M. Shaw

S8000 MMU CONFIGURATION

Inder M. Singh

INTRODUCTION

This note describes the revised MMU configuration for the S8000. The MMU setup in the current S8000 prototype hardware is quite appropriate for the phase one non-segmented software strategy. This revised MMU configuration maintains the support for the non-segmented software of phase one while providing a cleaner structure for segmented programs in phase two. It also provides better support for a segmented version of the operating system, in case we decide to follow this route. This also makes the hardware more attractive as an OEM product.

The only significant change between this version and the first draft is in the area of system access to user space. Changed sections are indicated by bars in the margin.

OVERVIEW

The CPU board contains three MMUs, referred to as M1 through M3.

A non-segmented operating system (OS) runs in segment 0, using slot 0 of M1, M2 and M3 for code, data and stack areas, respectively. M1 is used for translating program memory references, M2 or M3 are used for translating all other memory references. The selection between M2 or M3 is based on a comparison between the address and the contents of the System Break Register, a program accessible hardware register. Addresses lower than the SBR are treated as data addresses and are directed to M2. Addresses greater than or equal to the SBR are treated as stack addresses and are directed to M3.

A non-segmented user program runs in segment

63 (in fact, the hardware will allow any segment between 1 and 63 to be used). As in the case of the non-segmented OS, slot 63 in M1, M2 and M3 is used to provide separate code, data and stack areas respectively. The only difference is that the Normal Break Register (NBR) is used to distinguish between data and stack references instead of the SBR.

A segmented OS uses M1 to provide an address space consisting of up to 63 segments (segments 0 through 62). Segment 63 would be used to run non-segmented programs. No separation between code, data or stack segments is provided, or needed. The attribute flags in the segment descriptors of M3 can be used to configure different segments for this purpose.

A segmented user program uses M2 and M3 to provide an address space consisting of 126 or 128 segments, again without separating code, data and stack spaces. If the OS is non-segmented, segment numbers 0 and 64 are reserved for the OS, since it requires segment 0 slot of M2 and M3. With a segmented OS, all 128 segments are useable.

Figures 1 through 4 illustrate the OS and user address spaces in the various configurations, and the MMU slots (SDRs) which are used to map the different segments within the address spaces.

CONFIGURATION CONTROL

The MMU configuration is controlled by the OS software via two bits in an I/O port: Segmented System (SS) and Segmented User (SU). The operation in the different configurations, controlled by the SS and SU bits and the system/normal CPU state, is described below:

a) SS=0, SU=0. This configuration is intended for a non-segmented OS running non-segmented programs. The OS runs in segment 0, a user process runs in any of the segments 1 through 63, with 63 as the recommended segment.

   MMU M1 is enabled for program references, indicated by a CPU status code of 11xx. For other memory references, the operation is as follows: the address offset generated by the CPU is compared with the SBR if the segment number is 0, or with the NBR if the segment number is non-zero. If the result of the comparison is less than zero, MMU M2 is

enabled; otherwise, MMU M3 is enabled.

If a memory reference to segment 0 is made in normal mode, a segment trap is generated, and all three MMUs are disabled. The suppress signal is also asserted to protect memory.

If the OS (or parts of it) executes in segmented mode, the separation of code, data and stack spaces described above still applies. Note that the separation between data and stack spaces is based on the SBR for segment 0 and on the NBR for references to any other segment.

It is possible to run a segmented user process in this configuration, although the configuration with SU=1 is intended for this purpose. Such a process has a potential address space of 63 code and 63 data segments.

b) SS=0, SU=1. This configuration is used for a non-segmented OS running a segmented user program.

In system mode, if the segment number is zero, the operation is the same as in (a) above. Code, data and stack references are directed to M1, M2 and M3 respectively. The SBR contents are used to select between data and stack references.

In normal mode, MMU M2 is enabled for segment numbers 1 through 63, MMU M3 is enabled for segment numbers 65 through 127.

