bcc	title	prefix/class-number.revision
	ALLOCATION AND PROGRAM FORMAT IN SPL	SPLAPF/W-25

checked	authors	approval date / revision date
L. Peter Deutsch	Butler W. Lampson	8/15/69 /
checked		classification: Working Paper
approved: Mel	Butler W. Lampson	distribution: Company Private / pages: 6

ABSTRACT and CONTENTS

The format of SPL programs is specified. SPL facilities are defined which permit the location of code, arrays, scalars, common blocks and local environments in core to be specified. The default allocation strategies are described.

This document obsoletes SPLAL/W-16. It will be absorbed into a revision of the SPL manual soon.

Program Format

An SPL program is organized into blocks. Each block starts with a block:statement and ends with an end:statement.

block:statement = ("COMMON"/"PROGRAM") identifier

end:statement = "END"

The end:statement can be omitted, in which case it will be supplied automatically.

The old distinction between FUNCTION and ENTRY is removed. Either may appear anywhere in a program block and simply serves to specify a function entry point without carrying any implications about scopes. Everything between a PROGRAM and the matching END has the same defined variables and the same local environment.

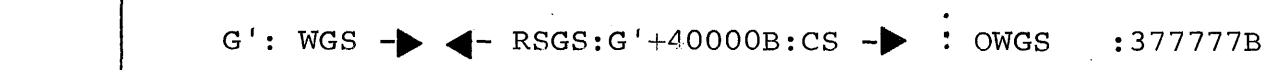
Within a block all include:statements must precede everything else. An origin:statement or fixed:statement, if present, must precede all other statements aside from include:statements. Declarations of variables must precede their use. There are no other restrictions on the order of statements.

A common block must lexically precede any block which includes it. Otherwise there are no restrictions on the ordering of blocks, except for ORIGINED common blocks (see below)

Layout of Core

The arrangement of memory relative to G is designed to group read-only things together and on separate pages from writeable things, so that the former can be protected by the hardware from modification. Later improvements will permit small programs to be packed together better.

Space is allocated in four main regions



WGS: Writeable global storage, starting at G. This area is allocated by a general storage allocator in the compiler in a piecemeal fashion; no attempt is made to keep related things together. Here are put all the writeable variables which appear in common blocks, together with fixed local environments. Some of the first 128 words may also be used for field and array descriptors, at the discretion of the compiler, except in the monitor ring, where this will never be done (unless forced by equivalences). The first few words, of course, are used for objects whose location is fixed by the hardware, like the stack descriptor.

The allocation strategy for this area may be modified by ORIGIN statements; see below.

Collision of this area with RSGS is a fatal error in the initial implementation. Later versions will cause it to overflow into

OWGS: Overflow writeable global storage, which is handled in the same way.

The stack is allocated space at the end of this area.

RSGS: Read-only scalar global storage. Here are put the constant scalars (e.g. array descriptors and initialized scalars) from common blocks, as well as function descriptors. This area is allocated by another incarnation of the general storage allocation used for WGS and on the same piecemeal basis.

CS: Code storage. Space here is allocated by block. All the code and constants generated by one program block, or all the non-scalar constants (strings, arrays and dope) generated by one common block, are collected together and allocated contiguously in that region. Transfer vectors also appear here. If block A precedes block B lexically (in the source), then the CS for A will precede the CS for B.

Origins

The origin:statement permits (most of the) storage of a block to be allocated at a fixed place.

origin:statement = "ORIGIN"[expr]

The expression, whose value is called the origin of the block, must evaluate to an integer at compile-time. The statement must appear in the block after any include:statement and before anything else.

If the block is a program block or a common block with no writeable variables declared, the origin tells where to start its space in CS. If the preceding block's space in CS extends past the specified origin, an error is recorded and the statement is ignored. This implies that origined blocks must appear in order of increasing origins. Note that the scalar storage of a common block is allocated in RSGS and is not affected by origin:statements.

If the block is a common block with writeable storage, then the origin tells when to start this storage. Two restrictions apply

1) The block must have no requirements for CS.
2) All blocks with origined WGS must appear before any non-origined blocks which require WGS, so that the space taken by origined blocks can be properly removed from the control of the storage allocator

All the WGS for an origined block is allocated together. A subsequent block may omit the expr from its origin:statement, in which case its WGS is allocated immediately following that of the preceding block.

Fixed Environments

A fixed local environment is specifed by a fixed:statement.

"FIXED" ["," "ORIGIN" expr]

The origin clause tells where to put the environment. The programmer is responsible for the security of the area he chooses,

which is not checked by the compiler. In the absence of the origin, the compiler will allocate the storage in WGS at its discretion.

The word FIXED may not appear in a function declaration.

Equivalence

An equivalence (SPL/M-1.1, p. 6) can be used to fix the location of a scalar or an array descriptor by writing an integer-valued expression for the object of the equivalence. Thus

```
DECLARE A = 40B, ARRAY B[30] = 41B;
```

allocates A at 40 and the descriptor for the array B at 41 and 42. The array itself is allocated according to the default rules. Restriction: the value of the equivalence must be in the range [G', G' +37777B]. An equivalence overrides all other methods of storage allocation. If a variable V has been equivalenced to a constant, or is declared in a common block, then @V is a constant whose value is the address assigned to V.

Fixed Fields

Descriptors for part-word fields are normally allocated in the first 128 words of the global environment by the compiler if there is room. This allocation can be suppressed and the field allocated in the function or common block like any other constant by prefixing FIXED to [SIGNED] FIELD in the declaration.

Arrays

The syntax of <dimension> (SPL/M-1.1, p.3) is extended as follows:

dimension = '[' expr $(','expr) [':'[expr][','expr]] ']'

The last optional expression tells where to put the first word of the array. Also, the expressions in the list of subscripts bounds may be null in a formal parameter; in this case the list serves simply to establish the dimensionality of the array.