A DOCUMENT ON THE KI10

A DOCUMENT ON THE KI10
MOTIVATION FOR DOCUMENT

MOTIVATION FOR DOCUMENT

The purpose of this document is to present a medium level discussion on the KI10 system covering as many aspects of the hardware and software as possible. It is the result of a seminar given by David Braithwaite to other Northeast Region software specialists on April 1973. It gathers together, organizes, and clarifies most KI10 documentation available.

David is to be commended for his own initiative in writing this document. Educational Services has edited and rewritten certain portions of the text. However, the document for the most part remains virtually intact. The references used were:

System Reference Manual
KI10 Processor Manual
Technical Summary
Large Buffer 188 (Internal Document)
Monitor Listings

GENERAL DESCRIPTION OF A KI10.

The motivation for this section is to give simple yet revealing answers to the question "what can you tell me about the KI10?"

Introductory Description

The KI10 is the processor utilized in the 1060, 1070, and the 1077 configurations. It offers virtual memory capabilities, double precision floating point hardware and a large address space. The virtual memory hardware segments a program into a series of fixed length (512 word) pages. These pages can be scattered throughout core memory; their original order being maintained by a "page map" which is utilized by the hardware. The programmer need not be concerned by either the segmentation nor the scattering.

The 6.01 TOPS10 monitor takes advantage of the KI10 hardware to do demand paging. In this scheme of memory management, it is not necessary for a complete program to be in core in order to be executed. Instead a "working set" of active pages are core resident with the remainder of the program residing on the virtual memory device (disk or drum). When a page is required which is not presently in the working set, an interrupt is generated, and the monitor transfers that page from the virtual memory device to core.

A DOCUMENT ON THE KI10
GENERAL DESCRIPTION OF A KI10

KI10 Hardware Features

Up to 4,096,000 36-bit words of memory.

User virtual address space of 256K 36-bit words.

Double precision floating point.

Four processor modes which provide protection in a multi-user environment.

Pipelining of data and instructions from memory. (Overlapped memory fetches.)

Different sets of general registers for user and executive programs.

SUPPORTING DATA OF A MORE TECHNICAL INTEREST

378 Instructions (366 on KA10).

Multi-level indirect addressing.

Immediate mode addressing (data is in the instruction).

Fixed-floating conversions.

Trap handling independent of the interrupt systems.

A DOCUMENT ON THE KI10
THE KI10 AS A PAGING MACHINE

General Concepts

Paging is not a new concept. It was first implemented on the ICL-ATLAS in the late 50's. The basic idea is simple. The user looks upon his program as a set of sequential addresses (no change here). The hardware, however, segments the program and places these segments anywhere in memory. A special hardware system translates between the linearity expected by the program and the actual physical location of the segments. This hardware is known by any of the following names:

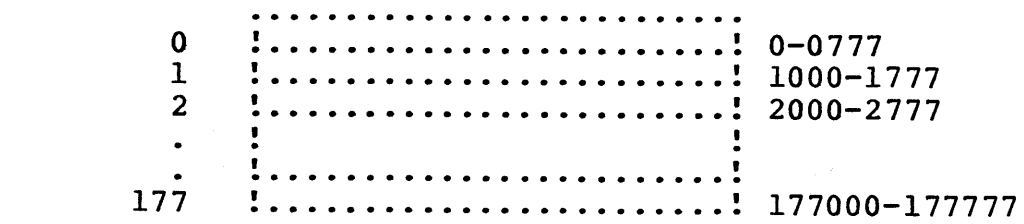
associative memory
paging box
relocation hardware
page table.

I will use the term "associative memory" throughout the remainder of this document.

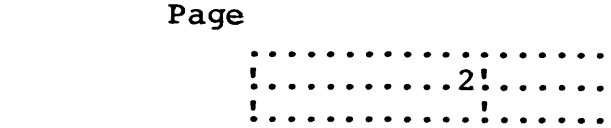
The Geometries of KI10 Memory

Physical core is looked upon as a sequence of 512 word pages (the segments referred to previously). Each page is addressed by its relative page number.

Picture a 64k system, the page numbers are then:



Thus to access word 15 in page 2 the address (2015) is viewed as:



When the hardware calculates the effective address it uses 13 bits for the page number giving a maximum of 8192 pages (4096k) and 9 bits for the increment (0-511).

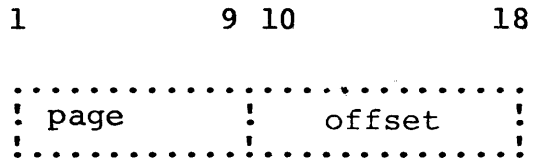
It is useful to think of addresses as being made up of these two parts (page, increment) since the hardware treats the parts separately.

User Address Space VS. Physical Address Space

The geometries explained above could have been for the KA10 if one merely changed a few numbers. However, there is little or no value in thinking of KA10 addresses as two part variables.

The user address space, known as user virtual address space, may be as large as 256k words (512 pages). This upper limit is enforced by the 18-bit address field of each instruction.

Again, it is useful to look upon the instruction's address field as two 9 bit fields:



The number of pages spanned by a program may be larger than physical core (given a properly smart operating system).

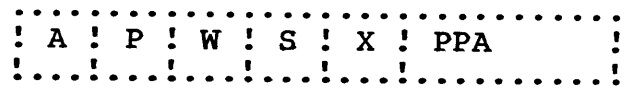
There is no natural relationship between the user page number and the physical page number. The relationship must be computed via a table lookup. This process is known as MAPPING.

A DOCUMENT ON THE KI10
THE KI10 AS A PAGING MACHINE

User Page Map

Associated with each user is a page known as the Process Table or "user page map page" (UPMP). The purpose of this page is to provide the information necessary to map the virtual addresses to physical addresses. This information is contained in a table of 512 entries, that is, one entry for each possible page in user's virtual address space.

