Minimizing Bank Selection Instructions for Partitioned Memory Architectures

Bernhard Scholz[1], Bernd Burgstaller[1], and Jingling Xue[2]

[1]The University of Sydney

[2]University of New South Wales

# Motivation

Huge market for microcontrollers (μCs):

- SIA forecast: 8-bit μC market reaches \$5.32 billion in 2006

- PIC Microchip sold 1 billion units between 2004 and 2005

μCs neglected by research community as "assembly language playground"

Productivity considerations require move to high-level languages

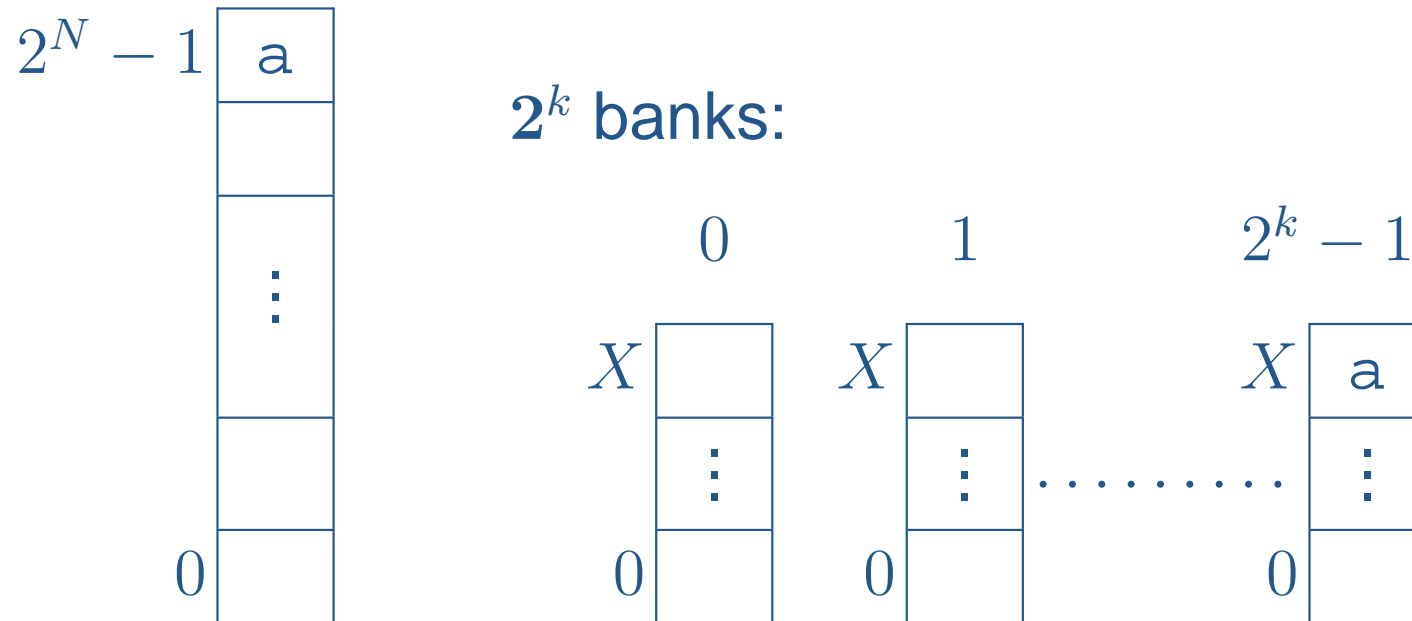Compiler optimizations for μCs required

Partitioned what???

$2^N - 1$ │ a │

⋮

0 │ │
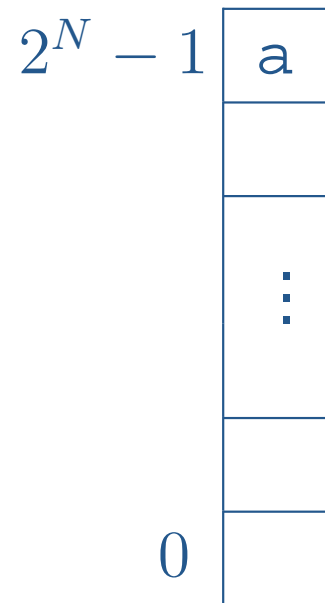
instruction encoding:

OPCODE	f

MOVWF   f

Operand: $0 \leq \mathtt{f} \leq 2^N - 1$

Operation: $(W) \rightarrow \mathtt{f}$

- effective address $\mathtt{f}$ encoded in N bits

- large instruction word sizes increase code size

Partitioned Address Space

$2^N - 1$ | a

$\vdots$

0

2^k banks:

0 1 $2^k - 1$

X X X | a

$\vdots$ $\vdots$ $\cdots\cdots$ $\vdots$

0 0 0

- requires *address disambiguation mechanism*
- reduces effective address to $N - k$ bits
- increases memory space without increasing address bus
- reduces instruction word size and memory footprint

