Inter-Office Memorandum

To	Mesa Floating Point Users	Date	November 4, 1980
From	Larry Stewart	Location	Palo Alto
Subject	Mesa 6 Floating Point Package	Organization	CSL

XEROX

Filed on: [Ivy]<CedarLib>Real>MesaFloat60.bravo, .press

This memo describes the Floating Point package for Mesa 6.

Compatibility

There have been many changes from the Mesa 5 floating point. In particular, the number format has changed to conform to the proposed IEEE floating point standard, the floating point definitions files have all changed, and compiler support is considerably different.

Distribution and support

For users within PARC/CSL, this package resides on [Ivy]<CedarLib>Real>. Messages about floating point should be directed to [Ivy]<CedarLib>Real>Support.dl. Users of the package should request (by sending a message to Support.dl) to be added to [Ivy]<CedarLib>Real>Users.dl.

There are various test programs available for floating point. If you need help, send a message to Support.dl, as above.

Number Format

A Mesa REAL is a 32 bit floating point number consisting of a sign bit, 8-bit excess 127 exponent, hidden bit, and 23 bit mantissa with interpretation as the fraction part of a number between 1 and 2. Mesa REALs conform to the proposed IEEE standard [1].

Major features of the proposal are:

Careful specification of formats and arithmetic operations.

Careful specification of conversion to and from Decimal.

Gradual underflow, which reduces the bad effects of underflow.

Not-A-Numbers, distinguished bit patterns which either trap or propagate unchanged through operations.

Client settable rounding mode, exceptions individually maskable, "sticky" exception flags.

The Mesa implementation of IEEE floating point provides fixed modes for handling of rounding, denormalized numbers, and infinity. The implementation also enables a fixed set of exceptions. An optional interface provides variable modes for access to the full generality of the standard.

New language support

The Mesa 6 Compiler provides the type REAL, and generates code for evaluation of expressions involving the operations on REALs +, -, *, /, _, Float (conversion from LONG INTEGER to REAL, a widening automatically supplied by the compiler), and compare (=, #, <, <=, >=, >). In addition, the Mesa 6 compiler supports REAL literals, REAL intervals, and the unary operations - (negation) and ABS [2]. The Mesa 6 debugger will print REALs without any added modules.

The Mesa 6 debugger will print REAL types correctly, but cannot handle REAL expressions or assign to a REAL.

About the implementation

The Mesa compiler generates instructions for the operations mentioned above. Additional operations (Fix, Round) are declared as MACHINE CODE, they generate instructions as well. Because microcode support is required, Mesa 6 floating point will not work on the Alto I or on Alto IIs with version 39 or earlier Mesa ROMs. The Alto II, D0, and Dorado all use different microcode, but the supporting mesa package is the same, hence it is possible to run the same compiled code on all machines. There is a compiler switch (/f, /-f) for floating point. It should be used as /f or just left off altogether.

Floating point microcode for the Alto II is supplied by the Mesa 6 version of RunMesa.run. Floating point microcode for the Dorado is included in the standard Cedar microcode release. Floating point microcode for the D0 and Dolphin should be available from your support organization.

Available Interfaces

The interface *Real* provides the basic facilities needed by most users of floating point. Additional numerical functions are provided by *RealFns*. The interface *RealOps* supplies the same operations as Real, but with all the modes explicit, giving the full generality of the IEEE standard. *RealConvert* supplies some procedures for conversion of Mesa 6 REALs to and from other formats. *Ieee* supplies access to internal details of floating point numbers. Some of these facilities are described below. Users should look at the definitions files in case of any uncertainty; they represent the truth.

What you need

Real, RealFns, RealOps, and Ieee are implemented by the configuration RealImpl.bcd; RealConvert is implemented by the module RealConvertImpl. You may wish to have RealImpl.symbols around in case you get any exceptions.

The most common pitfalls

Don’t forget to load and START RealImpl. You may get along fine for a while because most operations are in microcode, but "hard" operations and operands are handled by the supporting Mesa code, which must be there! The symptom will probably be a ControlFault or StackError.

Have the right microcode on your D0 or Dolphin

Don’t try to use REALs on an Alto I or Alto II with old Mesa ROMs.

