z/Architecture

Reference Summary

SA22-7871-11

z/Architecture

Reference Summary

SA22-7871-11

Twelfth Edition (May, 2022)

This revision differs from the previous edition by containing instructions related to the facilities marked by a bar under “Facility” in “Preface” and minor corrections and clarifications. Changes are indicated by a bar in the margin.

References in this publication to IBM® products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM’s program product may be used. Any functionally equivalent program may be used instead.

Additional copies of this and other IBM publications may be ordered or downloaded from the IBM publications web site at http://www.ibm.com/support/documentation.

Please direct any comments on the contents of this publication to:

Internet e-mail: mhvrcfs@us.ibm.com

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication is intended primarily for use by z/Architecture™ assembler-language application programmers. It contains basic machine information summarized from the *IBM z/Architecture Principles of Operation* (SA22-7832), about the IBM Z™ processors. It also contains frequently used information from *IBM ESA/390 Common I/O-Device Commands and Self Description* (SA22-7204), *IBM System/370 Extended Architecture Interpretive Execution* (SA22-7095), *The Load-Program-Parameter and the CPU-Measurement Facilities* (SC23-2260), and *IBM High Level Assembler for z/OS, z/VM & z/VSE Language Reference* (SC26-4940). This publication will be updated from time to time. However, the above publications and others cited in this publication are the authoritative reference sources and will be first to reflect changes.

The following instructions may be uninstalled or not available on a particular model:

Facility	Instruction
ASN-and-LX-reuse	EPAIR, ESAIR, PTI, SSAIR
BEAR-enhancement	LBEAR, LPSWEY, STBEAR
Compare-and-swap-and-store	CSST
Configuration-topology	PTF
Constrained-transactional-execution	TBEGINC
DAT-enhancement 1	CSPG, IDTE
DAT-enhancement 2	LPTEA
Decimal-floating-point	ADTR, AXTR, CDGTR, CDSTR, CDTR, CDUTR, CEDTR, CEXTR, CGDTR, CGXTR, CSDTR, CSXTR, CUDTR, CUXTR, CXGTR, CXSTR, CXTR, CXUTR, DDTR, DXTR, EEDTR, EEXTR, ESDTR, ESXTR, FIDTR, FIXTR, IEDTR, IEXTR, KDTR, KXTR, LDETR, LDXTR, LEDTR, LTDTR, LTXTR, LXDTR, MDTR, MXTR, QADTR, QAXTR, RRDTR, RRXTR, SDTR, SLDT, SLXT, SRDT, SRXT, SXTR, TDCDT, TDCET, TDCXT, TDGDT, TDGET, TDGXT
DEFLATE-conversion	DFLTCC
DFP-rounding	SRNMT
DFP-packed-conversion	CDPT, CPDT, CPXT, CXPT
DFP-zoned-conversion	CDZT, CXZT, CZDT, CZXT
Distinct-operands	AGHIK, AGRK, AHIK, ALGHSIK, ALGRK, ALHSIK, ALRK, ARK, NGRK, NRK, OGRK, ORK, SGRK, SLAK, SLGRK, SLLK, SLRK, SRAK, SRK, SRLK, XGRK, XRK
Enhanced-DAT 1	PFMF
Enhanced-DAT 2	CRDTE
Enhanced-sort	SORTL
Execute-extensions	EXRL
Execution-hint	BPP, BPRP, NIAI
Expanded-storage	PGIN, PGOUT
Extended-immediate	AFI, AGFI, ALFI, ALGFI, CFI, CGFI, CLFI, CLGFI, FLOGR, IIHF, IILF, LBR, LGBR, LGHR, LGFI, LHR, LLC, LLCR, LLGCR, LLGHR, LLH, LLHR, LLIHF, LLILF, LT, LTG, NIHF, NILF, OIHF, OILF, SLFI, SLGFI, XIHF, XILF
Extended-translation 2	CLCLU, MVCLU, PKA, PKU, TP, TROO, TROT, TRTO, TRTT, UNPKA, UNPKU
Extended-translation 3	CU14, CU24, CU41, CU42, SRSTU, TRTR
Extract-CPU-time	ECTG
Floating-point-extension	ADTRA, AXTRA, CDFBRA, CDFTR, CDGBRA, CDGTRA, CDLFBR, CDLFTR, CDLGBR, CDLGTR, CEFBRA, CEGBRA, CELFBR, CELGBR, CFDBRA, CFDTR, CFEBRA, CFXBRA, CFXTR, CGDBRA, CGDTRA, CGEBRA, CGXBRA, CGXTRA, CLFDBR, CLFDTR, CLFEBR, CLFXBR, CLFXTR, CLGDBR, CLGDTR, CLGEBR, CLGXBR, CLGXTR, CXFBRA, CXFTR, CXGBRA, CXGTRA, CXLFBR, CXLFTR, CXLGBR, CXLGTR, DDTRA, DXTRA, FIDBRA, FIEBRA, FIXBRA, LDXBRA, LEDBRA, LEXBRA, MDTRA, MXTRA, SDTRA, SRNMB, SXTRA
Floating-point-support-sign-handling	CPSDR, LCDFR, LNDFR, LPDFR

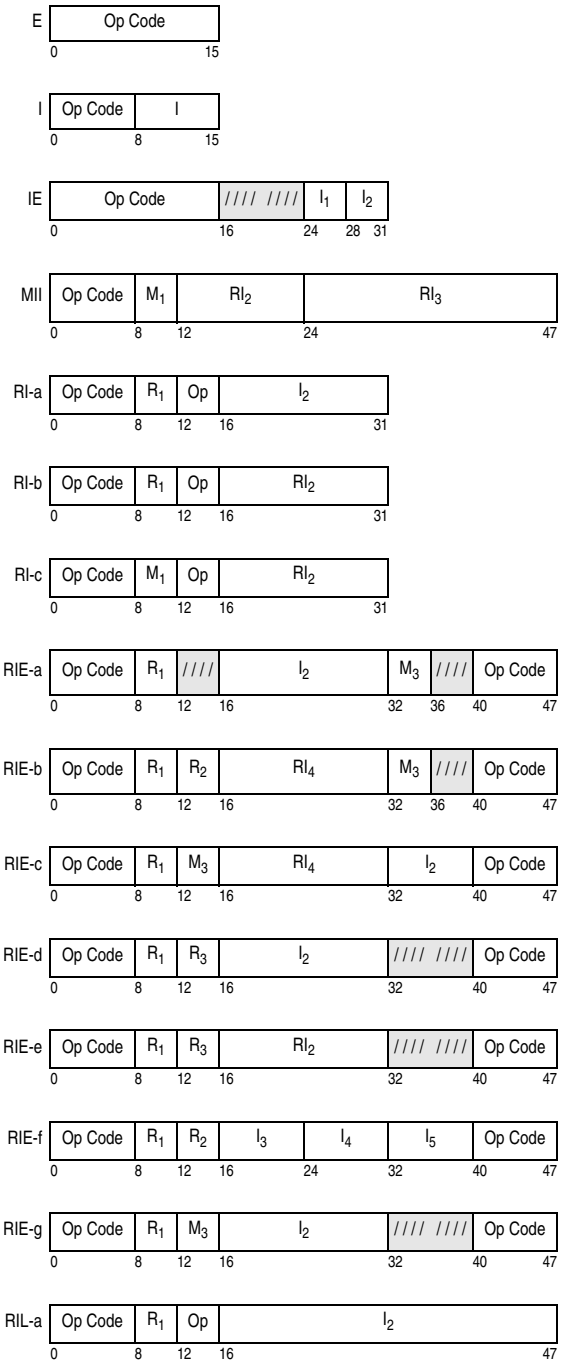
Facility	Instruction
FPR-GR-transfer	LDGR, LGDR
General-instructions-extension	ASI, AGSI, ALSI, ALGSI, CRB, CGRB, CRJ, CGRJ, CRT, CGRT, CGH, CHHSI, CHSI, CGHSI, CHRL, CGHRL, CIB, CGIB, CIJ, CGIJ, CIT, CGIT, CLRB, CLGRB, CLRJ, CLGRJ, CLRT, CLGRT, CLHHSI, CLFHSI, CLGHSI, CLIB, CLGIB, CLIJ, CLGIJ, CLFIT, CLGIT, CLRL, CLHRL, CLGRL, CLGHRL, CLGFRL, CRL, CGRL, CGFRL, ECAG, LAEY, LTGF, LHRL, LGHRL, LLHRL, LLGHRL, LLGFRL, LRL, LGRL, LGFRL, MVHHI, MVHI, MVGHI, MFY, MHY, MSFI, MSGFI, PFD, PFDRL, RNSBG, RXSBG, RISBG, ROSBG, STHRL, STRL, STGRL
Guarded storage	LGG, LGSC, LLGFSG, STGSC
HFP-multiply-and-add/subtract	MAD, MADR, MAE, MAER, MSD, MSDR, MSE, MSER
HFP-unnormalized extensions	MAY, MAYR, MAYH, MAYHR, MAYL, MAYLR, MY, MYH, MYL, MYR, MYHR, MYLR
High-word	AHHHR, AHHLR, AIH, ALHHHR, ALHHLR, ALSIH, ALSIHN, BRCTH, CHF, CHHR, CHLR, CIH, CLHF, CLHHR, CLHLR, CLIH, LBH, LHH, LFH, LLCH, LLHH, RISBHG, RISBLG, SHHHR, SHHLR, SLHHHR, SLHHLR, STCH, STHH, STFH
IEEE-exception-simulation	LFAS, SFASR
Insert-reference-bits-multiple facility	IRBM
Interlocked-access	LAA, LAAG, LAAL, LAALG, LAN, LANG, LAO, LAOG, LAX, LAXG, LPD, LPDG
Load-and-trap	LAT, LFHAT, LGAT, LLGFAT, LLGTAT
Load-and-zero-rightmost-byte	LLZRGF, LZRF, LZRG
Load/store-on-condition facility 1	LOC, LOCG, LOCGR, LOCR, STOC, STOCG
Load/store-on-condition facility 2	LOCFH, LOCFHR, LOCGHI, LOCHHI, LOCHI, STOCFH
Long displacement	AHY, ALY, AY, CDSY, CHY, CLIY, CLMY, CLY, CSY, CVBY, CVDY, CY, ICMY, ICY, LAMY, LAY, LB, LDY, LEY, LGB, LHY, LMY, LRAY, LY, MSY, MVIY, NIY, NY, OIY, OY, SHY, SLY, STAMY, STCMY, STCY, STDY, STEY, STHY, STMY, STY, SY, TMY, XIY, XY
Message-security-assist	KM, KMC, KIMD, KLMD, KMAC
Message-security-assist extension 3	PCKMO
Message-security-assist extension 4	KMCTR, KMF, KMO, PCC
Message-security-assist extension 5	PRNO
Message-security-assist extension 8	KMA
Message-security-assist extension 9	KDSA
Miscellaneous-instruction-extensions 1	CLT, CLGT, RISBGN
Miscellaneous-instruction-extensions 2	AGH, BIC, MG, MGH, MGRK, MSC, MSGC, MSGRKC, MSRKC, SGH
Miscellaneous-instruction-extensions 3	MVCRL, NCGRK, NCRK, NNGRK, NNRK, NOGRK, NORK, NXGRK, NXRK, OCGRK, OCRK, SELFHR, SELGR, SELR
Move-with-optional-specifications	MVCOS
Neural-network-processing-assist	NNPA, VCFN, VCLFNH, VCLFNL, VCNF, VCRNF
Parsing-enhancement	TRTE, TRTRE
Perform-floating-point-operation	PFPO
Population-count	POPCNT
Processor-activity-instrumentation	QPACI
Processor-assist	PPA
Reset-DAT-protection	RDP
Reset-reference-bits-multiple	RRBM
Store-clock fast	STCKF
Store-facility-list extended	STFLE
TOD-clock steering	PTFF
Transactional-execution	ETND, NTSTG, TABORT, TBEGIN, TEND
Test-pending-external-interruption	TPEI

Facility	Instruction
Vector-facility-for-z/Architecture	LCBB, VA, VAC, VACC, VACCC, VAVG, VAVGL, VCDG, VCDLG, VCEQ, VCGD, VCH, VCHL, VCKSM, VCLGD, VCLZ, VCTZ, VEC, VECL, VERIM, VERLL, VERLLV, VESL, VESLV, VESRA, VESRAV, VESRL, VESRLV, VFA, VFAE, VFCE, VFCH, VFCHE, VFD, VFEE, VFENE, VFI, VFLL, VFLR, VFM, VFMA, VFMS, VFPSO, VFS, VFSQ, VFTCI, VGBM, VGEF, VGEG, VGFM, VGFMA, VGM, VISTR, VL, VLBB, VLC, VLEB, VLEF, VLEG, VLEH, VLEIB, VLEIF, VLEIG, VLEIH, VLGV, VLL, VLLEZ, VLM, VLP, VLR, VLREP, VLVG, VLVGP, VMAE, VMAH, VMAL, VMALE, VMALH, VMALO, VMAO, VME, VMH, VML, VMLE, VMLH, VMLO, VMN, VMNL, VMO, VMRH, VMRL, VMX, VMXL, VN, VNC, VNO, VO, VPDI, VPERM, VPK, VPKLS, VPKS, VPOPCT, VREP, VREPI, VS, VSBCBI, VSBI, VSCBI, VSCEF, VSCEG, VSEG, VSEL, VSL, VSLB, VSLDB, VSRA, VSRAB, VSRL, VSRLB, VST, VSTEB, VSTEF, VSTEG, VSTEH, VSTL, VSTM, VSTRC, VSUM, VSUMG, VSUMQ, VTM, VUPH, VUPL, VUPLH, VUPLL, VX, WFC, WFK
Vector-enhancements facility 1	VBPERM, VFMAX, VFMIN, VFNMA, VFNMS, VMSL, VNN, VNX, VOC
Vector-enhancements facility 2	VLBR, VLBRREP, VLEBRF, VLEBRG, VLEBRH, VLER, VLLEBRZ, VSLD, VSRD, VSTBR, VSTEBRF, VSTEBRG, VSTEBRH, VSTER, VSTRS
Vector-packed-decimal	VAP, VCP, VCVB, VCVBG, VCVD, VCVDG, VDP, VLIP, VMP, VMSP, VPKZ, VPSOP, VRP, VSDP, VSRP, VSP, VTP, VUPKZ, VLRLR, VLRL, VSTRLR, VSTRL
Vector-packed-decimal-enhancement 2	VCLZDP, VCSPH, VPKZR, VSCHP, VSCSHP, VSRPR, VUPKZH, VUPKZL

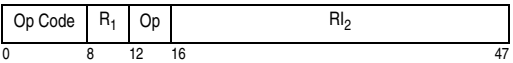
For information about Enterprise Systems Architecture/390® (ESA/390™) architecture, refer to *IBM Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201, and *IBM Enterprise Systems Architecture/390 Reference Summary*, SA22-7209.

Contents

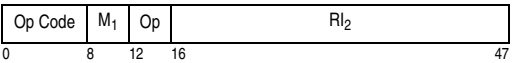
Machine Instruction Formats



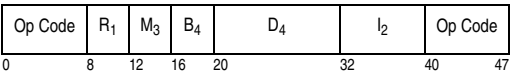
RIL-b



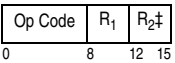
RIL-c



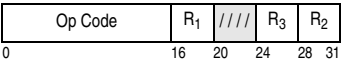
RIS



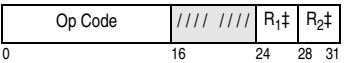
RR



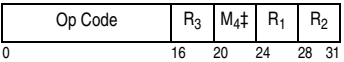
RRD



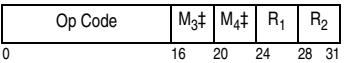
RRE



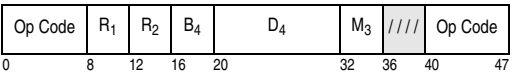
RRF-a,b



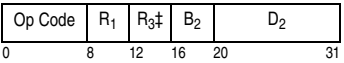
RRF-c-e



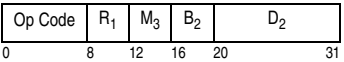
RRS



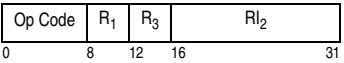
RS-a



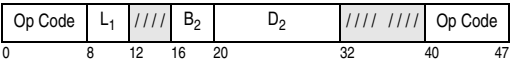
RS-b



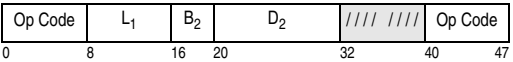
RSI



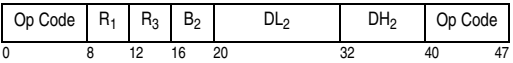
RSL-a



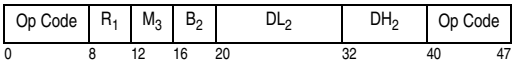
RSL-b



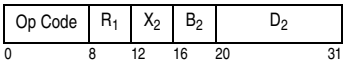
RSY-a



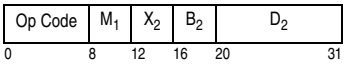
RSY-b



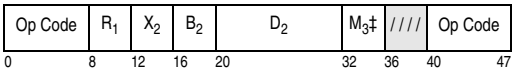
RX-a



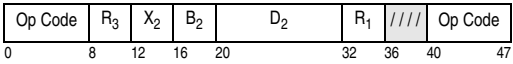
RX-b



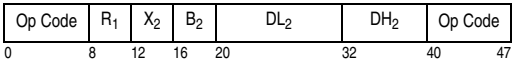
RXE



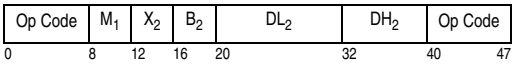
RXF



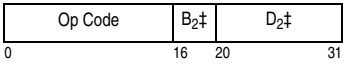
RXY-a



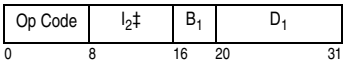
RXY-b



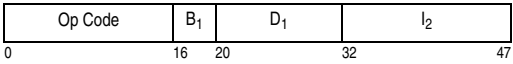
S



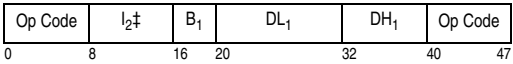
SI



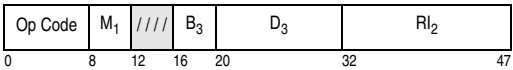
SIL



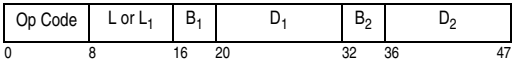
SIY



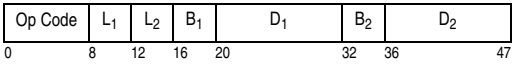
SMI



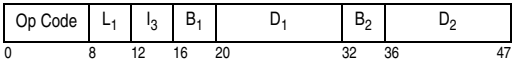
SS-a



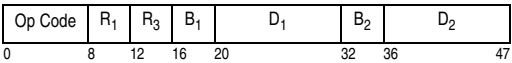
SS-b



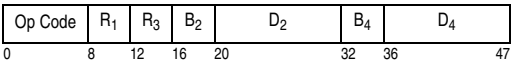
SS-c



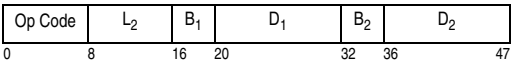
SS-d



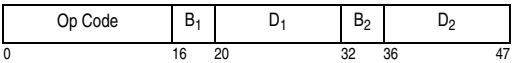
SS-e



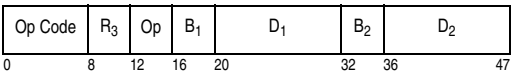
SS-f



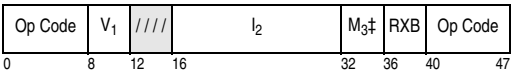
SSE



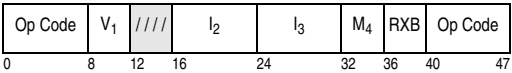
SSF



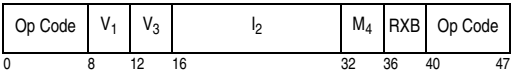
VRI-a



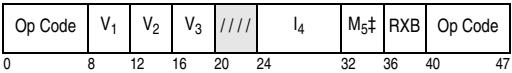
VRI-b



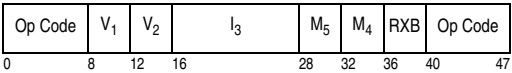
VRI-c



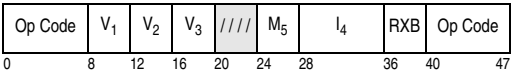
VRI-d



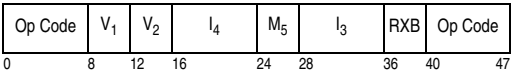
VRI-e



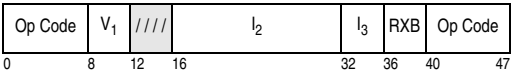
VRI-f



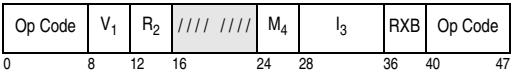
VRI-g

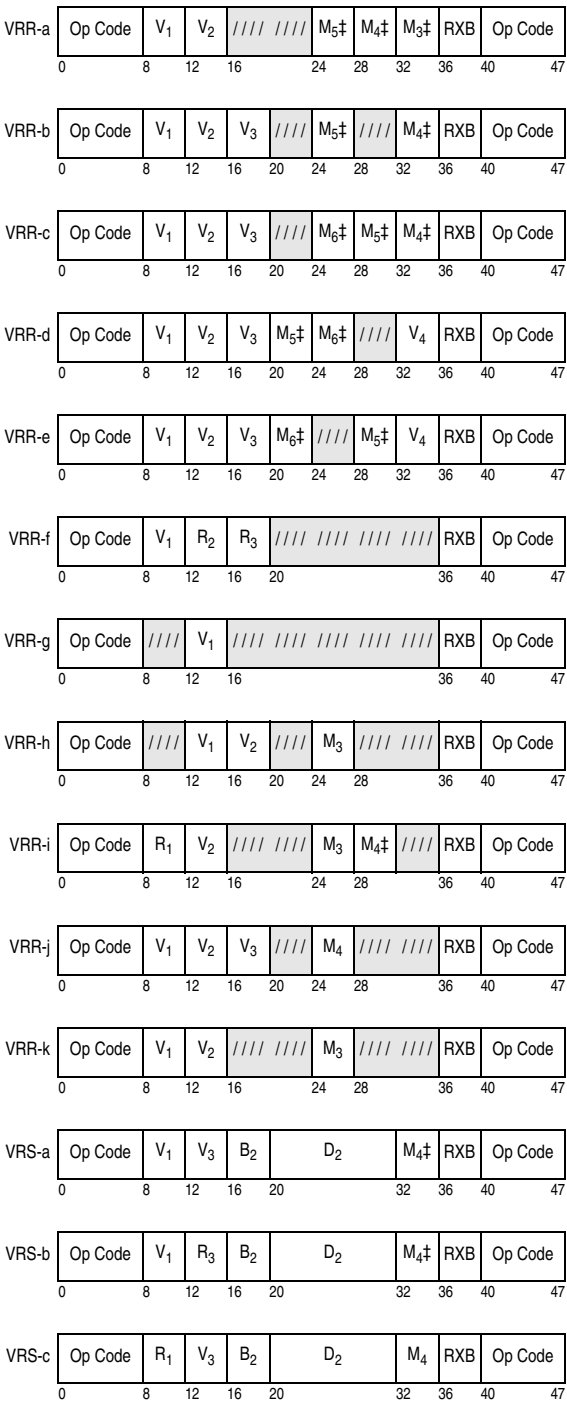


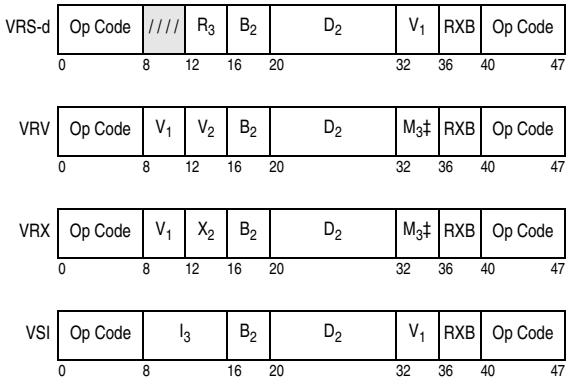
VRI-h



VRI-i




VRR-a
Op Code
V1
V2
//// ////
M5‡
M4‡
M3‡
RXB
Op Code
0 8 12 16 24 28 32 36 40 47
VRR-b
Op Code
V1
V2
V3
////
M5‡
////
M4‡
RXB
Op Code
0 8 12 16 20 24 28 32 36 40 47
VRR-c
Op Code
V1
V2
V3
////
M6‡
M5‡
M4‡
RXB
Op Code
0 8 12 16 20 24 28 32 36 40 47
VRR-d
Op Code
V1
V2
V3
M5‡
M6‡
////
V4
RXB
Op Code
0 8 12 16 20 24 28 32 36 40 47
VRR-e
Op Code
V1
V2
V3
M6‡
////
M5‡
V4
RXB
Op Code
0 8 12 16 20 24 28 32 36 40 47
VRR-f
Op Code
V1
R2
R3
//// //// //// ////
RXB
Op Code
0 8 12 16 20 36 40 47
VRR-g
Op Code
////
V1
//// //// //// //// ////
RXB
Op Code
0 8 12 16 36 40 47
VRR-h
Op Code
////
V1
V2
////
M3
//// ////
RXB
Op Code
0 8 12 16 20 24 28 36 40 47
VRR-i
Op Code
R1
V2
//// ////
M3
M4‡
////
RXB
Op Code
0 8 12 16 24 28 36 40 47
VRR-j
Op Code
V1
V2
V3
////
M4
//// ////
RXB
Op Code
0 8 12 16 20 24 28 36 40 47
VRR-k
Op Code
V1
V2
//// ////
M3
//// ////
RXB
Op Code
0 8 12 16 24 28 36 40 47
VRS-a
Op Code
V1
V3
B2
D2
M4‡
RXB
Op Code
0 8 12 16 20 32 36 40 47
VRS-b
Op Code
V1
R3
B2
D2
M4‡
RXB
Op Code
0 8 12 16 20 32 36 40 47
VRS-c
Op Code
R1
V3
B2
D2
M4
RXB
Op Code
0 8 12 16 20 32 36 40 47



1, 2, 3, 4, 5, 6	Denotes association with first, second, third, fourth, fifth, or sixth operand
a, b, c, d, e, f	Distinguishes among instances of the same basic instruction format
B_1, B_2, B_3, B_4	Base register designation field
D_1, D_2, D_3, D_4	Displacement field (including DH and DL for long-displacement forms)
I, I_2, I_3, I_4, I_5	Immediate operand field
L, L_1, L_2	Length field
M_1, M_3, M_4, M_5, M_6	Mask field
R_1, R_2, R_3	Register designation field
RI_2, RI_3, RI_4	Relative-immediate operand field
RXB	Most significant bits of vector registers designated by the V_1, V_2, V_3, V_4 fields, respectively
X_2	Index register designation field
‡	For certain instructions, this operand is not defined

Machine Instructions by Mnemonic

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
A	$R_1,D_2(X_2,B_2)$	Add (32)	RX-a	5A	c
AD	$R_1,D_2(X_2,B_2)$	Add Normalized (LH)	RX-a	6A	¤ c
ADB	$R_1,D_2(X_2,B_2)$	Add (LB)	RXE	ED1A	¤ c
ADBR	R_1,R_2	Add (LB)	RRE	B31A	¤ c
ADR	R_1,R_2	Add Normalized (LH)	RR	2A	¤ c
ADTR	R_1,R_2,R_3	Add (LD)	RRF-a	B3D2	¤ c TF
ADTRA	R_1,R_2,R_3,M_4	Add (LD)	RRF-a	B3D2	¤ c F
AE	$R_1,D_2(X_2,B_2)$	Add Normalized (SH)	RX-a	7A	¤ c
AEB	$R_1,D_2(X_2,B_2)$	Add (SB)	RXE	ED0A	¤ c
AEBR	R_1,R_2	Add (SB)	RRE	B30A	¤ c
AER	R_1,R_2	Add Normalized (SH)	RR	3A	¤ c
AFI	R_1,I_2	Add Immediate (32)	RIL-a	C29	c EI
AG	$R_1,D_2(X_2,B_2)$	Add (64)	RXY-a	E308	c N
AGF	$R_1,D_2(X_2,B_2)$	Add (64←32)	RXY-a	E318	c N
AGFI	R_1,I_2	Add Immediate (64←32)	RIL-a	C28	c EI
AGFR	R_1,R_2	Add (64←32)	RRE	B918	c N
AGH	$R_1,D_2(X_2,B_2)$	Add Halfword (64←16)	RXY-a	E338	c MI2
AGHI	R_1,I_2	Add Halfword Immediate (64←16)	RI-a	A7B	c N
AGHIK	R_1,R_3,I_2	Add Immediate (64←16)	RIE-d	ECD9	c DO
AGR	R_1,R_2	Add (64)	RRE	B908	c N
AGRK	R_1,R_2,R_3	Add (64)	RRF-a	B9E8	c DO
AGSI	$D_1(B_1),I_2$	Add Immediate (64←8)	SIY	EB7A	c GE
AH	$R_1,D_2(X_2,B_2)$	Add Halfword (32←16)	RX-a	4A	c
AHHHR	R_1,R_2,R_3	Add High (32)	RRF-a	B9C8	c HW
AHHLR	R_1,R_2,R_3	Add High (32)	RRF-a	B9D8	c HW
AHI	R_1,I_2	Add Halfword Immediate (32←16)	RI-a	A7A	c
AHIK	R_1,R_3,I_2	Add Immediate (32←16)	RIE-d	ECD8	c DO
AHY	$R_1,D_2(X_2,B_2)$	Add Halfword (32←16)	RXY-a	E37A	c LD
AIH	R_1,I_2	Add Immediate High (32)	RIL-a	CC8	c HW
AL	$R_1,D_2(X_2,B_2)$	Add Logical (32)	RX-a	5E	c
ALC	$R_1,D_2(X_2,B_2)$	Add Logical with Carry (32)	RXY-a	E398	c N3
ALCG	$R_1,D_2(X_2,B_2)$	Add Logical with Carry (64)	RXY-a	E388	c N
ALCGR	R_1,R_2	Add Logical with Carry (64)	RRE	B988	c N
ALCR	R_1,R_2	Add Logical with Carry (32)	RRE	B998	c N3
ALFI	R_1,I_2	Add Logical Immediate (32)	RIL-a	C2B	c EI
ALG	$R_1,D_2(X_2,B_2)$	Add Logical (64)	RXY-a	E30A	c N
ALGF	$R_1,D_2(X_2,B_2)$	Add Logical (64←32)	RXY-a	E31A	c N
ALGFI	R_1,I_2	Add Logical Immediate (64←32)	RIL-a	C2A	c EI
ALGFR	R_1,R_2	Add Logical (64←32)	RRE	B91A	c N
ALGHSIK	R_1,R_3,I_2	Add Logical with Signed Immediate (64←16)	RIE-d	ECDB	c DO
ALGR	R_1,R_2	Add Logical (64)	RRE	B90A	c N
ALGRK	R_1,R_2,R_3	Add Logical (64)	RRF-a	B9EA	c DO
ALGSI	$D_1(B_1),I_2$	Add Logical with Signed Immediate (64←8)	SIY	EB7E	c GE
ALHHHR	R_1,R_2,R_3	Add Logical High (32)	RRF-a	B9CA	c HW
ALHHLR	R_1,R_2,R_3	Add Logical High (32)	RRF-a	B9DA	c HW
ALHSIK	R_1,R_3,I_2	Add Logical with Signed Immediate (32←16)	RIE-d	ECDA	c DO
ALR	R_1,R_2	Add Logical (32)	RR	1E	c
ALRK	R_1,R_2,R_3	Add Logical (32)	RRF-a	B9FA	c DO
ALSI	$D_1(B_1),I_2$	Add Logical with Signed Immediate (32←8)	SIY	EB6E	c GE
ALSIH	R_1,I_2	Add Logical with Signed Immediate High (32)	RIL-a	CCA	c HW
ALSIHN	R_1,I_2	Add Logical with Signed Immediate High (32)	RIL-a	CCB	HW
ALY	$R_1,D_2(X_2,B_2)$	Add Logical (32)	RXY-a	E35E	c LD
AP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Add Decimal	SS-b	FA	¤ c
AR	R_1,R_2	Add (32)	RR	1A	c
ARK	R_1,R_2,R_3	Add (32)	RRF-a	B9F8	c DO

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
ASI	$D_1(B_1),I_2$	Add Immediate (32←8)	SIY	EB6A	c GE
AU	$R_1,D_2(X_2,B_2)$	Add Unnormalized (SH)	RX-a	7E	¤ c
AUR	R_1,R_2	Add Unnormalized (SH)	RR	3E	¤ c
AW	$R_1,D_2(X_2,B_2)$	Add Unnormalized (LH)	RX-a	6E	¤ c
AWR	R_1,R_2	Add Unnormalized (LH)	RR	2E	¤ c
AXBR	R_1,R_2	Add (EB)	RRE	B34A	¤ c
AXR	R_1,R_2	Add Normalized (EH)	RR	36	¤ c
AXTR	R_1,R_2,R_3	Add (ED)	RRF-a	B3DA	¤ c TF
AXTRA	R_1,R_2,R_3,M_4	ADD (ED)	RRF-a	B3DA	¤ c F
AY	$R_1,D_2(X_2,B_2)$	Add (32)	RXY-a	E35A	c LD
BAKR	R_1,R_2	Branch and Stack	RRE	B240	¤
BAL	$R_1,D_2(X_2,B_2)$	Branch and Link	RX-a	45	¤
BALR	R_1,R_2	Branch and Link	RR	05	¤
BAS	$R_1,D_2(X_2,B_2)$	Branch and Save	RX-a	4D	¤
BASR	R_1,R_2	Branch and Save	RR	0D	¤
BASSM	R_1,R_2	Branch and Save and Set Mode	RR	0C	¤
BC	$M_1,D_2(X_2,B_2)$	Branch on Condition	RX-b	47	¤
BCR	M_1,R_2	Branch on Condition	RR	07	¤
BCT	$R_1,D_2(X_2,B_2)$	Branch on Count (32)	RX-a	46	¤
BCTG	$R_1,D_2(X_2,B_2)$	Branch on Count (64)	RXY-a	E346	¤ N
BCTGR	R_1,R_2	Branch on Count (64)	RRE	B946	¤ N
BCTR	R_1,R_2	Branch on Count (32)	RR	06	¤
BIC	$M_1,D_2(X_2,B_2)$	Branch Indirect on Condition	RXY-b	E347	¤ MI2
BPP	$M_1,RI_2,D_3(B_3)$	Branch Prediction Preload	SMI	C7	¤ EH
BPRP	M_1,RI_2,RI_3	Branch Prediction Relative Preload	MII	C5	¤ EH
BRAS	R_1,RI_2	Branch Relative and Save	RI-b	A75	¤
BRASL	R_1,RI_2	Branch Relative and Save Long	RIL-b	C05	¤ N3
BRC	M_1,RI_2	Branch Relative on Condition	RI-c	A74	¤
BRCL	M_1,RI_2	Branch Relative on Condition Long	RIL-c	C04	¤ N3
BRCT	R_1,RI_2	Branch Relative on Count (32)	RI-b	A76	¤
BRCTG	R_1,RI_2	Branch Relative on Count (64)	RI-b	A77	¤ N
BRCTH	R_1,RI_2	Branch Relative on Count High (32)	RIL-b	CC6	¤ HW
BRXH	R_1,R_3,RI_2	Branch Relative on Index High (32)	RSI	84	¤
BRXHG	R_1,R_3,RI_2	Branch Relative on Index High (64)	RIE-e	EC44	¤ N
BRXLE	R_1,R_3,RI_2	Branch Relative on Index Low or Equal (32)	RSI	85	¤
BRXLG	R_1,R_3,RI_2	Branch Relative on Index Low or Equal (64)	RIE-e	EC45	¤ N
BSA	R_1,R_2	Branch and Set Authority	RRE	B25A	q
BSG	R_1,R_2	Branch in Subspace Group	RRE	B258	¤
BSM	R_1,R_2	Branch and Set Mode	RR	0B	¤
BXH	$R_1,R_3,D_2(B_2)$	Branch on Index High (32)	RS-a	86	¤
BXHG	$R_1,R_3,D_2(B_2)$	Branch on Index High (64)	RSY-a	EB44	¤ N
BXLE	$R_1,R_3,D_2(B_2)$	Branch on Index Low or Equal (32)	RS-a	87	¤
BXLEG	$R_1,R_3,D_2(B_2)$	Branch on Index Low or Equal (64)	RSY-a	EB45	¤ N
C	$R_1,D_2(X_2,B_2)$	Compare (32)	RX-a	59	c
CD	$R_1,D_2(X_2B_2),M_3$	Compare (LH)	RX-a	69	c
CDB	$R_1,D_2(X_2,B_2)$	Compare (LB)	RXE	ED19	¤ c
CDBR	R_1,R_2	Compare (LB)	RRE	B319	¤ c
CDFBR	R_1,R_2	Convert from Fixed (LB←32)	RRE	B395	¤
CDFBRA	R_1,M_3,R_2,M_4	Convert from Fixed (LB←32)	RRF-e	B395	¤ F
CDFR	R_1,R_2	Convert from Fixed (LH←32)	RRE	B3B5	¤
CDFTR	R_1,M_3,R_2,M_4	Convert from Fixed (LD←32)	RRF-e	B951	¤ F
CDGBR	R_1,R_2	Convert from Fixed (LB←64)	RRE	B3A5	¤ N
CDGBRA	R_1,M_3,R_2,M_4	Convert from Fixed (LB←64)	RRF-e	B3A5	¤ F
CDGR	R_1,R_2	Convert from Fixed (LH←64)	RRE	B3C5	¤ N
CDGTR	R_1,R_2	Convert from Fixed (LD←64)	RRE	B3F1	¤ TF
CDGTRA	R_1,M_3,R_2,M_4	Convert from Fixed (LD←64)	RRF-e	B3F1	¤ F
CDLFBR	R_1,M_3,R_2,M_4	Convert from Logical (LB←32)	RRF-e	B391	¤ F
CDLFTR	R_1,M_3,R_2,M_4	Convert from Logical (LD←32)	RRF-e	B953	¤ F
CDLGBR	R_1,M_3,R_2,M_4	Convert from Logical (LB←64)	RRF-e	B3A1	¤ F
CDLGTR	R_1,M_3,R_2,M_4	Convert from Logical (LD←64)	RRF-e	B952	¤ F

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
CDPT	$R_1,D_2(L_2,B_2),M_3$	Convert from Packed (To Long DFP)	RSL-b	EDAE	PC
CDR	R_1,R_2	Compare (LH)	RR	29	¤ c
CDS	$R_1,R_3,D_2(B_2)$	Compare Double and Swap (32)	RS-a	BB	¤ c
CDSG	$R_1,R_3,D_2(B_2)$	Compare Double and Swap (64)	RSY-a	EB3E	¤ c N
CDSTR	R_1,R_2	Convert from Signed Packed (LD←64)	RRE	B3F3	¤ TF
CDSY	$R_1,R_3,D_2(B_2)$	Compare Double and Swap (32)	RSY-a	EB31	¤ c LD
CDTR	R_1,R_2	Compare (LD)	RRE	B3E4	¤ c TF
CDUTR	R_1,R_2	Convert from Unsigned Packed (LD←64)	RRE	B3F2	¤ TF
CDZT	$R_1,D_2(L_2,B_2),M_3$	Convert from Zoned (to long DFP)	RSL-b	EDAA	¤ ZF
CE	$R_1,D_2(X_2,B_2)$	Compare (SH)	RX-a	79	¤ c
CEB	$R_1,D_2(X_2,B_2)$	Compare (SB)	RXE	ED09	¤ c
CEBR	R_1,R_2	Compare (SB)	RRE	B309	¤ c
CEDTR	R_1,R_2	Compare Biased Exponent (LD)	RRE	B3F4	¤ c TF
CEFBR	R_1,R_2	Convert from Fixed (SB←32)	RRE	B394	¤
CEFBRA	R_1,M_3,R_2,M_4	Convert from Fixed (SB←32)	RRF-e	B394	¤ F
CEFR	R_1,R_2	Convert from Fixed (SH←32)	RRE	B3B4	¤
CEGBR	R_1,R_2	Convert from Fixed (SB←64)	RRE	B3A4	¤ N
CEGBRA	R_1,M_3,R_2,M_4	Convert from Fixed (SB←64)	RRF-e	B3A4	¤ F
CEGR	R_1,R_2	Convert from Fixed (SH←64)	RRE	B3C4	¤ N
CELFBR	R_1,M_3,R_2,M_4	Convert from Logical (SB←32)	RRF-e	B390	¤ F
CELGBR	R_1,M_3,R_2,M_4	Convert from Logical (SB←64)	RRF-e	B3A0	¤ F
CER	R_1,R_2	Compare (SH)	RR	39	¤ c
CEXTR	R_1,R_2	Compare Biased Exponent (ED)	RRE	B3FC	¤ c TF
CFC	$D_2(B_2)$	Compare and Form Codeword	S	B21A	i ¤ c
CFDBR	R_1,M_3,R_2	Convert to Fixed (32←LB)	RRF-e	B399	¤ c
CFDBRA	R_1,M_3,R_2,M_4	Convert to Fixed (32←LB)	RRF-e	B399	¤ c F
CFDR	R_1,M_3,R_2	Convert to Fixed (32←LH)	RRF-e	B3B9	¤ c
CFDTR	R_1,M_3,R_2,M_4	Convert to Fixed (32←LD)	RRF-e	B941	¤ c F
CFEBR	R_1,M_3,R_2	Convert to Fixed (32←SB)	RRF-e	B398	¤ c
CFEBRA	R_1,M_3,R_2,M_4	Convert to Fixed (32←SB)	RRF-e	B398	¤ c F
CFER	R_1,M_3,R_2	Convert to Fixed (32←SH)	RRF-e	B3B8	¤ c
CFI	R_1,I_2	Compare Immediate (32)	RIL-a	C2D	c EI
CFXBR	R_1,M_3,R_2	Convert to Fixed (32←EB)	RRF-e	B39A	¤ c
CFXBRA	R_1,M_3,R_2,M_4	Convert to Fixed (32←EB)	RRF-e	B39A	¤ c F
CFXR	R_1,M_3,R_2	Convert to Fixed (32←EH)	RRF-e	B3BA	¤ c
CFXTR	R_1,M_3,R_2,M_4	Convert to Fixed (32←ED)	RRF-e	B949	¤ c F
CG	$R_1,D_2(X_2,B_2)$	Compare (64)	RXY-a	E320	c N
CGDBR	R_1,M_3,R_2	Convert to Fixed (64←LB)	RRF-e	B3A9	¤ c N
CGDBRA	R_1,M_3,R_2,M_4	Convert to Fixed (64←LB)	RRF-e	B3A9	¤ c F
CGDR	R_1,M_3,R_2	Convert to Fixed (64←LH)	RRF-e	B3C9	¤ c N
CGDTR	R_1,M_3,R_2	Convert to Fixed (64←LD)	RRF-e	B3E1	¤ c TF
CGDTRA	R_1,M_3,R_2,M_4	Convert to Fixed (64←LD)	RRF-e	B3E1	¤ c F
CGEBR	R_1,M_3,R_2	Convert to Fixed (64←SB)	RRF-e	B3A8	¤ c N
CGEBRA	R_1,M_3,R_2,M_4	Convert to Fixed (64←SB)	RRF-e	B3A8	¤ c F
CGER	R_1,M_3,R_2	Convert to Fixed (64←SH)	RRF-e	B3C8	¤ c N
CGF	$R_1,D_2(X_2,B_2)$	Compare (64←32)	RXY-a	E330	c N
CGFI	R_1,I_2	Compare Immediate (64←32)	RIL-a	C2C	c EI
CGFR	R_1,R_2	Compare (64←32)	RRE	B930	c N
CGFRL	R_1,RI_2	Compare Relative Long (64←32)	RIL-b	C6C	c GE
CGH	$R_1,D_2(X_2,B_2)$	Compare Halfword (64←16)	RXY-a	E334	c GE
CGHI	R_1,I_2	Compare Halfword Immediate (64←16)	RI-a	A7F	c N
CGHRL	R_1,RI_2	Compare Halfword Relative Long (64←16)	RIL-b	C64	c GE
CGHSI	$D_1(B_1),I_2$	Compare Halfword Immediate (64←16)	SIL	E558	c GE
CGIB	$R_1,I_2,M_3,D_4(B_4)$	Compare Immediate and Branch (64←8)	RIS	ECFC	¤ GE
CGIJ	R_1,I_2,M_3,RI_4	Compare Immediate and Branch Relative (64←8)	RIE-c	EC7C	¤ GE
CGIT	R_1,I_2,M_3	Compare Immediate and Trap (64←16)	RIE-a	EC70	GE
CGR	R_1,R_2	Compare (64)	RRE	B920	c N
CGRB	$R_1,R_2,M_3,D_4(B_4)$	Compare and Branch (64)	RRS	ECE4	¤ GE
CGRJ	R_1,R_2,M_3,RI_4	Compare and Branch Relative (64)	RIE-b	EC64	¤ GE

