MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall Semester, 1996

Lecture Notes, December 3 – Memory Management

List-Structured Memory

A good abstraction for memory hardware is a linear vector or storage locations with each location indicated by an "address" or "offset" from the start of memory, and with constant time access to any location of that memory:

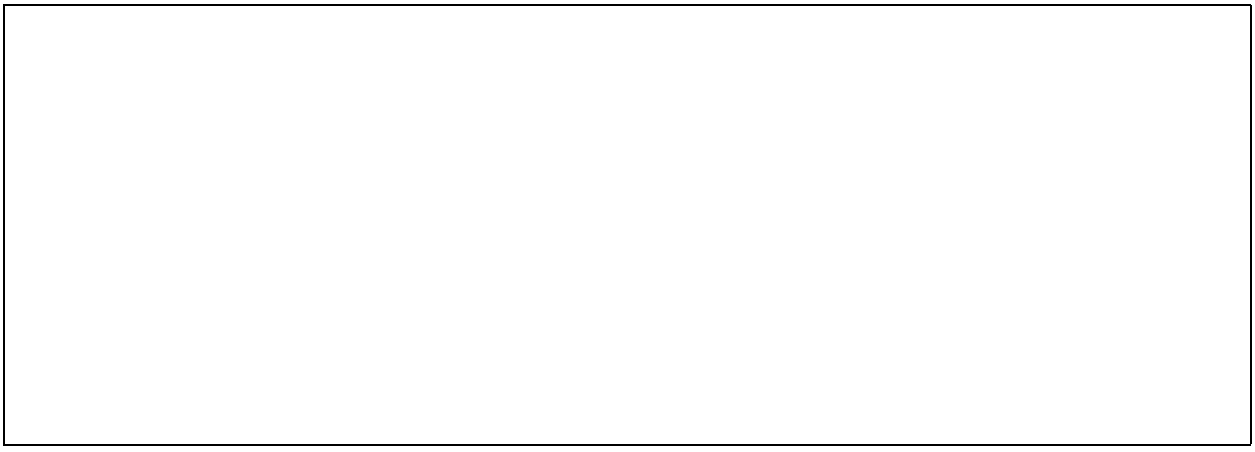
`(vector-ref` *<vector>* *<offset>*`)`
`(vector-set!` *<vector>* *<offset>* *<value>*`)`

The corresponding register machine primitives are

`(assign` *<reg-name>* `(op vector-ref) (reg` *<vector>*`)` *<offset>*`)`
`(perform (op vector-set!) (reg` *<vector>*`)` *<offset>* *<value>*`)`

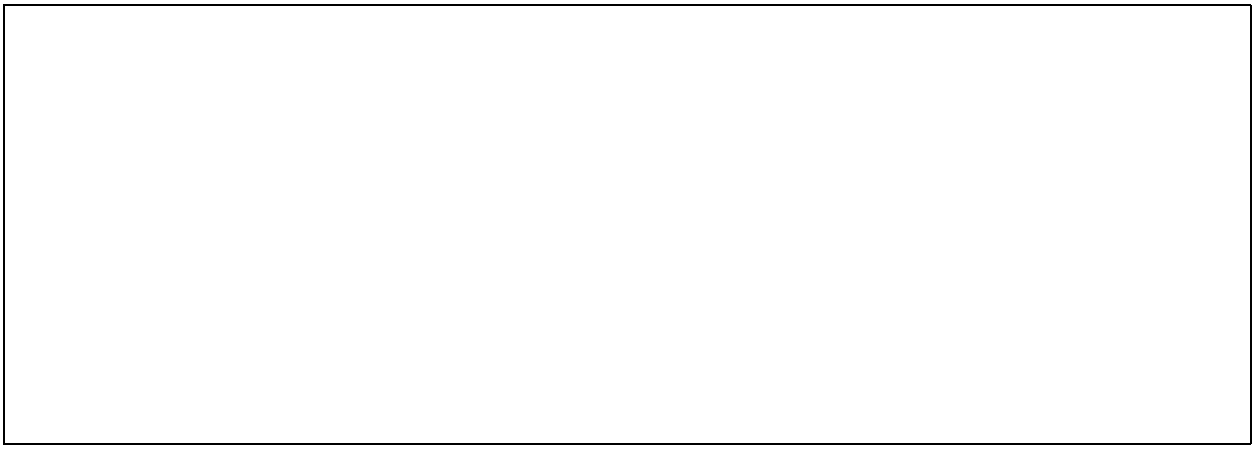
To implement cons cells, we use two special vectors, `the-cars` and `the-cdrs`. Below, copy a simple example of these vectors:

Our notation for typed pointers include the following:



An example to put in our memory:

```
(define a '(1 2 3))
(define b (cons a a))
```



Register Machine Implementation of Pair Abstraction

To implement pairs, we need to replace the "higher level" abstractions we have been using for `cons`, `car`, and `cdr` with the available vector-oriented machinery. For example,

