Code Density Concerns for New Architectures

Vincent M. Weaver
Cornell University

Sally A. McKee
Chalmers University of Technology

7 October 2009

Introduction

- Benchmark ported to 21 different assembly languages
- Hand-optimized for minimum size
- Code tested and works on all architectures

What ISA features lead to high code density?

Can this help designers of new ISAs?

New ISAs? Really?

- ISA design still a concern
- FPGAs make it easy
- Embedded architectures want dense code
- Linux has 12 embedded architectures and counting

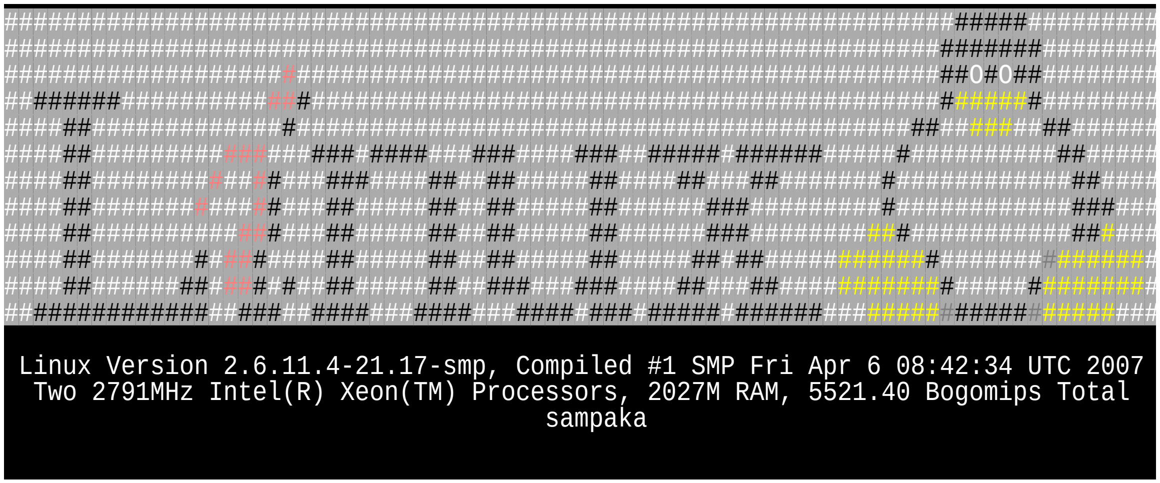
Benefits of Code Density

- L1 iCache holds more instructions
- More data fits in unified L2 cache
- Less bandwidth required to memory and disk
- Fewer TLB misses
- Compact loops can be executed from instruction buffer
- Smaller cache footprint can lead to energy savings

What about Performance?

- Hard to optimize performance
- Varies across implementations
- Dense code often performs well

The Benchmark



LZSS decompression

System calls, including disk I/O

String manipulation and search

Integer to ASCII conversion

ISA Categories

- VLIW
- CISC
- RISC
- Embedded
- 8/16-bit

VLIW Processors

ia64

- 16-byte bundle holds 3 instructions
- Instruction has 3 arguments
- Hundreds of integer registers
- Predication

CISC Processors

m68k, s390, VAX, x86, x86_64

- Variable instruction length, 1-54 bytes
- Instruction has 2 arguments
- 16 integer registers (x86 has 8)
- Status flags
- Unaligned loads
- Complex addressing modes

RISC Processors

Alpha, ARM, m88k, microblaze, MIPS, PA-RISC, PPC, SPARC

- 4 byte instruction length
- Instruction has 3 arguments
- 32 integer registers (except ARM, SPARC)
- Most have a zero register
- Many have branch delay slot

