

**Job Control
Language (JCL)**

Lesson 4: The EXEC
Statement

Lesson Objectives

- On completion of this lesson, you will be able to:
 - Explain use of the EXEC statement
 - Explain use of the PGM, REGION, TIME, ADDRSPC, ACCT COND parameters & IF/Else/EndIf



4.1: The EXEC Statement

Introduction

- Function: An EXEC statement identifies the step during the reading process when a job is submitted to the system.
- It defines the STEP and STEP level information to the system.
- Code Snippet:

```
//[stepname] EXEC parameters
```

- Remark:
 - Stepname is optional. When the stepname is omitted, no reference can be made to that step. A job can contain a maximum of 255 EXEC statements.



Copyright © Capgemini 2015. All Rights Reserved 3

Introduction to the EXEC Statement:

An EXEC statement identifies a step during the reading process when a job is submitted to the system. When an EXEC is found, the system accepts all the JCL statements that follow the step, until a delimiter is found. There are four possible delimiters for a step during the reading process:

Another EXEC statement in the input stream (it signals the end of [reading] one step and the beginning of [reading] of another)

A JOB statement

A null statement, that is, // (all JCL statements will be ignored except for a JOB statement)

End-of-file on the reading device; this means, there are no more statements to read.

General Syntax

```
//[stepname] EXEC parameters keyword parameter
```

Stepname: This is optional. When the **stepname** is omitted, no reference can be made. A job can contain not more than 255 steps.

4.1: The EXEC Statement

Step Name

- Must start from column 3 following two //
- Stepname is of 1 to 8 character alphanumeric or national character & UNIQUE
- Optional but mandatory in
 - Overriding
 - STEPNAME.DDNAME
 - Restart=STEPNAME
 - Referback
 - *.STEPNAME.DDNAME

4.2: The PGM Parameter

Using The PGM Parameter

- Function: The PGM parameter specifies the name of the program to be executed in a step.

- Code Snippet:

```
PGM=program-name(load-module)
```

- Remark:

- The program specified in PGM is always a member of a PDS which is called a load library or an executable program library.



Copyright © Capgemini 2015. All Rights Reserved 5

Using The PGM Parameter:

The PGM parameter identifies the program to be executed in a step.

General Syntax

```
PGM=pgmname           positional parameter
```

Pgmname: Name of the program to be fetched from the load library and executed.

The program specified in **PGM** is always a member of library (PDS). This library is commonly known as an executable program library or a load library. The **EXEC** statement can identify only the member. It has no parameter available to identify the library. If necessary, this must be done by using a **JOBLIB** or a **STEPLIB DD** statement.

4.2: The PGM Parameter

Using The PGM Parameter in EXEC - Example

■ Code Snippet

```
//Stepname      EXEC PGM=xyz
//Steplib       DD DSN=DA0001T.LIB.LOAD,DISP=SHR
```

Or

```
//DA0001TA      JOB .....
//joblib        DD      DSN=DA0001T.LIB.LOAD,DISP=SHR
```



Copyright © Capgemini 2015. All Rights Reserved 6

Using The PGM Parameter in EXEC – Example:

```
//DA0001TA  JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//JOBLIB    DD DSN=DA0001T.LIB.LOADLIB,DISP=SHR
//S1        EXEC PGM=ASSIGN1
```

OR

```
//DA0001TA  JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//S1        EXEC PGM=ASSIGN1
//STEPLIB   DD DSN=DA0001T.LIB.LOADLIB,DISP=SHR
```

4.3: Use of Other Parameters in EXEC

Using Other Parameters

- If neither JOBLIB nor STEPLIB is coded, system searches certain predefined libraries (that is, system default libraries). If the specified member is found, it is executed. If not found, the result is S806 ABEND failure.
- The following keyword parameters can be specified at the EXEC statement. The parameters are REGION, ADDRSPC, TIME, COND, ACCT, and PARM.

4.3: Use of Other Parameters in EXEC

Using Other Parameters (Contd.)


- When REGION and ADDRSPC parameter are coded in both the JOB and EXEC statements within a job, the value of the JOB statement will be used.
- If TIME parameter is coded in both the JOB and EXEC statements within a job, both will be in effect. Either of them can cause a S322 ABEND failure.
- These parameters will be used for that step only.
- Example


```
//DA0001TA EXEC PGM = PAYPGM1, PARM = '0791'
```


4.3: Use of Other Parameters in EXEC

Demo

EXEC Parameters



Capgemini
CONSULTING TECHNOLOGY BUSINESS

Copyright © Capgemini 2015. All Rights Reserved 9

The REGION Parameter:

This parameter specifies the available storage limit for the step within address space of the job.