Each entry contains the following information:



A	=0...unused entry (no page exists).
	=1...valid (used) entry.
P	=0...concealed page.
	=1...public page.
W	=0...write protected.
	=1...writeable.
S	.....undefined; for usage by monitor.
X	.....unused, reserved for hardware usage.
PPA	.....physical page address (13 bits)

Each entry occupies one half-word of storage. This means that over 200 locations in the UPMP are available for software.

Although the UPMP is associated with a user program, it is not part of that program's address space. The contents of this page is controlled by the monitor. The monitor also maintains another page map called the "exec page map page" (EPMP) which is used for mapping certain areas of the monitor address space.

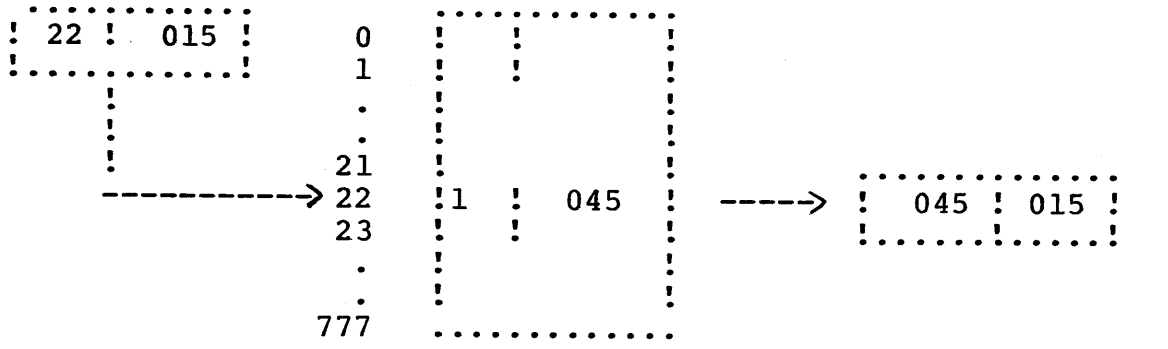
A DOCUMENT ON THE KI10
THE KI10 AS A PAGING MACHINE

Virtual to Physical: Mapping the Address Space

When an address is supplied to the processor from the user program it contains a user virtual page number. This must be translated into the proper physical page number. The operation of this is clear.

1. Access the proper entry in the user page table using the virtual page number as the index.

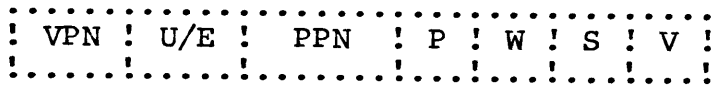
2. If a page exists, append the 13 bit page number found in this entry to the 9 bit increment. (This 22-bit result is the physical address).



With this scheme, each access to an addressed memory location actually takes two fetches. The necessity for two memory fetches is expensive and halves the effective speed of the memory. The addition of an "associative memory" eliminates many of these overhead memory fetches thus improving the overall efficiency of the machine.

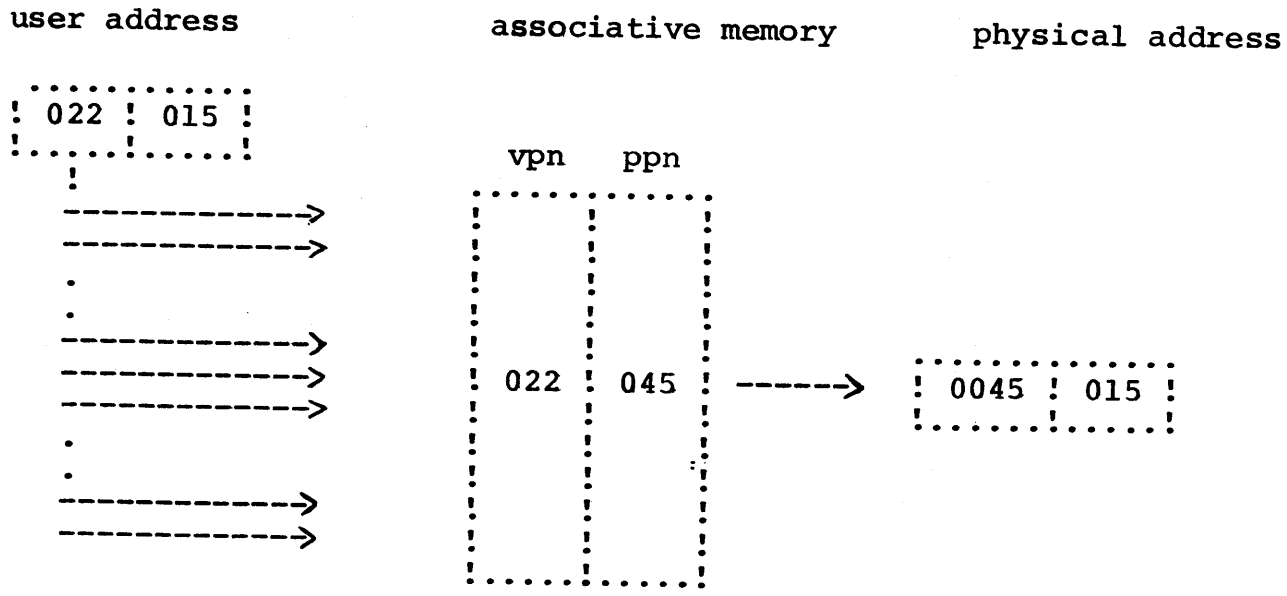
Associative Memory

The associative memory can be viewed as a table of 32 registers each of which contains the following information.



VPN	virtual page number (supplied by program).
U/E	=0...from executive page map page. =1...from user page map page.
PPN	physical page number (gotten from previous lookup into the page map page)
P	public or concealed.
W	writeable or non-writeable.
S	defined by the software
V	=0...this entry not in use. =1...this entry contains valid information.