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
CGRL	R_1,RI_2	Compare Relative Long (64)	RIL-b	C68	c GE
CGRT	R_1,R_2,M_3	Compare and Trap (64)	RRF-c	B960	GE
CGXBR	R_1,M_3,R_2	Convert to Fixed (64←EB)	RRF-e	B3AA	¤ c N
CGXBRA	R_1,M_3,R_2,M_4	Convert to Fixed (64←EB)	RRF-e	B3AA	¤ c F
CGXR	R_1,M_3,R_2	Convert to Fixed (64←EH)	RRF-e	B3CA	¤ c N
CGXTR	R_1,M_3,R_2	Convert to Fixed (64←ED)	RRF-e	B3E9	¤ c TF
CGXTRA	R_1,M_3,R_2,M_4	Convert to Fixed (64←ED)	RRF-e	B3E9	¤ c F
CH	$R_1,D_2(X_2,B_2)$	Compare Halfword (32←16)	RX-a	49	c
CHF	$R_1,D_2(X_2,B_2)$	Compare High (32)	RXY-a	E3CD	c HW
CHHR	R_1,R_2	Compare High (32)	RRE	B9CD	c HW
CHHSI	$D_1(B_1),I_2$	Compare Halfword Immediate (16←16)	SIL	E554	c GE
CHI	R_1,I_2	Compare Halfword Immediate (32←16)	RI-a	A7E	c
CHLR	R_1,R_2	Compare High (32)	RRE	B9DD	c HW
CHRL	R_1,RI_2	Compare Halfword Relative Long (32←16)	RIL-b	C65	c GE
CHSI	$D_1(B_1),I_2$	Compare Halfword Immediate (32←16)	SIL	E55C	c GE
CHY	$R_1,D_2(X_2,B_2)$	Compare Halfword (32←16)	RXY-a	E379	c LD
CIB	$R_1,I_2,M_3,D_4(B_4)$	Compare Immediate and Branch (32←8)	RIS	ECFE	¤ GE
CIH	R_1,I_2	Compare Immediate High (32)	RIL-a	CCD	c HW
CIJ	R_1,I_2,M_3,RI_4	Compare Immediate and Branch Relative (32←8)	RIE-c	EC7E	¤ GE
CIT	R_1,I_2,M_3	Compare Immediate and Trap (32←16)	RIE-a	EC72	GE
CKSM	R_1,R_2	Checksum	RRE	B241	¤ c
CL	$R_1,D_2(X_2,B_2)$	Compare Logical (32)	RX-a	55	c
CLC	$D_1(L,B_1),D_2(B_2)$	Compare Logical (character)	SS-a	D5	¤ c
CLCL	R_1,R_2	Compare Logical Long	RR	0F	i c
CLCLE	$R_1,R_3,D_2(B_2)$	Compare Logical Long Extended	RS-a	A9	¤ c
CLCLU	$R_1,R_3,D_2(B_2)$	Compare Logical Long Unicode	RSY-a	EB8F	¤ c E2
CLFDBR	R_1,M_3,R_2,M_4	Convert to Logical (32←LB)	RRF-e	B39D	¤ c F
CLFDTR	R_1,M_3,R_2,M_4	Convert to Logical (32←LD)	RRF-e	B943	¤ c F
CLFEBR	R_1,M_3,R_2,M_4	Convert to Logical (32←SB)	RRF-e	B39C	¤ c F
CLFHSI	$D_1(B_1),I_2$	Compare Logical Immediate (32←16)	SIL	E55D	c GE
CLFI	R_1,I_2	Compare Logical Immediate (32)	RIL-a	C2F	c EI
CLFIT	R_1,I_2,M_3	Compare Logical Immediate and Trap (32←16)	RIE-a	EC73	GE
CLFXBR	R_1,M_3,R_2,M_4	Convert to Logical (32←EB)	RRF-e	B39E	¤ c F
CLFXTR	R_1,M_3,R_2,M_4	Convert to Logical (32←ED)	RRF-e	B94B	¤ c F
CLG	$R_1,D_2(X_2,B_2)$	Compare Logical (64)	RXY-a	E321	c N
CLGDBR	R_1,M_3,R_2,M_4	Convert to Logical (64←LB)	RRF-e	B3AD	¤ c F
CLGDTR	R_1,M_3,R_2,M_4	Convert to Logical (64←LD)	RRF-e	B942	¤ c F
CLGEBR	R_1,M_3,R_2,M_4	Convert to Logical (64←SB)	RRF-e	B3AC	¤ c F
CLGF	$R_1,D_2(X_2,B_2)$	Compare Logical (64←32)	RXY-a	E331	c N
CLGFI	R_1,I_2	Compare Logical Immediate (64←32)	RIL-a	C2E	c EI
CLGFR	R_1,R_2	Compare Logical (64←32)	RRE	B931	c N
CLGFRL	R_1,RI_2	Compare Logical Relative Long (64←32)	RIL-b	C6E	c GE
CLGHRL	R_1,RI_2	Compare Logical Relative Long (64←16)	RIL-b	C66	c GE
CLGHSI	$D_1(B_1),I_2$	Compare Logical Immediate (64←16)	SIL	E559	c GE
CLGIB	$R_1,I_2,M_3,D_4(B_4)$	Compare Logical Immediate and Branch (64←8)	RIS	ECFD	¤ GE
CLGIJ	R_1,I_2,M_3,RI_4	Compare Logical Immediate and Branch Relative (64←8)	RIE-c	EC7D	¤ GE
CLGIT	R_1,I_2,M_3	Compare Logical Immediate and Trap (64←16)	RIE-a	EC71	GE
CLGR	R_1,R_2	Compare Logical (64)	RRE	B921	c N
CLGRB	$R_1,R_2,M_3,D_4(B_4)$	Compare Logical and Branch (64)	RRS	ECE5	¤ GE
CLGRJ	R_1,R_2,M_3,RI_4	Compare Logical and Branch Relative (64)	RIE-b	EC65	¤ GE
CLGRL	R_1,RI_2	Compare Logical Relative Long (64)	RIL-b	C6A	c GE
CLGRT	R_1,R_2,M_3	Compare Logical and Trap (64)	RRF-c	B961	GE
CLGT	$R_1,M_3,D_2(B_2)$	Compare Logical and Trap (64)	RSY-b	EB2B	MI1
CLGXBR	R_1,M_3,R_2,M_4	Convert to Logical (64←EB)	RRF-e	B3AE	¤ c F
CLGXTR	R_1,M_3,R_2,M_4	Convert to Logical (64←ED)	RRF-e	B94A	¤ c F
CLHF	$R_1,D_2(X_2,B_2)$	Compare Logical High (32)	RXY-a	E3CF	c HW

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
CLHHR	R_1,R_2	Compare Logical High (32)	RRE	B9CF	c HW
CLHHSI	$D_1(B_1),I_2$	Compare Logical Immediate (16←16)	SIL	E555	c GE
CLHLR	R_1,R_2	Compare Logical High (32)	RRE	B9DF	c HW
CLHRL	R_1,RI_2	Compare Logical Relative Long (32←16)	RIL-b	C67	c GE
CLI	$D_1(B_1),I_2$	Compare Logical Immediate	SI	95	c
CLIB	$R_1,I_2,M_3,D_4(B_4)$	Compare Logical Immediate and Branch (32←8)	RIS	ECFF	¤ GE
CLIH	R_1,I_2	Compare Logical Immediate High (32)	RIL-a	CCF	c HW
CLIJ	R_1,I_2,M_3,RI_4	Compare Logical Immediate and Branch Relative (32←8)	RIE-c	EC7F	¤ GE
CLIY	$D_1(B_1),I_2$	Compare Logical Immediate	SIY	EB55	c LD
CLM	$R_1,M_3,D_2(B_2)$	Compare Logical Char. under Mask (low)	RS-b	BD	c
CLMH	$R_1,M_3,D_2(B_2)$	Compare Logical Char. under Mask (high)	RSY-b	EB20	c N
CLMY	$R_1,M_3,D_2(B_2)$	Compare Logical Char. under Mask (low)	RSY-b	EB21	c LD
CLR	R_1,R_2	Compare Logical (32)	RR	15	c
CLRB	$R_1,R_2,M_3,D_4(B_4)$	Compare Logical and Branch (32)	RRS	ECF7	¤ GE
CLRJ	R_1,R_2,M_3,RI_4	Compare Logical and Branch Relative (32)	RIE-b	EC77	¤ GE
CLRL	R_1,RI_2	Compare Logical Relative Long (32)	RIL-b	C6F	c GE
CLRT	R_1,R_2,M_3	Compare Logical and Trap (32)	RRF-c	B973	GE
CLST	R_1,R_2	Compare Logical String	RRE	B25D	¤ c
CLT	$R_1,M_3,D_2(B_2)$	Compare Logical and Trap (32)	RSY-b	EB23	MI1
CLY	$R_1,D_2(X_2,B_2)$	Compare Logical (32)	RXY-a	E355	c LD
CMPSC	R_1,R_2	Compression Call	RRE	B263	i ¤ c
CP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Compare Decimal	SS-b	F9	¤ c
CPDT	$R_1,D_2(L_2,B_2),M_3$	Convert to Packed (From Long DFP)	RSL-b	EDAC	c PC
CPSDR	R_1,R_3,R_2	Copy Sign (L)	RRF-b	B372	¤ FS
CPXT	$R_1,D_2(L_2,B_2),M_3$	Convert to Packed (From Extended DFP)	RSL-b	EDAD	c PC
CPYA	R_1,R_2	Copy Access	RRE	B24D	¤
CR	R_1,R_2	Compare (32)	RR	19	c
CRB	$R_1,R_2,M_3,D_4(B_4)$	Compare and Branch (32)	RRS	ECF6	¤ GE
CRDTE	$R_1,R_3,R_2[,M_4]$	Compare and Replace DAT Table Entry	RRF-b	B98F	ED2 p c
CRJ	R_1,R_2,M_3,RI_4	Compare and Branch Relative (32)	RIE-b	EC76	¤ GE
CRL	R_1,RI_2	Compare Relative Long (32)	RIL-b	C6D	c GE
CRT	R_1,R_2,M_3	Compare and Trap (32)	RRF-c	B972	GE
CS	$R_1,R_3,D_2(B_2)$	Compare and Swap (32)	RS-a	BA	¤ c
CSCH		Clear Subchannel	S	B230	p c
CSDTR	R_1,R_2,M_4	Convert to Signed Packed (64←LD)	RRF-d	B3E3	¤ TF
CSG	$R_1,R_3,D_2(B_2)$	Compare and Swap (64)	RSY-a	EB30	¤ c N
CSP	R_1,R_2	Compare and Swap and Purge (32)	RRE	B250	p c
CSPG	R_1,R_2	Compare and Swap and Purge (64)	RRE	B98A	p c DE
CSST	$D_1(B_1),D_2(B_2),R_3$	Compare and Swap and Store	SSF	C82	¤ c
CSXTR	R_1,R_2,M_4	Convert to Signed Packed (128←ED)	RRF-d	B3EB	¤ TF
CSY	$R_1,R_3,D_2(B_2)$	Compare and Swap (32)	RSY-a	EB14	¤ c LD
CU12	$R_1,R_2[,M_3]$	Convert UTF-8 to UTF-16	RRF-c	B2A7	¤ c
CU14	$R_1,R_2[,M_3]$	Convert UTF-8 to UTF-32	RRF-c	B9B0	¤ c E3
CU21	$R_1,R_2[,M_3]$	Convert UTF-16 to UTF-8	RRF-c	B2A6	¤ c
CU24	$R_1,R_2[,M_3]$	Convert UTF-16 to UTF-32	RRF-c	B9B1	¤ c E3
CU41	R_1,R_2	Convert UTF-32 to UTF-8	RRE	B9B2	¤ c E3
CU42	R_1,R_2	Convert UTF-32 to UTF-16	RRE	B9B3	¤ c E3
CUDTR	R_1,R_2	Convert to Unsigned Packed (64←LD)	RRE	B3E2	¤ TF
CUSE	R_1,R_2	Compare until Substring Equal	RRE	B257	i c
CUTFU	$R_1,R_2[,M_3]$	Convert UTF-8 to Unicode	RRF-c	B2A7	¤ c
CUUTF	$R_1,R_2[,M_3]$	Convert Unicode to UTF-8	RRF-c	B2A6	¤ c
CUXTR	R_1,R_2	Convert to Unsigned Packed (128←ED)	RRE	B3EA	¤ TF
CVB	$R_1,D_2(X_2,B_2)$	Convert to Binary (32)	RX-a	4F	¤
CVBG	$R_1,D_2(X_2,B_2)$	Convert to Binary (64)	RXY-a	E30E	¤ N
CVBY	$R_1,D_2(X_2,B_2)$	Convert to Binary (32)	RXY-a	E306	¤ LD
CVD	$R_1,D_2(X_2,B_2)$	Convert to Decimal (32)	RX-a	4E	¤
CVDG	$R_1,D_2(X_2,B_2)$	Convert to Decimal (64)	RXY-a	E32E	¤ N
CVDY	$R_1,D_2(X_2,B_2)$	Convert to Decimal (32)	RXY-a	E326	¤ LD

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
CXBR	R_1,R_2	Compare (EB)	RRE	B349	¤ c
CXFBR	R_1,R_2	Convert from Fixed (EB←32)	RRE	B396	¤
CXFBRA	R_1,M_3,R_2,M_4	Convert from Fixed (EB←32)	RRF-e	B396	¤ F
CXFR	R_1,R_2	Convert from Fixed (EH←32)	RRE	B3B6	¤
CXFTR	R_1,M_3,R_2,M_4	Convert from Fixed (ED←32)	RRF-e	B959	¤ F
CXGBR	R_1,R_2	Convert from Fixed (EB←64)	RRE	B3A6	¤ N
CXGBRA	R_1,M_3,R_2,M_4	Convert from Fixed (EB←64)	RRF-e	B3A6	¤ F
CXGR	R_1,R_2	Convert from Fixed (EH←64)	RRE	B3C6	¤ N
CXGTR	R_1,R_2	Convert from Fixed (ED←64)	RRE	B3F9	¤ TF
CXGTRA	R_1,M_3,R_2,M_4	Convert from Fixed (ED←64)	RRF-e	B3F9	¤ F
CXLFBR	R_1,M_3,R_2,M_4	Convert from Logical (EB←32)	RRF-e	B392	¤ F
CXLFTR	R_1,M_3,R_2,M_4	Convert from Logical (ED←32)	RRF-e	B95B	¤ F
CXLGBR	R_1,M_3,R_2,M_4	Convert from Logical (EB←64)	RRF-e	B3A2	¤ F
CXLGTR	R_1,M_3,R_2,M_4	Convert from Logical (ED←64)	RRF-e	B95A	¤ F
CXPT	$R_1,D_2(L_2,B_2),M_3$	Convert from Packed (To Extended DFP)	RSL-b	EDAF	PC
CXR	R_1,R_2	Compare (EH)	RRE	B369	¤ c
CXSTR	R_1,R_2	Convert from Signed Packed (ED←128)	RRE	B3FB	¤ TF
CXTR	R_1,R_2	Compare (ED)	RRE	B3EC	¤ c TF
CXUTR	R_1,R_2	Convert from Unsigned Packed (ED←128)	RRE	B3FA	¤ TF
CXZT	$R_1,D_2(L_2,B_2),M_3$	Convert from Zoned (to extended DFP)	RSL-b	EDAB	¤ ZF
CY	$R_1,D_2(X_2,B_2)$	Compare (32)	RXY-a	E359	c LD
CZDT	$R_1,D_2(L_2,B_2),M_3$	Convert to Zoned (from long DFP)	RSL-b	EDA8	¤ ZF
CZXT	$R_1,D_2(L_2,B_2),M_3$	Convert to Zoned (from extended DFP)	RSL-b	EDA9	¤ ZF
D	$R_1,D_2(X_2,B_2)$	Divide (32←64)	RX-a	5D	¤
DD	$R_1,D_2(X_2,B_2)$	Divide (LH)	RX-a	6D	¤
DDB	$R_1,D_2(X_2,B_2)$	Divide (LB)	RXE	ED1D	¤
DDBR	R_1,R_2	Divide (LB)	RRE	B31D	¤
DDR	R_1,R_2	Divide (LH)	RR	2D	¤
DDTR	R_1,R_2,R_3	Divide (LD)	RRF-a	B3D1	¤ TF
DDTRA	R_1,R_2,R_3,M_4	Divide (LD)	RRF-a	B3D1	¤ F
DE	$R_1,D_2(X_2,B_2)$	Divide (SH)	RX-a	7D	¤
DEB	$R_1,D_2(X_2,B_2)$	Divide (SB)	RXE	ED0D	¤
DEBR	R_1,R_2	Divide (SB)	RRE	B30D	¤
DER	R_1,R_2	Divide (SH)	RR	3D	¤
DFLTCC	R_1,R_2,R_3	DEFLATE Conversion Call	RRF-a	B939	¤ c GZ
DIDBR	R_1,R_3,R_2,M_4	Divide to Integer (LB)	RRF-b	B35B	¤ c
DIEBR	R_1,R_3,R_2,M_4	Divide to Integer (SB)	RRF-b	B353	¤ c
DL	$R_1,D_2(X_2,B_2)$	Divide Logical (32←64)	RXY-a	E397	¤ N3
DLG	$R_1,D_2(X_2,B_2)$	Divide Logical (64←128)	RXY-a	E387	¤ N
DLGR	R_1,R_2	Divide Logical (64←128)	RRE	B987	¤ N
DLR	R_1,R_2	Divide Logical (32←64)	RRE	B997	¤ N3
DP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Divide Decimal	SS-b	FD	¤
DR	R_1,R_2	Divide (32←64)	RR	1D	¤
DSG	$R_1,D_2(X_2,B_2)$	Divide Single (64)	RXY-a	E30D	¤ N
DSGF	$R_1,D_2(X_2,B_2)$	Divide Single (64←32)	RXY-a	E31D	¤ N
DSGFR	R_1,R_2	Divide Single (64←32)	RRE	B91D	¤ N
DSGR	R_1,R_2	Divide Single (64)	RRE	B90D	¤ N
DXBR	R_1,R_2	Divide (EB)	RRE	B34D	¤
DXR	R_1,R_2	Divide (EH)	RRE	B22D	¤
DXTR	R_1,R_2,R_3	Divide (ED)	RRF-a	B3D9	¤ TF
DXTRA	R_1,R_2,R_3,M_4	Divide (ED)	RRF-a	B3D9	¤ F
EAR	R_1,R_2	Extract Access	RRE	B24F	
ECAG	$R_1,R_3,D_2(B_2)$	Extract CPU Attribute	RSY-a	EB4C	¤ GE
ECTG	$D_1(B_1),D_2(B_2),R_3$	Extract CPU Time	SSF	C81	¤ ET
ED	$D_1(L,B_1),D_2(B_2)$	Edit	SS-a	DE	¤ c
EDMK	$D_1(L,B_1),D_2(B_2)$	Edit and Mark	SS-a	DF	¤ c
EEDTR	R_1,R_2	Extract Biased Exponent (64←LD)	RRE	B3E5	¤ TF
EEXTR	R_1,R_2	Extract Biased Exponent (64←ED)	RRE	B3ED	¤ TF
EFPC	R_1	Extract FPC	RRE	B38C	¤
EPAIR	R_1	Extract Primary ASN and Instance	RRE	B99A	q RA

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
EPAR	R_1	Extract Primary ASN	RRE	B226	q
EPSW	R_1,R_2	Extract PSW	RRE	B98D	¤ N3
EREG	R_1,R_2	Extract Stacked Registers (32)	RRE	B249	¤
EREGG	R_1,R_2	Extract Stacked Registers (64)	RRE	B90E	¤ N
ESAIR	R_1	Extract Secondary ASN and Instance	RRE	B99B	q RA
ESAR	R_1	Extract Secondary ASN	RRE	B227	q
ESDTR	R_1,R_2	Extract Significance (64←LD)	RRE	B3E7	¤ TF
ESEA	R_1,R_2	Extract and Set Extended Authority	RRE	B99D	p N
ESTA	R_1,R_2	Extract Stacked State	RRE	B24A	¤ c
ESXTR	R_1,R_2	Extract Significance (64←ED)	RRE	B3EF	¤ TF
ETND	R_1	Extract Transaction Nesting Depth	RRE	B2EC	¤ TX
EX	$R_1,D_2(X_2,B_2)$	Execute	RX-a	44	¤
EXRL	R_1,RI_2	Execute Relative Long	RIL-b	C60	¤ XX
FIDBR	R_1,M_3,R_2	Load FP Integer (LB)	RRF-e	B35F	¤
FIDBRA	R_1,M_3,R_2,M_4	Load FP Integer (LB)	RRF-e	B35F	¤ F
FIDR	R_1,R_2	Load FP Integer (LH)	RRE	B37F	¤
FIDTR	R_1,M_3,R_2,M_4	Load FP Integer (LD)	RRF-e	B3D7	¤ TF
FIEBR	R_1,M_3,R_2	Load FP Integer (SB)	RRF-e	B357	¤
FIEBRA	R_1,M_3,R_2,M_4	Load FP Integer (SB)	RRF-e	B357	¤ F
FIER	R_1,R_2	Load FP Integer (SH)	RRE	B377	¤
FIXBR	R_1,M_3,R_2	Load FP Integer (EB)	RRF-e	B347	¤
FIXBRA	R_1,M_3,R_2,M_4	Load FP Integer (EB)	RRF-e	B347	¤ F
FIXR	R_1,R_2	Load FP Integer (EH)	RRE	B367	¤
FIXTR	R_1,M_3,R_2,M_4	Load FP Integer (ED)	RRF-e	B3DF	¤ TF
FLOGR	R_1,R_2	Find Leftmost One	RRE	B983	c EI
HDR	R_1,R_2	Halve (LH)	RR	24	¤
HER	R_1,R_2	Halve (SH)	RR	34	¤
HSCH		Halt Subchannel	S	B231	p c
IAC	R_1	Insert Address Space Control	RRE	B224	q c
IC	$R_1,D_2(X_2,B_2)$	Insert Character	RX-a	43	
ICM	$R_1,M_3,D_2(B_2)$	Insert Characters under Mask (low)	RS-b	BF	c
ICMH	$R_1,M_3,D_2(B_2)$	Insert Characters under Mask (high)	RSY-b	EB80	c N
ICMY	$R_1,M_3,D_2(B_2)$	Insert Characters under Mask (low)	RSY-b	EB81	c LD
ICY	$R_1,D_2(X_2,B_2)$	Insert Character	RXY-a	E373	LD
IDTE	R_1,R_3,R_2	Invalidate DAT Table Entry	RRF-b	B98E	p u DE
IEDTR	R_1,R_3,R_2	Insert Biased Exponent (LD←64&LD)	RRF-b	B3F6	¤ TF
IEXTR	R_1,R_3,R_2	Insert Biased Exponent (ED←64&ED)	RRF-b	B3FE	¤ TF
IIHF	R_1,I_2	Insert Immediate (high)	RIL-a	C08	EI
IIHH	R_1,I_2	Insert Immediate (high high)	RI-a	A50	N
IIHL	R_1,I_2	Insert Immediate (high low)	RI-a	A51	N
IILF	R_1,I_2	Insert Immediate (low)	RIL-a	C09	EI
IILH	R_1,I_2	Insert Immediate (low high)	RI-a	A52	N
IILL	R_1,I_2	Insert Immediate (low low)	RI-a	A53	N
IPK		Insert PSW Key	S	B20B	q
IPM	R_1	Insert Program Mask	RRE	B222	
IPTE	R_1,R_2	Invalidate Page Table Entry	RRF-a	B221	p
IRBM	R_1,R_2	Insert Reference Bits Multiple	RRE	B2AC	p IM
ISKE	R_1,R_2	Insert Storage Key Extended	RRE	B229	p
IVSK	R_1,R_2	Insert Virtual Storage Key	RRE	B223	q
KDB	$R_1,D_2(X_2,B_2)$	Compare and Signal (LB)	RXE	ED18	¤ c
KDBR	R_1,R_2	Compare and Signal (LB)	RRE	B318	¤ c
KDSA	R_1,R_2	Compute Digital Signature Authentication	RRE	B93A	¤ c M9
KDTR	R_1,R_2	Compare and Signal (LD)	RRE	B3E0	¤ c TF
KEB	$R_1,D_2(X_2,B_2)$	Compare and Signal (SB)	RXE	ED08	¤ c
KEBR	R_1,R_2	Compare and Signal (SB)	RRE	B308	¤ c
KIMD	R_1,R_2	Compute Intermediate Message Digest	RRE	B93E	¤ c MS
KLMD	R_1,R_2	Compute Last Message Digest	RRE	B93F	¤ c MS
KM	R_1,R_2	Cipher Message	RRE	B92E	¤ c MS
KMA	R_1,R_3,R_2	Cipher Message with Authentication	RRF-b	B929	¤ c M8
KMAC	R_1,R_2	Compute Message Authentication Code	RRE	B91E	¤ c MS

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
KMC	R_1,R_2	Cipher Message with Chaining	RRE	B92F	¤ c MS
KMCTR	R_1,R_3,R_2	Cipher Message with Counter	RRF-b	B92D	¤ c M4
KMF	R_1,R_2	Cipher Message with Cipher Feedback	RRE	B92A	¤ c M4
KMO	R_1,R_2	Cipher Message with Output Feedback	RRE	B92B	¤ c M4
KXBR	R_1,R_2	Compare and Signal (EB)	RRE	B348	¤ c
KXTR	R_1,R_2	Compare and Signal (ED)	RRE	B3E8	¤ cTF
L	$R_1,D_2(X_2,B_2)$	Load (32)	RX-a	58	
LA	$R_1,D_2(X_2,B_2)$	Load Address	RX-a	41	
LAA	$R_1,R_3,D_2(B_2)$	Load and Add (32)	RSY-a	EBF8	¤ c IA
LAAG	$R_1,R_3,D_2(B_2)$	Load and Add (64)	RSY-a	EBE8	¤ c IA
LAAL	$R_1,R_3,D_2(B_2)$	Load and Add Logical (32)	RSY-a	EBFA	¤ c IA
LAALG	$R_1,R_3,D_2(B_2)$	Load and Add Logical (64)	RSY-a	EBEA	¤ c IA
LAE	$R_1,D_2(X_2,B_2)$	Load Address Extended	RX-a	51	¤
LAEY	$R_1,D_2(X_2,B_2)$	Load Address Extended	RXY-a	E375	¤ GE
LAM	$R_1,R_3,D_2(B_2)$	Load Access Multiple	RS-a	9A	¤
LAMY	$R_1,R_3,D_2(B_2)$	Load Access Multiple	RSY-a	EB9A	¤ LD
LAN	$R_1,R_3,D_2(B_2)$	Load and AND (32)	RSY-a	EBF4	¤ c IA
LANG	$R_1,R_3,D_2(B_2)$	Load and AND (64)	RSY-a	EBE4	¤ c IA
LAO	$R_1,R_3,D_2(B_2)$	Load and OR (32)	RSY-a	EBF6	¤ c IA
LAOG	$R_1,R_3,D_2(B_2)$	Load and OR (64)	RSY-a	EBE6	¤ c IA
LARL	R_1,RI_2	Load Address Relative Long	RIL-b	C00	N3
LASP	$D_1(B_1),D_2(B_2)$	Load Address Space Parameters	SSE	E500	p c
LAT	$R_1,D_2(X_2,B_2)$	Load and Trap (32)	RXY-a	E39F	LT
LAX	$R_1,R_3,D_2(B_2)$	Load and Exclusive OR (32)	RSY-a	EBF7	¤ c IA
LAXG	$R_1,R_3,D_2(B_2)$	Load and Exclusive OR (64)	RSY-a	EBE7	¤ c IA
LAY	$R_1,D_2(X_2,B_2)$	Load Address	RXY-a	E371	LD
LB	$R_1,D_2(X_2,B_2)$	Load Byte (32←8)	RXY-a	E376	LD
LBEAR	$D_2(B_2)$	Load BEAR	S	B200	p BE
LBH	$R_1,D_2(X_2,B_2)$	Load Byte High (32←8)	RXY-a	E3C0	HW
LBR	R_1,R_2	Load Byte (32←8)	RRE	B926	EI
LCBB	$R_1,D_2(X_2,B_2),M_3$	Load Count to Block Boundary	RXE	E727	¤ c VF
LCDBR	R_1,R_2	Load Complement (LB)	RRE	B313	¤ c
LCDFR	R_1,R_2	Load Complement (L)	RRE	B373	¤ FS
LCDR	R_1,R_2	Load Complement (LH)	RR	23	¤ c
LCEBR	R_1,R_2	Load Complement (SB)	RRE	B303	¤ c
LCER	R_1,R_2	Load Complement (SH)	RR	33	¤ c
LCGFR	R_1,R_2	Load Complement (64←32)	RRE	B913	c N
LCGR	R_1,R_2	Load Complement (64)	RRE	B903	c N
LCR	R_1,R_2	Load Complement (32)	RR	13	c
LCTL	$R_1,R_3,D_2(B_2)$	Load Control (32)	RS-a	B7	p
LCTLG	$R_1,R_3,D_2(B_2)$	Load Control (64)	RSY-a	EB2F	p N
LCXBR	R_1,R_2	Load Complement (EB)	RRE	B343	¤ c
LCXR	R_1,R_2	Load Complement (EH)	RRE	B363	¤ c
LD	$R_1,D_2(X_2,B_2)$	Load (L)	RX-a	68	¤
LDE	$R_1,D_2(X_2,B_2)$	Load Lengthened (LH←SH)	RXE	ED24	¤
LDEB	$R_1,D_2(X_2,B_2)$	Load Lengthened (LB←SB)	RXE	ED04	¤
LDEBR	R_1,R_2	Load Lengthened (LB←SB)	RRE	B304	¤
LDER	R_1,R_2	Load Lengthened (LH←SH)	RRE	B324	¤
LDETR	R_1,R_2,M_4	Load Lengthened (LD←SD)	RRF-d	B3D4	¤ TF
LDGR	R_1,R_2	Load FPR from GR (L←64)	RRE	B3C1	¤ FG
LDR	R_1,R_2	Load (L)	RR	28	¤
LDXBR	R_1,R_2	Load Rounded (LB←EB)	RRE	B345	¤
LDXBRA	R_1,M_3,R_2,M_4	Load Rounded (LB←EB)	RRF-e	B345	¤ F
LDXR	R_1,R_2	Load Rounded (LH←EH)	RR	25	¤
LDXTR	R_1,M_3,R_2,M_4	Load Rounded (LD←ED)	RRF-e	B3DD	¤ TF
LDY	$R_1,D_2(X_2,B_2)$	Load (L)	RXY-a	ED65	¤ LD
LE	$R_1,D_2(X_2,B_2)$	Load (S)	RX-a	78	¤
LEDBR	R_1,R_2	Load Rounded (SB←LB)	RRE	B344	¤
LEDBRA	R_1,M_3,R_2,M_4	Load Rounded (SB←LB)	RRF-e	B344	¤ F
LEDR	R_1,R_2	Load Rounded (SH←LH)	RR	35	¤

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
LEDTR	R_1,M_3,R_2,M_4	Load Rounded (SD←LD)	RRF-e	B3D5	¤ TF
LER	R_1,R_2	Load (S)	RR	38	¤
LEXBR	R_1,R_2	Load Rounded (SB←EB)	RRE	B346	¤
LEXBRA	R_1,M_3,R_2,M_4	Load Rounded (SB←EB)	RRF-e	B346	¤ F
LEXR	R_1,R_2	Load Rounded (SH←EH)	RRE	B366	¤
LEY	$R_1,D_2(X_2,B_2)$	Load (S)	RXY-a	ED64	¤ LD
LFAS	$D_2(B_2)$	Load FPC and Signal	S	B2BD	¤ XF
LFH	$R_1,D_2(X_2,B_2)$	Load High (32)	RXY-a	E3CA	HW
LFHAT	$R_1,D_2(X_2,B_2)$	Load and Trap (32H←32)	RXY-a	E3C8	LT
LFPC	$D_2(B_2)$	Load FPC	S	B29D	¤
LG	$R_1,D_2(X_2,B_2)$	Load (64)	RXY-a	E304	N
LGAT	$R_1,D_2(X_2,B_2)$	Load and Trap (64)	RXY-a	E385	LT
LGB	$R_1,D_2(X_2,B_2)$	Load Byte (64←8)	RXY-a	E377	LD
LGBR	R_1,R_2	Load Byte (64←8)	RRE	B906	EI
LGDR	R_1,R_2	Load GR from FPR (64←L)	RRE	B3CD	¤ FG
LGF	$R_1,D_2(X_2,B_2)$	Load (64←32)	RXY-a	E314	N
LGFI	R_1,I_2	Load Immediate (64←32)	RIL-a	C01	EI
LGFR	R_1,R_2	Load (64←32)	RRE	B914	N
LGFRL	R_1,RI_2	Load Relative Long (64←32)	RIL-b	C4C	GE
LGG	$R_1,D_2(X_2,B_2)$	Load guarded (64)	RXY-a	E34C	¤ GF
LGH	$R_1,D_2(X_2,B_2)$	Load Halfword (64←16)	RXY-a	E315	N
LGHI	R_1,I_2	Load Halfword Immediate (64←16)	RI-a	A79	N
LGHR	R_1,R_2	Load Halfword (64←16)	RRE	B907	EI
LGHRL	R_1,RI_2	Load Halfword Relative Long (64←16)	RIL-b	C44	GE
LGR	R_1,R_2	Load (64)	RRE	B904	N
LGRL	R_1,RI_2	Load Relative Long (64)	RIL-b	C48	GE
LGSC	$R_1,D_2(X_2,B_2)$	Load guarded storage controls	RXY-a	E34D	¤ GF
LH	$R_1,D_2(X_2,B_2)$	Load Halfword (32←16)	RX-a	48	
LHH	$R_1,D_2(X_2,B_2)$	Load Halfword High (32←16)	RXY-a	E3C4	HW
LHI	R_1,I_2	Load Halfword Immediate (32←16)	RI-a	A78	
LHR	R_1,R_2	Load Halfword (32←16)	RRE	B927	EI
LHRL	R_1,RI_2	Load Halfword Relative Long (32←16)	RIL-b	C45	GE
LHY	$R_1,D_2(X_2,B_2)$	Load Halfword (32←16)	RXY-a	E378	LD
LLC	$R_1,D_2(X_2,B_2)$	Load Logical Character (32←8)	RXY-a	E394	EI
LLCH	$R_1,D_2(X_2,B_2)$	Load Logical Character High (32←8)	RXY-a	E3C2	HW
LLCR	R_1,R_2	Load Logical Character (32←8)	RRE	B994	EI
LLGC	$R_1,D_2(X_2,B_2)$	Load Logical Character (64←8)	RXY-a	E390	N
LLGCR	R_1,R_2	Load Logical Character (64←8)	RRE	B984	EI
LLGF	$R_1,D_2(X_2,B_2)$	Load Logical (64←32)	RXY-a	E316	N
LLGFAT	$R_1,D_2(X_2,B_2)$	Load and Trap (64←32)	RXY-a	E39D	LT
LLGFR	R_1,R_2	Load Logical (64←32)	RRE	B916	N
LLGFRL	R_1,RI_2	Load Logical Relative Long (64←32)	RIL-b	C4E	GE
LLGFSG	$R_1,D_2(X_2,B_2)$	Load logical and shift guarded (64←32)	RXY-a	E348	¤ GF
LLGH	$R_1,D_2(X_2,B_2)$	Load Logical Halfword (64←16)	RXY-a	E391	N
LLGHR	R_1,R_2	Load Logical Halfword (64←16)	RRE	B985	EI
LLGHRL	R_1,RI_2	Load Logical Halfword Relative Long (64←16)	RIL-b	C46	GE
LLGT	$R_1,D_2(X_2,B_2)$	Load Logical Thirty One Bits (64←31)	RXY-a	E317	N
LLGTAT	$R_1,D_2(X_2,B_2)$	Load Logical Thirty One Bits and Trap (64←31)	RXY-a	E39C	LT
LLGTR	R_1,R_2	Load Logical Thirty One Bits (64←31)	RRE	B917	N
LLH	$R_1,D_2(X_2,B_2)$	Load Logical Halfword (32←16)	RXY-a	E395	EI
LLHH	$R_1,D_2(X_2,B_2)$	Load Logical Halfword High (32←16)	RXY-a	E3C6	HW
LLHR	R_1,R_2	Load Logical Halfword (32←16)	RRE	B995	EI
LLHRL	R_1,RI_2	Load Logical Halfword Relative Long (32←16)	RIL-b	C42	GE
LLIHF	R_1,I_2	Load Logical Immediate (high)	RIL-a	C0E	EI
LLIHH	R_1,I_2	Load Logical Immediate (high high)	RI-a	A5C	N
LLIHL	R_1,I_2	Load Logical Immediate (high low)	RI-a	A5D	N
LLILF	R_1,I_2	Load Logical Immediate (low)	RIL-a	C0F	EI
LLILH	R_1,I_2	Load Logical Immediate (low high)	RI-a	A5E	N

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
LLILL	R_1,I_2	Load Logical Immediate (low low)	RI-a	A5F	N
LLZRGF	$R_1,D_2(X_2,B_2)$	Load Logical and Zero Rightmost Byte (32)	RXY-a	E33A	LZ
LM	$R_1,R_3,D_2(B_2)$	Load Multiple (32)	RS-a	98	
LMD	$R_1,R_3,D_2(B_2),D_4(B_4)$	Load Multiple Disjoint (64←32&32)	SS-e	EF	¤ N
LMG	$R_1,R_3,D_2(B_2)$	Load Multiple (64)	RSY-a	EB04	N
LMH	$R_1,R_3,D_2(B_2)$	Load Multiple High	RSY-a	EB96	N
LMY	$R_1,R_3,D_2(B_2)$	Load Multiple (32)	RSY-a	EB98	LD
LNDBR	R_1,R_2	Load Negative (LB)	RRE	B311	¤ c
LNDFR	R_1,R_2	Load Negative (L)	RRE	B371	¤ FS
LNDR	R_1,R_2	Load Negative (LH)	RR	21	¤ c
LNEBR	R_1,R_2	Load Negative (SB)	RRE	B301	¤ c
LNER	R_1,R_2	Load Negative (SH)	RR	31	¤ c
LNGFR	R_1,R_2	Load Negative (64←32)	RRE	B911	c N
LNGR	R_1,R_2	Load Negative (64)	RRE	B901	c N
LNR	R_1,R_2	Load Negative (32)	RR	11	c
LNXBR	R_1,R_2	Load Negative (EB)	RRE	B341	¤ c
LNXR	R_1,R_2	Load Negative (EH)	RRE	B361	¤ c
LOC	$R_1,D_2(B_2),M_3$	Load on Condition (32)	RSY-b	EBF2	L1
LOCFH	$R_1,D_2(B_2),M_3$	Load High on Condition (32)	RSY-b	EBE0	L2
LOCFHR	R_1,R_2,M_3	Load High on Condition (32)	RRF-c	B9E0	L2
LOCG	$R_1,D_2(B_2),M_3$	Load on Condition (64)	RSY-b	EBE2	L1
LOCGHI	R_1,I_2,M_3	Load Halfword Immediate on Condition (64←16)	RIE-g	EC46	L2
LOCGR	R_1,R_2,M_3	Load on Condition (64)	RRF-c	B9E2	L1
LOCHHI	R_1,I_2,M_3	Load Halfword High Immediate on Condition (32←16)	RIE-g	EC4E	L2
LOCHI	R_1,I_2,M_3	Load Halfword Immediate on Condition (32←16)	RIE-g	EC42	L2
LOCR	R_1,R_2,M_3	Load on Condition (32)	RRF-c	B9F2	L1
LPD	$R_3,D_1(B_1),D_2(B_2)$	Load Pair Disjoint (32)	SSF	C84	c IA
LPDBR	R_1,R_2	Load Positive (LB)	RRE	B310	¤ c
LPDFR	R_1,R_2	Load Positive (L)	RRE	B370	¤ FS
LPDG	$R_3,D_1(B_1),D_2(B_2)$	Load Pair Disjoint (64)	SSF	C85	c IA
LPDR	R_1,R_2	Load Positive (LH)	RR	20	¤ c
LPEBR	R_1,R_2	Load Positive (SB)	RRE	B300	¤ c
LPER	R_1,R_2	Load Positive (SH)	RR	30	¤ c
LPGFR	R_1,R_2	Load Positive (64←32)	RRE	B910	c N
LPGR	R_1,R_2	Load Positive (64)	RRE	B900	c N
LPQ	$R_1,D_2(X_2,B_2)$	Load Pair from Quadword (64&64←128)	RXY-a	E38F	¤ N
LPR	R_1,R_2	Load Positive (32)	RR	10	c
LPSW	$D_2(B_2)$	Load PSW	SI	82	p n
LPSWE	$D_2(B_2)$	Load PSW Extended	S	B2B2	p n N
LPSWEY	$D_1(B_1)$	Load PSW Extended	SIY	EB71	p n BE
LPTEA	R_1,R_3,R_2,M_4	Load Page-Table-Entry Address	RRF-b	B9AA	p c D2
LPXBR	R_1,R_2	Load Positive (EB)	RRE	B340	¤ c
LPXR	R_1,R_2	Load Positive (EH)	RRE	B360	¤ c
LR	R_1,R_2	Load (32)	RR	18	
LRA	$R_1,D_2(X_2,B_2)$	Load Real Address (32)	RX-a	B1	p c
LRAG	$R_1,D_2(X_2,B_2)$	Load Real Address (64)	RXY-a	E303	p c N
LRAY	$R_1,D_2(X_2,B_2)$	Load Real Address (32)	RXY-a	E313	p c LD
LRDR	R_1,R_2	Load Rounded (LH←EH)	RR	25	¤
LRER	R_1,R_2	Load Rounded (SH←LH)	RR	35	¤
LRL	R_1,RI_2	Load Relative Long (32)	RIL-b	C4D	GE
LRV	$R_1,D_2(X_2,B_2)$	Load Reversed (32)	RXY-a	E31E	N3
LRVG	$R_1,D_2(X_2,B_2)$	Load Reversed (64)	RXY-a	E30F	N
LRVGR	R_1,R_2	Load Reversed (64)	RRE	B90F	N
LRVH	$R_1,D_2(X_2,B_2)$	Load Reversed (16)	RXY-a	E31F	N3
LRVR	R_1,R_2	Load Reversed (32)	RRE	B91F	N3
LT	$R_1,D_2(X_2,B_2)$	Load and Test (32)	RXY-a	E312	c EI
LTDBR	R_1,R_2	Load and Test (LB)	RRE	B312	¤ c
LTDR	R_1,R_2	Load and Test (LH)	RR	22	¤ c

