U S A B L E X M L ' s

```
BLWP @XMLLNK      OR       LI  R0,>YYXX+>0100
DATA >XXYY                 XOP  6,6
```

EXAMPLE: CONVERT FLOATING TO INTEGER (CFI)

```
       EITHER BLWP @XMLLNK
              DATA >1200    (IN BASIC ENVIRONMENT)
       OR
              LI   R0,>0112
              XOP  6,6      (IN ANY ENVIRONMENT)
```

NOTE: "f.p." means "floating point"

>0100	ROUND$	Round f.p. number in FAC via guard digits in FAC+8
>0200	ROUNU$	Round f.p. number in FAC to position specified in FAC+10.
>0300	FSTAT	Sets status byte >837C according to f.p. number in FAC.
>0400	OVUN	
>0500	OVFL	Put overflow value (9.999999999999E+128) in FAC
>0600	FADD	f.p. FAC <-- ARG + FAC
>0700	FSUB	f.p. FAC <-- ARG - FAC
>0800	FMUL	f.p. FAC <-- ARG * FAC
>0900	FDIV	f.p. FAC <-- ARG / FAC
>0A00	FCOMP	Set status byte (>837C) based on the comparison ARG :: FAC

operations >0B00 to >0F00 use >836E as a pointer into an area of VDP RAM that is being used as a stack. The stack grows toward high memory, and >836E points to the top element. Push is pre-increment, pop is post-decrement.
NOTE: This is NOT the stack used by the Basic interpreter, and >836E should not be used in this way while in the Basic environment.

>0B00	VSADD	Pop f.p. number to ARG, then FADD
>0C00	VSSUB	Pop f.p. number to ARG, then FSUB
>0D00	VSMUL	Pop f.p. number to ARG, then FMUL
>0E00	VSDIV	Pop f.p. number to ARG, then FDIV
>0F00	VSCOMP	Pop f.p. number to ARG, then FCOMP
>1000	CSNV	Convert a string in VDP RAM to a f.p. number

in FAC. FAC+12 is a pointer to the start of the string on input, and to the first unconverted character on output. The normal convention is to terminate the string with an ascii null (>00) character.

>1100	CSNC	Similar to CSNV, except string is in CPU RAM.
>1200	CFI	Convert f.p. number in FAC to a 16 bit 2's complement integer in FAC. If an error occurs, the byte at FAC+10 will be set to a nonzero value.
>1700	VPUSH	Push the 8 bytes from FAC onto the VDP stack, using >836E as stack pointer.
>1800	VPOP	Pop 8 bytes into FAC from VDP stack, using >836E as a stack pointer.
>1001	GRINT	Apply greatest integer function to f.p. number in FAC.
>1101	PWR	f.p. FAC <-- ARG ^ FAC
>1201	SQR	f.p. FAC <-- SQR(FAC)
>1301	EXP	f.p. FAC <-- EXP(FAC)
>1401	LOG	f.p. FAC <-- LOG(FAC) (natural log)
>1501	COS	f.p. FAC <-- COS(FAC)
>1601	SIN	f.p. FAC <-- SIN(FAC)
>1701	TAN	f.p. FAC <-- TAN(FAC)
>1801	ATN	f.p. FAC <-- ATN(FAC)
>1901	CNS	Convert number in FAC to string according to INPUT: FAC+11 = 0 for free format (other inputs are ignored.) > 0 for width, excluding decimal point. FAC+12 = 0 for underflow to 0, overflow to EEEEEEEE. > 0 for E-format on over/underflow. FAC+13 >= 0 number of digits to right of decimal. <0 disables fixed mode. OUTPUT: FAC+12 (byte) is length FAC+13 (byte) is the least significant byte of a pointer to the answer. The most significant byte s always >83.
>1A01	CIF	Convert 16 bit 2's complement integer in FAC to an 8-byte f.p. numbe in FAC.
>4001	MULTI	Set up VDP Patern Name Table for multicolor mode. (0-31 four times, then 32-63 four times, etc.)

```
>4101    HIRES      Set up VDP Patern Name Table for high resolution mode.
                    (0-255 three times.)

>4201    DRAW       Draw a line in high resolution mode.
                    FAC : (byte) graphics mode (2,3,or 4.  One less
                          than Basic's CALL GRAPHICS mode.)
                    FAC+1 : (byte) color (fore/back)
                          each 0-15. (One less than Basic's
                          CALL COLOR values.)
                    FAC+2 : (byte) draw mode (-1,0 or 1)
                    FAC+4 : (word) Y1 (zero-based)
                    FAC+6 : (word) X1 (zero-based)
                    FAC+8 : (word) Y2 (zero-based)
                    FAC+10 : (word) X2 (zero-based)

>4301    FILL       Fill screen area in high resolution mode.
                    FAC : (byte) graphics mode (2,3,or 4.  One less
                          than Basic's CALL GRAPHICS mode.)
                    FAC+1 : (byte) color (fore/back)
                          each 0-15. (One less than Basic's
                          CALL COLOR values.)
                    FAC+4 : (word) Y  (zero-based)
                    FAC+6 : (word) X  (zero-based)
                    FAC+10 : (word) CPU stack pointer
                    FAC+12 : (word) CPU stack limit
                          The stack area is a scratch area used
                          by the fill routine.  A 2K byte
                          area is recommended.


                    X O P ' s

I.  screen handler xops.

    YPT        >837E   Y-pointer within current window
    XPT        >837F   X-pointer within current window
    ( the margins and screen width define the current window )
    LINLEN     >8478   table of line lengths
    ( One byte per line.  Smart scroll routine uses this
      table to scroll only nonblank parts of line.
      The clear-screen and display-character routines modify
      this table.)
    SCLEFT     >8490   Number of columns to skip at left of screen
    SCRRM      >8491   Width in characters of window
    SCRWID     >8492   Screen width (40 in text mode, otherwise 32)
    SCRTOP     >8493   Number of rows to skip at top of screen
    SCRBOT     >8494   Height of window, in rows

          XOP  0,6  scroll window
          XOP  1,6  clear window
          XOP  2,6  get character to R0 msb, based on XPT, YPT
          XOP  3,6  display character from R0 msb, based on
                    XPT, YPT
          XOP  4,6  forwardspace
          XOP  5,6  backspace

II.   link to XML routines
     R0 MSB = XML type (1,2 OR 3)
```

