Published: May 8, 1967

Identification

The Begin, Procedure, and Entry Statements

B. P. Goldberg

Introduction

This MSPM section is written in the format established for G.E. publications, since it will also be used as part of the G.E. 645 EPL Implementation Manual. According to present plans, this manual will follow the outline for the BN section of the MSPM manual. Section BN.6.04 begins at Paragraph IV of the chapter on Local Strategies.

IV. BEGIN, PROCEDURE, AND ENTRY STATEMENTS

A. Purpose

This section describes the Pass 1 and Pass 2 coding for internal and external procedures, entries to these two types of procedures, and the begin statement.

B. Simulation of Location Counters

An EPL object program consists of several independent sections; for example, the prologue, the epilogue, the main program, etc. Coding from several of these sections is illustrated in the examples in this writeup. Note that each section consists of disconnected pieces of coding. Since

EPLBSA only provides one location counter, EPL must generate transfer instructions to connect the pieces, e.g., tra p1.5. This method simulates the use of multiple location counters, but it is not as efficient.

In the future, EPLBSA will provide multiple location counters, and EPL will generate use pseudo-instructions at the beginning of each disjoint piece. This will enable EPLBSA to link the pieces of each section more efficiently. (See BN.6.01 for a discussion of the usage and implementation of multiple location counters.)

C. Procedure Statement

1. External Procedures

Pass 1 generates the following code for the initial procedure statement in a program:

```
dfxx  name,xx0024,bits,0,xxx,ext,entr,0,0,0
begin
entry name,xx0024,bits,0,xxx,ext,entr,0,0,0
```

The dfxx macro is the symbol definition for the procedure name. The begin macro signifies the beginning of the first block of the procedure. The entry macro marks the procedure name as an entry point to the procedure.

In the dfxx macro, the xx is the data type: fixed (fx), floating (fl), bit string (bs), character string (cs) or pointer (pt). EPL creates a

dffl macro, unless the procedure is a function whose value is not floating point.

The parameters in the df<u>xx</u> and entry macros have identical functions. <u>Name</u> denotes the source program name for the procedure. <u>Alias</u> is xx0024, since EPL built-in functions are assigned to the first 23 internal names in the compiler. The third parameter is the number of bits in the procedure value. If this is not specified by the programmer, it is the default precision for the data type.

Pass 2 then produces the following coding:

```
          link      xx0024,<procedure name>|[entry name]
p1.0:      tra      p1.1
s1.0:      tra      .y1
          name      procedure name
         entry      entry name
entry name:  eax7   .as1
          tsx0      .sv
          tsx1      p1.0
.y1:      null
          "
```

Here lp|xx0024 is defined as the location of the linkage to <u>entry name</u> in the linkage segment. The symbol .as1 denotes the number of words in the stack frame. All pointers, registers, etc. are saved by the .sv routine described in BN.5.02. Execution of the prologue is initiated by the tsx1 p1.0 instruction. The second statement in the program is generated following the null statement .y1.

2. Internal Procedures

Pass 1 generates the following macros for internal procedures:

```
dfxx  name,alias,bits,0,xxx,int,entr,0,level,0
begin
entry name,alias,bits,0,xxx,int,entr,0,level,0
```

The functions of these macros are the same as those described for external procedures.

Pass 2 then generates the following code for a procedure at level 1 (assuming the current block in the prologue is p5):

```
p5.0:  tsx0   .cp
       tra    p5.1
s5.0:  tra    .yx
alias: eax7   .as5
       tsx0   .sv
       tsx1   p5.0
 .yx:  null
       "
```

In this case, the procedure name is not located in the linkage section, since the procedure will never be invoked from the outside. The remaining code is the same as that for an external procedure, except for the transfer to .cp. The .cp routine copies the display (stack level) of the internal procedure's statically embracing block; i.e., if the internal procedure is at level 1, the .cp routine copies the display for the external procedure. (See BN.5.02 for an explanation of this concept.)

The following coding is generated for the .cp routine:

```
.cp:   eapbp   sp|26,*    Recovers arg list pointer
       ldx4    ap|0       Loads 0 into index register 4
       eapbp   ap|2,4*    Picks up copy of stack pointer, for procedure's
                             statically embracing block
       stpbp   sp|.ds     Stores copied stack pointer in the display
       lbrlp   bp|4  }    Redefines the linkage pointer relative to the
                             copied stack pointer
       lbrlb   bp|5  }
       stb     sp|0       Stores redefined values of the bases
       tra     0,0        Returns to the prologue
```

