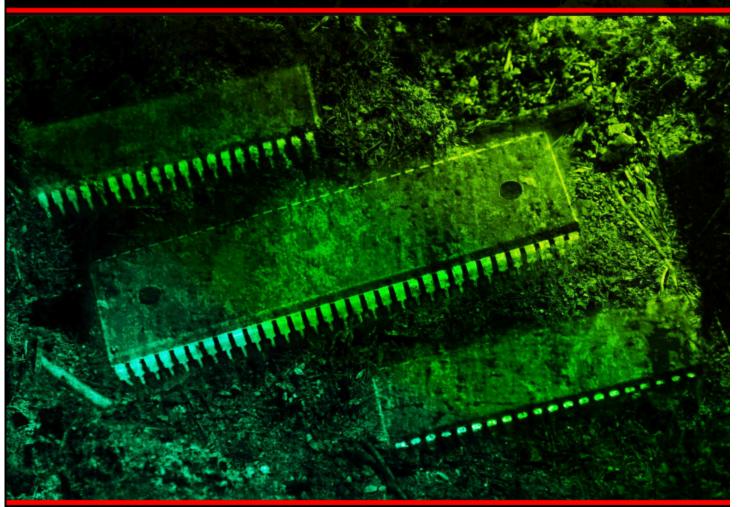
Learn Multiplatform Assembly Programming



with ChibiAkumas!





Cheatsheet Collection:

6502, 6280, 6309, 6809, 65816, 68000, 8086, ARM, ARM THUMB, IBM 370, MIPS, PDP-11, PowerPC, RISC-V, Super-H, TMS-9900, Z80, GBZ80, eZ80

Learn Multiplatform Assembly Programming with ChibiAkumas!

Ever wanted to make your own game for an old console, or learn about low level programming?

Enter the world of Assembly language, and learn new things about classic hardware!

"Learn Multiplatform Assembly Programming... with ChibiAkumas" is an introduction to retro programming. It gives the essential technical information you'll need in a 'down to earth' style that will be more accessible to the average computer user.

Volume 1 covers the Z80, 6502, 68000, 8086 and early ARM CPUs Volume 2 covers ARM Thumb, 65816, 6809, PDP-11 and Risc-V CPUs

Each book covers the terminology that relates to Assembly and classic hardware, an overview of the CPU and a list of the instruction set of that CPU, with clear simple descriptions.

For each CPU we'll look at some simple examples for an emulated computer or console to get you started, with details of how to compile and run them though an emulator on your Windows PC!



Get the book on Amazon now in Print or on Kindle. Find out more at: http://www.chibiakumas.com/book

Z80				
	Description Add registers and the corp. flag to the Accumulator A	Example	Parameters	Flags affected
	Add register r and the carry flag to the Accumulator A. Add 8 bit number # and the carry to A.	ADC B ADC 128	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL '#': 0-255 (\$00-\$FF)	S Z H V N C S Z H V N C
	Add 16 bit register rr and the carry to HL.	ADC HL,BC	'rr': BC DE HL SP	SZHVNC
	Add 16 bit register rr1 to 16 bit register rr2.	ADD HL,BC	'm1': HL IX IY 'm2': BC DE SP "HL IX IY"	H - N C
	Adds 8 bit register r to A. Adds 8 bit value # to A.	ADD B ADD B	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL '#': 0-255 (\$00-\$FF)	S Z H V N C S Z H V N C
	Logical AND of bits in register r with Accumulator A.	AND B	": (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL	SZHVNC
	Logical AND of bits in 8 bit value # with Accumulator A.	AND \$64	'#': 0-255 (\$00-\$FF)	SZHVNC
	Test bit b from 8 bit register r and set the Z flag to that bit.	BIT 7,B	'b': 0-7 (%76543210) 'r': (HL) (IX+#) (IY+#) A B C D E H L	s Z H v N -
	Call Subroutine at address addr	CALL \$1000	'addr': 0-65535 (\$0000-\$FFFF) 'addr': 0-65535 (\$0000-\$FFFF) 'c'c m nor p po pe z	
	Call Subroutine at address addr only IF condition c is true. Complement the Carry Flag. C flag will inverted	CALL Z,\$1000 CCF	'c': c m nc nz p po pe z	
	Compare the Accumulator to register r.	CP B	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHVNC
	Compare the Accumulator to 8 bit immediate value #.	CP 32	'#':0-255 (\$00-\$FF)	SZHVNC
	Compare A to the byte at address HL and decrease HL and BC. Compare A to the byte at address HL and Decrease and Repeat	CPD CPDR		SZHVN- SZHVN-
	Compare A to the byte at address HL and increase HL but decrease BC (Bytecount).	CPI		SZHVN-
	Compare A to the byte at addr HL and inc HL dec BC (Bytecount) and Rep until match or BC=0.	CPIR		SZHVN-
	Invert all bits of A (this is known as 'One's Complement').	CPL DAA		H - N -
	Decimal Adjust Accumulator (Binary Coded Decimal) Decrease value in 8 bit register r by one.	DEC B	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL	SZHV-C SZHVN-
	Decrease value in 16 bit register rr by one.	DEC HL	Valid registers for 'rr': BC DE HL IX IY SP	
	Disable Maskable Interrupts	DI	1.6.11100.1107	
	Decrease B and Jump if NonZero to address offset #. Enable Maskable Interrupts.	DJNZ label El	'ofst': -128 to +127	
	Exchange HL with the top item of the stack	EX (SP),HL		
EX AF,AF'	Exchange the Accumulator and Flags with the shadow Accumulator and Flags.	EX AF,AF'		SZHVNC
	Exchange HL and DE	EX DE,HL		
	Exchange the registers BC, DE and HL with the shadow registers Stop the CPU until an interrupt occurs.	EXX HALT		
	Enable Interrupt mode 0.	IM0		
	Enable Interrupt mode 1.	IM1		
	Enable Interrupt mode 2. Read in an 8 bit byte A from 8 bit port #.	IM2 IN A,(\$10)	'#': 0-255 (\$00-\$FF)	S Z H V N -
	Read in an 8 bit byte into register r from port (C)	IN A,(\$10)	'r': A B C D E H L	SZHVN-
INC r	Increase value in 8 bit register r by one.	INC B	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL	SZHVN-
	Increase value in 16 bit register r by one.	INC HL	'rr': BC DE HL IX IY SP	
	Read a byte IN from port (C) and save to address in HL, then Decrease HL and B. Read a byte IN from port (C) and save to address in HL. then Decrease HL and B, rep until B=0.	IND INDR		s Z h v N - s Z h v N -
	Read a byte IN from port (C) and save to address in HL, then increase HL and decrease B.	INI		s Z h v N -
	Read a byte IN from port (C) and save to the address in HL, inc HL and dec B, rep until B=0.	INIR		s Z h v N -
	Jump to the address in register HL. Jump to the 16 bit address addr.	JP (HL) JP \$4000	'addr': 0-65535 (\$0000-\$FFFF)	
	Jump to the 16 bit address addr. Jump to the 16 bit address addr only IF condition c is true in the flags register.	JP Z,\$4000	'addr': 0-65535 (\$0000-\$FFFF)	
	Jump to the 8 bit offset #.	JR TestLabel	'c'. c m nc nz p po pe z '#': -128 to +127	
	Jump to the 8 bit offset ofst IF condition c is true.	JR Z,TestLabel	'ofst': -128 to +127	
• •	Load the 8 bit value in the Accumulator into the address in register rr.	LD (DE),A	'm': BC DE HL IX+# IY SP	
	Load the 8 bit value in register r into the address in register rr. Load the 8 bit value in the Accumulator into memory address addr.	LD (HL),B LD (\$C000),A	% ABCDEHL % HLX# I/Y# 'addr': 0-65535 (\$0000-\$FFFF)	
	Load the 16 bit value in register pair rr into memory address addr.	LD (\$C000),A	India, o cerae (conon ceree)	
	Load the 8 bit value from the address in register rr into the Accumulator.	LD A,(DE)	'rr': BC DE HL IX+# IY SP	
	Load the 8 bit value from memory address addr into the Accumulator.	LD A,(\$C000)	'##': 0-65535 (\$0000-\$FFFF)	
	Load the 8 bit register r with value #.	LD B,32	Y: A B C D E H L IXH IXL IYH IYL #: 0-255 (\$00-\$FF)	
	Load the 8 bit value from the I register to the Accumulator. Load the 8 bit value from the R register to the Accumulator.	LD A,I LD A,R		SZHVN-
,	Load the 16 bit register pair rr from memory address addr.	LD BC,(\$C000)	'rr': BC DE HL IX IY SP 'addr': 0-65535 (\$0000-\$FFFF)	
	Load the 16 bit register pair rr with immediate value ####	LD BC,\$C000	'rr: BC DE HL IX IY SP 'addr': 0-65535 (\$0000-\$FFFF)	
	Load the 8 bit value from the Accumulator into the I register.	LD I,A		
	Load the R register with the 8 bit value in the Accumulator. Load the 16 bit Stack Pointer register SP with the value in HL.	LD R,A LD SP.HL		
	Load the 8 bit register r1 from register r2.	LD H,B	'r1' and 'r2': A B C D E H L IXH IXL IYH IYL	
	Load the 8 bit register r from the address in register rr.	LD B,(HL)	'Y: A B C D E H L 'm': HL IX+# IY+#	
	Load and Decrement. Copies bytes down from HL to DE with BC as a byte count.	LDD		H V N -
	Load, Decrement and Repeat. Copies bytes down from HL to DE with BC as a Byte count Load and Increment. Copies bytes upwards from HL to DE with BC as a byte count	LDDR LDI		H V N -
	Load, Decrement and Repeat. Copies bytes upwards from HL to DE with BC as byte count	LDIR		H V N -
	Negate the 8 bit value in the accumulator (Two's Complement of the number).	NEG		SZHVNC
	No Operation. This command has no effect on any registers or memory.	NOP OR B	Y: (HI) (IX+#) (IX+#) A B C D E H L IXH IXI IXI IXI	 S Z H V N C
	Logical OR of bits in register r with Accumulator A. Logical OR of bits in 8 bit value # with Accumulator A.	OR 8 OR \$64	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL '#': 0-255 (\$00-\$FF)	SZHVNC
OTDR	Out Decrement Repeat. Transfers B bytes from HL to port (C) moving downwards.	OTDR	\	s Z h v N -
	Out Increment Repeat. This command transfers B bytes from HL to port (C) moving upwards.	OTIR	I#I. O 255 (#00 #55\	s Z h v N -
	Output an 8 bit byte from A to 8 bit port #. On a system with 8 bit ports, this will output an 8 bit byte from register r to port (C).	OUT (\$10),A OUT (C),r	'#': 0-255 (\$00-\$FF) 'r': A B C D E H L	
	On a system with 8 bit ports, this will output an 8 bit byte zero to port (C).	OUT (C),0		
OUTD	Out and Decrement. This command transfers a byte from HL to port (C) moving downwards.	OUTD		s Z h v N -
	Out and Increment. This command transfers a byte from HL to port (C) moving upwards. Pop a pair of bytes off the stack into 16 bit register rr.	OUTI POP AF	'rr': AF BC DE HL IX IY	s Z h v N -
	Push a pair of bytes from 16 bit register rr onto the top of the stack.	PUSH AF	'rr': AF BC DE HL IX IY	all if AF / none
	Reset bit b from 8 bit register r to 0.	RES 7,B	b: 0-7 (%76543210) 'r': (HL) (IX+#) A B C D E H L	
RET	Return from a subroutine.	RET		
	Return from a subroutine only if condition c is true.	RET Z RETI	'c': c m nc nz p po pe z	
	Return from an interrupt. Return from a non maskable interrupt (NMI).	RETN		
RLr	Rotate bits in register r Left with Carry.	RL B	'r': (HL) (IX+#) (IY+#) A B C D E H L	S Z H P N C
	Rotate bits in register r Left and Copy the top bit to the Carry.	RLC B	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHPNC
	Rotate Left for binary coded Decimal. Rotate bits in register r Right with carry.	RLD RR B	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHVN- SZHPNC
	Rotate bits in register r Right and Copy the bottom bit to the Carry.	RLC B	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHPNC
RRD	Rotate Right for binary coded Decimal.	RRD		SZHVN-
	ReSeT function. RST is a single byte call to \$00xx address.	RST \$38	H. (III.) (IV. 45 (IV. 45 A.B. C.D. E. IV. 2012 20 20 20 20 20 20 20 20 20 20 20 20 20	
	Subtract register r and the carry flag from the Accumulator A. Subtract 8 bit number # and the carry from A.	SBC B SBC 128	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL '#': 0-255 (\$00-\$FF)	S Z H V N C S Z H V N C
	Subtract 16 bit register rr and the carry from HL.	SBC HL,BC	'r': BC DE HL SP	SZHVNC
	Set the carry flag to 1.	SCF		H - N C
	Set bit b from 8 bit register r to 1.	SET 7,B	'b': 0-7 (%76543210) 'f': (HL) (IX+#) (IY+#) A B C D E H L	
	Shift the bits register r Left for Arithmetic. Shift the bits in register r Left Logically (for unsigned numbers).	SLA A SLL A	'r': (HL) (IX+#) (IY+#) A B C D E H L 'r': (HL) (IX+#) (IY+#) A B C D E H L	S Z H P N C S Z H P N C
	Shift the bits in register r Left Logically (for unsigned numbers). Shift the bits in register r Right for Arithmetic. '	SRA A	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHPNC
SRL r	Shift the bits in register r Right Logically.	SRL A	'r': (HL) (IX+#) (IY+#) A B C D E H L	SZHPNC
	Subtract 8 bit register r from A.	SUB B	'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL	SZHVNC
		CLID OC		
SUB#	Subtract 8 bit value # from A. Logical XOR (eXclusive OR) of bits in register r with Accumulator A.	SUB 32 XOR B	'#': 0-255 (\$00-\$FF) 'r': (HL) (IX+#) (IY+#) A B C D E H L IXH IXL IYH IYL	S Z H V N C S Z H V N C

Bit	Flag	Name	Description
7	S	Sign	Positive / Negative
6	Z	Zero	Zero Flag (0=zero)
5	-		
4	Н	Half Carry	Used by DAA
3	-		
2	P / V	Parity / Overflow	Detects sign change / parity in bitshift ops
1	N	Add / Subtract	Used by DAA
0	С	Carry	Carry / Borrow

Condition	Meaning	Flag
Z	Zero	Z Set
NZ	Non Zero	Z Clear
С	Carry	C Set
NC	No Carry	C Clear
РО	Parity Odd	P/V Clear
PE	Parity Even	P/V Set
Р	Positive Sign	S Clear
М	Minus Sign	S Set

Address	Purpose	Detail
\$0000	RST0	Reset
\$0008	RST1	
\$0010	RST2	
\$0018	RST3	
\$0020	RST4	
\$0028	RST5	
\$0030	RST6	
\$0038	RST7	Interrupt Mode 1 Interrupt Handler
\$0066	NMI	Non Maskable Interrupt

