

ORACLE®

2010-0237



ORACLE®

**The Maxine Inspector:
A Specialized Tool for VM Development**

JVM Language Summit, July 28, 2010, Santa Clara, CA

Michael L. Van De Vanter
Researcher, Oracle Sun Labs

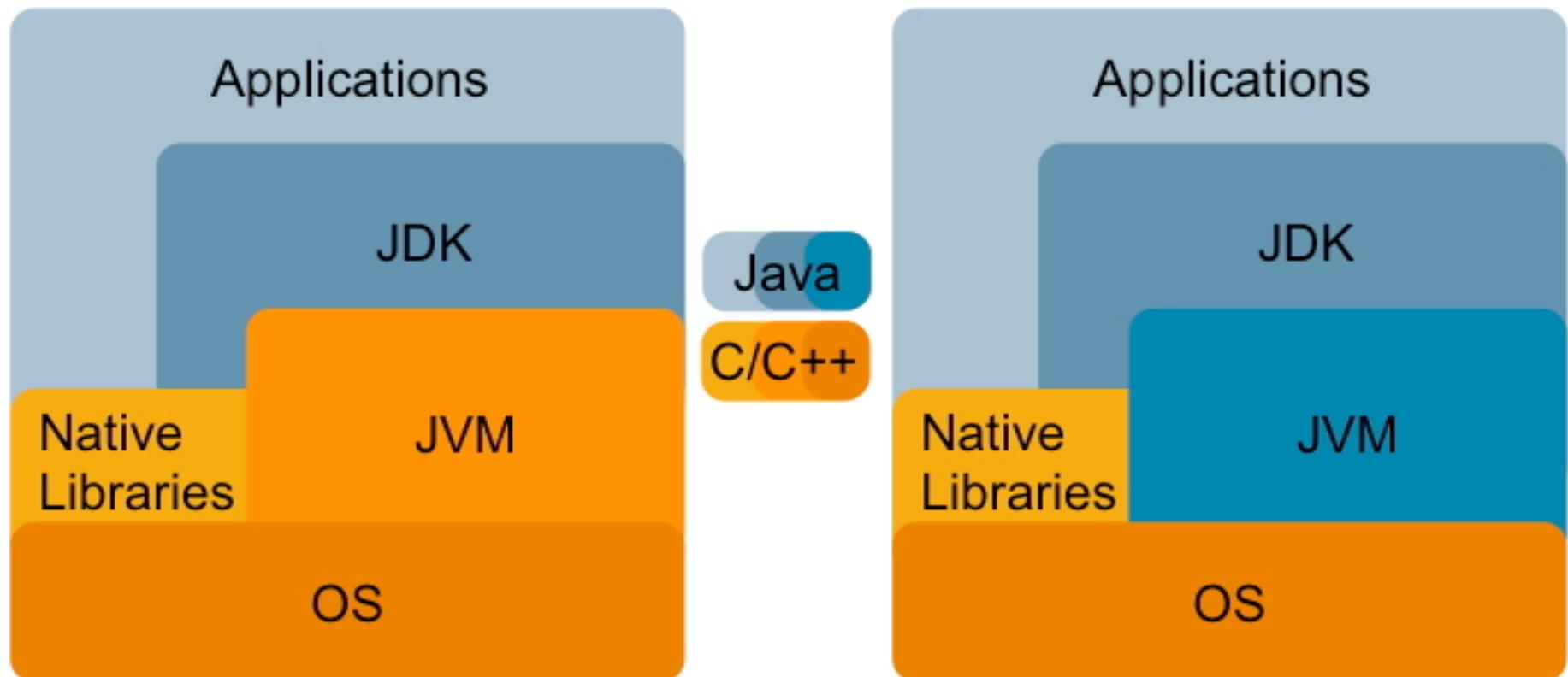
The Maxine Project at Oracle Sun Labs: highly productive and “approachable” JVM development

- Meta-circular VM, written in Java 1.5
- Binary compatible: works with standard JDK
- Load & build in standard IDEs
- Modular architecture:
 - Object layout
 - Object references
 - Garbage collector
 - Thread synchronization
 - VM startup sequence (e.g. Java)
 - Compilation strategy
 - Compiler(s)
- Maxine Inspector



Conventional vs. Metacircular

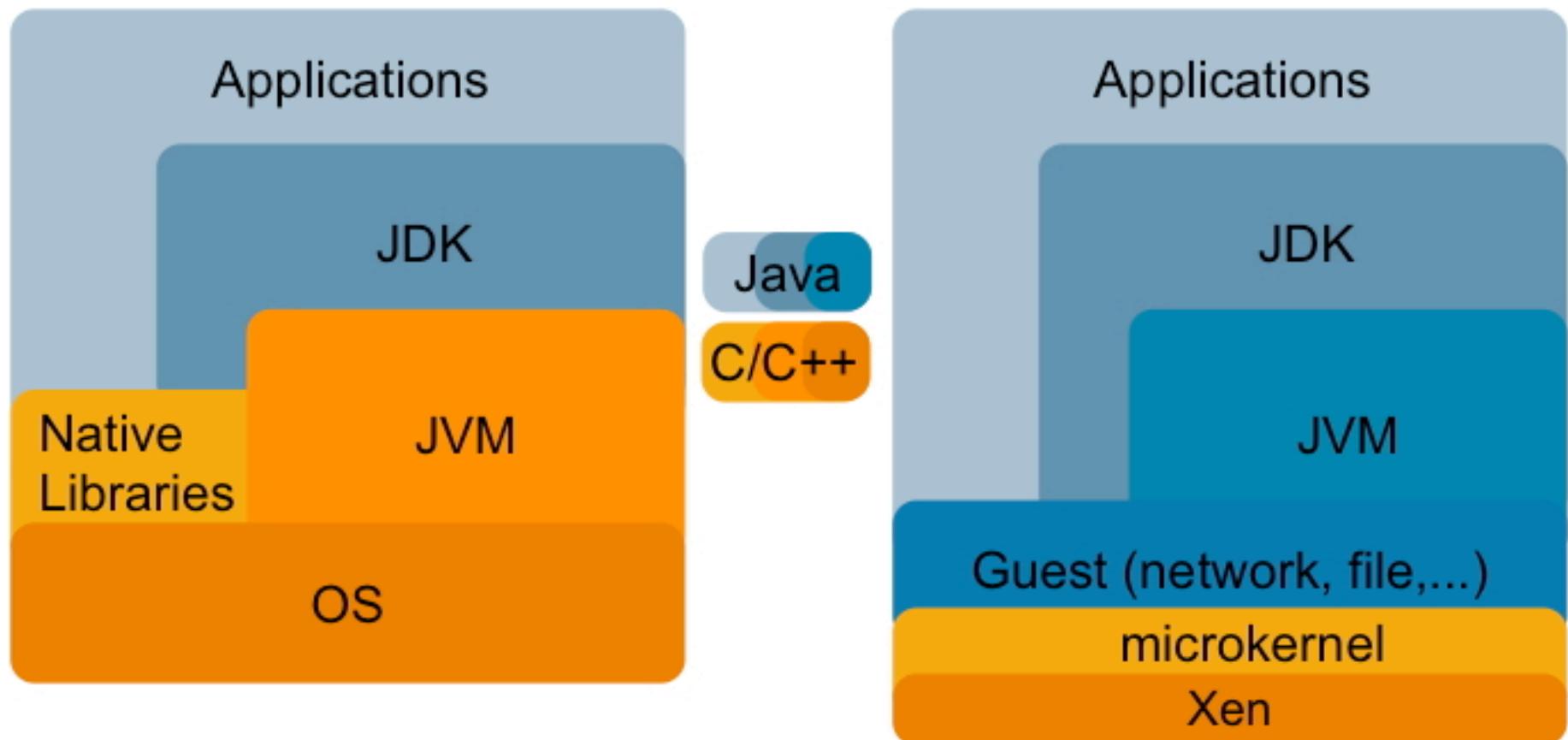
Maxine Standard Edition



ORACLE®

Conventional vs. Metacircular

Maxine Virtual Edition (formerly Guest VM)