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
LTDTR	R_1,R_2	Load and Test (LD)	RRE	B3D6	¤ c TF
LTEBR	R_1,R_2	Load and Test (SB)	RRE	B302	¤ c
LTER	R_1,R_2	Load and Test (SH)	RR	32	¤ c
LTG	$R_1,D_2(X_2,B_2)$	Load and Test (64)	RXY-a	E302	c EI
LTGF	$R_1,D_2(X_2,B_2)$	Load And Test (64←32)	RXY-a	E332	c GE
LTGFR	R_1,R_2	Load and Test (64←32)	RRE	B912	c N
LTGR	R_1,R_2	Load and Test (64)	RRE	B902	c N
LTR	R_1,R_2	Load and Test (32)	RR	12	c
LTXBR	R_1,R_2	Load and Test (EB)	RRE	B342	¤ c
LTXR	R_1,R_2	Load and Test (EH)	RRE	B362	¤ c
LTXTR	R_1,R_2	Load and Test (ED)	RRE	B3DE	¤ c TF
LURA	R_1,R_2	Load Using Real Address (32)	RRE	B24B	p
LURAG	R_1,R_2	Load Using Real Address (64)	RRE	B905	p N
LXD	$R_1,D_2(X_2,B_2)$	Load Lengthened (EH←LH)	RXE	ED25	¤
LXDB	$R_1,D_2(X_2,B_2)$	Load Lengthened (EB←LB)	RXE	ED05	¤
LXDBR	R_1,R_2	Load Lengthened (EB←LB)	RRE	B305	¤
LXDR	R_1,R_2	Load Lengthened (EH←LH)	RRE	B325	¤
LXDTR	R_1,R_2,M_4	Load Lengthened (ED←LD)	RRF-d	B3DC	¤ TF
LXE	$R_1,D_2(X_2,B_2)$	Load Lengthened (EH←SH)	RXE	ED26	¤
LXEB	$R_1,D_2(X_2,B_2)$	Load Lengthened (EB←SB)	RXE	ED06	¤
LXEBR	R_1,R_2	Load Lengthened (EB←SB)	RRE	B306	¤
LXER	R_1,R_2	Load Lengthened (EH←SH)	RRE	B326	¤
LXR	R_1,R_2	Load (E)	RRE	B365	¤
LY	$R_1,D_2(X_2,B_2)$	Load (32)	RXY-a	E358	LD
LZDR	R_1	Load Zero (L)	RRE	B375	¤
LZER	R_1	Load Zero (S)	RRE	B374	¤
LZRF	$R_1,D_2(X_2,B_2)$	Load and Zero Rightmost Byte (32)	RXY-a	E33B	LZ
LZRG	$R_1,D_2(X_2,B_2)$	Load and Zero Rightmost Byte (64)	RXY-a	E32A	LZ
LZXR	R_1	Load Zero (E)	RRE	B376	¤
M	$R_1,D_2(X_2,B_2)$	Multiply (64←32)	RX-a	5C	
MAD	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add (LH)	RXF	ED3E	¤ HM
MADB	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add (LB)	RXF	ED1E	¤
MADBR	R_1,R_3,R_2	Multiply and Add (LB)	RRD	B31E	¤
MADR	R_1,R_3,R_2	Multiply and Add (LH)	RRD	B33E	¤ HM
MAE	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add (SH)	RXF	ED2E	¤ HM
MAEB	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add (SB)	RXF	ED0E	¤
MAEBR	R_1,R_3,R_2	Multiply and Add (SB)	RRD	B30E	¤
MAER	R_1,R_3,R_2	Multiply and Add (SH)	RRD	B32E	¤ HM
MAY	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add Unnormalized (EH←LH)	RXF	ED3A	¤ UE
MAYH	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add Unnormalized (EH_H←LH)	RXF	ED3C	¤ UE
MAYHR	R_1,R_3,R_2	Multiply and Add Unnormalized (EH_H←LH)	RRD	B33C	¤ UE
MAYL	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Add Unnormalized (EH_L←LH)	RXF	ED38	¤ UE
MAYLR	R_1,R_3,R_2	Multiply and Add Unnormalized (EH_L←LH)	RRD	B338	¤ UE
MAYR	R_1,R_3,R_2	Multiply and Add Unnormalized (EH←LH)	RRD	B33A	¤ UE
MC	$D_1(B_1),I_2$	Monitor Call	SI	AF	¤
MD	$R_1,D_2(X_2,B_2)$	Multiply (LH)	RX-a	6C	¤
MDB	$R_1,D_2(X_2,B_2)$	Multiply (LB)	RXE	ED1C	¤
MDBR	R_1,R_2	Multiply (LB)	RRE	B31C	¤
MDE	$R_1,D_2(X_2,B_2)$	Multiply (LH←SH)	RX-a	7C	¤
MDEB	$R_1,D_2(X_2,B_2)$	Multiply (LB←SB)	RXE	ED0C	¤
MDEBR	R_1,R_2	Multiply (LB←SB)	RRE	B30C	¤
MDER	R_1,R_2	Multiply (LH←SH)	RR	3C	¤
MDR	R_1,R_2	Multiply (LH)	RR	2C	¤
MDTR	R_1,R_2,R_3	Multiply (LD)	RRF-a	B3D0	¤ TF
MDTRA	R_1,R_2,R_3,M_4	Multiply (LD)	RRF-a	B3D0	¤ F
ME	$R_1,D_2(X_2,B_2)$	Multiply (LH←SH)	RX-a	7C	¤
MEE	$R_1,D_2(X_2,B_2)$	Multiply (SH)	RXE	ED37	¤
MEEB	$R_1,D_2(X_2,B_2)$	Multiply (SB)	RXE	ED17	¤
MEEBR	R_1,R_2	Multiply (SB)	RRE	B317	¤
MEER	R_1,R_2	Multiply (SH)	RRE	B337	¤

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
MER	R_1,R_2	Multiply (LH←SH)	RR	3C	¤
MFY	$R_1,D_2(X_2,B_2)$	Multiply (64←32)	RXY-a	E35C	GE
MG	$R_1,D_2(X_2,B_2)$	Multiply (128←64)	RXY-a	E384	MI2
MGH	$R_1,D_2(X_2,B_2)$	Multiply Halfword(64←16)	RXY-a	E33C	MI2
MGHI	R_1,I_2	Multiply Halfword Immediate (64←16)	RI-a	A7D	N
MGRK	R_1,R_2,R_3	Multiply (128←64)	RRF-a	B9EC	MI2
MH	$R_1,D_2(X_2,B_2)$	Multiply Halfword (32←16)	RX-a	4C	
MHI	R_1,I_2	Multiply Halfword Immediate (32←16)	RI-a	A7C	
MHY	$R_1,D_2(X_2,B_2)$	Multiply Halfword (64←16)	RXY-a	E37C	GE
ML	$R_1,D_2(X_2,B_2)$	Multiply Logical (64←32)	RXY-a	E396	N3
MLG	$R_1,D_2(X_2,B_2)$	Multiply Logical (128←64)	RXY-a	E386	N
MLGR	R_1,R_2	Multiply Logical (128←64)	RRE	B986	N
MLR	R_1,R_2	Multiply Logical (64←32)	RRE	B996	N3
MP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Multiply Decimal	SS-b	FC	¤
MR	R_1,R_2	Multiply (64←32)	RR	1C	
MS	$R_1,D_2(X_2,B_2)$	Multiply Single (32)	RX-a	71	
MSC	$R_1,D_2(X_2,B_2)$	Multiply Single (32)	RXY-a	E353	c MI2
MSCH	$D_2(B_2)$	Modify Subchannel	S	B232	p c
MSD	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Subtract (LH)	RXF	ED3F	¤ HM
MSDB	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Subtract (LB)	RXF	ED1F	¤
MSDBR	R_1,R_3,R_2	Multiply and Subtract (LB)	RRD	B31F	¤
MSDR	R_1,R_3,R_2	Multiply and Subtract (LH)	RRD	B33F	¤ HM
MSE	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Subtract (SH)	RXF	ED2F	¤ HM
MSEB	$R_1,R_3,D_2(X_2,B_2)$	Multiply and Subtract (SB)	RXF	ED0F	¤
MSEBR	R_1,R_3,R_2	Multiply and Subtract (SB)	RRD	B30F	¤
MSER	R_1,R_3,R_2	Multiply and Subtract (SH)	RRD	B32F	¤ HM
MSFI	R_1,I_2	Multiply Single Immediate (32)	RIL-a	C21	GE
MSG	$R_1,D_2(X_2,B_2)$	Multiply Single (64)	RXY-a	E30C	N
MSGC	$R_1,D_2(X_2,B_2)$	Multiply Single (64)	RXY-a	E383	c MI2
MSGF	$R_1,D_2(X_2,B_2)$	Multiply Single (64←32)	RXY-a	E31C	N
MSGFI	R_1,I_2	Multiply Single Immediate (64←32)	RIL-a	C20	GE
MSGFR	R_1,R_2	Multiply Single (64←32)	RRE	B91C	N
MSGR	R_1,R_2	Multiply Single (64)	RRE	B90C	N
MSGRKC	R_1,R_2,R_3	Multiply Single (64)	RRF-a	B9ED	c MI2
MSR	R_1,R_2	Multiply Single (32)	RRE	B252	
MSRKC	R_1,R_2,R_3	Multiply Single (32)	RRF-a	B9FD	c MI2
MSTA	R_1	Modify Stacked State	RRE	B247	¤
MSY	$R_1,D_2(X_2,B_2)$	Multiply Single (32)	RXY-a	E351	LD
MVC	$D_1(L,B_1),D_2(B_2)$	Move (character)	SS-a	D2	¤
MVCDK	$D_1(B_1),D_2(B_2)$	Move with Destination Key	SSE	E50F	q
MVCIN	$D_1(L,B_1),D_2(B_2)$	Move Inverse	SS-a	E8	¤
MVCK	$D_1(R_1,B_1),D_2(B_2),R_3$	Move with Key	SS-d	D9	q c
MVCL	R_1,R_2	Move Long	RR	0E	i ¤ c
MVCLE	$R_1,R_3,D_2(B_2)$	Move Long Extended	RS-a	A8	¤ c
MVCLU	$R_1,R_3,D_2(B_2)$	Move Long Unicode	RSY-a	EB8E	¤ c E2
MVCOS	$D_1(B_1),D_2(B_2),R_3$	Move with Optional Specifications	SSF	C80	q c MO
MVCP	$D_1(R_1,B_1),D_2(B_2),R_3$	Move to Primary	SS-d	DA	q c
MVCRL	$D_1(B_1),D_2(B_2)$	Move Right to Left	SSE	E50A	¤ MI3
MVCS	$D_1(R_1,B_1),D_2(B_2),R_3$	Move to Secondary	SS-d	DB	q c
MVCSK	$D_1(B_1),D_2(B_2)$	Move with Source Key	SSE	E50E	q
MVGHI	$D_1(B_1),I_2$	Move (64←16)	SIL	E548	GE
MVHHI	$D_1(B_1),I_2$	Move (16←16)	SIL	E544	GE
MVHI	$D_1(B_1),I_2$	Move (32←16)	SIL	E54C	GE
MVI	$D_1(B_1),I_2$	Move Immediate	SI	92	
MVIY	$D_1(B_1),I_2$	Move Immediate	SIY	EB52	LD
MVN	$D_1(L,B_1),D_2(B_2)$	Move Numerics	SS-a	D1	¤
MVO	$D_1(L_1,B_1),D_2(L_2,B_2)$	Move with Offset	SS-b	F1	¤
MVPG	R_1,R_2	Move Page	RRE	B254	q c
MVST	R_1,R_2	Move String	RRE	B255	¤ c
MVZ	$D_1(L,B_1),D_2(B_2)$	Move Zones	SS-a	D3	¤

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
MXBR	R_1,R_2	Multiply (EB)	RRE	B34C	¤
MXD	$R_1,D_2(X_2,B_2)$	Multiply (EH←LH)	RX-a	67	¤
MXDB	$R_1,D_2(X_2,B_2)$	Multiply (EB←LB)	RXE	ED07	¤
MXDBR	R_1,R_2	Multiply (EB←LB)	RRE	B307	¤
MXDR	R_1,R_2	Multiply (EH←LH)	RR	27	¤
MXR	R_1,R_2	Multiply (EH)	RR	26	¤
MXTR	R_1,R_2,R_3	Multiply (ED)	RRF-a	B3D8	¤ TF
MXTRA	R_1,R_2,R_3,M_4	Multiply (ED)	RRF-a	B3D8	¤ F
MY	$R_1,R_3,D_2(X_2,B_2)$	Multiply Unnormalized (EH←LH)	RXF	ED3B	¤ UE
MYH	$R_1,R_3,D_2(X_2,B_2)$	Multiply Unnormalized (EH_H←LH)	RXF	ED3D	¤ UE
MYHR	R_1,R_3,R_2	Multiply Unnormalized (EH_H←LH)	RRD	B33D	¤ UE
MYL	$R_1,R_3,D_2(X_2,B_2)$	Multiply Unnormalized (EH_L←LH)	RXF	ED39	¤ UE
MYLR	R_1,R_3,R_2	Multiply Unnormalized (EH_L←LH)	RRD	B339	¤ UE
MYR	R_1,R_3,R_2	Multiply Unnormalized (EH←LH)	RRD	B33B	¤ UE
N	$R_1,D_2(X_2,B_2)$	AND (32)	RX-a	54	c
NC	$D_1(L,B_1),D_2(B_2)$	AND (character)	SS-a	D4	¤ c
NCGRK	R_1,R_2,R_3	AND with Complement (64)	RRF-a	B9E5	c MI3
NCRK	R_1,R_2,R_3	AND with Complement (32)	RRF-a	B9F5	c MI3
NG	$R_1,D_2(X_2,B_2)$	AND (64)	RXY-a	E380	c N
NGR	R_1,R_2	AND (64)	RRE	B980	c N
NGRK	R_1,R_2,R_3	AND (64)	RRF-a	B9E4	c DO
NI	$D_1(B_1),I_2$	AND Immediate	SI	94	c
NIAI	I_1,I_2	Next Instruction Access Intent	IE	B2FA	EH
NIHF	R_1,I_2	AND Immediate (high)	RIL-a	C0A	c EI
NIHH	R_1,I_2	AND Immediate (high high)	RI-a	A54	c N
NIHL	R_1,I_2	AND Immediate (high low)	RI-a	A55	c N
NILF	R_1,I_2	AND Immediate (low)	RIL-a	C0B	c EI
NILH	R_1,I_2	AND Immediate (low high)	RI-a	A56	c N
NILL	R_1,I_2	AND Immediate (low low)	RI-a	A57	c N
NIY	$D_1(B_1),I_2$	AND Immediate	SIY	EB54	c LD
NNGRK	R_1,R_2,R_3	NAND (64)	RRF-a	B964	c MI3
NNPA		Neural Network Processing Assist	RRE	B93B	c NN
NNRK	R_1,R_2,R_3	NAND (32)	RRF-a	B974	c MI3
NOGRK	R_1,R_2,R_3	NOR (64)	RRF-a	B966	c MI3
NORK	R_1,R_2,R_3	NOR (32)	RRF-a	B976	c MI3
NR	R_1,R_2	AND (32)	RR	14	c
NRK	R_1,R_2,R_3	AND (32)	RRF-a	B9F4	c DO
NTSTG	$R_1,D_2(X_2,B_2)$	Nontransactional Store (64)	RXY-a	E325	¤ TX
NXGRK	R_1,R_2,R_3	NOT Exclusive OR (64)	RRF-a	B967	c MI3
NXRK	R_1,R_2,R_3	NOT Exclusive OR (32)	RRF-a	B977	c MI3
NY	$R_1,D_2(X_2,B_2)$	AND (32)	RXY-a	E354	c LD
O	$R_1,D_2(X_2,B_2)$	OR (32)	RX-a	56	c
OC	$D_1(L,B_1),D_2(B_2)$	OR (character)	SS-a	D6	¤ c
OCGRK	R_1,R_2,R_3	OR with Complement (64)	RRF-a	B965	c MI3
OCRK	R_1,R_2,R_3	OR with Complement (32)	RRF-a	B975	c MI3
OG	$R_1,D_2(X_2,B_2)$	OR (64)	RXY-a	E381	c N
OGR	R_1,R_2	OR (64)	RRE	B981	c N
OGRK	R_1,R_2,R_3	OR (64)	RRF-a	B9E6	c DO
OI	$D_1(B_1),I_2$	OR Immediate	SI	96	c
OIHF	R_1,I_2	OR Immediate (high)	RIL-a	C0C	c EI
OIHH	R_1,I_2	OR Immediate (high high)	RI-a	A58	c N
OIHL	R_1,I_2	OR Immediate (high low)	RI-a	A59	c N
OILF	R_1,I_2	OR Immediate (low)	RIL-a	C0D	c EI
OILH	R_1,I_2	OR Immediate (low high)	RI-a	A5A	c N
OILL	R_1,I_2	OR Immediate (low low)	RI-a	A5B	c N
OIY	$D_1(B_1),I_2$	OR Immediate	SIY	EB56	c LD
OR	R_1,R_2	OR (32)	RR	16	c
ORK	R_1,R_2,R_3	OR (32)	RRF-a	B9F6	c DO
OY	$R_1,D_2(X_2,B_2)$	OR (32)	RXY-a	E356	c LD
PACK	$D_1(L_1,B_1),D_2(L_2,B_2)$	Pack	SS-b	F2	¤

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
PALB		Purge ALB	RRE	B248	p
PC	$D_2(B_2)$	Program Call	S	B218	q
PCC		Perform Cryptographic Computation	RRE	B92C	¤ c M4
PCKMO		Perform Crypto. Key Mgmt. Operations	RRE	B928	M3
PFD	$M_1,D_2(X_2,B_2)$	Prefetch Data	RXY-b	E336	¤ GE
PFDRL	M_1,RI_2	Prefetch Data Relative Long	RIL-c	C62	¤ GE
PFMF	R_1,R_2	Perform Frame Management Function	RRE	B9AF	p ED1
PFPO		Perform Floating-Point Operation	E	010A	¤ PF
PGIN	R_1,R_2	Page In	RRE	B22E	p c ES
PGOUT	R_1,R_2	Page Out	RRE	B22F	p c ES
PKA	$D_1(B_1),D_2(L_2,B_2)$	Pack ASCII	SS-f	E9	¤ E2
PKU	$D_1(B_1),D_2(L_2,B_2)$	Pack Unicode	SS-f	E1	¤ E2
PLO	$R_1,D_2(B_2),R_3,D_4(B_4)$	Perform Locked Operation	SS-e	EE	¤ c
POPCNT	$R_1,R_2[,M_3]$	Population Count	RRF-c	B9E1	c PK
PPA	R_1,R_2,M_3	Perform Processor Assist	RRF-c	B2E8	PA
PR		Program Return	E	0101	q n
PRNO	R_1,R_2	Perform Random Number Operation	RRE	B93C	M5
PT	R_1,R_2	Program Transfer	RRE	B228	q
PTF	R_1	Perform Topology Function	RRE	B9A2	c p CT
PTFF		Perform Timing-Facility Function	E	0104	q c
PTI	R_1,R_2	Program Transfer with Instance	RRE	B99E	q RA
PTLB		Purge TLB	S	B20D	p
QADTR	R_1,R_3,R_2,M_4	Quantize (LD)	RRF-b	B3F5	¤ TF
QAXTR	R_1,R_3,R_2,M_4	Quantize (ED)	RRF-b	B3FD	¤ TF
QPACI	$D_2(B_2)$	Query Processor Activity Counter	S	B28F	p PI
RCHP		Reset Channel Path	S	B23B	p c
RDP	$R_1,R_3,R_2[,M_4]$	Reset DAT Protection	RRF-b	B98B	p DP
RISBG	$R_1,R_2,I_3,I_4[,I_5]$	Rotate then Insert Selected Bits (64)	RIE-f	EC55	c GE
RISBGN	$R_1,R_2,I_3,I_4[,I_5]$	Rotate then Insert Selected Bits (64)	RIE-f	EC59	MI1
RISBHG	$R_1,I_2,I_3,I_4[,I_5]$	Rotate then Insert Selected Bits High (32)	RIE-f	EC5D	HW
RISBLG	$R_1,I_2,I_3,I_4[,I_5]$	Rotate then Insert Selected Bits Low (32)	RIE-f	EC51	HW
RLL	$R_1,R_3,D_2(B_2)$	Rotate Left Single Logical (32)	RSY-a	EB1D	N3
RLLG	$R_1,R_3,D_2(B_2)$	Rotate Left Single Logical (64)	RSY-a	EB1C	N
RNSBG	$R_1,R_2,I_3,I_4[,I_5]$	Rotate then AND Selected Bits (64)	RIE-f	EC54	c GE
ROSBG	$R_1,R_2,I_3,I_4[,I_5]$	Rotate then OR Selected Bits (64)	RIE-f	EC56	c GE
RP	$D_2(B_2)$	Resume Program	S	B277	q n
RRBE	R_1,R_2	Reset Reference Bit Extended	RRE	B22A	p c
RRBM	R_1,R_2	Reset Reference Bits Multiple	RRE	B9AE	p RB
RRDTR	R_1,R_3,R_2,M_4	Reround (LD)	RRF-b	B3F7	¤ TF
RRXTR	R_1,R_3,R_2,M_4	Reround (ED)	RRF-b	B3FF	¤ TF
RSCH		Resume Subchannel	S	B238	p c
RXSBG	$R_1,R_2,I_3,I_4[,I_5]$	Rotate then Exclusive OR Selected Bits (64)	RIE-f	EC57	c GE
S	$R_1,D_2(X_2,B_2)$	Subtract (32)	RX-a	5B	c
SAC	$D_2(B_2)$	Set Address Space Control	S	B219	q
SACF	$D_2(B_2)$	Set Address Space Control Fast	S	B279	q
SAL		Set Address Limit	S	B237	p
SAM24		Set Addressing Mode (24)	E	010C	¤ N3
SAM31		Set Addressing Mode (31)	E	010D	¤ N3
SAM64		Set Addressing Mode (64)	E	010E	¤ N
SAR	R_1,R_2	Set Access	RRE	B24E	¤
SCHM		Set Channel Monitor	S	B23C	p
SCK	$D_2(B_2)$	Set Clock	S	B204	p c
SCKC	$D_2(B_2)$	Set Clock Comparator	S	B206	p
SCKPF		Set Clock Programmable Field	E	0107	p
SD	$R_1,D_2(X_2,B_2)$	Subtract Normalized (LH)	RX-a	6B	¤ c
SDB	$R_1,D_2(X_2,B_2)$	Subtract (LB)	RXE	ED1B	¤ c
SDBR	R_1,R_2	Subtract (LB)	RRE	B31B	¤ c
SDR	R_1,R_2	Subtract Normalized (LH)	RR	2B	¤ c
SDTR	R_1,R_2,R_3	Subtract (LD)	RRF-a	B3D3	¤ c TF
SDTRA	R_1,R_2,R_3,M_4	Subtract (LD)	RRF-a	B3D3	¤ c F
SE	$R_1,D_2(X_2,B_2)$	Subtract Normalized (SH)	RX-a	7B	¤ c

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
SEB	$R_1,D_2(X_2,B_2)$	Subtract (SB)	RXE	ED0B	¤ c
SEBR	R_1,R_2	Subtract (SB)	RRE	B30B	¤ c
SELFHR	R_1,R_2,R_3,M_4	Select High (32)	RRF-a	B9C0	MI3
SELGR	R_1,R_2,R_3,M_4	Select (64)	RRF-a	B9E3	MI3
SELR	R_1,R_2,R_3,M_4	Select (32)	RRF-a	B9F0	MI3
SER	R_1,R_2	Subtract Normalized (SH)	RR	3B	¤ c
SFASR	R_1	Set FPC and Signal	RRE	B385	¤ XF
SFPC	R_1	Set FPC	RRE	B384	¤
SG	$R_1,D_2(X_2,B_2)$	Subtract (64)	RXY-a	E309	c N
SGF	$R_1,D_2(X_2,B_2)$	Subtract (64←32)	RXY-a	E319	c N
SGFR	R_1,R_2	Subtract (64←32)	RRE	B919	c N
SGH	$R_1,D_2(X_2,B_2)$	Subtract Halfword (64←16)	RXY-a	E339	c MI2
SGR	R_1,R_2	Subtract (64)	RRE	B909	c N
SGRK	R_1,R_2,R_3	Subtract (64)	RRF-a	B9E9	c DO
SH	$R_1,D_2(X_2,B_2)$	Subtract Halfword (32←16)	RX-a	4B	c
SHHHR	R_1,R_2,R_3	Subtract High (32)	RRF-a	B9C9	c HW
SHHLR	R_1,R_2,R_3	Subtract High (32)	RRF-a	B9D9	c HW
SHY	$R_1,D_2(X_2,B_2)$	Subtract Halfword (32←16)	RXY-a	E37B	c LD
SIE	$D_2(B_2)$	Start Interpretive Execution	S	B214	i p
SIGP	$R_1,R_3,D_2(B_2)$	Signal Processor	RS-a	AE	p c
SL	$R_1,D_2(X_2,B_2)$	Subtract Logical (32)	RX-a	5F	c
SLA	$R_1,D_2(B_2)$	Shift Left Single (32)	RS-a	8B	c
SLAG	$R_1,R_3,D_2(B_2)$	Shift Left Single (64)	RSY-a	EB0B	c N
SLAK	$R_1,R_3,D_2(B_2)$	Shift Left Single (32)	RSY-a	EBDD	c DO
SLB	$R_1,D_2(X_2,B_2)$	Subtract Logical with Borrow (32)	RXY-a	E399	c N3
SLBG	$R_1,D_2(X_2,B_2)$	Subtract Logical with Borrow (64)	RXY-a	E389	c N
SLBGR	R_1,R_2	Subtract Logical with Borrow (64)	RRE	B989	c N
SLBR	R_1,R_2	Subtract Logical with Borrow (32)	RRE	B999	c N3
SLDA	$R_1,D_2(B_2)$	Shift Left Double (64)	RS-a	8F	c
SLDL	$R_1,D_2(B_2)$	Shift Left Double Logical (64)	RS-a	8D	
SLDT	$R_1,R_3,D_2(X_2,B_2)$	Shift Significand Left (LD)	RXF	ED40	¤ TF
SLFI	R_1,I_2	Subtract Logical Immediate (32)	RIL-a	C25	c EI
SLG	$R_1,D_2(X_2,B_2)$	Subtract Logical (64)	RXY-a	E30B	c N
SLGF	$R_1,D_2(X_2,B_2)$	Subtract Logical (64←32)	RXY-a	E31B	c N
SLGFI	R_1,I_2	Subtract Logical Immediate (64←32)	RIL-a	C24	c EI
SLGFR	R_1,R_2	Subtract Logical (64←32)	RRE	B91B	c N
SLGR	R_1,R_2	Subtract Logical (64)	RRE	B90B	c N
SLGRK	R_1,R_2,R_3	Subtract Logical (64)	RRF-a	B9EB	c DO
SLHHHR	R_1,R_2,R_3	Subtract Logical High (32)	RRF-a	B9CB	c HW
SLHHLR	R_1,R_2,R_3	Subtract Logical High (32)	RRF-a	B9DB	c HW
SLL	$R_1,D_2(B_2)$	Shift Left Single Logical (32)	RS-a	89	
SLLG	$R_1,R_3,D_2(B_2)$	Shift Left Single Logical (64)	RSY-a	EB0D	N
SLLK	$R_1,R_3,D_2(B_2)$	Shift Left Single Logical (32)	RSY-a	EBDF	DO
SLR	R_1,R_2	Subtract Logical (32)	RR	1F	c
SLRK	R_1,R_2,R_3	Subtract Logical (32)	RRF-a	B9FB	c DO
SLXT	$R_1,R_3,D_2(X_2,B_2)$	Shift Significand Left (ED)	RXF	ED48	¤ TF
SLY	$R_1,D_2(X_2,B_2)$	Subtract Logical (32)	RXY-a	E35F	c LD
SORTL	R_1,R_2	Sort Lists	RRE	B938	c SL
SP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Subtract Decimal	SS-b	FB	¤ c
SPKA	$D_2(B_2)$	Set PSW Key from Address	S	B20A	q
SPM	R_1	Set Program Mask	RR	04	n
SPT	$D_2(B_2)$	Set CPU Timer	S	B208	p
SPX	$D_2(B_2)$	Set Prefix	S	B210	p
SQD	$R_1,D_2(X_2,B_2)$	Square Root (LH)	RXE	ED35	¤
SQDB	$R_1,D_2(X_2,B_2)$	Square Root (LB)	RXE	ED15	¤
SQDBR	R_1,R_2	Square Root (LB)	RRE	B315	¤
SQDR	R_1,R_2	Square Root (LH)	RRE	B244	¤
SQE	$R_1,D_2(X_2,B_2)$	Square Root (SH)	RXE	ED34	¤
SQEB	$R_1,D_2(X_2,B_2)$	Square Root (SB)	RXE	ED14	¤
SQEBR	R_1,R_2	Square Root (SB)	RRE	B314	¤

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
SQER	R_1,R_2	Square Root (SH)	RRE	B245	¤
SQXBR	R_1,R_2	Square Root (EB)	RRE	B316	¤
SQXR	R_1,R_2	Square Root (EH)	RRE	B336	¤
SR	R_1,R_2	Subtract (32)	RR	1B	c
SRA	$R_1,D_2(B_2)$	Shift Right Single (32)	RS-a	8A	c
SRAG	$R_1,R_3,D_2(B_2)$	Shift Right Single (64)	RSY-a	EB0A	c N
SRAK	$R_1,R_3,D_2(B_2)$	Shift Right Single (32)	RSY-a	EBDC	c DO
SRDA	$R_1,D_2(B_2)$	Shift Right Double (64)	RS-a	8E	c
SRDL	$R_1,D_2(B_2)$	Shift Right Double Logical (64)	RS-a	8C	
SRDT	$R_1,R_3,D_2(X_2,B_2)$	Shift Significand Right (LD)	RXF	ED41	¤ TF
SRK	R_1,R_2,R_3	Subtract (32)	RRF-a	B9F9	c DO
SRL	$R_1,D_2(B_2)$	Shift Right Single Logical (32)	RS-a	88	
SRLG	$R_1,R_3,D_2(B_2)$	Shift Right Single Logical (64)	RSY-a	EB0C	N
SRLK	$R_1,R_3,D_2(B_2)$	Shift Right Single Logical (32)	RSY-a	EBDE	DO
SRNM	$D_2(B_2)$	Set BFP Rounding Mode (2 bit)	S	B299	¤
SRNMB	$D_2(B_2)$	Set BFP Rounding Mode (3 bit)	S	B2B8	¤ F
SRNMT	$D_2(B_2)$	Set DFP Rounding Mode	S	B2B9	¤ TR
SRP	$D_1(L_1,B_1),D_2(B_2),I_3$	Shift and Round Decimal	SS-c	F0	¤ c
SRST	R_1,R_2	Search String	RRE	B25E	¤ c
SRSTU	R_1,R_2	Search String Unicode	RRE	B9BE	¤ c E3
SRXT	$R_1,R_3,D_2(X_2,B_2)$	Shift Significand Right (ED)	RXF	ED49	¤ TF
SSAIR	R_1	Set Secondary ASN with Instance	RRE	B99F	¤ RA
SSAR	R_1	Set Secondary ASN	RRE	B225	¤
SSCH	$D_2(B_2)$	Start Subchannel	S	B233	p c
SSKE	$R_1,R_2[,M_3]$	Set Storage Key Extended	RRF-d	B22B	p c
SSM	$D_2(B_2)$	Set System Mask	SI	80	p
ST	$R_1,D_2(X_2,B_2)$	Store (32)	RX-a	50	
STAM	$R_1,R_3,D_2(B_2)$	Store Access Multiple	RS-a	9B	
STAMY	$R_1,R_3,D_2(B_2)$	Store Access Multiple	RSY-a	EB9B	LD
STAP	$D_2(B_2)$	Store CPU Address	S	B212	p
STBEAR	$D_2(B_2)$	Store BEAR	S	B201	p BE
STC	$R_1,D_2(X_2,B_2)$	Store Character	RX-a	42	
STCH	$R_1,D_2(X_2,B_2)$	Store Character High (8)	RXY-a	E3C3	HW
STCK	$D_2(B_2)$	Store Clock	S	B205	¤ c
STCKC	$D_2(B_2)$	Store Clock Comparator	S	B207	p
STCKE	$D_2(B_2)$	Store Clock Extended	S	B278	¤ c
STCKF	$D_2(B_2)$	Store Clock Fast	S	B27C	¤ c SC
STCM	$R_1,M_3,D_2(B_2)$	Store Characters under Mask (low)	RS-b	BE	
STCMH	$R_1,M_3,D_2(B_2)$	Store Characters under Mask (high)	RSY-b	EB2C	¤ N
STCMY	$R_1,M_3,D_2(B_2)$	Store Characters under Mask (low)	RSY-b	EB2D	LD
STCPS	$D_2(B_2)$	Store Channel Path Status	S	B23A	p
STCRW	$D_2(B_2)$	Store Channel Report Word	S	B239	p c
STCTG	$R_1,R_3,D_2(B_2)$	Store Control (64)	RSY-a	EB25	p N
STCTL	$R_1,R_3,D_2(B_2)$	Store Control (32)	RS-a	B6	p
STCY	$R_1,D_2(X_2,B_2)$	Store Character	RXY-a	E372	LD
STD	$R_1,D_2(X_2,B_2)$	Store (L)	RX-a	60	¤
STDY	$R_1,D_2(X_2,B_2)$	Store (L)	RXY-a	ED67	¤ LD
STE	$R_1,D_2(X_2,B_2)$	Store (S)	RX-a	70	¤
STEY	$R_1,D_2(X_2,B_2)$	Store (S)	RXY-a	ED66	¤ LD
STFH	$R_1,D_2(X_2,B_2)$	Store High (32)	RXY-a	E3CB	HW
STFL	$D_2(B_2)$	Store Facility List	S	B2B1	p N3
STFLE	$D_2(B_2)$	Store Facility List Extended	S	B2B0	¤ c FL
STFPC	$D_2(B_2)$	Store FPC	S	B29C	¤
STG	$R_1,D_2(X_2,B_2)$	Store (64)	RXY-a	E324	N
STGRL	R_1,RI_2	Store Relative Long (64)	RIL-b	C4B	GE
STGSC	$R_1,D_2(X_2,B_2)$	Store guarded storage controls	RXY-a	E349	¤ GF
STH	$R_1,D_2(X_2,B_2)$	Store Halfword (16)	RX-a	40	
STHH	$R_1,D_2(X_2,B_2)$	Store Halfword High (16)	RXY-a	E3C7	HW
STHRL	R_1,RI_2	Store Halfword Relative Long (16)	RIL-b	C47	GE
STHY	$R_1,D_2(X_2,B_2)$	Store Halfword (16)	RXY-a	E370	LD

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
STIDP	$D_2(B_2)$	Store CPU ID	S	B202	p
STM	$R_1,R_3,D_2(B_2)$	Store Multiple (32)	RS-a	90	
STMG	$R_1,R_3,D_2(B_2)$	Store Multiple (64)	RSY-a	EB24	N
STMH	$R_1,R_3,D_2(B_2)$	Store Multiple High (32)	RSY-a	EB26	N
STMY	$R_1,R_3,D_2(B_2)$	Store Multiple (32)	RSY-a	EB90	LD
STNSM	$D_1(B_1),I_2$	Store Then And System Mask	SI	AC	p
STOC	$R_1,D_2(B_2),M_3$	Store on Condition (32)	RSY-b	EBF3	L1
STOCFH	$R_1,D_2(B_2),M_3$	Store High on Condition (32)	RSY-b	EBE1	L2
STOCG	$R_1,D_2(B_2),M_3$	Store on Condition (64)	RSY-b	EBE3	L1
STOSM	$D_1(B_1),I_2$	Store Then Or System Mask	SI	AD	p
STPQ	$R_1,D_2(X_2,B_2)$	Store Pair to Quadword (64,64→128)	RXY-a	E38E	¤ N
STPT	$D_2(B_2)$	Store CPU Timer	S	B209	p
STPX	$D_2(B_2)$	Store Prefix	S	B211	p
STRAG	$D_1(B_1),D_2(B_2)$	Store Real Address (64)	SSE	E502	p N
STRL	R_1,RI_2	Store Relative Long (32)	RIL-b	C4F	GE
STRV	$R_1,D_2(X_2,B_2)$	Store Reversed (32)	RXY-a	E33E	N3
STRVG	$R_1,D_2(X_2,B_2)$	Store Reversed (64)	RXY-a	E32F	N
STRVH	$R_1,D_2(X_2,B_2)$	Store Reversed (16)	RXY-a	E33F	N3
STSCH	$D_2(B_2)$	Store Subchannel	S	B234	p c
STSI	$D_2(B_2)$	Store System Information	S	B27D	p c
STURA	R_1,R_2	Store Using Real Address (32)	RRE	B246	p
STURG	R_1,R_2	Store Using Real Address (64)	RRE	B925	p N
STY	$R_1,D_2(X_2,B_2)$	Store (32)	RXY-a	E350	LD
SU	$R_1,D_2(X_2,B_2)$	Subtract Unnormalized (SH)	RX-a	7F	¤ c
SUR	R_1,R_2	Subtract Unnormalized (SH)	RR	3F	¤ c
SVC	I	Supervisor Call	I	0A	¤
SW	$R_1,D_2(X_2,B_2)$	Subtract Unnormalized (LH)	RX-a	6F	¤ c
SWR	R_1,R_2	Subtract Unnormalized (LH)	RR	2F	¤ c
SXBR	R_1,D_2	Subtract (EB)	RRE	B34B	¤ c
SXR	R_1,D_2	Subtract Normalized (EH)	RR	37	¤ c
SXTR	R_1,R_2,R_3	Subtract (ED)	RRF-a	B3DB	¤ c TF
SXTRA	R_1,R_2,R_3,M_4	Subtract (ED)	RRF-a	B3DB	¤ c F
SY	$R_1,D_2(X_2,B_2)$	Subtract (32)	RXY-a	E35B	c LD
TABORT	$D_2(B_2)$	Transaction Abort	S	B2FC	¤ TX
TAM		Test Addressing Mode	E	010B	¤ c N3
TAR	R_1,R_2	Test Access	RRE	B24C	¤ c
TB	R_1,R_2	Test Block	RRE	B22C	i p c
TBDR	R_1,M_3,R_2	Convert HFP to BFP (LB←LH)	RRF-e	B351	¤ c
TBEDR	R_1,M_3,R_2	Convert HFP to BFP (SB←LH)	RRF-e	B350	¤ c
TBEGIN	$D_1(B_1),I_2$	Transaction Begin (nonconstrained)	SIL	E560	¤ c TX
TBEGINC	$D_1(B_1),I_2$	Transaction Begin (constrained)	SIL	E561	¤ c CX
TCDB	$R_1,D_2(X_2,B_2)$	Test Data Class (LB)	RXE	ED11	¤ c
TCEB	$R_1,D_2(X_2,B_2)$	Test Data Class (SB)	RXE	ED10	¤ c
TCXB	$R_1,D_2(X_2,B_2)$	Test Data Class (EB)	RXE	ED12	¤ c
TDCDT	$R_1,D_2(X_2,B_2)$	Test Data Class (LD)	RXE	ED54	¤ TF
TDCET	$R_1,D_2(X_2,B_2)$	Test Data Class (SD)	RXE	ED50	¤ TF
TDCXT	$R_1,D_2(X_2,B_2)$	Test Data Class (ED)	RXE	ED58	¤ TF
TDGDT	$R_1,D_2(X_2,B_2)$	Test Data Group (LD)	RXE	ED55	¤ TF
TDGET	$R_1,D_2(X_2,B_2)$	Test Data Group (SD)	RXE	ED51	¤ TF
TDGXT	$R_1,D_2(X_2,B_2)$	Test Data Group (ED)	RXE	ED59	¤ TF
TEND		Transaction End	S	B2F8	¤ c TX
THDER	R_1,R_2	Convert BFP to HFP (LH←SB)	RRE	B358	¤ c
THDR	R_1,R_2	Convert BFP to HFP (LH←LB)	RRE	B359	¤ c
TM	$D_1(B_1),I_2$	Test under Mask	SI	91	c
TMH	R_1,I_2	Test under Mask High	RI-a	A70	c
TMHH	R_1,I_2	Test under Mask (high high)	RI-a	A72	c N
TMHL	R_1,I_2	Test under Mask (high low)	RI-a	A73	c N
TML	R_1,I_2	Test under Mask Low	RI-a	A71	c
TMLH	R_1,I_2	Test under Mask (low high)	RI-a	A70	c N
TMLL	R_1,I_2	Test under Mask (low low)	RI-a	A71	c N

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
TMY	$D_1(B_1),I_2$	Test under Mask	SIY	EB51	c LD
TP	$D_1(L_1,B_1)$	Test Decimal	RSL	EBC0	¤ c E2
TPEI	R_1,R_2	Test Pending External Interruption	RRE	B9A1	p c TE
TPI	$D_2(B_2)$	Test Pending Interruption	S	B236	p c
TPROT	$D_1(B_1),D_2(B_2)$	Test Protection	SSE	E501	p c
TR	$D_1(L,B_1),D_2(B_2)$	Translate	SS-a	DC	¤
TRACE	$R_1,R_3,D_2(B_2)$	Trace (32)	RS-a	99	p
TRACG	$R_1,R_3,D_2(B_2)$	Trace (64)	RSY-a	EB0F	p N
TRAP2		Trap	E	01FF	¤
TRAP4	$D_2(B_2)$	Trap	S	B2FF	¤
TRE	R_1,R_2	Translate Extended	RRE	B2A5	¤ c
TROO	$R_1,R_2[,M_3]$	Translate One to One	RRF-c	B993	¤ c E2
TROT	$R_1,R_2[,M_3]$	Translate One to Two	RRF-c	B992	¤ c E2
TRT	$D_1(L,B_1),D_2(B_2)$	Translate and Test	SS-a	DD	¤ c
TRTE	$R_1,R_2[,M_3]$	Translate and Test Extended	RRF-c	B9BF	¤ PE
TRTO	$R_1,R_2[,M_3]$	Translate Two to One	RRF-c	B991	¤ c E2
TRTR	$D_1(L,B_1),D_2(B_2)$	Translate and Test Reverse	SS-a	D0	¤ c E3
TRTRE	$R_1,R_2[,M_3]$	Translate and Test Reverse Extended	RRF	B9BD	¤ PE
TRTT	$R_1,R_2[,M_3]$	Translate Two to Two	RRF-c	B990	¤ c E2
TS	$D_2(B_2)$	Test and Set	SI	93	¤ c
TSCH	$D_2(B_2)$	Test Subchannel	S	B235	p c
UNPK	$D_1(L_1,B_1),D_2(L_2,B_2)$	Unpack	SS-b	F3	¤
UNPKA	$D_1(L_1,B_1),D_2(B_2)$	Unpack ASCII	SS-a	EA	¤ c E2
UNPKU	$D_1(L_1,B_1),D_2(B_2)$	Unpack Unicode	SS-a	E2	¤ c E2
UPT		Update Tree	E	0102	i ¤ c
VA	V_1,V_2,V_3,M_4	Vector Add	VRR-c	E7F3	¤ VF
VAC	V_1,V_2,V_3,V_4,M_5	Vector Add With Carry	VRR-d	E7BB	¤ VF
VACC	V_1,V_2,V_3,M_4	Vector Add Compute Carry	VRR-c	E7F1	¤ VF
VACCC	V_1,V_2,V_3,V_4,M_5	Vector Add With Carry Compute Carry	VRR-d	E7B9	¤ VF
VAP	V_1,V_2,V_3,I_4,M_5	Vector Add Decimal	VRI-f	E671	¤ c* VD
VAVG	V_1,V_2,V_3,M_4	Vector Average	VRR-c	E7F2	¤ VF
VAVGL	V_1,V_2,V_3,M_4	Vector Average Logical	VRR-c	E7F0	¤ VF
VBPERM	V_1,V_2,V_3	Vector Bit Permute	VRR-c	E785	¤ V1
VCDG	V_1,V_2,M_3,M_4,M_5	Vector FP Convert from Fixed 64-bit	VRR-a	E7C3	¤ VF
VCDLG	V_1,V_2,M_3,M_4,M_5	Vector FP Convert from Logical 64-bit	VRR-a	E7C1	¤ VF
VCEQ	V_1,V_2,V_3,M_4,M_5	Vector Compare Equal	VRR-b	E7F8	¤ c* VF
VCFN	V_1,V_2,M_3,M_4	Vector FP Convert From NNP	VRR-a	E65D	¤ NN
VCFPL	V_1,V_2,M_3,M_4,M_5	Vector FP Convert from Logical	VRR-a	E7C1	¤ V2
VCFPS	V_1,V_2,M_3,M_4,M_5	Vector FP Convert from Fixed	VRR-a	E7C3	¤ V2
VCGD	V_1,V_2,M_3,M_4,M_5	Vector FP Convert to Fixed 64-bit	VRR-a	E7C2	¤ VF
VCH	V_1,V_2,V_3,M_4,M_5	Vector Compare High	VRR-b	E7FB	¤ c* VF
VCHL	V_1,V_2,V_3,M_4,M_5	Vector Compare High Logical	VRR-b	E7F9	¤ c* VF
VCKSM	V_1,V_2,V_3	Vector Checksum	VRR-c	E766	¤ VF
VCLFNH	V_1,V_2,M_3,M_4	Vector FP Convert and Lengthen from NNP High	VRR-a	E656	¤ NN
VCLFNL	V_1,V_2,M_3,M_4	Vector FP Convert and Lengthen from NNP Low	VRR-a	E65E	¤ NN
VCLFP	V_1,V_2,M_3,M_4,M_5	Vector FP Convert to Logical	VRR-a	E7C0	¤ V2
VCLGD	V_1,V_2,M_3,M_4,M_5	Vector FP Convert to Logical 64-bit	VRR-a	E7C0	¤ VF
VCLZ	V_1,V_2,M_3	Vector Count Leading Zeros	VRR-a	E753	¤ VF
VCLZDP	V_1,V_2,M_3	Vector Count Leading Zero Digits	VRR-k	E651	¤ c* VD2
VCNF	V_1,V_2,M_3,M_4	Vector FP Convert to NNP	VRR-a	E655	¤ NN
VCP	V_1,V_2,M_3	Vector Compare Decimal	VRR-h	E677	¤ c VD
VCRNF	V_1,V_2,V_3,M_4,M_5	Vector FP Convert and Round to NNP	VRR-c	E675	¤ NN
VCSFP	V_1,V_2,M_3,M_4,M_5	Vector FP Convert to Fixed	VRR-a	E7C2	¤ V2
VCSPH	V_1,V_2,V_3,M_4	Vector Convert HFP to Scaled Decimal	VRR-j	E67D	¤ VD2
VCTZ	V_1,V_2,M_3	Vector Count Trailing Zeros	VRR-a	E752	¤ VF
VCVB	R_1,V_2,M_3	Vector Convert to Binary	VRR-i	E650	¤ c* VD
VCVBG	R_1,V_2,M_3	Vector Convert to Binary	VRR-i	E652	¤ c* VD
VCVD	V_1,R_2,I_3,M_4	Vector Convert to Decimal	VRI-i	E658	¤ c* VD

