UNIT 7

Conditional Execution.

Conditional Execution

- Objectives
- COND Parameter
- IF-THEN ELSE Statement

Objectives

- Understand the need for the Conditional statement
- Understand the different types of COND Parameter and their usage
- Understand the IF-THEN ELSE Statement and how to use it

COND - Introduction

Once a JOB starts executing, all of the steps are executed in the order in which they are coded.

JES returns a condition code after the execution of each Job Step in a job. This condition code has a special meaning in the execution process. A condition code of zero specifies a successful execution, while a code other than zero indicates some kind of error. The error code can be a value ranging from 0 to 4095.

Some typical non-zero condition codes (and their meaning) are:

```
COND CODE    0000     Program execution was completely successful.
COND CODE    0004     Execution was ok but cause warning messages.
COND CODE    0008     Program execution was seriously flawed.
COND CODE    0012     Program execution was very seriously flawed.
COND CODE    0016     Program failed disastrously.
```

Condition codes are different from ABEND codes. If a Job STEP abnormally terminates (ABENDS), then all subsequent Steps are bypassed. If a step returns a non-zero condition code, processing continues.

The COND parameter gives you the ability to determine which steps of the job should execute, based on the condition codes returned in previous steps.

The COND parameter is used on the JOB and EXEC statements.

Another way to control step execution based on condition codes is the IF-THEN-ELSE-ENDIF statement, discussed later in this unit.

Why Use The COND Parameter?

There are two reasons why we need the COND parameter:

1. To define conditions under which a step of a normally executing job is to be bypassed.

2. To process a step even if a previous step in a job has abended or only if a previous step has abended. Ordinarily MVS will terminate the entire job when a step abends. The COND parameter allows you to conditionally execute steps based on whether or not the job abended (such as executing a recovery step at the point of an abend).

Parameter –COND

The COND parameter can be coded on both JOB and EXEC statements. The flow chart below explains how the COND parameter works.

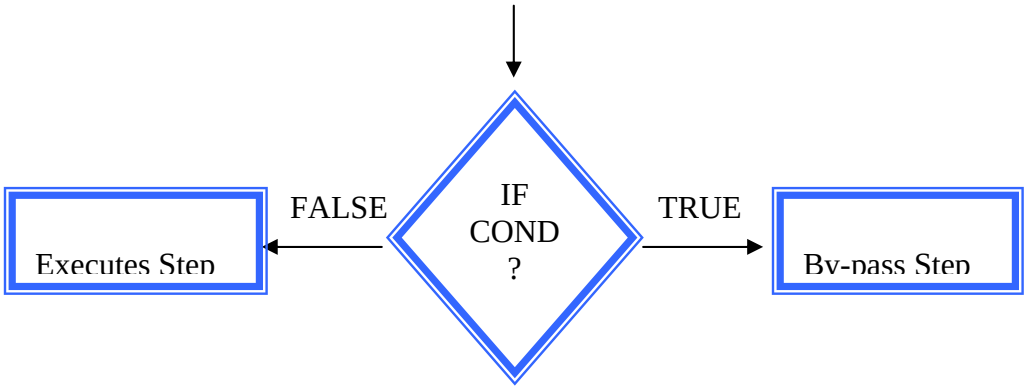

Figure 7-1

The COND parameter states that if a condition is true then the step in which this COND parameter is coded is to be bypassed. If the condition is false than the step is to be executed.

Syntax

The pattern to code COND parameter is

COND=(value, operator, stepname)

(0-4095) (GT — The name of a previous step that returned
LT — the condition code to be tested.
EQ
NE
GE
LE)

COND OPERATORS

GT	greater than
LT	less than
EQ	equal to
NE	not equal to
GE	greater than or equal to
LE	less than or equal to

Example:

```
//STEP08 EXEC PGM=IEWL,COND=(4,LT,STEP06)
```

As mentioned earlier, the COND parameter states that if a condition is true then the step in which this COND parameter is coded is to be bypassed. If the condition is false than the step is to be executed.

In the above example, if 4 is less than the condition code returned for STEP06, then STEP08 will be bypassed and will not execute.

If STEP06 has this CCODE	Will STEP08 Execute?	Why?
0000	Yes	Because 4 is not less than 0.
0001	Yes	Because 4 is not less than 1.
0002	Yes	Because 4 is not less than 2.
0003	Yes	Because 4 is not less than 3.
0004	Yes	Because 4 is not less than 4.
0005	No	Because 4 is less than 5.
0006	No	Because 4 is less than 6.

Compound Tests

You may specify up to eight individual tests for a single step.

The format is:

```
COND=(value,operator,stepname),(value,operator,stepname)
```

When the defined condition is satisfied it causes the step to be by-passed.