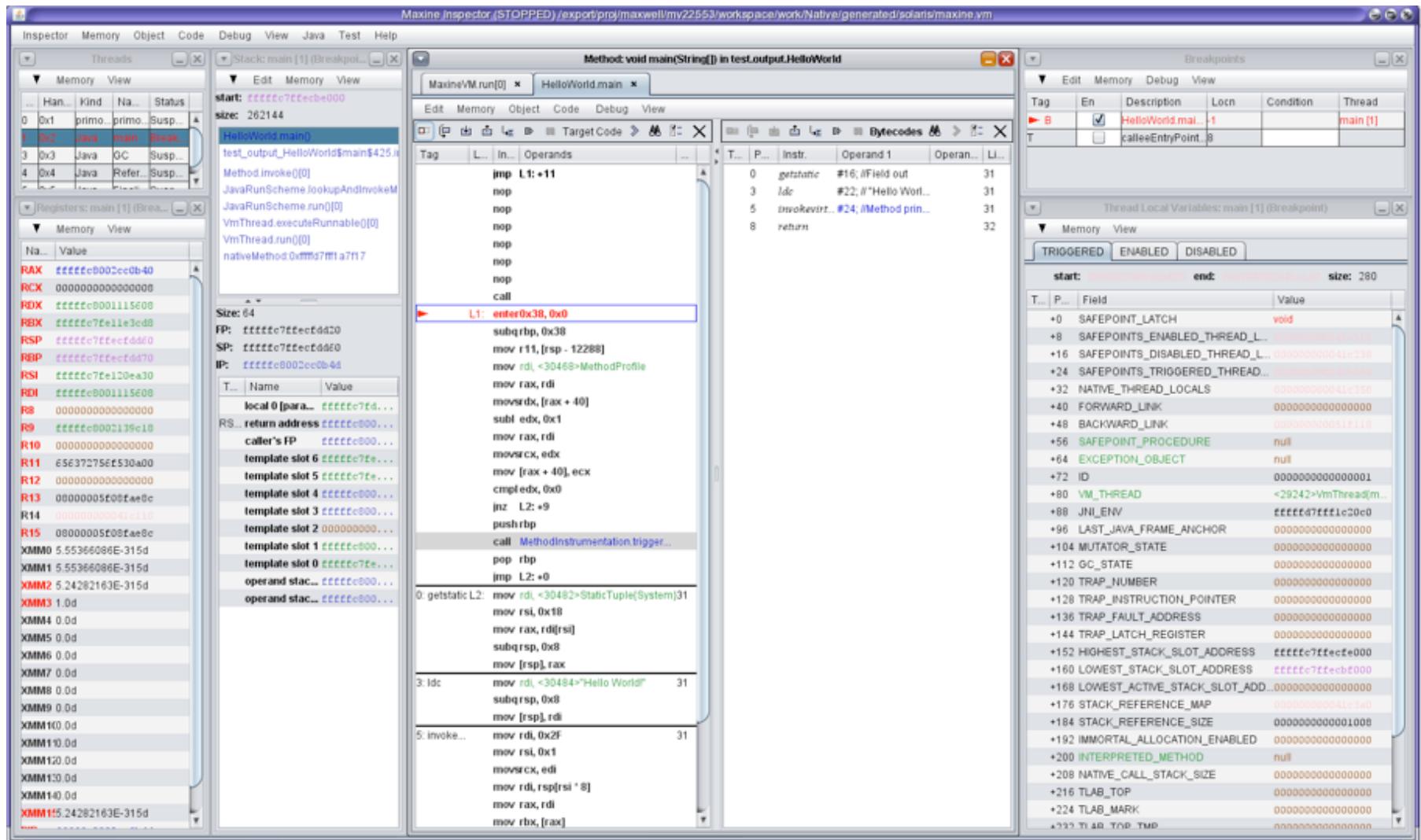
ORACLE®

The Maxine VM: Technical Overview

- Experiment in meta-circular design
 - Use Java bytecode instead of IR or assembly
 - E.g. JNI stubs, reflective invocation stubs
- Isolate configurable features with interfaces
- Build around an AOT optimizing compiler
- Compile-only execution model (JRockit, Jikes RVM)
- Annotations drive specialized compiler extensions
 - Permits low-level, unsafe memory access in Java
 - Effectively “systems programming in Java”
- Fast template-driven JIT

ORACLE®

The Maxine Inspector: Preview



ORACLE®

The Maxine Inspector: Debugger, Visualizer

“... a disassembler for the entire
VM execution state...”

- Randy Smith, Oracle Sun Labs

- Mission
 - Enable Maxine VM development & experimentation
 - Conventional development tools don't work
 - Set a new standard for VM “approachability”
 - Attract, inform, and accelerate developers
- Technical Overview
 - Specialized for Maxine: shares code, co-evolves
 - Out-of-process: controls process, reads memory
 - Runs on all VM platforms: Solaris, Linux, Mac OS
 - Requires very little support in VM: reads the bits, mostly

