

Published: 05/18/67

Identification

If Statement

B. Goldberg

1. INTRODUCTION

The if statement may have one of the following forms:

```
if scalar expression then unit-1; else unit-2;  
if scalar expression then unit-1;  
if scalar expression then; else unit-1;
```

In this discussion, the coding produced for the three formats is illustrated for case 1 through the use of simple statements in which tests are made for equality. (Comparison operations involving other types of expressions are covered in BN.6.03.) The general format of the coding is the same for all cases. Only the coding for the if clause differs. This is illustrated in the examples below.

A. Case 1: Simple Comparison Using No Long String Variables

For simplicity, all variables used in the following statements are single-precision floating-point numbers. (The same coding pattern is produced for fixed-point numbers or short strings.)

Statement 1: if a = b then c = d; else e = f;

Pass 1 Coding

<u>Macro</u>	<u>Comment</u>
ldfl <u>alias1</u> , <u>bits1</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} if a = b
ifeqf1 <u>bits1</u> ,0, <u>alias2</u> , <u>bits2</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
golb <u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
ldfl <u>alias4</u> , <u>bits4</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} then c = d
stfl <u>alias5</u> , <u>bits5</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
golb <u>alias6</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
dclb <u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
ldfl <u>alias7</u> , <u>bits7</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} else e = f
stfl <u>alias8</u> , <u>bits8</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
dclb <u>alias6</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	

Here the arrows indicate transfer of control. The first golb macro is produced to provide a false exit; i.e., if  $a \neq b$ , the else coding is executed.

If  $a = b$ , the then coding is executed and the second golb macro "skips around" the else coding.

#### Pass 2 Coding

	<u>Instruction</u>	<u>Macro</u>
	fld sp  <u>alias1</u>	ldfl
	fcmp sp  <u>alias2</u>	ifeqf1
	tnz <u>alias3</u>	golb
"		
	fld sp  <u>alias4</u>	ldfl
	fst sp  <u>alias5</u>	stfl
	tra <u>alias6</u>	golb
<u>alias3:</u>	null "	dclb
"		
	fld sp  <u>alias7</u>	ldfl
	fst sp  <u>alias8</u>	stfl
<u>alias6:</u>	null "	dclb

Statement 2: if a = b then c = d;

Pass 1 Coding

<u>Macro</u>	<u>Comment</u>
ldf1 <u>alias1</u> , <u>bits1</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} if a = b
ifeqf1 <u>bits1</u> ,0, <u>alias2</u> , <u>bits2</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
golb <u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
ldf1 <u>alias4</u> , <u>bits4</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} then c = d
stf1 <u>alias5</u> , <u>bits5</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
dclb <u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	

next macro

Since there is no else code, the golb macro transfers to the next macro. The next macro is also executed following the then code. This eliminates the need for a second golb macro.

Pass 2 Coding

<u>Instruction</u>	<u>Macro</u>
fld sp  <u>alias1</u>	ldf1
fcmp sp  <u>alias2</u>	ifeqf1
tnz <u>alias3</u>	golb
"	
fld sp  <u>alias4</u>	ldf1
fst sp  <u>alias5</u>	stf1
<u>alias3</u> : null "	dclb

Statement 3: if a = b then; else c = d;

Pass 1 Coding

	<u>Macro</u>	<u>Comment</u>
ldfl	<u>alias1</u> , <u>bits1</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} if a = b
ifeqfl	<u>bits1</u> ,0, <u>alias2</u> , <u>bits2</u> 0,xxx,int,auto,0, <u>level</u> ,0	
golb	<u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
golb	<u>alias4</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	then
dclb	, <u>alias3</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	
ldfl	<u>alias5</u> , <u>bits5</u> ,0,xxx,int,auto,0, <u>level</u> ,0	} else c = d
stfl	<u>alias6</u> , <u>bits6</u> ,0,xxx,int,auto,0, <u>level</u> ,0	
dclb	, <u>alias4</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	

This coding is the same as that for statement 1, except that the body of the then clause is missing. Note that two golb macros are produced, because of the inclusion of the else statement.

#### Pass 2 Coding

	<u>Instruction</u>	<u>Macro</u>
	fld sp  <u>alias1</u>	ldfl
	fcmp sp  <u>alias2</u>	ifeqfl
	tnz <u>alias3</u>	golb
"		
	tra <u>alias4</u>	golb
<u>alias3</u> :	null "	dclb
"		
	fld sp  <u>alias5</u>	ldfl
	fst sp  <u>alias6</u>	stfl
<u>alias4</u> :	null "	dclb

#### B. Case 2: Simple Comparison Using Long String Expressions in the If Clause

Here an entry to the string operation routine is called to make the comparison. (See BN.6.03). The following macros are then produced to load the result of the comparison (a 1-bit string) into the bit-string register and to provide a false exit.

```
ldbs      alias1,1,0,xxx,int,auto,0,level,0
iflb     alias2,144,0,xxx,con,xxxx,0,level,0
:
dclb     ,alias2,144,0,xxx,con,xxxx,0,level,0
```

Pass 2 translates this as follows:

```

eapap      splalias1+2,*
eapbp      splalias1,*
adbbp      ap|0
lda        bp|0
anaq       =v1/-1,71/0
tze        alias2

```

```
alias2: null      "
```

### C. Case 3: Use of Short Bit-String Expression in the If Clause

The following source code illustrates this case:

```

declare c bit (5);
if c then go to a;

```

Here the condition is true if any bit of c is a 1; the condition is false only if all bits of c are 0's.

