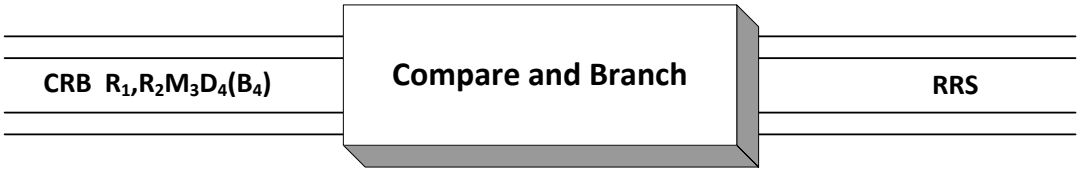


EC	R_1R_2	B_4D_4	D_4D_4	M_3////	F6

The Compare (Registers) and Branch instruction (**CRB**) combines a comparison operation followed by a branch operation into a single instruction. Interestingly, the condition code is unaffected by this operation. The rightmost 32 bits of the operand 1 register are treated as a signed binary integer and compared to the rightmost 32 bits of the operand 2 register. If the mask bit in the M_3 field corresponding to the comparison result is one, the instruction address in the current PSW is replaced by the branch address specified by the fourth operand. In other words, a branch is taken to the address specified in $B_4D_4D_4D_4$. In all other cases, instruction sequencing proceeds with the updated instruction address which will cause execution to resume with the instruction following **CRB.**

The mask consists of 4 bits with the following meanings:

Meaning:	Equal	Low	High	Unused
Bit:	0	1	2	3

For example, if the mask is specified in decimal as 4 = B'0100', the mask indicates we want to branch on a Low condition. If the mask is a decimal 6 = B'0110', the mask indicates we want to branch on a Low or High condition. Another way to describe that is to say we should branch "Not Equal". In every case bit 3 is reserved, and should be 0.

Here is a complete example that uses **CRB**:

Assume: R5 contains X'00000000FFFFFFFF'
R6 contains X'0000000000000001'

We execute: `CRB 5,6,4,THERE`

Register 5 is compared with register 6, and operand 1, which contains -1 in decimal, is lower than operand 2, which contains +1. The mask is designated as 4 = B'0100' and corresponds to a Low condition. As a result, the instruction address field in the PSW is replaced by the address of THERE causing execution to jump to the instruction at that address.

On the other hand,

Assume: R5 contains X'000000000000001A'
R6 contains X'000000000000000C'

We execute: `CRB 5,6,8,THERE`

The mask is designated as 8 = B'1000' and corresponds to an Equal condition. Since the contents of registers 5 and 6 are different, no branch is taken, and execution continues with the instruction following the **CRB**.

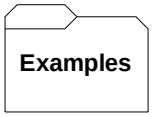
The high level assembler (HLASM) provides extended mnemonics to help you with this instruction. The extended-mnemonic names are formed by concatenating **CRB** with the suffix characters listed in the table below. For example CRBNH is an extended mnemonic for a **CRB** that uses a mask of 12 = B'1100'. Here's another example: Instead of coding,

```
    CRB     5,6,8,THERE
```

we can use the extended mnemonic and code,

```
    CRBE    5,6,THERE
```

Suffix Chars	Meaning	Mask Field
E	Equal	8
H	First operand high	2
L	First operand low	4
NE	Not equal	6
NH	First operand not high	12
NL	First operand not low	10



Some Unrelated CRB's

```
R4 =  X'FFFFFFFF'  -1
R5 =  X'00000028'  +40
R6 =  X'00000000'  ZERO
R7 =  X'00000000'  ZERO
```

```
   OBJECT CODE
EC45 C032 20F6       CRBH  R4,R5,THERE      NO BRANCH TAKEN
EC45 C032 40F6       CRBL  R4,R5,THERE      BRANCH TAKEN TO THERE
EC45 C032 60F6       CRBNE R4,R5,THERE      BRANCH TAKEN TO THERE
EC45 C032 C0F6       CRBNH R4,R5,THERE      BRANCH TAKEN TO THERE
EC45 C032 A0F6       CRBNL R4,R5,THERE      NO BRANCH TAKEN
EC67 C032 C0F6       CRB   R6,R7,12,THERE   BRANCH TAKEN TO THERE
EC67 C032 40F6       CRB   R6,R7,4,THERE    NO BRANCH TAKEN
```

Tips

1) Take advantage of **CRB** to shorten your code, replacing two instructions with one.
2) Make your code easier to read by using the extended mnemonics.