Revised 2/20/70, as Hardware Memo 2.

by J. Holloway

THE MEMORY PAGING DEVICE

General Description:

The Project MAC AI Group is currently constructing a memory paging device for the PDP-10. This is intended to be used primarily by the time-sharing system. The PDP-6 and the PDP-10 currently use a protection-relocation memory mapping scheme. A relocation register is used to offset virtual addresses generated by user programs to the absolute address inside the block of memory assigned to him. A protection register prevents reference to locations outside the user's block. A page-addressing scheme divides the user's virtual memory into small blocks or pages of fixed length, usually 1024 words. Each block has a page address contained in a page map that points to the absolute location of the page in memory. The map also contains an access permit bit that prevents reference to virtual pages that are not assigned to any location in memory. The user may see the memory accessible to him as occupying a homogeneous block in virtual memory, but the actual pages are located in any order and location. The purpose for providing a memory paging device is directed toward several problems. (See Figure 1.)

1. Efficient Core Utilization:

Currently all existent jobs for all users reside in core simultaneously. Since several users may be using the editor, DDT, MIDAS, LISP, etc., there are several essentially identical copies of the unchanging portion of each system program in core. It would only be necessary to have one copy of the pure portion of the procedure with data areas for each individual user. This can be accomplished by:

Modifying the program so that it makes all impure references by means of a base register;

Providing dual sets of protection-relocation registers for reference to pure and impure areas;

Or in a paging scheme where each user's map points to the same absolute pages for the pure area, and points to impure pages unique to each user.

2. More Efficient Core Management:

The current ITS contains a core allocator. It responds to user requests to increase or decrease the size of allocated memory. Its job is complicated by the requirement that the block of memory for each user occupy a contiguous area in core storage. It may find that it has to move one or several users in order to contiguate enough storage for an expanding user. This "core shuffling" would not be required with the paging system. Although

a negligible portion of actual machine time is spent moving users on the average, embarrassing delays sometimes result when large shuffles take place. The amount of core allocation overhead would probably tend to increase if users were swapped in and out of core.

3. More Flexibility in Referencing User Core Images:

Currently all transfers between user core images are done by calls to the system, invoking an appreciable overhead. With a paging device superior procedures could reference inferior locations directly by apending the inferiors' map to their own. Jobs could also reference portions of the system data areas or other user jobs after establishing the needed transformation in their map.

4. Segmentation Ability:

It is not necessary that the virtual pages allocated to the user be contiguous. Different data spaces that would like to expand dynamically could be separated in virtual memory and expansion of a segment would only require adding to the page map. Therefore, old addresses referencing these areas would be unchanged. This obviates the need for compacting garbage collectors and their ilk.

5. Demand Paging:

Pages nominally allocated to a user may not actually exist in core storage when the user is active, and the reference to the missing page would trap to the system and the user program would be continued when the page was recovered from the Disk. This scheme has dubious advantages but would be impossible without a paging device.

6. More Complex Access Control:

Page entries in the map can contain information describing the type of access permitted; for instance: NOT ACCESSIBLE, READ ONLY, NOT EXECUTABLE by pure procedures, etc.

IMPLEMENTATION

The PDP-10 has an 18-bit address. If virtual memory is divided into pages 1024 words long, memory will contain 256 pages. The address is logically divided into an 8-bit page number, and a ten-bit word address. The map must contain, per entry, 8 bits of page address and one or two bits of access code. The map requires 256 page registers or approximately 2,500 bits. This would require over 500 I.C.'s for the memory and would take over 200 usecs to load or store in memory.

In order to reduce the number of registers required to store page information, it is common to use an associative memory. Typically, the page map is stored in core, and copies of up to 16 of the page entries are stored in the AM. A reference to any page other than those described in the AM requires a reference to the page map in core, and one of the 16 AM words is replaced by the new information. In order to be efficient, the program must reference this page several times before it is replaced by some other page entry. The number of associative registers, the replacement algorithn, the size of the pages with respect to the size of the virtual memory, and the statistics of each program's memory references affect the efficiency of execution.

