

a JIT Smalltalk VM

written in itself



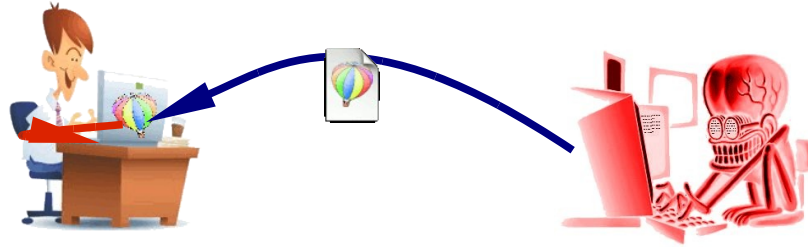
Javier Burroni



Gerardo Richarte



ESUG 2009 – JIT Security



test: arg
^self with: 0 with: inst with: arg

```
1 <13> PushSmallInteger0
2 <8C> PushInstance1
3 <55> PushArgument1
4 <0A> LoadSelf
5 <E2> SendSelector1
6 <48> Return
```

```
@1: 7F647D: push EBP
     7F647E: mov EBP, ESP
     7F6480: push EAX
     7F6481: mov ESI, EAX
     ...
@2: 7F6492: push 1 ; PushSmallInteger0
     7F6497: push [ESI] ; PushInstance1
     7F6499: push [EBP+8] ; PushArgument1
     7F649C: mov EAX, ESI ; LoadSelf
     7F649E: call 740207 ; SendSelector1
     7F64A3: mov ESP, EBP ; Return
     7F64A5: pop EBP
     7F64A6: mov ESI, [EBP-4]
     7F64A9: ret 4 NEAR
```

BytecodeNativizerPushR
assembler pushR

Assembler >> #pushR
self assembleByte: 16r50 " push eax "

New Goal

Smalltalk VM



New Goal

Smalltalk VM

Written in Smalltalk



New Goal

Smalltalk VM

Written in Smalltalk

Leveraging the environment



Smalltalk VM

- ◆ Written in Smalltalk
 - ◆ .exe generation
- ◆ Method lookup
- ◆ Object format
- ◆ Memory management
 - ◆ Object creation / copy
 - ◆ GC
 - ◆ Become
- ◆ Primitives
- ◆ FFI
- ◆ Processes
- ◆ Callbacks
- ◆ etc



Closure Bug (6-Aug-2010)

```
testThomasMuhrNestedBlocks
```

```
| abcdef |  
abcdef := [| var1 |  
  var1 := 'AB'.  
  [| var2 |  
    var2 := 'CD'.  
    [| var3 |  
      var3 := 'EF'.  
      [var1 , var2 , var3] value] value] value] value.  
self assert: abcdef = 'ABCDEF'
```

BOOM!



Fix #1 (7-Aug-2010)

```
IndirectEscapeBytecode >> #assemble
```

```
type := self nextByte.
```

```
argument := type \\ 4 = 0
```

```
ifTrue: [self nextIndex]
```

nextIndex + 1



```
ifFalse: [type \\ 4 + 3].
```

```
type < 5 ifTrue: [^self loadEnvironmentTemporaryIndirect].
```

```
type < 9 ifTrue: [^self pushEnvironmentTemporaryIndirect].
```

```
^self storeEnvironmentTemporaryIndirect
```

Closure Bug (6-Aug-2010)

testThomasMuhrNestedBlocks


```
| abcdef |  
abcdef := [| var1 |  
  var1 := 'AB'.  
  [| var2 |  
    var2 := 'CD'.  
    [| var3 |  
      var3 := 'EF'.  
      [var1 , var2 , var3] value] value] value] value.  
self assert: abcdef = 'ABCDEF'
```

BOOM!



Fix #3 (9-Aug-2010)

```
loc_1000ACFA:  
push    esi  
call    JIT_getNextIndex  
mov     edi, [esp+10h+aBytecodeNativizer]  
add     esp, 4  
dec     eax  
mov     [edi+4], eax  
jmp     short loc_1000AD29
```



Closure Bug (6-Aug-2010)

```
testThomasMuhrNestedBlocks
```

```
| abcdef |
```

```
abcdef := [| var1 |
```

```
  var1 := 'AB'.
```

```
  [| var2 |
```

```
    var2 := 'CD'.
```

```
    [| var3 |
```

```
      var3 := 'EF'.
```

```
      [var1 , var2 , var3] value] value] value] value.
```

```
self assert: abcdef = 'ABCDEF'
```

BOOM!



