The if statement may have one of the following forms:

\[
\text{if scalar expression then unit-1; else unit-2;}
\]

\[
\text{if scalar expression then unit-1;}
\]

\[
\text{if scalar expression then; else unit-1;}
\]

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
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**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>fld sp</td>
<td>alias1</td>
</tr>
<tr>
<td>fcmp sp</td>
<td>alias2</td>
</tr>
<tr>
<td>tnz alias3</td>
<td>golb</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>fld sp</td>
<td>alias4</td>
</tr>
<tr>
<td>fst sp</td>
<td>alias5</td>
</tr>
<tr>
<td>tra alias6</td>
<td>golb</td>
</tr>
<tr>
<td>alias3: null</td>
<td>&quot;</td>
</tr>
<tr>
<td>alias6: null</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

Macro

- \( 1df1 \) alias1,bits1,0,xxx,int,auto,0,level,0
- \( ifeq1 \) bits1,0,alias2,bits2,0,xxx,int,auto,0,level,0
- \( golb \) alias3,144,0,xxx,con,xxxx,0,level,0
- \( 1df1 \) alias4,bits4,0,xxx,int,auto,0,level,0
- \( stfl \) alias5,bits5,0,xxx,int,auto,0,level,0
- \( golb \) alias6,144,0,xxx,con,xxxx,0,level,0
- \( dclb \) ,alias3,144,0,xxx,con,xxxx,0,level,0
- \( 1df1 \) alias7,bits7,0,xxx,int,auto,0,level,0
- \( stfl \) alias8,bits8,0,xxx,int,auto,0,level,0
- \( dclb \) ,alias6,144,0,xxx,con,xxxx,0,level,0

\[ \text{if } a = b \]
\[ \text{then } c = d \]
\[ \text{else } e = f \]
Statement 2: if a = b then c = d;

Pass 1 Coding

<table>
<thead>
<tr>
<th>Macro</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1df1</td>
<td>alias1, bits1, 0, xxx, int, auto, 0, level, 0</td>
</tr>
<tr>
<td>ifeq1</td>
<td>bits1, 0, alias2, bits2, 0, xxx, int, auto, 0, level, 0</td>
</tr>
<tr>
<td>golb</td>
<td>alias3, 144, 0, xxx, con, xxx, 0, level, 0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1d</td>
<td>sp</td>
</tr>
<tr>
<td>fcmp</td>
<td>sp</td>
</tr>
<tr>
<td>tnz</td>
<td>alias3</td>
</tr>
<tr>
<td>f1d</td>
<td>sp</td>
</tr>
<tr>
<td>fst</td>
<td>sp</td>
</tr>
<tr>
<td>alias3:</td>
<td>null</td>
</tr>
</tbody>
</table>

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

Pass 1 Coding
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.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>fld</td>
<td>sp{alias1}</td>
</tr>
<tr>
<td>fcmp</td>
<td>sp{alias2}</td>
</tr>
<tr>
<td>tnz</td>
<td>alias3</td>
</tr>
<tr>
<td>tra</td>
<td>alias4</td>
</tr>
<tr>
<td>alias3: null</td>
<td>&quot;</td>
</tr>
<tr>
<td>fnl</td>
<td>sp{alias5}</td>
</tr>
<tr>
<td>fnt</td>
<td>sp{alias6}</td>
</tr>
<tr>
<td>alias4: null</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

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 translates this as follows:

```
eapap  sp|alias1+2, *
eapbp  sp|alias1, *
adbbp  ap|0
lda1  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
dblb ,alias2,144,0,xxx,con,xxxx,0,level,0
```

Pass 2 translates this coding as follows:

```
eapap  sp|alias1+2, *
eapbp  sp|alias1, *
adbbp  ap|0
lda1  bp|0
anaq  =v1/-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:

\[
\begin{align*}
\text{ldfx} & \quad \text{alias1}, \text{bits1}, 0, \text{xxx}, \text{int}, \text{auto}, 0, \text{level}, 0 \\
\text{iflb} & \quad \text{alias2}, 144, 0, \text{xxx}, \text{con}, \text{xxx}, 0, \text{level}, 0 \\
\text{dc1b} & \quad \text{alias2}, 144, 0, \text{xxx}, \text{con}, \text{xxx}, 0, \text{level}, 0
\end{align*}
\]

Pass 2 translates this code as follows:

\[
\begin{align*}
\text{lda} & \quad \text{sp|alias1} \\
\text{tze} & \quad \text{alias2} \\
\text{alias2: null} & \quad "\n\end{align*}
\]

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: \( \text{if } i = j \land j = k \text{ then go to } b; \)

Pass 1 produces the following macros for this statement:
Macro

ldfx alias1,bits1,0,xxx,int,auto,0,level,0
eqfx bits1,0,alias2,bits2,0,xxx,int,auto,0,level,0
stbs alias3,1,0,xxx,int,auto,0,level,0
ldfx alias4,bits4,xxx,int,auto,0,level,0
eqfx bits4,0,alias1,bits1,0,xxx,int,auto,0,level,0
ndbs 1,0,alias3,1,0,xxx,int,auto,0,level,0
bsbs 1,0,1,0
iflb alias5,144,0,xxx,con,xxx,0,level,0
golb alias6,144,0,con,xxx,0,level,0
dclb ,alias5,144,0,xxx,con,xxx,0,level,0

Comment

Loads j
Compares j with k
Stores result in temporary location
Loads i
Compares i with j
Ands the two results
Truncates answer to a 1-bit string
If the answer if a 0, goes to the next statement
If the answer is a 1, goes to b

Pass 2 translates this code as follows:

Instruction Macro/Comment
lda spl alias1
cmpa spl alias2
 tze *+3
lda 0,du
 tra *+2
lda 131072,du
eapap spl alias3+2,*
eabpp spl alias3,*
  adbpb apj0
 sta bpj0
lda spl alias4
cmpa spl alias1
tze *+3
lda 0,du
 tra *+2
lda 131072,du
eapap spl alias3+2,*
eabpp spl alias3,*
  adbpb apj0
 ana bpj0
anaq =v1/-1,71/0
 tze alias5
 tra alias6
 alias5: null

ldfx eqfx
Result is 10.....0 if j = k
 00.....0 if j ≠ k
{ }
stbs
Result is 10.....0 if i = j
 00.....0 if i ≠ j
{ }
ndbs
bsbs
Transfer if false; i.e., either j ≠ k or i ≠ j
If true go to b