HardBound: Architectural Support for Spatial Safety of the C Programming Language

Joe Devietti[*], Colin Blundell[†], Milo Martin[†], Steve Zdancewic[†]

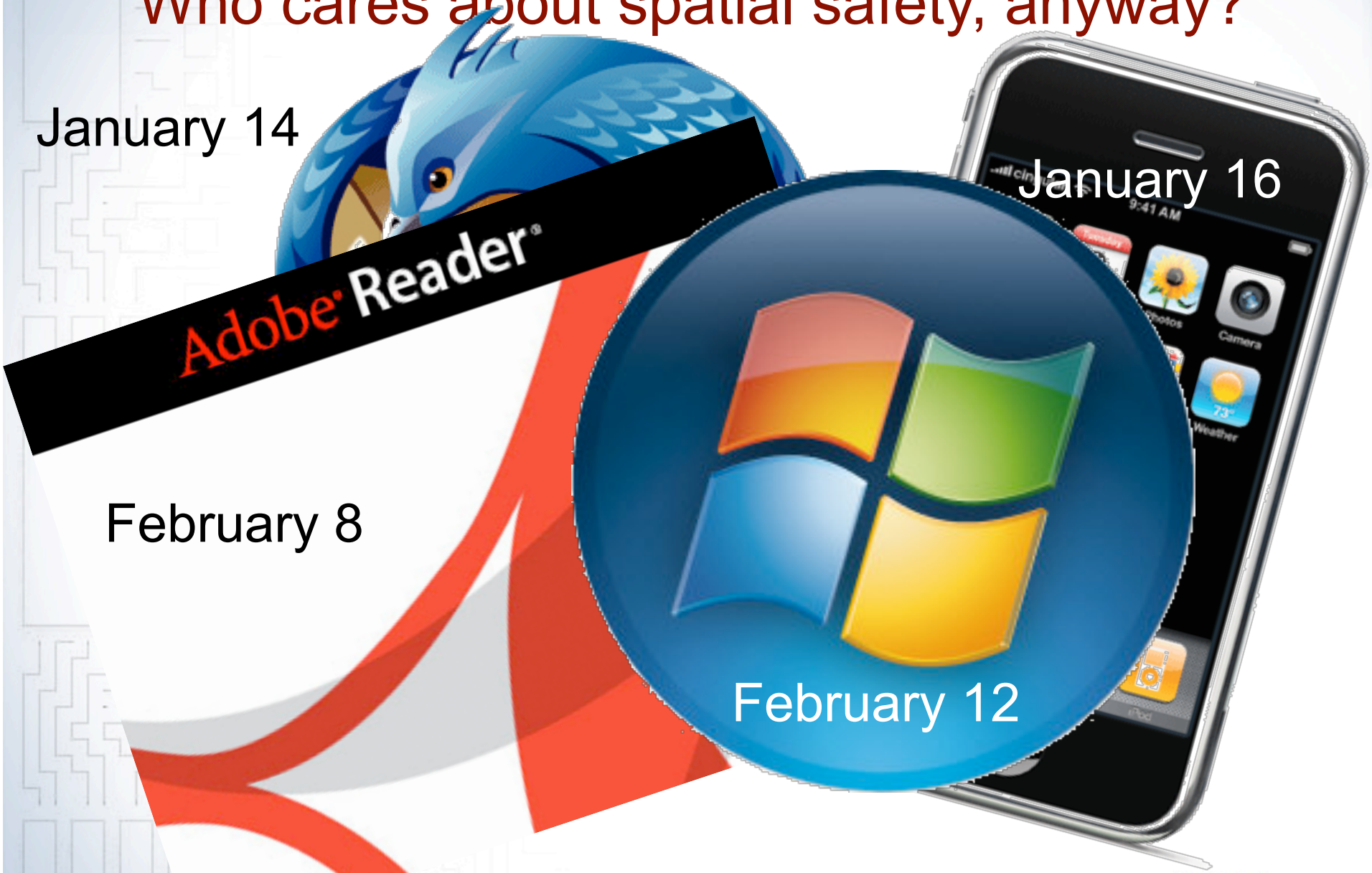
[*]University of Washington, [†]University of Pennsylvania

devietti@cs.washington.edu, {blundell, milom, stevez}@cis.upenn.edu

ACG University of Pennsylvania Architecture + Compilers Group

Who cares about spatial safety, anyway?

