## Table of Contents





# Notices

IBM and z/Architecture are registered trademarks of International Business Machines Corporation.

# Introduction

ASMA is somewhat behind in instruction implementation. This document describes the plan for the immediate future and generally going beyond that. Over time this document will be updated with the implementation status and next set of planned changes.

# ASMA Instruction Status

**In general, ASMA supports all machine instructions for all mainframe architectures starting with S/360 models through the systems defined by the SA22-7832-11 version of the *IBM® z/Architecture® Principles of Operation* released in September, 2017.**

Machine instructions are supported by ASMA for instructions defined by versions -11, -12, and -13 as described in the following three sections devoted to each version.

Extended Mnemonics are more restrictive. See “Extended Mnemonics” section for details on extended mnemonics requiring ASMA enhancements for support. Any missing extended mnemonic using only bits 8-15 should be reported as a bug.

Because ASMA instructions are defined by the MSL files (see the `asma/msl` directory), this is largely a plan for MSL. The last features implemented in `s390x-inst.msl` are those instructions added by the PoO manual SA22-7832-11, released in September, 2017.

The next version of the PoO manual was -12, released in September, 2019. The latest PoO manual, -13, was released in May, 2022. This makes ASMA instruction support three years behind. Most, but not all, new instructions are in the area of new floating point instructions and vector instructions. Of most interest by the users of ASMA are support for new instructions that are of more general use. “General use” includes privileged instructions for bare-metal programs.

This table documents the general status of ASMA instruction development for the -11, -12, and -13 PoO manuals by chapter. The “Type” column identifies whether the instructions defined by the chapter are available to all programs (general) or only programs executing in privileged state (privileged). The three status columns identify the status of the chapter contents within ASMA:

- Implemented – new instructions introduced by the chapter and implemented by ASMA,
- **Partial** – new instructions introduced by the chapter and only partially implemented by ASMA,
- **Implement** – new instructions introduced and ASMA does not yet support the new instructions, and
- No change – no new instructions were introduced by the chapter in the PoO manual version.

| Instructions | Type | Chapter | -11 Status | -12 Status | -13 Status |
|---|---|---|---|---|---|
| General | general | 7 | Implemented | Implemented | Implemented |
| Decimal | general | 8 | Implemented | No change | No change |
| FP Overview & Support | general | 9 | Implemented | No change | No change |
| Control | privileged | 10 | Implemented | Implemented | Implemented |
| I/O | privileged | 14 | Implemented | No change | No change |
| Hexadecimal FP | general | 18 | Implemented | No change | No change |

ASMA Instruction Status

| Instructions | Type | Chapter | -11 Status | -12 Status | -13 Status |
|---|---|---|---|---|---|
| Binary FP | general | 19 | Implemented | No change | No change |
| Decimal FP | general | 20 | Implemented | No change | No change |
| Vector Overview & Support | general | 21 | Implemented | **Implement** | No change |
| Vector Integer | general | 22 | Implemented | **Implement** | No change |
| Vector String | general | 23 | Implemented | **Implement** | No change |
| Vector FP | general | 24 | Implemented | **Implement** | No change |
| Vector Decimal | general | 25 | Implemented | No change | **Implement** |
| Specialized Function Assist | general | 26 | -- | -- | **Implement** |

Any instruction unsupported or incorrectly supported by ASMA introduced **prior to and including the -11 PoO** version, September, 2017, should be reported on the SATK `github` repository web site Issues page: https://github.com/s390guy/SATK.

Based upon the analysis of the remaining machine instructions requiring implementation, bringing ASMA current with the latest PoO version will proceed to completion. All chapters will eventually be shown as “Implemented” or “No change”. As reflected in the previous table, all of the required work to bring ASMA current with the latest PoO manual, -13, relates to vector instructions and the nine new instructions in Chapter 26.

Following implementation of all new instructions, the Machine Specification Language enhancements required for the new extended mnemonics using bits beyond 15 will be made. The **plan** for the actual extended mnemonics implementation will be addressed in the future. See the “Extended Mnemonics” section for base mnemonics present in Chapter 7 (-13) having defined extended mnemonics.

In the tables in the following sections, instructions, MSL formats, and programming notes in **bold** text require implementation. As implementation occurs, the bold text font will be changed to normal text font.

