DRAFT - Interoffice Memorandum

0243

SUBJECT: SEGMENTATION

DATE: October 14, 1970

TO: Gordon Bell
Dick Clayton
Bruce Delagi
Dave Parnas
Bill Wulf

FROM: Ad van de Goor
Larry Wade

This memo contains a very preliminary description of a segmentation scheme for the PDP-11 family.

The scheme attempts to accomplish the following:

1) Increase the user's virtual address space to 2↑24 bytes = 4 million bytes.

2) Give a hardware definition of the "working set" model.

3) Implement "sharing" and"protection".

4) Allow processes to handle private I/O devices.

5) The scheme is usable with or without paging.

6) Provide efficient protection between processes and different segments in a process.

7) Provide storage efficiency by allowing a large range of segment sizes.

8) Allow the user to work in virtual space only.

9) Provide a physical address space of 2↑25 bytes = 8 million bytes.

0244

1.0 Basic Solutions

The address generated by the PDP-11/20 is a 16-bit byte address. The bigger members of the PDP-11 family will interpret this address (now a virtual address) as a two dimensional (segmented) address as shown in Figure 1. The 16-bit virtual address "VA" is divided into two fields.

1) The Working Segment Field "WSF". This 3-bit field determines which of the 8 working segment registers "WSR's" has to be used to form the physical address of the data or instruction. The WSR's contain, among other things, pointers to the beginning (i.e. word 0) of a segment. Appendix A lists some reasons for considering 8 WSR's adequate.

2) The Displacement Field "DF". This is a 13-bit field which contains an address relative to the beginning of a segment. This allows for segment sizes of up to 8K bytes.

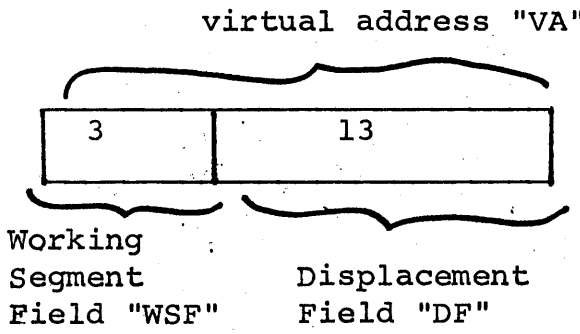
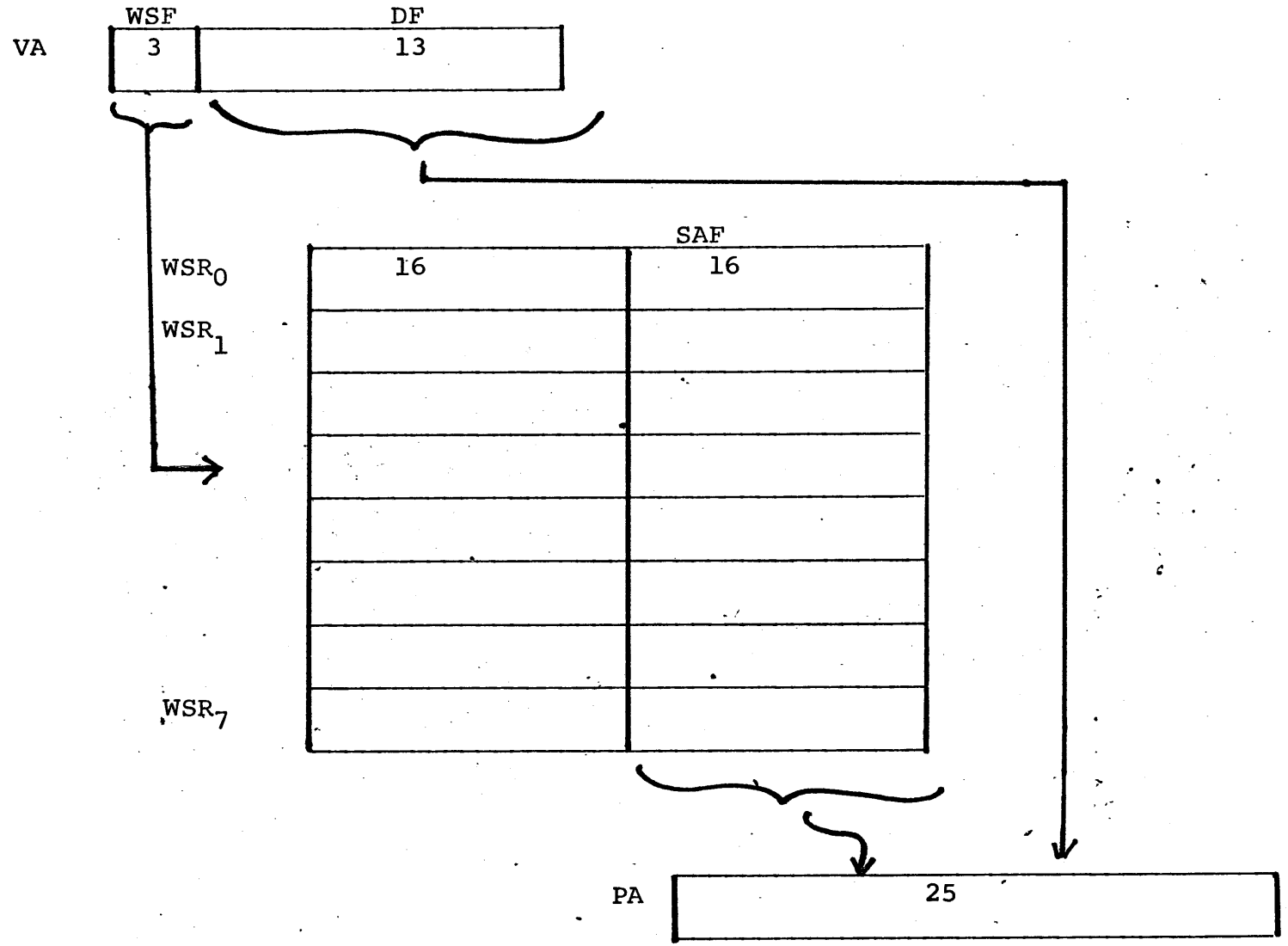