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
VCVDG	V_1,R_2,I_3,M_4	Vector Convert to Decimal	VRI-i	E65A	¤ c* VD
VDP	V_1,V_2,V_3,I_4,M_5	Vector Divide Decimal	VRI-f	E67A	¤ c* VD
VEC	V_1,V_2,M_3	Vector Element Compare	VRR-a	E7DB	¤ c VF
VECL	V_1,V_2,M_3	Vector Element Compare Logical	VRR-a	E7D9	¤ c VF
VERIM	V_1,V_2,V_3,I_4,M_5	Vector Element Rotate and Insert Under Mask	VRI-d	E772	¤ VF
VERLL	$V_1,V_3,D_2(B_2),M_4$	Vector Element Rotate Left Logical	VRS-a	E733	¤ VF
VERLLV	V_1,V_2,V_3,M_4	Vector Element Rotate Left Logical	VRR-c	E773	¤ VF
VESL	$V_1,V_3,D_2(B_2),M_4$	Vector Element Shift Left	VRS-a	E730	¤ VF
VESLV	V_1,V_2,V_3,M_4	Vector Element Shift Left	VRR-c	E770	¤ VF
VESRA	$V_1,V_3,D_2(B_2),M_4$	Vector Element Shift Right Arithmetic	VRS-a	E73A	¤ VF
VESRAV	V_1,V_2,V_3,M_4	Vector Element Shift Right Arithmetic	VRR-c	E77A	¤ VF
VESRL	$V_1,V_3,D_2(B_2),M_4$	Vector Element Shift Right Logical	VRS-a	E738	¤ VF
VESRLV	V_1,V_2,V_3,M_4	Vector Element Shift Right Logical	VRR-c	E778	¤ VF
VFA	V_1,V_2,V_3,M_4,M_5	Vector FP Add	VRR-c	E7E3	¤ VF
VFAE	$V_1,V_2,V_3,M_4[,M_5]$	Vector Find Any Element Equal	VRR-b	E782	¤ c* VF
VFCE	V_1,V_2,V_3,M_4,M_5,M_6	Vector FP Compare Equal	VRR-c	E7E8	¤ c* VF
VFCH	V_1,V_2,V_3,M_4,M_5,M_6	Vector FP Compare High	VRR-c	E7EB	¤ c* VF
VFCHE	V_1,V_2,V_3,M_4,M_5,M_6	Vector FP Compare High or Equal	VRR-c	E7EA	¤ c* VF
VFD	V_1,V_2,V_3,M_4,M_5	Vector FP Divide	VRR-c	E7E5	¤ VF
VFEE	$V_1,V_2,V_3,M_4[,M_5]$	Vector Find Element Equal	VRR-b	E780	¤ c* VF
VFENE	$V_1,V_2,V_3,M_4[,M_5]$	Vector Find Element Not Equal	VRR-b	E781	¤ c* VF
VFI	V_1,V_2,M_3,M_4,M_5	Vector Load FP Integer	VRR-a	E7C7	¤ VF
VFLL	V_1,V_2,M_3,M_4	Vector FP Load Lengthened	VRR-a	E7C4	¤ VF
VFLR	V_1,V_2,M_3,M_4,M_5	Vector FP Load Rounded	VRR-a	E7C5	¤ VF
VFM	V_1,V_2,V_3,M_4,M_5	Vector FP Multiply	VRR-c	E7E7	¤ VF
VFMA	V_1,V_2,V_3,V_4,M_5,M_6	Vector FP Multiply and Add	VRR-e	E78F	¤ VF
VFMAX	V_1,V_2,V_3,M_4,M_5,M_6	Vector FP Maximum	VRR-c	E7EF	¤ V1
VFMIN	V_1,V_2,V_3,M_4,M_5,M_6	Vector FP Minimum	VRR-c	E7EE	¤ V1
VFMS	V_1,V_2,V_3,V_4,M_5,M_6	Vector FP Multiply and Subtract	VRR-e	E78E	¤ VF
VFNMA	V_1,V_2,V_3,V_4,M_5,M_6	Vector FP Negative Multiply and Add	VRR-e	E79F	¤ V1
VFNMS	V_1,V_2,V_3,V_4,M_5,M_6	Vector FP Negative Multiply and Subtract	VRR-e	E79E	¤ V1
VFPSO	V_1,V_2,M_3,M_4,M_5	Vector FP Perform Sign Operation	VRR-a	E7CC	¤ VF
VFS	V_1,V_2,V_3,M_4,M_5	Vector FP Subtract	VRR-c	E7E2	¤ VF
VFSQ	V_1,V_2,M_3,M_4	Vector FP Square Root	VRR-a	E7CE	¤ VF
VFTCI	V_1,V_2,I_3,M_4,M_5	Vector FP Test Data Class Immediate	VRI-e	E74A	¤ VF
VGBM	V_1,I_2	Vector Generate Byte Mask	VRI-a	E744	¤ VF
VGEF	$V_1,D_2(V_2,B_2),M_3$	Vector Gather Element (32)	VRV	E713	¤ VF
VGEG	$V_1,D_2(V_2,B_2),M_3$	Vector Gather Element (64)	VRV	E712	¤ VF
VGFM	V_1,V_2,V_3,M_4	Vector Galois Field Multiply Sum	VRR-c	E7B4	¤ VF
VGFMA	V_1,V_2,V_3,V_4,M_5	Vector Galois Field Multiply Sum and Accumulate	VRR-d	E7BC	¤ VF
VGM	V_1,I_2,I_3,V_4	Vector Generate Mask	VRI-b	E746	¤ VF
VISTR	$V_1,V_2,M_3[,M_5]$	Vector Isolate String	VRR-a	E75C	¤ c* VF
VL	$V_1,D_2(X_2,B_2)$	Vector Load	VRX	E706	¤ VF
VLBB	$V_1,D_2(X_2,B_2),M_3$	Vector Load to Block Boundary	VRX	E707	¤ VF
VLBR	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Elements	VRX	E606	¤ V2
VLBRREP	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Element and Replicate	VRX	E605	¤ V2
VLC	V_1,V_2,M_3	Vector Load Complement	VRR-a	E7DE	¤ VF
VLEB	$V_1,D_2(X_2,B_2),M_3$	Vector Load Element (8)	VRX	E700	¤ VF
VLEF	$V_1,D_2(X_2,B_2),M_3$	Vector Load Element (32)	VRX	E703	¤ VF
VLEG	$V_1,D_2(X_2,B_2),M_3$	Vector Load Element (64)	VRX	E702	¤ VF
VLEH	$V_1,D_2(X_2,B_2),M_3$	Vector Load Element (16)	VRX	E701	¤ VF
VLEIB	V_1,I_2,M_3	Vector Load Element Immediate (8)	VRI-a	E740	¤ VF
VLEIF	V_1,I_2,M_3	Vector Load Element Immediate (32)	VRI-a	E743	¤ VF
VLEIG	V_1,I_2,M_3	Vector Load Element Immediate (64)	VRI-a	E742	¤ VF
VLEIH	V_1,I_2,M_3	Vector Load Element Immediate (16)	VRI-a	E741	¤ VF
VLER	$V_1,D_2(X_2,B_2),M_3$	Vector Load Elements Reversed	VRX	E607	¤ V2
VLGV	$R_1,V_3,D_2(B_2),M_4$	Vector Load GR from VR Element	VRS-c	E721	¤ VF
VLIP	V_1,I_2,I_3	Vector Load Immediate Decimal	VRI-h	E649	¤ VD

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
VLL	$V_1,R_3,D_2(B_2)$	Vector Load With Length	VRS-b	E737	¤ VF
VLEBRF	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Element (32)	VRX	E603	¤ V2
VLEBRG	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Element (64)	VRX	E602	¤ V2
VLEBRH	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Element (16)	VRX	E601	¤ V2
VLLEBRZ	$V_1,D_2(X_2,B_2),M_3$	Vector Load Byte Reversed Element and Zero	VRX	E604	¤ V2
VLLEZ	$V_1,D_2(X_2,B_2),M_3$	Vector Load Logical Element and Zero	VRX	E704	¤ VF
VLM	$V_1,V_3,D_2(B_2)[,M_4]$	Vector Load Multiple	VRS-a	E736	¤ VF
VLP	V_1,V_2,M_3	Vector Load Positive	VRR-a	E7DF	¤ VF
VLR	V_1,V_2	Vector Load	VRR-a	E756	¤ VF
VLREP	$V_1,D_2(X_2,B_2),M_3$	Vector Load and Replicate	VRX	E705	¤ VF
VLRL	$V_1,D_2(B_2),I_3$	Vector Load Rightmost with Length	VSI	E635	¤ VD
VLRLR	$V_1,R_3,D_2(B_2)$	Vector Load Rightmost with Length	VRS-d	E637	¤ VD
VLVG	$V_1,R_3,D_2(B_2),M_4$	Vector Load VR Element from GR	VRS-b	E722	¤ VF
VLVGP	V_1,R_2,R_3	Vector Load VR from GRs Disjoint	VRR-f	E762	¤ VF
VMAE	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Even	VRR-d	E7AE	¤ VF
VMAH	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add High	VRR-d	E7AB	¤ VF
VMAL	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Low	VRR-d	E7AA	¤ VF
VMALE	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Logical Even	VRR-d	E7AC	¤ VF
VMALH	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Logical High	VRR-d	E7A9	¤ VF
VMALO	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Logical Odd	VRR-d	E7AD	¤ VF
VMAO	V_1,V_2,V_3,V_4,M_5	Vector Multiply and Add Odd	VRR-d	E7AF	¤ VF
VME	V_1,V_2,V_3,M_4	Vector Multiply Even	VRR-c	E7A6	¤ VF
VMH	V_1,V_2,V_3,M_4	Vector Multiply High	VRR-c	E7A3	¤ VF
VML	V_1,V_2,V_3,M_4	Vector Multiply Low	VRR-c	E7A2	¤ VF
VMLE	V_1,V_2,V_3,M_4	Vector Multiply Logical Even	VRR-c	E7A4	¤ VF
VMLH	V_1,V_2,V_3,M_4	Vector Multiply Logical High	VRR-c	E7A1	¤ VF
VMLO	V_1,V_2,V_3,M_4	Vector Multiply Logical Odd	VRR-c	E7A5	¤ VF
VMN	V_1,V_2,V_3,M_4	Vector Minimum	VRR-c	E7FE	¤ VF
VMNL	V_1,V_2,V_3,M_4	Vector Minimum Logical	VRR-c	E7FC	¤ VF
VMO	V_1,V_2,V_3,M_4	Vector Multiply Odd	VRR-c	E7A7	¤ VF
VMP	V_1,V_2,V_3,I_4,M_5	Vector Multiply Decimal	VRI-f	E678	¤ c* VD
VMRH	V_1,V_2,V_3,M_4	Vector Merge High	VRR-c	E761	¤ VF
VMRL	V_1,V_2,V_3,M_4	Vector Merge Low	VRR-c	E760	¤ VF
VMSL	V_1,V_2,V_3,V_4,M_5,M_6	Vector Multiply Sum Logical	VRR-d	E7B8	¤ V1
VMSP	V_1,V_2,V_3,I_4,M_5	Vector Multiply and Shift Decimal	VRI-f	E679	¤ c* VD
VMX	V_1,V_2,V_3,M_4	Vector Maximum	VRR-c	E7FF	¤ VF
VMXL	V_1,V_2,V_3,M_4	Vector Maximum Logical	VRR-c	E7FD	¤ VF
VN	V_1,V_2,V_3	Vector AND	VRR-c	E768	¤ VF
VNC	V_1,V_2,V_3	Vector AND with Complement	VRR-c	E769	¤ VF
VNN	V_1,V_2,V_3	Vector NAND	VRR-c	E76E	¤ V1
VNO	V_1,V_2,V_3	Vector NOR	VRR-c	E76B	¤ VF
VNX	V_1,V_2,V_3	Vector Not Exclusive OR	VRR-c	E76C	¤ V1
VO	V_1,V_2,V_3	Vector OR	VRR-c	E76A	¤ VF
VOC	V_1,V_2,V_3	Vector OR with Complement	VRR-c	E76F	¤ V1
VPDI	V_1,V_2,V_3,M_4	Vector Permute Doubleword Immediate	VRR-c	E784	¤ VF
VPERM	V_1,V_2,V_3,V_4	Vector Permute	VRR-e	E78C	¤ VF
VPK	V_1,V_2,V_3,M_4	Vector Pack	VRR-c	E794	¤ VF
VPKLS	V_1,V_2,V_3,M_4,M_5	Vector Pack Logical Saturate	VRR-b	E795	¤ c* VF
VPKS	V_1,V_2,V_3,M_4,M_5	Vector Pack Saturate	VRR-b	E797	¤ c* VF
VPKZ	$V_1,D_2(B_2),I_3$	Vector Pack Zoned	VSI	E634	¤ VD
VPKZR	V_1,V_2,V_3,I_4,M_5	Vector Pack Zoned Register	VRI-f	E670	¤ c* VD2
VPOPCT	V_1,V_2,M_3	Vector Population Count	VRR-a	E750	¤ VF
VPSOP	V_1,V_2,I_3,I_4,M_5	Vector Perform Sign Operation Decimal	VRI-g	E65B	¤ c* VD
VREP	V_1,V_3,I_2,M_4	Vector Replicate	VRI-c	E74D	¤ VF
VREPI	V_1,I_2,M_3	Vector Replicate Immediate	VRI-a	E745	¤ VF
VRP	V_1,V_2,V_3,I_4,M_5	Vector Remainder Decimal	VRI-f	E67B	¤ c* VD
VS	V_1,V_2,V_3,M_4	Vector Subtract	VRR-c	E7F7	¤ VF
VSBCBI	V_1,V_2,V_3,V_4,M_5	Vector Subtract With Borrow Compute Borrow Indication	VRR-d	E7BD	¤ VF

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
VSBI	V_1,V_2,V_3,V_4,M_5	Vector Subtract With Borrow Indication	VRR-d	E7BF	¤ VF
VSCBI	V_1,V_2,V_3,M_4	Vector Subtract Compute Borrow Indication	VRR-c	E7F5	¤ VF
VSCEF	$V_1,D_2(V_2,B_2),M_3$	Vector Scatter Element (32)	VRV	E71B	¤ VF
VSCEG	$V_1,D_2(V_2,B_2),M_3$	Vector Scatter Element (64)	VRV	E71A	¤ VF
VSCHP	V_1,V_2,V_3,M_4,M_5	Decimal Scale and Convert to HFP	VRR-b	E674	¤ VD2
VSCSHP	V_1,V_2,V_3	Decimal Scale and Convert and Split to HFP	VRR-b	E67C	¤ VD2
VSDP	V_1,V_2,V_3,I_4,M_5	Vector Shift and Divide Decimal	VRI-f	E67E	¤ c* VD
VSEG	V_1,V_2,M_3	Vector Sign Extend to Doubleword	VRR-a	E75F	¤ VF
VSEL	V_1,V_2,V_3,V_4	Vector Select	VRR-e	E78D	¤ VF
VSL	V_1,V_2,V_3	Vector Shift Left	VRR-c	E774	¤ VF
VSLB	V_1,V_2,V_3	Vector Shift Left By Byte	VRR-c	E775	¤ VF
VSLD	V_1,V_2,V_3,I_4	Vector Shift Left Double by Bit	VRI-d	E786	¤ V2
VSLDB	V_1,V_2,V_3,I_4	Vector Shift Left Double By Byte	VRI-d	E777	¤ VF
VSP	V_1,V_2,V_3,I_4,M_5	Vector Subtract Decimal	VRI-f	E673	¤ c* VD
VSRA	V_1,V_2,V_3	Vector Shift Right Arithmetic	VRR-c	E77E	¤ VF
VSRAB	V_1,V_2,V_3	Vector Shift Right Arithmetic By Byte	VRR-c	E77F	¤ VF
VSRD	V_1,V_2,V_3,I_4	Vector Shift Right Double by Bit	VRI-d	E787	¤ V2
VSRL	V_1,V_2,V_3	Vector Shift Right Logical	VRR-c	E77C	¤ VF
VSRLB	V_1,V_2,V_3	Vector Shift Right Logical By Byte	VRR-c	E77D	¤ VF
VSRP	V_1,V_2,I_3,I_4,M_5	Vector Shift and Round Decimal	VRI-g	E659	¤ c* VD
VSRPR	V_1,V_2,V_3,I_4,M_5	Vector Shift and Round Decimal Register	VRI-f	E672	¤ c* VD2
VST	$V_1,D_2(X_2,B_2)[,M_3]$	Vector Store	VRX	E70E	¤ VF
VSTBR	$V_1,D_2(X_2,B_2),M_3$	Vector Store Byte Reversed Elements	VRX	E60E	¤ V2
VSTEB	$V_1,D_2(X_2,B_2),M_3$	Vector Store Element (8)	VRX	E708	¤ VF
VSTEBRF	$V_1,D_2(X_2,B_2),M_3$	Vector Store Byte Reversed Element (32)	VRX	E60B	¤ V2
VSTEBRG	$V_1,D_2(X_2,B_2),M_3$	Vector Store Byte Reversed Element (64)	VRX	E60A	¤ V2
VSTEBRH	$V_1,D_2(X_2,B_2),M_3$	Vector Store Byte Reversed Element (16)	VRX	E609	¤ V2
VSTEF	$V_1,D_2(X_2,B_2),M_3$	Vector Store Element (32)	VRX	E70B	¤ VF
VSTEG	$V_1,D_2(X_2,B_2),M_3$	Vector Store Element (64)	VRX	E70A	¤ VF
VSTEH	$V_1,D_2(X_2,B_2),M_3$	Vector Store Element (16)	VRX	E709	¤ VF
VSTER	$V_1,D_2(X_2,B_2),M_3$	Vector Store Elements Reversed	VRX	E60F	¤ V2
VSTL	$V_1,R_3,D_2(B_2)$	Vector Store With Length	VRS-b	E73F	¤ VF
VSTM	$V_1,V_3,D_2(B_2)[,M_4]$	Vector Store Multiple	VRS-a	E73E	¤ VF
VSTRC	$V_1,V_2,V_3,V_4,M_5[,M_6]$	Vector String Range Compare	VRR-d	E78A	¤ c* VF
VSTRL	$V_1,D_2(B_2),I_3$	Vector Store Rightmost with Length	VSI	E63D	¤ VD
VSTRLR	$V_1,R_3,D_2(B_2)$	Vector Store Rightmost with Length	VRS-d	E63F	¤ VD
VSTRS	$V_1,V_2,V_3,V_4,M_5[,M_6]$	Vector String Search	VRR-d	E78B	¤ c V2
VSUM	V_1,V_2,V_3,M_4	Vector Sum Across Word	VRR-c	E764	¤ VF
VSUMG	V_1,V_2,V_3,M_4	Vector Sum Across Doubleword	VRR-c	E765	¤ VF
VSUMQ	V_1,V_2,V_3,M_4	Vector Sum Across Quadword	VRR-c	E767	¤ VF
VTM	V_1,V_2	Vector Test Under Mask	VRR-a	E7D8	¤ VF
VTP	V_1	Vector Test Decimal	VRR-g	E65F	¤ c* VD
VUPH	V_1,V_2,M_3	Vector Unpack High	VRR-a	E7D7	¤ VF
VUPKZ	$V_1,D_2(B_2),I_3$	Vector Unpack Zoned	VSI	E63C	¤ VD
VUPKZH	V_1,V_2,M_3	Vector Unpack Zoned High	VRR-k	E654	¤ VD2
VUPKZL	V_1,V_2,M_3	Vector Unpack Zoned Low	VRR-k	E65C	¤ VD2
VUPL	V_1,V_2,M_3	Vector Unpack Low	VRR-a	E7D6	¤ VF
VUPLH	V_1,V_2,M_3	Vector Unpack Logical High	VRR-a	E7D5	¤ VF
VUPLL	V_1,V_2,M_3	Vector Unpack Logical Low	VRR-a	E7D4	¤ VF
VX	V_1,V_2,V_3	Vector Exclusive OR	VRR-c	E76D	¤ VF
WFC	V_1,V_2,M_3,M_4	Vector FP Compare Scalar	VRR-a	E7CB	¤ VF
WFK	V_1,V_2,M_3,M_4	Vector FP Compare and Signal Scalar	VRR-a	E7CA	¤ VF
X	$R_1,D_2(X_2,B_2)$	Exclusive OR (32)	RX-a	57	c
XC	$D_1(L,B_1),D_2(B_2)$	Exclusive OR (character)	SS-a	D7	¤ c
XG	$R_1,D_2(X_2,B_2)$	Exclusive OR (64)	RXY-a	E382	c N
XGR	R_1,R_2	Exclusive OR (64)	RRE	B982	c N

Mnemonic	Operands	Name	Format	Opcode	Class & Notes
XGRK	R_1,R_2,R_3	Exclusive OR (64)	RRF-a	B9E7	c DO
XI	$D_1(B_1),I_2$	Exclusive OR Immediate	SI	97	c
XIHF	R_1,I_2	Exclusive OR Immediate (high)	RIL-a	C06	c EI
XILF	R_1,I_2	Exclusive OR Immediate (low)	RIL-a	C07	c EI
XIY	$D_1(B_1),I_2$	Exclusive OR Immediate	SIY	EB57	c LD
XR	R_1,R_2	Exclusive OR (32)	RR	17	c
XRK	R_1,R_2,R_3	Exclusive OR (32)	RRF-a	B9F7	c DO
XSCH		Cancel Subchannel	S	B276	p c
XY	$R_1,D_2(X_2,B_2)$	Exclusive Or (32)	RXY-a	E357	c LD
ZAP	$D_1(L_1,B_1),D_2(L_2,B_2)$	Zero and Add	SS-b	F8	¤ c

Floating-Point Operand Lengths and Types:

E	Extended (binary, decimal or hex)	LB	Long binary
EB	Extended binary	LD	Long decimal
ED	Extended decimal	LH	Long hex
EH	Extended hex	S	Short (binary, decimal or hex)
EH_L	Extended hex (low-order part)	SB	Short binary
EH_H	Extended hex (high-order part)	SD	Short decimal
L	Long (binary, decimal or hex)	SH	Short hex

Notes:

- & Combination of fields
- ¤ One or more restrictions apply in the transactional-execution mode
- c Condition code set
- c* Condition code may be set based on control in the instruction
- i Interruptible instruction
- n New condition code loaded
- p Privileged instruction; restricted in the transactional-execution mode
- q Semiprivileged instruction; restricted in the transactional-execution mode
- u Condition code is unpredictable
- BE BEAR-enhancement facility
- CS Compare-and-swap-and-store facility
- CT Configuration topology facility
- CX Constrained-transactional-execution facility
- D2 DAT-enhancement facility 2
- DE DAT-enhancement facility
- DO Distinct-operands facility
- DP Reset-DAT-protection facility
- E2 Extended-translation facility 2
- E3 Extended-translation facility 3
- ED1 Enhanced-DAT facility 1
- ED2 Enhanced-DAT facility 2
- EH Execution-hint facility
- EI Extended-immediate facility
- ES Expanded-storage facility
- ET Extract-CPU-time facility
- F Floating-point-extension facility
- FG FPR-GPR-transfer facility
- FL Store-facility-list-extended facility
- FS Floating-point-support-sign-handling facility
- GE General-instructions-extension facility
- GF Guarded-storage facility
- GZ DEFLATE-conversion facility
- HM HFP multiply-and-add/subtract facility
- HW High-word facility
- IA Interlocked-access facility
- IM Insert-reference-bits-multiple facility
- L1 Load/store-on-condition facility 1
- L2 Load/store-on-condition facility 2
- LD Long-displacement facility
- LT Load-and-trap facility
- LZ Load-and-zero-rightmost-byte facility
- M3 Message-security assist extension 3
- M4 Message-security assist extension 4
- M5 Message-security assist extension 5
- M8 Message-security assist extension 8
- M9 Message-security assist extension 9
- MI1 Miscellaneous-instructions facility 1
- MI2 Miscellaneous-instructions facility 2
- MI3 Miscellaneous-instructions facility 3
- MO Move-with-optional-specifications facility
- MS Message-security assist
- N New in z/Architecture
- NN Neural-network-processing-assist facility
- N3 New in z/Architecture and added to ESA/390
- PA Processor-assist facility
- PC DFP-packed-conversion facility
- PE Parsing-enhancement facility
- PF PFPO facility
- PI Processor-activity-instrumentation facility
- PK Population-count facility
- RA ASN-and-LX-reuse facility
- RB Reset-reference-bits multiple facility
- SC Store-clock-fast facility
- SL Enhanced-sort facility
- TE Test-pending-external-interruption facility
- TF Decimal-floating-point facility
- TR Decimal-floating-point-rounding facility
- TS TOD-clock-steering facility
- TX Transactional-execution facility
- UE HFP unnormalized-extension facility
- VD Vector-packed-decimal facility
- VD2 Vector-packed-decimal-enhancement facility 2
- VF Vector facility for z/Architecture
- V1 Vector-enhancements facility 1
- V2 Vector-enhancements facility 2
- XF IEEE-exception-support facility
- XX Execute-extension facility
- ZF DFP zoned-conversion facility

Machine Instructions by Operation Code

OpCode	Mnemonic
0101	PR
0102	UPT
0104	PTFF
0107	SCKPF
010A	PFPO
010B	TAM
010C	SAM24
010D	SAM31
010E	SAM64
01FF	TRAP2
04	SPM
05	BALR
06	BCTR
07	BCR
0A	SVC
0B	BSM
0C	BASSM
0D	BASR
0E	MVCL
0F	CLCL
10	LPR
11	LNR
12	LTR
13	LCR
14	NR
15	CLR
16	OR
17	XR
18	LR
19	CR
1A	AR
1B	SR
1C	MR
1D	DR
1E	ALR
1F	SLR
20	LPDR
21	LNDR
22	LTDR
23	LCDR
24	HDR
25	LDXR
25	LRDR
26	MXR
27	MXDR
28	LDR
29	CDR
2A	ADR
2B	SDR
2C	MDR
2D	DDR
2E	AWR
2F	SWR
30	LPER
31	LNER
32	LTER
33	LCER
34	HER
35	LEDR
35	LRER
36	AXR
37	SXR
38	LER
39	CER
3A	AER
3B	SER
3C	MDER
3C	MER
3D	DER
3E	AUR
3F	SUR
40	STH

OpCode	Mnemonic
41	LA
42	STC
43	IC
44	EX
45	BAL
46	BCT
47	BC
48	LH
49	CH
4A	AH
4B	SH
4C	MH
4D	BAS
4E	CVD
4F	CVB
50	ST
51	LAE
54	N
55	CL
56	O
57	X
58	L
59	C
5A	A
5B	S
5C	M
5D	D
5E	AL
5F	SL
60	STD
67	MXD
68	LD
69	CD
6A	AD
6B	SD
6C	MD
6D	DD
6E	AW
6F	SW
70	STE
71	MS
78	LE
79	CE
7A	AE
7B	SE
7C	MDE
7C	ME
7D	DE
7E	AU
7F	SU
80	SSM
82	LPSW
83	Diagnose
84	BRXH
85	BRXLE
86	BXH
87	BXLE
88	SRL
89	SLL
8A	SRA
8B	SLA
8C	SRDL
8D	SLDL
8E	SRDA
8F	SLDA
90	STM
91	TM
92	MVI
93	TS
94	NI
95	CLI
96	OI

OpCode	Mnemonic
97	XI
98	LM
99	TRACE
9A	LAM
9B	STAM
A50	IIHH
A51	IIHL
A52	IILH
A53	IILL
A54	NIHH
A55	NIHL
A56	NILH
A57	NILL
A58	OIHH
A59	OIHL
A5A	OILH
A5B	OILL
A5C	LLIHH
A5D	LLIHL
A5E	LLILH
A5F	LLILL
A70	TMH
A70	TMLH
A71	TML
A71	TMLL
A72	TMHH
A73	TMHL
A74	BRC
A75	BRAS
A76	BRCT
A77	BRCTG
A78	LHI
A79	LGHI
A7A	AHI
A7B	AGHI
A7C	MHI
A7D	MGHI
A7E	CHI
A7F	CGHI
A8	MVCLE
A9	CLCLE
AC	STNSM
AD	STOSM
AE	SIGP
AF	MC
B1	LRA
B200	LBEAR
B201	STBEAR
B202	STIDP
B204	SCK
B205	STCK
B206	SCKC
B207	STCKC
B208	SPT
B209	STPT
B20A	SPKA
B20B	IPK
B20D	PTLB
B210	SPX
B211	STPX
B212	STAP
B214	SIE
B218	PC
B219	SAC
B21A	CFC
B221	IPTE
B222	IPM
B223	IVSK
B224	IAC
B225	SSAR
B226	EPAR
B227	ESAR

OpCode	Mnemonic
B228	PT
B229	ISKE
B22A	RRBE
B22B	SSKE
B22C	TB
B22D	DXR
B22E	PGIN
B22F	PGOUT
B230	CSCH
B231	HSCH
B232	MSCH
B233	SSCH
B234	STSCH
B235	TSCH
B236	TPI
B237	SAL
B238	RSCH
B239	STCRW
B23A	STCPS
B23B	RCHP
B23C	SCHM
B240	BAKR
B241	CKSM
B244	SQDR
B245	SQER
B246	STURA
B247	MSTA
B248	PALB
B249	EREG
B24A	ESTA
B24B	LURA
B24C	TAR
B24D	CPYA
B24E	SAR
B24F	EAR
B250	CSP
B252	MSR
B254	MVPG
B255	MVST
B257	CUSE
B258	BSG
B25A	BSA
B25D	CLST
B25E	SRST
B263	CMPSC
B276	XSCH
B277	RP
B278	STCKE
B279	SACF
B27C	STCKF
B27D	STSI
B28F	QPACI
B299	SRNM
B29C	STFPC
B29D	LFPC
B2A5	TRE
B2A6	CU21
B2A6	CUUTF
B2A7	CU12
B2A7	CUTFU
B2B0	STFLE
B2B1	STFL
B2B2	LPSWE
B2B8	SRNMB
B2B9	SRNMT
B2BD	LFAS
B2E8	PPA
B2EC	ETND
B2F8	TEND
B2FA	NIAI
B2FC	TABORT
B2FF	TRAP4
B300	LPEBR
B301	LNEBR
B302	LTEBR
B303	LCEBR

OpCode	Mnemonic
B304	LDEBR
B305	LXDBR
B306	LXEBR
B307	MXDBR
B308	KEBR
B309	CEBR
B30A	AEBR
B30B	SEBR
B30C	MDEBR
B30D	DEBR
B30E	MAEBR
B30F	MSEBR
B310	LPDBR
B311	LNDBR
B312	LTDBR
B313	LCDBR
B314	SQEBR
B315	SQDBR
B316	SQXBR
B317	MEEBR
B318	KDBR
B319	CDBR
B31A	ADBR
B31B	SDBR
B31C	MDBR
B31D	DDBR
B31E	MADBR
B31F	MSDBR
B324	LDER
B325	LXDR
B326	LXER
B32E	MAER
B32F	MSER
B336	SQXR
B337	MEER
B338	MAYLR
B339	MYLR
B33A	MAYR
B33B	MYR
B33C	MAYHR
B33D	MYHR
B33E	MADR
B33F	MSDR
B340	LPXBR
B341	LNXBR
B342	LTXBR
B343	LCXBR
B344	LEDBR
B344	LEDBRA
B345	LDXBR
B345	LDXBRA
B346	LEXBR
B346	LEXBRA
B347	FIXBR
B347	FIXBRA
B348	KXBR
B349	CXBR
B34A	AXBR
B34B	SXBR
B34C	MXBR
B34D	DXBR
B350	TBEDR
B351	TBDR
B353	DIEBR
B357	FIEBR
B357	FIEBRA
B358	THDER
B359	THDR
B35B	DIDBR
B35F	FIDBR
B35F	FIDBRA
B360	LPXR
B361	LNXR
B362	LTXR
B363	LCXR
B365	LXR

OpCode	Mnemonic
B366	LEXR
B367	FIXR
B369	CXR
B370	LPDFR
B371	LNDFR
B372	CPSDR
B373	LCDFR
B374	LZER
B375	LZDR
B376	LZXR
B377	FIER
B37F	FIDR
B384	SFPC
B385	SFASR
B38C	EFPC
B390	CELFBR
B391	CDLFBR
B392	CXLFBR
B394	CEFBR
B394	CEFBRA
B395	CDFBR
B395	CDFBRA
B396	CXFBR
B396	CXFBRA
B398	CFEBR
B398	CFEBRA
B399	CFDBR
B399	CFDBRA
B39A	CFXBR
B39A	CFXBRA
B39C	CLFEBR
B39D	CLFDBR
B39E	CLFXBR
B3A0	CELGBR
B3A1	CDLGBR
B3A2	CXLGBR
B3A4	CEGBR
B3A4	CEGBRA
B3A5	CDGBR
B3A5	CDGBRA
B3A6	CXGBR
B3A6	CXGBRA
B3A8	CGEBR
B3A8	CGEBRA
B3A9	CGDBR
B3A9	CGDBRA
B3AA	CGXBR
B3AA	CGXBRA
B3AC	CLGEBR
B3AD	CLGDBR
B3AE	CLGXBR
B3B4	CEFR
B3B5	CDFR
B3B6	CXFR
B3B8	CFER
B3B9	CFDR
B3BA	CFXR
B3C1	LDGR
B3C4	CEGR
B3C5	CDGR
B3C6	CXGR
B3C8	CGER
B3C9	CGDR
B3CA	CGXR
B3CD	LGDR
B3D0	MDTR
B3D0	MDTRA
B3D1	DDTR
B3D1	DDTRA
B3D2	ADTR
B3D2	ADTRA
B3D3	SDTR
B3D3	SDTRA
B3D4	LDETR
B3D5	LEDTR
B3D6	LTDTR

OpCode	Mnemonic
B3D7	FIDTR
B3D8	MXTR
B3D8	MXTRA
B3D9	DXTR
B3D9	DXTRA
B3DA	AXTR
B3DA	AXTRA
B3DB	SXTR
B3DB	SXTRA
B3DC	LXDTR
B3DD	LDXTR
B3DE	LTXTR
B3DF	FIXTR
B3E0	KDTR
B3E1	CGDTR
B3E1	CGDTRA
B3E2	CUDTR
B3E3	CSDTR
B3E4	CDTR
B3E5	EEDTR
B3E7	ESDTR
B3E8	KXTR
B3E9	CGXTR
B3E9	CGXTRA
B3EA	CUXTR
B3EB	CSXTR
B3EC	CXTR
B3ED	EEXTR
B3EF	ESXTR
B3F1	CDGTR
B3F1	CDGTRA
B3F2	CDUTR
B3F3	CDSTR
B3F4	CEDTR
B3F5	QADTR
B3F6	IEDTR
B3F7	RRDTR
B3F9	CXGTR
B3F9	CXGTRA
B3FA	CXUTR
B3FB	CXSTR
B3FC	CEXTR
B3FD	QAXTR
B3FE	IEXTR
B3FF	RRXTR
B6	STCTL
B7	LCTL
B900	LPGR
B901	LNGR
B902	LTGR
B903	LCGR
B904	LGR
B905	LURAG
B906	LGBR
B907	LGHR
B908	AGR
B909	SGR
B90A	ALGR
B90B	SLGR
B90C	MSGR
B90D	DSGR
B90E	EREGG
B90F	LRVGR
B910	LPGFR
B911	LNGFR
B912	LTGFR
B913	LCGFR
B914	LGFR
B916	LLGFR
B917	LLGTR
B918	AGFR
B919	SGFR
B91A	ALGFR
B91B	SLGFR
B91C	MSGFR
B91D	DSGFR

OpCode	Mnemonic
B91E	KMAC
B91F	LRVR
B920	CGR
B921	CLGR
B925	STURG
B926	LBR
B927	LHR
B928	PCKMO
B929	KMA
B92A	KMF
B92B	KMO
B92C	PCC
B92D	KMCTR
B92E	KM
B92F	KMC
B930	CGFR
B931	CLGFR
B938	SORTL
B939	DFLTCC
B93A	KDSA
B93B	NNPA
B93C	PRNO
B93E	KIMD
B93F	KLMD
B941	CFDTR
B942	CLGDTR
B943	CLFDTR
B946	BCTGR
B949	CFXTR
B94A	CLGXTR
B94B	CLFXTR
B951	CDFTR
B952	CDLGTR
B953	CDLFTR
B959	CXFTR
B95A	CXLGTR
B95B	CXLFTR
B960	CGRT
B961	CLGRT
B964	NNGRK
B965	OCGRK
B966	NOGRK
B967	NXGRK
B972	CRT
B973	CLRT
B974	NNRK
B975	OCRK
B976	NORK
B977	NXRK
B980	NGR
B981	OGR
B982	XGR
B983	FLOGR
B984	LLGCR
B985	LLGHR
B986	MLGR
B987	DLGR
B988	ALCGR
B989	SLBGR
B98A	CSPG
B98B	RDP
B98D	EPSW
B98E	IDTE
B98F	CRDTE
B990	TRTT
B991	TRTO
B992	TROT
B993	TROO
B994	LLCR
B995	LLHR
B996	MLR
B997	DLR
B998	ALCR
B999	SLBR
B99A	EPAIR
B99B	ESAIR

OpCode	Mnemonic
B99D	ESEA
B99E	PTI
B99F	SSAIR
B9A1	TPEI
B9A2	PTF
B9AA	LPTEA
B9AC	IRBM
B9AE	RRBM
B9AF	PFMF
B9B0	CU14
B9B1	CU24
B9B2	CU41
B9B3	CU42
B9BD	TRTRE
B9BE	SRSTU
B9BF	TRTE
B9C0	SELFHR
B9C8	AHHHR
B9C9	SHHHR
B9CA	ALHHHR
B9CB	SLHHHR
B9CD	CHHR
B9CF	CLHHR
B9D8	AHHLR
B9D9	SHHLR
B9DA	ALHHLR
B9DB	SLHHLR
B9DD	CHLR
B9DF	CLHLR
B9E0	LOCFHR
B9E1	POPCNT
B9E2	LOCGR
B9E3	SELGR
B9E4	NGRK
B9E5	NCGRK
B9E6	OGRK
B9E7	XGRK
B9E8	AGRK
B9E9	SGRK
B9EA	ALGRK
B9EB	SLGRK
B9EC	MGRK
B9ED	MSGRKC
B9F0	SELR
B9F2	LOCR
B9F4	NRK
B9F5	NCRK
B9F6	ORK
B9F7	XRK
B9F8	ARK
B9F9	SRK
B9FA	ALRK
B9FB	SLRK
B9FD	MSRKC
BA	CS
BB	CDS
BD	CLM
BE	STCM
BF	ICM
C00	LARL
C01	LGFI
C04	BRCL
C05	BRASL
C06	XIHF
C07	XILF
C08	IIHF
C09	IILF
C0A	NIHF
C0B	NILF
C0C	OIHF
C0D	OILF
C0E	LLIHF
C0F	LLILF
C20	MSGFI
C21	MSFI
C24	SLGFI

OpCode	Mnemonic
C25	SLFI
C28	AGFI
C29	AFI
C2A	ALGFI
C2B	ALFI
C2C	CGFI
C2D	CFI
C2E	CLGFI
C2F	CLFI
C42	LLHRL
C44	LGHRL
C45	LHRL
C46	LLGHRL
C47	STHRL
C48	LGRL
C4B	STGRL
C4C	LGFRL
C4D	LRL
C4E	LLGFRL
C4F	STRL
C5	BPRP
C60	EXRL
C62	PFDRL
C64	CGHRL
C65	CHRL
C66	CLGHRL
C67	CLHRL
C68	CGRL
C6A	CLGRL
C6C	CGFRL
C6D	CRL
C6E	CLGFRL
C6F	CLRL
C7	BPP
C80	MVCOS
C81	ECTG
C82	CSST
C84	LPD
C85	LPDG
CC6	BRCTH
CC8	AIH
CCA	ALSIH
CCB	ALSIHN
CCD	CIH
CCF	CLIH
D0	TRTR
D1	MVN
D2	MVC
D3	MVZ
D4	NC
D5	CLC
D6	OC
D7	XC
D9	MVCK
DA	MVCP
DB	MVCS
DC	TR
DD	TRT
DE	ED
DF	EDMK
E1	PKU
E2	UNPKU
E302	LTG
E303	LRAG
E304	LG
E306	CVBY
E308	AG
E309	SG
E30A	ALG
E30B	SLG
E30C	MSG
E30D	DSG
E30E	CVBG
E30F	LRVG
E312	LT
E313	LRAY

OpCode	Mnemonic
E314	LGF
E315	LGH
E316	LLGF
E317	LLGT
E318	AGF
E319	SGF
E31A	ALGF
E31B	SLGF
E31C	MSGF
E31D	DSGF
E31E	LRV
E31F	LRVH
E320	CG
E321	CLG
E324	STG
E325	NTSTG
E326	CVDY
E32A	LZRG
E32E	CVDG
E32F	STRVG
E330	CGF
E331	CLGF
E332	LTGF
E334	CGH
E336	PFD
E338	AGH
E339	SGH
E33A	LLZRGF
E33B	LZRF
E33C	MGH
E33E	STRV
E33F	STRVH
E346	BCTG
E347	BIC
E348	LLGFSG
E349	STGSC
E34C	LGG
E34D	LGSC
E350	STY
E351	MSY
E353	MSC
E354	NY
E355	CLY
E356	OY
E357	XY
E358	LY
E359	CY
E35A	AY
E35B	SY
E35C	MFY
E35E	ALY
E35F	SLY
E370	STHY
E371	LAY
E372	STCY
E373	ICY
E375	LAEY
E376	LB
E377	LGB
E378	LHY
E379	CHY
E37A	AHY
E37B	SHY
E37C	MHY
E380	NG
E381	OG
E382	XG
E383	MSGC
E384	MG
E385	LGAT
E386	MLG
E387	DLG
E388	ALCG
E389	SLBG
E38E	STPQ
E38F	LPQ