# Change History

| Change | Date | Description |
|---|---|---|
| 1 | 4 Sep 2022 | Initial release. |
| 2 | 5 Sep 2022 | -11 PoO fully supported with addition of two instructions that rename instructions released in the -10 PoO, chapter 24. |
| 3 | 6 Sep 2022 | Completed research for -12 and -13 PoO versions of chapters 7, 8, 10, and 14 instruction changes. Ready for implementation. |
| 4 | 8 Sep 2022 | Provided BIC extended mnemonic analysis. Implemented all instruction to current level (-13) for chapters 7, 8, 10, and 14. Added appendix describing issues and enhancements required for support of all extended mnemonics. |
| 5 | 13 Sep 2022 | Completed analysis of **all** instruction defining chapters required to bring ASMA **machine** instruction support current with the latest |

| Change | Date | Description |
|---|---|---|
| | | PoO (-13). |
| 6 | 14 Sep 2022 | Completed implementation of vector instructions missing from original release by PoO -10. All machine instructions prior to and including the -11 PoO are now supported by ASMA. Only some vector instructions and Chapter 26 still require work. |

# SA22-7832-11 – September, 2017

No new decimal instructions, Chapter 8, nor new I/O Instructions, Chapter 14, were added. Instructions that operate upon decimal floating point data, Chapter 20, and decimal data acted upon by vector instructions, Chapter 25 *were* added. Those additions have been implemented in ASMA.

Upon inspection of the actual MSL files, nearly all new instructions **WERE** already added to the MSL files for the -11 version of the PoO manual except for the mnemonic change for two instructions in Chapter 24.

During the analysis of floating point and vector instructions, ten vector instructions were found to be only partially implemented within the MSL. These ten were actually introduced in the -10 version of the PoO. By adding these ten instructions' mnemonics to the MSL `s390x-vector iset` statement (the statement defining the VECTOR FACILITY), the instructions were made visible to the assembler for z/Architecture target assemblies and those targets including all instructions.

By incorporating the mnemonic name changes, ASMA is current with the -11 version of the PoO manual for all -11 and previous machine instructions.

At this point ASMA is now requiring enhancements for -12, released in September, 2019 and -13, released in May, 2022. ASMA is now three years behind.

## General Instructions – Chapter 7

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Other Notes |
|---|---|---|---|---|---|
| AGH | 7-28 | 31 | RXYA | yes | |
| BIC | 7-39 | 31 | RXYB | yes | See BIC EM table below |
| KMA | 7-78 | 33 | RRFB3 | yes | |
| CLT | 7-155 | 26 | RSYB | yes | |
| CLGT | 7-155 | 26 | RSYB | yes | |
| LGG | 7-274 | 32 | RXYA | yes | |
| LGSC | 7-275 | 32 | RXYA | yes | |
| LLGFSG | 7-274 | 32 | RXYA | yes | |
| MG | 7-304 | 31 | RXYA | yes | |
| MGRK | 7-304 | 31 | RRFA1 | yes | |
| MGH | 7-305 | 31 | RXYA | yes | |
| MSC | 7-307 | 31 | RXYA | yes | |
| MSRKC | 7-307 | 31 | RRFA1 | yes | |
| MSGC | 7-307 | 31 | RXYA | yes | |
| MSGRKC | 7-307 | 31 | RRFA1 | yes | |
| PRNO | 7-346 | 29 | RRE | yes | |
| RISBGN | 7-363 | 26 | RIEF | yes | |
| STGSC | 7-383 | 32 | RXYA | yes | |
| SGH | 7-388 | 31 | RXYA | yes | |

## BIC Extended Mnemonics (EM)

The BIC instruction utilizes an extended operation code in bits 40-47. Because current extended mnemonic support utilizes the extended operation in bits 8-11, it is not possible to define the BIC extended mnemonics by overloading the extended operation field with the fixed mask. The new support for extended mnemonics is required for the BIC instruction.

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **BIO** | 7-40 | | **?** | M1 = 1 | |
| **BIP** | 7-40 | | **?** | M1 = 2 | |
| **BIH** | 7-40 | | **?** | M1 = 2 | |
| **BIM** | 7-40 | | **?** | M1 = 4 | |
| **BIL** | 7-40 | | **?** | M1 = 4 | |
| **BINZ** | 7-40 | | **?** | M1 = 7 | |
| **BINE** | 7-40 | | **?** | M1 = 7 | |
| **BIZ** | 7-40 | | **?** | M1 = 8 | |
| **BIE** | 7-40 | | **?** | M1 = 8 | |
| **BINM** | 7-40 | | **?** | M1 = 11 or B | |
| **BINL** | 7-40 | | **?** | M1 = 11 or B | |
| **BINP** | 7-40 | | **?** | M1 = 13 or D | |
| **BINH** | 7-40 | | **?** | M1 = 13 or D | |
| **BINO** | 7-40 | | **?** | M1 = 14 or E | |
| **BI** | 7-40 | | **?** | M1 = 15 or F | |