Selection	Z80 - GBZ		D/T			D./3		land-mil	0	B / T		land and the	0	0.45			0					D/T
Column	ADC A,(IX+d)	8E DD 8E d	1/7 - z v 3/19 - z v	c CALL addr c CALL c,addr	CD dr ad DC dr ad	3 / 17 20		LD (IY+d),A LD (IY+d),B	FD 77 d FD 70 d	3 / 19 3 / 19		LD IXL,A LD IXL,B	DD 6F DD 68	2/8 2/8		RES 7,(HL) RES 7,(IX+d)	CB BE DD CB d BE	2 / 15 4 / 23		SET 4,(HL) SET 4,(IX+d)	CB E6 DD CB d E6	2/15 4/23
18	ADC A,A ADC A,B	8F 88	1/4 - z v 1/4 - z v	c CALL nc,addr c CALL nz,addr	D4 dr ad C4 dr ad			LD (IY+d),D LD (IY+d),E	FD 72 d FD 73 d	3 / 19 3 / 19		LD IXL,D LD IXL,E	DD 6A DD 6B	2/8 2/8		RES 7,A RES 7,B	CB BF CB B8	2/8 2/8		SET 4,A SET 4,B	CB E7 CB E0	2/8
Section	ADC A,D ADC A,E	8A 8B	1/4 - z v 1/4 - z v	c CALL po,addr c CALL pe,addr	E4 dr ad EC dr ad	19/90 3/117710 19/90 3/117710 19/90 3/117710		LD (IY+d),L LD (IY+d),n	FD 75 d FD 36 d n	3 / 19 4 / 19		LD IXL,IXL LD IXL,n	DD 6D DD 2E n	2/8 3/11		RES 7,D RES 7,E	CB BA CB BB	2/8 2/8		SET 4,D SET 4,E	CB E2 CB E3	2/8
The column	ADC A,IXH ADC A,IYH	DD 8C	2/8 - z v	c CCF	3F	12/20	x	LD A,(BC)	0A	1/78		LD IYL,B	FD 68	2/8		RES 7,L	CB BD	2/8		SET 4,L	CB E5	2/8
Column	ADC A,IXL	DD 8D	2/8 - z v	c CP (IY+d)	FD BE d	3 / 19	- = v <	LD A,(HL) LD A,(IX+d)	DD 7E d	3 / 19		LD IYL,E	FD 6B	2/8		RET M	F8	1 / t11 f8 8/16		SET 5,(IY+d)	FD CB d EE	4 / 23
Section Control Cont	ADC A,n ADC HL,BC	CE n ED 4A	2/7 - Z V 2/15 - Z V	C CP B CP C	B8 B9	1/4 1/4	- = v < - = v <	LD A,A LD A,B	7F 78	1/4		LD IYL,IYL LD IYL,n	FD 6D FD 2En	2/8 3/11		RET NZ RET P	C0 F0	1 / t11 f8 8/16		SET 5,B SET 5,C	CB E8 CB E9	2/8
Column	ADC HL,DE ADC HL,HL ADC HL,SP	ED 6A	2/15 - ZV	C CP E	BB	1/4	- = v <	LD A,D	7A	1/4		LD SP,(addr)	ED 7B dr ad	4/20		RET PO	E0	1 / t11 f8 8/16		SET 5,E	CB EB	2/8
14	ADD A,(HL) ADD A,(IX+d)	DD 86 d	3/19 - z v	c CP IYH	FD BC	2/8	- = v <	LD A,IXH	DD 7C	2/8		LD SP,IX	DD F9	2/10		RETN	ED 45	2/14		SET 6,(HL)	CB F6	2 / 15
14	ADD A,A ADD A,B	87 80	1/4 - z v 1/4 - z v	C CP IXL CP IYL	DD BD FD BD	2/8	- = v < - = v <	LD A,I LD A,L	ED 57 7D	2/9 1/4	- z i -	LDD LDDR	ED A8 ED B8	2 / 216n221	BC -	RL (IX+d) RL (IY+d)	DD CB d 16 FD CB d 16	4/23 4/23	- z p r7 - z p r7	SET 6,(IY+d) SET 6,A	FD CB d F6 CB F7	4/23 2/8
Series Content of the	ADD A,C ADD A,D ADD A,E	82	1/4 - z v 1/4 - z v	c CPD	ED A9	2 / 16	? - BC -	LD A,IYL LD A,n	FD 7D	2/8 2/78		LDI (HL),A	22	2/8 2/8		RL B	CB 10	2/8	- z p r7	SET 6,C	CB F1 CB F2	2/8
Series Belle	ADD A,H ADD A,IXH ADD A,IYH	DD 84	2/8 - Z V	c CPIR	ED B2	2 / =16#21	? - BC -	LD A,(\$FF00+C)	F2	1/8		LDI	ED A0	2/16	BC -	RL E	CB 13	2/8	- z p r7	SET 6,H	CB F4	2/8
Selection of the content of the cont	ADD A,L ADD A,IXL	DD 85	1/4 - z v 2/8 - z v	C DAA C DEC (HL)	27 35	1 / 11	- z v -	LD B,(HL) LD B,(IX+d)	DD 46 d	3 / 19		NOP	0	1/4		RLA	CB 15 17	1/4	- z p r7 r7	SET 7,(HL) SET 7,(IX+d)	DD CB d FE	4 / 23
00 May 10	ADD A,n ADD HL,BC	C6 n 09	2/7 - Z V 1/11	c DEC (IY+d) c DEC A	FD 35d 3D	3/23 1/4	- Z V - - Z V -	LD B,A LD B,B	47 40	1/4		OR (IX+d) OR (IY+d)	DD B6 d FD B6 d	3 / 19 3 / 19	- z p - - z p -	RLC (IX+d) RLC (IY+d)	DD CB d 06 FD CB d 06	4/23 4/23	- z p r7 - z p r7	SET 7,A SET 7,B	CB FF CB F8	2/8
Service Servic	ADD HL,DE ADD HL,HL ADD HL,SP	29	1/11	c DEC BC	0B	1/6	BUG	LD B,D	42	1/4		OR B	B0	1/4	- z p -	RLC B	CB 00	2/8	- z p r7	SET 7,D	CB FA	2/8
SCHOOL COUNTY OF THE PARTY OF T	ADD IX,BC ADD IX,DE ADD IX.IX	DD 19	2/15	DEC DE	1B	1/6	BUG	LD B,IXH	DD 44	2/8		OR E	B3	1/4	- z p -	RLC E	CB 03	2/8	- z p r7	SET 7,L	CB FD	2/8
STATE 100	ADD IX,SP ADD IY,BC	DD 39 FD 09	2/15 2/15	DEC H DEC IXH	25 DD 25	1/4	- Z V - - Z V -	LD B,L LD B,IXL	45 DD 45	1/4		OR IXH OR IYH	DD B4 FD B4	2/8 2/8	- z p - - z p -	RLC L RLCA	CB 05 7	2/8 1/4	- z p r7 г7	SLA (IX+d) SLA (IY+d)	DD CB d 26 FD CB d 26	4/23 - z p 4/23 - z p
Note 1	ADD IY,IY ADD IY,SP	FD 29 FD 39	2/15 2/15	DEC HL DEC IX	2B DD 2B	1/6	BUG	LD B,n LD BC,(addr)	06 n ED 4B dr ad	2/78 4/20		OR IXL OR IYL	DD B5 FD B5	2/8 2/8	- z p - - z p -	RR (HL) RR (IX+d)	CB 1E DD CB d 1E	2 / 15 4 / 23	- z p r0 - z p r0	SLA B SLA C	CB 20 CB 21	2/8 - z p 2/8 - z p
Series of the control	ADD SP,n AND (HL) AND (IX+d)	A6	1/7 - z p 3/19 - z p	c DEC L	2D	1/4	- z v -	LD C,(HL) LD C,(IX+d)	4E DD 4E d	1/7 3/19		OTDR	ED BB	2 / z16nz21 2 / z16nz21	? - ? -	RR A	CB 1F	2/8 2/8	- z p r0	SLA E	CB 23	2/8 - z p 2/8 - z p
West	AND (IY+d) AND A AND B	FD A6 d A7	3/19 - z p 1/4 - z p	c DEC IYL c DEC SP	FD 2D 3B	1/6	- Z V -	LD C,(IY+d) LD C,A	DD 4E d 4F	1/4		OUT (C),A OUT (C),B	ED 79 ED 41	2 / 12 16		RR C RR D	CB 19 CB 1A	2/8	- z p r0 - z p r0	SLA L SLL (HL)	CB 25 CB 36	2/8 - z p 2/15 - z p
No. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.	AND C AND D	A1 A2	1/4 - z p 1/4 - z p	c DJNZ d c El	10 d FB	2/t13f8 1/4		LD C,C LD C,D	49 4A	1/1		OUT (C),D OUT (C),E	ED 51 ED 59	2 / 12 16 2 / 12 16		RR H RR L	CB 1C CB 1D	2/8	- z p r0 - z p r0	SLL (IY+d) SLL A	FD CB d 36 CB 37	4/23 - z p 2/8 - z p
No. 1. A. 14 19 19 SECRET. 19 11 11 11 11 11 11 11 11 11 11 11 11	AND H AND IXH	A4 DD A4	1/4 - z p 2/8 - z v	c EX (SP),IX c EX (SP),IY	DD E3 FD E3	2/23		LD C,H LD C,IXH	4C DD 4C	1/4		OUT (C),L OUT (n),A	ED 69 D3 n	2 / 12 16 2 / 11 12		RRC (HL) RRC (IX+d)	CB 0E DD CB d 0E	2 / 15 4 / 23	- z p r0 - z p r0	SLL C SLL D	CB 31 CB 32	2/8 - z p 2/8 - z p
Month, 11/2 17 12 12 12 12 12 12 12 12 12 12 12 12 12	AND IYH AND L AND IXL	A5	1/4 - z p	c EX DE,HL	EB	1/4		LD C,L	4D	1/4		OUTI	ED A3	2 / 16 25	? <>B? -	RRC A	CB 0F	2/8	- z p r0	SLL H	CB 34	2/8 - z p
	AND IYL AND n	E6 n	2/7 - z p	c IM 0	ED 46	2/8		LD C,IYL LD C,n	0E n	2/78		POP DE	D1	1 / 10		RRC D	CB 0A	2/8	- z p r0 - z p r0	SRA (IX+d)	DD CB d 2E	2 / 15 - z p 4 / 23 - z p
The color The	BIT 0,(IX+d) BIT 0,(IY+d)	CB DD 46 d CB FD 46 d	4/20 ?⇔b 4/20 ?⇔b	? - IM 2 ? - IN A,(C)	ED 5E ED 78	2 / 8 2 / 12 16	- z p -	LD D,(IX+d) LD D,(IY+d)	DD 56 d FD 56 d	3 / 19 3 / 19		POP IX POP IY	DD E1 FD E1	2 / 14 2 / 14		RRC H RRC L	CB 0C CB 0D	2/8	- z p r0 - z p r0	SRA A SRA B	CB 2F CB 28	2/8 - z p 2/8 - z p
Mart	BIT 0,A BIT 0,B BIT 0,C	CB 40	2/8 ?⇔b	?- IN B,(C)	ED 40	2 / 12 16 2 / 12 16	- z p -	LD D,B	50	1/4		PUSH BC	C5	1/11		RRD	ED 67	2 / 18	- z p -	SRA D SRA E	CB 2A	2/8 - z p
THE CASE 210 2-10 MILES SEED 1-10 MILES SEED SEED 1-10 MILES S	BIT 0,D BIT 0,E BIT 0.H	CB 43	2/8 ?⇔b	?- IN E,(C)	ED 58	2 / 12 16	- z p -	LD D,E	53	1/4		PUSH IX	DD E5	2 / 15		RST 16 (2)	D7	1 / 11 16		SRA L	CB 2D	2/8 - z p
The color 1.5	BIT 0,L BIT 1,(HL)	CB 45 CB 4E	2/8 ?⇔b 2/12 ?⇔b	?- IN L,(C) ?- INC (HL)	ED 68 34	1 / 11 12	- z p - - z v -	LD D,IYH	FD 54	2/8		RES 0,(IX+d)	DD CB d 86	4/23		RST 32 (4) RST 40 (5)	EF	1 / 11 16		SRL (IX+d) SRL (IY+d)	DD CB d 3E FD CB d 3E	4/23 - z p 4/23 - z p
THE CO. 10 29 29 29 29 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20	BIT 1,(IY+d) BIT 1,A	CB FD 4E d CB 1F	4/20 ?⇔b 2/8 ?⇔b	? - INC (IY+d) ? - INC A	FD 34 d 3C	1/4	- Z V - - Z V -	LD D,IXL LD D,IYL	DD 55 FD 55	2/8 2/8		RES 0,A RES 0,B	CB 87 CB 80	2/8 2/8		RST 56 (7) SBC A,(HL)	FF 9E	1/7	 -zvb	SRL B SRL C	CB 38 CB 39	2/8 - z p 2/8 - z p
Tright Card 24 24 24 24 24 24 24 2	BIT 1,C BIT 1,D	CB 49	2/8 ?⇔b	?- INC BC	3	1/68	BUG	LD DE,(addr)	ED 5B dr ad	4 / 20		RES 0,D	CB 82	2/8		SBC A,(IY+d)	FD 9Ed	3 / 19	- z v b	SRL E	CB 3B	2/8 - z p
	BIT 1,E BIT 1,H BIT 1,L	CB 4C	2/8 ?⇔b	?- INC DE	13	1/68		LD E,(IX+d)	DD 5E d	3 / 19		RES 0,L	CB 85	2/8		SBC A,C	99	1/4	- z v b	STOP	10 00	2/4
MITAL C857 219 79-71 MC MC 120 164 -805 MC 165 MC 1	BIT 2,(HL) BIT 2,(IY+d)	CB FD 56 d	4/20 ?⇔b	?- INC IXH	24 DD 24	2/8	- Z V - - Z V -	LD E,A LD E,B	5F 58	1/4		RES 1,(IX+d) RES 1,(IY+d)	FD CB d 8E	4/23		SBC A,H	9C	1/4	- z v b - z v b	SUB (IX+d) SUB (IY+d)	FD 96 d	3 / 19 - z v 3 / 19 - z v
INCLUDE CRESC 219 ***** ***** ***** **** ***** **** ***** **** ***** ***** ***** ***	BIT 2,A BIT 2,B	CB 57 CB 50	2/8 ?⇔b 2/8 ?⇔b	?- INC HL ?- INC IX	23 DD 23	1/68	BUG	LD E,D LD E,E	5A 5B	1/4 1/4		RES 1,B RES 1,C	CB 88 CB 89	2/8		SBC A,IYH SBC A,L	FD 9C 9D	2/8 1/4	- z v b - z v b	SUB B SUB C	90 91	1/4 - Z V 1/4 - Z V
	BIT 2,D BIT 2,E	CB 52 CB 53	2/8 ?⇔b 2/8 ?⇔b	?- INC L ?- INC IXL	2C DD 2C	1/4 2/8	- Z V -	LD E,IXH	DD 5C FD 5C	2/8 2/8		RES 1,E RES 1,H	CB 8B CB 8C	2/8 2/8		SBC A,IYL SBC A,n	FD 9D DE n	2/8	- z v b	SUB E SUB H	93	1/4 - z v 1/4 - z v
IT	BIT 2,H BIT 2,L BIT 3,(HL)	CB 55	2/8 ?⇔b	?- INC SP	33	1/6		LD E,IXL	DD 5D	2/8		RES 2,(HL)	CB 96	2/15		SBC HL,DE	ED 52	2 / 15	- z v b	SUB IYH	FD AC	2/8 - Z V
## 173.6 C6 58 28 7-697 P(H) E9 1/4 LD H, MY-0 F5 60 2/9 RES 2C C6 91 2/8 SET 0, MY-0 C6 20 40 4/2 SWAP A C8 37 2/4 -2V2 MY-0 MY	BIT 3,(IX+d) BIT 3,(IY+d)	CB FD 5E d	4/20 ?⇔b	?- INI	ED A2	2 / 16 25	?⇔B?-	LD H,(HL)	66	1/7		RES 2,A	CB 97	2/8		SCF	37	1/4		SUB IYL	FD AD	2/8 - z v
INTIAL CB 50 24 7-95*, Jepador C3 24 3195	BIT 3,B BIT 3,C	CB 58 CB 59	2/8 ?⇔b 2/8 ?⇔b	?- JP (HL) ?- JP (IX)	E9 DD E9	2/8		LD H,(IY+d) LD H,A	FD 66 d 67	3/19 1/4		RES 2,C RES 2,D	CB 91 CB 92	2/8 2/8		SET 0,(IX+d) SET 0,(IY+d)	DD CB d C6 FD CB d C6	4/23 4/23		SWAP A SWAP B	CB 37 CB 30	2/8 - Z V 2/8 - Z V
III A	BIT 3,E BIT 3,H	CB 5B CB 5C	2/8 ?⇔b 2/8 ?⇔b	?- JP addr ?- JP c,addr	C3 dr ad DA dr ad	3 / 10 12 3 / 10 12		LD H,C LD H,D	61 62	1/4		RES 2,H RES 2,L	CB 94 CB 95	2/8		SET 0,B SET 0,C	CB C0 CB C1	2/8		SWAP D SWAP E	CB 32 CB 33	2/8 - Z V 2/8 - Z V
ITALIFY C C C C C C C C C	BIT 3,L BIT 4,(HL) BIT 4,(IY+d)	CB 66 CB FD 66 d	2/12 ?⇔b 4/20 ?⇔b	?- JP nc,addr ?- JP nz,addr	D2 dr ad C2 dr ad	3 / 10 12 3 / 10 12		LD H,H LD H,L	64 65	1/4		RES 3,(IX+d) RES 3,(IY+d)	DD CB d 9E FD CB d 9E	4/23 4/23		SET 0,E SET 0,H	CB C3 CB C4	2/8		SWAP L SWAP (HL)	CB 35 CB 36	2/8 - Z V 2/16 - Z V
INT 4.C	BIT 4,(LY+d) BIT 4,A BIT 4,B	CB 67	2/8 ?⇔b	?- JP p,addr ?- JP po,addr	F2 dr ad E2 dr ad	3 / 10 12		LD H,n LD IXH,A	26 n DD 67	2/8		RES 3,A RES 3,B	CB 98	2/8		SET 0,L SET 1,(HL)	CB C5 CB CE	2 / 15		XOR (HL) XOR (IX+d)	AE DD AC d	1/7 - z p 3/19 - z p
IT 4	BIT 4,C BIT 4,D	CB 61 CB 62	2/8 ?⇔b 2/8 ?⇔b	? - JP z,addr ? - JR c,d	CA dr ad 38 d	3 / 10 12 2 / t12f7 8/12		LD IXH,C LD IXH,D	DD 61 DD 62	2/8 2/8		RES 3,D RES 3,E	CB 9A CB 9B	2/8 2/8		SET 1,(IY+d) SET 1,A	FD CB d CE CB CF	4/23 2/8		XOR A XOR B	AF A8	1/4 - z p
ITS (INF-d) CBO DE d 4/20 ?~0-0-7 D (laddr), A 20 20 20 20 20 20 20	BIT 4,H BIT 4,L	CB 64 CB 65	2/8 ?⇔b 2/8 ?⇔b	?- JR nc,d ?- JR nz,d	30 d 20 d	2 / t12f7 8/12 2 / t12f7 8/12		LD IXH,IXH LD IXH,IXL	DD 64 DD 65	2/8 2/8		RES 3,L RES 4,(HL)	CB 9D CB A6	2/8 2/15		SET 1,C SET 1,D	CB C9 CB CA	2/8 2/8		XOR D XOR E	AA AB	1/4 - z p
SIT 5, A CB 6F 2/8 7 - 9-7 LD Gaddry, ID CB 2/8 7 - 9-7 LD Gaddry, ID CB 2/8 7 - 9-7 LD Gaddry, ID CB 2/8 7 - 9-7 LD Gaddry, ID LD Gaddry, ID CB 2/8 7 - 9-7 LD Gaddry, ID CB 2/8 C	BIT 5,(HL) BIT 5,(IX+d) BIT 5,(IY+d)	CB DD 6E d CB FD6E d	4/20 ?⇔b 4/20 ?⇔b	? - LD (addr),A ? - LD (addr),BC	32 drad / EA drad ED 43 drad	3 / 13 4 / 20 24		LD IYH,A LD IYH,B	FD 67 FD 60	2/8 2/8		RES 4,(IY+d) RES 4,A	FD CB d A6 CB A7	4/23 2/8		SET 1,H SET 1,L	CB CC CB CD	2/8		XOR IXH XOR IYH	DD AC FD AD	2/8 - Z p 2/8 - Z p
Strong Color Col	BIT 5,A BIT 5,B BIT 5,C	CB 6F CB 68	2/8 ?⇔b	?- LD (addr),DE ?- LD (addr),HL	22 dr ad	3 / 16 24		LD IYH,D	FD 62	2/8 2/8		RES 4,B RES 4,C	CB A0 CB A1	2/8		SET 2,(HL) SET 2,(IX+d)	DD CB d D6	4/23		XOR IXL	DD AC	1/4 - z p 2/8 - z p
STT 5, L	BIT 5,D BIT 5,E	CB 6A CB 6B	2/8 ?⇔b 2/8 ?⇔b	?- LD (addr),IX ?- LD (addr),IY	DD 22 dr ad FD 22 dr ad	4 / 20 24 4 / 20 24		LD IYH,IYH LD IYH,IYL	FD 64 FD 65	2/8 2/8		RES 4,E RES 4,H	CB A3 CB A4	2/8		SET 2,A SET 2,B	CB D7 CB D0	2/8			EE n	
IT 6, ((((r/4)) CB D) 76 d 4/20 7-6) 2-7 D (((r/4)) CB	BIT 5,H BIT 5,L BIT 6,(HL)	CB 6D CB 76	2/8 ?⇔b 2/12 ?⇔b	? - LD (BC),A ? - LD (DE),A	2	1/7		LD HL,(addr) LD HL,(addr)	2A dr ad ED 6B dr ad	3 / 16 4 / 20		RES 5,(HL) RES 5,(IX+d)	CB AE DD CB d AE	2 / 15 4 / 23		SET 2,D SET 2,E	CB D2 CB D3	2/8 2/8		LD A, (\$FF00+n)	LDH (n),A LDH A,(n)	-
11	BIT 6,(IX+d) BIT 6,(IY+d) BIT 6,A	CB FD 76 d	4/20 ?⇔b	? - ? - LD (HL),A				LD HL,SP+n	F8 n	2 / 12		RES 5,A	CB AF	2/8		SET 2,L	CB D5	2/8		LDI A,(HL)	LD A,(HLI) or LD A,(H	HL+)
SIT 6,	BIT 6,B BIT 6,C	CB 70 CB 71	2/8 ?⇔b 2/8 ?⇔b	? - LD (HL),C ? - LD (HL),D	71 72	1/7		LD IX,(addr) LD IX,hilo	DD 2A dr ad DD 21 lo hi	4 / 20 4 / 14		RES 5,C RES 5,D	CB A9 CB AA	2/8 2/8		SET 3,(IX+d) SET 3,(IY+d)	DD CB d DE FD CB d DE	4/23 4/23		LD A,(C)	LD A,(\$FF00+C)	HLD)
HT 7,(HU) CB 7E 2/12 9-99-7.	BIT 6,D BIT 6,E BIT 6,H	CB 73 CB 74	2/8 ?⇔b 2/8 ?⇔b	? - LD (HL),H ? - LD (HL),L	74 75	1/7		LD IY,hilo LD L,(HL)	FD 21 lo hi 6E	4 / 14 1 / 7		RES 5,H RES 5,L	CB AC CB AD	2/8 2/8		SET 3,B SET 3,C	CB D8 CB D9	2/8 2/8		LD HL,SP+n	LDHL SP,n	
\$\text{\$1T}\$, (\text{V(rt)}) & \text{CB}\$ \text{Total}\$ \text{\$420}\$ \cdot ? \cdot \text{\$e\$} \text{\$1T}\$, \(\text{V(rt)}\) & \text{CB}\$ \text{\$1T}\$, \(\text{V(rt)}\) & \text{\$CB}\$ \text{\$T\$}\$ \\ \text{\$420}\$ \cdot ? \cdot \text{\$e\$}\$ \\ \text{\$1T}\$, \(\text{\$V\$}\) & \text{\$1T}\$, \(\text{\$V\$}\) & \text{\$1T\$}\$, \(\text{\$V\$}\) & \text{\$1T\$}\$	BIT 6,L BIT 7,(HL) BIT 7,(IX+d)	CB 7E CB DD 7E d	2/12 ?⇔b 4/20 ?⇔b	? - LD (\$FF00+C),A ? - LD (\$FF00+n), A	E2 E0 n	2/8 2/12		LD L,(IY+d)	FD 6E d 6F	3/19 1/4		RES 6,(IX+d)	DD CB d B6	4/23 4/23		SET 3,E	CB DB CB DC	2/8		&FE00-&FEFF	will cause sprit	es to flicker
BIT 7,C CB 79 2/8 ? ⇔b? - LD (IX+d),D DD 72 d 3/19 LD L,E 6B 1/4 RES 6,D CB 82 2/8 Blue: Z80 only BIT 7,D CB 7A 2/8 ? ⇔b? - LD (IX+d),E DD 73 d 3/19 LD L,H 6C 1/4 RES 6,E CB 83 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 71 d 3/19 LD L,L 6D 1/4 RES 6,L CB 84 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L D,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L D,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L D,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L D,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L D,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? - LD (IX+d),L DD 75 d 3/19 LD L,L 2E n 2/7 RES 6,L CB 85 2/8 BIT 7,H CB 7C 2/8 ? ⇔b? BIT 7,	BIT 7,(IY+d) BIT 7,A BIT 7,B	CB 7F	2/8 ?⇔b	?- LD (IX+d),B	DD 77 d DD 70 d	3 / 19		LD L,B LD L,C	69	1/4		RES 6,A RES 6,B	CB B0	2/8		SET 3,L	CB DD	2/8				
3HT7,H CB 7C 2/8 ?->b?- LD (IX+d),L DD 75 d 3/19 LD L,n 2En 2/7 RES 6,L CB B5 2/8	BIT 7,C BIT 7,D BIT 7.E	CB 79 CB 7A	2/8 ?⇔b 2/8 ?⇔b	?- LD (IX+d),D ?- LD (IX+d),E	DD 72 d DD 73 d	3 / 19 3 / 19		LD L,E LD L,H	6B 6C	1/4		RES 6,D RES 6,E	CB B2 CB B3	2/8 2/8		Blue: Z80 only Red: GBZ80 on	ıly					
	BIT 7,H BIT 7,L	CB 7C	2/8 ?⇔b	?- LD (IX+d),L	DD 75 d	3 / 19										, p.2. Opt 1111	<u>-</u>			1		