OpCode	Mnemonic
E390	LLGC
E391	LLGH
E394	LLC
E395	LLH
E396	ML
E397	DL
E398	ALC
E399	SLB
E39C	LLGTAT
E39D	LLGFAT
E39F	LAT
E3C0	LBH
E3C2	LLCH
E3C3	STCH
E3C4	LHH
E3C6	LLHH
E3C7	STHH
E3C8	LFHAT
E3CA	LFH
E3CB	STFH
E3CD	CHF
E3CF	CLHF
E500	LASP
E501	TPROT
E502	STRAG
E50A	MVCRL
E50E	MVCSK
E50F	MVCDK
E544	MVHHI
E548	MVGHI
E54C	MVHI
E554	CHHSI
E555	CLHHSI
E558	CGHSI
E559	CLGHSI
E55C	CHSI
E55D	CLFHSI
E560	TBEGIN
E561	TBEGINC
E601	VLEBRH
E602	VLEBRG
E603	VLEBRF
E604	VLLEBRZ
E605	VLBRREP
E606	VLBR
E607	VLER
E609	VSTEBRH
E60A	VSTEBRG
E60B	VSTEBRF
E60E	VSTBR
E60F	VSTER
E634	VPKZ
E635	VLRL
E637	VLRLR
E63C	VUPKZ
E63D	VSTRL
E63F	VSTRLR
E649	VLIP
E650	VCVB
E651	VCLZDP
E652	VCVBG
E654	VUPKZH
E655	VCNF
E656	VCLFNH
E658	VCVD
E659	VSRP
E65A	VCVDG
E65B	VPSOP
E65C	VUPKZL
E65D	VCFN
E65E	VCLFNL
E65F	VTP
E670	VPKZR
E671	VAP
E672	VSRPR
E673	VSP

OpCode	Mnemonic
E674	VSCHP
E675	VCRNF
E677	VCP
E678	VMP
E679	VMSP
E67A	VDP
E67B	VRP
E67C	VSCSHP
E67D	VCSPH
E67E	VSDP
E700	VLEB
E701	VLEH
E702	VLEG
E703	VLEF
E704	VLLEZ
E705	VLREP
E706	VL
E707	VLBB
E708	VSTEB
E709	VSTEH
E70A	VSTEG
E70B	VSTEF
E70E	VST
E712	VGEG
E713	VGEF
E71A	VSCEG
E71B	VSCEF
E721	VLGV
E722	VLVG
E727	LCBB
E730	VESL
E733	VERLL
E736	VLM
E737	VLL
E738	VESRL
E73A	VESRA
E73E	VSTM
E73F	VSTL
E740	VLEIB
E741	VLEIH
E742	VLEIG
E743	VLEIF
E744	VGBM
E745	VREPI
E746	VGM
E74A	VFTCI
E74D	VREP
E750	VPOPCT
E752	VCTZ
E753	VCLZ
E756	VLR
E75C	VISTR
E75F	VSEG
E760	VMRL
E761	VMRH
E762	VLVGP
E764	VSUM
E765	VSUMG
E766	VCKSM
E767	VSUMQ
E768	VN
E769	VNC
E76A	VO
E76B	VNO
E76C	VNX
E76D	VX
E76E	VNN
E76F	VOC
E770	VESLV
E772	VERIM
E773	VERLLV
E774	VSL
E775	VSLB
E777	VSLDB
E778	VESRLV
E77A	VESRAV

OpCode	Mnemonic
E77C	VSRL
E77D	VSRLB
E77E	VSRA
E77F	VSRAB
E780	VFEE
E781	VFENE
E782	VFAE
E784	VPDI
E786	VSLD
E787	VSRD
E785	VBPERM
E78A	VSTRC
E78B	VSTRS
E78C	VPERM
E78D	VSEL
E78E	VFMS
E78F	VFMA
E794	VPK
E795	VPKLS
E797	VPKS
E79E	VFNMS
E79F	VFNMA
E7A1	VMLH
E7A2	VML
E7A3	VMH
E7A4	VMLE
E7A5	VMLO
E7A6	VME
E7A7	VMO
E7A9	VMALH
E7AA	VMAL
E7AB	VMAH
E7AC	VMALE
E7AD	VMALO
E7AE	VMAE
E7AF	VMAO
E7B4	VGFM
E7B8	VMSL
E7B9	VACCC
E7BB	VAC
E7BC	VGFMA
E7BD	VSBCBI
E7BF	VSBI
E7C0	VCLGD
E7C0	VCLFP
E7C1	VCDLG
E7C1	VCFPL
E7C2	VCGD
E7C2	VCSFP
E7C3	VCDG
E7C3	VCFPS
E7C4	VFLL
E7C5	VFLR
E7C7	VFI
E7CA	WFK
E7CB	WFC
E7CC	VFPSO
E7CE	VFSQ
E7D4	VUPLL
E7D5	VUPLH
E7D6	VUPL
E7D7	VUPH
E7D8	VTM
E7D9	VECL
E7DB	VEC
E7DE	VLC
E7DF	VLP
E7E2	VFS
E7E3	VFA
E7E5	VFD
E7E7	VFM
E7E8	VFCE
E7EA	VFCHE
E7EB	VFCH
E7EE	VFMIN
E7EF	VFMAX

OpCode	Mnemonic
E7F0	VAVGL
E7F1	VACC
E7F2	VAVG
E7F3	VA
E7F5	VSCBI
E7F7	VS
E7F8	VCEQ
E7F9	VCHL
E7FB	VCH
E7FC	VMNL
E7FD	VMXL
E7FE	VMN
E7FF	VMX
E8	MVCIN
E9	PKA
EA	UNPKA
EB04	LMG
EB0A	SRAG
EB0B	SLAG
EB0C	SRLG
EB0D	SLLG
EB0F	TRACG
EB14	CSY
EB1C	RLLG
EB1D	RLL
EB20	CLMH
EB21	CLMY
EB23	CLT
EB24	STMG
EB25	STCTG
EB26	STMH
EB2B	CLGT
EB2C	STCMH
EB2D	STCMY
EB2F	LCTLG
EB30	CSG
EB31	CDSY
EB3E	CDSG
EB44	BXHG
EB45	BXLEG
EB4C	ECAG
EB51	TMY
EB52	MVIY
EB54	NIY
EB55	CLIY
EB56	OIY
EB57	XIY
EB6A	ASI
EB6E	ALSI
EB71	LPSWEY
EB7A	AGSI
EB7E	ALGSI
EB80	ICMH
EB81	ICMY
EB8E	MVCLU
EB8F	CLCLU
EB90	STMY
EB96	LMH
EB98	LMY
EB9A	LAMY
EB9B	STAMY
EBC0	TP
EBDC	SRAK
EBDD	SLAK
EBDE	SRLK
EBDF	SLLK
EBE0	LOCFH
EBE1	STOCFH
EBE2	LOCG
EBE3	STOCG
EBE4	LANG
EBE6	LAOG
EBE7	LAXG
EBE8	LAAG
EBEA	LAALG
EBF2	LOC

OpCode	Mnemonic
EBF3	STOC
EBF4	LAN
EBF6	LAO
EBF7	LAX
EBF8	LAA
EBFA	LAAL
EC42	LOCHI
EC44	BRXHG
EC45	BRXLG
EC46	LOCGHI
EC4E	LOCHHI
EC51	RISBLG
EC54	RNSBG
EC55	RISBG
EC56	ROSBG
EC57	RXSBG
EC59	RISBGN
EC5D	RISBHG
EC64	CGRJ
EC65	CLGRJ
EC70	CGIT
EC71	CLGIT
EC72	CIT
EC73	CLFIT
EC76	CRJ
EC77	CLRJ
EC7C	CGIJ
EC7D	CLGIJ
EC7E	CIJ
EC7F	CLIJ
ECD8	AHIK
ECD9	AGHIK
ECDA	ALHSIK
ECDB	ALGHSIK
ECE4	CGRB
ECE5	CLGRB
ECF6	CRB
ECF7	CLRB
ECFC	CGIB
ECFD	CLGIB
ECFE	CIB
ECFF	CLIB
ED04	LDEB
ED05	LXDB
ED06	LXEB
ED07	MXDB
ED08	KEB
ED09	CEB
ED0A	AEB
ED0B	SEB
ED0C	MDEB
ED0D	DEB
ED0E	MAEB
ED0F	MSEB
ED10	TCEB
ED11	TCDB
ED12	TCXB
ED14	SQEB
ED15	SQDB
ED17	MEEB
ED18	KDB
ED19	CDB
ED1A	ADB
ED1B	SDB
ED1C	MDB
ED1D	DDB
ED1E	MADB
ED1F	MSDB
ED24	LDE
ED25	LXD
ED26	LXE
ED2E	MAE
ED2F	MSE
ED34	SQE
ED35	SQD
ED37	MEE

OpCode	Mnemonic
ED38	MAYL
ED39	MYL
ED3A	MAY
ED3B	MY
ED3C	MAYH
ED3D	MYH
ED3E	MAD
ED3F	MSD
ED40	SLDT
ED41	SRDT
ED48	SLXT
ED49	SRXT
ED50	TDCET
ED51	TDGET
ED54	TDCDT
ED55	TDGDT
ED58	TDCXT
ED59	TDGXT
ED64	LEY
ED65	LDY
ED66	STEY
ED67	STDY
EDA8	CZDT
EDA9	CZXT
EDAA	CDZT
EDAB	CXZT
EDAC	CPDT
EDAD	CPXT
EDAE	CDPT
EDAF	CXPT
EE	PLO
EF	LMD
F0	SRP
F1	MVO
F2	PACK
F3	UNPK
F8	ZAP
F9	CP
FA	AP
FB	SP
FC	MP
FD	DP

Condition Codes

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	8	4	2	1
General Instructions				
Add	Zero	< Zero	> Zero	Overflow
Add Halfword	Zero	< Zero	> Zero	Overflow
Add Halfword Immediate	Zero	< Zero	> Zero	Overflow
Add High	Zero	< Zero	> Zero	Overflow
Add Immediate	Zero	< Zero	> Zero	Overflow
Add Immediate High	Zero	< Zero	> Zero	Overflow
Add Logical	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
Add Logical High	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
Add Logical Immediate	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
Add Logical with Carry	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
Add Logical with Signed Immediate	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
Add Logical with Signed Immediate High	Zero, no carry	Not zero, no carry	Zero, carry	Not zero, carry
AND	Zero	Not zero	—	—
AND Immediate	ANDed bits zero	ANDed bits not zero	—	—
AND with Complement	Zero	Not zero	—	—
Checksum	Checksum complete	—	—	CPU-determined completion
Cipher Message	Normal completion	Verification mismatch	—	Partial completion
Cipher Message with Authentication	Normal completion	Verification mismatch	Partial completion (LAAD or LPC zero)	Partial completion (time out)
Cipher Message with Chaining	Normal completion	Verification mismatch	—	Partial completion
Cipher Message with Cipher Feedback	Normal completion	Verification mismatch	—	Partial completion
Cipher Message with Counter	Normal completion	Verification mismatch	—	Partial completion
Cipher Message with Output Feedback	Normal completion	Verification mismatch	—	Partial completion
Compare	Equal	First op low	First op high	—
Compare and Form Codeword	Equal	First op low and ctl = 0, or first op high and ctl = 1	First op high and ctl = 0, or first op low and ctl = 1	—
Compare and Swap	Equal	Not equal	—	—
Compare and Swap and Store	Equal	Not equal	—	—
Compare Double and Swap	Equal	Not equal	—	—
Compare Halfword	Equal	First op low	First op high	—
Compare Halfword Immediate	Equal	First op low	First op high	—
Compare Halfword Relative Long	Equal	First op low	First op high	—
Compare High	Equal	First op low	First op high	—
Compare Immediate	Equal	First op low	First op high	—
Compare Immediate High	Equal	First op low	First op high	—
Compare Logical	Equal	First op low	First op high	—
Compare Logical Characters under Mask	Equal, or Mask is zero	First op low	First op high	—
Compare Logical High	Equal	First op low	First op high	—
Compare Logical Immediate	Equal	First op low	First op high	—
Compare Logical Immediate High	Equal	First op low	First op high	—
Compare Logical Long	Equal	First op low	First op high	—
Compare Logical Long Extended	Equal	First op low	First op high	CPU-determined completion
Compare Logical Long Unicode	Equal	First op low	First op high	CPU-determined completion
Compare Logical Relative Long	Equal	First op low	First op high	—
Compare Logical String	Equal	First op low	First op high	CPU-determined completion

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	8	4	2	1
Compare Relative Long	Equal	First op low	First op high	—
Compare until Substring Equal	Equal substring	Last bytes equal	Last bytes unequal	CPU-determined completion
Compression Call	Second op end	First op end, not second op end	—	CPU-determined completion
Compute Intermediate Message Digest	Normal completion	—	—	Partial completion
Compute Last Message Digest	Normal completion	—	—	Partial completion
Compute Message Authentication Code	Normal completion	Verification mismatch	—	Partial completion
Convert UTF-16 to UTF-32	Data processed	First op full	Invalid low surrogate	CPU-determined completion
Convert UTF-16 to UTF-8	Data processed	First op full	Invalid low surrogate	CPU-determined completion
Convert UTF-32 to UTF-16	Data processed	First op full	Invalid UTF-32 character	CPU-determined completion
Convert UTF-32 to UTF-8	Data processed	First op full	Invalid UTF-32 character	CPU-determined completion
Convert UTF-8 to UTF-16	Data processed	First op full	Invalid UTF-8 character	CPU-determined completion
Convert UTF-8 to UTF-32	Data processed	First op full	Invalid UTF-8 character	CPU-determined completion
Exclusive OR	Zero	Not zero	—	—
Exclusive OR Immediate	XORed bits zero	XORed bits not zero	—	—
Find Leftmost One	No one bit found	—	One bit found	—
Insert Characters under Mask	All zero, or mask is zero	Leftmost bit = 1	Not zero, but with leftmost bit = 0	—
Load and Test	Zero	< Zero	> Zero	—
Load Complement	Zero	< Zero	> Zero	Overflow
Load Negative	Zero	< Zero	—	—
Load Positive	Zero	—	> Zero	Overflow
Move Long	Operand lengths equal	First op shorter	First op longer	Overlap
Move Long Extended	Operand lengths equal	First op shorter	First op longer	CPU-determined completion
Move Long Unicode	Operand lengths equal	First op shorter	First op longer	CPU-determined completion
Move String	—	Second op moved	—	CPU-determined completion
Multiply Single (MSC, MSGC, MSGRKC, MRRKC)	Zero, no overflow	< Zero, no overflow	> Zero, no overflow	Overflow
NAND	Zero	Not zero	—	—
NOR	Zero	Not zero	—	—
NOT Exclusive OR	Zero	Not zero	—	—
OR	Zero	Not zero	—	—
OR Immediate	ORed bits zero	ORed bits not zero	—	—
OR with Complement	Zero	Not zero	—	—
Perform Cryptographic Computation	Normal completion	Verification mismatch	—	Partial completion
Perform Locked Operation (test bit zero)	Equal	First op not equal	First op equal, third op not equal	—
Perform Locked Operation (test bit one)	Code valid	—	—	Code invalid
Perform Random Number Operation	Normal completion	—	—	Partial completion
Population Count	Zero	Not zero	—	—

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	**8**	**4**	**2**	**1**
Rotate Then AND Selected Bits	Selected bits zero	Selected bits not zero	—	—
Rotate Then Exclusive OR Selected Bits	Selected bits zero	Selected bits not zero	—	—
Rotate Then Insert Selected Bits	Zero	< zero	> zero	—
Rotate Then OR Selected Bits	Selected bits zero	Selected bits not zero	—	—
Search String, Search String Unicode	—	Character found	Character not found	CPU-determined completion
Set Program Mask[4]	See Note	See Note	See Note	See Note
Shift Left (Double / Single)	Zero	< Zero	> Zero	Overflow
Shift Right (Double / Single)	Zero	< Zero	> Zero	—
Store Clock (STCK, STCKE or STCKF)	Set state	Not-set state	Error state	Stopped state or not operational
Store Facility List Extended	Complete list stored	—	—	Incomplete list stored
Subtract	Zero	< Zero	> Zero	Overflow
Subtract Halfword	Zero	< Zero	> Zero	Overflow
Subtract High	Zero	< Zero	> Zero	Overflow
Subtract Logical	—	Not zero, borrow	Zero, no borrow	Not zero, no borrow
Subtract Logical High	—	Not zero, borrow	Zero, no borrow	Not zero, no borrow
Subtract Logical Immediate	—	Not zero, borrow	Zero, no borrow	Not zero, no borrow
Subtract Logical with Borrow	Zero, borrow	Not zero, borrow	Zero, no borrow	Not zero, no borrow
Test Addressing Mode	24-bit mode	31-bit mode	—	64-bit mode
Test and Set	Leftmost bit zero	Leftmost bit one	—	—
Test under Mask (TM)	All zeros, or mask is zero	Mixed 0's and 1's	—	All ones
Test under Mask (TMH, TMHH, TMHL, TML, TMLH, TMLL)	All zeros or mask is zero	Mixed 0's and 1's and leftmost bit zero	Mixed 0's and 1's and leftmost bit one	All ones
Test under Mask High, Low	All zeros or mask is zero	Mixed 0's and 1's and leftmost bit zero	Mixed 0's and 1's and leftmost bit one	All ones
Transaction Begin	Successful	—	—	—
Transaction End	In TX mode	—	Not in TX mode	—
Translate and Test, Translate and Test Reverse	All zeros	Not zero, scan incomplete	Not zero, scan complete	—
Translate and Test Extended, Translate and Test Reverse Extended	All selected function codes zero	Nonzero function code selected	—	CPU-determined completion
Translate Extended	Data processed	First op byte equal test byte	—	CPU-determined completion
Translate One to One, One to Two, Two to One, Two to Two	Character not found	Character found	—	CPU determined completion
Unpack ASCII	Sign plus	Sign minus	—	Sign invalid
Unpack Unicode	Sign plus	Sign minus	—	Sign invalid
Update Tree	Compare equal at current node on path	Path complete, no nodes compared equal	—	Path not complete and compared register negative
Decimal Instructions				
Add Decimal	Zero	< Zero	> Zero	Overflow
Compare Decimal	Equal	First op low	First op high	—
Edit	Zero	< Zero	> Zero	—
Edit and Mark	Zero	< Zero	> Zero	—
Shift and Round Decimal	Zero	< Zero	> Zero	Overflow
Subtract Decimal	Zero	< Zero	> Zero	Overflow
Test Decimal	Digits and sign valid	Sign invalid	Digit invalid	Sign and digit invalid
Zero and Add	Zero	< Zero	> Zero	Overflow

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	8	4	2	1
Floating-Point Instructions				
Add	Zero	< Zero	> Zero	NaN
Add Normalized	Zero	< Zero	> Zero	—
Add Unnormalized	Zero	< Zero	> Zero	—
Compare (BFP)	Equal	First op low	First op high	Unordered
Compare (HFP)	Equal	First op low	First op high	—
Compare and Signal	Equal	First op low	First op high	Unordered
Compare Biased Exponent	Equal	First op low	First op high	Unordered
Convert BFP to HFP	Zero	< Zero	> Zero	Special case
Convert HFP to BFP	Zero	< Zero	> Zero	Special case
Convert to Fixed	Zero	< Zero	> Zero	Special case
Convert to Logical	Zero	< Zero	> Zero	Special case
Convert to Packed	Zero	< Zero	> Zero	Special case
Convert to Zoned	Zero	< Zero	> Zero	Special case
Divide to Integer	Remainder complete, quotient normal	Remainder complete, quotient overflow or NaN	Remainder incomplete, quotient normal	Remainder incomplete, quotient overflow or NaN
Load and Test (BFP)	Zero	< Zero	> Zero	NaN
Load and Test (HFP)	Zero	< Zero	> Zero	—
Load Complement (BFP)	Zero	< Zero	> Zero	NaN
Load Complement (HFP)	Zero	< Zero	> Zero	—
Load Negative (BFP)	Zero	< Zero	—	NaN
Load Negative (HFP)	Zero	< Zero	—	—
Load Positive (BFP)	Zero	—	> Zero	NaN
Load Positive (HFP)	Zero	—	> Zero	—
Perform Floating-Point Operation (T=0)	Normal result	Nontrap exception	Trap exception	—
Perform Floating-Point Operation (T=1)	Function valid	—	—	Function invalid
Subtract	Zero	< Zero	> Zero	NaN
Subtract Normalized	Zero	< Zero	> Zero	—
Subtract Unnormalized	Zero	< Zero	> Zero	—
Test Data Class	Zero (no match)	One (match)	—	—
Test Data Group	Zero (no match)	One (match)	—	—
Vector-Facility Instructions				
Load Count to Block Boundary	= 16	—	—	< 16
Vector Add Decimal[5]	No overflow	—	—	Overflow
Vector Compare Decimal	Equal	First op low	First op high	—
Vector Compare Equal[5]	All elements equal	Some elements equal	—	No element equal
Vector Compare High Logical[5]	All elements high	Some elements high	—	No element high
Vector Compare High[5]	All elements high	Some elements high	—	No element high
Vector Convert to Binary[5]	No overflow	—	—	Overflow
Vector Convert to Decimal[5]	No overflow	—	—	Overflow
Vector Count Leading Zero Digits[5]	Zero	< Zero	> Zero	Invalid digit or sign
Vector Divide Decimal[5]	Zero	< Zero	> Zero	Overflow
Vector Element Compare	Equal	Low	High	—
Vector Element Compare Logical	Equal	Low	High	—
Vector Find Any Element Equal[5]	None equal, zero found	Equal element found, no zeros if ZS=1	Equal element found and zero found	No equal elements, no zeros
Vector Find Element Equal[5]	Zero found	Equal element found, no zeros	Equal element found, and zero	Not equal, no zeros
Vector Find Element Not Equal[5]	Zero found	Not equal element found, less than	Not equal found, greater than	Equal, no zero
Vector FP Compare And Signal Scalar	Elements equal	First element low	First element high	Elements unordered

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	**8**	**4**	**2**	**1**
Vector FP Compare Equal[5]	All elements equal	Mix of equal and unequal (or unordered) elements	—	All elements not equal (or unordered)
Vector FP Compare High Or Equal[5]	All elements ≥	Mix of ≥ and <	—	All elements < (or unordered)
Vector FP Compare High[5]	All elements >	Mix of > and ≤	—	All elements ≤ (or unordered)
Vector FP Compare Scalar	Elements equal	First element low	First element high	Elements unordered
Vector FP Test Data Class Immediate	Match	Selected bit 1 for some (but not all) elements	—	No match
Vector Isolate String[5]	Zero element found	—	—	All elements nonzero
Vector Multiply and Shift Decimal[5]	No overflow	—	—	Overflow
Vector Multiply Decimal[5]	No overflow	—	—	Overflow
Vector Pack Logical Saturate[5]	No saturation	Some saturated		All saturated
Vector Pack Saturate[5]	No saturation	Some saturated		All saturated
Vector Pack Zoned Register[5]	Zero	< Zero	> Zero	Overflow or invalid digit or sign
Vector Perform Sign Operation Decimal[5]	Zero	< Zero	> Zero	Overflow
Vector Remainder Decimal[5]	Zero	< Zero	> Zero	Overflow
Vector Shift and Divide Decimal[5]	Zero	< Zero	> Zero	Overflow
Vector Shift and Round Decimal[5]	Zero	< Zero	> Zero	Overflow
Vector Shift and Round Decimal Register[5]	Zero	< Zero	> Zero	Overflow
Vector String Range Compare[5]	Zero found	At least one in ranges, no zero	At least one in ranges, zero found	No ranges match, no zeros
Vector String Search	No match and either ZS is 0 or no zero byte detected	No match and ZS is 1 and a zero byte detected	full match	partial match
Vector Subtract Decimal[5]	No overflow	—	—	Overflow
Vector Test Decimal[5]	Digits and sign valid	Sign invalid	Digit invalid	Sign and digit invalid
Vector Test Under Mask	All zeros or mask zero	Mixed	—	All ones
Control Instructions				
Compare and Replace DAT Table Entry	Equal	Not equal	—	—
Compare and Swap and Purge	Equal	Not equal	—	—
Diagnose[1]	See note	See note	See note	See note
Extract Stacked State	Branch state entry	Program call state entry	—	—
Insert Address Space Control	Primary-space mode	Secondary-space mode	Access register mode	Home-space mode
Load Address Space Parameters	Parameters loaded	Primary not available	Secondary not authorized or not available	Space-switch event
Load Page-Table-Entry Address	Address returned; STE.P=0	Address returned; STE.P=1	Invalid bit on in RTE or STE.	Exception condition exists.
Load PSW[3]	See note	See note	See note	See note
Load PSW Extended[3]	See note	See note	See note	See note
Load Real Address[2]	Translation available	Segment table entry invalid	Page table entry invalid	See note

Condition Code →	0	1	2	3
Mask Bit Value →	**8**	**4**	**2**	**1**
Move Page	Data moved	First op invalid, both valid in ES, locked, or ES error	Second op invalid	—
Move to Primary	Length ≤ 256	—	—	Length > 256
Move to Secondary	Length ≤ 256	—	—	Length > 256
Move with Key	Length ≤ 256	—	—	Length > 256
Move with Optional Specifications	Length ≤ 4096	—	—	Length > 4096
Page In	Operation completed	ES data error	—	ES block not available
Page Out	Operation completed	ES data error	—	ES block not available
Perform Timing Facility Function	Function performed	—	—	Function not available
Perform Topology Function	Initiated	—	Rejected	—
Program Return	See note	See note	See note	See note
Reset Reference Bit Extended	Ref = 0, Chg = 0	Ref = 0, Chg = 1	Ref = 1, Chg = 0	Ref = 1, Chg = 1
Resume Program[3]	See note	See note	See note	See note
Set Clock	Set	Secure	—	Not operational
Signal Processor	Accepted	Status stored	Busy	Not operational
Store System Information	Info provided	—	—	Info not available
Test Access	ALET = 0	ALET uses DUALD	ALET uses PSALD	ALET = 1 or causes ART exception
Test Pending External Interruption	None pending	One or more pending	—	—
Test Block	Usable	Unusable	—	—
Test Protection	Fetch and store allowed	Fetch allowed; no store allowed	No fetch or store allowed	Translation not available
Input/Output Instructions				
Cancel Subchannel	Function started	—	—	Not operational
Clear Subchannel	Function started	—	—	Not operational
Halt Subchannel	Function started	Non-intermediate status pending	Busy	Not operational
Modify Subchannel	Function executed	Status pending	Busy	Not operational
Reset Channel Path	Function started	—	Busy	Not operational
Resume Subchannel	Function started	Status pending	Not applicable	Not operational
Start Subchannel	Function started	Status pending	Busy	Not operational
Store Channel Report Word	CRW stored	Zeros stored	—	—
Store Subchannel	SCHIB stored	—	—	Not operational
Test Pending Interruption	Interruption not pending	Interruption code stored	—	—
Test Subchannel	IRB stored; status pending	IRB stored; not status pending	—	Not operational
Specialized-Function-Assist Instructions				
Compute Digital Signature Authentication	verify: signature verified; sign: normal completion	verify: public key not on curve; sign: KVP mismatch; sign & verify: reserved area	verify: signature incorrect or invalid; sign: random number not invertible if deterministic	partial completion

Condition Code ➔	0	1	2	3
Mask Bit Value ➔	8	4	2	1
DEFLATE Conversion Call	Normal completion	op1 length insufficient	op2 length insufficient or invalid input	CPU-determined completion
Neural Network Processing Assist	Normal completion	Response code stored	—	CPU-determined completion
Query Processor Activity Counter Information	Complete information stored	—	—	Incomplete information stored
Sort Lists	Normal completion	First or second operand length is insufficient to continue	incomplete or empty input list encountered	CPU-determined completion

Notes:

1 For Diagnose, the resulting condition code is model-dependent.

2 For Load Real Address, condition code 3 is set if address-space-control element not available, region-table entry outside table or invalid, segment-table entry outside table, or, for LRA in 24- or 31-bit mode when bits 0-32 of entry address not all zeros, segment- or page-table entry invalid.

3 For Load PSW, Load PSW Extended, and Resume Program, the condition code is loaded from the condition-code field of the second operand.

4 For Set Program Mask, the condition code is loaded from bit positions 2 and 3 of the first operand.

5 For various vector-facility instructions, the condition code is optionally set based on the CS control in the M^5 field of the instruction.

Assembler Instructions

Function	Mnemonic	Meaning
Option control	*PROCESS	Specify assembler options
	ACONTROL	Dynamically modify options
Data definition	CCW	Define channel command word
	CCW0	Define format-0 channel command word
	CCW1	Define format-1 channel command word
	DC	Define constant
	DS	Define storage
Program sectioning and linking	ALIAS	Rename external symbol
	AMODE	Specify addressing mode
	CATTR	Define class/part name and attributes
	COM	Identify common control section
	CSECT	Identify control section
	CXD	Cumulative length of external dummy section
	DSECT	Identify dummy section
	DXD	Define external dummy section
	ENTRY	Identify entry-point symbol
	EXTRN	Identify external symbol
	LOCTR	Specify multiple location counters
	RMODE	Specify residence mode
	RSECT	Identify read-only control section
	START	Start assembly
	WXTRN	Identify weak external symbol
	XATTR	Declare external symbol attributes
Base register assignment	DROP	Drop base address register
	USING	Use base address and register
Control of listing	AEJECT	Start new page in macro definition
	ASPACE	Space lines in macro definition
	CEJECT	Conditional start new page
	EJECT	Start new page
	PRINT	Control listing contents
	SPACE	Space listing
	TITLE	Identify assembly output
Program control	ADATA	Provide data for SYSADATA file
	CNOP	Conditional no operation
	COPY	Copy predefined source coding
	END	End assembly
	EQU	Equate symbol

Function	Mnemonic	Meaning
	EXITCTL	Program control data for I/O exits
	ICTL	Input format control
	ISEQ	Input sequence checking
	LTORG	Begin literal pool
	OPSYN	Equate operation code
	ORG	Set location counter
	POP	Restore ACONTROL, PRINT, or USING status
	PUNCH	Punch a record
	PUSH	Save current ACONTROL, PRINT, or USING status
	REPRO	Reproduce following record
Conditional assembly	ACTR	Conditional assembly branch counter
	AGO	Unconditional branch
	AIF	Conditional branch
	AINSERT	Create input record
	ANOP	Assembly no operation
	AREAD	Assign input record to SETC symbol
	GBLA	Define global SETA symbol
	GBLB	Define global SETB symbol
	GBLC	Define global SETC symbol
	LCLA	Define local SETA symbol
	LCLB	Define local SETB symbol
	LCLC	Define local SETC symbol
	MHELP	Trace macro flow
	MNOTE	Generate message
	SETA	Set arithmetic variable symbol
	SETAF	Set arithmetic variable symbol from external function
	SETB	Set binary variable symbol
	SETC	Set character variable symbol
	SETCF	Set character variable symbol from external function
Macro definition	MACRO	Macro definition header
	MEND	Macro definition trailer
	MEXIT	Macro expansion exit

Source: SC26-4940.

CNOP Alignment

Quadword															
Doubleword								Doubleword							
Fullword				Fullword				Fullword				Fullword			
Halfword		Halfword		Halfword		Halfword		Halfword		Halfword		Halfword		Halfword	
Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte
0,4		2,4		0,4		2,4		0,4		2,4		0,4		2,4	
0,8		2,8		4,8		6,8		0,8		2,8		4,8		6,8	
0,16		2,16		4,16		6,16		8,16		10,16		12,16		14,16	

For byte offset and boundary values greater than 16, see *IBM High Level Assembler for z/OS, z/VM & z/VSE Language Reference* (SC26-4940).

Extended-Mnemonic Instructions for Branch on Condition and Branch Indirect on Condition

	Extended Mnemonic for:				
Use	BC	BCR	BIC	Meaning	M_1 Value*
Control	B	BR	BI	Unconditional branch	15
	NOP	NOPR	—	No operation	0
After Compare Instructions (A:B)	BH	BHR	BIH	Branch on A High	2
	BL	BLR	BIL	Branch on A Low	4
	BE	BER	BIE	Branch on A Equal B	8
	BNH	BNHR	BINH	Branch on A Not High	13
	BNL	BNLR	BINL	Branch on A Not Low	11
	BNE	BNER	BINE	Branch on A Not Equal B	7
After Arithmetic Instructions	BP	BPR	BIP	Branch on Plus	2
	BM	BMR	BIM	Branch on Minus	4
	BZ	BZR	BIZ	Branch on Zero	8
	BO	BOR	BIO	Branch on Overflow	1
	BNP	BNPR	BINP	Branch on Not Plus	13
	BNM	BNMR	BINM	Branch on Not Minus	11
	BNZ	BNZR	BINZ	Branch on Not Zero	7

Use	Extended Mnemonic for: BC	BCR	BIC	Meaning	M_1 Value*
	BNO	BNOR	BINO	Branch on No Overflow	14
After Test under Mask Instruction	BO	BOR	BIO	Branch if Ones	1
	BM	BMR	BIM	Branch if Mixed	4
	BZ	BZR	BIZ	Branch if Zeros	8
	BNO	BNOR	BINO	Branch if Not Ones	14
	BNM	BNMR	BINM	Branch if Not Mixed	11
	BNZ	BNZR	BINZ	Branch if Not Zeros	7

Source: SC26-4940.

— Not applicable for BIC

* Extended mnemonic replaces the M_1 field; second operand, not shown, is $D_2(X_2,B_2)$ for RX and RXY formats and R_2 for RR format.

Extended-Mnemonic Instructions for Relative-Branch Instructions

Use	Extended Mnemonic	Meaning	Machine Instr.
General Branch Rel. on Condition	BRU or J	Unconditional Branch Relative	BRC 15,I_2
	BRUL or JLU	Unconditional Branch Relative	BRCL 15,I_2
	JNOP*	No Operation	BRC 0,I_2
After Compare Instructions	BRH or JH*	Branch Relative on A High	BRC 2,I_2
	BRL or JL*	Branch Relative on A Low	BRC 4,I_2
	BRE or JE*	Branch Relative on A Equal B	BRC 8,I_2
	BRNH or JNH*	Branch Relative on A Not High	BRC 13,I_2
	BRNL or JNL*	Branch Relative on A Not Low	BRC 11,I_2
	BRNE or JNE*	Branch Relative on A Not Equal B	BRC 7,I_2
After Arithmetic Instructions	BRP or JP*	Branch Relative on Plus	BRC 2,I_2
	BRM or JM*	Branch Relative on Minus	BRC 4,I_2
	BRZ or JZ*	Branch Relative on Zero	BRC 8,I_2
	BRO or JO*	Branch Relative on Overflow	BRC 1,I_2
	BRNP or JNP*	Branch Relative on Not Plus	BRC 13,I_2
	BRNM or JNM*	Branch Relative on Not Minus	BRC 11,I_2
	BRNZ or JNZ*	Branch Relative on Not Zero	BRC 7,I_2
	BRNO or JNO*	Branch Relative on No Overflow	BRC 14,I_2
After Test under Mask Instruction	BRO or JO*	Branch Relative if Ones	BRC 1,I_2
	BRM or JM*	Branch Relative if Mixed	BRC 4,I_2
	BRZ or JZ*	Branch Relative if Zeros	BRC 8,I_2
	BRNO or JNO*	Branch Relative if Not Ones	BRC 14,I_2
	BRNM or JNM*	Branch Relative if Not Mixed	BRC 11,I_2
	BRNZ or JNZ*	Branch Relative if Not Zeros	BRC 7,I_2
Other Branch Relative Instructions	JAS	Branch Relative and Save	BRAS R_1,I_2
	JASL	Branch Relative and Save Long	BRASL R_1,I_2
	JCT	Branch Relative on Count (32)	BRCT R_1,I_2
	JCTG	Branch Relative on Count (64)	BRCTG R_1,I_2
	JXH	Branch Relative on Index High (32)	BRXH R_1,R_3,I_2
	JXHG	Branch Relative on Index High (64)	BRXHG R_1,R_3,I_2
	JXLE	Br. Rel. on Index Low or Equal (32)	BRXLE R_1,R_3,I_2
	JXLEG	Br. Rel. on Index Low or Equal (64)	BRXLG R_1,R_3,I_2

Source: SC26-4940.

* To obtain BRCL instead of BRC, add L at the end of the B mnemonic or insert L after the J of the J mnemonic. For example, change BRNZ or JNZ to BRNZL or JLNZ.

Extended-Mnemonic Suffixes for Compare-and-Branch, and Compare-and-Trap Instructions

Suffix	Meaning	M_3 Value	Suffix	Meaning	M_3 Value
H	High	2	NH	Not High	12
L	Low	4	NL	Not Low	10
E	Equal	8	NE	Not Equal	6

Explanation:

These suffixes may be appended to the following mnemonics: CGIB, CGIJ, CGIT, CGRB, CGRJ, CGRT, CIB, CIJ, CIT, CLFIT, CLGIB, CLGIJ, CLGIT, CLGRB, CLGRJ, CLGRT, CLGT, CLIB, CLIJ, CLRB, CLRJ, CLRT, CLT, CRB, CRJ, CRT. When the suffix is coded, the M_3 operand must be omitted.

Extended-Mnemonic Suffixes for Load/Store-on-Condition Instructions

Suffix	Meaning	M_3 Value	Suffix	Meaning	M_3 Value
O *	One / Overflow	1	NO *	Not one / Not overflow	14
H	High	2	NH	Not High	13
P *	Plus	2	NP *	Not Plus	13
L	Low	4	NL	Not Low	11
M *	Minus / Mixed	4	NM *	Not Minus / Mixed	11
E	Equal	8	NE	Not Equal	7
Z *	Zero	8	NZ *	Not Zero	7

Explanation:

These suffixes may be appended to the following mnemonics: LOC, LOCG, LOCGHI, LOCGR, LOCHHI, LOCHI, LOCR, LOCFH, LOCFHR, STOC, STOCFH, STOCG. Suffixes marked with an asterisk (*) may not be available on earlier versions of the High Level Assembler.

Extended-Mnemonic Suffixes for Rotate-Then-Insert / AND / OR / Exclusive OR-Selected-Bits Instructions

Extended-Mnemonic Syntax		Basic-Mnemonic Equivalent		Meaning
LHHR	R_1,R_2	RISBHGZ	$R_1,R_2,0,31$	LOAD (HIGH ← HIGH)
LHLR	R_1,R_2	RISBHGZ	$R_1,R_2,0,31,32$	LOAD (HIGH ← LOW)
LLCHHR	R_1,R_2	RISBHGZ	$R_1,R_2,24,31$	LOAD LOG. CH. (HIGH ← HIGH)
LLCHLR	R_1,R_2	RISBHGZ	$R_1,R_2,24,31,32$	LOAD LOG. CH. (HIGH ← LOW)
LLCLHR	R_1,R_2	RISBLGZ	$R_1,R_2,24,31,32$	LOAD LOG. CH. (LOW ← HIGH)
LLHFR	R_1,R_2	RISBLGZ	$R_1,R_2,0,31,32$	LOAD (LOW ← HIGH)
LLHHHR	R_1,R_2	RISBHGZ	$R_1,R_2,16,31$	LOAD LOG. HW. (HIGH ← HIGH)
LLHHLR	R_1,R_2	RISBHGZ	$R_1,R_2,16,31,32$	LOAD LOG. HW. (HIGH ← LOW)
LLHLHR	R_1,R_2	RISBLGZ	$R_1,R_2,16,31,32$	LOAD LOG. HW. (LOW ← HIGH)
NHHR	R_1,R_2	RNSBG	$R_1,R_2,0,31$	AND HIGH (HIGH ← HIGH)
NHLR	R_1,R_2	RNSBG	$R_1,R_2,0,31,32$	AND HIGH (HIGH ← LOW)
NLHR	R_1,R_2	RNSBG	$R_1,R_2,32,63,32$	AND HIGH (LOW ← HIGH)
OHHR	R_1,R_2	ROSBG	$R_1,R_2,0,31$	OR HIGH (HIGH ← HIGH)
OHLR	R_1,R_2	ROSBG	$R_1,R_2,0,31,32$	OR HIGH (HIGH ← LOW)
OLHR	R_1,R_2	ROSBG	$R_1,R_2,32,63,32$	OR HIGH (LOW ← HIGH)
RISBGNZ	R_1,R_2,I_3,I_4,I_5	RISBGN	R_1,R_2,I_3,I_4+128,I_5	Set zero-remaining-bits control to 1.
RISBGZ	R_1,R_2,I_3,I_4,I_5	RISBG	R_1,R_2,I_3,I_4+128,I_5	Set zero-remaining-bits control to 1.
RISBHGZ	R_1,R_2,I_3,I_4,I_5	RISBHG	R_1,R_2,I_3,I_4+128,I_5	Set zero-remaining-bits control to 1.
RISBLGZ	R_1,R_2,I_3,I_4,I_5	RISBLG	R_1,R_2,I_3,I_4+128,I_5	Set zero-remaining-bits control to 1.
RNSBGT	R_1,R_2,I_3,I_4,I_5	RNSBG	R_1,R_2,I_3+128,I_4,I_5	Set test-results control to 1.
ROSBGT	R_1,R_2,I_3,I_4,I_5	ROSBG	R_1,R_2,I_3+128,I_4,I_5	Set test-results control to 1.
RXSBGT	R_1,R_2,I_3,I_4,I_5	RXSBG	R_1,R_2,I_3+128,I_4,I_5	Set test-results control to 1.
SLLHH	R_1,R_2,I_3	RISBHGZ	$R_1,R_2,0,31-I_3,I_3$	Shift Left Logical (HIGH ← HIGH)
SLLHL	R_1,R_2,I_3	RISBHGZ	$R_1,R_2,0,31-I_3,32+I_3$	Shift Left Logical (HIGH ← LOW)
SLLLH	R_1,R_2,I_3	RISBLGZ	$R_1,R_2,0,31-I_3,32+I_3$	Shift Left Logical (LOW ← HIGH)
SRLHH	R_1,R_2,I_3	RISBHGZ	$R_1,R_2,I_3,31,-I_3$	Shift Right Logical (HIGH ← HIGH)
SRLHL	R_1,R_2,I_3	RISBHGZ	$R_1,R_2,I_3,31,32-I_3$	Shift Right Logical (HIGH ← LOW)
SRLLH	R_1,R_2,I_3	RISBLGZ	$R_1,R_2,I_3,31,32-I_3$	Shift Right Logical (LOW ← HIGH)
XHHR	R_1,R_2	RXSBG	$R_1,R_2,0,31$	EXCL. OR HIGH (HIGH ← HIGH)
XHLR	R_1,R_2	RXSBG	$R_1,R_2,0,31,32$	EXCL. OR HIGH (HIGH ← LOW)
XLHR	R_1,R_2	RXSBG	$R_1,R_2,32,63,32$	EXCL. OR HIGH (LOW ← HIGH)

Source: SA22-7832

Extended-Mnemonics for NOT operation

Extended-Mnemonic Syntax		Base-Mnemonic Syntax		Meaning
NOTR	R_1,R_2	NORK	R_1,R_2,R_3	NOT (32)
NOTGR	R_1,R_2	NOGRK	R_1,R_2,R_3	NOT (64)

Source: SA22-7832

Extended-Mnemonics for Vector-Facility Instructions

See *z/Architecture Principles of Operation* (SA22-7832) Chapters 21-24

Summary of Constants

Type	Implied Length, Bytes	Default Alignment	Format	Truncation/ Padding
A	4	Word	Value of address or expression	Left
AD	8	Doubleword	Value of address or expression	Left
B	-	Byte	Binary digits	Left
C	-	Byte	Characters	Right
CA	-	Byte	Characters (ASCII)	Right
CE	-	Byte	Characters (EBCDIC)	Right
CU	Even	Byte	Characters, translated to Unicode	Right
D	8	Doubleword	Long hex floating point	Right
DB	8	Doubleword	Long binary floating point	Right
DD	8	Doubleword	Long decimal floating point	Right
DH	8	Doubleword	Long hex floating point	Right
E	4	Word	Short hex floating point	Right
EB	4	Word	Short binary floating point	Right
ED	4	Word	Short decimal floating point	Right
EH	4	Word	Short hex floating point	Right
F	4	Word	Fixed-point binary	Left
FD	8	Doubleword	Fixed-point binary	Left
G	Even	Byte	Graphic (double-byte) characters	Right
H	2	Halfword	Fixed-point binary	Left
J	4	Word	Symbol naming a DXD, DSECT, or class	Left
JD	8	Doubleword	Symbol naming a DXD, DSECT, or class	Left
L	16	Doubleword	Extended hex floating point	Right
LB	16	Doubleword	Extended binary floating point	Right
LD	16	Doubleword	Extended decimal floating point	Right
LH	16	Doubleword	Extended hex floating point	Right
LQ	16	Quadword	Extended hex floating point	Right
P	-	Byte	Packed decimal	Left
Q	4	Word	Symbol naming a DXD, DSECT, or part	Left
QD	8	Doubleword	Symbol naming a DXD, DSECT, or part	Left
QY	3	Halfword	Symbol naming a DXD, DSECT, or part in long-displacement form	–
R	4	Word	PSECT address value	Left
RD	8	Doubleword	PSECT address value	Left
S	2	Halfword	Address in base-displacement form	–
SY	3	Halfword	Address in base-and-long-displacement form	–
V	4	Word	Externally defined address value	–
VD	8	Doubleword	Externally defined address value	–
X	-	Byte	Hexadecimal digits	Left
Y	2	Halfword	Value of address or expression	Left
Z	-	Byte	Zoned decimal	Left

Source: SC26-4940.