Fix #2 (9-Aug-2010)

```
putCode: code
```

```
min: min
```

```
max: max
```

```
range: idx
```

```
(idx >= min and: [idx <= max])
```

```
  ifTrue: [self putNext: code - max + idx - 1]
```

```
  ifFalse: [
```

```
    self putNext: code.
```

```
    self putIndex: idx]
```

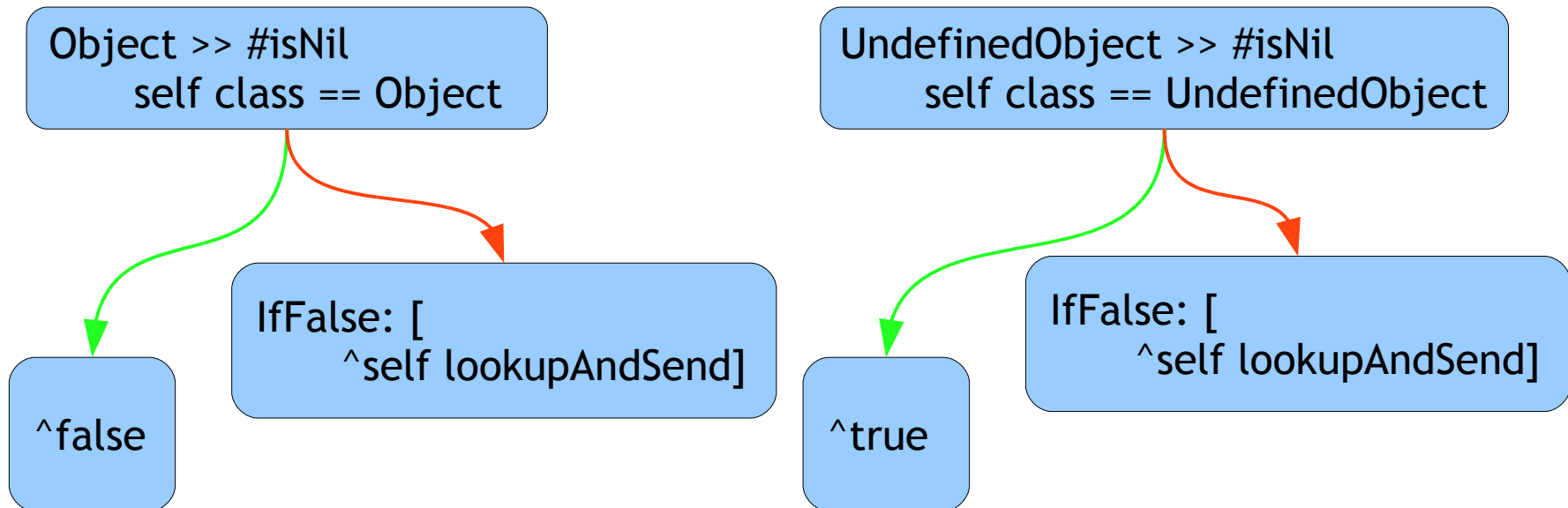
idx + 1



Smalltalks 2009 – code freezer

```
GenerationalGCBuilder >> #populateInternals
(nativizer := BigNativizer new)
  addAllMethodsFor: GCSPACE;
  addAllMethodsFor: ObjectGenerationalGC;
  at: #isSmallInteger add: SmallInteger;
  at: #bitXor: add: SmallInteger;
  at: #between:and: add: SmallInteger;
  at: #size addAll: (Array with: Dictionary with: Object);
  at: #isSmallInteger default: Object >> #isSmallInteger;
  at: #_generation default: ExtensionBytecode class >> #_generation;
  ...
```

