MANCHESTER 1824
The University of Manchester

MaxSim: A Simulation Platform for Managed Applications

Open-source: https://github.com/beehive-lab/MaxSim

Andrey Rodchenko, Christos Kotselidis,
Andy Nisbet, Antoniu Pop, Mikel Lujan

Advanced Processor Technologies Group,
School Of Computer Science,
The University of Manchester

Overview

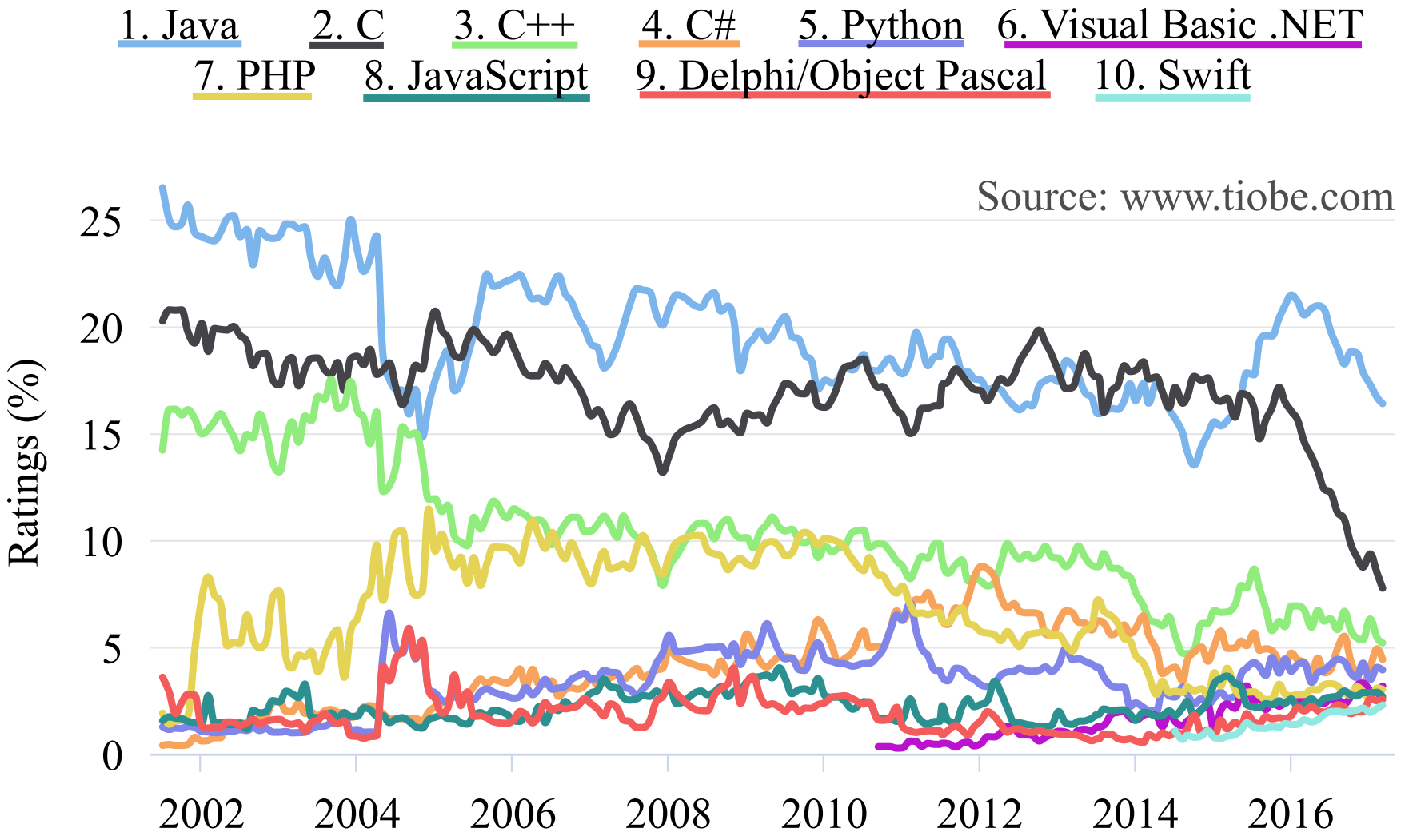
The University of Manchester

- What simulation platform for managed applications is needed and why?
- VM Selection Justification: Maxine VM
- Simulator Selection Justification: ZSim
- MaxSim: Overview and Features
- Use Cases: Characterization, Profiling, and HW/SW Co-design
- Conclusion

What simulation platform for managed applications is needed and why?

The University of Manchester

TIOBE Programming Community Index (March 2017)



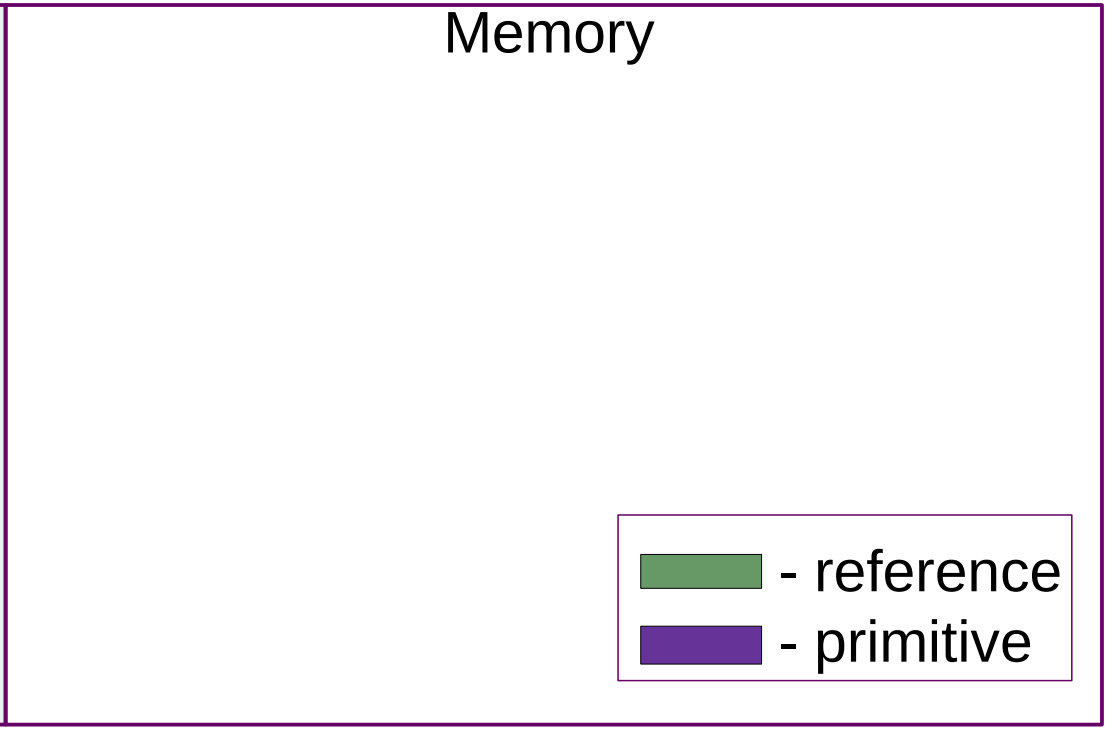
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

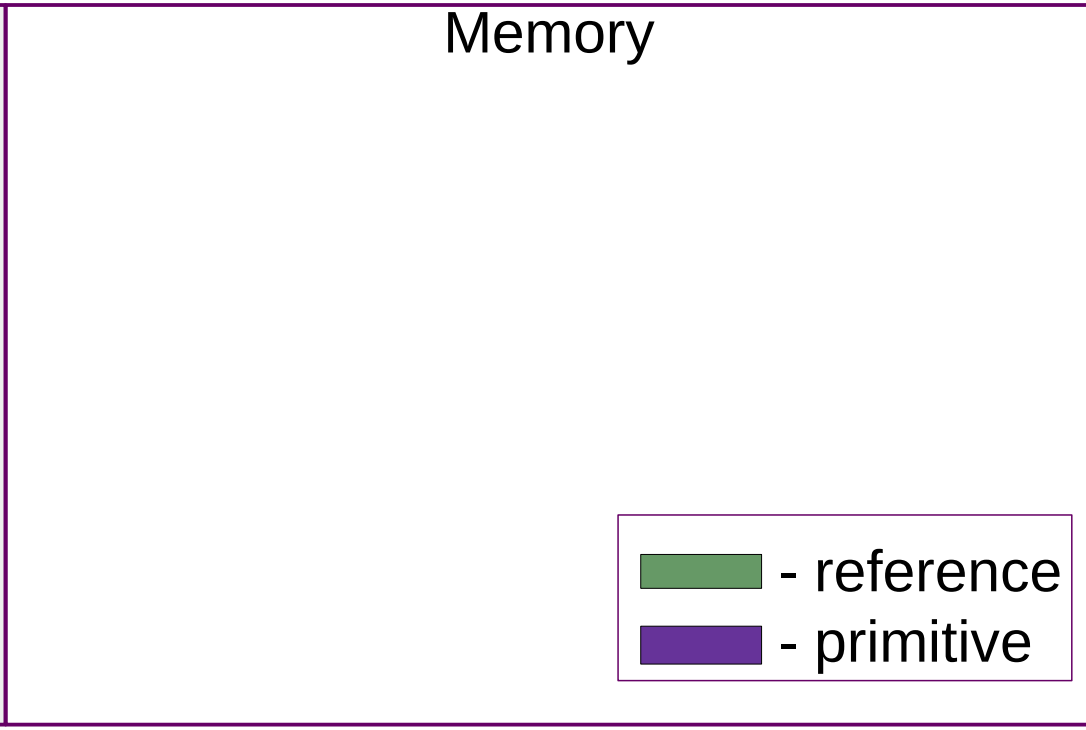
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

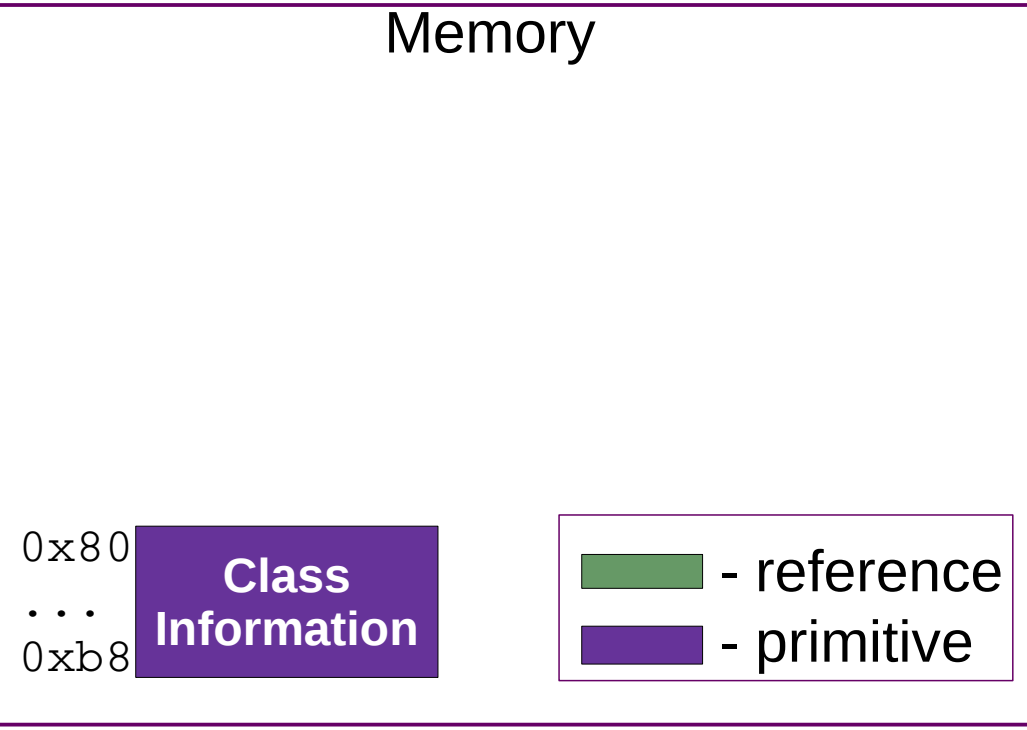
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

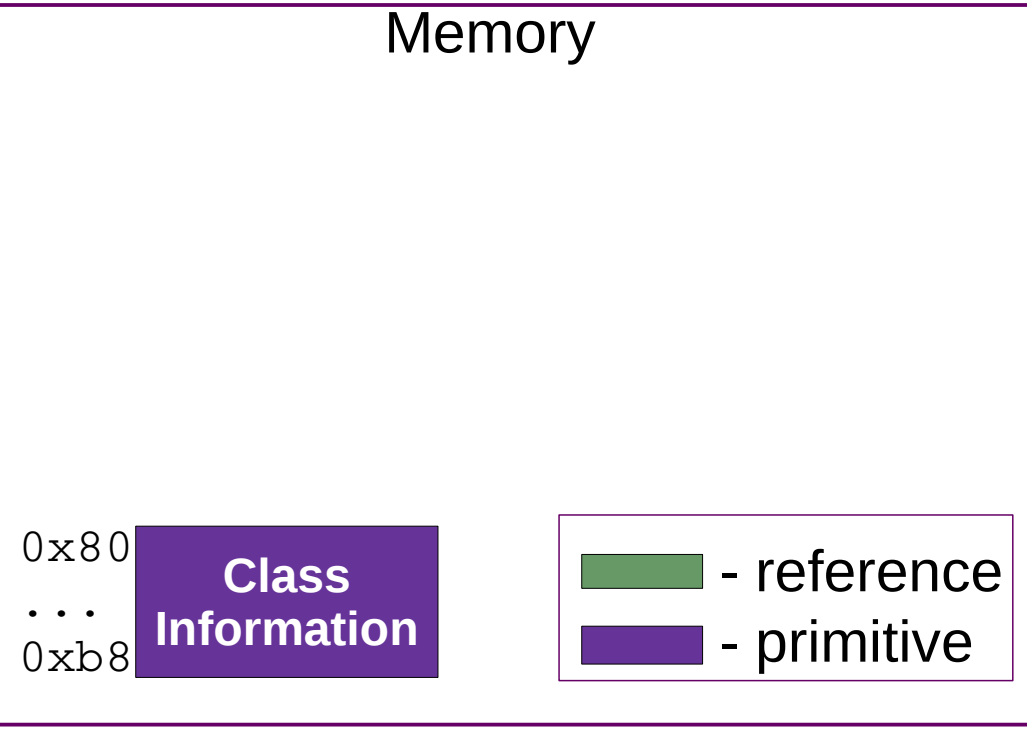
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

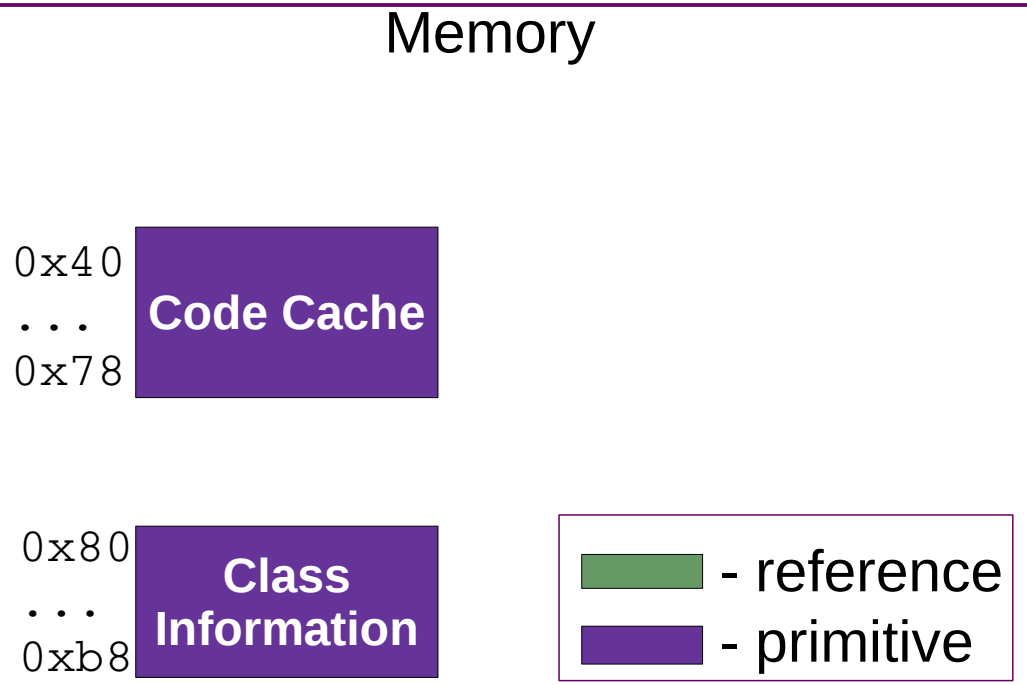
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

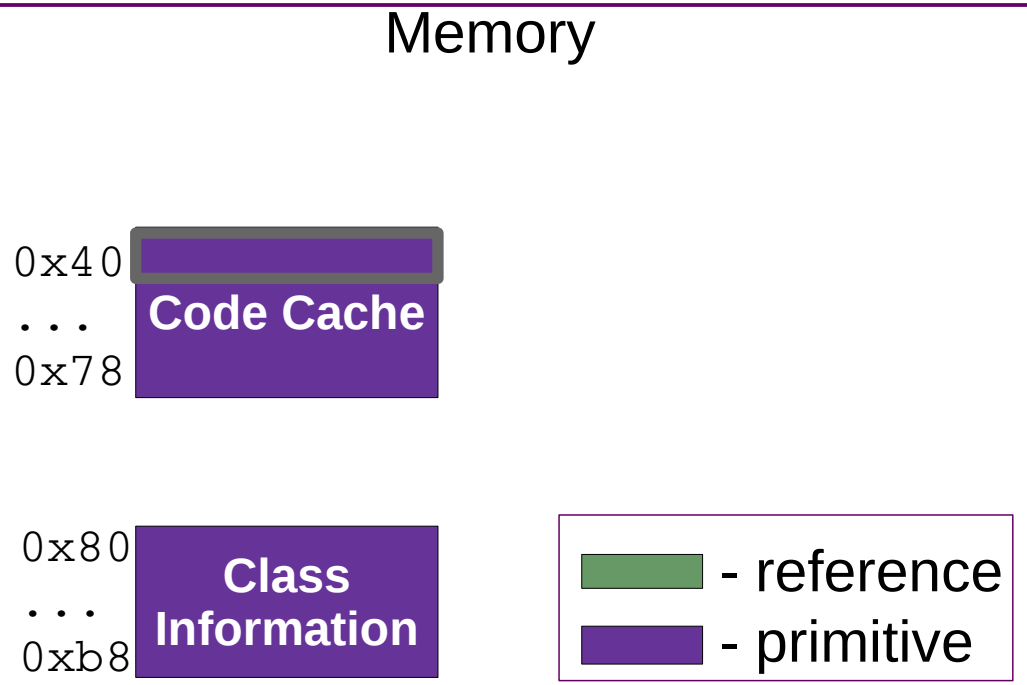
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

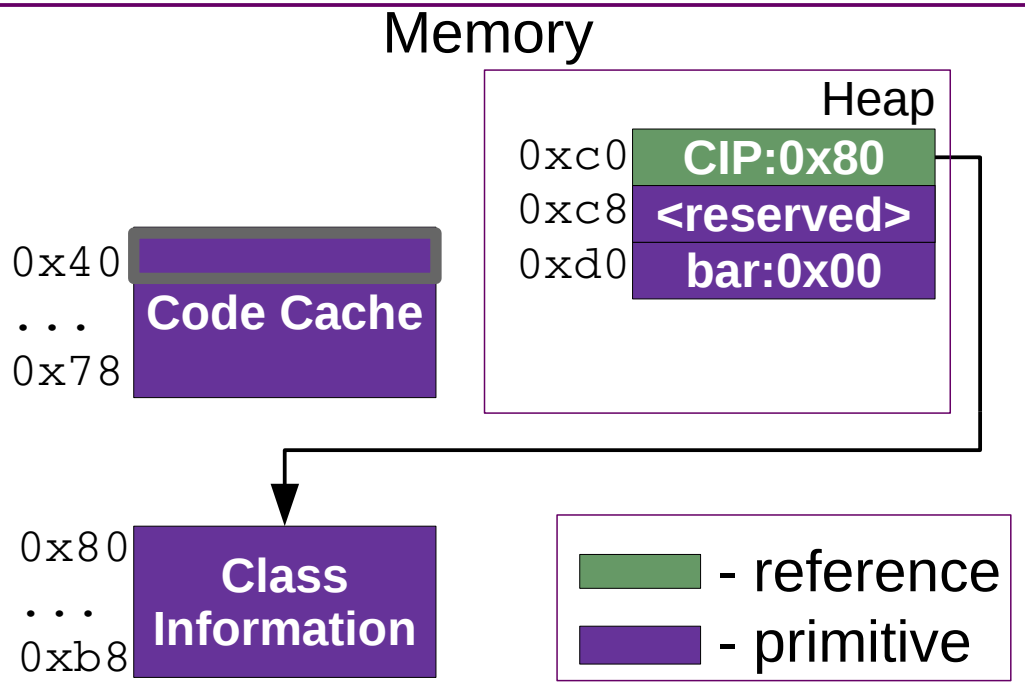
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- JIT compilation and interpretation
- Automatic memory management
- Object orientation and associated metadata

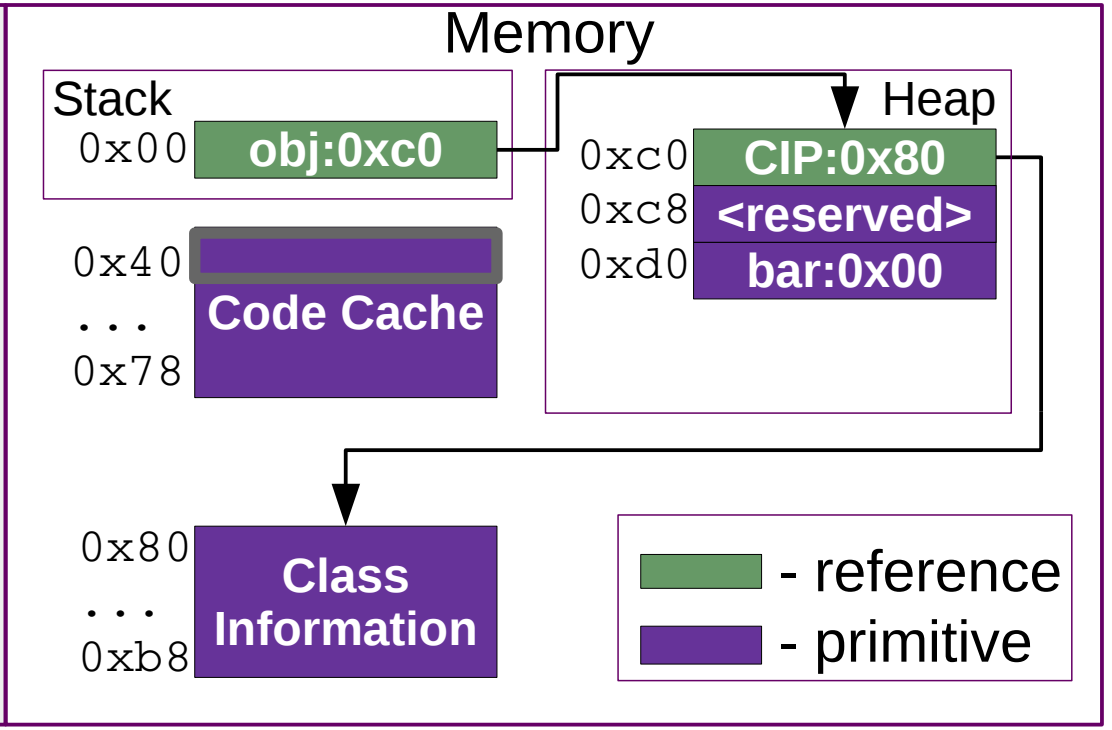
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