eZ80 Instruction	Opcodes	В/Т	Flags	Instruction	Opcodes	в/т	Flags	Instruction	Opcodes	В/Т	Flags	Instruction	Opcodes	B/T Flags	Instruction	Opcodes	В/Т	Flags	Instruction	Opcodes	В/Т	Flage
ADC A,(HL) ADC A,(IX+d) ADC A,(IY+d)	8E DD 8E d FD 8E d	1/2 3/4 3/4	szhpnc	CALL z,addr CCF CP (HL)			- h - n c	LD (IX+d),DE LD (IX+d),E LD (IX+d),H	DD 1F d DD 73 d DD 71 d	3/5+ 3/4 3/4		LD IY,(HL) LD IY,(IX+d) LD IY,(IX+d)	ED 31 DD 31 d	2/4+ 3/5+ 3/5+	RES 3,A RES 3,B RES 3,C	CB 9F CB 98 CB 99	2/2		SET 1,D SET 1,E SET 1,H	CB CA CB CB CB CC	2/2 2/2 2/2	
ADC A,(IT+U) ADC A,A ADC A,B	8F 88	1/1	* * h p n c	CP (IX+d) CP (IY+d)	DD BE d	3/4 **	ahpac ahpac	LD (IX+d),HL LD (IX+d),IX	DD 2F d DD 3F d	3/5+		LD IY,hilo LD IYH,A	FD 21 lo hi	4/4+ · · · · · · · · · · · · · · · · · · ·	RES 3,D RES 3,E	CB 9A CB 9B			SET 1,H SET 2,(HL)	CB CD CB D6	2/2	
ADC A,C ADC A,D	89 8A 8B	1/4 1/4 1/4	s z h p n c	CP A CP B CP C	BF B8 B9	1/4 **	ahpnc ahpnc ahpnc	LD (IX+d),IY LD (IX+d),L	DD 3E d DD 75 d DD 36 d n	3/5+ 3/4 4/5		LD IYH,B LD IYH,C LD IYH.D	FD 60 FD 61 FD 62	2/2 2/2	RES 3,H RES 3,L	CB 9C CB 9D CB A6	2/2		SET 2,(IY+d)	DD CB d D6 FD CB d D6 CB D7	4/5 4/5 2/2	
ADC A,E ADC A,H ADC A,IXH	8C DD 8C	1/4	* * h p n c	CP D CP E	BA BB	1/4 = 1	hpnc	LD (IX+d),n LD (IY+d),A LD (IY+d),B	FD 77 d FD 70 d	3/4		LD IYH,E LD IYH,IYH		2/2	RES 4,(HL) RES 4,(IX+d) RES 4,(IY+d)	DD CB d A6 FD CB d A6	4/5		SET 2,A SET 2,B SET 2,C	CB D0 CB D1	2/2	
ADC A,IXL ADC A,IYH	DD 8D FD 8C	2/2	szhpnc szhpnc szhpnc	CP H CP IXH CP IXL		2/2	: h p n c : h p n c	LD (IY+d),BC LD (IY+d),C	FD 0F d FD 71 d	3/5+ 3/4 3/4		LD IYH,IYL LD IYH,n	FD 26 n	2/2	RES 4,A RES 4,B	CB A7	212		SET 2,D SET 2,E	CB D2 CB D3	2/2	
ADC A,IYL ADC A,L ADC A,n	FD 8D 8D CE n	2/2 1/4 2/2	szhpnc	CP IYH CP IYL	FD BC	2/2 **	ahpac ahpac	LD (IY+d),D LD (IY+d),DE LD (IY+d),E	FD 72 d FD 1F d FD 73 d	3/5+		LD IYL,A LD IYL,B LD IYL,C	FD 68	2/2	RES 4,C RES 4,D RES 4,E	CB A1 CB A2 CB A3	2/2		SET 2,H SET 2,L SET 3,(HL)	CB D4 CB D5 CB DE	2/2	
ADC HL,BC ADC HL,DE	ED 4A ED 5A	2/2	szhpnc szhpnc szhpnc	CP L CP n		2/2 **	ahpac ahpac ahpa-	LD (IY+d),H LD (IY+d),HL	FD 74 d FD 2F d	3/4		LD IYL,D LD IYL,E		2/2	RES 4,H RES 4,L	CB A4 CB A5	2/2		SET 3,(IY+d)	DD CB d DE FD CB d DE	4/5	
ADC HL,HL ADC HL,SP ADD A,(HL)	ED 6A ED 7A 86	2/2 2/2 1/2	* z h p n c	CPD CPDR CPI	ED A9 ED B9 ED A1	?	. h p n -	LD (IY+d),IX LD (IY+d),IY LD (IY+d),L	FD 3F d FD 3E d FD 75 d	3/5+ 3/5+ 3/4		LD IYL,IYH LD IYL,IYL LD IYL,n	FD 6C FD 6D FD 2E n	2/2 2/2	RES 5,(HL) RES 5,(IX+d) RES 5,(IY+d)	CB AE DD CB d AE FD CB d AE	4/5		SET 3,A SET 3,B SET 3,C	CB DF CB D8 CB D9	2/2 2/2 2/2	
ADD A,(IX+d) ADD A,(IY+d)	DD 86 d FD 86 d	3/4	szhpnc szhpnc szhpnc	CPIR CPL	ED B1 2F	1/1	. h p n - . h - n -	LD (IY+d),n LD A,(addr)	FD 36 d n 3A dr ad	4/5 3/4+		LD L,(HL) LD L,(IX+d)	6E DD 6E d	1/2 3/4	RES 5,A RES 5,B	CB AF	212		SET 3,D SET 3,E	CB DA CB DB	2/2	
ADD A,A ADD A,B ADD A,C	87 80 81	1/1	* z h p n c	DAA DEC (HL) DEC (IX+d)		1/4 **	npn-	LD A,(BC) LD A,(DE) LD A,(HL)	0A 1A 7E	1/2 1/2 1/2		LD L,(IY+d) LD L,A LD L,B	FD 6E d 6F 68	3/4 1/1	RES 5,C RES 5,D RES 5,E	CB A9 CB AA CB AB	2/2		SET 3,H SET 3,L SET 4,(HL)	CB DC CB DD CB E6	2/2 2/2 2/3	
ADD A,D ADD A,E	82 83	1/1	szhpnc szhpnc	DEC (IY+d) DEC A	FD 35 d 3D	1/1 **	: h p n - : h p n -	LD A,(IX+d) LD A,(IY+d)	DD 7E d FD 7E d	3/4		LD L,C LD L,D	69 6A	1/1	RES 5,H RES 5,L	CB AC CB AD	2/2		SET 4,(IY+d)	DD CB d E6 FD CB d E6	4/5	
ADD A,H ADD A,IXH ADD A,IXL	84 DD 84 DD 85	1/1 2/2 2/2	* z h p n c	DEC B DEC BC DEC C	05 0B 0D	1/1	 . h p n -	LD A,A LD A,B LD A,C	7F 78 79	1/1 1/1 1/1		LD L,E LD L,H LD L,L	6B 6C 6D	1/1	RES 6,(IX+d) RES 6,(IX+d)	CB B6 DD CB d B6 FD CB d B6	4/5		SET 4,A SET 4,B SET 4,C	CB E7 CB E0 CB E1	2/2 2/2 2/2	
ADD A,IYH ADD A,IYL	FD 84 FD 85	2/2	szhpnc szhpnc szhpnc	DEC DE	15 1B	1/1	. h p n -	LD A,D LD A,E	7A 7B	1/1		LD L,n LD MB,A	ED 6D	2/2	RES 6,A RES 6,B	CB B7 CB B0	212		SET 4,D SET 4,E	CB E2 CB E3	2/2	
ADD A,L ADD A,n ADD HL,BC	85 C6 n 09	1/1 2/2 1/1	* z h p n c	DEC H DEC HL	1D 25 2B		. h p n -	LD A,H LD A,I LD A,IXH	7C ED 57 DD 7C	1/1 2/2 2/2	8 z h p n -	LD R,A LD SP,(addr) LD SP,hilo	ED 7B dr ad	2/2 4/5 3/3 ^{3/3}	RES 6,C RES 6,D RES 6,E	CB B1 CB B2 CB B3	2/2		SET 4,H SET 4,L SET 5,(HL)	CB E4 CB E5 CB EE	2/2 2/2 2/3	
ADD HL,DE ADD HL,HL	19 29	1/1	h-nc	DEC IX	DD 25	2/2 **	 . h p n -	LD A,IXL LD A,IYH	DD 7D FD 7C	2/2		LD SP,HL LD SP,IX		1/1	RES 6,H RES 6,L	CB B4 CB B5	2/2		SET 5,(IX+d) SET 5,(IY+d)	DD CB d EE FD CB d EE	4/5	
ADD HL,SP ADD IX,BC ADD IX,DE	39 DD 09 DD 19	1/1 2/2 2/2		DEC IXL DEC IY DEC IYH	FD 2B	2/2	 . h p n -	LD A,IYL LD A,L LD A,MB	FD 7D 7D ED 6E	2/2 1/1 2/2		LD SP,IY LDD LDDR	FD F9 ED A8 ED B8	2/2	RES 7,(HL) RES 7,(IX+d) RES 7,(IY+d)	CB BE DD CB d BE FD CB d BE	4/5		SET 5,A SET 5,B SET 5,C	CB EF CB E8 CB E9	2/2 2/2 2/2	
ADD IX,IX ADD IX,SP	DD 29 DD 39	2/2	h-nc	DEC IYL DEC L	2D	1/4 **	thpn-	LD A,n LD A,R	3E n ED 5F		s z h p n -	LDI LDIR	ED B0	2/5 h p n - 2/? h p n -	RES 7,A RES 7,B	CB BF CB B8	212		SET 5,D SET 5,E	CB EA CB EB	2/2	
ADD IY,BC ADD IY,DE ADD IY,IY	FD 09 FD 19 FD 29	2/2 2/2 2/2	h-nc	DEC SP DI DJNZ d	3B F3 10 d	1/1 1/1 2/2-4		LD B,(HL) LD B,(IX+d) LD B,(IY+d)	46 DD 46 d FD 46 d	1/2 3/4 3/4		LEA BC,IX+d LEA BC,IY+d LEA DE,IX+d	ED 02 d ED 03 d ED 12 d	3/3 3/3	RES 7,C RES 7,D RES 7,E	CB B9 CB BA CB BB	2/2		SET 5,H SET 5,L SET 6,(HL)	CB EC CB ED CB F6	2/2 2/2 2/3	
ADD IY,SP AND (HL)	FD 39 A6	2/2 1/2	h-nc	EI EX (SP),HL	FB E3	1/1		LD B,A LD B,C	47 41	1/1		LEA DE,IY+d LEA HL,IX+d	ED 13 d ED 22 d	3/3	RES 7,H RES 7,L	CB BC CB BD	2/2		SET 6,(IX+d) SET 6,(IY+d)	DD CB d F6 FD CB d F6	4/5	
AND (IX+d) AND (IY+d) AND A	DD A6 d FD A6 d A7	2/3 2/3 1/1	* * h p n c * * h p n c * * h p n c * * h p n c * * h p n c * * h p n c * * h p n c *	EX (SP),IX EX (SP),IY EX AF,AF		2/6-8		LD B,D LD B,E LD B.H	42 43 44	1/1 1/1 1/1		LEA HL,IY+d LEA IX,IX+d LEA IX.IY+d	ED 23 d ED 32 d ED 54 d	3/3	RET C RET M	C9 D8 F8	1/5+		SET 6,A SET 6,B SET 6,C	CB F7 CB F0 CB F1	2/2 2/2 2/2	
AND B AND C	A0 A1	1/1		EX DE,HL EXX	EB D9	1/1		LD B,IXH LD B,IXL	DD 44 DD 45	2/2		LEA IY,IX+d LEA IY,IY+d	ED 55 d ED 33 d	3/3	RET NC RET NZ	D0 C0	1/2+		SET 6,D SET 6,E	CB F2 CB F3	2/2	
AND D AND E AND H	A2 A3 A4	1/1 1/1 1/1	* * h p n c * * h p n c * * h p n c * * h p n c * * h p n c * * h p n c * * h p n c *	IM 0 IM 1	76 ED 46 ED 56	1/2		LD B,IYH LD B,IYL LD B.L	FD 44 FD 45 45	2/2 2/2 1/1		MLT BC MLT DE MLT HL	ED 4C ED 5C ED 6C	2/6 ······ 2/6 ······	RET P RET PE RET PO	F0 E8 E0	1/2+		SET 6,H SET 6,L SET 7,(HL)	CB F4 CB F5 CB FE	2/2 2/2 2/3	
AND IXH AND IXL	DD A4 DD A5	2/2	* z h p n c	IM 2 IN A,(BC)	ED 5E ED 78	1/2 2/3	 . h p n -	LD B,n LD BC,(addr)	06 n ED 4B dr ad	2/2 4/6		MLT SP NEG	ED 7C ED 44	2/6 2/2 ******	RET Z RETI	C8 ED 4D	1/2+		SET 7,(IX+d) SET 7,(IY+d)	DD CB d FE FD CB d FE	4/5 4/5	
AND IYH AND IYL AND L	FD A4 FD A5 A5	2/2 2/2 1/1	szhpnc szhpnc szhpnc	IN A,(n) IN B,(BC) IN C.(BC)		2/3	hpn-	LD BC,(HL) LD BC,(IX+d) LD BC,(IY+d)	ED 07 DD 07 d FD 07 d	2/4+ 3/5+ 3/5+		NOP OR (HL) OR (IX+d)	00 B6 DD B6 d	1/1 ***********************************	RETN RL (HL) RL (IX+d)	ED 45 CB 16 DD CB d 16	2/5	zhpac	SET 7,A SET 7,B SET 7,C	CB FF CB F8 CB F9	2/2 2/2 2/2	
AND n BIT 0,(HL)	E6 n CB 46	2/2	8 z h p n c 8 z h p n -	IN D,(BC) IN E,(BC)	ED 50	2/3 = 2	thpn-	LD BC,hilo LD C,(HL)	01 lo hi 4E	3/3		OR (IX+d) OR (IY+d)	FD B6 d B7	3/4 1/1 ********	RL (IY+d) RL A	FD CB d 16 CB 17	4/7 = 2 2/2 = 2	zhpac	SET 7,D SET 7,E	CB FA CB FB	2/2	
BIT 0,(IX+d) BIT 0,(IY+d) BIT 0.A	CB DD 46 d CB FD 46 d CB 47	4/5 4/5 2/2	* * * * * * * * * * * * * * * * * * *	IN H,(BC) IN L,(BC) IN0 A,(n)	ED 68	2/3 **	: h p n - : h p n -	LD C,(IX+d) LD C,(IY+d) LD C,A	DD 4E d DD 4E d 4F	3/4		OR B OR C OR D	B0 B1 B2	1/1 *= hpnc 1/1 *= hpnc 1/1 *= hpnc	RL B RL C RL D	CB 10 CB 11 CB 12	2/2 **	zhpnc zhpnc zhpnc	SET 7,H SET 7,L SLA (HL)	CB FC CB FD CB 26	2/2 2/2 2/5	******
BIT 0,B BIT 0,C	CB 40 CB 41	2/2	8 z h p n - 8 z h p n -	IN0 B,(n) IN0 C,(n)	ED 00 ED 08	2/4 = 2	hpn- hpn-	LD C,B LD C,D	48 4A	1/1		OR E OR H	B3 B4	1/1 * = h p n c 1/1 * = h p n c	RL E RL H	CB 13 CB 14	2/2 = 2	zhpnc	SLA (IX+d) SLA (IY+d)	DD CB d 26 FD CB d 26	4/7	szhpnc szhpnc
BIT 0,D BIT 0,E BIT 0.H	CB 42 CB 43 CB 44	2/2 2/2 2/2	8 z h p n - 8 z h p n - 8 z h p n -	INO D,(n) INO E,(n) INO H,(n)		2/4	: h p n - : h p n -	LD C,E LD C,H LD C,IXH	4B 4C DD 4C	1/1 1/1 2/2		OR IXH OR IXL OR IYH	DD B4 DD B5 FD B4	2/2 ** hpnc 2/2 ** hpnc 2/2 ** hpnc	RL L RLA RLC (HL)	CB 15 17 CB 06	1/1	2 h p n c - h - n c 2 h p n c	SLA A SLA B SLA C	CB 27 CB 20 CB 21	2/2	szhpnc szhpnc szhpnc
BIT 0,L BIT 1,(HL)	CB 45 CB 4E	2/2 2/3	s z h p n -	INO L,(n) INC (HL)		2/4 ***	hpn-	LD C,IXL LD C,IYH	DD 4D FD 4C	2/2		OR IYL OR L		2/2 ***********************************	RLC (IX+d) RLC (IY+d)	DD CB d 06 FD CB d 06	4/7 = 3	zhpnc	SLA D SLA E	CB 22 CB 23	2/2	szhpnc szhpnc
BIT 1,(IX+d) BIT 1,(IY+d)	CB DD 4E d CB FD 4E d		8 z h p n - 8 z h p n - 8 z h p n -	INC (IX+d) INC (IY+d) INC A	DD 34 d FD 34 d 3C	3/6 **	: h p n - : h p n -	LD C,IYL LD C,L	FD 4D 4D	2/2 1/1 2/2		OR n OTD2R OTDM	F6 n ED BC ED 8B	2/2 ***********************************	RLC A RLC B	CB 07 CB 00	2/2 **	zhpnc zhpnc zhpnc	SLA H SLA L SLP	CB 24 CB 25	212	szhpac
BIT 1,A BIT 1,B BIT 1,C	CB 1F CB 48 CB 49	2/2	8 z h p n - 8 z h p n -	INC B INC BC	04 03	1/1 **	. h p n -	LD C,n LD D,(HL) LD D,(IX+d)	0E n 56 DD 56 d	1/2		OTDMR OTDR	ED 9B ED BB	2/?	RLC C RLC D RLC E	CB 01 CB 02 CB 03	2/2 **	zhpnc	SRA (HL) SRA (IX+d)	ED 76 CB 2E DD CB d 2E		szhpac szhpac
BIT 1,D BIT 1,E BIT 1,H	CB 4A CB 4B CB 4C	2/2 2/2 2/2	8 z h p n - 8 z h p n - 8 z h p n -	INC C INC D INC DE	0C 14 13	1/1 **	: h p n -	LD D,(IY+d) LD D,A LD D.B	FD 56 d 57 50	3/4 1/1 1/1		OTDRX OTI2R OTIM	ED CB ED B4 ED 83	2/?	RLC H RLC L RLCA	CB 04 CB 05 07	2/2 **	zhpac zhpac -h-ac	SRA (IY+d) SRA A SRA B	CB 2F CB 28	2/2	szhpnc szhpnc szhpnc
BIT 1,L BIT 2,(HL)	CB 4D CB 56	2/2 2/3	s z h p n -	INC E INC H	1C 24	1/1 * 3	hpn-	LD D,B LD D,C LD D,E	51 53	1/1		OTIMR OTIR	ED 93 ED B3	2/?	RLD RR (HL)	ED 6F CB 1E	2/5 * 2	zhpnc	SRA C SRA D	CB 29 CB 2A	2/2	szhpnc szhpnc
BIT 2,(IX+d) BIT 2,(IY+d) BIT 2,A	CB DD 56 d CB FD 56 d CB 57		8 z h p n - 8 z h p n - 8 z h p n -	INC HL INC IX INC IXH				LD D,H LD D,IXH LD D,IXL	54 DD 54 DD 55	1/1 2/2 2/2		OTIRX OUT (BC),A OUT (BC),B	ED C3 ED 79 ED 41	2/?	RR (IX+d) RR (IY+d) RR A	DD CB d 1E FD CB d 1E CB 1F	4/7 * *	zhpac zhpac zhpac	SRA E SRA H SRA L	CB 2B CB 2C CB 2D		szhpnc szhpnc szhpnc
BIT 2,B BIT 2,C	CB 50 CB 51	2/2	8 z h p n - 8 z h p n -	INC IXL INC IY	DD 2C FD 23	2/2 = 2/2	. h p n -	LD D,IYH LD D,IYL	FD 54 FD 55	2/2		OUT (BC),C OUT (BC),D	ED 49 ED 51	2/3	RR B RR C	CB 18 CB 19	2/2 ***	zhpnc zhpnc	SRL (HL) SRL (IX+d)	CB 3E DD CB d 3E	2/5	szhpnc szhpnc
BIT 2,D BIT 2,E BIT 2,H	CB 52 CB 53 CB 54	2/2	8 z h p n - 8 z h p n -	INC IYH INC IYL INC L		2/2	: h p n - : h p n -	LD D,L LD D,n LD DE.(addr)	55 16 n ED 5B dr ad	2/2		OUT (BC),E OUT (BC),H OUT (BC),L		2/3 2/3	RR D RR E RR H	CB 1A CB 1B CB 1C	2/2 **	zhpac zhpac zhpac	SRL (IY+d) SRL A SRL B	CB 3F CB 38	2/2	szhpnc szhpnc szhpnc
BIT 2,L BIT 3,(HL)	CB 55	2/2	szhpn-	INC SP	33	1/1		LD DE,(HL) LD DE,(IX+d)	ED 17	2/4+3/5+		OUT (n),A OUT0 (n),A	D3 n	2/3	RR L RRA	CB 1D 1F	2/2 **	2 h p n c	SRL C SRL D	CB 39 CB 3A	2/2	szhpnc szhpnc
BIT 3,(IX+d) BIT 3,(IY+d) BIT 3,A	CB DD 5E d CB FD 5E d CB 5F	4/5	s z h p n - s z h p n -	IND2 IND2R INDM	ED 9C	2/?		LD DE,(IY+d) LD DE,hilo LD E,(HL)	FD 17 d 11 lo hi 5E	3/5+ 3/3 1/2		OUT0 (n),B OUT0 (n),C OUT0 (n),D	ED 09 n	3/4 ······ 3/4 ·····	RRC (HL) RRC (IX+d) RRC (IY+d)	CB 0E DD CB d 0E FD CB d 0E	4/7 **	zhpnc zhpnc zhpnc	SRL E SRL H SRL L	CB 3B CB 3C CB 3D	2/2	szhpnc szhpnc szhpnc
BIT 3,B BIT 3,C	CB 58 CB 59	2/2	szhpn-	INDMR INDR	ED 9A ED BA	2/? -		LD E,(IX+d) LD E,(IY+d)	DD 5E d FD 5E d	3/4		OUT0 (n),E OUT0 (n),H	ED 19 n	3/4	RRC A RRC B	CB 0F CB 08	2/2 ***	zhpnc	STMIX SUB (HL)	ED 7D 96	2/2	szhpnc
BIT 3,D BIT 3,E BIT 3.H	CB 5A CB 5B CB 5C	2/2	s z h p n - s z h p n - s z h p n -	INDRX INI INI2	ED A2	5 -	5 -	LD E,A LD E,B LD E.C	5F 58 59	1/1		OUTO (n),L OUTD OUTD2		3/4	RRC C RRC D RRC E	CB 09 CB 0A CB 0B	2/2 **	zhpac zhpac	SUB (IX+d) SUB (IY+d) SUB A	DD 96 d FD 96 d 97	2/4	szhpnc szhpnc szhpnc
BIT 3,L BIT 4,(HL)	CB 5D CB 66	2/22/3	8 z h p n -	INI2R INIM	ED 94 ED 82	2/? - 3		LD E,D LD E,H	5A 5C	1/1		OUTI OUTI2	ED A3 ED A4	2/5	RRC H RRC L	CB 0C CB 0D	2/2 ***	zhpnc	SUB B SUB C	90 91	1/1	* * h p n c
BIT 4,(IX+d) BIT 4,(IY+d) BIT 4,A	CB DD 66 d CB FD 66 d CB 67	4/5	8 z h p n - 8 z h p n -	INIMR INIR INIRX	ED B2	2/?		LD E,IXH LD E,IXL LD E,IYH	DD 5C DD 5D FD 5C	2/2		PEA IX+d PEA IY+d POP AF		3/5+ 3/5+ 1/3+ Popped	RRCA RRD RSMIX	0F ED 67 ED 7E	2/5 **	- h - n c	SUB D SUB E SUB H	92 93 94	1/1	szhpnc szhpnc szhpnc
BIT 4,B BIT 4,C	CB 60 CB 61	2/2	szhpn-	JP (HL) JP (IX)	E9 DD E9	1/3		LD E,IYL LD E,L	FD 5D 5D	2/2		POP BC POP DE	C1 D1	1/3+	RST 0 RST 16 (2)	C7 D7	1/5+		SUB IXH SUB IXL	DD AC DD AD	2/2	szhpnc szhpnc
BIT 4,D BIT 4,E BIT 4.H	CB 62 CB 63 CB 64	2/2	8 z h p n - 8 z h p n -	JP (IY) JP addr JP c.addr	C3 dr ad	2/4+		LD E,n LD H,(HL) LD H.(IX+d)	1E n 66 DD 66 d			POP HL POP IX POP IY	DD E1	1/3+ · · · · · · · · · · · · · · · · · · ·	RST 24 (3) RST 32 (4) RST 40 (5)	DF E7 EF	1/5+		SUB IYH SUB IYL SUB L	FD AC FD AD 95	2/2	szhpnc szhpnc -zvb
BIT 4,L BIT 5,(HL)	CB 65 CB 6E	2/2	szhpn-	JP m,addr JP nc,addr	FA dr ad	3/3+		LD H,(IY+d) LD H,A	FD 66 d 67	3/4		PUSH AF PUSH BC	F5	1/3+	RST 48 (6) RST 56 (7)	F7 FF	1/5+		SUB n TST A,(HL)	D6 n ED 34	2/2	s z h p n c
BIT 5,(IX+d) BIT 5,(IY+d) BIT 5,A	CB DD 6E d CB FD 6E d CB 6F	4/5	s z h p n - s z h p n -	JP nz,addr JP p,addr JP pe,addr	F2 dr ad	3/3+		LD H,B LD H,C LD H,D	60 61 62	1/1		PUSH DE PUSH HL PUSH IX	E5	1/3+ · · · · · · · · · · · · · · · · · · ·	RST 8 (1) SBC A,(HL) SBC A,(IX+d)	CF 9E DD 9Ed	1/5+ 1/2 1/2 3/4 1/2	 zhpnc	TST A,A TST A,B TST A,C	ED 3C ED 04 ED 0C	2/2	azhpac azhpac azhpac
BIT 5,B BIT 5,C	CB 68 CB 69	2/2	szhpn-	JP po,addr JP z,addr	E2 dr ad CA dr ad	3/3+		LD H,E LD H,H	63 64	1/1		PUSH IY RES 0,(HL)	FD E5 CB 86	2/4+	SBC A,(IX+d) SBC A,(IY+d) SBC A,A	FD 9Ed 9F	3 /4 ***	zhpnc	TST A,D TST A,E	ED 14 ED 1C	2/2	** * * * * * * * * * * * * * * * * * * *
BIT 5,D BIT 5,E BIT 5.H	CB 6A CB 6B CB 6C	2/2	s z h p n - s z h p n -	JR c,d JR d JR nc.d	18 d	2/3		LD H,L LD H,n	65 26 n 2A dr ad	2/2		RES 0,(IX+d) RES 0,(IY+d) RES 0.A	FD CB d 86	4/5 4/5	SBC A,B SBC A,C SBC A.D	98 99 9A	1/1 **	zhpac zhpac zhpac	TST A,H TST A,L TST A.n	ED 24 ED 2C ED 64 n	2/2	szhpnc szhpnc szhpnc
BIT 5,L BIT 6,(HL)	CB 6D CB 76	2/2	szhpn-	JR nz,d JR z,d	20 d	2/2+		LD HL,(addr) LD HL,(HL) LD HL,(IX+d)	ED 27	2/4+ 3/5+		RES 0,B RES 0,C	CB 80	2/2	SBC A,E SBC A,H	9B 9C	1/1 **	zhpnc zhpnc	TSTIO n XOR (HL)	ED 74 n AE	3/4	s z h p n c
BIT 6,(IX+d) BIT 6,(IY+d) BIT 6,A	CB DD 76 d CB FD 76 d CB 77	4/5	szhpn-	LD (addr),A LD (addr),BC LD (addr),DE	ED 43 dr ad	4/6		LD HL,(IY+d) LD HL,hilo LD HL,I	FD 27 d 21 lo hi ED D7	3/3		RES 0,D RES 0,E RES 0,H		2/2 2/2 2/2	SBC A,IXH SBC A,IXL SBC A,IYH	DD 9C DD 9D FD 9C	2/2 **	zhpnc zhpnc zhpnc	XOR (IX+d) XOR (IY+d) XOR A	DD AC d FD AC d AF	3/4	szhpnc szhpnc szhpnc
BIT 6,B BIT 6,C	CB 70 CB 71	2/2	szhpn-	LD (addr),HL LD (addr),IX	22 dr ad DD 22 dr ad	3/5		LD I,A LD I,HL	ED 47 ED C7	2/2		RES 0,L RES 1,(HL)	CB 85 CB 8E	2/2	SBC A,IYL SBC A,L	FD 9D 9D	2/2 ***	zhpnc zhpnc	XOR B XOR C	A8 A9	1/1	szhpnc
BIT 6,D BIT 6,E BIT 6,H	CB 72 CB 73 CB 74	2/2	s z h p n - s z h p n -	LD (addr),IY LD (addr),SP LD (BC),A		4/6		LD IX,(addr) LD IX,(HL) LD IX,(IX+d)		4/6 2/4+ 3/5+		RES 1,(IX+d) RES 1,(IY+d) RES 1,A	FD CB d 8E	4/5 4/5	SBC A,n SBC HL,BC SBC HL,DE	DE n ED 42 ED 52	2/2 **	zhpnc zhpnc zhpnc	XOR D XOR E XOR H	AA AB AC	1/1	szhpnc szhpnc szhpnc
BIT 6,L BIT 7,(HL)	CB 75 CB 7E	2/2	szhpn-	LD (DE),A LD (HL),A	12 77	1/2 -		LD IX,(IY+d) LD IX,hilo	FD 37 d DD 21 lo hi	3/5+ 4/4+		RES 1,B RES 1,C	CB 88 CB 89	2/2	SBC HL,HL SBC HL,SP	ED 62 ED 72	2/2 = 2	zhpnc zhpnc	XOR IXH XOR IXL	DD AC DD AC	2/2	szhpnc
BIT 7,(IX+d) BIT 7,(IY+d) BIT 7,A	CB DD 7E d CB FD 7E d CB 7F	4/5 2/2	s z h p n -	LD (HL),B LD (HL),BC LD (HL),C	70 ED 0F 71	2/4+		LD IXH,A LD IXH,B LD IXH,C	DD 67 DD 60 DD 61	2/2		RES 1,D RES 1,E RES 1,H	CB 8B	2/2 2/2	SCF SET 0,(HL) SET 0,(IX+d)	37 CB C6 DD CB d C6	2/3	- 5 - 5 0	XOR IYH XOR IYL XOR L	FD AD FD AD AD	2/2	azhpnc azhpnc azhpnc
BIT 7,B BIT 7,C	CB 78 CB 79	2/2	szhpn-	LD (HL),D LD (HL),DE	72 ED 1F	1/2		LD IXH,D LD IXH,E	DD 62 DD 63	2/2		RES 1,L RES 2,(HL)	CB 8D CB 96	2/2	SET 0,(IY+d) SET 0,A	FD CB d C6 CB C7	4/5 2/2		XOR n	EE n	2/2	szhpnc
BIT 7,D BIT 7,E BIT 7,H	CB 7A CB 7B CB 7C	2/2	8 z h p n - 8 z h p n -	LD (HL),E LD (HL),H LD (HL),HL	74	1/2		LD IXH,IXH LD IXH,IXL LD IXH,n	DD 64 DD 65 DD 26 n	2/2		RES 2,(IX+d) RES 2,(IY+d) RES 2,A	FD CB d 96	4/5 4/5 2/2	SET 0,B SET 0,C SET 0,D	CB C0 CB C1 CB C2	2/2		xxx.S xxx.L	52 xxx 49 xxx	1/1	
BIT 7,L CALL addr	CB 7D CD dr ad	2/2 3/5	s z h p n -	LD (HL),IX LD (HL),IY	ED 3F ED 3E	2/4+		LD IXL,A LD IXL,B	DD 6F DD 68	2/2		RES 2,B RES 2,C	CB 90 CB 91	2/2	SET 0,E SET 0,H	CB C3 CB C4	2/2		.IS .IS ADL	40 49	1/1	
CALL c,addr CALL m,addr CALL nc,addr	FC dr ad	3/6+ 3/6+ 3/6+		LD (HL),L LD (HL),n LD (IX+d),A	DD 77 d	2/3		LD IXL,C LD IXL,D LD IXL,E	DD 69 DD 6A DD 6B	2/2		RES 2,D RES 2,E RES 2,H	CB 93 CB 94	2/2 2/2	SET 0,L SET 1,(HL) SET 1,(IX+d)	CB C5 CB CE DD CB d CE	2/3		IL ADL	52 5B	1/1	
CALL p,addr CALL p,addr CALL pe,addr	C4 dr ad	3 / 6+ 3 / 6+		LD (IX+d),B LD (IX+d),BC LD (IX+d),C	DD 70 d DD 0F d	3/4		LD IXL,IXH LD IXL,IXL LD IXL.n	DD 6C DD 6D	2/2		RES 2,L RES 3,(HL) RES 3,(IX+d)	CB 95 CB 9E	2/2	SET 1,(IY+d) SET 1,A SET 1.B	CB CF CB C8	2/2		.SIS .LIL SLL is remove	40 5B	1/1	
CALL pe,addr CALL po,addr	E4 dr ad			LD (IX+d),C LD (IX+d),D		374		LD IXL,n LD IY,(addr)	FD 2A dr ad			RES 3,(IX+d)		4.70	SET 1,B	CB C9			LD B,B C,C D,		ved	

