Processing modes:

3 different modes of processing :

DL/I <BATCH>

BMP <BATCH MESSAGE PROCESSING>

MPP <MESSAGE PROCESSING PROGRAM>

1) 1) DL/I Batch :
   Similar to Non –IMS Batch Program
2) 2) BMP programs are same as DL/I batch programs except for the following differences :
   BMP programs can update online programs but DL/I programs cannot update
   Can use and share online data sets
   BMP 's can access the "ON-LINE" INPUT MESSAGE QUEUE, and can send messages to the OUTPUT MESSAGE
   QUEUE.
3) 3) MPP Programs :
   MPP (Message Processing Program) is what we normally refer to as the "ON-LINE IMS" or "IMS/DC" environment.
   The IMS/DC (Data Communications) system provides users on-line "real time" access to information in IMS databases.

MPP cal uses DC call to access transactions from the message queue , uses DB calls to access the databases and uses DC calls to send output messages back to user at terminal device
MPP transaction programs are NOT submitted with OS JCL. The On-line IMS Control Region automatically schedules
the MPP program necessary to process a transaction.

Components of IMS DC environment :

- IMS Control region
- IMS MPP region
- Terminals
- Transactions
- Message Queues
- On line log datasets
- On line IMS databases
- IMS/DC Application Programs PSB, ACB

IMS CONTROL REGION

The "on-line IMS system" is a control program that executes continuously in the IMS CONTROL REGION, usually called IMS CTL. This program can run either as an MVS "started task" or "JOB step".

The IMS CONTROL REGION "owns":

- IMS databases
- IMS message queues
- IMS terminals
- IMS on-line logs
- IMS transactions

the IMS CONTROL REGION:

- receives TRANSACTIONS from TERMINALS,
- LOGs the transactions,
- stores them in messages QUEUES,
- schedules IMS/DC PROGRAMS to process the TRANSACTIONS.

The IMS CONTROL REGION starts some related address spaces:
- DLI - where all online database access and message queue access are performed.
- DBRC - which is responsible for data sharing, control, and databases recovery functions

in most shops, there will be one (or more) PRODUCTION IMS CONTROL REGION(s) and one (or more) TEST IMS CONTROL REGION(s).

IMS MPP REGION

MESSAGE PROCESSING REGIONS, MPPS, are usually started by the IMS Master Terminal Operator (MTO),
and run either as an OS JOB step, or a started task.
All IMS/DC application program logic is executed in the MPP region.
There are usually multiple MPP regions, depending on the transaction workload. Some MPP regions may be "express" regions for short, quick response transactions (Similar to the express checkout lane in a supermarket). Other MPP regions may handle special classes of transactions.
One MPP region may run many application programs, but only one application at a time.

TERMINALS

Users enter TRANSACTIONS from TERMINALS. IMS TERMINALS may be dumb terminals, PCs, or special
terminals (e.g. Automated Teller Machines).
All terminals to be used by the online IMS system must be defined in the "IMSGEN". Every terminal is
assigned both a physical terminal name and a logical terminal name.
The application programs refer to logical terminals (so the program doesn't need to know if it's talking to a dumb terminal or a PC, for example)
If a PHYSICAL terminal breaks, another terminal can be "assigned" to the LOGICAL terminal name.

Transactions :

Transaction is a basic unit of work to be performed by the online IMS system for a user. All transactions to be used by the online system must be defined in IMSGEN.
TRANSACTION CODES are one of three types of "input messages" that the ON-LINE IMS system can process (the other two message types are "program-to-program message switches" and "IMS SYSTEM COMMANDS").

Message Queues

Input message transactions are sent to the IMS CONTROL REGION which places the message on an "input message queue".
Message queues may reside in main storage and on direct access storage devices (DASD).
The IMS CONTROL REGION schedules the appropriate application program to process the input message on the input message queue. IMS schedules the transaction with the highest priority.
The program may access IMS databases, and prepare an output message (to be sent back to the user requesting service) which is placed on the output message queue.
When messages have been processed or received by the user, they are de-queued and the space in the queue is reused.

ON-LINE LOG DATASETS

ON-LINE LOG DATASETS are used to:
- "log" information about IMS transactions.
- "log" information about changes made to IMS databases.
- "recover" an IMS system when a failure occurs.

