Information Processing in Japan Vol. 8, 1968

A Variable Length Storage Allocator and its Garbage Collector

YOSHIHIKO FUTAMURA*, KUNIKO FUTAMURA

This report describes a mechanism of storage management employed by a storage allocator to allocate variable length computer storages, as well as a mechanism of reclaiming used storages. As in LISP 1.5, this storage allocator divides the storage area into two sections: the free storage and the full word space. These two sections are each managed in different ways. Namely, in the full word space, the storages are relocated ("packed") when the used storages are collected. The method described in this report is advantageous for a computer with bit operation instructions to implement a symbol manipulation system with a mechanism of automatically reclaiming used storages such as LISP.

1. *Introduction*

It is advantageous to use multi-word items in list structures to attain savings not only in the amount of time required for list processing, but also in the storage required for the list structure [1]. When we were using a HITAC-5020 computer to implement HELP (Hitachi Experimental List Processor), a symbol manipulation system which is a dialect of LISP 1.5 [2], we decided to use multi-word blocks as the list elements containing BCD character strings or numbers. Consequently, the storage allocator must have the ability to allocate variable length storages and to reclaim used multi-word blocks. This report describes the methods of storage allocation used in this system.

The atom and number structure of HELP is shown in Fig. 1.

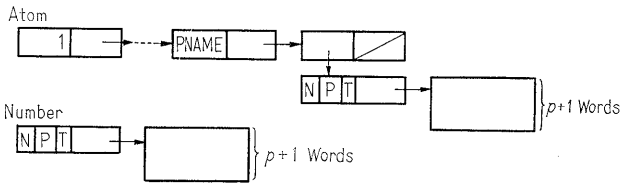

Fig. 1. Atom and number structure in HELP.

N is a 5-bit field; P is an 8-bit field; and T is a 3-bit field. (In the HITAC-5020, one computer word consists of 32 bits.) The content of N is a definite number indicating that the half-word (16-bit field) is not a pointer. The content of P is a number indicating the size of the block pointed to by the right half of the word. (In the diagram here, the contents of P are indicated by p, $0 \leqq p \leqq 255$.) The content of T is a number indicating the type of numerical value.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 8, No. 4 (1967), pp. 207~210.

* Section 7, Central Research Laboratory, Hitachi, Ltd.

This structure was based on the following concept. Data like BCD character strings or numbers, which may be more easily processed together in blocks than by analysis into a list, are stored as blocks. In making up these list structures, the HELP storage allocator allocates storages of variable lengths. In the same manner as in reference [2], the storage area is divided into two sections: a free storage for pointer words (computer words containing pointers) and a full word space for full words (computer words for blocks including BCD character strings or numbers). These two sections are each managed separately. If the storage area is exhausted during system execution, the HELP garbage collector is automatically called, and the used storages are collected so that they may be re-used. This operation is called "garbage collection". When the free storage is exhausted, the garbage collector will scan only the free storage to collect the used pointer words. Even though there may be an increased number of pointer words used to make up the list structure, it does not necessarily follow that there will be an increase in the number of full words used in the list structure. Therefore, in this case, there is no need to perform garbage collection in the full word space. However, when the full word space is exhausted, garbage collection is performed in both the free storage and full word space. When garbage collection is performed in the full word space, the blocks are relocated in order to eliminate gaps.

2. *Management of the Storage Area*

2.1. *Management of the Free Storage*

A bit table is provided corresponding to the free storage. There is a one-to-one correspondence between each word in the free storage and each bit in the bit table. When a word is available, its corresponding bit is set to 1. When a word is used, its corresponding bit is set to 0. (See Fig. 2.)

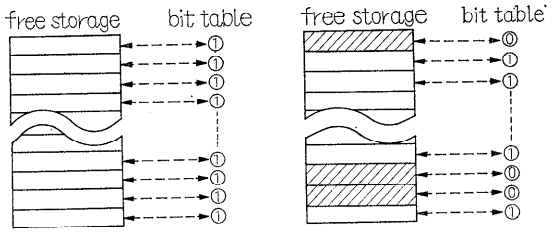

Fig. 2. Correspondence between free storage and bit table.
When all of the free storage words are available, all of the bits in the bit table are set to 1 (diagram on left). When a word is used, the corresponding bit is set to 0 (diagram on right). The shaded parts of the diagram indicate the used words. The bit table is located outside of the storage area but within the system area.

When a pointer word is required, the HELP storage allocator scans the bit table. If it finds a bit with value 1, it extracts the word corresponding to the first bit having the value 1. Then it changes the value of the bit to 0. The HITAC-5020 computer is equipped with a bit operation instruction (FTL) for finding the first bit with value 1 on the left side

in a 32-bit or a 64-bit field, and then changing its value to 0. Therefore, this method is more advantageous than the usual method of maintaining free storage linking all available storages in a list. If all of the bits in the bit table have a value 0, the garbage collector is called.

