SECTION 9

Storage Management

9.1 INTRODUCTION

This section discusses Explorer storage management in Areas and Regions. Garbage Collection (GC) and its special spaces are covered in the section on Garbage Collection.

Storage allocation for Lisp objects is implemented on top of a large uniform address space provided by the Virtual Memory System. The collection of all Lisp objects is known as the Lisp Object Space. Storage Allocation and Garbage Collection manage the mapping of Lisp Object Space to the virtual address space. Both storage allocation and garbage collection are logically above the virtual memory system. The virtual memory system does not understand and therefore cannot assist in a meaningful way in the allocation of address space to Lisp objects.

9.1.1 Areas and Regions.

The storage allocation system manages the address space by breaking it down into smaller segments called _regions_. A region is the smallest quantum of address space managed by the address space management system. Regions in turn belong to _areas_.

Areas are created by explicit commands (see MAKE-AREA). Area creation merely _defines_ the abstract area entity, whose regions are intended to contain objects related in some sense (such as all local objects used by a process). One initial region is created when the area is first made.

9.1.2 Address Space Allocation and Use. Virtual address space is _allocated_ to regions when they are created. The allocation a region is defined by the region's _origin_ (starting virtual address) and its length. Actual _occupation_ (use) of virtual address space by objects takes place in regions as storage requests are made by the various storage allocation primitives. The general term _consing_ is used to refer to any such allocation of storage, whether to actual cons cells or to other objects.

Storage is assigned to objects in a linear fashion starting at the origin address of a region. The subset of region locations occupied by objects is defined by the region's <u>free pointer</u>, which is the next offset at which an object can be given storage. All locations before this pointer have been used.

9.2 AREAS

The logical address space is divided into areas. An area defines a set of attributes on the virtual address space that it contains. While an area doesn't really have any virtual address space assigned directly to it, it does contain one or more regions which do. An area is identified by its area number, an integer between 0 and 255. The area number is used as an index into the various <u>area descriptor tables</u> described in Table 9-1 to obtain the appropriate characteristic of the area.

Table 9-1 Area Attributes

AREA-NAME	A symbol representing the name of the area.
AREA-REGION-LIST	The region number of the first region in this area.
AREA-REGION-BITS	A template for the REGION-BITS word for a region allocated in this area.
AREA-REGION-SIZE	The default size of this area's oldest regions.
AREA-MAXIMUM-SIZE	The maximum size this area is allowed to occupy or *MOST-POSITIVE-FIXNUM* if unlimited.

A further series of attributes are defined by the AREA-REGION-BITS word. When a new region is created it inherits the attributes of the area to which it belongs by using the AREA-REGION-BITS word as a template. These attributes will be described in detail in the paragraph on regions.

Logically, the <u>area descriptor table</u> is a table of 5-word entries as shown above. In reality, the table exists as five separate tables indexed by the area number, each table corresponding to one of the five words. This makes it easy for the microcode to index into the table and makes the code fairly insensitive to changes in the entry size.

Areas can be created by user commands. The area in which general consing occurs can also be controlled from Lisp (see the Storage Management chapter of the <u>Explorer Lisp Reference</u> manual). A program may use this feature to allocate related items in a contiguous portion of the virtual address space. This has the effect of increasing "locality of reference" on these data items, which can improve virtual memory paging performance.

9.3 REGIONS

A region is a block of contiguous virtual address space which is some multiple of the <u>Address Space Quantum Size</u> of 32 pages. When all the address space assigned to a region has been used by objects, new regions are automatically created as they are needed. The amount of address space allocated to an area's regions is generally unlimited, unless a particular <u>area maximum size</u> quantity was specified when the area was created.

Each region is identified by a number from 0 to 2047. Like an area number, this number is used as an index into any of the <u>region descriptor tables</u> in order to look up information about the region. The region characteristics found in these tables are summarized in Table 9-2.

Table 9-2 Region Characteristics

REGION-ORIGIN	Starting virtual address of the region.
REGION-LENGTH	The total amount of virtual address space assigned to this region in words.
REGION-FREE-POINTER	Offset into this region of the next free word that can be allocated.
REGION-GC-POINTER	Offset into this region of the next object which needs to be scavenged.
REGION-BITS	Defines various attributes of this region.
REGION-LIST-THREAD	Region number of the next region in this area's list of regions.