A simulation program was written for the PDP-6 to compare various paging techniques. This program interpreted PDP-6 programs and recorded the page replacement rate as a function of the number of associative registers and size of pages. The results are shown in Figure 2. The replacement rate is the number of legitimate references per reference to the page map in core. Two replacement algorithms were used:

1. The "oldest" page, in terms of the longest time since referenced, was replaced (Usage Counter).
2. Pages were chosen sequentially in the AM for replacement (Ring Buffer).

It appears that for an eighty-page program, efficiency might be assumed to be:

NUMBER OF ASSOCIATIVE REGISTERS	EFFICIENCY
4	87+%
8	96+%
16	98+%
32	Approx. 100%

It appears that the ring buffer replacement generates approximately one-third more page map references than the usage counter scheme. Of course, this efficiency is not guaranteed, and in some cases a program can function arbitrarily badly. If the main loop of a program references consecutively more than 16 different pages, it will never find the required page in the AM.

THE MAC PAGING SYSTEM

The Associative Memory

A 16-register associative memory is provided (see Figure 3), matching on the high order 8 bits (VMA 18-25) of the address plus one bit matching on the USER/EXEC mode. The output of the AM is 9 bits of address (MA 17-25) providing for a 19-bit address (512K). Two condition code bits (cc0, cc1) signify whether the page is: 00 NOT ACCESSIBLE, 01 READ ONLY, 10 READ/WRITE FIRST, 11 READ/WRITE. If the PURE bit of the PC word is a one then instruction fetches from READ/WRITE pages are trapped. READ/WRITE FIRST is used by the EXEC to detect if pages have been written into, so that it need not swap out unaltered pages.

Two additional 16-word memories are provided, the B Memory (BM) and the C Memory (CM). These are Block Associative Memories (BAM), and they consist of a single 5-bit associative match (VMA 18-21 + EXEC mode) keying a block of 16 pages. Each of these BAM's map a contiguous 16K block in virtual memory into sixteen 1K pages. Their usefulness is predicated on two facts.

1. Few programs require more than 48K of virtual memory, which is provided by AM, BM, and CM without page map updates.
2. Normally it can be predicted which area of a program will be referenced often, i.e. the area instruction fetches are directed to, or critical data areas. Usage of the BAM's will insulate the AM from a large percentage of the paging demands.

The location of the page map in memory is specified by three Descriptor Base Registers (DBR). DBR 1 is for virtual locations 0-377777 (pages 0-177_ of the user memory, DBR 2 is for locations 400000-777777 of the user memory, and DBR 3 is for 400000-777777 of the EXEC. Each DBR has a length register (DBRL) which is 7 bits long and describes the maiximum number of words in each of the three page maps. The page entries are stored two per word in the map, with even page numbers in the left half (see Figure 4).

At the beginning of each instruction that is going to be mapped, the program counter (PC) of the machine is stored in the old PC register (OPC). If a fault occurs, the instruction execution is aborted before any location becomes changed, but the PC may have been destroyed along with some of the PC word flags (AROV, BIS flag, etc.). The OPC register is used to restore machine

conditions to those just prior to the faulted instruction so that it may be re-executed later on. There is an 8-bit register (FA) which retains the number of the page that caused the fault.

