C. J. Saltzer

DRAFT: 2/24/69
J. D. Mills

PL/I LINKAGE GENERATOR

The linkage generator requires four segments, each of which is pointed to by a pointer in external static storage. These are:

	Segment	Name of Pointer
1.	Link segment	link_ptr
2.	Scratch segment for relocation info for link segment	link_reloc_ptr
3.	Def segment	def_ptr
4.	Scratch segment for relocation info for def segment	def_reloc_ptr

The scratch segments are word for word analogues of the link and def segments. Each word of the scratch segment is used to contain two relocation codes right justified in an 18 bit field. These codes correspond to the links and defs generated on a parallel, half-word by half-word basis. In pl1 these are later packed into the standard form in the symbol segment.

The def segment is, in pl1, a scratch segment which is later appended to the text segment. It may, of course, be the text segment itself.

The four pointers are assumed to be always pointing at the next locations in the segments where information is to be compiled. As links are generated the pointers are advanced.

The pl1 linkage generator is driven from the pl1 internal representation and as such is not generally useful. However, it uses a primitive which can be used outside of pl1 to generate a single link.

Compile_link is a procedure called as a function. It generates a Multics link, link definition, and relocation bits for same.

The calling sequence is:

```
link_loc=compile_link(string,size,trap,type);
```

where the declarations are:

```
dcl  link_loc        fixed bin(18),
     compile_link    external entry returns (fixed bin(18)),
     string          char(n),
     size            fixed bin(17),
     trap            bit(18),
     type            fixed bin(17);
```

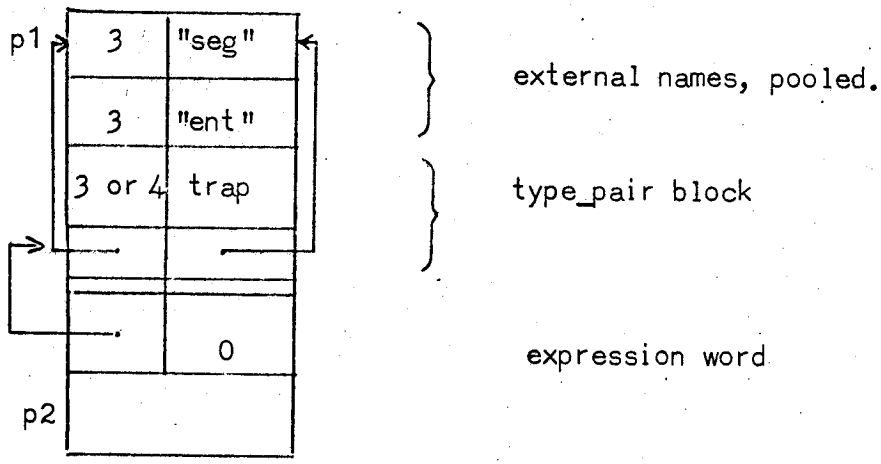
The value of compile_link is the word offset from the beginning of the linkage section of the link generated.

The parameter "string" is the external name to which a link is to be generated. It may have an imbedded dollar sign ($) to separate the segment name from the entry, e.g. a$b indicates entry b in segment a. If the dollar sign is the last character of the string, then a link is generated to location 0 of the segment. If there is no dollar sign, then the interpretation of the string depends on the value of the last parameter, type. If type is zero then string is both the segment name and the entry name. Otherwise, the segment is "stat_" and the location is string. Actually, "stat_" is taken from an external static variable and can be changed (by the RENAME option in pl1).

The third parameter, trap, is normally "0"b unless the link requires a trap-before-link. In the latter case the value of trap is a bit string representation of the offset from the origin of the definitions where the trap pointer word is.

Given this information compile_link generates the following:

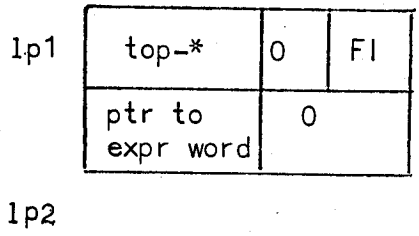
In the definition section.



p1 is the value of def_ptr upon entry.

p2 is the value of def_ptr upon exit.

In the linkage section.



lp1 is the value of link_ptr upon entry.

lp2 is the value of link_ptr upon exit.

The relocation codes are set in the corresponding relocation segments and the relocation ptrs are advanced.

The only pooling that is done is for the segment and entry names. The reason for this is that the links will be unique as generated by pl1 unless the user declares an external variable (entry or static) in more than one block.

This is not likely. However, names can be duplicated and should be shared among links for example, links to a, a$b, and a$c have four requirements for the string "a". One representation serves all four.

The pooling and building of external names is done by a procedure "name_assign" whose calling sequence is:

```
name_loc = name_assign(name,length);
```

where:

```
dcl  name_loc  bit(18),
     name_assign external entry returns (bit(18)),
     name char(n),
     length fixed bin(17);
```

It is required that name_assign be initialized by a call:

```
call  name_assign$initialize_name_assign;
```