TO: MSPM Distribution

FROM: W. S. Bartlett

DATE: December 9, 1966

SUBJECT: Simulation System Loader (BE.7.07)

The following write-up is being reissued at this time to include discussion of a) execution on the 645, and new segments associated with execution; b) modification to the Loader to have DSGBND and NMTBND assembly parameters, and c) new error messages.

W. S. BARTLETT

Published: 12/9/66
(Supercedes: BE.7.07 6/21/66;
BE.7.07 3/21/66
BE.7.07 10/13/66)

Identification

Loader
W. S. Bartlett

Purpose

The Loader is a system of 635 subroutines whose purpose is to create a process containing the segments that have been supplied for loading, either by the assemblers, by the user in object form or from the 645 segment library. As a part of this procedure it is necessary to establish a number of supporting segments: a Descriptor segment; a Name Table; a Length Table; a Fault Vector; an Interrupt Vector; where necessary, page tables for these segments; and page tables for the segments supplied by the user.[1]

Simulation vs. Execution

The Loader may be used to create a process either for simulation ~~on the 635 or 645~~ or for execution on the 645. The actions taken by the Loader with respect to the Fault Vector, Interrupt Vector, and GIOC mail box differ depending upon whether simulation or execution is to take place. In addition slightly different actions are taken after loading depending upon whether control is to be passed to the simulator, or to the process that has just been loaded. These topics are covered in the appropriate places in the explanations in this section.

Dynamic Loading

When a linkage fault takes place during either simulation or execution and the name of the desired segment is not in the name table, the 'search' segment of the pseudo-supervisor (BE.8) checks in a dictionary of the 645 Segment Library for the desired segment. If the segment is mentioned in the dictionary it is loaded from the Segment Library. This is referred to as dynamic loading

---

[1] These supporting segments are discussed in subsection "Special Segment Descriptions" below.

and is the only way in which it is possible to load symbol segments. The relationship of the Loader to the dynamic loading process is that it searches and reads the Segment Library under the direction of 'search'. During dynamic loading all storage allocation and bookkeeping entries are performed by 'search'.

Section Structure

This section is divided into the following subsections:

Loading Procedure

Storage Allocation

Special Inclusion Policy

Special Segment Descriptions

Error Messages

Loading Procedure

The major sequencing in the Loader is controlled by a GMAP routine called the "load list" which is written for each run either by the 64.5 Driver (section BE.6.01) or the Merge-Editor (section BE.5.02) and assembled during the run in which it is used. The values of the following parameters, described in BE.5.02, are written into the load list:

DSPGSZ	Descriptor Segment Page Size
NTPGSZ	Name Table Page Size
LSPGSZ	Linkage Section Page Size
PGSIZE	Page Size of Text Segments to be Loaded (initial value)
LODORG	Loading Origin
STPGSZ	Stack Page Size
TIME	Maximum Running Time in 1000's Cycles

Note: The Descriptor Segment and the Name Table are always paged. The page size will be 64 unless the user specifies 1024 in which case the page size will be 1024.

These parameters are made available to the Loader at the time it is initially called.

The values of the following parameters, originally described in BE.5.02 are assembled into the loader:

Parameter	Value	Description
DSGBND	(DSPGSZ*64)-1	Descriptor segment bound--maximum permissable offset in descriptor segment.
NMTBND	(NMPGSZ*64)-1	Name Table bound--maximum permissable offset in Name Table.
FLTBAS	256_{10}	Base of the Fault Vector

The Loader performs the following sequence of actions.

1. The NØGØ switch is checked. This switch may have been set by EPLBSA, BSA or by the Packer. It is set when one of the assemblers cannot produce a text or link segment or because of other errors that may have been discovered. The EPLBSA, BSA Postprocessor and Packer error messages are given in sections BE.7.04, BE.7.02, and BE.7.05, respectively.

2. The Loader determines the size of the simulated 645 memory. It is taken to be all the space between slave zero and the bottom of the 635 routines (which are loaded at the top of 635 slave core).

3. One page is allocated for each of the Descriptor Segment, Name Table, Length Table segments. Page tables are built for any of these segments which are paged. If additional pages are needed they are allocated only when required. The descriptors for all these segments are placed in the Descriptor Segment, their names in the Name Table and their lengths in the Length Table.

4. Descriptor Segment, Name Table, and Length Table space for the Linker and Segment Manager is set aside so that when the segments are loaded they may be given the correct segment numbers (see the description of the Pseudo-supervisor in BE.8).

