UNIT 1

Introduction to JCL

Introduction to JCL

- Objectives
- What is JCL?
- How MVS handles Jobs
- Initiators
- Structure of JCL
- Parameters
- Schematic representation of a Job-flow

Objectives

- Understand the need for JCL.
- Understand the concept of a Job.
- Visualize the processing flow of a Job.
- Understand the structure of coding JCL.
- Differentiate between the different types of parameters used in JCL.

What is JCL?

JCL stands for **Job Control Language.**

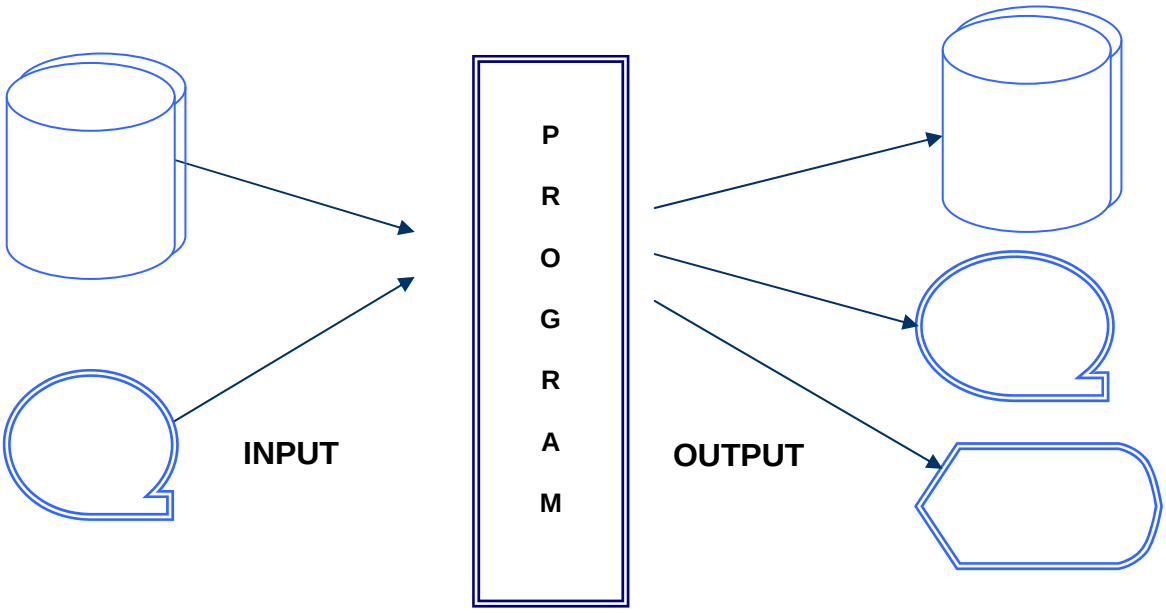


Figure: 1-1

JCL is a language that coordinates the activities of batch programs on the operating system. Using JCL you can tell the system:

- What program(s) you want to execute.
- What input and output datasets these program(s) will be using.
- What should be done with the datasets once the program(s) end(s).

A Batch Job is a set of JCL statements that are used to input this information to the system.

