COMPARE OPERATIONS

Compare instructions are provided to test specified relations between two numeric quantities and to provide byte testing capabilities.

The effect of the compare instructions is to set a bit called a condition bit. Twenty-four condition bits are provided and are grouped together to form special register S^0. The individual condition bits are identified as $c_0, c_1, \ldots, c_{23}$.

The compare instructions have the following formats:

(I)

(II)

The compare is done between the contents of registers R^j and R^k in format I and between the contents register X^j and the literal h in format II. In both formats the i field designates the bit (or bits) of the condition register which is to be set.

If a compare contains an i field greater than 23, that is, specifies a nonexistent condition bit, the result of the compare is lost. However, if an attempt is made to set c_{24} to 0, or c_{25} to 1, the condition check exception signal CC is generated.

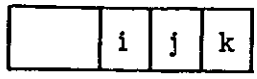
Although only two basic numerical comparison relations are provided in the instruction set (greater than or equal to, and equal to), all six possible relations can be tested either by interchanging the names in the j- and k-fields or by using the negation of the test result. Specifically:

Basic relation to test	Test for	Basic relation true if condition bit has value
$a > b$	$b \geq a$	0
$a \geq b$	$a \geq b$	1
$a = b$	$a = b$	1
$a \neq b$	$a = b$	0
$a \leq b$	$b \geq a$	1
$a < b$	$a \geq b$	0

In the floating point arithmetic section a bit configuration is defined to represent floating point numbers in the exponent overflow range. These numbers are symbolized by u and have the configuration of a 1 in bit zero and 0's in the remaining bits. When one or both operands are u in any of the floating point comparison operations, the result of the compare is made false (0).

The floating point compare operations may give an incorrect result if either or both operands are unnormalized. If either operand is unnormalized, the UO (unnormalized operand) exception bit is set to 1.

Compare, Greater or Equal, Normalized CGEN



The normalized single precision floating point numbers in A^j and A^k are compared. If the number in A^j is greater than or equal to the number in A^k, condition bit c_i is set to 1; otherwise c_i is set to 0.

For the special case when either or both operands are u, condition bit c_i is set to 0.

This instruction may give an incorrect result if either or both operands are unnormalized.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC

Compare, Equal, Normalized CEQN

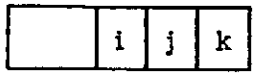
The normalized single precision floating point numbers in A^j and A^k are compared. If the numbers are equal, condition bit c_i is set to 1; otherwise c_i is set to 0.

For the special case when either or both operands are u, condition bit c_i is set to 0.

This instruction may give an incorrect result if either or both operands are unnormalized.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC

Compare, Greater or Equal, Double CGED

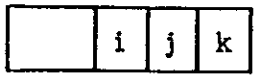


The normalized double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$ are compared. If the number in $A^{j,j+1}$ is greater than or equal to the number in $A^{k,k+1}$, condition bit c_i is set to 1; otherwise c_i is set to 0. The values of the j- and k-fields are assumed to be even.

For the special case when either or both operands are u, condition bit c_i is set to 0.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC
j or k odd	RS

Compare, Equal, Double CEQD

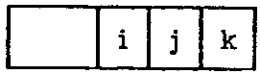


The normalized double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$ are compared. If the numbers are equal, condition bit c_i is set to 1; otherwise c_i is set to 0. The values of the j- and k-fields are assumed to be even.

For the special case when either or both operands are u, condition bit c_i is set to 0.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC
j or k odd	RS

Compare Magnitude, Greater or Equal, Normalized — CMGEN



The magnitudes of the normalized single precision floating point numbers in A^j and A^k are compared. If the magnitude of the number in A^j is greater than or equal to the magnitude of the number in A^k, condition bit c_i is set to 1. Otherwise c_i is set to 0.

For the special case when either or both operands are u, condition bit c_i is set to 0.

This instruction may give an incorrect result if either or both operands are unnormalized.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC

Compare Magnitude, Equal, Normalized — CMEQN

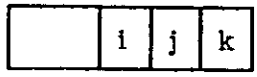
The magnitudes of the normalized single precision floating point numbers in A^j and A^k are compared. If the magnitudes of the numbers are equal, condition bit c_i is set to 1. Otherwise c_i is set to 0.

For the special case when either or both operands are u, condition bit c_i is set to 0.

This instruction may give an incorrect result if either or both operands are unnormalized.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC

Compare Magnitude Double, Greater or Equal — CMGED

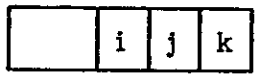


The magnitudes of the normalized double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$ are compared. If the magnitudes of the number in $A^{j,j+1}$ is greater than or equal to the magnitude of the number in $A^{k,k+1}$, condition bit c_i is set to 1. Otherwise, c_i is set to 0. The values of the j- and k-fields are assumed to be even.

For the special case when either or both operands are u, condition bit c_i is set to 0.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC
j or k odd	RS

Compare Magnitude Double, Equal — CMEQD

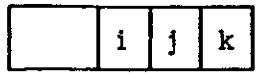


The magnitudes of the normalized double precision floating point numbers in $A^{j,j+1}$ and $A^{k,k+1}$ are compared. If the magnitudes of the numbers are equal, condition bit c_i is set to 1. Otherwise, c_i is set to 0. The values of the j- and k-fields are assumed to be even.

For the special case when either or both operands are u, condition bit c_i is set to 0.

