Inter-Office Memorandum

To	Mesa Users	Date	October 27, 1980
From	Jim Sandman, John Wick	Location	Palo Alto
Subject	**Mesa 6.0 XMesa Update**	Organization	SDD/SS/Mesa

XEROX

Filed on: [Iris]<Mesa>Doc>XMesa60.bravo (and .press)

This memo describes the changes in Mesa 6.0 runtime support which incorporate the facilities of XMesa 5.0 into the standard system.

Overview of Extended Memory Support

Mesa now uses the extended memory of Alto II XMs as additional swapping space for code. This means that code and data need not co-exist in the MDS, the primary 64K of memory. Mesa takes advantage of any available extra space automatically; standard Alto programs do not need to be modified to run. Support is provided for up to one million words of memory in blocks of 64K words.

Because Mesa uses extended memory for code segments, it includes a page-level storage allocator for the additional banks. Client programs may request storage in the additional banks by using extensions of the standard procedures in **SegmentDefs**. Mesa provides primitive mechanisms to read and write words in extended memory and to copy blocks of data between banks of memory, but gives no other assistance in accessing information in the extended memory. In particular, arbitrary use of **LONG POINTER**s is *not* supported on the Alto.

Public Interfaces

Unless otherwise stated, all of the facilities in this section are defined in **SegmentDefs**.

Configuration Information

The Mesa runtime system has an internal data structure that contains information about the hardware configuration of the machine on which it is running. Clients may obtain a copy of this data structure by calling **GetMemoryConfig** and should normally test for the existence of extended memory by examining the **useXM** field. The extant banks of memory are indicated by **MemoryConfig.banks**, which is a bit mask (*e.g.,* **MemoryConfig.banks=140000B** implies that banks zero and one exist). Note that this bit mask has been expanded to allow for up to sixteen banks; *constants used to test against it must be changed*.

```
BankIndex: TYPE = [0..17B];

ControlStoreType: TYPE = {Ram0, RamandRom, Ram3k, unknown};

MachineType: TYPE = {unknown0, AltoI, AltoII, AltoIIXM, . . . };

MemoryConfig: TYPE = MACHINE DEPENDENT RECORD [
  reserved: [0..37B],
  AltoType: MachineType,
  xmMicroCode: BOOLEAN,
  useXM: BOOLEAN,
  mdsBank: BankIndex,
  controlStore: ControlStoreType,
  banks: [0..177777B],
  mesaMicrocodeVersion: [0..177777B]];

memConfig: PUBILC READONLY MemoryConfig;

GetMemoryConfig: PROCEDURE RETURNS [MemoryConfig] = INLINE
  BEGIN RETURN[memConfig] END;
```

The field **memConfig.useXM** is true if and only if the following conditions hold:

1) the machine is an Alto II with XM modifications (**AltoType = AltoIIXM**),
2) the Alto has more than one memory bank installed (**banks ~= 100000B**),
3) the Alto has a 3K RAM, or it has a second ROM containing an appropriate version of the XMesa microcode.

The microcode version field tells only the *microcode* version, *not* the Mesa release number. (For example, for Mesa 6.0, **mesaMicrocodeVersion** is 41; Mesa 5.0 version 39 microcode is also supported, although not all features are available.)

Extended Memory Management

The facilities described in this section can be used regardless of the state of **useXM**.

Segments in extended memory are created with the usual primitives in **SegmentDefs**. However, additional "default" parameter values for those procedures that expect a VM base page number have been provided. **DefaultMDSBase** requests allocation anywhere in the MDS. **DefaultXMBase** requests allocation anywhere in the extended memory banks but not in the MDS. **DefaultBase0**, **DefaultBase1**, **DefaultBase2** and **DefaultBase3** request allocation in particular banks. **DefaultANYBase** requests allocation anywhere in the extended memory banks or the MDS. **DefaultBase** is equivalent to **DefaultANYBase** *if the segment is a code segment*, otherwise, it is equivalent to **DefaultMDSBase**.