How MVS handles Jobs

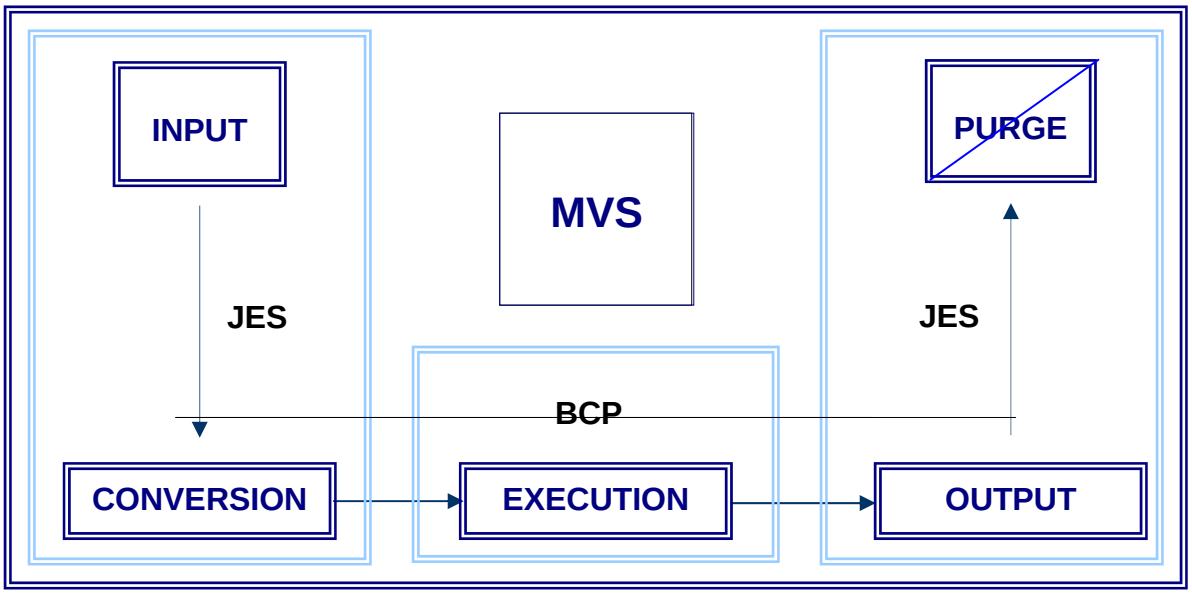


Figure: 1-2

Job Entry Subsystem (JES) handles the Jobs ‘before’ and ‘after’ the execution.

The Base Control Program (BCP) controls when a job gets the CPU for processing (this is called dispatching).

The various stages of a Job while processing are:

- **Input:** The Job is put into JES Input Queue waiting to be dispatched.
- **Conversion**: The Job is checked for Syntax and Validity.
- **Execution:** BCP handles the dispatching of the Job based on priority.
- **Output:** Job is sent to JES Output Queue from where it will be sent for printing.
- **Purge:** Job is in the JES Output Queue from where it can be purged from the system.