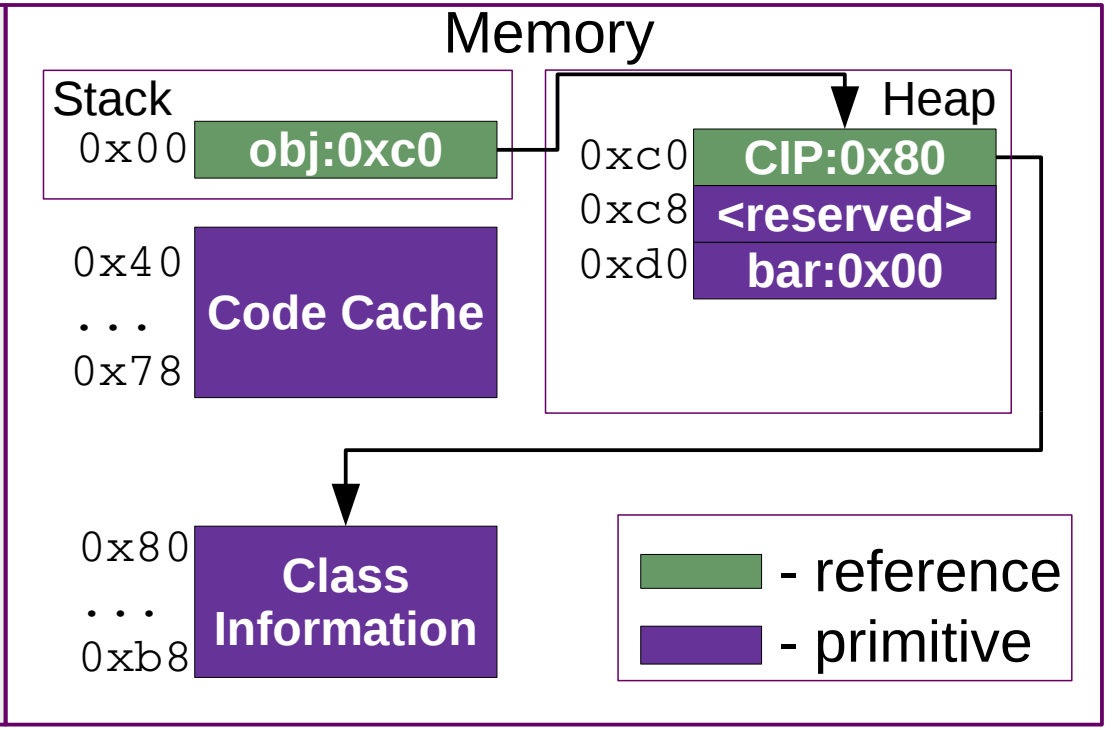
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

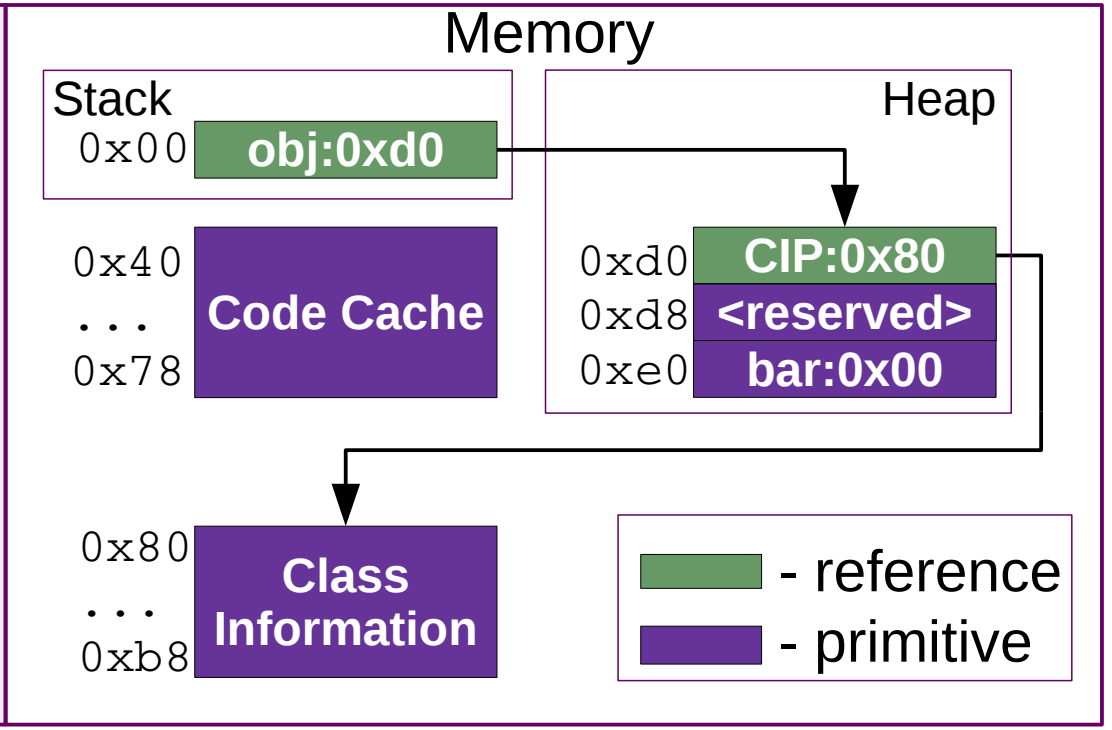
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

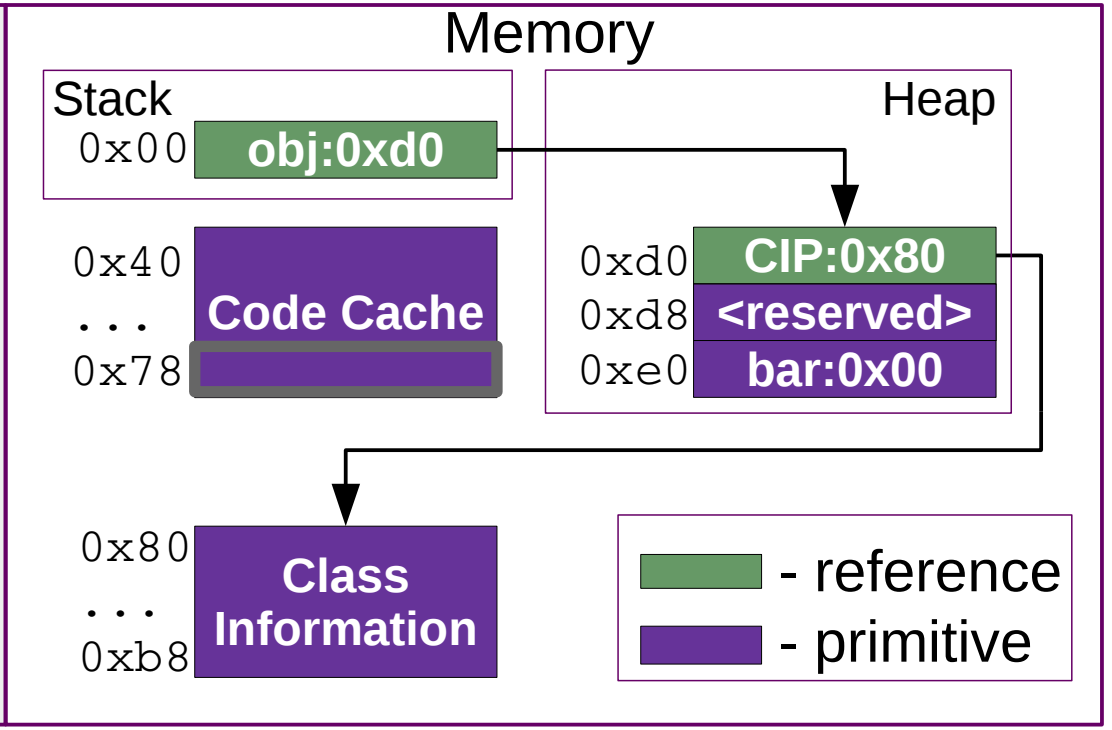
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

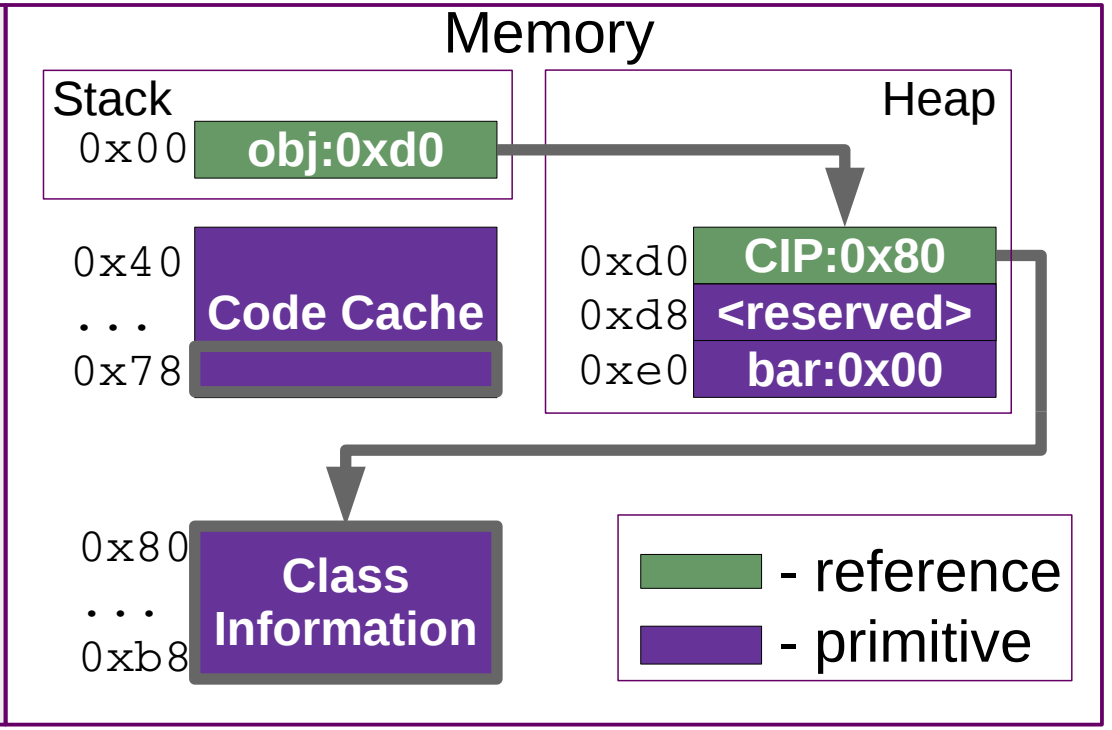
What simulation platform for managed applications is needed and why?