The address space allocated to this region are defined by its origin and its length characteristics. A region's allocation is only available for use by objects specifically created in this region's area; that is the address space cannot be used by just any storage request in general. The portion of a region's allocation actually used by objects occurs before the free pointer; the virtual address of the next free can be calculated by adding the region free pointer to the origin address.

All the regions in an area are linked together in a list. The list is anchored in the AREA-REGION-LIST entry for the area number. The next region in the list is found in the REGION-LIST-THREAD entry indexed by the first region; the third region is in the REGION-LIST-THREAD of the second entry; and so forth until a region number which is a negative number is encountered.

The free regions are also linked into a list. This is needed since when the garbage collector "flips" it frees a set of regions. Storage allocation needs to be able to find a free region easily.

9.3.1 Region Bits.

The REGION-BITS word defines a number of important attributes of the region. The fields within the region bits word are shown in Figure 9-1 and are discussed here.

```
 31 29   25 24       20  18   16  14    12    9 8 7  5 4 3 2    0
+----------+----------+----+-+------+-+--------+-+-+--+-+-+------+
|CC|DTP-FIX|  Status  |Rep |O| GEN  |U| Type   |S|V|VO|C|R| Swap |
+----------+----------+----+-+------+-+--------+-+-+--+-+-+------+
```

CC:	CDR Code
Res:	Reserved, unused
Status:	Access and status bits to be used in the hardware virtual memory map
Rep:	Representation type 0 = list 1 = structure 2,3 unused
O:	Oldspace meta bit. 0 = old space or free 1 = new space, static space or fixed
GEN:	Region generation
U:	Region usage (for future expansion)
Type:	Space type (see space type code table)
S:	Scavenger Enable. Value: 1 = Scavenger can touch this area.
V:	Region zero volatility lock
VO:	Region volatility
C:	Cache Inhibit (Explorer II only)
R:	Reserved, unused
Swap:	Number of pages the Virtual Memory System should try to swap at a time (Explorer II only)

Figure 9-1 Region Bits Area Entry Description

A regions can store one of two types of data: list objects or structure objects. A structure object is any object other than a list. This is termed a region's <u>representation type</u>

The Oldspace meta bit is a bit that is set to zero in the first-level hardware maps for this region when the region is flipped to Oldspace. The map bit can then be used to efficiently implement the Garbage Collector's Oldspace Read Barrier.

A region's generation, volatility, and zero volatility lock properties are <u>temporal</u> attributes used to support the Temporal Garbage Collection algorithm. These are described in the section on Garbage Collection.

The scavenge enable bit is set when this region is part of the Scavenge Space defined by the garbage collector for a collection.

The cache-inhibit bit and the swapin quantum field are used in the Explorer II only. The cache-inhibit means that the virtual memory cache should be disabled for all pages in this region. The swapin quantum is the log (base 2) of the number of pages to try to swap in simultaneously on virtual memory reads in this region. A value of 3, for example, means try to swap in up to 8 (2**3) pages.

The space type attribute defines the storage allocation scheme that is used. The encoding of this field is shown in Table 9-3.

Table 9-3 Space Type Codes

Code	Region Type
0	Free
1	Oldspace
2	Newspace
3-8	Unused
9	Static
10	Fixed (static, not growable, no consing)
11	Extra PDL
12	Copyspace
13	Reserved for future use
14-15	Unused

9.4 STANDARD AREAS

When a machine comes up after a cold boot there are 25 FIXED areas reserved for use by the system itself. The first 11 areas are permanently wired down (not allowed to be swapped out by the virtual memory system) because they are either referenced heavily by the microcode or are referenced at or below the level of the virtual memory system. The system parameters file (LROY-QCOM) specifies these areas and their sizes. As described above, an area is indentified by a unique area number. The assignment of these area numbers for the standard areas is made by the LROY-QCOM file.

The fixed areas each contain a single region. They are special areas in that their regions may be smaller than the minimum region quantum size of 32 pages. This requires that they be specially handled by the storage allocation mapping system (see the discussion of the Address Space Map Area below).

The standard areas are as follows:

* Resident Symbol Area (wired) - Contains T and NIL symbols. This area's single region currently always starts at virtual address 0.