5. A descriptor is set pointing to the fault vector, and the name 'fvectr' is placed in the Name Table position associated with this descriptor (see the discussion of the Fault Vector in the Special Segments section below).

6. A descriptor for the 645 simulated memory is established and the name 'memory' is placed in the Name Table position associated with this descriptor (see the discussion of Memory in the Special Segments section below).

7. A descriptor is set pointing to the interrupt vector and the name 'ivectr' is placed in the Name Table position associated with this descriptor (see the discussion of the Interrupt Vector in the Special Segments section below).

8. A descriptor is set pointing to the GIOC mail box and the name 'giocl' is placed in the Name Table position associated with this descriptor (see the discussion of the GIOC mail box in the Special Segments section).

9. Next the load list calls the Loader with the list of the names and associated control fields of the segments to be loaded from the Segment Library. This list is sorted into ascending ASCII sequence. Duplicates are eliminated by removing all but the first appearance of the name. The control field information for each segment is provided along with the name.

10. After this the load list calls Loader routines to change the page size (when necessary) and initiate loading of the supplied segments one at a time. The Loader is given the file name and descriptor control field for each user supplied segment. The name is used to find the correct text and linkage files created by the Packer or EPLBSA. After each segment has been located, the list of library names is searched for the presence of the text segment's name. If the name is in the library list, it is removed from the list. When this occurs the control field information used is that supplied by the user, not that supplied with the library list.

11. The segments loaded or created as a part of the loading appear in the following order in the Descriptor Segment:

    a. Page table for the text (if any)

    b. The text segment (unless it is the Segment Manager or the Linker, which have special Descriptor Segment positions, see Special Segments below).

    c. The linkage associated with the text segment.

    d. Page table for the linkage section (if any).

    Core storage is, however, allocated in the following order:

    a. The text segment area

    b. The text segment page table area (if any)

    c. The linkage segment area

    d. The linkage segment page table area (if any)

12. After having loaded all the user supplied segments the Loader loads the segments whose names remain in the library list, from the 645 Segment Library. Segments on the library whose names do not appear in the library list are not loaded. If a segment name appears in the list but cannot be found on the library the following diagnostic is written on the error file:

    segment _____ not on library

13. It is possible to have a text segment without associated linkage on the Library and have it loaded correctly. A text segment without linkage may <u>not</u> be supplied by the user using the TEXT+LINK command to the Merge-Editor or Driver. After loading all segments to be loaded the Loader sets up the 645 Fault Vector and Interrupt Vector entries as described in the Special Segments discussion.

14. Then the Stack segment is allocated. It is desired to have the Stack segment take up the remainder of simulated core in order that access to the remaining storage area be possible for the 645 routines. The attempt to do this is performed in the following sequence.

    a. If there is sufficient room for at least one Stack page and one page-table page (if the stack is paged), the set up is done as requested.

    b. If the Stack is paged, the page size is 1024 and sufficient room is not available for one Stack page and one page-table page, the Stack page size is set to 64. If there is still insufficient room (for one page plus one page-table page) then an attempt is made to fit in an unpaged Stack. If one full 64-word page is not available the Loader discontinues loading and prevents simulation.

    c. If the Stack is paged, the Stack page size is 64 and there are more than 260 64-word blocks left (256 for the stack and 4 for its page table[2]) there would be a descriptor-segment boundary-field overflow. In order to prevent this under these conditions, the Stack page size will be set to 1024.

    d. If the Stack is unpaged and the block size is 1024 and there are not 1024 words left (after starting at a 1024-word boundary) the Stack page size is set to 64.

    e. If the Stack is unpaged and the block size is 64 and there are more than 256 64-word blocks, the block size will be set to 1024.

    The user is given no notice of changes to the Stack paging or page size. (He may look on the dumps as to how they ended up.)

---

[2]This is based on the assumption that page tables have 64-word blocks, the current edition of the Loader forces this to be the case.

15. The forward pointer in the first frame of the stack is set to show that the first segment executed will use no temporary storage on the stack.