- **General Syntax**

REGION=value{K|M} - keyword parameter

Value : 1 to 2096128 if K (1024 bytes) is used. It should be an even number; it will be rounded to the next higher even number.
1 to 2047 if M (1024K or 1048576 bytes) is used. M is not available to **MVS/SP**, only to **MVS/XA** and **MVS/ESA**.

If the **REGION** parameter is omitted, the **REGION** parameter in the **EXEC** statements within the job is used. If the **REGION** parameter is coded in neither the **JOB** nor the **EXEC** statement, an installation-defined default is used. The default value of most installations is between 500K and 1000K.

If the **REGION** parameter is coded in both the **JOB** and an **EXEC** statement within the job, the value in the **JOB** statement is used.

The **REGION** parameter in the **JOB** statement is used much more often than the one in the **EXEC** statement. Coding the same value for all steps would have the same effect as the **REGION** parameter in the **JOB** statement.

- **Example :**

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//S1      EXEC PGM=ASSIGN1,REGION=500K
//STEPLIB DD DSN=DA0001T.LIB.LOADLIB,DISP=SHR
```

The TIME Parameter:

This parameter specifies the total amount of CPU time that the step is allowed to use.

- **General Syntax**

TIME=([minutes][,seconds] | [1440]) keyword parameter

Minutes: a number from 1 to 1439

Seconds: a number from 1 to 59

1440: The step is not timed for CPU. Note that **TIME=1440** is rarely used, and most installation disallow its use in a testing environment. **TIME=1440** should be used by an on-line system such as **CICS OR ADS/O**.

- **Use:**

- When the **TIME** parameter is omitted, an installation-defined default is used. This default is usually very high and can not cause an **S322 ABEND** failure.
- If the **TIME** parameter is also coded in the JOB statement, both will have effect and either can cause a S322 ABEND failure. It is not advisable to use them both.

- **Remark:**

It is possible for a step to get more CPU time than that is specified in the **TIME** parameter or the default by a maximum 10.5 seconds. This is due to the fact that the system checks for violations every 10.5 seconds.

- **Example:**

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID  
//S1      EXEC PGM=ASSIGN1,REGION=500K,TIME=(,3)  
//STEPLIB DD DSN=DA0001T.LIB.LOADLIB,DISP=SHR
```

4.4: The ADDRSPC Parameter

Using the ADDRSPC Parameter

- The characteristics significant for use are as below:

- ADDRSPC is a keyword parameter.
- It is optional.
- It is installation-dependent.

- Code Snippet:

```
ADDRSPC={VIRT|REAL}
```

- Remarks:

- This is the rarely used parameter because of the default. Note that ADDRSPC=REAL is a parameter that is disallowed in practically all installation because it can cause serious performance problems for other jobs.



Copyright © Capgemini 2015. All Rights Reserved 12

This parameter cannot override a specific ADDRSPC coded on the job statement, but may override the system default.

If ADDRSPC needs to be overridden for a given step of a called procedure this can be done by the inclusion of a procstepname parameter.

Syntax:

```
ADDRSPC [.procstepame] = {VIRT / REAL}
```

VIRT: Request virtual storage. The system can page the step

REAL: Request real storage. The system cannot page the job step and must place the job step in real storage

4.5: The ACCT Parameter

Introduction

- The characteristics significant for use are as below:

- Keyword parameter
- Optional
- Installation-dependent

- Code Snippet:

```
ACCT=(account-no,[additional account-info.])
```

- Remarks:

- The ACCT parameter specifies accounting information to be used for the step as opposed to the Accounting information in the JOB statement.



Copyright © Capgemini 2015. All Rights Reserved 13

The ACCT Parameter:

The parameter specifies accounting information to be used for the step as opposed to the accounting information in the JOB statement.

General Syntax:

```
ACCT=(acctno [,additional-acct-info])      keyword parameter
```

Acctno: The account number to be used for the step

additional-acct-info: same as in the JOB statement but without any JES2 meaning

Use:

The ACCT parameter is seldom used, and when it is, generally only the account number is displayed. This is used to charge resource utilization for a step to a different account number other than the one coded in the JOB statement. If an account number is also coded in the JOB statement, the account number in the EXEC statement is used.