### Programming Notes

| Note # | Description |
|---|---|
| 26 | Miscellaneous-Instruction Extensions Facility 1 |
| 29 | Message-Security-Assist Extension 5 |
| 31 | Miscellaneous-Instruction Extensions Facility 2 |
| 32 | Guarded Storage Facility |
| 33 | Message-Security-Assist Extension 8 |

# Control Instructions – Chapter 10

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| IRBM | 10-30 | 11 | RRE | yes | |
| TPEI | 10-169 | 12 | RRE | yes | |

### Programming Notes

| Note # | Description |
|---|---|
| 11 | Insert-Reference-Bits-Multiple Facility |
| 12 | Test Pending External Interruption Facility |

# Vector Overview and Support Instructions – Chapter 21

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| VLEF | 21-7 | A | VRX | yes | |
| VLEG | 21-7 | A | VRX | yes | |
| VLBB | 21-10 | A | VRX | yes | |
| VPKLS | 21-14 | A | VRRB | yes | |
| VSEL | 21-17 | A | VRRE1 | yes | |

### Programming Notes

| Note # | Description |
|---|---|
| A | Vector Facility introduced in PoO version -10 |

# Vector Integer Instruction – Chapter 22

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| VNC | 22-5 | A | VRRC2 | yes | |
| VCH | 22-8 | A | VRRB | yes | |
| VCHL | 22-9 | A | VRRB | yes | |

### Programming Notes

| Note # | Description |
|---|---|
| A | Vector Facility introduced in PoO version -10 |

# Vector Floating Point Instructions – Chapter 24

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| VFLL | 24-25 | | VRRA4 | yes | Changed Mnemonic from VLDE |
| VFLR | 24-26 | | VRRA | yes | Changed Mnemonic from VLED |

# SA22-7832-12 – September, 2019

## General Instructions – Chapter 7

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| NCRK | 7-34 | 34 | RRFA1 | yes | |
| NCGRK | 7-34 | 34 | RRFA1 | yes | |
| MVCRL | 7-300 | 34 | SSE2 | yes | |
| NNRK | 7-308 | 34 | RRFA1 | yes | |
| NNGRK | 7-308 | 34 | RRFA1 | yes | |
| NORK | 7-311 | 34 | RRFA1 | yes | |
| NOGRK | 7-311 | 34 | RRFA1 | yes | |
| NXRK | 7-311 | 34 | RRFA1 | yes | |
| NXGRK | 7-311 | 34 | RRFA1 | yes | |
| OCRK | 7-314 | 34 | RRFA1 | yes | |
| OCGRK | 7-314 | 34 | RRFA1 | yes | |
| POPCNT | 7-365 | 34 | RRFC | yes | |
| SELR | 7-376 | 34 | RRFA2 | yes | |
| SELGR | 7-376 | 34 | RRFA2 | yes | |
| SELFHR | 7-376 | 34 | RRFA2 | yes | |

### Programming Notes

| Note # | Description |
|---|---|
| 34 | Miscellaneous-Instruction-Extension Facility 3 |

## Vector Overview and Support Instructions – Chapter 21

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VLEBRH** | 21-7 | unnumbered | VRX | no | |
| **VLEBRF** | 21-7 | unnumbered | VRX | no | |
| **VLEBRG** | 21-7 | unnumbered | VRX | no | |
| **VLBRREP** | 21-8 | unnumbered | VRX | no | |
| **VLLEBRZ** | 21-8 | unnumbered | VRX | no | |
| **VLBR** | 21-9 | unnumbered | VRX | no | |
| **VLER** | 21-10 | unnumbered | VRX | no | |
| **VSTEBRH** | 21-22 | unnumbered | VRX | no | |
| **VSTEBRF** | 21-22 | unnumbered | VRX | no | |
| **VSTEBRG** | 21-22 | unnumbered | VRX | no | |
| **VSTBR** | 21-22 | unnumbered | VRX | no | |
| **VSTER** | 21-24 | unnumbered | VRX | no | |

### Programming Notes

| Note # | Description |
|---|---|
| unnumbered | **Vector-Enhancements Facility 2** |

# Vector Integer Instructions – Chapter 22

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VSLD** | 22-25 | unnumbered | VRID1 | no | |
| **VSRD** | 22-26 | unnumbered | VRID1 | no | |

