November 9, 1981

Dave -

This package is a basic introduction to the architecture simulation. The whole simulation occupies about 300 pages of Ada code in 75 files, so I felt it would be easier to start with a manageable subset of the operations, rather than deluge you with the entire structure. I will be happy to send you other pieces as you want them, and we can review the parts you don't have when you visit in December. We can also discuss any questions or comments you have by telephone.

I'll briefly describe what material I've included and outline the procedure I recommend for using it:

Section 1 includes a brief textual description of each of the instructions of the machine and of the operations on objects. The descriptions are not always complete or precise, but they do fairly well represent the intended semantics. We have not been meticulous in maintaining this documentation, so there may be a few places where the descriptions do not reflect most recent versions of some operations.

The package CODE defines the architectural view of the instruction set. The most important definition in the package is the type INSTRUCTION which occurs near the end. Also important are the types OPERAND_CLASS and OPERATOR which are defined on page 2.

The package STACKS defines almost all portions of the runtime representation of programs and the mechanisms by which instruction execution takes place. The type WORD_KIND on page 2 specifies all the kinds of objects which may appear on control stacks. The type CONTROL_QUAD on page 5 defines the actual formats of the control stack words. All actual data objects are either DATA_VAR's (where the object value is in the word on the control stack (in most cases)) or STRUCTURE_VAR's (records, arrays where the value lives in the data stack or in a collection). ENTRY_, FAMILY_, ACCEPT_, SELECT_, and DELAY_VAR's are the structures used in intertask communication. The ACTIVATION_STATE and ACTIVATION_LINK words are the mark stack control words. The other control words define module state information. The type TYPE_QUAD specifies the format of words in descriptors on the type stack.

The top level procedure of the simulation is SIMULATE, which uses the two main packages INSTRUCTION_UNIT and EXECUTION_UNIT. The operation of the simulation is embodied in the procedure CYCLE, which does the obvious Fetch / Dispatch.

INSTRUCTION_UNIT has responsibility for keeping the instruction buffer full, providing the next instruction, and maintaining the program counter.

The package EXECUTION_UNIT specifies a number of subpackages which execute particular subsets of the instructions and provides the procedure DISPATCH which essentially does the op decode and calls the appropriate package to do further decode and/or execute the instruction.

The package MOVE is one of the subpackages of EXECUTION_UNIT which implements the VAL, STO, and REF instructions.

Let me digress from the cataloging of sections for a moment to describe the execution of an instruction and how to use this documentation. Suppose one wanted to know what the VAL instruction does (the first instruction in the description) -

1. CODE says that the argument to a VAL instruction is an object reference, which is a lex level and delta.

2. In the cycle procedure of SIMULATE, the INSTRUCTION_UNIT.FETCH could return a VAL as the current instruction, and this would be sent to the execution unit for dispatch.

3. EXECUTION_UNIT.DISPATCH decodes the VAL opcode and calls the procedure VALUE of the subpackage MOVE with the lex level, delta.

4. MOVE.VALUE does a FRAME.READ which resolves the lex level, delta to an actual control stack address and reads the word at that address. VALUE changes the visibility of the copy to local and then does a CONTROL.PUSH which pushes the copy on top of the control stack.

The simulation acts in a similar fashion for all the instruction. Some are a bit more complicated, of course, but the principle is the same.

The package OBJECT_OP implements the DECL, VAR, and OP instructions vis-a-vis EXECUTION_UNIT, but basically OBJECT_OP just does an operand class decode and dispatches to other packages (INT_OP's, MODULE_OP's, ARRAY_OP's, etc.)

INT_OP is the subpackage of OBJECT_OP which implements operations on integers. For example, the instruction

```
CURRENT_INSTRUCTION  =  (OP     INT_CLASS, PLUS_OP)
```

```
would follow the chain  EXECUTION_UNIT.DISPATCH (CURRENT_INSTRUCTION)
                        OBJECT_OP.DISPATCH      (INT_CLASS, PLUS_OP)
                        INT_OP.ARITHMETIC_OP    (PLUS_OP)
```

and similarly for other instructions, including DECL (for type declaration) and VAR (for variable declaration).

In the simulation, all the memories are implemented as instantiations of a generic ADDRESS_SPACE package which maps a logical name consisting of a segment id and displacement to an actual data item. Each of the memory kinds (program, control, type, data, import, queue), however, has certain operations and addressing which is specific to it. Hence, each of the memories is defined in a package which provides the additional facilities. For example, the package CONTROL declares the CONTROL.MEMORY address space, but also maintains the top of stack register and operations to PUSH and POP as well as READ and WRITE.

