MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Machine Structures Group
Memo No. 1

Memorandum MAC-M-137
January 31, 1964

A Machine Structure for Dynamic Storage Allocation

J. B. Dennis

The computing structure described in this note is proposed with the following objectives in mind.

1. Dynamic storage allocation should be possible without physical transfer of information within the main storage medium.

2. The storage allocation features should be applicable to the user's problem as well as to the multiprogramming problem.

3. The structure should be practical with available technology.

The proposed structure is based on the following concepts:

1. Blocks of main storage are renamed through a control memory, for example as in Atlas I.

2. Programs are divided into segments of instructions and data which are the allocatable entities in the system.

3. Linear addressing holds for words within segments and segments may have any length up to the entire main memory.

4. Segments containing instructions are coded as pure procedures.

The structure is illustrated in Fig. 1. For simplicity only one processor is shown, although the scheme is clearly applicable to multiprocessor configurations as well.

The segment index memory is a content addressable memory arranged to implement equality searches. It could be realized by parallel search hardware or in the form of a pseudo-associative memory as outlined below.

Main memory is partitioned into many blocks, each block associated with an entry in the segment index. In a large system there might be 1024 blocks of 64 words each and 1024 blocks of 1024 words each. There are a number of blocks per independently accessible unit of main memory.

In general, the processor accesses main memory by supplying the information indicated in Fig. 2. This includes

1. PP - processor program number
2. PS - processor segment number
3. PI - processor word index

The program number distinguishes among all active programs. The segment number identifies the segments referenced by any program. The word index specifies a word within the segment by the usual linear selection rule. The word index WI is further subdivided into

a) PB - processor block number
b) PW - processor word number

which indicate the block number within the specified segment and the local address within the block.

There is one entry in the segment index memory for each block of main store. Each entry (Fig. 3) includes the program number IP, segment number IS, and block number IB within the segment to which the hardware memory block MB associated with the index memory entry is allocated.

A memory reference by the processor causes [PP, PS, PB] to be used as a key for search to match [IP, IS, IB] in the segment iddex. The MB field of the matching entry, together with PW form the hardware memory address required. The absence of a match implies that the referenced information

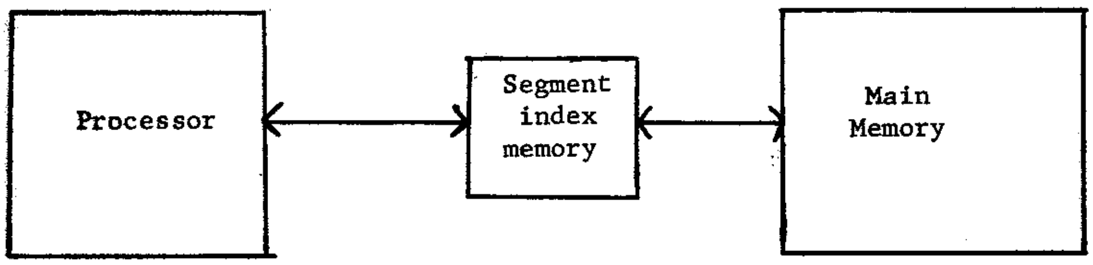

Fig. 1--Proposed machine structure

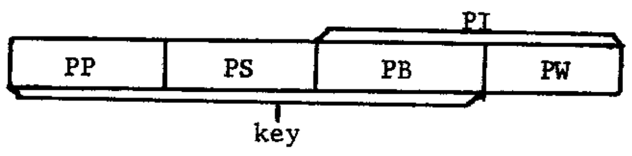

Fig. 2--Addressing data supplied by processor

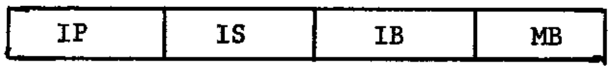

Fig. 3--Format of segment index entry

either does not exist, or is in auxiliary storage and a trap occurs to a supervisory routine.

To facilitate programming, the processor would be equipped with a program number register PN and several, say four, segment attachment registers, SAO, SA1, SA2, and SA3, as shown in Fig. 4. The PN register may be loaded only by a supervisory routine, and is the source of the PP data of a memory access request. The PS data originates from SAO for instruction fetch memory accesses, from SA1 for normal data accesses. For a certain class of data fetch and store instructions, and certain transfer and subroutine entry instructions, a tag in the instruction selects either SA2 or SA3 as designating the segment to be referenced. The PI data for a memory reference is an effective address generated in a conventional manner. The segment assignment registers consist of three parts, SN, BN, and HB. The SN field contains the number of the segment that it "attaches" to the processor, and is loaded by the action of a user's program. The BN field is loaded from PB whenever the processor uses the attachment register to make a memory reference. The HB field is loaded with the hardware block number associated with [SN, BN] when obtained from the index memory on the first reference to block BN of segment SN. Thus subsequent references to the same block do not require a look-up by the index memory.