The University of Manchester

Specific Characteristics of Managed Applications

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Object obj = new Foo();
  ...
  // GC can happen.
  ...
  // Type introspection.
  if (obj instanceof Foo) {
    ...
  }
}
```



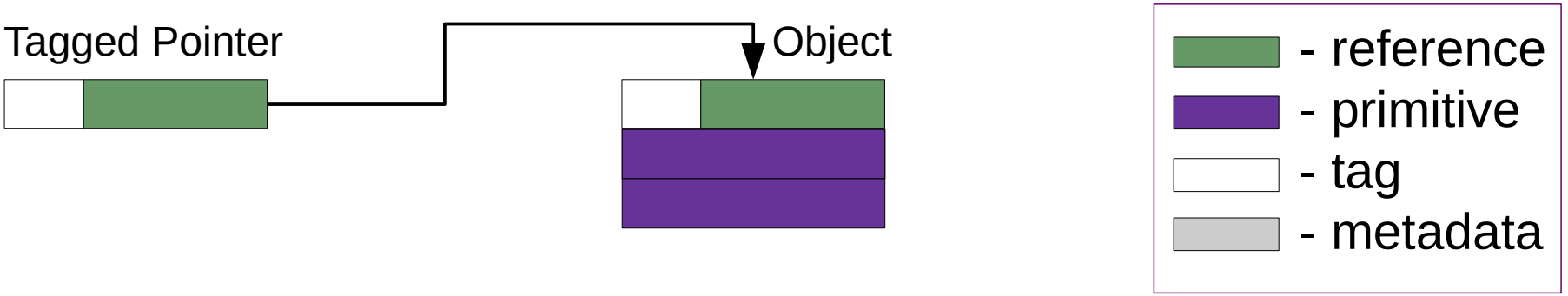
- Distributed in the verifiable bytecode format
- Automatic memory management
- JIT compilation and interpretation
- Object orientation and associated metadata

What simulation platform for managed applications is needed and why?

The University of Manchester

Support for Tagged Pointers

- An option for object metadata storage



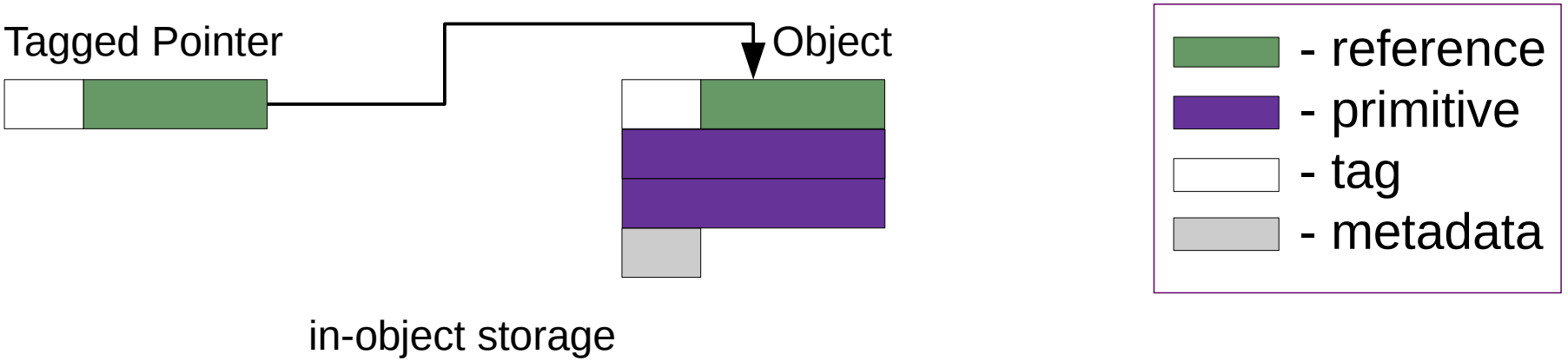
- Support in commodity 64-bit architectures
  - AArch64: [tag:8b | pointer:48b]
  - SPARC M7: [tag:8b | pointer:48b] - [tag:32b | pointer:32b]
  - x86-64: [signExtension | pointer:(48b|57b)]

What simulation platform for managed applications is needed and why?

The University of Manchester

Support for Tagged Pointers

- An option for object metadata storage



in-object storage

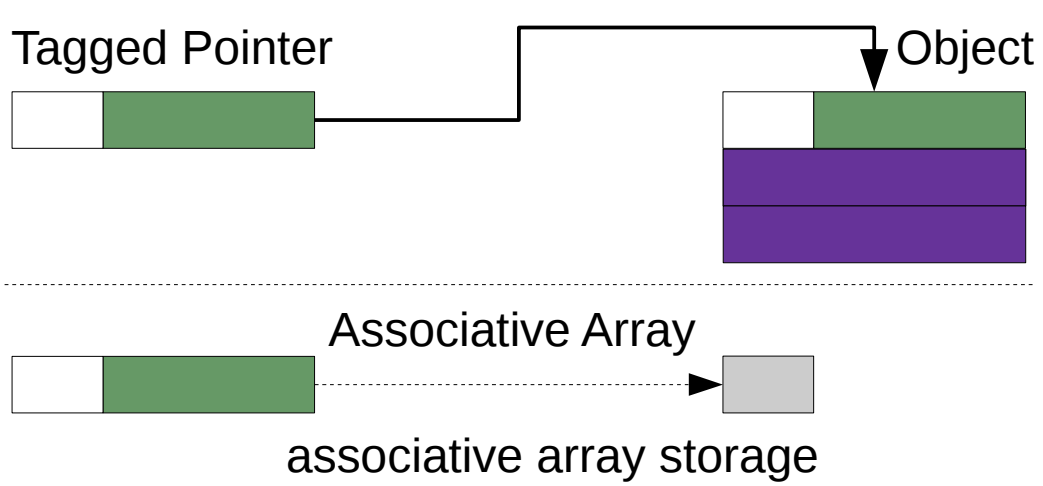
- Support in commodity 64-bit architectures
  - AArch64: [tag:8b | pointer:48b]
  - SPARC M7: [tag:8b | pointer:48b] - [tag:32b | pointer:32b]
  - x86-64: [signExtension | pointer:(48b|57b)]

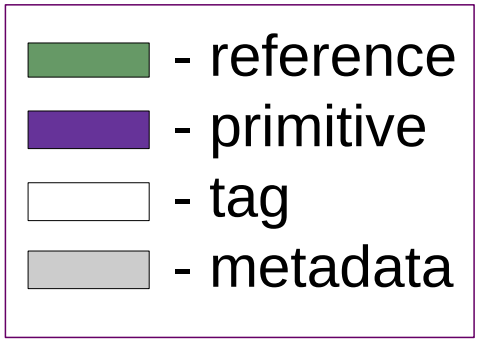
What simulation platform for managed applications is needed and why?

The University of Manchester

Support for Tagged Pointers

- An option for object metadata storage





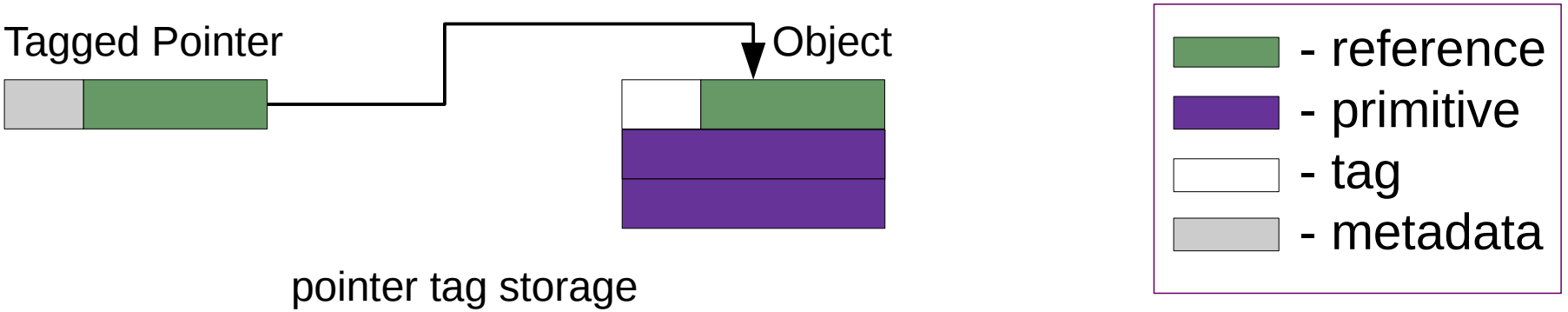
- Support in commodity 64-bit architectures
  - AArch64: [tag:8b | pointer:48b]
  - SPARC M7: [tag:8b | pointer:48b] - [tag:32b | pointer:32b]
  - x86-64: [signExtension | pointer:(48b|57b)]

What simulation platform for managed applications is needed and why?

The University of Manchester

Support for Tagged Pointers

- An option for object metadata storage



- Support in commodity 64-bit architectures
  - AArch64: [tag:8b | pointer:48b]
  - SPARC M7: [tag:8b | pointer:48b] - [tag:32b | pointer:32b]
  - x86-64: [signExtension | pointer:(48b|57b)]

What simulation platform for managed applications is needed and why?

The University of Manchester

Design Goals

- Productivity for research
  - VM modularity and support of other languages
  - High simulation speed (DaCapo benchmarks in one day on a single PC)
- Awareness of the VM in the simulator
- Advanced features
  - Support of tagged 64-bit pointers
  - Ability to experiment with different object layouts
  - Ability to perform power and energy modeling

VM Selection Justification: Maxine VM

The University of Manchester

Maxine VM[1]: A Platform for Research in VM Technology

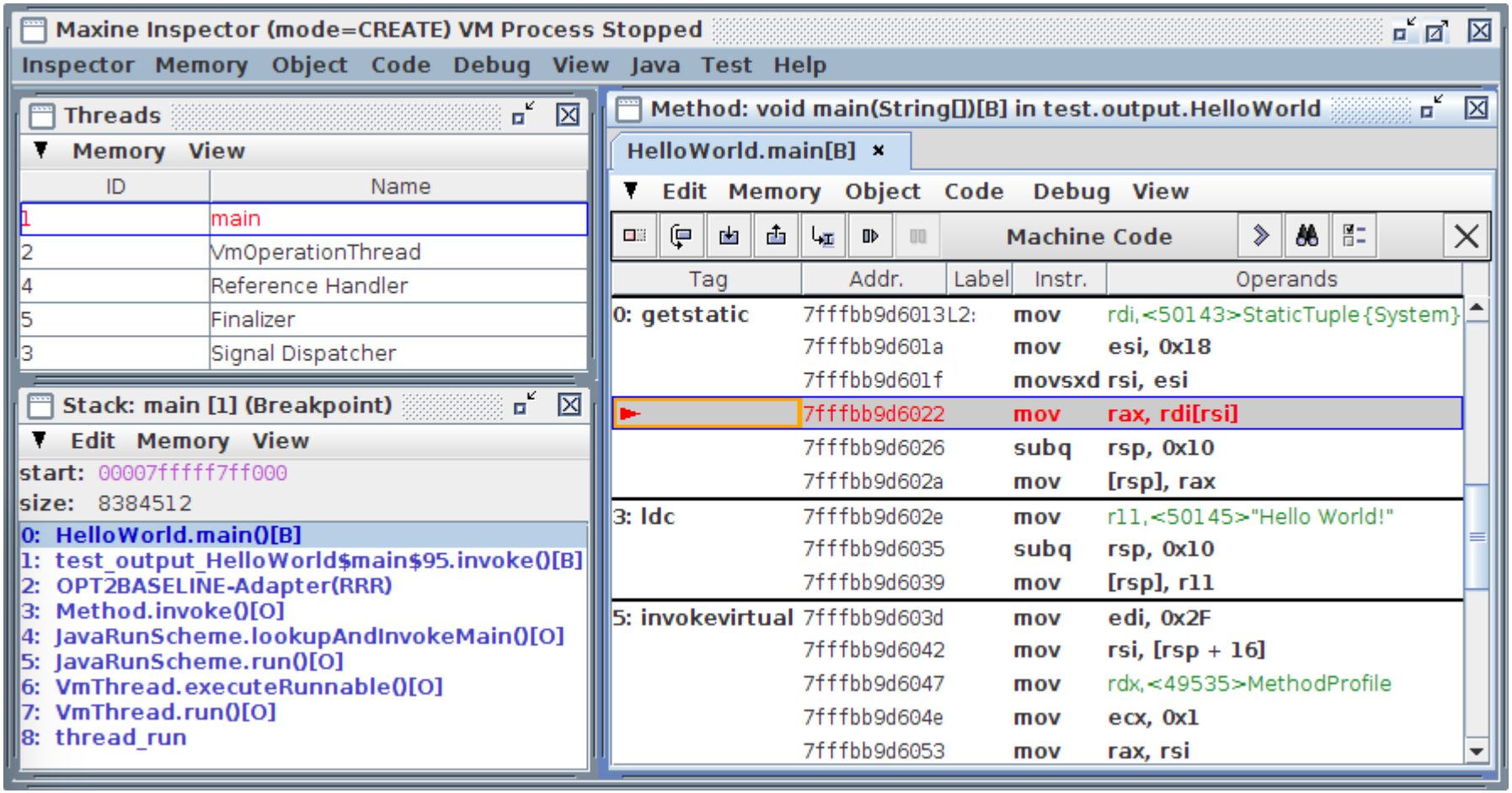
- Mostly written in Java, with a substrate written in C
- Modular design: schemes for object layouts, object references, heap and GC, thread synchronization, etc.
- Compilers: T1X (O0), C1X (O1), Graal (O2)
  - Graal supports other languages via Truffle (JavaScript, R, Ruby, others)
- Target ISAs: x86-64, ARMv7
- Class library: JDK 7

[1] Wimmer *et al.,* “Maxine: An approachable virtual machine for, and in, Java”, TACO, 2013

VM Selection Justification: Maxine VM

The University of Manchester

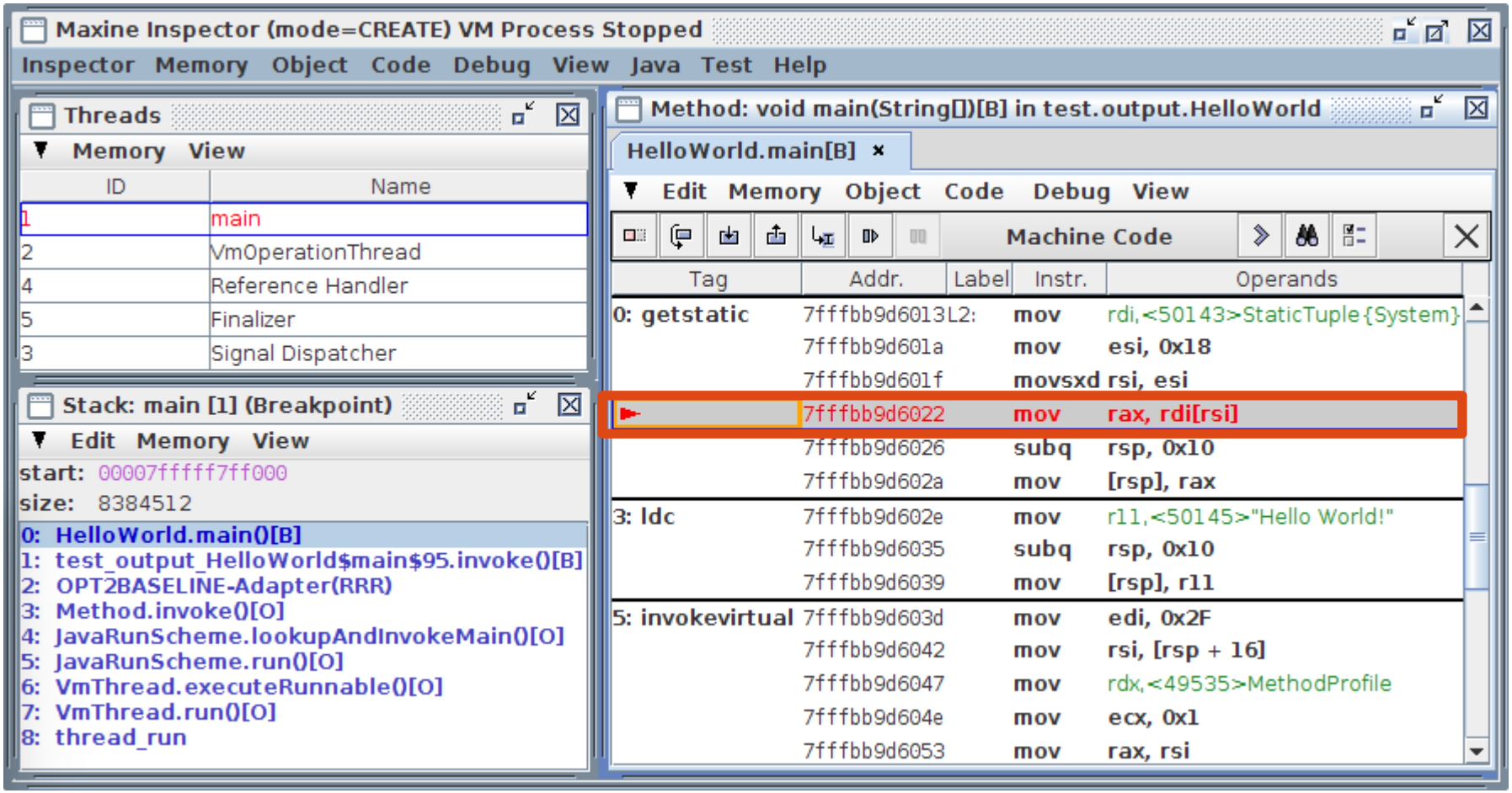
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

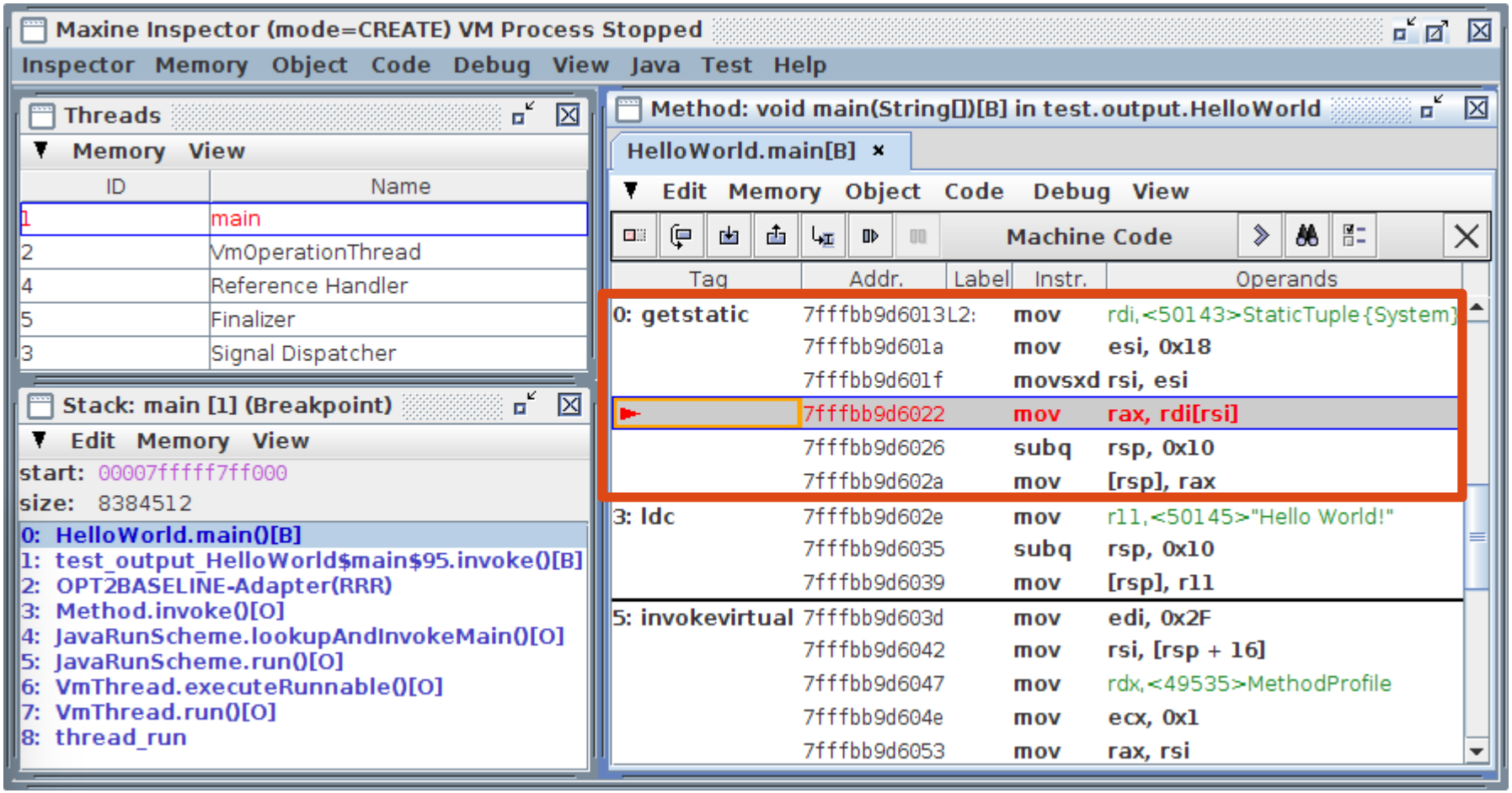
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

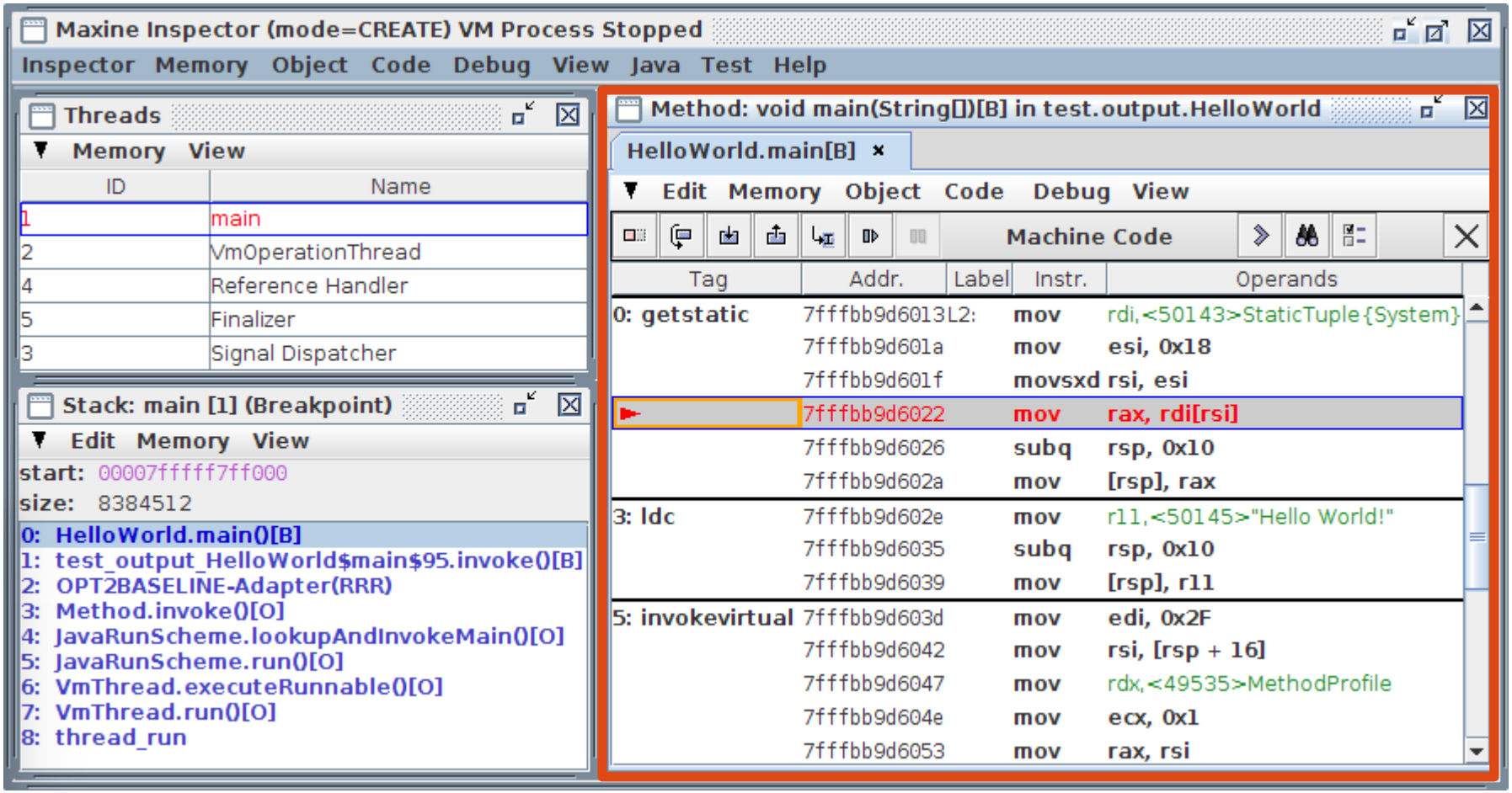
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

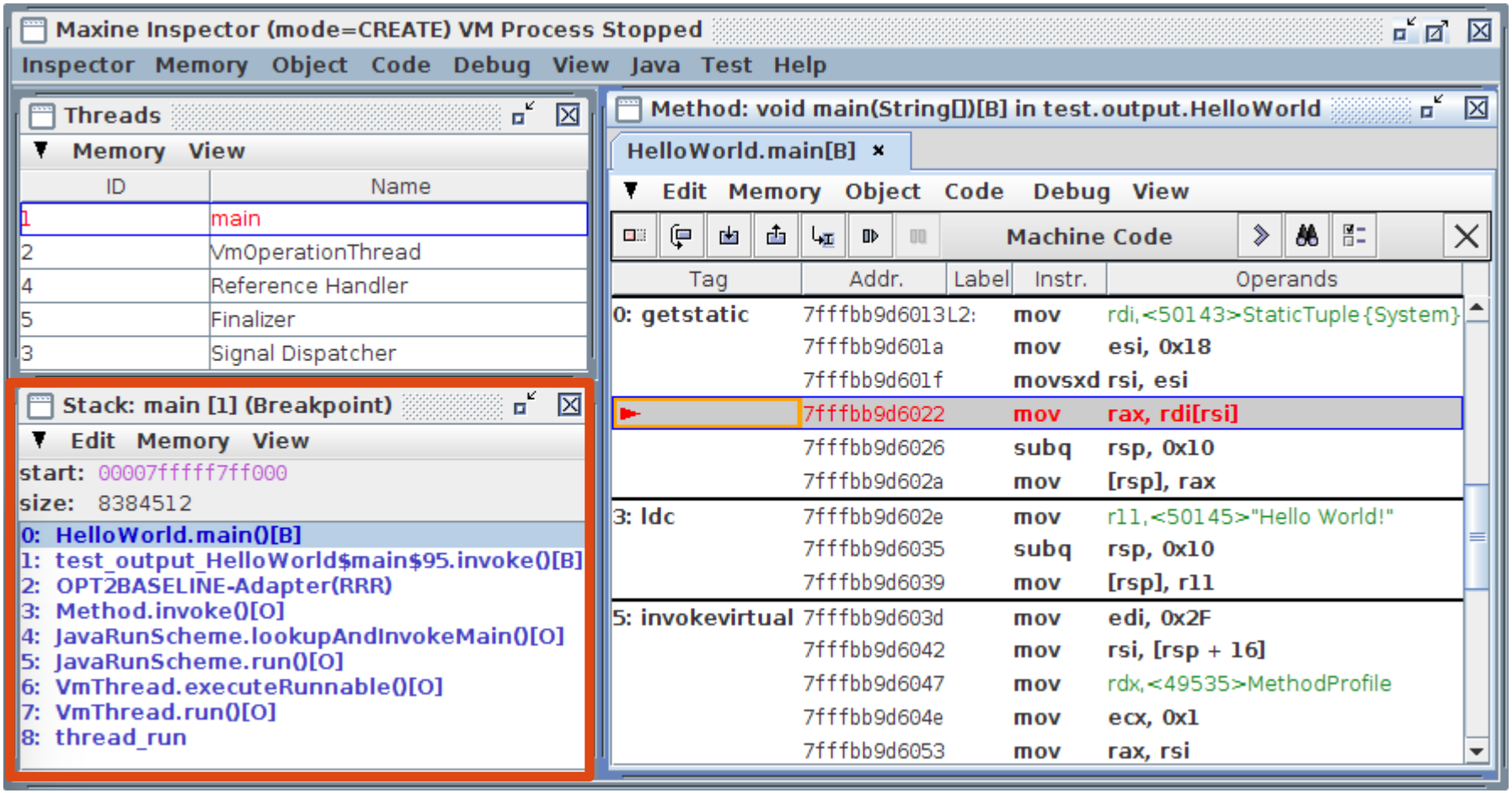
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

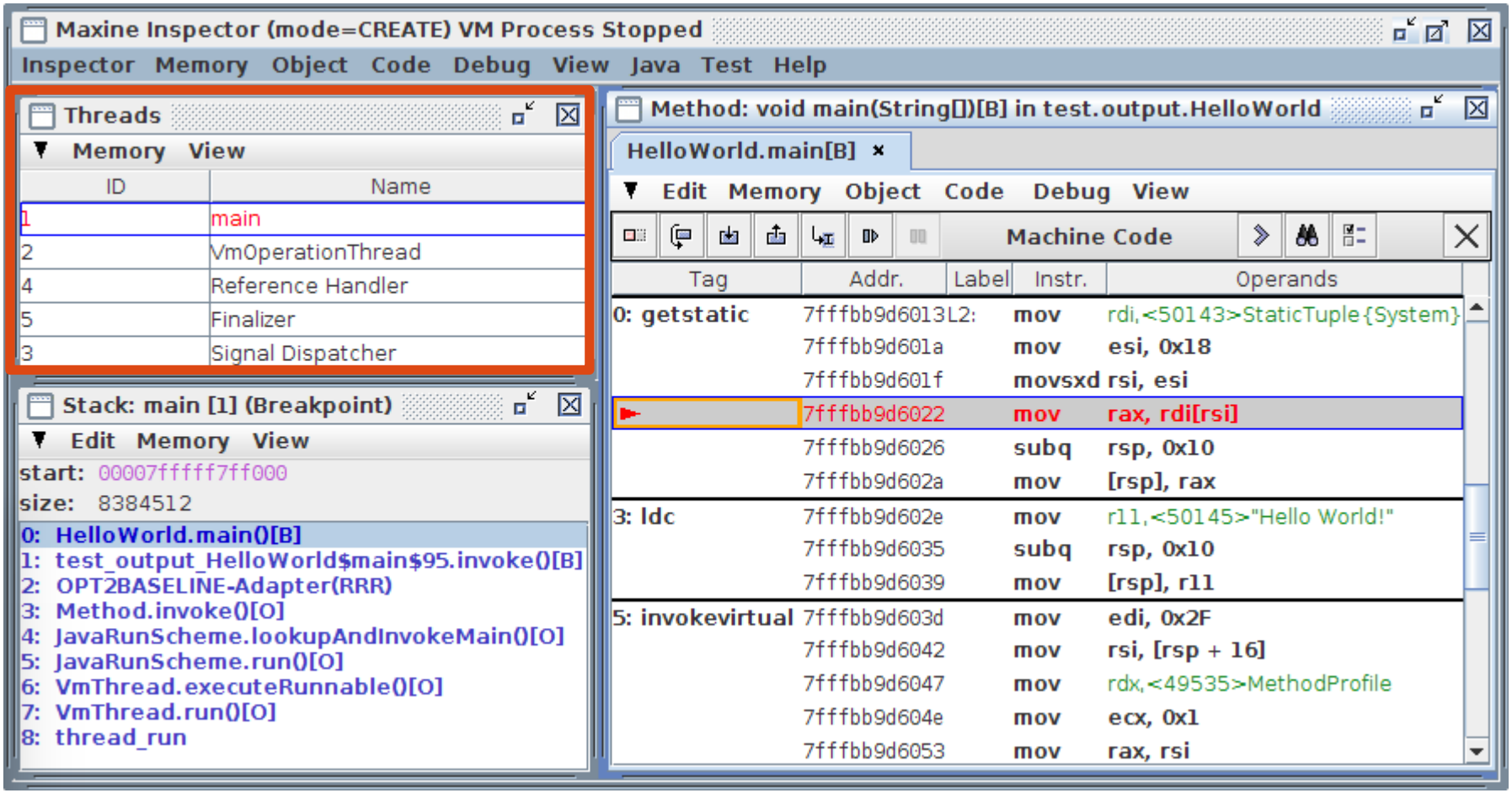
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

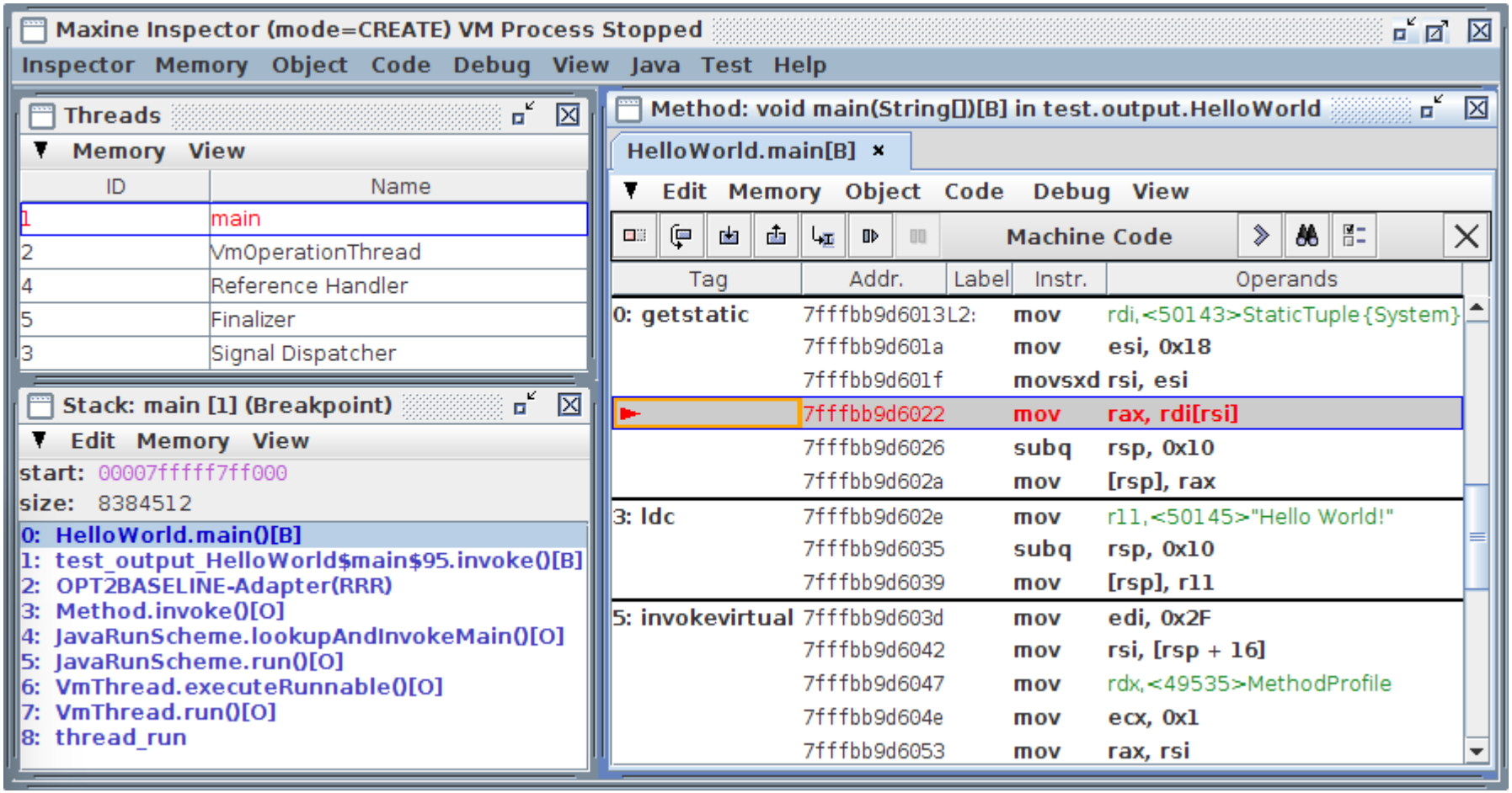
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

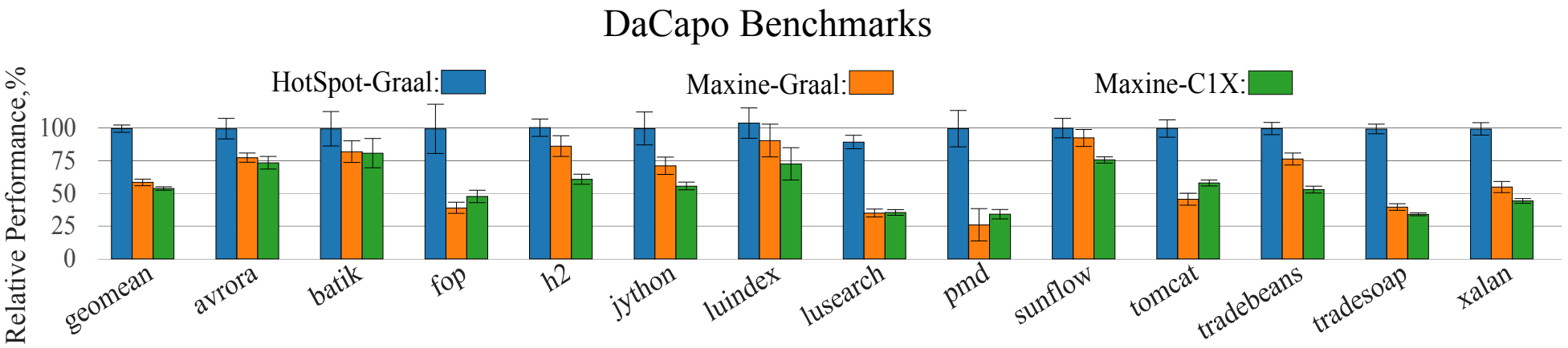
Maxine Inspector: Integrated Debugging Support



VM Selection Justification: Maxine VM

The University of Manchester

Maxine VM: Performance Comparison Against Hotspot VM

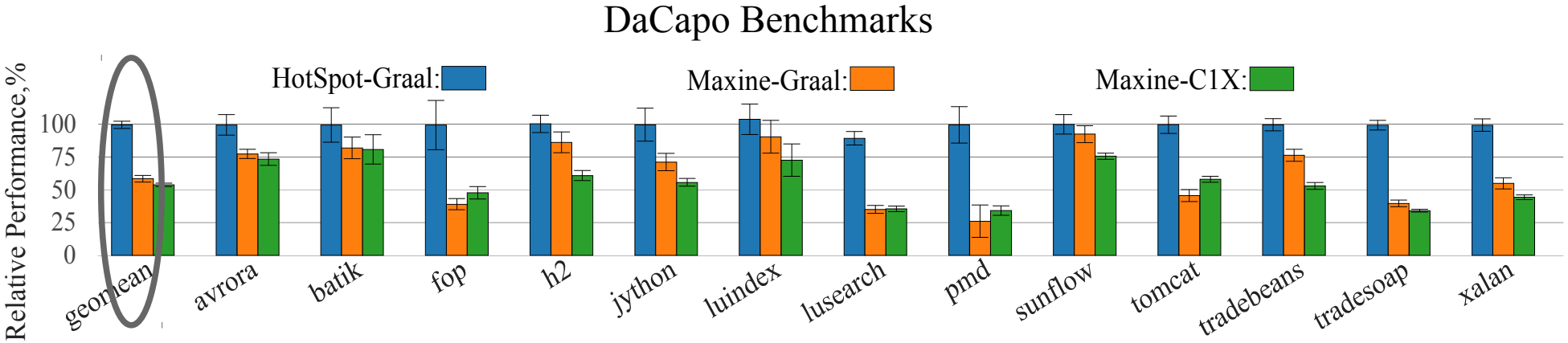


- Maxine VM performance is ~59% of the highly optimized Hotspot VM
- Graal (O2) compiler delivers 8% better performance than C1X (O1)

VM Selection Justification: Maxine VM