I have included just the visible part of the ADDRESS_SPACE generic, which is the interface of the simulation to the virtual memory system. The operations provided by ADDRESS_SPACE's are fairly low level, including creation, deletion, of segments, allocation of pages, and reading and writing.

The package SUBPROGRAM_OP is a subunit of EXECUTION_UNIT which implements the PROC instruction for declaring a procedure object, and the CALL and various EXIT instructions. Much of the complexity of subprogram call and exit is due to the possibility of having dependent tasks declared in a frame, or other structures which require deallocation upon exit. SUBPROGRAM_OP depends very heavily on several other packages.

The package FRAME maintains the limited display and is concerned with pushing and popping frames on subprogram call and exit.

Well, I hope the brief description above is adequate to help you understand what we have tried to do with the architecture simulation. I would like to discuss it with you by telephone later this week or next week sometime. We are all looking forward to working with you and anticipate your arrival.

Dave Stevenson

```
  SSSSSSSS      IIIIII      MM      MM
  SSSSSSSS      IIIIII      MM      MM
SS                II        MMMM  MMMM
SS                II        MMMM  MMMM
SS                II        MM  MM  MM
SS                II        MM  MM  MM
  SSSSSS          II        MM      MM
  SSSSSS          II        MM      MM
        SS        II        MM      MM
        SS        II        MM      MM
        SS        II        MM      MM
        SS        II        MM      MM
SSSSSSSS        IIIIII      MM      MM
SSSSSSSS        IIIIII      MM      MM




LL              OOOOOO        GGGGGGGG                      11
LL              OOOOOO        GGGGGGGG                      11
LL            OO      OO    GG                            1111
LL            OO      OO    GG                            1111
LL            OO      OO    GG                              11
LL            OO      OO    GG                              11
LL            OO      OO    GG                              11
LL            OO      OO    GG                              11
LL            OO      OO    GG  GGGGGG                      11
LL            OO      OO    GG  GGGGGG                      11
LL            OO      OO    GG      GG        ....          11
LL            OO      OO    GG      GG        ....          11
LLLLLLLLLL      OOOOOO        GGGGGG          ....        111111
LLLLLLLLLL      OOOOOO        GGGGGG          ....        111111



*START* Job SIM Req #201 for DHB    Date 27-Feb-82 15:55:23 Monitor: Rational Ma
File PS:<DHB.SIM>SIM.LOG.1, created: 27-Feb-82 15:45:00
        printed: 27-Feb-82 15:55:23
Job parameters: Request created:27-Feb-82 15:53:31   Page limit:27   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

```
[PHOTO:  Recording initiated  Sat 27-Feb-82 3:45PM]

End of <DHB>COMAND.CMD.5
@adas
Adas compiler - type Help; for help

-> open dim;
Can't open PS:<SIM.LIB>DIM.LIB

-> open sim;
Library PS:<SIM.LIB>SIM.LIB opened

-> fdir;
package       DEC10_INSTRUCTIONS
source PS:<ADAM.ENV>DEC10.ADA

package       TTY_IO
source PS:<ADAM.ENV>TTYIOV.ADA
source PS:<ADAM.ENV>TTYIOB.ADA  code PS:<ADAM.LIB>TTYIOB.REL

package       FILE_IO
source PS:<ADAM.ENV>FILIOV.ADA
source PS:<ADAM.ENV>FILIOB.ADA  code PS:<ADAM.LIB>FILIOB.REL

package       TEXT_IO
source PS:<ADAM.ENV>TXTIOV.ADA
source PS:<ADAM.ENV>TXTIOB.ADA  code PS:<ADAM.LIB>TXTIOB.REL

package       INPUT_OUTPUT            generic
source PS:<ADAM.ENV>GNRCIO.ADA

package       VT100
source PS:<ADAM.ENV>VT100V.ADA
source PS:<ADAM.ENV>VT100B.ADA  code PS:<ADAM.LIB>VT100B.REL

package       ENUMERATION_IO_UTILITIES
source PS:<SIM.CMD>ENUTIL.ADA
source PS:<SIM.CMD>ENUTIL.ADA  code PS:<SIM.CMD.REL>ENUTIL.REL

package       ENUM_IO                 generic
source PS:<SIM.CMD>ENUMIO.ADA

package       COMMAND_LINE_PROCESSOR generic
source PS:<SIM.CMD>CMDLIN.ADA

package       ID_STACK                generic
source PS:<SIM.MEM>IDSTAK.ADA

package       UNIQUE_ID               generic
source PS:<SIM.MEM>UNIQID.ADA

package       SIMULATION_IO
source PS:<SIM.CMD>SIMIOV.ADA
source PS:<SIM.CMD>SIMIOB.ADA  code PS:<SIM.CMD.REL>SIMIOB.REL

package       BASE ✓
source PS:<SIM.DEF>BASTYP.ADA
source PS:<SIM.DEF>BASTYP.ADA  code PS:<SIM.DEF.REL>BASTYP.REL
```