Initiators

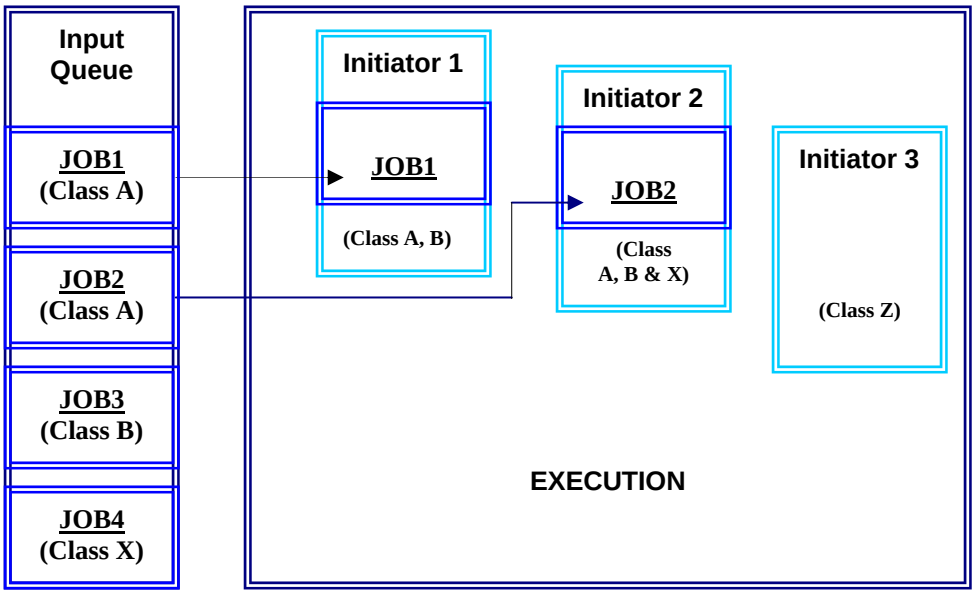


Figure: 1-3

- An Initiator is a special Address Space to which Batch Jobs are mapped during execution.
- A system has a predetermined number of Initiators.
- A Job must be mapped to an Initiator in order to be executed.
- Each Initiator is associated with one or more Classes and can take Jobs with those Classes only (Each Job has one Class).
- An Initiator holds a Job until the execution of the entire Job is over. Only then can it take on any other waiting Job.
- In Figure 1-3 for example,
  - Jobs Job1 and Job2 get mapped to Initiator1 and Initiator2 respectively since their Classes match with those of the Initiators,
  - Job Job3 will have to wait for an Initiator, as, even though Initiator3 is free, it cannot take a Job of Class ‘B’. Same case with Job4.

Initiators (Continued)

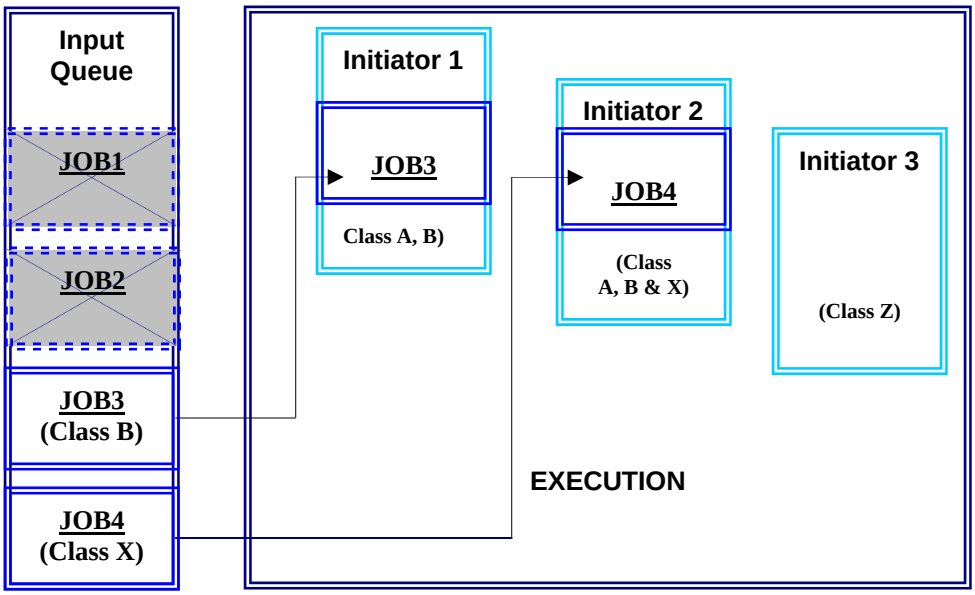


Figure: 1-4

What happens after Jobs Job1 and Job2 finish execution?

- Initiator1 and Initiator2 are now free,
- Job3 is the next Job in the Input Queue, so, it gets associated with Initiator1,
- Similarly, Job4 gets mapped to Initiator2,
- Initiator3 remains idle waiting for a Job with Class ‘Z’.