Embedded Processors

avr32, crisv32, sh3, ARM Thumb

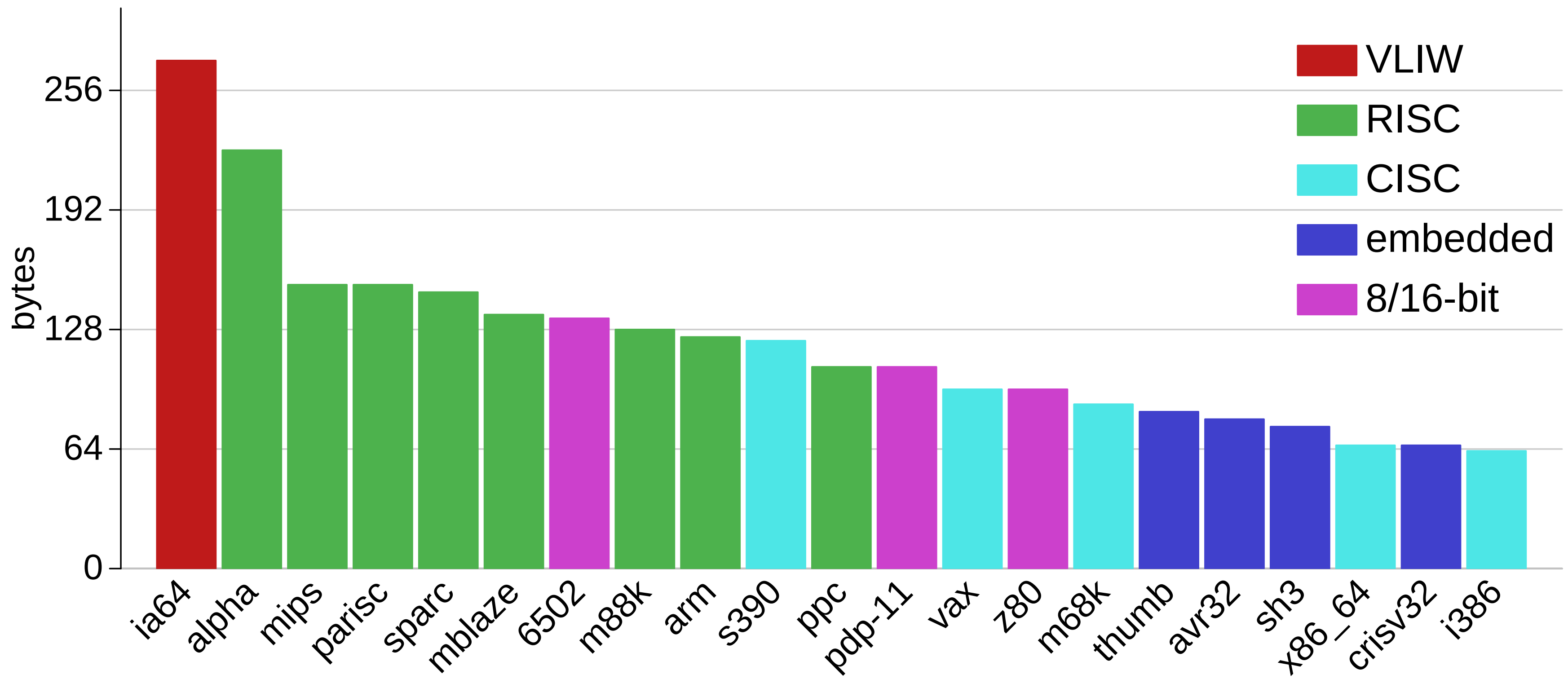
- 2 byte instruction length
- Instruction has 2 arguments
- Most have 16 integer registers
- Auto-incrementing loads
- Status flags