When the CPU is provided with an 18-bit address, the first 9 bits are stripped off and used as a key to search the associative memory. This search takes place in all 32 entries simultaneously. If the VPN and the U/E bit match in any entry, then the PPN is automatically appended to the low order 9 bits and the 22-bit address is thus formed. The cycle time is in the range of 100 nanoseconds.



There are several complications which should be noted. Since there are two page maps active at any given time (user and exec), there is a possibility of two pages in the associative memory with the same virtual page number. There is, however, no ambiguity since the search includes the matching of the User/Exec mode bit (U/E). An interesting question is "what happens when we switch to another user?" This is a valid question since all users have the same virtual page numbers available, but each user's pages are in different physical locations. The associative memory, however, does not know about different users, only about virtual page numbers. Thus, the monitor must clear all entries in the associative memory whenever user page maps are changed.

If the user page number is not found in the associative memory, then the hardware does a lookup using the UPMP (or EPMP) for the physical page number. It then loads an entry of the associative memory and supplies the 22-bit address to the processor. This operation can cause a "page fault" for any of the following reasons:

1. Protection violation.

2. Page fault (no physical page referenced in the UPMP).

3. Other request failure types..(See later)

PROCESSOR MODES

On the KA10 there are two modes, executive and user. In executive mode, there is no relocation, and all of memory is in the range of your instructions. In user mode, there is two-segment relocation and protection. This is provided by dual protection-relocation registers. The KI10 does not have these registers; it performs the functions by another means.

The associative memory in the KI10 provides the vehicle for both relocation and protection. On a page level, portions of a job can be made "writeable" or "non-writeable", "public" or "concealed". Attempts to violate these conditions will result in a page fault which the monitor will handle.

A DOCUMENT ON THE KI10
PROCESSOR MODES

EXECUTIVE MODE

KERNEL MODE	SUPERVISOR MODE
HANDLE INTERUPTS	
PERFORMS I/O	GENERAL MANAGEMENT FUNCTIONS.
PERFORMS FUNCTIONS AFFECTING ALL USERS, (I.E., CONSTRUCTING PAGE MAPS)	PERFORMS FUNCTIONS AFFECTING A SINGLE USER.
CAN ADDRESS UP TO 112k DIRECTLY	EXECUTES IN EXECUTIVE VIRTUAL ADDRESS SPACE USING EXEC PAGE MAP

USER MODE

CONCEALED MODE	PUBLIC MODE
PROTECTION FOR PROPRIETARY PROGRAMS	NORMAL USER PROGRAMS
CAN READ, WRITE, EXECUTE, TRANSFER TO ANY LOCATION IN A PUBLIC PAGE	256k WORD ADDRESS SPACE
	PERMIT ALL INSTRUCTIONS UNLESS THEY COMPROMISE SYSTEM INTEGRITY.
	CAN ONLY TRANSFER TO CONCEALED MODE AT "PORTALS".

A DOCUMENT ON THE KI10
PROCESSOR MODES

Inter-Mode Protection and the mode of the processor

The processor is always operating in one of the 4 modes. Each mapped page is classified as a public page or as a concealed page. Depending on this classification and the mode of the processor the hardware enforces certain access restrictions.

Page classified as Public or Concealed...

Write protection...A page which is write protected cannot be written into from any mode (except kernel mode direct addressing). However, kernel mode programs can change the write protection of any page.

Page classified as Public...

Processor in Kernel mode...may read or write data as it pleases either thru the exec page map or via direct addressing (discussed later). If an instruction in such a page is executed, the processor changes to supervisor mode; returning to kernel only thru such defined ways as 1) issuing an MUUO, 2) an interrupt, 3) Entering a concealed page at a PORTAL instruction.

Processor in Supervisor mode...There are no restrictions since this is essentially the Public form of Exec mode.

Processor in Concealed mode...Like kernel mode, the processor can read or write data from public pages at will. When executing an instruction from such a page, the processor will change its mode to public.

Processor in public mode...No restrictions. This is the mode in which user programs will usually run.

Page classified as Concealed...

Processor in Kernel mode...There are no restrictions as the processor and the page have essentially the same classification.

Processor in Supervisor mode...Data may be read from concealed pages, but may not be written into concealed pages. Only PORTAL instructions may be accessed for execution from such pages; at which time the processor changes to kernel mode. Supervisor mode programs usually cannot access concealed pages referenced through the user page map. However, by setting the Disable-Bypass bit (bit 0 of the PC) this restriction can be overcome.

Processor in Concealed mode...No restrictions since the processor mode and the page mode are the same.

Processor in Public mode...No access to pages in this mode for data manipulation (either read or write). Instruction access only thru PORTAL areas. When such an instruction is accessed, the Processor changes to concealed mode.

KI10 INSTRUCTION SET DIFFERENCES FROM THE KA10

This section covers information on the KI10 instruction set which is difference from the KA10.

Note that the KI10 cannot set flags or traps when instructions are executed as interrupt instructions (this is done very rarely if ever in normal practice).

A DOCUMENT ON THE KI10
KI10 INSTRUCTION SET DIFFERENCES FROM THE KA10

Stack, Byte, and Shifting Instructions.

PUSH ...push down..When incrementing the pointer, the two halves are handled independently.

POP ...pop up..same restrictions as push.

byte instructions ..in the KA10, if the "Y" (address) field of the pointer overflows it invalidates the index portion of the pointer..This does not happen in the KI10.

shift instructions ..KI10 eliminates redundant movement by shifting by MOD (E,72) rather than a max of 255 as in the KA10.

Conversion Instructions

FIX fix..float to fixed, round down.

FIXR fix and round..float to fixed and round up if fractional part .5.

FLTR float and round..fixed to float

Double Precision Instructions

..note that these instructions give 62 bits of precision, whereas only 54 bits were possible in the KA10.