Assigned Storage Locations

Hex Addr	Dec Addr	Addr Type	Function
0-7	0-7	A	IPL PSW†
8-F	8-15	A	CCW-type IPL: IPL CCW1†
10-17	16-23	A	CCW-type IPL: IPL CCW2†
10-13	16-19	A	LD-IPL: Machine-loader execution-space size†
14-17	20-23	A	LD-IPL: System-IPL parameter-list pointer†
80-83	128-131	R	External-interruption parameter
84-85	132-133	R	CPU address associated with external interruption, or zeros
86-87	134-135	R	External-interruption code (see table on page 48)
88-8B	136-139	R	SVC-interruption identification: 0-12 zeros, 13-14 ILC, 15 zero, 16-31 code
8C-8F	140-143	R	Program-interruption identification: 0-12 zeros, 13-14 ILC, 15 zero, 16-31 code (see table on page 48)
90-93	144-147	R	Data-exception code or vector-exception code: 0-23 zeros, 24-31 code (for DXC, see table on page 49; for VXC, see table on page 49)
94-95	148-149	R	Monitor-class number: 0-7 zeros, 8-15 number
96-97	150-151	R	PER code, ATMID, AI (see table on page 51)

Hex Addr	Dec Addr	Addr Type	Function
98-9F	152-159	R	PER address
A0	160	R	Exception access identification: 0-3 zeros, 4-7 access-register number
A1	161	R	PER access identification: 0-3 zeros, 4-7 access-register number
A2	162	R	Operand access identification (if page-translation exception recognized by MOVE PAGE): 0-3 R_1, 4-7 R_2
A3	163	A/R	Store-status/machine-check architectural-mode identification: 0-6 zeros, 7 one
A8-AF	168-175	R	Translation-exception identification (see table on page 50)
B0-B7	176-183	R	Monitor code
B8-BB	184-187	R	Subsystem-identification word: 0-12 zeros, 13-14 SSID,15 one, 16-31 subchannel number
BC-BF	188-191	R	I/O-interruption parameter
C0-C3	192-195	R	I/O-interruption-identification word: 0-1 zeros, 2-4 I/O-interruption subclass, 5-31 zeros
C8-CB	200-203	R	STFL facility list (see “Facility Indications” on page 52)
E8-EF	232-239	R	Machine-check-interruption code (see diagram on page 51)
F4-F7	244-247	R	External-damage code (see diagram on page 52)
F8-FF	248-255	R	Failing-storage address
100-107	256-263	R	Enhanced-Monitor Counter-Array Origin
108-10B	264-267	R	Enhanced-Monitor Counter-Array Size
10C-10F	268-271	R	Enhanced-Monitor Exception Count
110-117	272-279	R	Breaking-event address
120-12F	288-303	R	Restart old PSW
130-13F	304-319	R	External old PSW
140-14F	320-335	R	Supervisor-call old PSW
150-15F	336-351	R	Program old PSW
160-16F	352-367	R	Machine-check old PSW
170-17F	368-383	R	Input/output old PSW
1A0-1AF	416-431	R	Restart new PSW
1B0-1BF	432-447	R	External new PSW
1C0-1CF	448-463	R	Supervisor-call new PSW
1D0-1DF	464-479	R	Program new PSW
1E0-1EF	480-495	R	Machine-check new PSW
1F0-1FF	496-511	R	Input/output new PSW
11B0-11B7	4528-4535	R	Machine-check-extended-save-area address
11C0-11FF	4544-4607	R	Available for programming
1200-127F	4608-4735	A/R	Store-status/machine-check floating-point-register save area
1280-12FF	4736-4863	A/R	Store-status/machine-check general-register save area
1300-130F	4864-4879	A/R	Store-status PSW save area or machine-check fixed-logout area‡
1318-131B	4888-4891	A	Store-status prefix save area
131C-131F	4892-4895	A/R	Store-status/machine-check floating-point-control-register save area
1324-1327	4900-4903	A/R	Store-status/machine-check TOD-programmable-register save area
1328-132F	4904-4911	A/R	Store-status/machine-check CPU-timer save area
1331-1337	4913-4919	A/R	Store-status/machine-check clock-comparator bits 0-55 save area (zeros at 4912)
1338-133F	4920-4927	A/R	Store-status/machine-check Breaking-Event-Address-Register save area
1340-137F	4928-4991	A/R	Store-status/machine-check access-register save area
1380-13FF	4992-5119	A/R	Store-status/machine-check control-register save area
1500-1507	5376-5383	R	Cryptography-Counter Designation
1508-150F	5384-5391	R	Activity-Instrumentation-Control Designation
1800-18FF	6144-6399	R	Program-interruption TDB (see diagram on page 75)

A Absolute address.
R Real address.
A/R A if store status; R if machine check.
† When the configuration-z/Architecture-architectural-mode (CZAM) facility is installed.
‡ Contents may vary among models; see System Library manuals.

External-Interruption Codes

At real-storage locations 134-135 (86-87 hex)

Code (Hex)	Condition
0040	Interrupt key
1004	Clock comparator
1005	CPU timer
1007	Warning-track interruption
1200	Malfunction alert
1201	Emergency signal
1202	External call
1406	Timing alert
1407	Measurement alert
2401	Service signal

Program-Interruption Codes

At real-storage locations 142-143 (8E-8F hex)

Code (Hex)	Condition	ILC Set	Instr. Ending
0001	Operation exception	1 2 3	S
0002	Privileged-operation exception	2 3	S
0003	Execute exception	2 3	S
0004	Protection exception	1 2 3	S T
0005	Addressing exception	1 2 3	S T
0006	Specification exception	0 1 2 3	C S
0007	Data exception	1 2 3	C S T
0008	Fixed-point-overflow exception	1 2 3	C
0009	Fixed-point-divide exception	1 2 3	C S
000A	Decimal-overflow exception	2 3	C
000B	Decimal-divide exception	2 3	S
000C	HFP-exponent-overflow exception	1 2 3	C
000D	HFP-exponent-underflow exception	1 2 3	C
000E	HFP-significance exception	1 2	C
000F	HFP-floating-point-divide exception	1 2	S
0010	Segment-translation exception	1 2 3	N
0011	Page-translation exception	1 2 3	N
0012	Translation-specification exception	1 2 3	S
0013	Special-operation exception	1 2 3	S
0015	Operand exception	2	S
0016	Trace-table exception	1 2	N
0018	Transaction constraint	1 2 3	S
001B	Vector-processing	3	S
001C	Space-switch event	0 1 2	C
001D	HFP-square-root exception	2	S
001F	PC-translation-specification exception	2	S
0020	AFX-translation exception	1 2	N
0021	ASX-translation exception	1 2	N
0022	LX-translation exception	2	N
0023	EX-translation exception	2	N
0024	Primary-authority exception	2	N
0025	Secondary-authority exception	1 2	N
0026	LFX-translation exception	2	N
0027	LSX-translation exception	2	N
0028	ALET-specification exception	1 2 3	S
0029	ALEN-translation exception	1 2 3	N
002A	ALE-sequence exception	1 2 3	N
002B	ASTE-validity exception	1 2 3	N
002C	ASTE-sequence exception	1 2 3	N
002D	Extended-authority exception	1 2 3	N
002E	LSTE sequence	2	N
002F	ASTE instance	1 2 3	N
0030	Stack-full exception	2	N
0031	Stack-empty exception	1 2	N

Code (Hex)	Condition	ILC Set	Instr. Ending
0032	Stack-specification exception	1 2	N
0033	Stack-type exception	1 2	N
0034	Stack-operation exception	1 2	N
0038	ASCE-type exception	1 2 3	N
0039	Region-first-translation exception	1 2 3	N
003A	Region-second-translation exception	1 2 3	N
003B	Region-third-translation exception	1 2 3	N
003D	Secure-storage-access exception	1 2 3	N
0040	Monitor event	2	C
0080	PER basic event (code may be combined with another code)	0 1 2 3	C
0080	PER nullification event	0	N
0119	Crypto-operation exception	2	N
0200	Transactional-execution-aborted event	1 2 3	C

C Completed
ILC Instruction-length code
N Nullified
S Suppressed
T Terminated

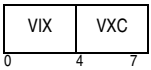
Data-Exception Code (DXC)

At real-storage location 147 (93 hex) and in byte 2 of floating-point-control register

Code (Hex)	Data Exception
00	General operand
01	AFP register
02	BFP instruction
03	DFP instruction
04	Quantum exception
07	Simulated quantum exception
08	IEEE inexact and truncated
0B	IXS inexact
0C	IEEE inexact and incremented
10	IEEE underflow, exact
13	IXS underflow, exact
18	IEEE underflow, inexact and truncated
1B	IXS underflow, inexact
1C	IEEE underflow, inexact and incremented
20	IEEE overflow, exact
23	IXS overflow, exact
28	IEEE overflow, inexact and truncated
2B	IXS overflow, inexact
2C	IEEE overflow, inexact and incremented
40	IEEE division by zero
43	IXS division by zero
80	IEEE invalid operation
83	IXS invalid operation
FE	Vector instruction
FF	Compare-and-trap or load-and-trap instruction

Vector-Exception Code (VXC)

At real-storage location 147 (93 hex) and in byte 2 of floating-point-control register



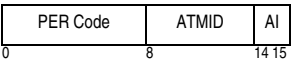
Vector Index (VIX)
Index to the leftmost element that recognized the exception

Vector Interrupt Code (VIC)

Value	Meaning
0001	IEEE invalid operation
0010	IEEE division by zero
0011	IEEE overflow
0100	IEEE underflow
0101	IEEE inexact

PER Code, ATMID, and AI

At real-storage locations 150-151



Program-Event-Recording (PER Code)

Bit	Meaning
0	Successful-branching event
1	Instruction-fetching event
2	Storage-alteration event
3	Storage-key-alteration event
4	Store-using-real-address event
5	Zero-address-detection event
6	Transaction-end event
7	Instruction-fetch-nullification event

Addressing and-Translation-Mode ID (ATMID)

Bit	Meaning
8	PSW bit 31
9	ATMID-validity bit
10	PSW bit 32
11	PSW bit 5
12-13	PSW bits 16-17

PER ASCE Identification (AI)

14-15	0 - primary; 1 - AR-specified; 2 - secondary; 3 - home

Translation-Exception Identification

At real-storage locations 168-175 (A8-AF hex)

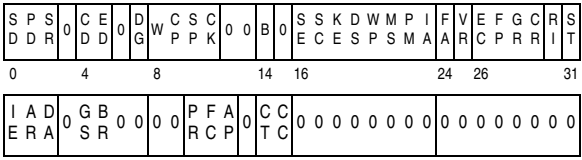
Interruption Code (Hex)	Exception or Event	Format of Information Stored*
0004	Protection	If 61 zero: rest unpredictable, If 61 one: suppression, 0-51 address; 52-53 access-exception fetch/store indication; bits 56, 60, and 61 form a 3-bit protection code, 62-63 ASCE identification, rest unpredictable, location 160 valid; if DAT was off, rest unpredictable
0010	Segment translation	0-51 address; 52-53 access-exception fetch/store indication; 54-61 unpredictable, 62-63 ASCE identification
0011	Page translation	0-51 address; 52-53 access-exception fetch/store indication; 54-60 unpredictable, if 61, zero, not MOVE PAGE; if 61 one, MOVE PAGE (see location 162); 62-63 ASCE identification
001C	Space switch	From primary-space mode: 32 old primary-space-switch-event control, 33-47 zeros, 48-63 old PASN From home-space mode: 32 home-space-switch-event control, 33-63 zeros
0020	AFX translation	32-47 zeros, 48-63 address-space number
0021	ASX translation	32-47 zeros, 48-63 address-space number
0022	LX translation	32-43 zeros, 44-63 program-call number
0023	EX translation	32-43 zeros, 44-63 program-call number
0024	Primary authority	32-47 zeros, 48-63 address-space number
0025	Secondary authority	32-47 zeros, 48-63 address-space number
0026 0027	LFX translation LSX translation	When bit 44 is 0: 32-43 zeros, 44-63 program-call number. When bit 44 is 1, 32-63 program-call number
0038	ASCE type	0-51 address; 52-53 access-exception fetch/store indication; 54-61 unpredictable, 62-63 ASCE identification
0039	Region-first translation	0-51 address; 52-53 access-exception fetch/store indication; 54-61 unpredictable, 62-63 ASCE identification
003A	Region-second translation	0-51 address; 52-53 access-exception fetch/store indication; 54-61 unpredictable, 62-63 ASCE identification
003B	Region-third translation	0-51 address; 52-53 access exception fetch/store indication; 54-61 unpredictable, 62-63 ASCE identification

Interruption Code (Hex)	Exception or Event	Format of Information Stored*
003D	Secure-storage access	If 61 zero: rest unpredictable, If 61 one: 0-51 address; 52-53 access-exception fetch/store indication; 54-60 unpredictable; 62-63 ASCE identification

* Bits 0-31 (bytes 168-171) unchanged if not described.

Machine-Check Interruption Code

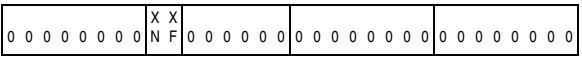
At real-storage locations 232-239 (E8-EF hex)



Bit	Meaning
0	(SD) System damage
1	(PD) Instruction-processing damage
2	(SR) System recovery
4	(CD) Timing-facility damage
5	(ED) External damage
7	(DG) Degradation
8	(W) Warning
9	(CP) Channel report pending
10	(SP) Service-processor damage
11	(CK) Channel-subsystem damage
14	(B) Backed up
16	(SE) Storage error uncorrected
17	(SC) Storage error corrected
18	(KE) Storage-key error uncorrected
19	(DS) Storage degradation
20	(WP) PSW-MWP validity
21	(MS) PSW mask and key validity
22	(PM) PSW program-mask and condition-code validity
23	(IA) PSW-instruction-address validity
24	(FA) Failing-storage-address validity
25	(VR) Vector-register validity
26	(EC) External-damage-code validity
27	(FP) Floating-point-register validity
28	(GR) General-register validity
29	(CR) Control-register validity
30	(RI) Reserved for IBM use
31	(ST) Storage logical validity
32	(IE) Indirect storage error
33	(AR) Access-register validity
34	(DA) Delayed-access exception
36	(GS) Guarded-storage-registers validity
37	(BR) Breaking-event-address-register validity
42	(PR) TOD-programmable-register validity
43	(FC) Floating-point-control-register validity
44	(AP) Ancillary report
46	(CT) CPU-timer validity
47	(CC) Clock-comparator validity

External-Damage Code

At real-storage address 244-247 (F4-F7 hex)



Bit	Meaning
8	(XN) Expanded storage not operational
9	(XF) Expanded-storage control failure

Facility Indications

The first 32 facility indications are stored at real-storage locations 200-203 (C8-CB hex) by STFL; the specified number of doublewords of facility indications are stored at second-operand location by STFLE.

Bit	Meaning when Bit is One
0	The instructions marked "N3" in the instruction-summary figures in Chapters 7 and 10 are installed.
1	The z/Architecture architectural mode is installed.
2	The z/Architecture architectural mode is active. When bits 2 and 168 are both zero, the ESA/390 architectural mode is active; when bit 2 is zero and bit 168 is one, the ESA/390 compatibility mode is active.
3	The DAT-enhancement facility is installed in the z/Architecture architectural mode. The DAT-enhancement facility includes the INVALIDATE DAT TABLE ENTRY (IDTE) and COMPARE AND SWAP AND PURGE (CSPG) instructions.
4	INVALIDATE DAT TABLE ENTRY (IDTE) performs the invalidation-and-clearing operation by selectively clearing TLB segment-table entries when a segment-table entry or entries are invalidated. IDTE also performs the clearing-by-ASCE operation. Unless bit 4 is one, IDTE simply purges all TLBs. Bit 3 is one if bit 4 is one.
5	INVALIDATE DAT TABLE ENTRY (IDTE) performs the invalidation-and-clearing operation by selectively clearing TLB region-table entries when a region-table entry or entries are invalidated. Bits 3 and 4 are ones if bit 5 is one.
6	The ASN-and-LX reuse facility is installed in the z/Architecture architectural mode.
7	The store-facility-list-extended facility is installed.
8	The enhanced-DAT facility 1 is installed in the z/Architecture architectural mode.
9	The sense-running-status facility is installed in the z/Architecture architectural mode.
10	The conditional-SSKE facility is installed in the z/Architecture architectural mode.
11	The configuration-topology facility is installed in the z/Architecture architectural mode.
13	The IPTE-range facility is installed in the z/Architecture architectural mode.
14	The nonquiescing key-setting facility is installed in the z/Architecture architectural mode.
16	The extended-translation facility 2 is installed.
17	The message-security assist is installed.
18	The long-displacement facility is installed in the z/Architecture architectural mode.
19	The long-displacement facility has high performance. Bit 18 is one if bit 19 is one.
20	The HFP-multiply-add/subtract facility is installed.
21	The extended-immediate facility is installed in the z/Architecture architectural mode.
22	The extended-translation facility 3 is installed in the z/Architecture architectural mode.
23	The HFP-unnormalized-extension facility is installed in the z/Architecture architectural mode.
24	The ETF2-enhancement facility is installed.
25	The store-clock-fast facility is installed in the z/Architecture architectural mode.
26	The parsing-enhancement facility is installed in the z/Architecture architectural mode.
27	The move-with-optional-specifications facility is installed in the z/Architecture architectural mode.
28	The TOD-clock-steering facility is installed in the z/Architecture architectural mode.
30	The ETF3-enhancement facility is installed in the z/Architecture architectural mode.
31	The extract-CPU-time facility is installed in the z/Architecture architectural mode.
32	The compare-and-swap-and-store facility is installed in the z/Architecture architectural mode.
33	The compare-and-swap-and-store facility 2 is installed in the z/Architecture architectural mode.
34	The general-instructions-extension facility is installed in the z/Architecture architectural mode.
35	The execute-extensions facility is installed in the z/Architecture architectural mode.
36	The enhanced-monitor facility is installed in the z/Architecture architectural mode.
37	The floating-point extension facility is installed in the z/Architecture architectural mode.

Bit	Meaning when Bit is One
39	Assigned to IBM internal use.
40	The set-program-parameters facility is installed in the z/Architecture architectural mode.
41	The floating-point-support-enhancement facilities (FPR-GR-loading, FPS-sign-handling, and DFP-rounding) are installed in the z/Architecture architectural mode.
42	The DFP (decimal-floating-point) facility is installed in the z/Architecture architectural mode.
43	The DFP (decimal-floating-point) facility has high performance. Bit 42 is one if bit 43 is one.
44	The PFPO instruction is installed in the z/Architecture architectural mode.
45	The distinct-operands, fast-BCR-serialization, high-word, and population-count facilities, the interlocked-access facility 1, and the load/store-on-condition facility 1 are installed in the z/Architecture architectural mode.
47	The CMPSC-enhancement facility is installed in the z/Architecture architectural mode.
48	The decimal-floating-point zoned-conversion facility is installed in the z/Architecture architectural mode.
49	The execution-hint, load-and-trap, and processor-assist facilities and the miscellaneous-instruction-extensions facility 1, are installed in the z/Architecture architectural mode.
50	The constrained transactional-execution facility is installed in the z/Architecture architectural mode. This bit is meaningful only when bit 73 is one.
51	The local-TLB-clearing facility is installed in the z/Architecture architectural mode.
52	The interlocked-access facility 2 is installed.
53	The load/store-on-condition facility 2 and load-and-zero-rightmost-byte facility are installed in the z/Architecture architectural mode.
57	The message-security-assist-extension 5 is installed in the z/Architecture architectural mode.
58	The miscellaneous-instruction-extensions facility 2 is installed in the z/Architecture architectural mode.
61	The miscellaneous-instruction-extensions facility 3 is installed in the z/Architecture architectural mode.
66	The reset-reference-bits-multiple facility is installed in the z/Architecture architectural mode.
67	The CPU-measurement counter facility is installed in the z/Architecture architectural mode.
68	The CPU-measurement sampling facility is installed in the z/Architecture architectural mode.
73	The transactional-execution facility is installed in the z/Architecture architectural mode. Bit 49 is one when bit 73 is one.
74	The store-hypervisor-information facility is installed in the z/Architecture architectural mode (see *z/VM CP Programming Services* [SC24-6179]).
75	The access-exception-fetch/store-indication facility is installed in the z/Architecture architectural mode.
76	The message-security-assist-extension 3 is installed in the z/Architecture architectural mode.
77	The message-security-assist-extension 4 is installed in the z/Architecture architectural mode.
78	The enhanced-DAT facility 2 is installed in the z/Architecture architectural mode.
80	The decimal-floating-point packed-conversion facility is installed in the z/Architecture architectural mode.
81	The PPA-in-order facility is installed in the z/Architecture architectural mode.
129	The vector facility for z/Architecture is installed in the z/Architecture architectural mode.
130	The instruction-execution-protection facility is installed in the z/Architecture architectural mode.
131	The side-effect-access facility and the enhanced-suppression-on-protection facility 2 are installed in the z/Architecture architectural mode.
133	The guarded-storage facility is installed in the z/Architecture architectural mode.
134	The vector-packed-decimal facility is installed in the z/Architecture architectural mode.
135	The vector-enhancements facility 1 is installed in the z/Architecture architectural mode.
138	The configuration z/Architecture architectural mode facility is installed.
139	The multiple-epoch facility is installed in the z/Architecture architectural mode.
142	The store-CPU-counter-multiple facility is installed.
144	The test-pending-external-interruption facility is installed in the z/Architecture architectural mode.
145	The insert-reference-bits-multiple facility is installed in the z/Architecture architectural mode.
146	The message-security-assist-extension 8 is installed in the z/Architecture architectural mode.
148	The vector-enhancements facility 2 is installed in the z/Architecture architectural mode.
149	The move-page-and-set-key facility is installed in the z/Architecture architectural mode.
150	The enhanced-sort facility is installed in the z/Architecture architectural mode.

Bit	Meaning when Bit is One
151	The DEFLATE-conversion facility is installed in the z/Architecture architectural mode.
152	The vector-packed-decimal-enhancement facility is installed in the z/Architecture architectural mode.
155	The message-security-assist-extension 9 is installed in the z/Architecture architectural mode.
158	The ultravisor-call facility is installed in the z/Architecture architectural mode.
161	The secure-execution-unpack facility is installed in the z/Architecture architectural mode.
165	The neural-network-processing-assist facility is installed in the z/Architecture architectural mode.
168	The ESA/390-compatibility-mode facility is installed in the configuration.
169	The storage-key-removal facility is installed in the z/Architecture architectural mode.
192	The vector-packed-decimal-enhancement facility 2 is installed in the z/Architecture architectural mode.
193	The BEAR-enhancement facility is installed in the z/Architecture architectural mode.
194	The reset-DAT-protection facility is installed in the z/Architecture architectural mode.
196	The processor-activity-instrumentation facility is installed in the z/Architecture architectural mode.
197	The processor-activity-instrumentation extension 1 is installed in the z/Architecture architectural mode.

Control Registers

CR	Bits	Name of Field	Associated with	Init*
0	8	Transactional-execution control	Transactional-execution	0
	9	Program-interruption filtering override	Transactional-execution	0
	10	Clock-comparator sign control	TOD clock	0
	13	Cryptography counter control	processor-activity instrumentation	0
	14	processor-activity instrumentation-extension control	processor-activity instrumentation	0
	30	Warning-track-interruption enablement	Virtual machines	0
	32	Trace TOD-clock control	TOD clock	0
	33	SSM-suppression control	SSM instruction	0
	34	TOD-clock-sync control	TOD clock	0
	35	Low-address-protection control	Low-address protection	0
	36	Extraction-authority control	Instruction authorization	0
	37	Secondary-space control	Instruction authorization	0
	38	Fetch-protection-override control	Key-controlled protection	0
	39	Storage-protection-override control	Key-controlled protection	0
	40	Enhanced-DAT-enablement control	Dynamic address translation	0
	43	Instruction-execution-protection-enablement control	Instruction-execution protection	0
	44	ASN-and-LX-reuse control	Instruction authorization	0
	45	AFP-register control	Floating point	0
	46	Vector enablement control	Vector facility for z/Architecture	0
	48	Malfunction-alert subclass mask	External interruptions	0
	49	Emergency-signal subclass mask	External interruptions	0
	50	External-call subclass mask	External interruptions	0
	52	Clock-comparator subclass mask	External interruptions	0
	53	CPU-timer subclass mask	External interruptions	0
	54	Service-signal subclass mask	External interruptions	0
	56	Unused (See note)		1
	57	Interrupt-key subclass mask	External interruptions	1
	58	Unused (See note)		1
	59	ETR subclass mask	External interruptions	0
	61	Crypto control	Cryptography	0

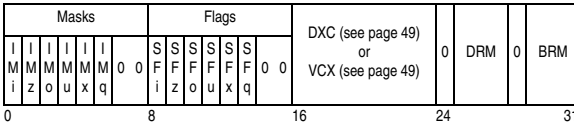
CR	Bits	Name of Field	Associated with	Init*
1	0-63	Primary address-space-control element	Dynamic address translation	0
	0-51	Primary region-table or segment-table origin or real-space token origin	Dynamic address translation	0
	54	Primary subspace-group control	Subspace groups	0
	55	Primary private-space control	Dynamic address translation	0
	56	Primary storage-alteration-event control	Program-event recording	0
	57	Primary space-switch-event control	Program interruptions	0
	58	Primary real-space control	Dynamic address translation	0
	60-61	Primary designation-type control	Dynamic address translation	0
	62-63	Primary table length	Dynamic address translation	0
2	0-8	Reserved for IBM use	System Controls	0
	33-57	Dispatchable-unit-control-table origin	Access-register translation	0
	59	Guarded-storage facility enablement	Guarded-storage facility	0
	61	Transaction diagnostic scope	Transactional execution	0
	62-63	Transaction diagnostic control	Transactional execution	0
3	0-31	Secondary ASTE Instance Number	Instruction authorization	0
	32-47	PSW-key mask	Instruction authorization	0
	48-63	Secondary ASN	Address spaces	0
4	0-31	Primary ASTE Instance Number	Instruction authorization	0
	32-47	Authorization index	Instruction authorization	0
	48-63	Primary ASN	Address spaces	0
5	33-57	Primary-ASTE origin	Access-register translation	0
6	32-39	I/O-interruption subclass mask	I/O interruptions	0
7	0-63	Secondary address-space-control element	Dynamic address translation	0
	0-51	Secondary region-table or segment-table origin or real-space token origin	Dynamic address translation	0
	54	Secondary subspace-group control	Subspace groups	0
	55	Secondary private-space control	Dynamic address translation	0
	56	Secondary storage-alteration-event control	Program-event recording	0
	58	Secondary real-space control	Dynamic address translation	0
	60-61	Secondary designation-type control	Dynamic address translation	0
	62-63	Secondary table length	Dynamic address translation	0
8	16-31	Enhanced-monitor masks	MONITOR CALL instruction	0
	32-47	Extended authorization index	Access-register translation	0
	48-63	Monitor masks	MONITOR CALL instruction	0
9	32	Successful-branching-event mask	Program-event recording	0
	33	Instruction-fetching-event mask	Program-event recording	0
	34	Storage-alteration-event mask	Program-event recording	0
	35	Storage-key-alteration-event mask	Program-event recording	0
	36	Store-using-real-address-event mask	Program-event recording	0
	37	Zero-address-detection-event mask	Program-event recording	0
	38	Transaction-end-event mask	Program-event recording	0
	39	Instruction-fetching-nullification-event mask	Program-event recording	0
	40	Branch-address control	Program-event recording	0
	41	Event-suppression control	Program-event recording	0
	42	Storage-alteration-space control	Program-event recording	0
10	0-63	PER starting address	Program-event recording	0
11	0-63	PER ending address	Program-event recording	0
12	0	Branch-trace control	Tracing	0
	1	Mode-trace control	Tracing	0
	2-61	Trace-entry address	Tracing	0
	62	ASN-trace control	Tracing	0
	63	Explicit-trace control	Tracing	0

CR	Bits	Name of Field	Associated with	Init*
13	0-63	Home address-space-control element	Dynamic address translation	0
	0-51	Home region-table or segment-table origin or real-space token origin	Dynamic address translation	0
	54	Home subspace-group control	Subspace groups	0
	55	Home private-space control	Dynamic address translation	0
	56	Home storage-alteration-event control	Program-event recording	0
	57	Home space-switch-event control	Program interruptions	0
	58	Home real-space control	Dynamic address translation	0
	60-61	Home designation-type control	Dynamic address translation	0
	62-63	Home table length	Dynamic address translation	0
14	32	Unused (See note)		1
	33	Unused (See note)		1
	35	Channel-report-pending subclass mask	I/O machine-check handling	0
	36	Recovery subclass mask	Machine-check handling	0
	37	Degradation subclass mask	Machine-check handling	0
	38	External-damage subclass mask	Machine-check handling	1
	39	Warning subclass mask	Machine-check handling	0
	42	TOD-clock-control-override control	TOD clock	0
	44	ASN-translation control	Instruction authorization	0
	45-63	ASN-first-table origin	ASN translation	0
15	0-60	Linkage-stack-entry address	Linkage-stack operations	0

* Value after initial CPU reset.

Note: This bit is not used but is initialized to one for consistency with the System/370 definition.

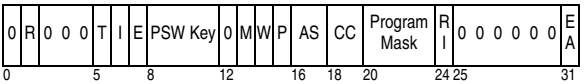
Floating-Point-Control (FPC) Register

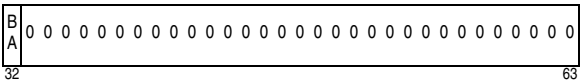


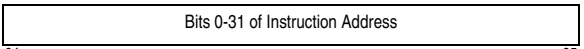
Bit	Meaning
0	(IMi) IEEE-invalid-operation mask
1	(IMz) IEEE-division-by-zero mask
2	(IMo) IEEE-overflow mask
3	(IMu) IEEE-underflow mask
4	(IMx) IEEE-inexact mask
5	(IMq) Quantum-exception mask
8	(SFi) IEEE-invalid-operation flag
9	(SFz) IEEE-division-by-zero flag
10	(SFo) IEEE-overflow flag
11	(SFu) IEEE-underflow flag
12	(SFx) IEEE-inexact flag
13	(SFq) Quantum-exception flag
16-23	(DXC) Data-exception code (see table on page 49)
25-27	(DRM) DFP Rounding mode 000 Round to nearest with ties to even 001 Round toward 0 010 Round toward +∞ 011 Round toward -∞ 100 Round to nearest with ties away from 0 101 Round to nearest with ties toward 0 110 Round away from 0 111 Round to prepare for shorter precision
29-31	(BRM) BFP Rounding mode 000 Round to nearest 001 Round toward 0 010 Round toward +∞ 011 Round toward -∞ 111 Round to prepare for shorter precision

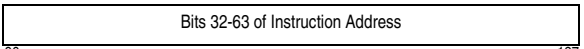
Program-Status Word (PSW)

z/Architecture PSW



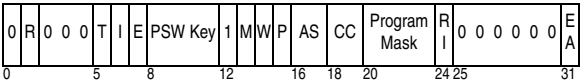


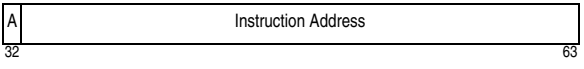




Bit	Meaning
1	(R) Program-event-recording mask
5	(T = 1) DAT mode
6	(I) Input/output mask
7	(E) External mask
12	Zero indicates a 16-byte PSW
13	(M) Machine-check mask
14	(W = 1) Wait state
15	(P = 1) Problem state
16-17	xx Real mode (T = 0) 00 – Primary-space mode (T = 1) 01 – Access-register mode (T = 1) 10 – Secondary-space mode (T = 1) 11 – Home-space mode (T = 1)
18-19	(CC) Condition code
20	Fixed-point-overflow mask
21	Decimal-overflow mask
22	HFP-exponent-underflow mask
23	HFP-significance mask
24	Reserved for IBM use
31/32	Extended/basic addressing mode 00 – 24-bit mode 01 – 31-bit mode 10 – Invalid 11 – 64-bit mode

Short-Format PSW

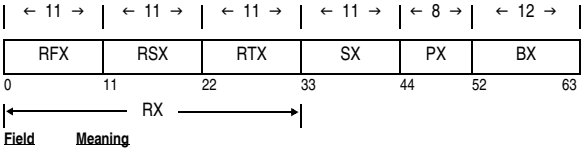




Bit	Meaning
12	One indicates a short-format PSW

Dynamic Address Translation

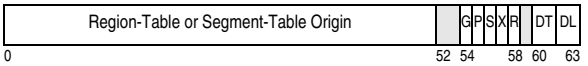
Virtual-Address Format



Field	Meaning
RX	Region index (region = 2G bytes)
RFX	Region first index
RSX	Region second index
RTX	Region third index
SX	Segment index (segment = 1M bytes)
PX	Page index (page = 4K bytes)
BX	Byte index

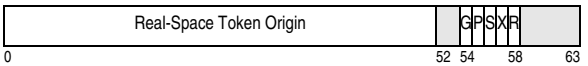
Address-Space-Control Element (ASCE)

Region-Table or Segment-Table Designation (RTD or STD)



Bit	Meaning
54	(G) Subspace-group control
55	(P) Private-space control
56	(S) Storage-alteration-event control
57	(X) Space-switch-event control
58	(R) Real-space control (R = 0)
60-61	(DT) Designation-type control 11 Region-first-table 10 Region-second-table 01 Region-third-table 00 Segment-table
62-63	(DL) Designation length (× 4K bytes)

Real-Space Designation (RSD)



Bit	Meaning
58	(R) Real-space control (R = 1)

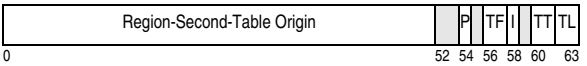
Note: Other bits are as in RTD or STD.

Table Values

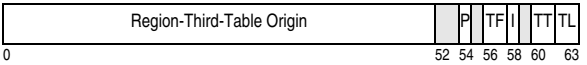
Table	Increment	Incr. Size	Incr. Entries	Max. Size	Max. Entries	Max Table Maps	
						Regions	Bytes
Region First	1-4	4KB	512	16KB	2K	8G	$16E = 16 \times 2^{60}$
Region Second	1-4	4KB	512	16KB	2K	4M	$8P = 8 \times 2^{50}$
Region Third	1-4	4KB	512	16KB	2K	2K	$4T = 4 \times 2^{40}$
Segment	1-4	4KB	512	16KB	2K	1	$2G = 2 \times 2^{30}$
Page	1	2KB	256	2KB	256	—	$1M = 2^{20}$

Region-Table Entry (RTE)

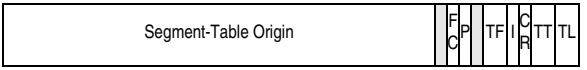
Region-First-Table Entry (RFTE)



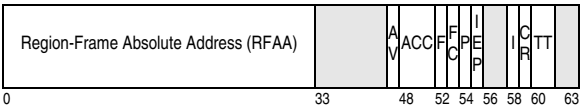
Region-Second-Table Entry (RSTE)



Region-Third-Table Entry (RTTE, FC=0)

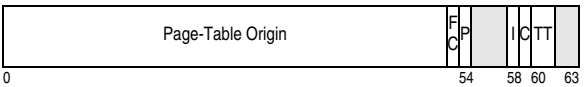


Region-Third-Table Entry (RTTE, FC-1)

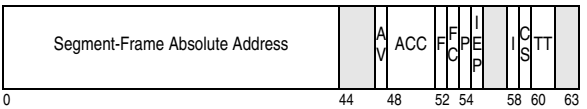


Bit	Meaning
47	(AV) Access-control (ACC) and fetch-protection (F) validity bit
48-51	(ACC) Access-control bits
52	(F) Fetch-protection bit
53	(FC) Format control
54	(P) DAT protection bit
55	(IEP) Instruction-execution-protection bit (format-1 RTTE only)
56-57	(TF) Table offset (for next-lower-level table)
58	(I) Invalid bit (for set of regions in RFTE or RSTE, or for region in RTTE
59	(CR) Common-region bit
60-61	(TT) Table-type bits (for this table) 11=Region first table 10=Region second table 01=Region third table
62-63	(TL) Table length (for next-lower-level table) (× 4K bytes)

Segment-Table Entry (STE, FC=0)



Segment-Table Entry (STE, FC=1)



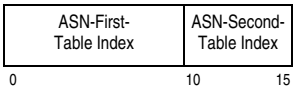
Bit	Meaning
47	(AV) Access-control (ACC) and fetch-protection (F) validity bit
48-51	(ACC) Access-control bits
52	(F) Fetch-protection bit
53	(FC) Format control
54	(P) DAT-protection bit
55	(IEP) Instruction-execution-protection bit (format-1 STE only)
58	(I) Segment-invalid bit
59	(CS) Common-segment bit
60-61	(TT) Table-type bits (for this table): 00=Segment table

Page-Table Entry (PTE)

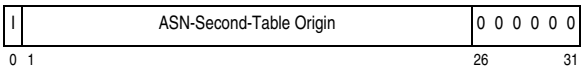
Bit	Meaning
53	(I) Page-invalid bit
54	(P) DAT-protection bit
55	(IEP) Instruction-execution-protection bit (format-1 STE only)

ASN Translation

Address-Space Number (ASN)

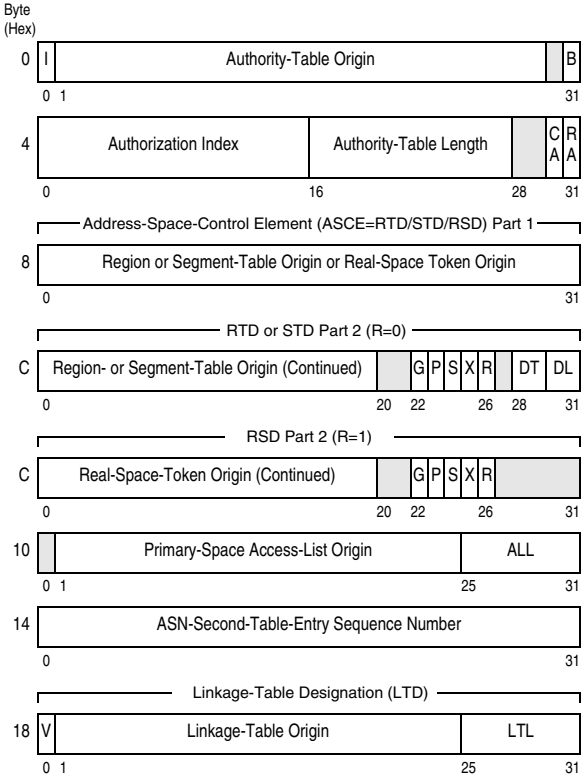


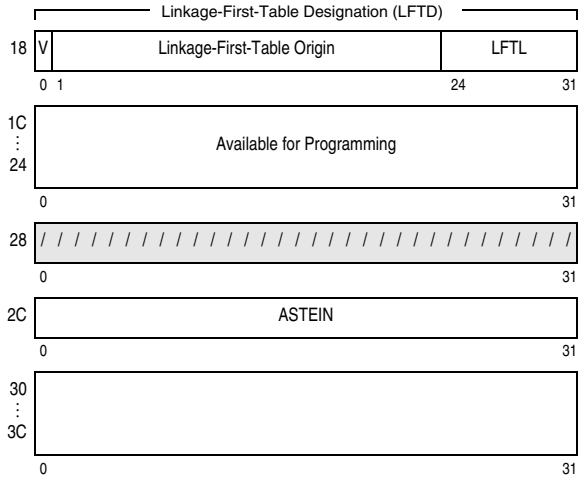
ASN-First-Table Entry



Bit	Meaning
0	(I) AFX-invalid bit

ASN-Second-Table Entry (ASTE)

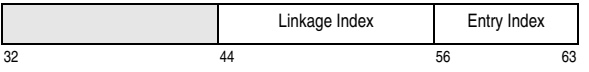




Byte.Bit	Meaning
0.0	(I) ASX-invalid bit
0.31	(B) Base-space bit
4.30	(CA) Controlled-ASN bit
4.31	(RA) Reusable-ASN bit
10.25-31	(ALL) Access-list length (× 128 bytes)
18.0	(V) Subsystem-linkage control
18.25-31	(LTL) Linkage-table length (× 128 bytes)
18.24-31	(LFTL) Linkage-first-table length (× 256 bytes)

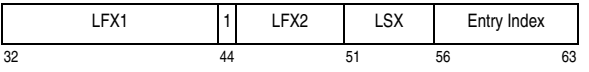
PC-Number Translation

Program-Call Number (20-Bit)

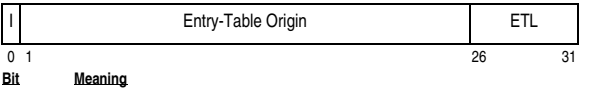


Program-Call Number (32-Bit, Bit 44=0)

Program-Call Number (32-Bit, Bit 44=1)

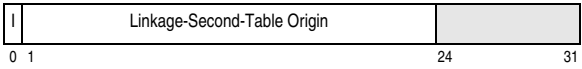


Linkage-Table Entry (LTE)



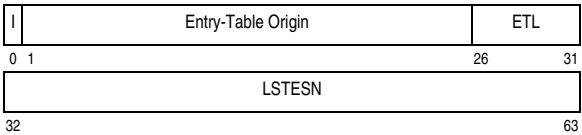
Bit	Meaning
0	(I) LX-invalid bit
26-31	(ETL) Entry-table length (× 128 bytes)

Linkage-First-Table Entry (LFTE)



Bit	Meaning
0	(I) LFX-invalid bit

Linkage-Second-Table Entry (LSTE)

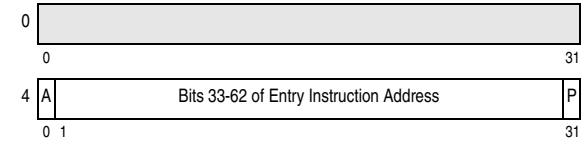


Bit	Meaning
0	(I) LSX-invalid bit
26-31	(ETL) Entry-table length (× 128 bytes)

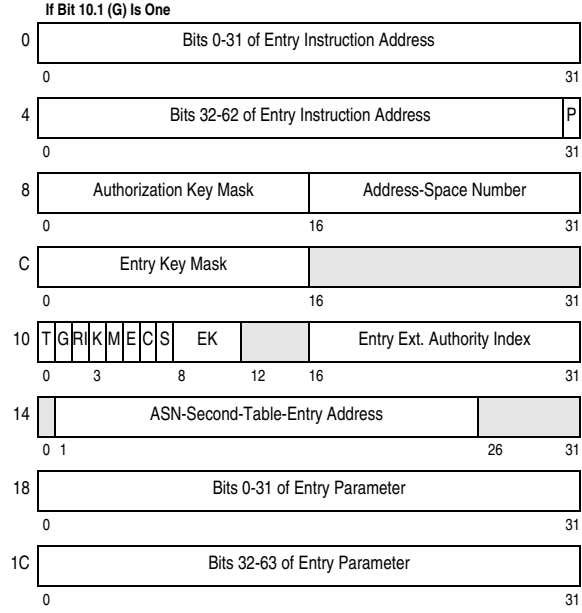
Entry-Table Entry (ETE)

Byte
(Hex)

