MATHEMA

Type-Safe, Efficient, Low-Level-Programming for the Java Virtual Machine

CON6037

Michael Wiedeking

michael.wiedeking@mathema.de

MATHEMA Software GmbH

Henkestraße 91

91052 Erlangen

Germany

Copyright © 2012, MATHEMA Software GmbH

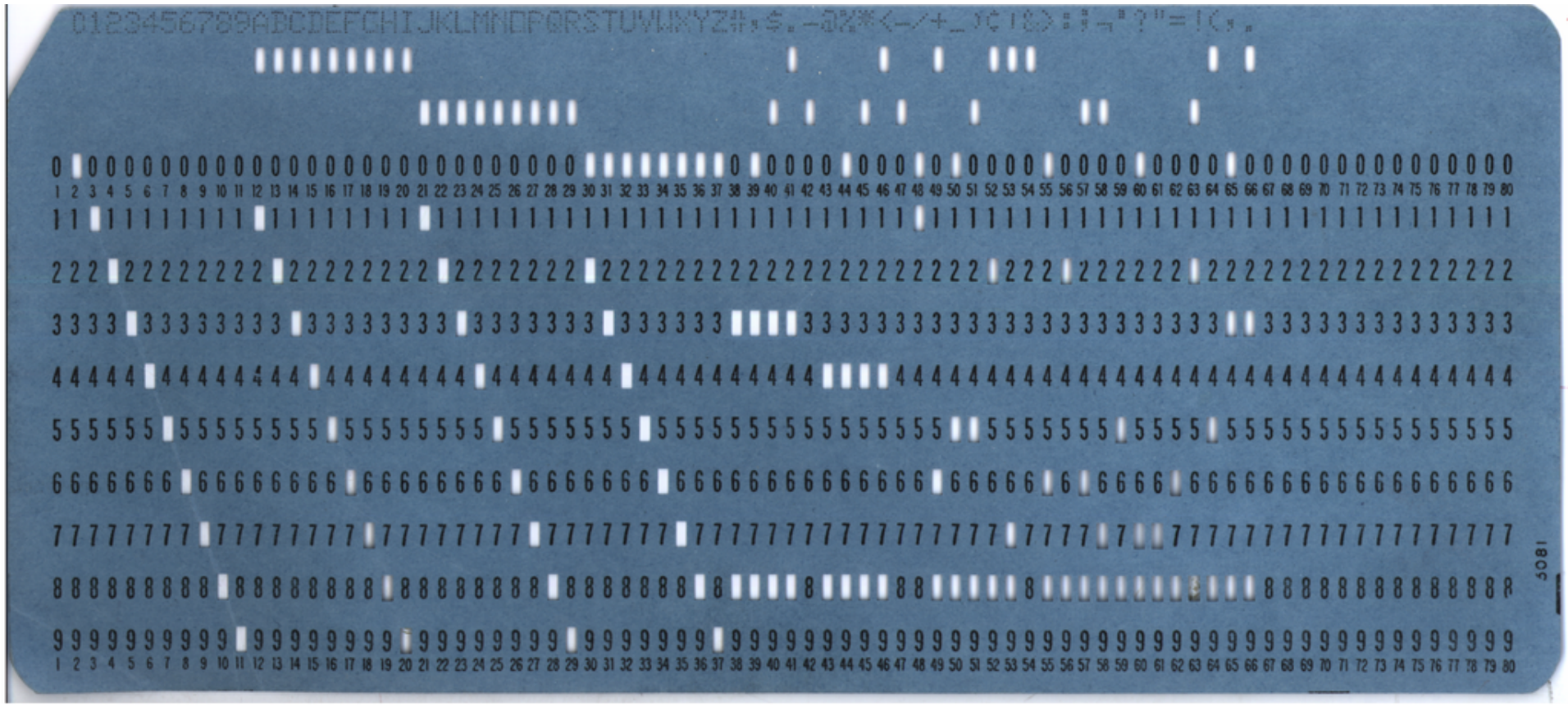
Motivation

Copyright © 2012, MATHEMA Software GmbH

$$x \diamond y$$

Copyright © 2012, MATHEMA Software GmbH

Punch card



Copyright © 2012, MATHEMA Software GmbH

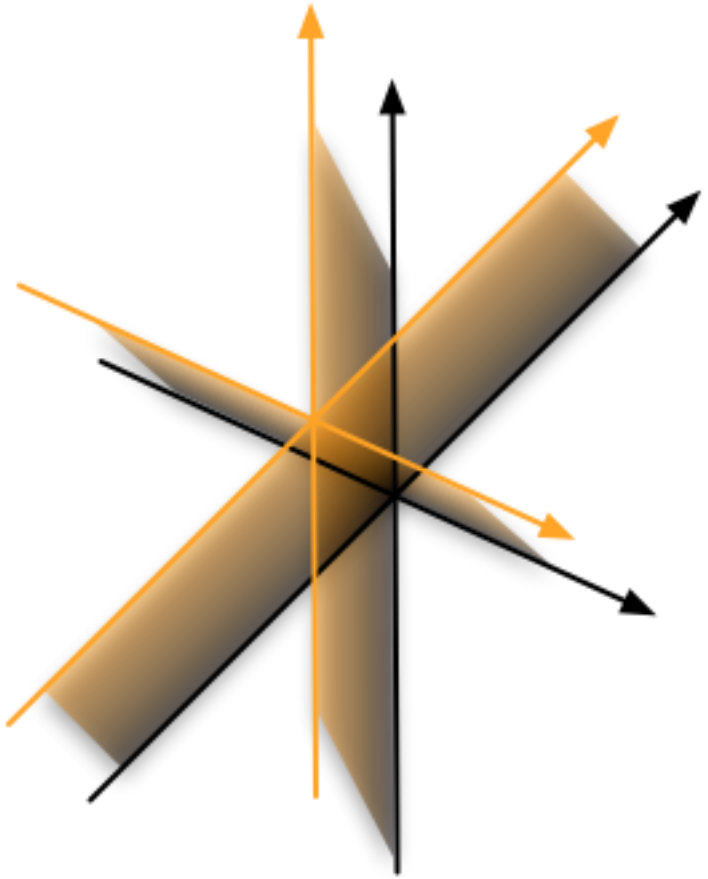
$$x = 2$$

Copyright © 2012, MATHEMA Software GmbH

$$x \neq 2$$

Copyright © 2012, MATHEMA Software GmbH

- Machine independence
- Problem independence
- Human independence
- Time independence



Daniel J. Salomon

Four Dimensions of Programming-Language Independence

ACM SIGPLAN Notices, Volume 27, No. 3, March 1992

Copyright © 2012, MATHEMA Software GmbH

Definition of Independence

A programming language can be said to be independent of a classification of the elements of a domain if it:

(1) supplies the same level of computational power to all groups in the classification, and

(2) meets the computational needs of each of the groups in the classification.

Copyright © 2012, MATHEMA Software GmbH

Architecture Independence

Applying the given definition of independence, an architecture-independent language would allow one to:

(1) Run any program written for one architecture on any other architecture.

(2) Take advantage of all the special features of any particular architecture.

Copyright © 2012, MATHEMA Software GmbH

A short introduction to the

AL_X Language Family

Copyright © 2012, MATHEMA Software GmbH

Unified Syntax Elements

- Command Statement

  keyword *token-list*

- Block Statement

  keyword [*token-list* **begin-keyword**]$^?$

  …

  end [**keyword** *optional-tokens*]$^?$

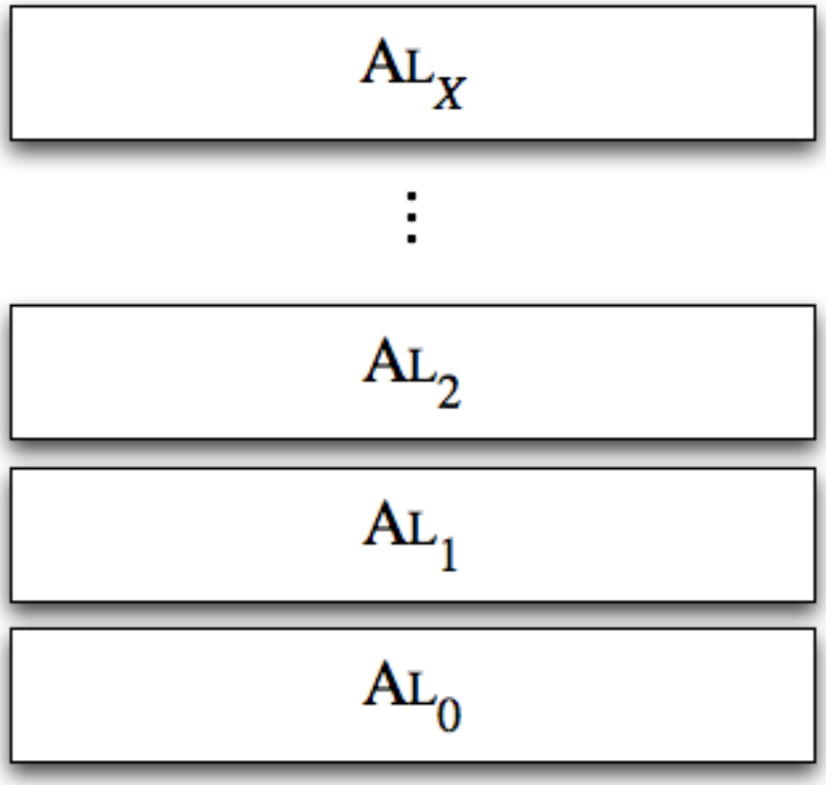
- Expression (Statement)
  - Assignment $x \leftarrow \ldots$; $x :\leftarrow \ldots$; $x \twoheadleftarrow \ldots$; $x ::= \ldots$; …
  - Procedure call p(…);
  - Everything else $x + y$; $x < y < z$; x^y; $x[y]$; $\lfloor x \rfloor$; …

Copyright © 2012, MATHEMA Software GmbH

```
namespace //mathema.de/example/Example

import java://java/lang/String

function example(x : ℤ, y : ℤ) → ℤ is

    z :← add(x, y)

    if greater(x, y) then
        z ← subtract(z, y)
    end

    return z

end function example
```


Copyright © 2012, MATHEMA Software GmbH

Language Layers



Copyright © 2012, MATHEMA Software GmbH

AL_x-Levels

- AL_0

  random memory access (C, LLVM, …)

- AL_1

  managed memory (JVM, CLR, …)

- AL_2

  control structures, records (Pascal, C, …)

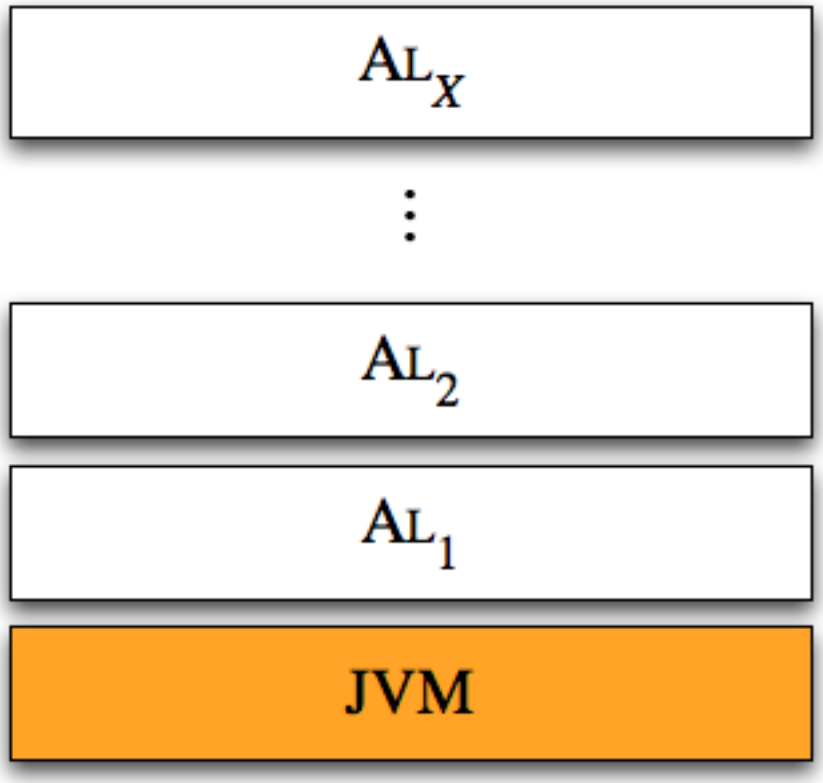
- AL_3

  objects (Java, C#, C++, …)

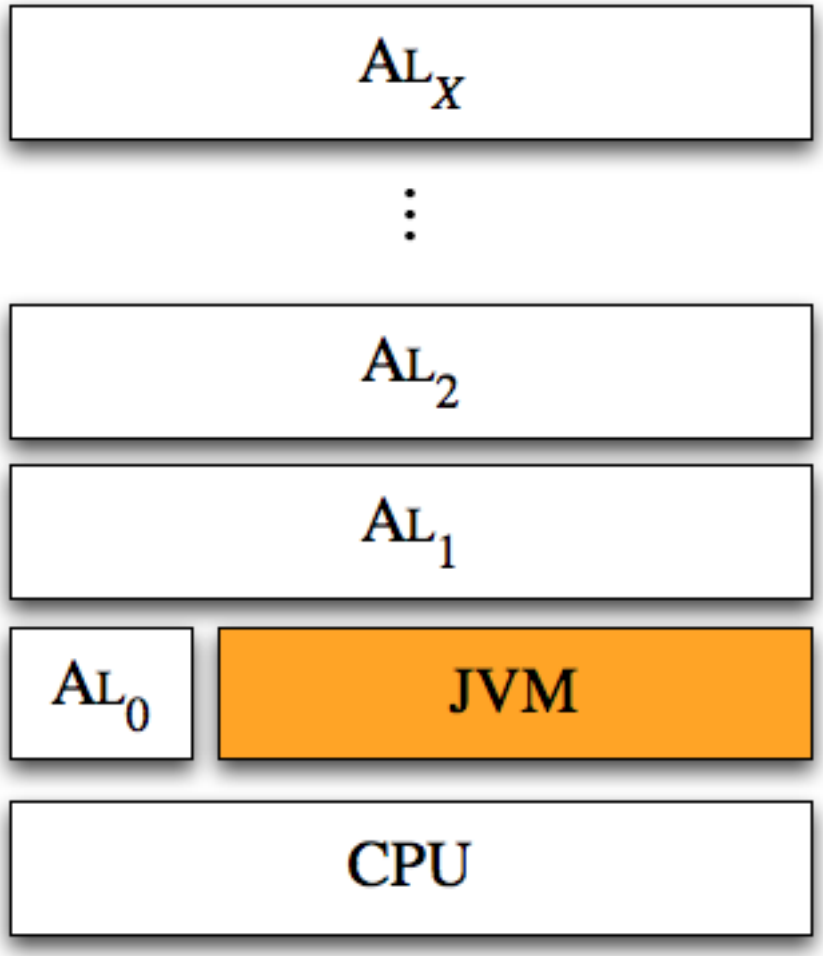
- $AL_?$

  everything else (Fortress, X10, …)

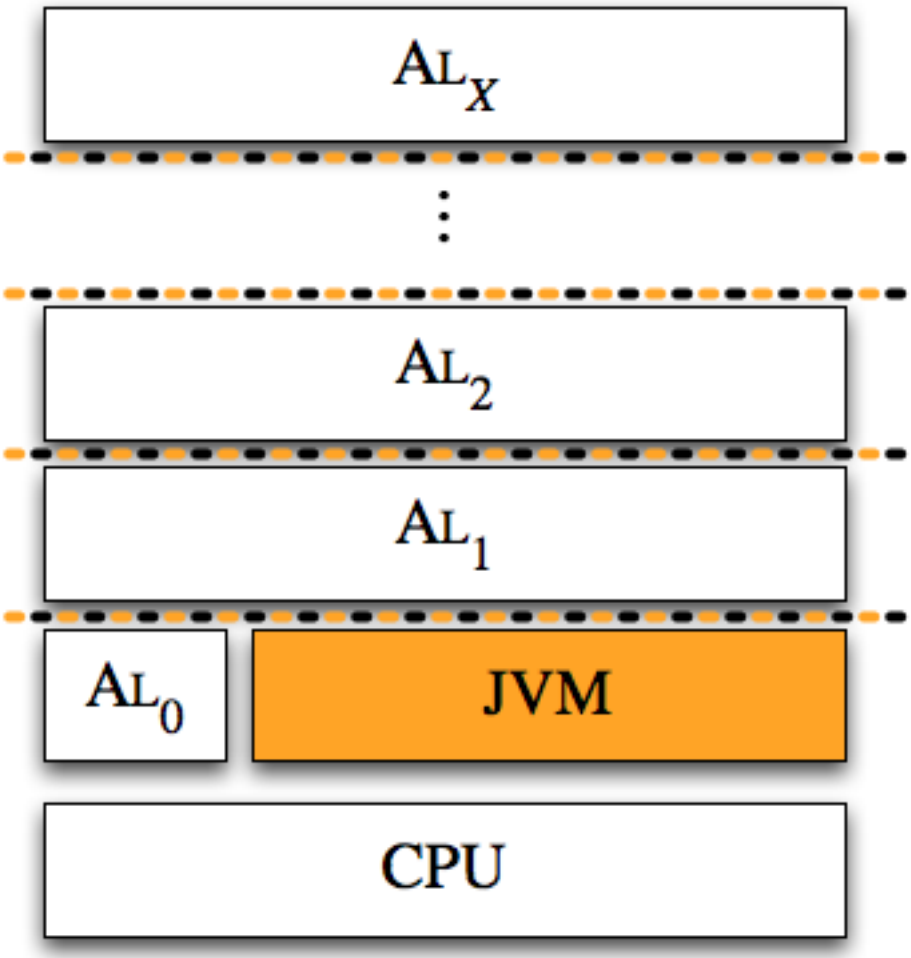
Copyright © 2012, MATHEMA Software GmbH



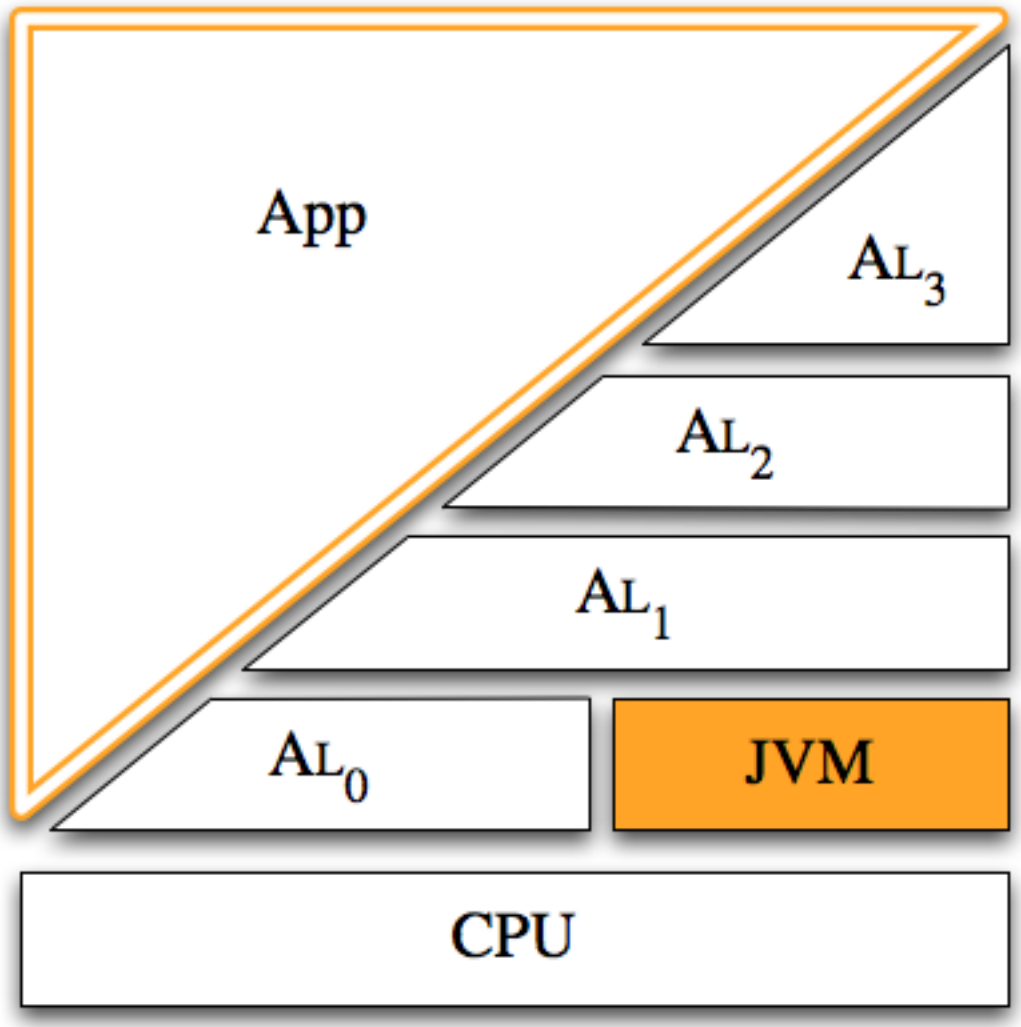
Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH

Source File Format

Structured file format (like XML) containing

- Supported AL_X version
- UTF-8 encoding, supported Unicode version, and reference to Private Use Area
- Compiler options, digital signature, …
- Owner, copyright, license, …
- Internal File System
  - Version history, change log, …
  - Images, Resources, …

Copyright © 2012, MATHEMA Software GmbH

How the JVM works

Copyright © 2012, MATHEMA Software GmbH

```
class Simple {
    int half(int x) {
        return x / 2;
    }
    int f(int x, int y) {
        int r = x + y;
        r = half(r);
        if (r < 50) {
            return 50;
        }
        return r;
    }
}
```

```
public static void main(String[] a) {
    Simple s = new Simple();
    System.out.println(s.f(37, 44));
}
```


Copyright © 2012, MATHEMA Software GmbH

```
int half(int x) {
    return x / 2;
}
```

```
int half(int);

  0:   iload_1
  1:   iconst_2
  2:   idiv
  3:   ireturn
```


Copyright © 2012, MATHEMA Software GmbH

```
int f(int x, int y) {
    int r = x + y;
    r = half(x);
    if (r < 50) {
        return 50;
    }
    return r;
}
```

```
int f(int, int);

  0:  iload_1
  1:  iload_2
  2:  iadd
  3:  istore_3
  4:  aload_0
  5:  iload_3
  6:  invokevirtual  #20; //Method half:(I)I
  9:  istore_3
 10:  iload_3
 11:  bipush 50
 13:  if_icmpge  19
 16:  bipush 50
 18:  ireturn
 19:  iload_3
 20:  ireturn
```


Copyright © 2012, MATHEMA Software GmbH

```
public static void main(java.lang.String[]);

    0:   new     #1; //class Simple
    3:   dup
    4:   invokespecial  #27; //Method "<init>":()V
    7:   astore_1
    8:   getstatic    #28; //Field java/lang/System.out:Ljava/io/PrintStream;
   11:   aload_1
   12:   bipush 37
   14:   bipush 44
   16:   invokevirtual  #34; //Method f:(II)I
   19:   invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
   22:   return
```

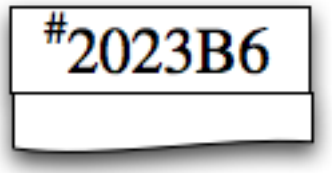

Copyright © 2012, MATHEMA Software GmbH

public static void main(java.lang.String[]);

```
 0:  new     #1; //class Simple
 3:  dup
 4:  invokespecial  #27; //Method "<init>":()V
 7:  astore_1
 8:  getstatic     #28; //Field java/lang/System.out:Ljava/io/PrintStream;
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
22:  return
```

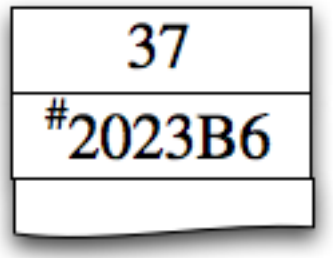

Copyright © 2012, MATHEMA Software GmbH

```
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
```



