Inter-Office Memorandum

To	Mesa Users	Date	May 31, 1978
From	John Wick	Location	Palo Alto
Subject	Mesa 4.0 System Update	Organization	SDD/SD

XEROX

Filed on: [IRIS]<MESA>DOC>SYSTEM40.BRAVO

This memo outlines changes made in the Mesa system code since the last release (October 17, 1977). It also dicusses a number of internal changes made in the system and the microcode; see also the Mesa 4.0 Microcode Update. (In addition, the list of change requests closed by Mesa 4.0 will appear as part of the Software Release Description.)

External Interfaces

Names in square brackets refer to sections of the *Mesa System Documentation* which has also been updated. More details can be found there and in other documentation accompanying this release.

Alto Reserved Locations

Mesa software now conforms to the most recent allocation of Alto reserved locations (*Alto: A Personal Computer System, Hardware Manual*, February, 1978, Appendix H). The only page one location reserved by Mesa is the disaster flag (location 456B).

Basic Mesa

Basic Mesa has been reclassified as released software. To facilitate development of special purpose systems, Basic Mesa no longer includes a keyboard handler; the procedure for adding one is described in the documentation on the Keyboard Package. **[Section 3]**

CheckPoint/Restart

Procedures have been added for writing checkpoint image files, and the bootstrap loader has been modified to load them. Checkpoint files contain only the data (so creating them and loading them is fast); unlike **MakeImage**, **MakeCheckPoint** does not copy code or any other files to the image file. Note this means that none of the files referenced by the checkpoint can be updated or modified in any way. **[Image Files]**

Code Links

The loader has been extended to optionally write external links in the code rather than the global frame (assuming code links were specified in the configuration description). To effect this option, the loader /l switch must be used. Note that if a module calls for code links, loading it will be slower, as the code segment must be swapped in and rewritten. To decrease resident storage requirements, all standard Mesa systems are configured with links in the code. **[Modules]**

Convert

Due to extremely limited microcode space, the convert instruction is no longer part of the Mesa instruction set. It can be simulated with **BitBlt** (see the system display package). [Display]

Debugger Call

A method of invoking the debugger explicitly, without generating a signal, is now available; it causes minimal disruption of the current state of the debuggee. The inline **CallDebugger** is defined in **MiscDefs**. **[Miscellaneous]**

Deleting Configurations

The procedure **UnNewConfig** is now available for unloading a configuration from a running system. The configuration's global frames are deallocated and its code segments are released. In addition, all existing modules are checked for bindings to the configuration being unloaded. **[Modules]**

Display Package

A smaller display package, similar to the Bcpl version, is now standard. It supports a simple display oriented teletype-like interface and an optional typescript file (see **FontDefs** and **DisplayDefs**). The window package is available as a separate configuration (which is no longer supported); see *Window Package*. **[Display Package]**

File Lengths

File lengths and file length hints are no longer kept as a permanent part of each file object. A separate (and optional) length object is allocated only when the length of a file is requested. Since length objects are not required for files containing code segments, this substantially reduces the amount of resident object space required to handle a large number of BCDs. **[File Package]**

Free Storage Package

The free storage package has been modified so that it protects each zone (including the system supplied free storage heap) with a monitor. This enables several processes to share the heap. **[Storage Management]**

Interrupts

For compatability with the new process mechanism, a Nova interrupt now causes a "naked notify" to one of sixteen condition variables. Pointers to these condition variables are contained in fixed locations in page zero (see **ProcessDefs**). **[Processes and Monitors]**

Images

All symbol table references and options have been removed from the system, and the interface to **MakeImage** has been changed to reflect this (the **symbolsToImage** parameter has been dropped). The image file format has also been revised to support checkpoint/restart files (see above). **[Image Files]**

KeyStreams

The keyboard routines have been revised to utilize the new process mechanism; a condition variable is notified when characters are available in the current keystream. The "idle procedure" has been replaced by a WAIT on this condition variable. **[Keyboard Package]**

Memory Management

A number of options have been added to the memory management and swapping facilities. Unlocked read-only segments, such as fonts, are now swapped automatically (previously, this applied only to code segments). The interface to swapping procedures has been expanded to include an (optional) **AllocInfo** parameter, which provides more information about how the memory should be allocated. The new facilities are defined in **AllocDefs**; swap stratagies and swapping procedures are now defined there. **[Segment Package]**

Pause to Debugger

A switch has been added to the NEW command which will invoke the debugger as soon as a configuration has been loaded (in command line mode, before it is started). Also, holding down the control-swat keys while an image file is loaded will now work correctly (the debugger will be invoked as soon as possible). **[Section 4]**

Process Structure

A new process mechanism which supports monitors and condition variables, WAITs and NOTIFYs, and FORK and JOIN has been implemented. The BLOCK operation has been eliminated (a **Yield** procedure is available). Note that several refinements (and some revisions) of the original proposal (in the *Pilot Functional Specification*) have been made. A new chapter in the *Mesa Language Manual* provides complete documentation; additional facilities which are not part of the language are described in the system documention (and **ProcessDefs**). **[Processes and Monitors]**

Warning: In general, facilities provided by the system *are not protected* by monitors. Since Pilot will be available soon, we have not redesigned and retrofitted the Alto/Mesa system to support preemptive processes (other than simple interrupt routines, as before). Except for the free storage package (see above), system facilities shared by more than one process must be protected by a user supplied set of monitor entry procedures.