If a memory reference is made to segment 0 or 64 in normal mode, a segment trap is generated, and all three MMUs are disabled. The suppress signal is also asserted to protect memory. This protects the system data and stack areas from being accessed by the user program.

In system mode, if the segment number of a user segment is generated (1-63 or 65-127), the translation is the same as in normal mode. Separation of code, data and stack spaces is deactivated; MMU M2 is enabled for segments 1 through 63, MMU M3 is enabled for segments 65 through 127. This allows the OS to access any user location directly.

c) SS=1, SU=0. This configuration is used by a segmented OS running a non-segmented user process.

MMU M1 is enabled for all memory references in system mode.

In normal mode, separation between code, data and stack references is activated. As described above in (a), M1 is enabled for program references. Other memory references are directed to M2 or M3 based on a comparison between the address offset, and the contents of the NBR for any non-zero segment number.

d) SS=1, SU=1. This configuration is used for a segmented OS running a segmented user process.

MMU M1 is enabled in system mode. In normal mode, MMU M2 is enabled for segment numbers 0 through 63; MMU M3 is enabled for segment numbers 64 through 127. A user process can access all 128 segments in this configuration.

BREAK REGISTERS

There are two 4-bit registers accesible as I/O ports: the System Break Register (SBR) and the Normal Break Register (NBR). During any memory reference, the 16-bit address offset is compared to the break value given by the contents of either the SBR or the NBR extended with 12 low-order zeroes. The SBR is used for the break value if the segment number is zero, the NBR if it is non-zero. If the MMU configuration calls for separation of program, data and stack spaces, and the CPU status code indicates a non-program reference (status code 10xx) then the result of this comparison selects between data and stack references. If the address offset is less than the break value, the current reference is for data, and MMU M2 is enabled. Otherwise it is a stack reference, and MMU M3 is enabled.

SYSTEM ACCESS TO USER SPACE

To access a user segment, the OS can use a free segment slot, and set up its Segment

Descriptor Register (SDR) to point to the same memory area as the target user segment's SDR.

A non-segmented OS running a non-segmented user process can directly access the user data and stack areas by going into system mode and using the user segment number. To access the user's code segment, one of the unused segment slots, such as 62, is set up to point to the code segment. The SDR's for this slot in M2 and M3 are both set up to point to the code segment, so that the contents of the NBR don't matter.

A non-segmented OS running a segmented user process can directly access any part of the user space by going into segmented mode.

A segmented OS has to map one or more of the 64 segments in its space to the user segments it needs to access by setting the corresponding SDRs in MMU M1.

Some Nitty-Gritty Details

The high-order segment line (SN6) into the MMUs is hard-wired to a logical zero, so that the MMUs only see segment numbers 0 through 63. When US is set to one, and the CPU is in normal mode, either M2 or M3 is enabled depending on whether SN6 (out of the CPU) is 0 or 1. Thus even though M3 is used for translating user segments above 63, the Upper Range Select (URS) bit in the MMUs should always be zero.

The hardware uses the N/S- input of the MMUs to select the appropriate MMU. The MST bit in the MMUs should be set to 1 to ebable this mode. The NMS bit should be set to 0. Since the MMUs donot see the CPU's system/normal state, system segments cannot be protected from user access by using the System-only (SYS) bit in the SDRs. The protection is obtained by separating the address spaces when both the OS and the user program are running segmented (SS=1, US=1). A Non-segmentmented user program cannot access outside its segment, so this problem doesn't exist. When a segmented program is run under a non-segmented OS (SS=0, US=1), external hardware is used to prohibit accesses to system segments in normal mode. In this configuration, access to segments 0 or 64 in normal mode generate segment traps.

