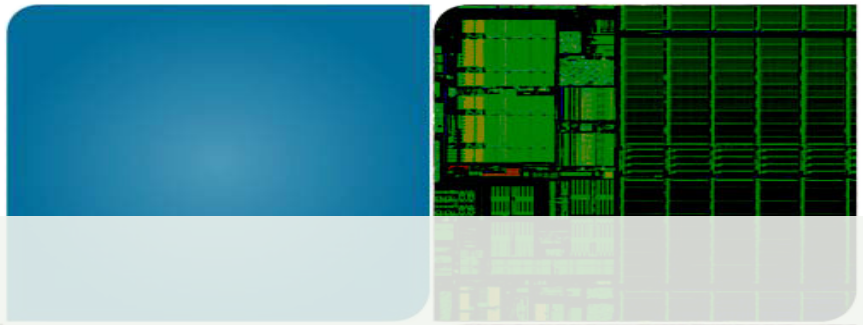
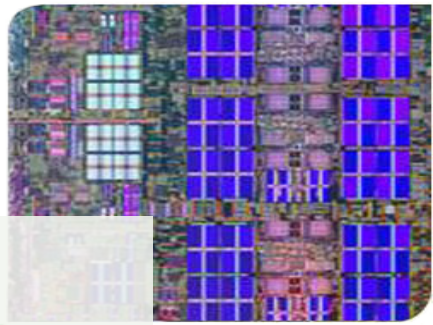
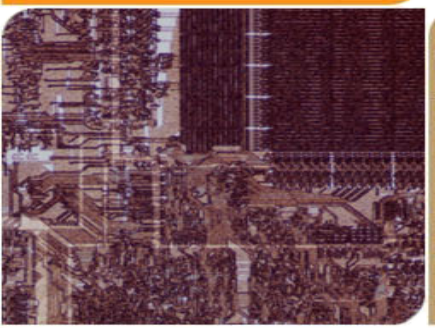




Hardware-based Always-On Heap Memory Safety

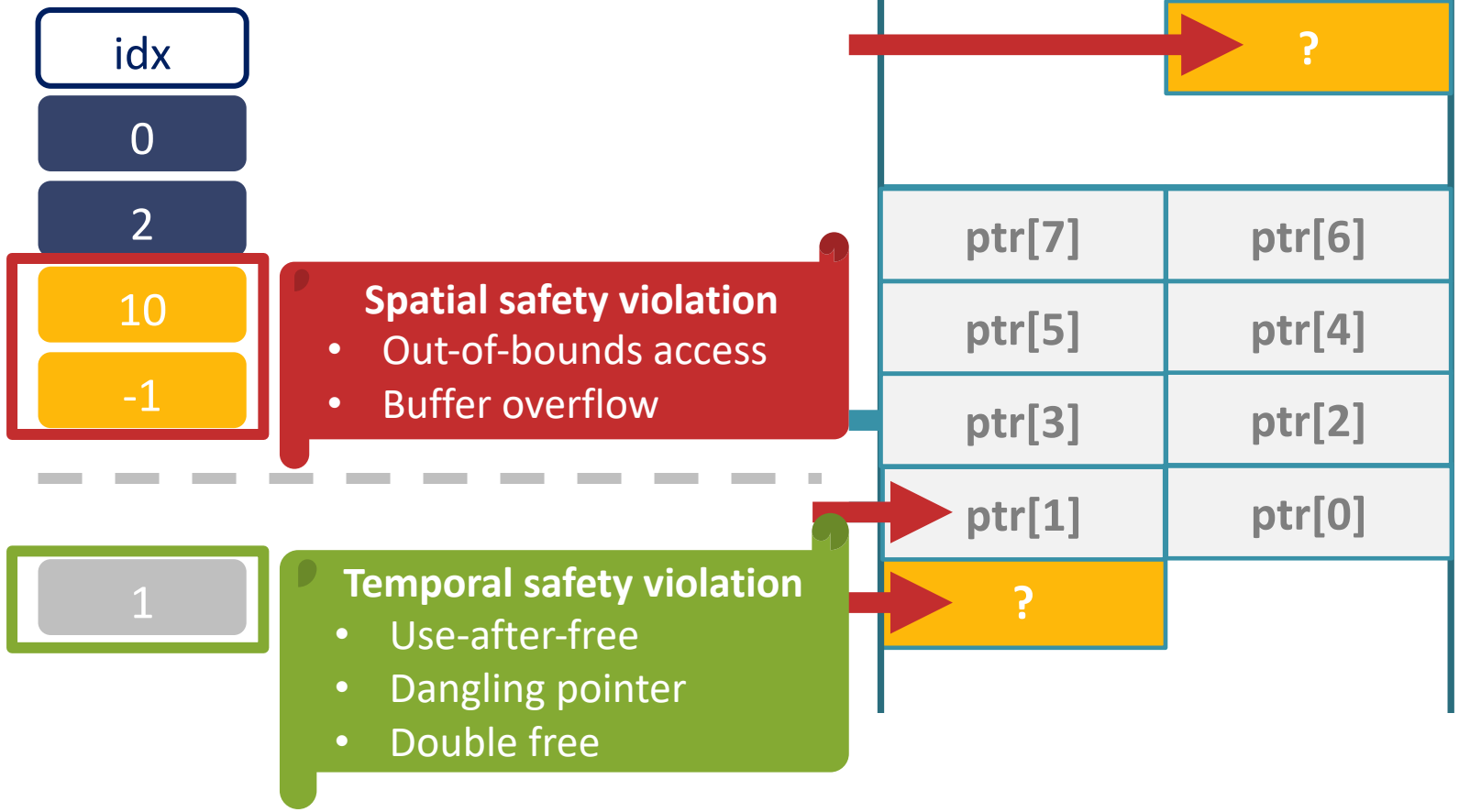
Yonghae Kim, Georgia Tech
Jaekyu Lee, Arm Research
Hyesoon Kim, Georgia Tech