January 14

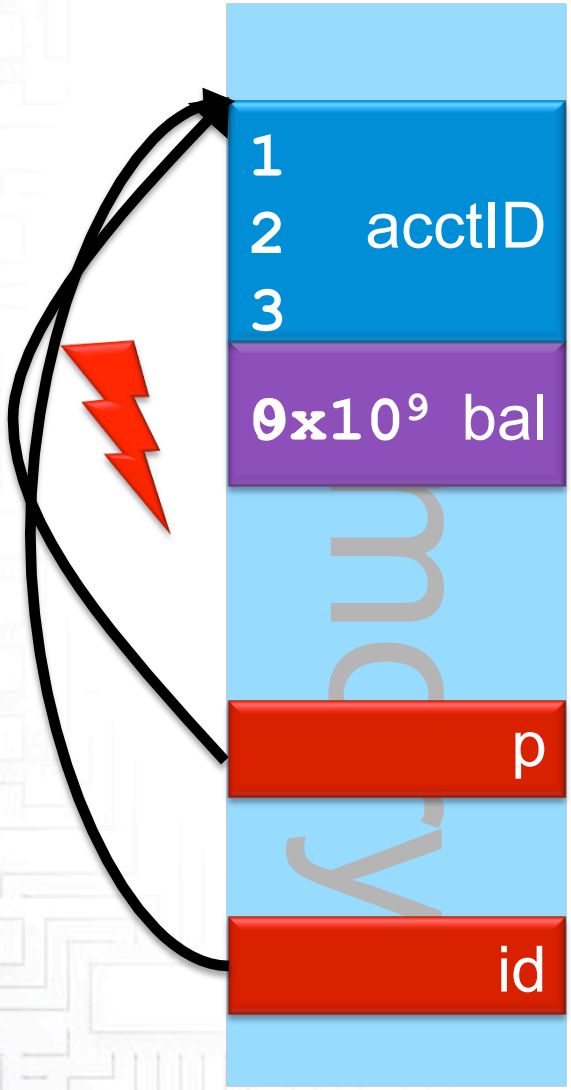


February 8

HardBound: Spatial Safety for C

- Bounded pointer primitive in hardware
- A new hardware/software contract for pointers
  - Software **finds** where pointers are created
  - Hardware **checks** and **propagates** pointers
- Inspired by software-only fat pointer proposals
  - Pointers become (pointer, base, bound) triples
  - \+ Can provide complete spatial safety
  - − Changes memory layout
- Hardware support enables:
  - Unchanged memory layout → high compatibility
  - Efficient fat pointer encodings → low overheads (5%)

Spatial Violation Example




```
struct BankAccount {
  char acctID[3];
  int balance;
} b;
...
b.balance = 0;
char* id = b.acctID;
inputID(id);
...
void inputID(char* p) {
  while (*p = readchar())
    p++;
}
```

Preventing Spatial Violations

treat the language

a safe subset

use a safe

add bounds

write perfect

checking to C

address space randomization

union

pointer

return addresses

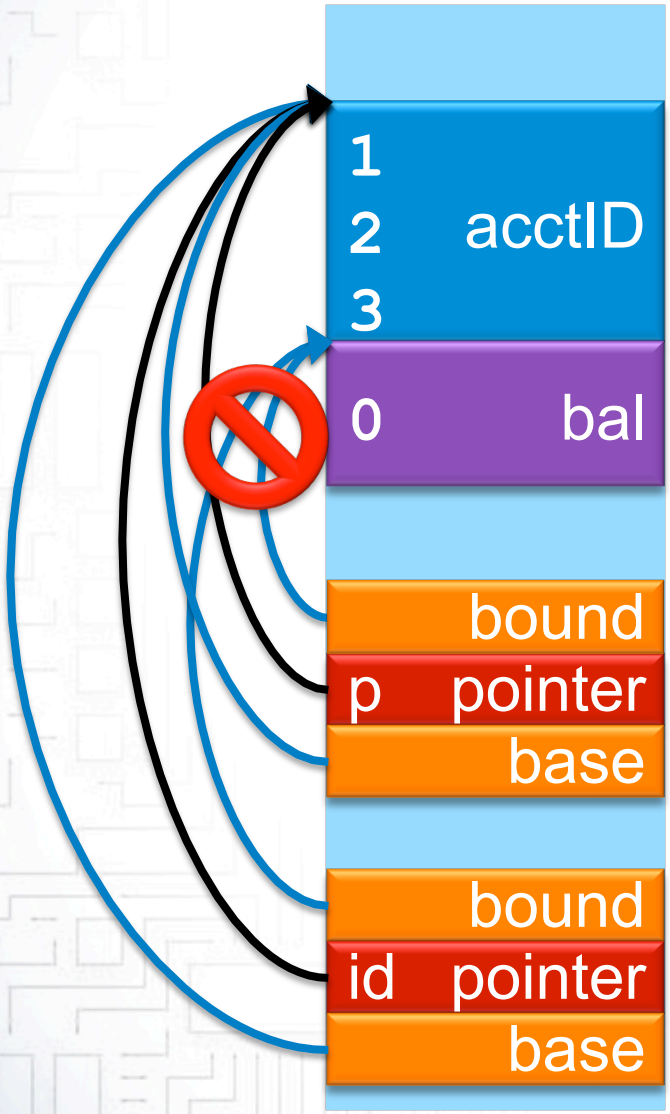
all just existing code?

what about complete

Software-Only Bounds Checking for C

approach	can handle sub-objects?	compatibility?	examples
object lookup			
fat pointer			

