EECS 507: Introduction to Embedded Systems Research
Memory Hierarchies in Embedded Systems

Robert Dick

University of Michigan

Announcements

W. Strunk Jr. and E. B. White, *The Elements of Style*. Macmillan Publishing Co., Inc., 2000.

Memory hierarchies in embedded systems: short version

Application specific.

Real-time.

Design-time information.

Memory hierarchies in embedded systems

Highly varied.

MMU-less.

MP-less.

Marker-based debugging.

OS controlled swap relatively rare.

Overlays sometimes used.

Special hardware memory bus controllers common.

Common to find SRAM, DRAM, EEPROM, and FLASH in same system.

Low-power sleep mode might clear RAM.

Streaming systems that avoid transition through RAM.

Scratchpad memory.

Cache locking for real-time systems.

Outline

Problem background

RAM quantity limits application functionality

RAM price dropping but usage growing faster

Secure Internet access, email, music, and games

How much RAM?

- Functionality
- Cost
- Power consumption
- Size

Ideal hardware–software design process

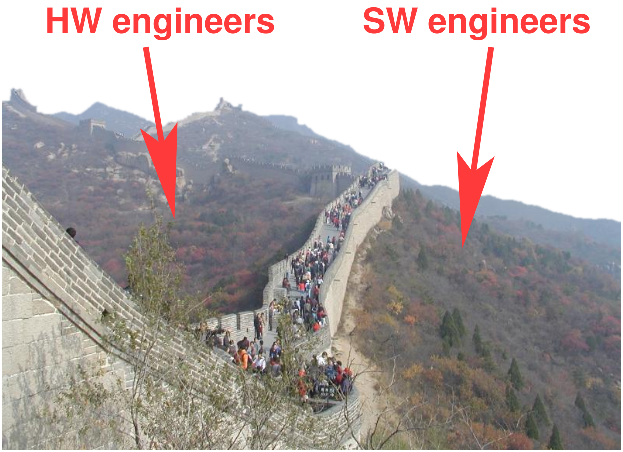
Ideal case

Hardware and software engineers collaborate on system-level design from start to finish

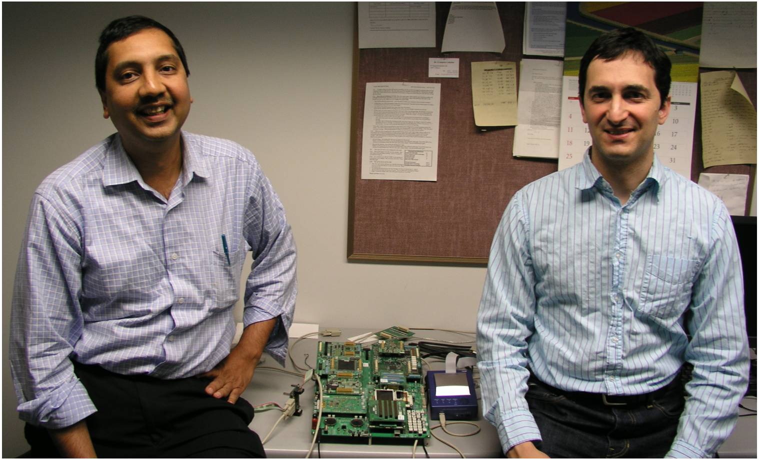
We teach the advantages of this in our classes