```
package          PROCESSORS
source PS:<SIM.DEF>PROCSR.ADA
source PS:<SIM.DEF>PROCSR.ADA   code PS:<SIM.DEF.REL>PROCSR.REL

package          TASKS
source PS:<SIM.DEF>TASKSV.ADA
source PS:<SIM.DEF>TASKSB.ADA   code PS:<SIM.DEF.REL>TASKSB.REL

package          EXCEPTIONS
source PS:<SIM.DEF>EXCPTV.ADA

package          OFFSET
source PS:<SIM.DEF>OFFSET.ADA
source PS:<SIM.DEF>OFFSET.ADA   code PS:<SIM.DEF.REL>OFFSET.REL

package          CODE ✓
source PS:<SIM.DEF>CODE.ADA
source PS:<SIM.DEF>CODE.ADA   code PS:<SIM.DEF.REL>CODE.REL

package          STACKS ✓
source PS:<SIM.DEF>STACKS.ADA
source PS:<SIM.DEF>STACKS.ADA   code PS:<SIM.DEF.REL>STACKS.REL

package          CONVENTION ✓
source PS:<SIM.DEF>CNVENT.ADA
source PS:<SIM.DEF>CNVENT.ADA   code PS:<SIM.DEF.REL>CNVENT.REL

package          ENTRYS
source PS:<SIM.DEF>ENTRYS.ADA
source PS:<SIM.DEF>ENTRYS.ADA   code PS:<SIM.DEF.REL>ENTRYS.REL

package          MESSAGES
source PS:<SIM.DEF>MESSAG.ADA
source PS:<SIM.DEF>MESSAG.ADA   code PS:<SIM.DEF.REL>MESSAG.REL

package          EVENTS
source PS:<SIM.DEF>EVENTS.ADA
source PS:<SIM.DEF>EVENTS.ADA   code PS:<SIM.DEF.REL>EVENTS.REL

package          DESCRIPTOR ✓
source PS:<SIM.DEF>DSCRPT.ADA
source PS:<SIM.DEF>DSCRPT.ADA   code PS:<SIM.DEF.REL>DSCRPT.REL

package          TRACE
source PS:<SIM.CMD>TRACEV.ADA
source PS:<SIM.CMD>TRACEB.ADA   code PS:<SIM.CMD.REL>TRACEB.REL

package          OPS ✓
source PS:<SIM.CPU>OPSV.ADA

package          RAISES
source PS:<SIM.CPU>RAISEV.ADA

package          ADDRESS_SPACE           generic
source PS:<SIM.MEM>ADRSPC.ADA

package          PROGRAM
source PS:<SIM.MEM>PRGRMV.ADA
source PS:<SIM.MEM>PRGRMB.ADA   code PS:<SIM.MEM.REL>PRGRMB.REL
```

```
package       CONTROL
source PS:<SIM.MEM>CNTRLV.ADA
source PS:<SIM.MEM>RFCNTB.ADA  code PS:<SIM.MEM.REL>RFCNTB.REL

package       TYP
source PS:<SIM.MEM>TYPEV.ADA
source PS:<SIM.MEM>TYPEB.ADA  code PS:<SIM.MEM.REL>TYPEB.REL

package       WORDS
source PS:<SIM.MEM>WORDSV.ADA
source PS:<SIM.MEM>WORDSB.ADA  code PS:<SIM.MEM.REL>WORDSB.REL

    package       BITWISE
    source PS:<SIM.MEM>BITWB.ADA  code PS:<SIM.MEM.REL>BITWB.REL


package       DATA
source PS:<SIM.MEM>DATAV.ADA
source PS:<SIM.MEM>DATAB.ADA  code PS:<SIM.MEM.REL>DATAB.REL