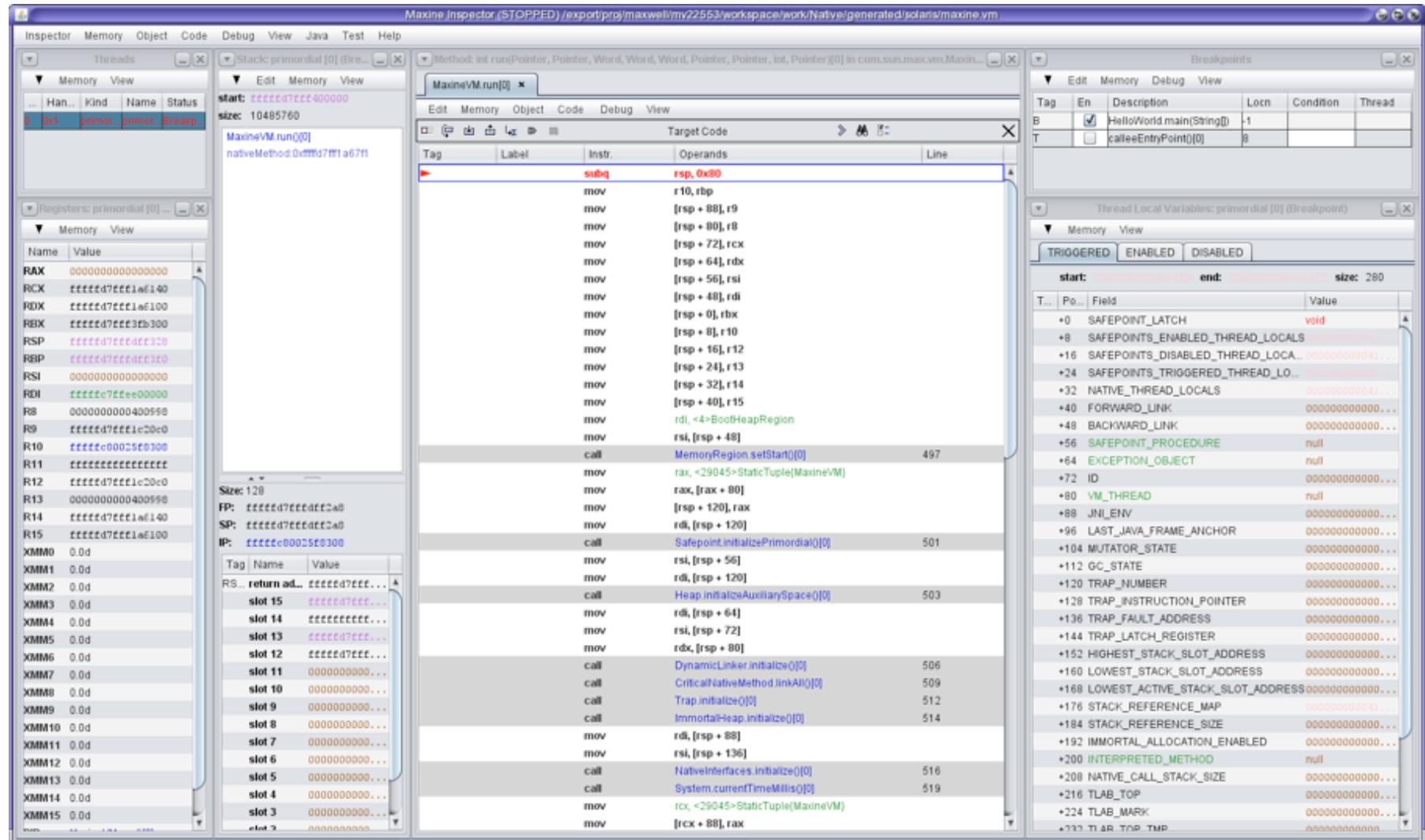


A Virtual Demo

- Screen snapshots follow
- More detailed video demo available (HD, no narration)
 - <http://www.youtube.com/watch?v=taJOxIUXJrc>