Structure of JCL

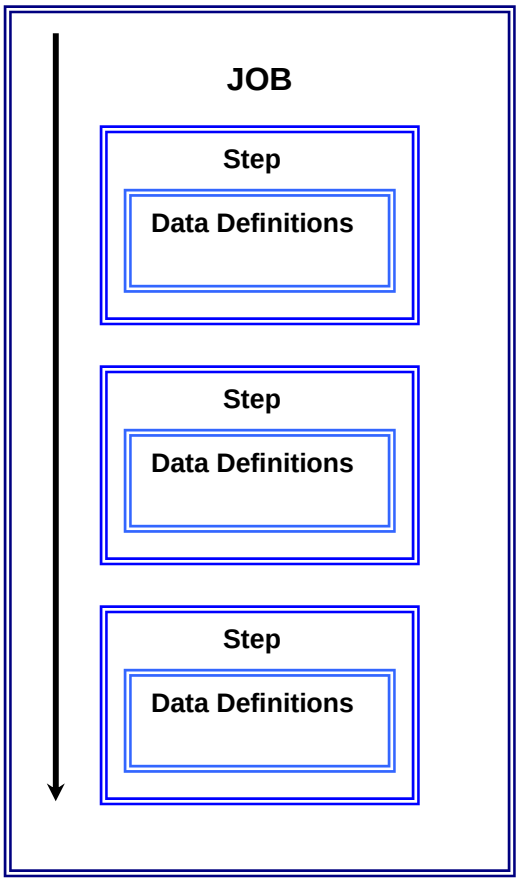


Figure: 1-5

- A Job contains one or more Steps.
- Each Step signifies the execution of a Program.
- Each Step has its own set of Data Definitions. These define the datasets to be used in the step.
- All the Steps in a Job are executed sequentially.
- Max No. of steps can be upto 255.
- Max No. of DD statements can be upto 3273.

Structure of JCL (Continued 1)

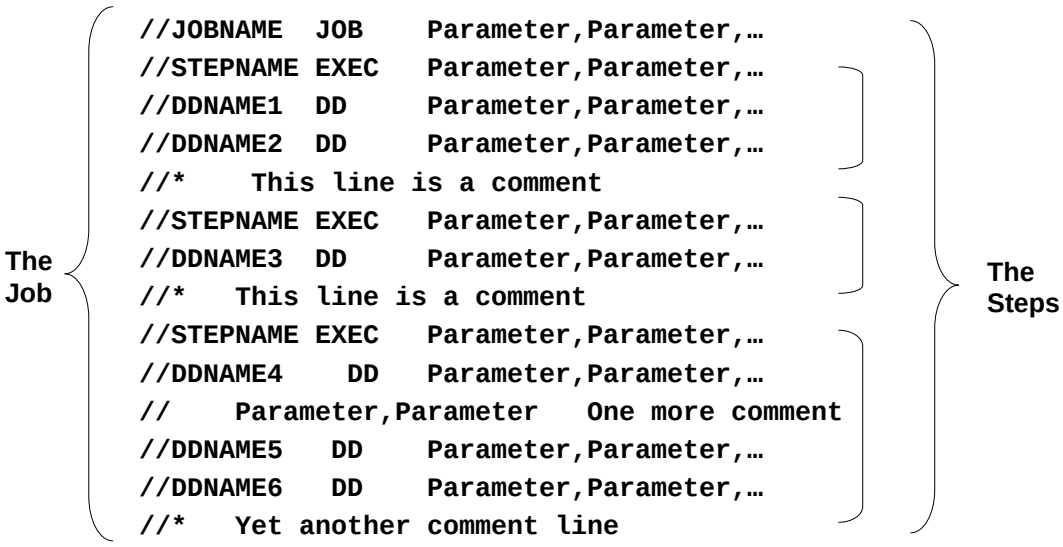


Figure: 1-6

A Job generally looks as displayed above.

There are three primary operations (statement types) in JCL: JOB, EXEC and DD.

A JOB statement signifies the beginning of a Job and also describes the attributes for the entire Job in the form of parameters.

An EXEC statement signifies the beginning of a Step (Execution of a program) and, optionally, the attributes of that Step.

All DD statements following an EXEC are the data definitions for that step. They identify the input and output datasets in the step.

These are only a few of the statements available in JCL. There are more, which will be discussed later in this course.