**Programming Notes**

| Note # | Description |
|---|---|
| unnumbered | **Vector-Enhancements Facility 2** |

# Vector String Instructions – Chapter 23

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VSTRS** | 23-8 | 2 | VRRD | no | |

**Programming Notes**

| Note # | Description |
|---|---|
| 2 | **Vector-Enhancements Facility 2** |

# Vector Floating Point Instructions – Chapter 24

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VCFPS** | 24-15 | unnumbered | VRRA | no | Rename VCDG |
| **VCFPL** | 24-17 | unnumbered | VRRA | no | Rename VCDLG |
| **VCSFP** | 24-18 | unnumbered | VRRA | no | Rename VCGD |
| **VCLFP** | 24-20 | unnumbered | VRRA | no | Rename VCLDG |

**Programming Notes**

| Note # | Description |
|---|---|
| unnumbered | **Vector-Enhancements Facility 2** |

# SA22-7832-13 – May, 2022

## Control Instructions – Chapter 10

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| LBEAR | 10-51 | 13 | S0 (zero) | yes | |
| LPSWEY | 10-57 | 13 | SIY0 (zero) | yes | |
| QPACI | 10-123 | 15 | S0 (zero) | yes | |
| RDP | 10-124 | 14 | RRFB2 | yes | |
| STBEAR | 10-145 | 13 | S0 (zero) | yes | |

### Programming Notes

| Note # | Description |
|---|---|
| 13 | BEAR-Enhancement Facility |
| 14 | Reset DAT-Protection Facility |
| 15 | Processor-Activity-Instrumentation Facility |

## Vector Decimal Instructions – Chapter 25

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VSCSFP** | 25-4 | 2 | VRRC | no | |
| **VSCHP** | 25-5 | 2 | VRRB | no | |
| **VCSPH** | 25-11 | 2 | VRRC | no | |
| **VCLZDP** | 25-11 | 2 | VRRI | no | |
| **VPKZR** | 25-18 | 2 | VRIF | no | |
| **VSRPR** | 25-26 | 2 | VRIF | no | |
| **VUPKZH** | 25-30 | 2 | **VRRK** | no | VRRK in `formats.msl` but not committed |
| **VUPKZL** | 25-31 | 2 | **VRRK** | no | VRRK in `formats.msl` but not committed |

### Programming Notes

| Note # | Description |
|---|---|
| **2** | **Vector-Packed-Decimal-Enhancement Facility 2** |

## Specialized-Function-Assist Instructions – Chapter 26

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **KDSA** | 26-2 | 1 | RRE | no | |
| **DFLTCC** | 26-17 | 2 | RRFA1 | no | |
| **NNPA** | 26-61 | 4 | RRE1 | no | |
| **SORTL** | 26-96 | 3 | RRE | no | |
| **VCLFNH** | 26-121 | 4 | VRRA4 | no | |
| **VCLFNL** | 26-122 | 4 | VRRA4 | no | |
| **VCRNF** | 26-123 | 4 | VRRC3 | no | |

| Mnemonic | Page # | Prog. Notes | MSL Format | Implemented | Notes |
|---|---|---|---|---|---|
| **VCFN** | 26-123 | 4 | VRRC3 | no | |
| **VCNF** | 26-124 | 4 | VRRA4 | no | |

## Programming Notes

| Note # | Description |
|---|---|
| 1 | **Message-Security-Assist Extension 9** |
| 2 | **DEFLATE-Conversion Facility** |
| 3 | **Enhanced-Sort Facility** |
| 4 | **Neural-Network-Processing-Assist Facility** |

# Extended Mnemonics

This section documents the current status of extended mnemonics in ASMA. Support for the base machine instruction mnemonics are described in the preceding sections.

Refer to SA22-7832-13, Appendix J, for details. The following table only reflects the base mnemonic for which extended mnemonics are defined. Page numbers are for the -13 version.

All of the extended mnemonics requiring the MSL enhancements were defined by SA22-7832-04 or later versions and are *not* presently supported by ASMA.

All extended mnemonics up to and including ESA/390 either separate from z/Architecture or ESA/390 on a z/Architecture system *are* supported by ASMA. While ESA/390 continued to be available on z/Architecture until the release of the CZAM facility, ESA/390 ceased being enhanced on z/Architecture beyond PoO -02.

