Table of Contents

Notices

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.

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.

Extended Mnemonics are more restrictive. See below where within the plan extended mnemonics using any bits of the instruction beyond bit 15 are addressed. 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 instruction development for the -11, -12, and -13 PoO manuals by chapter.

Instructions	Chapter	-11 Status	-12 Status	-13 Status
General	7	Implemented	Implementing	No change
Decimal	8	No change	No change	No change
Control	10	Implemented	No change	Implementing
I/O	14	No change	No change	No change
Vector FP	24	Implemented	?	?

Following implementation of the new instructions, the Machine Specification Language enhancements required for many of the new extended mnemonics using bits beyond 15 will be made. The **plan** for the actual extended mnemonics and the implementation of the new floating point and vector instructions will be addressed in the future.

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.

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.

New vector instructions were added as documented in Chapters 21-25.

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.

By incorporating the mnemonic name changes, ASMA is current with the -11 version of the PoO manual for 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 extend 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 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

No new instructions were added to Chapters 8, 10, or 14.

Chapters 17-26 require future research.

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

SA22-7832-13 – May, 2022

No new instructions were added to chapters 7, 8, or 14.

Chapters 17-26 require future research.

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

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.

ASMA Instruction Status

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.