* System Communication Area (wired) - This area contains various values used by I/O routines and systems utilities. See discussion of SCA below.

* Scratch Pad Init Area (wired, read only) - Micro code variables are loaded into this area at startup.

* Micro Code Link Area (wired, read only) - Contains the microcode entry points for the miscellaneous operations.

* Region Origin Area (wired) - Contains the starting address for each region, indexed by region number.

* Region Length Area (wired) - Contains the length of each region, indexed by region number.

* Region Bits Area (wired) - Contains the region bits information for each region, indexed by region number.

* Region Free Pointer Area (wired) - Contains the region free pointer for each region, indexed by region number.

* Device Descriptor Area (wired) - Contains device descriptors for the I/O system.

* Disk Page Map Area (wired) - Contains the Disk Page Map Table for the Virtual Memory System

* Address Space Map Area (wired) - Contains the Address Space Map. The Address Space Map is a table indexed by the virtual address quantum and indicates the region number of the virtual address. If the region number in the address space map is zero, then either the virtual address has not been allocated to a region or the virtual address belongs to a fixed area. When a zero is found in the Address Space Map the fixed areas are searched to determine which area contains the virtual address. The region number is then determined from the area number, since for fixed areas the region number and area number are the same.

* Region GC Pointer Area (fixed) - Contains the GC pointer information for each region, indexed by region number.

* Region List Thread Area (fixed) - Contains the list thread for each region, indexed by region number.

* Area Name Area (fixed) - Contains the name of each area, indexed by area number.

* Area Region List Area (fixed) - Contains the first region number in each area, indexed by area number.

* Area Region Bits Area (fixed) - Contains the Region Bits word for each area, indexed by area number.

* Area Region Size Area (fixed) - Contains the default region size for each area, indexed by area number.

* Area Maximum Size (fixed) - Contains the maximum size for each area, indexed by area number.

* Support Entry Vector (fixed, read only) - Contains Lisp functions which are callable by microcode.

* Extra PDL Area (fixed) - The Extra PDL Area, or number consing area, is used to reduce the garbage generated when evaluating arithmetic expressions. All bignums and floating point numbers are first consed in the Extra PDL Area. Pointers into the Extra PDL Area are only allowed "in the machine" (see the chapter on garbage collection for a description of the parts of the processor that are "in the machine"). Before a pointer is written into main memory, a check is made to see if the pointer points into the Extra PDL Area. If the pointer being written points into the Extra PDL Area, then the object is copied out of the Extra PDL Area into the default consing area and the pointer is modified to reflect the

number's new address.

When the Extra PDL Area is full, all of the pointers in the machine are checked to see if they point into the Extra PDL Area. If a pointer into the Extra PDL Area is found, then the object is copied out of the Extra PDL Area into the default consing area and the pointer is replaced by a pointer to the copy. When there are no more pointers in the machine that point into the Extra PDL Area, then the Extra PDL Area contains only garbage. The address space is then reclaimed by setting the free pointer for each region in the Extra PDL Area to zero (currently there is exactly one region in the Extra PDL Area).

- Microcode Entry Area (fixed) - Contains indices into the Microcode Link Area for each microcoded function (see discussion of DTP-U-ENTRY in chapter on data types).

- Microcode Entry Debug Info Area (fixed) - Micro entry address or locative indirect micro-code-symbol-area.

- Scavenger State Area (fixed) - This single region area contains the stack used by garbage collection depth-first scavenging.

- Linear PDL Area (fixed) - The Linear PDL Area contains the Linear PDL (Push Down List, or stack) for the initial process. The Linear PDL (usually just called PDL) is the runtime stack for the process. The currently executing process will have the top part of its PDL cached in the processor's PDL Buffer.

Any memory reference to this area results in a page fault so that the virtual memory system can check if the target of the memory reference is really in the PDL Buffer.

- Linear Bind PDL Area (fixed) - contains the Special PDL for the initial process.

- Working Storage Area - The default cons area; most objects created by users are created in this area.

- Permanent Storage Area - Permanent data structures are placed here.

- Property List Area - Contains the property lists for symbols.

- Print Name String Area - Contains the print names for symbols.

- Control Tables Area

* Non-Resident Symbol Area - Contains most of the symbols in the kernel.

* Macro Compiled Program Area (read only) - Contains all compiled functions.