FIGURE 1. Intepretation of a Virtual Address.

The formation of a Physical Address "PA" is shown in Figure 1A. The (WSF) of the VA is used to address one of the WSR's. The Segment Address Field "SAF" of the addressed WSR is used together with the (DF)[1] to form the PA. The PA is, as will be shown later, a 25-bit byte address.

The 8 WSR's can be loaded with Segment Descriptor Words "SDW's" from the Segment Table "ST" under control of the process.

[1] Note: (X) means "the contents of X".



VA = Virtual Address
PA = Physical Address
WSF = Working Segment Field
DF = Displacement Field
WSR_i = Working Segment Register i
SAF = Segment Address Field

Figure 1A. Formation of a Physical Address.

1.1 The Segment Descriptor Word

The Segment Descriptor Word is a double word (32 bits) containing information relevant to a particular segment. It contains 5 fields as shown in Figure 2. Detailed descriptions are given in subsequent sections.

1) Use and Validity Field "UVF". This 4-bit field is the only field which is subject to change during execution of a segment.[1] The UVF consists of two sub-fields:

   a) The Use Field "UF". This is a 1-bit field which indicates whether the segment has ever been used.

   b) The Validity Field "VF". This is a 3-bit field describing the validity state of segment. These states will be discussed further on.

2) Software Use Field "SUF". This 4-bit field allows for 16 encoded states four of which are assigned already and describe the movability and size flexibility of a segment.
   a) Move Freely (can be swapped out).
   b) Move in core only (cannot be swapped out).
   c) Do not move (specifically for segments containing I/O addresses).
   d) Do allow size changes (e.g. for the stack segment).

3) Access Control Field "ACF".

   This is a 3-bit field describing whether Read, Write and Execute are allowed.

[1] All changes are done under hardware control.

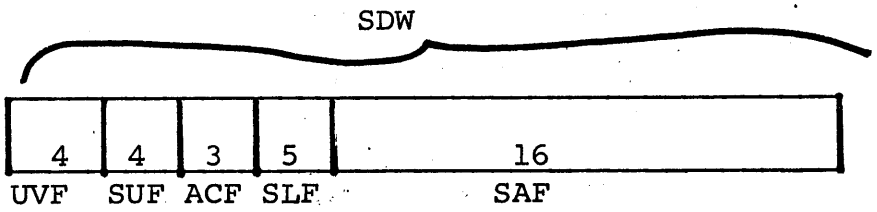
4) Segment Length Field "SLF". This is a 5-bit field describing the length of the segment.

5) Segment Address Field "SAF". This 16-bit field contains a pointer to physical word 0 of the segment.

1.1.1 The Validity States

The 8 possible validity states are encoded in the validity field VF. These 8 states are listed in Table 1 below. The column "core assigned" indicates whether any physical core has been reserved. The column "core valid" indicates whether the assigned section of core contains valid information. The column "valid copy" indicates whether a backup copy (on the disk/drum) is available.

In order to get a better understanding of Table 1, the state transitions of Table 2 should be consulted.