R0 LSB = table/entry selector

```
XOP  6,6
```

III. I/O xops.

R0=pointer to name length byte in PAB
R1=search type. (8 for normal DSR I/O.)

```
XOP  7,6     link to DSR, with PAB and buffer in VDP ram.
XOP  8,6     link to DSR, with PAB and buffer in CPU ram.
```

Notes: DSR's assume that map file 0 at >8000 is the current active map, and that map file 1 at >8040 is available for scratch usage. All DSR's must be called in 4A mamory map mode. Interrupts must be off. For CPU I/O, the PAB and buffer may not be in the last two pages of memory (>E000 to >FFFF) since these are used by the DSR's for scratch work. The >4xxx and >5xxx pages must be mapped to >FF4000 and >FF5000.

IV. Memory management xops

These XOPs use a workspace at >8410. This workspace must be maintained across calls, as global memory allocation is kept there.

>841C to 1F: "fence" below which the memory allocation routine will not go. Initially a low value, to allow DSR's to allocate as much memory as desired on power-up. An application program will usually set this to memory top to freeze the DSR allocation.

>8420 to 23: Memory top. This is the first byte not available to the application. At power up this is set to physical memory top, and then reduced as DSR's allocate blocks of memory.

XOP 3 will be used to access the memory mapper via the shadow map file at >8300. The form of the call is

```
XOP pointer-location,3
DATA register-location
```

The "register-location" is used to select a map register from 0 to 15. The "location" is taken to be the address of a two word physical address to be loaded into that map register. The effect of the call is

```
MOV  @pointer-location,@register-location
MOV  @pointer-location+2,@register-location+2
MOVB @LMAP1,@MAPPER
```

(Where LMAP1 contains the byte opcode to load the map file from the shadow file #1 and MAPPER is the memory-mapped location of the mapper command register.) Register-location will be in the range of >8000 to >803C for the registers in the shadow map file #1.
In this instance, the XOP call takes two or three words instead of the nine required for the direct code. This is its function, since it will be slower. As examples, consider

```
XOP R0,3
DATA >8000+(10*4)
```

to map memory pointed to by registers R0 and R1 into >A000.
Also,

```
XOP @VAR0,3
```

```
     DATA >8000+(12*4)
```
to map memory pointed to by the two-word value at VAR0 into the page at >C000.

XOP 4 will function like XOP 3, but with the second shadow map file.

XOP 5 will have several functions, selected by its operand, which should be a register number.

```
    XOP 0,5
```
Allocates a block of physical ram at power-up time. The amount of ram needed will be taken from the caller's R0/R1. The ID tag will be taken from the caller's R12. Upon return, the first 4K of the allocated block will be mapped in at >E000. The EQ bit will be set if not enough ram was available. Absolute pointers should not be used in these blocks, as they may move during allocation/deallocation.

```
    XOP 1,5
```
Change the size of an allocated block. New size from caller's R0/R1. A value of zero will delete the block. The first 4K of the block will be mapped in at >E000 upon return. The EQ bit will be set if the operation cannot be performed.

CAUTION:
Changing the allocation may affect running application programs, such as BASIC.

```
    XOP 2,5
```
Find the allocation block with tag matching caller's R12. Map in the first 4K bytes at >F000. EQ bit will be set if not found.

```
    XOP 3,5
```
Find the allocation block with tag matching caller's R12. Map in the first 4K bytes at >E000. EQ bit will be set if not found.

```
    XOP  4,5
```
Used by the system at power-up or reset to determine the amount of memory present, and initialize the memort-top variable.

LINK TP GPL

```
  (1) save current grom address, if desired
  (2) place target grom address in >84A6
  (3) load GPLWS
  (4) BL @>24     (gpllnk)
  (5) load your won workspace
  (6) restore grom address, if saved
```

SYSTEM VARIABLES (In addition to 99/4A variables)
 (see also screen variables under xop description)

```
>841C (double word) Memory allocation fence    CMFENCE
  (first byte available to DSR allocations.)
>8420 (double word) Memory allocation boundary CMTOP
  (first byte allocated to DSR's.)
>849C (word) keyboard debounce delay value.  Default is 1500.
  Make it zero to eliminate delay within the keyscan routine.
  (but application is then responsible for ensuring that
  kscan is not called again within 10ms.)
>849E (word) Speed control delay value.  Default to >210 for
  approximate 99/4A speed match.  Set to zero for full speed.
  Used as a decrementer on each VDP interrupt.  Too large a
```

value will cause system lockup.
>84A0 (word) pointer to Sprite Attribute List in VDP RAM.
defaults to >0300, which was only value supported on 4A.
Used by automotion routine.
>84A2 (word) pointer to Sprite Velocity List in VDP RAM.
defaults to >0780, which was only value supported on 4A.
Used by automotion routine.
>84A4 (word) used by gpl-link routine for return.
>84A6 (word) set up by user as gpl address of routine
called via gpl-link.
>8400 to >84AF: all reserved for system usage.