LOGICAL OPERATIONS

A comprehensive set of logical operations is included on the arithmetic registers, the index registers, and condition bits. For most of the logical instructions the two operands are treated as either 1-, 24-, or 48-bit quantities and a logical connective is applied bit by bit. However, for the "count" instructions a function is computed, not on corresponding pairs of bits of different operands, but on all 24 or 48 bits of one operand.

All logical operations have the short format:

```
   op  i  j  k
```

where the j- and k-fields designate the operand registers or bits and the i-field designates the result register or bit. The contents of the operand registers or bits are not changed by the execution of a logical operation.

The basic set of logical operations provides for eight logical connectives, applied bit by bit on the operands. The truth tables for these eight functions are:

<table>
<thead>
<tr>
<th>function</th>
<th>function value</th>
<th>common names of function</th>
<th>base mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>a ∧ b</td>
<td>a 0 0 1 1 b 0 1 0 1</td>
<td>and, logical product</td>
<td>AND</td>
</tr>
<tr>
<td>a ∧ ē</td>
<td>0 0 0 1</td>
<td>logical difference</td>
<td>TAF</td>
</tr>
<tr>
<td>ā ∧ ē</td>
<td>0 0 1 0</td>
<td>nor, Peirce stroke</td>
<td>FAF</td>
</tr>
<tr>
<td>a ∨ b</td>
<td>1 0 0 0</td>
<td>or, logical sum</td>
<td>OR</td>
</tr>
<tr>
<td>a ∨ ē</td>
<td>0 1 1 1</td>
<td>cover</td>
<td>TOF</td>
</tr>
<tr>
<td>ā ∨ ē</td>
<td>1 0 1 1</td>
<td>nand, Scheffer stroke</td>
<td>FOF</td>
</tr>
<tr>
<td>a = b</td>
<td>1 0 0 1</td>
<td>equivalence</td>
<td>EQ</td>
</tr>
<tr>
<td>a ≠ b</td>
<td>0 1 1 0</td>
<td>not equal, exclusive or,</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modulo 2 sum</td>
<td></td>
</tr>
</tbody>
</table>
It should be noted that all sixteen possible Boolean functions of two variables can be computed by these eight operations by interchanging the names in the j- and k-fields or by setting k equal to j. In particular are the following common functions (where R may be interpreted as either A, X, or C):

- **move**: \( R_i + R_j \) \( R_i + R_j \land R_j \)
- **complement and move**: \( R_i + \bar{R}_j \) \( R_i + R_j \land \bar{R}_j \)
- **set to 0's**: \( R_i + 0's \) \( R_i + R_j \land \bar{R}_j \)
- **set to 1's**: \( R_i + 1's \) \( R_i + R_j \lor \bar{R}_j \)

In addition to the operations included in this section, the shift instructions and certain move instructions provide logical (i.e. bit by bit) functions.
Logical Operations, Arithmetic Unit

ANDA \( A_i ^i + A_j \wedge A_k \)

TAFA \( A_i ^i + A_j \wedge \overline{A}_k \)

FAFA \( A_i ^i + \overline{A}_j \wedge \overline{A}_k \)

ORA \( A_i ^i + A_j \vee A_k \)

TOFA \( A_i ^i + A_j \vee \overline{A}_k \)

FOFA \( A_i ^i + \overline{A}_j \vee \overline{A}_k \)

EQA \( A_i ^i + A_j = A_k \)

XORA \( A_i ^i + A_j \neq A_k \)

Exceptions: none

Logical Operations, Index Unit

ANDX \( X_i ^i + X_j \wedge X_k \)

TAFX \( X_i ^i + X_j \wedge \overline{X}_k \)

FAFX \( X_i ^i + \overline{X}_j \wedge \overline{X}_k \)

ORX \( X_i ^i + X_j \vee X_k \)

TOFX \( X_i ^i + X_j \vee \overline{X}_k \)

FOFX \( X_i ^i + \overline{X}_j \vee \overline{X}_k \)

EQX \( X_i ^i + X_j = X_k \)

XORX \( X_i ^i + X_j \neq X_k \)

Exceptions: none
Logical Operations, Condition Bits

<table>
<thead>
<tr>
<th>Operation</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDC</td>
<td>$c_i + c_j \land \overline{c}_k$</td>
</tr>
<tr>
<td>TAPC</td>
<td>$c_i + c_j \land \overline{c}_k$</td>
</tr>
<tr>
<td>FAFC</td>
<td>$c_i + \overline{c}_j \land \overline{c}_k$</td>
</tr>
<tr>
<td>ORC</td>
<td>$c_i + c_j \lor c_k$</td>
</tr>
<tr>
<td>TOFC</td>
<td>$c_i + c_j \lor \overline{c}_k$</td>
</tr>
<tr>
<td>FOFC</td>
<td>$c_i + \overline{c}_j \lor \overline{c}_k$</td>
</tr>
<tr>
<td>EQC</td>
<td>$c_i + c_j = c_k$</td>
</tr>
<tr>
<td>XORC</td>
<td>$c_i + c_j \neq c_k$</td>
</tr>
</tbody>
</table>

Exception

$c_{24}$ set to 0 or $c_{25}$ set to 1

Exception bit

CC
Count Total Ones, Arithmetic

The contents of register $A_i^j$ are replaced by the number of bits of register $A_i^j$ which have the value 1.

Exceptions: none

Count Leading Alike, Arithmetic

The contents of register $A_i^j$ are replaced by the number of leading bits of register $A_i^j$ which have the value of the bit $A_0^k$. The bits of $A_i^j$ are examined in the order $A_0^j$, $A_1^j$, $A_2^j$, and so on.

Note that if the $k$-field specifies $A_0^0$, the effect is to count leading 0's.

Exceptions: none

Count Leading Different, Arithmetic

The contents of register $A_i^j$ are replaced by the number of leading bits of register $A_i^j$ which are different in value from the value of bit $A_0^k$ (that is, have the value $A_0^k$). The bits of $A_i^j$ are examined in the order $A_0^j$, $A_1^j$, $A_2^j$, and so on.

Note that if the $k$-field specifies $A_0^0$, the effect is to count leading 1's.

Exceptions: none
Count Leading Alike, Index

The contents of register $X^i$ are replaced by the number of leading bits of register $X^j$ which have the value of the bit $X^k_0$. The bits $X^j$ are examined in the order $X^j_0$, $X^j_1$, $X^j_2$, and so on.

Note that if the $k$-field specifies $X^0$, the effect is to count leading 0's.

Exceptions: none

Count Leading Different, Index

The contents of register $X^i$ are replaced by the number of leading bits of register $X^j$ which are different in value from the value of bit $X^k_0$ (that is, have the value $X^k_0$). The bits of $X^j$ are examined in the order $X^j_0$, $X^j_1$, $X^j_2$, and so on.

Note that if the $k$-field specifies $X^0$, the effect is to count leading 1's.

Exceptions: none