Each statement accepts a list of parameters. Some parameters are optional, others are mandatory.

Structure of JCL (Continued 2)

A JCL statement consists of one or more 80-byte records.

A Continued JCL statement can begin in column 4-16.

Each JCL statement is divided into five fields:

1. **IDENTIFIER Field (‘//’, ‘/*’, or ‘//*’)**

   Indicates a JCL statement and its type. They are:
   - a. // in columns 1 and 2, define all JCL statements except the delimiters and comments.
   - b. /* in columns 1 and 2 as a delimiter. This signifies the end of in-stream control statements. Installation-designated characters may also be used.
   - c .//* in columns 1,2, and 3 depicts a comment statement

2. **NAME Field**

   Identifies a statement so that it can be referred to later. The name field identifies the name of the job, the name of the step, etc,:
   - Must begin in column 3.
   - 1-8 characters in length (alphanumeric or national (#, @, $))
   - First character must be alphabetic or national.
   - Must be followed by at least one blank.

   Example:

   ```
   //ABC    JOB  ‘MY ACCOUNT’,’PROGRAMMER NAME’,CLASS=A
   ```

   The example above shows the beginning of job ‘ABC’, as defined in the name field of the Job statement.

Structure of JCL (Continued 3)

3. OPERATION Field

Specifies the type of statement or command (e.g. JOB, EXEC, DD, etc.):

i. Follows the NAME Field
ii. Must be preceded and followed by at least one blank.

Examples:

```
//ABC    JOB  'MY ACCOUNT','PROGRAMMER NAME',CLASS=A
//STEP01 EXEC  PGM=USERPGM
```

The first example contains the 'JOB' operation, which identifies the JCL statement as a JOB card.

The second example contains the 'EXEC' operation, which identifies the statement as a step (called STEP01).

4. PARAMETER Field

Contains parameters separated by commas:

i. Follows the OPERATION field.
ii. Must be preceded and followed be at least one blank
iii. Consists of two types:
iv. Positional
v. Keyword

Example:

```
//ABC    JOB  'MY ACCOUNT','PROGRAMMER NAME',CLASS=A
```

The above JOB statement contains three parameters. Notice that they are each separated by a comma, with no spaces in between.

Structure of JCL (Continued 4)

5. COMMENT Field:

Comments can contain any information and are used to document elements of the JCL. The Comment Field:

i. Follows the PARAMETER field

ii. Must be preceded by at least one blank. Comment statement is recommended.

Example:

```
//ABC    JOB  'MY ACCOUNT','PROGRAMMER NAME',CLASS=A
//STEP01 EXEC  PGM=APGEN  GENERATES THE MONTHLY A/P REPORT
```

Anything to the right of a space that follows the last parameter on the line is a comment. Notice that there are no parameters following PGM=APGEN, so everything else on the line is treated as a comment.

```
//ABC    JOB  'MY ACCOUNT','PROGRAMMER NAME',CLASS=A
//STEP01 EXEC  PGM=APGEN
//*  THIS STEP GENERATES THE MONTHLY A/P REPORT
```

The Comment statement is also used to document the purpose of JCL statements, or of the job itself.