With the proposed structure, memory becomes a rather fluid commodity which can be attached to or released from individual segments by a simple executive action. Segments may serve a variety of functions; they may serve as

1. large or small directly addressable blocks of data pr program that are entirely in main memory when active

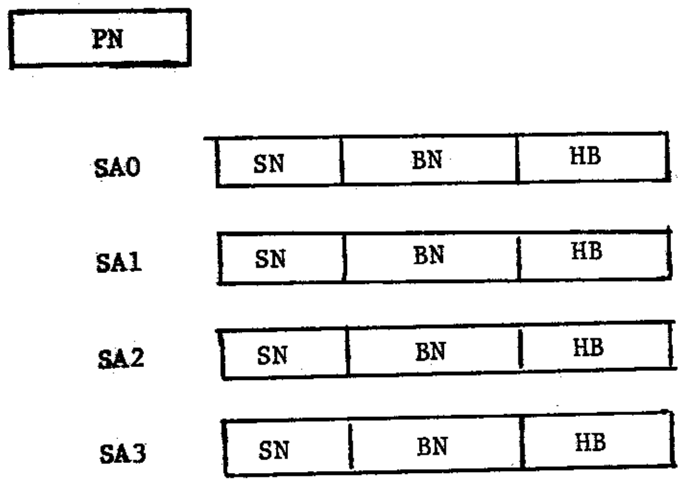

Fig. 4--Special processor registers

2. stacks always kept in main memory when active
3. push down lists in which depp entries are filed on auxiliary storage
4. ring buffers } which are mostly in auxiliary storage
5. files }

The realization of the segment index memory as a pseudo-associative memory is shown in Fig. 5. A standard core memory plus control logic is used. The addressing scheme uses a calculated ("hash") address to locate the origin of a chain of index memory entries having equal calculated addresses. The format of an entry in the index memory includes forward and backward pointers FP and BP to link the chain. This is done to facilitate the entry and deletion of index memory entries while maintaining the base entry of each chain at the calculated address. Analysis of this addressing technique shows that; with good statistics, the average number of memory cycles required for a search is 1.5. Preliminary simulation of the scheme has yielded results of 1.6 to 1.7 memory cycles for a 512 entry memory.

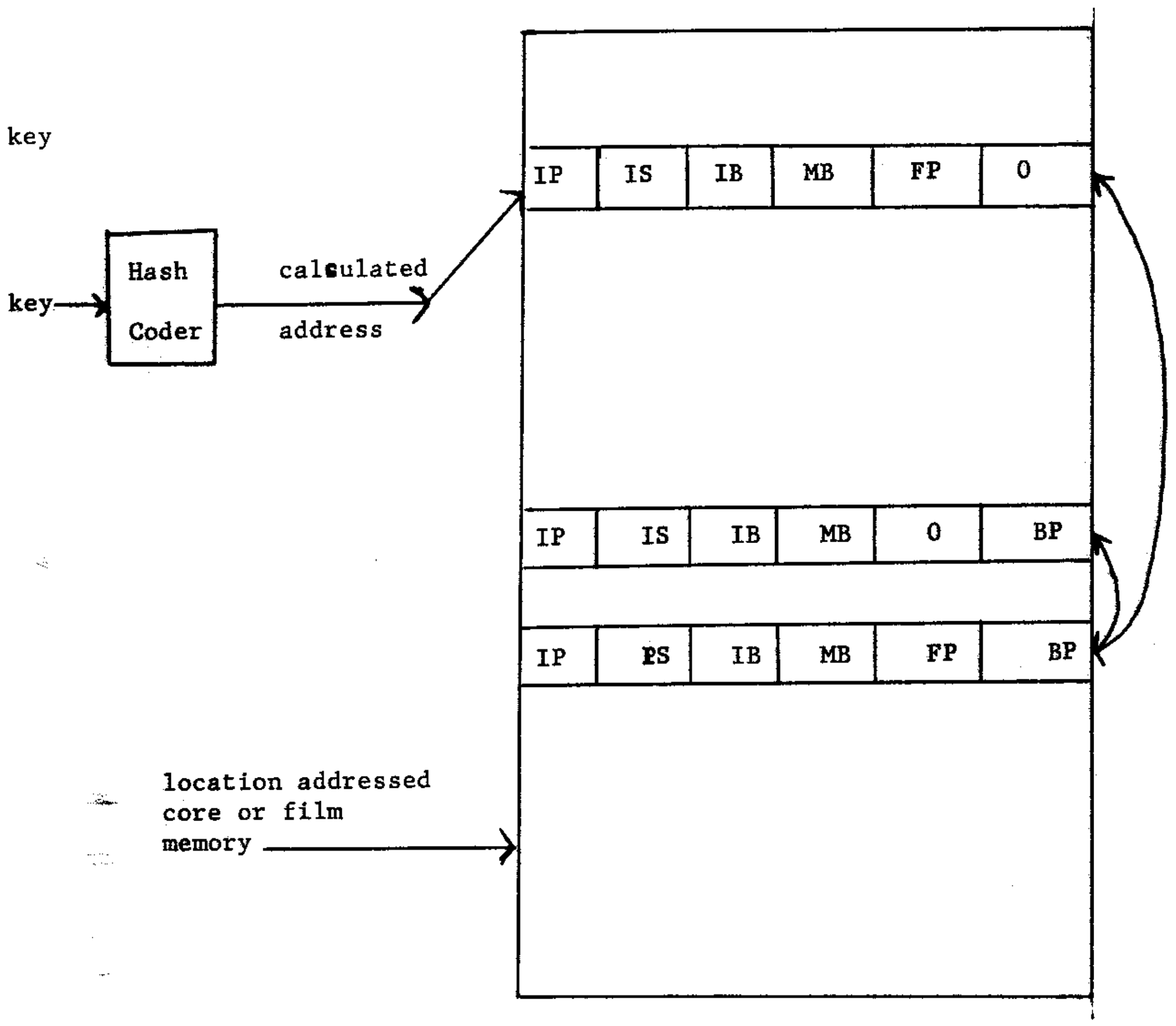

Fig. 5--Pseudo-associative realization of the index memory.