16. An all-zero fence word is placed in the entry following the last entry in the Name Table segment.

17. a. If the process being loaded is to be simulated the CYCLES (from TIME), ABSM (= absolute mode), IC (= absolute address of first word of 'init'[3]), TØM, DBR, PBR (= segment number of the 'init' segment), and FVECTR (base of the Fault Vector parameters are passed to the simulator. If the loading is successful control passes to the simulator at the START entry. If the NOGO switch was on at the start, or if the loader could not successfully complete loading control passes to the simulator at the SUICID entry.

b. If the process being loaded is to be executed the communication region (for communication between the supplement and the escape coding[4]) is established.

If the loading is successful the bases of the Fault Vector and Interrupt Vector are handed to the Supplement, which at the same time switches the descriptor for the slave procedure containing the Loader to MASPRC. The Loader then loads the Descriptor word of the Descriptor Segment of the process just loaded and transfers indirectly through an its-pair to 'init'.

If the loading is not successful the Loader transfers to the escape coding and requests that the dumping process begin.

---

[3] See Special Segments discussion below.

[4] See BE.7.12 for a more complete discussion.

Storage Allocation

The Loader performs storage allocation (for historical reasons not associated with the present author) in the following way: A pointer is set to the LODORG. Each time a new block is requested the next higher block above the current value of the pointer, starting at the correct origin[5], is allocated and the pointer advanced by adding the number of words requested. Pages passed over (multiples of 64 that are not multiples of 1024, when requesting a 1024-word block) are lost. Hence mixtures of 64-word and 1024-word pages will result in the loss of the use of much core.[6]

Special Inclusion Policy

Whenever the 645 Loader is part of a 6.36 job, the Merge-Editor and 64.5 Driver create a library list which includes the names of segments 'init', 'linker', 'segman', and 'f2catc' and the names of a minimal set of escape and EPL run time routines.[7] These segments are included on the 645 Segment Library. User supplied copies of any of these segments will be loaded in place of the standard copies. Additional portions of the 645 escape coding and run time segments for EPL are also on the library and are available for dynamic loading.

Special Segment Descriptions

1. Descriptor Segment

   The name of the Descriptor Segment is 'dseg', the descriptor control field is DATA, the segment number is 0.

2. Name Table

   The Name Table segment is created by the Loader for the convenience of the Linker and of GEBUG. The Name Table segment is a collection of ASCII strings, four

---

[5] Multiple of 64 for 64-word pages, multiple of 1024 for 1024-word pages.

[6] The basic problem is that the Loader must load unpaged segments without knowing how long they are.

[7] 'linker', 'segman', and 'f2catc' are discussed in BE.8.

9-bit characters (called bytes) per word, loosely packed (each new string begins a new word). These strings are the names of the segments loaded, (the first byte contains the number of characters in the name[8]), in the order in which their descriptors appear in the Descriptor Segment. Immediately following the last entry in the table, a fence word of zero appears.

In the body of the table, fill bytes (that is, bytes beyond the counted ones in the last word of an entry) have no conventional setting. Because entries in the table occupy a variable number of words, meaningful scans of this segment must start from relative location zero.

The name of the Name Table segment is 'namtab' the descriptor control field is DATA, SLVACC, the segment number is 3.

3. Fault Vector

A segment named 'fvectr' is painted over the 64-word Fault Vector and the two following 64-word pages. The base of 'fvectr' is at the base of the Fault Vector. The first 64-word page contains scu, tra instruction pairs. The second 64-word page contains the its pairs for the scu instruction, the third 64-word page, the its pairs for the tra instruction. The entries for the Fault 2 Catcher are initialized by the Loader. All other entries contain (777777001000)8, "escape -1" for the simulator or "mmel -1" for the 645 processor.

The control field is DATA, the segment number is 4.

4. 645 Memory

For the benefit of the escape coding a segment named 'memory' is painted over the entire memory available to the process if it is to be simulated, and if it is to be executed, over the entire slave memory allocated to this activity. The control field is DATA, SLVACC, WPERMT and the segment number is 5.

---

[8]Note: The maximum number of characters in a name is 31.

5. GIOC Mail Box[9]

When the process is to be executed a simulated GIOC Mail Box area is set aside by the supplement for the use of the process. The address and size of the Mail Box is obtained directly from the supplement. If the process is to be simulated a dummy Mail Box is created. The contents of the Mail Box is not initialized by the Loader. The segment is unpaged, its descriptor is DATA and it is segment number 6, the Name Table entry is 'giocl'.