## General Instructions – Chapter 7

| Base Mnemonic | Page # | MSL Format | Base ASMA Target | MSL | Notes |
|---|---|---|---|---|---|
| BCR | 7-40 | | s360 (yes) | | |
| BC | 7-40 | | s360 (yes) | | |
| BRAS | 7-45 | | e390 (yes) | | |
| BRC | 7-46 | | e390 (yes) | | |
| BRCT | 7-47 | | e390 (yes) | | |
| BRXH | 7-48 | | e390 (yes) | | |
| BRXLE | 7-48 | | e390 (yes) | | |
| BRASL | 7-45 | | s390 (yes) | | |
| BRCL | 7-46 | | s390 (yes) | | |
| BRASL | 7-45 | | s390 (yes) | | |
| BRCL | 7-46 | | s390 (yes) | | |
| BRCTG | 7-47 | | s390x-00 (yes) | | |
| BRCTH | 7-47 | | s390x-00 (yes) | | |
| BRXHG | 7-48 | | s390x-00 (yes) | | |
| BRXLG | 7-48 | | s390x-00 (yes) | | |
| IILF | 7-266 | | s390x-04 | | |
| LLILF | 7-284 | | s390x-04 | | |
| CRB | 7-137 | | s390x-06 | | |
| CGRB | 7-137 | | s390x-06 | | |
| CRJ | 7-137 | | s390x-06 | | |
| CGRJ | 7-137 | | s390x-06 | | |
| CRT | 7-150 | | s390x-06 | | |
| CGRT | 7-150 | | s390x-06 | | |
| CIB | 7-137 | | s390x-06 | | |

# ASMA Instruction Status

| Base Mnemonic | Page # | MSL Format | Base ASMA Target | MSL | Notes |
|---|---|---|---|---|---|
| CGIB | 7-137 | | s390x-06 | | |
| CIJ | 7-137 | | s390x-06 | | |
| CGIJ | 7-137 | | s390x-06 | | |
| CIT | 7-150 | | s390x-06 | | |
| CGIT | 7-150 | | s390x-06 | | |
| CLRB | 7-155 | | s390x-06 | | |
| CLGRB | 7-155 | | s390x-06 | | |
| CLRJ | 7-155 | | s390x-06 | | |
| CLGRJ | 7-155 | | s390x-06 | | |
| CLRT | 7-156 | | s390x-06 | | |
| CLGRT | 7-156 | | s390x-06 | | |
| CLIB | 7-155 | | s390x-06 | | |
| CLGIB | 7-155 | | s390x-06 | | |
| CLIJ | 7-155 | | s390x-06 | | |
| CLGIJ | 7-155 | | s390x-06 | | |
| CLFIT | 7-157 | | s390x-06 | | |
| CLGIT | 7-157 | | s390x-06 | | |
| RNSBG | 7-372 | | s390x-06 | | |
| **RNSBGT** | 7-373 | | s390x-06 | | Treat this base as an extended mnemonic |
| RXSGB | 7-372 | | s390x-06 | | |
| RISBG | 7-374 | | s390x-06 | | |
| ROSBG | 7-372 | | s390x-06 | | |
| BRCTH | 7-47 | | s390x-08 | | |
| LOCR | 7-287 | | s390x-08 | | |
| LOCGR | 7-287 | | s390x-08 | | |
| LOC | 7-287 | | s390x-08 | | |
| LOCG | 7-287 | | s390x-08 | | |
| CLT | 7-156 | | s390x-08 | | |
| RISBHG | 7-374 | | s390x-08 | | |
| **RISBHGZ** | 7-375 | | s390x-08 | | Treat this base as an extended mnemonic |
| RISBLG | 7-375 | | s390x-08 | | |
| **RISBLGZ** | 7-375 | | s390x-08 | | Treat this base as an extended mnemonic |
| STOC | 7-397 | | s390x-08 | | |
| STOCG | 7-397 | | s390x-08 | | |
| CLGT | 7-156 | | s390x-09 | | |
| RISBGN | 7-374 | | s390x-09 | | |
| LOCHHI | 7-280 | | s390x-10 | | |
| LOCHI | 7-280 | | s390x-10 | | |
| LOCGHI | 7-280 | | s390x-10 | | |
| LOCFHR | 7-287 | | s390x-10 | | |
| LOCFH | 7-287 | | s390x-10 | | |
| STOCFH | 7-397 | | s390x-10 | | |
| BIC | 7-39 | | s390x-11 | | |
| NORK | 7-314 | | s390x-12 | | |
| NOGRK | 7-314 | | s390x-12 | | |

# ASMA Instruction Status

| Base Mnemonic | Page # | MSL Format | Base ASMA Target | MSL | Notes |
|---|---|---|---|---|---|
| SELR | 7-380 | | s390x-12 | | |
| SELGR | 7-380 | | s390x-12 | | |
| SELFHR | 7-380 | | s390x-12 | | |

# Appendix A – Extended Mnemonic Limitation

This appendix explains the current (2022) situation with regards to extended mnemonics and in general terms what enhances are needed within MSL and where those enhancements affect the rest of ASMA.

