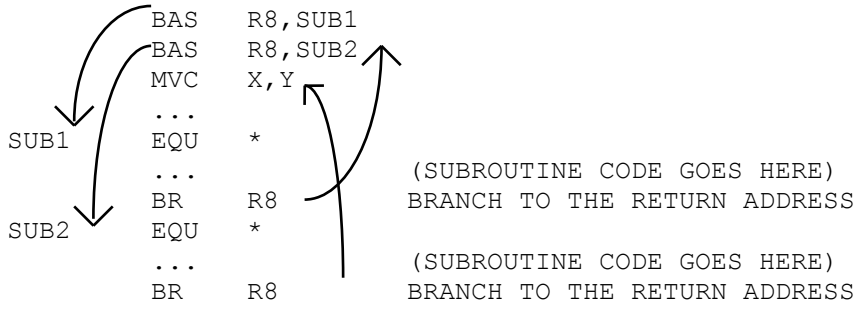
BAS R1,D2(X2,B2) | **Branch and Save** | **RX**

OP Code	R_1X_2	B_2D_2	D_2D_2

BAS is a RX instruction which is used to support internal subroutines. When executed, the address of the instruction which follows the **BAS**, a return address, is stored in the operand 1 register, and a branch is taken to the address specified by operand 2. **BAS** is used in combination with **BR** to construct internal subroutines (routines that are contained in the same control section). Consider the instruction sequence below



When the first **BAS** instruction is executed, the address of the next instruction (BAS R8,SUB2) is loaded into R8. After this return address is loaded, a branch occurs to the address denoted by SUB1. This begins execution of the code in the subroutine. At completion of the subroutine, an unconditional branch (BR R8) occurs to the address in R8. Execution resumes at the second **BAS** instruction. The second **BAS** causes the address of the next instruction (MVC X,Y) to be loaded into R8. A branch is taken to the subroutine denoted by SUB2. After execution of the subroutine, the unconditional branch (BR R8) at the end of the subroutine returns control at the **MVC** instruction.

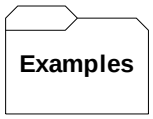
BAS replaces an older instruction called "**BAL**" which stands for "Branch and Link". Both of these instructions load the address of the next instruction into operand 1. The difference in their operation depends on the addressing mode that the machine is using:

In 24 bit mode: BAL loads bits 0 - 7 of operand 1 with linkage information (instruction length code, condition code, program mask)
BAL loads bits 8 - 31 of operand 1 with the 24 bit return address

BAS loads bits 0 - 7 with eight 0's
BAS loads bits 8 - 31 of operand 1 with a 24 bit return address

In 31 bit mode BAS and BAL load bit 0 with a 1 indicating 31 bit mode addressing
BAS and BAL load bits 1 - 31 with a 31 bit return address

The information that was provided by BAL in bits 0 - 7, can now be obtained using the **IPM** (insert Program Mask) instruction.



Some Unrelated BAS's

```
      BAS    R4,HERE  LOAD NEXT ADDRESS IN R4, BRANCH TO "HERE"
      BAS    R8,THERE LOAD NEXT ADDRESS IN R8, BRANCH TO "THERE"
      BAS    R10,YON  LOAD NEXT ADDRESS IN R10, BRANCH TO "YON"
```

Tips

1) Use **BAS** instead of **BAL** to avoid non-zero bits being placed in the high-order byte of the stored address.

2) When creating internal subroutines, consider saving the linkage register on entry to the subroutine and restoring it just before exiting:

```
            SUB1    EQU    *
                    ST     R8,SAVE8      SAVE THE RETURN ADDRESS
LOCALLY
                    ...    (SUBROUTINE CODE GOES HERE)
                    L      R8,SAVE8      RESTORE THE RETURN ADDRESS
                    BR     R8
                    DS    0F
            SAVE8   DS     F
```

By saving and restoring the linkage register, the subroutine is free to call other internal subroutines with the same linkage register.