package       IMPORT
source PS:<SIM.MEM>IMPRTV.ADA
source PS:<SIM.MEM>IMPRTB.ADA  code PS:<SIM.MEM.REL>IMPRTB.REL

package       QUEUE
source PS:<SIM.MEM>QUEUEV.ADA
source PS:<SIM.MEM>QUEUEB.ADA  code PS:<SIM.MEM.REL>QUEUEB.REL

package       ALLOCATOR
source PS:<SIM.MEM>ALLOCV.ADA
source PS:<SIM.MEM>ALLOCB.ADA  code PS:<SIM.MEM.REL>ALLOCB.REL

package       BLOCK_DATA_OP
source PS:<SIM.MEM>BLKOPV.ADA
source PS:<SIM.MEM>BLKOPB.ADA  code PS:<SIM.MEM.REL>BLKOPB.REL

package       CODE_IO
source PS:<SIM.CMD>CODIOV.ADA
source PS:<SIM.CMD>CODIOB.ADA  code PS:<SIM.CMD.REL>CODIOB.REL

package       WORD_IO
source PS:<SIM.CMD>WRDIOV.ADA
source PS:<SIM.CMD>WRDIOB.ADA  code PS:<SIM.CMD.REL>WRDIOB.REL

package       INTERFACE
source PS:<SIM.CMD>SIMIFV.ADA
source PS:<SIM.CMD>SIMIFB.ADA  code PS:<SIM.CMD.REL>SIMIFB.REL

    package       COMMAND_INTERFACE
    source PS:<SIM.CMD>CMDIFB.ADA  code PS:<SIM.CMD.REL>CMDIFB.REL

        package       TOPS20_FORKER
        source PS:<SIM.CMD>FORKTB.ADA  code PS:<SIM.CMD.REL>FORKTB.REL

        package       MEMORY_INTERFACE
        source PS:<SIM.CMD>MEMIFB.ADA  code PS:<SIM.CMD.REL>MEMIFB.REL

    package       INITIATOR
    source PS:<SIM.INIT>INITB.ADA  code PS:<SIM.INIT.REL>INITB.REL
```

```
package       OBJECT_FILE
source PS:<SIM.OBJ>OBFILV.ADA

package       INSTRUCTION_GENERATOR
source PS:<SIM.OBJ>INSGNV.ADA
source PS:<SIM.OBJ>INSGNB.ADA  code PS:<SIM.OBJ.REL>INSGNB.REL

    package       SEGMENT_NAMES
    source PS:<SIM.OBJ>SEGMNB.ADA  code PS:<SIM.OBJ.REL>SEGMNB.REL

    package       REFERENCES
    source PS:<SIM.OBJ>REFERB.ADA  code PS:<SIM.OBJ.REL>REFERB.REL

    package       LABELS
    source PS:<SIM.OBJ>LABELB.ADA  code PS:<SIM.OBJ.REL>LABELB.REL

    package       LITERALS
    source PS:<SIM.OBJ>LITERB.ADA  code PS:<SIM.OBJ.REL>LITERB.REL


package       BUILD_STANDARD
source PS:<SIM.INIT>BUILDV.ADA
source PS:<SIM.INIT>BUILDB.ADA  code PS:<SIM.INIT.REL>BUILDB.REL

package       VALUE_OP
source PS:<SIM.CPU>VALOPV.ADA
source PS:<SIM.CPU>VALOPB.ADA  code PS:<SIM.CPU.REL>VALOPB.REL

package       DISCRETE_OPS
source PS:<SIM.CPU>DSCOPV.ADA
source PS:<SIM.CPU>DSCOPB.ADA  code PS:<SIM.CPU.REL>DSCOPB.REL

package       DISCRETE_TYPES
source PS:<SIM.CPU>DSCTPV.ADA
source PS:<SIM.CPU>DSCTPB.ADA  code PS:<SIM.CPU.REL>DSCTPB.REL

package       FLOAT_OPS
source PS:<SIM.CPU>FLTOPV.ADA
source PS:<SIM.CPU>FLTOPB.ADA  code PS:<SIM.CPU.REL>FLTOPB.REL

package       FLOAT_TYPES
source PS:<SIM.CPU>FLTYPV.ADA
source PS:<SIM.CPU>FLTYPB.ADA  code PS:<SIM.CPU.REL>FLTYPB.REL

package       DECLARE_OP
source PS:<SIM.CPU>DCLOPV.ADA
source PS:<SIM.CPU>DCLOPB.ADA  code PS:<SIM.CPU.REL>DCLOPB.REL

package       TYPE_DESCRIPTOR_SIZE
source PS:<SIM.CPU>TYPSZV.ADA
source PS:<SIM.CPU>TYPSZB.ADA  code PS:<SIM.CPU.REL>TYPSZB.REL

package       FAULT
source PS:<SIM.CPU>FAULTV.ADA
source PS:<SIM.CPU>FAULTB.ADA  code PS:<SIM.CPU.REL>FAULTB.REL

package       CHECK
source PS:<SIM.CPU>CHECKV.ADA
source PS:<SIM.CPU>CHECKB.ADA  code PS:<SIM.CPU.REL>CHECKB.REL
```