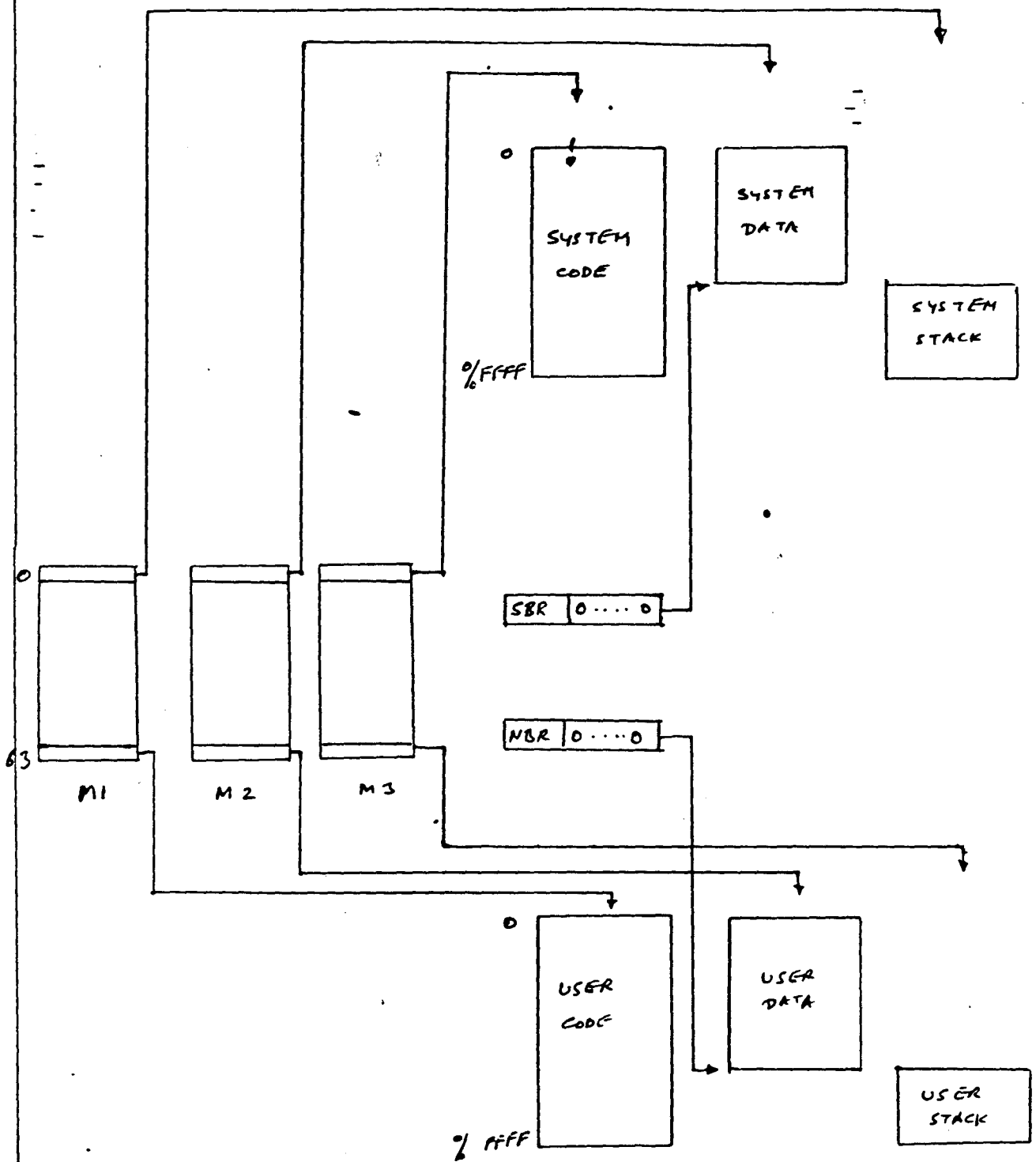

Figure 1. Non-segmented OS running non-segmented user (SS = 0, SU = 0)