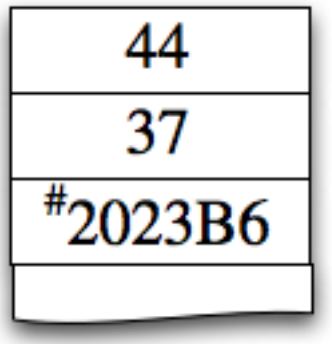
Copyright © 2012, MATHEMA Software GmbH

```
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
```



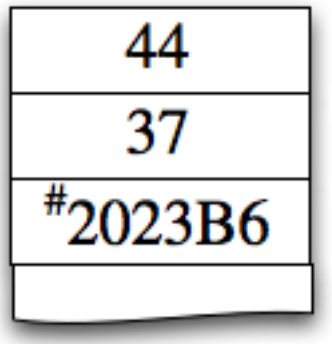
Copyright © 2012, MATHEMA Software GmbH

```
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
```



Copyright © 2012, MATHEMA Software GmbH

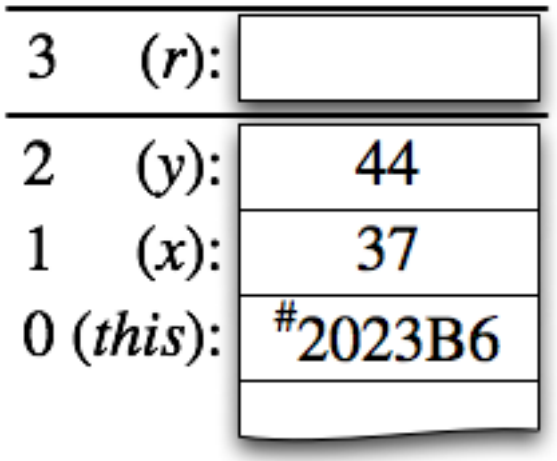
```
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

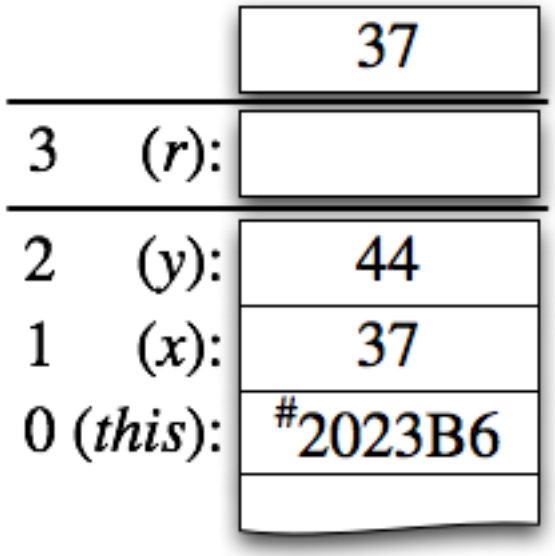
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

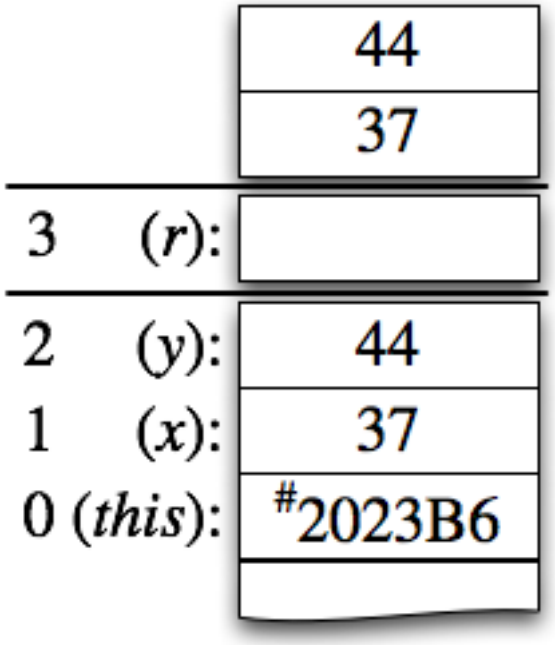
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

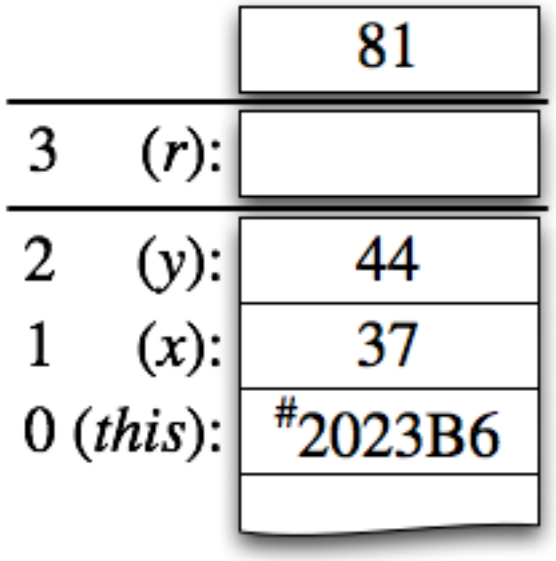
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

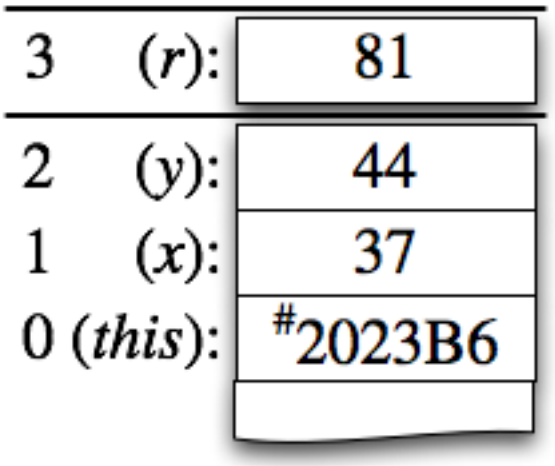
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

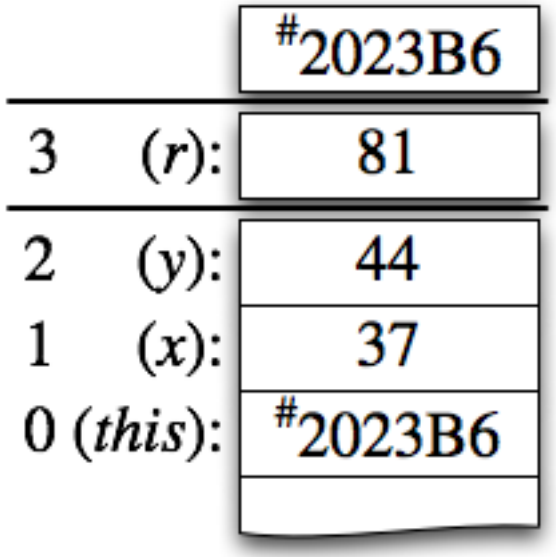
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

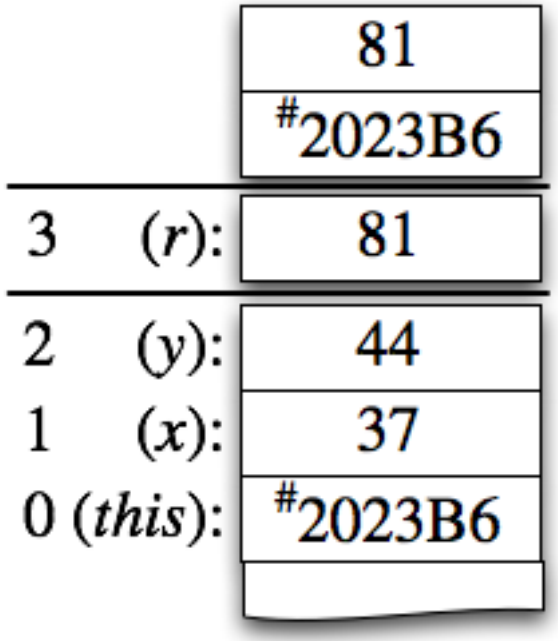
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

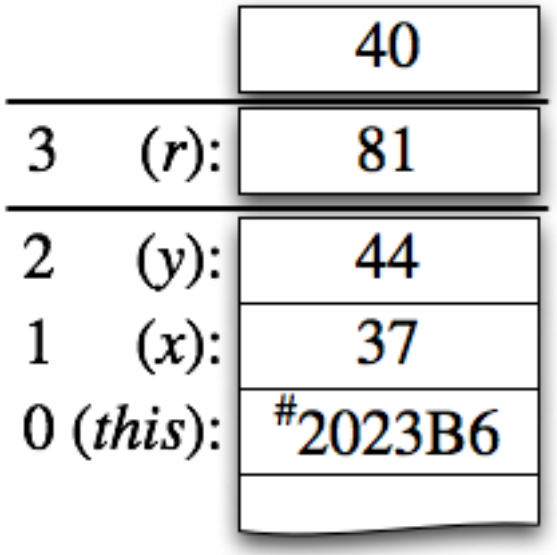
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

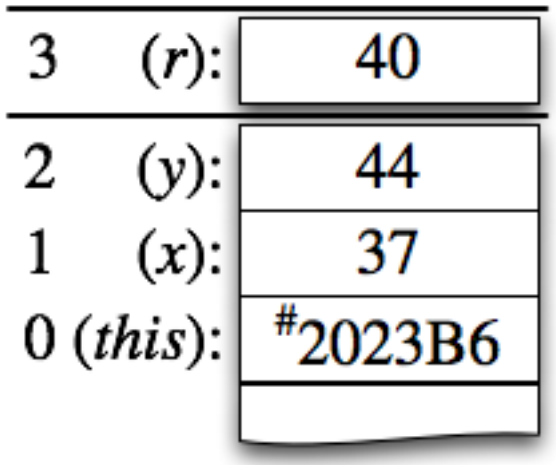
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

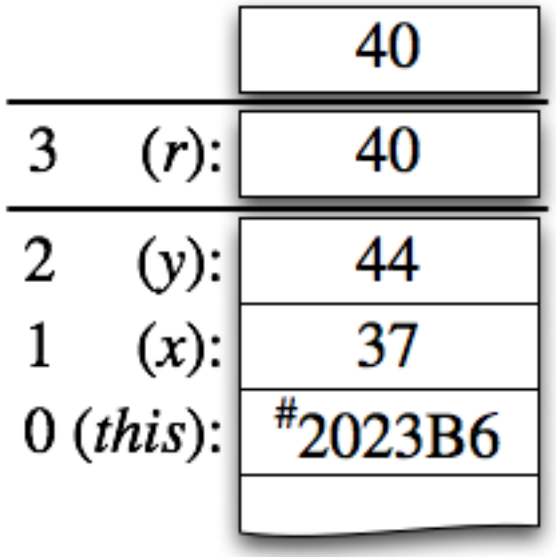
```
 0:   iload_1
 1:   iload_2
 2:   iadd
 3:   istore_3
 4:   aload_0
 5:   iload_3
 6:   invokevirtual  #20;
          //Method half:(I)I
 9:   istore_3
10:   …
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

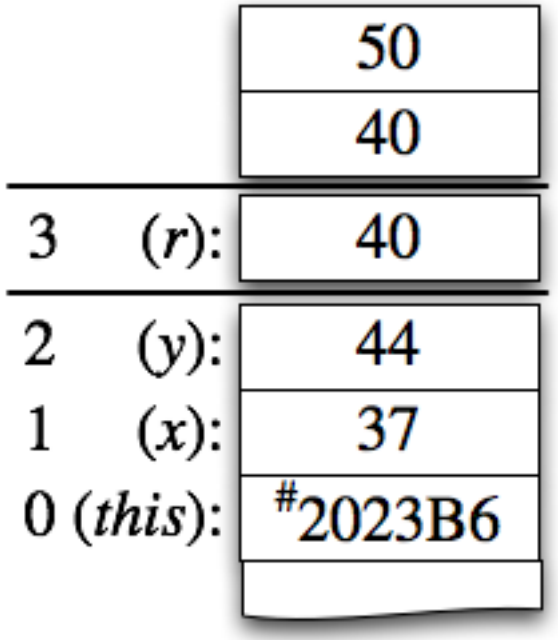
```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

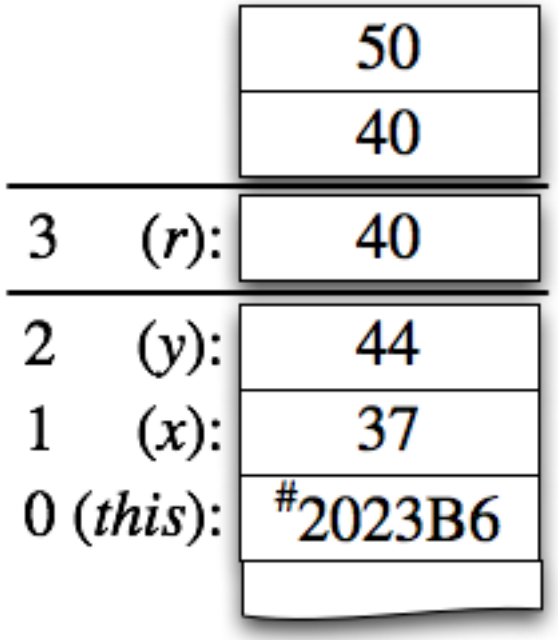
```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

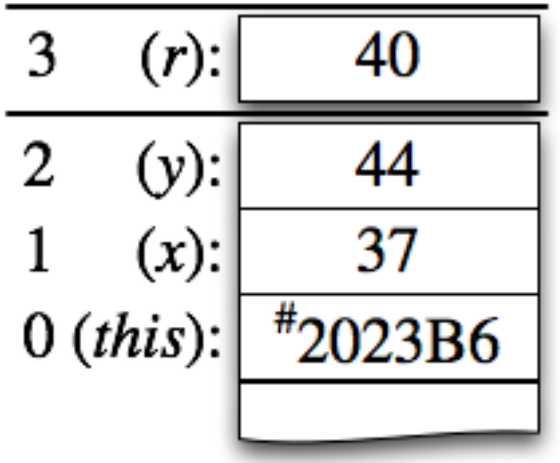
```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

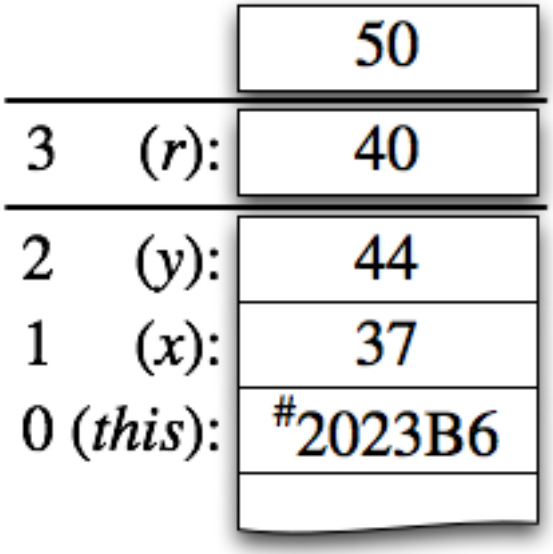
```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```



Copyright © 2012, MATHEMA Software GmbH

int f(int, int);

```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```



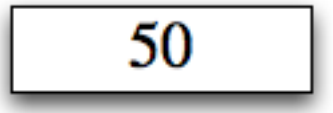
Copyright © 2012, MATHEMA Software GmbH

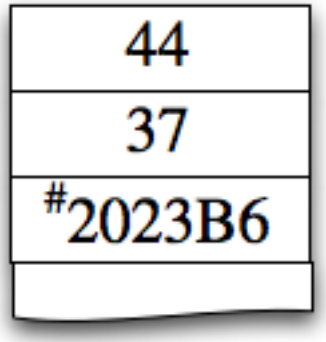
int f(int, int);

```
 6:  …
 9:  istore_3
10:  iload_3
11:  bipush 50
13:  if_icmpge 19
16:  bipush 50
18:  ireturn
19:  iload_3
20:  ireturn
```

Copyright © 2012, MATHEMA Software GmbH

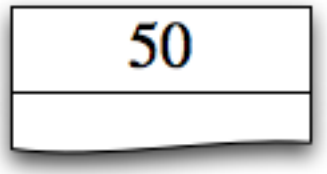
```
11:  aload_1
12:  bipush 37
14:  bipush 44
16:  invokevirtual  #34; //Method f:(II)I
19:  invokevirtual  #36; //Method java/io/PrintStrea
```





Copyright © 2012, MATHEMA Software GmbH

```
11:   aload_1
12:   bipush 37
14:   bipush 44
16:   invokevirtual  #34; //Method f:(II)I
19:   invokevirtual  #36; //Method java/io/PrintStream.println:(I)V
```

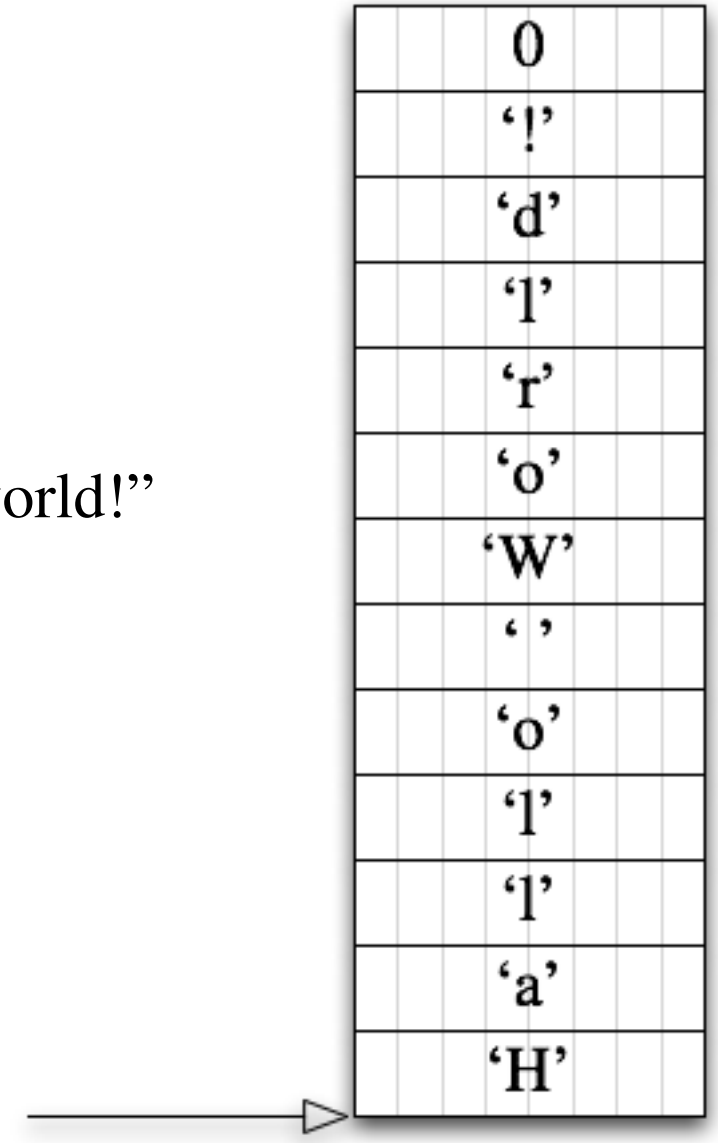


Copyright © 2012, MATHEMA Software GmbH

A short Introduction to AL$_1$

Copyright © 2012, MATHEMA Software GmbH

```
namespace //mathema.de/hello/Hello

import //alx/compatibility/c/Stdio/print

[[read_only]]
data DataSegment is
    greetings : Sq[[Byte, 20]] ← "Hello, world!"
end data

procedure main() is

    s⃗ :← ⊕greetings
    print(s⃗)

