GC Overview:
February 6, 1986
Jan Pedersen & Greg Nuyens

Reference counts are not contiguous with their objects, but rather are kept in three tables: \HTMAIN (the hashed main ref count table), \HTCOLL (the collision table), and \HTBIGCOUNT (the big ref count table). \HTMAIN is a hash table, where the hash function is based on the address of the object, while \HTCOLL and \HTBIGCOUNT handle two aspects of hash table overflow.

Since most (?) objects have refcount 1 (e.g. cons cells of a list), they are not explicitly represented; if an object (of a refcountable type) is not present in the tables, its refcount is 1.

If several objects hash to the same entry in \HTMAIN, then the entries are kept in a linked list of ref count entries in \HTCOLL.

If an object has a refcount equal to \MAXHTCNT (63), its refcount is stored in a "big" refcount table (\HTBIGCOUNT).

The Lisp function \HTFIND can handle all cases and is the punt function for the opcode (GCREF), which handles only the simplest case of no collision and no big refcount. Many opcodes call GCREF as a microcode subroutine and if GCREF punts, an entry is made in the table \HTOVERFLOW. Before opcode completion, the microcode calls the Lisp function \GC.HANDLEOVERFLOW, which processes \HTOVERFLOW by explicitly calling \HTFIND on each entry.

\HTMAIN is a locked down table of 32K word sized entries (or 64K bytes).

\HTCOLL is a paged table of 32K double word sized entries (or 128K bytes), where the first word in a pair is a ref count entry and the second is a 16 bit offset to the next entry in the chain.

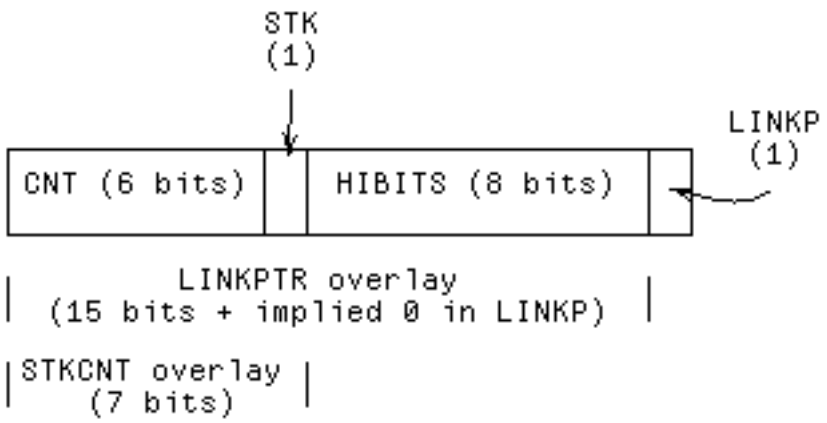
\HTBIGCOUNT is a linearly searched table of big ref count entries; new pages are allocated as needed.

Hashing:

Given an address, the hash function computes a 15 bit offset into \HTMAIN by logically shifting the low 16 bits of the address right one bit.

GC record structure:

Each entry in \HTMAIN is a word (16 bits) long, and is a record of type GC. Its structure is the following:



REFCOUNT TABLE ENTRY

CNT is the refcount (0 to 63, excluding 1)
STK is on if this object is referenced from the stack.
HIBITS contains the top 8 bits of the PTR address of the object (the offset at which you found this entry gives you the lower 15 bits (only even objects are refcounted)).
LINKP is on if this entry is a pointer to the collision table.

Organization of \HTCOLL:

\HTCOLL points to a double word entry, the first word of which is a 16 bit offset from \HTCOLL where the first entry on the linked free list may be found. The secord word is a 16 bit offset from \HTCOLL where the first free double word in sequential storage may be found.

At startup the first word is zero, indicating an empty free list and the second word is four, indicating that the first free double word in sequential storage is at offset four from the top of \HTCOLL.

A new entry is allocated by first looking at the free list. If the free list is empty, a new entry is allocated from sequential storage.

It is an error to allocate double words from sequential storage at offsets greater than or equal to \HTCOLLTHRESHOLD (65528 or 177770Q) -- if this occurs the garbage collector is turned off.

The double word entries in \HTCOLL are similar to those in \HTMAIN in that the first word in a pair is an instance of record type GC. The adjacent word is a 16 bit link offset (field name NXTPTR) from the top of \HTCOLL to the next double word entry in this collision chain. A NXTPTR value of zero indicates end-of-chain.

