PUBLISHED: 6/02/66

Identification.

The Processor Stack
Chester Jones

Purpose.

The Processor Stack is a per-processor data base that is used for safe-storing the processor state when the processor accepts an external interrupt and for making the calls necessary for handling the interrupt during which a page-not-in-core fault cannot be tolerated. The Processor Stack is maintained as part of the Processor Data Segment, a block of "wired-in" core space which is allocated to each processor in the system. (See Section BK.1.01 for a description of the Processor Data Segment.) Since the Processor Data Segment is passed along from process to process, the Processor Stack is always accessible to the running process.

Usage of the Processor Stack.

When a processor accepts an external interrupt, control passes to the System Interrupt Interceptor (Section BK.2.02) which executes as part of the process which happens to be running at the time of the interrupt. In the System Interrupt Interceptor, the processor state (including the state of the interrupt mask) is safe-stored in the Processor Stack, space is made available for safe-storing the processor state should another interrupt arrive, interrupts of equal or lower priority are masked, and the appropriate interrupt handler is called. The interrupt handlers are brief and to the point. They cannot contain programmed faults (e.g. page-not-in-core faults) and, therefore they use the Processor Stack. They may relinquish control of the processor only to call Wakeup (Section BJ.7) to allow some other process which really belongs to this interrupt to schedule itself using this processor (control is guaranteed to return). When a return is made from the interrupt handler, the processor state (including the previous state of the interrupt mask) is restored and control is returned to the point at which the interrupt occurred. Once the processor state has been restored, the core space that is used for safe-storing the processor state is automatically returned to the Processor Stack.

In the Traffic Controller (Section BJ) design, the time-out interrupt is assigned lower priority than the external interrupts so that the time-out interrupt is recognized only after all external interrupts have been handled. Therefore, when a processor recognizes the time-out interrupt and switches processes, the new process is passed an "empty" Processor Stack.

Organization of the Processor Stack.

The first physical locations of the Processor Stack contain pointers which are used in maintaining the remainder of the Processor Stack. The remainder of the Processor Stack contains interrupt frames and associated interim stacks. An interrupt frame is used for safe-storing the processor state following an external interrupt. An interim stack is created following an external interrupt and is used for calling the appropriate modules for handling that interrupt. An interim stack contains one or more stack frames, each of which is produced by the execution of the standard CALL, SAVE sequence. (See Section BD.7.02 for a description of the standard CALL, SAVE, and RETURN sequence.) Each interim stack is independent of the other interim stacks.

Processor Stack Base.

The usage of the first physical locations at the base of the Processor Stack is as follows:

base location	0-1	stb pointer. This points to the base location of the interrupt frame to be used when an interrupt occurs (i.e. "current" interrupt frame). This pointer is used in storing the address base registers when an interrupt occurs.
	2-3	sreg pointer. This points to the area reserved for the arithmetic registers within the current interrupt frame.
	4-5	scu pointer. This points to the area reserved for the processor control unit within the current interrupt frame.

The base location of the Processor Stack must be an even location since each of the pointers contained in the first physical locations must occupy an even-odd word pair. Also, the base location of each Processor Stack must be offset an equal amount within the Processor Data Segments for all processors.

Interrupt Frame Format.

The usage of the various locations within an interrupt frame is as follows:

base location	0-7	save address base registers
	8-15	save arithmetic registers
	16-21	save processor control unit
	22-23	back pointer to the base location of the preceding interrupt frame

The base location of the interrupt frame must equal zero (modulo 8) since the instructions for storing the address base registers, arithmetic registers, and processor control unit require addresses which equal zero (modulo 8).

Interim Stack Format.