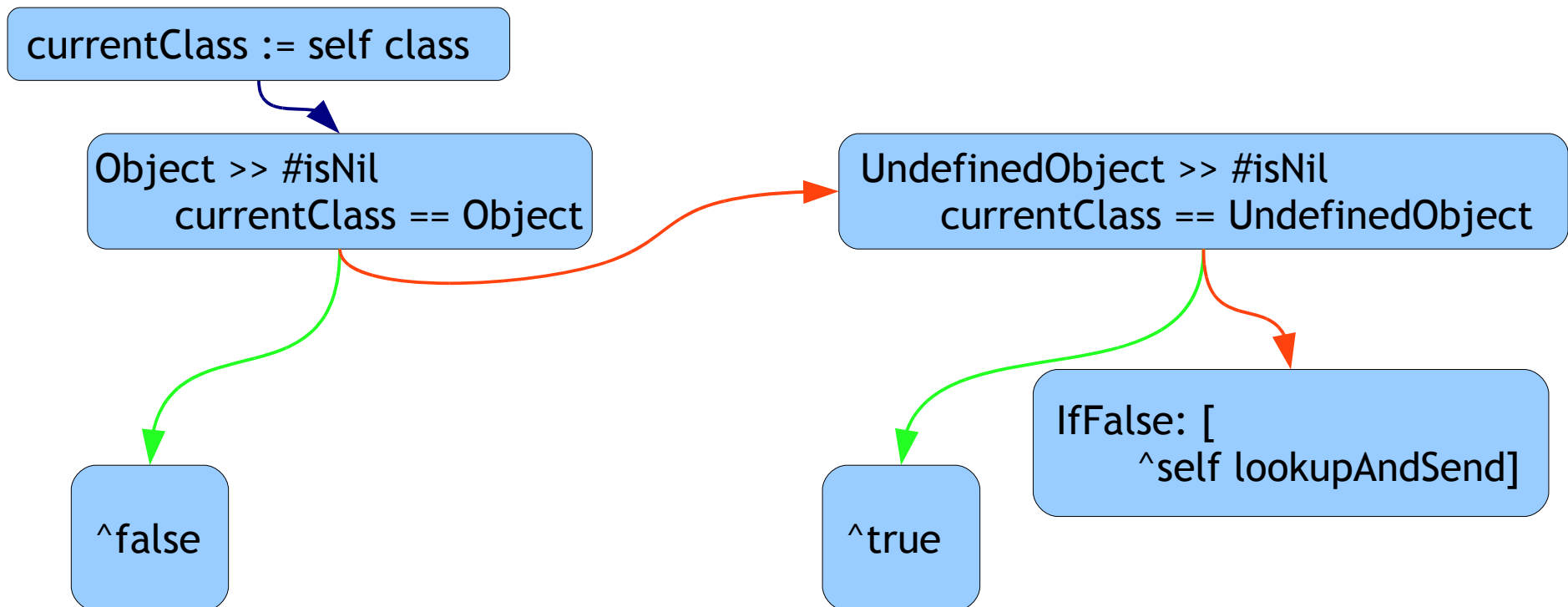
Lookup



Frozen Code: “Lookup”

nativizer

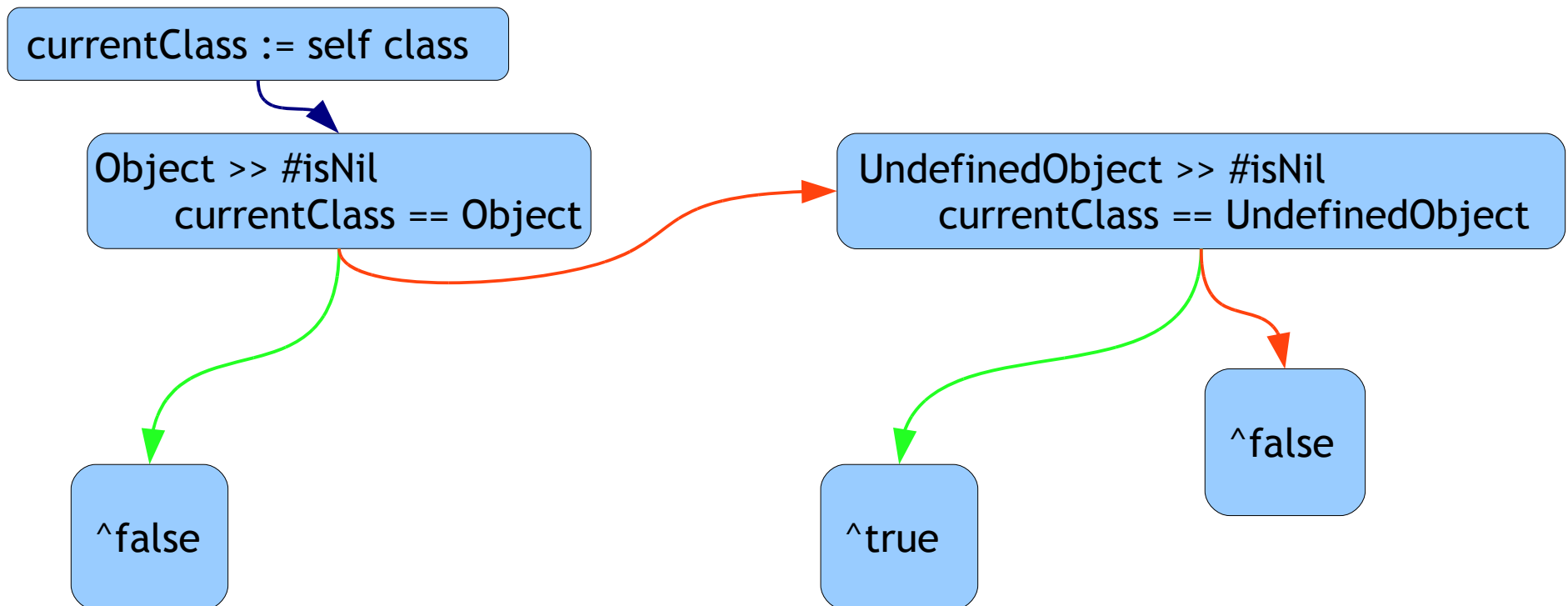
at: #isNil addAll: (Array with: Object with: UndefinedObject)