Using the ACCT Parameter (Contd.):

Example:

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//S1      EXEC PGM=IEFBR14,ACCT=('es0013,hr4200,iefbr14')
//DD1      DD
DSN=DA0001T.SHEELA.EMPFILE,disp=(MOD,DELETE),
//          SPACE=(TRK,0),UNIT=SYSDA
```

4.6: The PARM Parameter

Using the PARM Parameter

- The characteristics significant for use are as below:

- Keyword parameter
- Optional
- Installation-dependent

- Code Snippet:

```
PARM=string (a max. of 100 characters)
```

- Remarks: The PARM parameter provides a way to supply data of limited size to the executing program.
- Note : If commas are parts of the string, the entire field must be enclosed in parenthesis or apostrophe's.



Copyright © Capgemini 2015. All Rights Reserved 14

Using the PARM Parameter:

This parameter provides a way to supply data of limited size to the executing program

General Syntax

```
PARM=string      keyword parameter
```

String: A string of characters up to 100. If commas are a part of the string, the entire field must be enclosed in parentheses (or apostrophes). If any portion of the string contains special characters (other than hyphen), that portion of the entire string must be enclosed in apostrophes.

Note: Any parentheses that are used are counted as characters, whereas apostrophes do not. All information after the "=" in the PARM parameter, excluding apostrophes, is saved by the system within the step's own space. When the program begins execution by using the appropriate instructions, it can find the saved information in the storage space.

4.6: The PARM Parameter

Using the PARM Parameter - Examples

- In COBOL the following must be coded:

1. PARM=(A,B,C,D) or 'A,B,C,D'
2. PARM=1005

LINKAGE SECTION.

01 PARM.

05 plength PIC S9(4) COMP.

05 string PIC X(100).

PROCEDURE DIVISION USING PARM.

- plength contains the length of the data passed to the program and data is placed in the variable string.

Using the PARM Parameter – Examples:

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//S1      EXEC PGM=ASSIGN2,PARM='G2 ',TIME=(,1)
//STEPLIB DD DSN=DA0001T.LIB.LOADLIB,DISP=SHR
//INFILE  DD DSN=DA0001T.EMPFILE,DISP=SHR
```

 -Access the data in a Cobol program:

```
Identification Division.
Data Division.
Linkage Section.
01 Parm-data-Area.
                                05 Parm-len  pic s9(4) comp.
                                05 Parm-data1.
                                10 Data1-in  X(100).

Procedure Division using Parm-Data-Area.
Display Data1-in.
```

Parm-len contains the length of the data passed to the program and data is placed in the variable string.

Using the PARM Parameter – Examples:

Rules for continuation

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//COB      EXEC PGM=IKFCBL00,REGION=1024K,
//          PARM=('notrunc,nodynam,lib,size=4096k,buf=116k',
//              'apost,nores,seq')
```

OR

```
//DA0001TA JOB LA2719,CG,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//COB      EXEC PGM=IKFCBL00,REGION=1024K,
//          PARM=(notrunc,nodynam,lib,'size=4096k','buf=116k',
//              apost,nores,seq)
```

Note that an expression in quotes cannot be continued, we need to enclose the string in parentheses and field containing special characters in apostrophes.

Ex.1 PARM='29/06/00' or ('29/06/00')

Ex.2 PARM=(A,B,C,D) or 'A,B,C,D'

PARM parameter (Contd ..)

Pass data vide PARM in the JCL:

```
//Step010 EXEC PGM=Load Module, PARM='Hello World'
```

Coding Rules:

Use apostrophes around the passed data if the data consists of a space

```
PARM='001 JOHN'
```

If passed data comprises an apostrophe or an ampersand, code two such characters consecutively.

```
PARM="'Hello World'"
PARM='ABC&&4'
```

To continue coding data on the next line, code a comma and continue. Also coding apostrophes around continued data is a must. Comma gets included in the byte count.