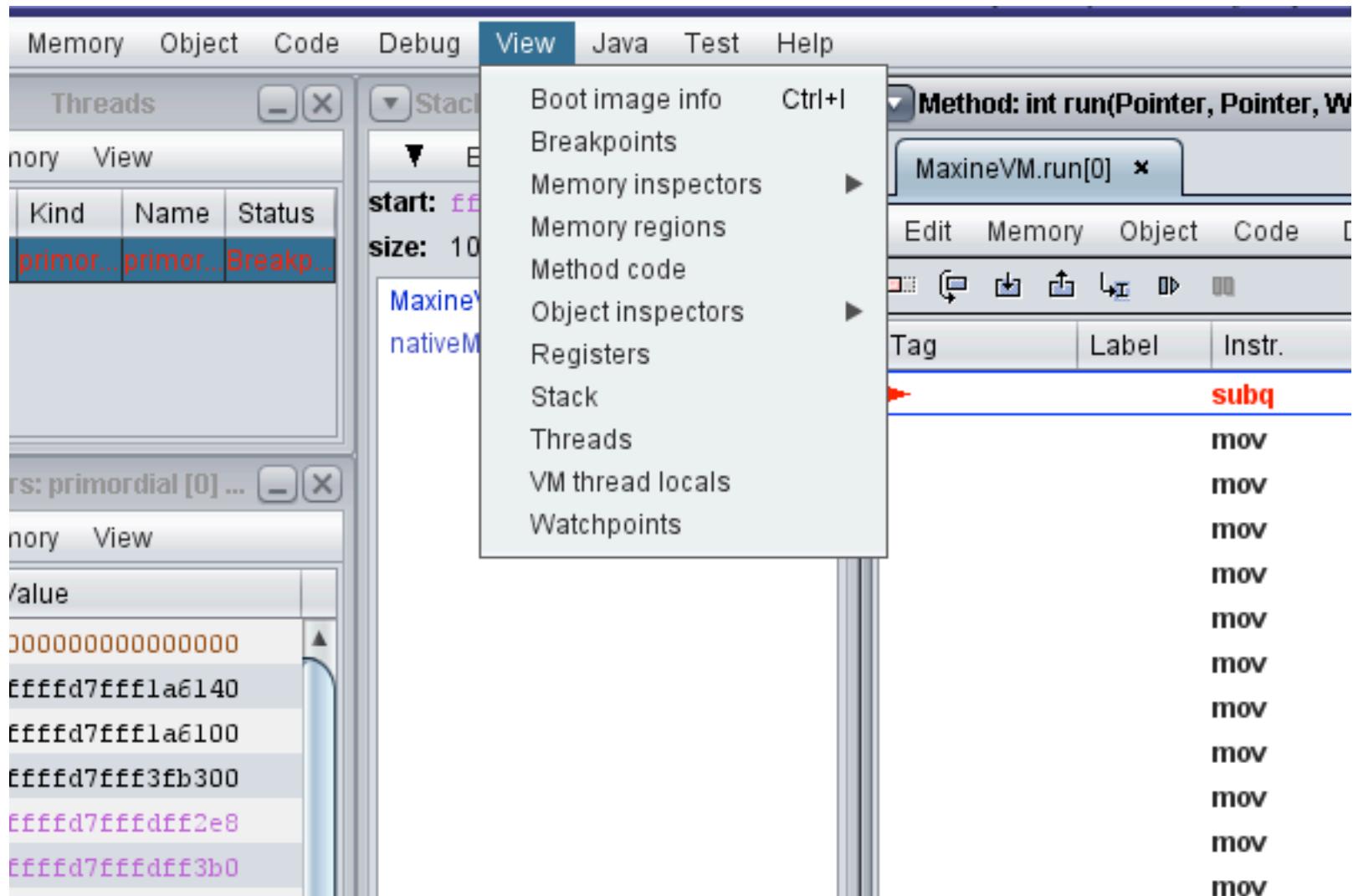


1: Session Begin



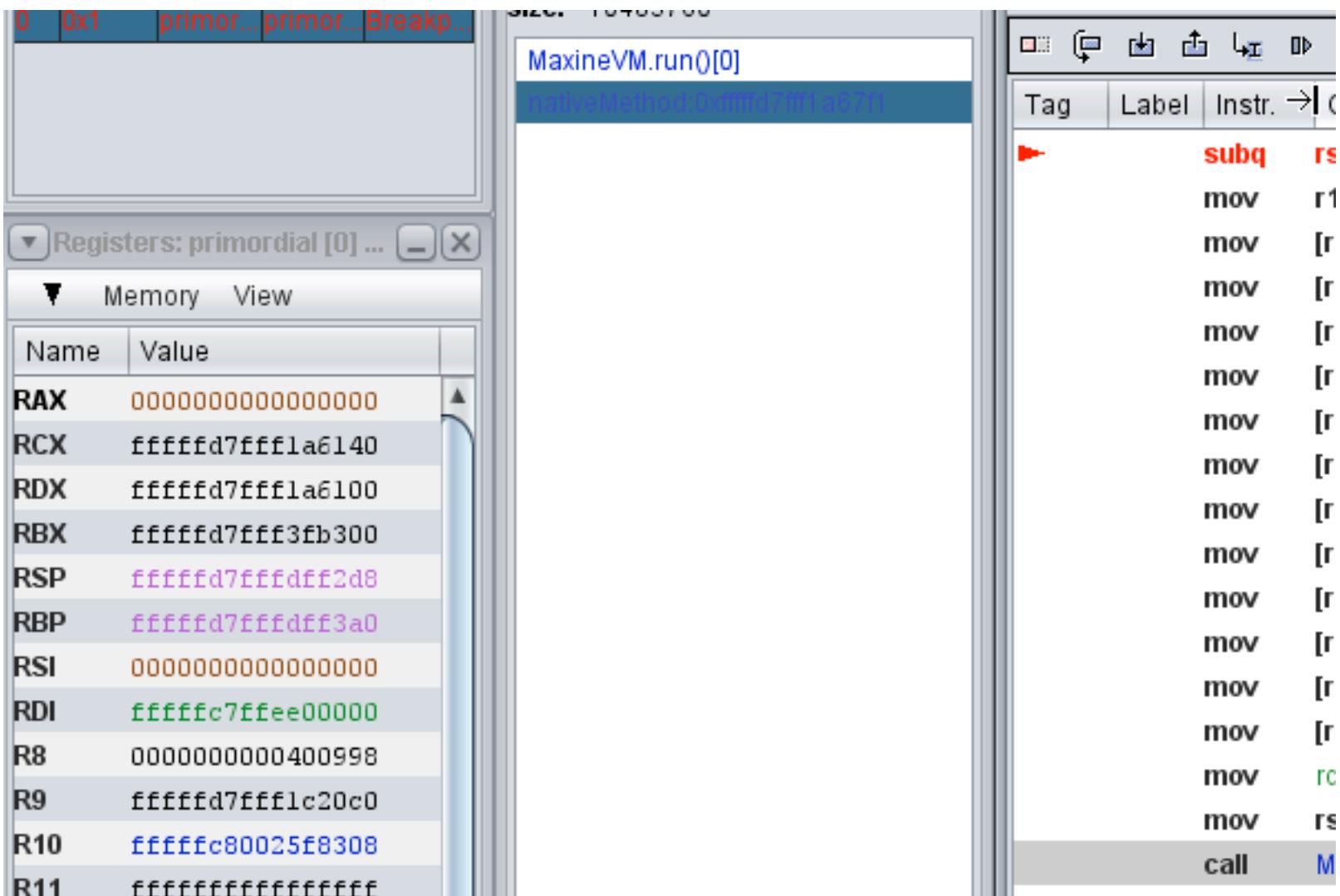
ORACLE®

2: View Menu



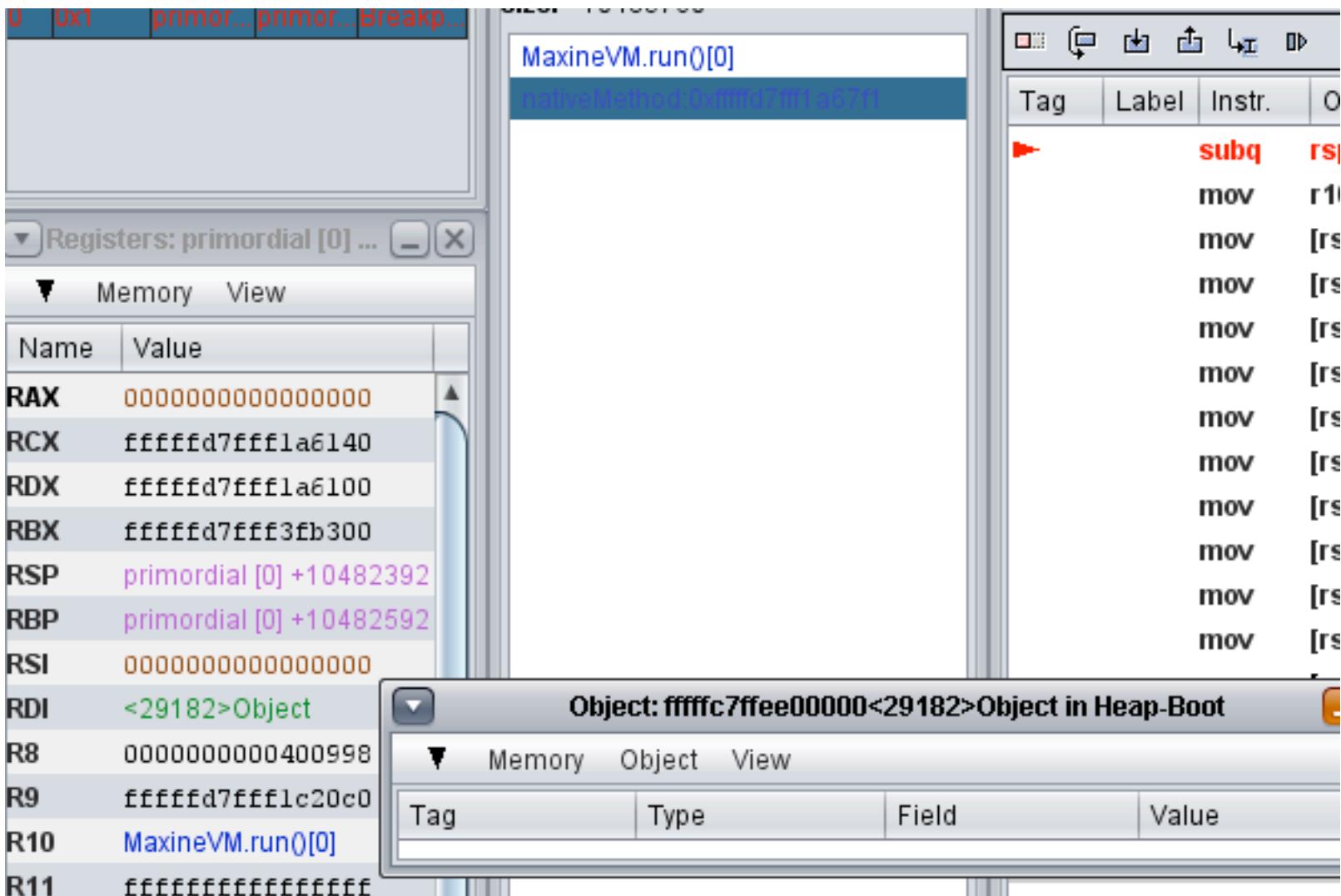
ORACLE®

3: Interpreted Data Display, Numeric Mode



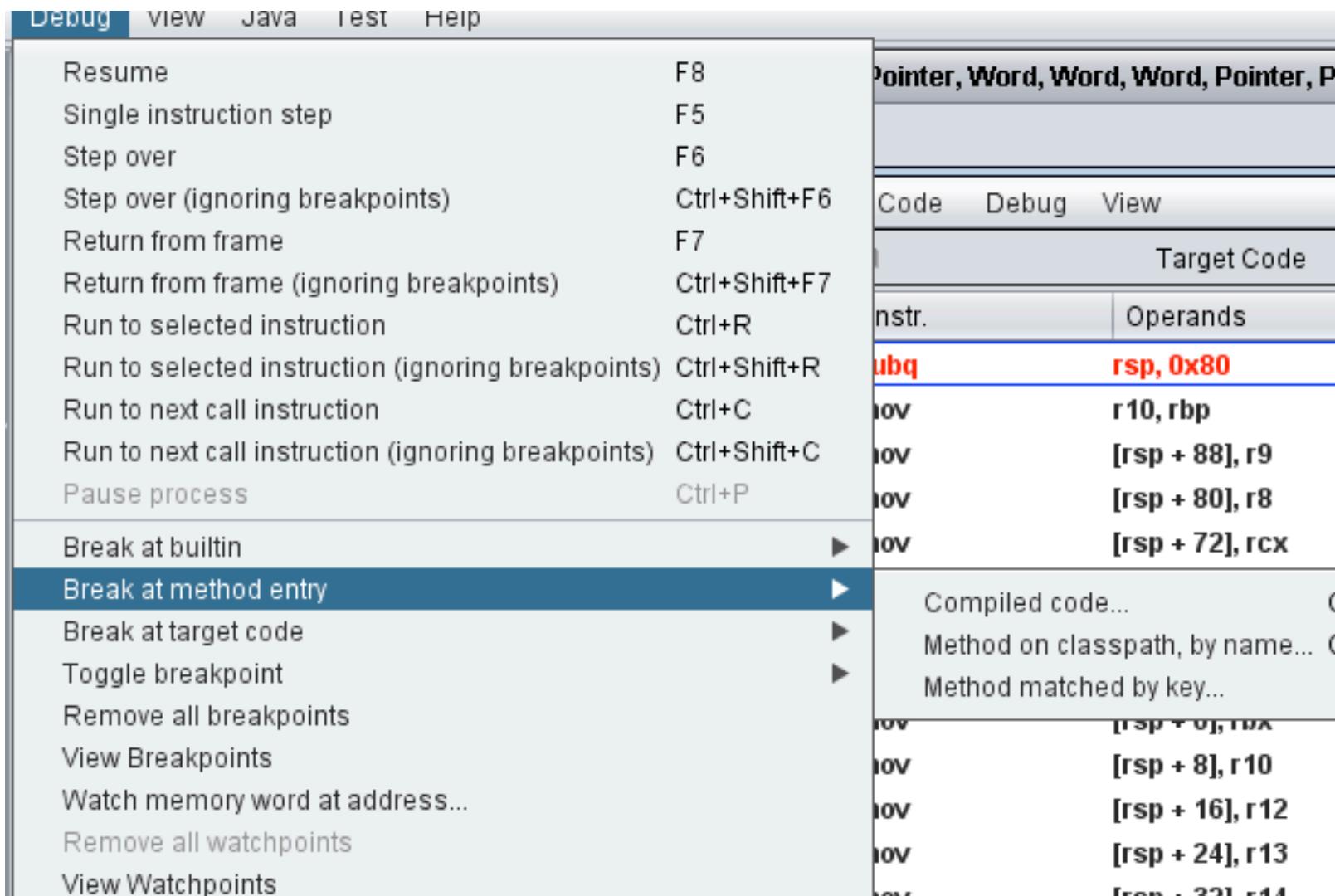
ORACLE®

4: Interpreted Data Display, An Object Inspector

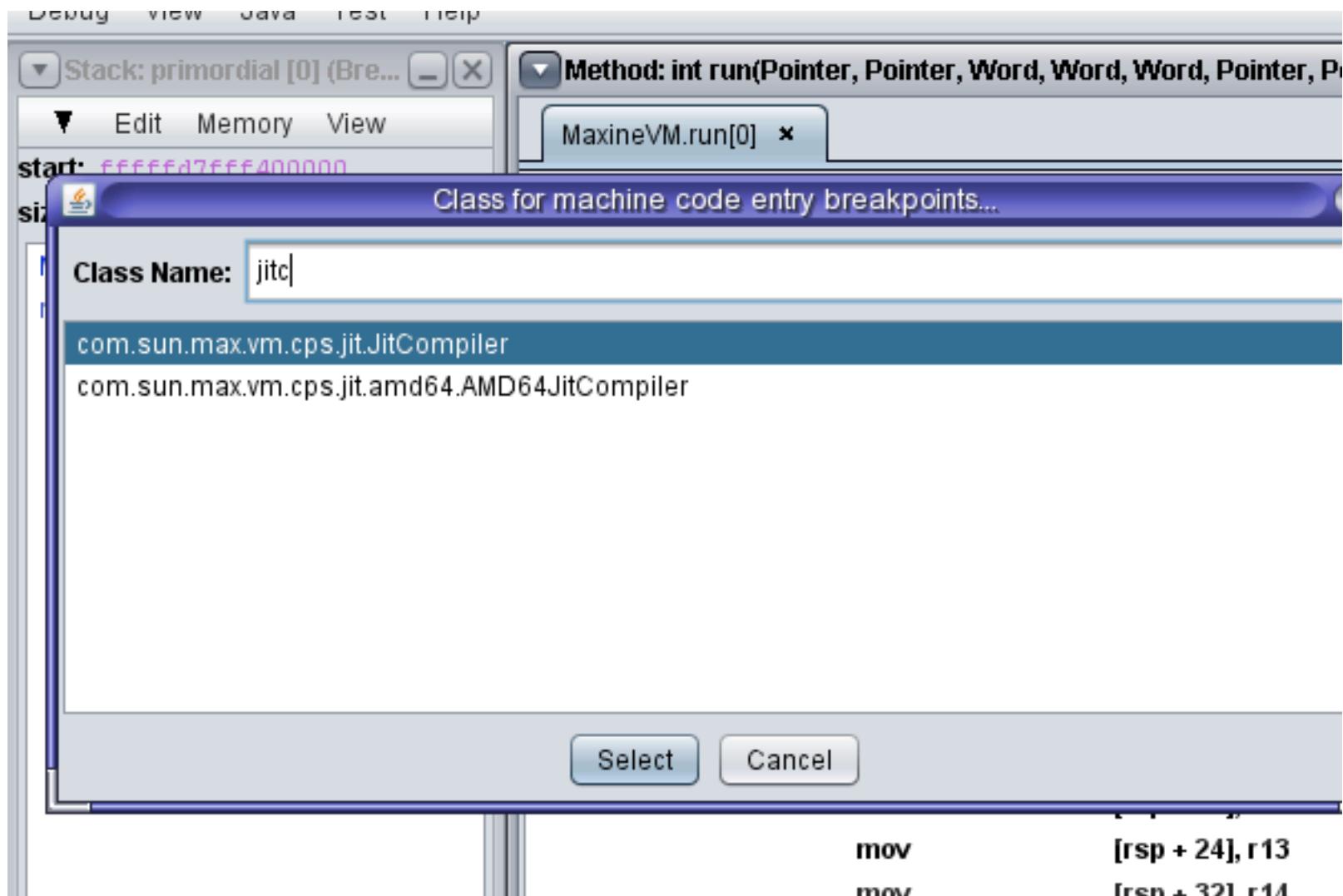


ORACLE®

5: Debugging Menu



6: Setting a Breakpoint



7: VM Meta-Information represented as objects

The screenshot shows a debugger interface with two main panes. The left pane displays memory dump information with columns for Type, Name, and Value. The right pane shows assembly code and a detailed view of an object's fields.

Memory Dump (Left Pane):

T...	Name	Value
	return addr...	fffffc8002a1cd12
slot 7	fffffc7fdea...	
slot 6	fffffc7fdea...	
slot 5	000000000000...	
slot 4	fffffc7fdea...	
slot 3	fffffc8001c...	
slot 2	fffffc7fdea...	
slot 1	fffffc7fdea...	
slot 0	fffffc7fdea133a0<30026>MethodActor{newInstance()}	