6. Interrupt Vector[9]

A segment named 'ivectr' is painted over the 64-word interrupt vector which starts at location 448_{10} and extends for 3 64-word pages. The even locations of the first page are initialized to scu instructions indirect through the corresponding location pair in the second page and the odd locations of the first page are initialized to rcu instructions indirect through the corresponding location pair in the third page. The second and third page are not initialized. It is intended that the Interrupt Vector be used only by processes to be executed on the 645. The Interrupt Vector is unpaged, has descriptor control field DATA and is segment number 7.

7. Length Table

The Loader generates a segment Length Table named 'lentab'. The Length Table contains a single word entry for each segment in the process, the entries are in the same order as the Descriptor Segment. Each entry contains the maximum word length of the segment in bits 0-17 and the current word length in bits 18-35.

The actual entries are the offsets, not the numbers of words (offset = number of words -1). The maximum length of an unpaged segment is one less than the product of the number of blocks comprising the segment and the size of the blocks. The maximum length of a paged segment is one less than the product of the number of blocks in its page table, the number of words per page table block, and the size of the segment's pages. The descriptor information for segment 'lentab' is DATA, SLVACC.

---

[9] See BE.7.12 for a more complete discussion.

8. Initializer

The Loader arranges that simulation or execution always begins at the first location of segment 'init'.

The 'init' program is charged with

(a) Initializing all fault locations except for those associated with the fi tag,

(b) Initializing all base address registers,

(c) Other absolute initializing (e.g., interrupt locations) which may be specified from time to time, and

(d) Terminating with

call <.init>|[..init]

The descriptor control field of 'init' is MASPRC.

9. .init

The Merge-Editor (BE.5.02) or the 64.5 Driver (BE.6.01) generate (for assembly by EPLBSA) the '.init' segment. The '.init' segment has an entry point '..init'. '.init' calls the user's entry point. If the user returns, '.init' calls <escape>|[finish]. The user does not supply his own '.init' segment.

10. Page Tables

For each paged segment loaded or created, the Loader generates a separate segment containing the former segment's page table. The name of the page table of segment number n is 'ptoooo', where oooo is the octal form of n. The descriptor control field of each page table segment, in the Descriptor Segment, is DATA. The control field within each page table word is MASPRC, SLVACC, WPERMT.

11. Linkage Segments

There is a linkage segment generated by the Loader for each segment which has linkage. The name of the linkage segment for segment number n is 'lkoooo' where oooo is the octal form of n. The descriptor control field of each linkage segment is SLVPRC, SLVACC, WPERMT.

Note: Segments without linkage may be loaded from the Library but not from the Text and Link files.

12. Stack

The name of the Stack segment is 'stack'. The descriptor control field is DATA, SLVACC, WPERMT. The Stack segment is built out of the remainder of free memory after other loading. The page size of the stack may be changed by the Loader from that specified, see number 14 in the description of the loading process above.

Error Messages

After discovery of an error condition, the Loader writes a message on the error file (file code ER) and gives control to the routine providing core dumps. Each message is of the form 'loading aborted in subroutine _____ for reason _'.

SUBROUTINE	REASON	MEANING
ADDSEG	1	Insufficient space in Descriptor Segment (see DSGBND parameter at the start of this section).
	2	Insufficient space in Name Table (see NMTBND parameter at the start of this section).
ADDWRD	1	Loader errors.
	⋮	
	4	
ALØCAT	1	Core allocation routine could find no more space.
FIND	0	Region missing from global file. System error, since should be caught by Merge-Editor or Driver.
	1	Empty region in global file. System error in EPLBSA, BSA, or Packer.
	2	Empty linkage segment
	3	System error in GECOS
	⋮	
	7	
INITLD	0	NOGO switch set by EPLBSA, BSA or Packer
LIBLST	1	Identification record but no text record on Library.
LODTXT	1	No room for text segment being loaded.
	2	No room for text segment page table.
	3	Unused
	4	Descriptor segment boundary field overflow for text segment being loaded.

SUBROUTINE	REASON	MEANING
LODLKG	1	No room for linkage segment being loaded.
	2	No room for linkage segment page table.
	3	Linkage segment does not have an id code of 1.
	4	Descriptor-segment boundary-field overflow for linkage segment.
LREC	1	GEFRC error return--probably hardware problem
	2 ⋮	GEFRC error returns--probably from GET.
SETAPP	1 ⋮ 6	Loader Errors.
TERMLD	1	No room for Stack segment.
	2	No room for Name Table Fence word--(see NMTBND parameter at the start of this section).