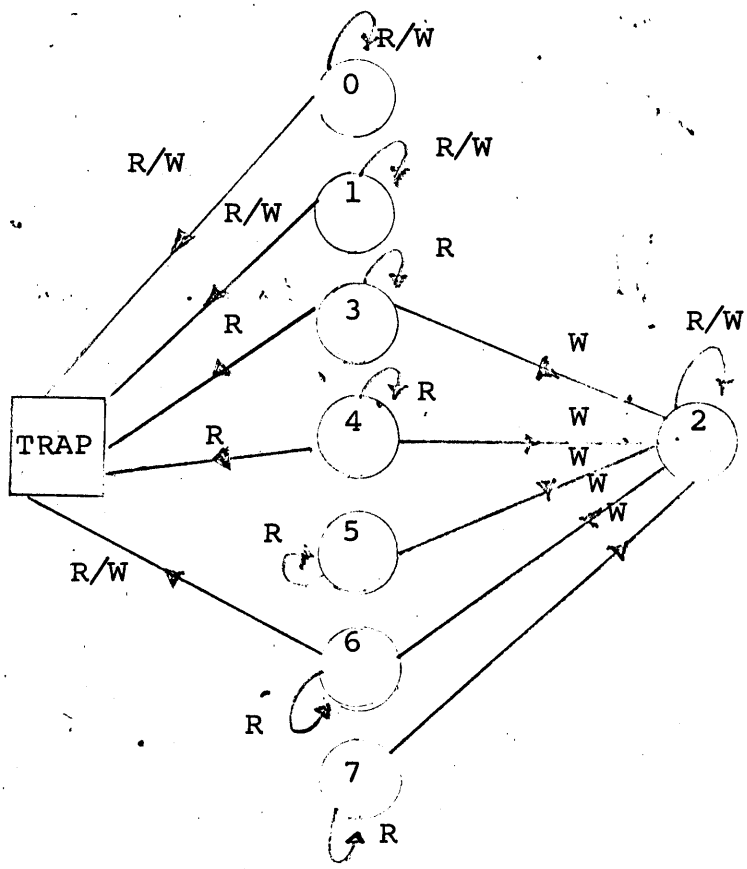
SDW = Segment Descriptor Word
UVF = Use and Validity Field
SUF = Software Use Field
ACF = Access Control Field
SLF = Segment Length Field
SAF = Segment Address Field

Figure 2. Layout of a Segment Descriptor Word.

	CORE ASSIGNED	CORE VALID	VALID COPY	TRAP AFTER WRITE	COMMENT
0	No	—	No	—	Empty Segment
1	No	—	Yes	—	Segment on backup storage
2	Yes	Yes	No	No	Core copy only
3	Yes	No	No	No	Core reserved for segment
4	Yes	No	Yes	No	Core reserved & backup copy available
5	Yes	Yes	Yes	No	State after transfer from backup
6	Yes	No	Yes	Yes	
7	Yes	Yes	Yes	Yes	

Table 1. The 8 Validity States

	READ OR EXECUTE	WRITE
0	0 Trap	0 Trap
1	1 Trap	1 Trap
2	2	2
3	3 Trap	2
4	4 Trap	2
5	5	2
6	6 Trap	2 Trap
7	7	2 Trap



NOTE: R = Read or Execute
W = Write
Trap is an ACTION - not a state.

Table 2. Validity State Transition Table & Flow Diagram

1.1.2 The Access Control States

The access control state of a segment is described in the 3-bit access control field "ACF". Table 3, below, shows the 8 states.

	READ	WRITE	EXECUTE	COMMENT
0	—	—	—	This state allows for passing segments
1	—	—	X	Execute only segment
2	—	W	—	Write only segment
3	—	W	X	Useful?
4	R	—	—	Read only data segment
5	R	—	X	"Normal" shared segment
6	R	W	—	R/W Data Segment
7	R	W	X	"Garden Variety" Segment

Table 3. Access Control States

1.1.3 The Segment Length

This is described in the 5-bit segment length field "SLF". Small segments are incorporated for storage efficiency and to allow for "private I/O", e.g. to allow users in a time-sharing system to have their own I/O devices. The meaning of the encoded bits is as shown in Table 4 below.

(SLF) = 0 means that the segment descriptor word is void, i.e. it does not describe a segment.

(SLF) = 15 indicating a shared segment, means that the SDW points to a string of SDW's (of length 1 or more) the last one of which contains the actual length of the segment.

For the smaller segments the length is a power of 2. The bigger segments, however, have a size which is a multiple of 256 words for storage efficiency reasons. (See Section 5.0)

The maximum size of a segment can be derived from the 13-bit displacement field of Figure 1. By requiring that any item in the segment be direct, byte addressable the 13-bit displacement has to be interpreted as a 13-bit byte address, limiting the maximum segment size to 2^{12} = 4096 words.

(SLF)	LENGTH OF SEGMENT IN WORDS
0	invalid segment
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	NOT USED
10	
11	
12	
13	
14	
15	shared segment

The numbers 16-31 in the SLF indicate the following lengths:

(X-15) * 256 where $16 \leq X \leq 31$

This allows for 256 word pages.

Table 4. Interpretation of the Segment Length.

1.1.4 The Segment Address

The physical address of the first word (i.e. word 0) of the segment is contained in the 16-bit segment address field "SAF". The interpretation of this 16-bit quantity is as follows.

1) If $1 \leq (SLF) \leq 8$ then the 16-bit quantity is interpreted as a word address. This means that "small" segments (i.e. those with a length between 1 and 128 words) have to be located in the first 65K words of core memory.

2) If $(SLF) > 15$ then the 16-bit quantity is interpreted as a "page address", i.e. an address of a 256 word quantity. This allows for a maximum physical word address of $2 \uparrow 16 * 2 \uparrow 8 = 2 \uparrow 24$ words or $2 \uparrow 25$ bytes.

2.0 Layout of the Segment Table

The Segment Table "ST" contains all the segment descriptor words "SDW's" belonging to a certain process.

The ST, itself, is a segment and its maximum size, therefore, is limited to $2 \uparrow 13$ bytes = 4K words. Considering the length of a SDW (4 bytes) the ST can contain $2 \uparrow 11 = 2K$ SDW's maximally. This gives a maximum virtual memory per process of: (max. segment size) * (max. # of segments) = $2 \uparrow 13 * 2 \uparrow 11 = 2 \uparrow 24$ bytes.

The layout of the ST is shown in Figure 3. The top 16 words of the ST are not used to store SDW's for reasons to be explained later. Currently these words are used as follows.

1) The first 8 words (W0-W7) are used to contain Segment Numbers "S#'s". A S# of j in Wi of Figure 3 indicates that WSRi is loaded with SDW#j. So the S#'s loaded into W0-W7 of the ST indicate the SDW's the WSR's are loaded with. Because the maximum # of SDW's in a ST is 2↑11 a S# does not have to be bigger than 11 bits.

2) Word #10_8 (W10) contains the stack pointer (R6) when the process is inactive.

3) The remainder of the words (W11-W17) are reserved for software use.

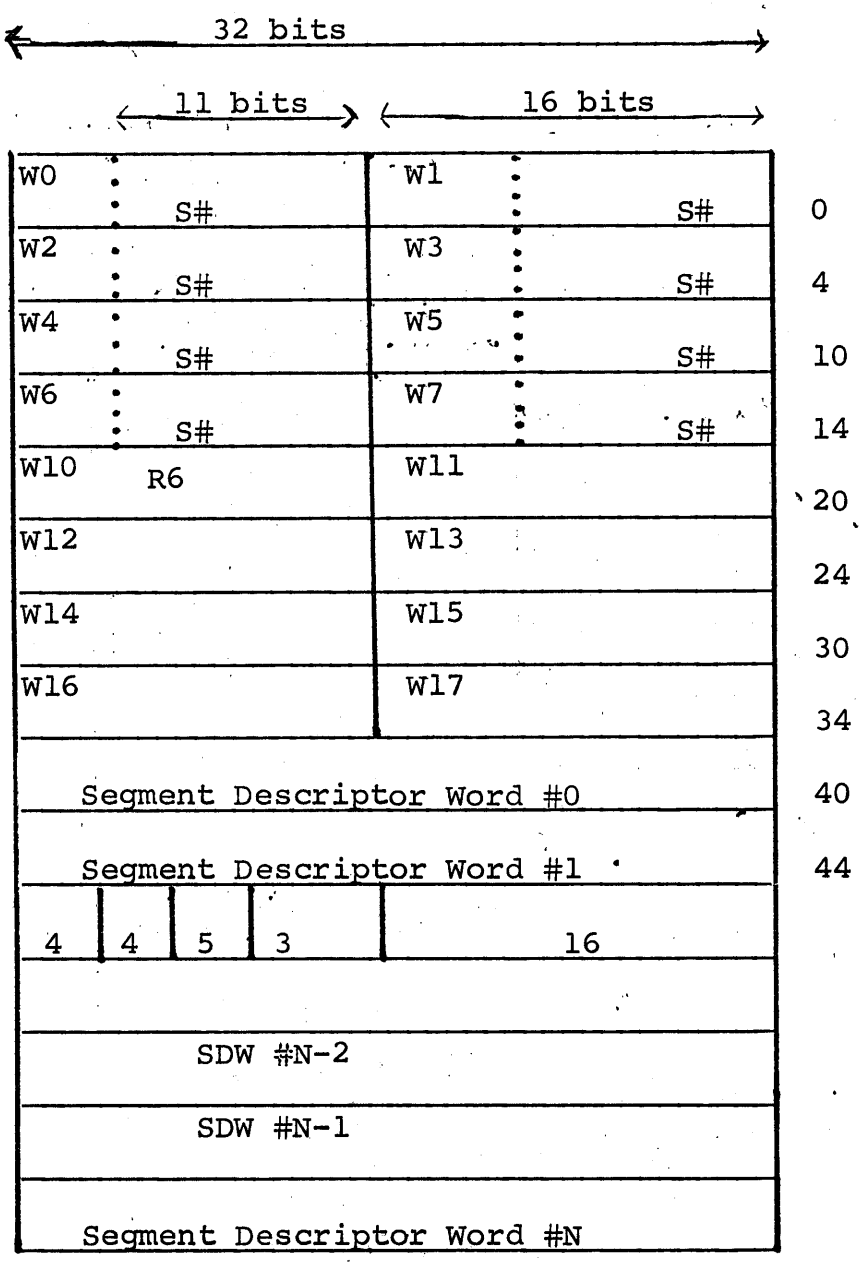
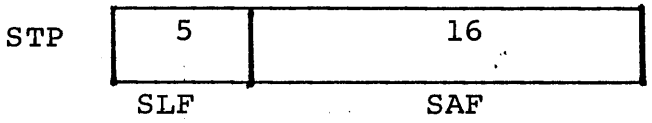


Figure 3. Layout of Segment Table

3.0 Master Control Process "MCP"

This process has the authority to allocate and de-allocate resources in the system. Core management and the creation and deletion of segments belong to its responsibility. This is the only process which is allowed to add, delete, or modify ST's and SDW's. Other processes have no control over their ST and SDW's.

Every process in the system is completely specified by its ST. When a process is in control, a hardware register, the Segment Table Pointer "STP", points to the ST of the process. The STP is a 21-bit register (see Figure 4) and has the same layout as the low order 21 bits of the SDW of Figure 2.



SLF = Segment Length Field
STP = Segment Table Pointer
SAF = Segment Address Field

Figure 4. Layout of the Segment Table Pointer "STP"

The MCP has a special segment, called the Segment-Segment Table, "SST", which contains Segment Table Descriptor Words "STDW's" pointing to all processes in the system, including the MCP. The location of the SST is known to the MCP because

it is one of its segments. Figure 5 shows the layouts of the different tables. The SST contains M STDW's indicating that there are M processes in the system.

Process 0 "Pr.0" is the MCP. Note that the SST is the first segment in the MCP's ST and is therefore under complete control of the MCP. It is quite obvious that the SST should not be a shared segment. The process in control is Pr. 1 because the STP (Segment Table Pointer) points to it.

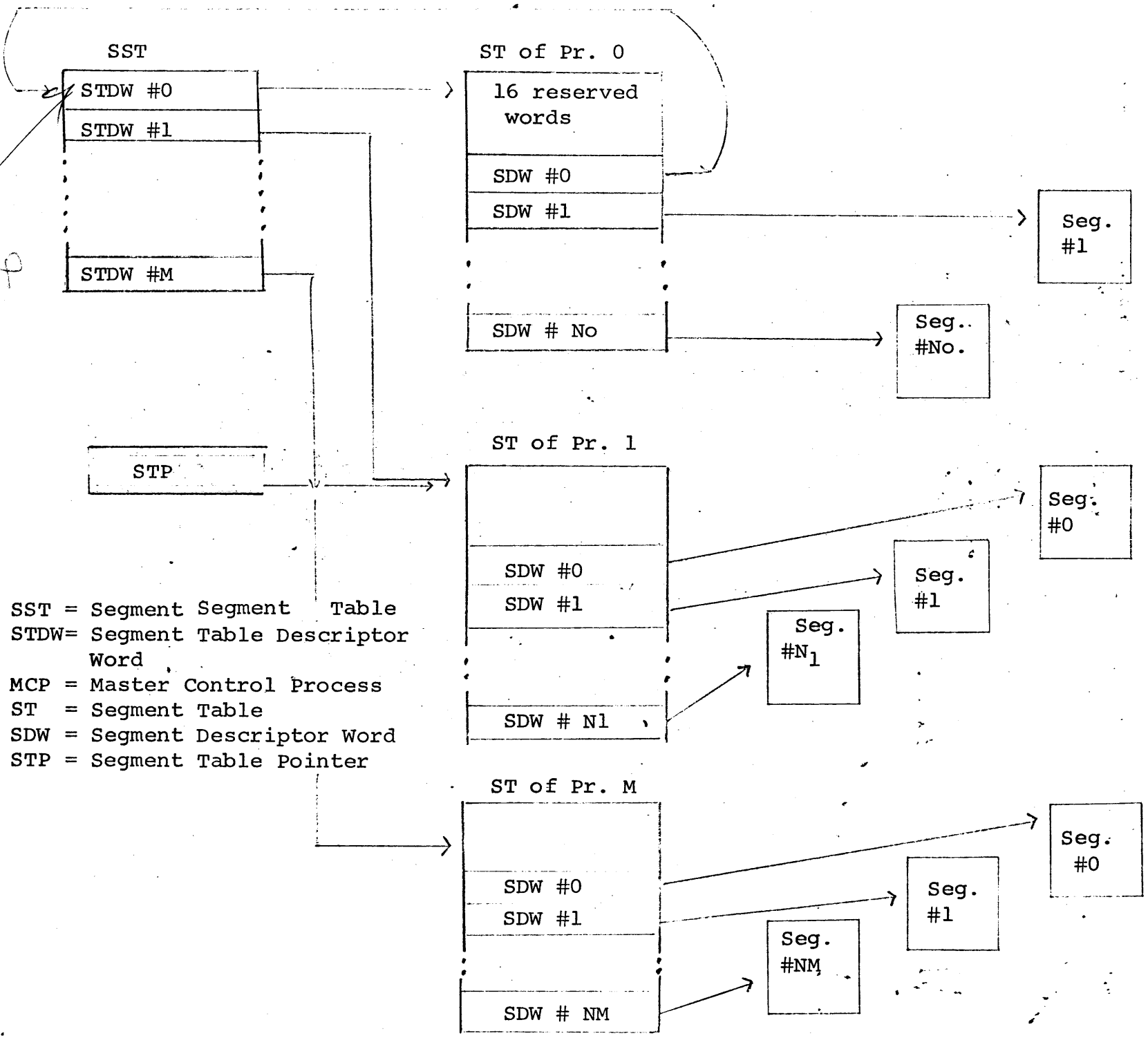


Figure 5. Layout of SST and ST Tables.

4.0 Interruptability

Fast interrupt response is a requirement especially because the machine might be used in real-time applications. The state of a running process is determined by the following:

1) The program counter "PC"
2) The stack pointer "SP"
3) The program status word "PS"
4) The location of the ST which is the (STP)
5) The contents of the 8 WSR's
6) The contents of the accumulators "AC's"

The interrupt response time can be divided into two groups:

1) The time needed to save the current status, called "Save Time".
2) The time needed to set up the new status, called the "Restore Time".

4.1 Reduce Save Time

In order to reduce the Save Time, the following two facilities are introduced:

1) The saving of the AC's is done optionally through Save AC's bit "SAC" in the PS word (see Figure).

2) The saving of the WSR's is made not necessary because of the scheme discussed below. In order to allow for this, two requirements have to be satisfied.

   a) Duplicate copies of the contents of the WSR's have to be available in core memory.

   b) Knowledge as to which SDW's are loaded in which WSR's has to be available to allow for a correct restore operation.

Part a) is satisfied by guaranteeing that the (WSR's) are always the same as the corresponding SDW's in the ST. This can be done relatively easily at the expense of very little overhead because the SDW's do not change very often when they are loaded in the WSR's. Only 4 bits of the SDW can change while a "segment is working"(i.e. loaded in a WSR). These are the Use and Validity bits (see Section 1.1)

   a1) The Use bit changes,at most, once while the segment is working, namely when it is used the first time.

   a2) The validity bits can change only a few times after which they end up in a stable state or cause a trap (see Table 2).

Because the changes mentioned under a1 and a2 are so infrequent, they are made in the WSR and the corresponding SDW simultaneously (i.e. in a non-interruptable sequence).

