BMC Mainframe: Using z/OS Assembler

COURSE ABSTRACT

COURSE CODE

» MGRS-UZOA-2021

APPLICABLE VERSIONS

» Not Applicable

DELIVERY METHOD

» Instructor-led Training (ILT)

COURSE DURATION

» 5 Days

PREREQUISITES

» Experience of using z/OS, including using TSO/ISPF and submitting batch jobs. For programmers it is an advantage to have a good knowledge of either a high level (example, COBOL or PL/I) or a procedural language (example, CLISTs or REXX)

RECOMMENDED TRAININGS

» NA

Course Overview

The course is developed and delivered by © RSM Technology.

This is the definitive Assembler course. It is suitable for all systems and application programmers who need to understand Assembler, either to install and maintain systems software or to maintain and amend application programs or packages written in Assembler.

This course describes and explains how Assembler works, and teaches how to read, interpret, and modify Assembler routines.

Target Audience

Systems Programmers and Software Developers who will need to 'read, mend and modify' Assembler programs.

Learner Objectives

» Describe the format of machine instructions
» Demonstrate an understanding of addressing and base registers
» Explain how control sections and dummy sections are used
» Describe how to use a working set of machine instructions
» Explain the difference between machine and Assembler instructions
» Define constants and storage
» Code instructions which will perform arithmetic, data manipulation, program control, file handling and table handling
» Explain how condition codes and masks are used
» Read dumps
» Explain how module linkage works
» Use BASSM and BSM instructions
» Understand the concept of 31 & 64 bit addressing and its effect on existing machine instruction

BMC Mainframe: Using z/OS Assembler

COURSE ABSTRACT

COURSE ACTIVITIES

» Classroom Presentations
» Demonstration

BMC MAINFRAME INFRASTRUCTURE AND PLATFORMS LEARNING PATH

» https://www.bmc.com/education/courses/find-courses.html#filter/%7B%22type%22%3A%22edu-specific-types-159150236%22%7D

CERTIFICATION PATHS

» This course is not part of a BMC Certification Path.

DISCOUNT OPTIONS

» Have multiple students? Contact us to discuss hosting a private class for your organization
» **Contact us for additional information**

Course Modules

Introduction to Assembler Programs

» Programming languages
» The Assembly process
» The Linking process
» Assembler program structure
» Assembler directives
» Macro instructions
» Assembler instructions
» Instruction formats
» Storage locations and addresses
» Machine instructions
» Symbolic statements
» Syntax
» Statement format
» The Operation & operand fields
» Hexadecimal - a shorthand form of binary
» Character

Handling Character Data

» Defining fields
» Define Storage & Constants
» Examples of DS & DC
» Defining areas with zero repetition
» Move instructions
» Move Long instruction
» Literals, constants, and immediate data
» Dummy Sections (DSECTs)
» Equates (EQU)

Comparisons, Branching & Structure

» Comparison instructions
» Branching Instructions
» Branch Mnemonics (BC instruction)
» BCT/BCTR for looping
» Program skeleton
» Executable code
» The location counter
» Addressing
» Register convention
» Save area chaining
» Relative addressing
» Indexing
» Declarative part
» Instructions to the Assembler
» Subroutines
» Branch and Link Instructions
» BASSM & BSM instructions
» Branch Register instruction
» BXH / BXLE for looping
» Execute
» Program linkage with BAKR
» Linkage Stack
» Linkage Stack operations - PICs

Binary Instructions

» Signed binary arithmetic instructions
» Types of Binary Instruction
» Loading and Storing
» Load and Store Instructions
» The Binary Add Instructions
» Binary Subtraction Instructions
» Binary comparisons
» Convert to Binary
» Convert to Decimal
» Multiplying Fullwords
» Binary Division
» Bit shifting instructions

Macros & Input/Output

» Macros
» File handling
» Building a control block
» The DCB
» DCB tied to File
» The DCB explained
» DCB parameters explained

BMC Mainframe: Using z/OS Assembler

COURSE ABSTRACT

- The OPEN macro
- The GET macro
- The PUT macro
- The CLOSE macro
- Printed output data sets
- Clearing the print line

Packed Decimal Arithmetic

- Summary of 'Decimal Manipulating Instructions'
- Summary of decimal instructions
- Packed Decimal
- Add Packed instruction
- SP and ZAP instructions
- Pack and Unpack instructions
- Move Zone Instruction
- Move Numeric Instruction
- Decimal Condition Codes
- Editing Packed
- Decimal Data
- Constructing patterns
- EDIT example
- EDMK instruction
- Packed Decimal multiplication
- Packed Decimal division

Translations and Bit Manipulations

- Summary of bit manipulating instructions
- 'OR'ing
- 'AND'ing
- Exclusive 'OR'ing
- Testing the bits
- Translate instruction
- Translation Tables + 'ORG'
- TRT- Translate and Test instruction

MVS Error Reporting, Dumps & Binder

- System error reporting
- MVS dumps
- Stand-Alone Dump (SADUMP)
- SVC Dumps
- User ABEND Dumps
- Generating a User ABEND Dump
- System Generated ABEND dump
- Snap dumps
- Symptom dumps
- Program Interruption Codes (PICs)
- Explanations of the Assembler program listings
- The External Symbol Dictionary (ESD)
- The Source and Object program listing
- The Relocation Dictionary
- The Symbol and Literal cross-reference (HLASM)
- The Diagnostic Cross Reference and Summary (HLASM)
- Options Summary (HLASM)
- Dump labs

24, 31 & 64 Bit Programming

- Impact of 31-bit addressing
- Addressing Memory - when and where?
- Extended Addressing
- 31-bit Virtual Addressing
- Dynamic Address Translation
- Addressing-mode control
- Instruction differences
- Different addressing
- Setting the mode when branching
- Branch-And-Save-and- Set-Mode (BASSM)
- Branch-and-Set-Mode (BSM)
- Coding examples
- The z/Architecture differences
- z/Architecture - the bottom line
- Tri- modal addressing mode
- Memory boundaries
- Registers in a z/Architecture machine
- 64-bit registers and instructions
- z/Architecture PSW
- The 64-bit virtual address
- Changing to 64-bit mode
- Setting the mode when branching
- Indicating 64-bit mode in pointers
- Branch-And- Save-and-Set-Mode (BASSM)
- A z/OS address space
- Characteristics of Memory Objects