Published: 05/19/67

Identification

Do Statement

B. P. Goldberg

Introduction

The basic pattern of the do statement coding is the same for all types of expressions. The expressions used in the examples below were chosen for simplicity. (See Section BN.6.03 for a description of the evaluation of other types of expressions.) A simple assignment statement (a = a;) is used as the program element in each example. Assume that a is a single-precision floating-point number and is defined as the first variable in the program; i.e., xx0026.

Use of a Variable or a Signed Constant in the "by expression"

Source Code:
```
do i = j to k by 1 :
    a = a;
    end;
```

Pass 1 generates the following code:

```
                Macro                                                    Comment
ldfx    alias1,bits1,0,xxx,int,auto,0,level,0                    Loads j
stfx    alias2,bits2,0,xxx,int,auto,0,level,0                    Stores j in i
golb    alias3,144,0,xxx,con,xxxx,0,level,0                      Transfers to test
dclb    ,alias4,144,0,xxx,con,xxxx,0,level,0
ldfx    alias2,bits2,0,xxx,int,auto,0,level,0                    Loads i
adfx    bits2,0,alias5,bits5,0,xxx,int,auto,0,level,0            Adds 1 to i
fxfx    bitsn,0,17,0                                             Changes precision of result
stfx    alias2,bits2,0,xxx,int,auto,0,level,0                    Stores result in i
dclb    ,alias3,144,0,xxx,con,xxxx,0,level,0
ldfx    alias2,bits2,0,xxx,int,auto,0,level,0                    Loads i
```

```
sbfx    bits2,0,alias6,bits6,0,xxx,int,auto,0,level,0   Subtracts k
dofx    alias5,bits5,0,xxx,int,auto,0,level,0           Negates difference if 1 ≥ 0
outlb   alias7,144,0,xxx,con,xxxx,0,level,0             Transfers if difference < 0;
                                                          i.e., satisfied do
ldfl    xx0026,27,0,xxx,int,auto,0,level,0    }         Program element
stfl    xx0026,27,0,xxx,int,auto,0,level,0    }

gogolb  alias4,144,0,xxx,con,xxxx,0,level,0             Transfers to incrementation
                                                          coding
dclb    ,alias7,144,0,xxx,con,xxxx,0,level,0
```

Pass 2 translates the macros shown above as follows:

```
        lda     sp|alias1       }
        sta     sp|alias2       }  Initialization of i
        tra     alias3          }
alias4: null    "
        lda     sp|alias2       }
        ada     sp|alias5       }  Incrementing of i
        sta     sp|alias2       }
alias3: null    "
        lda     sp|alias2       }
        sba     sp|alias6       }
        szn     sp|alias5       }
        tmi     *+2             }  Testing of i
        neg                     }
        als     0               }
        tmi     alias7          }
"
        fld     sp|xx0026       }
        fst     sp|xx0026       }  Program element
"
        tra     alias4              Next iteration
alias7: null    "
```

Use of an Unsigned Constant in the "by expression"

```
Source Code:    do i = j to k;
                a = a;
                end;
```

In this example, the expression "by 1" is understood. The initialization and iteration coding is similar to that for Case 1. However, the testing

is different. Here Pass 1 produces the following macros to determine whether the do is satisfied and to transfer out of the loop:

Macros	Comment
ldfx alias1,bits1,0,xxx,int,auto,0,level,0	Loads i
iflefx bits1,0,alias2,bits2,0,xxx,int,auto,0,level,0	If i ≤ limit, then continues;
golb alias3,144,0,xxx,con,xxxx,0,level,0	otherwise, do is satisfied.

Pass 2 then produces the following code:

Instruction		Comment
lda	sp|alias1	
cmpa	sp|alias2	
tze	*+2	Transfers to program element
tpl	alias3	Transfers out of loop

While Clause

Source Code: do while (i = j);

Here Pass 1 produces the following macros to determine whether the do is satisfied and to transfer out of the loop:

Macros	Comment
ldfx alias1,bits1,0,xxx,int,auto,0,level,0	Loads i
ifeqfx bits1,0,alias2,bits2,0,xxx,int,auto,0,level,0	If i = j, then continues;
golb alias3,144,0,xxx,con,xxxx,0,level,0	otherwise do is satisfied.