Heap Memory Bugs

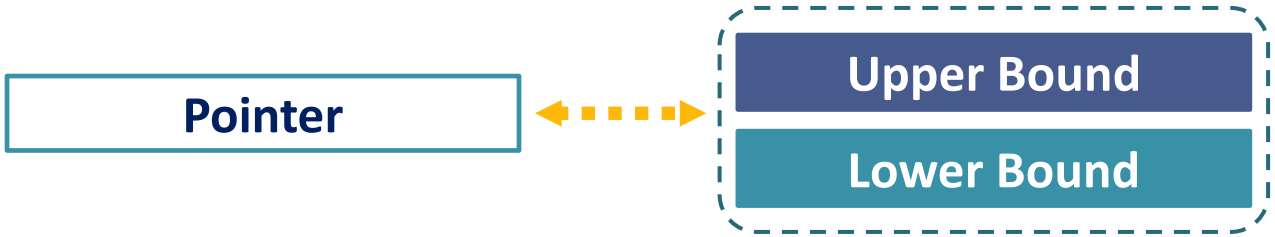

```
int *ptr, idx, val;
ptr = malloc(sizeof(int)*8);

scanf("%d", &idx);
// Vulnerable read
val = ptr[idx];

free (ptr);
// Use-after-free
val = ptr[1];
```

Controlled by a user



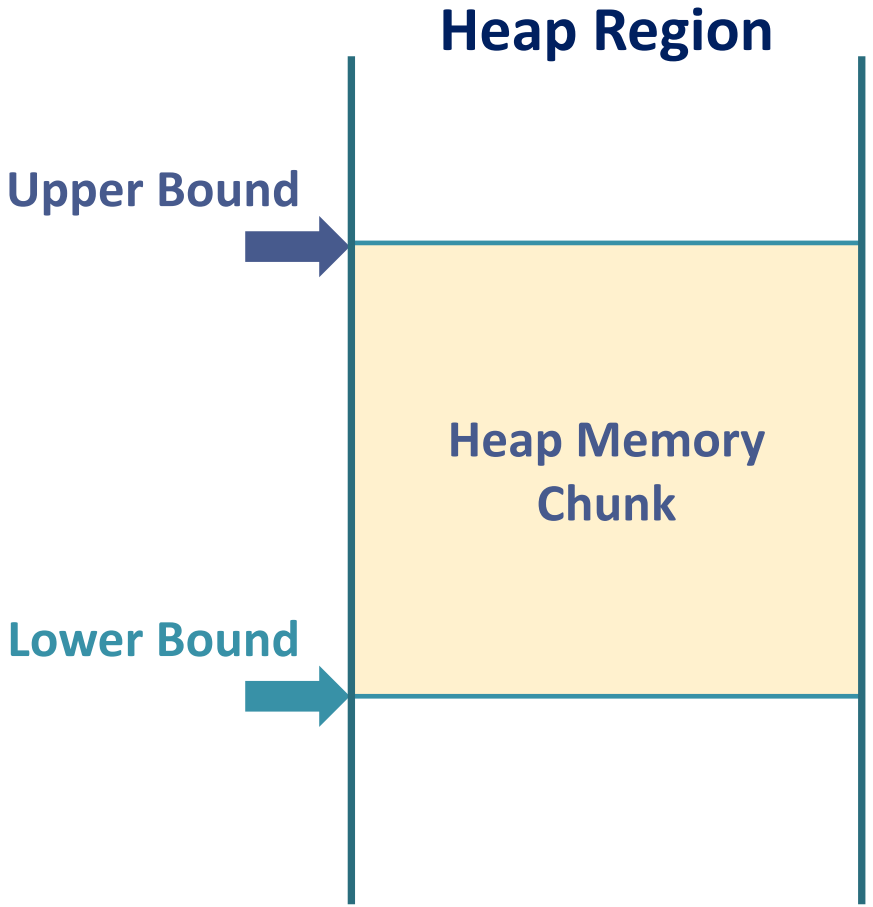
How to Prevent Such Violations?

