INTERNAL USE ONLY

INTERNATIONAL COMPUTERS AND TABULATORS LIMITED

Scientific Programming Department
I.C.T. 1900 Series

Superceded by Note 10 and 10A

FORTRAN NOTE 3
27.3.65.

A Proposed Method of Describing and Implementing Program Overlays for the FØRTRAN IV (ICT) Compiler

General

If there exists a suitable "backing store" for the object program there will have to exist a facility through which more than one segment of a program may be compiled to occupy the same main store locations, being read from the backing store to the main store when necessary.

The notes which follow give a proposal for program overlay facilities within the framework of FØRTRAN IV (ICT) without adding to the FØRTRAN language.

For the purpose of this document the following terms are now defined.

1. Overlay Segment - A program segment which is obeyed from the same area of main store as another segment.

2. Overlay Area - an area of main store from which overlay segments may be obeyed. At any time an overlay area contains only one complete overlay segment. An overlay segment of an overlay area has its first main store program address at the beginning of that area.

3. Overlay Area Number - A small integer which 'labels' a particular overlay area.

FØRTRAN Overlays

A FØRTRAN IV (ICT) program may consist of a main section of program with no overlay segments; or it may contain a main section of program which may not be overlaid, and two or more overlay segments which may be obeyed from one or more overlay areas. No one segment may be obeyed from more than one overlay area. At any time an overlay area contains only one overlay segment.

For FØRTRAN IV (ICT) it is proposed that the only segments which may be overlay segments - and are therefore the only segments which may themselves be overlaid - are segments which are introduced by a statement of the form.

SUBRØUTINE Sub

where "Sub" is the name of a segment. "Sub" may <u>not</u> have an argument list.

All overlay segments which are to be obeyed from a particular overlay area are described to the FØRTRAN IV (ICT) compiler by a statement in the "Program Description" of the form.

ØVERLAY n, Sub1 = Sub2 = ... = Sub k

where "n" is the overlay area number, and the "Sub i" are the names of the segments which will be obeyed from that overlay area.

(For details of the "Program Description" read "A Proposed Method of Describing a FØRTRAN Program to the FØRTRAN IV (ICT) Compiler" - FORTRAN NOTE 2)

Program Communication between Overlay Segments

Within FØRTRAN segments, at the source program level, there is no indication as to whether a given segment of the approporate type is an overlay segment. Thus, regardless of the "status" of a segment the programmer will still write

CALL Sub

to enter the segment "Sub"; and from within that segment, the statement

RETURN

will cause the original segment to be re-entered at the statement following the CALL statement.

The actual communication between any segment and a possible overlay segment which is being called proceeds via a subroutine (%FOVL).

In the "calling" routine (call it A), the statement CALL B causes the following sequence to be compiled.

```
CALL 1 %FØVL
       M1
       M2
BRN    Start of B
```

where M_1 is the address of a "Constants" area which contains the name A. Similarly M_2 is the address of the name "B". In both cases the first cases the first character of the name is the number of words in the name (including that first character). Thus if the name of "A" is JACK, the "name" in the Constants area will be "2 JACK sp sp sp".

From within B, the statement RETURN causes the following instruction to be compiled

BRN %FØVL + 1

Note that these sequences are only compiled if B is of a form which might allow it to be used as an overlay segment.

The Operation of %FØVL

This routine ensures that the correct program segment is in store when a call for that segment is made. It also keeps track of which segments have been "calling" possible overlay segments so that on execution of a RETURN statement it can ensure that the original calling segment is in store.

To perform these operations, %FØVL makes use of two (possibly three) lists. The compiler generates a list of Overlay areas and overlay addresses along with the names of the overlay segments. %FØVL generates a pushdown list (stack) of the names of segments which have used it to enter possible overlay segments. This stack also contains the links for these segments. For certain types of Backing Store there may have to be a third list (generated by the program loader) of where on the Backing Store the overlay segments have been stored. Even for magnetic tape storage the tape deck must be known to %FØVL.

%FØVL has two entries - at the first word to enter a possible overlay segment, and at the second word to return from such a segment. When %FØVL is entered at the first word, the word following the call to %FØVL contains the address of an area holding the name of the calling routine. The second word following the call contains the address of the name of the called routine. In each case, the first character of the name gives the number of words in the name (including that first character). The third word following the call contains a branch to the desired subroutine.

