XOP DEFINITIONS FOR ARMADILLO

The Armadillo system rom will implement the following XOPs. These will only be accessible when the system rom is enabled.

XOP 0 was previously used for assembly language breakpoints in a debugger environment. Since this has never been used in a product release situation, it is subject to redefinition.

XOP 1 will be left with the vector >FFD8,>FFF8, as a user XOP for assembly language routines (such as DEBUG).

XOP 2 will be left with the vector >83A0,>8300, as a user XOP for compatabity, even though it is not known to have been ever used.

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

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

The "register-number" 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 3,R0
      DATA >8000+(10*4)
```

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

```
      XOP 3,@VAR0
      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 5,0
Allocates a block of physical ram at power-up time. The amount of ram needed will be taken from the callers RO. The ID tag will be taken from the callers R12. Upon return, the first 4K of the allocated block will be mapped in at >F000. 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 5,1
Change the size of an allocated block. New size from callers RO. A value of zero will delete the block. The first 4K of the block will be mapped in at >F000 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 5,2
Find the allocation block with tag matching callers R12. Map in the first 4K bytes at >F000. EQ bit will be set if not found.

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