Code Example with Fat Pointers

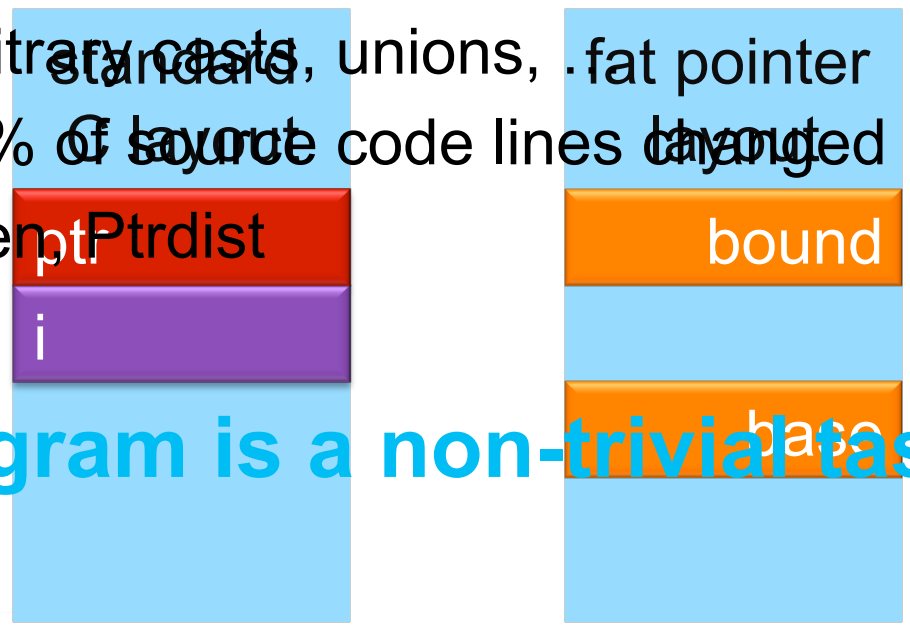



```
struct BankAccount {
  char acctID[3];
  int balance;
} b;
...
b.balance = 0;
char* id = b.acctID;
inputID(id);
...
void inputID(char* p) {
  while (*p = readchar())
    p++;
}
```

CCured

+ Uses fat pointers only where needed
  - determined by whole-program type inference
  - significantly reduces overhead
− Curing a program requires programmer help
  - Interfacing with non-cured libraries
  - C nastiness: arbitrary casts, unions, ...
  - CCured team: 1% of source code lines changed
    - Spec95, Olden, Ptrdist

```
struct {
  int *ptr;
  int i;
} s;
```



Curing a program is a non-trivial task

HardBound: A New Hope

- Bounded pointer hardware primitive
  - implemented via shadow memory and registers
- Compiler **identifies** where pointers are created
  - examples: malloc, &variable (intraprocedural analysis)
  - using `setbound` instruction
- Hardware **checks** and **propagates** the pointers

+ Fat pointer representation **hidden** by hardware
  + compatible (memory layout unchanged)
  + enables optimized metadata encodings

Rest of Talk

- How do we make HardBound work?
  - pointer propagation and checking
  - memory layout
- How do we make HardBound fast?
  - compressing non-pointers
  - compressing pointers

Metadata Propagation and Checking

- All registers and memory have metadata
- Example operations:
  - `add R1 <- R2 + imm`

```
R1.value <- R2.value + imm
```