ON-LINE LOG DATASETS are kept on direct access storage devices. They are periodically "archived" to tape for storage for a longer period of time.

ON-LINE IMS DATABASES

ON-LINE IMS DATABASES are accessed:

- by on-line IMS/DC application programs using DL/I calls (the same used in DL/I batchIMS/DB programs).
- - to either retrieve data to be displayed on the user terminal, or allow the user to add, update, or delete data in one or more IMS database records.
- - These databases must be defined in IMSGEN. They can be accessed simultaneously by multiple transactions .They cannot be accessed by DL/I batch programs.

IMS DC APPLICATION PROGRAM

IMS/DC Programs require a Program Specification Block (PSB) just like DL/I IMS/DB programs. The PSB defines the logical view of the IMS databases required by the application program.
The Application Control Block (ACB) is the combination of the PSB and DataBase Descriptors (DBD) for the application program.

FLOW OF IMS ON-LINE TRANSACTION

1) The **ON-LINE IMS CONTROL REGION** must be started.This is done by an operator in the computer room,
and is often referred to as "bringing IMS up".
2) One or more **MESSAGE PROCESSING PROGRAM (MPP)** regions must be started by the MASTER TERMINAL
OPERATOR (MTO).
3) Our USER in the Human Resources department needs to do some work - let's say, updating some records on the
EMPLOYEE database...
4)The user "signs on" to the **ON-LINE IMS SYSTEM**, providing her USER ID and PASSWORD (SECURITY is an important consideration for most online systems, especially PERSONNEL/PAYROLL applications)
5) 5) A "Main Menu" is presented. This shows all application systems our user has access to.
6)Our user chooses PERSONNEL from the main menu, and is presented with a PERSONNEL menu, showing options available in this application system.
7) When our user selects option 1 - EMPLOYEE UPDATE,the associated TRANSACTION CODE is used to build
an input MESSAGE, which is edited by **MESSAGE FORMAT SERVICES (MFS)**...
8) The message is stored in the message queue buffer,or message queue dataset (if the buffer is full).
9) Message scheduling is performed by the **IMS CONTROL REGION.** Based on a scheduling algorithm, IMS CONTROL
REGION selects a message for processing.
10) Using the **TRANSACTION CODE,** the appropriate PSB and DBD are selected and loaded.
11) The IMS/DC application program is loaded into one of the MESSAGE PROCESSING PROGRAM (MPP) regions.
12) The application program is executing now :

- Input message is retrieved from message queues
- database calls are made to GET and REPLACE the appropriate database records
- an appropriate output message is sent back to the user, first being placed on the output message queue, and
transmitted to the terminal as appropriate resources are available.

- the program tries to retrieve another message, and receives a 'QC' status code, which means there are no more
messages. The program ends.

13) The **MESSAGE REGION** is now available to schedule and execute another program for the same user
or another user.

Components and structure of an IMS DC Program

IO-PCB

An IMS/DC program requires a data communications PCB for handling input messages from and output messages to
the originating terminal. This is commonly referred to as the IO-PCB.
You will need to code an IO-PCB Mask in the LINKAGE SECTION of your COBOL program.
The IO-PCB must be the first PCB listed in the 'ENTRY' statement in the PROCEDURE DIVISION.
Since IMS automatically provides an IO-PCB for the PSB of MPP and BMP programs, you do not need to include any code for it in the PSB.

ALT-IO-PCB

The ALT-PCB is used to send a message somewhere other than to the originating terminal. It is used for output only and cannot be used for input.

DB-PCB-MASK

Our COBOL IMS PROGRAM contains a DB-PCB "mask" or "structure" for each database our program will use.
If your IMS/DC program accesses an IMS database, you will need:
- a DB PCB Mask in the LINKAGE SECTION of your COBOL program,
- a reference to the DB PCB in the 'ENTRY' statement of your COBOL program,
- a PCB in the PSB.

A typical **DC** call looks as below :

```
CALL CBLTDLI USING  FUNCTION
                    IO-PCB
                    IO-AREA
```

CHNG or change call is used to set the destination of the modifiable alternate PCB in the application.
PURG is used to tell IMS the message is complete
CHKP –Check point tells IMS that a unit of work is complete. Primarily used in BMP
XRST - primarily used in BMPs - if the program has been using CHKPs, then the program can be restarted, if this call is the first
call in the program.