Address Disambiguation

$$2^N - 1 \quad \boxed{\textbf{a}}$$

$$\vdots$$

$$0$$

0   1   7

X X X $\boxed{\textbf{a}}$

$\vdots$ $\vdots$ $\cdots\cdots$ $\vdots$
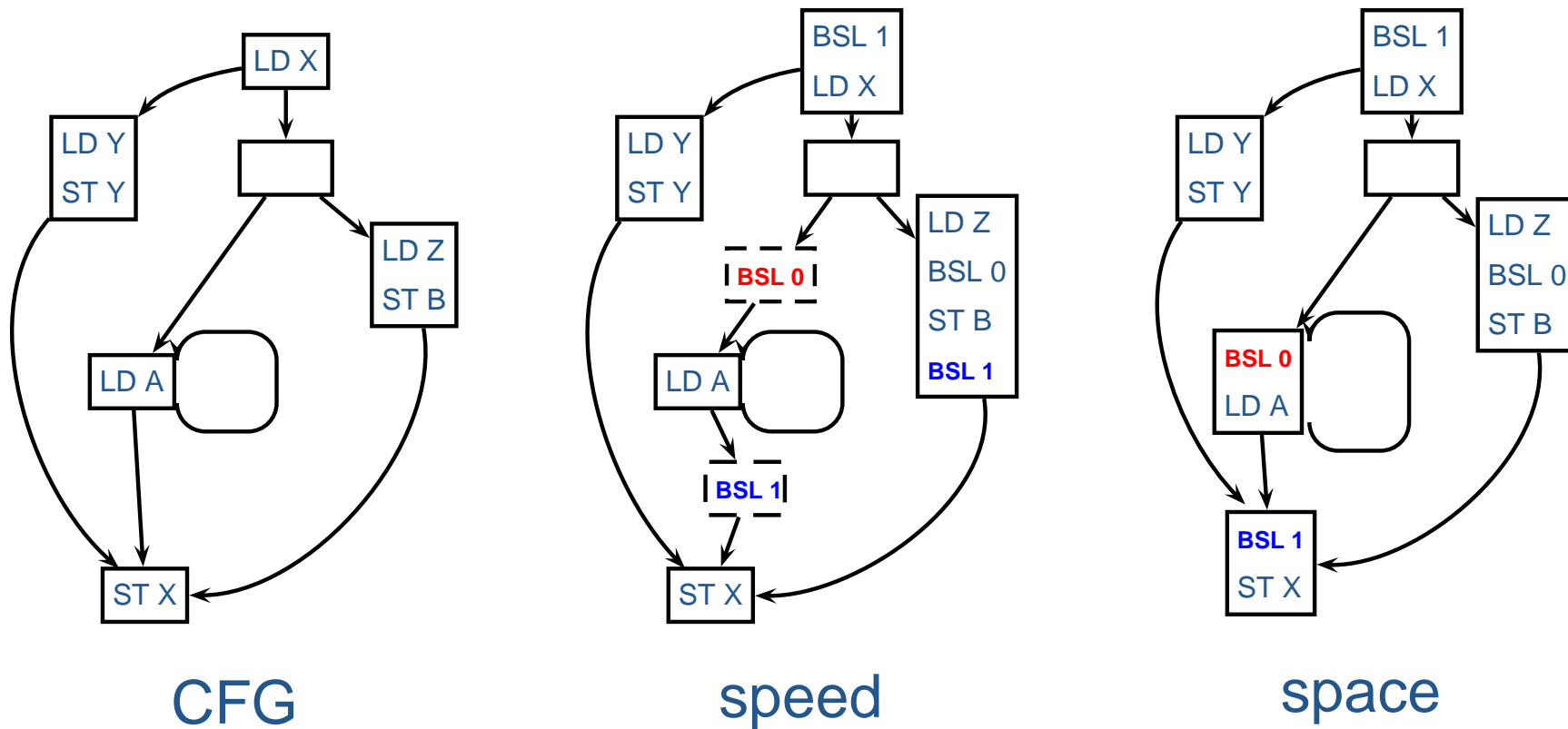
0   0   0
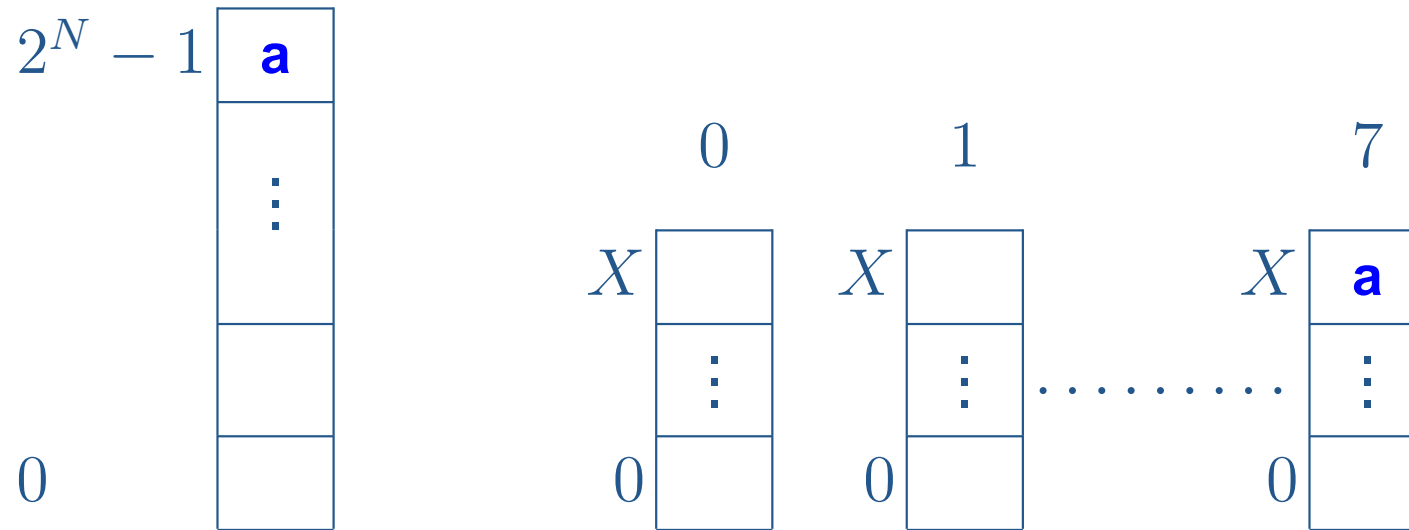
- CPU can access one bank at a time

- active bank stored in CPU bank register

- bank selection instruction `BSL <nr>` switches banks

- access of entity **a**:   `BSL 7`

                          `MOVWF` X

Bank Selection Instruction Placement

bank 0: A,B

bank 1: X,Y,Z

CFG   speed   space

Bank Switching in Prevalent μC Architectures

Used for data, code, and CPU registers

- Motorola 68HC11
- PIC 12*, 14* and 16* families
- Intel 8051 family
- Ubicom SX, Toshiba T900
- partitioned SRAMs for WSNs

Significant potential for compiler improvement

- to reduce bank switching overhead

Problem Statement

Minimize number of bank selection instructions of a program for

- a static allocation of variables to banks

- a cost metric (speed, space, etc.)