It doesn't always happen

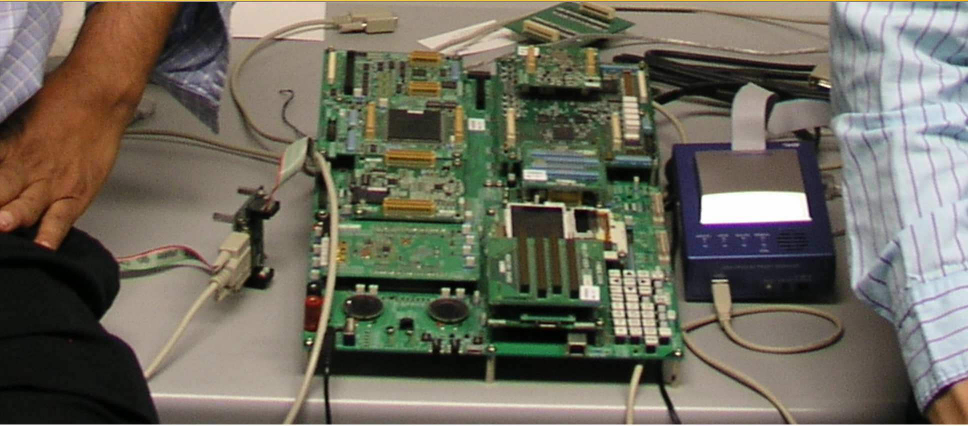
Real hardware–software design process



NEC collaborators and cellphone prototype



NEC collaborators and cellphone prototype



Options

Option 1: Add more memory

Implications: Hardware redesign, miss shipping target, get fired

Option 2: Rip out memory-hungry application features

Implications: Lose market to competitors, fail to recoup design and production costs, get fired

Option 3: Make it seem as if memory increased

- Do not change hardware
- Do not change applications
- Do not decrease performance
- Do not increase power

Nobody knew how to do this

Options

Option 1: Add more memory

Implications: Hardware redesign, miss shipping target, get fired

Option 2: Rip out memory-hungry application features

Implications: Lose market to competitors, fail to recoup design and production costs, get fired

Option 3: Make it seem as if memory increased

- Do not change hardware
- Do not change applications
- Do not decrease performance
- Do not increase power

Nobody knew how to do this

Goals

Allow application RAM requirements to overrun initial estimates even after hardware design

Reduce physical RAM, negligible performance and energy cost

Improve functionality or performance with same physical RAM

Hardware RAM compression: Tremaine 2001, Benini 2002, Moore 2003

- Hardware (de)compression unit between cache and RAM
- Hardware redesign and application-specific compression hardware
- Past work claimed application-specific hardware essential to keep power and performance overhead low

Conventional view: Hardware required

"Compression speed becomes the primary cost measure for assessing the quality of a compression scheme; this automatically rules out any software-based solution, and makes hardware-assisted approaches the only viable option in this context [cache or memory compression] . . ."

– A famous IEEE Fellow, TVLSI 2002

Code compression: Lekatsas 2000, Xu 2004

- Store code compressed, decompress during execution
- Compress off-line, decompress on-line
- For RAM, less important than on-line data compression

Compression for swap performance

Compressed caching

- Douglis 1993, Russinovich 1996, Wilson 1999, Kjelso 1999
- Add compressed software cache to VM

Swap compression

- RamDoubler, Cortez 2000, Roy 2001, Chihaia 2005
- Compress swapped-out pages and store them in software cache

Both techniques

- Target: general purpose system with disks
- Goal: improve system performance
- Interface to backing store (disk)

Outline

CRAMES subproblems

Page selection

Scheduling compression and decompression

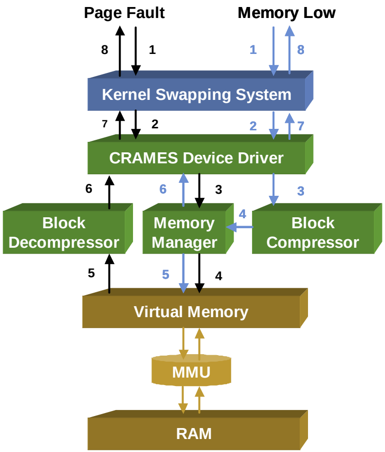
Organizing compressed and uncompressed regions

Dynamically adjust compressed region size

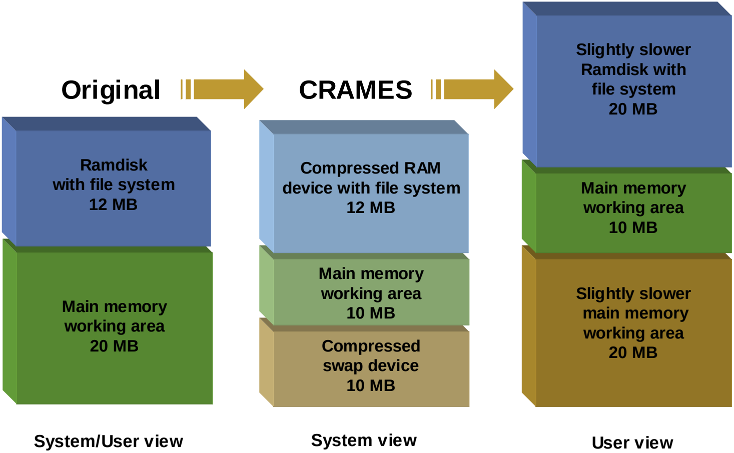
Compression scheme

- High performance
- Energy efficient
- Good compression ratio
- Low memory requirement