- Bounds checking.
  - Associate bounds metadata with a pointer to protect.
  - Perform bounds checking upon pointer dereferencing.
  - Can detect any out-of-bounds (OOB) access.



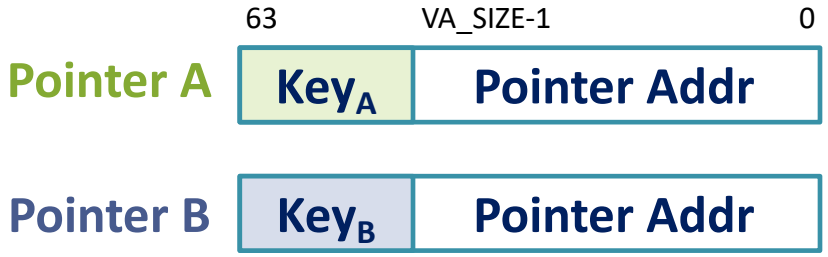
How to efficiently maintain bounds?

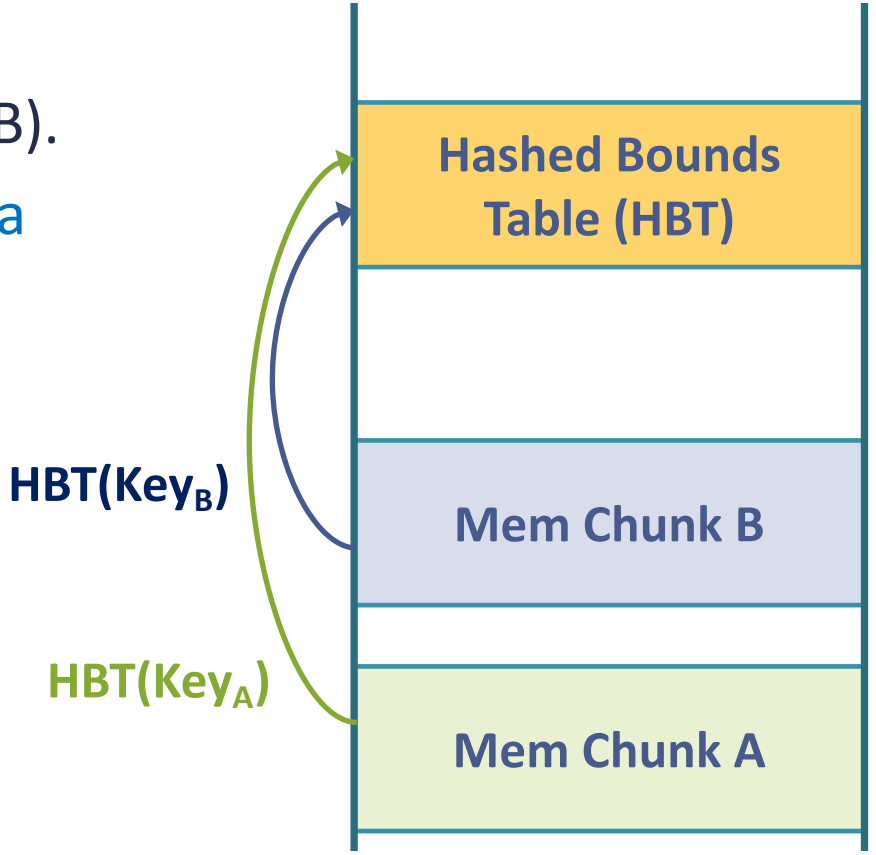
How to efficiently access bounds?