end callable
```



Copyright © 2012, MATHEMA Software GmbH

data DataSegment **is**

a_0 : Byte ← $(01)_{16}$

a_1 : Byte ← $(23)_{16}$

a_2 : Byte ← $(45)_{16}$

a_3 : Byte ← $(67)_{16}$

end data

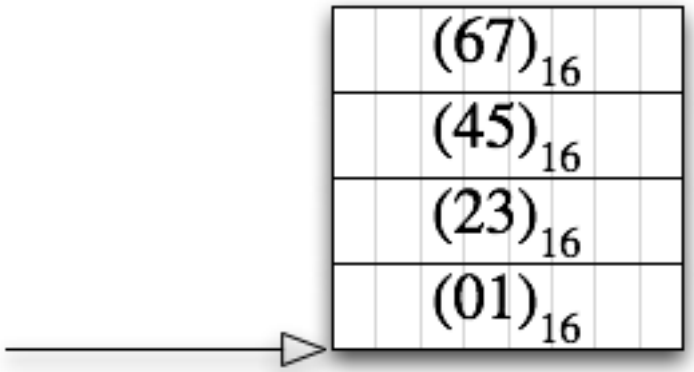
procedure main() **is**

t :← load [[Tetra]] ($\oplus a_0$)

¶ *little-endian:* $t = (67\ 45\ 23\ 01)_{16}$

¶ *big-endian:* $t = (01\ 23\ 45\ 67)_{16}$

end callable



Copyright © 2012, MATHEMA Software GmbH

assume *config.byte_order* = BIG_ENDIAN

data DataSegment **is**

$a_0 : \text{Byte} \leftarrow (01)_{16}$

$a_1 : \text{Byte} \leftarrow (23)_{16}$

$a_2 : \text{Byte} \leftarrow (45)_{16}$

$a_3 : \text{Byte} \leftarrow (67)_{16}$

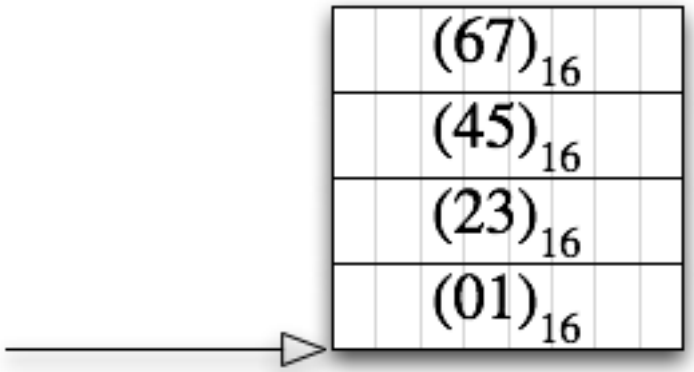
end data

procedure main() **is**

$t :\leftarrow \text{load}[\![\text{Tetra}]\!](\oplus a_0)$

$\P\ t = (01234567)_{16}$

end callable



Copyright © 2012, MATHEMA Software GmbH

Find First 0-Byte in Sequence …

[[inline]]

function markZeroBytes($x : \mathbb{M}$) $\rightarrow \mathbb{M}$ **is**

$\mathrm{M} = (7\mathrm{F}\ldots7\mathrm{F})_{16}$

$r :\leftarrow \mathrm{and}(x, \mathrm{M})$

$r \leftarrow \mathrm{addUnsigned}(r, \mathrm{M})$

$r \leftarrow \mathrm{or}(r, x)$

$r \leftarrow \mathrm{or}(r, \mathrm{M})$

$r \leftarrow \mathrm{not}(r)$

return r

end callable

Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

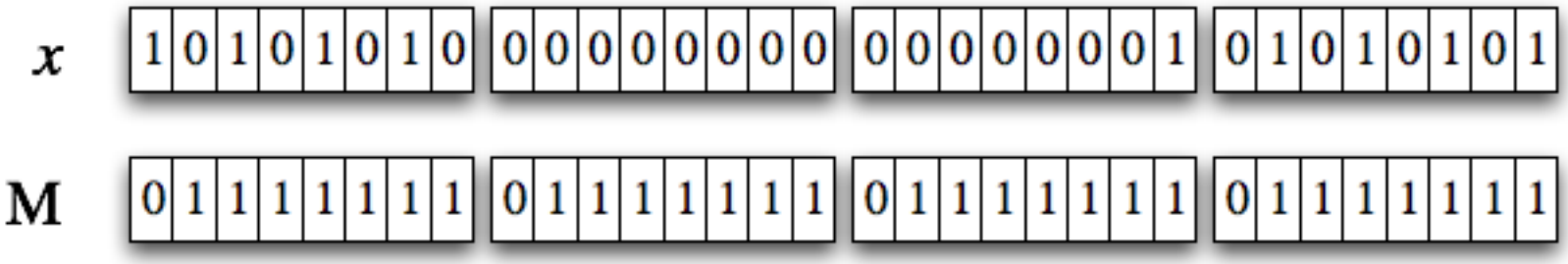
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

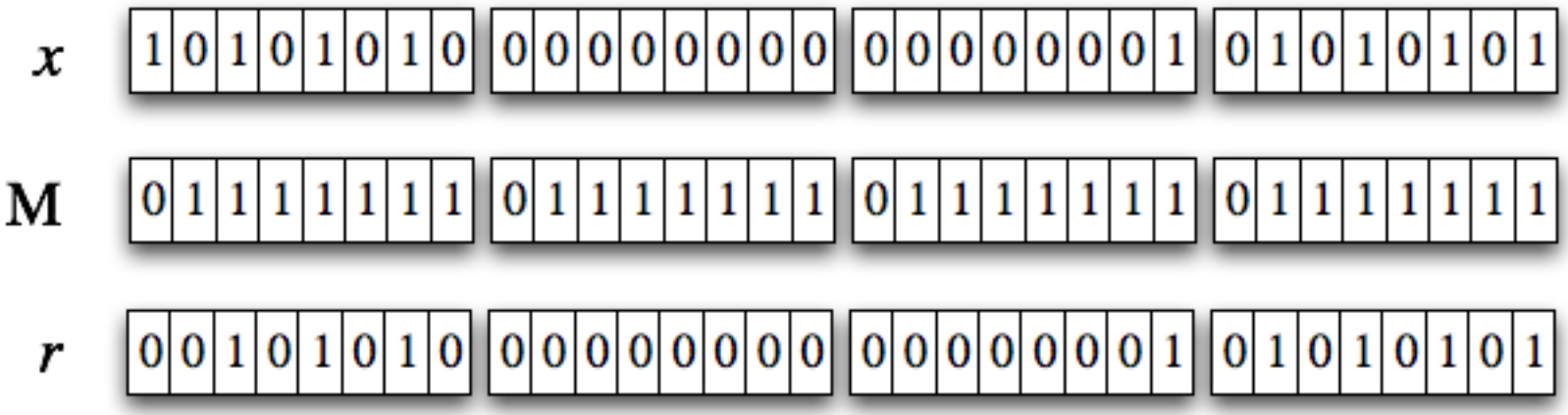
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

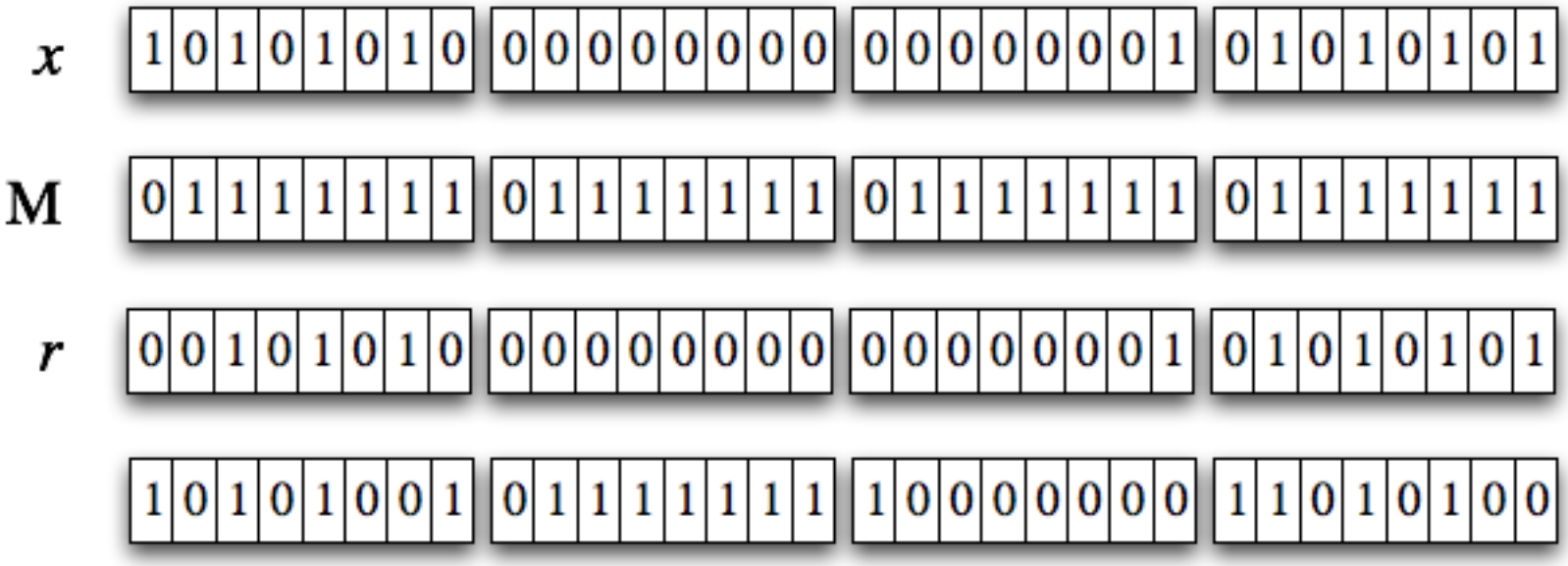
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

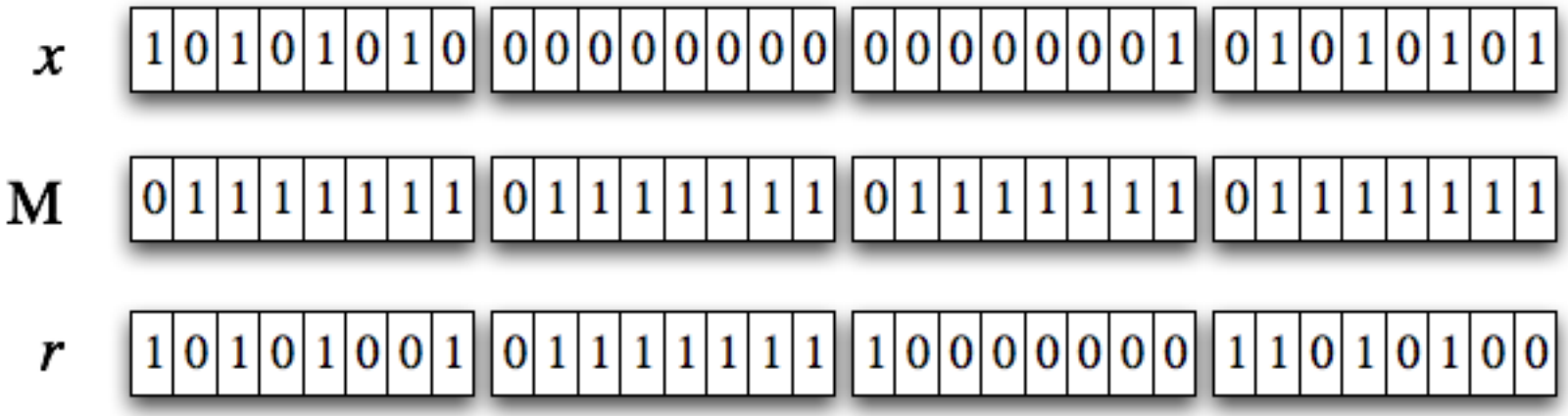
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

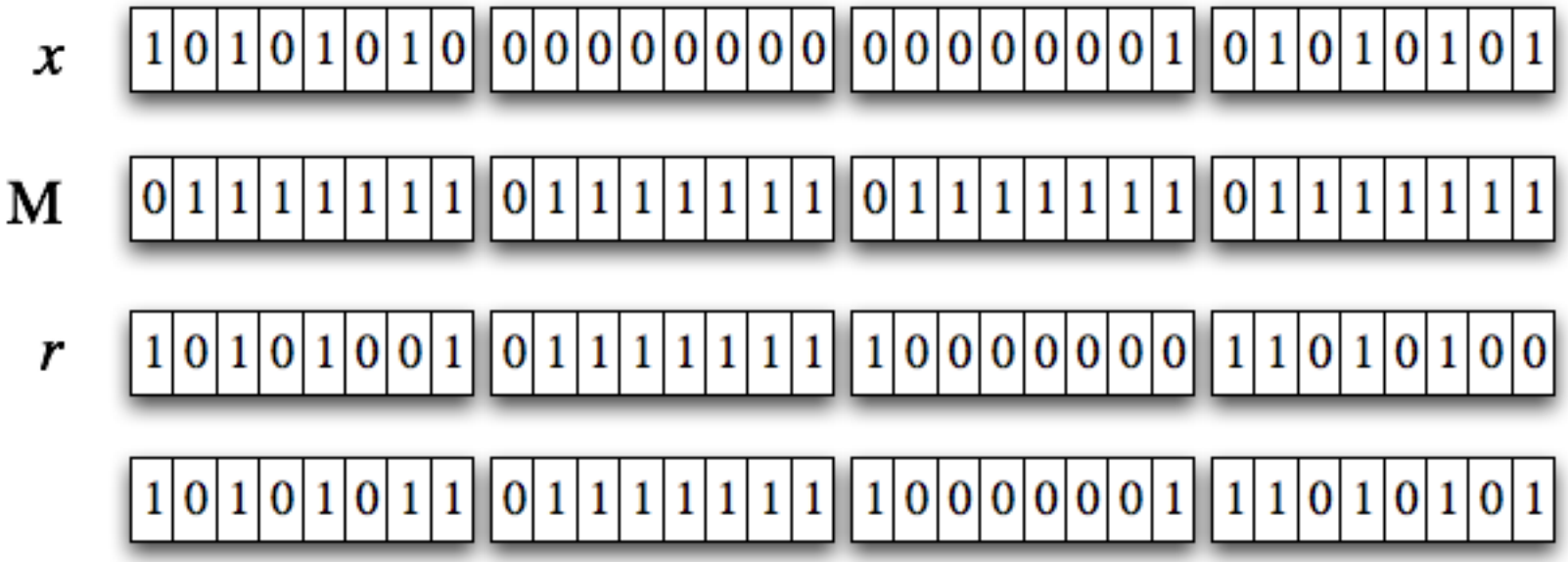
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

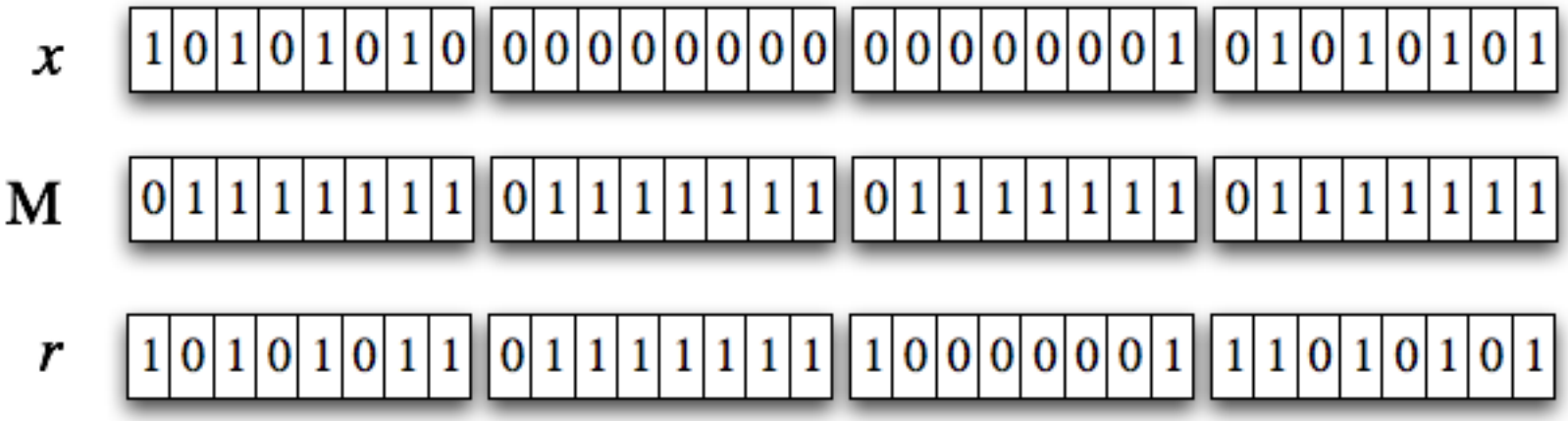
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$

Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

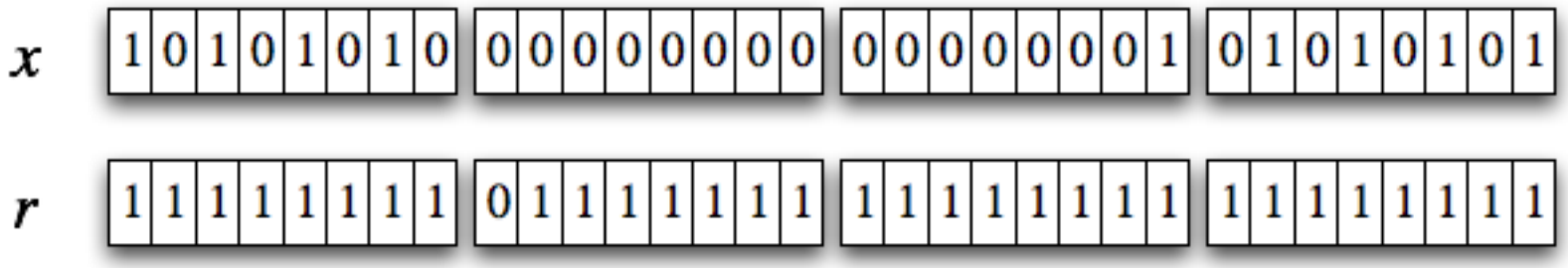
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

$r \leftarrow \text{not}(r)$



Copyright © 2012, MATHEMA Software GmbH

$M = (7F\ldots7F)_{16}$

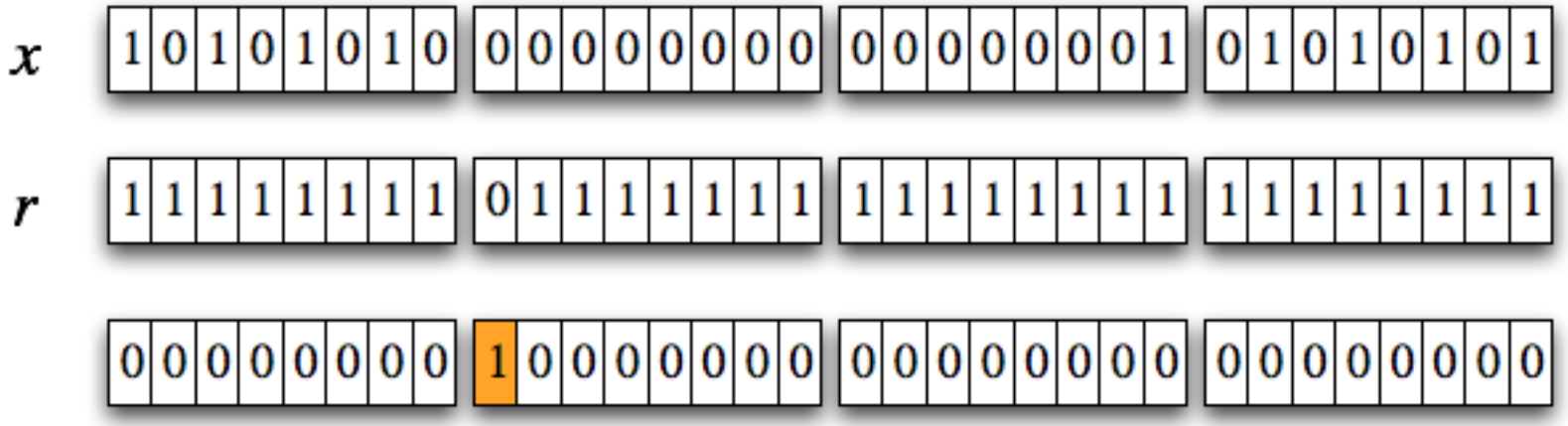
$r :\leftarrow \text{and}(x, M)$

$r \leftarrow \text{addUnsigned}(r, M)$

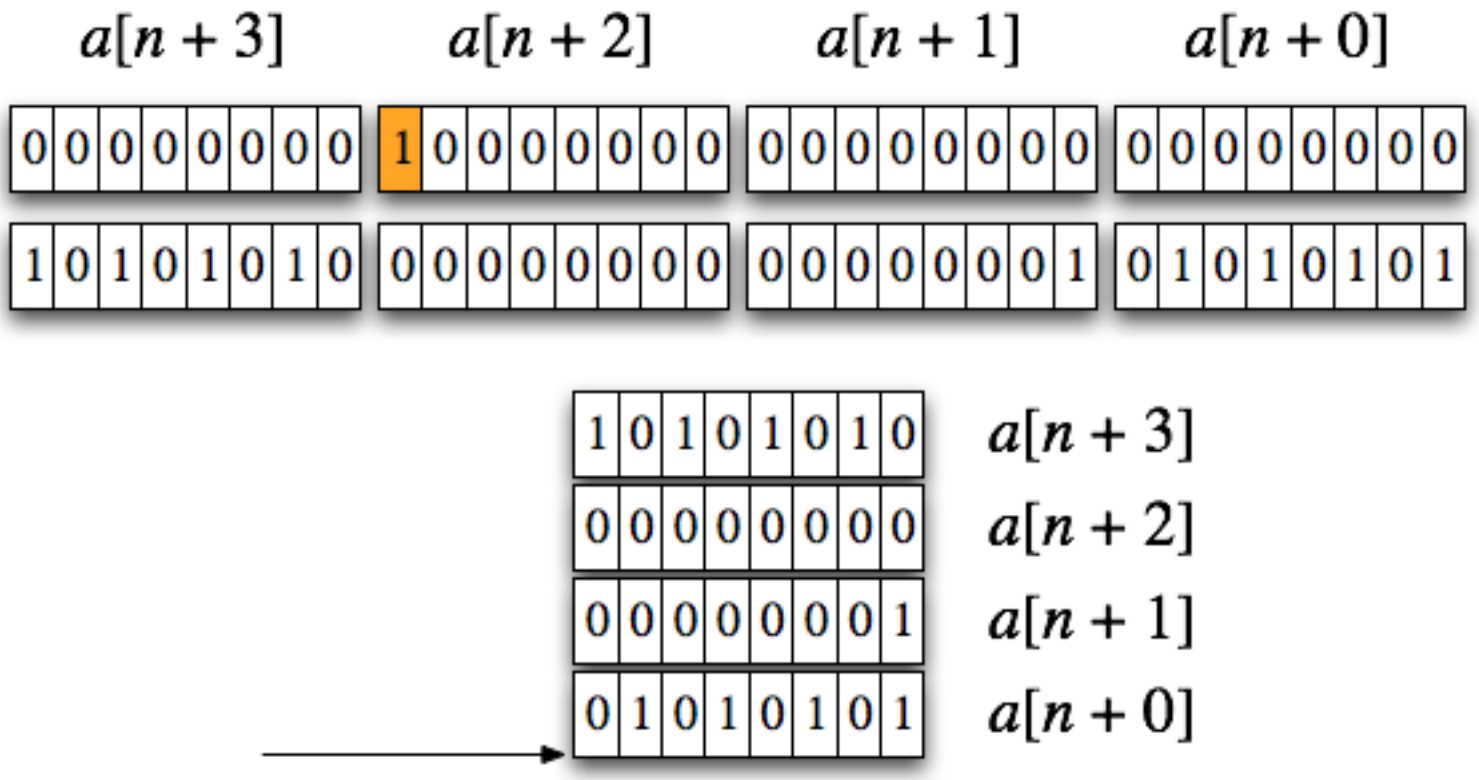
$r \leftarrow \text{or}(r, x)$

$r \leftarrow \text{or}(r, M)$

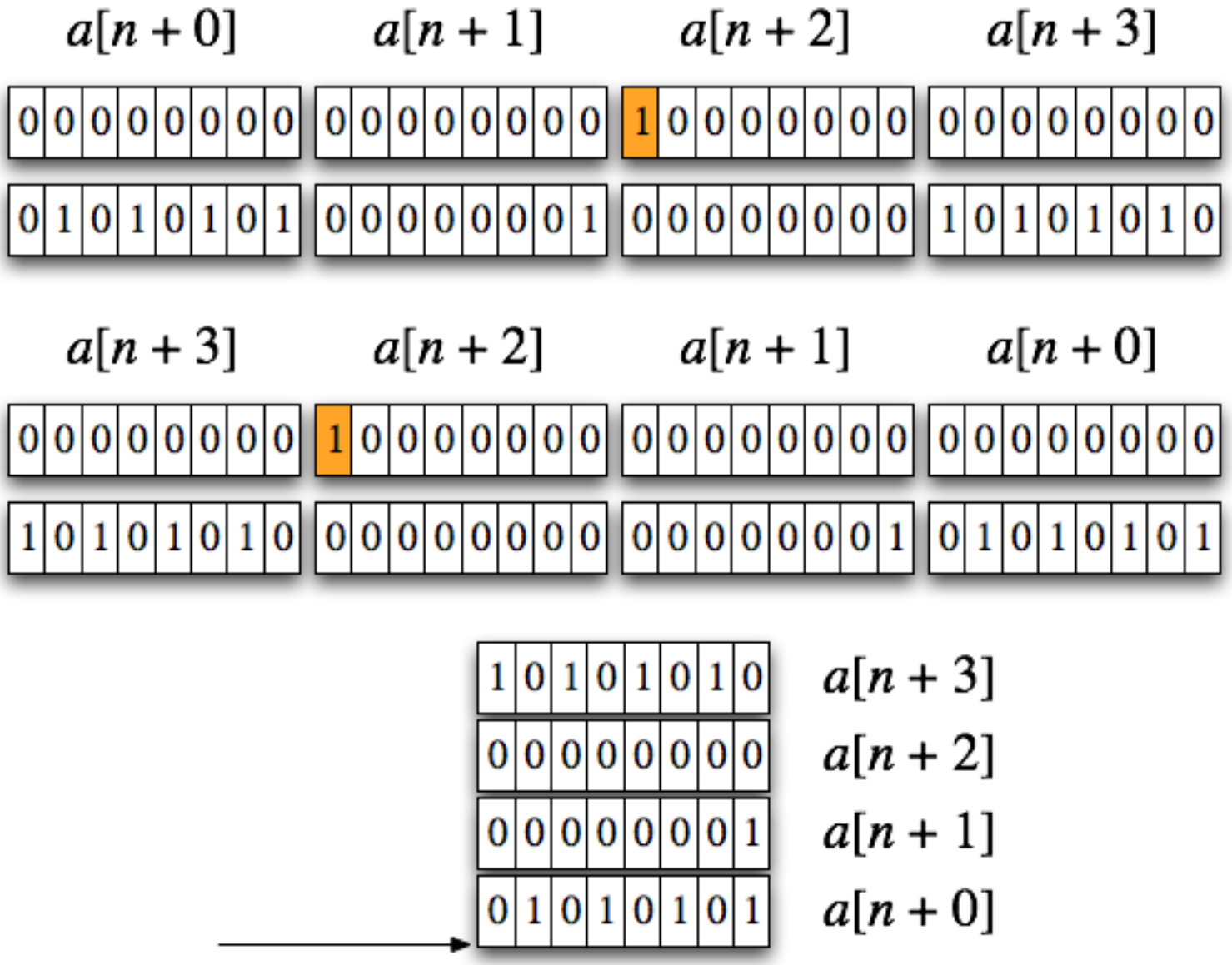
$r \leftarrow \text{not}(r)$



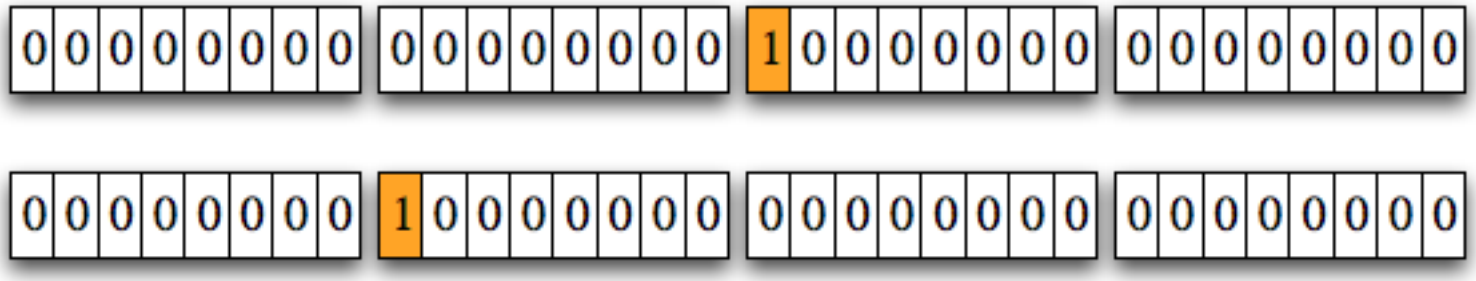
Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH

Find First 0-Byte in Sequence (cont'd) …

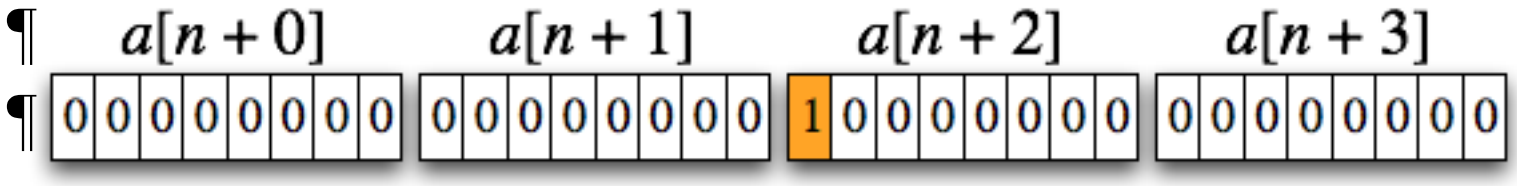
```
[[inline]]
function findZeroByte(x : 𝕄) → 𝕄 is
    r :← markZeroBytes(x)
    variant
    ■ "big endian" …
        …
    ■ "little endian" …
        …
    end variant
    return r
end callable
```


Copyright © 2012, MATHEMA Software GmbH

Find First 0-Byte in Sequence (cont'd)

variant ¶ Reminder: zero found at *a*[2]

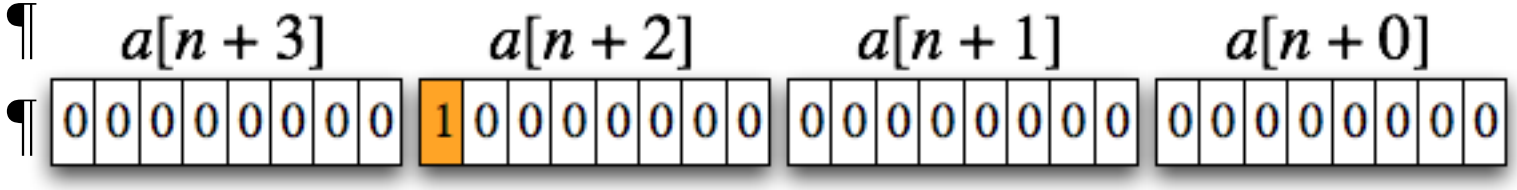
■ "big endian" **requires** {*config*.*byte_order* = BIG_ENDIAN}:



nlz :← numberOfLeadingZeroBits(*r*)

r ← shiftRightUnsigned(*nlz*, 3) ¶ *nlz* / 8

■ "little endian" **requires** {*config*.*byte_order* = LITTLE_ENDIAN}:



ntz :← numberOfTrailingZeroBits(*r*)

r ← shiftRightUnsigned(*ntz*, 3) ¶ *ntz* / 8

end variant

Copyright © 2012, MATHEMA Software GmbH

Types

Copyright © 2012, MATHEMA Software GmbH

- $\mathbb{M}_p$ (*Machine*) *p*-Byte word
- $\mathbb{I}_n$ (*Integer*) signed integer numbers
  $\{-2^{n-1}, \ldots, 2^{n-1} - 1\}$
- $\mathbb{U}_n$ (*Unsigned*) unsigned integer numbers
  $\{0, \ldots, 2^n - 1\}$
- $\mathbb{J}_n$ (*Jumble*) numbers fitting in both $\mathbb{I}_n$ and $\mathbb{U}_n$
  $\{0, \ldots, 2^{n-1} - 1\}$
- $\mathbb{F}_n$ (*Float*) floating point numbers (base 2)
- $\mathbb{D}_n$ (*Decimal*) decimal numbers (base 10)
- $\mathbb{B}$ (*Boolean*) boolean values FALSE (0) and TRUE (1)

Copyright © 2012, MATHEMA Software GmbH

Machine Word $\mathbb{M}_p$

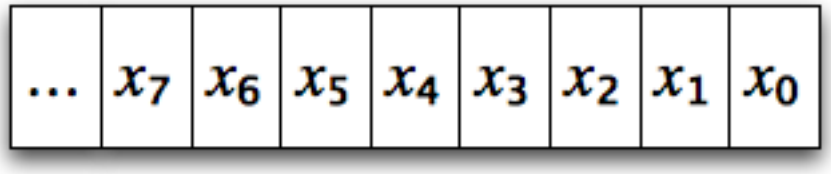
- $\mathbb{M}_p$ $p = 2^k$

- $\mathbb{M}$ *preferred machine word*
- $\mathbb{M}_1$ Byte
- $\mathbb{M}_2$ Wyde
- $\mathbb{M}_4$ Tetra
- $\mathbb{M}_8$ Octa
- $\mathbb{M}_{16}$ Hexadec
- $\mathbb{M}_{32}$ *Doxadec*
- $\mathbb{M}_{64}$ *Texadec*
- $\mathbb{M}_{128}$ *Oxadec*
- $\mathbb{M}_{256}$ *Hexadex*
- $\mathbb{M}_{512}$ *Doxadex*
- $\mathbb{M}_{1024}$ *Texadex*
- $\mathbb{M}_{2048}$ *Oxadex*

Copyright © 2012, MATHEMA Software GmbH

Machine Word $\mathbb{M}_p$

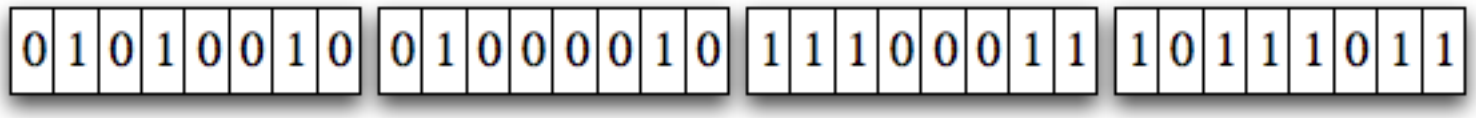
- [[const]] *width* : $\mathbb{N}$[Bit] $= 8 \cdot p$ Bit
- [[const]] *size* : $\mathbb{N}$[Byte] $= p$ Byte
- [[const]] *element_size* : $\mathbb{N}$[Byte] $= p$ Byte
- [[const]] *alignment* : $\mathbb{N}$ $= p$



- *signed* : $\mathbb{I}_{n=8p}$ $= -2^{n-1} \cdot x_k + \sum[0 \le k < n-1]\ 2^k \cdot x_k$
- *unsigned* : $\mathbb{U}_{n=8p}$ $= \sum[0 \le k < n]\ 2^k \cdot x_k$

Copyright © 2012, MATHEMA Software GmbH

- [[const]] *width* : ℕ[Bit] = 32 Bit
- [[const]] *size* : ℕ[Byte] = 4 Byte
- [[const]] *element_size* : ℕ[Byte] = 4 Byte
- [[const]] *alignment* : ℕ = 4



- *signed* : $\mathbb{I}_{32}$ $[-2^{31}; 2^{31}[$
- *unsigned* : $\mathbb{U}_{32}$ $[0; 2^{31} - 1[$

Copyright © 2012, MATHEMA Software GmbH

Machine Word Access $\mathbb{M}_p$

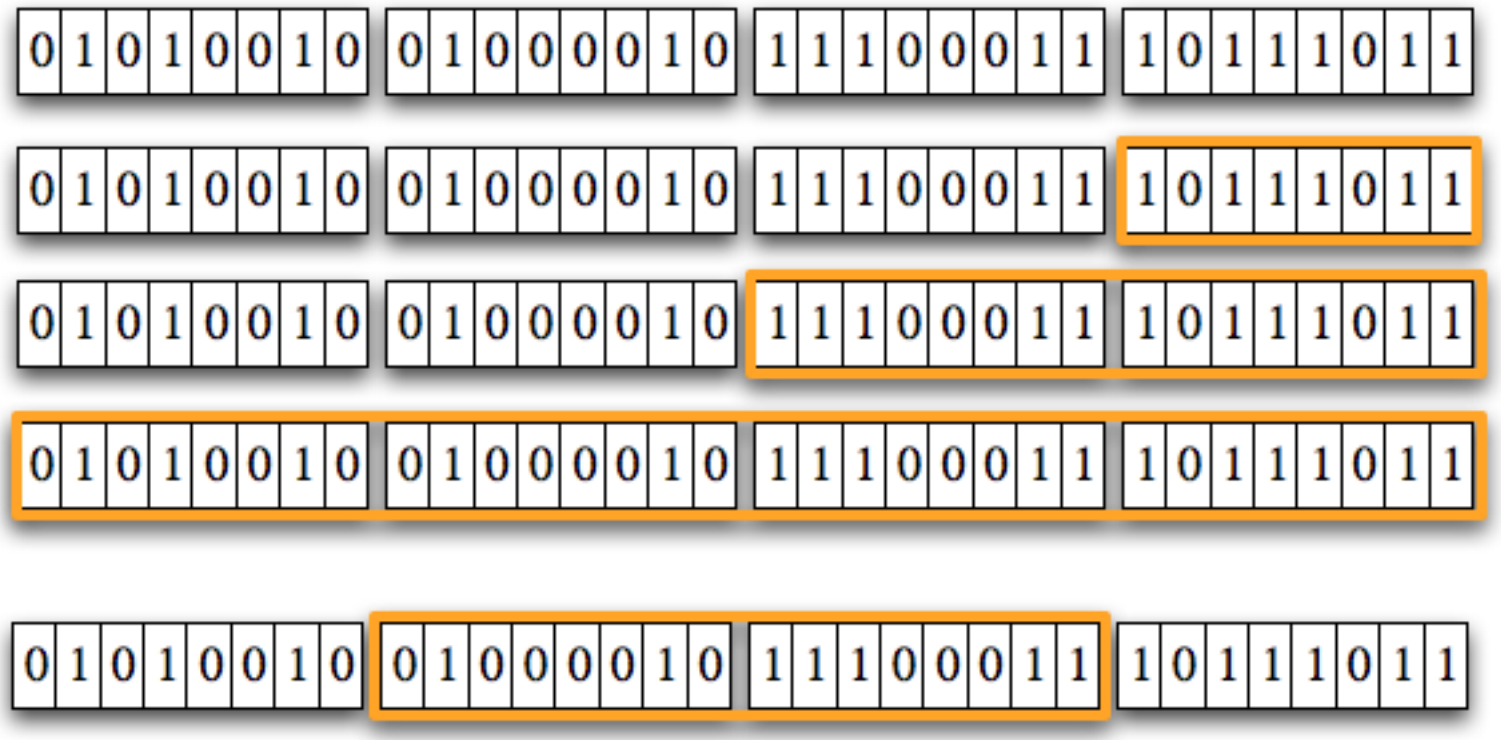
- *high* : $\mathbb{M}_{p/2}$
  low : $\mathbb{M}_{p/2}$
- *bytes* : Sq⟦Byte, *p*⟧
  big_endian_bytes : Sq⟦Byte, *p*⟧
  little_endian_bytes : Sq⟦Byte, *p*⟧
- *wydes* : Sq⟦Wyde, *p*/2⟧
  big_endian_wydes : Sq⟦Wyde, *p*/2⟧
  little_endian_wydes : Sq⟦Wyde, *p*/2⟧
- *tetras* : Sq⟦Tetra, *p*/4⟧
  big_endian_tetras : Sq⟦Tetra, *p*/4⟧
  little_endian_tetras : Sq⟦Tetra, *p*/4⟧
- *bits* : BitSq⟦8*p*⟧
  big_endian_bits : BitSq⟦8*p*⟧
  little_endian_bits : BitSq⟦8*p*⟧

Copyright © 2012, MATHEMA Software GmbH

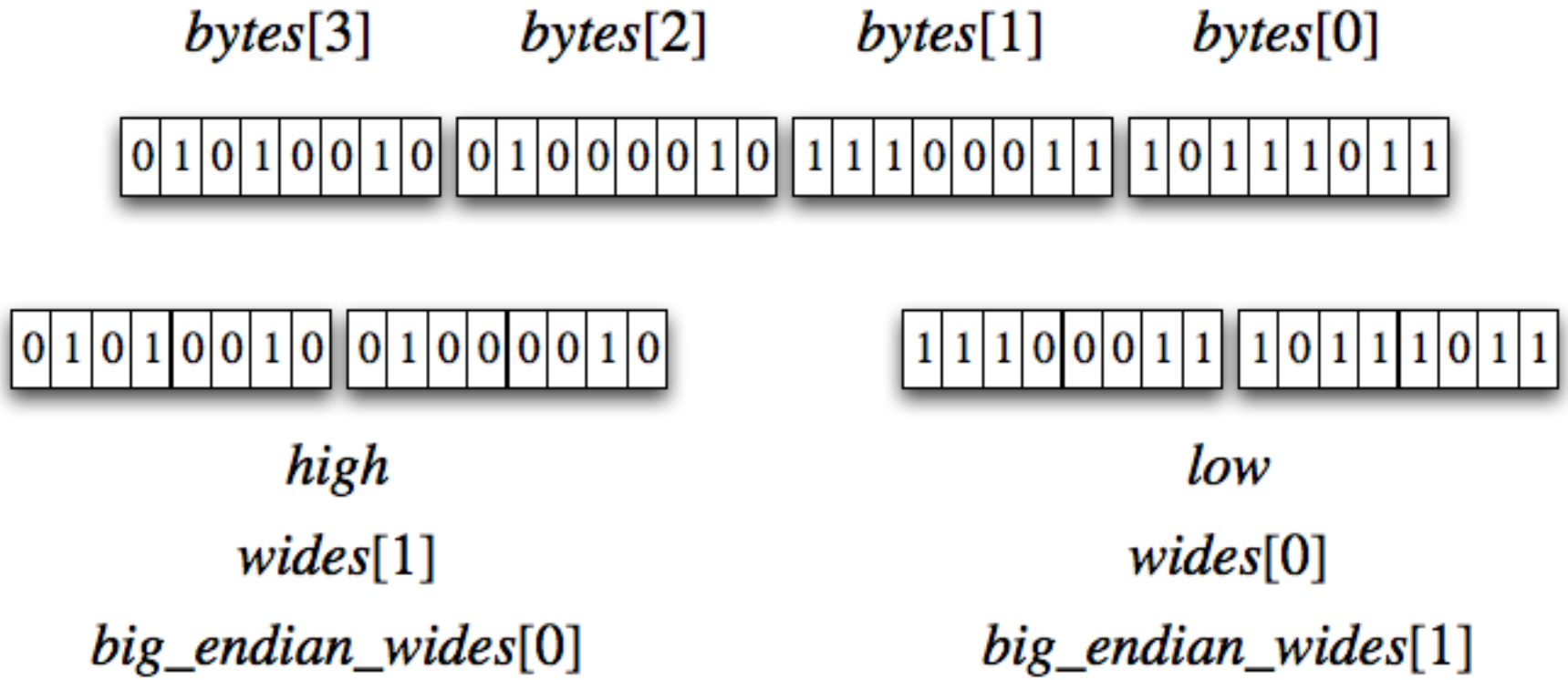
- *high* : Wyde
  low : Wyde
- *bits* : BitSq⟦32⟧
  big_endian_bits : BitSq⟦32⟧
  little_endian_bits : BitSq⟦32⟧
- *bytes* : Sq⟦Byte, 4⟧
  big_endian_bytes : Sq⟦Byte, 4⟧
  big_endian_bytes : Sq⟦Byte, 4⟧
- *wydes* : Sq⟦Wyde, 2⟧
  big_endian_wydes : Sq⟦Wyde, 2⟧
  little_endian_wydes : Sq⟦Wyde, 2⟧

Copyright © 2012, MATHEMA Software GmbH

Accessing Parts of a Machine Word



Copyright © 2012, MATHEMA Software GmbH



$signed = 1\ 380\ 115\ 387$

$unsigned = 1\ 380\ 115\ 387$

$high.signed = 21\ 058$

$low.signed = -7\ 237$

Copyright © 2012, MATHEMA Software GmbH

Intrinsic Functions

Copyright © 2012, MATHEMA Software GmbH

- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{not}(x : \mathrm{T}) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{and}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{or}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{xor}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{andNot}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{orNot}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{nand}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{nor}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$ $\quad \mathrm{nxor}(x : \mathrm{T}, y : \mathbb{M}_p) \to \mathrm{T}$

Copyright © 2012, MATHEMA Software GmbH

Bit Counts

- numberOfLeadingZeroBits($x : \mathbb{M}_p$) → $\mathbb{J}\{0, \ldots, 8 \cdot p\}$
- numberOfTrailingZeroBits($x : \mathbb{M}_p$) → $\mathbb{J}\{0, \ldots, 8 \cdot p\}$
- numberOfOneBits($x : \mathbb{M}_p$) → $\mathbb{J}\{0, \ldots, 8 \cdot p\}$

- numberOfOneBits.*is_native*

Copyright © 2012, MATHEMA Software GmbH

- add($x : \mathbb{I}_n, y : \mathbb{I}_n$) $\rightarrow \mathbb{I}_n$ **throws** Overflow

Copyright © 2012, MATHEMA Software GmbH

Addition

- add($x : \mathbb{I}_n, y : \mathbb{I}_n$) $\rightarrow \mathbb{I}_n$ **throws** Overflow
- addUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n$) $\rightarrow \mathbb{U}_n$

Copyright © 2012, MATHEMA Software GmbH

Addition

- add($x : \mathbb{I}_n, y : \mathbb{I}_n$) $\rightarrow \mathbb{I}_n$ **throws** Overflow
- addUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n$) $\rightarrow \mathbb{U}_n$
- addWithCarry($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{J}_n\{0, 1\}$) $\rightarrow (\mathbb{I}_n, \mathbb{J}_n\{0, 1\})$

Copyright © 2012, MATHEMA Software GmbH

Addition

- add($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** Overflow
- addUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$
- addWithCarry($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{I}_n, \mathbb{J}_n\{0, 1\})$
- addWithCarryUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{U}_n, \mathbb{J}_n\{0, 1\})$

Copyright © 2012, MATHEMA Software GmbH

Addition

- add($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** Overflow
- addUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$
- addWithCarry($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{I}_n, \mathbb{J}_n\{0, 1\})$
- addWithCarryUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{U}_n, \mathbb{J}_n\{0, 1\})$

- addUnchecked($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$

Copyright © 2012, MATHEMA Software GmbH

Addition

- add($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** Overflow
- addUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$
- addWithCarry($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{I}_n, \mathbb{J}_n\{0, 1\})$
- addWithCarryUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{U}_n, \mathbb{J}_n\{0, 1\})$

- addUnchecked($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$
- addUnsignedChecked($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** Overflow

Copyright © 2012, MATHEMA Software GmbH

Subtraction

- subtract($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** Overflow
- subtractUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$
- subtractWithBorrow($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{I}_n, \mathbb{J}_n\{0, 1\})$
- subtractWithBorrowUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n, c : \mathbb{J}_n\{0, 1\}) \rightarrow (\mathbb{U}_n, \mathbb{J}_n\{0, 1\})$
- subtractUnchecked($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$
- subtractUnsignedChecked($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** Overflow

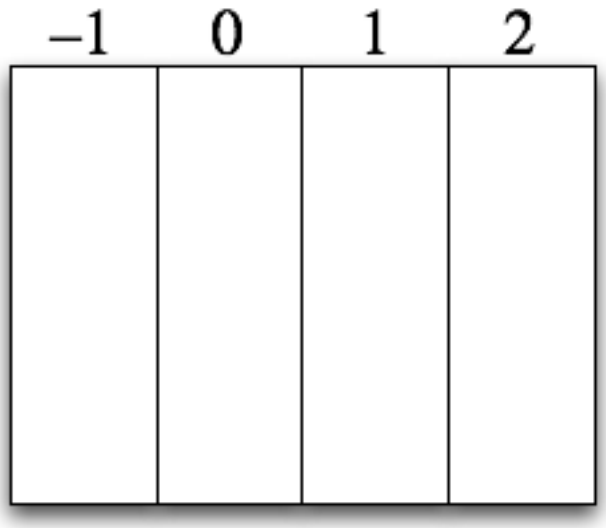
Copyright © 2012, MATHEMA Software GmbH

- $\mathrm{div}(x : \mathbb{I}_n, y : \mathbb{I}_n) \to \mathbb{I}_n$ **throws** {Overflow}
  $\mathrm{divUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n$
  - $x / 2^n$ should be identical to $x \gg n$
    (which is not true for Java et al. if $x < 0$)
  - Necessity for different division and modulo functions:

Name		***Function***	***Round toward***
div(x, y);	mod(x, y)	$\lfloor x / y \rfloor$	$-\infty$
ratio(x, y);	residue(x, y)	round(x / y)	nearest
group(x, y);	pad(x, y)	$\lceil x / y \rceil$	$+\infty$
quot(x, y);	rem(x, y)	$\mathrm{sign}(x \cdot y) \cdot \mathrm{div}(\lvert x \rvert, \lvert y \rvert)$	0

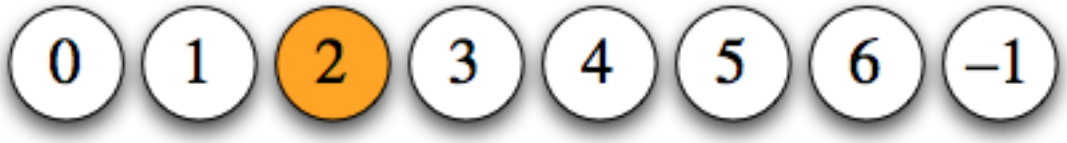
Copyright © 2012, MATHEMA Software GmbH

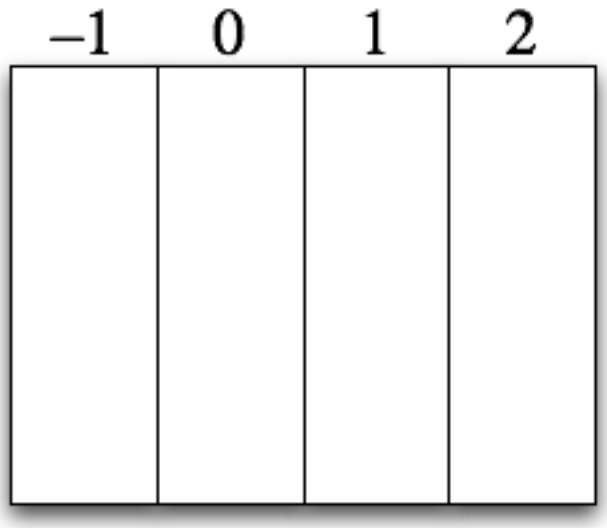
How To Group Into Groups



Copyright © 2012, MATHEMA Software GmbH

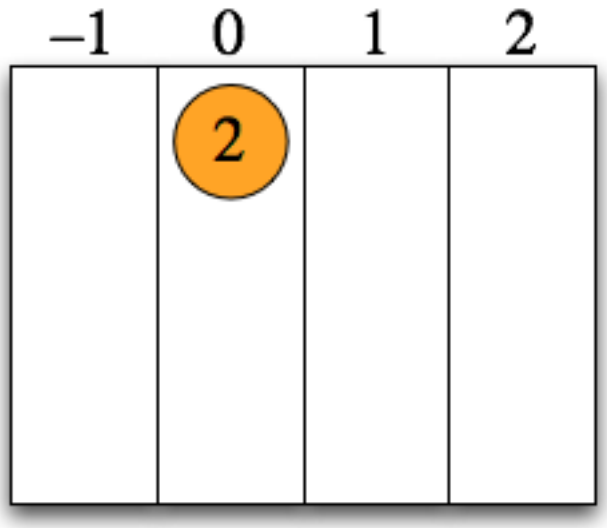
How To Group Into Groups





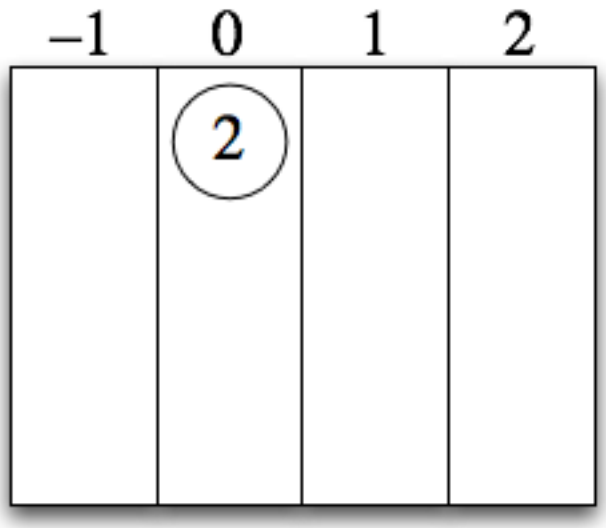
Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups



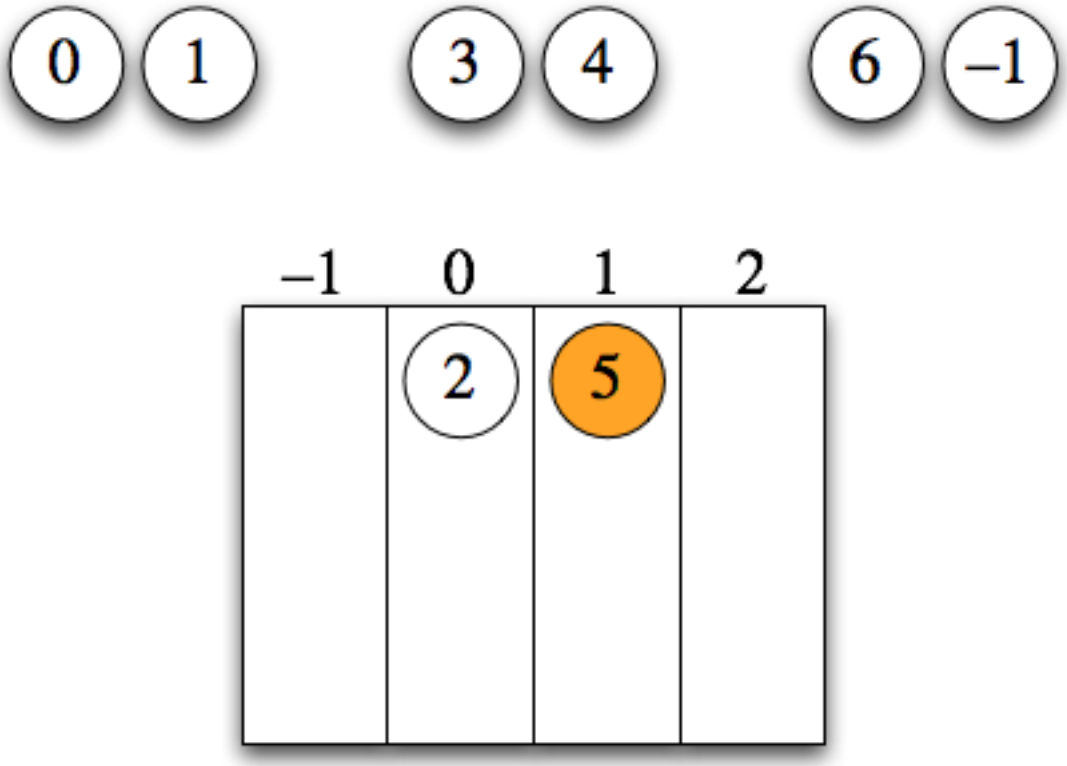
Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups



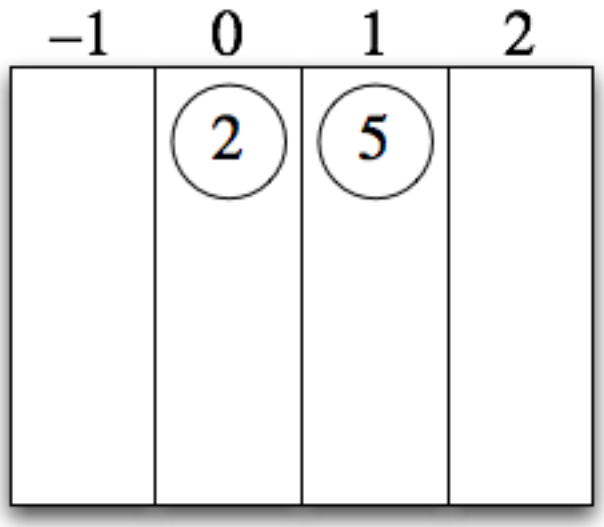
Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups



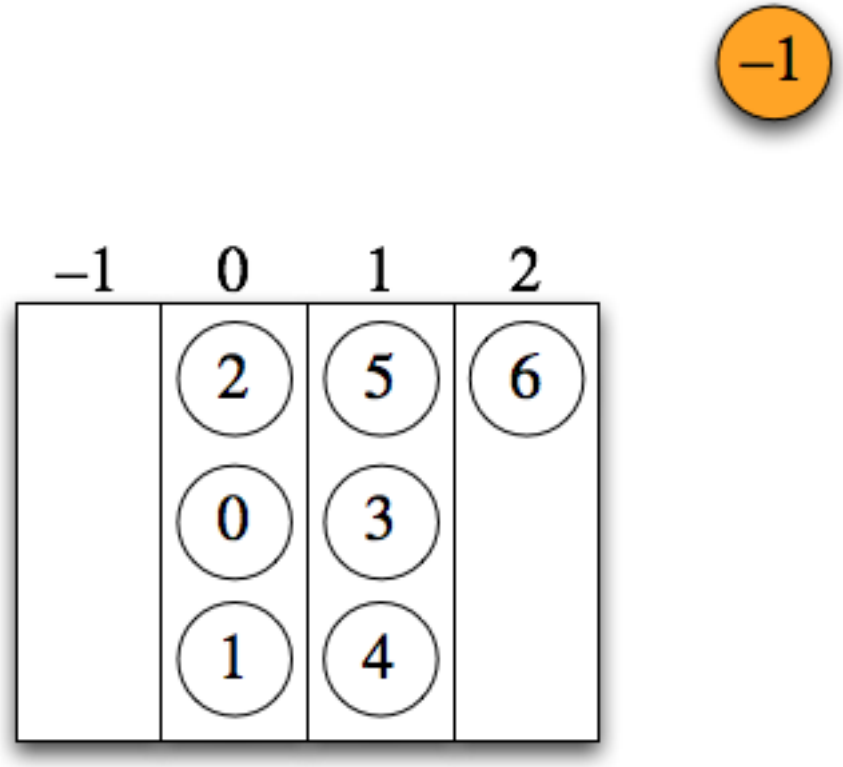
Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups



Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups

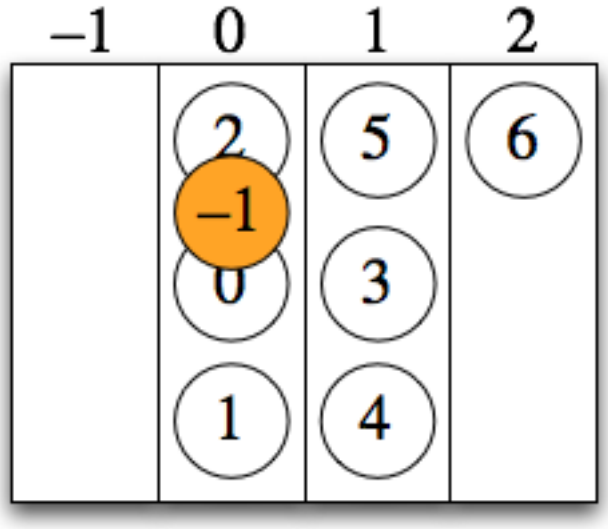


Copyright © 2012, MATHEMA Software GmbH

How To Group Into Groups

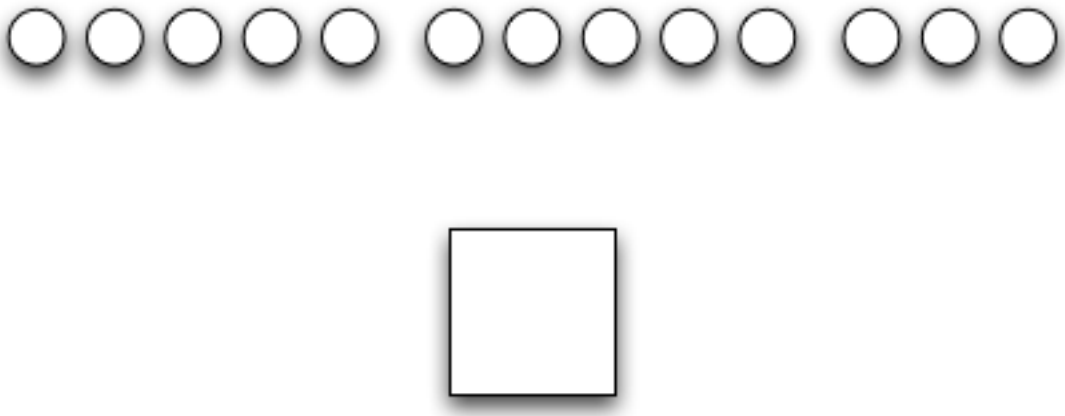
Java: $-1 / 3 = 0$

AL_X: divide$(-1, 3) = -1$



Copyright © 2012, MATHEMA Software GmbH

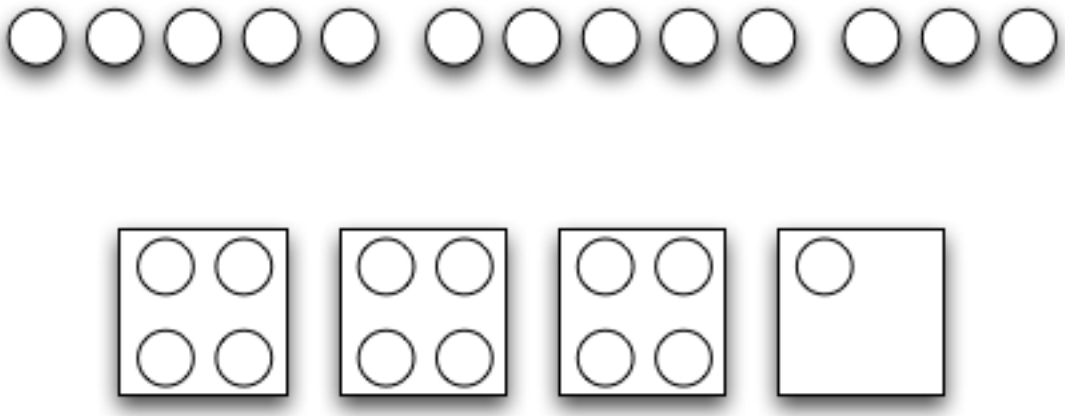
How To Group Into Boxes



Copyright © 2012, MATHEMA Software GmbH

How To Group Into Boxes

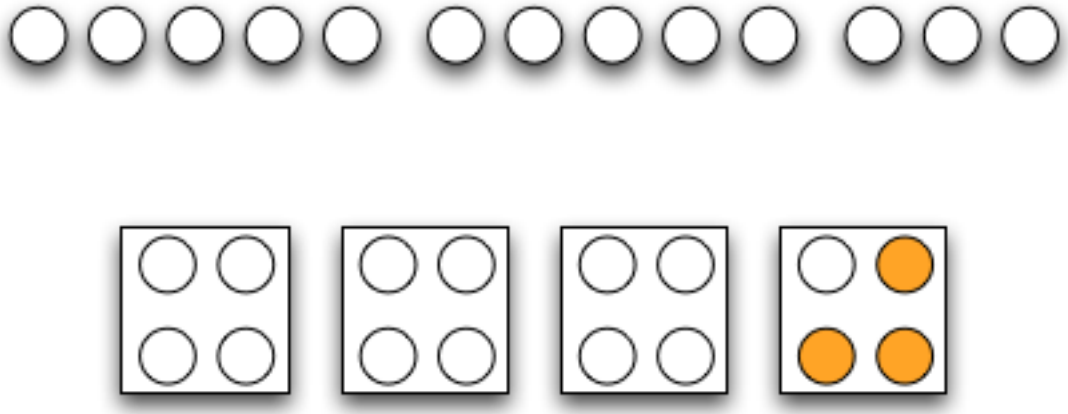
$$group(13, 4) = 4$$



Copyright © 2012, MATHEMA Software GmbH

How To Group Into Boxes

$$pad(13, 4) = 3$$



Copyright © 2012, MATHEMA Software GmbH

Multiplication

- multiply($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** Overflow
- multiplyUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$
- multiplyWithCarry($x : \mathbb{I}_n, y : \mathbb{I}_n, c : \mathbb{I}_n) \rightarrow (p : \mathbb{I}_n, c : \mathbb{I}_n)$
- multiplyWithCarryUnsigned($x : \mathbb{U}_n, y : \mathbb{U}_n, c : \mathbb{U}_n) \rightarrow (\mathbb{U}_n, \mathbb{U}_n)$

- multiplyUnchecked($x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$
- multiplyUnsignedChecked($x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** Overflow

Copyright © 2012, MATHEMA Software GmbH

Optimized Offsets

- $\text{times2plus}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n$
- $\text{times4plus}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n$
- $\text{times8plus}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n$
- $\text{times16plus}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n$
- …

$$\text{times2plus}(x, x) = 3 \cdot x$$

$$\text{times4plus}(x, x) = 5 \cdot x$$

Copyright © 2012, MATHEMA Software GmbH

Shift

- $\text{shiftLeft}(x : \mathbb{I}_n, k : \mathbb{I}_n) \to \mathbb{I}_n$ **throws** {Overflow}
- $\text{shiftRight}(x : \mathbb{I}_n, k : \mathbb{I}_n) \to \mathbb{I}_n$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_n]\!]$ $\text{shiftLeftUnsigned}(x : \mathrm{T}, k : \mathbb{I}_n) \to \mathrm{T}$
- $[\![\mathrm{T} \trianglelefteq \mathbb{M}_n]\!]$ $\text{shiftRightUnsigned}(x : \mathrm{T}, k : \mathbb{I}_n) \to \mathrm{T}$

- $\text{shift}_2(x : \mathbb{I}_n, k : \mathbb{I}_n) \to \mathbb{I}_n$ **throws** {Overflow}
- $\text{shift}_{10}(x : \mathbb{I}_n, k : \mathbb{I}_n) \to \mathbb{I}_n$ **throws** {Overflow}

Copyright © 2012, MATHEMA Software GmbH

- $\mathrm{abs}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** {Overflow} ¶ $|x|$

  $\mathrm{nabs}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ ¶ $-|x|$

- $\mathrm{pow}(x : \mathbb{I}_n, k : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** {Overflow, Invalid} ¶ x^k

  $\mathrm{pow}(x : \mathbb{U}_n, k : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** {Overflow, Invalid}

  $\mathrm{powUnsigned}(x : \mathbb{U}_n, k : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** {Invalid}

- $\mathrm{sqrt}(x : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** {Invalid} ¶ $\lfloor \sqrt{x} \rfloor$

  $\mathrm{sqrtUnsigned}(x : \mathbb{U}_n) \rightarrow \mathbb{U}_n$

- $\mathrm{ceilLog}_2(x : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** {Invalid} ¶ $\lfloor \log_2 x \rfloor$

  $\mathrm{floorLog}_2(x : \mathbb{I}_n) \rightarrow \mathbb{I}_n$ **throws** {Invalid} ¶ $\lceil \log_2 x \rceil$

  $\mathrm{ceilLog}_2(x : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** {Invalid} ¶ $\lfloor \log_2 x \rfloor$

  $\mathrm{floorLog}_2(x : \mathbb{U}_n) \rightarrow \mathbb{U}_n$ **throws** {Invalid} ¶ $\lceil \log_2 x \rceil$

Copyright © 2012, MATHEMA Software GmbH

Comparisons

- $\mathrm{compare}(x : \mathbb{I}_n, y : \mathbb{I}_n) \to \mathbb{J}\{-1, 0, 1\}$

  $\mathrm{compareUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{J}\{-1, 0, 1\}$

- $\mathrm{sign}(x : \mathbb{I}_n) \to \mathbb{J}\{-1, 0, 1\}$

  $\mathrm{signUnsigned}(x : \mathbb{U}_n) \to \mathbb{J}\{0, 1\}$

- $\mathrm{min}(x : \mathbb{I}_n, y : \mathbb{I}_n) \to \mathbb{I}_n\{x, y\}$

  $\mathrm{minUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n\{x, y\}$

- $\mathrm{max}(x : \mathbb{I}_n, y : \mathbb{I}_n) \to \mathbb{I}_n\{x, y\}$

  $\mathrm{maxUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \to \mathbb{U}_n\{x, y\}$

Copyright © 2012, MATHEMA Software GmbH

Control

Copyright © 2012, MATHEMA Software GmbH

Unconditional Jump

- **goto** ([[static]] *label* : 𝕃)