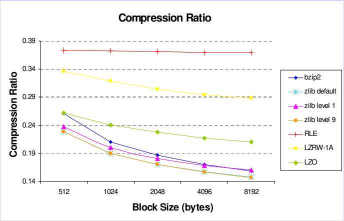
CRAMES design

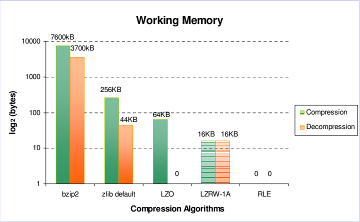


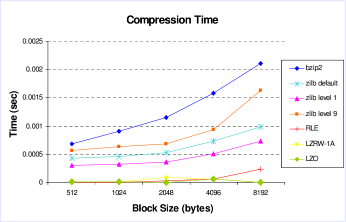
System configuration

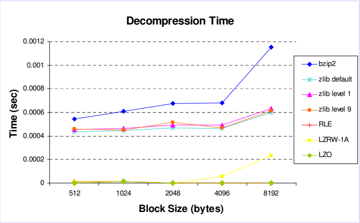


Compression algorithm

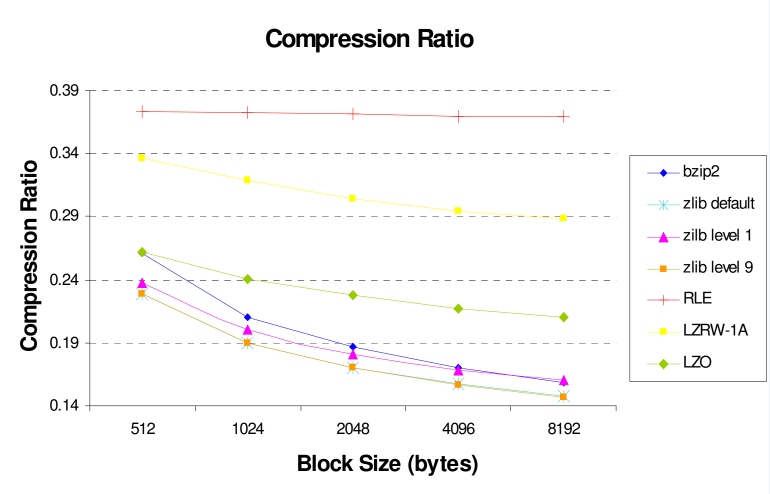




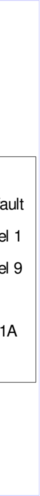


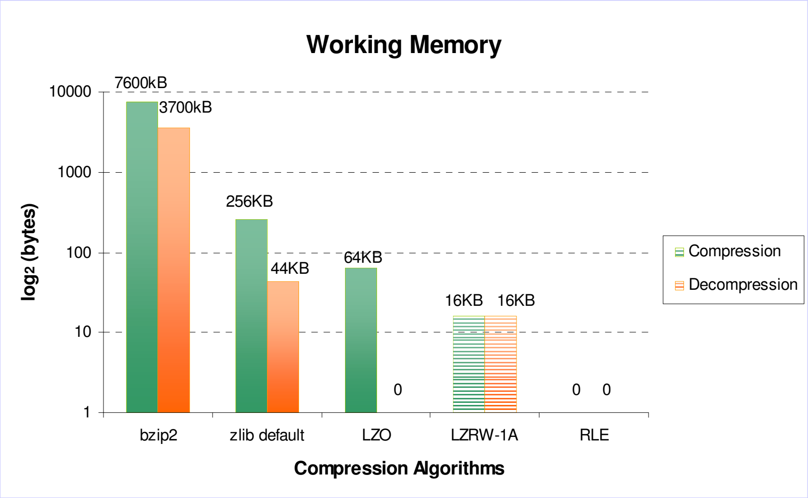


Compression algorithm

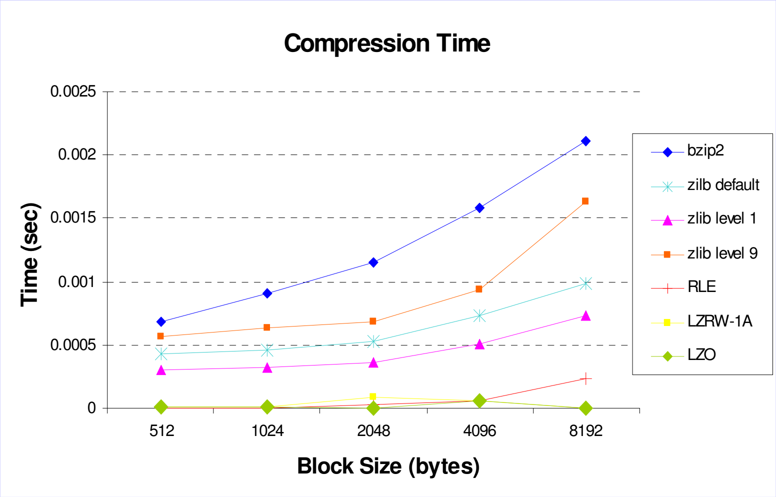


Compression algorithm

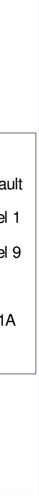


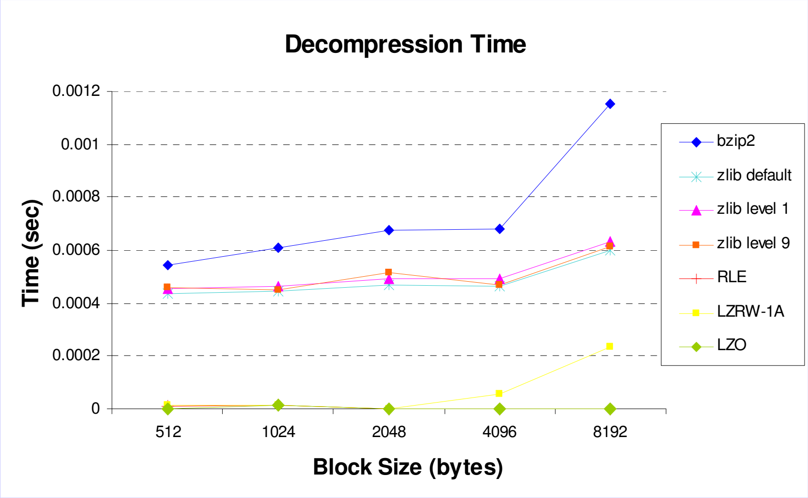


Compression algorithm

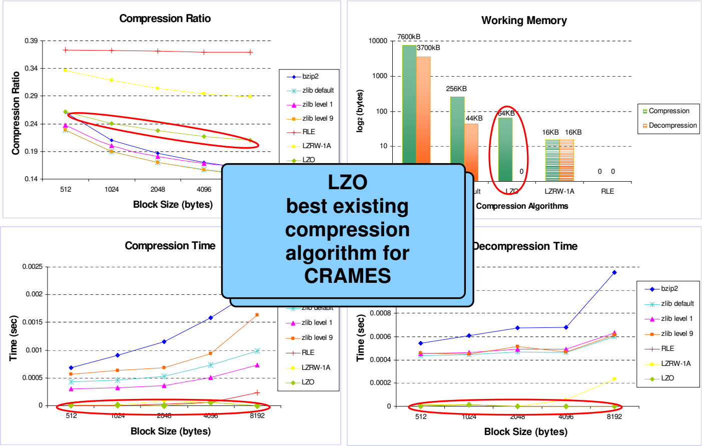


Compression algorithm

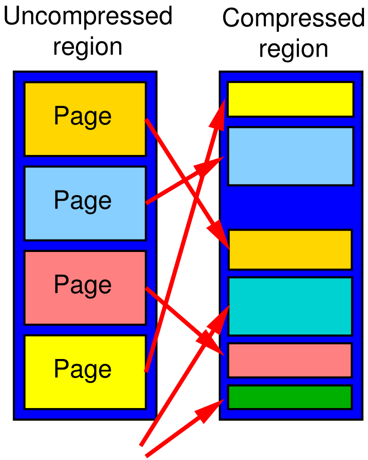




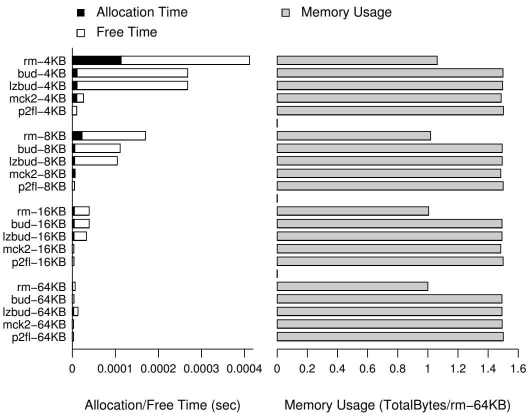
Compression algorithm



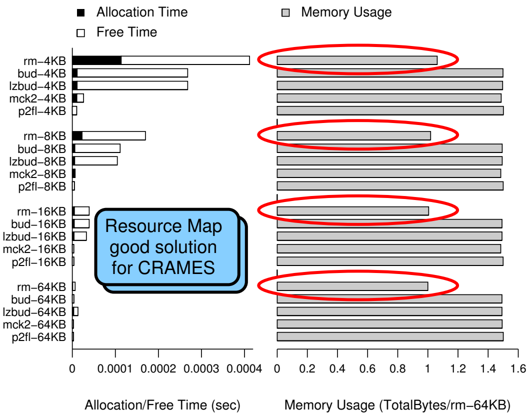
Memory management



Compressed page allocation



Compressed page allocation



Original problem solved, raise the goal

Applications that used to run still run fast

Applications requiring twice as much memory run

Can we go farther?

What happens if we remove half the RAM and run the same applications?

Original problem solved, raise the goal

Applications that used to run still run fast

Applications requiring twice as much memory run

Can we go farther?