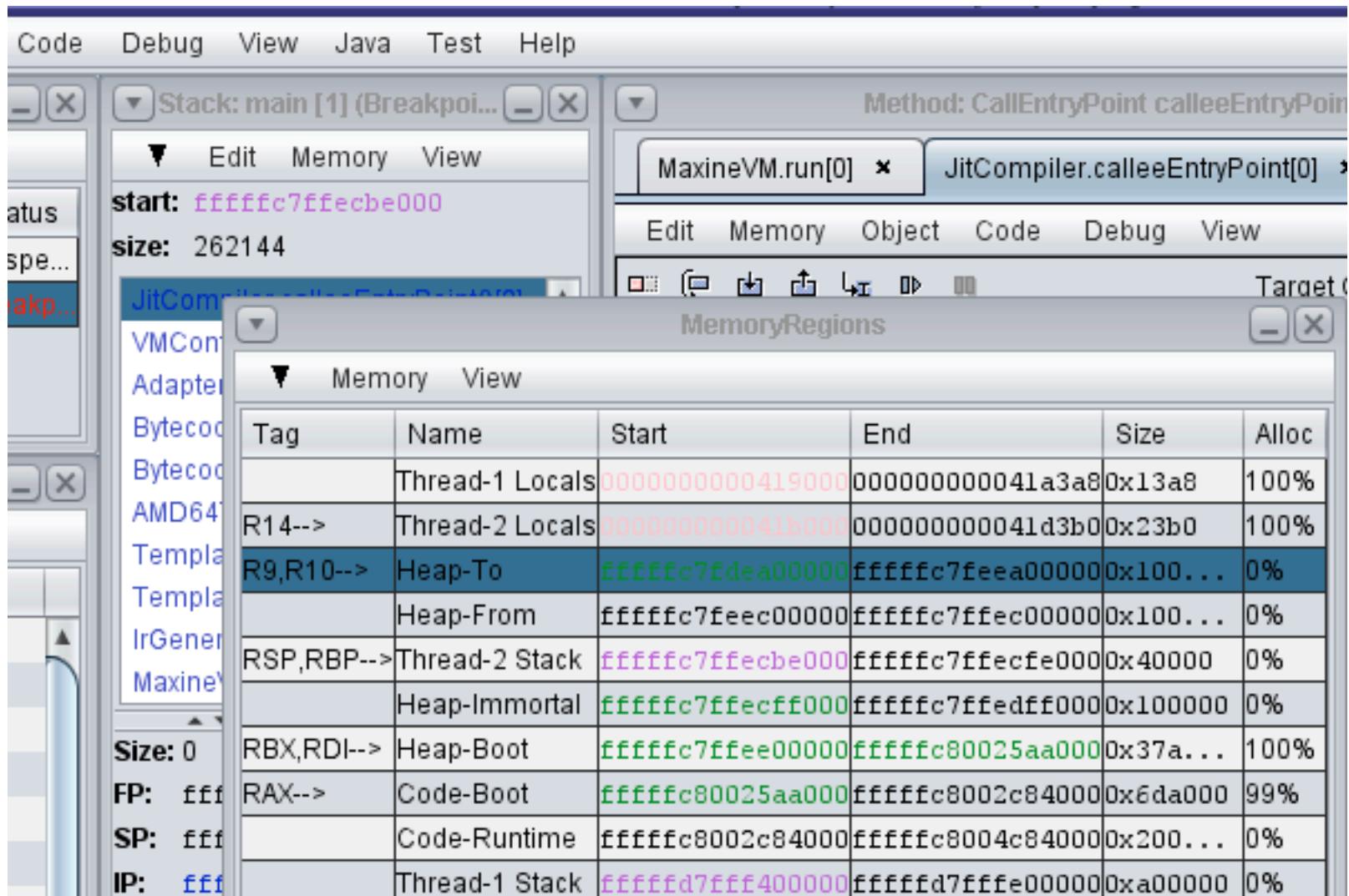
Object Details (Right Pane):

Object: fffffc7fdea133a0<30026>MethodActor{newInstance()} in He...

...	Type	Field	Value
	Utf8Constant	name	<29635>"newInstance"
	int	flags	Flags: 0x1001
	Descriptor	descriptor	<29636>SignatureDescriptor\$SignatureDescrip...
	ClassActor	holder	<29985>ClassActor{java_security_AccessContr...
	char	memberIndex	'0'