..all instructions use adjacent pairs of accumulators and adjacent memory locations. AC 17 wraps around to AC 0. Memory locations wrap around from 777777 to 0.

..high order bit of second word is ignored.

..results are always normalized in addition and subtraction, not guaranteed in multiplication and division if inputs were non-normalized.

A DOCUMENT ON THE KI10
KI10 INSTRUCTION SET DIFFERENCES FROM THE KA10

DFAD double floating add.
DFSB double floating subtract.
DFMP double floating multiply.
DFDV double floating divide.

Double Move Instructions

DMOVE double move to AC's

DMOVEM double move to memory...results not guaranteed if moving AC,AC+1 to AC+1,AC+2

DMOVN double move to AC negate..if operand in software (KA10) double precision, use DFN instead.

DMOVNM double move to memory negate..same caution as DMOVEM.

Arithmetic Testing

AOBJP add one to both halves of AC and jump if positive. ..unlike the KA10, the two halves are treated independently.

AOBJN add one to both halves of AC and jump if negative. ..same caution as AOBJP.

program control ..several of the jump instructions save the contents of the PC and flags.

JSR jump to subroutine..executed as an interrupt instruction, this forces the processor to executive-kernel mode.

JSP jump and save PC..same into as JSR.

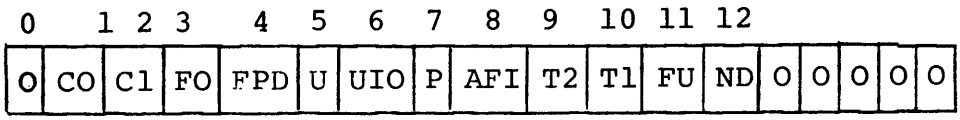
JRST 1, PORTAL ..valid entry point from a public to a concealed area.

TRNA fastest skip instruction in the KI10

PUSHJ push down and jump..trap 2 set on pushdown overflow.

POPJ pop up and jump..trap 2 set on overflow.. trap 2 cannot be set if used as an interrupt instruction.

The PC flags saved by these instructions are illustrated on the next page.



O overflow (fixed arithmetic, ASH, ASHC, FIX, FIXR, floating arithmetic, no divide)

..in KI10 executive mode bit zero represents the disable bypass flag which is related to the executive XCT covered later.

CO, C1 carry flags.

FO floating overflow ..can be set by KI10 instructions (FLTR,DMOVN,DMOVNM,DFDV).

FPD first part done ..in KI10 has implications that a page failure occurred while processing a byte or second word of a DMOVEM, or in a noninterrupt data IO instruction that results from a block IO instruction, also has same implications as KA10.

U user ..processor in user mode.

UIO user in-out ..normal KA10 implications if user mode set, has executive XCT implications (covered later) if executive mode.

P public ..last instruction was fetched from a public area, (user-public, or executive-supervisor).

AFI address failure inhibit ..an address failure cannot occur during next instruction (see later discussion).

T2 trap 2 ..arithmetic overflow, if traps are enabled (covered later) this will take one.

T1 trap 1 ..pushdown overflow, if traps are enabled (covered later) this will take one immediately.

FU floating underflow

ND no divide.

UUO — Unimplemented user operation.

LUUO — local uuo..if executed in executive mode uses location 40-41 in the executive process table; in user mode-location 40-41 in user's virtual pg. Ø.

MUUO — monitor uuo..behaves as follows..

store instruction code, AC, and effective address in location 424 of user process table.

save flags and PC in location 425.

set up flags and PC from a third location and start at specified point.

third location depends on mode of processor at time of muuo, and whether or not it was given as a result of a trap.

mode	execution	location
kernel	no trap	430
kernel	trap	431
supervisor	no trap	432
supervisor	trap	433
concealed	no trap	434
concealed	trap	435
public	no trap	436
public	trap	437

Note that the new PC can set up the flags any way it pleases, thus it may cause the mode to change.

A DOCUMENT ON THE KI10
KI10 INSTRUCTION SET DIFFERENCES FROM THE KA10

Input-Output Instructions

..the KI10 allows users to execute IO instructions to any device whose device code is greater than 740.

DETAILED VIEW OF KI10

KI10 Operation

Page Maps

There are ALWAYS two page maps in use. One is the Executive Page Map. The other is the User Page Map. The location of these maps is under the control of the monitor. In the instruction:

DATAO PAG,E

The effective address E is a word which contains:

1. User base address. This is the physical address of the UPMP.

2. Exec base address. This is the physical address of the EPMP. In 506 this is always zero.

3. AC block. Accumulator block used in user mode (exec mode always uses block 0).

4. Small user bit. If a user is designated as "small user", then he is restricted by hardware checks to keeping his virtual address space within pages 0-15 and 256-271. This gives a total user space of 32K.

5. User address compare. Used in conjunction with the console paging switches for address break control.

6. Overflow trap enable.

There are also two bits which allows one to change the user information only, the executive information only, or both. Whenever this instruction is used, the associative memory is "cleared".

Executive Mode Address Space

Executive address space is partitioned into 4 areas. Each area is specified by the manner in which the hardware makes access to it. Thus the areas are:

1. 0-17 the AC's . . . . use Block 0.
2. 20-337777 the first 112K. . . . is "unmapped" and is thusly accessed directly (a la the KA10).
3. 340000-377777 16K mapped location by the UPMP.
4. 400000-777777 128K mapped by the EPMP.

