

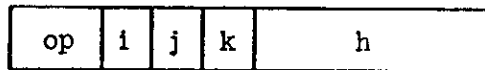
BRANCH AT EXIT OPERATIONS

Branch-at-Exit instructions form the basic set which permits alteration of sequential execution of instructions.

To specify a change in the sequence (i. e. , a branch) three decisions are required: (1) whether or not the branch is to be taken, that is, the condition determination; (2) when the branch is to be taken, the exit point specification; and (3) the address to which the branch is to be made, the effective address calculation.

Condition Determination

The conditional Branch-at-Exit instructions have the long format:



The i- and j-fields designate the bits of the condition register used to determine whether or not the branch is taken. The k-field designates an X-register which with the literal h-field is used to compute the effective branch address.

Whether or not the branch is to be taken is computed as a function of two bits selected from the condition register c (special register S⁰). The i- and j-fields select the bits of c; the function which is computed is specified by the operation code. If the value of the function is TRUE (1), the branch is called successful and the alteration of sequence is effected at the next EXIT instruction. If the value of the function is FALSE (0), the branch is called unsuccessful and no alteration of sequence occurs.

Eight functions can be specified:

$c_i \wedge c_j$	$c_i \vee c_j$
$c_i \wedge \bar{c}_j$	$c_i \vee \bar{c}_j$
$\bar{c}_i \wedge \bar{c}_j$	$\bar{c}_i \vee \bar{c}_j$
$c_i = c_j$	$c_i \neq c_j$

A branch controlled by a single bit may be specified by setting j equal to i . An unconditional branch may be specified by the true function $c_i = c_i$ for any i .

If any of the (non-existent) condition bits 24 through 31 is addressed, the bit value 0 is used.

There is a single unconditional Branch-at-Exit instruction which has the short format:



The i - and j -fields of this instruction are ignored, and the condition value TRUE is used so that this branch is always successful.

Exit Point

The sequential nature of instruction execution is not altered by the Branch-at-Exit instruction itself. Rather, the branch point is marked by an EXIT instruction, and, when a branch is successful, the actual alteration of instruction flow occurs at the EXIT. Instructions between the branch instruction and the EXIT are executed normally, independent of whether the branch is successful or unsuccessful.

When two or more branch instructions occur without an intervening EXIT, the branch instructions are examined in order. The first branch which is successful governs the next EXIT; the other branch instructions which follow the successful branch but precede the EXIT are ignored. The set of branch instructions which relate to a single EXIT need not be in adjacent storage locations but may be interspersed with other instructions (except EXITs).

If an EXIT occurs without a successful branch having been executed since the last previous EXIT, the instruction flow continues in a sequential manner.

Effective Branch Address

The effective branch address, eba , designates the location of the instruction to which the instruction execution sequence will be altered if the branch is successful. The point of alteration is determined by an EXIT instruction.

The eba may be specified in either of two ways: in the 24-bit unconditional branch instruction eba is given directly by the contents of index register k ; in 48-bit instructions eba is the modulo 2^{24} sum of index register k and the 24-bit literal field of the instruction.

ADVANCED COMPUTING SYSTEMS

Volume : 1A
Chapter : 02
Section : Appendix

IBM REGISTERED CONFIDENTIAL

ACS-1 Development Workbook

Page: 9-3

Date: 4/17/67

instruction format

short

long

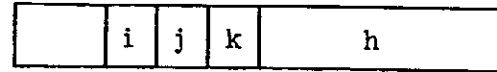
eba calculation

$eba + X^k$

$eba + X^k + h$

If the branch is successful and if the eba designates a missing address, at the next EXIT exception bit MI is set to 1 and the program is interrupted (see the section on Sequencing for further details). If the branch is unsuccessful, no exception can occur.

Branch at Exit, Conditional



<u>mnemonic</u>	<u>function</u>
BAND	$c_i \wedge c_j$
BTAF	$c_i \wedge \bar{c}_j$
BFAF	$\bar{c}_i \wedge \bar{c}_j$
BOR	$c_i \vee c_j$
BTOF	$c_i \vee \bar{c}_j$
BFOF	$\bar{c}_i \vee \bar{c}_j$
BEQ	$c_i = c_j$
BXOR	$c_i \neq c_j$

Exceptions: none

Branch at Exit, Unconditional



<u>mnemonic</u>	<u>function</u>
BU	identically TRUE

Exceptions: none

Exit Operations

An EXIT instruction serves to mark a branch point, where one sequential pattern of instruction execution terminates and another sequential pattern begins.

Two exit operations are provided. The EXIT instruction serves only to designate a branch point. The EXITL instruction does three functions in the following logical order: it sets the skip state to "not skipping", it performs the function of the MLX instruction, and it designates a branch point.

A branch point designation cannot be skipped. Thus, if an EXIT instruction is flagged as skippable, the flag is ignored. If an EXITL is flagged, its first two functions may be skipped but the branch point designation may not.

Exit

EXIT

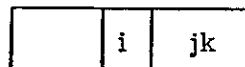


The branching action for any previous branch instruction occurs at the point designated by this instruction.

Exceptions: none

Exit, Save Location and Stop Skipping

EXITL



This instruction is logically identical to the three instructions:

SKTAF 31,31

MLX i,jk

EXIT

Exceptions: none