Rainer Leupers
leupers@iss.rwth-aachen.de

Overview

- Motivation
- Overview of Offset Assignment Problem
- Coalescing Simple Offset Assignment Technique
- Example of using CSOA Technique
- Experimental Results
- Conclusions and Future Work

Motivation

Stack	a	c	d	b	e

- Code size constraints
- Address computation is expensive
- ***Indirect addressing*** is suitable to embedded processors:
  - Implements ***fast address computation***
  - Enables the design of ***short instructions***
- ***Post-increment/decrement*** addressing modes

Offset Assignment Problem

- **Simple Offset Assignment Problem**: Almost all solutions are based in MWPC [Bartley'92, Liao et al'96, Leupers et al'96, Xiaotong'03, ...].

- **General Offset Assignment Problem**: Almost all solutions try to partition the variables and then resolves each partition as SOA problem [Liao et al'96, Leupers et al'96, Xiaotong'03, ...].

Coalescing based Simple Offset Assignment

- Based in the MWPC solution;
- At each step, chooses between:

  (1) insert the maximal weight edge in the path; or

  (2) coalesce two variables

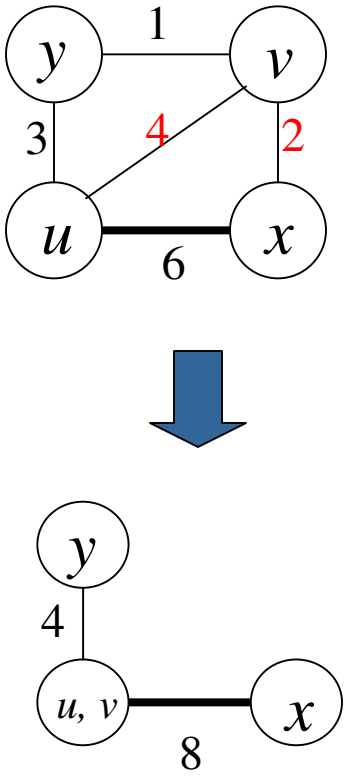
Coalescing based Simple Offset Assignment

- **Conditions to coalesce two variables *u* and *v*:**
  - $(u, v) \notin$ Interference Graph;
  - Coalesce u and *v* does not create a cycle, considering only the selected edges;
  - Coalesce u and *v* does not cause the new coalesced vertex to have degree greater than two, considering only the selected edges.

Coalesce-based Simple Offset Assignment

How to calculate the savings in offset cost when coalescing two variables *u* and *v*:

- For each $x \in (Adj_{sel}(u) - Adj_{sel}(v))$, add the weight of the edge *(x, v)* to the cost;
- For each $x \in (Adj_{sel}(v) - Adj_{sel}(u))$, add the weight of the edge *(x, u)* to the cost;
- Add the weight of the edge *(u, v)* to the cost, if this edge was not selected yet.