Run Image

A module is now available which will invoke a Mesa image file (or a Bcpl run file) from the Mesa environment, without returning to the Alto Executive. Any Mesa subsystem which supports command line input (e.g., the compiler or binder, or even the Mesa system itself) can be invoked considerably faster using this facility. **[Image Files]**

StreamIO

To make instantiating multiple instances of this module easier, **StreamIO** no longer takes parameters specifying the input and output streams. Procedures are available which override the default settings. **[StreamIO Package]**

StringDefs

To facilitate conversion from binary to alphanumeric data, **AppendNumber** and other related procedures have been added to **StringDefs**. **WordsForString** now expects a cardinal. Bcpl strings now use packed arrays. **[String Package]**

SystemDefs

A simplified interface to the new memory management facilities has been added in the form of two additional procedures: **AllocateResidentSegment** and **AllocateResidentPages**. [Storage Management]

TrapDefs

Documentation on a number of system generated traps is now available in a new section of the system document. Most traps are converted into signals of the same name: **StartFault**, **ControlFault**, **UnboundProcedure**, **StackError**, etc. **[Traps]**

UnNew

This procedure no longer supports the option of adding the module's frame to the free frame heap (it will be returned there only if it was allocated from there). Note that this procedure does not check for other modules bound to the one being deleted. *Beware of dangling references!* [**Modules**]

Unsigned Compare

The unsigned compare operation (USC) has been removed from **InlineDefs**. Use of the appropriate signed or unsigned comparison operators should be controlled by the type of the variables involved: CARDINAL (unsigned) or INTEGER (signed). See the *Mesa Language Manual* and the *Mesa 4.0 Compiler Update* for more information.

User Interface

Modules to be loaded into the standard Mesa system (and Basic Mesa) can now be specified on the command line (and in command files). A number of switches are available to control loading options; these are described in the *Mesa User's Handbook.*

VMnotFree

The signal **VMnotFree** was inadvertently respelled (it used to be **VMNotFree**). [**Segment Package**]

Window Package

The Mesa window package is no longer part of the standard Mesa system; it is available as a separate configuration. **WindEx** replaces **WManager** for optional use in the debugger. The definitions of **BitBlt** and **Convert** have been removed from **RectangleDefs**. Support for a blinking cursor has been added. [**Window Package**]

Internal Interfaces

The following changes are internal to the implementation and do not affect public interfaces. They may affect performance and/or space requirements, however. For several of these items, furthur information can be founded in the *Mesa 4.0 Microcode Update.*

Alto/Mesa Microcode

The microcode has been completely rewritten to improve its execution speed. The major changes are: 1) several instructions must now be aligned on word boundaries and, 2) certain instructions (notably jumps) require the evaluation stack to be empty except for their operands. (As a side effect of the reorganization, new opcode numbers have been assigned to most instructions.) We have observed improvements in raw execution speed of 20-50%, depending on the dynamic instruction mix.

Alto File System

The hint in the DiskDescriptor containing the number of free disk pages is now maintained properly. The declarations describing Sys.Log have been deleted, since it is no longer supported by the Bcpl OS (versions 14 or later).

Alto Time Standard

The time conversion package is now part of standard system. **UnpackedTime** and **PackDT** have been extended to support GMT, time zones, and daylight savings time (for compatability with Bcpl OS versions 14 or greater).

Bcd Format

This structure has been revised to achieve a space reduction of about 10%. A segment table and name table have been added, as well as support for packed code segments and code links. Provision for a source version stamp has also been included.

BitBlt

BitBlts are now performed entirely in microcode, using the ROM subroutine. They are not only faster, but interruptable as well. **BitBltDefs** now contains the interface to this operation; it has been updated to include the extended memory option.

Cleanup Procedures

Adding a cleanup procedure must now specify the conditions under which the procedure should be called. Several cleanup procedures are no longer called on swapping to and from the debugger.

Warning: Since the interrupt key may preempt a process holding a monitor lock, cleanup procedures must not attempt to enter any monitor. This severly restricts the operations that can be safely performed by cleanup procedures.