`(assign` *<reg1>* `(op car) (reg` *<reg2>*`))`

becomes

`(assign` *<reg1>* `(op vector-ref) (reg the-cars) (reg` *<reg2>*`))`

Similarly,

`(assign` *<reg1>* `(op cdr) (reg` *<reg2>*`))`

becomes

`(assign` *<reg1>* `(op vector-ref) (reg the-cdrs) (reg` *<reg2>*`))`

Allocation of a cons cell depends upon some additional conventions, specifically about where to find free or available cons cells. Here we assume that `free` points to the first free location in memory, and that everything else below `free` is also available for use. Thus, `cons` can be implemented as in this example:

`(assign` *<reg1>* `(op cons) (reg` *<reg2>*`) (const` *<value>*`))`

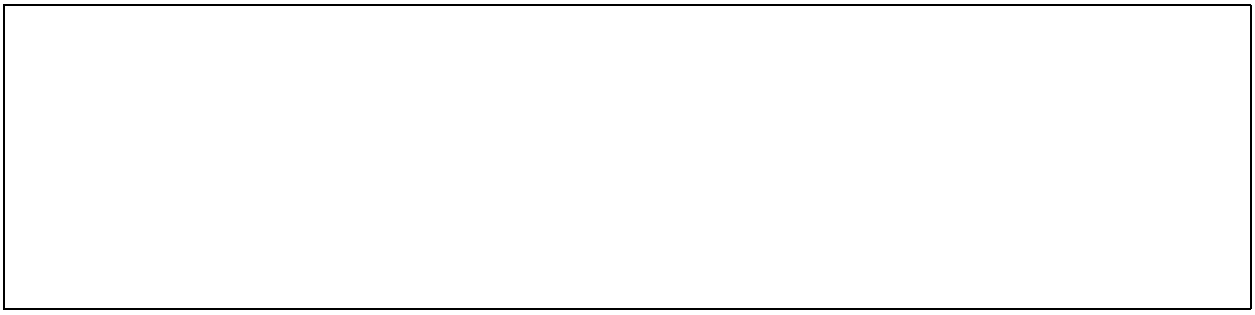
becomes

```
(perform (op vector-set!) (reg the-cars) (reg free) (reg <reg2>))
(perform (op vector-set!) (reg the-cdrs) (reg free) (const <value>))
(assign <reg1> (reg free))
(assign free (op +) (reg free) (const 1))
```

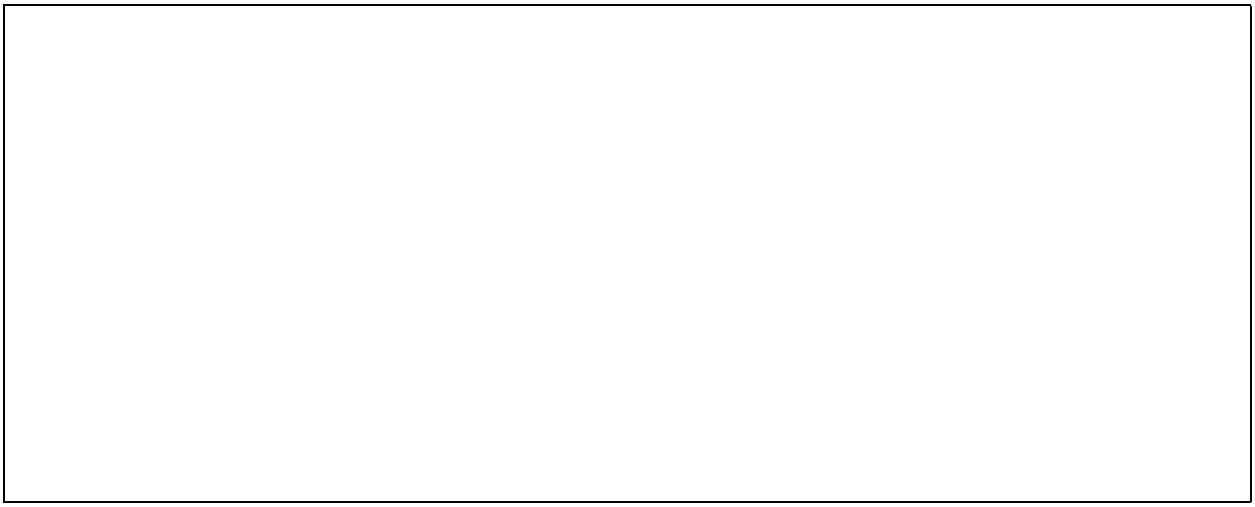
Garbage Collection

Method 1: Reference Count

The basic idea is:

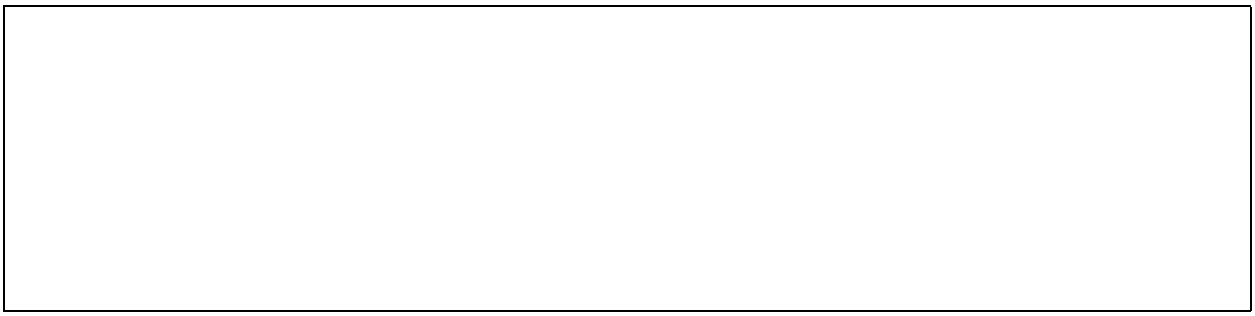


An example:

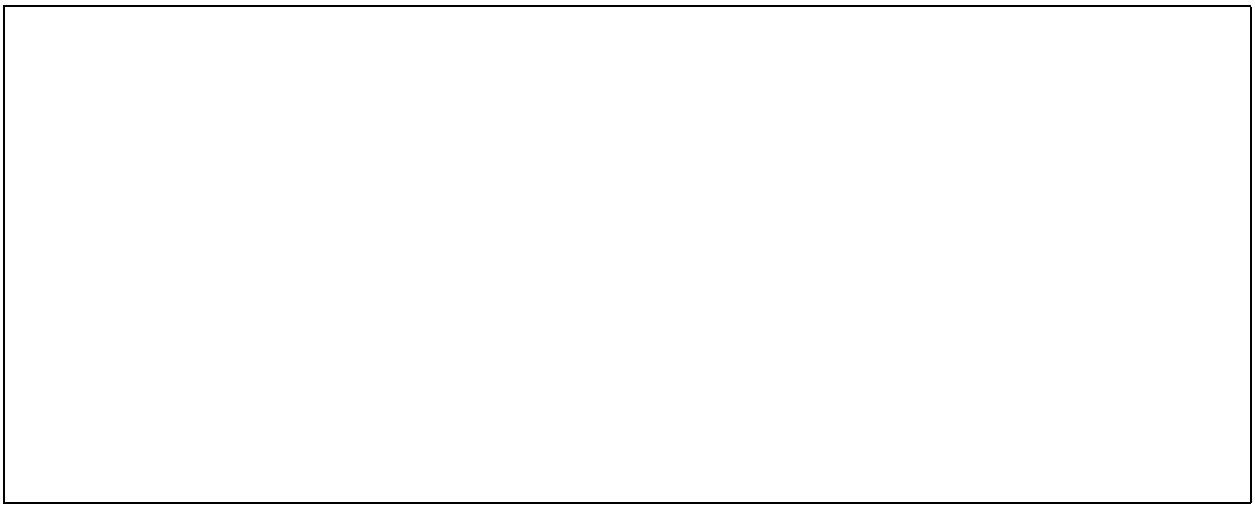


Method 2: Mark-Sweep

The basic idea is:

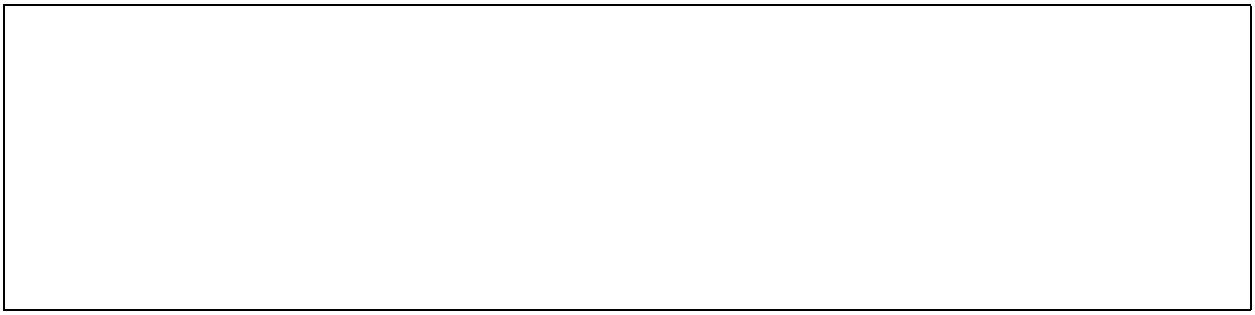


An example:



Method 3: Stop and Copy

The basic idea is:



An example: