XEROX

INFORMATION PRODUCTS GROUP

System Development Division

September 26, 1977

XEROX SDD ARCHIVES
I have read and understood
Pages______To______
Reviewer______Date______
# of Pages______Ref. 77SDD-339

To:	File
From:	John Wick
Subject:	Mesa 3.0 Tasks
Keywords:	Mesa, Work Plan
Filed On:	<WICK>MESADONE30.BRAVO

This memo identifies those items from the Mesa task list that have been implemented in Mesa 3.0. An update of the task list which reflects these changes will be forthcomming shortly.

Items are organized alphabetically by major component. References are enclosed in square brackets. To distinguish among multiple works by the same person, a terse abbreviation for the title of the reference has been appended to the author's name (e.g., [LampsonCD*]).

Some items (marked with #) have been retained temporarily in Mesa 3.0 to ease conversion; these will be removed in the first maintenance release.

Language

Boolean Enumerated Type

Defining the built-in type BOOLEAN as an enumerated type would allow it to be used as a variant record tag. This case occurs frequently in Diamond data structures. [WickLWG]

Data Modules (#)

The destinction between DATA and PROGRAM modules is no longer meaningful, since all modules are started only once under the new scheme proposed for the binder. *Data modules will be retained temporarily for compatability.* [Sweet]

Directory Names

In conjunction with other changes which improve type checking (primarily export checking for the new binder), the compiler will now verify that the name in the DIRECTORY statement matches the one in the corresponding definitions or program module. The binder will also make this check. [MesaLM]

Empty Intervals

Allowing empty intervals (especially for array index sets) in type definitions would provide some help in implementing sequences until they are incorporated into the language.

External Declarations (#)

The new binding scheme requires that all references to types defined outside a module must be obtained from included modules. The EXTERNAL attribute will therefore be removed from the language. *It will be retained temporarily for compatability.* [Binder, Sweet]

Implementing/Sharing (#)

The IMPLEMENTING construct will be replaced by EXPORTS. For compatability with IMPORTS and EXPORTS, SHARING will be replaced by SHARES. *Both will be retained temporarily for compatability.* [Binder, Sweet]

Implicit Stop

The implicit STOP statement inserted by the compiler between the initialization code and the module's main body will be removed. This implies that all modules need be started only once, regardless of the presence of main body code. [MesaLM]

Imports/Exports

The IMPORTS and EXPORTS lists will be added to the program header. The compiler will now check that a module's procedures match the corresponding definitions in the exported interface. [Binder]

Module Instantiation

Parameter passing will move from NEW to START, allowing a module to be instantiated without running any of its code. This permits the binder to run as a preprocessor. The RESTART statement will be added to absorb the current uses of START without parameters. [Binder]

Module Instantiation - Copy

A form of NEW which copies the bindings from an existing module instance would greatly speed this operation in what is thought to be a common case. [WickUMI]

Overlayed Variant Records

The constructions for referencing computed variant records could be simplified considerably by allowing access without descrimination to those fields which have unique names. [SimonyiMR]

Packed Arrays

Since many applications cannot afford the storage overhead associated with strings, packing array components (at least with byte granularity) seems necessary. The PACKED attribute has been proposed (automatic packing is potentially incompatable with existing software). [GeschkeS]

String Bodies

Some housecleaning on STRINGS is needed, especially if sequences are implemented; at the least, a name should be attached to the string body. The body name could also be used to make clear the pointerness of strings during initialization, a common source of trouble among users. [GeschkeS]

Compiler

Compiling Options

A number of compiling options have been proposed (optimization, cross reference data, warning messages); some user interface for specifying options needs to be designed. The primitive Bcpl switch mechanism could be used. Some options might be included in the source text.

Cross Reference Data

The routines which produce the cross reference dribble file need to be incorporated into the early passes of the compiler. Some way of enabling this option needs to be specified.

Module Format

The format of an object module must be expanded to contain the information needed by the new binder. *This will require recompilation of all existing Mesa programs with the next release.* [Binder]

Program Parameters

The next release of the compiler will type check module parameters (now passed by START). Passing more than five parameters will also be supported.