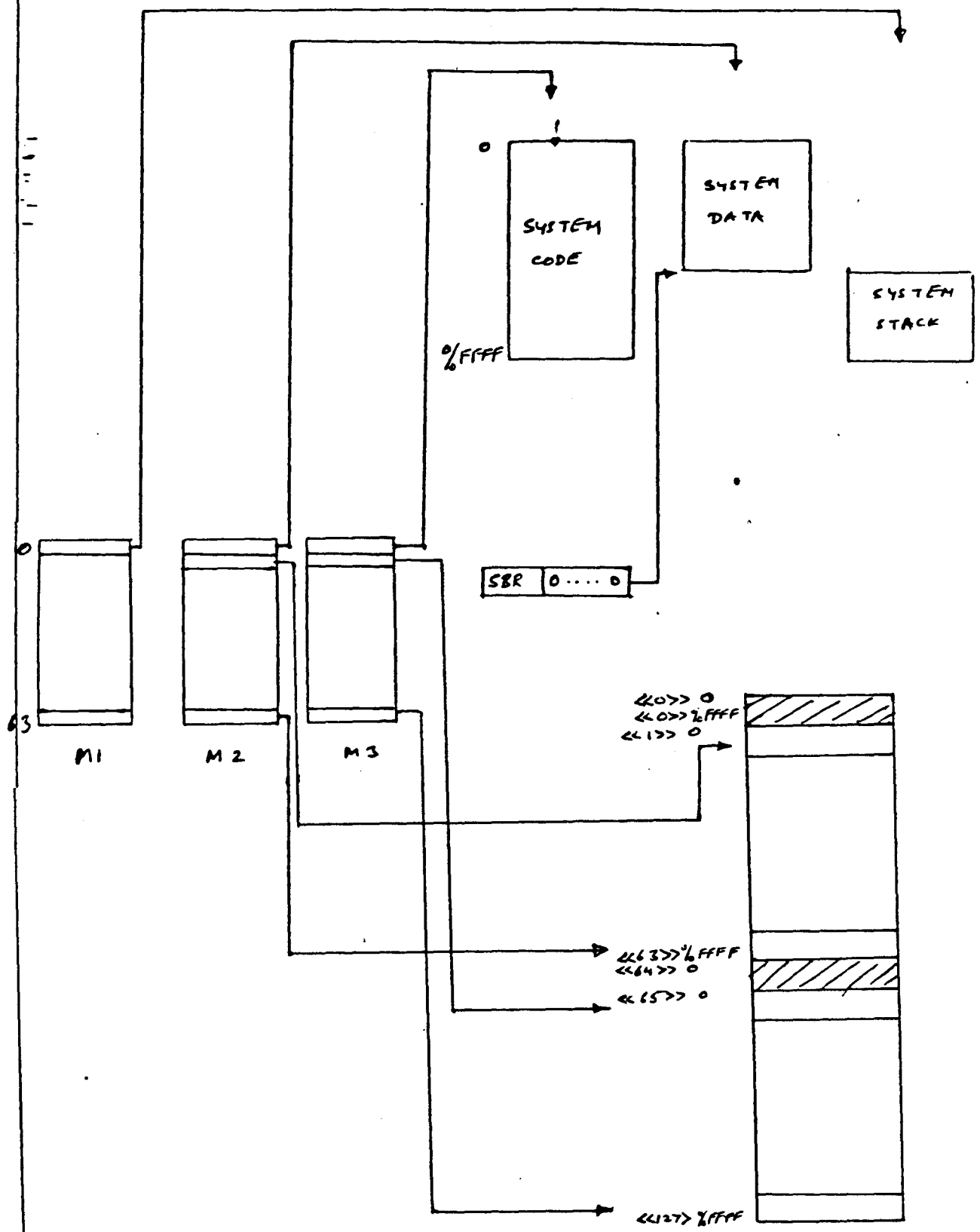

Figure 2. Non-segmented OS running segmented user (SS=0, SU=1)