insn

```
R1.base <- R2.base
R1.bound <- R2.bound
```

propagation

  - `load R1 <- Memory[R2]`

bounds check

```
assert(R2.base <= R2.value < R2.bound)
```

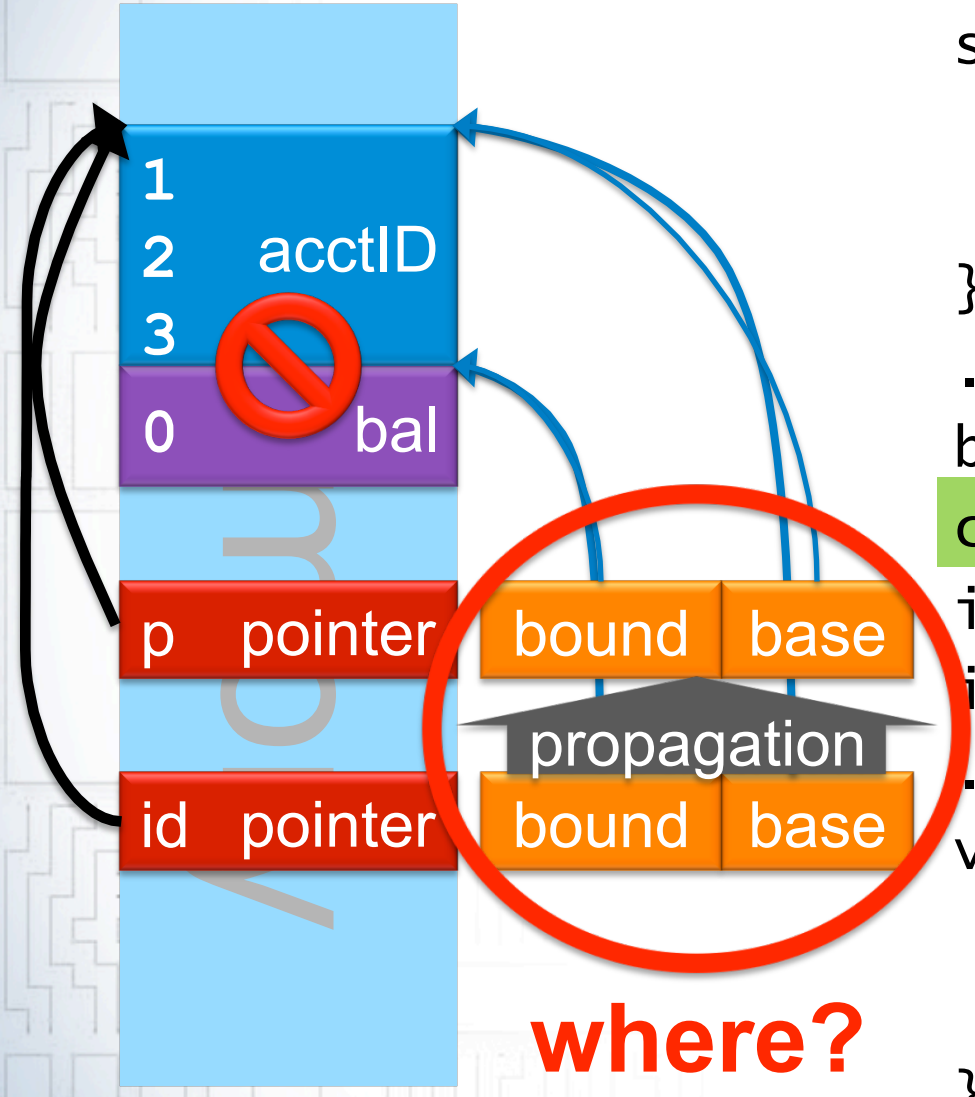
```
R1.value <- Memory[R2.value].value
```

insn

```
R1.base <- Memory[R2.value].base
R1.bound <- Memory[R2.value].bound
```

propagation

The HardBound Hardware/Software Approach

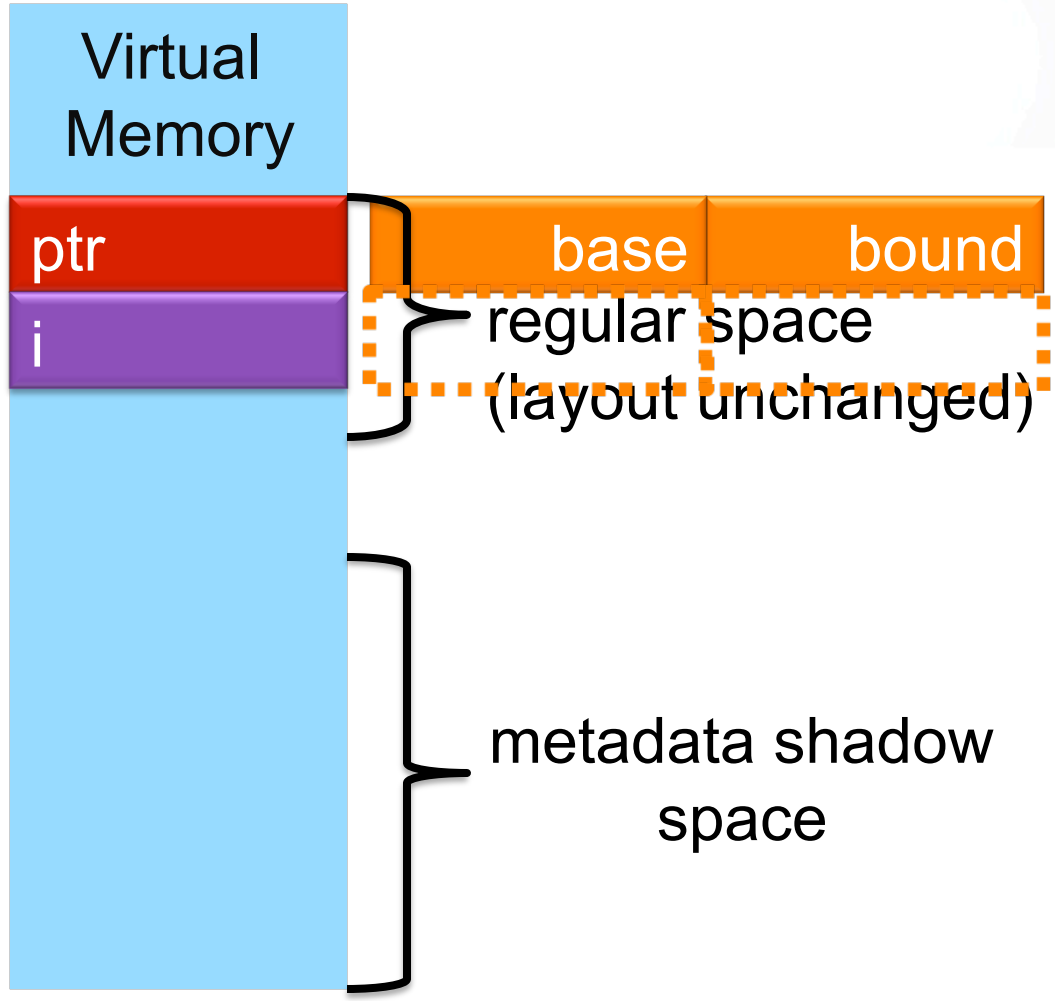



```
struct BankAccount {
  char acctID[3];
  int balance;
} b;
...
b.balance = 0;
char* id;
id = b.acctID;
setbound(b.acctID,3);
inputID(id);
...
void inputID(char* p) {
  while (*p = readchar())
    p++;
}
```