Syntax

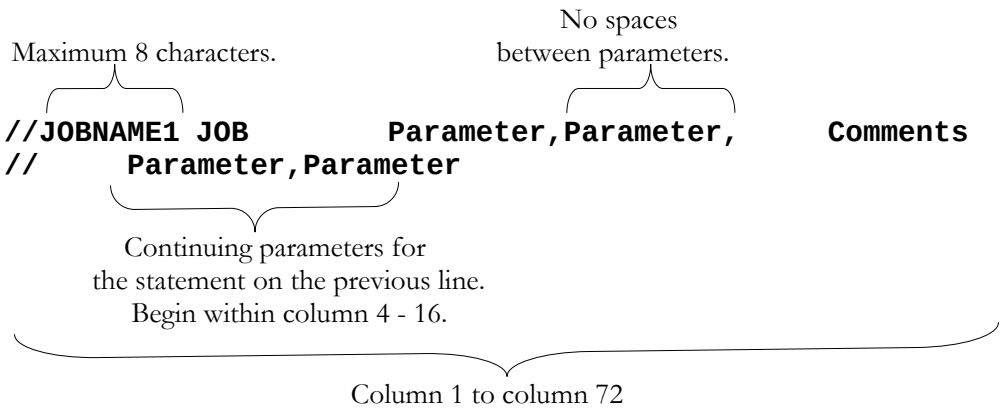


Figure: 1-7

Figure 1-7 shows an example of how a JOB statement might be coded.

Important points to note:

- All JCL statements start with two forward-slashes (//) starting from column 1.
- Code the name for the statement (optional). Do not leave any blanks between the slashes and the name assigned to the statement.
- Leave at least one blank and code the Operation (In this case, 'JOB').
- Leave no blanks between two parameters.
- The parameters for a single statement can span across multiple lines.
- Parameters continued from a previous line must always begin between columns 4 and 16.
- Separate the individual parameters by a comma (,).

If two forward-slashes followed by an asterisk (//*) are found starting from column 1, the entire line is treated as a comment.

Caution: If a blank is left between two parameters on the same line, all the parameters on that line following the blank are treated as comment entries.

Parameters

There are two types of parameters: Positional parameters and Keyword parameters.

1) Positional parameters

If coded, must appear in a particular order.

2) Keyword parameters

Can appear in any order.

Example:

```
//JOBNAME  JOB A,B,X=1,Y=2,Z=3
```

- In the above example, assume A and B to be positional parameters and, X,Y and Z to be keyword parameters.

- Hence, the statement may also be coded as,

```
//JOBNAME  JOB A,B,Z=3,Y=2,X=1
```

- But NOT as,

```
//JOBNAME  JOB B,A,X=1,Y=2,Z=3
```

If you do not want to code any value for the first positional parameter (A) then, you have to notify the absence by coding a positional comma before (B), i.e.

```
//JOBNAME  JOB ,B,Y=2,X=1,Z=3
```

Observe that keyword parameters are usually identifiable by an equal-to (=) operator. In other words, they accept values in the form of one or more constants or parameters.

Subsets of these two types of parameters are listed in the next page.

Parameters (Continued)

Positional Keyword parameters

Even though PGM is a keyword parameter (accepts program name), if it is coded, it must appear immediately after EXEC.

Example:

```
//STEPNAME EXEC PGM=Program1
```

Keyword parameters that accept positional sub-parameters

There are sub-parameters that may also be positional.

Example:

```
//DDNAME DD DISP=(A,B,C)
```

CANNOT be coded as

```
//DDNAME DD DISP=(B,C,A)
```

Keyword parameters that accept keyword parameters

There are also sub-parameters that are not positional.