Signal Descriptors

In the next release, signal values will be structured like procedure descriptors, with a module-unique signal index replacing the entry point number. Like procedure descriptors, signal descriptors can be constructed on the fly, eliminating the need to store the signal value in the global frame. [JohnssonMRI]

Source/Object Mapping

The table produced by the compiler which records source text/object code correspondence will be expanded to include the beginning of each source statement, rather than the beginning of each source line. This will allow finer contol over breakpoint setting. [SatterthwaiteMI, GeschkeMDI]

Static Link

Catch phrases and nested procedures could achieve a better instruction mix if their static links pointed to the first local variable of the enclosing context instead of to the beginning of that frame. [GeschkeMop]

Unwind

In the next release, UNWIND will be a predefined symbol, rather than an external signal. Binding to this symbol will no longer be required.

Variant Record Packing

The current algorithm for packing fields in variant records can leave unused bits in some variants, forcing the compiler to disallow comparisons (the unused bits contain garbage). Removing the holes would make some variants one word longer.

Warning Messages

A number of potential errors could be recognized by the compiler and translated into warning messages ("variable not referenced", for example). Some compiler option is needed to specify whether these messages should be ignored or printed. Each condition needs to be analysed carefully; warnings do no good if they are routinely ignored.

System Building

Binding (#)

The current binder is being replaced by a preprocessor designed to be run before any loading takes place. It inputs a Configuration Description and combines all the modules it specifies into a single package -- a Binary Configuration Description or BCD -- that can be loaded as a unit. *To ease conversion to the new scheme, the old binder will be retained temporarily.* [Binder]

Bootstrapping

Converting the compiler, the new and old binders, the system, and the debugger to the new binding scheme is a straightforward but time consuming task (see **Operations**). At least three full bootstraps will be required. Conversion of the system bootstrapping code (BootMesa) will require a substantial rewrite of that system. In the interim, it can continue to use the old binding scheme.

Loading

A new loader is required which understands Binary Configuration Descriptions. Eventually, it should be capable of loading (and unloading) an arbitrary number of BCDs into a running system. The initial implementation may be limited to a single user BCD. [Binder]

Packaging

Configurations can be packaged in several ways; for example, a configuration may or may not include sysmbol tables for debugging purposes. Utility functions will eventually be required to package the code, symbols, and BCDs according to a number of options. [Binder]

SubSystems

The procedures which save the state of a running system in an Image file must be redesigned to process BCDs. [Binder]

Version Checking

The increase in packaging options made possible with configurations requires more thorough version checking to insure that all of the BCDs, the runtime system, and the debugger are consistent. Version identification must be propagated not only among compiled modules, but BCDs and Image files as well. [Binder]

Architecture

Byte Arrays

The present read and write byte instructions (RSTR and WSTR) have the Mesa string format built into them. To support packed arrays, this offset needs to be specified in an alpha instruction field. [GeschkeMop]

Procedure Descriptor Format

The next release of the system will contain a new procedure descriptor format (the global frame and entry point indexes will be interchanged, in anticipation of conversion to the Dstar instruction set). Complete compatability with the Ds may be included, depending on available microstore.

System

Code Segment Prefix

The format of the code segment prefix has been revised to include the offset and length of the linkage vector (including external signals) in the global frame. *This format restricts the number of global frame table entries per module to a maximum of four, and therefore the number of entry points per module to 128.* These restrictions are consistent with the Dstar design.

Deleting Modules

A kernal function which deleted a module instance would enable applications to move once-only code into separate modules, allowing the space to be reclaimed after initialization. Dangling references to deleted modules might cause a problem; a check could be made, but it involves swapping in every code segment in the system. [JohnssonMRI]

Disk Stream Extensions

A number of more convenient procedures have been proposed, including one which creates a stream from a file name instead of a file handle, as well as a procedure for backing up a stream index.

Global Frame Breakage

The packaging of several modules into a single configuration will allow the loader to allocate all the global frames in a single segment. Provided modules are combine into configurations prior to loading, most breakage will be eliminated. [JohnssonMRI]

Interrupt Handling

The inefficiency of disabling/enabling interrupts has been a major problem for the Mesa FTP and Juniper projects. The complexity of the interface (with the Nova emulator) has been a major cause of system bugs. This revision proposes to implement these operations in microcode. [JohnssonRHI]

Kernal Function Calls

A number of proposals involve additions (and deletions) to the system dispatch table (see *Module Instantiation - Copy, Constant Initialization, Deleting Modules, Start Trap*).

Memory Statistics

In order to provide for gathering memory usage statistics, the allocators provided with the system need to keep better track of their data structures (primarily the frame heap and free storage heap). This would also enable the heaps to be pruned.