- a fixed instruction schedule (no instruction re-ordering).

Outline of Solution

Optimization within basic blocks

Intra-procedural optimization

- modelled as a discrete optimization problem

Extension to the interprocedural case

Mapping of the discrete optimization problem to a solver

Basic Block Optimization

Performs linear scan of a basic block,
memorizing the state of the bank selection register

Inserts bank selection statements except

- before the first bank-sensitive statement

- after the last bank-sensitive statement

Example:

```
Input:  1 LD (bank 1) X      Output:  1 LD X            b?
        2 CALL foo                     2 CALL foo        b?
        3 LD (bank 0) A                3 BSL 0; LD A     0
        4 ST (bank 1) Y                4 BSL 1; ST Y     1
        5 ST (bank 1) Z                5 ST Z            1
```
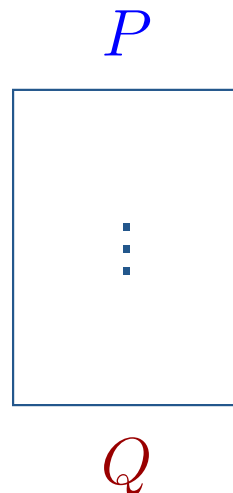
Intra-Procedural Optimization

Introduce controlling variables P and Q for each basic block
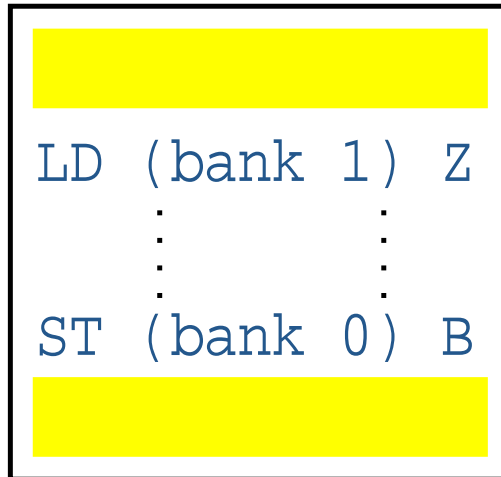
Domain: $\mathbb{D} = \{0, \ldots, 2^k - 1, b_?\}$

- P specifies bank register state upon entry (guarantee)

- Q specifies bank register state upon exit (obligation)
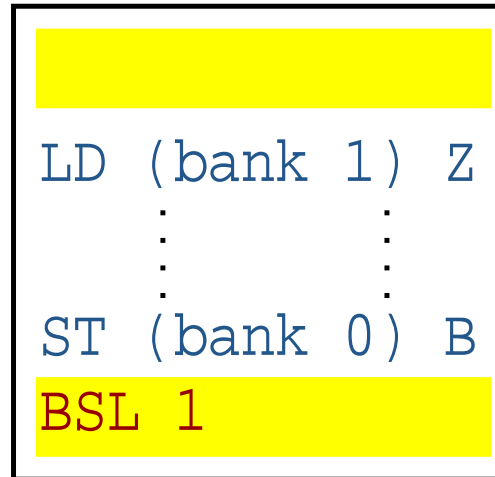
Discrete optimization "attaches" to controlling variables

$$P$$

$$\vdots$$

$$Q$$

Controlling Variables on Bank-Sensitive Basic Blocks

$$P$$

```
LD (bank 1) Z
    .        .
    .        .
    .        .
ST (bank 0) B
```

$$P = 1$$

```
LD (bank 1) Z
    .        .
    .        .
    .        .
ST (bank 0) B
BSL 1
```