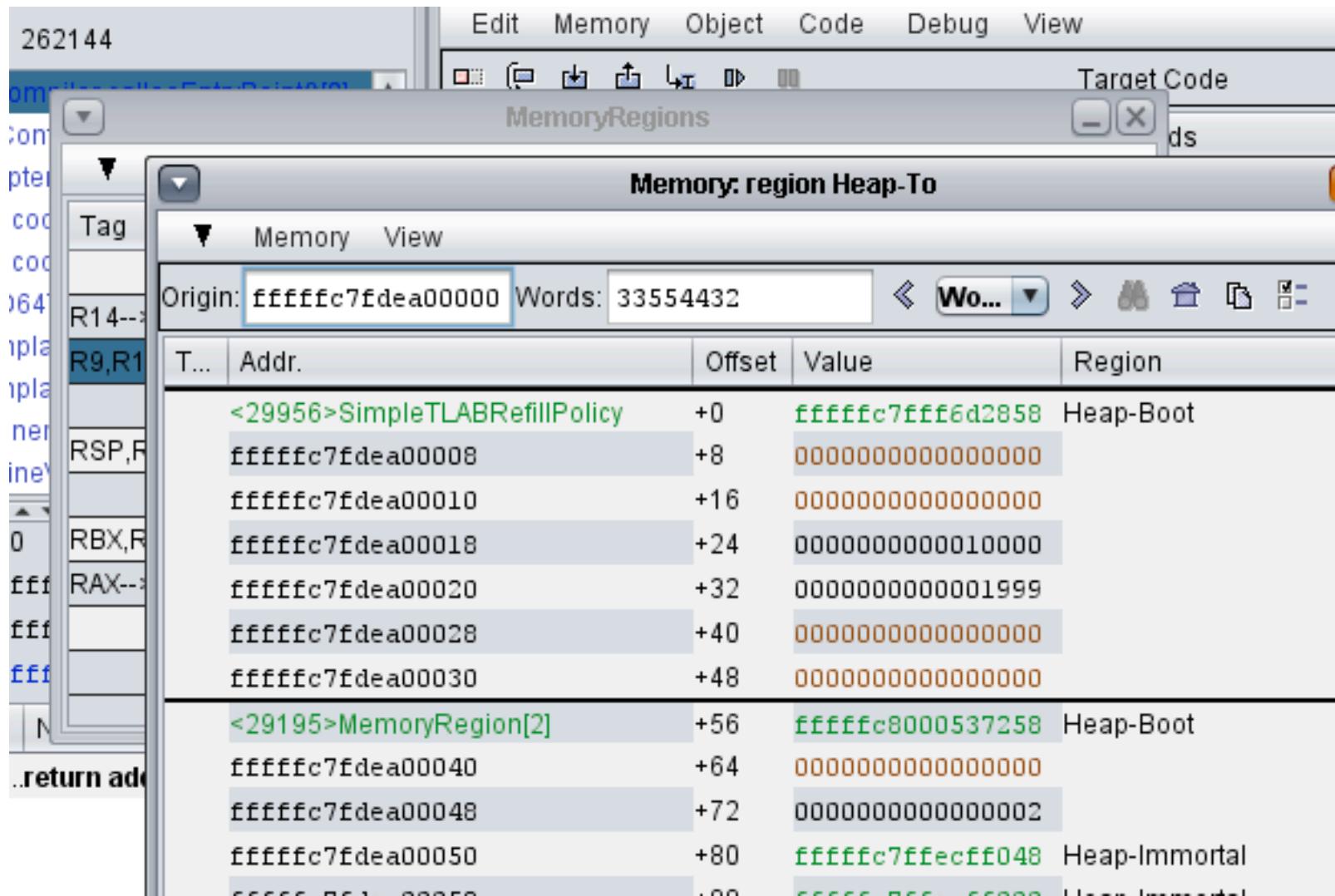
ORACLE®

8: Allocated Memory Regions



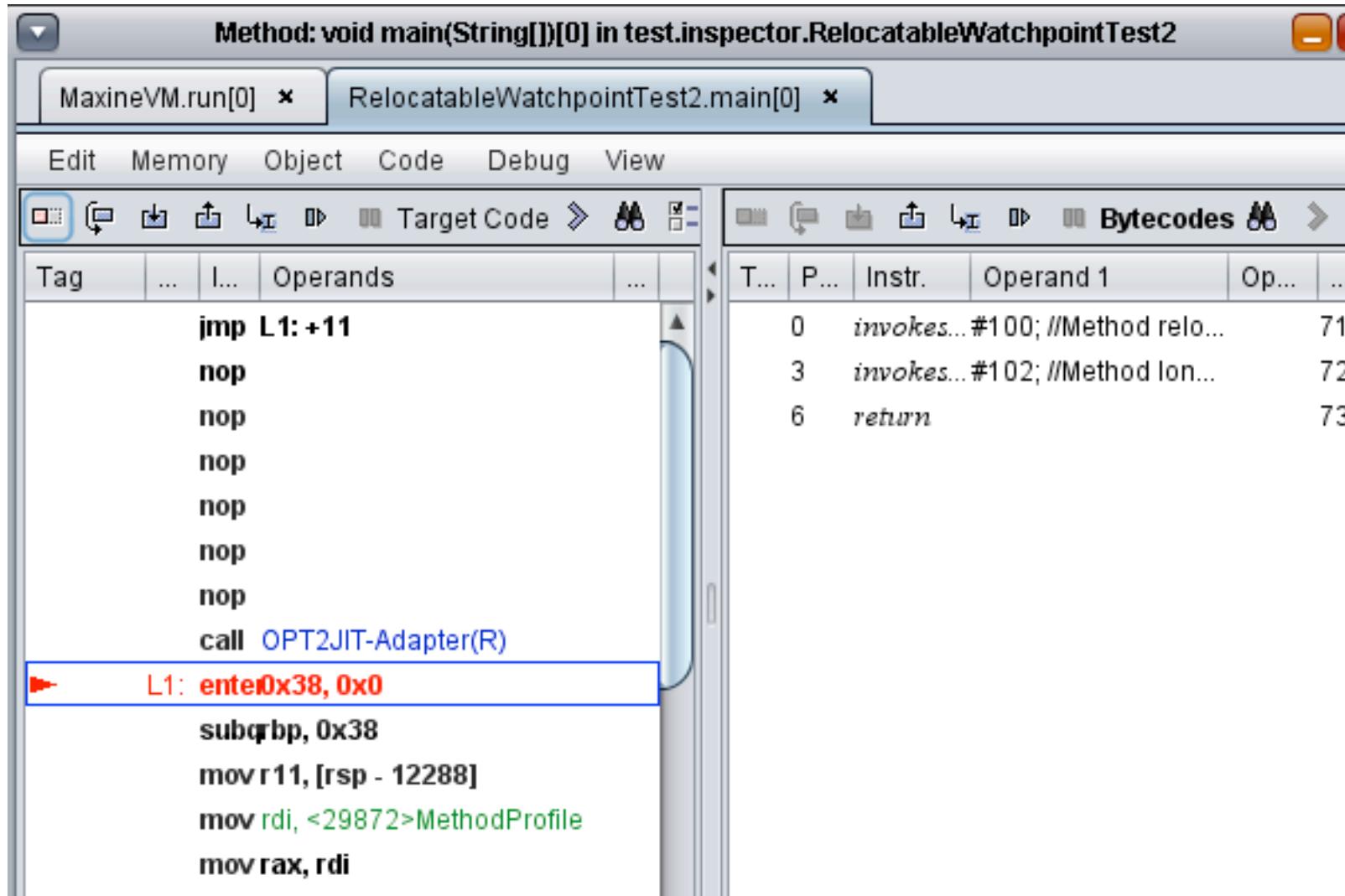
ORACLE

9: Memory Inspection



ORACLE

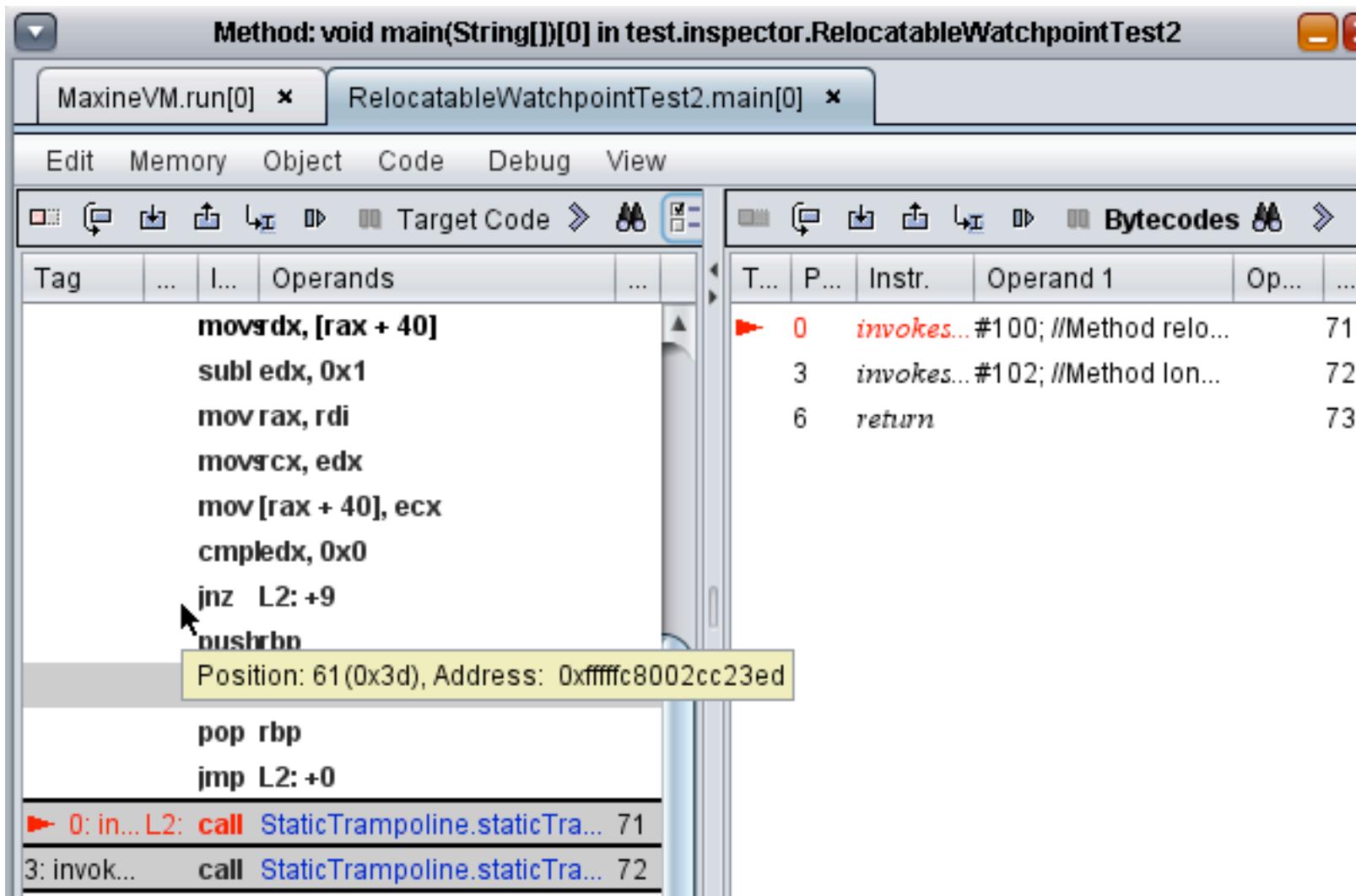
10: Method Viewing: Machine Code & Bytecode