Don’t forget to request to be added to the Users distribution list. The maintainers are unlikely to be sympathetic if you hit an already fixed bug.

in Real (exported by RealImpl)

The configuration RealImpl.bcd IMPORTS StringDefs and EXPORTS Real, RealFns, Ieee, and RealOps. RealImpl also IMPORTS InlineDefs, but inline procedures only.

InitReals: PROCEDURE;

RealControl: PROGRAM;

InitReals must be called or RealControl STARTed before any floating point operations are executed (including any initialization other than REAL literals which occur in a global frame containing a call to InitReals). It is ok to call InitReals more than once. Fine points: STARTing the configuration RealImpl by including it on the command line is good enough. There should not be any need to call InitReals on return from a MakeImage.

Exceptions

```
Flag: TYPE = BOOLEAN _ FALSE;

Exception: TYPE = {fixOverflow, inexactResult, invalidOperation, divisionByZero, overflow, underflow};

ExceptionFlags: TYPE = PACKED ARRAY Exception OF Flag;

UsualExceptions: ExceptionFlags = [fixOverflow: TRUE, invalidOperation: TRUE, divisionByZero: TRUE,
    overflow: TRUE];
```

RealException: SIGNAL [flags: ExceptionFlags, vp: POINTER TO Extended] RETURNS [POINTER TO Extended];

POINTER above will be replaced by REF for Cedar Mesa.

RealError: ERROR;

RealException is raised whenever an enabled exception occurs. The parameter flags indicates which exceptions occurred during the faulted operation (including any which occurred but were not enabled). For operations called through the interface Real (or by the Mesa compiler), the only enabled exceptions are those listed under UsualExceptions above. Usually a catch phrase for RealException can RESUME. The client is expected to fix up the initial result represented by vp, then return the new value to use as the result of the faulted operation. If the client returns NIL, a default result specified by the IEEE standard will be used.

The exception invalidOperation when raised as a result of compare or any of the real to fixed point conversion operations is not resumable. If the client RESUMEs anyway, the ERROR RealError will be raised.

Fine point: because of the way SIGNAL catching is implemented, a catch phrase on a MACHINE CODE procedure (such as Fix or Round) or a catch phrase on a block enclosing floating point operations will not work; however, a catch phrase at least one procedure call removed from a floating point operation will work.

Clients wishing to test for uninitialized storage of REAL types may wish to set that storage to the value Real.TrappingNaN. This distinguished operand will produce a SIGNAL if it is ever picked up as an operand of a floating point operation (other than assignment).

Default results of exceptional conditions

fixOverflow: returns least significant bits of the correct result.
inexactResult: returns rounded version of preliminary result
invalidOperation: returns Not-a-Number characteristic of the faulted operation.
divisionByZero: returns Not-a-Number DivideInfinityNaN.
overflow: returns infinity of the correct sign.
underFlow: returns de-normalized number.

Conversion operations

Fix: PROCEDURE [REAL] RETURNS [LONG INTEGER];

FixI: PROCEDURE [REAL] RETURNS [INTEGER];

FixC: PROCEDURE [REAL] RETURNS [CARDINAL];

RoundLI: PROCEDURE [REAL] RETURNS [LONG INTEGER];

RoundI: PROCEDURE [REAL] RETURNS [INTEGER];

RoundC: PROCEDURE [REAL] RETURNS [CARDINAL];

The Fix operations use rounding mode round-to-zero (truncation). The Round operations use mode round-to-nearest These operations can raise exceptions such as fixOverflow or invalidOperation.

Float: PROCEDURE [LONG INTEGER] RETURNS [REAL];

Float is generated automatically by the compiler, but a client may wish to use it explicitly.

Input and Output

ReadReal: PROCEDURE [get: PROCEDURE RETURNS[CHARACTER],
putback: PROCEDURE[CHARACTER] _ DefaultPutback] RETURNS [REAL];

StringToReal: PROCEDURE [STRING] RETURNS [REAL];

The procedures which convert to REAL may generate exceptions depending on the value represented by the input. ReadReal must read past the end of the number input; it uses putback to return the last character read. DefaultPutback discards the character.