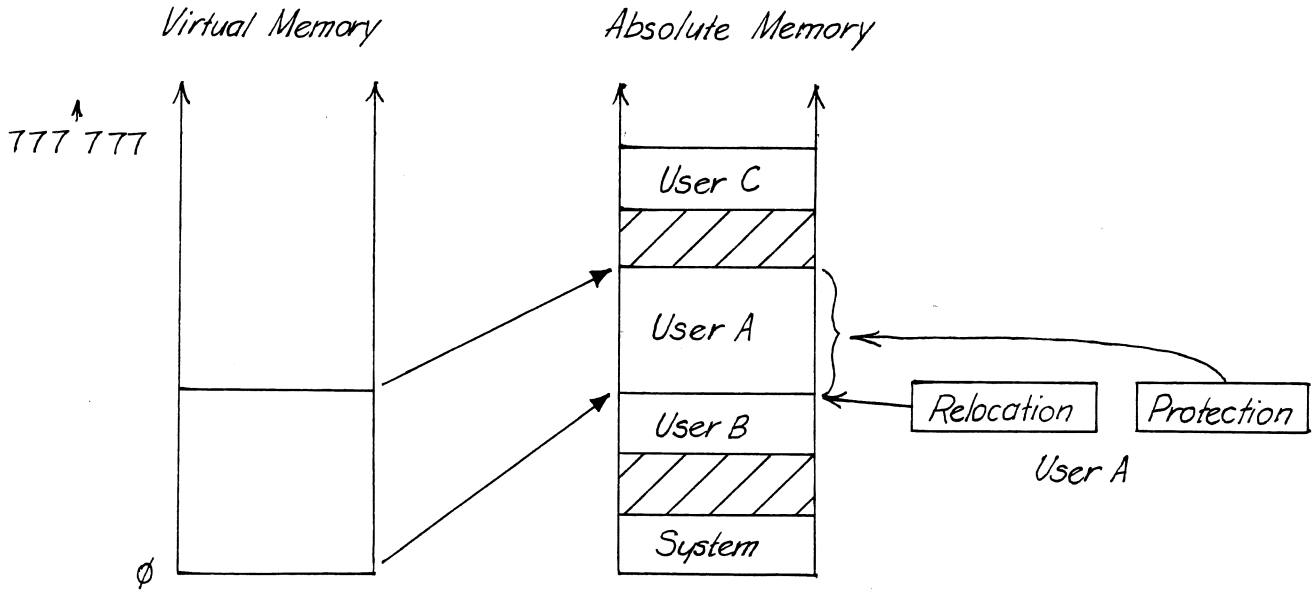
If at any time during the execution of a user mode instruction, the PC is changed by the successful execution of a transfer instruction, the OPC register is immediately saved in the Jump PC (JPC) register. This is merely a debugging feature for user programs.

There is an Execute Relocated Instruction (XCTR) which is used by the EXEC to reference user core images. This instruction is similar to an Execute, but depending upon bits in the AC field of the instruction (bits 9-12) fetch, store, or AC references of the instruction executed are relocated as if in user mode. Bit 10 (AC 4) specifies relocated fetch cycles, bit 11 (AC 2) relocates store cycles, and if bit 12 (AC 1) is on, then all AC index registers references will go to a block of 16 locations specified by an AC pointer (ACP). This allows uniform reference to user addresses, independent of whether they are located in the user state word or is some scattered page. If a fault occurs during an XCTR (except for READ/WRITE FIRST conditions), an interrupt will not be generated, but the floating overflow flag will be set. This may be tested by a JFCL 1,.