Reorganized Interfaces

Because the new binding scheme requires all external procedure and signal definitions to be obtained from included modules, some reorganization of the system definitions files (mostly additions) was required.

Signaller Improvements

Because the signaller was one of the first modules implemented in Mesa, it makes rather poor use of the facilities which are now available. Several modifications have also been proposed to improve its efficiency.

Start Trap

To avoid a flurry of code swapping (to run initialization code) when systems are first loaded, a trap could be generated when a call to a module which has not been initalized is attempted. A kernal function could clean things up and retry the transfer. [Binder]

Debugger

Configuration Descriptions

A number of debugger commands need to be rethought and reimplemented based on the information now contained in configuration descriptions. The debugger must deal with several new data structures built by the binder and the loader. [GeschkeMDI, Koalkin]

Module Format

The debugger must be modified to accept the new module format, and perform some minimal processing on the BCDs loaded into the user's core image. *For compatability, the old owner link, binding entry, binding path, and symbol table pointer in the global frame will be retained, allowing the debugger to be converted gradually.* [Binder, Sweet]

Source/Object Mapping

As the pressure for code optimizations increases (and open coded procedures become available in the language), the accuracy and attractiveness of source level debugging will be lost. A compiler option which turns off optimization seems to be the only alternative. [SatterthwaiteMI, GeschkeMDI]

Operations

Version Identification

In conjunction with the new release policy, a set of version numbers will be assigned to all released software. Whatever scheme is chosen, it should be compatable with the one used by the Tools Librarian.

Support

Documentation

Obviously more documentation will be required with all future releases. Major components needed for the upcoming release all center on the binder. A new document on configurations and system building is needed, and several chapters of the *Mesa Language Manual* will require revision. Shortly thereafter, a major revision of the *Mesa System Document* (including the debugger section) will be required.

References

Binder Working Group. *Binding in Mesa.* February 18, 1977.
Geschke. *Mesa debugger issues.* January 10, 1977.
______. *Sequences.* January 28, 1977.
Geschke, Johnsson. *Modifications to Mesa opcodes.* February 28, 1977.
Geschke, Johnsson, Sweet. *Enrty vectors and control links.* March 25, 1977.
Geschke, Satterthwaite. *Loose ends in Mesa.* March 11, 1976.
Guarino. *Results of meeting . . .* June 5, 1977.
Johnsson. *Mesa runtime issues.* January 11, 1977.
______. *New system display facilities.* February 1, 1977.
______. *Revised handling of interrupts in Alto/Mesa.* May 26, 1977.
______. *Minutes of D0 conversion meeting.* June 27, 1977.
Koalkin. *Plans for the Mesa Debugger.* April 21, 1977.
Lampson. *Mesa issues.* January 6, 1977.
______. *Proposed OIS processor changes.* January 14, 1977.
______. *Mesa disk space.* January 19, 1977.
Lampson, et all. *Report on the Programming Language Euclid.* July, 1976.
Lampson, Lynch, Shultz. *Mesa priorities.* April 18, 1977.
Lampson, Wick. *Conversion to D* environment.* January 19, 1977.
Lynch. *Summary of Mesa issues state.* January 24, 1977.
Maybury, Mitchell. *Mesa Language Manual.* October 1, 1976.
Parsley. *A proposed design for the core debugger interface.* May 11, 1977.
Redell. *Proposed syntax and semantics of Mesa processes* (draft). April 28, 1977.
Satterthwaite. *Unresolved Mesa issues.* September 10, 1976.
______. *Mesa issues.* January 10, 1977.
______. *Mesa language issues.* January 28, 1977.
______. *Attributes of variables in Mesa.* March 29, 1977.
Simonyi. *Mesa issues.* December 22, 1976.
______. *Mesa binding.* December 30, 1976.
______. *Mesa requests.* February 18, 1977.
Smokey. *Mesa runtime and Mesa debugger evolution.* April 11, 1977.
Sweet. *Binder working group minutes.* January-February, 1977.
Swinehart. *Humus Conventions . . .* March 15, 1977.
Thacker. *OIS Processor Principles of Operation.* April, 1977.
Wick. *Unresolved Mesa issues -- binding.* January 11, 1977.
______. *Global frames.* January 31, 1977.
______. *Mesa language working group minutes.* January-February, 1977.