The University of Manchester

Maxine VM: Performance Comparison Against Hotspot VM

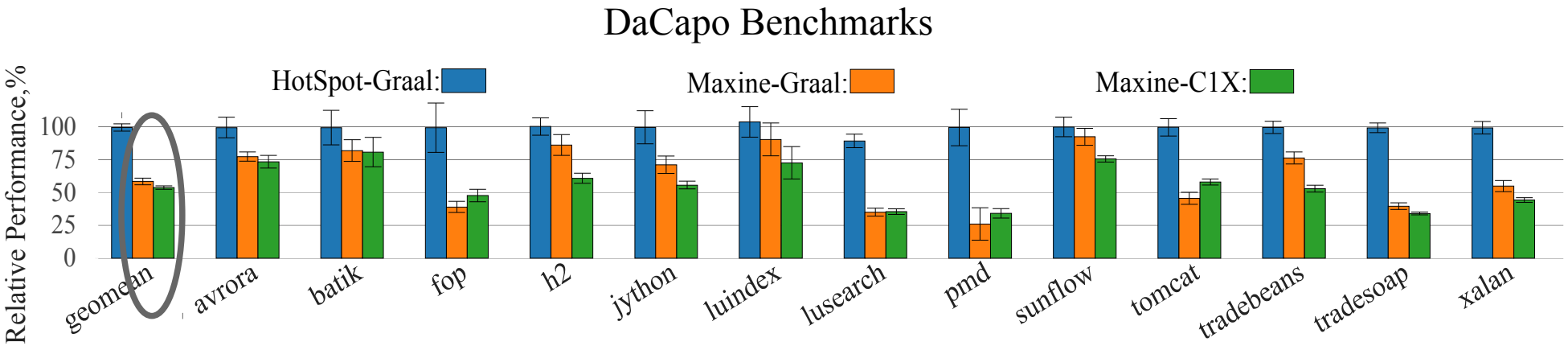


- Maxine VM performance is ~59% of the highly optimized Hotspot VM
- Graal (O2) compiler delivers 8% better performance than C1X (O1)

VM Selection Justification: Maxine VM

The University of Manchester

Maxine VM: Performance Comparison Against Hotspot VM



- Maxine VM performance is ~59% of the highly optimized Hotspot VM
- Graal (O2) compiler delivers 8% better performance than C1X (O1)

Simulator Selection Justification: ZSim

The University of Manchester

ZSim[1]: Fast and Accurate Microarchitectural Simulation

- x86-64 execution-driven timing simulator based on Pin
- Bound-weave technique for scalable simulation
- Lightweight user-level virtualization
- Comparison with open simulators supporting managed applications

Simulator	Engine	Full-System	Simulation Speed
gem5	Emulation	yes	~100-300 KIPS
Sniper *	DBT	no	~1-3 MIPS
ZSim	DBT	no	~7-20 MIPS

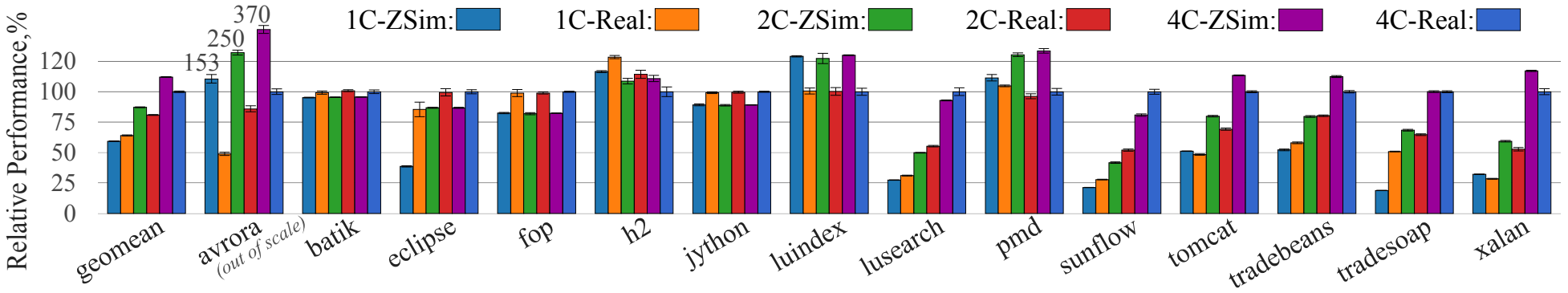
[1] Sanchez *et al.* "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems", ISCA, 2013

* Sniper can simulate DaCapo benchmarks on 32-bit Jikes RVM only.

Simulator Selection Justification: ZSim

The University of Manchester

ZSim Validation: DaCapo on Maxine VM

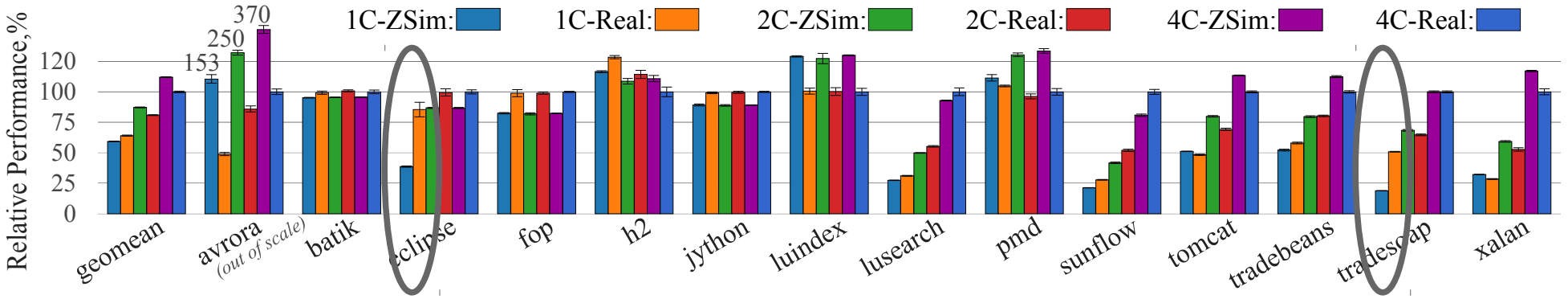


- 100% pass rate and ~10% geomean simulation error at ~12 MIPS
- Inconsistencies:
  - eclipse, tradesoap (1C-*): Round Robin vs CFS scheduling
  - avrora: spends more than 50% of execution in the kernel

Simulator Selection Justification: ZSim

The University of Manchester

ZSim Validation: DaCapo on Maxine VM

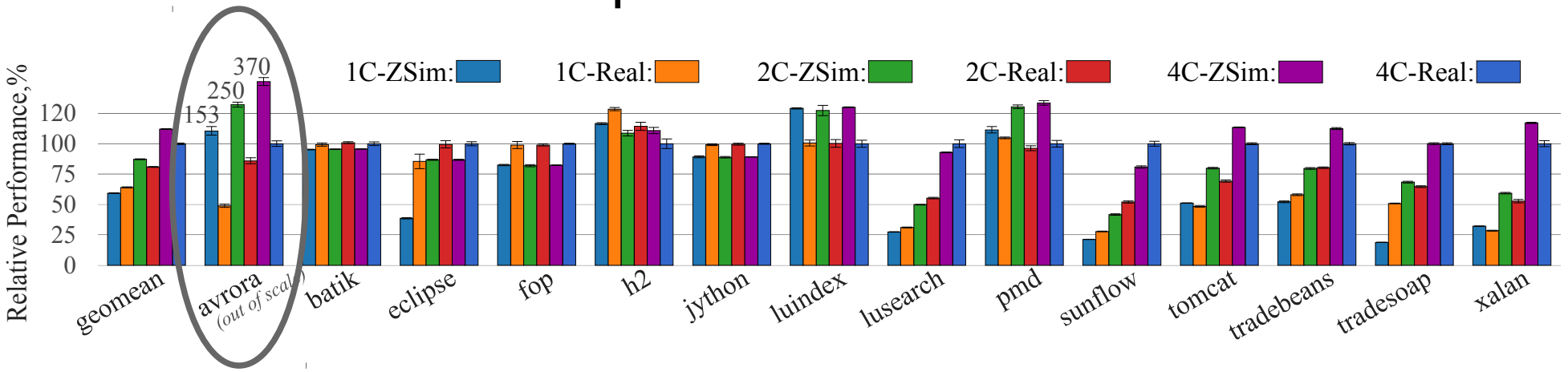


- 100% pass rate and ~10% geomean simulation error at ~12 MIPS
- Inconsistencies:
  - eclipse, tradesoap (1C-*): Round Robin vs CFS scheduling
  - avrora: spends more than 50% of execution in the kernel

Simulator Selection Justification: ZSim

The University of Manchester

ZSim Validation: DaCapo on Maxine VM

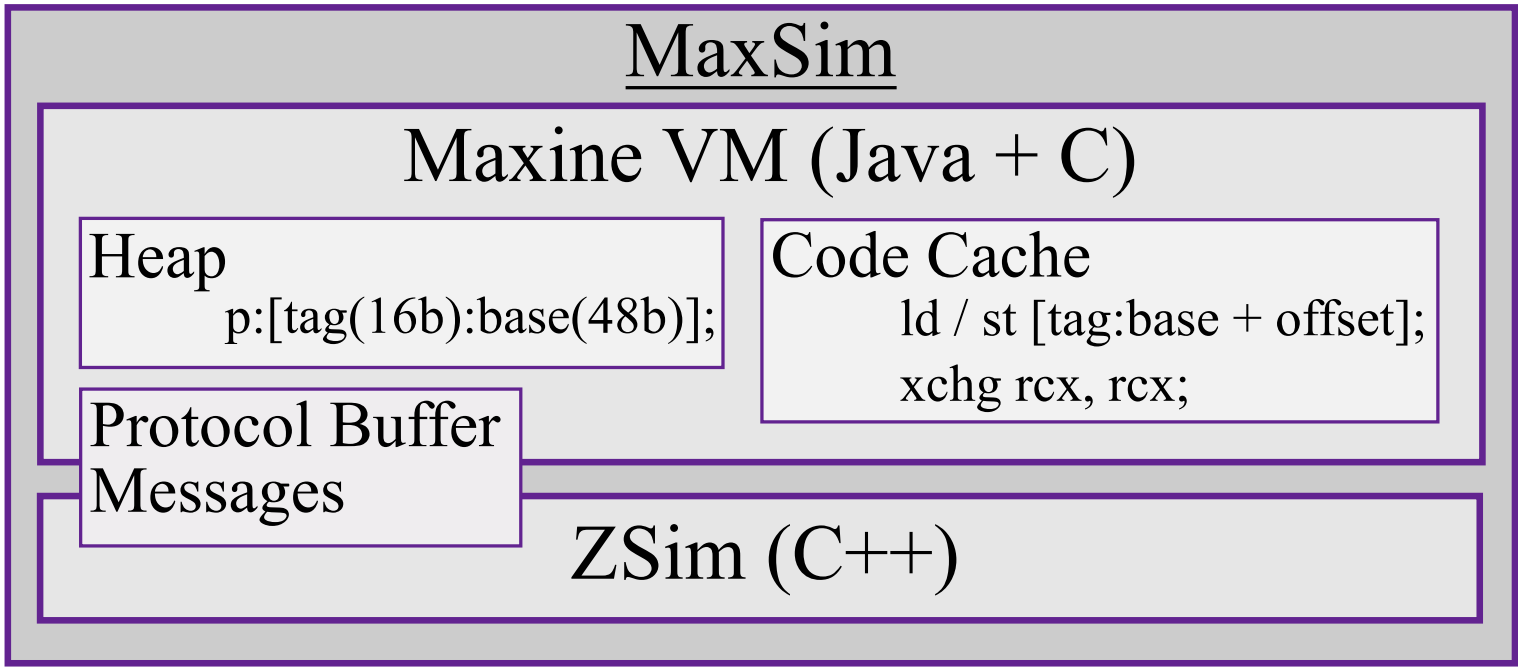


- 100% pass rate and ~10% geomean simulation error at ~12 MIPS
- Inconsistencies:
  - eclipse, tradesoap (1C-*): Round Robin vs CFS scheduling
  - avrora: spends more than 50% of execution in the kernel

MaxSim: Overview and Features

The University of Manchester

Maxine-ZSim Integration Scheme

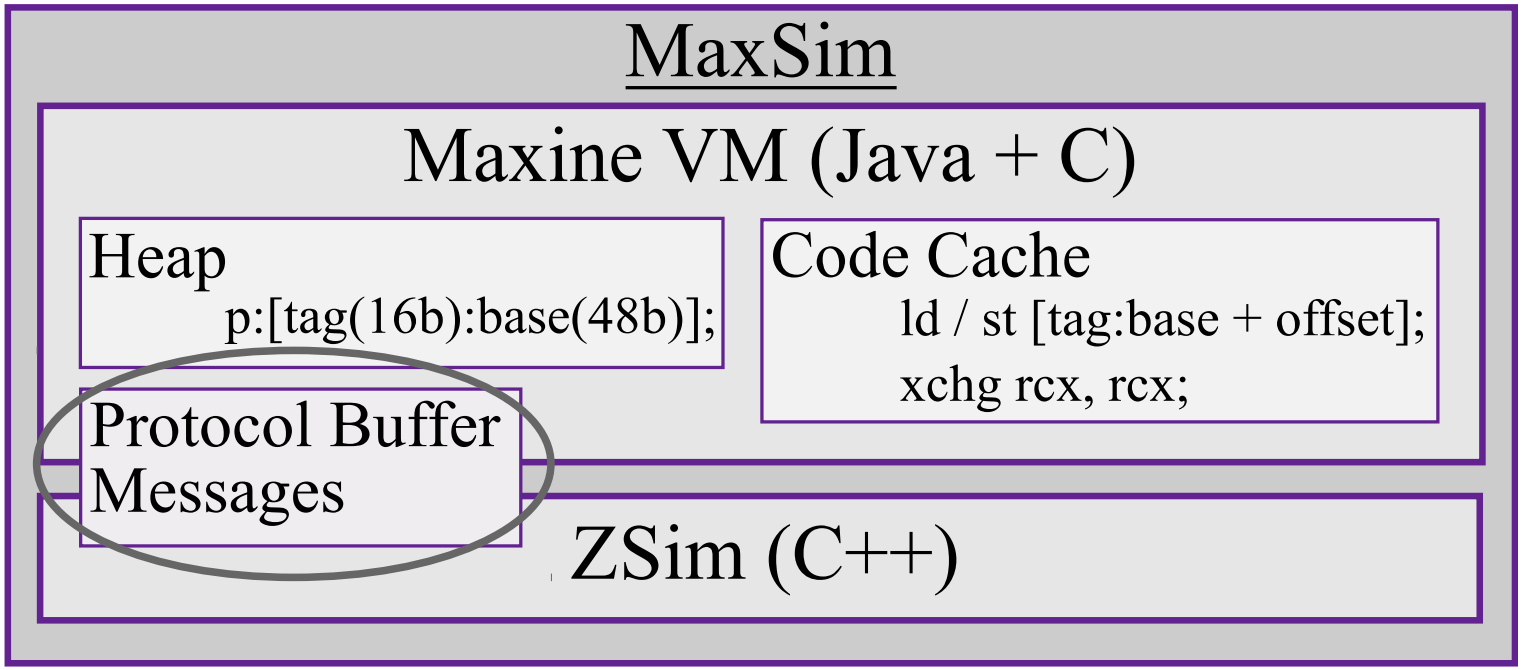


- Protocol Buffer Messages
  - Interface definition
  - Configuration
  - Profile serialization
- Magic NOPs
  - Simulation control
  - VM awareness
  - Sending/receiving protocol buffer messages
- Tagged Pointers
  - VM awareness
  - Profiling

MaxSim: Overview and Features

The University of Manchester

Maxine-ZSim Integration Scheme

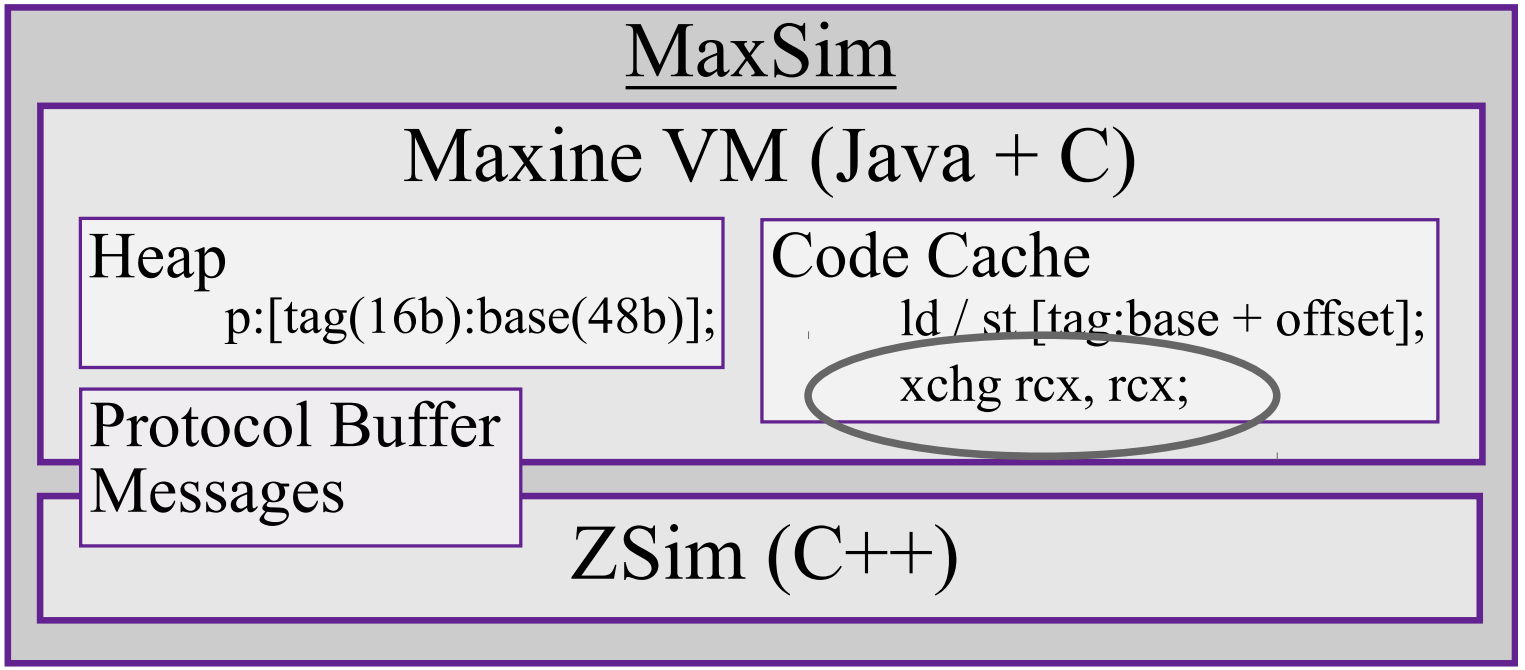


- Protocol Buffer Messages
  - Interface definition
  - Configuration
  - Profile serialization
- Magic NOPs
  - Simulation control
  - VM awareness
  - Sending/receiving protocol buffer messages
- Tagged Pointers
  - VM awareness
  - Profiling

MaxSim: Overview and Features

The University of Manchester

Maxine-ZSim Integration Scheme

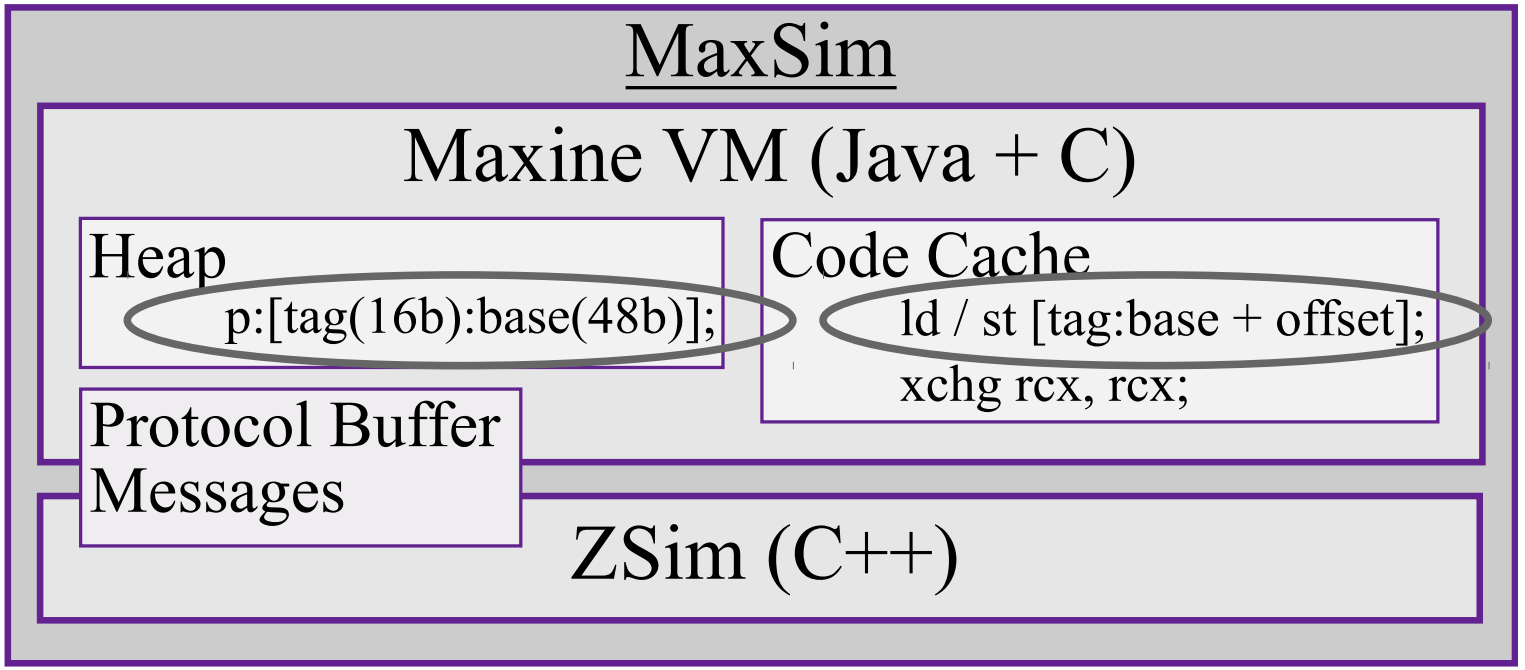


- Protocol Buffer Messages
  - Interface definition
  - Configuration
  - Profile serialization
- Magic NOPs
  - Simulation control
  - VM awareness
  - Sending/receiving protocol buffer messages
- Tagged Pointers
  - VM awareness
  - Profiling

MaxSim: Overview and Features

The University of Manchester

Maxine-ZSim Integration Scheme



- Protocol Buffer Messages
  - Interface definition
  - Configuration
  - Profile serialization
- Magic NOPs
  - Simulation control
  - VM awareness
  - Sending/receiving protocol buffer messages
- Tagged Pointers
  - VM awareness
  - Profiling

MaxSim: Overview and Features

The University of Manchester

VM Awareness in the Simulator

- VM memory regions
  - Stack
  - TLS
  - Heap
  - Code cache
  - Native code
  - Others
- VM operations
  - Garbage collection
  - Object allocation
- Object binding
  - To its class
  - To its allocation site

MaxSim: Overview and Features

The University of Manchester

Pointer Tagging

- Two types of pointer tagging are supported
  - Class ID tagging
  - Allocation site ID tagging