There is an implied OR relationship in any compound COND test. Any one condition that is satisfied is sufficient to by-pass the step.

Example:

```
//LKED EXEC PGM=IEWL,COND=((4,LT,COB),(4,LT,PC))
```

In this example, the step ‘LKED’ is conditionally executed based on two separate steps – ‘COB’ and ‘PC’.

Coding The COND Parameter On The JOB Statement

The JOB statement COND parameter performs the same return code tests for every step in a job. If the JOB statement's return code test is satisfied, the job terminates.

The JOB COND parameter performs its return code tests for every step in the job, even when EXEC statements also contain COND parameters.

If any step satisfies the return code test in the JOB statement, the job terminates. The job terminates regardless of whether or not any EXEC statements contain COND parameters and whether or not an EXEC return code test would be satisfied.

If the JOB statements return code test is not satisfied, the system then checks the COND parameter on the EXEC statement for the next step. If the EXEC statement return code test is satisfied, the system bypasses that step and begins processing of the following step, including return code testing.

Examples of JOB Statement Return Code Tests

Example 1:

```
//JOB1  JOB  ,'LEE BURKET',COND=((10,GT),(20,LT))
```

This example asks 'Is 10 greater than the return code or is 20 less than the return code?' If either is true, the system skips all remaining job steps. If both are false the system executes all job steps.

For example, if a step returns a code of 12, neither test is satisfied, the next step is executed. However, if a step returns a code of 25, the first test is false, but the second test is satisfied: 20 is less than 25. The system bypasses all remaining job steps.

Example 2:

```
//J2  JOB  ,'D WEISKOPF',COND=((50,GE),(60,LT))
```

This example says 'If 50 is greater than or equal to a return code, or 60 is less than a return code, bypass the remaining job steps.' In other words, the job continues as long as the return codes are 51 through 60.

COND parameter coded in the EXEC statement

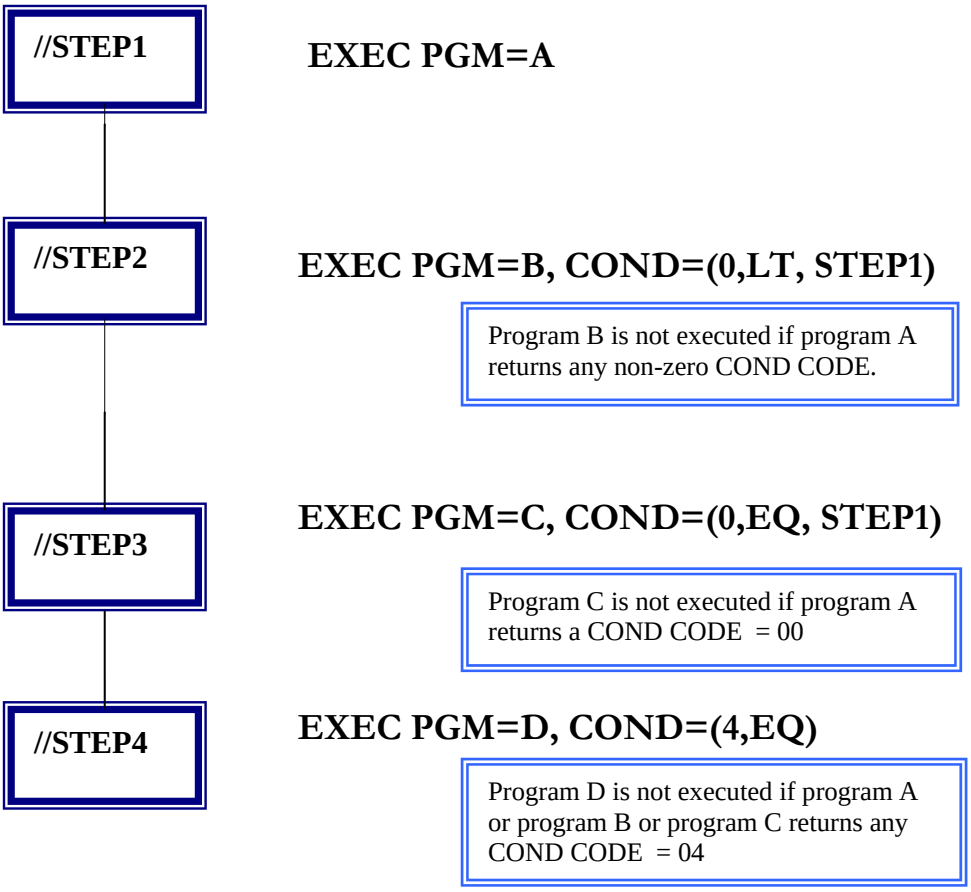

Figure 7-2

The COND parameter can be coded on second and subsequent steps of a job stream.