Algorithm:

\HTFIND is called with two arguments, PTR (a pointer to the object) and CASE (oneof \ADDREFCASE, \DELREFCASE, \SCANREFCASE or \UNSCANREFCASE). The \SCANREFCASE means mark the object as referenced on the stack, likewise \UNSCANREFCASE. \HTFIND returns PTR if the result is zero ref count, else NIL.

First, \HTFIND hashes the address into a 15 bit offset from the top of \HTMAIN, pointing at a refcount table entry. If the resulting entry is empty (all bits 0) then the macro .NEWENTRY. handles it.

.NEWENTRY. does the following:
updates the HIBITS.
and by case does:

- \ADDREFCASE sets CNT to 2 (since its absence from the table implied previous refcount 1)
- \DELREFCASE leaves the CNT at 0 and returns the PTR
- \SCANREFCASE Sets count to 1 and sets the STK bit.

If the entry from the hash has the same HIBITS then this main table entry is the entry for PTR, so the macro .MODENTRY. handles it.

.MODENTRY. does the following:
if CNT is \MAXHTCNT, then this entry is superceded by an entry in the big refcount table (\HTBIGCOUNT), so call \GC.MODIFY.BIGREFCNT.

Otherwise, by refcount case:

- \ADDREFCASE if incrementing CNT makes it equal to \MAXHTCNT, then call \GC.ENTER.BIGREFCNT, otherwise just increment CNT.
- \DELREFCASE decrements CNT
- \SCANREFCASE sets STK.
- \UNSCANREFCASE clears STK.

If resulting refcount is 1and no STK (STKCNT = 2), return T signalling removal.

If the HIBITS don't match, then a new collison has occured. The collision resolution is basically to construct a linked list of refcount table entries out of cells from \HTCOLL

New collision:
-- Get two double word entries (LINK and PREV) from \HTCOLL by calling the .GETLINK. macro.
-- Make the ref count contents of PREV equivalent to the ref count contents of the entry in \HTMAIN and link PREV to the next double word entry, LINK.
-- Smash the \HTMAIN entry so that the LINKPTR overlay is an offset to PREV.
-- Mark LINK as empty, and the end of a chain.
-- proceed with LINK as an empty entry (.NEWENTRY.)

where .GETLINK. does the following
-- fetch the FREEPTR field of \HTCOLL (should be an offset to the front of the free list).
-- if FREEPTR is non-zero, fetch the first free cell and update the FREEPTR field of \HTCOLL
-- else fetch the NEXTFREE of \HTCOLL (should be an offset to the first free double word in sequential storage). If NEXTFREE is not GEQ to \HTCOLLTHRESHOLD, allocate that double word and increment NEXTFREE by 2.

If the entry from the hash has LINKP set, then a collision has already occurred and the entry is a pointer to a chain in the collision table. The LINKPTR overlay is used as a (16 bit) offset into \HTCOLL (the first 15 bits of the entry are used as is, with the last bit (LINKP) masked to zero). Note that the LINKP test should occur before testing HIBITS, but we place the explanation here since this case is more complex than a new collision.

So LINKPTR is an offset to the beginning of a chain of double word entries in \HTCOLL. This chain is searched sequentially (by following NXTPTR offsets down the chain), resulting in either finding an entry for PTR, or hitting the end of the chain (NXTPTR offset equal to zero).

If an entry is found, then .MODENTRY. handles it as before except that if the refcount goes to zero the link is deleted from the chain (by .DELLINK.).

.DELLINK. takes three args, (LINK -- the link to be deleted, PREV the previous link in the chain (can be NIL), and ENTRY -- the \HTMAIN table entry)
-- If PREV, than update PREV to point to the link following LINK, else update ENTRY to point to the link following LINK.
-- Place LINK back on the free list (.FREELINK.)
-- If ENTRY now points to a chain one entry long, then update ENTRY to have the refcount table entry contents of the remaining link, and free the remaining link.

.FREELINK. takes one arg, the LINK cell to be freed
-- zero LINK
-- place LINK on free list of \HTCOLL by update the first word (field name FREEPTR) of \HTCOLL

If no entry in the chain is found for PTR, then allocate an entry (.GETLINK.) and put it on the end of the chain. Treat the new LINK as an empty entry (.NEWENTRY.).

Stay tuned for further updates...