What happens if we remove half the RAM and run the same applications?

Weak link: compression algorithm

LZO average performance penalty 9.5% when RAM reduced to 40%

Developed simulation environment to permit profiling

Compression and decompression were taking most time

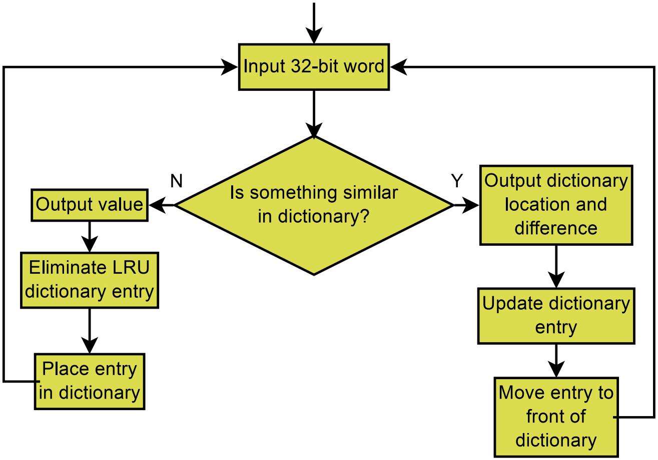
Needed a better compression algorithm

Weak link: compression algorithm

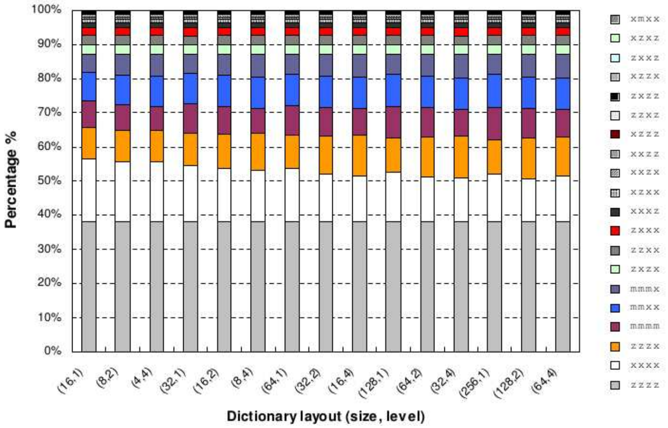
- LZO average performance penalty 9.5% when RAM reduced to 40%
- Developed simulation environment to permit profiling
- Compression and decompression were taking most time
- Needed a better compression algorithm