Code Packing

All resident code now employs the packed code option implemented by the binder.

Code Segments

The format of code segments has been revised to accomodate the new options for handling external links (storing them in the code and storing them backwards from the frame or code base). Also note that when several modules are packed into the same code segment, only the LRU bit of the first module is examined by the swapper.

ControlDefs

The declarations of control links and local and global frames now use overlaid variant records. The **AV**, **SD**, and **GFT** have been preassigned constant values as in the PrincOps; the REGISTER construct has been revised accordingly (see also *Trap Parameters*). The assignment of System Data indicies is now contained in **SDDefs**.

Descriptor Instructions

The descriptor instructions (DESCB and DESCBS) described in the PrincOps have been implemented.

External Links

External links are now stored and indexed backwards from the global frame base (or code base); this eliminates the "effective" minimum frame size overhead of eighteen. The total number of external procedures, programs, signals, and errors per module must be less than 256.

Field Descriptors

The format of field descriptors has been revised to agree with the D0 design. The read field stack (RFS) and read field code (RFC) instructions have been implemented.

Frame Allocation

The ALLOC and FREE instructions are now implemented in microcode; thus the overhead for large (greater than five word) parameter and result records has been drastically reduced.

Global Frame Format

The global frame overhead has been reduced from ten to three words; all fields relating to the old binding scheme have been eliminated (see also *External Links* and *Main Body Procedure*).

Kernel Function Calls

Several new kernel functions have been added as a result of other extensions (see **SDDefs**). Most entries of public interest are now defined as inlines in definitions modules (e.g. **FrameDefs**, **LoaderDefs**). Provision has been made for all of the traps defined in the PrincOps.

Load State Format

A change in format has reduced the size of the load state substantially. The maximum number of BCDs which can be loaded into a single image file is now about forty.

Long Integers

Addition, subtraction, and comparison of long (32-bit) integers are now implemented in microcode. Multiplication and division are done by software (and are therefore slow).

Main Body Procedure

The main body of a module is now executed in a separate local frame, instead of using the global frame. This eliminates three words from the global frame overhead (the access link, saved pc, and return link).

Nil Pointers

To enable conversion and comparison of both long and short values of null pointers, the value of NIL has been changed.

Novacode interface

To accomodate the implementation of the process opcodes in Nova code (and the removal of block, convert, and bitblt), the interface to the Nova has been revised.

OSStaticDefs

The format of the OS statics region has been revised to reflect the changes in OS version 14 (see the *Alto Operating System Reference Manual*).

Pair Instructions

A number of the pair instructions described in the PrincOps have been implemented on the Alto (notably the RXLP, RILP, and RIGP families).

Processes and Monitors

Due to severe space limitations, all of the process/monitor opcodes (enter, wait, reenter, notify, broadcast, exit, and requeue) are implemented in Nova code, and therefore are considerably slower (relative to other instructions) than they will be on the D0.

Real Data Type

The compiler now generates KFCBs to perform real arithmetic. The SD contains entries for the following floating point operations: FADD, FSUB, FMUL, FDIV, FCOMP, FIX, FLOAT. Note, however, that no implementation of these operations is provided or planned.

Realtime Clock

The low order ten realtime clock bits (maintained in a micro processor register) can now be read by the programmer using the REGISTER construct. This feature was added in conjunction with the program monitoring facilities (see *Xfer Traps*).

Segment and File Objects

Descriptors for files and segments are now allocated from a single pool, rather than from separate tables. This eliminates considerable breakage, at some cost in the speed of enumeration procedures (which are performed rarely). As a consequence, objects are now represented as true variant records (not computed or overlaid), and a number of procedures take either data or file segments as parameters.

Shared Code Segments

A bit has been added to the global frame which indicates if the code is shared by other module instances. In the common case (a single instance of each module), this will eliminate searches of the global frame table every time the code is swapped out.

Start Command

The Mesa Executive's Start command now FORKs to the module, running it as a separate process. This insures that the executive will survive and continue to accept commands even if the user's process is aborted.

Trap Parameters

To eliminate the possibility of clobbering the stack when (possibly nested) traps occur, all trap parameters are now passed in registers (of the micro processor). They are made available to the trap routine using the REGISTER construct.

Uncaught Signals

The method of handling uncaught signals has been revised to accomodate the new process mechanism. Each process no longer includes an instance of the debugger's "nub" as its root; instead, a nub is spliced into the call stack dynamically when the uncaught signal occurs.

Xfer Traps

A mechanism has been added which will optionally cause a trap routine to be invoked for each XFER operation. In addition to performance monitoring, this facility is also useful for finding a large class of bugs, especially clobbers (see TrapDefs).

Distribution:
Mesa Users
Mesa Group