Frozen Code: “Lookup”

nativizer

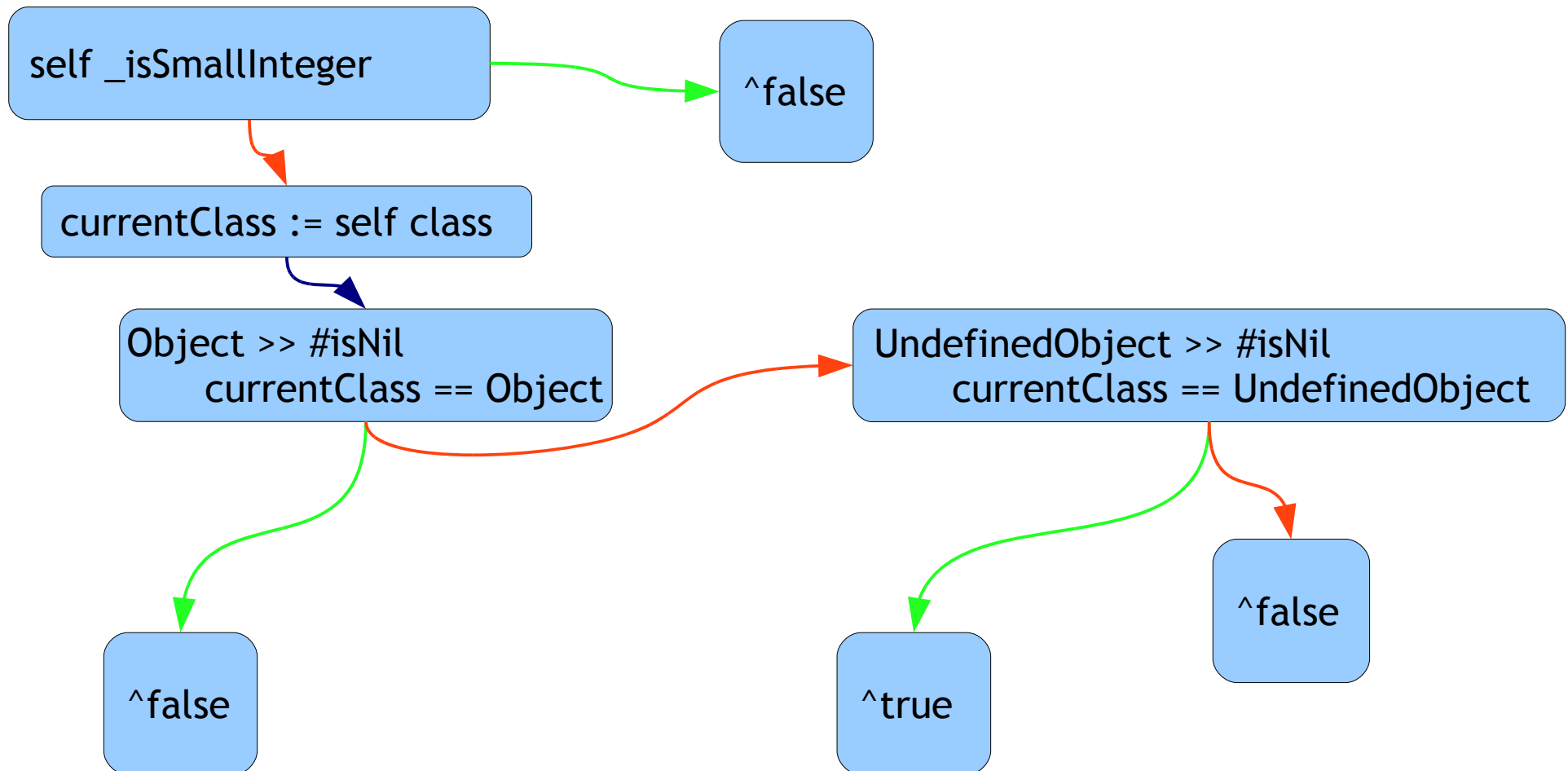
```
at: #isNil addAll: (Array with: Object with: UndefinedObject);  
default: Object >> #isNil
```



Frozen Code: "Lookup"

nativizer

```
at: #isNil addAll: (Array with: Object with: UndefinedObject);  
default: Object >> #isNil;  
at: #isNil add: SmallInteger
```