HardBound Memory Layout

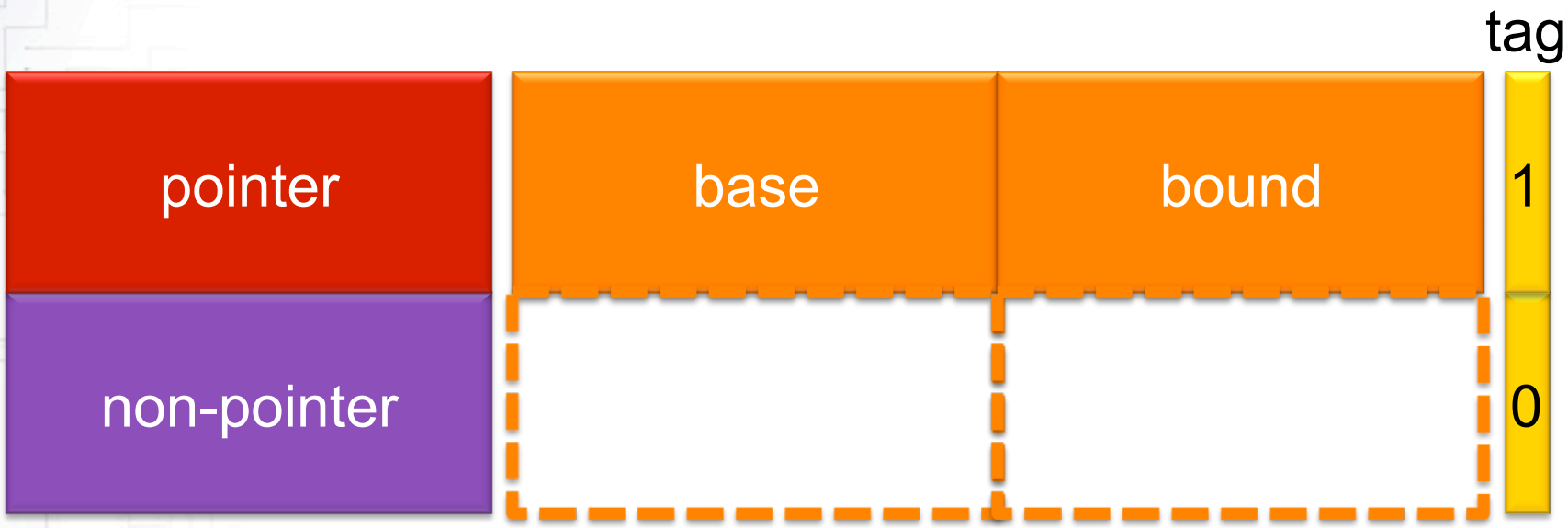
```
struct {
   int* ptr;
   int i;
} s;
```

enables compatibility… …how do we make it fast?