8/16-bit Processors

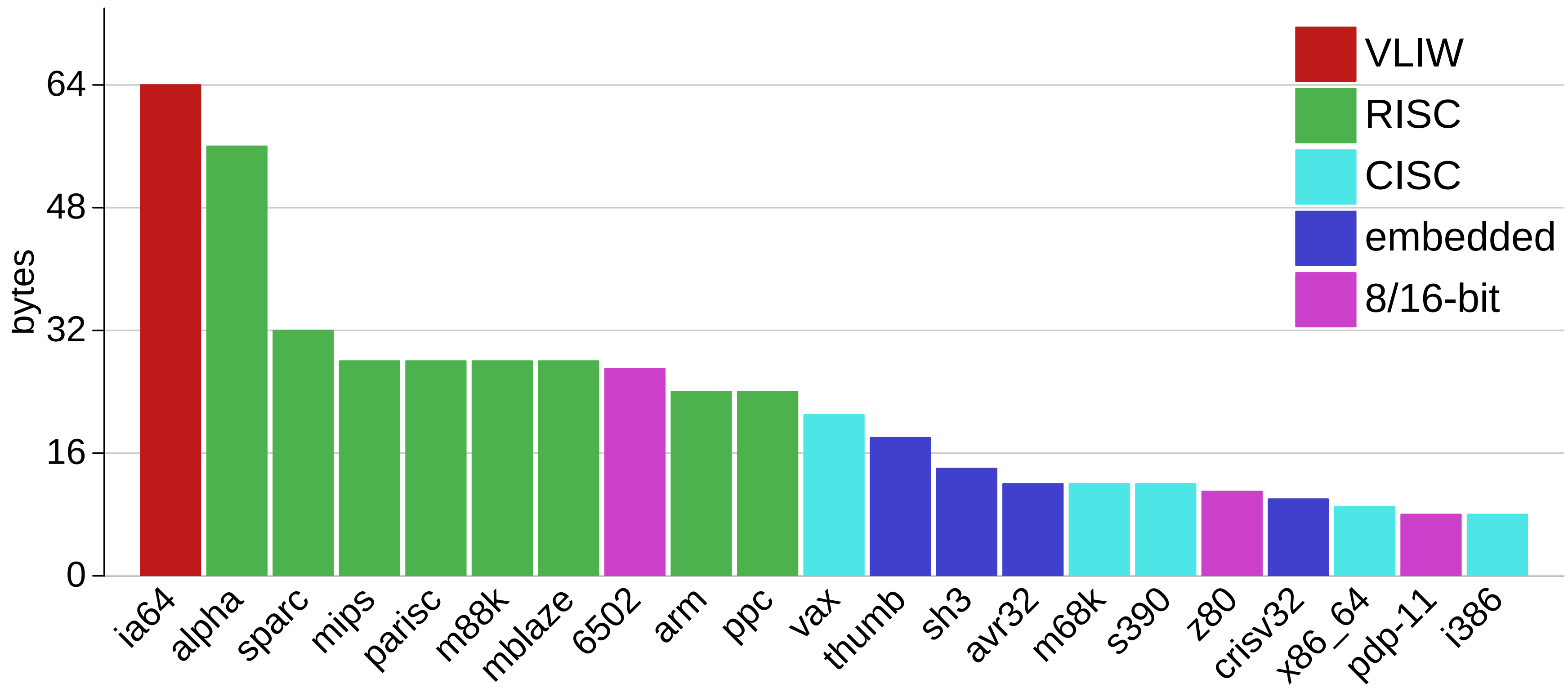
6502, PDP-11, z80

- Variable instruction length (1-6 bytes)
- Instruction has 1-2 arguments
- Status flags