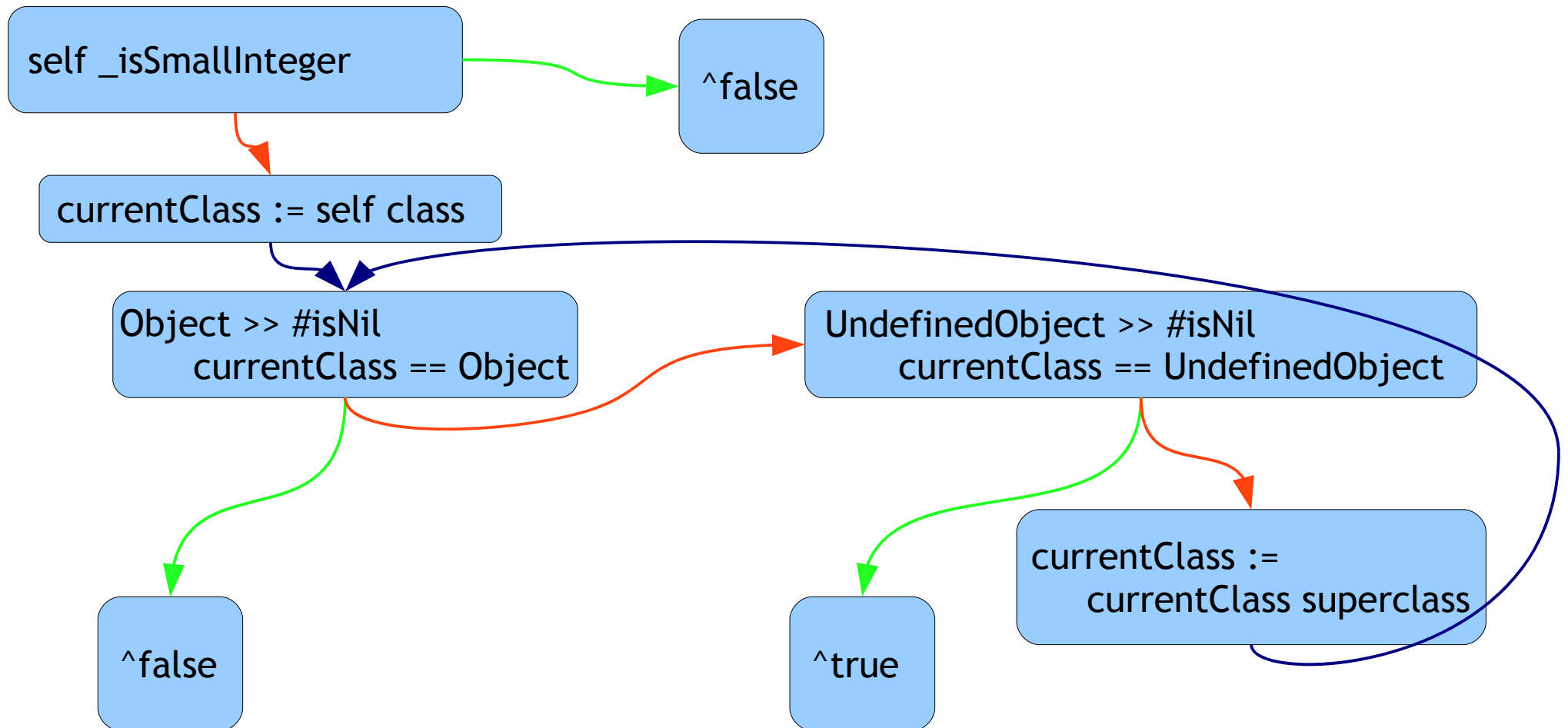


Frozen Code: Lookup

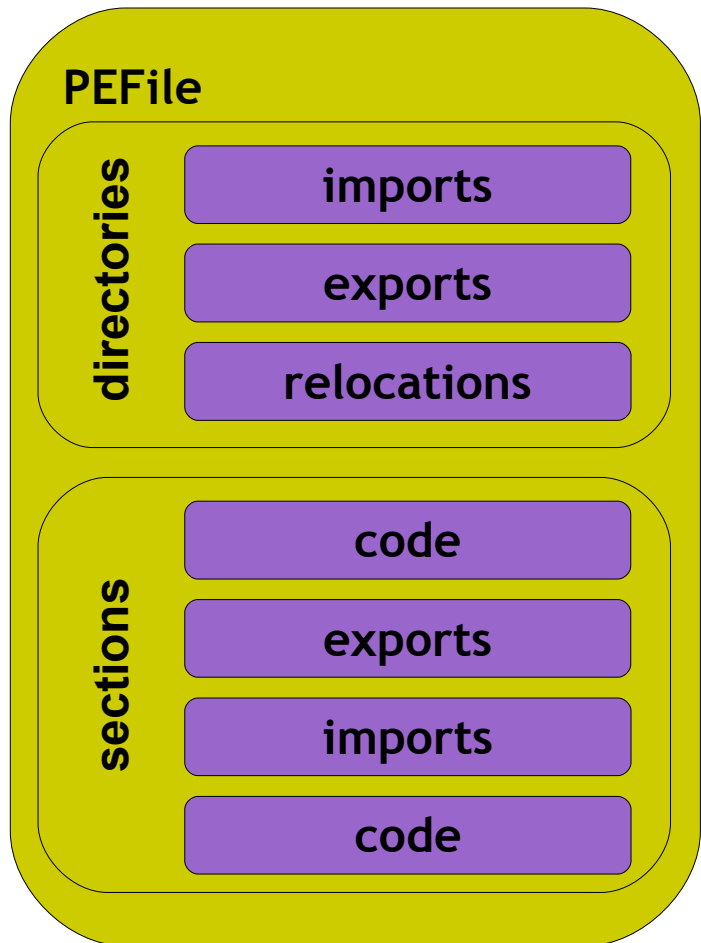
nativizer

at: **#isNil** addAll: (Array with: Object with: UndefinedObject);

at: **#isNil** add: SmallInteger



.exe generation



```
_commit: newSize  
  assembler  
    loadArgFromContext: index;  
    convertArgToNative;  
    pushArg;  
    convertToNative;  
    pushR;  
    pushConstant: 4;  
    pushConstant: 4096;  
    pushArg;  
    pushR;  
    callTo: 'VirtualAlloc' from: 'KERNEL32.DLL';  
    convertToSmalltalk
```

nativizer

addAllMethodsFor:

(Array with: **GenerationalGC** with: **GenerationalGC** class with: **GCSpace**);

at: #_commit: _primitive: **ExtensionBytecode** >> #_commit

GCSpace

Object subclass: #GCSpace
InstanceVariableNames: '
 contents
 base
 nextFree
 commitLimit
 reservedLimit '
classVariableNames: ''
poolDictionaries: ''

old

A

B

- **old** includes: **anObject**
- **to** reset
- **from** isReferredBy: **object**
- **old** growTo: **newLimit**
- **moved := to** shallowCopy: **object**
- **from** flipWith: **to**

includes: **anObject**

^**anObject** isSmallInteger not

and: [**anObject** _oop between: **self** base and: **self** committedLimit]

reset

self nextFree: **self** base

GenerationalGC

Object subclass: #GenerationalGC
InstanceVariableNames: '
old from to
rememberSet literalsReferences
weakContainers ephemeron
rescuedEphemeron'
classVariableNames: ''
poolDictionaries: ''

old

from

to

- flipper purgeRoots
- flipper follow: object
- flipper moveToOldOrTo: object
- flipper
fixReferencesOrSetTombstone:
weakContainer
- flipper addInterrupt

collect
self

```
followRoots;  
walkStack;  
rescueEphemeron;  
traverseWeakContainers;  
flipSpaces
```

GenerationalGC

GenerationalGC

old

from

to

collect

self

followRoots;

walkStack;

follow: **object**

| **size** |

size := 0.

object _isBytes ifFalse: [

object _hasWeaks

ifTrue: [self **addWeakOrEphemeron**: **object**]

ifFalse: [**size** := **object** _size]].

self follow: **object** count: 1 + **size** startingAt: 0

GenerationalGC

GenerationalGC

old

from

to

collect

self

followRoots;

walkStack;

follow: **objects** count: **size** startingAt: **base**

| **index** |

index := **base** - 1.

size timesRepeat: [| **object** |

index := **index** + 1.

object := **objects** **_basicAt:** **index**.

(**fromSpace** includes: **object**) ifTrue: [

moved := self **moveToOldOrTo:** **object**.

objects **_basicAt:** **index** **put:** **moved**.

self **follow:** **moved**]]

GenerationalGC

GenerationalGC

old

from

to

collect

self

followRoots;

walkStack;

follow: **objects** count: **size** startingAt: **base**

| **index** |

index := **base** - 1.

size timesRepeat: [| **object** |

index := **index** + 1.

object := **objects** **_basicAt:** **index**.

(**fromSpace** includes: **object**) **ifTrue:** [

object **_isProxy**

ifTrue: [**objects** **_basicAt:** **index** **put:** **object** **_proxee**]

ifFalse: [| **moved** |

moved := **self** **moveToOldOrTo:** **object**.

objects **_basicAt:** **index** **put:** **moved**.

self **follow:** **moved**]]]

GenerationalGC

GenerationalGC

old

from

to

collect

self

traverseWeakContainers

traverseWeakContainers

weakContainers

do: [:weakContainer |

self fixReferencesOrSetTombstone: weakContainer]

self reset: weakContainers

fixReferencesOrSetTombstone: weakContainer

| size |

size := weakContainer _size.

1 to: size do: [:index | | instance |

instance := weakContainer _basicAt: index.

(fromSpace includes: instance)

ifTrue: [| referenceOrThombstone |

referenceOrThombstone := instance _isProxy

ifTrue: [instance _proxee]

ifFalse: [residueObject].

weakContainer _basicAt: index put: referenceOrThombstone]]

GenerationalGC

GenerationalGC

old

from

to

collect

self

rescueEphemeron

rescueEphemeron

| unknowns rescan |

unknowns := self localStack.

rescan := false.

[ephemeron isEmpty] whileFalse: [

rescan := self followEphemeronCollectingIn: unknowns.

rescan

ifTrue: [ephemeron addAll: unknowns]

ifFalse: [

unknowns do: [:ephemeron | self rescueEphemeron: ephemeron]].

self reset: unknowns]