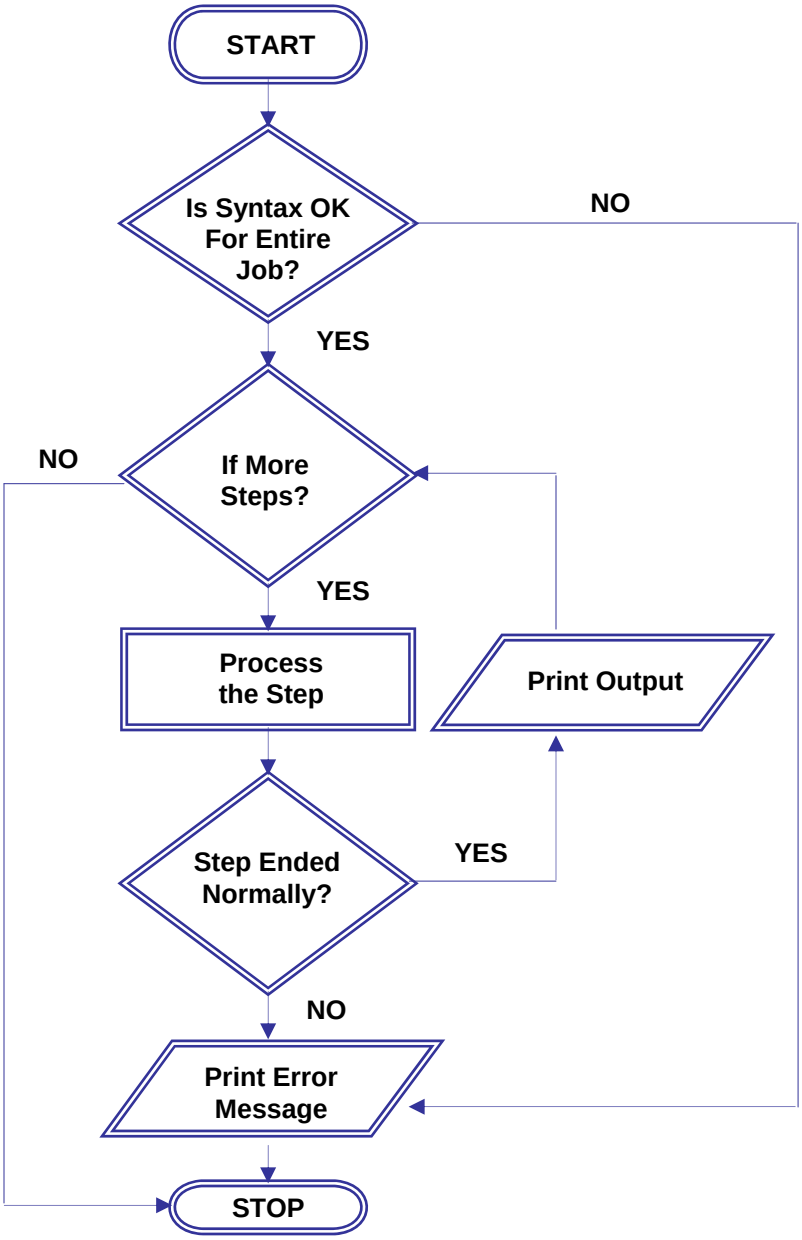
Example:

```
//DDNAME  DD DCB=(A=1,B=2)
```

CAN be coded as

```
//DDNAME  DD DCB=(B=2,A=1)
```

Schematic representation of a Job-flow



When a job is submitted, JES prepares it for execution, and temporarily stores its information on DASD until OS/390 is ready to accept it.

When an initiator becomes available, JES selects the job for OS/390 execution. As OS/390 executes the JOB there is syntax check.

If a JCL syntax error is detected, the system bypasses the entire job, notify the user and does not attempt job execution.

Otherwise the job is executed step by step, simultaneously printing the output after execution of each step.

Figure: 1-8

Unit 1 Exercises

Complete the following:

1. **Which of the following is not a stage during a Job's processing? (*Circle One*)**

   A. Input
   B. Conversion
   C. Execution
   D. Substantiation
   E. Output
   F. Purge

2. **The MVS subsystem that handles a Job before and after execution is called**

   _____.

3. **A special Address Space to which Batch Jobs are mapped during execution is called a**

   _____.

4. **In order for a Job to use an Initiator, the Job must have the same _____**

   as the initiator.

5. **The _____ statement signifies the beginning of a Job.**

6. **The _____ statement signifies the beginning of a step.**

7. **The _____ statement contains the data definitions for that step.**

8. **Which of the following is not a field in a JCL statement? *(Circle One)***

   A. The Identifier field
   B. The Summary field
   C. The Name field
   D. The Operation field
   E. The Parameter field
   F. **The Comment field**

Notes