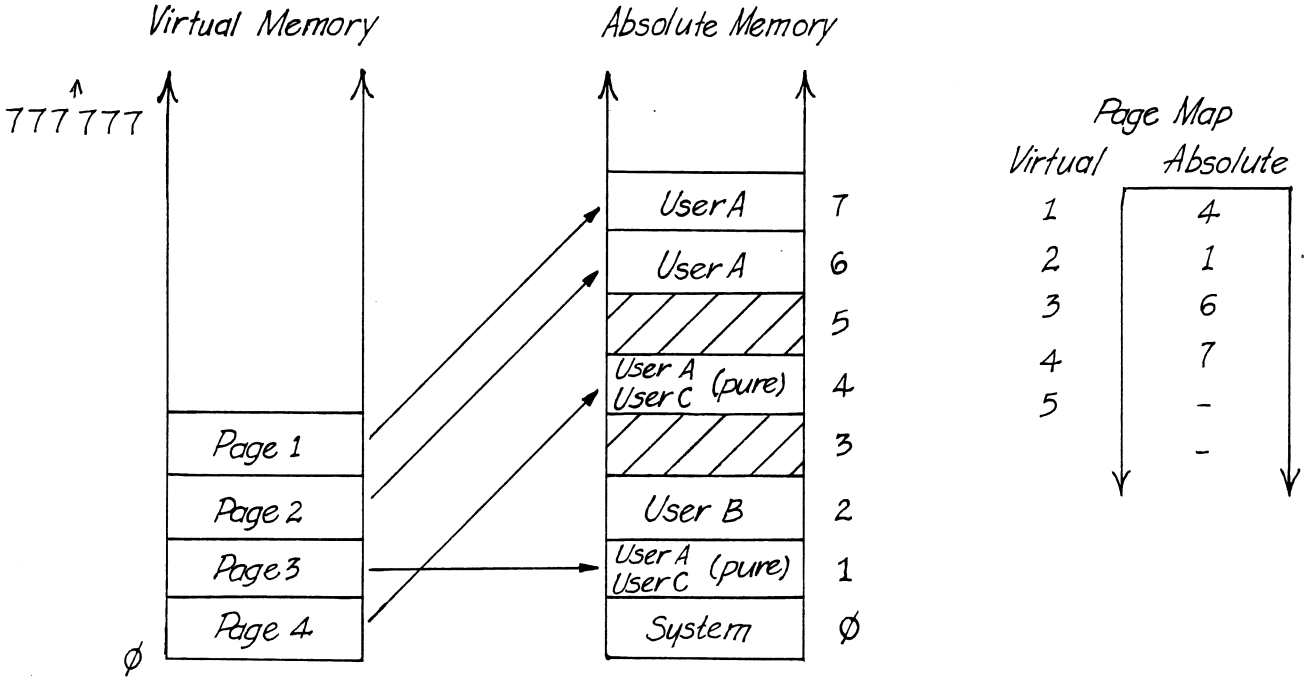
There is an 18-bit quantum counter, that counts at 1 MHz except when the machine is PI in progress. When this timer counts out, it generates an interrupt on the processor channel.

There is a memory address register (MAR) which is 18 bits plus two condition bits. The condition bits specify Interrupt on: 00 - always, 01 - on write cycles, 10 - on PC fetches, 11 - never. If the condition is true when the location pointed to by the MAR is referenced, an interrupt will occur.

There are two instructions, Load Page Memory (LPM), and Store Page Memory (SPM), which transfer eight words of random registers at the location specified by their effective address (see Figure 5).



Protection, Relocation Hardware



Paging

FIG. 1

REPLACEMENT RATE

	No. Pages	No. Associative Registers			
		4	8	16	32
LISP	40	12	64	1800	-----
LISP	80	11	35	122	40,000
LISP	80R*	7	25	101	11,000
LISP	160	8	22	62	410
LISP COMPILED	160	6	24	245	718
LISP	160R*	6	16	43	223
MIDAS	32	21	100	1200	40,000
MIDAS	64	13	48	162	3,600
MIDAS	128	9	24	73	1,200
Average		9	40	423	12,144

*R means the replacement algorithm was Ring Buffer.

Figure 2.

