Debug Tool: Introduction

z/OS Mainframe Utilities & Development Tools

Performance Objectives

You will learn:
• The features and facilities of the Debug Tool.
• Identify and use the Debug Tool interfaces.
• Compiler options associated with installation specific programming languages.
• Use the Full Screen Tool sections.
• Setting a breakpoint.
• Displaying and changing values.
• List and Monitor commands.
• Become aware of the Debug Tool subset of PL/1 commands.
Debug Tool

• Debug Tool helps in testing programs and examining, monitoring, and controlling the execution of programs written in Assembler, C, C++, COBOL, or PL/1 on a z/OS system.

• Debug Tool provides a disassembly view that provides the capability for debugging portions of an application at the machine code level.

Debug Tool Interfaces

Full-screen Mode
• Debug Tool provides an interactive full-screen interface on a 3270 device, with debugging information displayed in three windows: Source, Log and Monitor.

Batch Mode
• Debug Tool command files can be utilized on a running batch application.

Remote Debug Mode
• In remote debug mode, the host application uses a TCP/IP connection to communicate with a remote debugger.
Debug Tool Utilities and Advanced Functions

These tools assist in:

- Preparing high-level language and Assembler programs for debugging.
- Conducting analysis on test cases:
  - Analyzing load modules and program objects in order to identify the language translator used to generate the object for each CSECT.
  - Editing a TEST runtime option data set that the Debug Tool Language Environment user exit uses to start a debug session.
- Invoking IBM File Manager for z/OS.

Proper Compiler Options

- **Compiler Options for use with COBOL Programs**
  - The TEST compiler option provides suboptions to refine debugging capabilities.

- **Compiler Options for use with PL/1 Programs**
  - All PL/1 programs must use the TEST compiler option and suboptions with the following stipulations.
  - Programs compiled with the PL/1 for MVS or OS PL/1 compilers must also specify the SOURCE suboption.
  - The syntax for the TEST compiler option of the Enterprise PL/1 compilers is slightly different.
Starting Debug Tool

• Debug Tool Utilities are used to start the Language Environment program.
• Include the TEST run-time option as part of the command.
  – Example:
    
    `CALL 'USERID1.MYLIB(MYPROGRM)' '/TEST'`

Full Screen Tool

• The Full Screen Tool screen is divided into four sections:
  – **Session panel header**
    • The header fields describe the programming language and the location in the program.
  – **Monitor window**
    • It displays the results of the AUTOMONITOR and MONITOR commands.
  – **Source window**
    • The Source window displays the source or listing file.
  – **Log window**
    • It records your interactions with Debug Tool and the results of those interactions.
Single Stepping

- Stepping through a program means that a program is run through one line at a time.
- After each line is run, changes in program flow and storage can be observed.
- These changes are displayed in the: Monitor window, Source window, and Log window.
- The STEP command is used for stepping through a program.

Running a Program to a Specific Line

It is possible to run from one point in a program to another point by using one of the following methods:

- Setting a breakpoint and using the GO command.
  – The RUN command is synonymous with the GO command.

- The GOTO command will resume a program at a specified point.
  – The point is specified in the GOTO command.

- The RUNTO command runs a program to a specified point.
  – The point is specified in the RUNTO command.
Setting a Breakpoint

- Breakpoints can indicate a stop point in a program and a stopping point in time.
- Breakpoints can also contain activities, such as:
  - Instructions to run.
  - Calculations to perform.
  - Changes to make.
- A basic breakpoint indicates a stopping point in a program.
  - Example:
    - In order to stop on line 100 of a program, enter:
      AT 100
    - In the Log window, the message AT 100 ; will appear.

Complex Instructions

- A breakpoint can contain complex instructions.
- Example:
  - When Debug Tool reaches line 100, it alters the contents of the variable myvar if the value of the variable mybool is true:
    \[
    \text{AT 100 if (mybool == TRUE) myvar = 10 ;}
    \]
Displaying Values

- After becoming familiar with setting breakpoints and running through a program, the value of a variable can be displayed.
- The value of a variable can be displayed in one of the following ways:
  - One-time display in the Log window is useful for checking the value of a variable.
  - Continuous display in the Monitor window is useful for observing the value of a variable over time.
  - A combination of one-time and continuous display, where the value of variables coded in the current line are displayed, is useful for observing the value of variables when the variables are used.

List and Monitor Commands

- For one-time display, enter the following command on the command line:
  LIST (x)
  – where x is the name of the variable.

- For continuous display, enter the following command on the command line:
  MONITOR LIST ( x )
  – where x is the name of the variable:
Changing Values

• Changing the value of a variable depends on the programming language that is being debugged.
• In Debug Tool, the rules and methods for the assignment of values to variables are the same as programming language rules and methods.

Skipping a Breakpoint

• The DISABLE command can be used to temporarily disable a breakpoint.
• The ENABLE command can be used to re-enable the breakpoint.
Clearing a Breakpoint

- When a breakpoint is no longer required, it can be cleared.
- Clearing the breakpoint removes any of the instructions associated with that breakpoint.
  - Example:
    CLEAR AT 100
    This will clear the breakpoint.

Stopping Debug Tool

- The QUIT command is used for stopping the Debug Tool.
Preparing a PL/1 Program

• When a program is under development, the program should be compiled with the TEST(ALL) compiler option in order to utilize all of the Debug Tool’s capabilities.
• Depending on the compiler and compiler options that have been selected, it will be necessary to save the source, listing, or separate debug file. The Debug Tool requires that information to display the source.

PL/1 - TEST Compiler Option

• The PL/1 compiler provides the TEST compiler option and its suboptions to control the placement of hooks and symbol tables.
• The suboptions regulate the points at which the compiler inserts hooks.
  – BLOCK
  – STMT
  – PATH
  – ALL
  – NONE
Debug Tool Subset of PL/1 Commands

Command
- Description
  • Assignment
    – Scalar and vector assignment.
  • BEGIN
    – Composite command grouping.
  • CALL
    – Debug Tool procedure call.
  • DECLARE or DCL
    – Declaration of session variables.

Debug Tool Subset of PL/1 Commands

Command
- Description
  • DO
    – Iterative looping and composite command grouping.
  • IF
    – Conditional execution.
  • ON
    – Define an exception handler.
  • SELECT
    – Conditional execution.