Our Approach: Always-On Memory Safety (AOS)

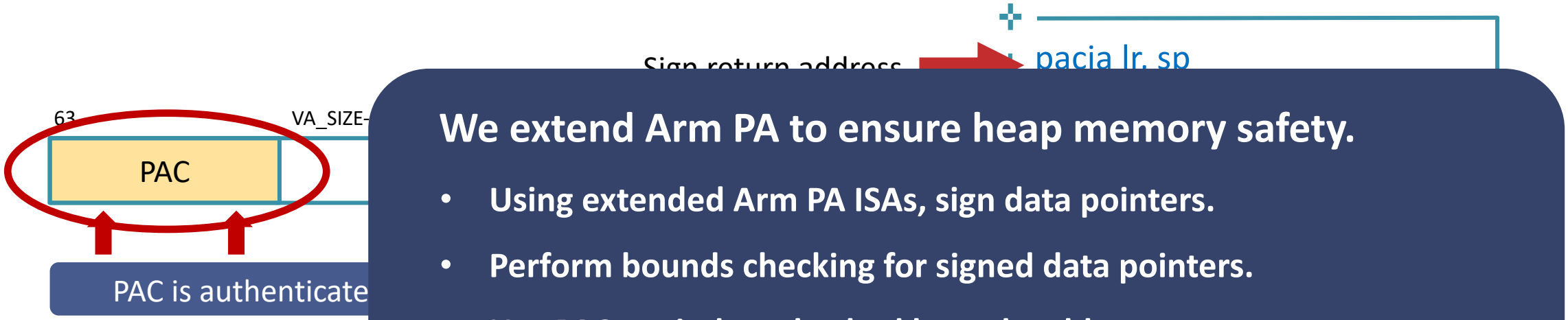
- Idea: Use the unused upper bits of pointers!
  - Under typical VA schemes, 11 to 32 bits are available.
- However, not enough to store bounds metadata (16B).
- Solution: place a key in a pointer and use the key as a hash to index a hashed bounds table, where bounds are stored.





Background: Arm Pointer Authentication (PA)

- To ensure pointer integrity,
  - [Signing]: Place pointer authentication code (PAC) into upper bits.
  - [Authentication]: Check the integrity of PAC before use.
- However, it does not provide spatial and temporal safety.

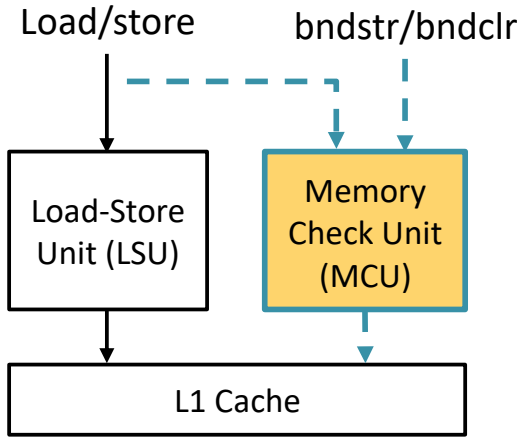


We extend Arm PA to ensure heap memory safety.

- **Using extended Arm PA ISAs, sign data pointers.**
- **Perform bounds checking for signed data pointers.**
- **Use PACs to index a hashed bounds table.**
- **Furthermore, we propose effective iterative bounds search.**

AOS Overview

- ISA extensions.
  - Arm PA extensions: AOS pointer signing (pacma, xpacm, ...).
  - New instructions: Bounds store/clear (bndstr/bndclr).
- Hardware Extension.
  - Memory Check Unit (MCU).
    - Bounds store/clear.
    - Selective bounds checking.
  - Support for new AOS pointer signing.
- Hashed Bounds Table (HBT).
  - A multi-way bounds table with gradual resizing.
  - Allocated in the memory by OS when a process is initiated.



< Memory Check Unit (MCU) >

Way 0			...			Way T-1		
BND	BND	...			...			...
BND	BND	...			...			...
BND	BND	...	BND	BND	...	BND	BND	...
	BND	...	BND		...			...