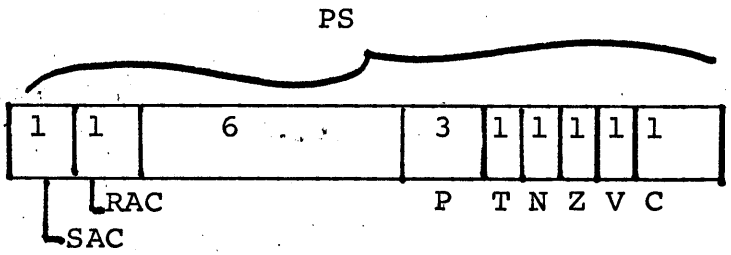
Part b) is satisfied by reserving in the ST 8 words which contain the SDW #'s loaded in the corresponding WSR's (see Figure 3).

The additional requirement in loading a WSR is that the SDW # has to be loaded in the corresponding entry in the ST.

4.2 Reduce the Restore Time

The restore time can be reduced by

1) Conditionally restore the AC's. This is done through the Restore AC bit "RAC" in the PS word (see Figure 4).
2) Selectively restore the WSR's. This is done through an 8-bit mask, the Restore WSR mask "RWSR", in the Virtual Memory Descriptor "VMD" of Figure 5.
3) Conditionally Change Address Space.
   This is done through the change address space bit "CAS" in the VMD. The exact operation of this bit needs some more work.



SAC = Save AC's

RAC = Restore AC's

P = Priority

T = Trace

N = Negative

Z = Zero

V = Overflow

C = Carry

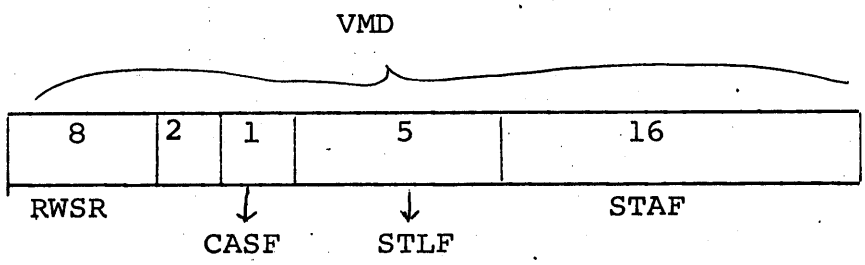
PS = Program Status Word

Figure 4. Layout of the Program Status Word

5.0 Interrupts and Traps

The interrupt and trap vectors, as currently exist on the PDP-11/20, have to be redefined in order to guarantee efficient operation. Instead of the "old" interrupt/trap vectors consisting of a PC and a PS word, we will now have a Virtual Memory Descriptor "VMD", see Figure 5. These VMD's are located in physical core, they are also 2 words long, and can therefore replace the interrupt/trap vectors.

The VMD contains all the information necessary to start a process operating in virtual memory, rather than interrupt/trap handlers operating in physical address space. The above feature allows processes to handle their own interrupts/traps.



VMD = Virtual Memory Descriptor
RWSR = Restore Working Segment Register Mask
CASF = Change Address Space Field
STLF = Segment Table Length Field
STAF = Segment Table Address Field

Note: The STLF is similar to the SLF.
The STAF is similar to the SAF.

Figure 5. Layout of the Virtual Memory Descriptor

The saving of the state of an interrupted process consists of the following steps.

1) Test the SAC bit in the PS (see Figure 4) and conditionally push the AC's on the stack of the interrupted process.
2) Push the PC and PS on the stack of the interrupted process.
3) Store SP in $W10_8$ of the ST of the interrupted process.
4) Invalidate the WSR's by clearing them. This is to safeguard the interrupting process from accidentally being able to access the interrupted process's VM space.

Restoring the state of the interrupting process consists of the following steps.

1) Store (STP) in a temporary location "TSTP".
2) Pick up VMD from interrupt/trap vector and store it in the STP.
3) Store (TSTP) in W12 and W13 of the ST of the interrupting VM space.
4) Restore R6 from W10 of the ST of the interrupting process.

5) Pop PS and PC.

6) Test RAC bit of popped PS and conditionally restore the AC's by popping them from the stack.

7) Selectively restore the WSR's under control of the RWSR mask in the VMD.

6.0 Indirect Addressing and Shared Segments

Instruction as well as data addresses are virtual addresses "VA's". Because the (WSF's)[1] can be different for instructions and data, instructions and addresses can come from different segments The two possible cases for direct addressing are shown in Figure 6.

6.1 Indirect Addressing

Indirect Addressing is handled in a way very similar to direct addressing. Now, however, three VA's are generated: 1 for the instruction; 1 for the indirect address; and 1 for the data. This leads to the five possible addressing cases shown in Figure 7.

[1] See Figure 1 and 1A.