```
// Example of a class.
class Foo {
  public long bar;
}

// Source code example.
{
  // Allocation site.
  Foo obj = new Foo();
  obj.bar = 42;
}
```

- Tagging/untagging of all pointers at arbitrary places of execution
  - Enables simulation fast-forwarding

- After tagging the following properties are preserved:
  - Pointers to the same object are tagged with the same tag
  - Tags are immutable between an allocation and a garbage collection
  - Objects are accessed using `[tag:base + offset]` addressing mode

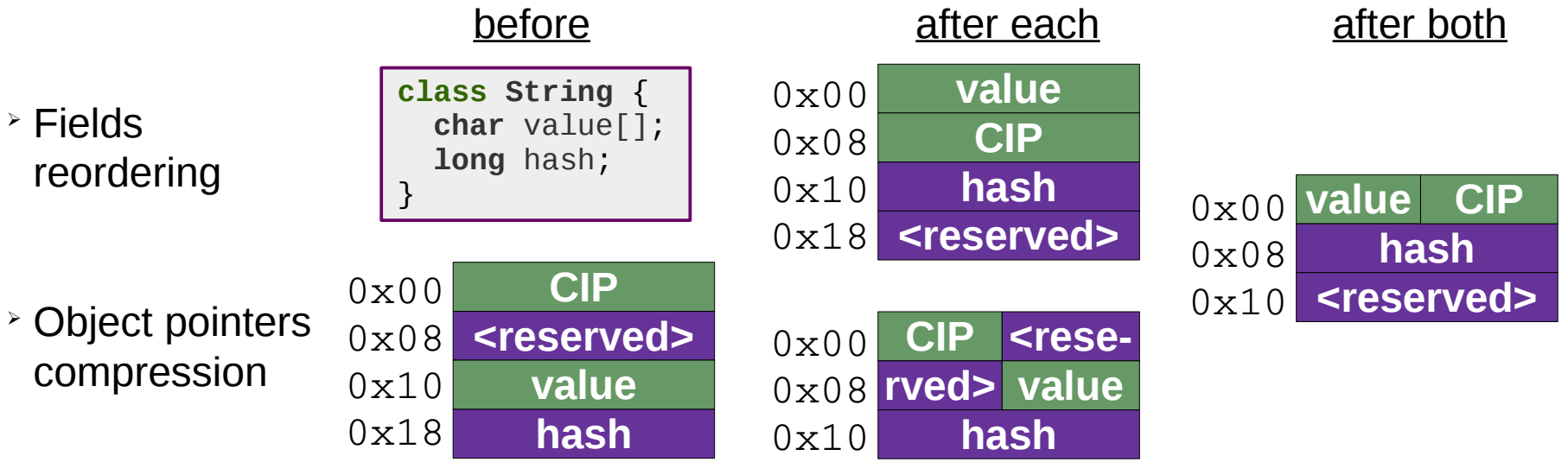
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing

- Motivation: easy experimentation with object layouts without adding extra complexity or breaking modularity of Maxine VM

- Supports two object layout transformations



- Makes use of two properties of MaxSim
  - Flexibility of Maxine VM to expand object fields
  - Ability of ZSim to remap memory addresses

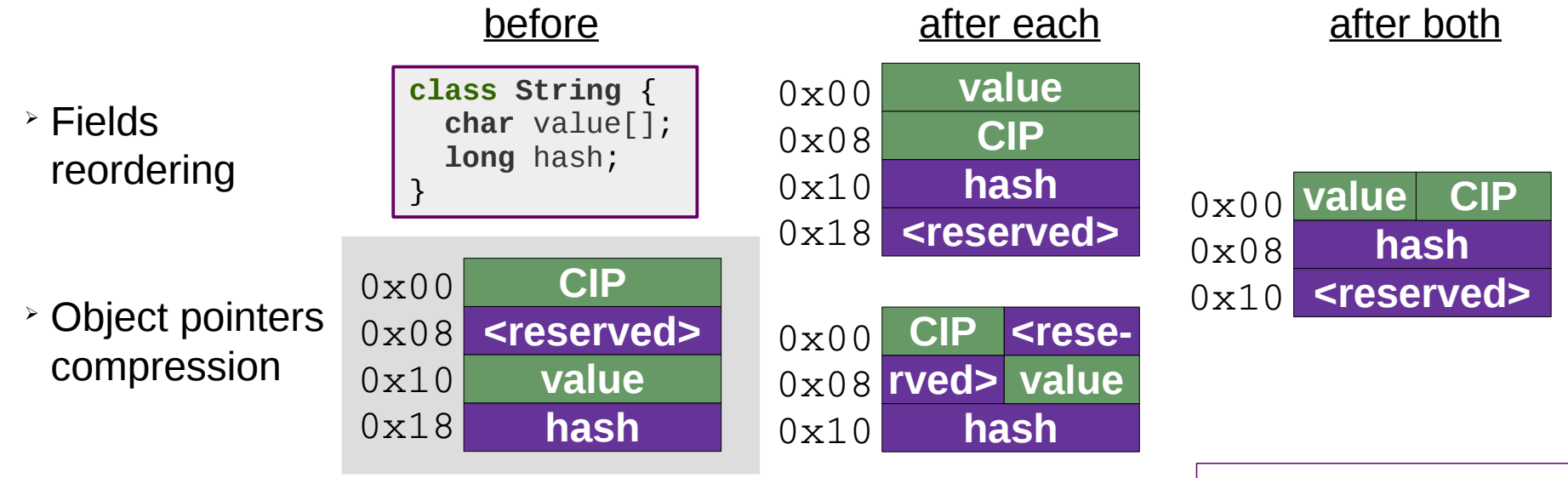
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing

- Motivation: easy experimentation with object layouts without adding extra complexity or breaking modularity of Maxine VM

- Supports two object layout transformations



- Makes use of two properties of MaxSim
  - Flexibility of Maxine VM to expand object fields
  - Ability of ZSim to remap memory addresses

MaxSim: Overview and Features

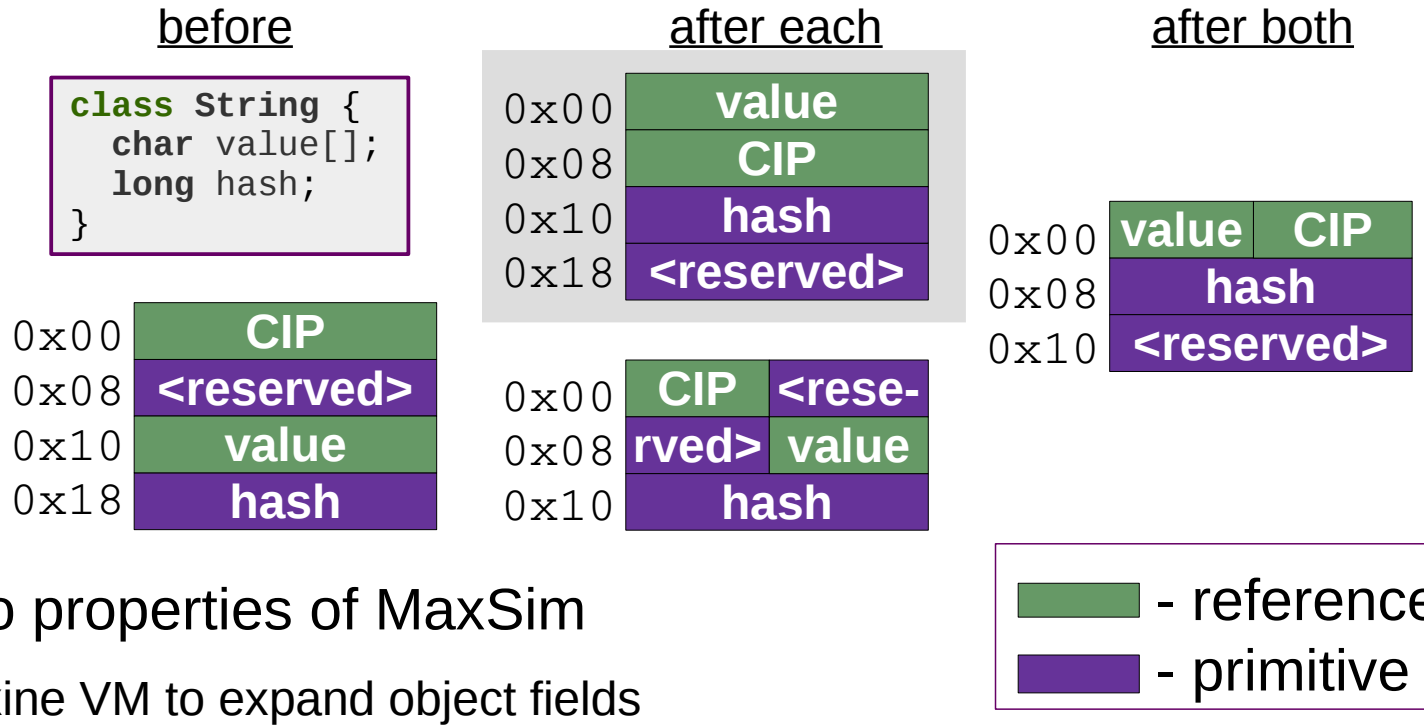
The University of Manchester

Address Space Morphing

- Motivation: easy experimentation with object layouts without adding extra complexity or breaking modularity of Maxine VM

- Supports two object layout transformations

  - Fields reordering
  - Object pointers compression



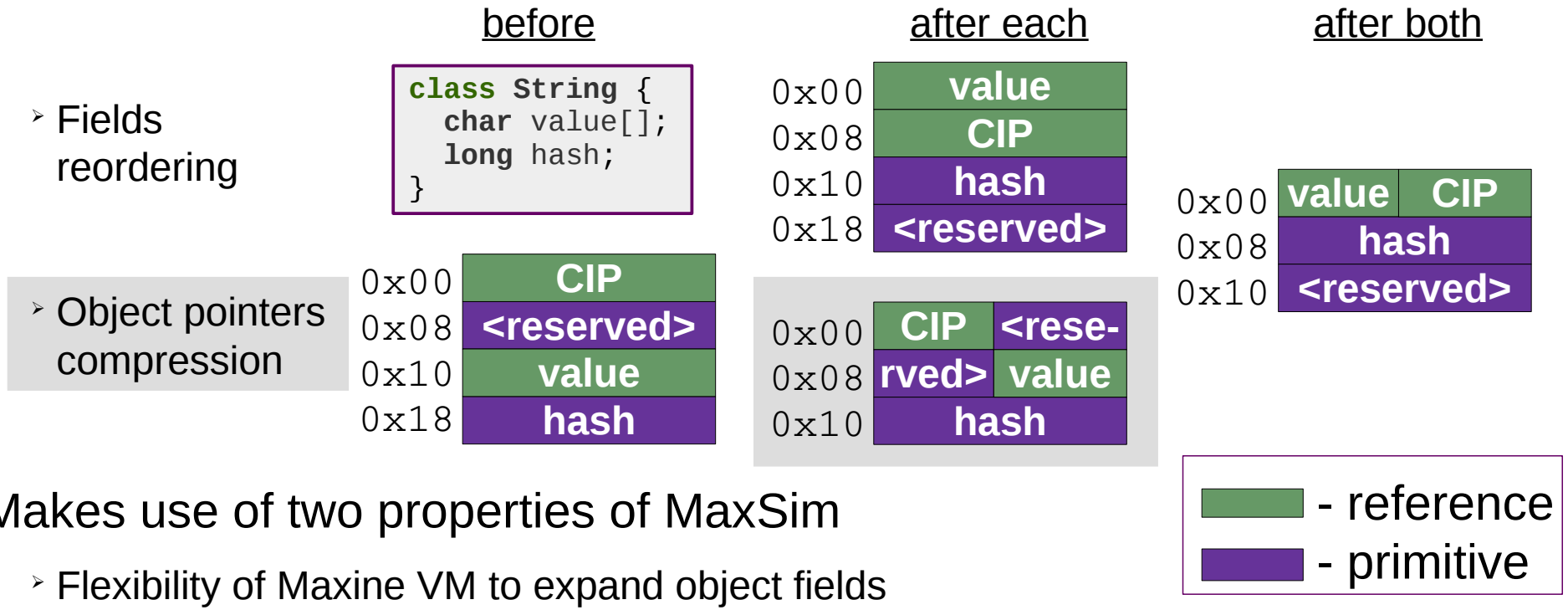
- Makes use of two properties of MaxSim
  - Flexibility of Maxine VM to expand object fields
  - Ability of ZSim to remap memory addresses

MaxSim: Overview and Features

The University of Manchester

Address Space Morphing

- Motivation: easy experimentation with object layouts without adding extra complexity or breaking modularity of Maxine VM
- Supports two object layout transformations



- Makes use of two properties of MaxSim
  - Flexibility of Maxine VM to expand object fields
  - Ability of ZSim to remap memory addresses

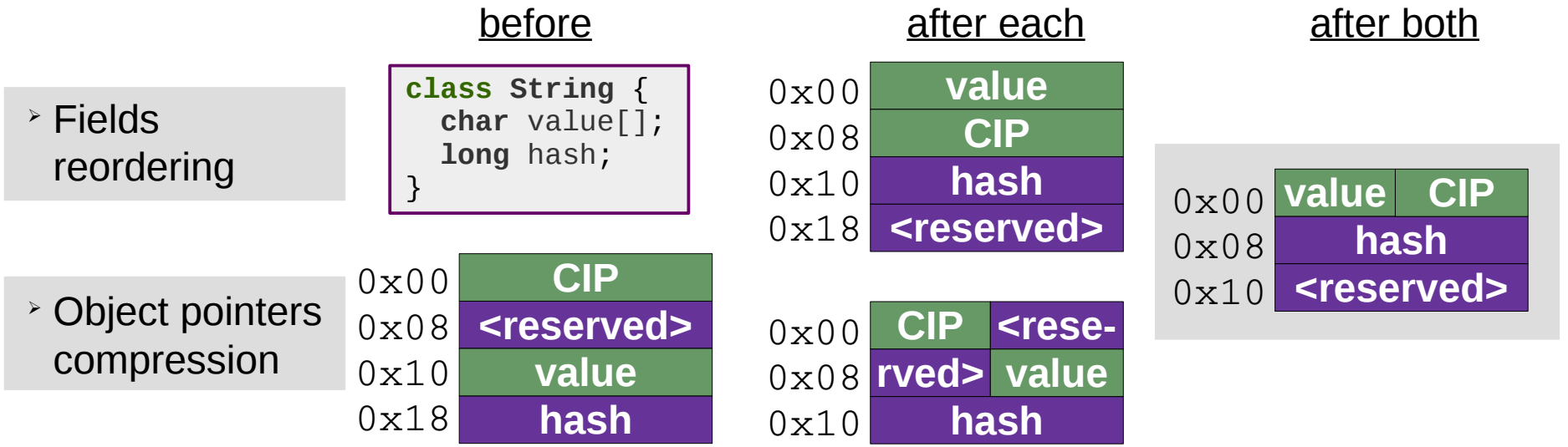
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing

- Motivation: easy experimentation with object layouts without adding extra complexity or breaking modularity of Maxine VM

- Supports two object layout transformations



- Makes use of two properties of MaxSim
  - Flexibility of Maxine VM to expand object fields
  - Ability of ZSim to remap memory addresses

MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

f (1:2)

Layout

0x00	ref.0
0x08	prim.1
0x10	ref.2
0x18	prim.3

0x00	ref.2	ref.0
0x08	prim.3	
0x10	prim.1	

Addressing

$[b_o + o_o]$

Fields Reordering Map

0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m

- reference
- primitive

MaxSim: Overview and Features

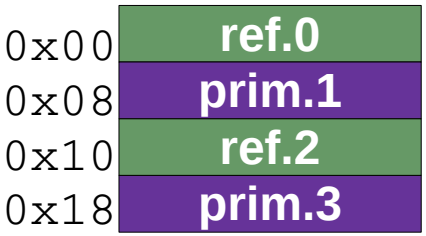
The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion

in Maxine VM →

Layout



Addressing

$[b_o + o_o]$

Fields Reordering Map

0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_o

- reference
- primitive

MaxSim: Overview and Features

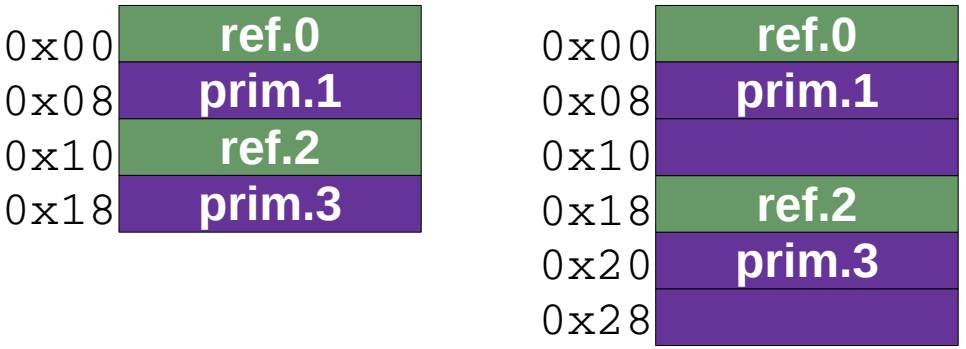
The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion

in Maxine VM

Layout



Addressing

$[b_o+o_o]$

Fields Reordering Map

0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_o

- reference
- primitive

MaxSim: Overview and Features

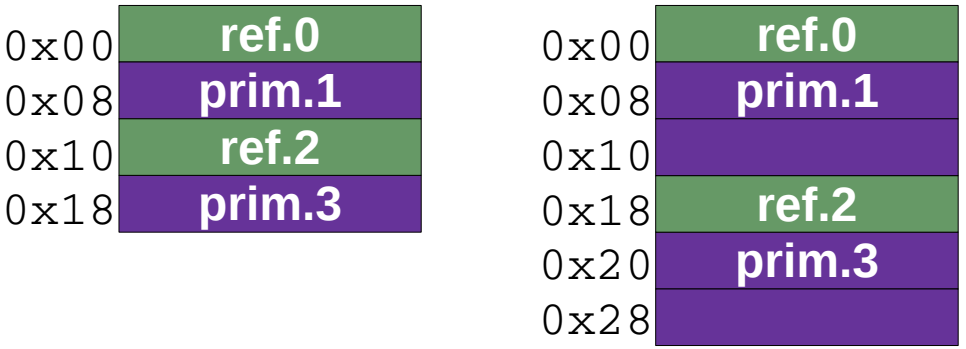
The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion

in Maxine VM →

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$

Fields Reordering Map

0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_o

- reference
- primitive

MaxSim: Overview and Features

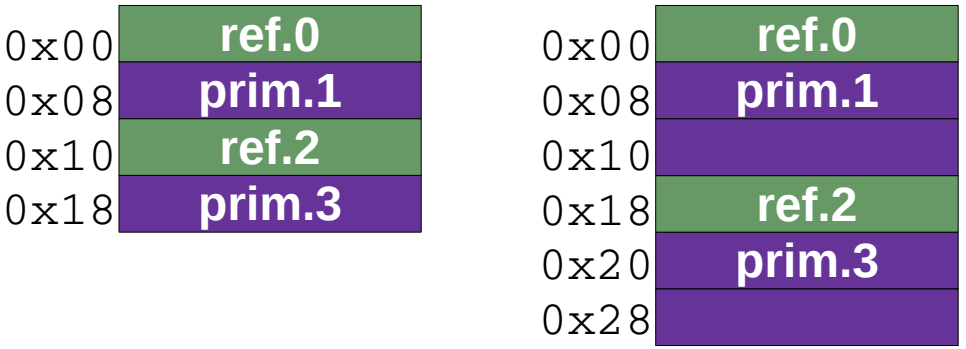
The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion

in Maxine VM →

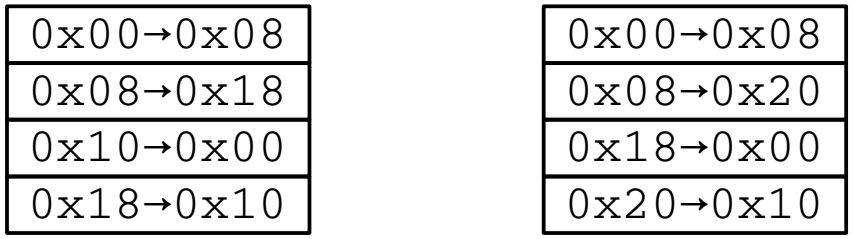
Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$