6502				
Mnemonic	Description	Example	Addressing Modes	Flags
ADC <ea></ea>	Add <ea> and the carry flag to the Accumulator A.</ea>	ADC #61	Imm ; ZeroPg ; ZeroPg,X ; Abs ; Abs,X ;	N Z C V
ADO (car	Add sear and the early may to the Accumulator A.	ADO #01	Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	1
AND <ea></ea>	Logical AND of bits in 8 bit value <ea> with Accumulator</ea>	AND \$12	Accum; ZeroPg; ZeroPg,X; Abs; Abs,X	N Z C
ASL <ea></ea>	Shift <ea> Left for Arithmetic.</ea>	ASL	Accum; ZeroPg; ZeroPg,X; Abs; Abs,X	
Bcc ofst	Branch to the 8 bit offset ofst IF condition cc is true.	BEQ TestLal		
BIT <ea></ea>	Test bits in Accumulator compared to <ea></ea>	BIT \$61	Imm {65c02}; ZeroPg; ZeroPg,X {65c02}; Abs; Abs,X {65c02}	N Z V
BRK	Stop the CPU and execute an interrupt.	BRK	, , ,	I
CLC	Clear the Carry Flag. C flag will be set to Zero.	CLC		C
CLD	Clear the Decimal Flag. (BCD off)	CLD		D -
CLI	Clear the Interrupt Flag. (Enable Interrupts)	CLI		I
CLV	Clear the oVerflow Flag. V flag will be set to Zero.	CLV		V
CMP <ea></ea>	Compare the Accumulator to <ea>.</ea>	CMP #10	Imm; ZeroPg; ZeroPg,X; Abs; Abs,X;	N Z C
ODV 1		057/ //40	Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	
CPX <ea></ea>	Compare the X register to <ea>.</ea>	CPX #10	Imm ; ZeroPg ; Abs	N Z C
CPY <ea></ea>	Compare the Y register to <ea>.</ea>	CPY #10	Imm; ZeroPg; Abs	N Z C
DEC <ea></ea>	Decrease the 8 bit value <ea> by one.</ea>	DEC \$10	Accum {65c02}; ZeroPg ; ZeroPg,X ; Abs ; Abs,X	
DEX	Decrease register X by one.	DEX		N Z
DEY	Decrease register Y by one.	DEY		N Z
EOR <ea></ea>	Logical EOR (Exclusive OR) of bits in <ea> with A</ea>	EOR <ea></ea>	Imp; ZeroPg; ZeroPg,X; Abs; Abs,X; Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	N Z
INC <ea></ea>	Increase the 8 bit value <ea> by one.</ea>	INC \$10	Accum {65c02}; ZeroPg ; ZeroPg,X ; Abs ; Abs,X	N Z
INX	Increase register X by one.	INX		N Z
INY	Increase register Y by one.	INY		N Z
JMP addr	Jump to the 16 bit address addr.	JMP \$4000	Abs ; (Ind Abs,X) {65c02} ; (Ind)	
JSR addr	Jump to Subroutine at address addr.	JSR addr	Abs	
LDA <ea></ea>	Load the 8 bit value from <ea> into the Accumulator.</ea>	LDA #100	Imm; ZeroPg; ZeroPg,X; Abs; Abs,X; Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	N Z
LDX <ea></ea>	Load the 8 bit value from <ea> into the X register.</ea>	LDX #100	Imm ; ZeroPg ZeroPg,Y ; Abs ; Abs,Y	N Z
LDY <ea></ea>	Load the 8 bit value from <ea> into the Y register.</ea>	LDY #100	Imm ; ZeroPg ZeroPg,X ; Abs ; Abs,X	N Z
LSR <ea></ea>	Shift the bits of <ea> Right Logically.</ea>	LSR \$1000	Accum; ZeroPg; ZeroPg,X; Abs; Abs,X	N Z C
NOP	No Operation.	NOP		
ORA <ea></ea>	Logical OR of bits in 8 bit value <ea> with Accumulator</ea>	ORA #61	Imm; ZeroPg; ZeroPg,X; Abs; Abs,X; Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	N Z C V
PHA	Push a byte from register A onto the top of the stack.	PHA		
PHP	Push the flags (P) onto the stack.	PHP		
PLA	Pull a byte off the stack into register A.	PLA		
PLP	Pull a byte off the stack into register A.	PLP		NZCIDV
ROL <ea></ea>	Rotate bits of <ea> Left with the Carry.</ea>	ROL \$40	Accum; ZeroPg; ZeroPg,X; Abs; Abs,X	N Z C
ROR <ea></ea>	Rotate bits of <ea> Right with the Carry.</ea>	ROR \$40	Accum; ZeroPg; ZeroPg,X; Abs; Abs,X	N Z C
RTI	Return from an interrupt.	RTI		NZCIDV
RTS	Return from a subroutine.	RTS	Insure - Zeve Der - Zeve Der V - Al - Al - V	
SBC <ea></ea>	Subtract <ea> and the carry flag from the Accumulator</ea>	SBC #61	Imm; ZeroPg; ZeroPg,X; Abs; Abs,X; Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	N Z C V
SEC	Set the carry flag to 1.	SEC		C
SED	Set the Decimal Flag. (BCD on)	SED		D -
SEI	Set the Interrupt Flag.	SEI		I
STA <ea></ea>	Store the Accumulator into memory address <ea>.</ea>	STA \$10	ZeroPg; ZeroPg,X; Abs; Abs,X; Abs,Y; (ind) {65c02}; (Ind,X), (Ind),Y	
STX <ea></ea>	Store the X register into memory address <ea>.</ea>	STX \$10	ZeroPg; ZeroPg,Y; Abs	
STY <ea></ea>	Store the Y register into memory address <ea>.</ea>	STY \$10	ZeroPg ; ZeroPg,X ; Abs	
TAX	Transfer the Accumulator into register X.	TAX		N Z
TAY	Transfer the Accumulator into register Y.	TAY		N Z
TSX	Transfer the Stack pointer into register X.	TSX		N Z
TXA	Transfer the X register into the Accumulator.	TXA		N Z
TXS	Transfer the X Register into the Stack pointer.	TXS		
TYA	Transfer the Y register into the Accumulator.	TYA		N Z
Mnomonio	Description	Evample	Addressing Mades	El age
Mnemonic BRA ofst	Description Pranch always to the 8 hit offset ofst (without condition)	Example BRA TestLat	Addressing Modes	Flags
PHY	Branch always to the 8 bit offset ofst (without condition).	PHY	UGI	