Pass 2 translates these macros as follows:

```
lda     sp|alias1
cmpa    sp|alias2
tnz     sp|alias3
```

Do Statement with a Simple Specification

```
Source Code :    do i = 1;
                 a = a;
                 end;
```

Pass 1 first produces macros to set i equal to 1. It then generates the following code:

	Macro	Commment
golb	alias1,144,0,xxx,con,xxxx,0,level,0	Goes to program element
dclb	,alias2,144,0,xxx,con,xxxx,0,level,0	
golb	alias3,144,0,xxx,con,xxxx,0,level,0	Dummy test
dclb	,alias1,144,0,xxx,con,xxxx,0,level,0	
ldfl	xx0026,27,0,xxx,int,auto,0,level,0	Program element
stfl	xx0026,27,0,xxx,int,auto,0,level,0	
gogolb	alias2,144,0,xxx,con,xxxx,0,level,0	Goes to dummy test
dclb	,alias3,144,0,xxx,con,xxxx,0,level,0	

Here the do is satisfied immediately after the computation is performed.

Therefore, no testing is necessary.

Pass 2 then produces the following code:

	Instruction		Macro
	tra	alias1	golb
alias2:	null	"	dclb
	tra	alias3	golb
alias1:	null	"	dclb
"	fld	sp|xx0026	
"	fst	sp|xx0026	program element
	tra	alias2	gogolb
alias3:	null	"	dclb

Multiple Specifications

Source Code:
```
do i = 4,9,16,25;
a = a;
end;
```

Pass 1 first creates macros to set i = 4. It then generates the following code:

Macro	Comment
dflb ,alias1,144,0,xxx,0,int,auto,0,level,0	Defines temporary location
ldlb alias2,144,0,xxx,con,xxxx,0,level,0	Sets up transfer from program element
stlb alias1,144,0,xxx,int,auto,0,level,0	
golb alias3,144,0,xxx,con,xxxx,0,level,0	Transfers to program element
dclb ,alias2,144,0,xxx,con,xxxx,0,level,0	Program element transfers here
golb alias4,144,0,xxx,con,xxxx,0,level,0	Transfers to code for next specification
dclb ,alias3,144,0,xxx,con,xxxx,0,level,0	
golb alias5,144,0,xxx,con,xxxx,0,level,0	Goes to program element
dclb ,alias4,144,0,xxx,con,xxxx,0,level,0	
coding for remaining specifications	
dclb ,alias5,144,0,xxx,con,xxxx,0,level,0	
ldfl xx0026,27,0,xxx,int,auto,0,level,0	Program element
stfl xx0026,27,0,xxx,int,auto,0,level,0	
gogolb alias1,144,0,xxx,int,auto,0,level,0	Transfers to code for next specification
dclb ,alias9, 144,0,xxx,con,xxxx,0,level,0	Declares label for transfer to next statement

The gogolb macro transfers to the "contents of " alias1; i.e., alias2. Again, since there is no test involved, an immediate transfer is made to the coding for the next specification. This has the same pattern as the coding shown above. The temporary defined by alias1 is used for each specification, but it is only defined once.

This process is shown more graphically by the Pass 2 coding:

```
                    Instruction                     Macro

                equ     alias1,location             dflb
                tra     s.x
s.x:            eapbp   alias2
                stpbp   sp|.u1+4                    ldlb
                stpsp   sp|.u1+6
                eapbp   sp|alias1
                ldaq    sp|.u1+4
                staq    bp|0                        stlb
                ldaq    sp|.u1+6
                staq    bp|2
                tra     alias3                      golb
alias2:         null    "                           dclb
                tra     alias4                      golb
alias3:         null    "                           dclb
                tra     alias5                      golb
alias4:         null    "                           dclb
                 .
                 .
                 .
alias5:         null    "
"
                fld     sp|xx0026           program element
                fst     sp|xx0026
"
                tra     sp|alias1,*                 gogolb
alias9:         null    "                           dclb
```