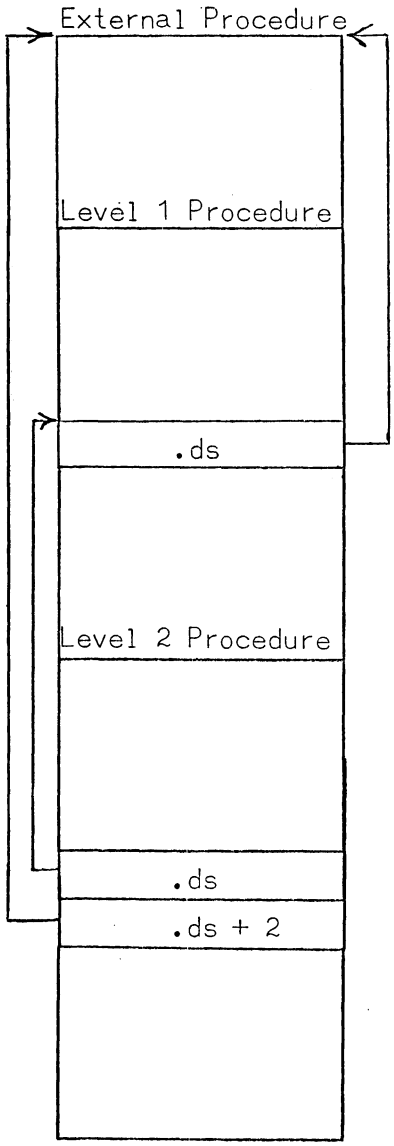

Figure 1. Creation of Display for Level 2 Procedure

Pass 2 produces the following instructions below the tsx0 .cp for a procedure at level 2:

```
        eapbp   sp|.ds,*
        ldaq    bp|.ds
        staq    sp|.ds+2
```

The first instruction picks up the stack pointer for the level 1 procedure. (This duplicates the .cp instruction eapbp ap|2,4*.) The two-word display from the level 1 procedure is then stored immediately below the two words at .ds at level 2. This is illustrated in Figure 1.

For all procedures below level 2, the tsx0 .cp instruction is followed by the coding shown below:

```
        eax4    2
        tsx0    .cp1
         .
         .
cp1:    eapbp   sp|.ds,*
        ldaq    bp|.ds,4
        staq    sp|.ds+2,4
        sblx4   2,du
        tpl     *-3
        tra     0,0
```

Assume the procedure is at level 3. In this case, the first instruction in the .cp1 routine picks up the stack pointer for the level 2 procedure. The four-word display from the level 2 procedure is then stored immediately below the two words at .ds at level 3.

D. Begin Statement

The begin statement is interpreted as an internal procedure preceded by a dummy call to the procedure. Pass 1 generates the following macros for

this statement:

```
dfxx     ,alias,bits,0,xxx,int,entr,0,level,0
callxx   alias,bits,0,xxx,int,entr,0,level,0
begin
entry    ,alias,bits,0,xxx,int,entr,0,level,0
```

If the begin statement has a label, the above coding is preceded by a dclb macro.

The following Pass 2 coding is generated for a begin statement at level 1:

```
         tra     s1.2
s1.2:    eapbp   sp|0
         stpbp   sp|.a1+2
         ldaq    =v18/0,18/2,36/0
         staq    sp|.a1
         call    alias (sp|.a1)
         tra     s1.3
p2.0:    tsx0    .cp
         tra     p2.1
s2.0:    tra     .y3
alias:   eax7    .as2
         tsx0    .sv
         tsx1    p2.0
.y3:     null
```

Here the first six instructions set up the call. The stack pointer is transferred in place of an argument. The remaining coding constitutes the procedure and thus is identical with that for an internal procedure. The changes in this coding for lower level statements are also the same as those shown for internal procedures (See Paragraph C.2.).

E. Entry Statement

1. External Entries

Pass 1 processes the external entry statement as follows:

```
dfxx    name,alias,bits,0,xxx,ext,entr,0,0,0
entry   name,alias,bits,0,xxx,ext,entr,0,0,0
```

Note that this coding is the same as that for an external procedure, except that, in this case, there is no begin macro.

Pass 2 translates this code as follows (assuming this is the second statement):

```
            tra     s1.2
            link    alias,<name>|[entry name]
s1.2:       tra     .yx
            entry   entry name
entry name: eax7    .as1
            tsx0    .sv
            tsx1    p1.0
.yx:        null
             "
```

This coding is described in Paragraph C.1.

2. Internal Entries

Pass 1 produces the following macros for an internal entry:

```
dfxx    name,alias,bits,0,xxx,int,entr,0,level,0
entry   name,alias,bits,0,xxx,int,entr,0,level,0
```

The following example illustrates the coding produced by Pass 2:

Example:

```
            tra         s5.2
s5.2:       tra         .y8
xx0056:     eax7        .as5
            tsx0        .sv
            tsx1        p5.0
.y8:        null
             "
```

Here xx0056 is the alias for the entry name. This name is not located in the linkage section, since the entry will never be invoked from the outside. The remaining code is the same as that for an external entry.

F. End

All procedure blocks and begin blocks are terminated by bend macros. The following example illustrates the Pass 2 translation of the bend macro:

Example:

```
          tra     e6.0
e6.0:     tra     .rt
p6.3:     tra     0,1
          equ     .a6,50
          equ     .u6,56
          equ     .as6,64
          equ     .w6,42
          equ     .m6,48
```

The equ pseudo-instructions define locations in the current stack frame. The symbols used are defined in BN.3.02.