Offset Cost saved: 6

Coalescing based Simple Offset Assignment

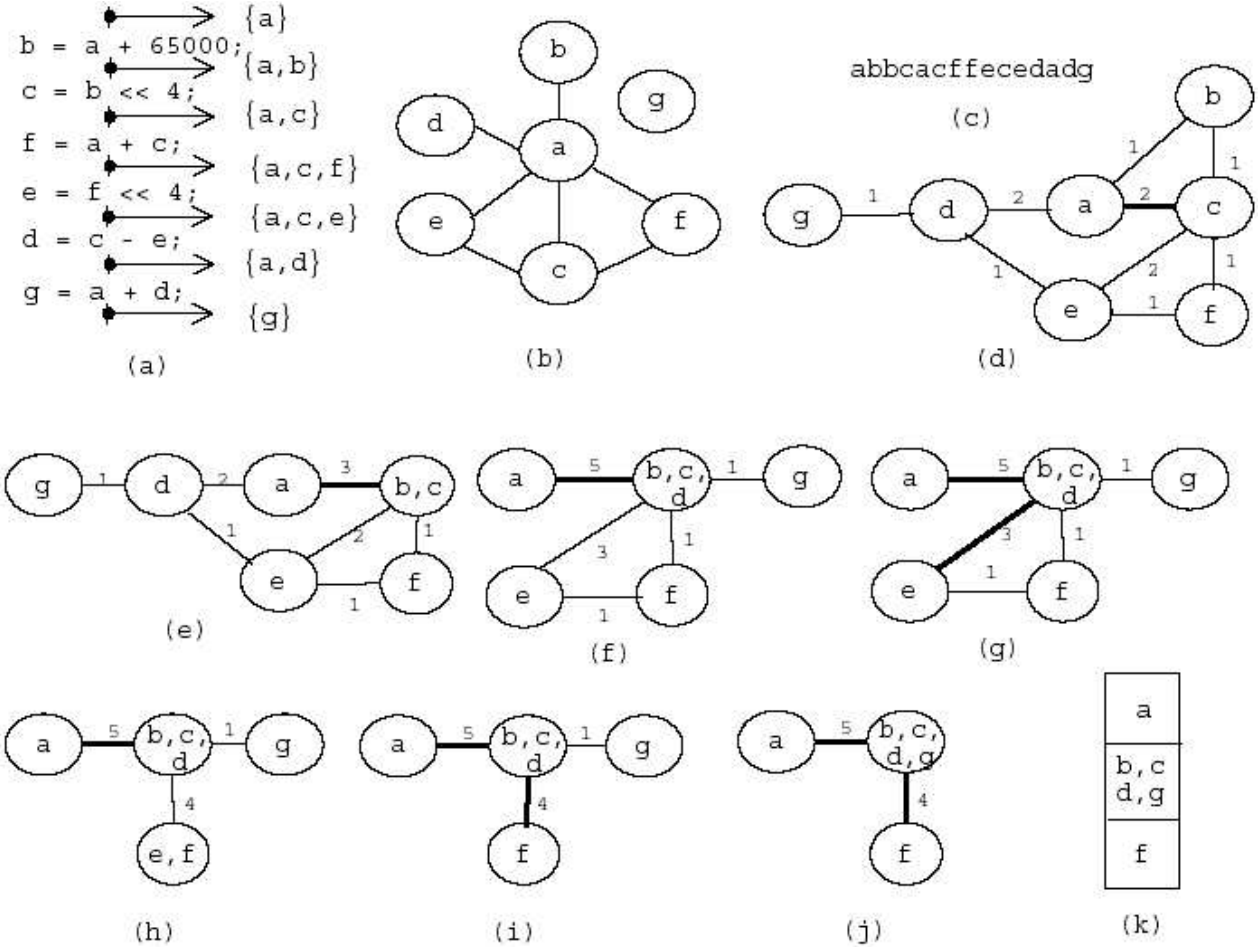
Algorithm 1 Coalescing-Based SOA

```
(1)  G_A(V_A, E_A) ← BuildAccessGraph(L_AS);
(2)  L = sorted list of the E_A;
(3)  coal ← false;
(4)  sel ← false;
(5)  repeat
(6)      rebuild ← false;
(7)      (coal, u, v, csave) ← FindCandidatePair(G_I, u, v);
(8)      sel ← FindEdgeValidNotSel(L, e);
(9)      if (coal && sel )
(10)         if (csave ≥ w(e))
(11)             rebuild ← true;
(12)         else
(13)             mark e as selected;
(14)     else
(15)         if (coal)
(16)             rebuild ← true;
(17)         else if (sel)
(18)             mark e as selected;
(19)     if (rebuild)
(20)         RebuildAccessGraph(G_A, u, v);
(21)         RebuildInterferenceGraph(G_I, u, v);
(22)         RebuildL(L);
(23) until (!(coal || sel))
(24) return BuildOffset(G_A);
```

Input: the access sequence L_{AS},
the interference graph $G_I(V_I, E_I)$.
Output: the offset assignment.

Coalescing based Simple Offset Assignment

- **Example of algorithm execution**



Experimental Results – Offset Costs (relative SOA-Liao costs [Liao et al'96])

- Implementation using LANCE and OffsetStone:
  - TB [Leupers et al'1996]
  - GA [Leupers et al'1998]
  - INC-TB [Leupers'2003]
  - SOA-Color [Ottoni et al'03]
  - CSOA [Ottoni et al'03]

Benchmarks	TB	GA	INC-TB	SOA-Color	CSOA
adpcm	89.1%	89.1%	89.1%	55.8%	45.6%
epic	96.8%	96.6%	96.6%	74.3%	50.2%
g721	96.2%	96.2%	96.2%	50.6%	27.9%
gsm	96.3%	96.3%	96.3%	26.6%	19.4%
jpeg	96.9%	96.7%	96.7%	52.6%	32.2%
mpeg2	97.3%	97.1%	97.2%	60.2%	34.3%
pegwit	91.1%	90.7%	90.7%	75.2%	38.8%
pgp	94.9%	94.8%	94.8%	55.0%	32.2%
rasta	98.6%	98.5%	98.5%	33.2%	21.1%
Average	95.2%	95.1%	95.1%	51.2%	32.1%

Experimental Results – *Memory Savings*

Benchmarks	SOA-Color			CSOA		
	Code	Data	Code+Data	Code	Data	Code+Data
adpcm	89.9%	15.7%	71.5%	87.5%	27.8%	72.7%
epic	92.0%	11.6%	74.1%	84.6%	27.0%	71.8%
g721	90.7%	13.3%	68.0%	86.4%	25.1%	68.5%
gsm	94.3%	7.0%	71.9%	93.7%	21.8%	75.3%
jpeg	94.9%	14.6%	81.0%	92.7%	34.9%	82.7%
mpeg2	92.8%	12.5%	75.1%	88.0%	31.9%	75.7%
pegwit	97.4%	9.7%	78.2%	93.6%	35.3%	80.8%
pgp	99.6%	12.8%	98.3%	99.4%	31.6%	98.4%
rasta	84.4%	9.9%	71.2%	81.6%	26.1%	71.8%
Average	92.8%	11.6%	76.2%	89.6%	28.7%	77.1%

• Memory Savings relative to memory used by Liao's algorithm.

Conclusions and Future Works

Conclusions:

- Contrary to what Liao suspected in his thesis (MIT'96), coalescing can improve OA.

- CSOA heuristic produces good results, dramatically reducing the OA cost.

Future Work:

- CSOA can be used to resolve GOA problem.