Published: 04/25/68

Identification

Core Control Initialization Procedures
P. G. Neumann, M. R. Wagner and R. C. Daley

Purpose

This section describes the operation of the procedures used during system initialization to initialize the core map and to interface with the various aspects of system initialization. The procedures described here are contained in three different segments all of which may be discarded after system initialization is complete.

Summary

The following list summarizes the various procedures and entries used in the initialization of core control. A detailed description of each entry and its arguments follow the summary.

init_core_control	core initialization control program
ccinit1$initcm	initializes the core map
ccinit1$assign_at_loc	updates core map after collection 1 has been loaded
ccinit1$setup_64	sets size of 64-word map
ccinit2$del_temp	deletes temporary segments
ccinit2$set_sst_ptr	sets the sst relative pointer in a block map entry

init_core_control

The init_core_control procedure is called by the Multics initializer (see BC.5.01) after the loading of collection 1. This procedure is run only once during the initialization of collection 1 and is deleted before the loading of collection 2 is begun. The init_core_control procedure is invoked by means of the following call.

```
call init_core_control;
```

Upon receiving this call, init_core_control takes the following steps.

1. The MMCT (see BK.4.04) is consulted to determine the current core configuration and ccinit1$initcm is called to initialize the core map. The core map is initialized to show that all available core is free.

2. The core assignments already made during the loading of collection 1 are reflected in the core map by calls to ccinit1$assign_at_loc for each (non-temporary) segment loaded in collection 1. Since core used in loading temporary segments is not reflected in the core map, this core may be reused during the loading of collection 2.

3. The final size of the 64-word block map is established by a call to ccinit1$setup_64.

4. The core management module (see BG.5.02) is initialized by a call to core_man$init and control is returned to the initializer.

ccinit1$initcm

The core map is initialized by the following call issued by init_core_control.

```
call initcm (p);
```

In this call p is a pointer to the following structure which describes all of the core available to the Multics system.

```
dcl  1 core based (p),
     2 controllers fixed,
     2 mem (8),
       3 loc fixed bin (18),
       3 size fixed;
```

controllers- contains the number (1-8) of system controllers available to the system.

mem- is an array which specifies the base location (loc) and size (size) of each available controller in units of 64-word blocks.

Upon receiving the above call, the core map header and the 1024-word block map are initialized. The status of each 1024-word block is initially set to free by placing the corresponding 1024-word block map entries in the 1024-word free list. The index of the first entry in the 64-word map is established and saved in the core map header. However, no core is made available in the 64-word map at this time.

ccinit1$assign_at_loc

The assign_at_loc call is used by init_core_control to reflect core assignments already made during the loading of collection 1 and is invoked by means of the following call.

```
call ccinit1$assign_at_loc (loc, sps, blocks, stat, page);
```

The arguments used in this call are declared by the following PL/I statement and are described in detail below.

```
dcl  loc fixed bin (18),
     sps fixed bin (1),
     blocks fixed bin (18),
     stat fixed bin (2),
     page fixed bin (8);
```

loc- is the high order 18 bits of the absolute 24-bit memory address of a contiguous set of core blocks to be assigned (after the fact).

sps- is a switch indicating if ON(1) that the core blocks are to be assigned to the 64-word map and if OFF(0) that the core blocks are to be assigned to the 1024-word map.

blocks- is the number of 64-word or 1024-word blocks affected by this call.

stat- indicates the status the assigned blocks are to have upon completion of this call (0=removable, 1=wired, 2=permanent). The constant 2 is added to this quantity to obtain the desired type code as described in BG.5.01.

page- is the page number (0-255) of the first block if the block is currently used as a page of a pageable segment. Individual pages of pageable segments must be assigned one at a time (blocks=1) in order to properly specify this quantity.

If the sps parameter specifies 1024-word blocks, the corresponding 1024-word block map entries are removed from the 1024-word free list and placed in the list appropriate to the stat argument. If sps specifies 64-word blocks, the containing 1024-word block(s) may have to be split if not already split by a previous call. When this is the case, the corresponding 1024-word block is split and the 16 corresponding entries are entered in the 64-word free list. After any necessary splitting is done, the 64-word map entries corresponding to the blocks specified in the call are removed from the 64-word free list and placed in the list appropriate to the stat argument.

Whenever this call causes a 64-word map entry to be placed on the 64-word permanent list, a check is made to see if an entire 1024-word block has been assigned to the 64-word permanent list. If this case is discovered, the 1024-word block map entry is placed on the 1024-word permanent list and the 16 corresponding 64-word map entries are discarded. The vacated slots in the 64-word map may now be reused for splitting other 1024-word blocks. This strategy is used to reduce the core space required by the core map itself.

ccinit1$setup_64

After the core map has been updated to relect all current core assignments the final size of the 64-word map is established by means of the following call.

```
call ccinit1$setup_64;
```

If the number of available 64-word blocks as reflected in the 64-word map is less than the number required for normal operation, additional 1024-word blocks must be split and made available as 64-word blocks. This is done by removing the required number of 1024-word block map entries from the 1024-word free list, splitting them and placing the resulting 64-word entries on the 64-word free list.

ccinit2$del_temp

During the loading of collection 2, all core used in loading temporary segments is assigned in temporary status. All temporary segments loaded during collection 1 or 2 may be deleted by means of the following call.

```
call ccinit2$del_temp;
```

Upon receiving this call, the segment descriptors (SDWs) of all temporary segments are zeroed and the deleted switch is set on in the corresponding SLT entries. Once this is done, any core assigned in temporary status is returned to the free list.

ccinit2$set_sst_ptr

After loading collection 2, SST entries (see BG.2) are constructed for all (non-permanent) segments which may be removed to secondary storage. At this time, core map entries for pages of these segments must be related to their corresponding entries in the SST. This relationship is set up by update_sst (see BL.10.02) which makes use of the following call.

```
call ccinit2$set_sst_ptr(loc, sstep);
```

The arguments used in this call are declared in the following PL/I statement and are described in detail below.

```
dcl  loc fixed bin (18),
     sstep ptr;
```

loc- is the page address of a page to be related to the SST entry of the segment of which it is a part.

sstep- is a pointer to SST entry of the segment which contains this page.

Upon receiving this call, the block map entry corresponding to loc is updated to contain a relative pointer (i.e., rel (sstep)) to the specified SST entry.