Exceptions	Exception bit
unnormalized operand	UO
c_{24} set to 0 or c_{25} set to 1	CC
j or k odd	RS

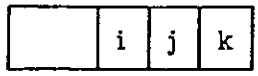
Compare, Greater or Equal,Integer CGEI



The single precision integers in A^j and A^k are compared. If the number in A^j is greater than or equal to the number in A^k, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare, Equal, Integer CEQI



The single precision integers in A^j and A^k are compared. If the numbers are equal, condition bit c_i is set to 1; otherwise c_i is set to 0.

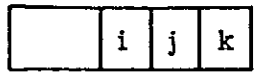
Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Unsigned, Greater or Equal, Integer CUGEI

The contents of registers A^j and A^k are considered as 48-bit unsigned integers. If the number in A^j is greater than or equal to the number in A^k, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

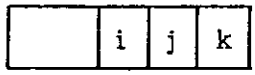
Compare, Greater or Equal, Index CGEX



The index integers in X^j and X^k are compared. If the number in X^j is greater than or equal to the number in X^k, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

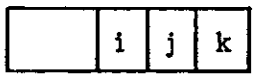
Compare, Equal, Index CEQX



The index integers in X^j and X^k are compared. If the numbers are equal, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Unsigned, Greater or Equal, Index CUGEX



The contents of registers X^j and X^k are considered as 24-bit unsigned integers. If the number in X^j is greater than or equal to the number in X^k, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Index with Constant, Greater or Equal — CGEXK

The index integers in X^j and in the literal h-field are compared. If the number in X^j is greater than or equal to the number in the h-field, condition bit c_i is set to 1; otherwise, c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Index with Constant, Equal — CEQXK

The index integers in X^j and in the literal h-field are compared. If the numbers are equal, condition bit c_i is set to 1; otherwise, c_i is set to 0.

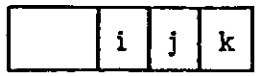
Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Unsigned Index with Constant, Greater or Equal — CUGEXK

The contents of register X^j and the literal h-field are considered as 24-bit unsigned integers. If the number in X^j is greater than or equal to the number in the h-field, condition bit c_i is set to 1; otherwise c_i is set to 0.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Bytes, Arithmetic CBA

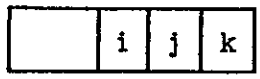


The 48-bit contents of register A^k are considered as six 8-bit operand bytes: the first byte is $A^k_{0,1,\ldots,7}$; the second byte is $A^k_{8,9,\ldots,15}$; and so on. The low order 8 bits of register A^j are considered as one test byte. The test byte is compared with each of the six operand bytes.

Condition bit c_i is set to 1 if the test byte is identical to one or more of the operand bytes; it is set to 0 if the test byte is not identical to any operand byte.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Bytes, Multiple, Arithmetic CBMA



The 48-bit contents of register A^k are considered as six 8-bit operand bytes: the first byte is $A^k_{0,1,\ldots,7}$; the second byte is $A^k_{8,9,\ldots,15}$; and so on. The low order 8 bits of register A^j are considered as one test byte. The test byte is compared with each of the six operand bytes.

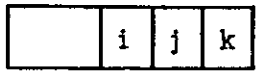
Condition bit c_i is set to 1 if the test byte is identical to one or more of the operand bytes; it is set to 0 if the test byte is not identical to any operand byte.

Condition bit c_{i+1} is set to 1 or 0 according as the test byte is identical to the first operand byte or not; bit c_{i+2} is set to 1 or 0 according as the test byte is identical to the second operand byte or not; and so on through bit c_{i+6}.

Only the leading two bits of the i-field of the instruction are interpreted to determine which condition bits are set, thereby effectively partitioning the condition register into segments: bits 0 to 6, bits 8 to 14, bits 16 to 22, and bits 24 to 30.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Bytes, Index CBX

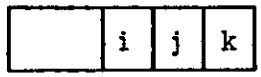


The 24-bit contents of register X^k are considered as three 8-bit operand bytes: the first byte is $X^k_{0,1,\ldots,7}$; the second byte is $X^k_{8,9,\ldots,15}$; and so on. The low order 8 bits of register X^j are considered as one test byte. The test byte is compared with each of the three operand bytes.

Condition bit c_i is set to 1 if the test byte is identical to one or more of the operand bytes; it is set to 0 if the test byte is not identical to any operand byte.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC

Compare Bytes, Multiple, Index CBMX



The 24-bit contents of register X^k are considered as three 8-bit operand bytes: the first byte is $X^k_{0,1,\ldots,7}$; the second byte is $X^k_{8,9,\ldots,15}$; and so on. The low order 8 bits of register X^j are considered as one test byte. The test byte is compared with each of the three operand bytes.

Condition bit c_i is set to 1 if the test byte is identical to one or more of the operand bytes; it is set to 0 if the test byte is not identical to any operand byte.

Condition bit c_{i+1} is set to 1 or 0 according as the test byte is identical to the first operand byte or not; bit c_{i+2} is set to 1 or 0 according as the test byte is identical to the second operand byte or not; and bit c_{i+3} is set to 1 or 0 according as the test byte is identical to the third operand byte or not.

Only the leading three bits of the i-field of the instruction are interpreted to determine which condition bits are set, thereby effectively partitioning the condition register into segments: bits 0 to 3, bits 4 to 7, and so on.

Exception	Exception bit
c_{24} set to 0 or c_{25} set to 1	CC