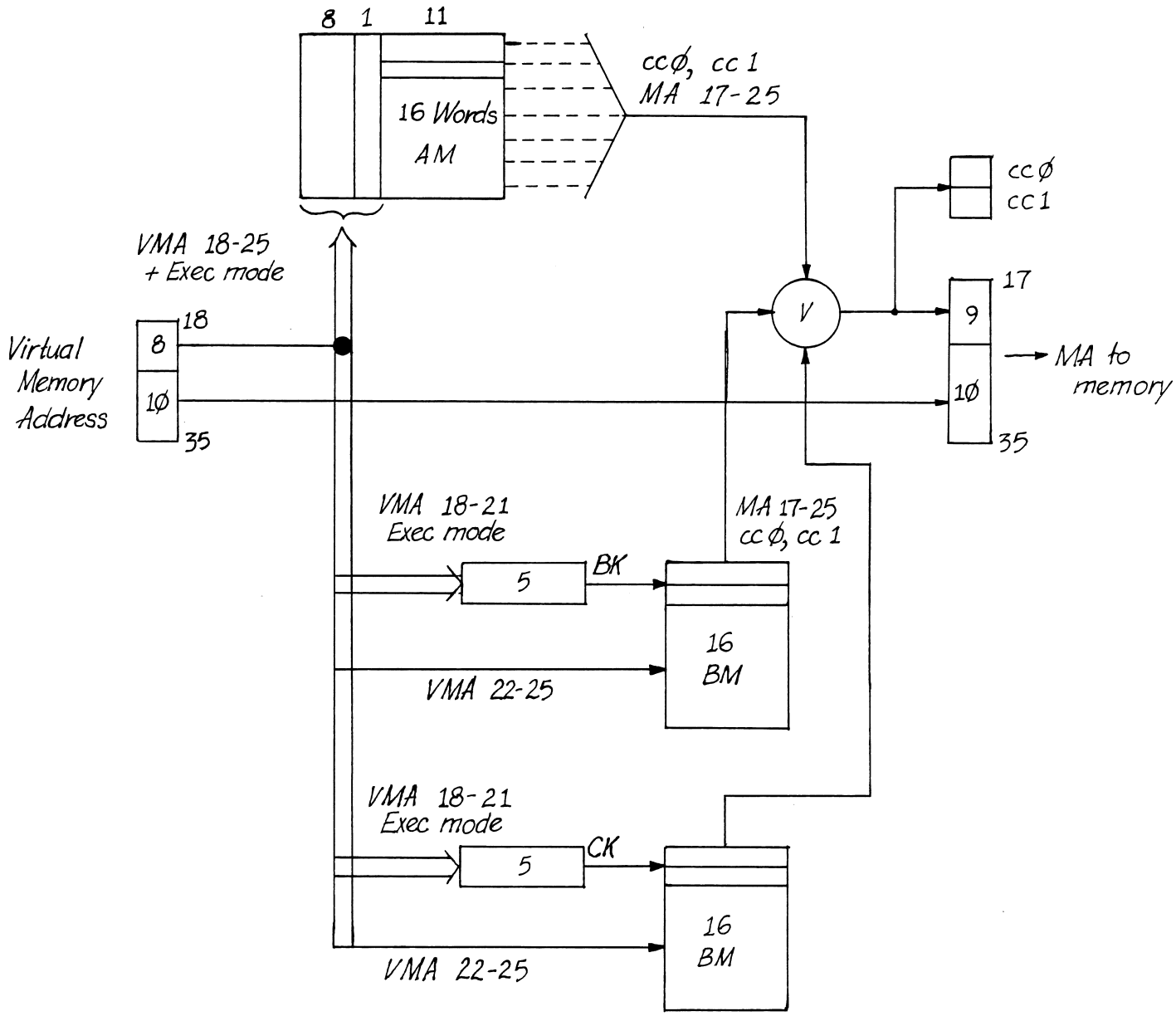

8
1
11
16 Words
AM
cc∅, cc 1
MA 17-25
cc∅
cc1
17
9
10
35
MA to memory
V
VMA 18-25
+ Exec mode
18
8
10
35
Virtual Memory Address
VMA 18-21
Exec mode
5
BK
16
BM
VMA 22-25
MA 17-25
cc∅, cc 1
VMA 18-21
Exec mode
5
CK
16
BM
VMA 22-25