Eliminating Metadata for Non-pointers

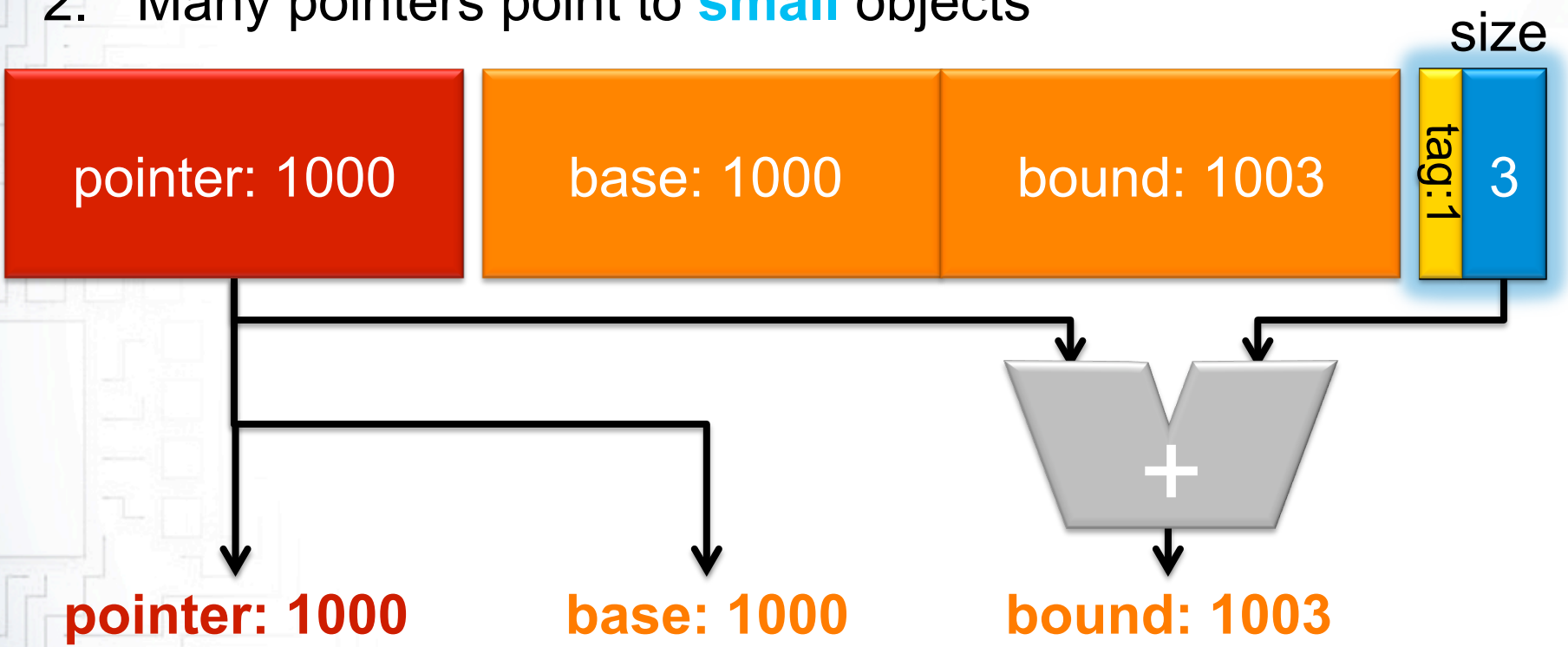
- Most data are not pointers



- add 1-bit tag to act as a filter
  - prevents an expensive base/bound lookup
- lives in virtual memory
  - 1 bit per word (3% overhead)