```
package         CHECK_COMPATIBILITY
source PS:<SIM.CPU>CHCOMV.ADA
source PS:<SIM.CPU>CHCOMB.ADA  code PS:<SIM.CPU.REL>CHCOMB.REL

package         CHECK_OBJECT
source PS:<SIM.CPU>CHKOBV.ADA
source PS:<SIM.CPU>CHKOBB.ADA  code PS:<SIM.CPU.REL>CHKOBB.REL

package         TEST_CONSTRAINT
source PS:<SIM.CPU>CNSTRV.ADA
source PS:<SIM.CPU>CNSTRB.ADA  code PS:<SIM.CPU.REL>CNSTRB.REL

package         SCOPES
source PS:<SIM.CPU>SCOPEV.ADA
source PS:<SIM.CPU>SCOPEB.ADA  code PS:<SIM.CPU.REL>SCOPEB.REL

package         COPY
source PS:<SIM.CPU>COPYV.ADA
source PS:<SIM.CPU>COPYB.ADA  code PS:<SIM.CPU.REL>COPYB.REL

package         FRAME
source PS:<SIM.CPU>FRAMEV.ADA
source PS:<SIM.CPU>FRAMEB.ADA  code PS:<SIM.CPU.REL>FRAMEB.REL

package         CHILD
source PS:<SIM.CPU>CHILDV.ADA
source PS:<SIM.CPU>CHILDB.ADA  code PS:<SIM.CPU.REL>CHILDB.REL

package         BLOCK
source PS:<SIM.CPU>BLOCKV.ADA
source PS:<SIM.CPU>BLOCKB.ADA  code PS:<SIM.CPU.REL>BLOCKB.REL

package         CODE_SEGMENTS
source PS:<SIM.CPU>CDSEGV.ADA
source PS:<SIM.CPU>CDSEGB.ADA  code PS:<SIM.CPU.REL>CDSEGB.REL

package         IMPORT_OP
source PS:<SIM.CPU>IMPOPV.ADA
source PS:<SIM.CPU>IMPOPB.ADA  code PS:<SIM.CPU.REL>IMPOPB.REL

package         ARRAY_TYPES
source PS:<SIM.STRUCTURES>ARRTYV.ADA
source PS:<SIM.STRUCTURES>ARRTYB.ADA  code PS:<SIM.STRUCTURES.REL>ARRTYB.REL

package         ARRAY_VARIABLES
source PS:<SIM.STRUCTURES>ARRVRV.ADA
source PS:<SIM.STRUCTURES>ARRVRB.ADA  code PS:<SIM.STRUCTURES.REL>ARRVRB.REL

package         VARIANT_RECORD_TYPES
source PS:<SIM.STRUCTURES>VRCTYV.ADA
source PS:<SIM.STRUCTURES>VRCTYB.ADA  code PS:<SIM.STRUCTURES.REL>VRCTYB.REL

package         VARIANT_RECORD_VARIABLES
source PS:<SIM.STRUCTURES>VRCVRV.ADA
source PS:<SIM.STRUCTURES>VRCVRB.ADA  code PS:<SIM.STRUCTURES.REL>VRCVRB.REL

package         COLLECTIONS
source PS:<SIM.STRUCTURES>CLCTNV.ADA
source PS:<SIM.STRUCTURES>CLCTNB.ADA  code PS:<SIM.STRUCTURES.REL>CLCTNB.REL
```