< Hashed Bounds Table (HBT) >

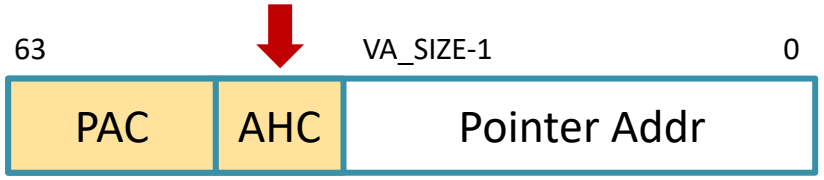
Pointer Signing & Bounds Store

- Which pointer to check?
- Sign data pointers and store bounds.
  - pacma <pointer>, <modifier>, <size>.
    - Compute PAC and AHC (Address Hashing Code) and insert into a pointer.
  - bndstr <pointer>, <size>.
    - Compress bounds to 8 bytes.
    - Store bounds in HBT.

```
ptr = malloc (size);
pacma ptr, sp, size;   // sign data pointer
bndstr ptr, size;      // store bounds
```

Blue instructions are inserted at the compile time.

Nonzero AHC indicates a signed pointer.

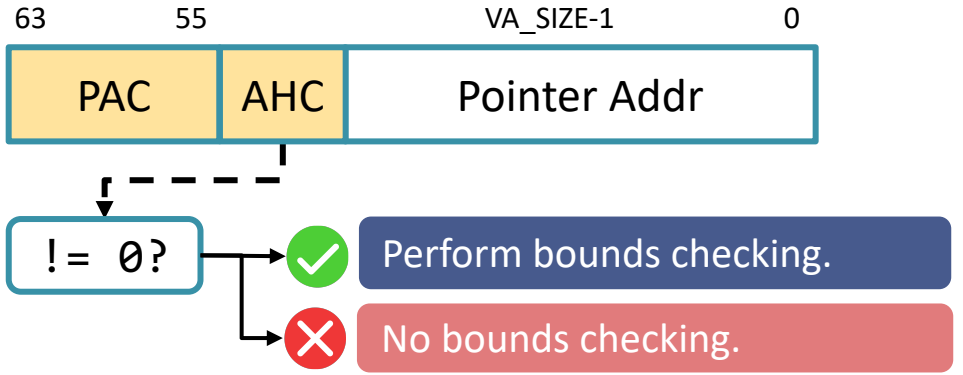


< AOS data pointer signing >

Bounds Checking

- Validate memory access.
  - If the pointer has been signed, perform bounds checking.
    - Bounds checking requires no explicit instructions.
    - Metadata such as PAC and AHC are propagated without any overhead.
- Iterative bounds searching in HBT.
  - Iterate until bounds are found, which the pointer address belongs to.
  - If it cannot find the bounds, it fails!

```
ptr = malloc (size);
pacma ptr, sp, size;  // sign data pointer
bndstr ptr, size;     // store bounds
// ...
val = *ptr;           // Trigger bounds checking
ptr2 = ptr + 1;       // Pointer arithmetic
*ptr2 = 0;            // Trigger bounds checking
```