Outline

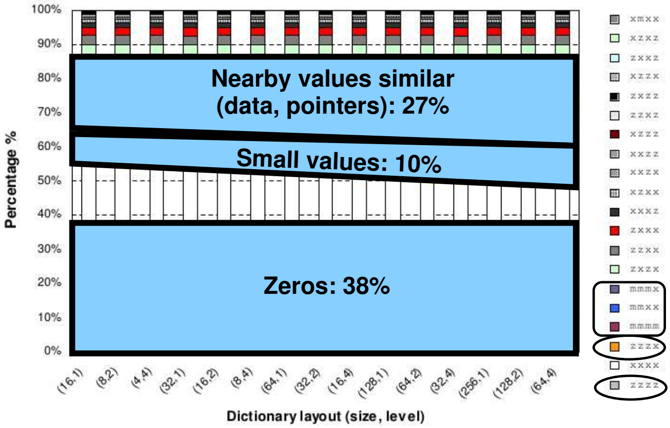
Pattern-based partial dictionary match coding



Data regularity



Data regularity



Pattern-based partial match

Algorithm

- Consider each 32-bit word as an input
- Allow partial dictionary match
- Detect frequent patterns selected based on statistical analysis

Optimizations

- Two-way associative LRU 16-entry hash-mapped dictionary
- Optimized coding scheme
- Early termination for incompressible data
- Fine-grained operation parallelization

Outline

Experimental setup

Sharp Zaurus SL-5600 PDA

- Intel XScale PXA250
- 32 MB flash memory
- 32 MB RAM
- Embedix (Linux 2.4.18 kernel)
- Qt/Qtopia PDA edition

Benchmarks

PBPM twice as fast as LZO

Benchmark applications

Results for

- ADPCM: Speech compression application from MediaBench
- JPEG: Image encoding application from MediaBench
- MPEG2: Video CODEC application from MediaBench
- Straight-forward matrix multiplication
  - Intentionally difficult for CRAMES

Also tested on

- 10 GUI applications that came with Qtopia
- Next-generation cellphone prototype

Results

Base case: 20 MB RAM, no compression

Reduced RAM from 20 MB to 8 MB

Without CRAMES

- All suffered significant performance penalties
- Matrix multiplication cannot execute

With CRAMES

- LZO average case 9.5% overhead, worst case 29%
- PBPM average case 2.1% overhead, worst case 9.2%

Also works on arbitrary in-RAM filesystems

Outline

1. Introduction, motivation, and past work
2. CRAMES design
3. Compression algorithm design
4. Experimental evaluation
5. Commercialization
6. Action items

Status

CRAMES used in NEC cellphones, first in N904i, June 2007

NEC was selling 10 million cellphones per year

Technology won Computerworld Horizon Award in 2007

What is the impact of this technology, essentially?

What did this require?

DRAM process shrink?

What did this require?

~~DRAM process shrink?~~

No! One good Ph.D. student. One good student can have a big impact.

Other compression work

CMP cache compression and data migration algorithms

- Move to many-core increases pressure on cache
- New technique that unifies cache compression and data migration
- Evaluated with SPEC OMP, NASA Parallel, SPEC2000, DIS
- 34% average throughput relative to private L2 caches

Efficient cache compression hardware

- Most cache compression research assumes efficient hardware
- However, fast cache compression hardware design is challenging
- Developed PBPM-based cache compression hardware unit
- 58% system-wide compression ratio, 6.6 ns latency in 65 nm
- DCC'08

Outline

Assignments

12 Sep: R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: A design alternative for cache on-chip memory in embedded systems," in *Proc. Int. Wkshp. Hardware/Software Co-Design*, May 2002, pp. 73–78.

17 Sep: C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. of the ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.