```
package       STRUCTURE_OP
source PS:<SIM.STRUCTURES>STROPV.ADA
source PS:<SIM.STRUCTURES>STROPB.ADA  code PS:<SIM.STRUCTURES.REL>STROPB.REL

package       VARIANT_ARRAY_OP
source PS:<SIM.STRUCTURES>VARARV.ADA
source PS:<SIM.STRUCTURES>VARARB.ADA  code PS:<SIM.STRUCTURES.REL>VARARB.REL

package       ARRAY_DECL_OP
source PS:<SIM.STRUCTURES>ARRDCV.ADA
source PS:<SIM.STRUCTURES>ARRDCB.ADA  code PS:<SIM.STRUCTURES.REL>ARRDCB.REL

package       MODULE_OPS
source PS:<SIM.TASKS>MODOPV.ADA
source PS:<SIM.TASKS>MODOPB.ADA  code PS:<SIM.TASKS.REL>MODOPB.REL

package       MODULE_TYPES
source PS:<SIM.TASKS>MDTYPV.ADA
source PS:<SIM.TASKS>MDTYPB.ADA  code PS:<SIM.TASKS.REL>MDTYPB.REL

package       TERMINAL_IO
source PS:<SIM.TASKS>TRMIOV.ADA
source PS:<SIM.TASKS>TRMIOB.ADA  code PS:<SIM.TASKS.REL>TRMIOB.REL

package       SELECT_OP
source PS:<SIM.TASKS>SELOPV.ADA
source PS:<SIM.TASKS>SELOPB.ADA  code PS:<SIM.TASKS.REL>SELOPB.REL

package       TASKING_OPS
source PS:<SIM.TASKS>TSKOPV.ADA
source PS:<SIM.TASKS>TSKOPB.ADA  code PS:<SIM.TASKS.REL>TSKOPB.REL

package       TASKING_VARIABLES
source PS:<SIM.TASKS>TSKVRV.ADA
source PS:<SIM.TASKS>TSKVRB.ADA  code PS:<SIM.TASKS.REL>TSKVRB.REL

package       DATA_TRANSLATOR
source PS:<SIM.TASKS>XLATEV.ADA
source PS:<SIM.TASKS>XLATEB.ADA  code PS:<SIM.TASKS.REL>XLATEB.REL

package       MODULE
source PS:<SIM.TASKS>MODULV.ADA
source PS:<SIM.TASKS>MODULB.ADA  code PS:<SIM.TASKS.REL>MODULB.REL

package       CONTROL_BLOCK
source PS:<SIM.TASKS>MODCNV.ADA
source PS:<SIM.TASKS>MODCNB.ADA  code PS:<SIM.TASKS.REL>MODCNB.REL

package       ELABORATION
source PS:<SIM.TASKS>ELBRTV.ADA
source PS:<SIM.TASKS>ELBRTB.ADA  code PS:<SIM.TASKS.REL>ELBRTB.REL

package       SYSTEM_MODULE_OP
source PS:<SIM.TASKS>SYMODV.ADA
source PS:<SIM.TASKS>SYMODB.ADA  code PS:<SIM.TASKS.REL>SYMODB.REL

package       RENDEZVOUS
source PS:<SIM.TASKS>RNDZVV.ADA
source PS:<SIM.TASKS>RNDZVB.ADA  code PS:<SIM.TASKS.REL>RNDZVB.REL
```

```
    package        QUEUE_OP
    not compiled

    package        CHECKS
    source PS:<SIM.TASKS>CHCKSB.ADA


package        INSTRUCTION_UNIT
source PS:<SIM.ADA>IFETCV.ADA
source PS:<SIM.ADA>IFETCB.ADA   code PS:<SIM.REL>IFETCB.REL

package        DELAY_QUEUE
source PS:<SIM.ADA>DELAYV.ADA
source PS:<SIM.ADA>DELAYB.ADA   code PS:<SIM.REL>DELAYB.REL


;BKPT GC-OVERFLOW
q;

;Q UNBOUND VARIABLE
;;

.

;DOT CONTEXT ERROR


;Q UNBOUND VARIABLE
.

;DOT CONTEXT ERROR
^C
@pop

[PHOTO:  Recording terminated Sat 27-Feb-82 3:51PM]
```

```
BBBBBBBB        AAAAAA        SSSSSSSS    TTTTTTTTTT    YY      YY
BBBBBBBB        AAAAAA        SSSSSSSS    TTTTTTTTTT    YY      YY
BB      BB    AA      AA    SS                  TT        YY      YY
BB      BB    AA      AA    SS                  TT        YY      YY
BB      BB    AA      AA    SS                  TT          YY  YY
BB      BB    AA      AA    SS                  TT          YY  YY
BBBBBBBB      AA      AA      SSSSSS          TT            YY
BBBBBBBB      AA      AA      SSSSSS          TT            YY
BB      BB    AAAAAAAAAA            SS        TT            YY
BB      BB    AAAAAAAAAA            SS        TT            YY
BB      BB    AA      AA            SS        TT            YY
BB      BB    AA      AA            SS        TT            YY
BBBBBBBB      AA      AA    SSSSSSSS          TT            YY
BBBBBBBB      AA      AA    SSSSSSSS          TT            YY
```