FIG. 3

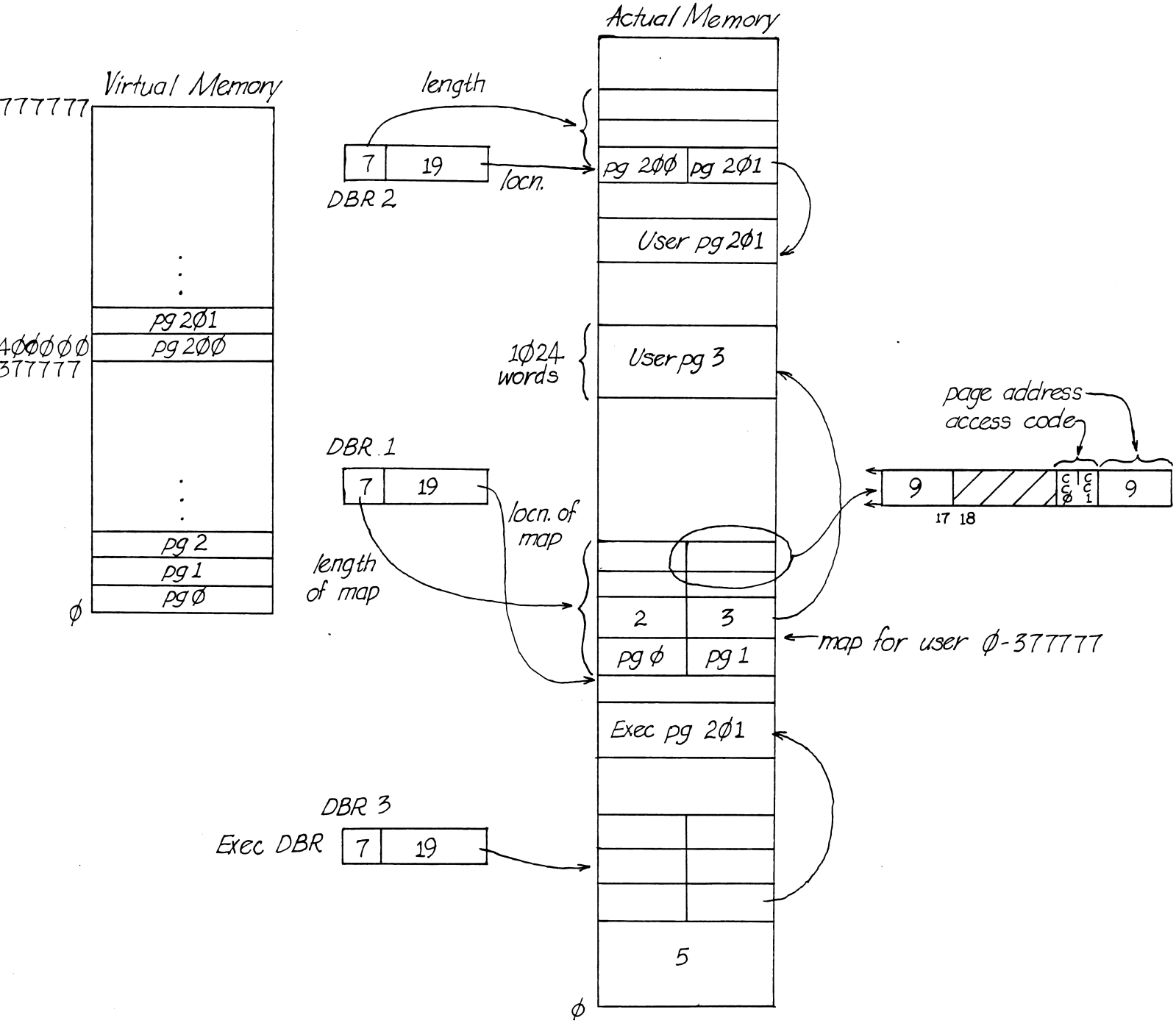