Compressing In-Memory Metadata

1. Many pointers point to the **beginning** of an object
2. Many pointers point to **small** objects

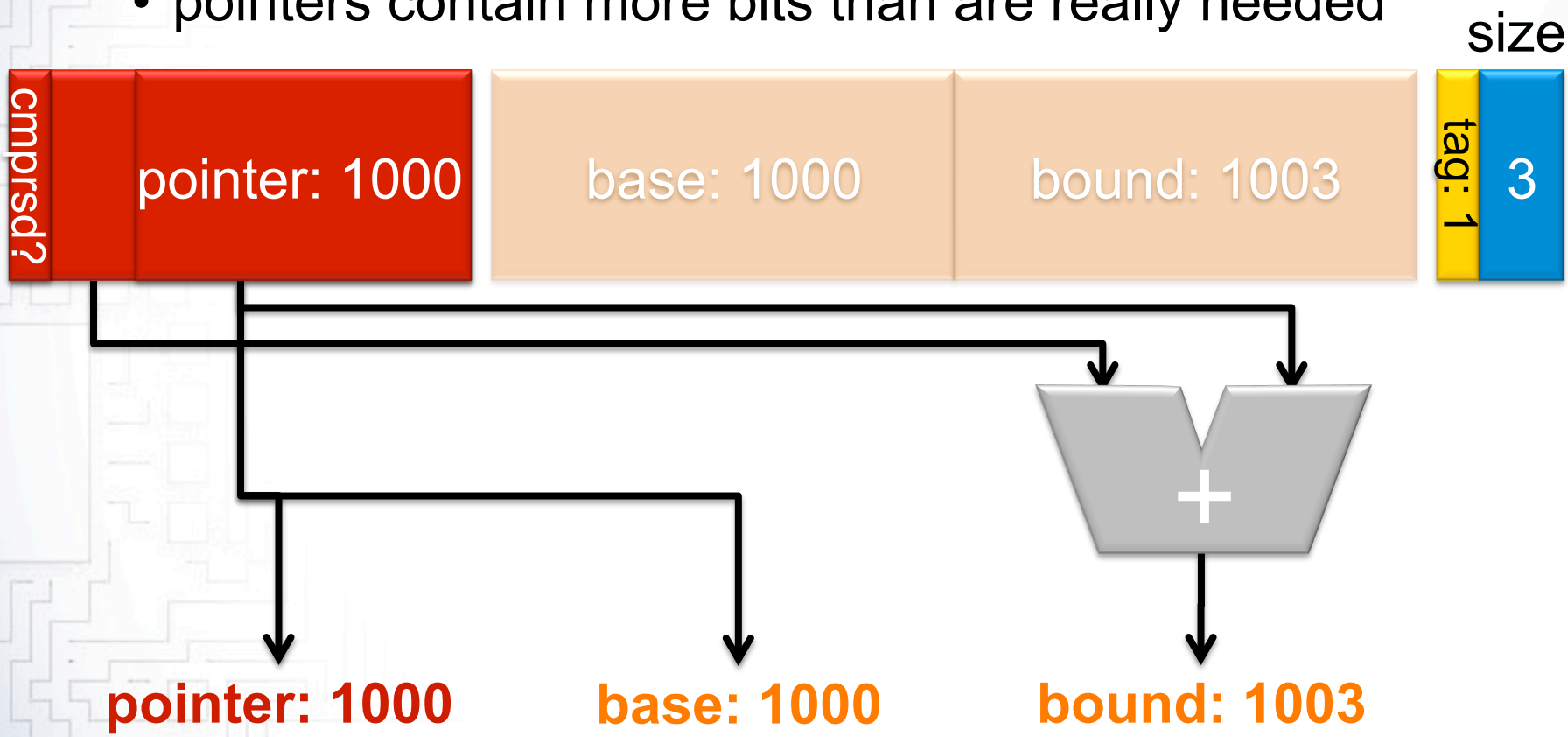


used opportunistically on stores

External encoding: fold size into tag

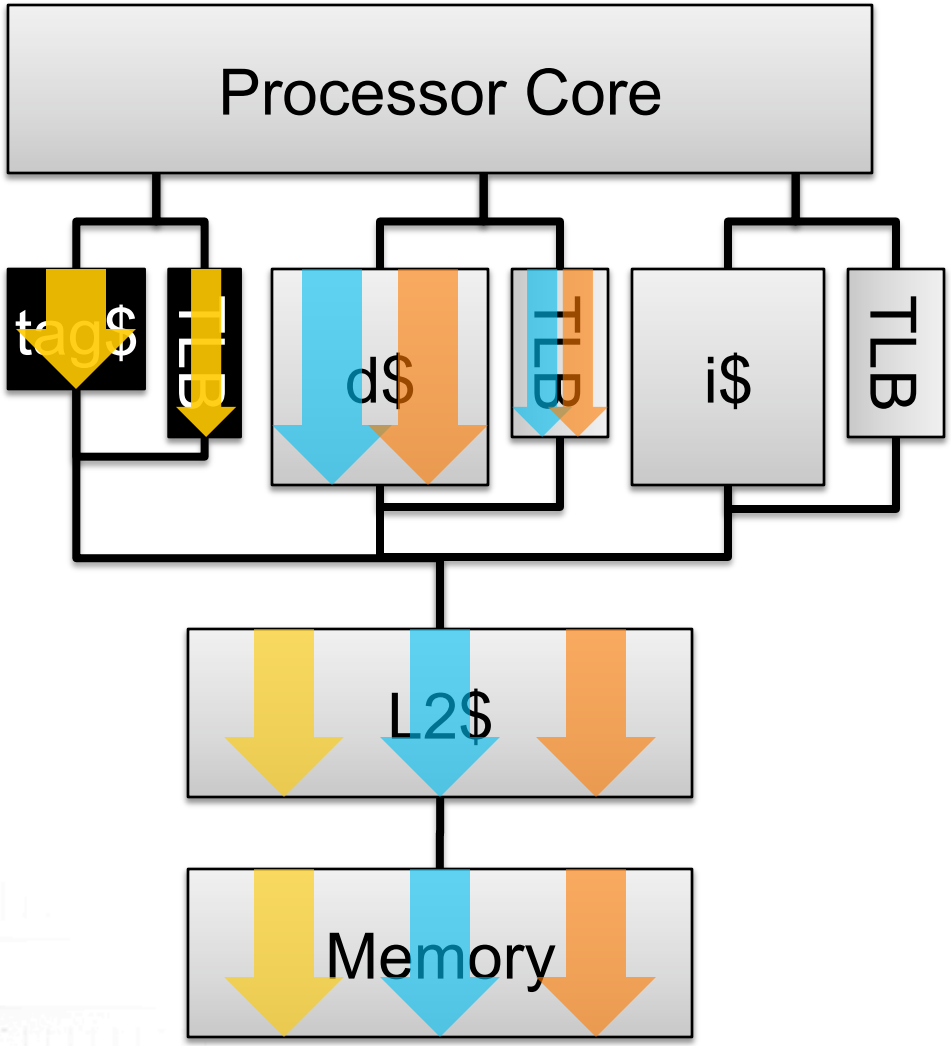
Compressing Metadata for Pointers

- Most programs don't use much of virtual memory
  - pointers contain more bits than are really needed



Internal encoding: hide offset in pointer

HardBound Metadata Lookup



Experiments

- Experimental methodology
  - Source-to-source compiler written in CIL
  - FeS_2 x86 inorder single-cycle processor with Simics
- Tested correctness with a suite of 291 spatial violations [Kratkiewicz & Lippmann, 2005]
  - 286 are compatible with our simulation environment
  - **no false positives, no false negatives**