```
  AAAAAA      DDDDDDDD        AAAAAA                        11
  AAAAAA      DDDDDDDD        AAAAAA                        11
AA      AA    DD      DD    AA      AA                    1111
AA      AA    DD      DD    AA      AA                    1111
AA      AA    DD      DD    AA      AA                      11
AA      AA    DD      DD    AA      AA                      11
AA      AA    DD      DD    AA      AA                      11
AA      AA    DD      DD    AA      AA                      11
AAAAAAAAAA    DD      DD    AAAAAAAAAA                      11
AAAAAAAAAA    DD      DD    AAAAAAAAAA                      11
AA      AA    DD      DD    AA      AA        ....          11
AA      AA    DD      DD    AA      AA        ....          11
AA      AA    DDDDDDDD      AA      AA        ....        111111
AA      AA    DDDDDDDD      AA      AA        ....        111111
```

```
*START* Job BASTYP Req #188 for DHB    Date 27-Feb-82 15:38:25 Monitor: Rational
File PS:<SIM.DEF>BASTYP.ADA.13, created: 25-Feb-82 13:48:03
        printed: 27-Feb-82 15:38:35
Job parameters: Request created:27-Feb-82 15:37:54   Page limit:9   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

```
package BASE is

    type REAL      is digits 18;

    type DISCRETE is new INTEGER;       -- range - 2 ** 63 .. 2 ** 63 - 1;

    subtype SHORT_LITERAL      is DISCRETE range - 2 **  8 .. 2 **  8 - 1;

    subtype DISCRETE_LITERAL   is DISCRETE range - 2 ** 31 .. 2 ** 31 - 1;

    subtype CHARACTER_LITERAL is DISCRETE range   0 ..  2 ** 8 - 1;

    type ARRAY_LITERAL          is array (0 .. 3) of CHARACTER_LITERAL;

end BASE;


package body BASE is end BASE;
```

```
  OOOOOO      PPPPPPPP        SSSSSSSS    VV      VV
  OOOOOO      PPPPPPPP        SSSSSSSS    VV      VV
OO      OO    PP      PP    SS            VV      VV
OO      OO    PP      PP    SS            VV      VV
OO      OO    PP      PP    SS            VV      VV
OO      OO    PP      PP    SS            VV      VV
OO      OO    PPPPPPPP        SSSSSS      VV      VV
OO      OO    PPPPPPPP        SSSSSS      VV      VV
OO      OO    PP                    SS    VV      VV
OO      OO    PP                    SS    VV      VV
OO      OO    PP                    SS      VV  VV
OO      OO    PP                    SS      VV  VV
  OOOOOO      PP          SSSSSSSS          VV
  OOOOOO      PP          SSSSSSSS          VV



  AAAAAA      DDDDDDDD        AAAAAA                      999999
  AAAAAA      DDDDDDDD        AAAAAA                      999999
AA      AA    DD      DD    AA      AA                  99      99
AA      AA    DD      DD    AA      AA                  99      99
AA      AA    DD      DD    AA      AA                  99      99
AA      AA    DD      DD    AA      AA                  99      99
AA      AA    DD      DD    AA      AA                    99999999
AA      AA    DD      DD    AA      AA                    99999999
AAAAAAAAAA    DD      DD    AAAAAAAAAA                          99
AAAAAAAAAA    DD      DD    AAAAAAAAAA                          99
AA      AA    DD      DD    AA      AA        ....              99
AA      AA    DD      DD    AA      AA        ....              99
AA      AA    DDDDDDDD      AA      AA        ....        999999
AA      AA    DDDDDDDD      AA      AA        ....        999999


*START* Job OPSV Req #366 for DHB     Date  1-Mar-82 14:37:28 Monitor:  Rational M
File PS:<SIM.CPU>OPSV.ADA.90, created:  26-Feb-82 22:38:28
        printed:  1-Mar-82 14:37:28
Job parameters: Request created: 1-Mar-82 14:37:27   Page limit:9   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