ORACLE®

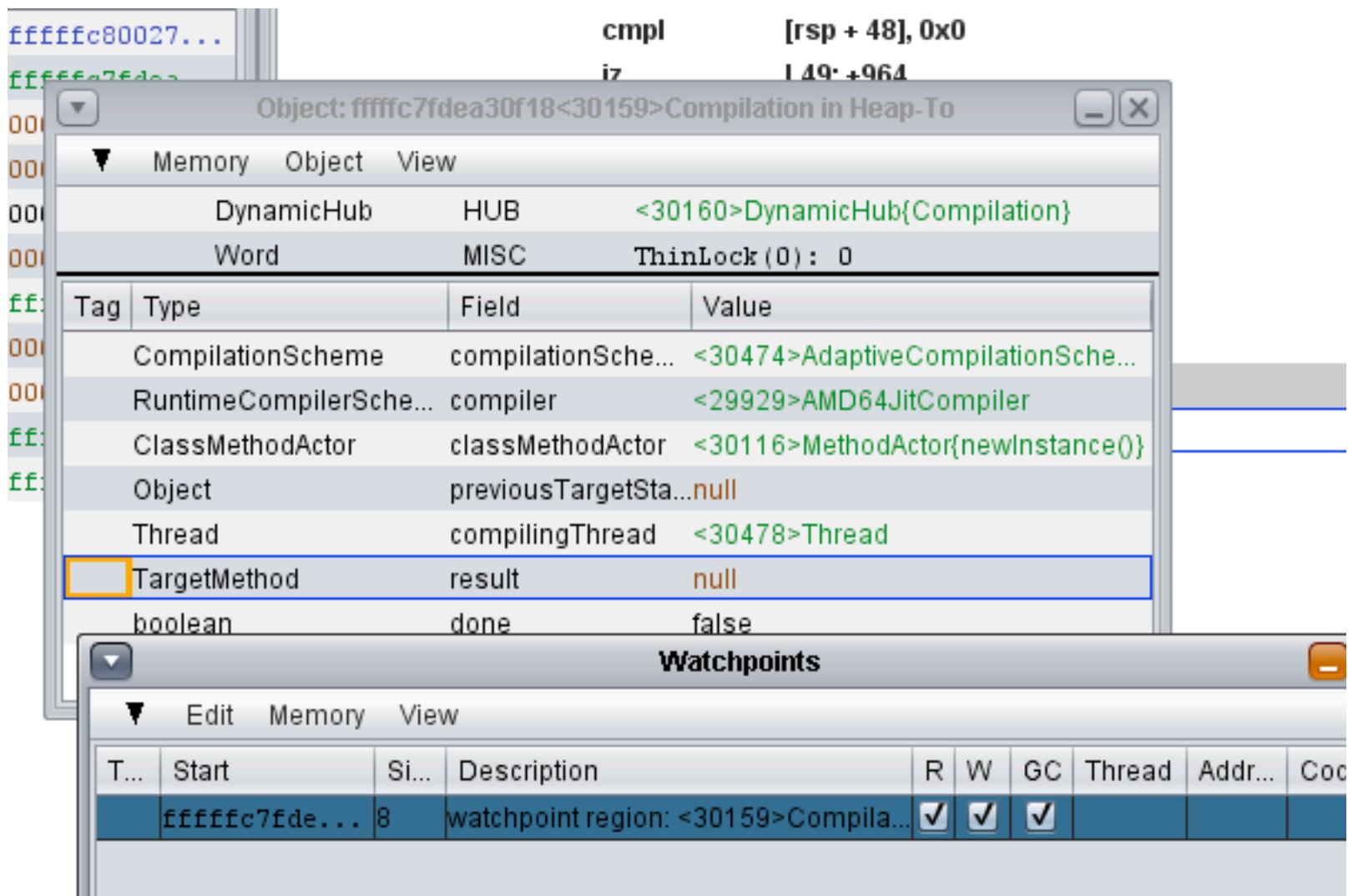
11: Dual Mode Debugging

(when instruction maps available)



ORACLE

12: Object Headers, Relocatable Field Watchpoints



ORACLE

Other capabilities

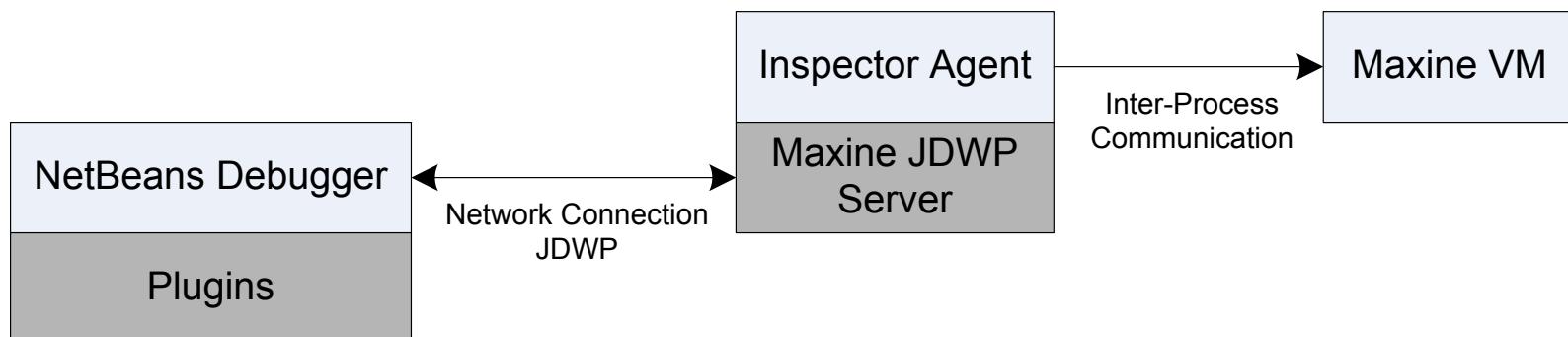
- Optional “columns” in every display, e.g
 - Address
 - Offset
 - Tags (pointed to by registers, breaks)
 - Region pointed-to
- Standard code locations for breaks
 - e.g. GC: start, stop, memory allocate/resize
- Double-click breakpoints, watchpoints
- Object Inspector lifetime awareness:
 - Dead objects (can only be determined at GC end)
 - Forwarded objects (for relocating GC implementations)
- Thread locals

Maxine Project Status

- VM runs on:
 - Solaris-x64
 - Linux-x64
 - Darwin-x64
 - Xen-x64 (Virtual Edition)
- VM executes SPEC JVM98 and DaCapo benchmarks
 - (usually)
- Inspector works on all platforms
 - Watchpoints require Solaris or VE (for now)
- VM executes 2x - 5x slower than HotSpot
 - Bottleneck: Existing CPS compiler
 - Being addressed by new C1X compiler

Maxine Inspector - Directions

- *Task-specific* extensions, e.g. GC development
- *Platform-specific* extensions, e.g. for Virtual Edition
- *Implementation-specific* extensions, e.g. GC algorithm
- Re-engineer Inspector architecture into two layers:
 - *Inspector Agent*: extracts, models Maxine VM state
 - *Inspector Client*: presentation, interaction, analysis
 - Code separation and interface design underway (background)
- Summer 2008 experiment: standard Java debugging



ORACLE®

Maxine Project Acknowledgements

Team:

Laurent Daynès

Mick Jordan

Doug Simon (PI)

Former:

Bernd Mathiske

Ben Titzer

Greg Wright

Manager: Mario Wolczko

Others: Tom Rodriguez

Christian Wimmer

Ken Russell

Robert Griesemer

Paul Caprioli

Interns

Athul Acharya

Aritra Bandyopadhyay

Michael Bebenita

Marcelo Cintra

Michael Duller

Abdulaziz Ghuloum

Yi Guo

Christos Kotselidis

Puneet Lakhina

David Liu

Karthik Manivannan

Sumeet Panchal

Hannes Payer

Sunil Soman

Lukas Stadler

Simon Wilkinson

Thomas Würthinger

Hiroshi Yamauchi

Thank You

- <http://labs.oracle.com/projects/maxine/>
- <http://wikis.sun.com/display/MaxineVM/Home>
- <http://wikis.sun.com/display/MaxineVM/Inspector>



ORACLE®