The executive page map only contains mapping information on pages 400-777 (octal). Thus a reference such as ADD 2,270022 is performed directly without use of the paging hardware. Should the address not exist in physical memory, a normal KA10 type NXM (no access to memory) interrupt will occur. Using the above information, the following examples illustrate the method used for address calculation.

```
ADD 2,17        From the accumulators.
ADD 2,20        Direct addressing (no paging).
ADD 2,337777    Direct addressing (no paging).
ADD 2,340000    Thru UPMP (location 400).
ADD 2,377777    Thru UPMP (location 401).
ADD 2,400000    Thru EPMP (location 200).
ADD 2,777777    Thru EPMP (location 377).
ADD 2,0                 From AC 0.
```

Thus with the processor in exec mode, the hardware will use the AC's, direct memory addressing, the user's page map, or the exec page map depending on which of the 4 areas the address falls into.

Accumulator blocks

There are 4 blocks of accumulators, numbered 0-3. Block 0 is always used during exec mode processing. Normally users will get block 1, however realtime jobs will be allowed to use blocks 2 and 3 (in later monitors). The purpose of multiple AC blocks is to eliminate the storing and restoring of accumulators as programs oscilate between user and exec mode processing (MUUO's). This does not eliminate the need for saving the accumulators during context switching.

Extended Functions Demanded By The KI10 Architecture

With the architecture providing for an executive address space separate from the user address space, there are two necessary functions which the KI10 instruction set given so far doesn't handle.

1. The monitor, executing in the EXEC mode uses AC block 0. However, it is often necessary to be able to access the user's AC blocks (1-3) from this state.

2. While processing a UUO, the monitor must often access arguments in the user's core area thru the user page map.

Also, there are two functions which, while not necessary, are sufficient to provide for a great deal of transparency in the handling of UUO arguments.

1. UUO arguments which are in registers may be in the specified AC block or in the user's shadow area. It would be helpful if this distinction were transparent to the processing routine.

2. If the monitor itself were to use the UUO mechanism by going thru the procedure of:
   a) store the arguments in the AC's,
   b) save the AC's,
   c) issue the UUO,

   then it would be helpful if the correct set of saved AC's could be accessed transparently by the processing routine.

All of these functions, both necessary and desired, are provided for by the special modes of the XCT instruction.

The Executive Mode XCT Instruction

The preceding section was merely to set the atmosphere for this relatively difficult area. It presented several items necessary and/or desireable; all of which are accomplished via the XCT instruction.

To re-capitulate, the exec XCT will allow the monitor to access data in user AC Blocks and user core. It will also allow the monitor routines to be coded independent of the need to know the exact location of the AC's (AC block or shadow area) and independent of who issued the UUO (user or monitor).

The XCT instruction is written thusly:

```
XCT     AC1, ADDRESS (AC2)
```

In the KA10, and during User-mode processing in the KI10, the field "AC1" is ignored. During Exec-mode processing in the KI10, the bits in this field are used as flags to control the location of data during read and write operations.

Control Flags:

Bit 11=0 .....Write in the monitor address space.
=1.....Write in the caller's address space.

Bit 12=0 .....Read from the monitor address space.
=1.....Read from the caller's address space. (note that this effects instructions which are read-modify-write [ɔs]).

USER I/O Bit in PC
=0.....The monitor issued the UUO, the caller space is thus the same as the monitor address space except that the routine which issued the call saved the AC's.

=1.....The UUO was issued from User-mode. The caller's core space will be accessed via the User Page Map.

CURRENT USER AC BLOCK
If the current user AC block (as defined by the argument to the DATAO PAG,E instruction) is non-zero, then the callers AC's are in this block, otherwise, they are in the user's shadow area.

It is important to notice that the USER I/O bit in the PC and the Current User AC Block are set before the processing routines are entered, and thus are "global" control items which allow the correct core and AC areas to be accessed automatically. The read/write bits are "frozen" in the code of the processing routines and thus are not controlled dynamically by the caller or the hardware.

Consider the following examples:

1. The User Issues a UUO.....
The hardware automatically picks up a new PC from location 434 or 436 of the UPMP. This PC will have the USER I/O bit set to 1 and will start execution of the UUO processing routines.

.....no flags set.....
XCT 0, (MOVE AC1,OPERAND)
XCT 0, (MOVEM AC1,OPERAND)

Both instructions behave as a normal XCT since the read and

write flags are both zero (indicating read/write from monitor space).

.....Read flag set.....(effective only if Eff. Adr is read or read-modified-written from)
XCT 1, (MOVE AC1,OPERAND)

The address OPERAND will be in the user's address space. If it accesses an AC, then the data will be read from the shadow area or the designated AC Block (see Current user AC Block above.).

XCT 1, (MOVEM AC1,OPERAND)

The data in the monitor's AC, AC1, will be stored in the address OPERAND which is in the monitor address space.

Note: This operates no differently from normal XCT since MOVEM does not perform a read from the effective adr calculated.

.....write flag set....(effective only if instr. writes into E)

XCT 2, (MOVE AC1,OPERAND)

The data in the monitor location OPERAND will be stored in the monitor's AC1.

XCT 2, (MOVEM AC1,OPERAND)

The data in monitor AC1 will be stored in the address in the user space referenced by OPERAND. If that is an AC then it will be appropriately the shadow or designated block, or if core, it will be accessed thru the UPMP.

.....both read and write flags set.....(has meaning for BLT & PUSH instrs).

XCT 3,(MOVE AC1,OPERAND)
XCT 3,(MOVEM AC1, OPERAND)

These instructions still operate as illustrated above, since the read and write bits only effect the effective address.

XCT 3,(BLT AC1,DEST+N)
where: AC = source,,dest

After address resolutions the operands, source & dest, will be both read and written into user address space.

2. The Monitor Issues a UUO

Prior to issuing a UUO, the monitor must save the AC's. These can be saved in 16 contiguous locations in the UPMP. The exact location in this page is maintained by the "executive AC Stack Pointer", a hardware register set by the CONO PAG, instruction. The contents of this register can be changed and thus UUO's can be issued while processing other UUO's* with the changed register pointing to new save areas.

When the UUO is issued, the new PC is gotten from location 430 or 432 of the UPMP and will have the USER I/O bit set to 0.

