Reference Counting

*Reference counting:* a way to know whether a record has other users

- Attatch a count to every record, starting at 0
- When installing a pointer to a record (into a register or another record), increment its count
- When replacing a pointer to a record, decrement its count
- When a count is decremented to 0, decrement counts for other records referenced by the record, then free it
Reference Counting

Top boxes are the registers $expr$, $todos$, etc.

Boxes in the blue area are allocated with $malloc$
Reference Counting

Adjust counts when a pointer is changed...
Reference Counting

... freeing a record if its count goes to 0
Reference Counting

Same if the pointer is in a register
Reference Counting

Adjust counts after frees, too...
Reference Counting

... which can trigger more frees
Reference Counting And Cycles

An assignment can create a cycle...
Reference Counting And Cycles

Adding a reference increments a count
Reference Counting And Cycles

Lower-left records are inaccessible, but not deallocated

In general, cycles break reference counting
Garbage Collection

**Garbage collection:** a way to know whether a record is *accessible*

- A record referenced by a register is *live*
- A record referenced by a live record is also live
- A program can only possibly use live records, because there is no way to get to other records
- A garbage collector frees all records that are not live
- Allocate until we run out of memory, then run a garbage collector to get more space
Garbage Collection Algorithm

• Color all records **white**

• Color records referenced by registers **gray**

• Repeat until there are no gray records:
  ○ Pick a gray record, $r$
  ○ For each white record that $r$ points to, make it gray
  ○ Color $r$ **black**

• Deallocate all white records
Garbage Collection

All records are marked white
Garbage Collection

Mark records referenced by registers as gray
Garbage Collection

Need to pick a gray record

Red arrow indicates the chosen record
Garbage Collection

Mark white records referenced by chosen record as gray
Garbage Collection

Mark chosen record black
Garbage Collection

Start again: pick a gray record
Garbage Collection

No referenced records; mark black
Garbage Collection

Start again: pick a gray record
Garbage Collection

Mark white records referenced by chosen record as gray
Garbage Collection

Mark chosen record black
Garbage Collection

Start again: pick a gray record
Garbage Collection

No referenced white records; mark black
Garbage Collection

No more gray records; deallocate white records

Cycles *do not* break garbage collection
Two-Space Copying Collectors

A **two-space** copying collector compacts memory as it collects, making allocation easier.

**Allocator:**

- Partitions memory into **to-space** and **from-space**
- Allocates only in **to-space**

**Collector:**

- Starts by swapping **to-space** and **from-space**
- Coloring gray ⇒ copy from **from-space** to **to-space**
- Choosing a gray record ⇒ walk once though the new **to-space**, update pointers
Two-Space Collection

Left = from-space
Right = to-space
Two-Space Collection

Mark gray = copy and leave forward address
Two-Space Collection

Choose gray by walking through to-space
Two-Space Collection

Mark referenced as gray
Two-Space Collection

Mark black = move gray-choosing arrow
Two-Space Collection

Nothing to color gray; increment the arrow
Two-Space Collection

Color referenced record gray
Two-Space Collection

Increment the gray-choosing arrow
Two-Space Collection

Referenced is already copied, use forwarding address
Two-Space Collection

Choosing arrow reaches the end of to-space: done
Two-Space Collection

Right = from-space
Left = to-space
Two-Space Collection on Vectors

• Everything is a number:
  ○ Some numbers are immediate integers
  ○ Some numbers are pointers

• An allocated record in memory starts with a tag, followed by a sequence of pointers and immediate integers
  ○ The tag describes the shape
Two-Space Vector Example

• 26-byte memory (13 bytes for each space), 2 registers
  ○ Tag 1: one integer
  ○ Tag 2: one pointer
  ○ Tag 3: one integer, then one pointer

Register 1: 7                  Register 2: 0

From:  1  75  2  0  3  2  10  3  2  2  3  1  4
Two-Space Vector Example

• 26-byte memory (13 bytes for each space), 2 registers
  ○ Tag 1: one integer
  ○ Tag 2: one pointer
  ○ Tag 3: one integer, then one pointer

Register 1: 7       Register 2: 0

From:  1  75  2  0  3  2  10  3  2  2  3  1  4
Addr:  00  01  02  03  04  05  06  07  08  09  10  11  12
Two-Space Vector Example

• 26-byte memory (13 bytes for each space), 2 registers
  ○ Tag 1: one integer
  ○ Tag 2: one pointer
  ○ Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 7</th>
<th>Register 2: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: 1 75 2 0 3 2 10 3 2 2 3 1 4</td>
<td>Addr: 00 01 02 03 04 05 06 07 08 09 10 11 12</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>From:</th>
<th>1</th>
<th>75</th>
<th>2</th>
<th>0</th>
<th>3</th>
<th>2</th>
<th>10</th>
<th>3</th>
<th>2</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addr:</td>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

^ ^ ^ ^ ^

| To: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

^
Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 0</th>
<th>Register 2: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: 1 75 2 0 3 2 10 99 0 2 3 1 4</td>
<td></td>
</tr>
<tr>
<td>Addr: 00 01 02 03 04 05 06 07 08 09 10 11 12</td>
<td></td>
</tr>
<tr>
<td>^ ^ ^ ^ ^</td>
<td></td>
</tr>
<tr>
<td>To: 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

arrow
## Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 0</th>
<th>Register 2: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>Addr:</td>
</tr>
<tr>
<td>99 3 2 0 3 2 10 99 0 2 3 1 4</td>
<td>00 01 02 03 04 05 06 07 08 09 10 11 12</td>
</tr>
<tr>
<td></td>
<td>^ ^ ^ ^ ^ ^ ^</td>
</tr>
<tr>
<td>To:</td>
<td></td>
</tr>
<tr>
<td>3 2 2 1 75 0 0 0 0 0 0 0 0 0</td>
<td>^</td>
</tr>
</tbody>
</table>
Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 0</th>
<th>Register 2: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: 99 3 99 5 3 2 10 99 0 2 3 1 4</td>
<td>Addr: 00 01 02 03 04 05 06 07 08 09 10 11 12</td>
</tr>
<tr>
<td></td>
<td>^ ^ ^ ^ ^</td>
</tr>
<tr>
<td>To: 3 2 5 1 75 2 0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>
Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 0</th>
<th>Register 2: 3</th>
</tr>
</thead>
</table>

**From:**

<table>
<thead>
<tr>
<th>99</th>
<th>3</th>
<th>99</th>
<th>5</th>
<th>3</th>
<th>2</th>
<th>10</th>
<th>99</th>
<th>0</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>^</td>
<td>^</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Addr:**

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
<th>5</th>
<th>1</th>
<th>75</th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**To:**

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
<th>5</th>
<th>1</th>
<th>75</th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two-Space Vector Example

- 26-byte memory (13 bytes for each space), 2 registers
  - Tag 1: one integer
  - Tag 2: one pointer
  - Tag 3: one integer, then one pointer

<table>
<thead>
<tr>
<th>Register 1: 0</th>
<th>Register 2: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: 99 3 99 5 3 2 10 99 0 2 3 1 4</td>
<td></td>
</tr>
<tr>
<td>Addr: 00 01 02 03 04 05 06 07 08 09 10 11 12</td>
<td></td>
</tr>
<tr>
<td>^ ^ ^ ^ ^ ^</td>
<td></td>
</tr>
<tr>
<td>To: 3 2 5 1 75 2 3 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>^</td>
<td></td>
</tr>
</tbody>
</table>