```
with BASE,
     CODE,
     STACKS;

package OPS is  -- miscellaneous functions for code and stack manipulation

    function  MAKE_BOOLEAN  (VALUE : BOOLEAN)
                                return STACKS.CONTROL_WORD
                                            (OF_KIND => STACKS.DISCRETE_VAR);

    function  MAKE_DISCRETE (VALUE : BASE.DISCRETE) return STACKS.CONTROL_WORD
                                            (OF_KIND => STACKS.DISCRETE_VAR);

    function  MAKE_FLOAT    (VALUE : BASE.REAL) return STACKS.CONTROL_WORD
                                            (OF_KIND => STACKS.FLOAT_VAR);

    function  MAKE_DISCRETE_VAR (TYPE_LINK : STACKS.TYPE_LINK)
                                    return STACKS.CONTROL_WORD
                                            (OF_KIND => STACKS.DISCRETE_VAR);

    function  MAKE_VALUE_VAR (VALUE     : STACKS.VALUE;
                              TYPE_LINK : STACKS.TYPE_LINK)
                                                return  STACKS.CONTROL_WORD;

    function  MAKE_INDIRECT_VAR (LOCATION  : STACKS.DATA_REFERENCE;
                                 TYPE_LINK : STACKS.TYPE_LINK)
                                                return  STACKS.CONTROL_WORD;

    function  MAKE_TYPED_VAR (TYPE_LINK  : STACKS.TYPE_LINK)
                                                return STACKS.CONTROL_WORD;

    function  MAKE_VALUE_REF (VALUE_TYPE : STACKS.TYPE_LINK;
                              DATA_REF   : STACKS.DATA_REFERENCE)
                                                return STACKS.CONTROL_WORD;

    function  VALUE_TYPE_FOR_REF (REF_TYPE : STACKS.TYPE_LINK)
                                                return STACKS.TYPE_LINK;

    function  MAKE_ACCESS_REF
                        (ACCESS_VAL  : STACKS.VALUE
                                        (OF_KIND => STACKS.ACCESS_VAR);
                         ACCESS_INFO : STACKS.TYPE_WORD
                                        (OF_KIND => STACKS.ACCESS_INFO))
                return STACKS.DATA_REFERENCE;
    procedure GET_DISCRETE_BOUNDS (FOR_DISCRETE : STACKS.CONTROL_WORD;
                                   LOWER_BOUND  : out BASE.DISCRETE;
                                   UPPER_BOUND  : out BASE.DISCRETE);

    function  IN_SCALAR_BOUNDS (VALUE  : STACKS.VALUE;
                                BOUNDS : STACKS.TYPE_WORD
                                            (OF_KIND => STACKS.SCALAR_BOUNDS);
                                CLASS  : STACKS.SCALAR_VAR) return BOOLEAN;

    function  EXTRACT_TYPE (TYPED_VAR : STACKS.CONTROL_WORD)
                                                return STACKS.TYPE_LINK;

    function  DISCRETE_VAR_SIZE (TYPE_ARG : STACKS.TYPE_LINK
                                        (OF_KIND => STACKS.DISCRETE_VAR))
                                                return STACKS.VALUE_SIZE;

    function  FIXED_VAR_SIZE (OF_KIND  : STACKS.TYPED_VAR)
                              -- (OF_KIND => FIXED_SIZE_VAR | FIXED_SIZE_REF);
                                                return STACKS.VALUE_SIZE;

    function  TYPED_VAR_SIZE (TYPE_ARG : STACKS.TYPE_LINK)
                                                return STACKS.BIT_COUNT;

    function  TYPED_VAR_SIZE (TYPE_ARG : STACKS.CONTROL_WORD)
                                                return STACKS.BIT_COUNT;

    function TYPED_VAR_SIZE (TYPE_ARG : STACKS.TYPE_LINK;
                             DATA_REF : STACKS.DATA_REFERENCE)
                        return STACKS.BIT_COUNT;

    procedure MAKE_LOCAL   (EXPORT : in out STACKS.CONTROL_WORD);

    procedure MAKE_VISIBLE (IMPORT : in out STACKS.CONTROL_WORD);

    function  MAX (X, Y : INTEGER)  return INTEGER;

    function  GET_PARENT (TYPE_LINK : STACKS.TYPE_LINK) return STACKS.NAME;

    procedure POP_AND_WRITE_UTILITY (PATH : STACKS.TYPE_REFERENCE);

    function GET_COMPLETE_TYPE_INFO (PATH : STACKS.TYPE_REFERENCE)
                        return STACKS.TYPE_WORD (OF_KIND => STACKS.TYPE_INFO);

end OPS;
```