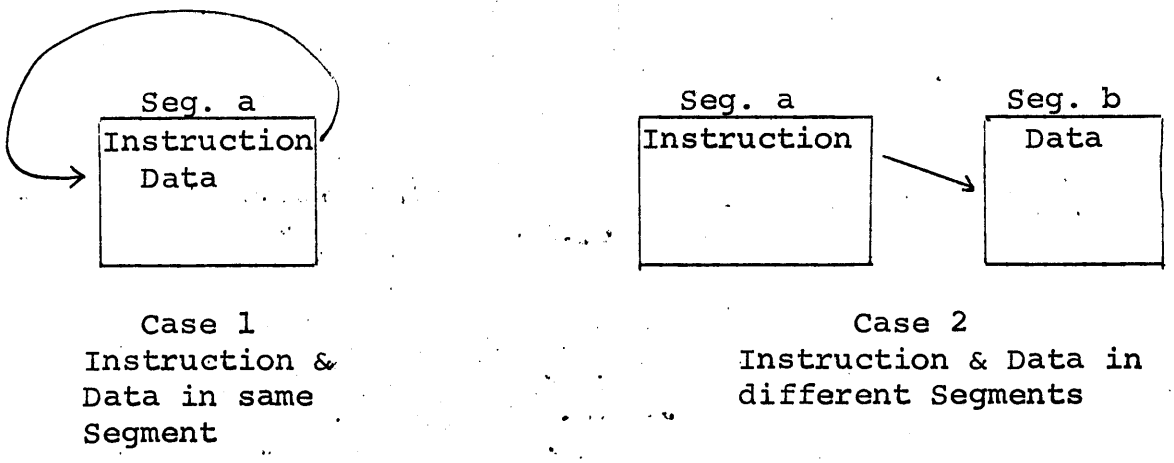


Figure 6. Direct Addressing Cases

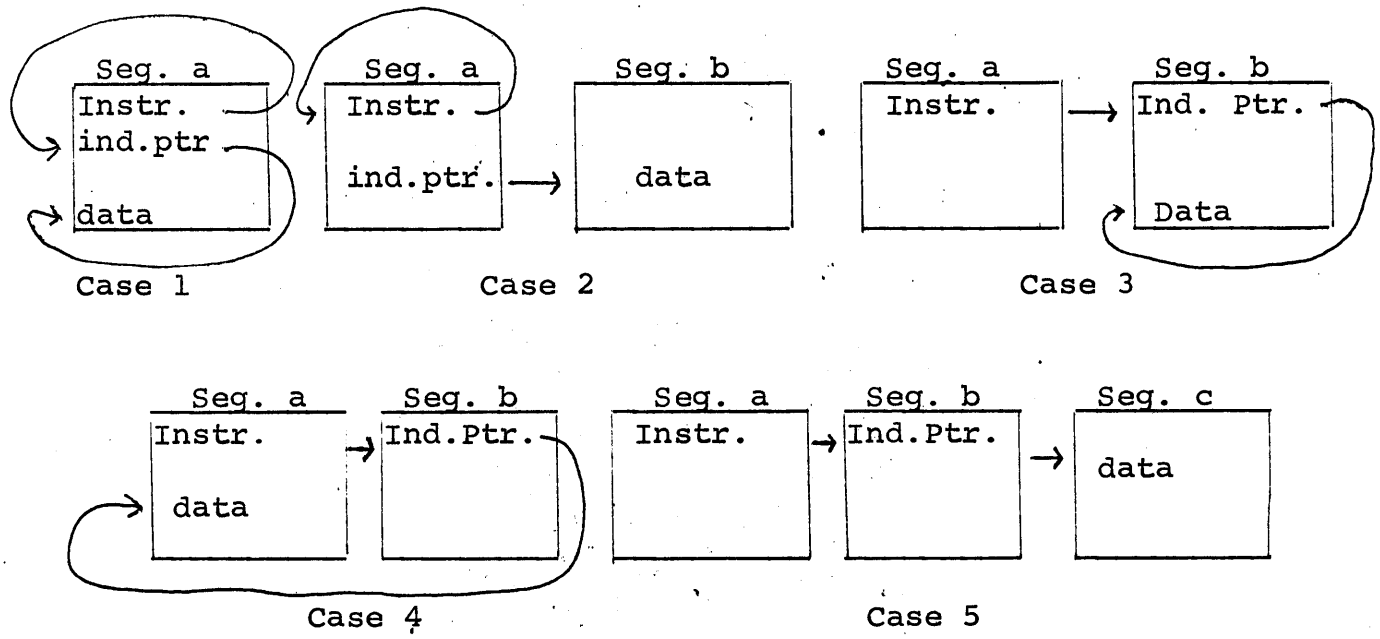


Figure 7. Indirect Addressing Cases