$$P = 0$$

```
BSL 1
LD (bank 1) Z
    .        .
    .        .
    .        .
ST (bank 0) B
BSL 1
```
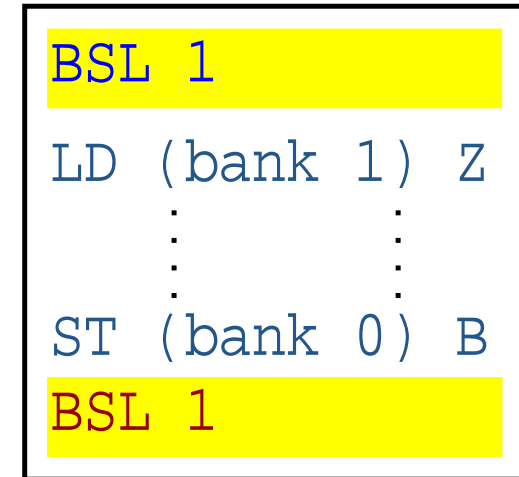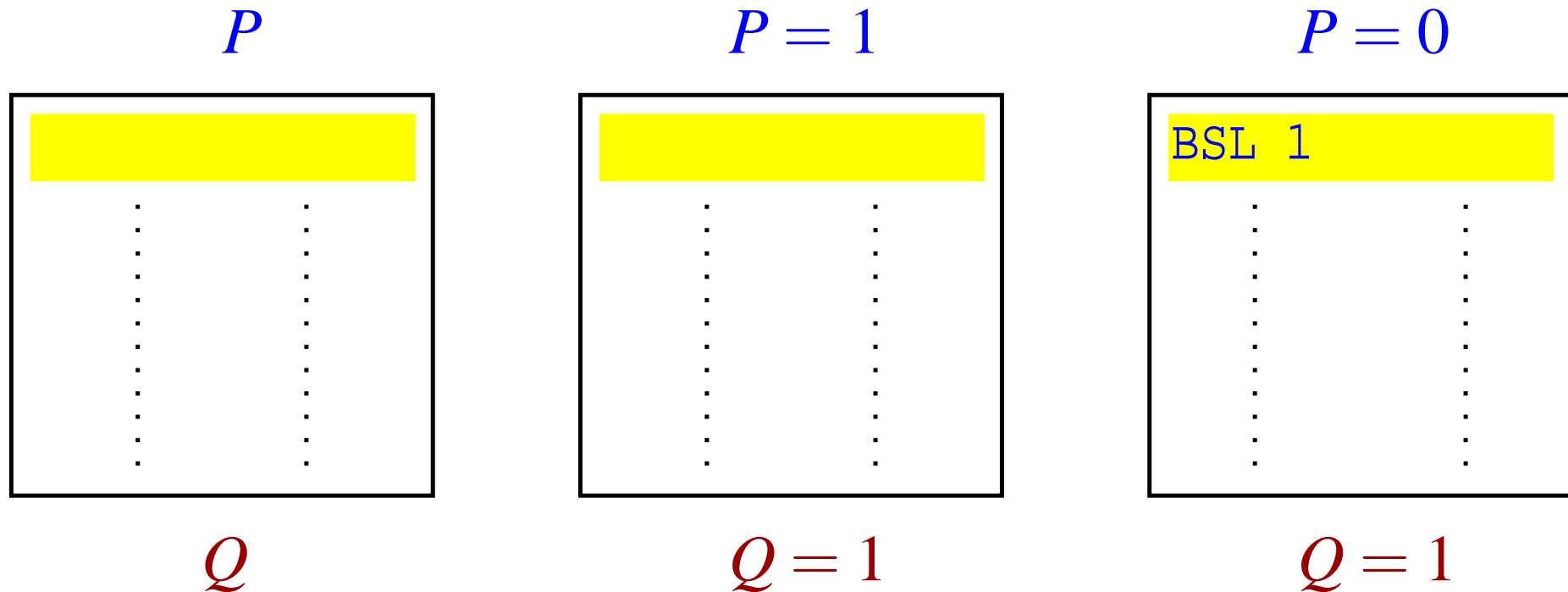
$$Q \qquad\qquad Q = 1 \qquad\qquad Q = 1$$

Cost function *n-cost*(P)

- accounts for the costs at the entry of the basic block

Cost function *e-cost*(Q)

- accounts for the costs at the exit of the basic block

Controlling Variables on Transparent Basic Blocks

P $\qquad\qquad$ $P = 1$ $\qquad\qquad$ $P = 0$

Q $\qquad\qquad$ $Q = 1$ $\qquad\qquad$ $Q = 1$
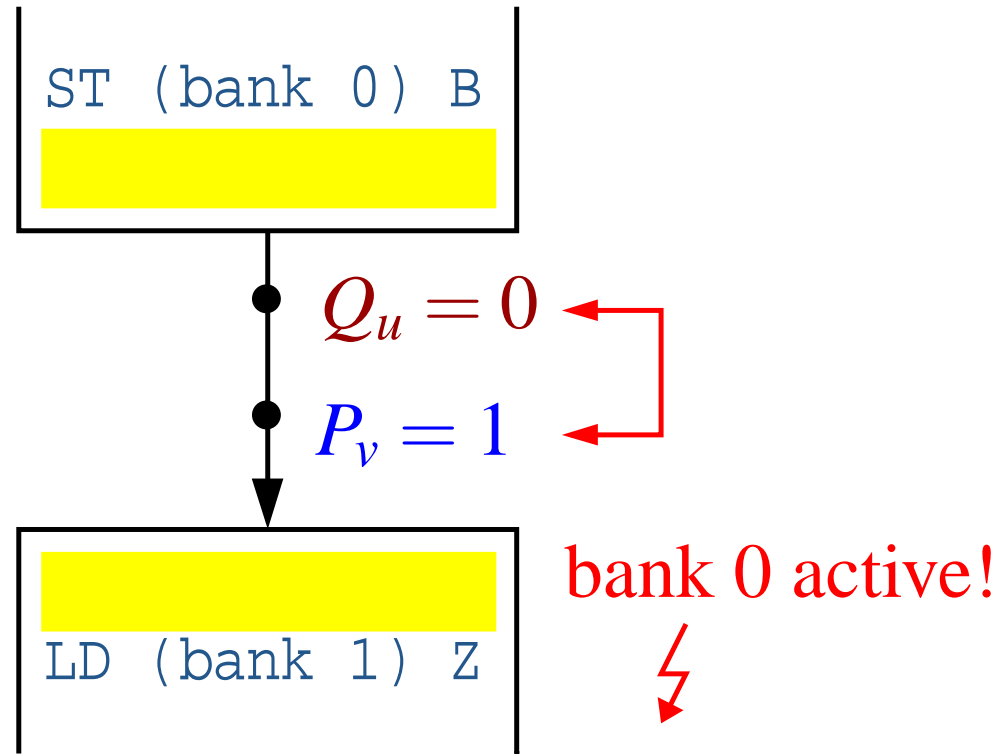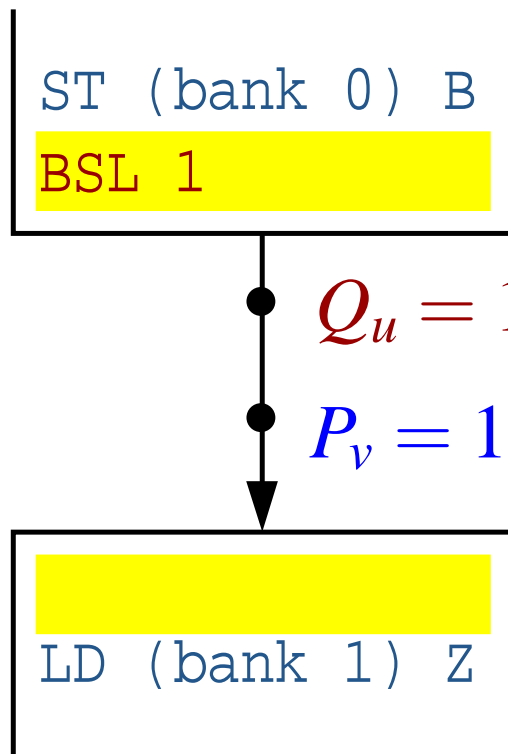
Cost function *t-cost*(P, Q)

- accounts for the costs at a transparent basic block

Cost functions can model arbitrary costs (speed, space, power consumption, ...)