[Hayes1997] Barry Hayes, *Ephemeron: a new finalization mechanism*

GenerationalGC

GenerationalGC

old

from

to

```
collect  
self  
flipSpaces
```

flipSpaces

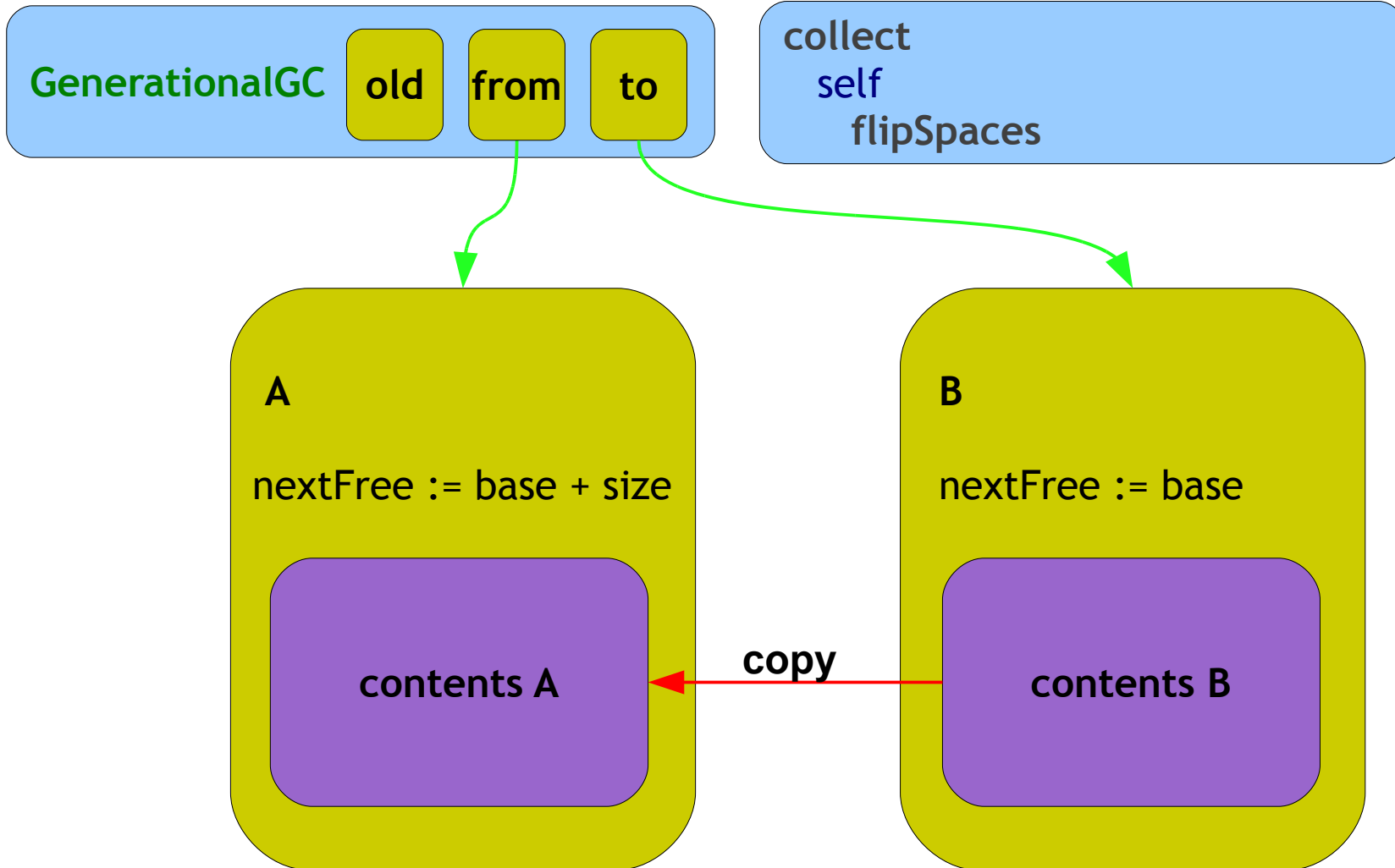
```
fromSpace flipWith: toSpace.  
toSpace reset.
```

```
GCSpace >> flipWith: space  
| aux |
```