Thus:

XCT 1, (MOVE AC1,AC15)

Will read AC15 from the save AC area which is marked by the "Executive AC Stack Pointer" register and will write the data into the monitor's AC1.

If effective adr is 17, then XCT acts as a normal XCT.

*NOTE: The current Monitor has no nested UUO calls. It simply branches to the necessary code.

3. The Automatic Hardware Functions

Given the instruction:

XCT2,(MOVEM AC1,OPERAND AC2)

THE FOLLOWING OCCURS:
..The addresses are computed. Thus, the value in AC2 (Block 0 AC's) is added to the address operand to determine the effective address.
..the XCT instruction is executed.

.....User UUO...Current AC Block =1 (2 or 3), and the effective address is greater than 17. The value in AC1 of Block 0 is moved to the effective address using the User Page Map.

.....User UUO...Current AC Block =1 (2 or 3), and the effective address is less than 20.

The value in AC1 of Block 0 is moved to the effective address which is an AC in Block 1.

.....User UUO...Current Block =0.

The value in AC1 is gotten from the monitor's Block 0. This value is stored in the manner illustrated above. If the effective address is less than 20, it is stored into the Shadow AC's.

.....Monitor UUO...

The value of AC1 is gotten from Monitor's Block 0 and is stored into the effective address. If that address is 0-17, then it is stored into the "stacked" AC's area in the UPMP.

One more flag (this makes 5 flags to understand), the DISABLE-BYPASS which is bit 0 of the PC gives a supervisor mode program extra privileges. When =0, the supervisor mode routine is allowed access to concealed user pages, when =1, such access is prohibited.

It should be re-emphasized that the instruction executed by the XCT is from the exec address space. Also the address calculations are done in exec address space. Thus in the monitor instruction:

```
XCT 1, (SKIPL @T2)
```

The affected address is calculated using the value found in T2 from the monitor AC's. If this calculation leads to another indirect address, that is also taken from monitor space, and so on until an address is finally resolved.

NOTE: This hardware capability for saving Exec ACS in the UPMP is not currently used in the DEC-10 Monitor.

Page Failure

When a page failure occurs, the following information is saved:

1. User or Exec mode flag.

2. The number of the virtual page in question.

3. A set of flags indicating the reason for the page failure.

   Hard Failures:

   address failure -- Page failure forced by the satisfaction of an address condition set on the console. It indicates that the ADDRESS BREAK switch was on and the ADDRESS INHIBIT BIT (in the PC) was clear.

   page refill failure -- This is a hardware failure.

   small user violation -- Violates the small user policy explained earlier.

   proprietary violation -- A public mode program has attempted to reference a concealed page or has attempted to transfer control to a non-portal instruction in such a page.

   Soft Failures:

   Soft failures merely indicated that the instruction was a read or write and reveal the access, write protection and software bits from the users page table. The monitor must then determine the significance of this information.

When the page failure occurs the above information is placed in the UPMP and control branches in kernel mode to the location specified by 420 in the faulters page table (could be UPMP or EPMP). If the fault occured during exec mode the failure information is stored in 427; in user mode 426.

If the page had an entry in the associative memory when the fault occurred, this entry is cleared (eg ., it could be a write-protection fault).

A probable use of the Software bit in the paging information is to control unnecessary swapping. Any time a page is swapped in, it could be write protected. The first write reference will cause a fault. The monitor will then set the software bit. Only those pages with this bit set need be swapped out. (Not implemented in 6.01).

Priority Interrupt System and Traps

The priority interrupt system in the KI10 can function in the same manner as the KA10, but can also take advantage of "smart" devices. In the latter case the following takes place.

1. An interrupt on channel N is received.

2. The KI10 sends an "interrupt-granted" signal for this channel.

3. The first device on the bus transfers the signal to the second, the second to the third, and so on; until (at last) the device which requested the interrupt receives it. The devices terminates the signal path and sends an interrupt function word back to the processor.

   The function word contains a function code, an interrupt address, and an increment (perhaps a device number relative to a certain controller).

   The function code may specify that the interrupt be handled like the KA10. It may specify that the processor dispatch to the interrupt address or to the address formed by adding the value found at the interrupt address to the increment (in effect, a dispatch table). Or it may specify that a DATAI/DATAO be performed using the interrupt address as the arguement.

KA10 devices work with no modification on the KI10 since no response to the interrupt-granted signal is used as a flag for standard processing.

The first 2 instructions executed in response to an interrupt are referred to as being "executed as interrupt instructions". Some instructions, when so executed, have different effects than when executed normally. In general, these effects are: flags not being set during arithmetic instructions, overflows in PUSH and POP instructions being ignored, and page faults merely being held until the interrupt is dismissed.

The only reasonable interrupt instructions outside of BLKI/BLKO are JSR, JSP, PUSHJ, MUUO; each of which hold onto the interrupt and begin operation with the interrupt PC. Some others, namely AOSX, SKIPX, SOSX, CONSX operate similarly to the BLKX and thus have uses in special areas. All other instructions, in general, cause the processor to dismiss the interrupt.

Traps

Overflow conditions on the KI10 are handled by a trapping mechanism somewhat similar in behavior to what happens when a MUUO is issued. Only in this case, the next instruction (not a new PC word) comes from Loc. 421 of the UPMP/EPMP depending on the mode. (Loc. 422 is similarly handled for Push Down Overflows.)

When control comes to Loc. 421, the instruction is executed in the current hardware mode and control passes back to the program causing the trap. (The mode is unchanged.) The only mechanism for allowing the monitor to gain control when this condition occurs is to place a MUUO in Loc. 421.

The trap vector MUUO is handled similarly to a normal MUUO. That is, the new PC word is picked up from the UPMP. Only in this case, depending on the current mode of the machine, it will come from slots 431, 433, 435, or 437. (Not 430, 432, 434, or 436).