Correctness Criteria for Intra-Procedural Optimization

"adjacent controlling variables have to agree":

$$\text{forall edges (u,v): } (P_v \neq b_?) \Rightarrow (Q_u = P_v)$$

```
ST (bank 0) B
BSL 1
```

$Q_u = 1$

$P_v = 1$

```
LD (bank 1) Z
```

```
ST (bank 0) B
```

$Q_u = 0$

$P_v = 1$

```
LD (bank 1) Z
```

bank 0 active!

The Bank Selection Placement Optimization Problem

Objective function f accounts for the costs of basic blocks

Controlling variables determine costs of transformations

$$\text{s.t. } \forall u \in V : P_u, Q_u \in \mathbb{D}$$

$$P_s = b_?$$

$$\forall (u, v) \in E : (P_v \neq b_?) \Rightarrow (Q_u = P_v)$$

$$\min f = \sum_{u \in V} \textit{cost}_u(P_u, Q_u)$$

$$= \underbrace{\sum_{u \in S} \textit{n-cost}_u(P_u) + \sum_{u \in S} \textit{e-cost}_u(Q_u)}_{\text{bank-sensitive}} + \underbrace{\sum_{u \in T} \textit{t-cost}_u(P_u, Q_u)}_{\text{transparent}}$$
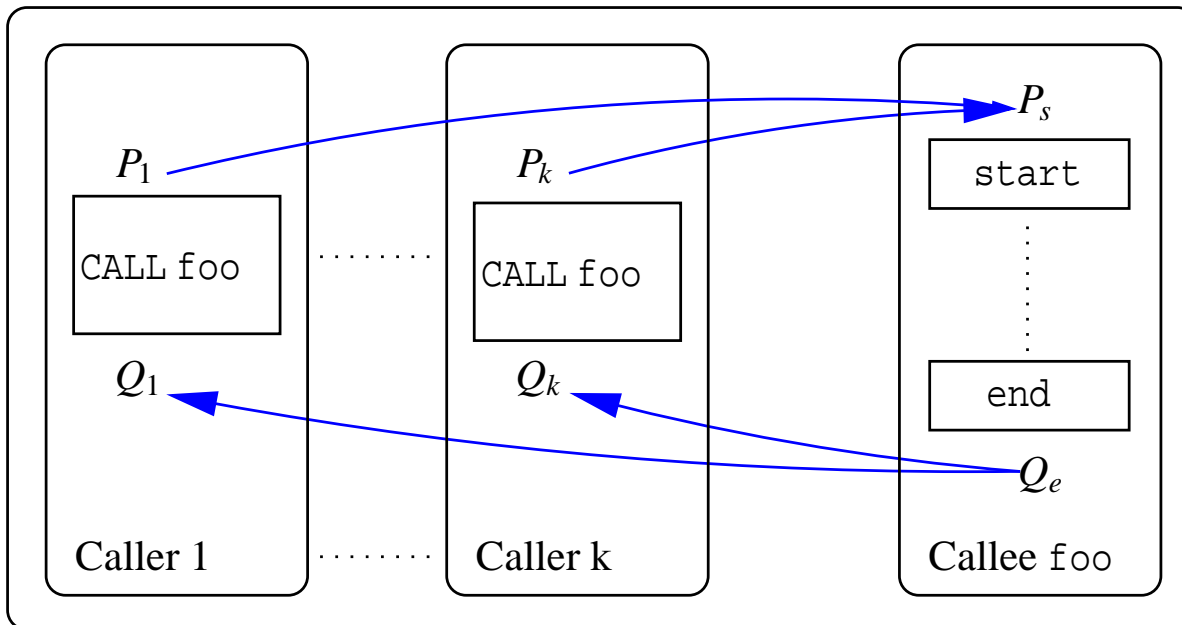
Interprocedural Extension

Bank selection instructions can be hoisted across call sites

Additional placement of `BSL` instructions:

- at the entry/exit of a subroutine, and before a call

Caller/Callee correctness constraints:

$$(P_s \neq b_?) \Rightarrow (P_s = P_i)$$

$$(Q_i \neq b_?) \Rightarrow (Q_i = Q_e)$$

- one discrete optimization problem for the whole program

Experimental Results

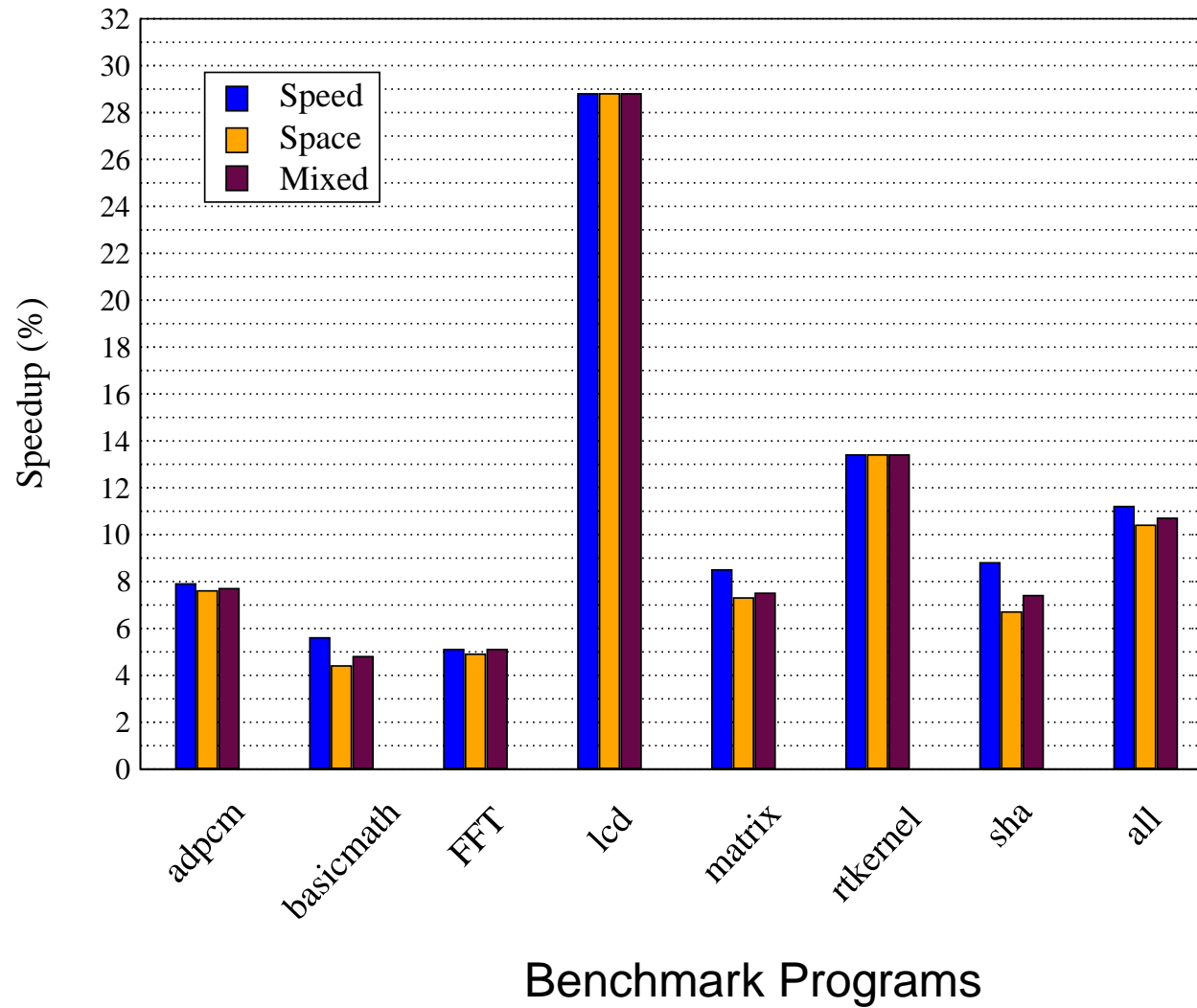Implementation of optimizations for a PIC16F877A μC