Mnemonic	Description	Example Addressing Modes	Flags
BRA ofst	Branch always to the 8 bit offset ofst (without condition).	BRA TestLabel	
PHX	Push a byte from register X onto the top of the stack.	PHX	
PHY	Push a byte from register Y onto the top of the stack.	PHY	
PLY	Pull a byte off the stack into register Y.	PLY	
STZ <ea></ea>	Clear the 8 bit value in memory address <ea>.</ea>	STZ \$1000	

Mnemonic	Description	Condition
BCC label	Branch to labe	C=0
BCS label	Branch to labe	C=1
BEQ label	Branch to labe	Z=1
BNE label	Branch to labe	Z=0
BMI label	Branch to labe	N=1
BPL label	Branch to labe	N=0
BVC label	Branch to labe	V=0
BVS label	Branch to labe	V=1

Bit	Flag	Name	Description
7	N	Negative	Positive / Negative
6	V	Overflow	1=True. (sign changed)
5	-		unused
4	В	Break	Interrupt was a break (in pushed flags only)
3	D	Decimal mode	1=Binary Coded Decimal mode
2	1	IRQ	1=Disable Interrupts
1	Z	Zero	Zero Flag (1=zero)
0	С	Carry	Carry (1=Carry one) / Borrow (0=Borrow one)

From	То	Purpose
\$0000	\$00FF	Zero Page
\$0100	\$01FF	Stack
\$FFFA	\$FFFB	NMI: Non Maskab
\$FFFC	\$FFFD	Reset Vector
\$FFFE	\$FFFF	IRQ: Interrupt Ved