KI10 Processor Conditions

The KI10 can have two priority interrupt channels assigned to itself. One is for errors and the other is for the clock. The most interesting condition trapped here is the automatic restart after power failure. This feature is restricted to those times when the voltage on the power mains drops below specification with the CPU running; and is restored with the CPU still in the run state. Thus automatic restart can never take place as a function of the on/off switch.

This feature needs the help of the monitor. The monitor must enable the feature and be able to respond to it in 4 millisecs. When power is restored, the CPU will automatically branch to location 70 in kernel mode.

In a nutshell, if the monitor can save the state of the processor within 4 msec after a power failure is detected, then it can restart with honor (assuming of course that users can live with scrambled up disk transfers).

Memory Control

Associative memory control

Digging deeper in the associative memory we find a few more unmentioned tidbits.

Refill Counter--The refill counter points at one of the 32 associative memory entries. As its name implies, it controls the point at which the next entry into this memory will be made. Its location can be determined by a CONI PAG,E. Its location can be set by a CONO PAG,E.

When left alone, the counter is updated automatically and it behaves in a most rational manner; after the entry to which it points is filled, the counter is advanced (MOD 32), and if a mapping is made using the entry to which it now points, the counter is again advanced. This characteristic tends to push it away from active entries. Thus it should point at the most inactive entry.

Besides the CONO/CONI, datao/datai, one may issue a MAP instruction to the PAG device. With this, one can determine if a given virtual page is in the associative memory. If it is, then the corresponding physical page and control flags are made available.

Overlap, pipelining, and prefetch in core memory

A portion of the KI10 is a memory control unit which sits functionally between the memory and the processor. The processor merely makes data requests, and the memory control does all the necessary interfacing with the associative memory and various core memories to supply the data.

Overlapping

Overlapping refers to the ability to overlap two read cycles in different memories. The memories must be "timed" as are ME10, MF10, as oppossed to "untimed" (asynchronous) as are MA,MB,MD. They may, however be of different speeds. With interleved operation, the fetch of a two-word operand is greatly speeded up by overlapping. This feature also is made use of by the instruction prefetch feature. Note that the proper buffering or delaying is done in memory control so that the processor is not concerned with whether overlap took place or not.

Pipelining

Memory control consists of 6 memory subroutines:

Subroutine call~page delay (the control)
Page check, memory go, recycle
Request cycle ~read restart
Read return and refill entry
Write restart
AC references

These subroutines are operationally independent and thus the operations may be pipelined. E.g., page checking is done with the address on the address bus while the MA register contains a different address to be sent on the memory bus. This pipelining is most useful during double word fetches and prefetching. It is not dependent on overlapping being possible.

Prefetch

The prefetch feature is simply a request from the processor to fetch the next instruction before the present one is completed. This can be done if the present one leaves the results in the accumulators and does not change the sequence of program execution (PC is incremented by 1). Instruction processing can be broken into 4 steps:

Instruction fetch and decoding
Operand fetch
Execution
Result storage

In a two instruction sequence like:

```
AA:     ADD     4,FOO
BB:     SUB     4,OOF
```

Instruction AA does not change the PC sequence and leaves its results in AC 4. Thus, as soon as a request is made for the value at FOO, the processor may request the next instruction BB. The memory control will insure that the operand gets to the processor first.

It is this feature which makes the ability to do pipelining and memory overlap worthwhile since the percentage of double word fetching instructions is quite low.

Different Memories

The KI10 divides memories into 3 classes: fast, slow, and immediate. Fast and slow memories are those which have the necessary logic to overlap. Memory control takes into account the speed differences when doing overlap to a fast and a slow memory. Immediate memories do not have this logic and thus are unable to do overlap. Each box of memory is tied to one of 3 request busses depending on its class and it is this that allows memory control to descriminate among the 3 types of memories.

Some Efficiencies in Programming

Even if the effective address is in the AC's, a certain amount of running around in the memory control is done for all such computed (operand portion of instruction) addresses. Thus where it is possible, the instruction MOVEI T1,(T2) should be used to transfer data from register T2 to register T1. This particular example is more efficient than either MOVE T1,T2 or MOVEM T2,T1. Also it is often more efficient to BLT data into the AC's and then execute a loop from core than to put the loop in the AC's. This is contrary to the KA10 where the latter would always win.

MONITOR USAGE OF THE KI10

Memory Referencing Capabilities

The monitor can no longer reference all possible addresses in physical core. Thus at any given point a key question is "what is addressable now?" The primary answer is that the first 112K is directly addressable, and that other pages are available through the user's page map table, UPMP, or through the exec page map table EPMP. When the monitor has the necessity to reference specific pages in the current users program (e.g., during MUUO's), it does so through reserved slots in the UPMP. When the monitor must reference pages in a non-current user's area (during I/O), slots must have been set up within the EPMP.

The slots used for mapping a non-current user are called the "exec virtual memory (EVM)". EVM consists of the last 128 pages (128-255) of the executive address space. Because of the finite number of slots available, EVM is contended for as a shared resource by user programs.

Using EVM

The major tasks performed for the non-current user are I-O and swapping. For I-O on devices connected through a DF10 there is no need for EVM. The monitor sets up channel command lists in the monitor free area formed by a string of 4-word blocks. If a user's buffer is split across two or more non-contiguous pages, the monitor builds one command list entry for each physical segment. Devices which have monitor buffers like TTY's and PTY's do not require EVM either.

For devices connected to the I-O bus enough slots in EVM must be allocated to map the largest buffer. In the case of buffered I-O, this size will not change and subsequent requests will utilize the same slots with up-dated information in them. In the case of dump mode I-O, enough slots are allocated to cover the largest area specified by a single IOWD.

The monitor keeps track of which slots are allocated to which devices through the DEVEVM field in the DDB.