- RISC-based Harvard architecture

- 8-bit data bus

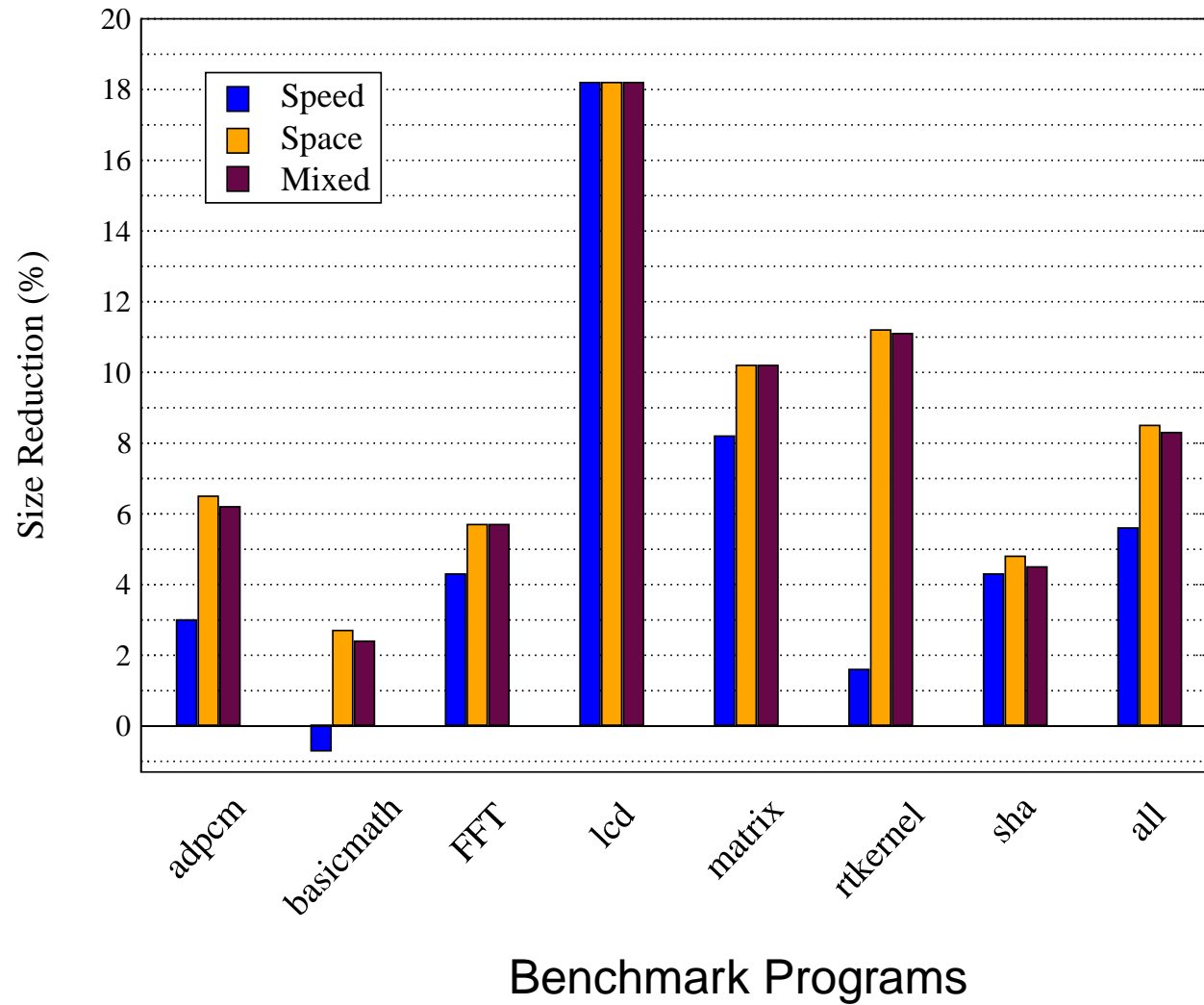- 4 data banks, 368 bytes data memory

Evaluated optimizations for

- MiBench (as applicable),

- DSPStone (as applicable),

- a μC real-time kernel, and

- μC driver routines

Reference point: HI-TECH PICC high-performance C-compiler

Experimental Results (speedup)

# Experimental Results (code size reduction)

Optimally solved procedures, all of MiBench

	Benchmark	proc	CFG node	CFG edge	CFG (simpl.) node	CFG (simpl.) edge	PBQP P,Q	optproc	
autom.	basicmath	6	97	143	174	220	348	6	100.0%
	bitcount	15	88	102	124	138	248	13	86.7%
	qsort	4	40	55	59	74	118	3	75.0%
	susan	19	698	1102	1104	1509	2208	11	57.9%
consum.	jpeg	375	6998	10255	10948	14211	21896	262	69.9%
	lame	213	5720	8510	8735	11528	17470	148	69.5%
	mad	1	3	2	3	2	6	1	100.0%
	tiff	510	10735	15668	15678	20628	31356	320	62.7%
	typeset	399	20650	31675	31322	42348	62644	251	62.9%
nw	dijkstra	12	138	184	198	246	396	10	83.3%
	patricia	5	160	227	217	285	434	2	40.0%
office	ghostscript	3551	47967	65967	67117	85120	134234	2734	77.0%
	ispell	107	2368	3611	3757	5001	7514	51	47.7%
	rsynth	53	1326	2012	2074	2760	4148	32	60.4%
	sphinx	684	10597	14895	15732	20043	31464	522	76.3%
	stringsearch	13	176	240	262	326	524	7	53.8%
secur.	blowfish	14	237	344	373	483	746	13	92.9%
	pgp	320	7381	10959	11207	14788	22414	195	60.9%
	rijndael	7	157	223	220	286	440	2	28.6%
	sha	7	56	70	78	92	156	7	100.0%
telec.	adpcm	5	79	105	103	132	206	3	60.0%
	CRC32	4	32	41	47	56	94	3	75.0%
	FFT	6	83	114	124	156	248	6	100.0%
	gsm	65	1589	2244	2097	2754	4194	44	67.7%
	all	6395	117375	168748	171753	223186	343506	4646	72.7%