```
        goto INIT↓
LOOP:
        …
        goto LOOP↑
        …
INIT:
        …
        goto LOOP↑
```

```
        goto ①↓
        …
①:
        goto ①↑
        …
①:
        …
        goto ①↑
```


Copyright © 2012, MATHEMA Software GmbH

case offset **of**

- 0:
  goto ⓪↓
- 1:
  goto ①↓
- 2:
  goto ②↓
- 3:
  goto ③↓

end

case offset **of**

- 0, … , 3:
  goto ⓪↓
- 4, 10, …, 12:
  goto ①↓
- :
  goto ②↓

end

Copyright © 2012, MATHEMA Software GmbH

Conditional Jumps

- **if** *condition* **then**
  goto L
  end if
- **unless** *condition* **then**
  goto L
  end if

if positive(*r*) **then**
goto END
end

Copyright © 2012, MATHEMA Software GmbH

Conditional Assignment

- **if** *condition* **then**

  $v \leftarrow r$

  end if

- **unless** *condition* **then**

  $v \leftarrow r$

  end if

$r :\leftarrow 0$

if greater(x, y) **then**

  $r \leftarrow 1$

end

Copyright © 2012, MATHEMA Software GmbH

Predicates

- $\text{null}(x : \mathbb{M}_p) \to \mathbb{B}$

  $\text{nonNull}(x : \mathbb{M}_p) \to \mathbb{B}$

- $\text{zero}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{nonZero}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{zero}(x : \mathbb{U}_n) \to \mathbb{B}$

  $\text{nonZero}(x : \mathbb{U}_n) \to \mathbb{B}$

- $\text{positive}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{nonPositive}(x : \mathbb{I}_n) \to \mathbb{B}$

- $\text{negative}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{nonNegative}(x : \mathbb{I}_n) \to \mathbb{B}$

- $\text{even}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{odd}(x : \mathbb{I}_n) \to \mathbb{B}$

  $\text{even}(x : \mathbb{U}_n) \to \mathbb{B}$

  $\text{odd}(x : \mathbb{U}_n) \to \mathbb{B}$

Copyright © 2012, MATHEMA Software GmbH

- $\text{equal}(x : \mathbb{M}_p, y : \mathbb{M}_p) \rightarrow \mathbb{B}$

  $\text{nonEqual}(x : \mathbb{M}_p, y : \mathbb{M}_p) \rightarrow \mathbb{B}$

- $\text{less}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{B}$

  $\text{lessUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{B}$

- $\text{lessOrEqual}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{B}$

  $\text{lessOrEqualUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{B}$

- $\text{greater}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{B}$

  $\text{greaterUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{B}$

- $\text{greaterOrEqual}(x : \mathbb{I}_n, y : \mathbb{I}_n) \rightarrow \mathbb{B}$

  $\text{greaterOrEqualUnsigned}(x : \mathbb{U}_n, y : \mathbb{U}_n) \rightarrow \mathbb{B}$

Copyright © 2012, MATHEMA Software GmbH

Two Syntactical Exceptions

Copyright © 2012, MATHEMA Software GmbH

Syntactical “Exception” #1: Iverson Operator

⟦intrinsic⟧

function [.] : (𝔹) → 𝕁{0, 1}

$$\text{sign}: (x) \mapsto [\text{positive}(x)] - [\text{negative}(x)]$$

$$\text{sign}: (x) \mapsto [x > 0] - [x < 0]$$

$x < 0$	0	−	1	= −1
$x = 0$	0	−	0	= 0
$x > 0$	1	−	0	= +1

Copyright © 2012, MATHEMA Software GmbH

Syntactical “Exception” #2: Target Operator

$[\![\text{intrinsic}]\!]$

function $\oplus.$: $([\![\text{var}]\!]\ \text{T}) \to \mathbb{A}\,[\![\text{T}]\!]$

data X **is**

$x : \mathbb{I}_{32}$

end data

$i :\leftarrow \text{load}(\oplus x)$

$v :\leftarrow \text{load}\,[\![\text{Tetra}]\!]\,(\oplus x)$

Copyright © 2012, MATHEMA Software GmbH

Side Effects

Copyright © 2012, MATHEMA Software GmbH

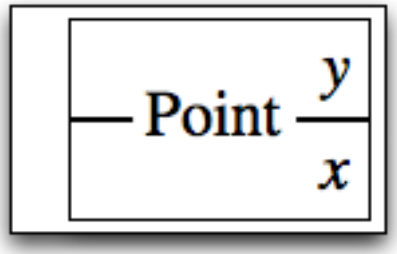
```
class Point {
    int x;
    int y;
}
class Line {
    Point p1;
    Point p2;
}
Line line = new Line(...)
```


Copyright © 2012, MATHEMA Software GmbH

```
class Point extends Object {

    int x;
    int y;

}

class Line extends Object {

    Point p1;
    Point p2;

}

Line line = new Line(…)
```

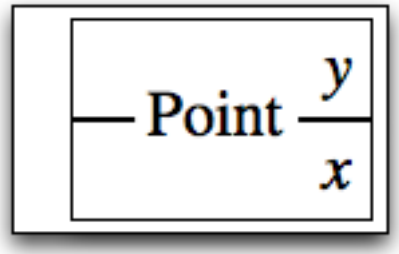

Copyright © 2012, MATHEMA Software GmbH

```
class Point extends Object {
    int x;
    int y;
}
class Line extends Object {
    Point p1;
    Point p2;
}
Line line = new Line(...)
```

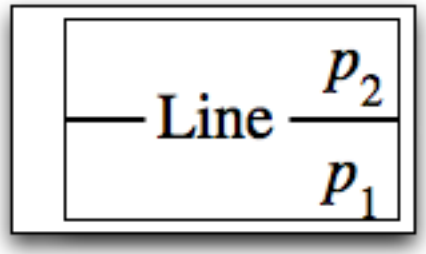


Copyright © 2012, MATHEMA Software GmbH