650	2 – 65816 – 6280																							
		Implied	Relative	Accum	Immediate	Zero Page	Zero Pg,X	Zero PG,Y		Absolute,X	Absolute,Y	Abs,X Indir	Indirect	(Indirect,X)	(Indirect),Y	CZIDBVN	Absolute Long	Abs Indir Long	Direct Pg Indirect	Direct Pg Ind Lng	Abs Long,X	(Long Indirect),Y	Stack Relative	SR Indirect Indexed
		no params	jr	works on A	#nn &nn	\$nn (&00nn)	\$nn,X (&00nn+X)	\$nn,Y (&00nn+Y)	\$0100 (&0100)	\$0100,X (&0100+X)	\$0100,Y (&0100+Y)	(\$nnnn,X) ((&nnnn +X))	(\$nnnn) ((&nnnn))	(\$nn,X) ((&00nn+X))	(\$nn),Y ((&00nn)+Y)		\$100000 (\$100000)	[\$1000] ((\$1000))	(\$nn) (\$dpnn)	[\$nn] (\$dpdpnn)	\$010000,X (&010000+X)	[\$nn],Y ((&dpdpnn)+Y)	\$88,S (\$88+S)	(\$88,S),Y ((\$88,S)+Y)
ADC AND	Add with Carry Logical AND				\$69 2 2 \$29 2 2	\$65 2 3 \$25 2 3	\$75 2 4 \$35 2 4		\$6D 3 4 \$2D 3 4	\$7D 3 4/5 \$3D 3 4/5			\$72 3 \$32 3	\$61 2 6 \$21 2 6	\$71 2 5/6 \$31 2 5/6	Ovf7 2 +- 7	\$6F 3 4 \$2F 4 5		\$72 2 5 \$32 2 5	\$67 2 6 \$27 2 4	\$7F 4 5 \$3F 4 5	\$77 2 6 \$37 2 6	\$63 2 4 \$23 2 4	\$73 2 7 \$33 2 7
ASL	Arithmetic Shift Left			\$0A 1 2	\$29 2 2	\$06 2 5	\$16 2 6		\$2D 3 4 \$0E 3 6	\$1E 3 7	\$39 3 4/5		\$32.3	\$2120	\$31 2 5/6	7 Z 7	\$2F 4 5		\$32.25	\$21 2 4	\$3F 4 5	\$37.2.0	\$23 2 4	\$33.2.7
BCC BCS	Branch if Carry Clear C=1 (Aka BLT) Branch if Carry Set C=0 (Aka BGE)		\$90 2 2-4 \$B0 2 2-4																					
BEQ	Branch if Equal to Zero		\$F0 2 2-4																					
BIT BMI	Bit Test (set flags like AND) Branch if Minus (S = 1)		\$30 2 2-4		\$89 2	\$24 2 3	\$34 2		\$2C 3 4	\$3C 3						- z 6 7								
BNE	Branch if Not Equal to Zero		\$D0 2 2-4																					
BPL BRK	Branch if Plus (S = 0) Break	\$00 1 7	\$10 2 2-4													=1								
BVC	Branch if Overflow Clear	400 1 7	\$50 2 2-4																					
BVS	Branch if Overflow Set Clear Carry Flag		\$70 2 2-4													=0								
CLD	Clear Decimal Mode	\$D8 1 2														=0								
CLV	Clear Interrupt Mask (Enable Interrupts) Clear Overflow Flag	\$58 1 2 \$B8 1 2														=0								
CMP	Compare Accumulator to Memory	\$50 T Z			\$C9 2 2	\$C5 2 3	\$D5 2 4		\$CD 3 4	\$DD 3 4/5	\$D9 3 4/5		\$D2 3	\$C1 2 6	\$D1 2 5/6		\$CF 4 5		\$D2 2 5	\$C7 2 6	\$DF 4 5	\$D7 2 6	\$C3 2 4	\$D3 2 7
CPX	Compare with Index Register X Compare with Index Register Y				\$E0 2 2 \$C0 2 2	\$E4 2 3 \$C4 2 3			\$EC 3 4 \$CC 3 4							> = 7								
DEC	Decrement (Aka DEA)			\$3A	\$C0 2 2	\$C6 2 5	\$D6 2 6		\$CE 3 6	\$DE 3 7						- z 7								
DEX	Decrement Index Register X Decrement Index Register Y	\$CA 1 2 \$88 1 2														- z 7 - z 7								
DEY	Logical Exclusive-OR (XOR)	\$00 T Z			\$49 2 2	\$45 2 3	\$55 2 4		\$4D 3 4	\$5D 3 4/5	\$59 3/4/5		\$52.3	\$4126	\$51 2 5/6	- z 7	\$4F 4 5		\$52 2 5	\$47 2 6	\$5F 4 5	\$57 2 6	\$43 2 4	\$53 2 7
INC	Increment (Aka INA)			\$1A		\$E6 2 5	\$F6 2 6		\$EE 3 6	\$FE 3 7						- z 7 - z 7								
INX	Increment Index Register X Increment Index Register Y	\$E8 1 2 \$C8 1 2														- 2 7								
JMP	Jump to New Location (or JML for long)								\$4C 3 3			\$7C 3	\$6C 3.5				\$5C 4 4	\$DC 36						
JSR	Jump to Subroutine (or JSL for long)						an- :		\$20 3 6	\$FC 3 8	ans -			04:	05:	7	\$22 4 8		05.		an-	000	0.4.7	up.
LDA LDX	Load Accumulator Load Index Register X				\$A9 2 2 \$A2 2 2	\$A5 2 3 \$A6 2 3	\$B5 2 4	\$B6 2 4	\$AD 3 4 \$AE 3 4	\$BD 3 4/5	\$B9 3 4/5 \$BE 3 4/5		\$B2 3	\$A1 2 6	\$B1 2 5	- 2 7	\$AF 4 5		\$B2 2 5	\$A7 2 6	\$BF 4 5	\$B7 2 6	\$A3 2 4	#B3 2 7
LDY	Load Index Register Y			6.1		\$A4 2 3	\$B4 2 4		\$AC 3 4	\$BC 3 4/5						- z 7								
LSR NOP	Logical Shift Right No Operation	\$EA 1 2		\$4A 1 2		\$46 2 5	\$56 2 6		\$4E 3 6	\$5E 3 7						- z 7								
ORA	Logical (Inclusive) OR				\$09 2 2	\$05 2 3	\$15 2 4		\$0D 3 4	\$1D 3 4/5	\$19 3 4/5		\$12.3	\$01 2 6	\$11 2 5/6	- z 7	\$0F 4 5		\$12 2 5	\$07 2 6	\$1F 4 5	\$07 2 6	\$03 2 4	\$13 2 7
PHA	Push Accumulator onto Stack Push Processor Status	\$48 1 3 \$08 1 3																						
PLA	Pull Accumulator from Stack	\$68 1 4														- z 7								
PLP ROL	Pull Processor Status Rotate Left through Carry	\$28 1 4		\$2A 1 2		\$26 2 5	\$36 2 6		\$2E 3 6	\$3E 3 7						S S S S S S S Old7 2 7								
ROR	Rotate Right through Carry			\$6A 1 2		\$66 2 5	\$76 2 6		\$6E 3 6	\$7E 3 7						old0 Z 7								
RTI RTS	Return from Interrupt (RETI) Return from Sub (RET) (or RTL for long)	\$40 1 6 \$60 1 6														8888888								
	Subtract with Carry	\$6B 1 6			\$E9 2 2	\$E5 2 3			\$ED 3.4	\$FD 3 4/5	SEO 3 1/5		\$F23	\$E126	\$F1 2 5/6	Ovf7 2 7	\$EF 4 5		\$E2.25	\$E7 2 6	SEE 4.5	\$F7 2 6	\$E3 2 4	\$E3.2.7
SBC	Set Carry (SCF)	\$38 1 2			φE9 Z Z	φE3 2 3			\$ED 3 4	фгD 3 4/3	φr 9 3 4/3		φr23	\$E120	φF123/0	1	⊅ EF 4 5		Ģ FZ Z 3	\$E7.20	ŞFF 4 5	\$F720	φE3 2 4	\$F3 2 1
SED	Set Decimal Flag Set Interrupt Mask (Disable Interrupts)	\$F8 1 2 \$78 1 2														1								
SEI	Store Accumulator	\$7012				\$85 2 3	\$95 2 4		\$8D 3 4	\$9D 3 5	\$99 3 5		\$923	\$81 2 6	\$9126		\$8F 4 5		\$92 2 5	\$87 2 6	\$9F 4 5	\$97 2 6	\$83 2 4	\$93 \$2 7
STX	Store Index Register X					\$86 2 3	****	\$96 2 4	\$8E 3 4															
STY	Store Index Register Y Transfer Accumulator to Index Reg X	\$AA 1 2				\$84 2 3	\$94 2 4		\$8C 3 4							- z 7								
TAY	Transfer Accumulator to Index Reg Y	\$A8 1 2														- z 7								
TSX	Transfer Stack Pointer to X Transfer Index Register X to Accumulator	\$BA 1 2 \$8A 1 2														- z 7 - z 7								
TXS	Transfer X to Stack Pointer	\$9A 1 2																						
TYA BRA	Transfer Index Register Y to Accumulator Branch Relative Always (JR)	\$98 1 2	\$80 2													- 2 7								
COP	(BRL for long) Coprocessor Enable	\$02 2 7	\$82 3 4													ID								
MVN	Block Move Next (LDIR) MVN M,N A bytes from MX->NY (Alters DBR)				\$54 3 ?																			
MVP	Block Move Previous (LDDR) MVP M,N A bytes from MX→NY (Alters DBR)				\$44 3 ?																			
PEA	Push Effective Absolute address Push Effective Indirect Address							\$F4 3 5											\$D4 2 6					
PEI PER	Push effective PC Relative Indirect Addr		\$62 3 6																\$D4 2 6					
PHB	Push 8 bit Data Bank Reg (DBR)	\$8B 1 3 \$0B 1 4																						
PHD PHK	Push 16bit Direct Page Register Push 8 bit Program Bank Register (PBR)	\$4B 1 3																						
PHX	Push X	\$DA 1																						
PHY	Push Y Pull 8 bit Data Bank Reg (DBR)	\$5A 1 \$AB 1 4														- z 7								
PLD	Pull 16bit Direct Page Register	\$2B 1 5														- z 7 - z 7								
PLX PLY	Pull X Pull Y	\$FA 1 \$7A 1														- z 7								
REP	Reset Status Bits				\$C2 2 3											,,,,,,,,								
SEP	Set Status Bit Stop processor until next RST	\$DB 1			\$E2 2 3																			
STZ	Store Zero to address					\$64.2	\$74.2		\$9C 3	\$9E 3														
TCD TDC	Transfer A to Direct page register (aka TAD) Transfer Direct Page register to A (aka TDA)	\$5B 1 2 \$7B 1 2														- z 7								
TCS	Transfer Accumulator to SP (aka TAS)	\$1B 1 2														- 2 7								
TRB TSB	Test and Reset Bits with A Test and Set Bits with A					\$14 2 \$04 2			\$1C 3 \$0C 3							- z								
TSC	Transfer SP to Accumulator (aka TSA)	\$3B 1 2														- z								
TXY TYX	Transfer X to Y Transfer Y to X	\$9B 1 2 \$BB 1 2														- z 7 - z 7								
WAI	Wait until any interrupt	\$CB 1																						
WDM XBA	Reserved for future use! Exchange A and B (aka SWA)	\$42.2 \$EB.1.3														- z 7								
XCE	Exchange Carry (C) and Emu bits (E)	\$FB 1 2														Еив								
BBR BBS	Branch if bit n is Reset (also on some 65c02) Branch if bit n is Reset (also on some 65c02)					\$0F-\$7F 2 \$8F-\$FF 2										D								
BSR	Branch to subroutine (Call Relative)	\$44 2 8																						
CLX	Clear X Clear Y	\$82 1 2 \$C2 1 2																						
CLY	Change Speed High (7.16 MHz)	\$D4 1 3																						
CSL	Change Speed Low (1.78 MHz)	\$54 1 3				\$07.677																		
RMB SAX	Reset Memory Bit n (also on some 65c02) Swap A and X	\$22 1 3				\$07-\$77																		
SAY	Swap A and Y	\$42 1 3																						
SET SMB	Set T flag Set Memory Bit n (also on some 65c02)	\$F4 1 2				\$87-\$F7																		
ST0	ST0 - Store (HuC6270) VDC No. 0				\$03 2 5																			
ST1 ST2	ST1 - Store (HuC6270) VDC No. 1 ST2 - Store (HuC6270) VDC No. 2				\$13 2 5 \$23 2 5																			
SXY	Swap X and Y registers	\$02 1 3																						
TAI TAM	Transfer Alternate Increment Transfer Accumulator to MPR				\$F3 7 17+ \$53 2 5									>=6502 002, 65816										
TIA	Transfer Accumulator to MPR Transfer Increment Alternate				\$E3 7 17+									2,6280 NO										
TII	Transfer Increment Increment Transfer Increment				\$73 7 17+ \$D3 7 17+									6280										
TIN TMA	Transfer Increment Transfer MPR to Accumulator				\$D3 7 17+ \$43 2 4									6280 + 658 65816										
	Test Bits at n2 with n1					\$83 3 7	\$A3 3 7	\$93 4 8	\$B3 4 8				16 bit	in 65816 M										

650	2 – 6280															
000	2 – 0200	Implied	Relative	Accum	Immediate	Zero Page	Zero Pg,X	Zero PG,Y	Absolute	Absolute,X	Absolute,Y	Abs,X Indir	Indirect	(Indirect,X)	(Indirect),Y	CZIDBVN
		Implied	relative	Accum	#nn	\$nn	\$nn,X	\$nn,Y	\$0100	\$0100,X	\$0100,Y	(\$nnnn,X)	(\$nnnn)	(\$nn,X)	(\$nn),Y	
		no params	jr	works on A	&nn	(&00nn)	(&00nn+X)	(&00nn+Y)	(&0100)	(&0100+X)	(&0100+Y)	((&nnnn +X))	((&nnnn))	((&00nn+X))	((&00nn)+Y)	
ADC	Add with Carry				\$69 2 2 \$29 2 2	\$65 2 3	\$75 2 4 \$35 2 4		\$6D 3 4 \$2D 3 4	\$7D 3 4/5	\$79 3 4/5 \$39 3 4/5		\$72 3 \$32 3	\$61 2 6 \$21 2 6	\$71 2 5/6 \$31 2 5/6	Ovf7 Z +- 7
AND ASL	Logical AND Arithmetic Shift Left			\$0A 1 2	\$29 2 2	\$25 2 3 \$06 2 5	\$1626		\$2D 3 4 \$0E 3 6	\$3D 3 4/5 \$1E 3 7	\$39 3 4/5		\$32.3	\$2126	\$31 2 5/0	7 Z 7
BCC	Branch if Carry Clear C=1 (Aka BLT)		\$90 2 2-4	φολίτΣ		400 2 0	Ψ1020		ΨΟΣ Ο Ο	QIE 07						
BCS	Branch if Carry Set C=0 (Aka BGE)		\$B0 2 2-4													
BEQ	Branch if Equal to Zero		\$F0 2 2-4													
BIT	Bit Test (set flags like AND)				\$89 2	\$24 2 3	\$34 2		\$2C 3 4	\$3C 3						- z 6 7
ВМІ	Branch if Minus (S = 1)		\$30 2 2-4													
BNE	Branch if Not Equal to Zero		\$D0 2 2-4 \$10 2 2-4													
BPL BRK	Branch if Plus (S = 0) Break	\$00 1 7	\$10 2 2-4													=1
BVC	Branch if Overflow Clear	Ψοσιί	\$50 2 2-4													
BVS	Branch if Overflow Set		\$70 2 2-4													
CLC	Clear Carry Flag															=0
CLD	Clear Decimal Mode	\$D8 1 2														=0
CLI	Clear Interrupt Mask (Enable Interrupts)	\$58 1 2														=0
CLV	Clear Overflow Flag	\$B8 1 2							***					****		=0 - > = 7
CMP	Compare Accumulator to Memory				\$C922	\$C5 2 3	\$D5 2 4		\$CD 3 4	\$DD 3 4/5	\$D9 3 4/5		\$D2 3	\$C1 2 6	\$D1 2 5/6	>=7
CPX	Compare with Index Register X Compare with Index Register Y				\$E0 2 2 \$C0 2 2	\$E4 2 3 \$C4 2 3			\$EC 3 4 \$CC 3 4							>=7
DEC	Decrement (Aka DEA)			\$3A	ψ30 Z Z	\$C6 2 5	\$D6 2 6		\$CE 3 6	\$DE 3 7						- z 7
DEX	Decrement Index Register X	\$CA 1 2		J 0.1		,	,_0_0		,	,,						- z 7
DEY	Decrement Index Register Y	\$88 1 2														- z 7
EOR	Logical Exclusive-OR (XOR)				\$49 2 2	\$45 2 3	\$55 2 4		\$4D 3 4	\$5D 3 4/5	\$59 3/4/5		\$523	\$41 2 6	\$51 2 5/6	- Z 7
INC	Increment (Aka INA)			\$1A		\$E6 2 5	\$F6 2 6		\$EE 3 6	\$FE 3 7						- z 7
INX	Increment Index Register X	\$E8 1 2														- z 7
INY	Increment Index Register Y	\$C8 1 2							640 2 2			670	***			- z 7
JMP	Jump to New Location (or JML for long)								\$4C 3 3			\$7C 3	\$6C 3.5			
JSR	Jump to Subroutine (or JSL for long)								\$20 3 6	\$FC 3 8						
LDA	Load Accumulator				\$A9 2 2	\$A5 2 3	\$B5 2 4		\$AD 3 4	\$BD 3 4/5	\$B9 3 4/5		\$B2 3	\$A1 2 6	\$B1 2 5	- z 7
LDX	Load Index Register X				\$A2 2 2	\$A623	CD404	\$B6 2 4	\$AE 3 4	\$BC 3 4/5	\$BE 3 4/5					- Z 7
LDY LSR	Load Index Register Y Logical Shift Right			\$4A 1 2	\$A0 2 2	\$A4 2 3 \$46 2 5	\$B4 2 4 \$56 2 6		\$AC 3 4 \$4E 3 6	\$5E 3 7						- z 7
NOP	No Operation	\$EA 1 2		ψ+Λ 1 2		ψ-10 2 3	ψ50 2 0		Ψ-L 3 0	Ψ3Ε 3 7						
ORA	Logical (Inclusive) OR	,			\$09 2 2	\$05 2 3	\$15 2 4		\$0D 3 4	\$1D 3 4/5	\$19 3 4/5		\$123	\$01 2 6	\$11 2 5/6	- z 7
PHA	Push Accumulator onto Stack	\$48 1 3														
PHP	Push Processor Status	\$08 1 3														
PLA	Pull Accumulator from Stack	\$68 1 4														- z 7
PLP	Pull Processor Status	\$28 1 4														S S S S S S S S old7 Z 7
ROL	Rotate Left through Carry			\$2A 1 2 \$6A 1 2		\$26 2 5 \$66 2 5	\$36 2 6 \$76 2 6		\$2E 3 6 \$6E 3 6	\$3E 3 7 \$7E 3 7						old/ Z 7
ROR	Rotate Right through Carry Return from Interrupt (RETI)	\$40 1 6		\$6A 1 2		\$00 2 5	\$/020		\$0E 3 0	\$/E 3 /						SSSSSS
RTI RTS	Return from Sub (RET) (or RTL for long)	\$60 1 6														
		\$6B 1 6									250045		0500			067 #
	Subtract with Carry	£20.4.0			\$E9 2 2	\$E5 2 3			\$ED 3 4	\$FD 3 4/5	\$F9 3 4/5		\$F2 3	\$E1 2 6	\$F1 2 5/6	Ovf7 Z 7
SEC SED	Set Carry (SCF) Set Decimal Flag	\$38 1 2 \$F8 1 2														1
SEI	Set Interrupt Mask (Disable Interrupts)	\$78 1 2														1
STA	Store Accumulator	7.012				\$85 2 3	\$95 2 4		\$8D 3 4	\$9D 3 5	\$9935		\$923	\$81 2 6	\$91 2 6	
	Store Index Register X					\$86 2 3		\$96 2 4	\$8E 3 4							
STY	Store Index Register Y					\$84 2 3	\$94 2 4		\$8C 3 4							
TAX	Transfer Accumulator to Index Reg X	\$AA 1 2														- z 7
TAY	Transfer Accumulator to Index Reg Y	\$A8 1 2														- z 7
TSX	Transfer Stack Pointer to X	\$BA 1 2														- z 7 - z 7
TXA TXS	Transfer Index Register X to Accumulator Transfer X to Stack Pointer	\$8A 1 2 \$9A 1 2														/
TYA	Transfer Index Register Y to Accumulator	\$98 1 2														- z 7
	Branch Relative Always (JR)		\$80 2													
	(BRL for long)	CDA 4	\$82 3 4													
PHX PHY	Push X Push Y	\$DA 1 \$5A 1														
PLX	Pull X	\$5A 1														- z 7
PLY	Pull Y	\$7A 1														- z 7
STZ	Store Zero to address					\$64 2	\$74 2		\$9C 3	\$9E 3						
														>=6502		
													650	02, 65816	, 6280	
													65C0	2,6280 NO	T 65816	
														6280	146	
													'	6280 + 658 65816	, 10	
													16 bit	in 65816 N	1=0 / X=0	
ı	ı	1	1		ı	ı	l			I	ı				•	ı l