2.2. *Management of the Full Word Space*

A bit table is not provided corresponding to the full word space. Instead, a system register TAVB (Top of AVailable Block) is used to point to the first available word in the full word space. If a block of n words is required to contain a BCD character string or a number, the HELP storage allocator will extract n words, beginning with the word indicated

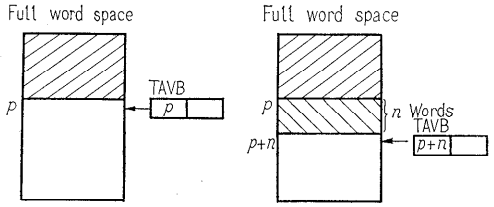

Fig. 3. Full word space and TAVB.
TAVB points to the first available word in the full word space. When n words are used, the state shown in the diagram on the left changes to that shown in the diagram on the right.

by the system register TAVB. Then the contents of TAVB will be increased by n. (See Fig. 3.) However, if the full word space does not have the required number of available words, the garbage collector is called.

3. *The HELP Garbage Collector*

The following terms are to be used with the definitions given below:

Active word (or block): A word (or block) in the storage area which can be traced by pointers from the left half-words of the system registers Base 1, Base 2, ... Base l.

Used word (or block): A word (or block) in the storage area which is not an active word (or block).

3.1. *Garbage Collection When Free Storage Is Exhausted*

Step 1.

All the bits on the bit table are set to 1, and i is set to 1. The stack is set to empty. Then step 2 is performed.

Step 2.

If $i>l$, garbage collection is stopped. If $i\leq l$, step 3 is performed. In this case, the word indicated by the pointer of the left half-word of Base i (the left pointer) will be the word to be examined.

Step 3.

Let x be the word to be examined. When x is a pointer word, step 4 will be performed if the bit corresponding to x is 0. If the bit corresponding to x is 1, this bit will

be changed to 0 and step 5 will be performed, with x as the word to be examined.

If x is not a pointer word, step 4 will be performed.

Step 4.

If the stack is empty, $i+1$ is substituted for i, and step 2 is performed.

If the stack is not empty, the word to be examined will be the word which is pointed to by the right pointer (the contents of the right half-word) of the word at the top of the stack. The stack will be popped up, and step 3 will be performed.

Step 5.

Let y be the word to be examined. In this case, y is stacked and step 3 is performed, the word to be examined being the word indicated by the left pointer of y.

By means of steps 1 through 5, all of the bits corresponding to garbage (*i. e.*, used words) in the free storage are set to 1, and all of the bits corresponding to active words are set to 0.

3. 2. *Garbage Collection When Full Word Space Is Exhausted*

When the full word space is exhausted, the garbage collector will relocate only the active blocks, beginning from the first address in the full word space, in order to eliminate gaps. When this "packing" operation is completed, the system register TAVB will be set to point to the first available word. (See Fig. 4.)

When active blocks are being relocated, the garbage collector uses the work area (256 core words) and the drum area (taken in the drum and less than the full word space by 256 words.)

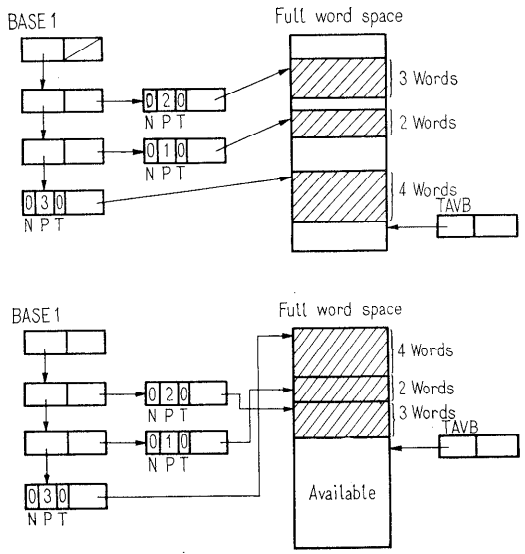

Fig. 4. State of the full word space before and after garbarge collection.

When garbage collection occurs in the state shown in the upper diagram, the state will change to the one shown in the lower diagram. (In this case, $l=1$.)

Step 1.

All of the bits on the bit table are set to 1, and i is set to 1. The stack is set to empty. The left pointer of the system register TAVB is set to the first address in the full word space. The work area and the drum area are both set to empty. Then step 2 is performed.

Step 2.

If $i>l$, the active contents of the drum area are loaded into the full word space, beginning from the top, and the active contents of the work area are transferred to the addresses immediately following. (See Fig. 5.)

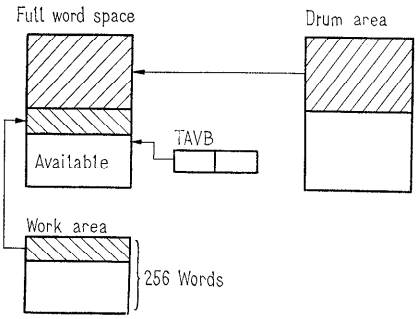

Fig. 5. Transfer of active full words.
The shaded parts of the drum area and of the work area (*i. e.*, the active full words) are transferred to the full word space as indicated by the arrow.