  In addition, any constant objects, such as lists, are also loaded into this area. This causes naive users to get mysterious error messages about trying to write in a read-only area when they try to do destructive operations (such as RPLACA) on constant objects in compiled functions.

* PDL Area - contains the linear PDL, except for the one used by the initial process which is in wired memory.

* SG And Bind PDL Area - The SG and Bind PDL Area contains the stack groups and the Special, or Binding, PDL for each process. The Special PDL contains the variable binding information for a process.

* Indirection Cell Area - This area is always pointed to by DTP-GC-YOUNG-POINTER's. It is used to keep track of pointers from older to younger generations.

* FASL Table Area - The FASL Table is placed here at load time.

* FASL Temp Area - Temporary structures created by the Loader are placed here.

* Debug Info Area - Contains debug info structures and fields.

9.5 SYSTEM COMMUNICATION AREA

The Systems Communication Area contains miscellaneous words that are needed for basic operation and do not rely on the rest of the machine operating. This information is shared by the microcode and Lisp. The file SYS:UCODE;LROY-QCOM contains the definitions of items in this area. The Systems Communication Area is wired and at the fixed address of 1000 (octal).

A map of systems communications areas is shown in Figure 9-2.

Octal
Addresses

1000-1046 : Miscellaneous words
1047-1577 : Not used
1600- : swap-in-rqb-origin

Figure 9-2 Map of Systems Communications Area

The miscellaneous words (1000 - 1046) are:

1. Area Origin Pointer - Virtual address of the Area Origin Area, which lists the starting virtual address of all fixed areas.
2. Valid Size - Number of words used in a saved band.
3. Object Array Pointer - Unused Unused
4. Ether Free List - Ethernet interrupt-handler variable.
5. Ether Transmit List - Ethernet interrupt-handler variable.
6. Ether Receive List - Ethernet interrupt-handler variable.
7. Band Format - Encodes format number in a saved band: 2000 -> new compressed format, otherwise old expanded format.
8. GC Generation Number - Reserved for value of %GC-GENERATION-NUMBER
9. Device Interrupt Table - Points to the Device Interrupt Table.
10. Temporary - Microcode bashes this at extra-pdl-purge.
11. Free Area Number List - Threaded through area-region-list, end=0.
12. Free Region Number List - Threaded through region-list-thread, end=0.
13. Memory Size - Number of words of main memory.
14. Wired Size - Words of low memory wired down; not all of these words are wired, this is really the virtual

address of the start of normal pageable memory.

15. Chaos Free List - Chaosnet interrupt-handler variable.

16. Chaos Transmit List - Chaosnet interrupt-handler variable.

17. Chaos Receive List - Chaosnet interrupt-handler variable.

18. Debugger Requests

19. Debugger Keep Alive

20. Debugger Data 1

21. Debugger Data 2

22. Major Version - Major version number of SYSTEM.

23. Desired Microcode Version - Microcode version this world expects. Note: this word may be stored with its data type field starting at bit 24 even though pointer fields are now 25 bits.

24. Highest Virtual Address - (Note: Should have this much room in the paging partition)

25. Pointer Width - 25

26. Descriptor Space Free Pointer - Current allocation pointer in the Device-Descriptor-Area.

27. Page Device Table - Unused

28. System Nupi Descriptor - Pointer to descriptor for system nupi

29. Processor Slot - Ucode stores A-SLOT-IM-IN here.

30. Overtemp Event - SIB will post overtemp events here.

31. Fiber Optic Warning Event - Microcode posts fiber optic warning event her.

32. Nupi Overtemp Event - NUPI Special event: microcode will post overheat special event here with formatter number embedded in bits <3:5> and non-zero value in bits <2:0>.

33. Physical Memory Map - Pointer to a memory table of memory board addresses in A-Memory.

34. Keybd Error Event - Keybd error condition; log of possible flaky keyboards.

35. Disk Retry Event - Disk retry condition; log of successful retries after disk errors.

36. Unused SIB Event - Place this address in all the unused SIB event locations

37. Parity Error Event - NuBus parity error; high 8 bits (0-7) with bit 8 on.

38. Parity Error Event 2 - NuBus parity error; part 2 - low 24 bits.

39. Syslog Wrap Event - System log wrap around event.