```
class Point extends Object {
    int x;
    int y;
}
```




```
class Line extends Object {
    Point p1;
    Point p2;
}
```




```
Line line = new Line(...)
```


Copyright © 2012, MATHEMA Software GmbH

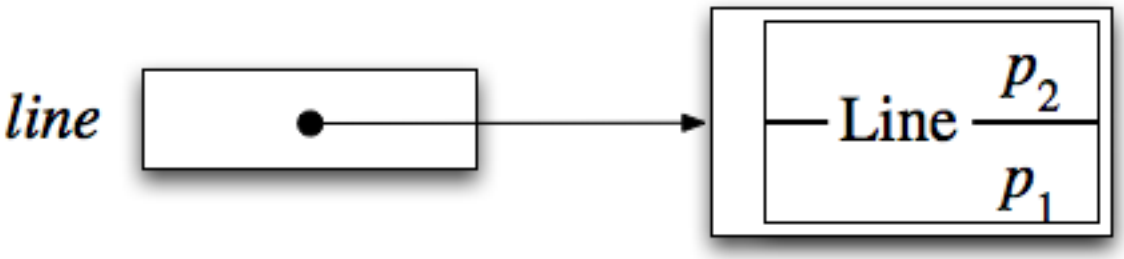
Line *line*;

line = **new** Line()

$line.p_1$ = **new** Point(…);

$line.p_2$ = **new** Point(…);

Copyright © 2012, MATHEMA Software GmbH



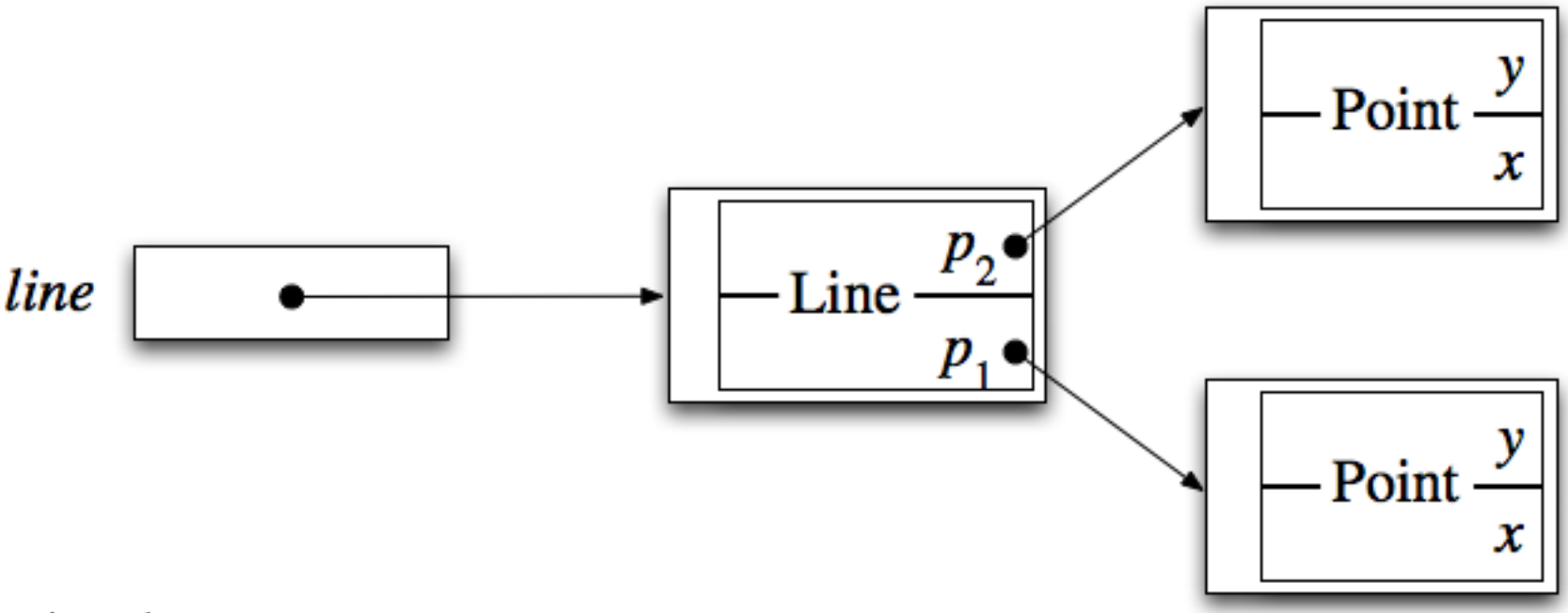
Line *line*;

line = **new** Line()

$line.p_1$ = **new** Point(…);

$line.p_2$ = **new** Point(…);

Copyright © 2012, MATHEMA Software GmbH



Line *line*;

line = **new** Line()

$line.p_1 = \textbf{new}\ \text{Point}(\ldots);$

$line.p_2 = \textbf{new}\ \text{Point}(\ldots);$

Copyright © 2012, MATHEMA Software GmbH

Line *line*;

line = **new** Line();

Point *p*;

p = **new** Point();

$line.p_1 = p$;

Copyright © 2012, MATHEMA Software GmbH

Line *line*;	*line* : Line
line = **new** Line();	*line* ← allocateManaged(Line)
Point *p*;	*p* : Point
p = **new** Point();	*p* ← allocateManaged(Point)
$line.p_1 = p$;	storeFieldReference(*line*, *p*, Line.p_1)

Copyright © 2012, MATHEMA Software GmbH

Line *line* = **new** Line();	*line* :← allocateManaged(Line)
Point *p* = **new** Point();	*p* :← allocateManaged(Point)
$line.p_1 = p$;	storeFieldReference(*line*, *p*, Line.p_1)

Copyright © 2012, MATHEMA Software GmbH

Line *line* = **new** Line();	*line* :← allocateManaged(Line)
Point *p* = **new** Point();	Line.init(*line*)
$line.p_1 = p$;	*p* :← allocateManaged(Point)
	Point.init(*p*)
	storeFieldReference(*line*, *p*, Line.p_1)

Copyright © 2012, MATHEMA Software GmbH

- load $[\![\textbf{implicit } T \trianglelefteq \mathbb{I}_n]\!](\vec{a} : \mathbb{P}[\![T]\!]) \to T$
- loadUnsigned $[\![\textbf{implicit } T \trianglelefteq \mathbb{U}_n]\!](\vec{a} : \mathbb{P}[\![T]\!]) \to T$

data DataSegment **is**

$value : \mathbb{I}_{32}$

end data

$\vec{a} : \mathbb{P}[\![\text{Tetra}]\!] \leftarrow \bigoplus value$

$v :\leftarrow \text{load}[\![\text{Tetra}]\!](\vec{a})$

$v \leftarrow \text{load}(\vec{a})$

Copyright © 2012, MATHEMA Software GmbH

- store $[\![\textbf{implicit } \mathrm{T} \trianglelefteq \mathbb{I}_n]\!]$

  $(\vec{a} : \mathbb{P}\,[\![\mathrm{T}]\!], \mathit{value} : (\mathbb{I}_k | \mathbb{U}_k))$ **throws** {Overflow} **where** $\{k \geq n\}$

- storeUnsigned $[\![\textbf{implicit } \mathrm{T} \trianglelefteq \mathbb{M}_p]\!]$

  $(\vec{a} : \mathbb{P}\,[\![\mathrm{T}]\!], \mathit{value} : (\mathbb{I}_k | \mathbb{U}_k))$ **where** $\{k \geq 8p\}$

$\vec{a} : \mathbb{P}\,[\![\mathrm{Tetra}]\!] \leftarrow \oplus \mathit{value}$

store($\vec{a}$, *i32*)

store($\vec{a}$, *i64*)

store($\vec{a}$, *i16*)

data DataSegment **is**

$\mathit{value} : \mathbb{I}_{32}$

end data

Copyright © 2012, MATHEMA Software GmbH

- **type** FieldDescriptor ⟦OwningClass ⊴ Class, FieldType ⊴ Any⟧
- ⟦C ⊴ Class⟧ loadField ⟦**implicit** T ⊴ $\mathbb{M}_p$⟧
  ($\vec{a}$: $\mathbb{P}$ ⟦C⟧, ⟦static⟧ *fd* : FieldDescriptor ⟦C, T⟧) → T

$\vec{a}$: $\mathbb{P}$ ⟦Point⟧ ← $\oplus p$

x :← loadField ⟦Tetra⟧ ($\vec{a}$, Point.*x*)

x ← loadField($\vec{a}$, Point.*x*)

record Point **is**

x : $\mathbb{I}_{32}$

y : $\mathbb{I}_{32}$

end record

data DataSegment **is**

p : Point

end data

Copyright © 2012, MATHEMA Software GmbH

Calculate Field Address

$[\![C \trianglelefteq \text{Class}, T \trianglelefteq \text{Any}]\!]$ calculateFieldAddress

$(\vec{a} : \mathbb{P}[\![C]\!],\ [\![\text{static}]\!]\ fd : \text{FieldDescriptor}[\![C, T]\!]) \to \mathbb{P}[\![T]\!]$

$\vec{a} : \mathbb{P}[\![\text{Line}]\!] \leftarrow \oplus line$

$\vec{p}_1 :\leftarrow \text{calculateFieldAddress}[\![\text{Point}]\!](\vec{a}, \text{Line}.p_1)$

$\vec{p}_1 \leftarrow \text{calculateFieldAddress}(\vec{a}, \text{Line}.p_1)$

$x :\leftarrow \text{loadField}(\vec{p}_1, \text{Point}.x)$

record Point **is**

$x : \mathbb{I}_{32}; y : \mathbb{I}_{32}$

end record

record Line **is**

p_1 : Point; p_2 : Point

end record

data DataSegment **is**

line : Line

end data

Copyright © 2012, MATHEMA Software GmbH

- loadElement⟦**implicit** T ⊴ $\mathbb{M}_p$⟧($\vec{a}$: $\mathbb{P}_*$⟦Sq⟦T, k⟧⟧, i : ($\mathbb{I}_n$|$\mathbb{U}_n$)) → T

  requires $\{0 \le i < k\}$

  loadElementUnsigned⟦**implicit** T ⊴ $\mathbb{M}_p$⟧

  ($\vec{a}$: $\mathbb{P}_*$⟦Sq⟦T, k⟧⟧, i : ($\mathbb{I}_n$|$\mathbb{U}_n$)) → T

  requires $\{0 \le i < k\}$

$\vec{a}$: $\mathbb{P}_*$⟦Sq⟦$\mathbb{I}_{32}$, 10⟧⟧ ← $\bigoplus si$

x :← loadElement⟦Tetra⟧($\vec{a}$, i)

x ← loadElement($\vec{a}$, i)

data DataSegment **is**

si : Sq⟦$\mathbb{I}_{32}$, 10⟧

sp : Sq⟦Point, 2⟧⟧

end data

Copyright © 2012, MATHEMA Software GmbH

Calculate Array Element Address

- calculateElementAddress $[\![$**implicit** T $\trianglelefteq$ Any$]\!]$

  $(\vec{a} : \mathbb{P}_* [\![\mathrm{Sq}[\![\mathrm{T}, k]\!]]\!], i : (\mathbb{I}_n | \mathbb{U}_n)) \to \mathbb{P}[\![\mathrm{T}]\!]$

  requires $\{0 \le i < k\}$

$\vec{a} : \mathbb{P}_* [\![\mathrm{Sq}[\![\mathrm{Point}, 3]\!]]\!] \leftarrow \bigoplus sp$

$\vec{e} :\leftarrow \mathrm{calculateElementAddress}[\![\mathrm{Point}]\!](\vec{a}, i)$

$\vec{e} \leftarrow \mathrm{calculateElementAddress}(\vec{a}, i)$

$\vec{x} \leftarrow \mathrm{loadElement}(\vec{e}, \overrightarrow{\mathrm{Point}.x})$

record Point **is**

$x : \mathbb{I}_{32}; y : \mathbb{I}_{32}$

end record

data DataSegment **is**

sp : Sq$[\![$Point, 3$]\!]$

end data

Copyright © 2012, MATHEMA Software GmbH

Special Variants of load/storeUnsigned

- loadReference/Pointer
- loadFieldReference/Pointer
- loadElementReference/Pointer

- storeReference/Pointer
- storeFieldReference/Pointer
- storeElementReference/Pointer

Copyright © 2012, MATHEMA Software GmbH

[[restricted permission=RandomMemoryAccess]]

- $\mathrm{load}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!]) \to \mathbb{I}_{8p}$

  $\mathrm{load}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{offset} : (\mathbb{I}_n|\mathbb{U}_n)) \to \mathbb{I}_{8p}$

  $\mathrm{loadUnsigned}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!]) \to \mathbb{U}_{8p}$

  $\mathrm{loadUnsigned}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{offset} : (\mathbb{I}_n|\mathbb{U}_n)) \to \mathbb{U}_{8p}$

- $\mathrm{store}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{value} : (\mathbb{I}_k|\mathbb{U}_k))$ **throws** {Overflow}

  $\mathrm{store}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{offset} : (\mathbb{I}_n|\mathbb{U}_n), \mathit{value} : (\mathbb{I}_k|\mathbb{U}_k))$ **throws** {Overflow}

  where $\{k \geq 8p\}$

  $\mathrm{storeUnsigned}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{value} : \mathbb{M}_q)$

  $\mathrm{storeUnsigned}\,[\![\mathbb{M}_p]\!]\,(\vec{a} : \mathbb{P}\,[\![?]\!], \mathit{offset} : (\mathbb{I}_n|\mathbb{U}_n), \mathit{value} : \mathbb{M}_q)$

  where $\{q \geq p\}$

Copyright © 2012, MATHEMA Software GmbH

⟦restricted permission=RandomMemoryAccess⟧

- loadField $⟦\mathbb{M}_p⟧(\vec{a} : \mathbb{P}⟦?⟧,\ ⟦\text{static}⟧\ fd : \text{FieldDescriptor}⟦?, ?⟧) \to \mathbb{I}_{8p}$

  loadFieldUnsigned $⟦\mathbb{M}_p⟧($——————— " ———————$) \to \mathbb{U}_{8p}$

- storeField $⟦\mathbb{M}_p⟧$

  $(\vec{a} : \mathbb{P}⟦?⟧,\ ⟦\text{static}⟧\ fd : \text{FieldDescriptor}⟦?, ?⟧,\ value : (\mathbb{I}_n|\mathbb{U}_n))$

  throws {Overflow} **where** $\{n \geq 8p\}$

  storeFieldUnsigned $⟦\mathbb{M}_p⟧($——————— " ———————$)$

  where $\{n \geq 8p\}$

- calculateFieldAddress

  $(\vec{a} : \mathbb{P}⟦?⟧,\ ⟦\text{static}⟧\ fd : \text{FieldDescriptor}⟦?, ?⟧) \to \mathbb{P}⟦?⟧$

Copyright © 2012, MATHEMA Software GmbH

[[restricted permission=RandomMemoryAccess]]

- loadElement $[\![\mathbb{M}_p]\!](\vec{a} : \mathbb{P}[\![?]\!], i : (\mathbb{I}_n|\mathbb{U}_n)) \rightarrow \mathbb{I}_k$

  loadElementUnsigned $[\![\mathbb{M}_p]\!](\vec{a} : \mathbb{P}[\![?]\!], (\mathbb{I}_n|\mathbb{U}_n)) \rightarrow \mathbb{U}_k$

- storeElement $[\![\mathbb{M}_p]\!](\vec{a} : \mathbb{P}[\![?]\!], i : (\mathbb{I}_n|\mathbb{U}_n), value : (\mathbb{I}_k|\mathbb{U}_k))$

  throws {Overflow} **where** $\{k \geq 8p\}$

  storeElementUnsigned $[\![\mathbb{M}_p]\!](\vec{a} : \mathbb{P}[\![?]\!], i : (\mathbb{I}_n|\mathbb{U}_n), value : (\mathbb{I}_k|\mathbb{U}_k))$

  where $\{k \geq 8p\}$

- calculateElementAddress $[\![\mathrm{T}]\!](\vec{a} : \mathbb{P}[\![?]\!], i : (\mathbb{I}_k|\mathbb{U}_k)) \rightarrow \mathbb{P}[\![?]\!]$

Copyright © 2012, MATHEMA Software GmbH

Standard Annotations

- [[uncached]]
- [[unchecked]]
- [[checked]]
- [[tailcall]]
- [[unaligned]]
- [[volatile]]
- [[ordered]]
- …

Copyright © 2012, MATHEMA Software GmbH

References and Pointers

Copyright © 2012, MATHEMA Software GmbH

- allocateManaged(Class ⟦T⟧) → ℗ ⟦T⟧

- allocateUnmanaged(Class ⟦T⟧) → ℙ ⟦T⟧
- freeUnmanaged(ℙ ⟦T⟧)

- allocateLocal ⟦T⟧) → ℙ ⟦T⟧

Copyright © 2012, MATHEMA Software GmbH

object Point **is**

$x : \mathbb{I}_{32}$

$y : \mathbb{I}_{32}$

end

object Line **is**

p_1 : Point

p_2 : Point

end

record @Point **extends** Object **is**

$x : \mathbb{I}_{32}$

$y : \mathbb{I}_{32}$

end

Point ≡ ℧[[@Point]]

record @Line **extends** Object **is**

p_1 : Point

p_2 : Point

end

Line ≡ ℧[[@Line]]

Copyright © 2012, MATHEMA Software GmbH

Line[] *lines* = **new** Line[3];

line = **new** Line();

lines[1] = *line*;

$line.p_1$ = **new** Point();

Copyright © 2012, MATHEMA Software GmbH

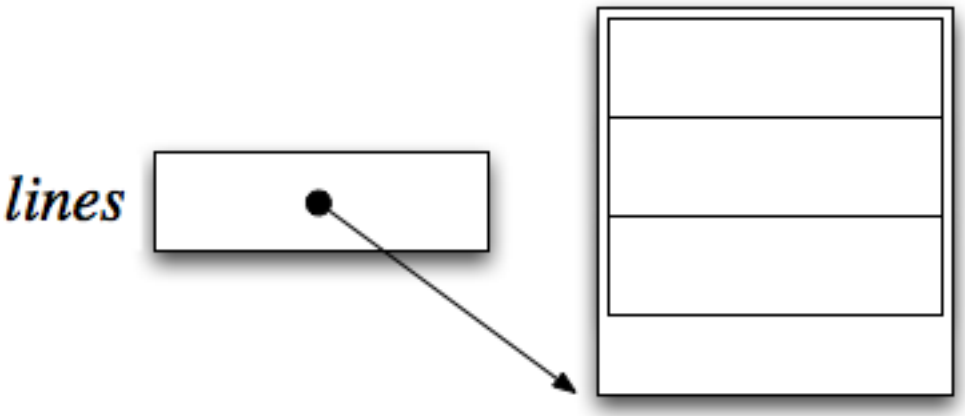
Line[] *lines* = **new** Line[3];

line = **new** Line();

lines[1] = *line*;

$line.p_1$ = **new** Point();

Copyright © 2012, MATHEMA Software GmbH



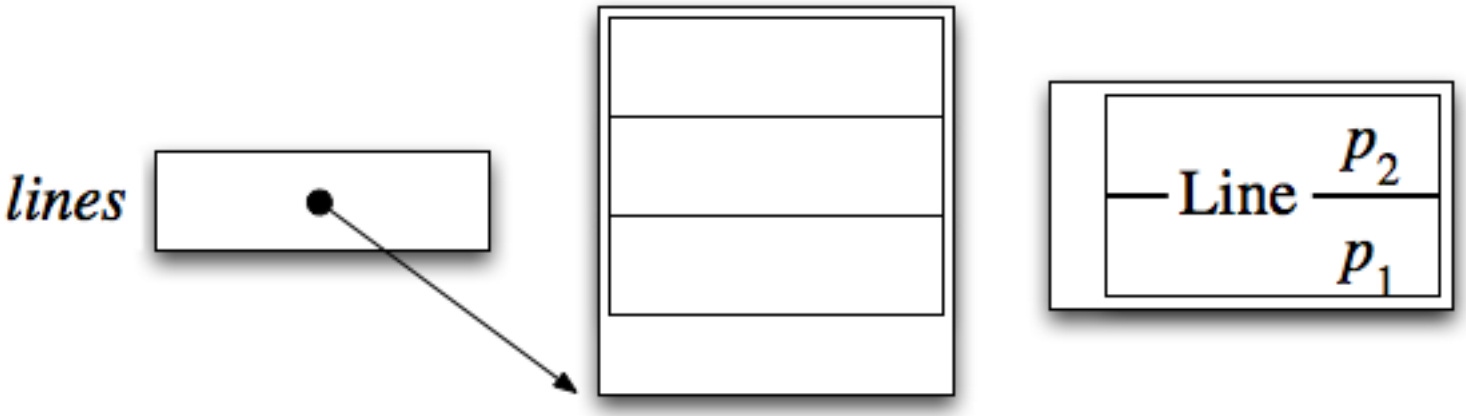
Line[] *lines* = **new** Line[3];

line = **new** Line();

lines[1] = *line*;

$line.p_1$ = **new** Point();

Copyright © 2012, MATHEMA Software GmbH



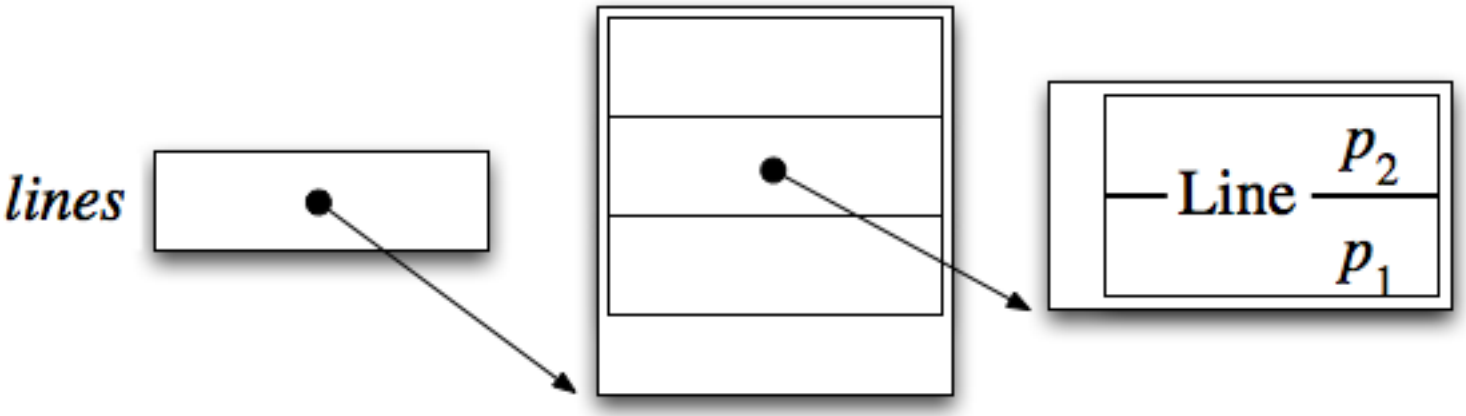
Line[] *lines* = **new** Line[3];

Line *line* = new Line();

lines[1] = *line*;

line.p_1 = **new** Point();

Copyright © 2012, MATHEMA Software GmbH



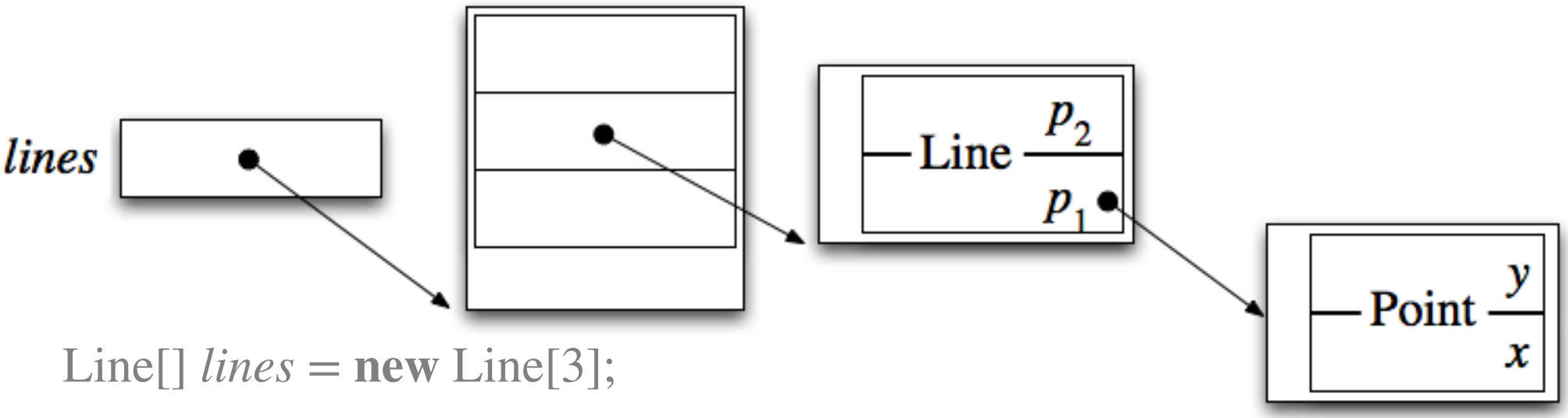
Line[] *lines* = **new** Line[3];

Line *line* = **new** Line();

lines[1] = *line*;

line.p_1 = **new** Point();

Copyright © 2012, MATHEMA Software GmbH



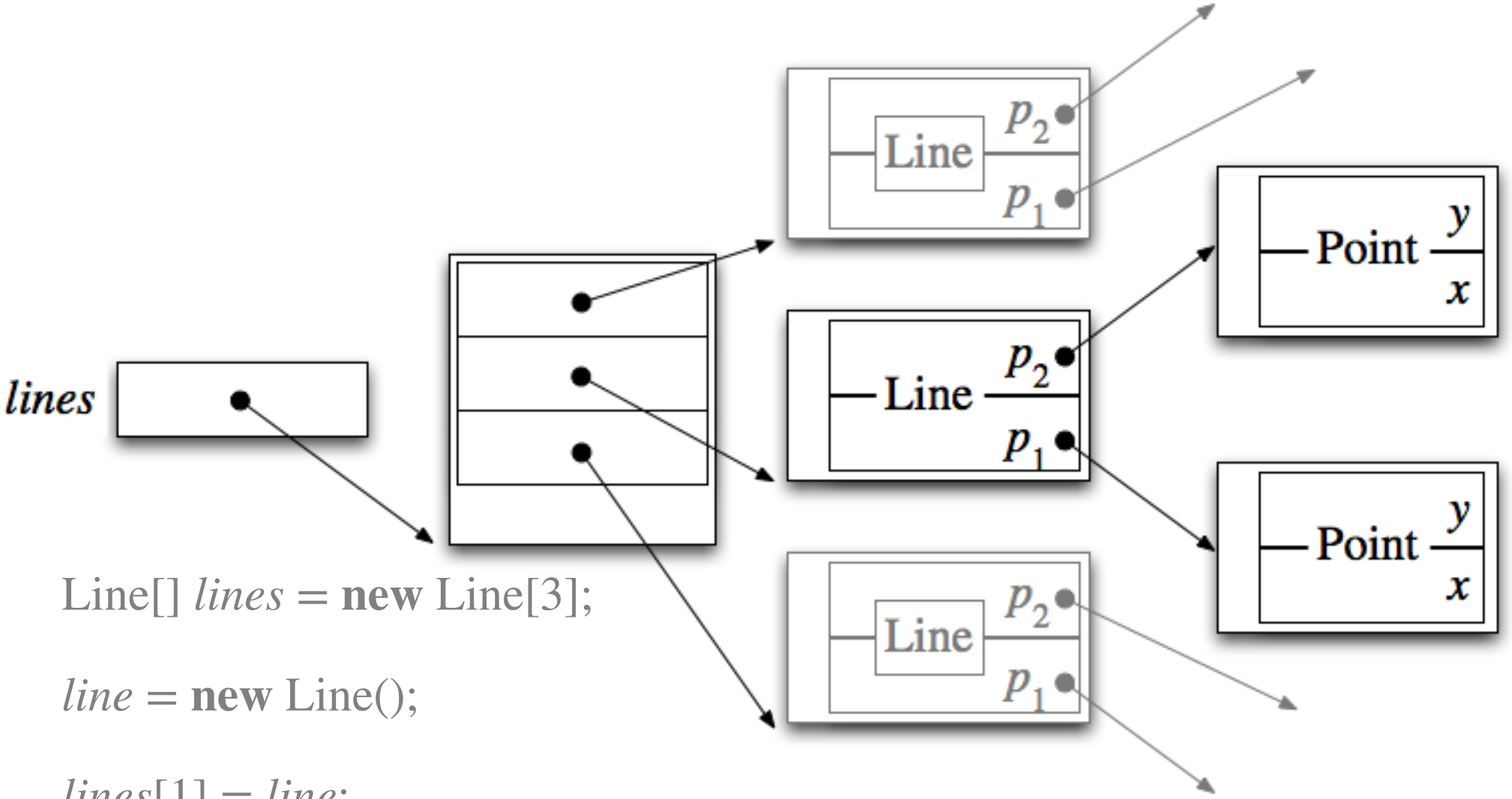
Line[] *lines* = **new** Line[3];

line = **new** Line();

lines[1] = *line*;

line.p_1 = **new** Point();

Copyright © 2012, MATHEMA Software GmbH



Line[] *lines* = **new** Line[3];

line = **new** Line();

lines[1] = *line*;

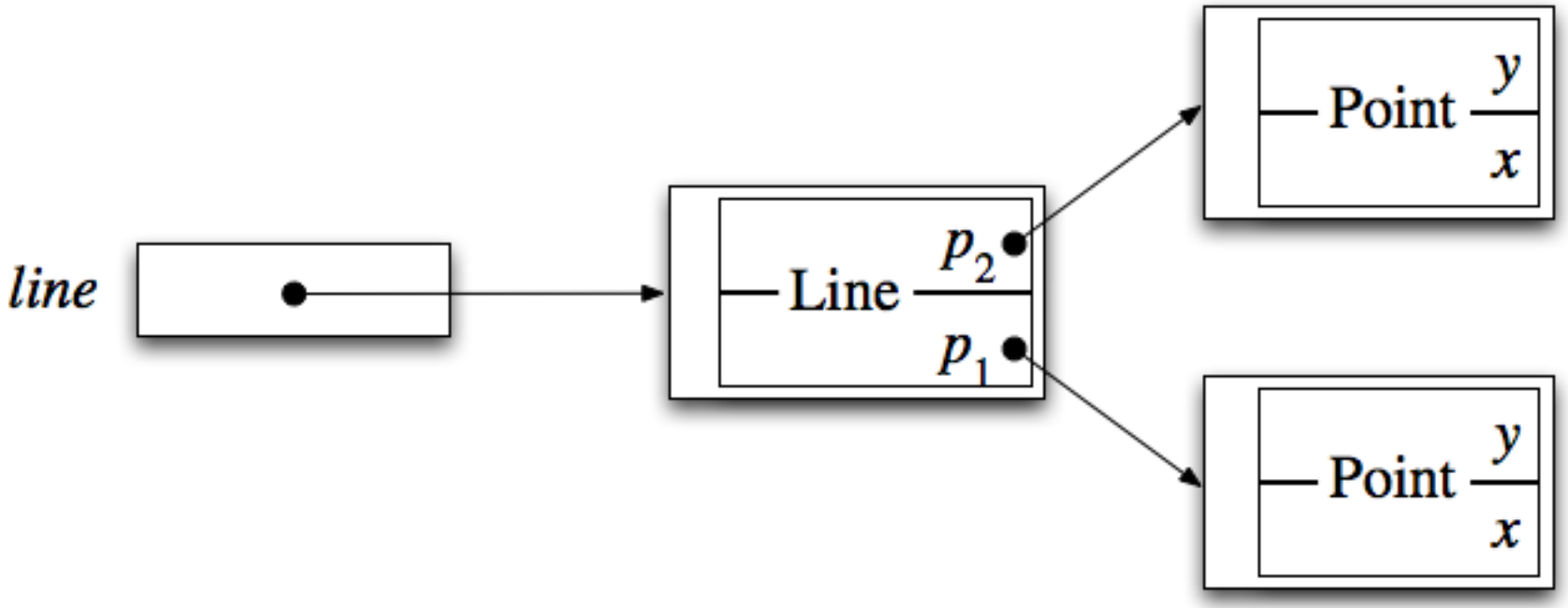
$line.p_1$ = **new** Point();

Copyright © 2012, MATHEMA Software GmbH

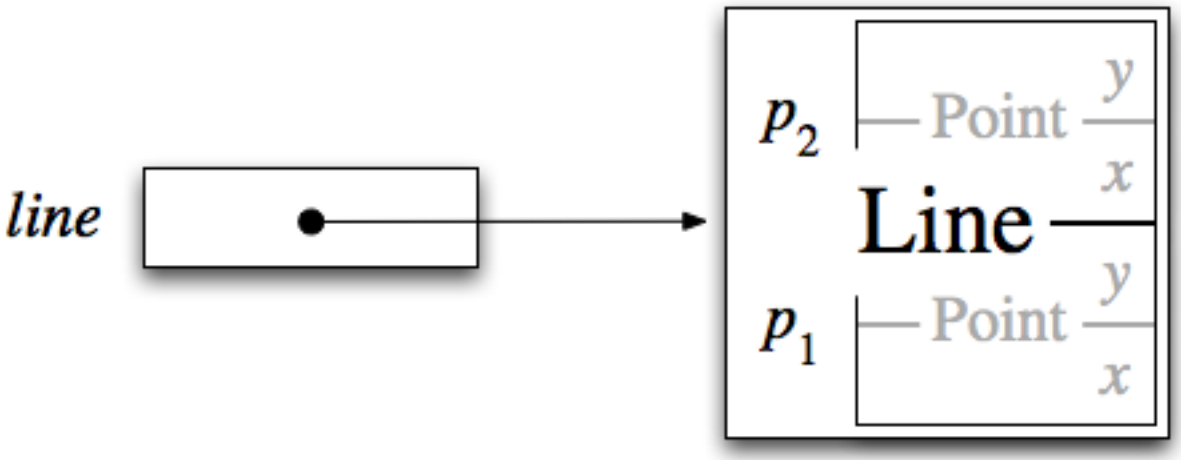
- O: java.lang.Object “overhead” (12 Bytes + *alignment*)
- R: reference size (4 Byte)
- I: size of int (4 Bytes)
- n: number of lines

$$
\begin{aligned}
\mathit{total} &= \textbf{sizeof}(\text{Line}[n]) + n \cdot \textbf{sizeof}(\text{Line}) + 2n \cdot \textbf{sizeof}(\text{Point}) \\
&= (O + n \cdot R) + n \cdot (O + 2 \cdot R) + 2n \cdot (O + 2 \cdot I) \\
&= O + n \cdot R + n \cdot O + 2n \cdot R + 2n \cdot O + 4n \cdot I \\
&= (1 + n + 2n) \cdot O + (n + 2n + 4n) \cdot 4 \\
&= (3n + 1) \cdot O + 7n \cdot 4 \\
&= (9n + 3 + 7n) \cdot 4 \\
&= 64n + 12
\end{aligned}
$$

Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH

record Point **is**

$x : \mathbb{I}_{32}$

$y : \mathbb{I}_{32}$

end

object Line **is**

p_1 : Point

p_2 : Point

end

object Point **is**

$x : \mathbb{I}_{32}$

$y : \mathbb{I}_{32}$

end

object Line **is**

[[embed]]

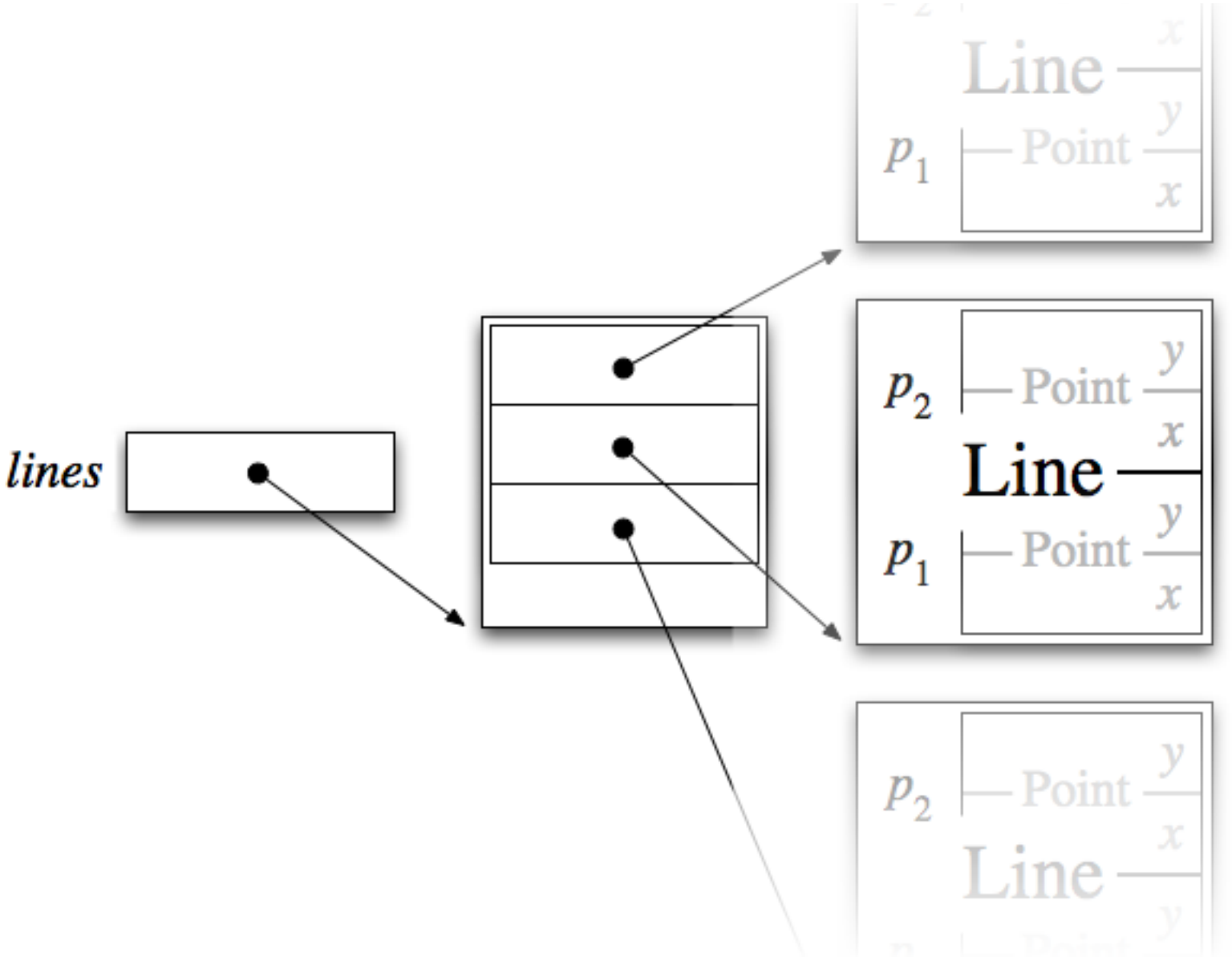
p_1 : Point

[[embed]]

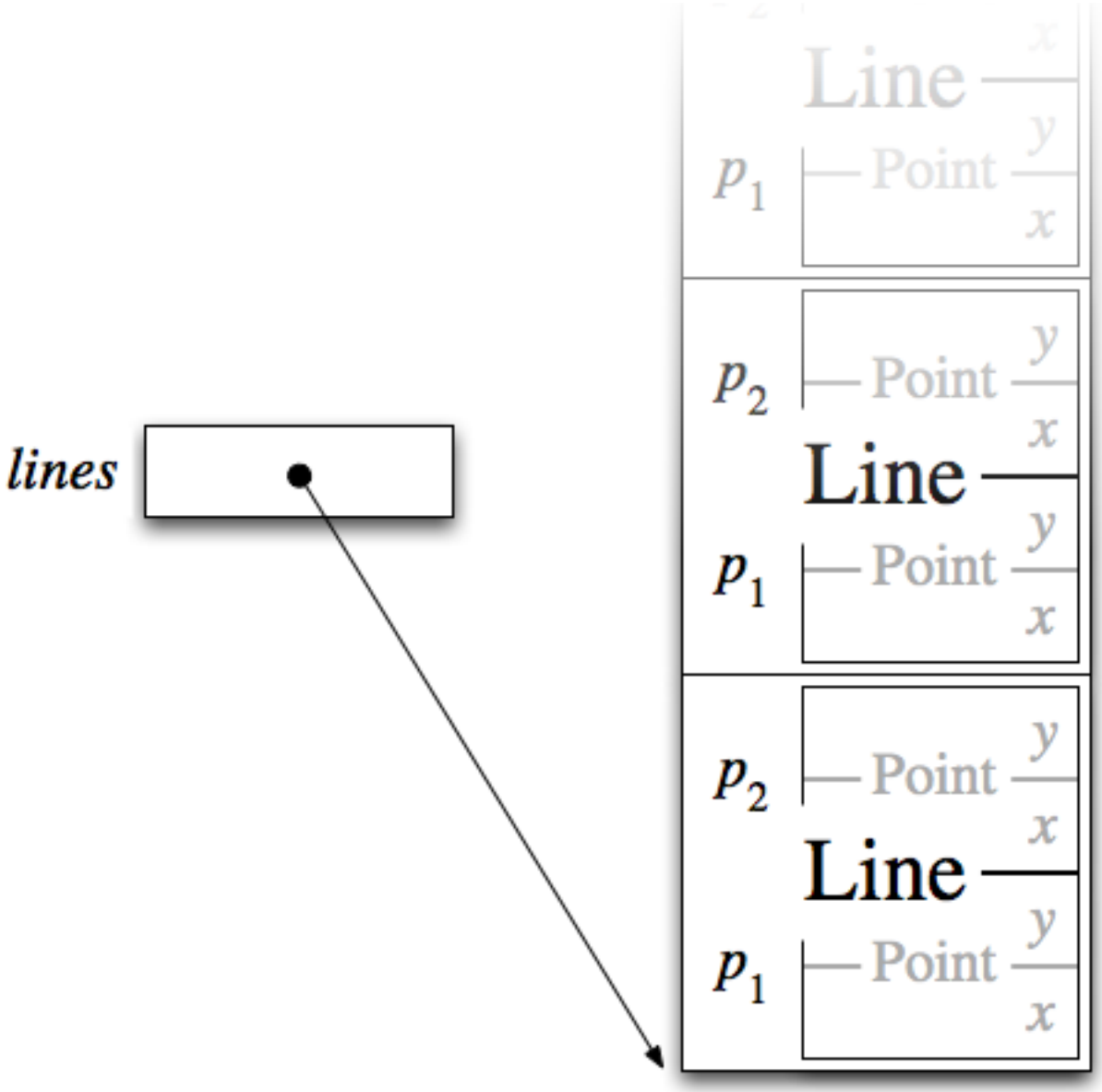
p_2 : Point

end

Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH



Copyright © 2012, MATHEMA Software GmbH

- O: java.lang.Object “overhead” (12 Bytes + alignment)
- R: reference size (4 Byte)
- I: size of int (4 Bytes)
- n: number of lines

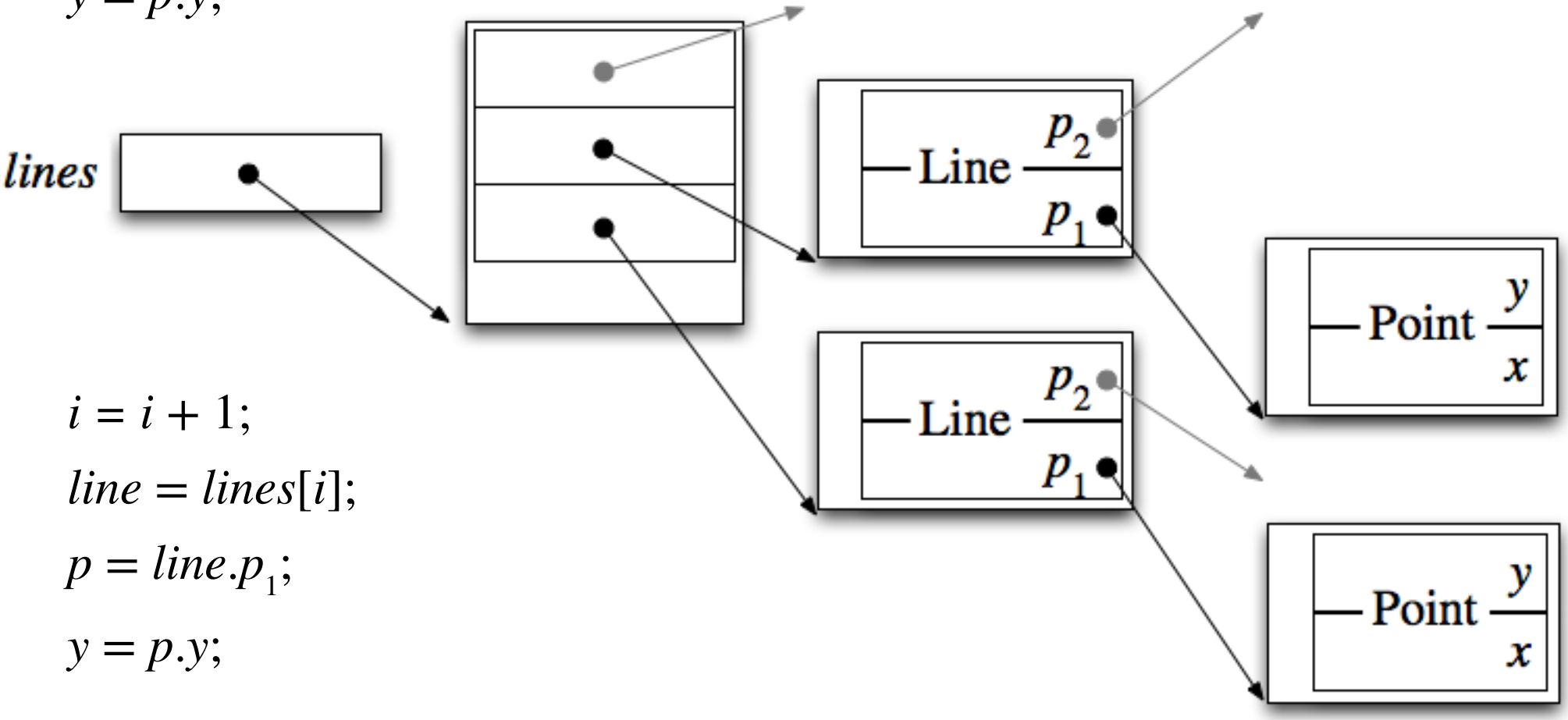
$$
\begin{aligned}
total &= \text{sizeof(Line[}n\text{])} + n \cdot \text{sizeof(Line)} + 2n \cdot \text{sizeof(Point)} \\
&= \text{sizeof(Line[}n\text{])} + n \cdot (\text{sizeof(Line)} + 2 \cdot \text{sizeof(Point)}) \\
&= \text{sizeof(Line[}n\text{])} + n \cdot (2 \cdot \text{sizeof(Point)}) \\
&= O + n \cdot (2 \cdot \text{sizeof(Point)}) \\
&= O + n \cdot (2 \cdot 2 \cdot I) \\
&= 16n + 12 < 64n + 12
\end{aligned}
$$

Copyright © 2012, MATHEMA Software GmbH

$line = lines[i];$

$p = line.p_1;$

$y = p.y;$



$i = i + 1;$

$line = lines[i];$

$p = line.p_1;$

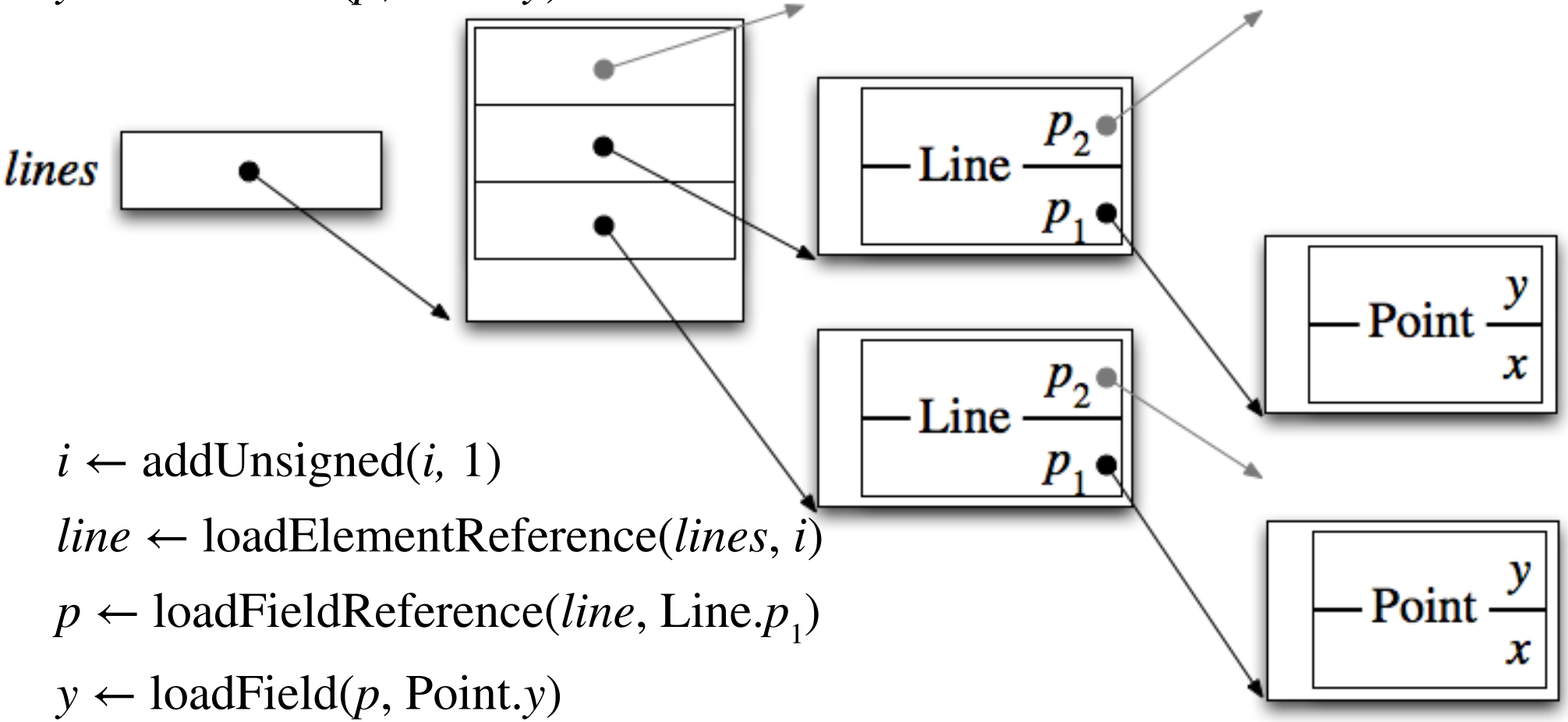
$y = p.y;$

Copyright © 2012, MATHEMA Software GmbH

line ← loadElementReference(lines, i)

p ← loadFieldReference(*line*, Line.p_1)

y ← loadField(p, Point.y)



i ← addUnsigned(i, 1)

line ← loadElementReference(*lines*, i)

p ← loadFieldReference(*line*, Line.p_1)

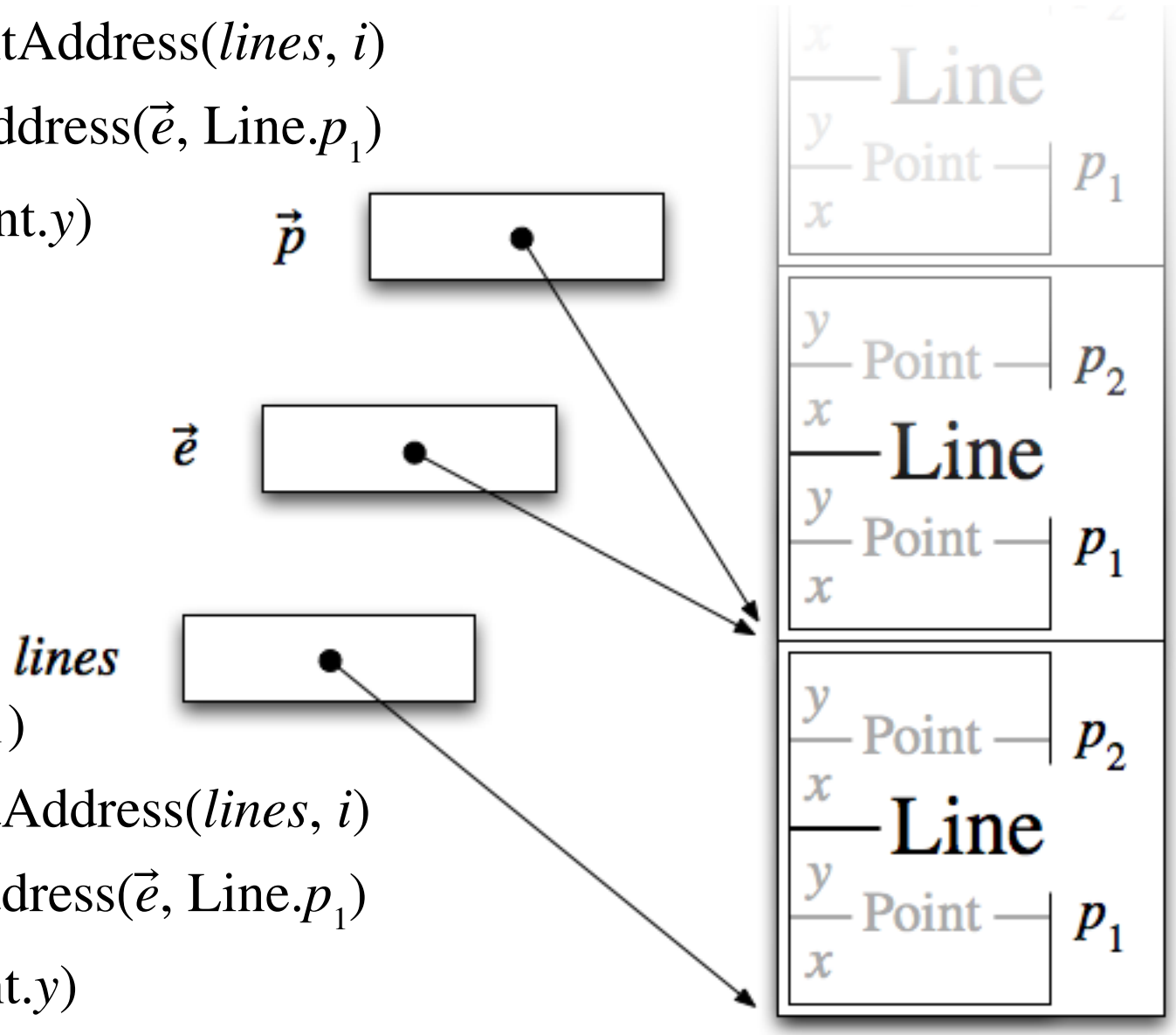
y ← loadField(p, Point.y)

Copyright © 2012, MATHEMA Software GmbH

$\vec{e}$:← calculateElementAddress(*lines*, *i*)

$\vec{p}$:← calculateFieldAddress($\vec{e}$, Line.p_1)

y :← loadField($\vec{p}$, Point.*y*)



i :← addUnsigned(*i*, 1)

$\vec{e}$ ← calculateElementAddress(*lines*, *i*)

$\vec{p}$ ← calculateFieldAddress($\vec{e}$, Line.p_1)

y ← loadField($\vec{p}$, Point.*y*)

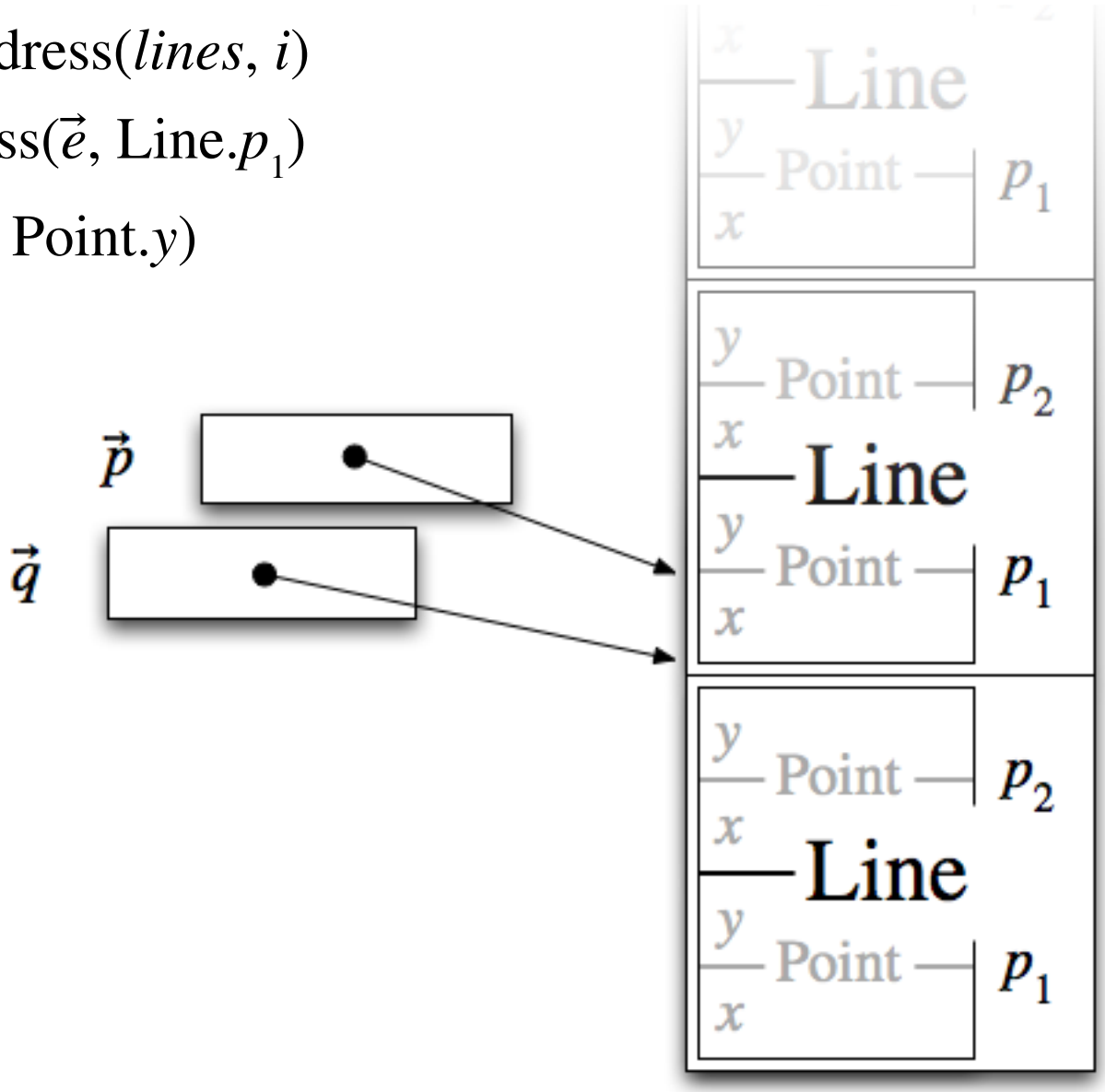
Copyright © 2012, MATHEMA Software GmbH

$\vec{e}$:← calculateElementAddress(*lines*, *i*)

$\vec{q}$:← calculateFieldAddress($\vec{e}$, Line.p_1)

$\vec{p}$:← loadFieldAddress($\vec{q}$, Point.*y*)

y ← load($\vec{p}$)



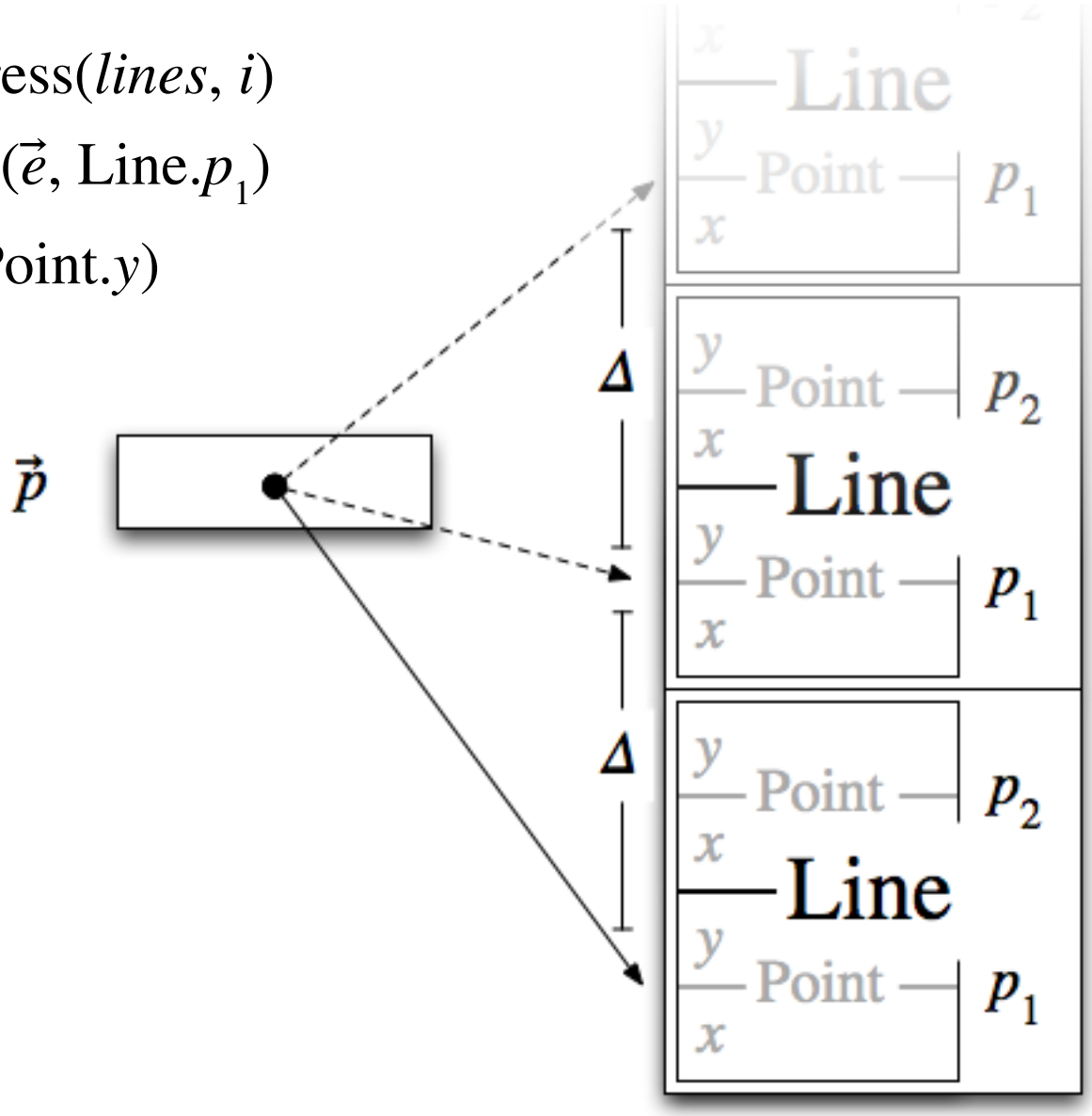
Copyright © 2012, MATHEMA Software GmbH

$\vec{e}$:← calculateElementAddress(*lines*, *i*)

$\vec{q}$:← calculateFieldAddress($\vec{e}$, Line.p_1)

$\vec{p}$:← loadFieldAddress($\vec{q}$, Point.*y*)

y ← load($\vec{p}$)



Copyright © 2012, MATHEMA Software GmbH

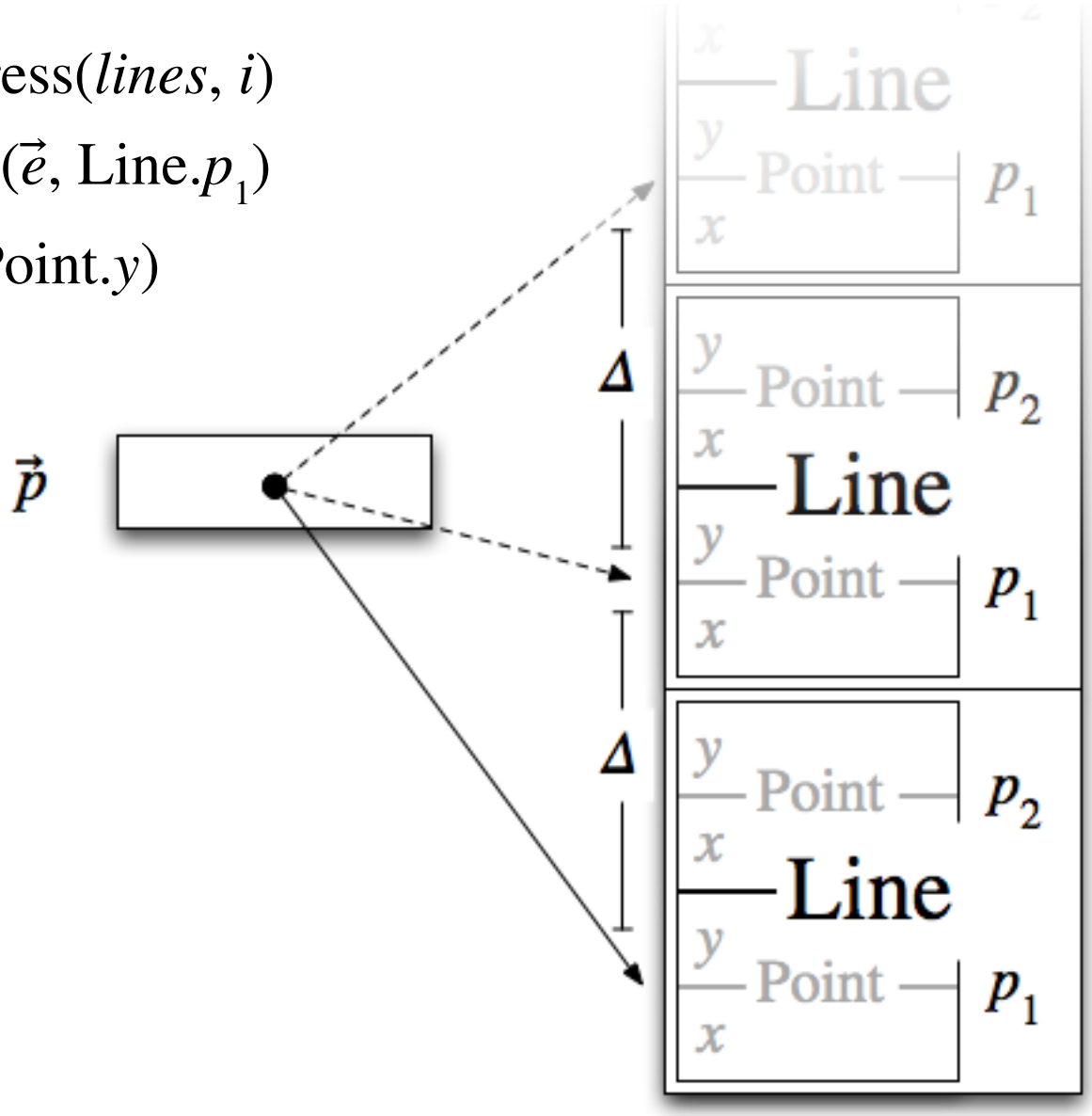
$\vec{e}$:← calculateElementAddress(*lines*, *i*)

$\vec{q}$:← calculateFieldAddress($\vec{e}$, Line.p_1)

$\vec{p}$:← loadFieldAddress($\vec{q}$, Point.*y*)

y ← load($\vec{p}$)

$\Delta := 16$



Copyright © 2012, MATHEMA Software GmbH

$\vec{e}$:← calculateElementAddress(*lines*, *i*)

$\vec{q}$:← calculateFieldAddress($\vec{e}$, Line.p_1)

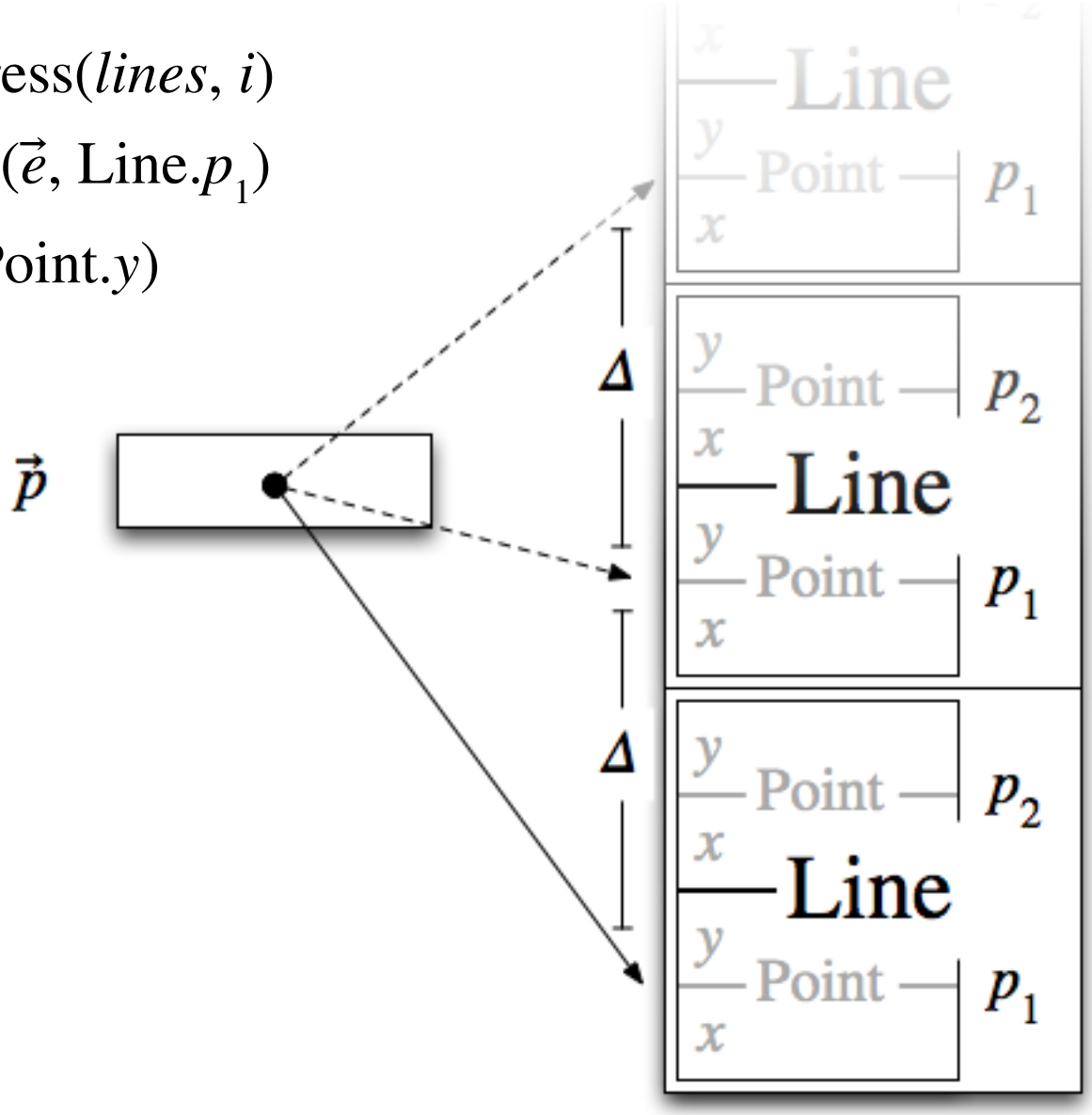
$\vec{p}$:← loadFieldAddress($\vec{q}$, Point.*y*)

y ← load($\vec{p}$)

Δ := 16

$\vec{p}$ ← add($\vec{p}$, Δ)

y ← load($\vec{p}$)



Copyright © 2012, MATHEMA Software GmbH

Questions?

Copyright © 2012, MATHEMA Software GmbH

Contact

Michael Wiedeking
michael.wiedeking@mathema.de

MATHEMA Software GmbH
Henkestraße 91
91052 Erlangen
Germany

Copyright © 2012, MATHEMA Software GmbH