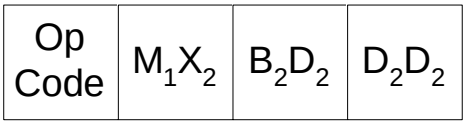
BC M1,D2(X2,B2) | **Branch on Condition** | **RX**



The Branch on Condition instruction examines the 2-bit condition code in the PSW and branches (or not) based on the value it finds. Operand 1 is a self-defining term which represents a 4-bit mask (binary pattern) indicating the conditions under which the branch should occur. Operand 2 is the target address to which the branch will be made if the condition indicated in Operand 1 occurs. If the condition code has one of the values specified in the mask, the instruction address in the PSW is replaced with the target address in the instruction. This causes the processor to fetch the instruction located at the target address as the next instruction in the fetch/execute cycle. See **System 390 Architecture** for more details.

There are four possible values for the condition code:

Condition Code	**Meaning**
00	Zero or Equal
01	Low or Minus
10	High or Plus
11	Overflow

When constructing a mask for Operand 1, each bit (moving from the high-order bit to the low-order bit) represents one of the four conditions in the following order: Zero/Equal, Low/Minus, High/Plus, Overflow. Consider the following instruction,

```
BC   8,THERE
```

The first operand,"8", is a decimal self-defining term and represents the binary mask B'1000'. Since the first bit is a 1, the mask indicates that a branch should occur on a zero or equal condition. Since the other bits are all 0, no branch will be taken on the other conditions. The first operand could be designated as any equivalent self-defining term. For example, the following instruction is equivalent to the one above.

```
BC   B'1000',THERE
```

Extended mnemonics were developed to replace the awkward construction of having to code a mask. The extended mnemonics are easier to code and read. A listing of the extended mnemonics follows below.

BE	Branch Equal	BNE	Branch Not Equal
BZ	Branch Zero	BNZ	Branch Not Zero
BL	Branch Low	BNL	Branch Not Low
BM	Branch Minus	BNM	Branch Not Minus
BH	Branch High	BNH	Branch Not High

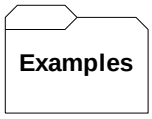
```
BP      Branch Positive     BNP      Branch Not Positive
NOP     No Operation        B        Unconditional Branch
```

Using extended mnemonics we could replace the previous Branch On Condition instruction with the following,

```
BZ   THERE
```

The “BZ” means “Branch on Condition Zero”. When the assembler processes **BZ**, it generates the mask as B’1000’. The table below indicates the possible mask values and the equivalent extended mnemonics.

Eq/Low	Low/Min	High/Plus	Overflow	Decimal Condition	Extended Mnemonic
0	0	0	0	0	NOP
0	0	0	1	1	BO
0	0	1	0	2	BH, BP
0	0	1	1	3	No mnemonic
0	1	0	0	4	BL, BM
0	1	0	1	5	No mnemonic
0	1	1	0	6	No mnemonic
0	1	1	1	7	BNE, BNZ
1	0	0	0	8	BE, BZ
1	0	0	1	9	No mnemonic
1	0	1	0	10	No mnemonic
1	0	1	1	11	BNL, BNM
1	1	0	0	12	No mnemonic
1	1	0	1	13	BNH, BNP
1	1	1	0	14	No mnemonic
1	1	1	1	15	B



Some Unrelated Branch on Conditions

```
         LTR  R8,R8     SET THE CONDITION CODE
         BP   THERE     BRANCH IF CONDITION CODE IS POSITIVE
         ...            OTHERWISE FALL THROUGH TO NEXT INSTRUCTION
THERE    EQU  *

         CLC  X,Y       SET THE CONDITION CODE
         BE   THERE     BRANCH IF X = Y
         ...            OTHERWISE FALL THROUGH TO NEXT INSTRUCTION
THERE    EQU  *

         CLC  X,Y       SET THE CONDITION CODE
         BH   THERE     BRANCH IF X > Y
         ...            OTHERWISE FALL THROUGH TO NEXT INSTRUCTION
THERE    EQU  *
```