MaxSinglePrecision: CARDINAL = 9;
DefaultSinglePrecision: CARDINAL = 7;

WriteReal: PROCEDURE [cp: PROCEDURE[CHARACTER], r: REAL,
precision: CARDINAL _ DefaultSinglePrecision, forceE: BOOLEAN _ FALSE];

AppendReal: PROCEDURE [s: STRING, r: REAL,
precision: CARDINAL _ DefaultSinglePrecision, forceE: BOOLEAN _ FALSE];

The procedures which convert from REAL will not generate exceptions. If forceE is TRUE or the value is not IN [0.000001..1000000) then the number will be printed in scientific notation. The output value will have precision significant digits.

in WFReal (exported by WFRealImpl)

These procedures are intended for use as print procedures in the WriteFormatted package. See the WriteFormatted package documentation. WFReal and WFRealImpl are distributed with the WriteFormatted package.

InitWFReals: PROCEDURE;

WFWriteReal: PROCEDURE [rp: UNSPECIFIED, f: STRING, p: PROCEDURE [CHARACTER]];

WFWriteEReal: PROCEDURE [rp: UNSPECIFIED, f: STRING, p: PROCEDURE [CHARACTER]];

in RealFns (exported by RealImpl)

Exp: PROCEDURE [REAL] RETURNS [REAL];

For an input argument n, returns e^n (e=2.718...). Computed by continued fractions.

Log: PROCEDURE [base,arg: REAL] RETURNS [REAL];

Computes logarithm to the base base of arg. Computed by Ln(arg)/Ln(base).

Ln: PROCEDURE [REAL] RETURNS [REAL];

Computes the natural logarithm (base e) of the input argument.

SqRt: PROCEDURE [REAL] RETURNS [REAL];

Calculates the square root of the input value by Newton's iteration.

Root: PROCEDURE [index,arg: REAL] RETURNS [REAL];

Calculates the index root of arg by $e^{(Ln(arg)/index)}$.

Power: PROCEDURE [base,exponent: REAL] RETURNS [REAL];

Calculates base to the exponent power by $e^{(exponent*Ln(base))}$.

Sin: PROCEDURE [radians: REAL] RETURNS [REAL];

SinDeg: PROCEDURE [degrees: REAL] RETURNS [REAL];

Cos: PROCEDURE [radians: REAL] RETURNS [REAL];

CosDeg: PROCEDURE [degrees: REAL] RETURNS [REAL];

Tan: PROCEDURE [radians: REAL] RETURNS [REAL];

TanDeg: PROCEDURE [degrees: REAL] RETURNS [REAL];

Computes the trigonometric function by polynomial; good to 7.33 decimal places.

ArcTan: PROCEDURE [y, x: REAL] RETURNS [radians: REAL];

ArcTanDeg: PROCEDURE [y, x: REAL] RETURNS [degrees: REAL];

Good to 8.7 decimal places.

AlmostZero: PROC [x: REAL, distance: [-126..127]] RETURNS [BOOLEAN];

Returns TRUE if ABS[x]<2^distance.

AlmostEqual: PROC [x, y: REAL, distance: [-126..127]] RETURNS [BOOLEAN];

Returns TRUE if (ABS[x-y]/MAX[ABS[x],ABS[y]])<2^distance.

in RealConvert (exported by RealConvertImpl)

The module RealConvertImpl IMPORTS InlineDefs (inline procedures only).

Mesa5ToIeee: PROC [LONG UNSPECIFIED] RETURNS [REAL];

Converts a Mesa 5 REAL to an equivalent Mesa 6 REAL.

BcplToIeee: PROC [LONG UNSPECIFIED] RETURNS [REAL];

Converts a Bcpl REAL to an equivalent Mesa 6 REAL. Bcpl REALs occur in some Press format files.

IeeeToBcpl: PROC [REAL] RETURNS [LONG UNSPECIFIED];

Converts a Mesa 6 REAL to an equivalent Bcpl REAL. Bcpl REALs occur in some Press format files. This operation may raise the signal below on impossible conversions.

CVError: ERROR;

References

[1] "An Implementation Guide to a Proposed Standard for Floating Point Arithmetic", Jerome T. Coonen, *Computer Magazine*, January, 1980.

[2] Mesa 6.0 Compiler Update, Ed Satterthwaite, [Iris]<Mesa>Doc>Compiler60.memo, May 22, 1980.