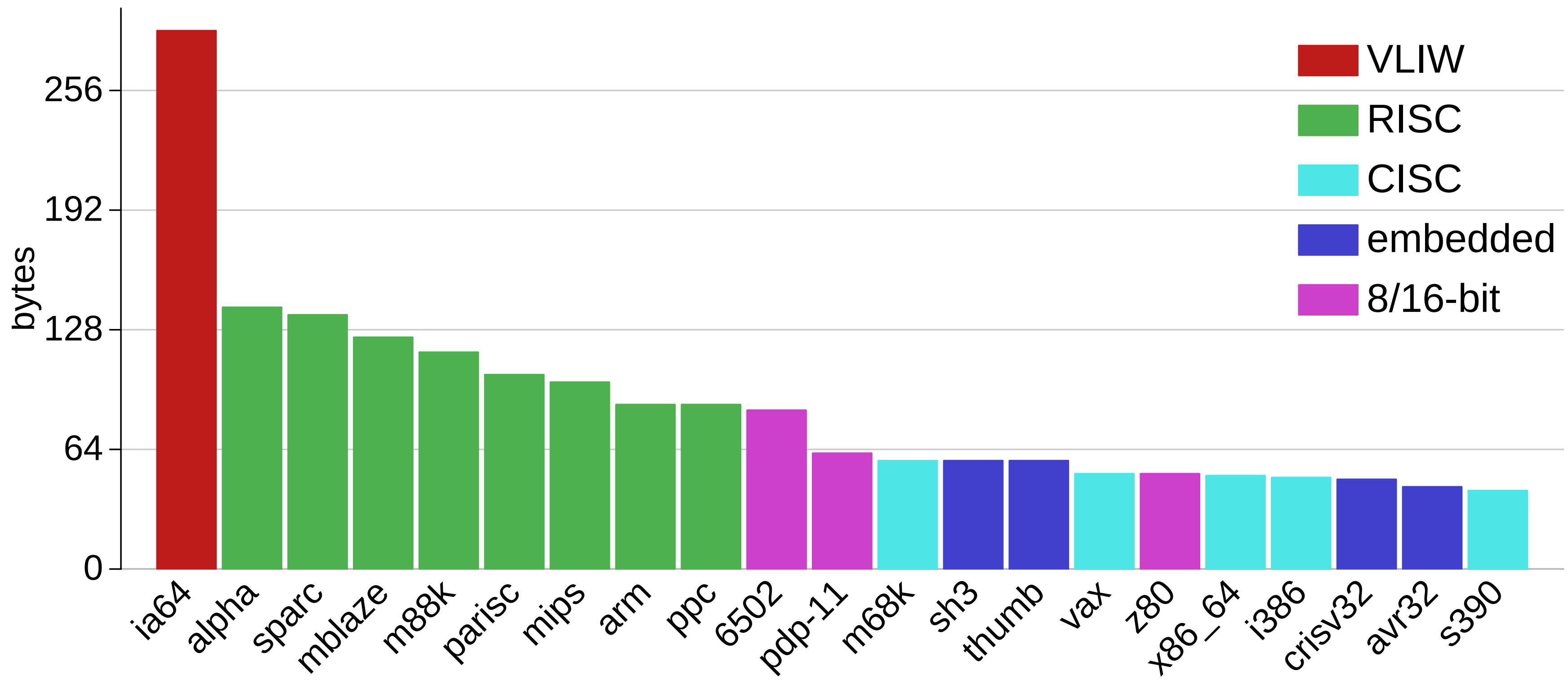
Results – LZSS Decompression



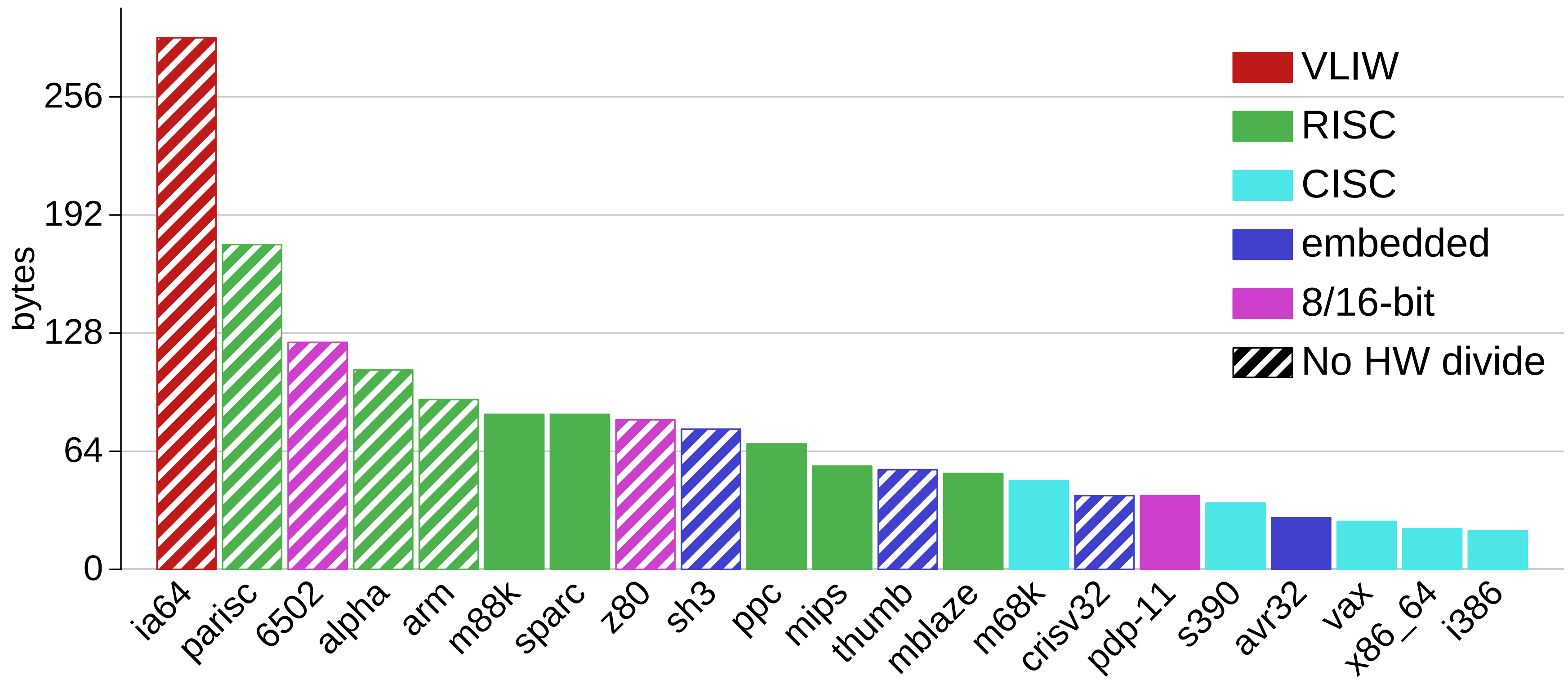
Results – String Concatenation



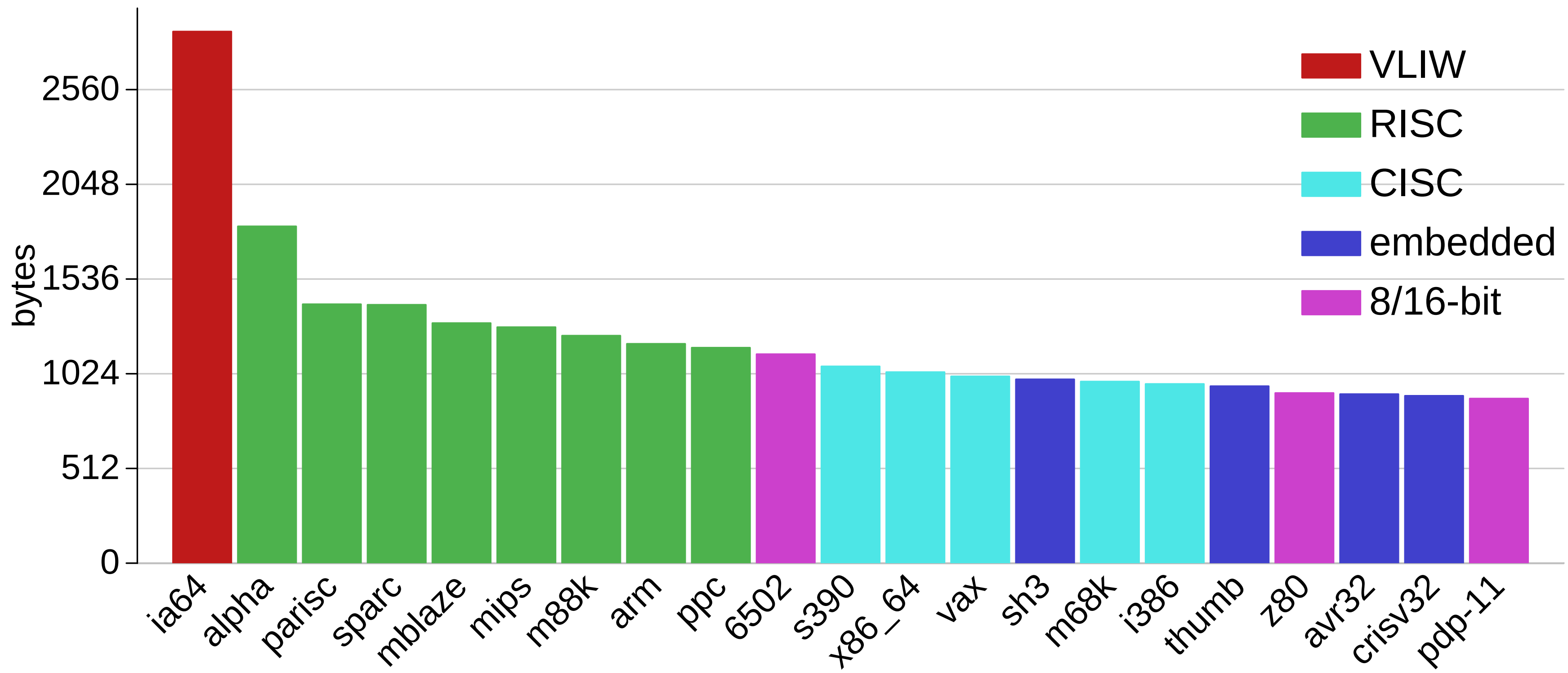
Results – String Search



Results – Integer → ASCII



Results – Overall



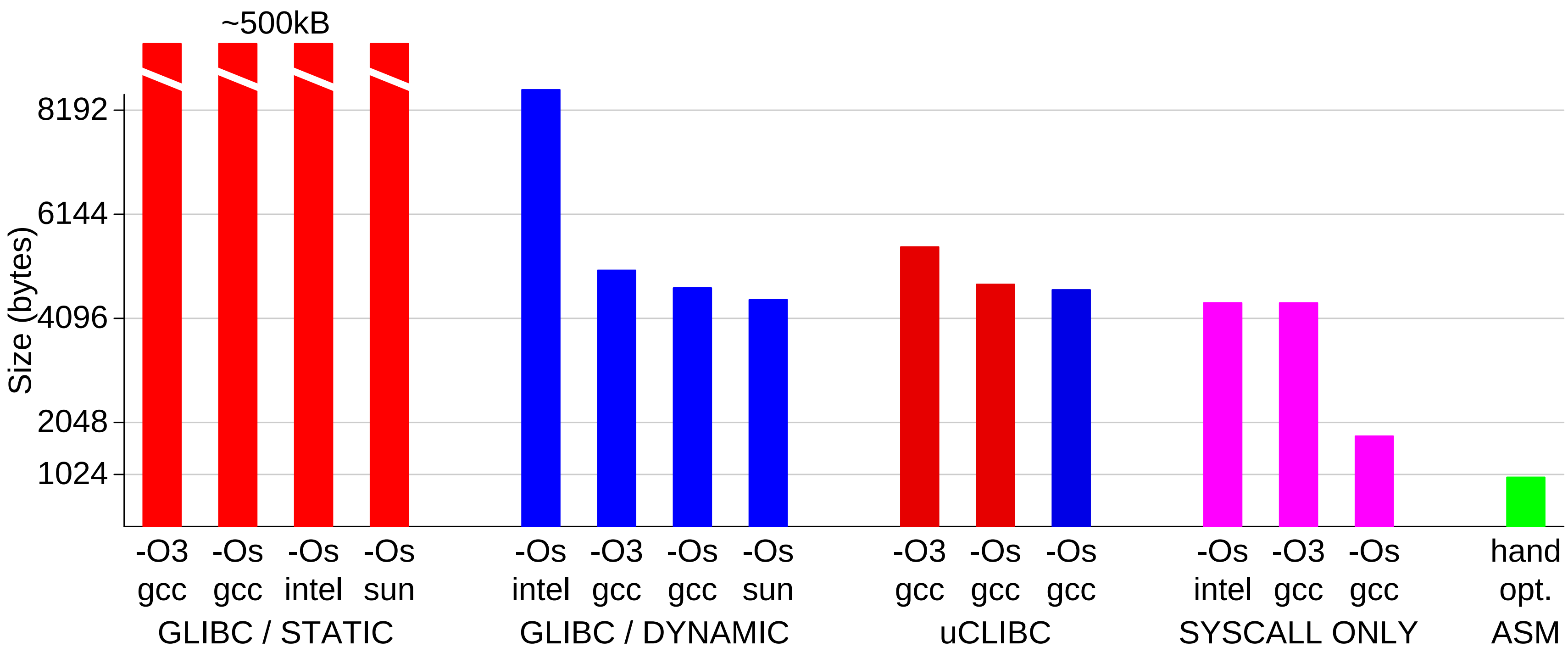
Correlations

Correlation Coefficient	Architectural Parameter
0.9381	Smallest possible instruction length
0.9116	Low number of integer registers
0.7823	Low Virtual address of first instruction
0.6607	Architecture lacks a zero register
0.6159	Low Bit-width
0.4982	Few operands in each instruction
0.3854	Hardware divide in ALU

More Correlations

Correlation Coefficient	Architectural Parameter
0.3653	Unaligned load/store available
0.3129	Year the architecture was introduced
0.2521	Hardware status flags (zero/overflow/etc.)
0.2121	Auto-incrementing addressing scheme
0.0809	Machine is big-endian
0.0021	Branch delay slot

Results – C Comparison (x86/Linux)



What is holding back the C version?

- Stack frame (Calling convention)
- Pointer aliasing
- Full program register allocation
- Constant loading optimizations
- String instructions

Related Work

- RISC Code Compression
- Kozuch and Wolfe – investigate VAX, MIPS, SPARC, m68k, RS6000, PPC
- Hasegawa et al. – gcc generated code on m68k, x86, i960, Sparclite, SPARC, MIPS, AMD29k, m88k, Alpha, RS6000
- Flynn et al. – synthetic architectures

Conclusions / Future Work

- New ISAs are continually being developed; code density is still a concern
- Short instruction codings are key
- High code density requires co-operation of ISA, operating system, system libraries, and compiler
- More architectures should be investigated, as well as more and larger benchmarks

Questions?

All code is available:

http://www.deater.net/weave/vmwprod/asm/ll