Conversational MPP consists of the following :
Many related transactions or input messages and many responses and output messages. The data needs to be saved here

Non conversational MPP's have only one output message and one transaction .the data need not be saved at all.

ALT-IO-PCB :

ALT-IO-PCB on ISRT call is used to send output to another device or terminal.
Modifiable ALT-IO-PCB is used when the source is not known…

CHNG – call…: - sets the output message destination.
- verifies the destination by checking IMSGEN.
- Destination remains in effect until:
  - a SYNC point is reached (we'll learn about SYNC points later in this module)
  - another CHNG call is issued

Here's the format for coding CHNG:

CALL 'CBLTDLI' USING CHNG-FUNC, ALT-PCB, TERM-NAME.

NOTE: TERM-NAME is a PIC X(08) field defined in the WORKING STORAGE SECTION of the COBOL program.

Check the "STATUS-CODE" for SPACES.

If not SPACES, the CHNG call was not successful.

Possible bad status codes for the CHNG call:

A1 = destination not defined in the IMSGEN
A2 = not a modifiable ALT-PCB, or a partial message has been inserted without the PURG call.
A4 = security violation.

PURG CALL

• signifies to IMS that your message is complete.

Typically the need for this call arises either because you have sent the last segment of a multiple-segment message to the same terminal, or ...you are sending the same message to multiple terminals. In the latter situation, the PURG call terminates the message to one terminal before sending a message to another terminal.

The PURG command has two formats:

CALL 'CBLTDLI' USING PURG-FUNC, ALT-PCB.
CALL 'CBLTDLI' USING PURG-FUNC, ALT-PCB, MSG-AREA.

The first format (without the MSG-AREA), is used most often. It merely signals the end of one message.

The second format (with the MSG-AREA), is equivalent to a PURG call followed by an ISRT call to the message queue with the contents of the WORKING-STORAGE area defined as MSG-AREA.

Check the "STATUS-CODE" for SPACES.

If not SPACES, the PURG call was not successful. In addition, if the MSG-AREA option was specified and the call was not successful, that MSG-AREA did not get inserted.

Program to program Message Processing :

Program A gets a message from the queue using the GU call.
The originating logical terminal name is contained in the prefix (IMS does this for us).
Program A determines it will send a message for processing to Program B. It does this by using an ISRT call to the ALT-PCB.
If the ALT-PCB was defined as fixed, the destination is a TRANSACTION-CODE (rather than a TERMINAL-NAME).
If the ALT-PCB was defined as modifiable, Program A would have to issue a CHNG call, to indicate that the new destination was the TRANSACTION-CODE to be processed by Program B.
Even though this second message did not originate from a terminal, the prefix would contain the name of the terminal that sent the original message.

Program-to-program message switching allows IMS to perform the work required by one transaction in multiple programs.

This is especially useful, for example, if many updates must take place. Rather than doing them immediately, and slowing down the response time back to the user, transaction messages can be put on the queue for later processing by another MPP or BMP.

Another use of program-to-program message switching is for conversational programs. One program may be a Menu program, which
may call another sub-menu program, and still other programs depending on the menu option chosen, and fields entered by the user.

If while updating a database something unexpected happens then the following needs to be done if we want to know whether the updation took place ot not

A SYNC POINT is ...

the point in time when the on-line program tasks of

- output message queueing and
- database updates

are considered to be completely finished.

When a SYNC point occurs...

- IMS removes the input message transaction from the input queue.
- IMS considers all database updates final.
  - all output messages are released.
  - **WHEN DOES A SYNC POINT OCCUR ?**
  - • when the program issues a GU call to the IO-PCB.
  - • when the program does a GOBACK.
  - • when the program issues a CHKP call to the IO-PCB
    for the MODE=MULT transaction (instead of a GU call);
    (this is beyond the scope of this course).

WHAT HAPPENS WHEN YOUR IMS/DC PROGRAM ABENDS ?

• based on the application program control (we will see how in a moment), IMS will "back out" any data base changes made since the last SYNC point occured..
• IMS will delete any output messages not yet on the destination output queue.
• IMS will remove the input message transaction that caused the problem from the input message queue.
• IMS will send an information message about the abend to the IMS MASTER TERMINAL.