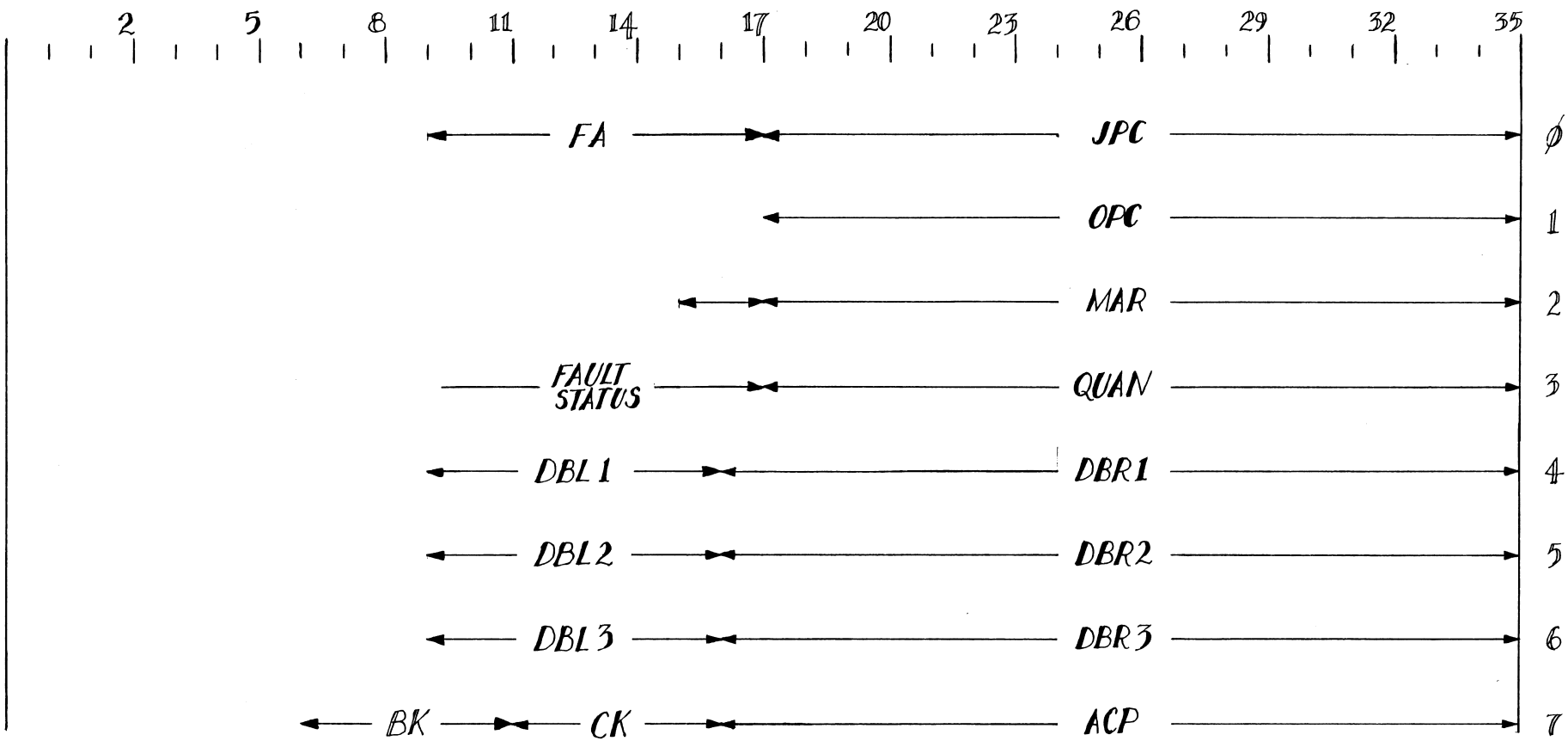