Extended mnemonics when ASMA was under active development was limited to use of the mask field (bits 8-11) in various branch-type instructions. At the same time, there were a few instructions that utilized some or all of bits 8-15 for an extended operation field. Extended operation fields are used for operation codes in excess of eight bits. This influenced the design of the Machine Specification Language (MSL). MSL was from the start created to support an extended operation field, referred to as the `XOP` field in MSL formats.

Lacking multi-year imagination, the implementation decision was made to overload the XOP field as the implied mask for extended mnemonics. XOP at the time was a good decision. The contents of the XOP field is part of the instruction definition, not the format. So an extended mnemonic does, while requiring a specific format for the source fields in the assembly, can be shared with all instructions using bits 8-11 for the mask.

In this paradigm, an instruction can define bits in 8-11, or 8-15 as part of an extended mnemonic (the `BC` mask field) or an extended operation code (`SAMx` instructions), but not both.

Fast forward a decade and a half and ASMA is faced with a different situation. For one, an instruction may have an extended operation code field in bits other than 8-15. As it turns out MSL from the beginning supported that structure. However, some instructions now have extended mnemonics that use bits 8-11 for a mask **and** an extended operation code. The BIC instruction is a case in point. The overloading of the function of XOP precludes definition of extended mnemonics for this instruction.

Additionally, for a number of complex instructions, fields outside of bits 8-11 are used for the extended mnemonic. MSL has no way to describe what is to be placed in those extended mnemonic fields. It is this enhancement that is required of the MSL for these new extended mnemonics.

This enhancement touches three main portions of ASMA:

- MSL itself (`doc/asma/MSL.odt` and `doc/asma/MSL.pdf`), the
- MSL database built by ASMA at program start (`asma/msldb.py`), and the
- instruction builder (`asma/insnbldr.py`) to fill the extended mnemonic fields for the instruction.

The database creation is really conversion of the MSL text files into Python constructs. `msldb.py` is essentially the syntax analyzer of the MSL. The Python constructs are passed to the instruction builder to fill in the values of the instruction fields. It identifies the source of for the machine instruction. In the context of a language, `insnbldr.py` performs semantic processing of MSL.

Two MSL statements participate in instruction definition: the

- `inst` statement, and the
- `format` statement.

`inst` defines which format is used by the instruction and machine instruction constant content, such as the operation code or extended operation code.

`format` maps the instruction's source operands to the destination field within the machine instruction into which the operand value is inserted.

A new extended mnemonic format is required for those instructions that have extended mnemonics. This is because the instruction's source syntax changes as a result of the extended mnemonic.

However, the values used by each instruction in the extended mnemonic is instruction specific and should appear within the individual extended mnemonic's `inst` statement, as is the case for the `XOP` field content currently supported by MSL.

Additionally, certain extended mnemonics effect specific bits in certain ways, and some bits should be ignored. A mechanism for support of these individual cases require addition to MSL. And example is the `ROTATE THEN…` instructions. Others may appear. In the general case a special routine within ASMA must be triggered by the MSL to handle these cases.

It should be noted that a number of MSL parameters as they appear in the text file and are placed within the database are tightly coupled with operand syntax parsing and the instruction builder. Arbitrary changes to source operand names or machine instruction fields should not be made.

Once these enhancements have been made, it will be possible to define the missing extended mnemonics within the MSL files and assemble them.

## Implementation and Testing Considerations

The above is written as though the MSL language can be enhanced separate from instruction definition and implementation by ASMA. The realities of implementation and testing is that, while MSL changes must precede instruction definition, testing can not occur without all three

components being enhanced. Yes, `mslrpt.py` allows testing of the MSL language. However, instructions that utilize the MSL enhancements and that can then be successfully assembled (`insnbldr.py`) is the real test. Those tests may result in changes to the MSL language or adjustments to `msldb.py` or `insnbldr.py` to achieve successful assembly.

In as much as the BIC instruction's extended mnemonics are already defined, testing will probably proceed with these extended mnemonics.