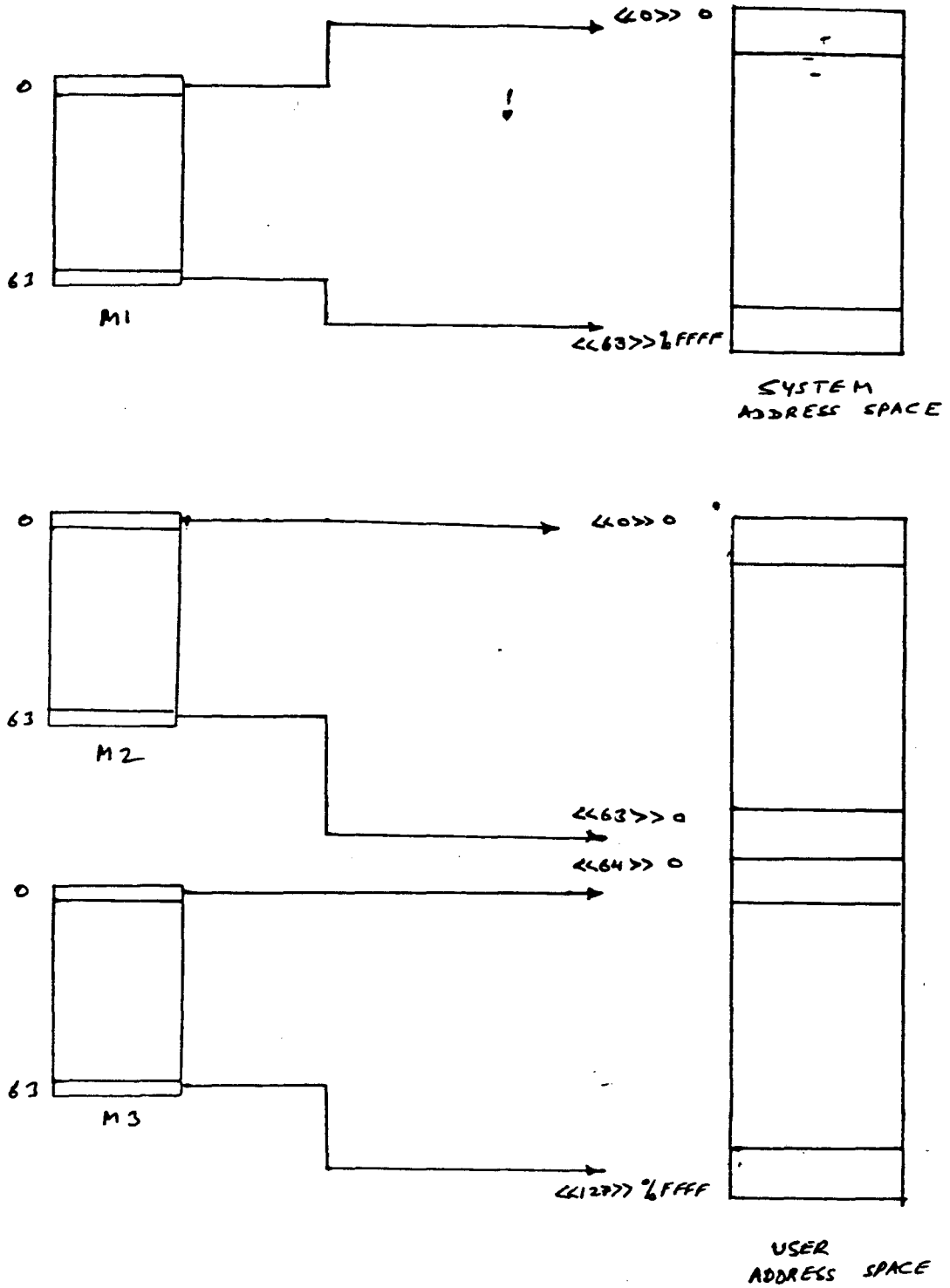

Figure 3. Segmented OS running ~~segmented~~ user
(SS = 1, SU = 1)