Solve times wrt. the number of banks, all of MiBench
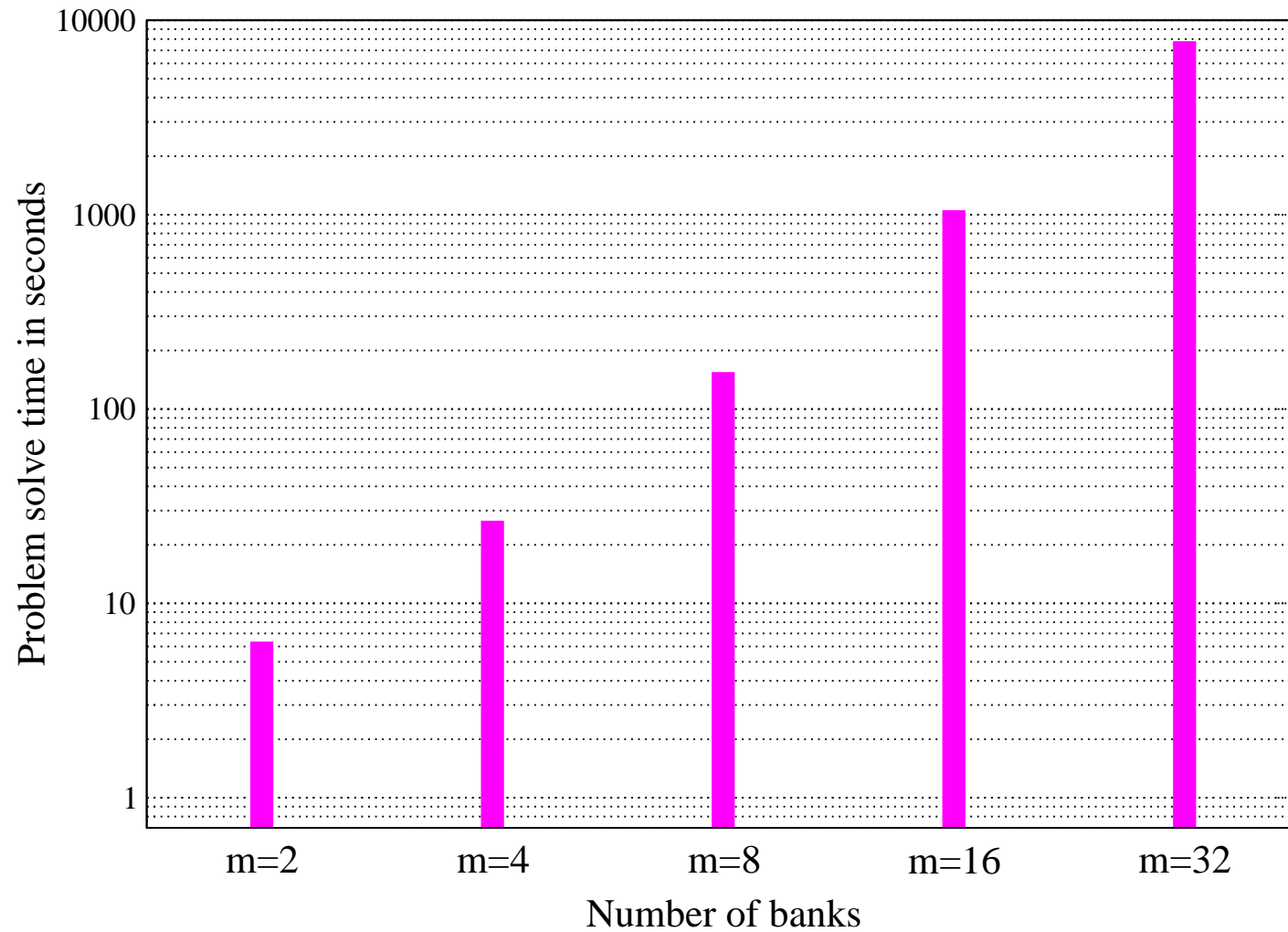
Solve times wrt. the number of CFG edges, all of MiBench
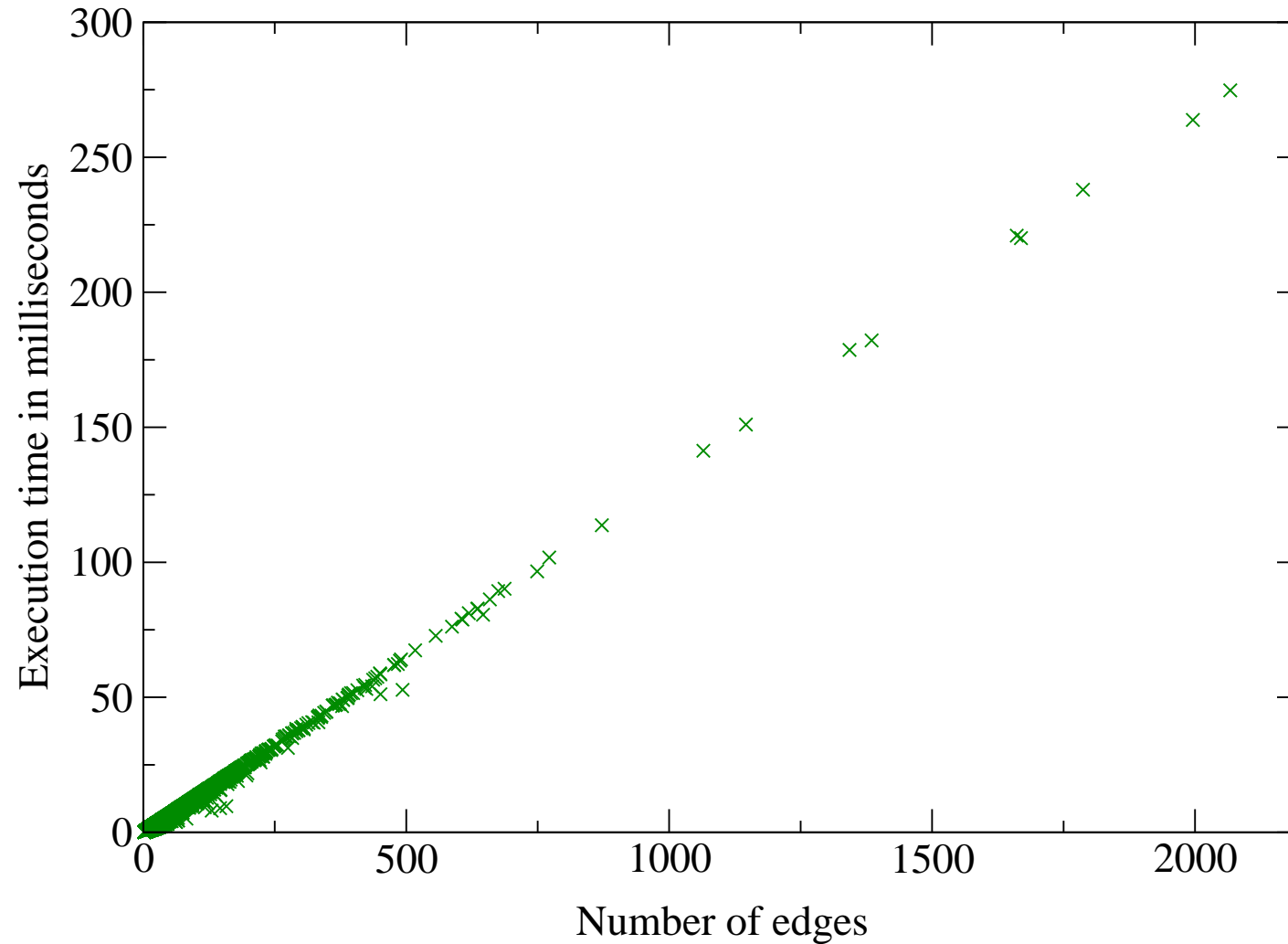
Summary

Devised algorithm to minimize bank selection instructions for

- a given instruction order

- and a given data partitioning

Formulated bank selection instruction placement as a discrete optimization problem

Optimization objectives are formulated as cost metrics

Experimental results for PIC16F877A μ-controller:

- code size reduction between 2.7% and 18.2%

- speedup between 5.1% and 28.8%

- 100% of PIC benchmarks solved optimally, 72% of MiBench

Toolchain