```
TTTTT  EEEEE  RRRR   M   M   III   N   N   OOO   L        OOO    GGGG  Y   Y
  T    E      R   R  MM MM    I    N   N  O   O  L       O   O  G      Y   Y
  T    E      R   R  M M M    I    NN  N  O   O  L       O   O  G       Y Y
  T    EEEE   RRRR   M   M    I    N N N  O   O  L       O   O  G        Y
  T    E      R R    M   M    I    N  NN  O   O  L       O   O  G GGG    Y
  T    E      R  R   M   M    I    N   N  O   O  L       O   O  G   G    Y
  T    EEEEE  R   R  M   M   III   N   N   OOO   LLLLL    OOO    GGG     Y


DDDD    OOO    CCCC            1
D   D  O   O  C               11
D   D  O   O  C                1
D   D  O   O  C                1
D   D  O   O  C                1
D   D  O   O  C          ..    1
DDDD    OOO    CCCC     ..   111


*START* Job TERMIN Req #806 for EGB    Date 25-Aug-82 12:09:31 Monitor: Rational
File RM:<SIM.DOC.PATENT>TERMINOLOGY.DOC.1, created: 16-Jun-82 12:47:21
        printed: 25-Aug-82 12:09:32
Job parameters: Request created:25-Aug-82 12:08:11   Page limit:18   Forms:NORMAL
File parameters: Copy: 1 of 1   Spacing:SINGLE   File format:ASCII   Print mode:AS
```

In an attempt to clarify and standardize the usage of certain frequent, often overloaded, often abused terms, the following suggested uses and definitions are proposed. It is hoped that by using terminology in a consistent manner in all written and oral discussions of the R1000 architecture and environment, we may minimize the confusion both among ourselves and among our customers.

## 1. OBJECT

An OBJECT is an entity together with some means for identification and for interpretation of the entity. The means of interpretation may provide attributes of the entity, such as value, text representation, etc., may provide transformations into other objects, or may provide the mechanism for interpreting other objects.

In the architecture, there are three (N?) principal categories of objects, instruction objects, control stack objects, and type stack objects, each such object consisting of a specified number of bits in memory. Instruction objects are identified by their presence in program memory and may be decoded to cause particular operations to occur in the machine. Control stack objects are identified by their presence in control memory and contain a further means for their interpretation in the form of unique patterns of bits in a particular position (a tag). Type stack objects are identified by their presence in type memory and are interpreted only (generally) by a means (type link) contained in associated control stack objects.

The notion of instructions as objects in the R1000 is of secondary importance, so it is suggested that explicit qualification as 'instruction object' be used whenever it is intended. Furthermore, since type stack objects are not manipulated directly as a result of the execution of instructions and are interpreted only (generally) by means of control stack objects, the common usage of 'object' should be in reference to control stack objects, and the term 'type descriptor' should be used to describe type stack objects.

In ADA, an object is introduced in a declaration which includes an identifier for the object and a type indication for the object. The type defines the means for interpretation of the object within the semantics of the language.

In the programming environment, the use of the term object generally means an object as defined in ADA, however certain notions such as a 'program' are not objects in ADA, but may be manipulated in the environment. The means for the interpretation of environment objects which do not lie within the semantics of ADA may be either explicit or implicit.

## 2. CLASS

The term CLASS refers to membership in a relatively small set of categories of data which are recognized and manipulated by the R1000. The tag of a control stack object indicates membership in one or more classes. The class of an object determines its representation in memory of the machine and, hence, specifies the primitive means for interpreting the object. Many instruction objects are represented in such a way that the specification of a class of applicability is extracted during decoding. From the perspective of the (architecture) (microcode), conversion among objects of different classes is not possible.

correct usage:
"plus is applicable to discrete class objects"
"if the operand is of array class"

3. KIND

The term KIND has (at least) two accepted usages in the architecture: to distinguish the distinct varieties of memory (ie. Program, Control, Type, Data, Import, and Queue); and to distinguish a particular representation for a word in memory. In the latter usage, KIND is similar to CLASS except that it does not necessarily refer to (explicit or implicit) objects, but may be used more abstractly to indicate the means of interpretation of the all objects of a category.

correct usage:
"generates a reference to memory of program kind"
"if the kind of control word is a discrete var"

4. TYPE

The term TYPE should always be used in a context relating to the declaration of (keyword) types in a source program, and hence, as described in the LRM should characterize "a set of values and a set of operations applicable to those values". One may speak of the type of an object on a control stack, because such objects include a reference to a type descriptor which contains sufficient information to specify the values for the type. The operations for a type are specified implicitly by the class of the type and explicitly by source language subprograms within the scope of declaration of the type.

correct usage:
"certain conversions among types are permitted by the machine"

incorrect usage:
"the tag indicates the type of the object"

5. REFERENCE

REFERENCE may be used to indicate an R1000 address either in the form of lex level/delta in an instruction (usually termed an "object reference", since given any frame, it addresses either some object, or is illegal) or in the form of a machine logical address (consisting of memory kind, name, offset, eg. "control reference").

6. SPACE

The term SPACE may be used either to describe the total logically addressed memory of the machine or that portion of the total logically addressed memory which may be reference using a specific logical name.

7. SEGMENT

SEGMENT should be used to indicate a portion of the full logical address space of the machine which be referenced using a specific memory kind and specific name. The term STACK is synonymous with SEGMENT in the case of control, type, and data memory kinds.

correct usage:
"code segment 4732"
"control stack 16"

8. MODULE

The term MODULE is to be used to indicate the logical group of segments which are associated with the value of a task, package, or collection object, and which share the same name. A given module will always have a control segment physically allocated, but may or may not have type, data, or queue segments allocated.

9. POINTER

The usage of the term POINTER should be restricted to access types and access variables.

Incorrect usage:
"the type link contains a pointer to the type stack"