Actual Memory
Virtual Memory
length
7
19
DBR 2
locn.
pg 2ØØ
pg 2Ø1
User pg 2Ø1
1Ø24 words
User pg 3
page address access code
DBR 1
locn. of map
length of map
9
9
17 18
2
3
pg Ø
pg 1
map for user Ø-377777
Exec pg 2Ø1
DBR 3
Exec DBR
5
Ø
777777
4ØØØØØ
377777
pg 2Ø1
pg 2ØØ
pg 2
pg 1
pg Ø
Ø

FIG. 4



LPM ADR
SPM ADR

Format of Words Stored

FIG 5

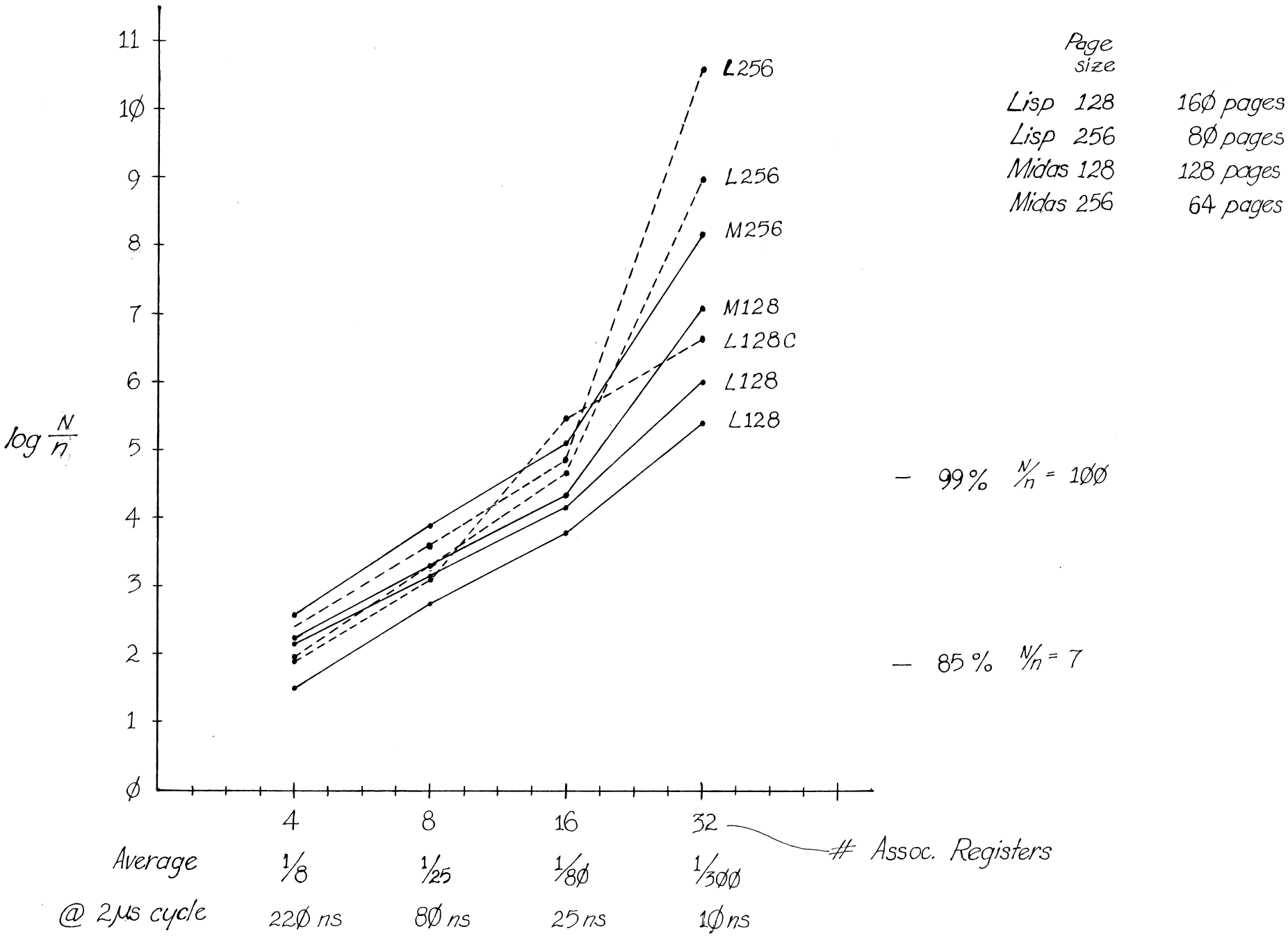

11
10
9
8
7
6
5
4
3
2
1
0
log N/n
L256
L256
M256
M128
L128C
L128
L128
Page size
Lisp 128 160 pages
Lisp 256 80 pages
Midas 128 128 pages
Midas 256 64 pages
— 99% N/n = 100
— 85% N/n = 7
4
8
16
32
Assoc. Registers
Average
1/8
1/25
1/80
1/300
@ 2μs cycle
220 ns
80 ns
25 ns
10 ns

FIG. 6