Mnemonic	Description	Example	Valid Lengths Addressing Modes	Flags
ABCD Dm,Dn ABCD -(Am),-(An)	Adds two 8 bit Binary Coded Decimal numbers with eXtend	ABCD D1,D2	В	X n Z v C
ADD <ea>,Dn ADD Dn,<ea></ea></ea>		,		-
ADDA <ea>,An</ea>		4DD D4 D0	D.W.I	VN7V0
ADDI #, <ea> ADDQ #,<ea></ea></ea>	Adds two numbers together. Adds a short immediate value # to <ea>.</ea>	ADD D1,D2 ADDQ #1,A1	B,W,L B,W,L	X N Z V C X N Z V C
AND <ea>,Dn AND Dn,<ea></ea></ea>				
ANDI #, <ea></ea>	logically ANDs two numbers together.	AND D1,D2	B,W,L	XNZVC
ANDI #,CCR ANDI ##,SR	logically ANDs immediate value # with the CCR is only available in Supervisor Mode.	ANDI #\$F0,CCR ANDI #\$0F,SR	B W	X N Z V C X N Z V C
ASL Dm,Dn ASL # <data>,Dn</data>				
ASL <ea></ea>	Shift the bits Left for Arithmetic	ASL.W D1,D2	B,W,L	XNZVC
ASR Dm,Dn ASR # <data>,Dn</data>				
ASR <ea> Bcc #</ea>	Shift the bits Right for Arithmetic Branch to offset/Label # if the condition cc is true.	ASR.W D1,D2 BCC TestLabel	B,W,L B,W	XNZVC
BCHG Dn, <ea> BCHG #,<ea></ea></ea>	Test Bit Dn / # of destination <ea>, and flip bit in <ea></ea></ea>	BCHG #1,D1	B,L	Z
BCLR Dn, <ea></ea>				
BCLR #, <ea></ea>	Test Bit Dn / # of destination <ea>, and zero bit in <ea> BRA TestLabel</ea></ea>	BCLR #1,D1 BRA TestLabel	B,L B,L	Z
BSET Dn, <ea></ea>				
BSET #, <ea></ea>	Test Bit Dn / # of destination <ea>, and set bit in <ea> Branch to Subroutine at relative offset #.</ea></ea>	BSET #1,D1 BSR TestLabel	B,L B,W	Z
BTST Dn, <ea></ea>				
BTST #, <ea> CHK <ea>,Dn</ea></ea>	Test Bit Dn or # of destination <ea></ea>	BTST #1,D1	B,L	Z
CHK #,Dn CLR <ea></ea>	Compare Dn to upper bound # Trap 6 if out of range	CHK #1000,D1 CLR.B \$1000	W, L {on 68020+}	- Nzvc
CMP <ea>,Dn</ea>	Clear <ea> setting it to zero.</ea>	CLR.B \$1000	B,W,L	- N Z V C
CMPA <ea>,An CMPI #.<ea></ea></ea>			B, W, L	
CMPM (Am)+,(An)+ DBcc Dn,#	CMP compares <ea> to Dn. Decrease Dn, if Dn > -1 and branch if cc not true</ea>	CMPI.B #\$FF,(A1) DBRA D0,TestLabel	(W,L for CMPA) B,W	- N Z V C
DIVS <ea>,Dn</ea>	Divide Signed numbers. Dn is divided by <ea>. Dn=Dn / <ea>.</ea></ea>	DIVS #4,D1	L = L/w	- N Z V C
DIVU <ea>,Dn</ea>	DIVide Unsigned numbers. Effectively Dn=Dn / <ea>.</ea>	DIVU #4,D1	L = L/w	- N Z V C
EOR Dn, <ea> EORI #,<ea></ea></ea>	Logical EOR (Exclusive OR) of bits in Dn or # with <ea>.</ea>	EOR #\$20,D1	B,W,L	- N Z V C
EORI #,CCR EORI ##,SR	Logical EOR # with the CCR is only available in Supervisor Mode.	EORI #\$F0,CCR EORI #\$F0,SR	В	X N Z V C X N Z V C
EXG Dn,Dm				
EXG An,Am EXT Dn	Exchange the contents of registers Dn and Dm. Sign extend register Dn, either extending a Byte to Word.	EXG D1,D2 EXT.W D1	L W,L	 - N Z V C
ILLEGAL JMP #	execute "Illegal Instruction Vector" (Trap 4).	ILLEGAL IMP Tooth about		
JSR#	Jump to absolute address #. Jump to Subroutine at absolute address #.	JMP TestLabel JSR TestLabel	L L	
LEA <ea>,An LINK An,#</ea>	Load the effective address <ea> into An.</ea>	LEA (Label,PC),A1		
LSL Dm,Dn	Creates a 'Tepmporary area' on the stack for work	LINK A1,#-4		
LSL #,Dn LSL <ea></ea>	Shift the bits in register Dn Left Logically by Dm or # bits.	LSL #1,D1		XNZVC
LSR Dm,Dn	, , ,			
LSR #,Dn LSR <ea></ea>	Shift the bits in register Dn Right Logically by Dm or # bits.	LSR #1,D1	B, W, L	XNZVC
MOVE <ea>,<ea2> MOVEA <ea>,An</ea></ea2></ea>	Move the contents of source <ea> to the destination <ea2>.</ea2></ea>	MOVE #15,D1	B, W, L	- N Z V C
MOVE <ea>,CCR</ea>	moves a 16 bit value from <ea> to the CCR</ea>	MOVE D0,CCR	w	XNZVC
MOVE SR, <ea> MOVE <ea>,SR</ea></ea>	Move to or from the Status Register	MOVE SR,D0	W	XNZVC
MOVE USP,An MOVE An,USP	Transfer the User Stack Pointer to or from address register An.	MOVE USP,A0	L	
MOVEM <ea>,<regs> MOVEM <regs>,<ea></ea></regs></regs></ea>	· · · · · · · · · · · · · · · · · · ·			
MOVEP Dn,(#,An)		MOVEM.L (A1),D0/D3		
MOVEP (#,An),Dn MOVEQ #,Dn	Move 16 or 32 bits to a set of memory mapped byte data ports. adds short immediate # to the register Dn.	MOVEP.L D0,(4,A1) MOVEQ #1,D1	W,L L	
MULS <ea>,Dn</ea>	Multiply Signed numbers. Dn=Dn* <ea>.</ea>	MULS #4,D1	L=W*W	- N Z V C
MULU <ea>,Dn NBCD Dn</ea>	Multiply Unsigned numbers. Dn=Dn* <ea>. Negates BCD byte with eXtend. Dn. Dn=(0-Dn)-{X flag}</ea>	MULU #4,D1 NBCD D1	L=W*W B	- N Z V C X n Z v C
NEG <ea></ea>	Negate <ea></ea>	NEG D0	B, W, L	XNZVC
NEGX <ea></ea>	Negate <ea> with eXtend No Operation.</ea>	NEGX <ea></ea>	B, W, L	XNZVC
NOT <ea></ea>	Invert/Flip all the bits of <ea>.</ea>	NOT.L D1	B, W, L	- N Z V C
OR <ea>,Dn OR Dn,<ea></ea></ea>				
ORI #, <ea> ORI #,CCR</ea>	logically ORs two numbers together. logically ORs immediate value # with the CCR	OR D1,D2 ORI #\$0F,CCR	B, W, L B	- N Z V C X N Z V C
ORI ##,SR	logically ORs immediate value ## with the Status Register.	ORI #\$0F,SR	W	XNZVC
PEA <ea>,An RESET</ea>	Push the effective address <ea> onto the stack. Sends an "RSTO" signal</ea>	PEA (Label,PC) RESET	L	
ROL Dm,Dn ROL #,Dn				
ROL <ea></ea>	Rotate bits in Dn to the Left by a number of bits	ROL.B #8,D1	B, W, L	- N Z V C
ROR Dm,Dn ROR #,Dn				
ROR <ea></ea>	Rotate bits in Dn to the Right by a number of bits	ROR.B #8,D1	B, W, L	- N Z V C
ROXL Dm,Dn ROXL #,Dn				
ROXL <ea> ROXR Dm,Dn</ea>	Rotate bits in Dn to the Left, with the eXtend bit	ROXL.B #8,D1	B, W, L	XNZVC
ROXR #,Dn ROXR <ea></ea>	Rotate bits in Dn to the Right, with the eXtend bit	ROXR.B #8,D1	B, W, L	XNZVC
RTE	Return from Exception.	RTE	u,, u	XNZVC
RTR RTS	Return and Restore condition codes. Return from a Subroutine.	RTR RTS		XNZVC
SBCD Dm,Dn				
SBCD -(Am),-(An) Scc <ea></ea>	Subtracts two 8 bit Binary Coded Decimal with eXtend carry Set <ea> to 255 or 0 according to condition cc.</ea>	SBCD D1,D2 SEQ.B TestLabel	B B	X n Z v C
STOP ##	Load the SR Status register with 16 bit immediate ## and halt			
SUB <ea>,Dn SUB Dn,<ea></ea></ea>				
SUBA <ea>,An SUBI #,<ea></ea></ea>	Subtracts two numbers.	SUBI.B #1,(A1)	B, W, L	XNZVC
SUBQ #, <ea></ea>	Subtracts a short immediate value # from <ea>.</ea>	SUBQ #1,A1	B, W, L	XNZVC
SUBX Dm,Dn SUBX -(Am),-(An)	Subtracts with the eXtend bit.	SUBX D1,D2	B, W, L	XNZVC
SWAP Dn TAS <ea></ea>	Swap the high and low words of register Dn. Test and set <ea>.</ea>	SWAP D1 TAS D1	W B	- N Z V C - N Z V C
TRAP#	causes a jump to exception vector number #.	TRAP #1		
TRAPV	If the oVerflow flag (V) is set, call overflow trap vector Set the flags according to <ea>.</ea>	TRAPV TST.B D1	B, W, L	 - N Z V C
TST <ea></ea>				

Flag	Name	Description
Т	Trace bit	1 if Trace enabled (Jump to address at Trace Vector \$000024 every instruction for debugging)
S	supervisor	1 if in supervisor mode, 0 in user mode
I	Interrupt	Interrupt level (0-7)
Х	eXtend	Used for transferring data in/out of registers in rotation
N	Negative	1 if top-most bit of result is 1 (meaning the value is negative)
Z	Zero	1 if result of previous operation was zero
٧	Overflow	1 if arithmetic overflow occurred (last operation caused accidental sign change)
С	Carry	1 if the last operation resulted in a Carry or borrow

CC	Description	Flags					
CC	carry clear	C=0					
CS	carry set	C=1					
EQ	Equal	Z=1					
GE	Greater than or equal	(N=1 & V=1) or (N=0 & V=0)					
GT	Greater than	(N=1 & V=1 & Z=0) or (N=0 & V=0 & Z=0)					
HI	Higher than	C=0 & Z=0					
LE	Less than or equal	Z=1 or (N=1 & V=0) or (N=0 & V=1)					
LS	Lower than or same	C=1 or Z=1					
LT	Less than	(N=1 and V=0) or (N=0 and V=1)					
MI	Minus	N=1					
NE	Not equal	Z=0					
PL	Plus	N=0					
Т	True	=0					
F	False	=1					
VC	Overflow clear	V=0					
vs	Overflow Set	V=1					