< AOS bounds checking >

Hashed Bounds Table Access

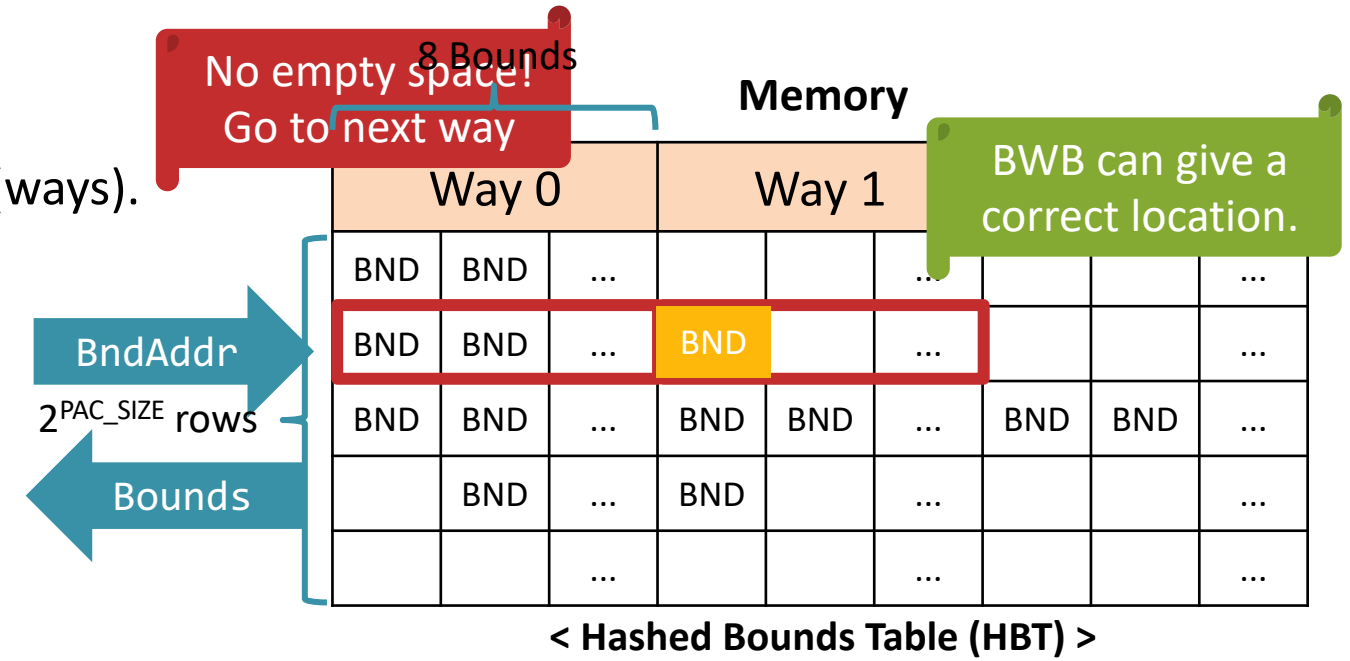
- Indexed by PACs.
  - Simple bounds address calculation.
- Parallel bounds searching.
  - Store 8 bounds (8 x 8B) in one way.
  - Load 8 bounds at a time.
- Bounds Way Buffer (BWB)
  - Keep track of recent bounds locations (ways).

Multiple iteration rarely occurs.

→ **With a 16-bit PAC size,**

1-way HBT can cover up to 512K (= 2^{16} x 8) bounds.

	Iterate until
bndstr	An empty space is found.
load/store	Bounds checking succeeds.
bndclr	Bounds to clear are found.

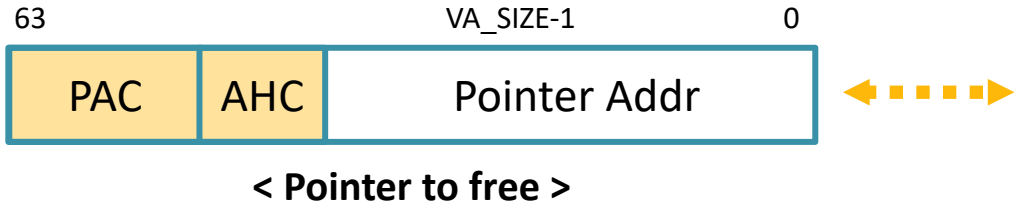


Ensuring Temporal Safety

- When a pointer is freed, clear bounds.
  - bndclr <pointer>.
    - Clear the corresponding bounds in HBT.
  - xpacm <pointer>.
    - Temporarily strip both PAC and AHC from a pointer.
    - Used to avoid unnecessary bounds checking during free().
  - pacma <pointer>, <modifier>, <size>.
    - Re-sign a pointer to prevent further use.

```
bndclr ptr          // clear bounds
xpacm ptr           // strip data pointer
free (ptr);
pacma ptr, sp, xzr  // re-sign data pointer
```

Blue instructions are inserted at the compile time.



Bounds-checking failure occurs
→ Its bounds do not exist anymore!
→ Detect temporal errors



Optimizations

- Bounds compression.
  - Compress bounds information to 8 bytes.
- Bounds table access during resizing.
  - Mitigate the cost of HBT resizing.
- Bounds cache (L1-B cache).
  - An optional L1-B cache reduces cache pollution.
- Bounds store-to-load forwarding.
  - Reduce memory accesses.

Please refer to the paper for the details.