The internal organization of an interim stack is shown in the diagram. (For further information, see Stack Usage in Section BD.7.02.) For the purpose of the diagram, the System Interrupt Interceptor, SII, calls proc1 and proc1 calls proc2. Each time a procedure is called, the value of the stack pointer, sp, is adjusted to point to a new stack frame.

sp when in SII	0-7	save bases when calling proc1
(base location	8-15	save registers when calling proc1
of interim stack)	16-17	not used since this is first frame
	18-19	value of sp when in proc1 (next sp)
	20-21	return when calling proc1
	22-23	(used by master/execute only procedures)
	24-25	(used for alternate return to SII)
	26-27	value of ap when calling proc1
		temporary storage for SII The temporary storage for SII includes space required to save memory controller mask register(s). (Two words are required for each memory controller for which this processor is designated control processor.)
sp when in proc1	0-7	save bases when calling proc2
	8-15	save registers when calling proc2
	16-17	value of sp when in SII (last sp)
	18-19	value of sp when in proc2 (next sp)
	20-21	return when calling proc2
	22-23	(used by master/execute only procedures)
	24-25	(used for alternate return to proc1
	26-27	value of ap when calling proc2
		temporary storage for proc1
sp when in proc2	0-7	....

The base location of each stack frame (i.e. the value of sp) must equal zero (modulo 8) since the instructions for storing the bases and registers require addresses which equal zero (modulo 8).

Management of the Processor Stack.

When a processor accepts an external interrupt, the following actions are taken by the System Interrupt Interceptor with regard to the Processor Stack:

1. The processor state is stored in the current interrupt frame. Pointers to the current interrupt frame are maintained at the base of the Processor Stack. The processor state is stored in three steps:

   a. The processor control unit is stored (store control unit instruction) using the scu pointer.

   b. The arithmetic registers are stored (store registers instruction) using the sreg pointer.

   c. The address base registers are stored (store bases instruction) using the stb pointer.

2. A new interrupt frame is allocated in preparation for the next external interrupt. Since a processor may be interrupted in the course of handling an earlier interrupt, care is taken to prevent a possible overlapping of the new interrupt frame and the interim stack in use at the instant of the interrupt. A new interrupt frame is allocated as follows:

   a. The value of the Processor Stack pointer, sp, is obtained. If the Processor Stack is not "empty", the sp-sb base register pair (stored in Step 1) points to the base location of the stack frame in use at the instant of the interrupt and sp/18 points to the base location of the next (intended) stack frame (i.e. next sp). In this case, a constant, k, is added to the value of the next sp to obtain the base location of the new interrupt frame. (Currently, k is equal to 32.) If the Processor Stack is "empty" at the instant an external interrupt occurs, then the new interrupt frame is allocated immediately following the current interrupt frame. In this case, the base location of the new interrupt frame is obtained by adding 24 (i.e. the length of an interrupt frame) to the stb pointer at the base of the Processor Stack.

   b. A back pointer to the previous interrupt frame is fabricated and stored into locations 22-23 of the new interrupt frame.

   c. The stb, sreg, and scu pointers stored at the base of the Processor Stack are adjusted to point to the appropriate areas within the new interrupt frame.

3. A new interim stack is created immediately following the new interrupt frame. The sp-sb address base register pair is set to point to the base location of the new interim stack.

4. The value of the next sp is determined and stored at sp/18 in the first stack frame of the new interim stack.

The amount of temporary storage required by the System Interrupt Interceptor is used in determining the value to be stored at sp/18.

5. The memory controller interrupt mask register is stored in the temporary storage of the first stack frame of the new interim stack. This step is repeated for all memory controllers for which this processor is designated control processor.

6. A mask that will inhibit recognition of interrupts of equal or lower priority is stored in the memory controller interrupt mask register. This step is repeated for all memory controllers for which this processor is designated control processor.

7. A standard CALL is issued to the appropriate module to handle the interrupt. This module may CALL other modules. Until a RETURN is made to the System Interrupt Interceptor, all Processor Stack management is performed by the standard CALL, SAVE, and RETURN sequence. When control returns to the System Interrupt Interceptor, the sequence of steps presented here continues.

8. The scu, sreg, and stb pointers stored at the base of the Processor Stack are restored from locations 22-23 of the current interrupt frame to point to the base location of the previous interrupt frame.

9. The memory controller interrupt mask register is restored to its state at the instant the interrupt occurred. This step is repeated for all memory controllers for which this processor is designated control processor.

10. The processor is restored to its state at the instant the interrupt occurred. This is done as follows:

    a. The arithmetic registers are restored (load registers instruction) using the sreg pointer.

    b. The address base registers are restored (load bases instruction) using the stb pointer.

    c. The processor control unit is restored (restore control unit instruction) using the scu pointer. When the processor control unit is restored, control returns to the point at which the interrupt occurred.