Pass 1 produces the following macros to load the bit string and provide the false exit:

```

ldbs      alias1,5,0,xxx,int,auto,0,level,0
bsbs      5,0,5,0
iflb      alias2,144,0,xxx,con,xxxx,0,level,0
:
dclb      ,alias2,144,0,xxx,con,xxxx,0,level,0

```

Pass 2 translates this coding as follows:

```

eapap      splalias1+2,*
eapbp      splalias1,*
adbbp      ap|0
lda        bp|0
anaq       =v5/-1,67/0
tze        alias2

```

```
alias2: null      "
```

D. Case 4: Use of Long Bit-String Expression in the if Clause

EPL calls the `ixbs_` entry to the string operation (`stgop_`) routine when the scalar expression in the if statement is a long or varying string. (See BN.6.03 for a description of the `ixbs_` entry.) The arguments are: the string, "1"b, and a temporary location in which to store the result. This result is a fixed-point number: 0 if there are no 1-bits in the string, or the position of the first 1-bit in the string. Pass 1 then produces the following macros to load this number and to provide a false exit:

```
ldfx  alias1,bits1,0,xxx,int,auto,0,level,0
iflb  alias2,144,0,xxx,con,xxxx,0,level,0
      .
      .
dc1b  ,alias2,144,0,xxx,con,xxxx,0,level,0
```

Pass 2 translates this code as follows:

```
lda   sp|alias1
tze   alias2
alias2: null  "
```

E. Case 5: Use of a Compound Condition in the If Clause

The coding produced for this case can be understood by examining the following source statement: `if i = j & j = k then go to b ;`

Pass 1 produces the following macros for this statement:

	<u>Macro</u>	<u>Comment</u>
ldfx	<u>alias1</u> , <u>bits1</u> ,0,xxx,int,auto,0, <u>level</u> ,0	Loads j
eqfx	<u>bits1</u> ,0, <u>alias2</u> , <u>bits2</u> ,0,xxx,int,auto,0, <u>level</u> ,0	Compares j with k
stbs	<u>alias3</u> ,1,0,xxx,int,auto,0, <u>level</u> ,0	Stores result in temporary location
ldfx	<u>alias4</u> , <u>bits4</u> ,xxx,int,auto,0, <u>level</u> ,0	Loads i
eqfx	<u>bits4</u> ,0, <u>alias1</u> , <u>bits1</u> ,0,xxx,int,auto,0, <u>level</u> ,0	Compares i with j
ndbs	1,0, <u>alias3</u> ,1,0,xxx,int,auto,0, <u>level</u> ,0	Ands the two results
bsbs	1,0,1,0	Truncates answer to a 1-bit string
iflb	<u>alias5</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	If the answer is a 0, goes to the next statement
golb	<u>alias6</u> ,144,0,con,xxxx,0, <u>level</u> ,0	If the answer is a 1, goes to b
dclb	, <u>alias5</u> ,144,0,xxx,con,xxxx,0, <u>level</u> ,0	

Pass 2 translates this code as follows:

<u>Instruction</u>	<u>Macro/Comment</u>
lda sp, <u>alias1</u>	ldfx
cmpa sp, <u>alias2</u>	eqfx
tze *+3	} Result is 10.....0 if j = k 00.....0 if j ≠ k
lda 0,du	
tra *+2	} stbs
lda 131072,du	
eapap sp, <u>alias3</u> +2,*	
eapbp sp, <u>alias3</u> ,*	
adbbp ap,0	
sta bp,0	ldfx
lda sp, <u>alias4</u>	eqfx
cmpa sp, <u>alias1</u>	} Result is 10.....0 if i = j 00.....0 if i ≠ j
tze *+3	
lda 0,du	} ndbs
tra *+2	
lda 131072,du	
eapap sp, <u>alias3</u> +2,*	
eapbp sp, <u>alias3</u> ,*	
adbbp ap,0	} bsbs
ana bp,0	
anaq =v1/-1,71/0	Transfer if false; i.e., either j ≠ k or i ≠ j
tze <u>alias5</u>	If true go to b
tra <u>alias6</u>	
<u>alias5</u> : null "	