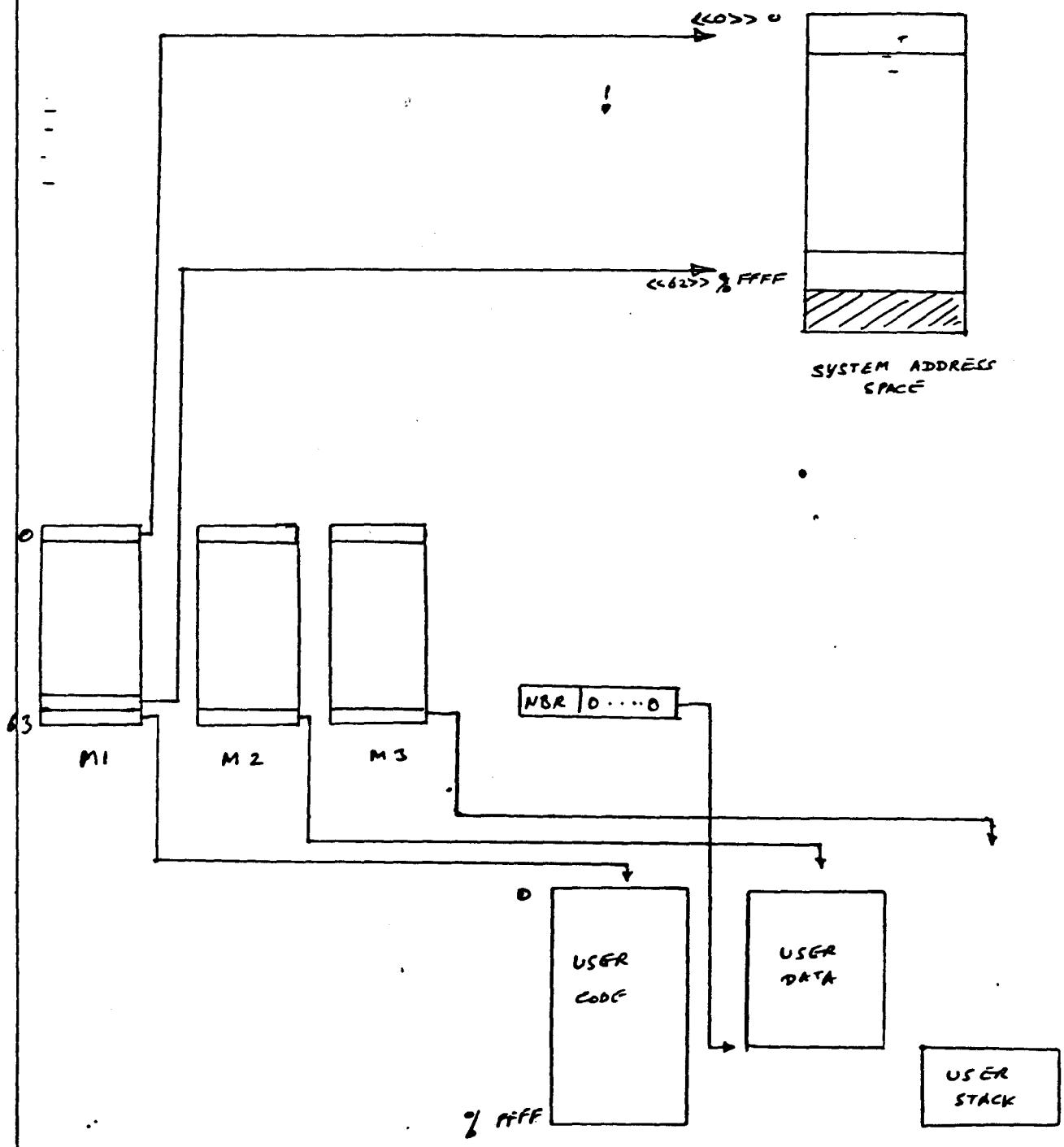

Figure 4. Segmented OS running non-segmented user (SS=1, SU=0)