With the exception of Step 7, all of the above steps are executed in master mode with interrupts inhibited.

Processor Stack Examples.

In the attached diagrams, the Processor Stack is shown at three instances:

1. While the Processor Stack is empty. (Figure 1)

2. After the processor has accepted one interrupt. (Figure 2)

3. After the processor has accepted three "cascaded" interrupts. (Figure 3)

In Figure 1, PDS stands for the segment number of the Processor Data Segment and psb stands for the offset within the Processor Data Segment at which the Processor Stack begins.

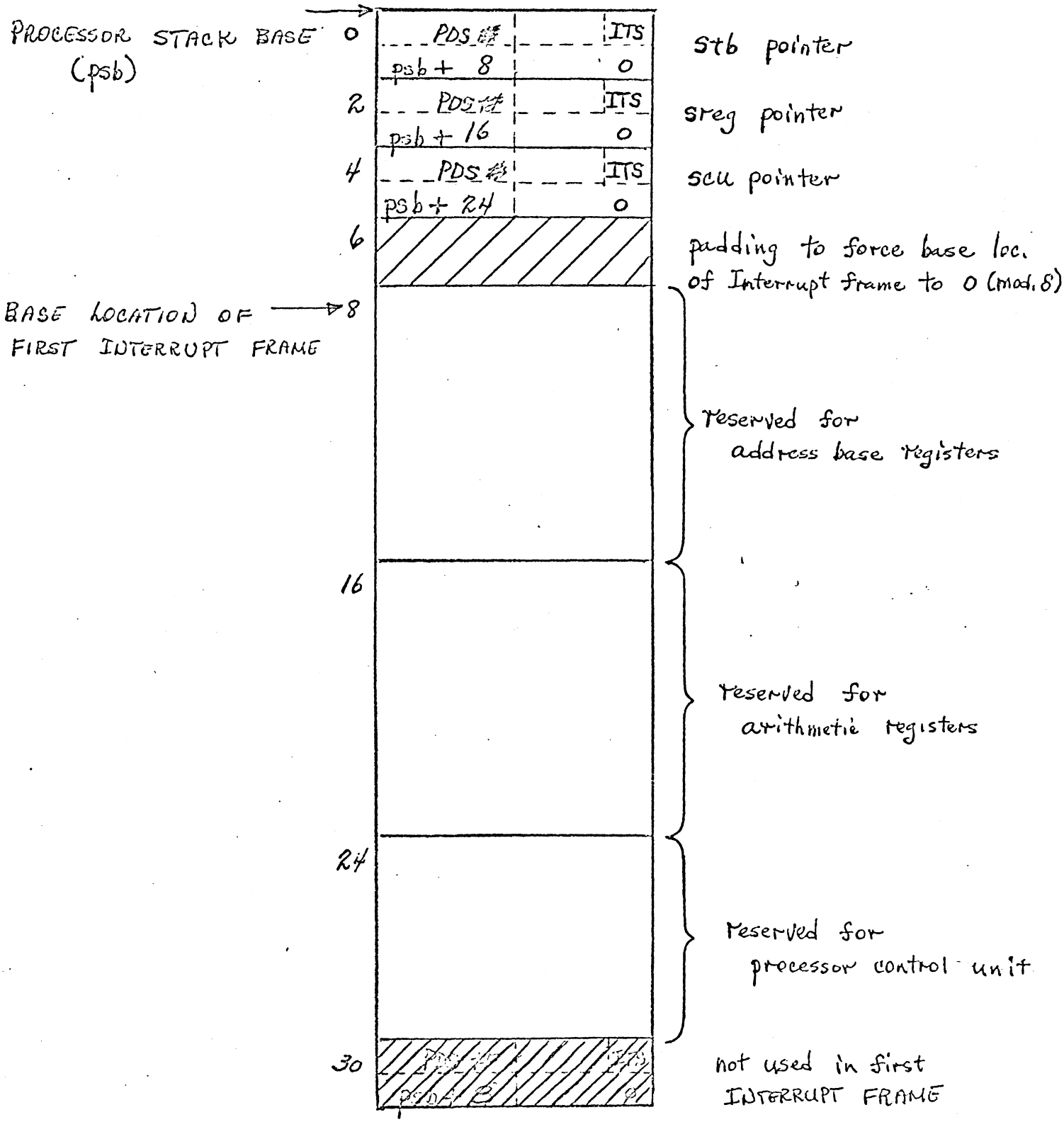

Figure 1

EMPTY PROCESSOR STACK

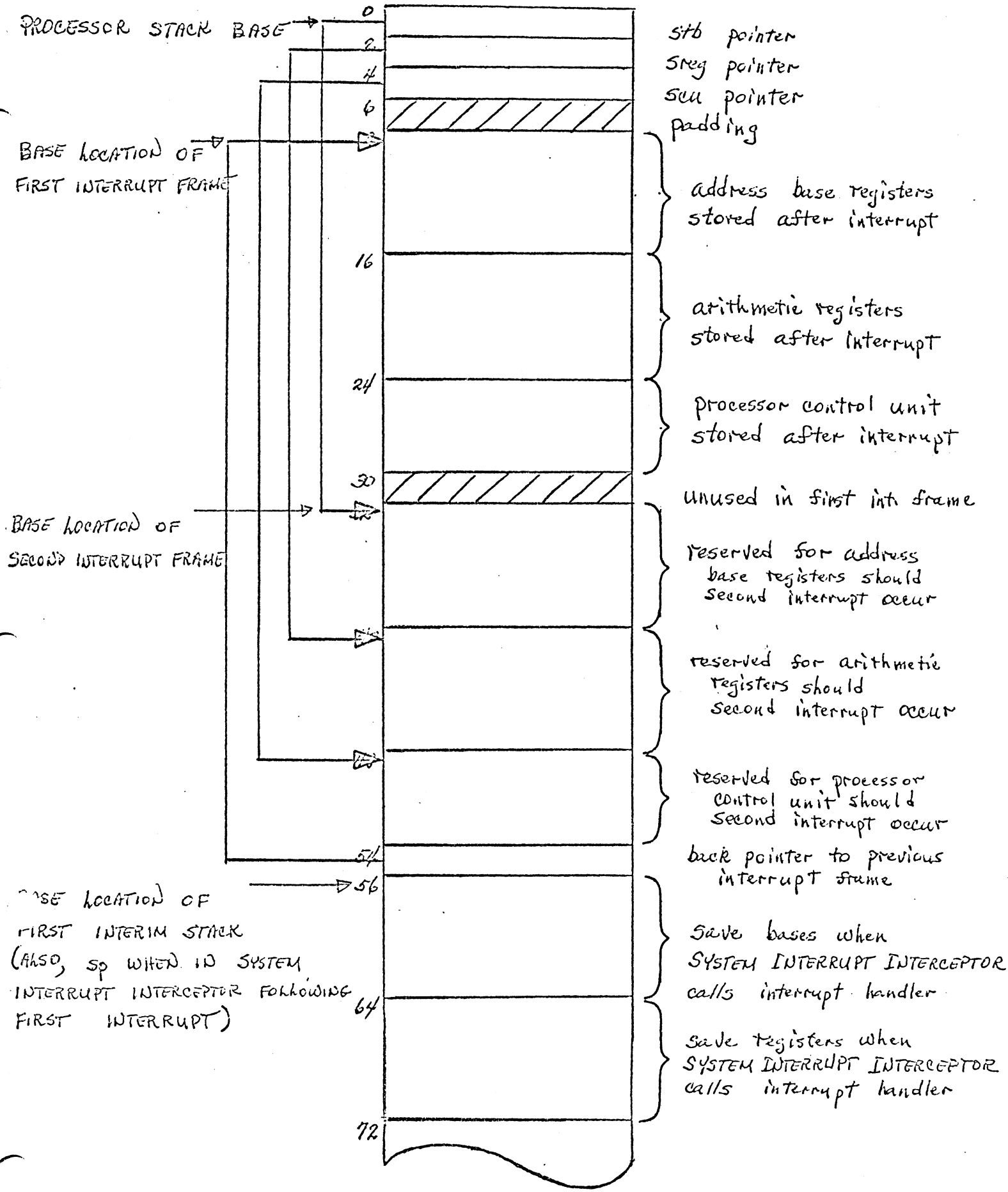

Figure 2.