Fields Reordering Map



m_o m_e

- reference
- primitive

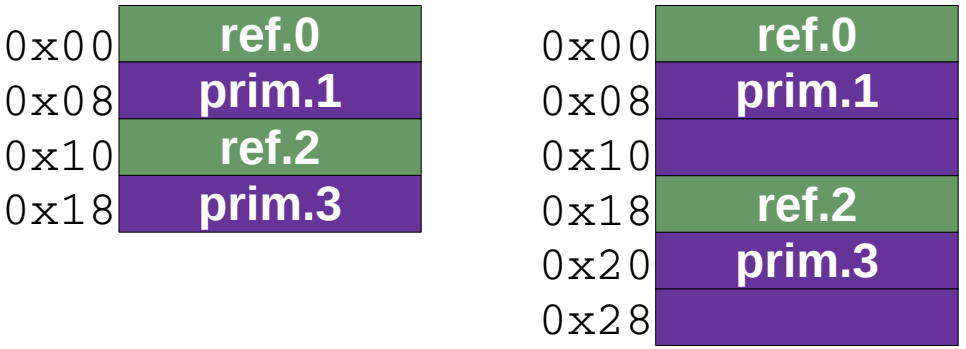
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim →

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

- reference
- primitive

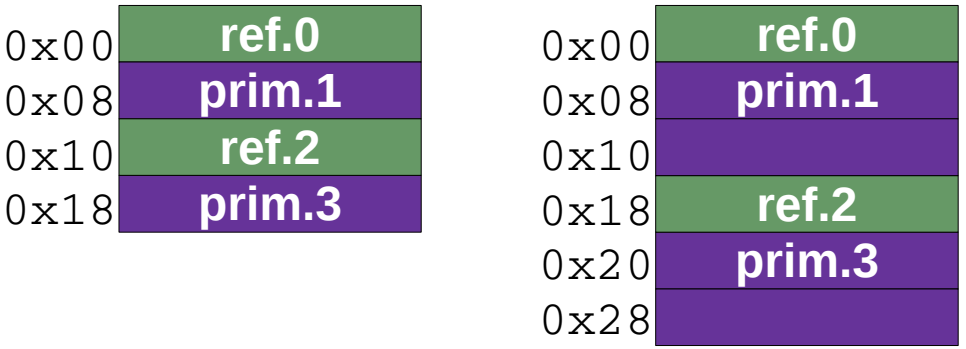
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim →

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

- reference
- primitive

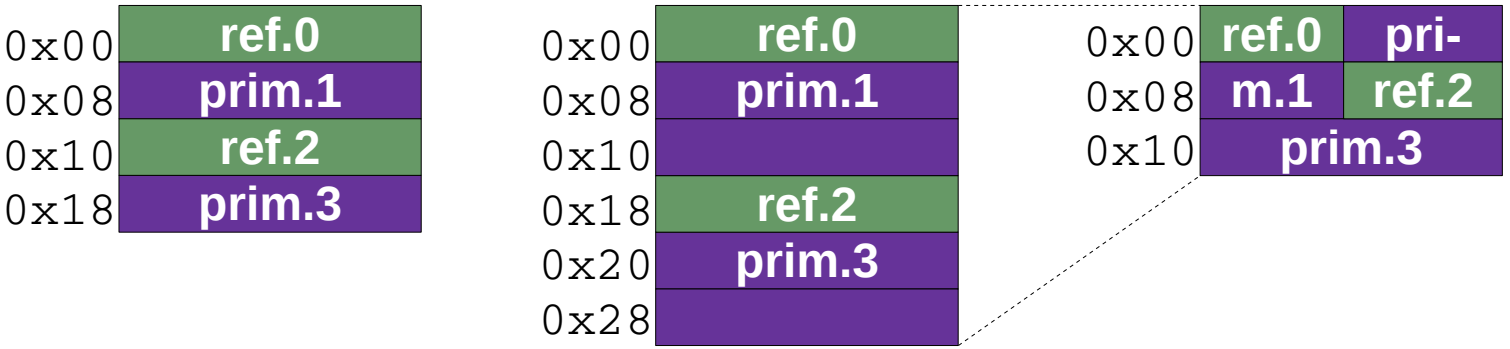
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim →

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

- reference
- primitive

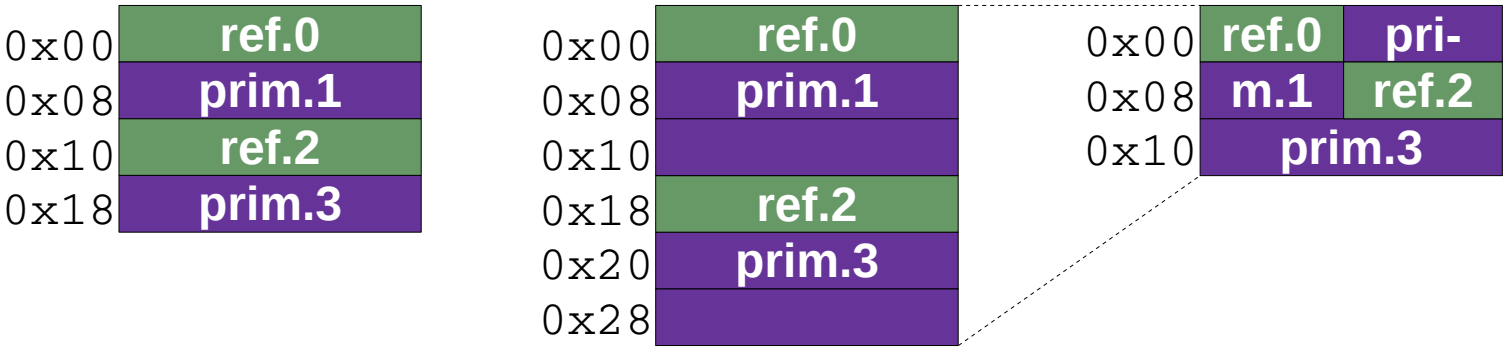
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim →

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

m_c
0x00→0x04
0x04→0x10
0x0C→0x00
0x18→0x08

- reference
- primitive

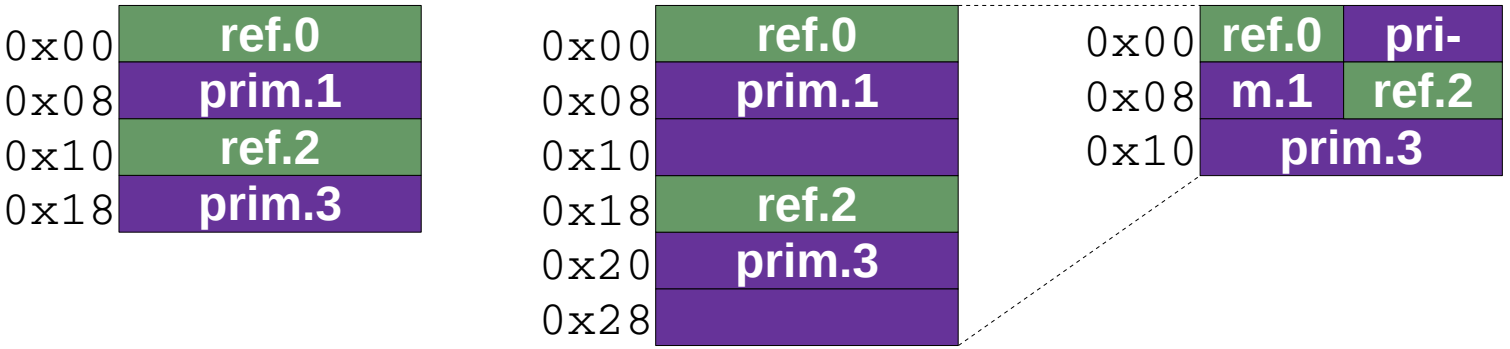
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim → $f_r(m_c)$ - reordering in ZSim

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

m_c
0x00→0x04
0x04→0x10
0x0C→0x00
0x18→0x08

- reference
- primitive

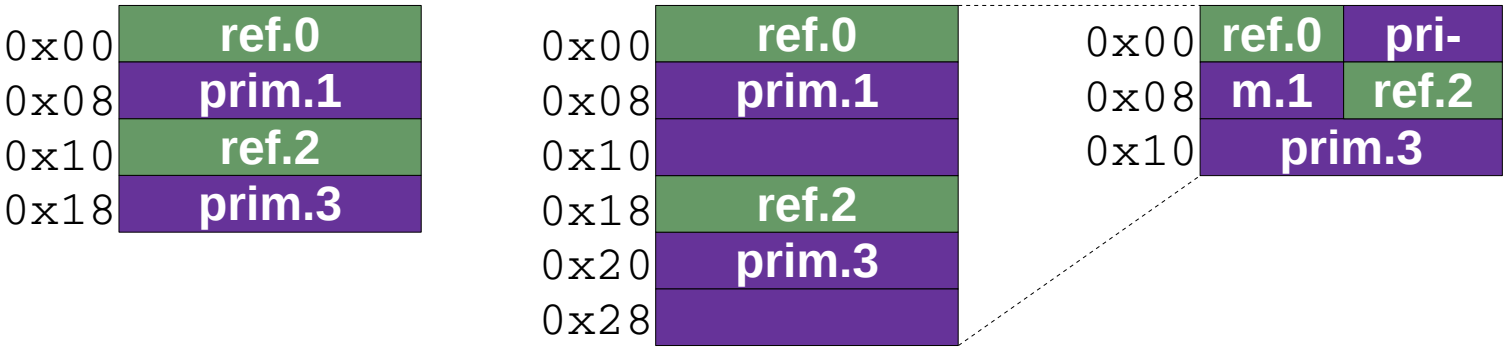
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim → $f_r(m_c)$ - reordering in ZSim

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$ $[b_c+m_c(o_c)]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

m_c
0x00→0x04
0x04→0x10
0x0C→0x00
0x18→0x08

- reference
- primitive

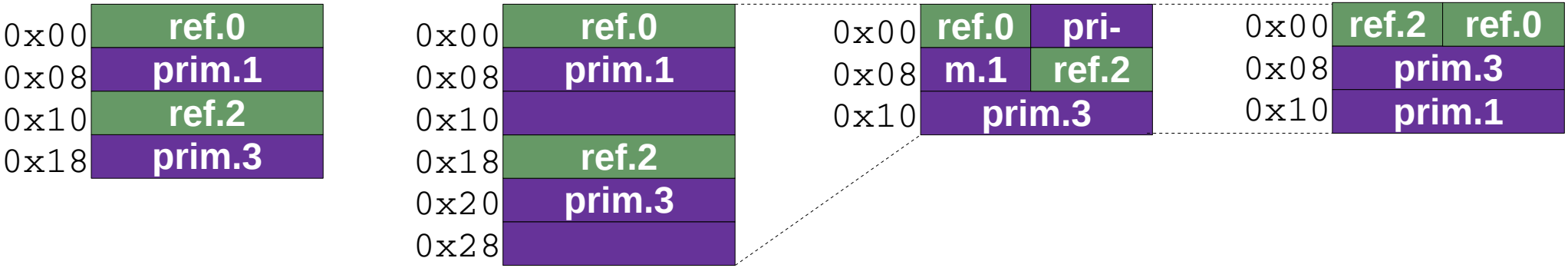
MaxSim: Overview and Features

The University of Manchester

Stages of Address Space Morphing

$f_e(1,2)$ - expansion in Maxine VM → $f_c(2)$ - contraction in ZSim → $f_r(m_c)$ - reordering in ZSim

Layout



Addressing

$[b_o+o_o]$ $[f_e(b_o)+f_e(o_o)]$ $[b_e/2+o_e/2]$ $[b_c+m_c(o_c)]$

Fields Reordering Map

m_o
0x00→0x08
0x08→0x18
0x10→0x00
0x18→0x10

m_e
0x00→0x08
0x08→0x20
0x18→0x00
0x20→0x10

m_c
0x00→0x04
0x04→0x10
0x0C→0x00
0x18→0x08

- reference
- primitive

MaxSim: Overview and Features

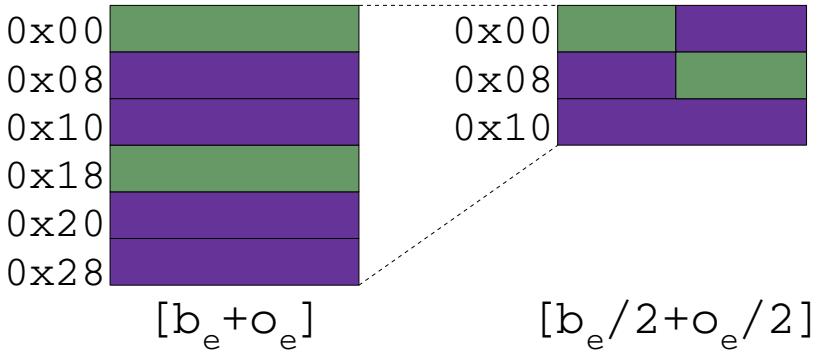
The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) {
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```

$f_c(2)$ – contraction in ZSim



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

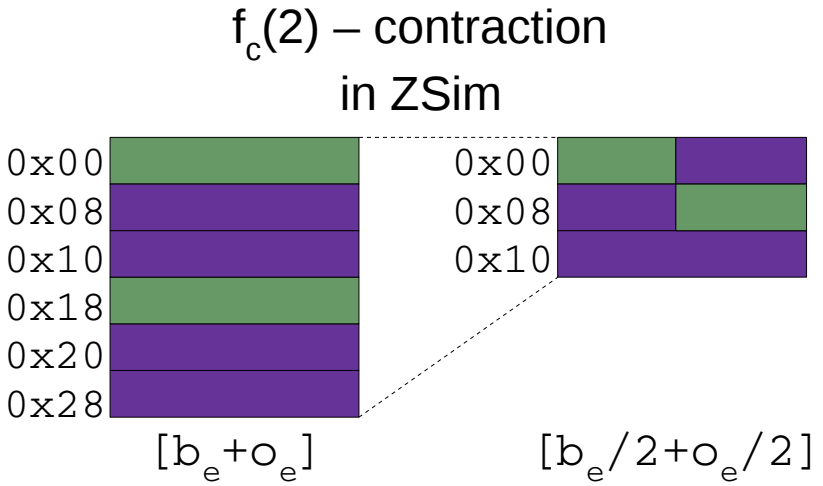
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) {
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

MaxSim: Overview and Features

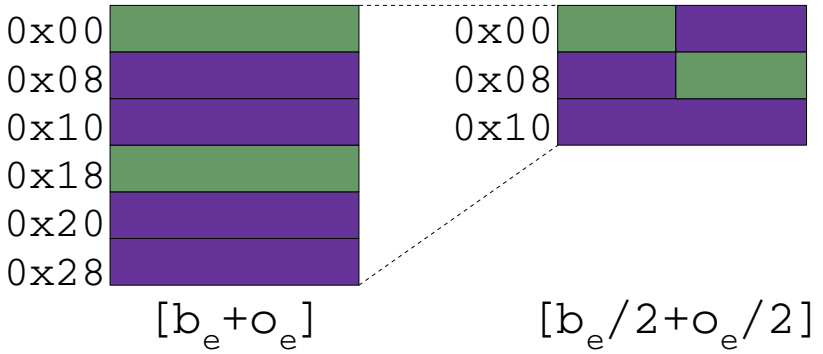
The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) {
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```

$f_c(2)$ – contraction in ZSim



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

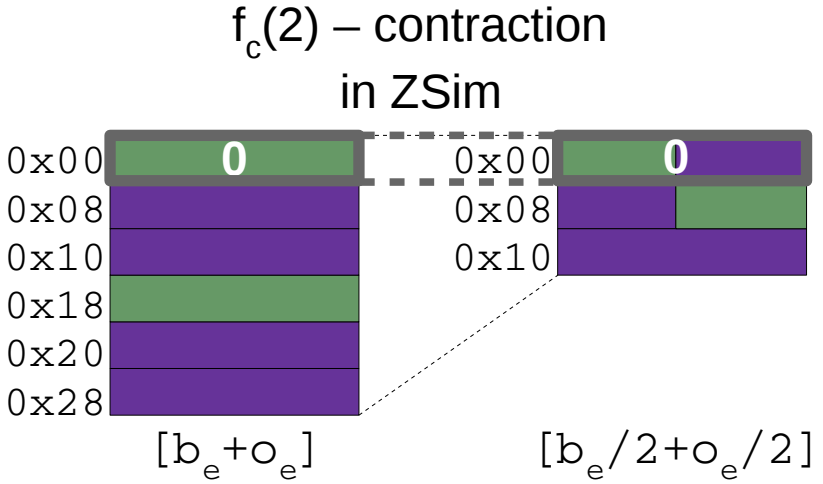
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 0
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

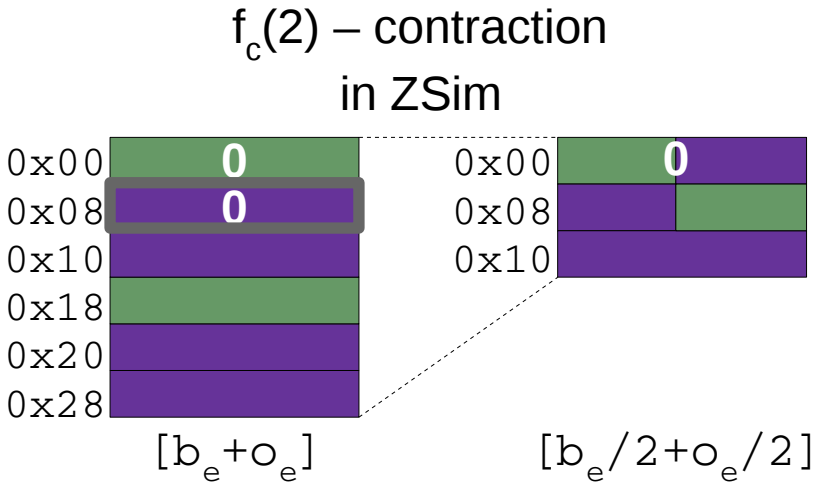
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 1
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

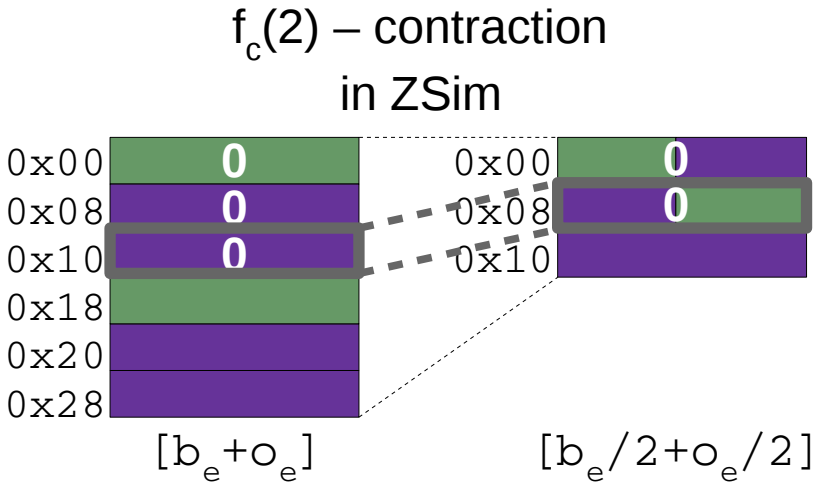
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 2
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

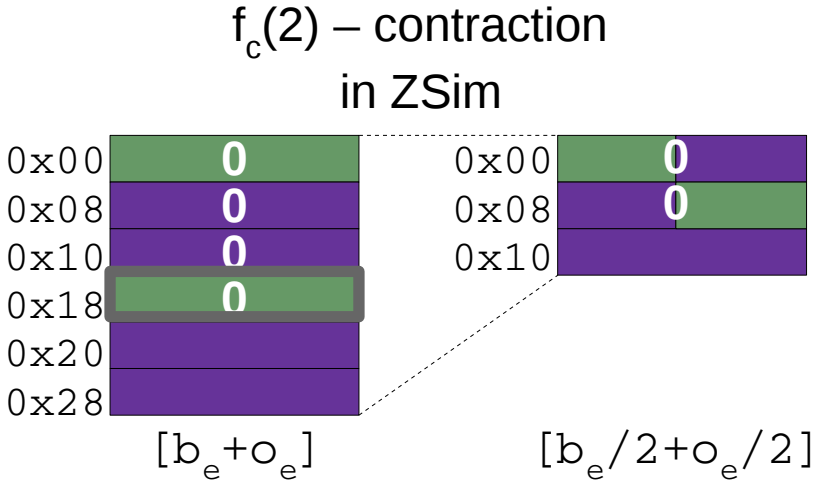
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 3
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

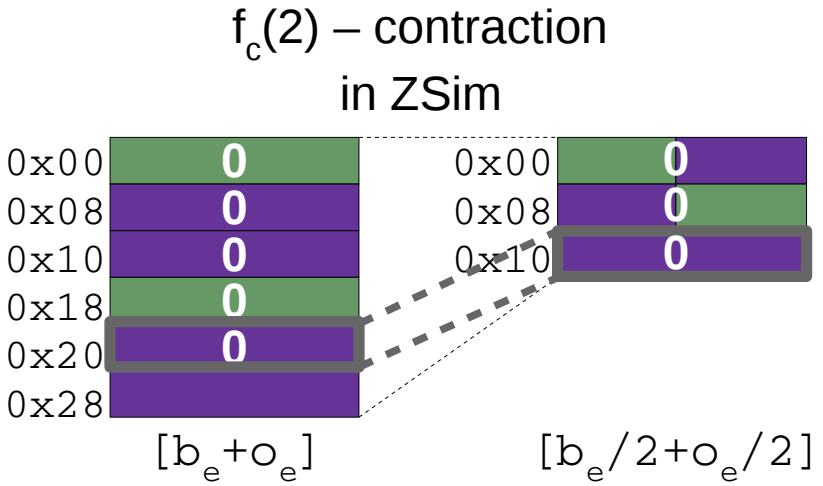
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 4
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

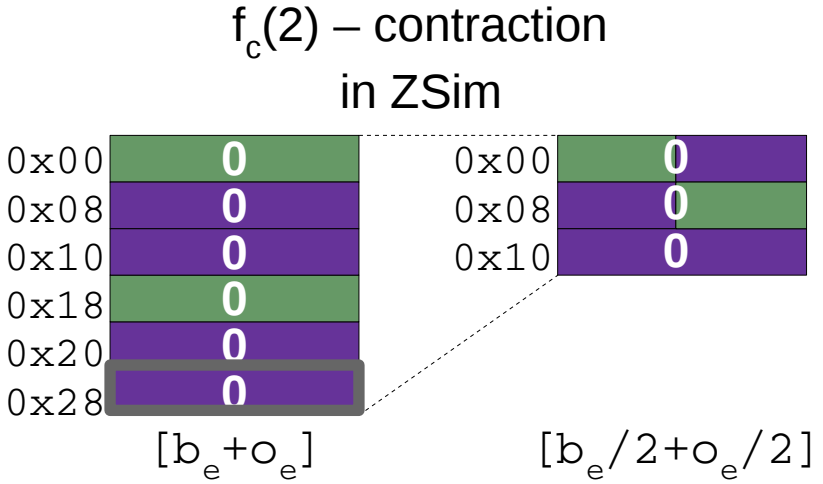
MaxSim: Overview and Features

The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) { // i = 5
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

MaxSim: Overview and Features

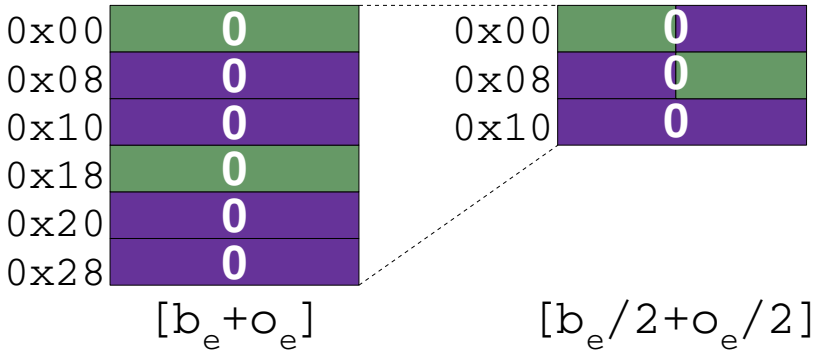
The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) {
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```

$f_c(2)$ – contraction in ZSim



- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

MaxSim: Overview and Features

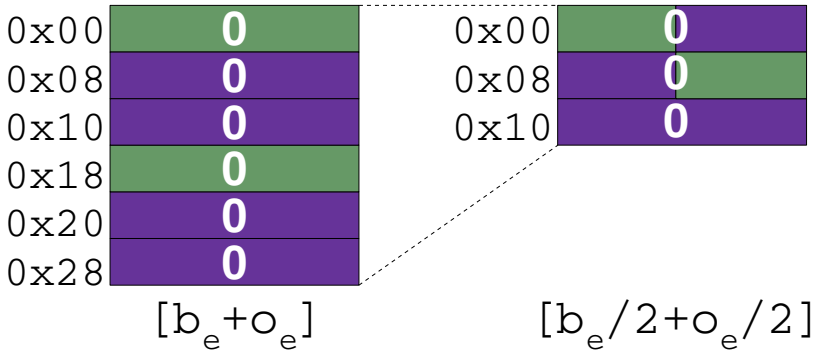
The University of Manchester

Address Space Morphing: Special Cases and Validation

- Simulation filtering of copying and initialization