If Bit 10.1 (G) Is Zero



If Bit 10.1 (G) Is One

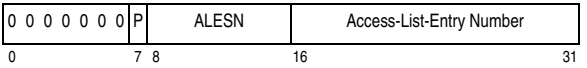


Byte.Bit	Meaning
4.0	(A) Entry addressing mode
4.31	(P) Entry problem state
10.0	(T) PC-type bit (zero: basic; one: stacking)
10.1	(G) Entry extended addressing mode

10.2	(RI) Reserved for IBM use
10.3	(K) PSW-key control (zero: unchanged; one: replace if stacking
10.4	(M) PSW-key-mask control (zero: Or; one: replace if stacking)
10.5	(E) EAX control (zero: unchanged; one: replace if stacking)
10.6	(C) Address-space-control control
10.7	(S) Secondary-ASN control
10.8-11	(EK) Entry key

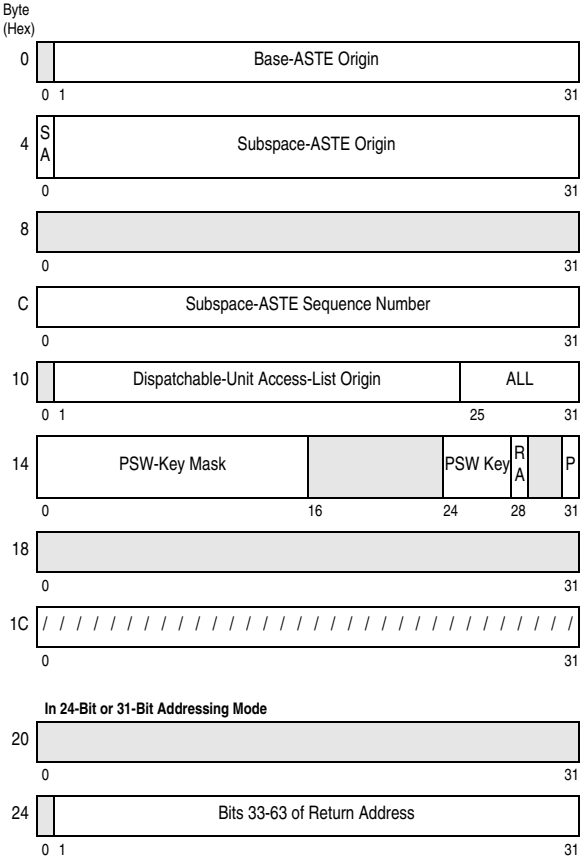
Access-Register Translation

Access-List-Entry Token (ALET)

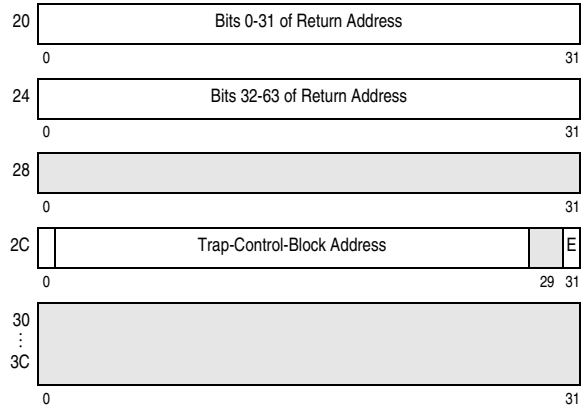


Bit	Meaning
7	(P) Primary-list bit (zero: use DUCT; one: use primary ASTE)
8-15	(ALESN) Access-list-entry sequence number

Dispatchable-Unit-Control Table (DUCT)

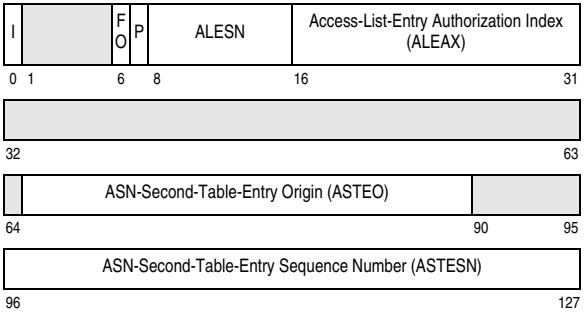


In 64-Bit Addressing Mode



Byte.Bit	Meaning
4.0	(SA) Subspace-active bit
10.25-31	(ALL) Access-list length (× 128 bytes)
14.28	(RA) Reduced-authority bit
14.31	(P) Problem-state bit
2C.31	(E) TRAP-enabled bit
///	Available for programming

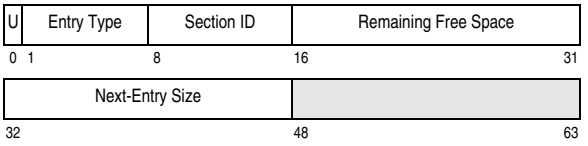
Access-List Entry (ALE)



Bit	Meaning
0	(I) ALEN-invalid bit
6	(FO) Fetch-only bit
7	(P) Private bit
8-15	(ALESN) Access-list-entry sequence number

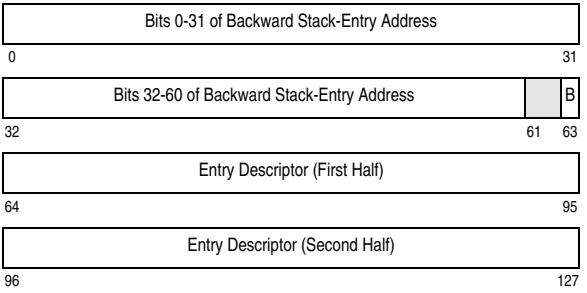
Linkage-Stack Entries

Entry Descriptor



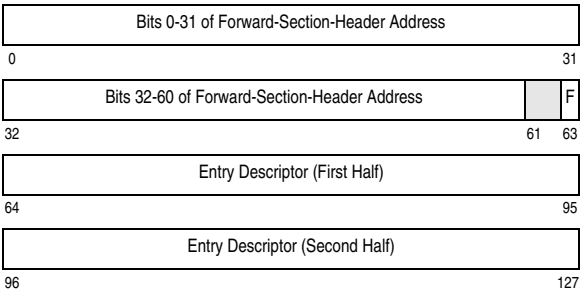
Bit	Meaning
0	(U) Unstack-suppression bit
1-7	Entry type: Header entry = 0001001 binary Trailer entry = 0001010 binary Branch state entry = 0001100 binary Program-call state entry = 0001101 binary Available for program use = 1xxxxxx binary

Header Entry (Entry Type 0001001)



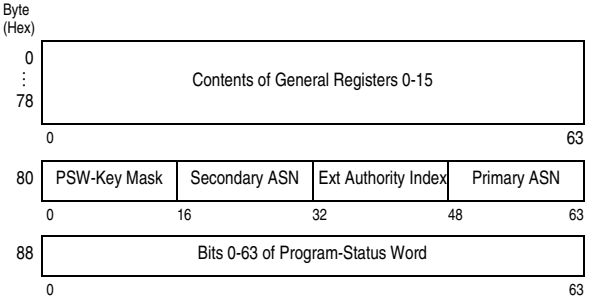
Bit	Meaning
63	(B) Backward stack-entry validity bit

Trailer Entry (Entry Type 0001010)

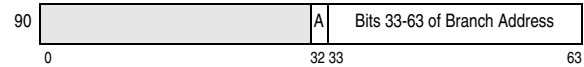


Bit	Meaning
63	(F) Forward-section validity bit

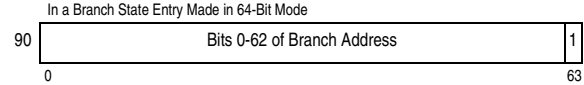
Branch State Entry (Entry Type 0001100) and Program-Call State Entry (Entry Type 0001101)



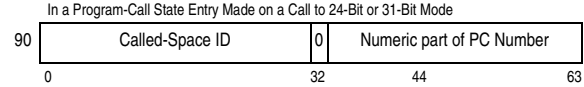
In a Branch State Entry Made in 24-Bit or 31-Bit Mode



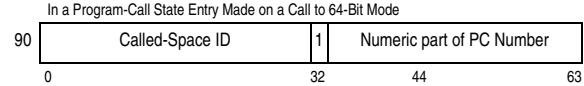
In a Branch State Entry Made in 64-Bit Mode

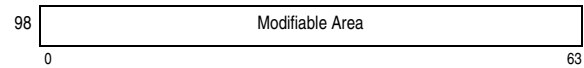


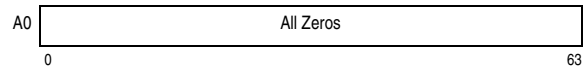
In a Program-Call State Entry Made on a Call to 24-Bit or 31-Bit Mode

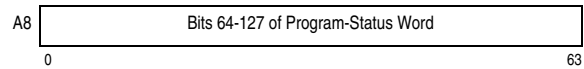


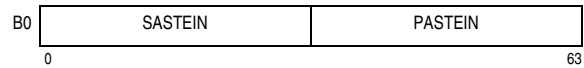
In a Program-Call State Entry Made on a Call to 64-Bit Mode

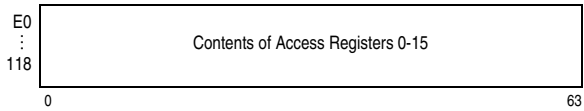


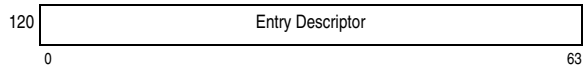








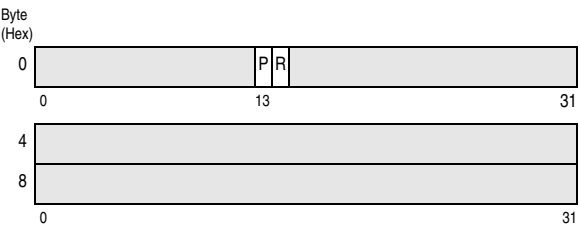


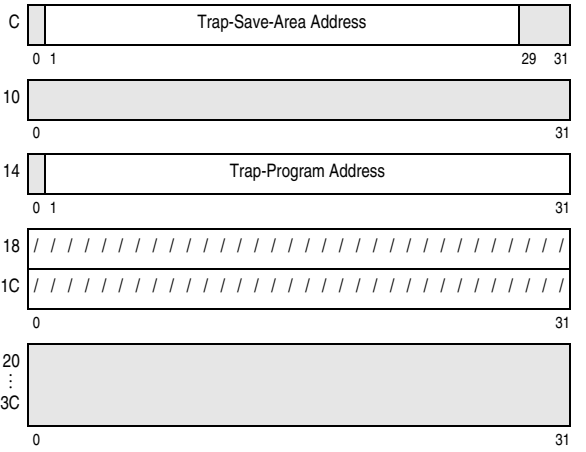


Byte.Bit	Meaning
90.32	(A) Addressing mode (in branch state entry)

Trapping

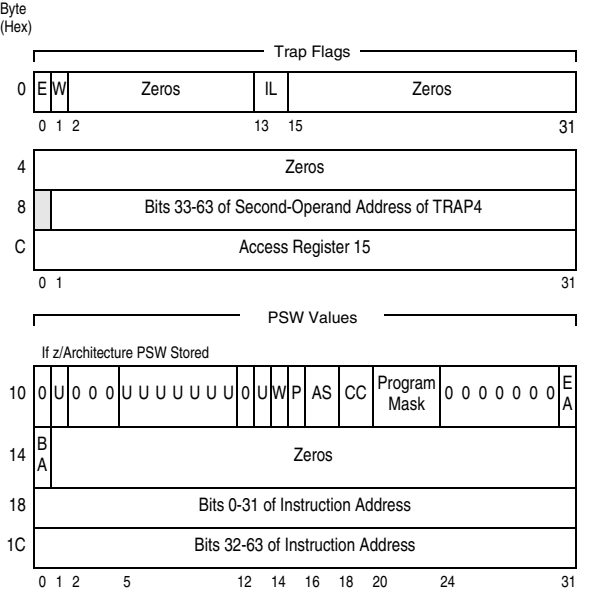
Trap Control Block

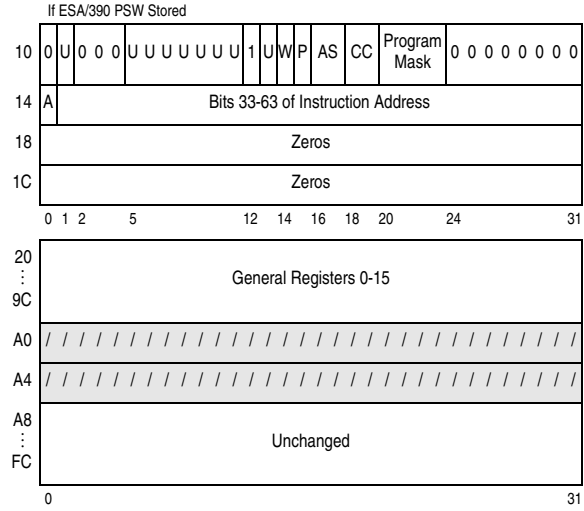




Byte.Bit	Meaning
0.13	(P) PSW control (zero: PSW.31 must be zero, ESA/390 PSW stored; one: z/Architecture PSW stored)
0.14	(R) General-register control (zero: bits 32-63 stored; one: bits 0-63 stored)
///	Available for programming

Trap Save Area





Byte.Bit	Meaning
0.0	(E) TRAP was target of EXECUTE
0.1	(W) TRAP is TRAP4 (not TRAP2)
0.13-14	(IL) Instruction-length code
10-1F	PSW values (see PSW on page 58)
U	Unpredictable
///	Available for programming

Trace-Entry Formats

Identification of Trace Entries

Trace-Entry Bits			Trace Entry	
0-7	8-11	12-15	Type	Format
00000000			Branch	1
00010000		000N	Set Secondary ASN	1
00100001			Program Call	1[1]
00100010			Program Call	2[1]
00100001		0	Program Call	3[1]
00100010		0	Program Call	4[1]
00100010		100E	Program Call	5[1]
00100010		101E	Program Call	6[1]
00100011		111E	Program Call	7[1]
00110001		000N	Program Transfer	1
00110001		100N	Program Transfer	2
00110010		0000	Program Return	1
00110010		0010	Program Return	2
00110010		1000	Program Return	4
00110010		1010	Program Return	5
00110010		110N	Program Transfer	3
00110011		0011	Program Return	3
00110011		1011	Program Return	6
00110011		1100	Program Return	7
00110011		1110	Program Return	8

Trace-Entry Bits			Trace Entry	
0-7	8-11	12-15	Type	Format
00110100		1111	Program Return	9
01000001			Branch in Subspace Group	1
01000010			Branch in Subspace Group	2
01010001	0010		Mode Switch	2
01010001	0011		Mode Switch	1
01010001	1010		Mode-Switching Branch	1
01010001	1011		Mode-Switching Branch	2
01010010	0110		Mode Switch	3
01010010	1100		Branch	3
01010010	1111		Mode-Switching Branch	3
0111	0		Trace	1
0111	1		Trace	2
1			Branch	

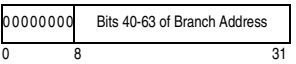
1 Format-1 and -2 entries are made when the ASN-and-LX-reuse facility (ALRF) is not enabled. Entries of formats 3-7 are made when the facility is enabled.

E Indicates, when one, that the extended-addressing-mode bit, PSW bit 31, was set to one.

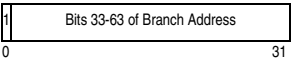
N Indicates, when one, that an entry was made because of PTI or SSAIR.

Branch

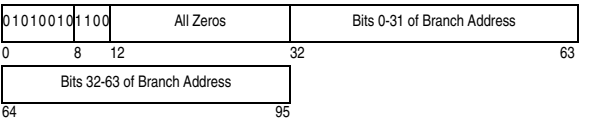
F1 (Branch, RP, or TRAP2/4 to 24-Bit Mode)



F2 (Branch, RP, or TRAP2/4 to 31/64-Bit Mode)



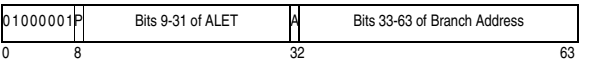
F3 (Branch, RP, or TRAP2/4 to 64-Bit Mode)



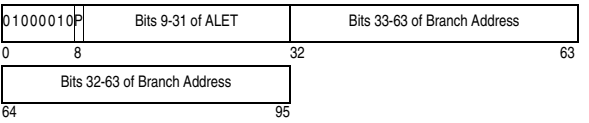
Note: "Branch" is BAKR, BALR, BASR, BASSM, BSA, or BSG.

Branch in Subspace Group (if ASN Tracing on)

F1 (in 24/31-Bit Mode)

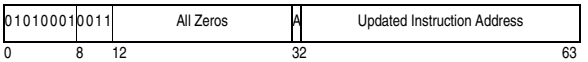


F2 (in 64-Bit Mode)

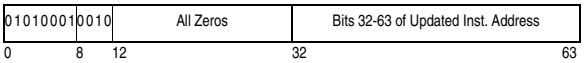


Mode Switch

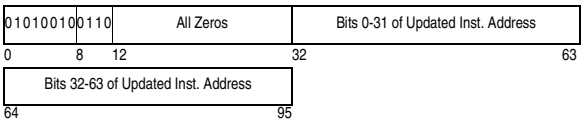
F1 (BASSM, BSM, PC, PR, RP, or SAM64 from 24/31-Bit to 64-Bit Mode)



F2 (BASSM, BSM, PC, PR, RP, SAM24/31 from 64-Bit to 24/31-Bit Mode)

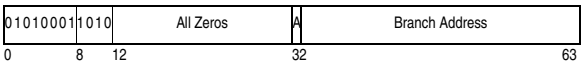


F3 (BASSM, BSM, PC, PR, RP, SAM24/31 from 64-Bit to 24/31-Bit Mode)

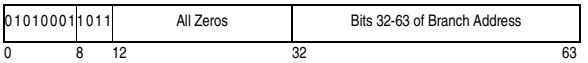


Mode-Switching Branch

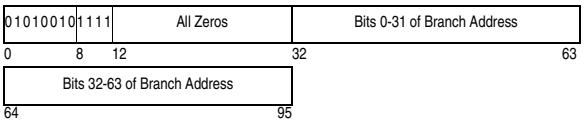
F1 (BASSM or RP from 64-Bit to 24/31-Bit Mode)



F2 (BASSM or RP from 24/31-Bit to 64-Bit Mode)

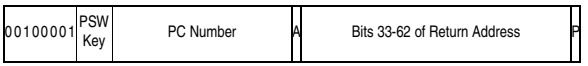


F3 (BASSM or RP from 24/31-Bit to 64-Bit Mode)

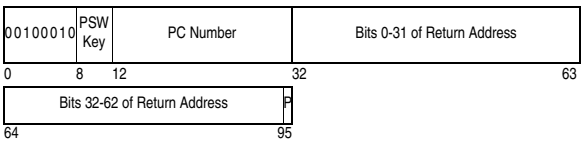


Program Call

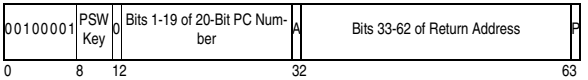
F1 (in 24/31-Bit Mode, ALRF Not Enabled)



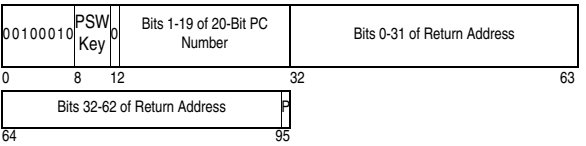
F2 (in 64-Bit Mode, ALRF Not Enabled)



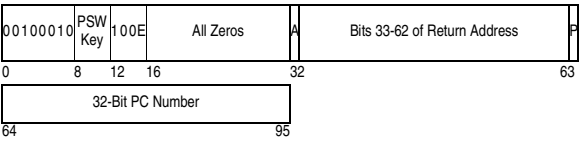
F3 (in 24/31-Bit Mode, ALRF Enabled, 20-Bit PC Number)



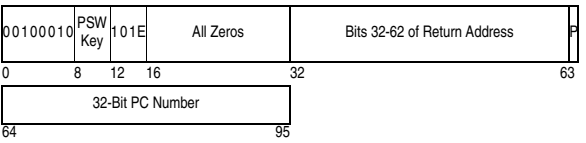
F4 (in 64-Bit Mode, ALRF Enabled, 20-Bit PC Number)



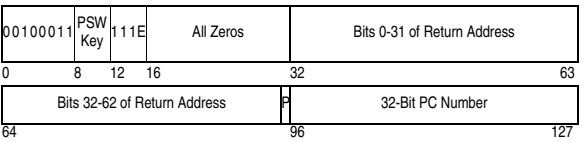
F5 (in 24/31-Bit Mode, ALRF Enabled, 32-Bit PC Number)



F6 (in 64-Bit Mode, ALRF Enabled, 32-Bit PC Number, Bits 0-31 of Return Address All Zeros)

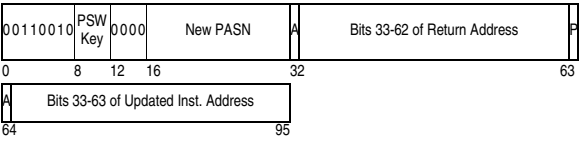


F7 (in 64-Bit Mode, ALRF Enabled, 32-Bit PC Number, Bits 0-31 of Return Address Not All Zeros)

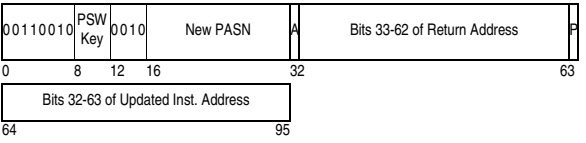


Program Return

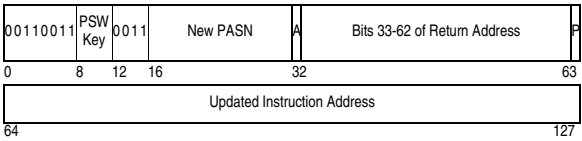
F1 (in 24/31-Bit to 24/31-Bit Mode)



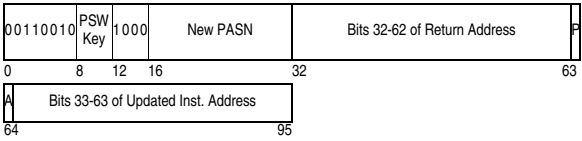
F2 (in 64-Bit to 24/31-Bit Mode)



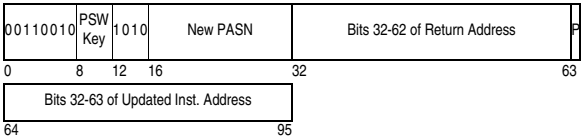
F3 (in 64-Bit to 24/31-Bit Mode)



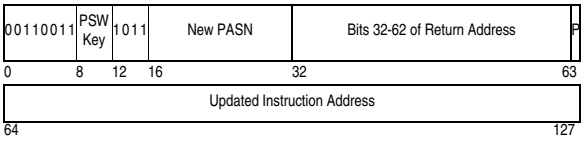
F4 (in 24/31-Bit to 64-Bit Mode)



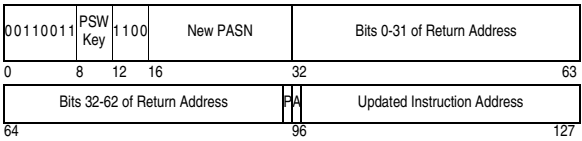
F5 (in 64-Bit to 64-Bit Mode)



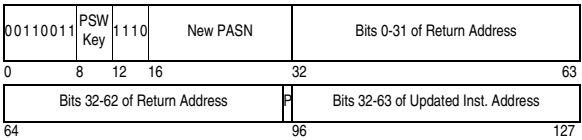
F6 (in 64-Bit to 64-Bit Mode)



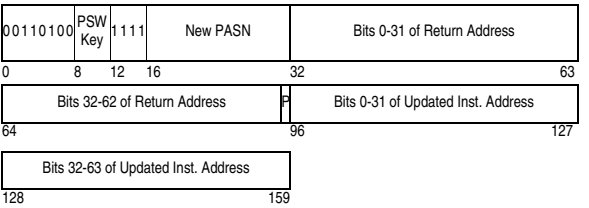
F7 (in 24/31-Bit to 64-Bit Mode)



F8 (in 64-Bit to 64-Bit Mode)

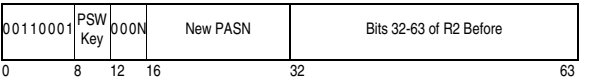


F9 (in 64-Bit to 64-Bit Mode)

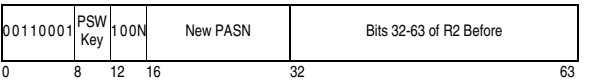


Program Transfer

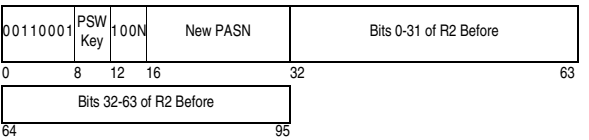
F1 (in 24/31-Bit Mode)



F2 (in 64-Bit Mode, Bits 0-31 of R_2 All Zeros)

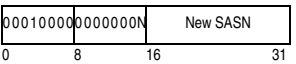


F3 (in 64-Bit Mode, Bits 0-31 of R_2 Not All Zeros)



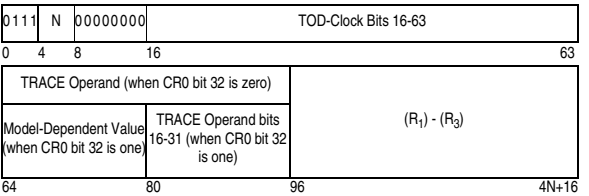
Set Secondary ASN

F1

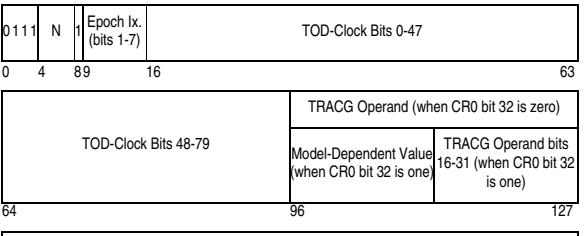


Trace

F1 (TRACE)

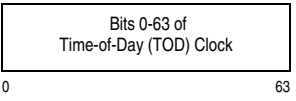


F2 (TRACG)



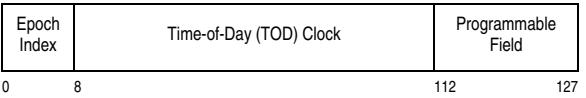
Bit	Meaning
4-7	(N) One less than the number of registers in the trace entry.

Operand of Store Clock and Store Clock Fast



Note: Bit 51 of the TOD clock corresponds to one microsecond.

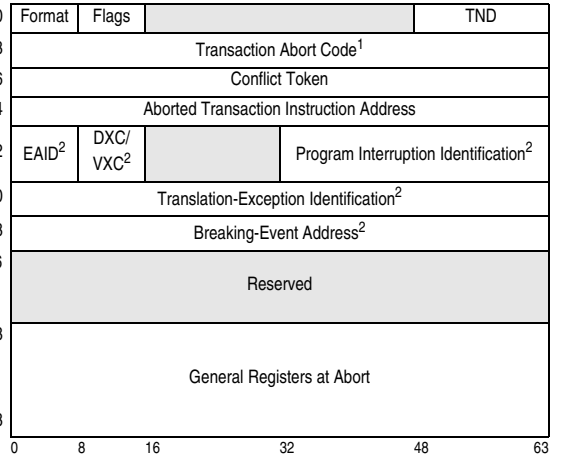
Operand of Store Clock Extended



Note: Bit 51 of the TOD clock (bit 59 of the operand) corresponds to one microsecond.

Transaction Diagnostic Block (TDB)

TBEGIN-specified TDB is the operand of the TBEGIN instruction when B1 ≠ 0; Program-interruption TDB is at real locations 6,144 - 6,399.



Explanation:

1 Transaction abort codes:

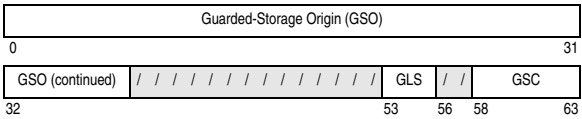
- 2 – External interruption
- 4 – Non-filtered program interruption
- 5 – Machine-check interruption
- 6 – I/O interruption
- 7 – Fetch overflow
- 8 – Store overflow
- 9 – Fetch conflict
- 10 – Store conflict
- 11 – Restricted instruction
- 12 – Filtered program interruption
- 13 – Nesting depth exceeded
- 14 – Cache fetch-related condition
- 15 – Cache store-related condition
- 16 – Cache other condition
- 19 – Guarded-storage event recognized
- 255 – Miscellaneous condition
- >255 – TABORT instruction

2 The field is stored only in the TBEGIN-specified TDB; otherwise, the field is unpredictable. The field is valid only for certain filtered program-interruption conditions. When the field is not valid, its contents are unpredictable.

TND Transaction nesting depth

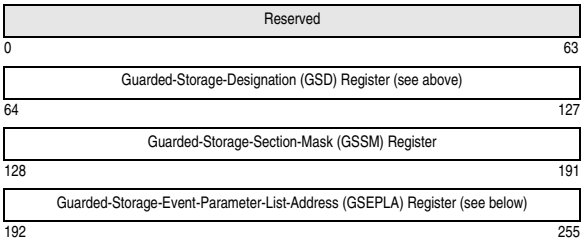
Guarded-Storage Facility Registers and Parameters

Guarded-Storage-Designation (GSD) Register

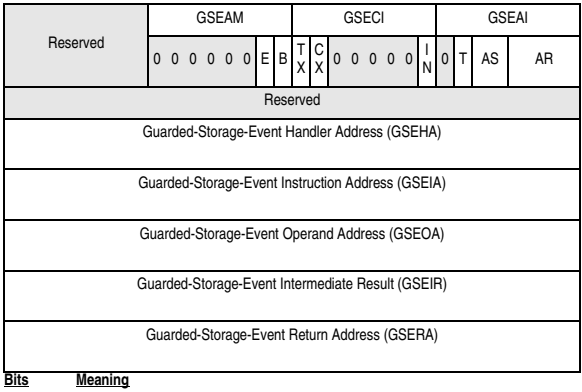


Bit	Meaning
0-J	Guarded-storage origin (GSO), where J is 64-GSC
53-55	Guarded-load shift (GLS); valid values are 0-4
58-63	Guarded-storage characteristic (GSC); valid values are 25-56

Guarded-Storage Control Block



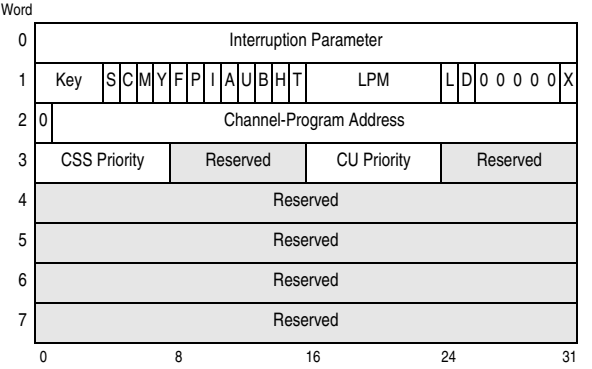
Guarded-Storage-Event Parameter List



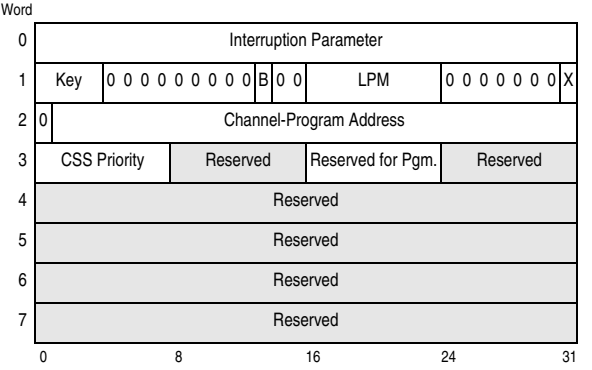
Bits	Meaning
0-7	Reserved
8-15	Guarded-storage-event addressing mode (GSEAM) 14 - Extended-addressing mode (E) 15 - Basic Addressing mode (B)
16-23	Guarded-storage-event cause indication (GSECI) 16 - CPU was in the transactional-execution mode (TX) 17 - CPU was in the constrained transactional-execution mode (CX) 23 - Instruction causing the event; 0-LGG, 1=LLGFSG
24-31	Guarded-storage-event access information (GSEAI) 25 - DAT mode (copy of PSW bit 5) 26-27 - Address-space indication (copy of PSW bits 16-17) 28-31 - AR number (when in the AR mode; otherwise unpredictable
32-63	Reserved
64-127	Guarded-storage-event handler address (GSEHA)
128-191	Guarded-storage-event instruction address (GSEIA)
192-255	Guarded-storage-event operand address (GSEOA)
256-319	Guarded-storage-event intermediate result (GSEIR)
320-383	Guarded-storage-event return address (GSERA)

Operation-Request Block (ORB)

Command-Mode ORB



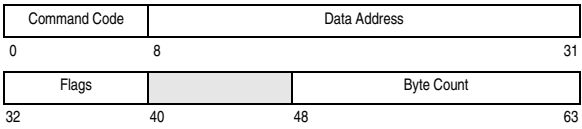
Transport-Mode ORB



Word.Bit	Meaning
1.0-3	(Key) Subchannel key
1.4	(S) Suspend control
1.5	(C) Streaming-mode control
1.6	(M) Modification control
1.7	(Y) Synchronization control
1.8	(F) CCW-format control
1.9	(P) Prefetch control
1.10	(I) Initial-status-interruption control
1.11	(A) Address-limit-checking control
1.12	(U) Suppress-suspended-interruption control
1.13	(B) Channel-Program Type
1.14	(H) Format-2-IDAW control
1.15	(T) 2K-IDAW control
1.16-23	(LPM) Logical-path mask
1.24	(L) Incorrect-length-suppression mode
1.25	(D) Modified-CCW-indirect-data-addressing control
1.31	(X) ORB-extension control
3.0-7	Channel-subsystem priority
3.16-23	Control-unit priority

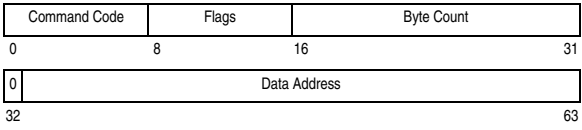
Channel-Command Word (CCW)

Format-0 CCW



Bit	Meaning
32	(CD) Causes use of data-address portion of next CCW
33	(CC) Causes use of command code and data address of next CCW
34	(SLI) Causes suppression of possible incorrect-length indication
35	(Skip) Suppresses transfer of information to main storage
36	(PCI) Causes an intermediate-interruption condition to occur
37	(IDA) Causes bits 8-31 of CCW to specify location of first IDAW
38	(Suspend) Causes suspension before execution of this CCW
39	(MIDA) Causes bits 8-31 of CCW to specify location of first MIDAW

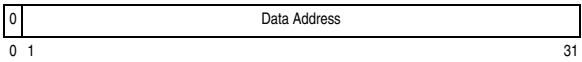
Format-1 CCW



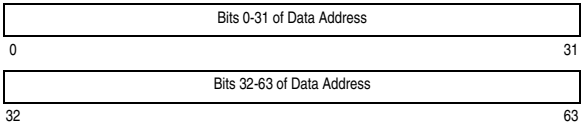
Bit	Meaning
8	(CD) Causes use of data-address portion of next CCW
9	(CC) Causes use of command code and data address of next CCW
10	(SLI) Causes suppression of possible incorrect-length indication
11	(Skip) Suppresses transfer of information to main storage
12	(PCI) Causes an intermediate-interruption condition to occur
13	(IDA) Causes bits 33-63 of CCW to specify location of first IDAW
14	(Suspend) Causes suspension before execution of this CCW
15	(MIDA) Causes bits 33-63 of CCW to specify location of first MIDAW

Indirect-Data-Address Word (IDAW)

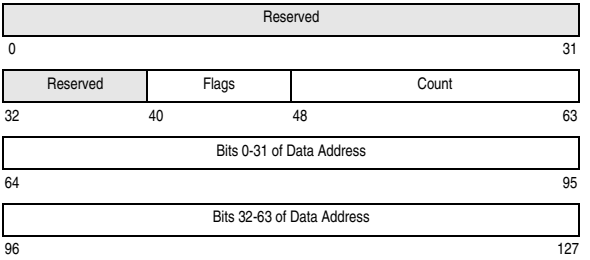
Format-1 IDAW



Format-2 IDAW

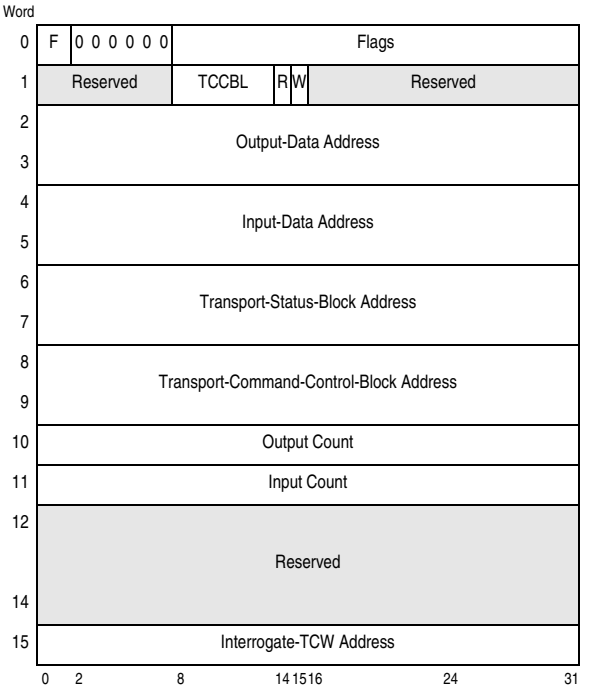


Modified-CCW-Indirect-Data-Address Word (MIDAW)



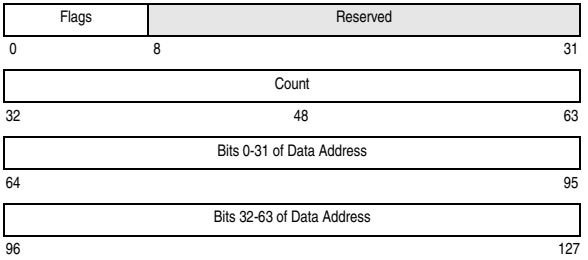
Bit	Meaning
40	Last MIDAW
41	Skip
42	Data-transfer-interruption control
43-47	Reserved

Transport Control Word (TCW)



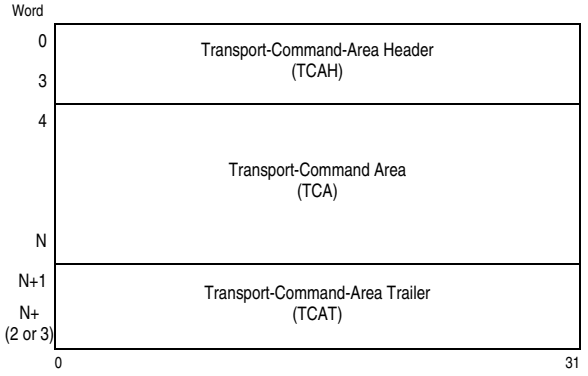
Word.Bit	Meaning
0.0-1	Format
0.13	Input transport-indirect-data addressing (TIDA)
0.14	Transport-command-control-block TIDA
0.15	Output TIDA
0.16-17	TIDAW Format
1.8-13	(TCCBL) Transport-Command-Control-Block Length
1.14	(R) Read Operations
1.15	(W) Write Operations

Transport-Indirect-Data-Address Word (TIDAW)

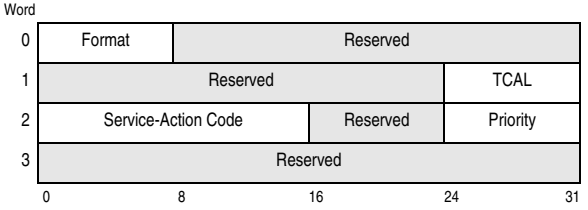


Bit	Meaning
0	Last TIDA
1	Skip
2	Data-transfer-interruption control
3	(TTIC) TIDAW Transfer In Channel
4	Insert CBC Control
5-7	Reserved

Transport Command Control Block (TCCB)

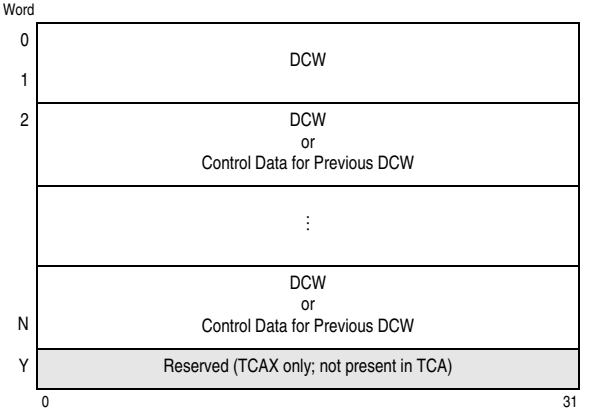


Transport Command Area Header (TCAH)

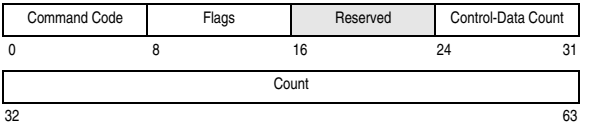


Word.Bit	Meaning
1.24-31	(TCAL) Transport-Command-Area Length

Transport-Command Area (TCA) and Transport-Command-Area Extension (TCAX)

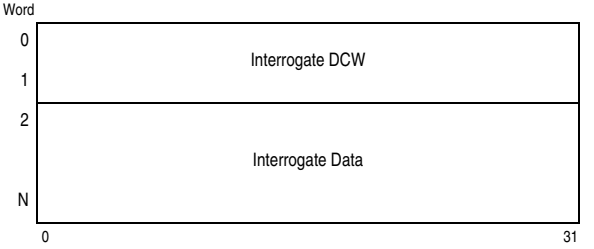


Device-Command Word (DCW)

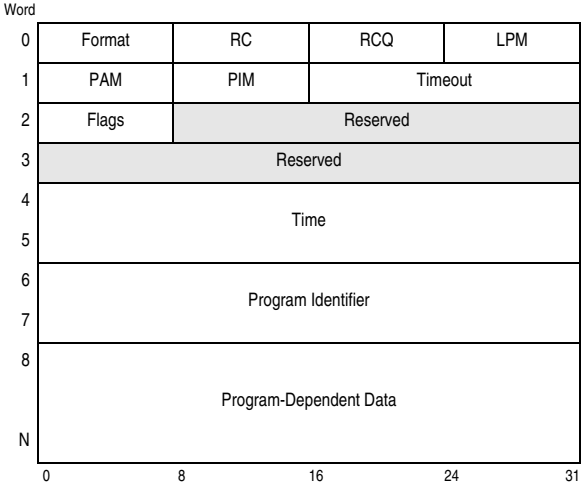


Bit	Meaning
9	(CC) Causes use of next DCW
10	(SLI) Suppresses incorrect-length indication

Interrogate TCA

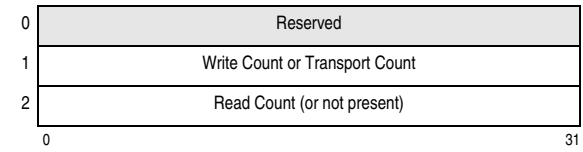


Interrogate Data

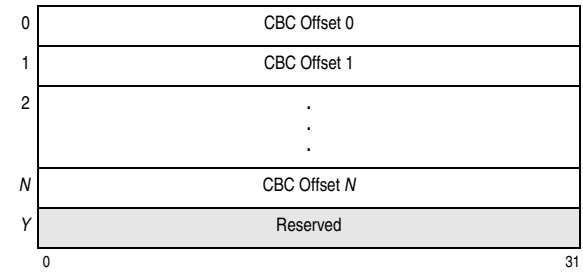


Word.Bit	Meaning
0.8-15	(RC) Reason code 0 Interrogate reason not specified 1 Timeout
0.16-23	(RCQ) Reason-code qualifier 0 Interrogate reason qualifier not specified 1 Primary 2 Secondary
0.24-31	(LPM) Logical-path mask
1.0-7	(PAM) Path-available mask
1.8-15	(PIM) Path-installed mask
2.0-7	Flags 0 Multipath mode 1 Program path recovery 2 Critical