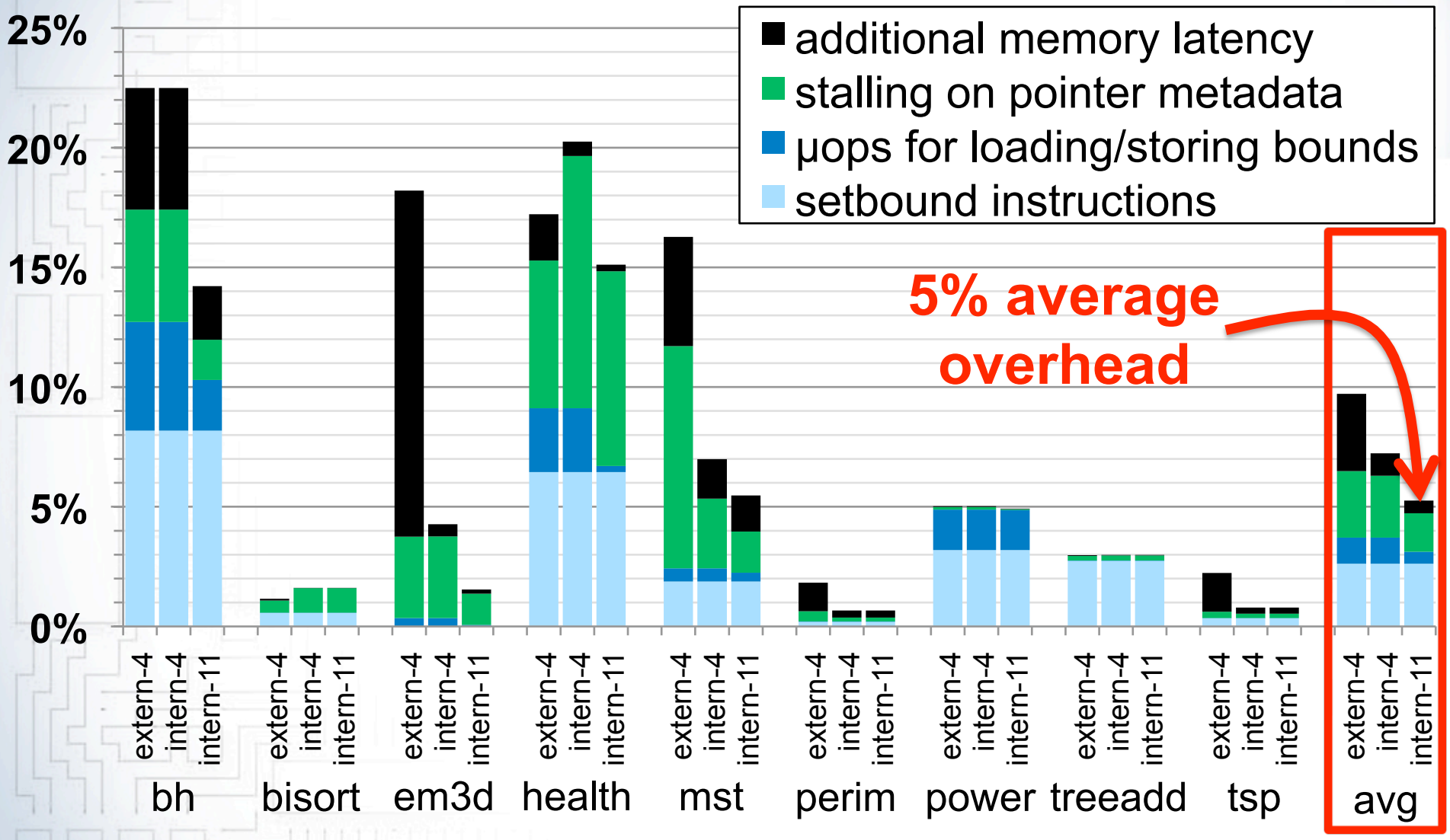
Performance Experiments

- Olden pointer-intensive benchmark suite
  - no source code modifications required for correctness
    - we even found a bug in em3d
  - we made two small performance-only modifications
    - bh: manually inlined two functions
      - reduce number of redundant `setbound` insns
    - mst: manually inserted three `setbound` insns
      - to tighten bounds for better compression

Experimental Configurations

encoding	size of tag space	tag cache	encodes pointers up to
external 4-bit			
internal 4-bit			
internal 11-bit			

Normalized Runtime Results



Conclusions

- HardBound is a hardware/software approach that provides spatial safety for C
- Hardware-managed pointer metadata enables:
  - **high compatibility** (doesn't change memory layout)
  - **low overhead** (enables encoding tricks)
- Our experiments show that HardBound is:
  - **effective** (no false positives, no false negatives)
  - **efficient** (compressing metadata works)
- Future work
  - further reduce overhead using CCured's static analysis
  - temporal safety (dangling pointers)

ACG

University of Pennsylvania

Architecture + Compilers Group

Penn

University of Pennsylvania