Whether or not the called routine is actually an overlay segment, the name of the calling segment and the link (X1) is stored in the "Stack". %FØVL then compares the name indicated by the second word following the call to its overlay list. If the name is not in the list or if the name is in the list but the segment is already in the main store, the third word following the call is obeyed as an instruction.

If the name is in the Overlay List and the segment is not in the main store, %FØVL causes the segment to be read into store at the address specified by the starting point for the correct overlay area, after which the third word following the call for that segment is obeyed as an instruction. Note that in case the calling segment is in the same overlay area as the called segment the third word following the call must be obeyed from within %FØVL's own working store.

When %FØVL is entered at the second word (to return from a subroutine), the name at the top of the stack is compared with the items in the overlay list. If the required segment is not in store it is read down from the Backing store. The Link (also in the stack) is extracted and both the link and the segment name are removed from the stack. The link is loaded into X1 and the instruction "EXIT 1 3" is obeyed.

Note that any operations involving the reading of an overlay segment into store requires that the overlay list for the particular overlay area be updated to indicate the correct segment in store.

%FØVL Lists

1. Overlay List

   This list is generated by the compiler and occupies the Common area %FØVT.

The List contains two types of item.

Type 1. Overlay Area Item. This consists of two parts.
part 1 (one word) - The main store address of the start of the Overlay area.
part 2 (one word) - The address of the List item of type 2 (Overlay Segment Items) for the segment currently in the overlay area.

Type 2 Overlay Segment Item. This consists of three parts.
part 1 (one word) - The address of the previous item of type 2 in the list. If there is no "previous item", this word is zero.
part 2 (one word) - The address of the list item of type 1 for the particular overlay area.
part 3 - The name of the overlay segment. The first character of the name gives the length of the name in words (e.g. for a name JACK, this part of the list item will be 2JACK sp sp sp)

These will be as many items of type 1 as there are overlay areas

There will be as many items of type 2 as there are overlay segments.

The first word of the Common area %FØVT gives the address of the last item of type 2 put into the list.

2. %FØVL Pushdown List (Stack)

This consists of a word pointing to the last item put in the stack.

An item consists of two parts.

Part 1 (one word) the link for the segment.
Part 2 The name of the segment. The first character of the name gives the length of the name in words.

The Compiler - Consolidator - Loader Communication for Overlays

When the Compiler processes an ØVERLAY statement within a Program Description it generates a standard blank cue for the overlay routine (%FØVL) but also generates on "Overlay Area" cue for each overlay area number and an "Overlay Segment" cue for each segment name mentioned in the ØVERLAY statement. This overlay segment cue is of the same family as "unsatisfied" (blank) cues but it has attributes which indicate which overlay area the segment will finally occupy.

When the Compiler processes any segment (including overlay segments) it generates a "Program" cue which gives the length of that segment.

When the Consolidator finds a program cue when it already has an overlay cue for that segment, the overlay cue is marked as "satisfied" and the value in the program cue is compared with the value of the associated overlay area cue. The greater of the two values becomes the new value for the overlay area cue. The original program cue is then forgotten. If the Consolidator finds a program cue but does not have an overlay cue for that segment, the Consolidator takes the program cue itself as completely describing the program segment.

In the Consolidated Leader, a cue for a program segment is either a "program" cue or an "overlay" cue. A program cue will contain the starting address of the program segment. An overlay cue will give the overlay area number for the segment. If a program segment is defined in an overlay cue, the Consolidator will also have generated an "overlay area" cue for the particular overlay area number. This overlay area cue will contain the starting address of the overlay area.

The Loader now has enough information from the Consolidated Leader to load each segment of the object program.

Program Segments represented by Program Cues in the Consolidated Leader are loaded correctly. Segments represented by Overlay Cues are loaded into a "Temporary Overlay Area" of the loader and then written to the Backing Store in binary form. Depanding on the Backing Store the loader may have to generate a list of Backing Store addresses, the starting addresses in the Backing store for the various segments. This list will have to be in an area accessible to FØVL (or at least an index word to the list will have to be accessible.)

V. K. Taylor