```
// Loop used for initialization.
void setWords(Pointer p, int n) {
  ZSIM_MAGIC_NOP(BEGIN_LOOP_FILTERING);
  for (int i = 0; i < n; i++) {
    p.writeWord(i, 0);
  }
  ZSIM_MAGIC_NOP(END_LOOP_FILTERING);
}
```

$f_c(2)$ – contraction in ZSim



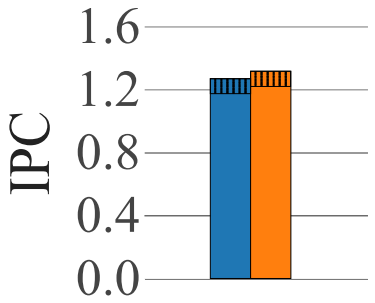
- Special cases for fast simulation
  - Array of primitives and code cache objects are handled differently
- Validation
  - References and primitives were expanded twice in Maxine VM and contracted twice in ZSim
  - Less than 1% difference in comparison with the original layout

MaxSim: Use Cases

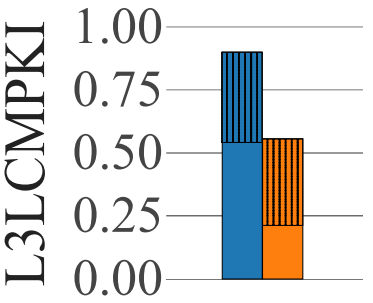
The University of Manchester

DaCapo Tomcat Characterization

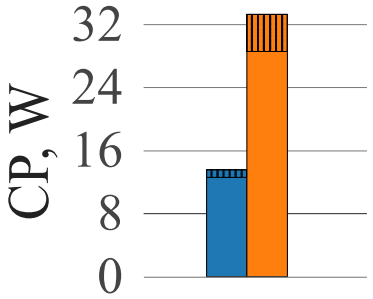
1 Core 2MB LLC: 4 Cores 8MB LLC: GC part:



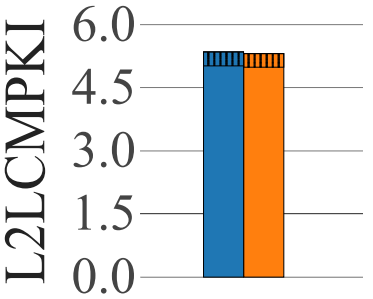
Instructions per Clock



L3 Load Cache Misses per Kilo Instruction



Consumed Power



L2 Load Cache Misses per Kilo Instruction

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

String.class bytecode

bci	Instr.	Line
20	*getfield*	980
23	*arraylength*	980
24	*istore_3*	980

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

String.class bytecode

bci	Instr.	Line
20	*getfield*	980
23	*arraylength*	980
24	*istore_3*	980

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

String.class bytecode

bci	Instr.	Line
20	*getfield*	980
23	*arraylength*	980
24	*istore_3*	980

String.java source code

```
974  public boolean equals(Object anObject) {
975    if (this == anObject) {
976      return true;
977    }
978    if (anObject instanceof String) {
979      String anotherString = (String) anObject;
980      int n = value.length;
981      if (n != anotherString.value.length)
982        return false;
983      ...
```

MaxSim: Use Cases

The University of Manchester

Analysis of L2 Cache Misses via Profiling

MaxSim output of class profiling information

```
char[]([C)(i:43 mf:57163720 (s:56(200337) ... r2m:722499 w2m:158200 r3m:108784 w3m:7723):
    ...
    (o:16 f:.35 r:18602074 w:759093 r2m:596449 w2m:62251 r3m:80211 w3m:161)
    ...
```

Maxsim output of cache miss site profiling information

```
...
[java.lang.String.equals(Object)+108(k:I bci:23)](m:539629 i:43 ol:16 oh:16)
...
```

String.class bytecode

bci	Instr.	Line
20	*getfield*	980
23	*arraylength*	980
24	*istore_3*	980

String.java source code

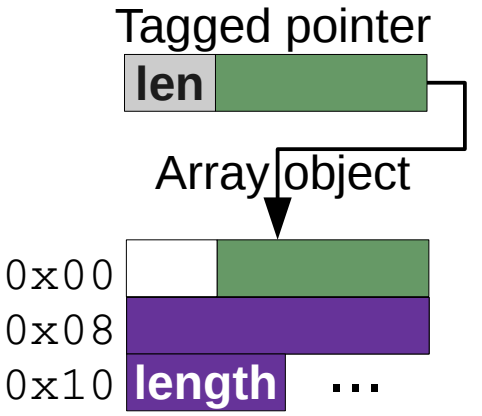
```
974  public boolean equals(Object anObject) {
975    if (this == anObject) {
976      return true;
977    }
978    if (anObject instanceof String) {
979      String anotherString = (String) anObject;
980      int n = value.length;
981      if (n != anotherString.value.length)
982        return false;
983      ...
```

MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator



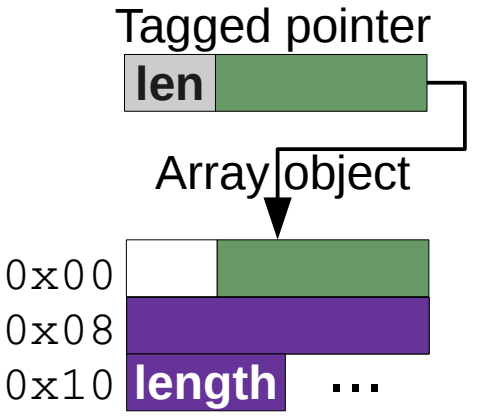
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

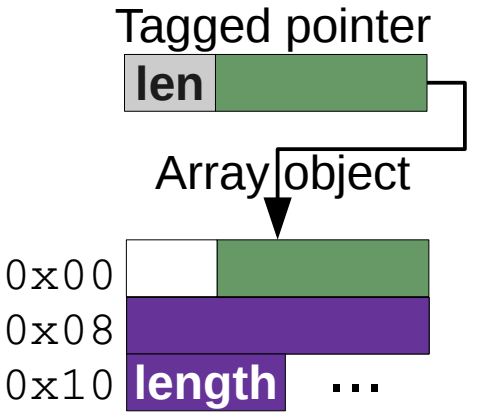
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

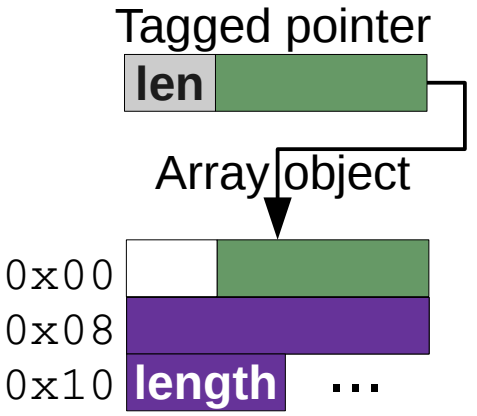
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

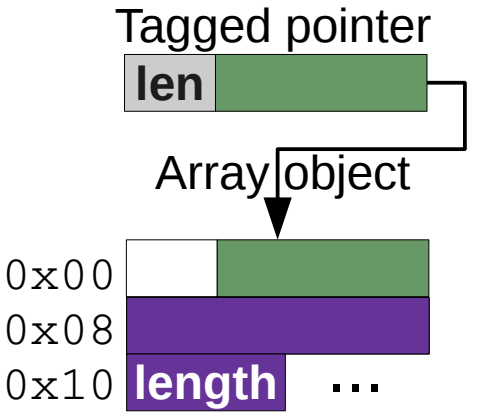
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

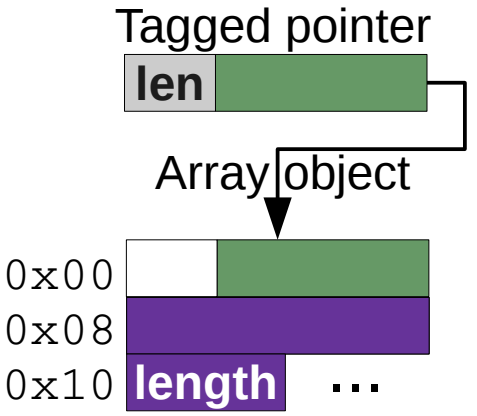
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

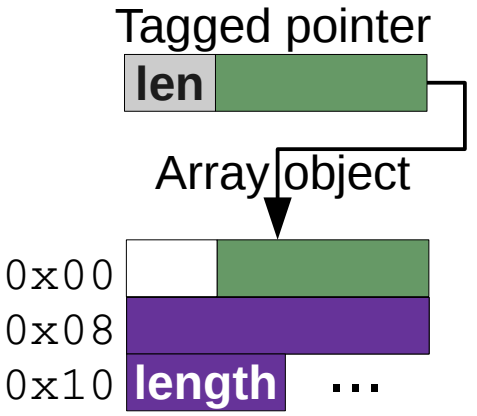
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

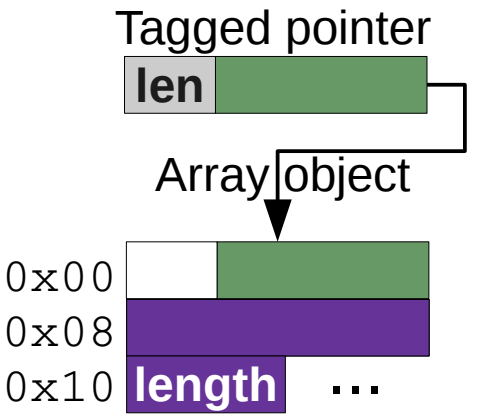
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

x86-64 assembler

```
// objectAddress in   %rdi
    movq    %rdi,     %rax
    shrq    $48,      %rax
    cmpq    $65535,   %rax
    jne     .L1
    movq    16(%rdi), %rax
.L1:
// array length in    %rax
```

- Dynamic execution height of 4.5 instructions of 19 bytes
- Originally 1 instruction of 4 bytes

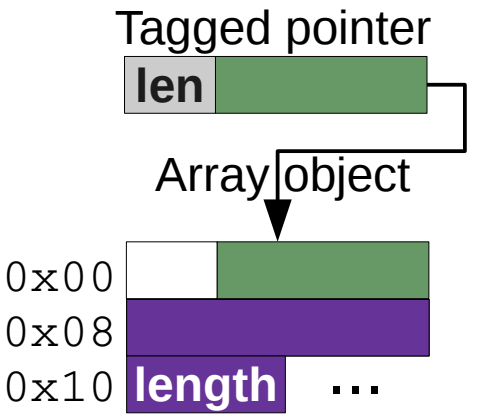
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

x86-64 assembler

```
// objectAddress in   %rdi
    movq    %rdi,     %rax
    shrq    $48,      %rax
    cmpq    $65535,   %rax
    jne     .L1
    movq    16(%rdi), %rax
.L1:
// array length in    %rax
```

- Dynamic execution height of 4.5 instructions of 19 bytes
- Originally 1 instruction of 4 bytes

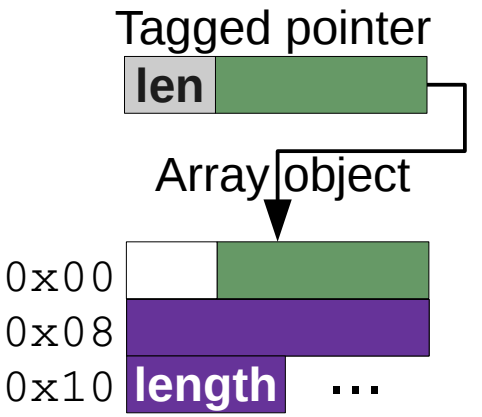
MaxSim: Use Cases

The University of Manchester

Storing Array Length in a Pointer Tag

- Having 16-bit-tagged pointers it is possible to store a range of array lengths [0;0xFFFE], when 0XFFFF is Not an Array Length (NaAL) indicator

- Array length retrieval in software



Source code