If your IMS/DC program discovers a problem severe enough to want to back out database changes made for the transaction in progress, you may do this using one of these DL/I calls: -
ROLL & ROLB
Both of these calls cause IMS to back out all the database changes since the last SYNC point. The difference between the two calls is the location to which program control is transferred.

ROLL Causes an ABEND and control is not returned to your program. **CALL 'CBLTDLI' USING ROLL-FUNC.**
ROLB Does not cause an ABEND, but it re-queues the last input message for processing by your program.
CALL 'CBLTDLI' USING ROLB-FUNC.
NOTE: When using either ROLL or ROLB no output messages are sent unless they are inserted to an ALTERNATE PCB defined as EXPRESS=YES. Refer to your shop standards for online abends, or your DBA for assistance.

BTS TESTING – IBM PRODUCT

BTS uses the following libraries:

- DBDLIB, PSBLIB (ACBLIB)
- PGMLIB

Application programmers will use two data sets:

- BTSIN
- BTSOUT

BTSIN

BTSIN contains BTS commands and "simulator statements" which define the
- transactions
- programs, and
- terminals

to be simulated.

BTSIN may also contain
- DL/I call trace options, and
- - output listing options

BTSOUT

BTSOUT is a 133 byte sequential file created by BTS.

It will contain a "trace" of every- thing that happened with your on-line program. BTS will produce a hard copy listing of your IMS/DC program's logic flow, including:
- commands entered by the user.
- screen images, before & after input.
- IMS/DB PCB-mask, SSAs, and IO-AREA.
- IMS/DC PCB-mask, and IO-AREA.
- MFS screen images.
- BTS statistics.

=-=

PAGING CONCEPT

Paging is a facility that lets messages be broken down into components for display, for processing or for both.

MFS provides two kinds of paging:-

1.**Physical paging**: A message with a fixed number of segments is further divided into parts that are sized properly for a particular device.

***** You can define this through the DFLD statement like pos=(8,12)

***** If you want multiple physical pages just specify MULT=YES in the DPAGE statement in format set.

2.**Logical paging**: A logical page is simply a group of related fields either on the terminal srceen or in a message.

* Can send multiple occurances of the same screen but with different data, as part of one output message.

* You can define it by coding device page(DPAGE)message page(LPAGE) statements

=-=

DIV TYPE=INOUT

You code INOUT for a 3270 display station because an output format becomes the input format for the next message.

The primary function of the DPAGE control statement is to let you specify a device format that will handle multiple logical pages.You may need to use DPAGE statement for initial cursor positioning and field Fill.

=-=

How to code message descriptor control statements

MSG : Identifies the beginning of a message descriptor.
LPAGE : Identifies the beginning of a series of control statements for a logical page subordinate to a MSG statement.
PASSWORD : Identifies the beginning of a series of MFLD statements that are used to construct the password for an input message.
You may code only one PASSWORD statement subordinate to an LPAGE statement.
SEG : Identifies the beginning of a series of MFLD statements the make up a message segment.You may code multiple SEG
statements subordinate to an LPAGE statement.
MFLD : Identifies a message field subordinate to either a PASSWORD statement or a SEG.
MESGEND : Identifies the end of a message descriptor.

=-

LOGICAL PAGING

A logical page is usually associated with a single screen image, just like a physical page. However with logical paging you can send multiple occurrences of the same screen, but with different data as part of one output message. A logical paging is often used for application that produce relatively large amounts of output. A logical page is simply a group of fields either on the terminal screen or in a message.

A group of fields as seen at a device is called a device page(usually a single screen image).
The data that's received from or sent to a device page by an application program is called a message page.

You can code the DPAGE statement to define a device page and the LPAGE statement to define a message page.

With operator logical paging the terminal user has more control over which logical pages of a message are displayed. The user can enter paging commands to tell MFS which page to display next. To define logical pages you need to know the proper coding for the DPAGE & LPAGE statements.