You can define one or more conditions on a step. If any condition in a COND is true the step is by-passed.

If you omit the stepname in the COND parameter, the test applies to all previous steps (such as in //STEP4 in the example above).

EVEN and ONLY

The job can progress in any one of two modes

1. Normal processing mode.
2. Abnormal termination mode.

All jobs start up in normal processing mode, but when a program abends MVS changes the job to abnormal termination mode. In normal processing mode MVS recognizes each EXEC step and attempts to execute the program coded at it. If any COND conditions on an EXEC are true MVS does not run that step but goes on to the next step.

A job gets into an abnormal termination mode because a program has abended. MVS dumps out the program that fails and flushes out the reminder of the job with no further processing. MVS will acknowledge the presence of the steps remaining after the step that failed, but will not process them. EVEN and ONLY are used under such cases.

EVEN and ONLY prevents MVS from flushing out the rest of the steps when an abend occurs and prevents exiting the program.

If COND=EVEN is coded on an EXEC statement, MVS will process the step even when the job was already put into abend mode by the failure of a previous step.

If COND=ONLY is coded, MVS will process the step *only* if the job is in abend mode. Such a step will be ignored in normal processing mode.

EVEN or ONLY can be coded in a step to gain MVS's attention in abend mode, but both can't be coded together.

```
//STEP2 EXEC PGM=IEBGENER,COND=EVEN
//STEP3 EXEC PGM=IEBGENER,COND=ONLY
```

EVEN or ONLY can be used along with normal COND CODE test.

```
//STEP4 EXEC PGM=IEBGENER,COND=((4,LT,STEP1),EVEN)
```

In this example, MVS will process STEP4 even if a previous step abends.

EVEN and ONLY do not guarantee that a Job step will be executed if a previous step has abended. Only control comes to this step and then a check of all JOB and EXEC level COND parameters are done before actually executing the step.

Effects of EVEN and ONLY

	Normal processing Mode	Abnormal termination mode
No COND code -or- COND return code tests only	• **Step is processed,** • **Return code tests can cause step to be skipped.**	• **Step is acknowledged but not processed.**
COND=EVEN	• **Step is processed, return code tests can cause step to be skipped.**	• **Step is processed** • **Return code tests can cause step to be skipped.**
COND=ONLY	• **Step is acknowledged but not processed.**	• **Step is processed** • **Return code tests can cause step to be skipped.**

IF/THEN/ELSE/ENDIF

The way you code the IF/THEN/ELSE/ENDIF statement construct determines whether the statement construct tests all job steps, a single job step, or a procedure step.

Job Level Evaluation

If you do not code a stepname, the IF/THEN/ELSE/ENDIF statement construct evaluates the return code, abend condition, or run condition of every previous step in the job. If the condition (return code, abend condition or run condition) is satisfied, based on the steps in the job that have executed thus far, the system executes the THEN clause.

Step Level Evaluation

To test a single step, code the stepname of the step you want to test. To test a procedure step, code the stepname.procstepname of the procedure step you want to test. If the step or procedure step that you are evaluating did not execute, was cancelled or ended abnormally, the result of the evaluation is false.

You can code the IF/THEN/ELSE/ENDIF statement construct anywhere in the job after the JOB statement. Code it as follows:

```
//[Name]        IF   (relational expression) THEN
//STEPTRUE      EXEC
//[Name]        ELSE
//STEPFALS      EXEC
//               ENDIF
```

Example:

```
//MODAL2    JOB         '0.2AMIP',…..
//STEP1     EXEC      PGM=IDCAMS
//SYSPRINT  DD          SYSOUT=*
//SYSIN     DD           *
       REPRO                          -
       INDATASET(COBOL.SOURCE.FILE1)  –
       OUTDATASET(COBOL.SOURCE.FILE2)

        REPRO                         -
        INDATASET(COBOL.SOURCE.FILE3) -
        OUTDATSET(COBOL.SOURCE.FILE4)
         IF   MAXCC GT 0
               THEN
                SET   MAXCC=16
         ELSE
                DELETE(COBOL.SOURCE.FILE1)
                DELETE(COBOL.SOURCE.FILE3)
/*
```

For the above example given, first it does the repro (copy) for the files cobol.source.file2, cobol.source.file4 from the files cobol.source.file1, cobol.source.file3, if MaxCC is greater than zero then set that to 16 i.e. (MaxCC=16), else it will delete the input datasets.

The Relational Expression

There are four types of relational operators:

- Comparison operators
- Logical operators
- Not (¬) operators
- Relational expression keywords.

Comparison operators (GT, LT, etc.) compare a relational expression keyword to a numeric value. The comparison results in a true or false condition.