Methodology

- Implemented using the gem5 simulator.
- Added new passes in LLVM.
- SPEC CPU 2006 workloads with reference input sets.
- Evaluated four system configurations,
  - **Watchdog**[1]: Prior work that features user-after-free and bounds checking.
  - **PA**[2]: Arm PA-based pointer integrity solution.
  - **AOS**: AOS bounds-checking mechanism.
  - **PA+AOS**: AOS integrated with PA.

[1] "Watchdog: Hardware for safe and secure manual memory management and full memory safety," S. Nagarakatte et al., ISCA (2012).
[2] "PAC it up: Towards pointer integrity using ARM pointer authentication," H. Liljestrand et al., USENIX Security (2019).

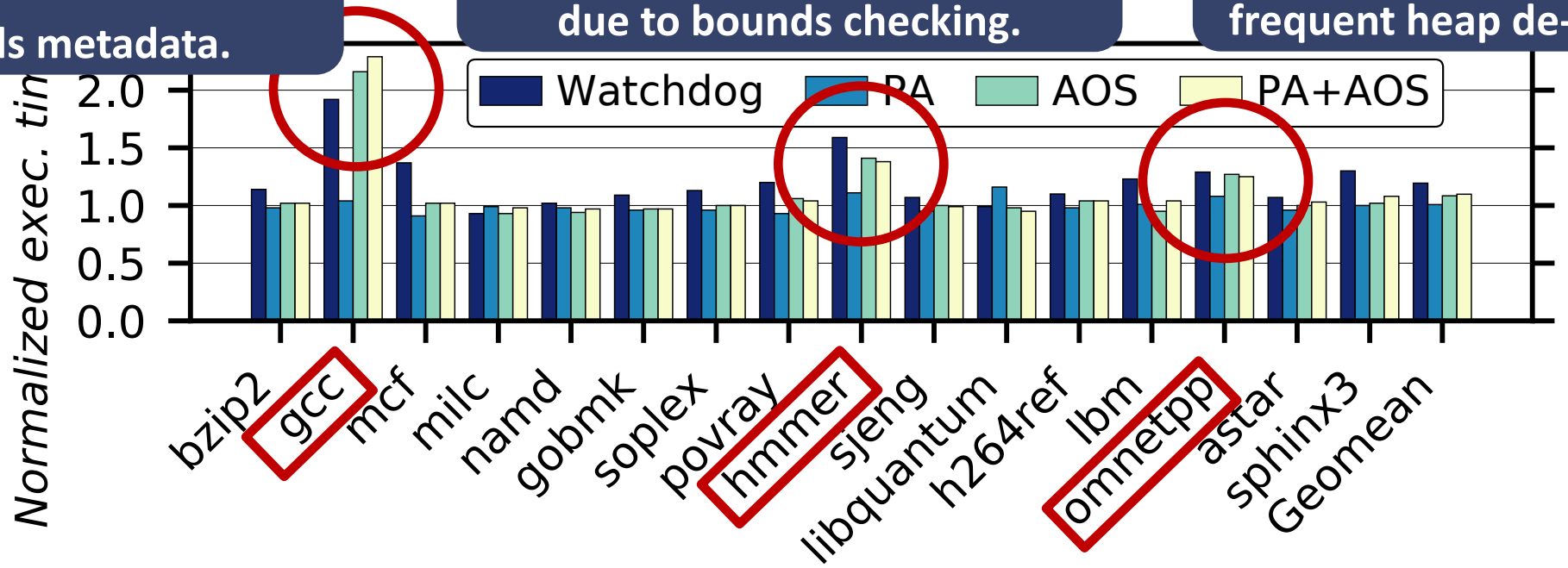
Performance Evaluation

- AOS shows an 8.4% overhead on average.
- Watchdog incurs 19.4% overhead on average.
- PA shows negligible overhead (< 1%) for most applications

Cache pollution caused by bounds metadata.

Delayed instruction retirement due to bounds checking.

Instruction overhead from frequent heap de-/allocations.





Conclusion

- We proposed AOS, a defense mechanism for heap spatial and temporal safety.
- In AOS, we introduced:
  - A data pointer signing scheme for selective bounds checking.
  - A micro-architectural unit (MCU) for efficient bounds operations.
  - A multi-way bounds table with gradual resizing for an efficient bounds metadata management.
- AOS achieved marginal performance overhead (8.4%) while providing strong security guarantees.
- We believe that AOS can serve as an effective runtime safety solution.



Thank you!

All questions are welcome.

Contact information: yonghae@gatech.edu