=--=-=-=-=-==-=-=-=-=--=-=-=-=-=--=

HOW TO DEFINE A DEVICE PAGE

To define a logical page as it's displayed on a terminal (device page) use the DPAGE statement.Each device page represents a complete screen image that MFS can select when it sends output to a terminal. Subordinate to each DPAGE statement you code all the DFLD statements necessary to define a complete screen image.
When you develop format sets that define a single device page, you still have to code the DPAGE statement if you want to
specify an initial cursor position on the screen. You can code the FILL parameter on the DPAGE statement, even if the format set contains just one DPAGE. When you code multiple DPAGE statements to define multiple device pages within one format set, you don't have to code any parameters other than the ones you have already seen. You must supply a one-to eight character label for each DPAGE statement so it can be linked to message page. MFS uses the names you assign to DPAGE statements to determine which device page to use when it maps an output message to a terminal.

=--=-=-=-=-==-=-=-=-=--=-=-=-=-=--=

HOW TO DEFINE MESSAGE PAGES

To define a message page you code the LPAGE statement in a message descriptor.
Unlike the DPAGE statement, the LPAGE statement has no function other than to delimit multiple message pages.
When you do define a message that has multiple logical pages, you identify each with an LPAGE statement.
Subordinate to each LPAGE statement you code MFLD statements to define the fields that make up that message page.
You can specify LPAGEs for both input and output messages. To relate the message page an LPAGE statement defines to its corresponding device page, you code the SOR parameter and specify the label of the related DPAGE statement.

Suppose the out put LPAGE statement relates a message page named SUMLPAGE to device format logical page named SUMDPAGE.
(SUMLPAGE LPAGE SOR=SUMDPAGE,……..) then the MFLD statements that follow the SUMLPAGE LPAGE statement are mapped into DFLDs that are part of SUMDPAGE.

=--=-=-=-=-==-=-=-=--=-=-=-=-=--=

World Depend on IMS

It has been said that IMS is a niche product and that the niche is the Fortune 1000 and world-wide equivalent companies of which State Farm is a top member. According to the IBM corporation, 90 percent of the worlds largest companies use IMS. Never was that more apparent when a prior release of IMS (Version 4) was made available before it was stable. IBM had so much adverse reaction from so many large and important companies that they reorganized the IMS development and support areas completely and are now putting money back (15% increase) into IMS development instead of removing IMS's profits for more hi-profile endeavors.

Worldwide IMS serves 150 million end users processing nearly 8 billion transactions every day and manages 1 billion gigabytes of mission critical data. Money and corporate masterfiles are entrusted to IMS by the worlds largest companies. When you use an ATM, it most likely is an IMS terminal. Some companies geographically familiar to State Farm that use IMS are Mitsubishi Motors, Caterpillar, BankOne, Allstate, and John Deere. Banks, manufacturing companies, insurance companies, airlines, and the U.S. government all rely on IMS software to safely and effectively get the job done. **Mitsubishi Bank of Japan recently set a new system uptime record by having an IMS system (handling ATM transactions) up for over 5 years: absolutely no down time for 5 years. The only software in the world that can do that is IMS.**

IBM calls IMS it's premier MVS/ESA and OS/390 transaction manager and it's premier hierarchical database manager. There is no mainframe data management software that guarantees better message and database integrity, is a faster transaction manager, supports more open communications, or is more flexible particularly when appropriately combined with DB2. No doubt, IMS along with DB2, will play a vital roll in future client/server applications via the IBM mainframe as a super-fast and super-safe data server. Worldwide IBM IMS licenses are growing not decreasing. There are 7000 IMS Database Manager customers and 2000 IMS Transaction Manager Customers worldwide (on MVS/OS/390 mainframes only). IBM's worldwide revenues from IMS are $800 million, almost a billion dollars per year. The future of IMS at State Farm and at other enterprise companies is bright indeed.

From a recent Gartner Group Research Note: "IBM's IMS will remain a viable DBMS well into the 21st century. IBM will focus on enhancing the product's strengths and adding advanced database functionality to the IMS product family."

Recently an IBM employee took a look at our transaction rates and remarked that State Farm had a significant percentage of the worldwide IMS transaction rates. Of the 2000 worldwide IMS Transaction Manager customers the average customer executes 4 million IMS transactions per day. State Farm averages almost 21 million: 5 times the average IMS customer.

IMS is a true success story at State Farm, due in part to the quality of the software involved, but more importantly to the State Farm people involved in supporting IMS. IMS and the people that support it are critical, one might say vital to State Farm Insurance.