There are certain constraints placed on a job with the EVM resource. The job may not be swapped out since the location of his physical pages will change when he is swapped in. Thus EVM must be released before a job sleeps (including during DECTAPE dead reckoning).

Since the monitor has certain overhead tasks when working with users programs, certain slots in the EPMP are reserved for special purposes.

Page 400...For creating or moving a UPMP.

Page 401...Swapping checksum--to insure that the first page of a segment is addressable while the segment is being swapped.

Pages 402-410...One page per PI level in case there is a need to reference a page, e.g., memory parity routine needs a slot when sweeping core.

Page 411... Contains the SKPCUP macro for dual CPU systems. This offers code independence since this can reflect a different physical address in each of the CPU's page maps.

Pages 412-431...Contains the pages of where PAGETAB resides in core.

Pages 432-451...Contains the pages of where MEMTAB resides in core.

As the operating system acquires capability, additional slots for the EPMP will be reserved (cutting down on the available EVM).

Exec Slots in the UPMP

During MUUO's the monitor may have to reference the user's core area. To simplify this task, the 32 pages (340-377) in the Exec Address Space automatically get their physical page locations from the UPMP. The following information is stored into these pages when a user is swapped.

Page 340...UPMP address.

Page 341...Physical page number of JOBDAT area.

Page 342...Physical page number of high segment vestigal data area.

Page 343...Temporary slot for random monitor requests.

Pages 344-346...Physical pages in some other job to satisfy the requirements of the JOBPEK MUUO.

In the JOBPEK MUUO, when job-A needs information from job-B, the monitor copies the necessary entries from job-B's UPMP into slots 344-346 of job-A's UPMP, and then does a BLT using exec address 344000 plus the appropriate offset.

Managing the Physical Pages

A few new tables have been defined, and a few old ones redefined in order to keep track of the physical pages and their users.

PAGTAB

A table with one entry per physical page. Each entry is linked to one of several chains. One such chain is the Free-Page list which is pointed at by PAGPTR. Otherwise there is one chain per job.

JBTUPM

One entry per job. This table contains the pointer to PAGTAB such that it references the UPMP.

JBTADR

These entries now contain the value 341000 for each job which is swapped in.

During the processing of I-O requests and other things which require the monitor to access pages of the non-current job, the following is done. Read and save the pointer to the present UPMP. Set the hardware pointer to the UPMP to be that of the desired job. Do whatever juggling has to be done since this job is now the current job. Restore the pointer in the hardware to the former job. Clear the associative memory. This last step is very important since the search key to this memory is the virtual address which is ambiguous where more than one user is concerned.

KI10 CONSOLE NOTES
by Dave Gross

LIGHTS AND SWITCHES

Real address of a parity error on a parity stop is found in the MA lights (bay 1, lower right).

Address of a HALT is found in the AR lights (bay 2, second row). ECO KI10-00020/PS521 fixes a bug in this feature.

Here is a quick handwave on the lights and switches. The indicator panels at the top of the machine show internal registers and time states of the machine. Registers and time states are named the same as the corresponding components of the KA10. Of interest to programmers are the MA lights (bay 1, lower right), the PI GEN lights (bay 3, lower left), and the switch-selectable lights (bay 3, top left). A rotary switch (maintenance panel, third from right) controls the function of the selectable lights (MB=memory data, IOB= transitory data on the I/O Bus, P-R=paper tape reader buffer, UEBR=data to page box (associative memory)).

In general, push buttons light when pushed. Some toggle, some toggle machanically. DATA LOCK (on the maintenance panel) locks the data switches. CONSOLE LOCK locks everything else except the mechanical switches.

CONSOLE FUNCTIONS

The console functions much like the KA10 console. But there are significant differences. Some are:

1. Addressing via the Address Switches

   There are 2 switches to the left of the address switches labelled EXEC PAGING and USER PAGING. If both are off, the address switches refer to the 22-bit physical address. Moreover, AC references via EXAMINE or DEPOSIT refer to the AC block number contained in the switches in the top row of the maintenance panel. If EXEC/USER PAGING PAGING=10 or 11, EXAMINE and DEPOSIT requests (18 bits only) are mapped via the Exec Page Map. If EXEC/USER PAGE=01, the User Page Map is used.

2. Address Following:

   If EXEC/USER PAGING =
   - 01 No address following
   - 10 The MI follows exec memory references
   - 11 The MI follows exec references if the USER ADR COMP bit in the pager is on
   - 01 The MI follows user references if the USER ADR COMP bit pager is on

3. Address Stop or Break

   Same dependency on EXEC/USER PAGING and USER ADR COMP as address following. Address break is actually a form of page failure. There is a DATAO to load the address switches, EXEC and USER PAGING, ADDRESS BREAK, and the 3 associated conditions.

4. Page Map violations caused by the console light the KEY PF light (top center of the console) and do NOT cause a page failure in the program.

5. There is no single-memory-cycle mode. Instead, there is a single-clock-pulse mode. The mode is entered by the SINGLE PULSE switch in the maintenance panel (third row down). Clock pulses are triggered by the SINGLE PULSER switch (lower left of the console). The pulser switch lights when the clock is ready to be triggered.

6. If any switch is on which should be off for normal operation, the MAINT MODE indicator (top center of the console) lights up.

7. All action switches are effective while the machine is running. In particular, READIN, START (starts in kernal mode), EXAMINE NEXT, and DEPOSIT NEXT work in the KI10 while the machine is running but not in the KA10. If this worries you, use console lock.

8. If an action switch is being repeated via the REPEAT function, the switch remains lit until REPEAT is turned off and the function occurs one more time. Meanwhile, all other action switches except RESET and the memory-continue function of CONT are disabled. STOP, CONT, and READ IN can be repeated. These functions retrigger whenever the processory executes a halt instruction. But note that the STOP switch will be disabled if the program fails to ever execute a HALT (use RESET if necessary).