The logical operators & (AND) and | (OR) indicate that the system should evaluate the Boolean result of two or more relational expressions.

The ¬ (NOT) operator reverses the testing of the relational expression.

Relational expression keywords indicate that you are evaluating a return code, ABEND condition, or ABEND completion code.

The THEN clause or ELSE clause must contain at least one EXEC statement. The EXEC statement indicates a job step that the system executes based on its evaluation of the relational expression. A THEN or ELSE clause that does not contain an EXEC statement is a null clause.

You can nest IF/THEN/ELSE/ENDIF statement constructs up to 15 levels of nesting.

Example 1: This example tests the return code for a step.

```
//RCTEST     IF   (STEP1.RC GT 20|STEP2.RC = 60)   THEN
//STEP3      EXEC PGM=U
//ENDTEST    ENDIF
//NEXTSTEP   EXEC
```

The system executes STEP3 if:

- The return code from STEP1 is greater than 20, or
- The return code from STEP2 equals 60.

If the evaluation of the relational expression is false, the system bypasses STEP3 and continues processing with step NEXTSTEP.

The relational expression (Continued)

Example 1: This example tests the return code for a step.

```
//RCTEST     IF   (STEP1.RC GT 20|STEP2.RC = 60)   THEN
//STEP3      EXEC PGM=U
//ENDTEST    ENDIF
//NEXTSTEP   EXEC
```

The system executes STEP3 if

- The return code from STEP1 is greater than 20, or the return code from STEP2 equals 60.
- If the evaluation of the relational expression is false, the system bypasses STEP3 and continues processing with step NEXTSTEP.

Example 2: This example tests for an ABEND condition in a procedure step**.**

```
//ABTEST     IF     (STEP4.LINK.ABEND=FALSE) THEN
//BADPROC    ELSE
//CLEANUP    EXEC   PGM=ERRTN
//ENDTEST    ENDIF
//NEXTSTEP   EXEC
```

The relational expression tests that an ABEND did not occur in procedure LINK, called by the EXEC statement in STEP4. If the relational expression is true, no ABEND occurred. The null THEN statement passes control to step NEXTSTEP. If the relational expression is false, an ABEND occurred. The ELSE clause passes control to the program called ERRTN. This example implies the following naming convention – STEP4.LINK.ABEND=FALSE – is how you would test for an ABEND in STEP4.LINK – meaning that you type the job step and procstep followed by a dot (.) and the work ABEND.

Example 3: This example tests for a user abend completion code in the job.

```
//CCTEST   IF   (ABENDCC = U0100) THEN
//GOAHEAD  EXEC  PGM=CONTINUE
//NOCC     ELSE
//EXIT     EXEC  PGM=CLEANUP
//ENDIF
```

If any job step produced the user abend completion code 0100, the EXEC statement GOAHEAD calls the procedure CONTINUE. If no steps produced the completion code, the EXEC statement EXIT calls program CLEANUP.

Unit 7 Exercises

1. **COND parameter can be coded on _____ and _____ statements.**

2. **The COND parameter on the _____ statement is checked before any other step cond parameters.**

3. **COND=_____ specifies that the step is to be executed whether or not a step abended.**

4. **(T/F) If a job level COND parameter is satisfied all job steps are bypassed.**

5. **STEP5 is coded with COND=(8,LT,STEP4) and STEP4 returns a condition code of 0008. Will STEP5 execute? _____**

6. **STEP08 is coded with COND=(5,GE,STEP06). Answer the following questions:**

If STEP06 has this CCODE	Will STEP08 Execute?	Why?
0004	Yes / No	_____
0005	Yes / No	_____
0006	Yes / No	_____

Unit 7 Lab Exercises

Logon to TSO/ISPF and perform the following exercises. Wherever you see "userid" in lower case, substitute your valid security userid.

1. **Code a two-step JOB where by the second step should not be executed if step 1 goes through successfully.**

2. **You can make use of the utility program IEFBR14 to create a new dataset in step1.**

EX:

```
//JOBNAME JOB,,NOTIFY=APARNA,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//STEP1 EXEC PGM=IEFBR14,COND=ONLY
//SYSPRINT DD SYSOUT=*
//DD1 DD DSN=APARNA.JCL.EX,
//        DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,VOL=SER=LP2WK1,
//        SPACE=(TRK,(5,5),RLSE)
//SYSIN DD DUMMY
//*
//STEP2 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=APARNA.JCL.COBOL,DISP=OLD
//SYSUT2 DD DSN=APARNA.JCL.EX,DISP=OLD
//SYSIN DD DUMMY
//*
```

The above ex. tells that ,if step1 terminates then step2 gets executed, else if step1 executes successfully then it is not passed to step2.