```
PARM='001,
JOHN'
```

To code till column 71 and then continue, leave column 72 blank and continue on the next line from column 16

4.7: The COND Parameter

Introduction

- The characteristics significant for use are as below:
 - The COND parameter causes conditional execution of steps within a job.
 - It can be coded in the JOB or EXEC statement or both.
 - Return code (or Condition code) is a number between 0 and 4095, issued by an executing program just before the execution is finished.



Copyright © Capgemini 2015. All Rights Reserved 17

Introduction:

Characteristics of the COND parameter are as follows:

The COND parameter can be coded in the JOB as well as the EXEC statement. It is mostly used in the EXEC statement. The main tool for controlling the execution of steps within a job is the COND parameter.

A Return (or Condition) code:

A return code is a number between 0 and 4095, issued by an executing program just before its execution is finished.

It is intended to identify an important event found (or not found) during the execution. For example, a program may issue a return code of 21 to indicate that a problematic event (such as a record is out of sequence) was detected during the execution. Alternatively, a program may issue a return code of 0 to indicate that the execution was trouble-free.

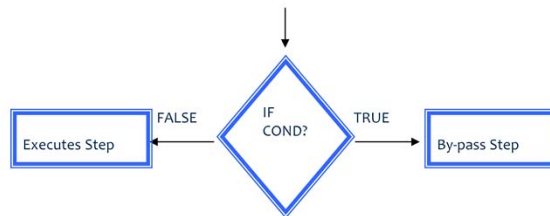
The return code issued by a program is saved by the system for the duration of the job. Any subsequent step of the same job can interrogate this return code by using the COND parameter either in the JOB or EXEC statement. The result of this interrogation is to permit or bypass the execution of the step.

Note: The return code is never available to a job other than the one which issued it. In other words, the step that interrogates the return code must be in the job same as the step that issued it, and it should be subsequent to the step that issued the return code.

4.7: The COND Parameter

The JOB Statement – COND

- 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 then the step is to be executed.



Copyright © Capgemini 2015. All Rights Reserved 18

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.

4.7: The COND Parameter

Using COND in the JOB Statement

■ Code Snippet:

```
COND=((code, operator) [,....])
      code = number between 0 to 4095
      operator = LT, GT, LE, GE, NE, EQ
```

■ Rules:

- Condition is read from left to right.
- Maximum of 8 tests.
- If COND evaluates to TRUE the step is BYPASSED.
- If COND evaluates to FALSE the step is EXECUTED.



Copyright © Capgemini 2015. All Rights Reserved 19

Using COND in the JOB Statement:

IBM-established conventions are as follows:

Return code of 0 indicates a complete success.

Return code of 4 indicates a warning. The warning is benign, so a return code will normally be treated as acceptable.

Return code of 8 indicates questionable results.

Return code of 12 indicates bad results.

Return code of 16 indicates a terminal condition.

Example

There can be a maximum of eight tests in the COND parameter. Condition is evaluated from left to right and if a test is satisfied, the job stops execution at that point.

```
//Step010 Exec Pgm=CBL1
//Step020 Exec Pgm=CBL2,Cond=(0,LT,Step010)
Pgm CBL2 is not executed if pgm CBL1 returns any CC > 0000
//Step030 Exec Pgm=CBL3,Cond=(0,Eq,Step010)
Pgm CBL33 is not executed if pgm CBL1 returns CC 0000
```

4.7: The COND Parameter

Using COND in the JOB Statement - Example

- Consider a job with five steps. Assume that none will ABEND.

```
//DA0001TA JOB LA2719,CG,COND=((12,LT),(8,EQ))
```

- STEP1 issues a return code of 0
- STEP2, if executed, issues a return code of 4
- STEP3, if executed, issues a return code of 16
- STEP4, if executed, issues a return code of 0
- STEP5, if executed, issues a return code of 4



Copyright © Capgemini 2015. All Rights Reserved 20

Using COND in the JOB Statement:

STEP 1 is executed by default, since no previous return codes exist, the COND parameter in the JOB statement is ignored for the first step.

Before STEP2 begins execution, the system interrogates the existing return code (0), using the tests in the COND parameter and reading the test from left to right,

Is 12 less than 0? The answer is "no". The first test of the COND parameter is not satisfied. The second test is tested.

Is 8 equal to 0? . The answer is "no". Neither of the two tests is satisfied, and therefore, STEP2 is executed.

Before STEP3 begins execution, the system interrogates the existing return codes (0 and 4), using the tests in the same COND parameter. Since the result for return code 0 is already known, only 4 is tested:

Is 12 less than 4? The answer is "no". The first test of the COND parameter is not satisfied. The second test is tested.

Is 8 equal to 4 . The answer is "no". Neither of the two tests is satisfied, and therefore, STEP3 is executed.

Before STEP4 begins execution, the system interrogates the existing return codes (0 , 4 and 16), using the tests in the same COND parameter. Since the results for return code 0 and 4 are already known, only 16 is be tested:

Is 12 less than 16? The answer is "yes". The first test of the COND parameter was satisfied. There is no need for the second test. Executions of the job stops. STEP 4 and the remaining steps do not get executed.

A message is displayed as the output: IEF2011 DA0001TA STEP4-JOB TERMINATED BECAUSE OF CONDITION CODES.

4.7: The COND Parameter

Using COND in the EXEC Statement

■ Code Snippet:

```
COND=((code,operator[,stepname])[,...][,EVEN/ONLY])
```

- EVEN requests that execution be permitted even though any previous step has ABENDED (If no step has ABENDED then the EVEN condition is ignored)
- ONLY requests that execution be permitted only if previous step has ABENDED (If no step ABENDs the COND parameter is ignored)



Copyright © Capgemini 2015. All Rights Reserved 21

Using COND in the EXEC Statement:

The COND parameter can perform a test (or multiple tests) before a step begins execution against the return (condition) codes issued by previous steps. If a test is satisfied (reading from left to right), the step is not executed.

General Syntax

```
COND=((code,operator[,stepname])[,(code,operator[,stepname])]....
```

[,EVEN|ONLY]) keyword

parameter

Code : This is a number between 0 and 4095.

Operator: This provides a comparison between a return code and the code. There are six operators: LT, LE, NE, EQ, GT, GE

Stepname: This identifies the name of the preceding step whose return code is interrogated. It can also appear as two names procexec.stepname where procexec identifies the name of the EXEC statement invoking a procedure and "stepname" the stepname within the procedure.

EVEN : This requests that execution be permitted even though a previous (any previous) step has ABENDED.

ONLY : This requests that execution be permitted only if a previous (any previous) step has ABENDED.

4.7: The COND Parameter

Using COND in the EXEC Statement (contd.)

- If the COND parameter is coded in both the JOB and EXEC statements, the COND parameter of the JOB statement is tested first and then the COND of the EXEC statement is tested.



Copyright © Capgemini 2015. All Rights Reserved 22

Using COND in the EXEC Statement (Contd.):

There can be a maximum of eight tests in the COND parameter. EVEN or ONLY count toward eight. Condition is evaluated from left to right and if a test is satisfied, only that step is not executed.

Remark:

EVEN and ONLY cannot make reference to a particular step. They refer to any previous step that has ABENDED.

EVEN and ONLY are mutually exclusive.

EVEN and ONLY have no positional significance. Each can be coded anywhere in the COND parameter in relation to other tests.

Following an ABEND failure, a step cannot be executed unless it contains EVEN or ONLY in the COND parameter of its EXEC statement.

The first step is always executed unless COND=ONLY appears in the EXEC statement.

COND=ONLY causes the first step to be bypassed, since no previous ABEND failures has occurred. Any other COND parameter in the first EXEC statement is ignored (that is, COND=(4,LT) or COND=EVEN) or results in JCL error.

(that is, COND=(5,LT,stepname)) The reason is, there are no previous step.

A step that is not executed issues no return code because a program responsible for issuing the return code was not even loaded into the storage. As a result, no return code exists. An attempt to interrogate the return code of such a step in the COND parameter of a subsequent step is ignored.

4.7: The COND Parameter

Using the COND Parameter

- The following table will help you in solving queries related with the COND parameter.

COND	JOB	EXEC
TRUE	TERMINATE	BYPASSED
FALSE	CONTINUES	EXECUTED

Using The COND Parameter:

A step that ABENDs (carries out ABENDING) issues 'no return code' because a program always issues a return code (conditionally or by default) if it reaches the end of its execution and intentionally returns control to the system. When an ABEND occurs, the program loses control instantly, and is evicted from execution by the system. This results in a specific effect: when a step that ABENDs 'no return code' exists (a completion code exists), an attempt to interrogate the return code of such a step in the COND parameter of a step is ignored until it contains EVEN or ONLY.

Using COND, JOB, EXEC, and PGM:

If the COND parameter is coded neither at the JOB nor at the EXEC statement:

The step is executed regardless of previous return codes.

However, this does not happen if the previous step has ABENDED.

If the COND parameter is coded in both the JOB statement as well as in the EXEC statement within the JOB:

Both are tested.

The COND parameter of the JOB statement is tested first.

If none of its tests are satisfied, then the COND parameter of the EXEC statement is tested.

If a test is satisfied, none of the steps from that point on wards are executed.

4.7: The COND Parameter

IF/THEN/ELSE/ENDIF

- IF/THEN/ELSE/ENDIF statement construct determines for conditionally executing one or more steps.
- Nesting is possible up to 15 levels.
- If the coded condition is true, the following steps till else will be executed and if the condition is false then the steps coded on the else part will be executed.



Copyright © Capgemini 2015. All Rights Reserved 24

Do not specify JOBLIB, JCLLIB,, JOBCAT, STEPCAT, JOB within the THEN or ELSE scope of IF statement.

Relational expression field consists of:

Comparison operators - GT, LT, NG, NL, EQ, NE, GE, LE

Connect operators – AND(&), OR(!), NOT (¬) operators

Keywords:

RC	Indicates Return code
ABEND	Indicates occurrence of Abend
¬ ABEND	Indicates non occurrence of Abend
ABENDCC	Indicates a specific system or user abend code
RUN	Indicates execution of the specified Step
¬RUN	Indicates non execution of the specified step

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:

```
//
DSRP039A JOB (),NOTIFY=&SYSUID
//STEP01 EXEC PGM=IEBGENER
//XYZ IF (STEP01.RC=0) THEN
//STEP02 EXEC PGM=IEBCOPY
//XYZ ELSE
//STEP03 EXEC PGM=IEFBR14
//XYZ ENDIF
//
```


4.7: The COND Parameter

IF/THEN/ELSE/ENDIF (contd..)

■ Syntax

```
//Name IF (Relational expression) THEN  
// Steps to processed when relational expression is true  
//Name ELSE  
// Step to processed when relational expression is false  
//Name ENDIF
```

- Name field is optional.
- Operation fields are IF, THEN, ELSE and ENDIF.
 - ELSE is an optional clause.
 - ENDIF marks the end of the statement

4.8: Using the COND, JOB, EXEC, and PGM Parameters

Example1

```
//DA0001TA      JOB .....  
//S1      EXEC  PGM=P1                      (4)  
//S2      EXEC  PGM=P2,  
//          COND=(0,LT,S1),EVEN)            (12)  
//S3      EXEC  PGM=P3,COND=(8,LT,S2)        (0)  
//S4      EXEC  PGM=P4,COND=(4,LT)           (8)  
//S5      EXEC  PGM=P5,  
          COND=((4,LT,S1),(0,LT,S3)        ABEND  
//S6      EXEC  PGM=P6,  
          COND=((EVEN,(0,LE,S5))            (16)  
//S7      EXEC  PGM=P7,  
          COND=((0,LT,S1),(12,LT,S3))        (0)  
//S8      EXEC  PGM=P8,  
          COND=(16,EQ,S6),ONLY)              (0)  
//S9      EXEC  PGM=P9, COND=EVEN            (4)  
//S10     EXEC  PGM=P10, COND=ONLY           (0)
```

4.8: Using the COND, JOB, EXEC, and PGM Parameters

Example2

//DA0001TA	JOB	LA2719,CG,COND=(10,LT)		
//STEP1	EXEC	PGM=AAA	(6)	
//STEP2	EXEC	PGM=BBB,		
		COND=((2,EQ),(4,EQ))		(2)
//STEP3	EXEC	PGM=CCC, COND=ONLY	(4)	
//STEP4	EXEC	PGM=DDD,		(6)
		COND=(5,GT,STEP1),(2,EQ))		
//STEP5	EXEC	PGM=EEE	(9)	
//STEP6	EXEC	PGM=FFF,	(10)	
		COND=((8,GT,STEP5),EVEN)		
//STEP7	EXEC	PGM=GGG,		(12)
		COND=(4,GT,STEP4)		
//STEP8	EXEC	PGM=HHH	--	
//STEP9	EXEC	PGM=III, COND=ONLY		--

Summary

- The PGM parameter identifies the program to be executed in a step.
- The REGION parameter specifies the limit of available storage for the step.
- The TIME parameter specifies the total amount of CPU time that the step is allowed to use.
- The ADDRSPC parameter specifies if the step will use real or virtual storage.
- The ACCT parameter specifies accounting information to be used for the step.
- The PARM parameter provides a way to supply data of limited size to the executing program.
- Controlling the execution of steps within a job is done by specifying the COND parameter.