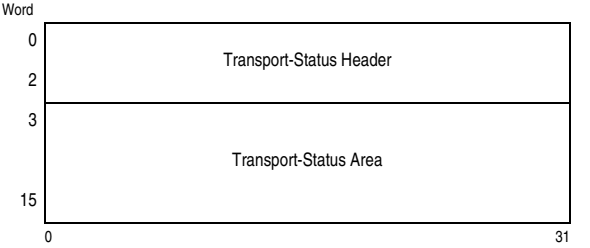
Transport Command Area Trailer (TCAT)



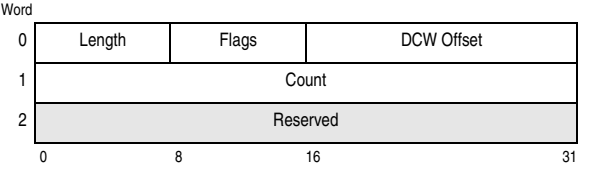
CBC-Offset Block (COB)



Transport Status Block (TSB)

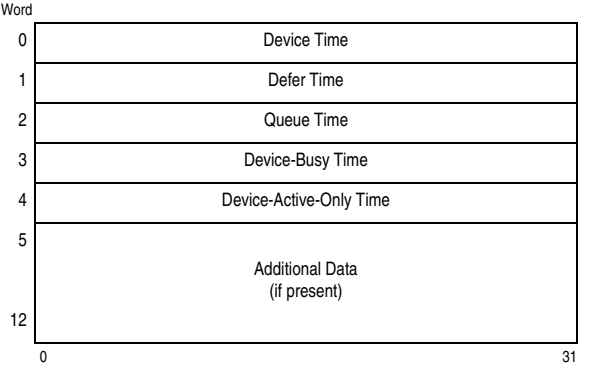


Transport Status Block Header (TSH)

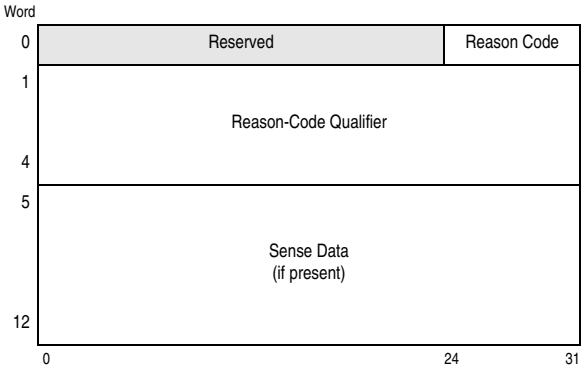


Word.Bit	Meaning
0.8	DCW-offset field valid
0.9	Count field valid
0.10	Cache miss
0.11	Time fields valid
0.13-15	Transport-Status Area (TSA) Format 0 TSA contents have no meaning 1 I/O-status TSA 2 Device-detected-program-check TSA 3 Interrogate TSA

I/O-Status TSA



Device-Detected-Program-Check TSA



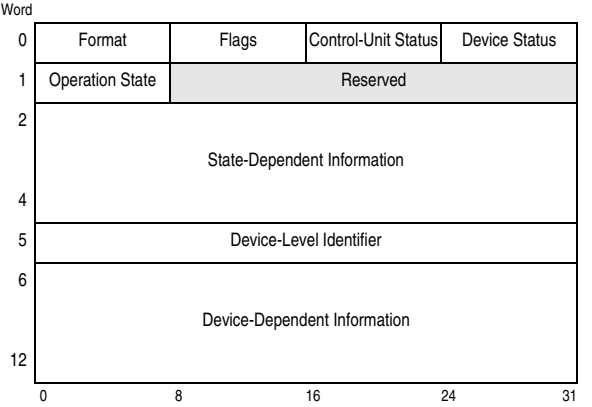
Word.Bit **Meaning**

0.24-31 (RC) Reason Code

- 0 No information
- 1 TCCB transport failure
  - (RCQ) Reason-code-qualifier byte 0 (1.0-7)
    - 0 No additional information
    - 1 TCCB transport size error
    - 2 TCCB CBC error
- 2 Invalid CBC detected on output data
  - RCQ word 0: Offset of first output-data byte for which error was detected
  - RCQ word 1: Offset of last output-data byte for which error was detected
- 3 Incorrect TCCB length specification
  - RCQ byte 0
    - 0 No additional information
    - 1 TCAL value not 8 greater than TCW TCCBL value
    - 2 TCAL value is less than 20 or greater than 252
- 4 TCAH specification error
  - RCQ byte 0
    - 0 No additional information
    - 1 Format field specification error
    - 2 Reserved field specification error
    - 3 Service-action-code field specification error
- 5 DCW specification error
  - RCQ byte 0
    - 0 No additional information
    - 1 Reserved field specification error
    - 2 Flags field command-chaining specification error
    - 3 Control-data-count field specification error
    - 4 TCOB location error
    - 5 TCOB duplication error
    - 6 TCOB multiple-count error
    - 7 TCOB direction error
    - 8 TCOB chaining error
    - 9 TCOB count-specification error
    - 10 TTE location error
    - 11 TTE duplication error
    - 12 TTE CD-count specification error
    - 13 TTE count specification error
    - 14 TTE direction error:
    - 15 TTE chaining error
    - 16 TCAX specification error
- 6 Transfer-direction specification error
  - RCQ byte 0
    - 0 No additional information
    - 1 Read-direction specification error
    - 2 Write-direction field specification error
    - 3 Read-write-conflict specification error

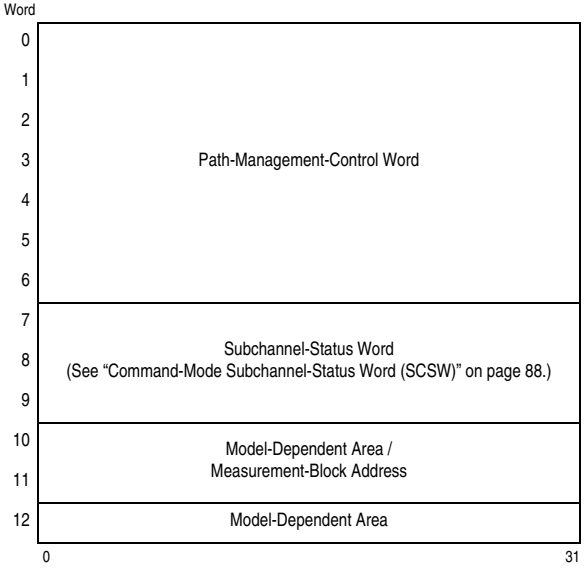
7 Transport-count specification error
RCQ byte 0
0 No additional information
1 Read-count specification error
2 Write-count specification error
8 Two I/O operations active
RCQ: No additional information
9 CBC-offset specification error
RCQ word 0: Byte offset of COB CBC-offset entry

Interrogate TSA

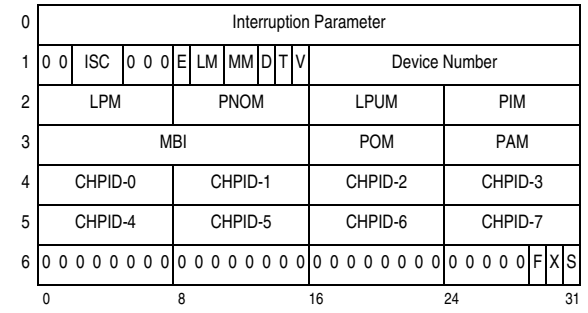


Word.Bit	Meaning
0.8	Control-unit state valid
0.9	Device-state valid
0.10	Operation-state valid
0.16-23	(CS) Control-unit state 0 Busy 1 Recovery 2 Interrogate maximum
0.24-31	(DS) Device-unit state 0 Path-Group identification (in state-dependent-information field) 1 Long busy 2 Recovery
1.0-7	(OS) Operation state 0 No I/O operation present. 1 An I/O operation is present and executing. 2 An I/O operation is present and awaiting completion of another operation initiated by another configuration. 3 An I/O operation is present and awaiting completion of another operation initiated for the same device extent. 4 An I/O operation is present and waiting to perform a device-dependent operation.

Subchannel-Information Block (SCHIB)



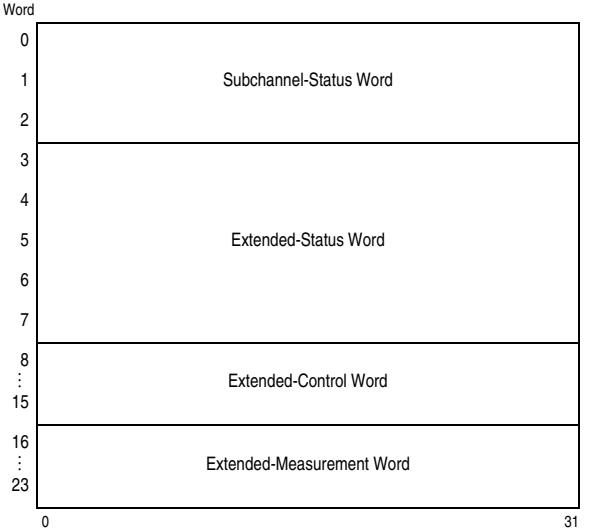
Path-Management-Control Word (PMCW)



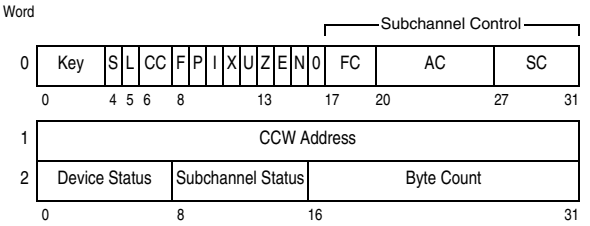
Word.Bit	Meaning
1.2-4	(ISC) Interruption-subclass code
1.8	(E) Subchannel enabled
1.9-10	(LM) limit mode
	00 No Checking
	01 Data address must be ≥ limit
	10 Data address must be < limit
	11 Reserved
1.11-12	(MM) Measurement-mode enable
	00 Neither mode enabled
	01 Device-connect-time-measurement enabled
	10 Measurement-block-update enabled
	11 Both modes enabled

1.13	(D) Multipath mode
1.14	(T) Timing facility available
1.15	(V) Device number valid
2.0-7	(LPM) Logical-path mask
2.8-15	(PNOM) Path-not-operational mask
2.16-23	(LPUM) Last-path-used mask
2.24-31	(PIM) Path-installed mask
3.0-15	(MBI) Measurement-block index
3.16-23	(POM) Path-operational mask
3.24-31	(PAM) Path-available mask
4.0-7	(CHPID-0) Channel-path ID for logical path 0 (typical)
6.29	(F) Measurement-block-format control
6.30	(X) Extended-measurement-word-mode enable
6.31	(S) Concurrent sense

Interruption-Response Block (IRB)



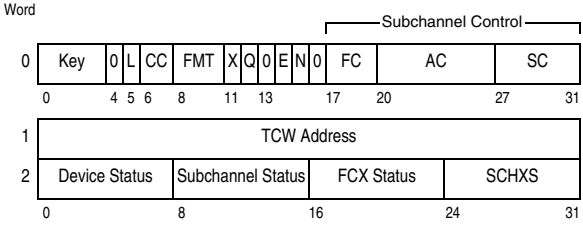
Command-Mode Subchannel-Status Word (SCSW)



Word.Bit	Meaning
0.0-3	(Key) Subchannel key
0.4	(S) Suspend control
0.5	(L) Extended-status-word format (logout stored)
0.6-7	(CC) Deferred condition code 00 Normal I/O interruption 01 Status in SCSW 10 Reserved 11 Path not operational
0.8	(F) CCW-format control
0.9	(P) Prefetch control

0.10	(I) Initial-status-interruption control	
0.11	(X) IRB-format control	
0.12	(U) Suppress-suspended-interruption control	
0.13	(Z) Zero condition code	
0.14	(E) Extended control (information stored in ECW of IRB)	
0.15	(N) Path not operational (PNOM nonzero)	
0.17-19	(FC) Function control	
	17 (40) Start, 18 (20) Halt, 19 (10) Clear	
0.20-26	(AC) Activity control	
	20 (08) Resume pending	24 (80) Subchannel active
	21 (04) Start pending	25 (40) Device active
	22 (02) Halt pending	26 (20) Suspended
	23 (01) Clear pending	
0.27-31	(SC) Status control	
	27 (10) Alert	30 (02) Secondary
	28 (08) Intermediate	31 (01) Status pending
	29 (04) Primary	
2.0-15	Device status (0-7)	Subchannel status (8-15)
	0 (80) Attention	8 (80) Program-controlled interruption
	1 (40) Status modifier	9 (40) Incorrect length
	2 (20) Control-unit end	10 (20) Program check
	3 (10) Busy	11 (10) Protection check
	4 (08) Channel end	12 (08) Channel-data check
	5 (04) Device end	13 (04) Channel-control check
	6 (02) Unit check	14 (02) Interface-control check
	7 (01) Unit exception	15 (01) Chaining check

Transport-Mode Subchannel-Status Word (SCSW)



Word.Bit	Meaning	
0.0-3	(Key) Subchannel key	
0.5	(L) Extended-status-word format (logout stored)	
0.6-7	(CC) Deferred condition code	
	00 Normal I/O interruption	
	01 Status in SCSW	
	10 Reserved	
	11 Path not operational	
0.8-10	(FMT) Format	
0.11	(X) IRB-format control	
0.12	(Q) Interrogate complete	
0.14	(E) Extended control (information stored in ECW of IRB)	
0.15	(N) Path not operational (PNOM nonzero)	
0.17-19	(FC) Function control	
	17 (40) Start, 18 (20) Halt, 19 (10) Clear	
0.20-26	(AC) Activity control	
	21 (04) Start pending	23 (01) Clear pending
	22 (02) Halt pending	25 (40) Device active
0.27-31	(SC) Status control	
	27 (10) Alert	30 (02) Secondary
	28 (08) Intermediate	31 (01) Status pending
	29 (04) Primary	

2.0-15	Device status (0-7)	Subchannel status (8-15)
	0 (80) Attention	8 (80) —
	1 (40) —	9 (40) Incorrect length
	2 (20) Control-unit end	10 (20) Program check
	3 (10) Busy	11 (10) Protection check
	4 (08) Channel end	12 (08) Channel-data check
	5 (04) Device end	13 (04) Channel-control check
	6 (02) Unit check	14 (02) Interface-control check
	7 (01) Unit exception	15 (01) Channel-subsystem retry failed
2.16-23	FCX status (16-23)	
	23 (01) TSB valid	
2.24-31	(SCHXS) Subchannel-extended status	
	24 (80) (F) Interrogate failed	
	25-31 (SESQ) SCHSX qualifier	

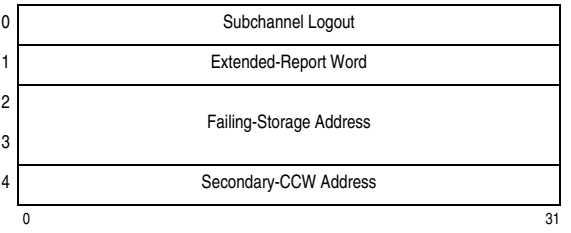
0	No status available.
1	Storage-request limit exceeded.
2	Program check when not an interrogate operation, TCW read/write data count not zero, and CE only or CE+DE only status received.
3	Transport mode not supported by the I/O device.
4	Transport mode not supported by the selected channel path.
6	Program check on TCW.
7	Device-detected program check condition due to indeterminate cause.
8	Device-detected program check.
9	Program check on TIDAW - failing-storage-address (FSA) valid in ESW (see below) and contains TIDAW address.
32	TCW access exception - FSA field valid and contains TCW address.
33	TSB access exception - FSA field valid and contains TSB address.
34	TCCB access exception - FSA field valid and contains TCCB address.
35	TIDAW access exception - FSA field valid and contains TIDAW address.
36	Data access exception - FSA field valid and contains address of data.
64	Invalid CBC error on read data.
66	Link protocol error condition.
67	Device-level recovery operation failed.
68	IFCC due to failed device-level recovery operation - program, protection, or data check may also be set in subchannel status.
70	Invalid CBC on status portion of transport response from device.
71	Invalid CBC on TSB transported from device.

Note: If FSA field valid for cases other than noted above, FSA field contains address of current TCW.

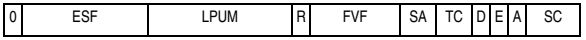
Extended-Status Word (ESW)

See chart on page 92 to determine the appropriate ESW format.

Format-0 ESW

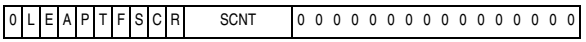


Format-0 ESW Word 0 (Subchannel Logout)



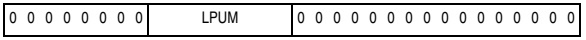
Bit	Meaning
1-7	(ESF) Extended-status flags (1 key check, 2 measurement-block program check, 3 measurement-block data check, 4 measurement-block protection check, 5 CCW check, 6 IDAW check, 7:0)
8-15	(LPUM) Last-path-used mask
16	(R) Ancillary Report
17-21	(FVF) Field-validity flags (17 LPUM, 18 TC, 19 SC, 20 device status, 21 CCW address)
22-23	(SA) Storage-access code (00 access type unknown, 01 read, 10 write, 11 read backward)
24-25	(TC) Termination code (00 halt signal issued, 01 stop, stack, or normal termination, 10 clear signal issued)
26	(D) Device status check
27	(E) Secondary error
28	(A) I/O-error alert
29-31	(SC) Sequence code

Format-0 ESW Word 1 (Extended-Report Word)



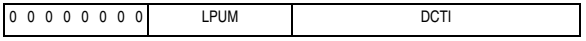
Bit	Meaning
1	(L) Request logging only
2	(E) Extended-subchannel-logout pending
3	(A) Authorization check
4	(P) Path-verification-required
5	(T) Channel-path timeout
6	(F) Failing-storage-address validity
7	(S) Concurrent sense
8	(C) Secondary-CCW-address validity
9	(R) Failing-storage-address format (zero: 1-31 of word 2; one: words 2 and 3)
10-15	(SCNT) Concurrent-sense count

Format-1 ESW Word 0[1]



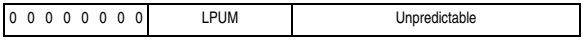
Bit	Meaning
8-15	(LPUM) Last-path-used mask

Format-2 ESW Word 0[1]



Bit	Meaning
8-15	(LPUM) Last-path-used mask
16-31	(DCTI) Device-connect-time interval

Format-3 ESW Word 0[1]



Bit	Meaning
8-15	(LPUM) Last-path-used mask

1. Word 1 is the same as word 1 of a format-0 ESW. Words 2, 3, and 4 are zeros.

Information Stored in ESW

Subchannel Conditions under which ESW Is Stored by Test Subchannel Instruction											Extended-Status Word (ESW)	
Subchannel-Status Word							Path-Management-Control Word					
Status-Control Field						Suspended Bit	Timing-Facility Bit	Device-Connect-Time Measurement-Mode Enable Bit	Device-Connect-Time Measurement-Mode Active	Format	Contents Word 0 Byte	
A	I	P	S	X	L Bit						0	1 2 3
-	-	-	-	0	-	*	*	*	No / Yes	U	*	* * *
*	*	0	0	1	1	*	*	*	No / Yes	0	R	R R R
*	*	1	*	1	1	*	*	*	No / Yes	0	R	R R R
1	0	0	1	1	1	*	*	*	No / Yes	0	R	R R R
0	0	0	0	1	0	*	*	*	No / Yes	U	*	* * *
0	0	0	1	1	0	*	*	*	No / Yes	3	Z	M * *
1	0	0	*	1	0	*	*	*	No / Yes	3	Z	M * *
*	*	1	*	1	0	*	0	*	No / Yes	1	Z	M Z Z
*	*	1	*	1	0	*	1	0	No / Yes	1	Z	M Z Z
*	*	1	*	1	0	*	1	1	No	1	Z	M Z Z
*	*	1	*	1	0	*	1	1	Yes	2	Z	M D D
0	1	0	0	1	0	0	*	*	No / Yes	U	*	* * *
0	1	0	0	1	0	1	0	*	No / Yes	1	Z	M Z Z
0	1	0	0	1	0	1	1	0	No / Yes	1	Z	M Z Z
0	1	0	0	1	0	1	1	1	No	1	Z	M Z Z
0	1	0	0	1	0	1	1	1	Yes	2	Z	M D D
0	0	0	1	1	1	These combinations do not occur.						
1	1	0	0	1	0							
*	1	0	1	1	*							

Bit **Meaning**

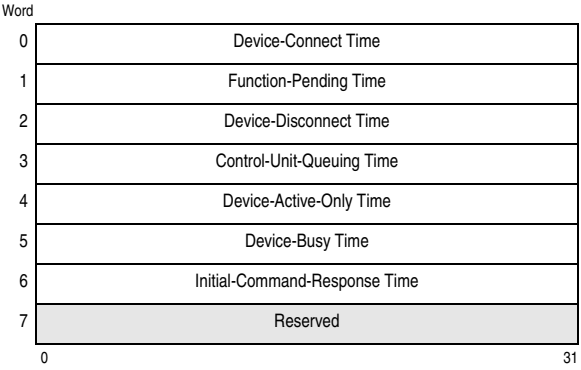
- \- Not meaningful.
- * Bits may be zeros or ones.
- A Alert status.
- D Accumulated device-connect-time-interval (DCTI) value stored in bytes 2 and 3.
- I Intermediate status.
- L Extended-status-word format.
- M Last-path-used mask (LPUM) stored in byte 1.
- P Primary status.
- R Subchannel-logout information stored in bytes 0-3.
- S Secondary status.
- U No format defined.
- X Status pending.
- Z Bits are stored as zeros.

Extended-Control Word (ECW)

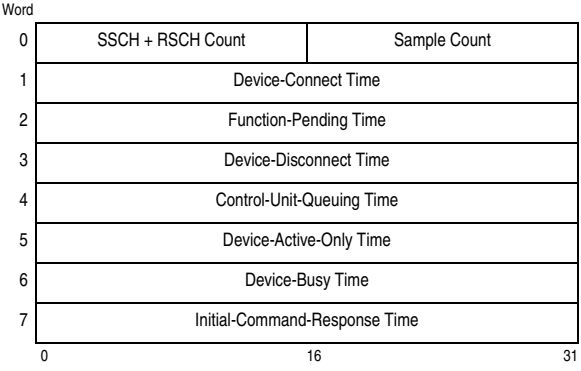
SCSW Bits 5	SCSW Bits 14	ERW Bit 7	ERW Bits 10-15	ECW Words 0-7
0	0	0	Zeros	Unpredictable
0	1	1	Number of concurrent-sense bytes[a]	Concurrent-sense information[a]
1	0	0	Zeros	Unpredictable
1	1	0	Zeros	Model-dependent information
1	1	1	Number of concurrent-sense bytes	Concurrent-sense information

a. The contents of the ECW are specified by bits 5 and 14 of word 0 of the SCSW. The combination of SCSW bit 5 zero, SCSW bit 14 one, and ERW bit 7 zero does not occur.

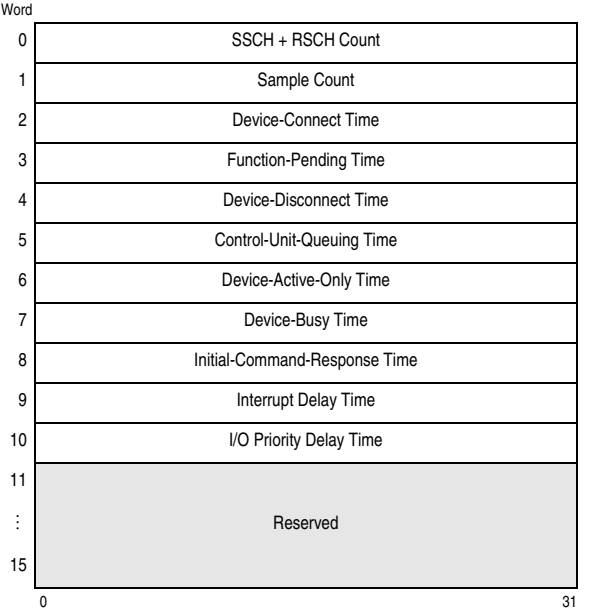
Extended-Measurement Word



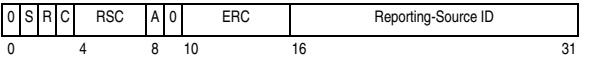
Format 0 Measurement Block



Format 1 Measurement Block



Channel-Report Word (CRW)



Bit	Meaning
1	(S) Solicited CRW
2	(R) Overflow (one or more CRWs lost)
3	(C) Chaining (meaningless if bit 2 is one)
4-7	(RSC) Reporting-source code (see Reporting-Source table)
8	(A) Ancillary report
10-15	(ERC) Error-recovery code (see Error-Recovery-Code table)
16-31	Reporting-source ID (see Reporting-Source table)

Error-Recovery Codes

ERC	Condition
0 0 0 0 0 1	Available
0 0 0 0 1 0	Initialized
0 0 0 0 1 1	Temporary error
0 0 0 1 0 0	Installed parameters initialized
0 0 0 1 0 1	Terminal
0 0 0 1 1 0	Permanent error with facility not initialized
0 0 0 1 1 1	Permanent error with facility initialized
0 0 1 0 0 0	Installed parameters modified

Reporting Source

The reporting-source-ID format depends on the RSC field of the channel-report word, as follows:

RSC	Reporting Source	Reporting-Source ID
0 0 1 0	Monitoring facility	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1	Subchannel (first or only CRW)	X X X X X X X X X X X X X X X X
0 0 1 1	Subchannel (chained CRW)	0 0 0 0 0 0 0 0 0 0 S S 0 0 0 0
0 1 0 0	Channel path	0 0 0 0 0 0 0 0 Y Y Y Y Y Y Y Y
1 0 0 1	Configuration-alert facility	0 0 0 0 0 0 0 0 Y Y Y Y Y Y Y Y
1 0 1 1	Channel subsystem	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

S = Subchannel-set identifier (SSID) when the MSS facility is installed and the CRW is chained immediately following a CRW for a subchannel.
X = Subchannel number
Y = Channel-path ID (CHPID)

I/O Command Codes

Standard Command-Code Assignments (CCW and DCW Bits 0-7)

x x x x	0 0 0 0	Invalid Command
mmmm	mm0 1	Write (a)
mmmm	mm1 0	Read (a)
0 0 0 0	0 0 1 0	— Read IPL
mmmm	mm1 1	Control
0 0 0 0	0 0 1 1	— Control no operation
0 1 mm	0 0 0 0	Transport (b)

x – Bit Ignored
m – Modifier bit for specific type of I/O device

mmmm	0 1 0 0	Sense
0 0 0 0	0 1 0 0	— Basic Sense
1 1 1 0	0 1 0 0	— Sense ID
x x x x	1 0 0 0	Transfer in channel (c)
0 0 0 0	1 0 0 0	Transfer in channel (d)
mmmm	1 0 0 0	Invalid command (e)
mmmm	1 1 0 0	Read backwards (f)

a May designate control data in a DCW
b DCW only
c Format-0 CCW
d Format-1 CCW
e Format-1 CCW and nonzero m bit
f CCW only

Standard Meanings of Bits of First Sense Byte

Bit	Designation	Bit	Designation
0	Command reject	4	Data check
1	Intervention required	5	Overrun
2	Bus-out check	6	(Device dependent)
3	Equipment check	7	(Device dependent)

Hexadecimal and Decimal Conversion

Use the following figure to preform hexadecimal and decimal conversions.

From hex: locate each hex digit in its corresponding column position and note the decimal equivalents. Add these to obtain the decimal value.

From decimal: (1) locate the largest decimal value in the table that will fit into the decimal number to be converted, and (2) note its hex equivalent and hex column position. (3) Find the decimal remainder. Repeat the process on this and subsequent remainders.

Note: Hexadecimal equivalents of all numbers from 0 to 255 are listed in “Character Assignments” on page 100.

Word															
Halfword								Halfword							
Byte				Byte				Byte				Byte			
Bits: 0123		4567		0123		4567		0123		4567		0123		4567	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	2,684,354,560	A	167,772,160	A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	2,952,790,016	B	184,549,376	B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	3,221,225,472	C	201,326,592	C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
8		7		6		5		4		3		2		1	

Powers of 2 and 16

m	*n*	2^m and 16^n	Symbol
0	0	1	
1		2	
2		4	
3		8	
4	1	16	
5		32	
6		64	
7		128	
8	2	256	
9		512	
10		1 024	K (kilo)
11		2 048	
12	3	4 096	
13		8 192	
14		16 384	
15		32 768	
16	4	65 536	
17		131 072	
18		262 144	
19		524 288	
20	5	1 048 576	M (mega)
21		2 097 152	
22		4 194 304	
23		8 388 608	
24	6	16 777 216	
25		33 554 432	
26		67 108 864	
27		134 217 728	
28	7	268 435 456	
29		536 870 912	
30		1 073 741 824	G (giga)
31		2 147 483 648	
32	8	4 294 967 296	
33		8 589 934 592	
34		17 179 869 184	
35		34 359 738 368	
36	9	68 719 476 736	
37		137 438 953 472	
38		274 877 906 944	
39		549 755 813 888	
40	10	1 099 511 627 776	T (tera)
41		2 199 023 255 552	
42		4 398 046 511 104	
43		8 796 093 022 208	
44	11	17 592 186 044 416	
45		35 184 372 088 832	
46		70 368 744 177 664	
47		140 737 488 355 328	
48	12	281 474 976 710 656	
49		562 949 953 421 312	
50		1 125 899 906 842 624	P (peta)
51		2 251 799 813 685 248	
52	13	4 503 599 627 370 496	
53		9 007 199 254 740 992	
54		18 014 398 509 481 984	
55		36 028 797 018 963 968	
56	14	72 057 594 037 927 936	
57		144 115 188 075 855 872	
58		288 230 376 151 711 744	
59		576 460 752 303 423 488	
60	15	1 152 921 504 606 846 976	E (exa)
61		2 305 843 009 213 693 952	
62		4 611 686 018 427 387 904	
63		9 223 372 036 854 775 808	

m	*n*	2^m and 16^n	Symbol
64	16	18 446 744 073 709 551 616	
65		36 893 488 147 419 103 232	
66		73 786 976 294 838 206 464	
67		147 573 952 589 676 412 928	
68	17	295 147 905 179 352 825 856	
69		590 295 810 358 705 651 712	
70		1 180 591 620 717 411 303 424	Z (zetta)
71		2 361 183 241 434 822 606 848	
72	18	4 722 366 482 869 645 213 696	
73		9 444 732 965 739 290 427 392	
74		18 889 465 931 478 580 854 784	
75		37 778 931 862 957 161 709 568	
76	19	75 557 863 725 914 323 419 136	
77		151 115 727 451 828 646 838 272	
78		302 231 454 903 657 293 676 544	
79		604 462 909 807 314 587 353 088	
80	20	1 208 925 819 614 629 174 706 176	Y (yotta)
81		2 417 851 639 229 258 349 412 352	
82		4 835 703 278 458 516 698 824 704	
83		9 671 406 556 917 033 397 649 408	
84	21	19 342 813 113 834 066 795 298 816	
85		38 685 626 227 668 133 590 597 632	
86		77 371 252 455 336 267 181 195 264	
87		154 742 504 910 672 534 362 390 528	
88	22	309 485 009 821 345 068 724 781 056	
89		618 970 019 642 690 137 449 562 112	
90		1 237 940 039 285 380 274 899 124 224	(see note)
91		2 475 880 078 570 760 549 798 248 448	
92	23	4 951 760 157 141 521 099 596 496 896	
93		9 903 520 314 283 042 199 192 993 792	
94		19 807 040 628 566 084 398 385 987 584	
95		39 614 081 257 132 168 796 771 975 168	
96	24	79 228 162 514 264 337 593 543 950 336	
97		158 456 325 028 528 675 187 087 900 672	
98		316 912 650 057 057 350 374 175 801 344	
99		633 825 300 114 114 700 748 351 602 688	
100	25	1 267 650 600 228 229 401 496 703 205 376	(see note)
101		2 535 301 200 456 458 802 993 406 410 752	
102		5 070 602 400 912 917 605 986 812 821 504	
103		10 141 204 801 825 835 211 973 625 643 008	
104	26	20 282 409 603 651 670 423 947 251 286 016	
105		40 564 819 207 303 340 847 894 502 572 032	
106		81 129 638 414 606 681 695 789 005 144 064	
107		162 259 276 829 213 363 391 578 010 288 128	
108	27	324 518 553 658 426 726 783 156 020 576 256	
109		649 037 107 316 853 453 566 312 041 152 512	
110		1 298 074 214 633 706 907 132 624 082 305 024	(see note)
111		2 596 148 429 267 413 814 265 248 164 610 048	
112	28	5 192 296 858 534 827 628 530 496 329 220 096	
113		10 384 593 717 069 655 257 060 992 658 440 192	
114		20 769 187 434 139 310 514 121 985 316 880 384	
115		41 538 374 868 278 621 028 243 970 633 760 768	
116	29	83 076 749 736 557 242 056 487 941 267 521 536	
117		166 153 499 473 114 484 112 975 882 535 043 072	
118		332 306 998 946 228 968 225 951 765 070 086 144	
119		664 613 997 892 457 936 451 903 530 140 172 288	
120	30	1 329 227 995 784 915 872 903 807 060 280 344 576	(see note)
121		2 658 455 991 569 831 745 807 614 120 560 689 152	
122		5 316 911 983 139 663 491 615 228 241 121 378 304	
123		10 633 823 966 279 326 983 230 456 482 242 756 608	
124	31	21 267 647 932 558 653 966 460 912 964 485 513 216	
125		42 535 295 865 117 307 932 921 825 928 971 026 432	
126		85 070 591 730 234 615 865 843 651 857 942 052 864	
127		170 141 183 460 469 231 731 687 303 715 884 105 728	
128	32	340 282 366 920 938 463 463 374 607 431 768 211 456	

Note: No Système international d'unités (SI) symbols greater than Y (yotta) are defined.

Character Assignments

Dec	Hex	EBCDIC[1]	ISO-8[2]
0	00	NUL	NUL
1	01	SOH	SOH
2	02	STX	STX
3	03	ETX	ETX
4	04	SEL	EOT
5	05	HT	ENQ
6	06	RNL	ACK
7	07	DEL	BEL
8	08	GE	BS
9	09	SPS	HT
10	0A	RPT	LF
11	0B	VT	VT
12	0C	FF	FF
13	0D	CR	CR
14	0E	SO	SO
15	0F	SI	SI
16	10	DLE	DLE
17	11	DC1	DC1
18	12	DC2	DC2
19	13	DC3	DC3
20	14	RES/ENP	DC4
21	15	NL	NAK
22	16	BS	SYN
23	17	POC	ETB
24	18	CAN	CAN
25	19	EM	EM
26	1A	UBS	SUB
27	1B	CU1	ESC
28	1C	IFS	IFS
29	1D	IGS	IGS
30	1E	IRS	IRS
31	1F	ITB/IUS	IUS
32	20	DS	SP
33	21	SOS	!
34	22	FS	"
35	23	WUS	#
36	24	BYP/INP	$
37	25	LF	%
38	26	ETB	&
39	27	ESC	'
40	28	SA	(
41	29	SFE	)
42	2A	SM/SW	*
43	2B	CSP	+
44	2C	MFA	,
45	2D	ENQ	-
46	2E	ACK	.
47	2F	BEL	/
48	30		0
49	31		1
50	32	SYN	2
51	33	IR	3
52	34	PP	4
53	35	TRN	5
54	36	NBS	6
55	37	EOT	7
56	38	SBS	8
57	39	IT	9
58	3A	RFF	:
59	3B	CU3	;
60	3C	DC4	<
61	3D	NAK	=
62	3E		>
63	3F	SUB	?

Dec	Hex	EBCDIC[1]	ISO-8[2]
64	40	SP	@
65	41	RSP	A
66	42	â	B
67	43	ä	C
68	44	à	D
69	45	á	E
70	46	ã	F
71	47	å	G
72	48	ç	H
73	49	ñ	I
74	4A	¢	J
75	4B	.	K
76	4C	<	L
77	4D	(	M
78	4E	+	N
79	4F	|	O
80	50	&	P
81	51	é	Q
82	52	ê	R
83	53	ë	S
84	54	è	T
85	55	í	U
86	56	î	V
87	57	ï	W
88	58	ì	X
89	59	ß	Y
90	5A	!	Z
91	5B	$	[
92	5C	*	\
93	5D	)	]
94	5E	;	^
95	5F	¬	_
96	60	-	`
97	61	/	a
98	62	Â	b
99	63	Ã	c
100	64	À	d
101	65	Á	e
102	66	Ã	f
103	67	Å	g
104	68	Ç	h
105	69	Ñ	i
106	6A	¦	j
107	6B	,	k
108	6C	%	l
109	6D	_	m
110	6E	>	n
111	6F	?	o
112	70	ø	p
113	71	É	q
114	72	È	r
115	73	Ë	s
116	74	Ë	t
117	75	Í	u
118	76	Î	v
119	77	Ï	w
120	78	Ì	x
121	79	`	y
122	7A	:	z
123	7B	#	{
124	7C	@	|
125	7D	'	}
126	7E	=	~
127	7F	"	•

Dec	Hex	EBCDIC[1]	ISO-8[2]
128	80	Ø	
129	81	a	
130	82	b	BPH
131	83	c	NBH
132	84	d	IND
133	85	e	NEL
134	86	f	SSA
135	87	g	ESA
136	88	h	HTS
137	89	i	HTJ
138	8A	«	VTS
139	8B	»	PLD
140	8C	ð	PLU
141	8D	ý	RI
142	8E	þ	SS2
143	8F	±	SS3
144	90	°	DCS
145	91	j	PU1
146	92	k	PU2
147	93	l	STS
148	94	m	CCH
149	95	n	MW
150	96	o	SPA
151	97	p	EPA
152	98	q	SOS
153	99	r	
154	9A	ª	SCI
155	9B	º	CSI
156	9C	æ	ST
157	9D	¸	OSC
158	9E	Æ	PM
159	9F	¤	APC
160	A0	µ	RSP
161	A1	~	¡
162	A2	s	¢
163	A3	t	£
164	A4	u	¤
165	A5	v	¥
166	A6	w	¦
167	A7	x	§
168	A8	y	¨
169	A9	z	©
170	AA	¡	ª
171	AB	¿	«
172	AC	Ð	¬
173	AD	Ý	SHY
174	AE	þ	®
175	AF	®	¯
176	B0	^	°
177	B1	£	±
178	B2	¥	²
179	B3	·	³
180	B4	©	´
181	B5	§	µ
182	B6	¶	¶
183	B7	¼	·
184	B8	½	¸
185	B9	¾	¹
186	BA	[	º
187	BB	]	»
188	BC	ä	¼
189	BD	¨	½
190	BE	´	¾
191	BF	×	¿

Dec	Hex	EBCDIC[1]	ISO-8[2]
192	C0	{	À
193	C1	A	Á
194	C2	B	Â
195	C3	C	Ã
196	C4	D	Ä
197	C5	E	Å
198	C6	F	Æ
199	C7	G	Ç
200	C8	H	È
201	C9	I	É
202	CA	SHY	Ê
203	CB	ô	Ë
204	CC	ö	Ì
205	CD	ò	Í
206	CE	ó	Î
207	CF	õ	Ï
208	D0	}	Ð
209	D1	J	Ñ
210	D2	K	Ò
211	D3	L	Ó
212	D4	M	Ô
213	D5	N	Õ
214	D6	O	Ö
215	D7	P	×
216	D8	Q	Ø
217	D9	R	Ù
218	DA	¹	Ú
219	DB	û	Û
220	DC	ü	Ü
221	DD	ù	Ý
222	DE	ú	Þ
223	DF	ÿ	ß
224	E0	\	à
225	E1	÷	á
226	E2	S	â
227	E3	T	ã
228	E4	U	ä
229	E5	V	å
230	E6	W	æ
231	E7	X	ç
232	E8	Y	è
233	E9	Z	é
234	EA	²	ê
235	EB	Ô	ë
236	EC	Ò	ì
237	ED	Ó	í
238	EE	Õ	î
239	EF	0	ï
240	F0	0	ð
241	F1	1	ñ
242	F2	2	ò
243	F3	3	ó
244	F4	4	ô
245	F5	5	õ
246	F6	6	ö
247	F7	7	÷
248	F8	8	ø
249	F9	9	ù
250	FA	³	ú
251	FB	Û	û
252	FC	Ü	ü
253	FD	Ù	ý
254	FE	Ú	þ
255	FF	EO	ÿ

Notes:

1 The EBCDIC characters are based on code page 037.

2 The ISO-8 controls are from ISO 6429, and the graphics are from ISO 8859-1. The ISO-8 graphics are code page 00819, named ISO/ANSI Multilingual.

Control Character Representations

ACK	Acknowledge
BEL	Bell
BS	Backspace
BYP	Bypass
CAN	Cancel
CR	Carriage Return
CSP	Control Sequence Prefix
CU1	Customer Use 1
CU3	Customer Use 3
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
DEL	Delete
DLE	Data Link Escape
DS	Digit Select
EM	End of Medium
ENP	Enable Presentation
ENQ	Enquiry
EO	Eight Ones
EOT	End of Transmission
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
FF	Form Feed
FS	Field Separator
GE	Graphic Escape
HT	Horizontal Tab
IFS	Interchange File Separator
IGS	Interchange Group Separator
INP	Inhibit Presentation
IR	Index Return
IRS	Interchange Record Separator
IT	Indent Tab
ITB	Intermediate Transmission Block
IUS	International Unit Separator
LF	Line Feed
MFA	Modify Field Attribute
NAK	Negative Acknowledge
NBS	Numeric Backspace
NL	New Line
NUL	Null
POC	Program-Operator Communication
PP	Presentation Position
RES	Restore
RFF	Required Form Feed
RNL	Required New Line
RPT	Repeat
SA	Set Attribute
SBS	Subscript
SEL	Select
SFE	Start Field Extended
SI	Shift In
SM	Set Mode
SO	Shift Out
SOH	Start of Heading
SOS	Start of Significance
SPS	Superscript
STX	Start of Text
SUB	Substitute
SW	Switch
SYN	Synchronous Idle
TRN	Transparent
UBS	Unit Backspace
VT	Vertical Tab
WUS	Word Underscore

Additional ISO-8 Control Character Representations

APC	Application Program Command
BPH	Break Permitted Here
CCH	Cancel Character
CSI	Control Sequence Introducer
DCS	Device Control String
ESA	End of Selected Area
HTJ	Character Tabulation w/ Justification
HTS	Character Tabulation Set
IFS	Information Separator Four
IGS	Information Separator Three
IND	Index
IRS	Information Separator Two
MW	Message Waiting
NBH	No Break Here
NEL	Next Line
OSC	Operating System Command
PLD	Partial Line Down
PLU	Partial Line Up
PM	Privacy Message
PU1	Private Use One
PU2	Private Use Two
SCI	Single Character Introducer
SOS	Start of String
SPA	Start of Guarded Area
SSA	Start of Selected Area
SS2	Single Shift Two
SS3	Single Shift Three
ST	String Terminator
STS	Set Transmit State
US	Information Separator One
VTS	Line Tabulation Set

Formatting Character Representations

NSP	Numeric Space
RSP	Required Space
SP	Space
SHY	Syllable Hyphen

Two-Character BSC Data Link Controls

Function	EBCDIC	ASCII
ACK-0	DLE,X'70'	DLE,0
ACK-1	DLE,X'61'	DLE,1
WACK	DLE,X'68'	DLE,;
RVI	DLE,X'7C'	DLE,<

Commonly Used Editing Pattern Characters

Code (Hex)	Meaning
20	Digit selector
21	Start of significance
22	Field separator
40	Blank
4B	Period
5B	Dollar sign
5C	Asterisk
6B	Comma
C3D9	CR (credit)
C4C2	DB (debit)

ANSI-Defined Printer Control Characters

(**A** in **RECFM** field of DCB)

Code	Action before Printing Record
blank	Space 1 line
0	Space 2 lines
-	Space 3 lines
+	Suppress space
1	Skip to line 1 on new page

SA22-7871-11