Interpretation & Just in Time

Jan Vraný

Department of Computer Science and Engineering
Czech Technical University In Prague
Faculty of Electrical Engineering

December 16, 2008

Outline

Bytecode

Interpretation & Just in time

Context & Block Optimization

Smalltalk bytecode set

PUSH_MARG
PUSH_MVAR
PUSH_BARG
PUSH_BVAR
PUSH_IVAR
PUSH_GLOB
PUSH_OBVAR
PUSH_OBARG
PUSH_LIT
PUSH_CONTEXT
DROP
DUP

STORE_MVAR
STORE_BVAR
STORE_IVAR
STORE_GLOB
STORE_OBVAR
RET_TOP
HOME_RET_TOP
SEND
SUPER_SEND
JMP_FALSE
JMP
MAKE_BLOCK

Example (SELF)

Method...

```
1 method:  marg
2    | mloc |
3    mloc ←Array with:marg.
4    mloc do:[:e|ivar←e].
5    ↑mloc
```

Example (SELF)

Method...

```
1 method:  marg
2    | mloc |
3    mloc ←Array with:marg.
4    mloc do:[:e|ivar←e].
5    ↑mloc
```

...and its bytecode

```
3 PUSH_GLOB Array
  PUSH_MARG 1
  SEND with:
  STORE_MVAR 1
4 PUSH_MVAR 1
  MAKE_BLOCK ...
  PUSH_BARG 1
  DUP
  STORE_IVAR 1
  RET_TOP

  SEND_DROP do:
5 PUSH_MVAR 1
  RET_TOP
```

Open-coding

Several selectors are open-coded by the bytecode compiler for performance:

Method...

```
1 method2:  bool
2   bool
3     ifTrue:[↑0]
4     ifFalse:[↑10]
```

Open-coding

Several selectors are open-coded by the bytecode compiler for performance:

Method...

```
1 method2:  bool
2   bool
3     ifTrue:[↑0]
4     ifFalse:[↑10]
```

...and its bytecode

```
2 PUSH_MARG 1
  JUMP_FALSE FalseBlock
3 PUSH_LIT 0
  RET_TOP
4 FalseBlock:
  PUSH_LIT 10
  RET_TOP
```

Open-coding

Several selectors are open-coded by the bytecode compiler for performance:

Method...

```
1 method2:  bool
2   bool
3     ifTrue:[↑0]
4     ifFalse:[↑10]
```

...and its bytecode

```
2 PUSH_MARG 1
  JUMP_FALSE FalseBlock
3 PUSH_LIT 0
  RET_TOP
4 FalseBlock:
  PUSH_LIT 10
  RET_TOP
```

That's bad, very bad!

SELF, The Power of Simplicity

Idea behing

Smalltalk is extremely complex (bloated) language. We need a new language purely based on message-sending.

In Smalltalk. . .

```
1 method:  marg
2    | mloc |
3    mloc←Array with:marg.
4    mloc do:[:e|ivar←e].
5    ↑mloc
```

SELF, The Power of Simplicity

Idea behind

Smalltalk is extremely complex (bloated) language. We need a new language purely based on message-sending.

In Smalltalk...

```
1 method: marg
2     | mloc |
3     mloc ← Array with: marg.
4     mloc do:[:e— ivar ←e].
5     ↑mloc
```

SELF, The Power of Simplicity

Idea behind

Smalltalk is extremely complex (bloated) language. We need a new language purely based on message-sending.

In SELF...

```
1 method: marg
2    | mloc |
3    mloc: (Array with: marg).
4    mloc do:[:e|ivar: e].
5    mloc
```

The only SELF bytecode set

PUSH_SELF

PUSH_LIT

SEND

DELEGATE

NON_LOCAL_RETURN

Example (SELF)

Method...

```
1 method:  marg
2    | mloc |
3    mloc:(Array with:marg).
4    mloc do:[:e|ivar:  e].
5    mloc
```

Example (SELF)

Method...

```
1 method:  marg
2    | mloc |
3    mloc:(Array with:marg).
4    mloc do:[:e|ivar:  e].
5    mloc
```

...and its bytecode

```
3 PUSH_SELF
  PUSH_SELF
  SEND Array
  PUSH_SELF
  SEND marg
  SEND with:
  SEND mloc:
4 SEND mloc
  PUSH_LIT 1
  SEND do:
5 SEND mloc
```

Outline

Bytecode

Interpretation & Just in time

Context & Block Optimization

Marcus Denker's slides

Outline

Bytecode

Interpretation & Just in time

Context & Block Optimization

Context & Block Optimization

C-stack for message sends.

Cheap block/full blocks.

Non-local variable access optimization.