Then the left pointer of TAVB is set to the first available word, and garbage collection is stopped.

If $i\leq l$, step 3 is performed. In this case, the word to be examined is the word indicated by the left pointer of Base i.

Step 3.

Let x be the word to be examined. When x is a pointer word, step 4 will be performed if the bit corresponding to x is 0. If the bit corresponding to x is 1, this bit will be changed to 0 and step 5 will be performed, with x as the word to be examined.

If x is not a pointer word, step 4 will be performed.

Step 4.

If the stack is empty, $i+1$ will be substituted for i, and step 2 will be performed.

If the stack is not empty, the word to be examined will be the word which is indicated by the right pointer of the word at the top of the stack. The stack will be popped up, and step 3 will be performed.

Step 5.

Let y be the word to be examined. If the right pointer of y points to a full word, a block, the number of whose word is one more greater than that of the P-field of y, is selected. This block begins from the full word indicated by the pointer. It is packed into the first available space in the work area. The left pointer of TAVB is substituted for the

right pointer of y. Then the size of the block is added to the left pointer of TAVB, and step 4 is performed. If the work area has become full, its contents are transferred to the first available space in the drum area, and the next packing job is performed beginning from the top of the work area. (See Fig. 6.)

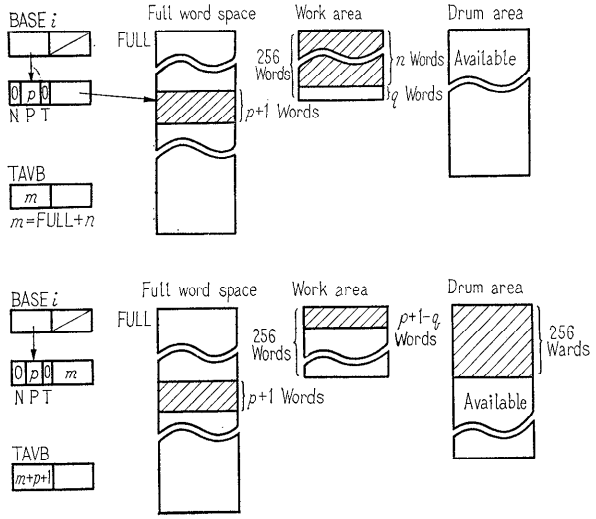

Fig. 6. Packing of blocks into the work area.
In the state shown in the upper diagram, when packing occurs when $p \geq q$, q words in the block are packed in the work area. Next, the contents of the work area are transferred to the drum area. Finally, the remaining $p+1-q$ words in the block are packed at the top of the work area. (Lower diagram)

If the right pointer of y does not point to a full word, y is stacked. Then step 3 is performed, the word to be examined being the word indicated by the left pointer of y.

By means of steps 1 through 5, all of the bits corresponding to active words are set to 0, and all of the bits corresponding to used words are set to 1. Furthermore, only the active blocks are packed in the full word space.

4. *Conclusion*

This storage allocator is used with HELP (Hitachi Experimental List Processor), a symbol manipulation system which is a dialect of LISP 1.5. Since a drum is used during garbage collection of the full word space, in the HITAC-5020 computer (core access time 2 μs, drum access time 10 ms) approximately 1 second is required for garbage collection of 9,216 words in free storage and 1,024 words in the full word space. (See Table 1.)

However, since almost all of the storages used for the execution of non-numerical computations consist of pointer words, garbage collection of the full word space rarely occurs in ordinary symbol manipulation problems. Consequently, even though slightly more time may be required in the garbage collection, the method described here is considered to be advan-

Table 1. Time required for garbage collection (GC).

Job No.	Number of executed CONS	Number of executed GC	Execution time of GC			Total time of job
			Minimum	Maximum	Mean	
1	85314	12(P)	0.4 sec	1.3 sec	0.6 sec	4 min 28.6 sec
2	190534	50(P)	1.1 sec	1.2 sec	1.2 sec	11 min 48.5 sec
3	81603	23(P), 1(F)	1.2 sec	2.1 sec(F) 1.9 sec(P)	2.1 sec(F) 1.6 sec(P)	9 min 9.4 sec
4	98728	11(P), 16(F)	0.2 sec	0.4 sec	0.3 sec	5 min 26.1 sec

Each of the P and the F in the table denotes that GC is caused by the exhausion of the free storage and the full word space, respectively.

tageous for the following two considerations:

(1) Savings can be made in the processing time (in input-output routines and arithmetical routines) of BCD character strings and numbers, and also in the storage required for them.

(2) Savings can be made in garbage collection time of the free storage.

Acknowledgment The authors are indebted to Dr. Shozo Shimada and Mr. Kazuma Yoshimura of the Hitachi Central Research Laboratory for their valuable advices in the implementation of this system.

References

[1] Comfort, W. T., Multiword list items, *Comm. ACM, 7*, 6 (June, 1964) 357-362.

[2] McCarthy, J., et al., *LISP 1.5 Programmer's manual*. Cambridge, Mass., M. I. T. Press, 1962.