PROCESSOR STACK AFTER ONE INTERRUPT

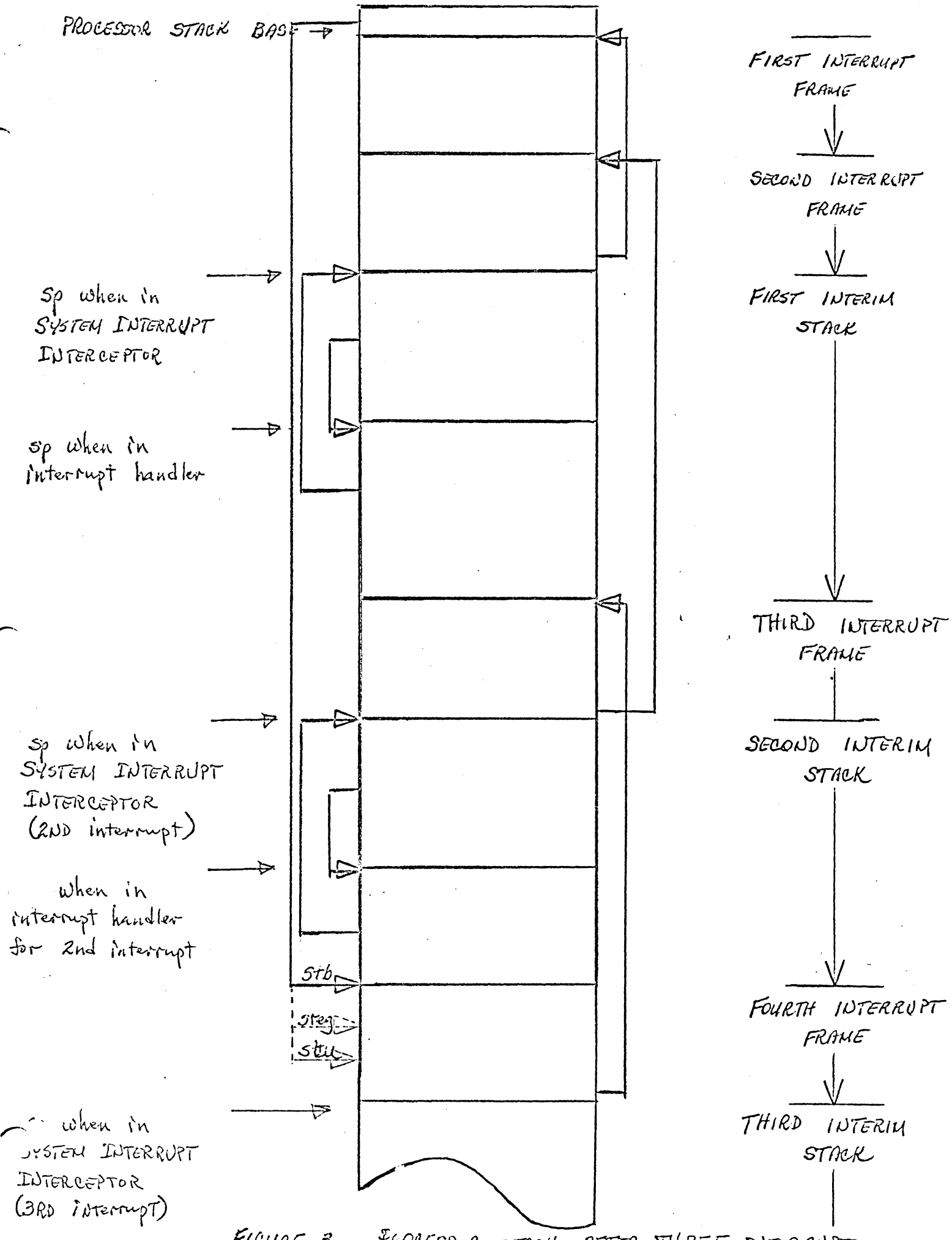

FIGURE 3. PROCESSOR STACK AFTER THREE INTERRUPTS