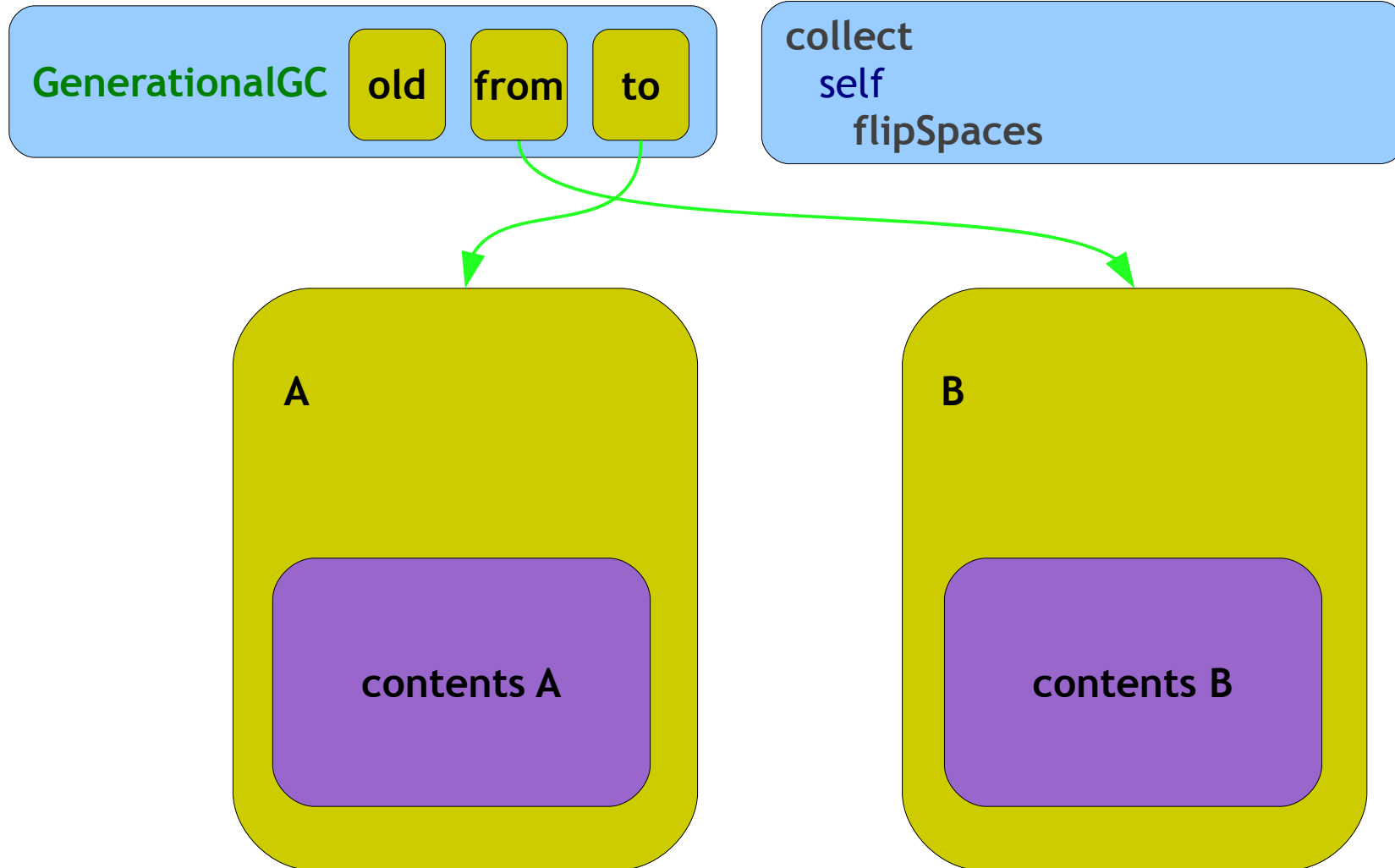
```
aux := space base.  
space base: self base.  
self base: aux.
```

...

GenerationalGC



GenerationalGC



GenerationalGC: testing

GenerationalGC

old

from

to

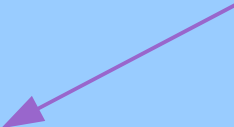
collect

self

followRoots

testFollowObject

```
| flipper from array string byteArray root flipSize currentSize |
flipper := self defaultFlipper.
from := flipper fromSpace.
array := from shallowCopy: (Array new: 3).
string := from shallowCopy: 'a String' copy.
from shallowCopy: 'leaked' copy.
byteArray := from shallowCopy: #[1 2 3] copy.
array
  at: 1 put: 1;
  at: 2 put: string;
  at: 3 put: byteArray.
root := Array with: array.
string := byteArray := array := nil.
flipper addRoot: root; followRoots.
flipSize := flipper toSpace nextFree - flipper toSpace base.
currentSize := flipper fromSpace nextFree - flipper fromSpace base.
self
  assert: flipSize < currentSize;
  assert: currentSize - flipSize = (8 + 'leaked' size roundTo: 4)
```



GenerationalGC: testing

testFollowObject

```
| flipper from array string byteArray root flipSize currentSize |  
flipper := self defaultFlipper.  
from := flipper fromSpace.  
array := from shallowCopy: (Array new: 3).  
string := from shallowCopy: 'a String' copy.  
from shallowCopy: 'leaked' copy.  
byteArray := from shallowCopy: #[1 2 3] copy.
```

testFollowObject

```
| flipper from array string byteArray root flipSize currentSize |  
flipper := self defaultFlipper.  
self  
  execute: [:from |  
    array := from shallowCopy: (Array new: 3).  
    string := from shallowCopy: 'a String' copy.  
    from shallowCopy: 'leaked' copy.  
    byteArray := from shallowCopy: #[1 2 3] copy]  
  proxying: flipper fromSpace.
```

aDemo



VMBuilder installGenerational

```
copy: object to: space  
  | copy |  
  counter := counter + 1.  
  copy := space shallowCopy: object.  
  object _proxee: copy.  
  ^copy
```

anotherDemo



JIT enableCoverage

setActivatedFlag

| delta |

delta := 2 - self prologueSize - (assembler sizeof: #call:).

assembler

loadTempPC;

pushR;

loadConstant: 3;

storeByteIntoTemporary: delta;

popR

What now?

- ◆ Closure for JIT
- ◆ Memory Management
 - ◆ Mark & Sweep
 - ◆ new
 - ◆ become:
- ◆ Method Lookup (strategies)
- ◆ etc.



¡gracias!