The following procedures convert between segment handles and long pointers, and work for segments anywhere in the 20-bit address space.

```
LongVMtoSegment: PROCEDURE [a: LONG POINTER] RETURNS [SegmentHandle];
```

LongSegmentAddress: PROCEDURE **[seg**: **SegmentHandle]** RETURNS **[**LONG POINTER**]**;

LongVMtoDataSegment: PROCEDURE **[a**: LONG POINTER**]** RETURNS
[DataSegmentHandle];

LongDataSegmentAddress: PROCEDURE **[seg**: **DataSegmentHandle]**
RETURNS **[**LONG POINTER**]**;

LongVMtoFileSegment: PROCEDURE **[a**: LONG POINTER**]** RETURNS
[FileSegmentHandle];

LongFileSegmentAddress: PROCEDURE **[seg**: **FileSegmentHandle]**
RETURNS **[**LONG POINTER**]**;

The following definitions have been added to **AltoDefs**; they define parameters of the extended memory system.

MaxVMPage: CARDINAL = **7777B**;
MaxMDSPage: CARDINAL = **377B**;
PagesPerMDS: CARDINAL = **MaxMDSPage+1**;

The following procedures convert between page numbers and long pointers, and are analogous to **AddressFromPage** and **PageFromAddress**.

LongAddressFromPage: PROCEDURE **[page: AltoDefs.PageNumber]**
RETURNS **[lp**: LONG POINTER**]**;

PageFromLongAddress: PROCEDURE **[lp**: LONG POINTER**]**
RETURNS **[page**: **AltoDefs.PageNumber]**;

The following procedures check the validity of long pointers and page numbers and raise the indicated errors.

ValidateVMPage: PROCEDURE **[page**: UNSPECIFIED**]**;

InvalidVMPage: ERROR **[page**: UNSPECIFIED**]**;

ValidateLongPointer: PROCEDURE **[a**: LONG UNSPECIFIED**]**;

InvalidLongPointer: ERROR **[lp**: LONG UNSPECIFIED**]**;

The signal **ImmovableSegmentInXM** is raised when **MakeImage** (or **CheckPoint**) discovers a segment in the extended memory banks that cannot be swapped out. (See the section on restrictions, below, for more information about image files).

Long Pointer Support

The facilities described in this section should be used only when **useXM** (see above) is TRUE.

XCOPY is no longer implemented; clients should use **InlineDefs.LongCOPY**. It may only be called when **memConfig.xmMicrocode** is TRUE.

LongCOPY: PROCEDURE **[from**: LONG POINTER, **nwords**: CARDINAL, **to**: LONG POINTER**];**

LongCOPY makes no attempt to validate the long pointers; if they exceed 20 bits or reference non-existent memory, **LongCOPY** will produce unpredictable results.

XBitBlt is no longer implemented; the following extension is not supported by XMesa 5.0 ROMs. The normal AltoIIXM **sourcealt** and **destalt** fields of the BitBlt record (**BitBltDefs.BBTable**) should be used (*do not use the long pointer options*). In addition, if the **unused** word in the **BBTable** is nonzero, the microcode sets the emulator bank register to that value for the duration of the BitBlt. In effect, BitBlt can only be used to move data within a single bank or between the MDS (bank zero) and some other bank.

Restrictions, Limitations, and "Features"

Images and Checkpoints. **MakeImage** cannot preserve the contents of extended memory in the image file it constructs. If **MakeImage** is invoked when **useXM** is TRUE, it will swap out all unlocked file segments in extended memory. (It will also move any locked code segments to the MDS.) If any segments then remain in extended memory, **MakeImage** will refuse to build the image file. Analogous comments apply to **CheckPoint**.

Bank Registers. Mesa assumes it has exclusive control of the emulator bank register on AltoIIXMs. Client programs must not attempt to alter the bank register, but rather must use the public interfaces for moving data to and from extended memory (see **LongCOPY** and **BitBlt**, above).

Segment Alignment. Segments may not cross bank boundaries. The first page of each non-MDS bank is reserved for internal allocation tables.

Swapper Algorithms. The swapper loads a segment into extended memory by first swapping it into primary memory, then copying it to extended memory and releasing the MDS memory space. Thus, if the MDS is so full that the requested segment cannot be swapped in, **InsufficientVM** will be raised, even though sufficient space for the segment may exist in other banks. (Analogous comments apply when swapping out segments that must be written to disk.)

Distribution:
- Mesa Users
- Mesa Group
- SDSupport