```
inline int retrieveArrayLength(Pointer_t objectPointer)
{
    TAG_t tag = extractTAG(objectPointer);
    if (tag != NaAL) {
        return (int) tag;
    }
    return * ((int *) (objectPointer + 0x10));
}
```

x86-64 assembler

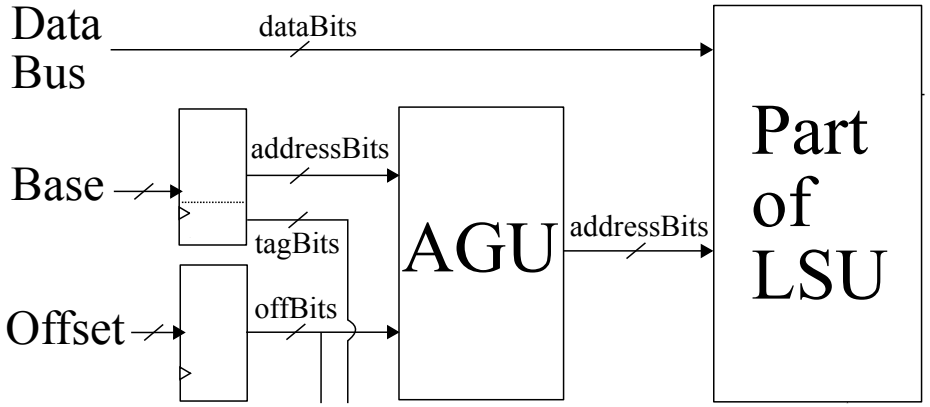
```
// objectAddress in    %rdi
     movq    %rdi,      %rax
     shrq    $48,       %rax
     cmpq    $65535,    %rax
     jne     .L1
     movq    16(%rdi),  %rax
.L1:
// array length in     %rax
```

- Dynamic execution height of 4.5 instructions of 19 bytes
- Originally 1 instruction of 4 bytes

MaxSim: Use Cases

The University of Manchester

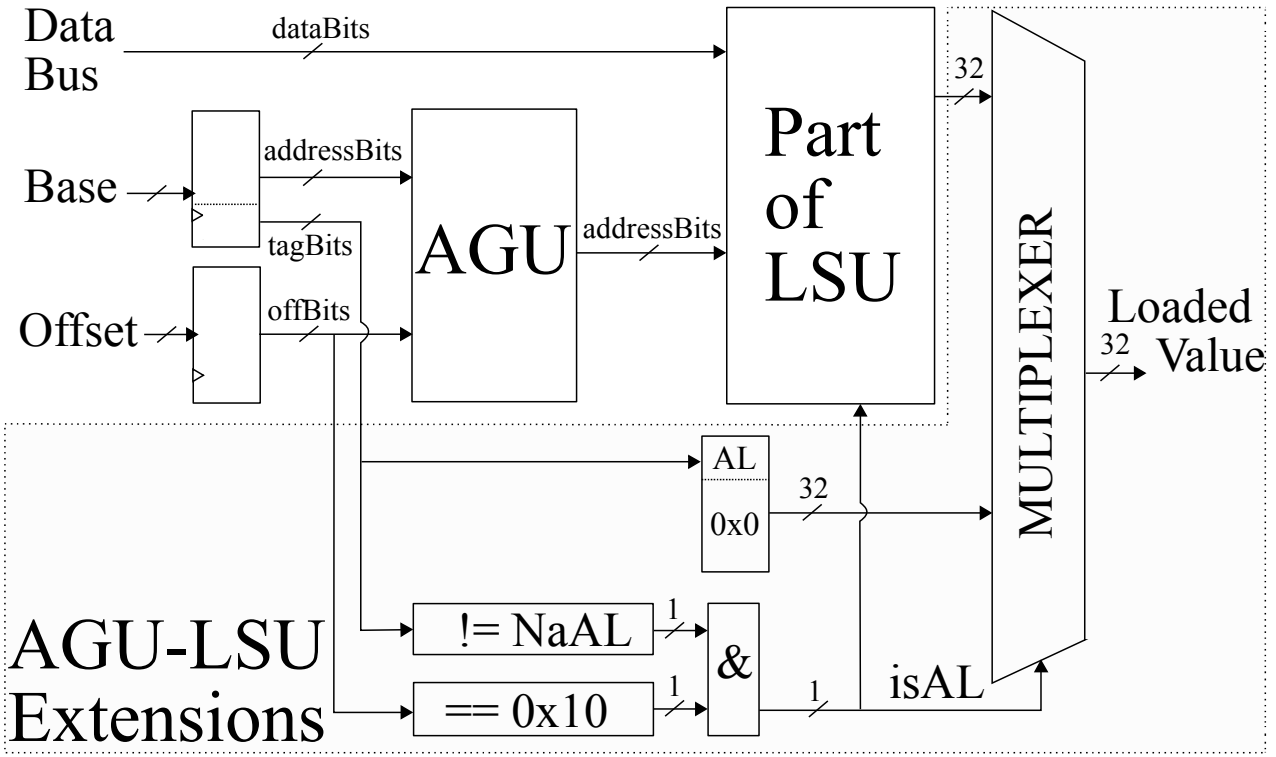
HW-Assisted Array Length Retrieval from Tagged Pointers



MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

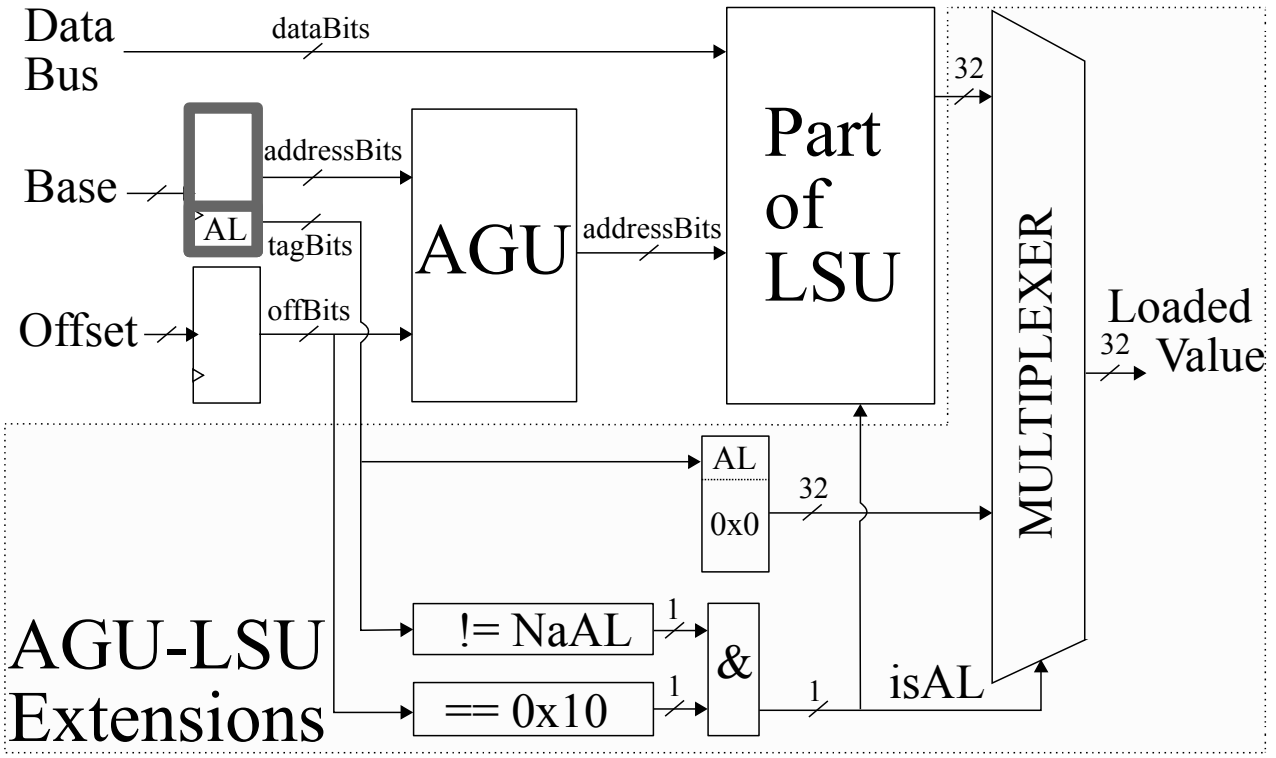
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

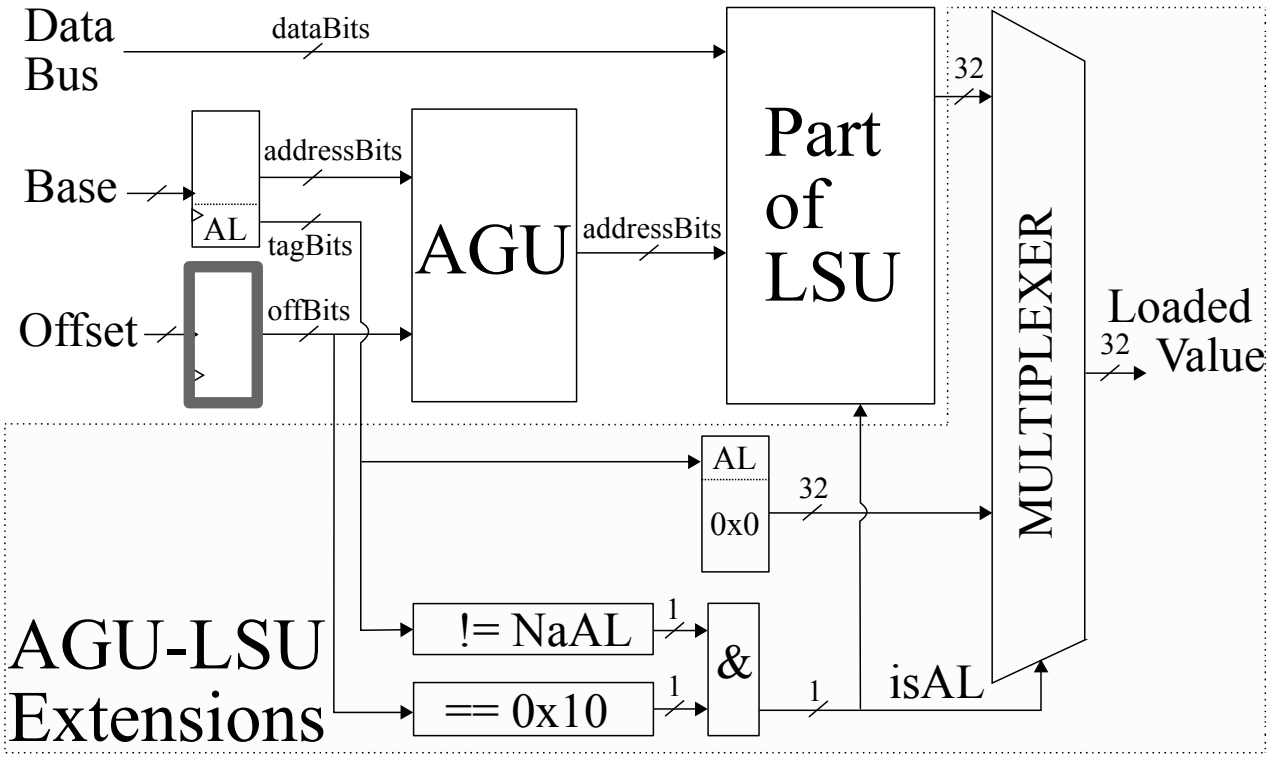
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

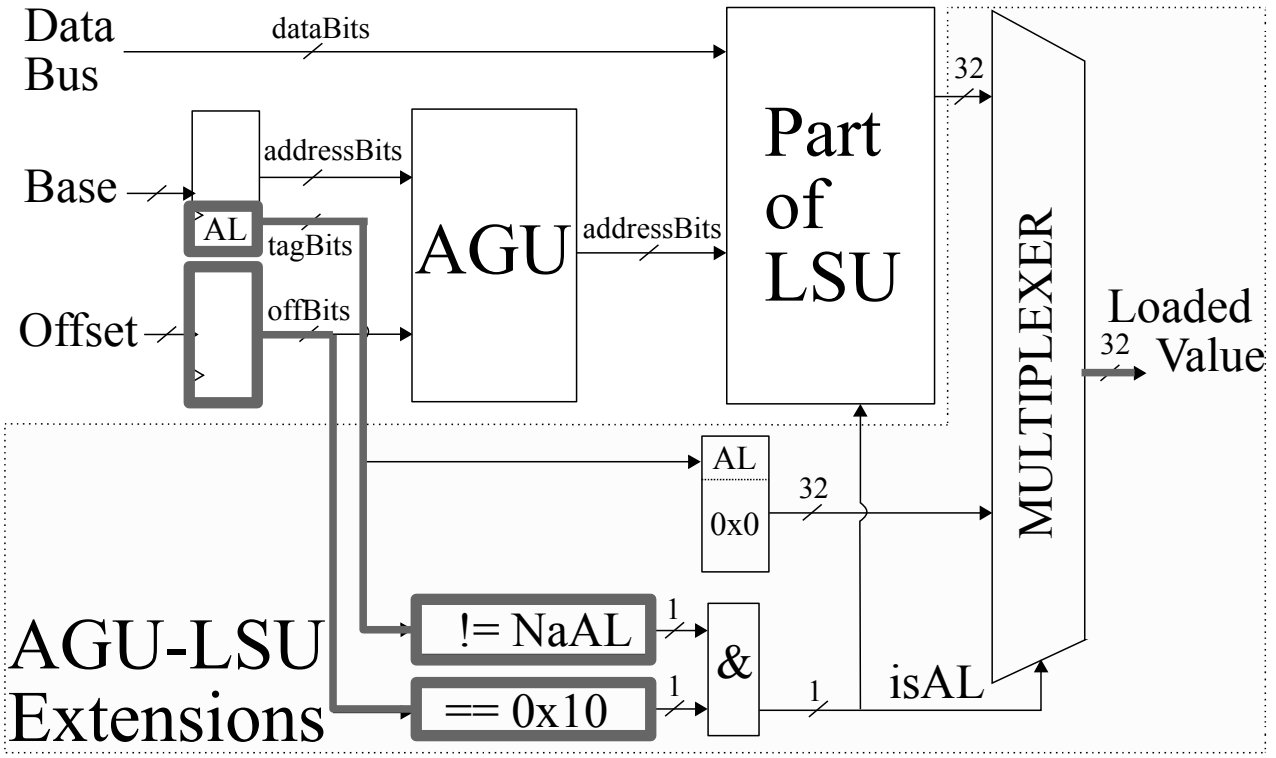
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

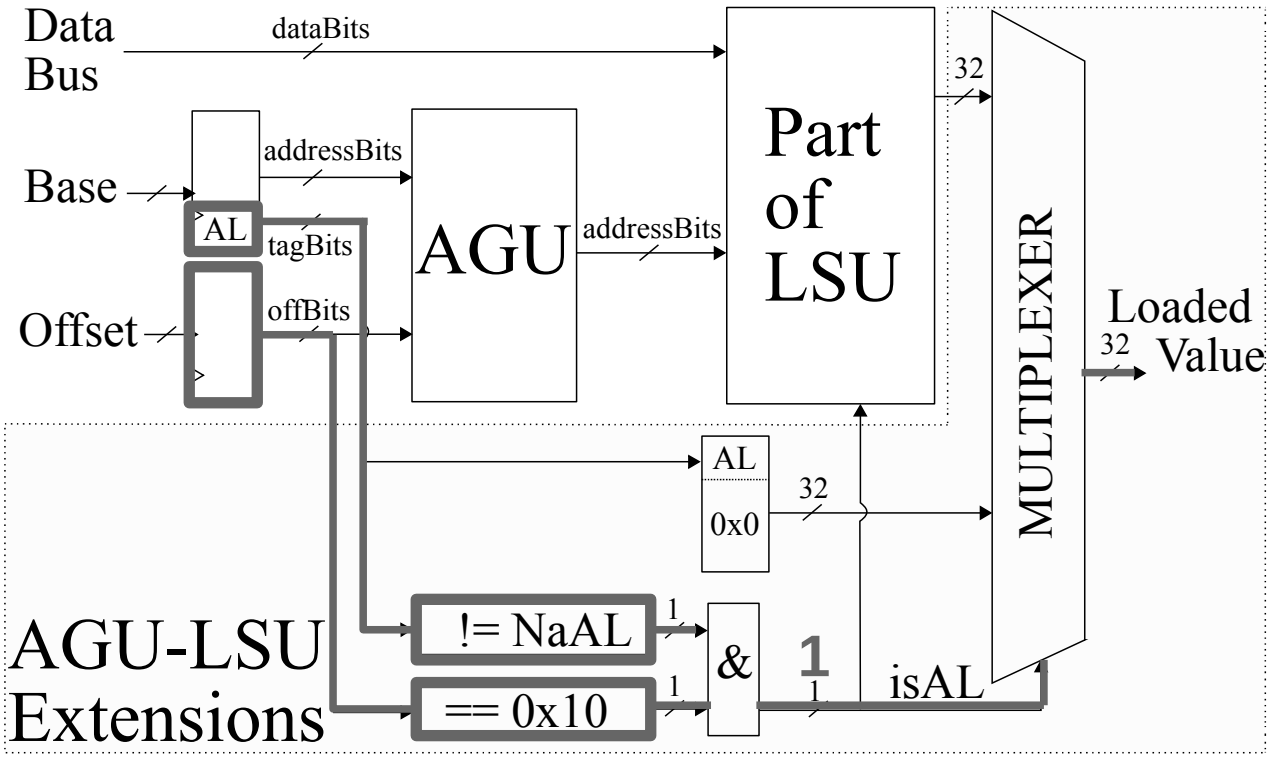
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

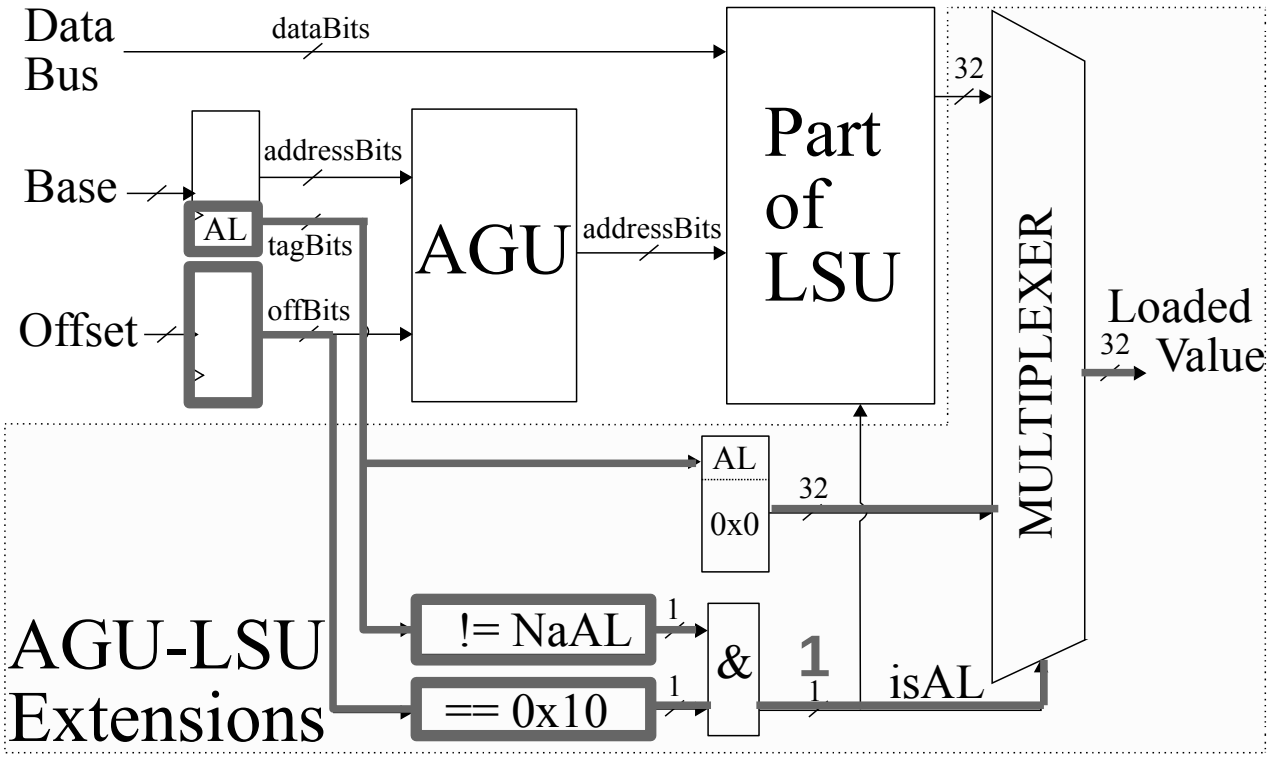
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

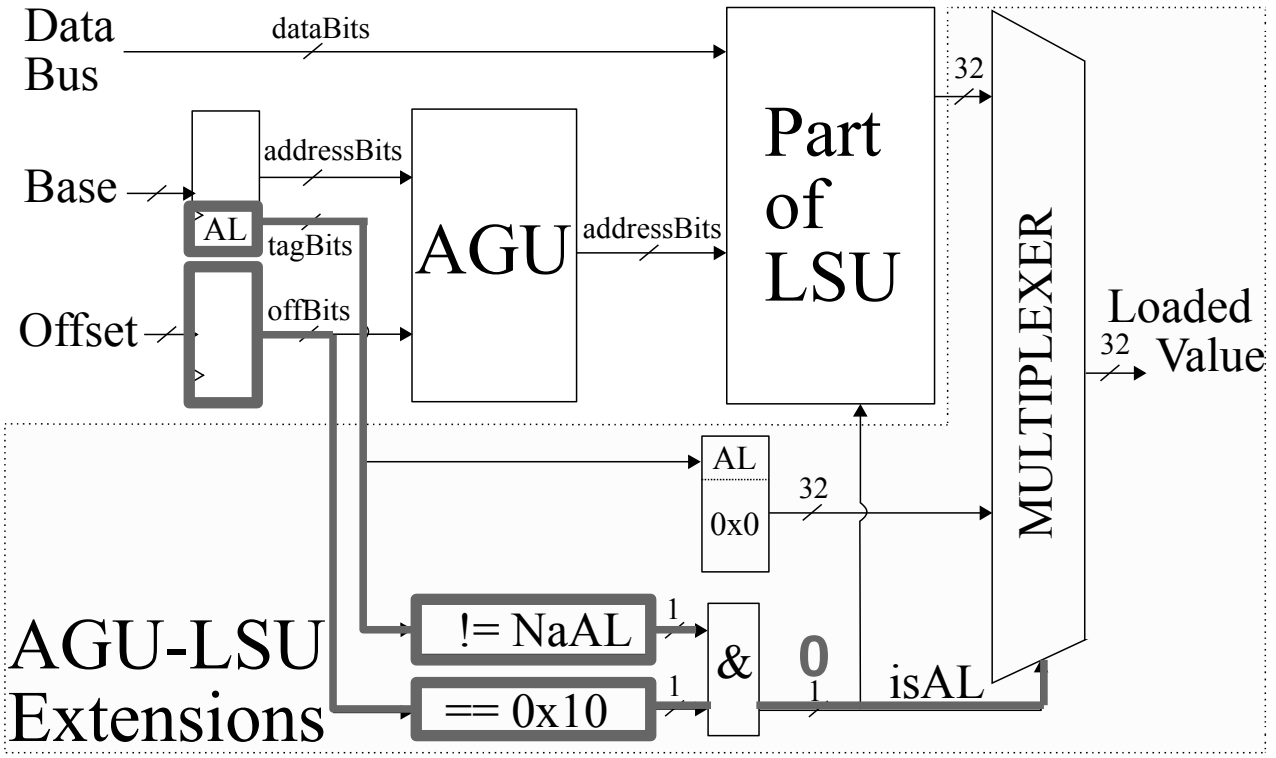
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

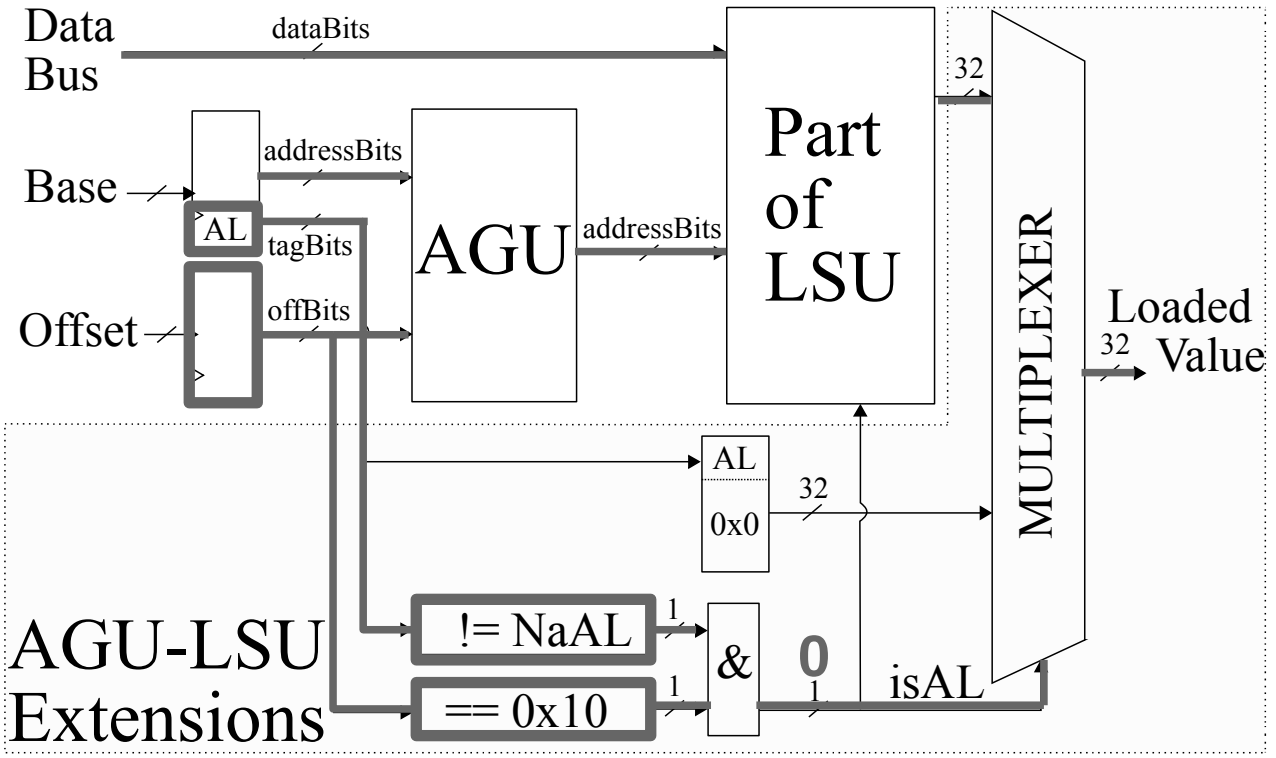
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

MaxSim: Use Cases

The University of Manchester

HW-Assisted Array Length Retrieval from Tagged Pointers



- Array length retrieval in one instruction

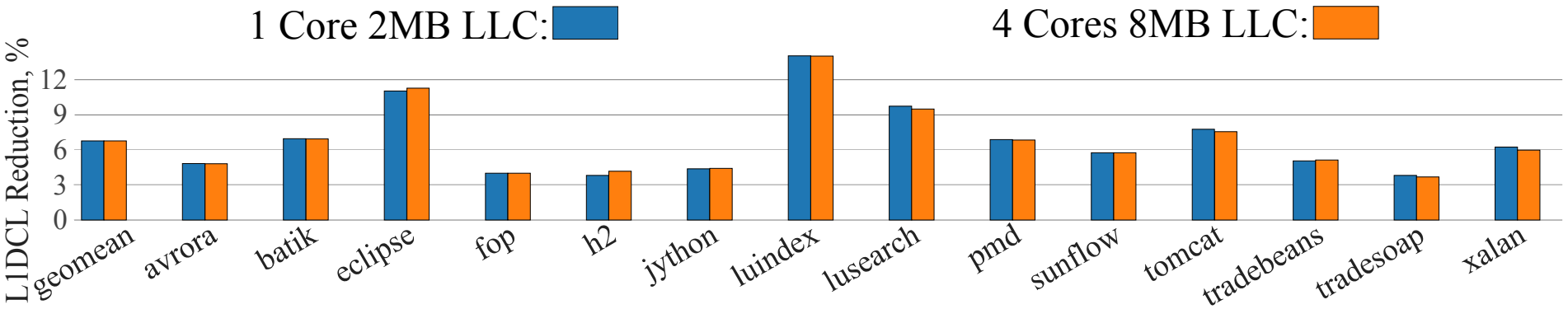
```
inline int retrieveArrayLength(Address_t objectAddress) {
    return * (( CIP_t *) (objectAddress + 0x10));
}
```

```
// objectAddress in %rdi
movq    16(%rdi), %rax
// array length in %rax
```

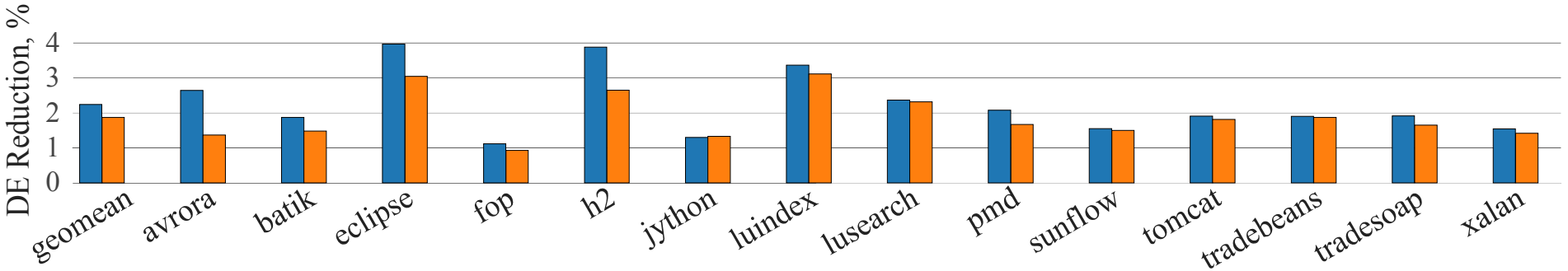
MaxSim: Use Cases

The University of Manchester

Evaluation of HW-Assisted Array Length Retrieval



L1 Data Cache Loads Reduction (~6% in geomean) on DaCapo Benchmarks

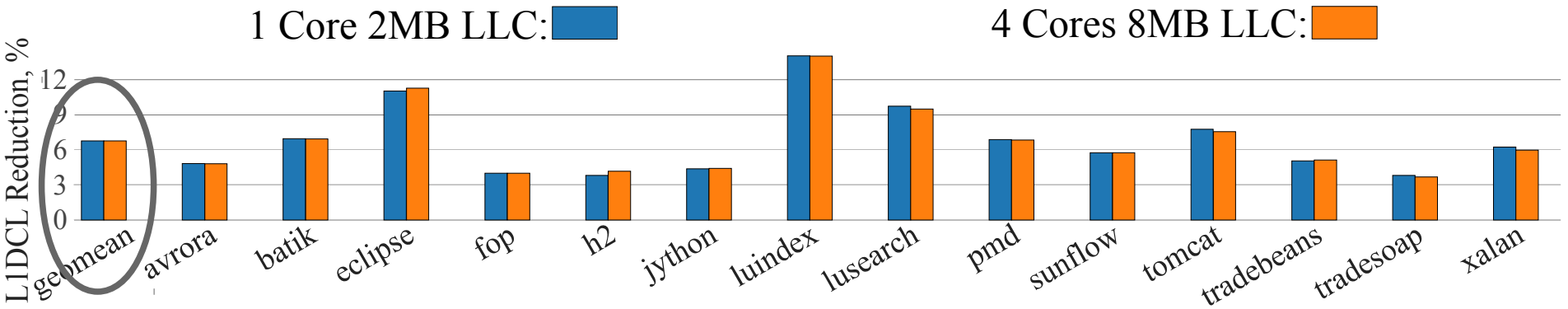


Dynamic Energy Reduction (~2% in geomean) on DaCapo Benchmarks

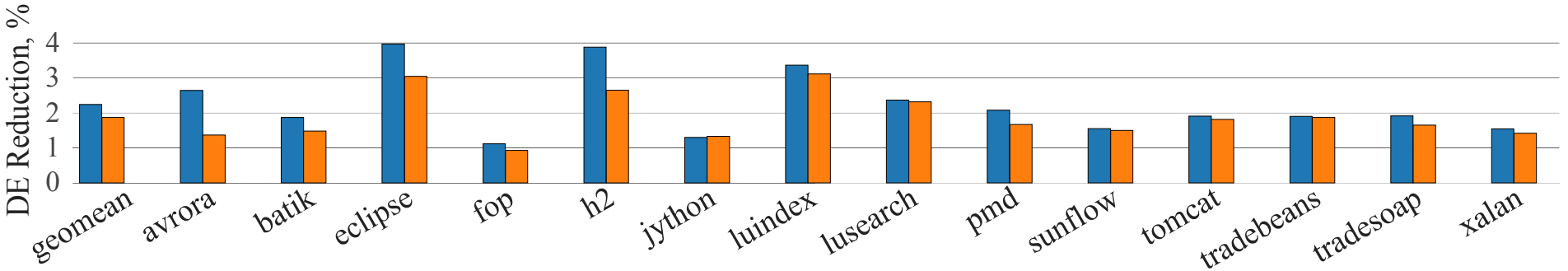
MaxSim: Use Cases

The University of Manchester

Evaluation of HW-Assisted Array Length Retrieval



L1 Data Cache Loads Reduction (~6% in geomean) on DaCapo Benchmarks

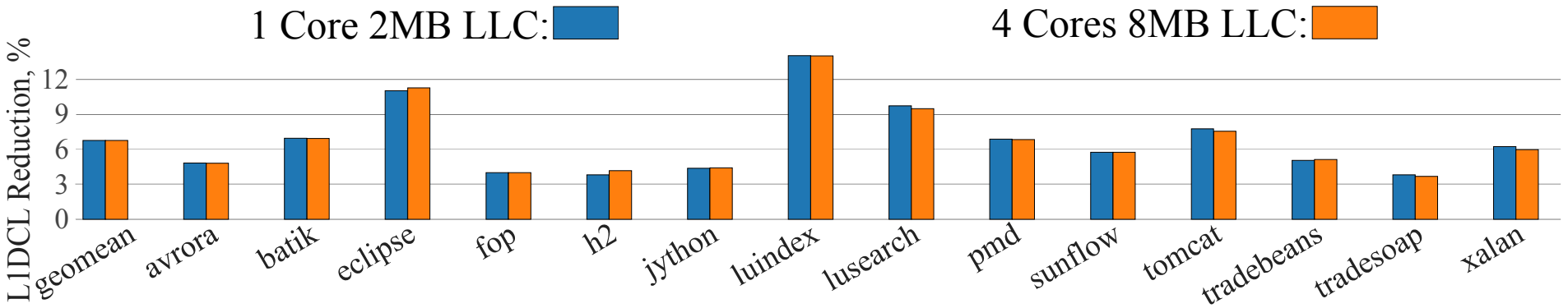


Dynamic Energy Reduction (~2% in geomean) on DaCapo Benchmarks

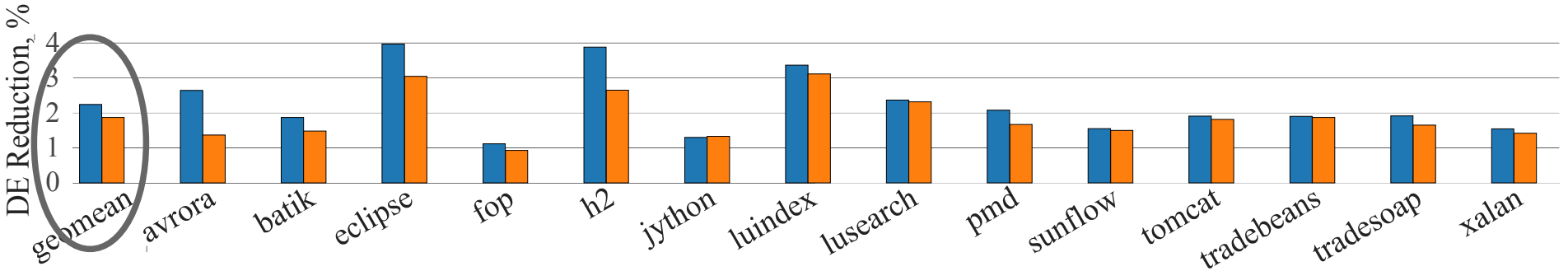
MaxSim: Use Cases

The University of Manchester

Evaluation of HW-Assisted Array Length Retrieval



L1 Data Cache Loads Reduction (~6% in geomean) on DaCapo Benchmarks



Dynamic Energy Reduction (~2% in geomean) on DaCapo Benchmarks

Conclusion

The University of Manchester

- Novel simulation platform for managed applications
  - Based of the state-of-the art VM and simulator
  - Awareness of the VM in the simulator
  - Simulation of 16-bit tagged pointers on x86-64
  - Low-overhead memory access profiling
  - Address-space morphing technique
- Use cases
  - Workload characterization and profiling
  - HW/SW co-design and exploration of architectural specialization for managed applications
  - Easy experimentation with object layout transformations
- Open-source platform is available at:

  https://github.com/beehive-lab/MaxSim