6809	Manina	Inharant	luuma aliata	Direct	Eutodod	lander/landin	Deletive	U.N. #. V. C
ABX ADCA	Meaning Add B to X Add with Carry to A	Inherent \$3A (3/1)	Immediate \$89 (2/2)	\$99 (4/2)	\$B9 (5/3)	Indx/Indir \$A9 (4+/2+)	Relative	H N Z V C
ADCB ADDA	Add with Carry to B Add to A		\$C9 (2/2) \$8B (2/2)	\$D9 (4/2) \$9B (4/2)	\$F9 (5/3) \$BB (5/3)	\$E9 (4+/2+) \$AB (4+/2+)		* * * * *
ADDB ADDD	Add to B add to AB (16 bit)		\$CB (2/2) \$C3 (4/3)	\$DB (4/2) \$D3 (6/2)	\$FB (5/3) \$F3 (7/3)	\$EB (4+/2+) \$E3 (6+/2+)		* * * * *
ANDA ANDB	And with A And with B		\$84 (2/2) \$C4 (2/2)	\$94 (4/2) \$D4 (4/2)	\$B4 (5/3) \$F4 (5/3)	\$A4 (4+/2+) \$E4 (4+/2+)		- * * 0 - - * * 0 -
ANDCC ASL ASLA	And with ConditionCode Arithmatic Shift Left Arithmatic Shift Left A	\$48 (2/1)	\$1C (3/2)	\$08 (6/2)	\$78 (7/3)	\$68 (6+/2+)		7 8 * * * * 8 * * *
ASLB ASR	Arithmatic Shift Left B Arithmatic Shift Right	\$58 (2/1)		\$07 (6/2)	\$77 (7/3)	\$67 (6+/2+)		8 * * * *
ASRA ASRB	Arithmatic Shift Right A Arithmatic Shift Right B	\$47 (2/1) \$46 (2/1)		,, (, ,	, , ,	, ,		8 * * - *
BCC BCS	Branch if Carry Clear C=0 Branch if Carry Set C=1						\$24 (3/2) \$25 (3/2)	
BEQ BGE	Branch if Equal Z=1 Branch if Greater than or equal to zero						\$27 (3/2) \$2C (3/2)	
BGT BHI BHS	Branch if Greater than Zero Branch if Higher Z+C=0 Branch if Higher or Same C=0						\$2E (3/2) \$22 (3/2) \$24 (3/2)	
BITA BITB	Bit Test A Bit Test B		\$85 (2/2) \$C5 (2/2)	\$95 (4/2) \$D5 (4/2)	\$B5 (5/3) \$F5 (5/3)	\$A5 (4+/2+) \$E5 (4+/2+)	\$24 (3/2)	8 * * 0 * 8 * * 0 *
BLE	Branch if Less than or Equal to Zero Branch if Lower C=1		\$55 (E.Z)	¥== (<u>=</u> /	4. 5 (5.5)	, , , , , , , , , , , , , , , , , , ,	\$2F (3/2) \$25 (3/2)	
BLS BLT	Branch if Lower or Same C+Z=1 Branch if Less Than Zero						\$23 (3/2) \$2D (3/2)	
BMI BNE	Branch if Minus N=1 Branch if Not Equal to Zero Z=0						\$2B (3/2) \$26 (3/2)	
BPL BRA BRN	Branch if Plus N=0 Branch Always Branch Never						\$2A (3/2) \$20 (3/2) \$21 (3/2)	
BSR BVC	Branch to Subroutine Branch if Overflow Clear V=0						\$8D (3/2) \$28 (3/2)	
BVS CLR	Branch if Overflow Set V=1 Clear			\$0F (6/2)	\$7F (7/3)	\$6F (6+/2+)	\$29 (3/2)	- 0 1 0 0
CLRA CLRB	Clear A Clear B	\$4F (2/1) \$5F (2/1)						- 0 1 0 0 - 0 1 0 0
CMPA CMPB	Compare with A Compare with A Compare with AB		\$81 (2/2) \$C1 (2/2) \$10.93 (5/4)	\$91 (4/2) \$D1 (4/2) \$10.03 (7/3)	\$B1 (5/3) \$F1 (5/3) \$10 B2 (9/4)	\$A1 (4+/2+) \$E1 (4+/2+) \$10 A2 (7+/2+)		8 * * * * - * * * *
CMPD CMPS CMPU	Compare with AB Compare with S Compare with U		\$10 83 (5/4) \$11 8C (5/4) \$11 83 (5/4)	\$10 93 (7/3) \$11 9C (7/3) \$11 93 (7/3)	\$10 B3 (8/4) \$11 BC (8/4) \$11 B3 (8/4)	\$10 A3 (7+/3+) \$11 AC (7+/3+) \$11 A3 (7+/3+)		- * * * * - * * * *
CMPX CMPY	Compare with X Compare with Y		\$11 83 (5/4) \$8C (4/3) \$10 8C (5/4)	\$11 93 (7/3) \$9C (6/2) \$10 9C (7/3)	\$BC (7/3) \$10 BC (8/4)	\$AC (6+/2+) \$10 AC (7+/3+)		- * * * *
COMA	Complement Complement A	\$43 (2/1)		\$03 (6/2)	\$73 (7/3)	\$63 (6/2)		- * * 0 1 - * * 0 1
COMB	Complement B And with CC and Wait	\$53 (2/1)	\$3C (20/2)					- * * 0 1 7
DAA DEC DECA	Decimal Adjust after Addition Decrement Decrement A	\$19 (2/1) \$40 (2/1)		\$0A (6/2)	\$7A (7/3)	\$6A (6+/2+)		- * * 0 * - * * * -
DECA DECB EORA	Decrement A Decrement B Exclusive Or A (Xor)	\$4A (2/1) \$5A (2/1)	\$88 (2/2)	\$98 (4/2)	\$B8 (5/3)	\$A8 (4+/2+)		- * * * - - * * 0 -
EORB EXG	Exclusive Or B (Xor) Exchange Register Contents		\$C8 (2/2) \$1E (8/2)	\$D8 (4/2)	\$F8 (5/3)	\$E8 (4+/2+)		- * * 0 -
INC INCA	Increment Increment A	\$4C (2/1)	, (, ,	\$0C (6/2)	\$7C (7/3)	\$6C (6+/2+)		- * * * -
JMP	Increment B Jump	\$5C (2/1)		\$0E (3/2)	\$7E (4/3)		\$6E (3+/2+)	- * * * -
JSR LBCC	Jump to Subroutine Long Branch if Carry Clear C=0			\$9D (7/2)	\$BD (8/3)		\$AD (7+/2+) \$10 24 (5+/4)	
LBCS LBEQ LBGE	Long Branch if Carry Set C=1 Long Branch if Equal Z=1 Long Branch if Greater than or equal to zero						\$10 25 (5+/4) \$10 27 (5+/4) \$10 2C (5+/4)	
LBGT LBHI	Long Branch if Greater than Zero Long Branch if Higher Z+C=0						\$10 2E (5+/4) \$10 22 (5+/4)	
LBHS LBLE	Long Branch if Higher or Same C=0 Long Branch if Less than or Equal to Zero						\$10 24 (5+/4) \$10 2F (5+/4)	
LBLO LBLS	Long Branch if Lower C=1 Long Branch if Lower or Same C+Z=1						\$10 25 (5+/4) \$10 23 (5+/4)	
LBLT LBMI LBNE	Long Branch if Less Than Zero Long Branch if Minus N=1 Long Branch if Not Equal to Zero Z=0						\$10 2D (5+/4) \$10 2B (5+/4) \$10 26 (5+/4)	
LBPL LBRA	Long Branch if Plus N=0 Long Branch Always						\$10 26 (5+/4) \$10 2A (5+/4) \$16 (5/3)	
LBRN LBSR	Long Branch Never Long Branch to Subroutine						\$10 21 (5/4) \$17 (9/3)	
LBVS	Long Branch if Overflow Clear V=0 Long Branch if Overflow Set V=1						\$10 28 (5+/6) \$10 29 (5+/6)	
LDA LDB	Load A Load B		\$86 (2/2) \$C6 (2/2)	\$96 (4/2) \$D6 (4/2)	\$B6 (5/3) \$F6 (5/3)	\$A6 (4+/2+) \$E6 (4+/2+)		- * * 0 - - * * 0 -
LDD LDS LDU	Load AB Load S Load U		\$CC (3/3) \$10 CE (4/4) \$CE (3/3)	\$DC (5/2) \$10 DE (6/3) \$DE (5/2)	\$FC (6/3) \$10 FE (7/4) \$FE (6/3)	\$EC (5+/2+) \$10 EE (6+/3+) \$EE (5+/2+)		- * * 0 - - * * 0 -
LDX LDY	Load X Load Y		\$8E (3/3) \$10 8E (4/4)	\$9E (5/2) \$10 9E (6/3)	\$BE (6/3) \$10 BE (7/4)	\$AE (5+/2+) \$10 AE (6+/3+)		- * * 0 - - * * 0 -
LEAS LEAU	Load Effective Address into S Load Effective address into U					\$32 (4+/2+) \$33 (4+/2+)		
LEAY	Load Effective Address into X Load Effective Address into Y			\$00 (0(0))	ATO (T/O)	\$30 (4+/2+) \$31 (4+/2+)		* *
LSLA LSLB	Logical Shift Left Logical Shift Left A Logical Shift Left B	\$48 (2/1) \$58 (2/1)		\$08 (6/2)	\$78 (7/3)	\$68 (6+/2+)		- * * * *
LSR LSRA	Logical Shift Right Logical Shift Right A	\$44 (2/1)		\$04 (6/2)	\$74 (7/3)	\$64 (6+/2+)		- 0 * - *
LSRB MUL	Logical Shift Right B Multiply A*B – result in AB	\$54 (2/1) \$3D (11/1)						- 0 * - * * - 9
NEGA NEGA	Negate A	\$40 (2/1)		\$00 (6/2)	\$70 (7/3)	\$60 (6+/2+)		8 * * * * 8 * * * *
NEGB NOP ORA	Negate B No Operation Or A	\$50 (2/1) \$12 2/1	\$8A (2/2)	\$9A (4/2)	\$BA (5/3)	\$AA (4+/2+)		8 * * * *
ORB ORCC	Or A Or B Or Condition Code		\$8A (2/2) \$CA (2/2) \$1A (3/2)	\$9A (4/2) \$DA (4/2)	\$BA (5/3) \$FA (5/3)	\$AA (4+/2+) \$EA (4+/2+)		- * * 0 - - * * 0 -
PSHS PSHU	Push onto S stack (PC U Y X DP B A CC) Push onto U stack (PC S Y X DP B A CC)		\$34 (3/2) \$36 (3/2)					
PULS PULU	Pull off S stack (PC U Y X DP B A CC) Pull off U stack (PC S Y X DP B A CC)		\$35 (3/2) \$37 (3/2)					
ROLA	Rotate Left through Carry Rotate Left through Carry A	\$49 (2/1)		\$09 (6/2)	\$79 (7/3)	\$69 (6+/2+)		- * * * * - * * * *
ROLB ROR RORA	Rotate Left through Carry B Rotate Right through Carry Rotate Right through Carry A	\$59 (2/1) \$46 (2/1)		\$06 (6/2)	\$76 (7/3)	\$66 (6+/2+)		- * * * * - * * - *
RORB RTI	Rotate Right through Carry B Return from Interrupt	\$56 (2/1) \$58 (6/15)						- * * - *
RTS SBCA	Return From Subroutine Subtract with Carry from A	\$39 (5/1)	\$82 (2/2)	\$92 (4/2)	\$B2 (5/3)	\$A2 (4+/2+)		8 * * * *
SEX	Subtract with Carry from B Sign Extend B into AB	\$1D (2/1)	\$C2 (2/2)	\$D2 (4/2)	\$F2 (5/3)	\$E2 (4+/2+)		8 * * * * - * * 0 - - * * 0 -
STA STB STD	Store A Store B Store AB			\$97 (4/2) \$D7 (4/2) \$DD (5/2)	\$B7 (5/3) \$F7 (5/3) \$FD (6/3)	\$A7 (4+/2+) \$E7 (4+/2+) \$ED (5+/2+)		- * * 0 - - * * 0 -
STS STU	Store S Store U			\$10 DF (6/3) \$DF (5/2)	\$10 FF (7/4) \$FF (6/3)	\$ED (5+/2+) \$10 EF (6+/3+) \$EF (5+/2+)		- * * 0 - - * * 0 -
STX	Store X Store Y			\$9F (5/2) \$10 9F (6/3)	\$BF (6/3) \$10 BF (7/4)	\$AF (5+/2+) \$10 AF (6+/3+)		- * * 0 - - * * 0 -
SUBA SUBB	Subtract from A Subtract from B		\$80 (2/2) \$C0 (2/2)	\$90 (4/2) \$D0 (4/2)	\$B0 (5/3) \$F0 (5/3)	\$A0 (4+/2+) \$E0 (4+/2+)		8 * * * *
SUBD	Subtract from AB Software Interrupt	\$3F (19/1)	\$83 (4/3)	\$93 (6/2)	\$B3 (7/3)	\$A3 (6+/2+)		- * * * *
SWI2 SWI3 SYNC	Software Interrupt 2 Software Interrupt 3 Syncronise to Ext Event (wait for interrupt)	\$10 3F (20/2) \$11 3F (20/2) \$13 (2/1)						
TFR TST	Transfer Register to Register (X,Y,U,S,A,B,D,PC,CC) Test (Set flags)	\$10 (M1)	\$1F (7/2)	\$0D (6/2)	\$7D (6/3)	\$6D (6+/2+)		
TSTA TSTB	Test A Test B	\$4D (2/1) \$5D (2/1)				,		- * * 0 - - * * 0 -

6309	Manufact					1,	Deletion
Cmd	Meaning	Inherent	Immediate	Direct	Exteded	Indx/Indir	Relative EFHINZVC
DCD	Add Memory Word plus Carry with Accumulator D Add Source Register plus Carry to Destination Register		\$18 09 (4/4-5) \$10 31(3/4)	\$10 99 (3/5-7)	\$10 B9 (6-8)	\$10 A9 (6-7/3)	* * * *
DDE	Add Memory Byte to 8-Bit Accumulator E		\$11 8B (3/3)	\$11 9B (4-5/3)	\$11 BB (3+/5+)	\$11 AB (3+/5+)	*-**
ADDF	Add Memory Byte to 8-Bit Accumulator F		\$11 CB (3/3)	\$11DB (5/4)	\$11 FB (4/5-6)	\$11 EB (3+/5+)	* _ * * *
ADDW ADDR	Add Memory Word to 16-Bit Accumulator W Add Source Register to Destination Register		\$10 8B (4/4-5) \$10 30 (3 /4)	\$10 9B (3/5-7)	\$10 BB (4/6-8)	\$10 AB (3+/6+)	* * * * * * * *
AIM	Logical AND of Immediate Value with Memory Byte		\$10.30 (3.74)	\$02 (3/6)	\$72 (4/7)	\$62 (3+/7+)	* * 0 -
ANDD	Logically AND Memory Word with Accumulator D		\$10 84 (4/4-5)	\$10 94 (3/5-7)	\$10 B4 (4/6-8)	\$10 A4 (3+/6+)	**0 -
ANDR	Logically AND Source Register with Destination Register		\$10 34 (3/4)				**0-
ASLD ASRD	Arithmetic Shift Left of Accumulator D Arithmetic Shift Right of Accumulator D	\$10 84 (2/2-3) \$10 3F (2/2-3)					* - * * - *
BAND	Logically AND Register Bit with Memory Bit	\$10.31 (2/2-3)		\$11 30 (4/6-7)			
BEOR	Exclusive-OR Register Bit with Memory Bit			\$11 34 (4/6-7)			
BIEOR	Exclusively-OR Register Bit with Inverted Memory Bit			\$11 35 (4/6-7)			
BIOR BITD	Logically OR Register Bit with Inverted Memory Bit Bit Test Accumulator D with Memory Word Value		\$10 85 (4/4-5)	\$11 33 (4/6-7) \$10 95 (3/5-7)	\$10 B5 (4/6-8)	\$10 A5 (3+/6+)	**0-
BITMD	Bit Test the MD Register with an Immediate Value		\$11 3C (3/4)	\$10 00 (0/0 1)	\$10 D0 (110 0)	ψ10710 (0×70×)	*
BOR	Logically OR Memory Bit with Register Bit			\$11 32 (4/6-7)			
CLRD	Load Zero into Accumulator		\$10 4F (2/2-3)				0100 0100
CLRE	Load Zero into Accumulator Load Zero into Accumulator		\$11 4F (2/2-3) \$11 5F (2/2-3)				0100
CLRW	Load Zero into Accumulator		\$10 5F (2/2-3)				0100
CMPE	Compare Memory Byte from 8-Bit Accumulator		\$11 81 (3/3)	\$11 91 (3/4-5)	\$11 B1 (3/5-6)	\$11 A1 (3/4-5)	*-**
CMPF CMPW	Compare Memory Byte from 8-Bit Accumulator Compare Memory Word from 16-Bit Register		\$11 C1 (3/3) \$10 81 (4/4-5)	\$11 D1 (3/4-5) \$10 91 (3/5-7)	\$11 F1 (3/5-6) \$10 B1 (4/6-8)	\$11 E1 (3/4-5) \$10 A1 (3+/6+)	* _ * * * * *
CMPR	Compare Nemory Word from 18-Bit Register Compare Source Register from Destination Register		\$10.81 (4/4-5)	₩10 91 (3/3-1)	ψ10 D1 (4/0-0)	ψ10 A1 (37/0+)	***
COMD	Complement Accumulator		\$10 43 (2/2-3)				**01
COME	Complement Accumulator		\$11 43 (2/2-3)				**01
COMF	Complement Accumulator Complement Accumulator		\$11 53 (2/2-3) \$10 43 (2/2-3)				**01 **01
DECD	Decrement Accumulator		\$10 45 (2/2-3)				**
DECE	Decrement Accumulator		\$11 4A (2/2-3)				* * * _
DECF	Decrement Accumulator		\$11 5A (2/2-3)				* * * _
DECW	Decrement Accumulator Signed Divide of Accumulator D by 8-bit value in Memory		\$10 4A (2/2-3) \$11 8D (3/25)	\$11 9D (3/26-27)	\$11 BD (4/27-28)	\$11 AD (3+/27+)	
DIVQ	Signed Divide of Accumulator Q by 16-bit value in Memory		\$11 8E (4/34)		\$11 BE (4/36-37)		* * * *
EIM	Exclusive-OR of Immediate Value with Memory Byte			\$05 (3/6)	\$65 (3+/7+)	\$75 (4/7)	**0 -
EORD EORR	Exclusively-OR Memory Word with Accumulator D Exclusively-OR Source Register with Destination Register		\$10 88 (4/4-5) \$10 36 (3/4)	\$10 98 (3/5-7)	\$10 B8 (4/6-8)		**0-
NCD	Increment Accumulator		\$10 30 (3/4) \$10 4C (2/2-3)				* * * _
NCE	Increment Accumulator		\$11 4C (2/2-3)				***-
NCF	Increment Accumulator		\$11 5C (2/2-3)				* * * _
NCW LDE	Increment Accumulator Load Data into 8-Bit Accumulator		\$10 5C (2/2-3) \$11 86 (3/3)	\$11 96 (3/3)	\$11 B6 (3/3)	\$11 A6 (3/3)	* * 0 _
_DF	Load Data into 8-Bit Accumulator		\$11 C6 (3/3)	\$11 D6 (3/3)	\$11 F6 (3/3)	\$11 E6 (3/3)	**0-
_DW	Load Data into 16-Bit Register		\$10 86 (4/4)	\$10 96 (3/5-6)	\$10 B6 (4/6-7)	\$10 A6 (3+/6+)	* * 0 -
LDBT LDMD	Load Memory Bit into Register Bit Load an Immediate Value into the MD Register		\$11 3D (3/5)	\$11 36 (4/6-7)			
_DQ	Load 32-bit Data into Accumulator Q		\$CD (5/5)	\$10 DC (3/7-8)	\$10 FC (4/8-9)	\$10 EC (3+/8+)	**0-
LSLD	Logical Shift Left of Accumulator D	\$10 48 (2/2-3)	. ,	, ,	. ,		* * * *
LSRD	Logical Shift Right of 16-Bit Accumulator	\$10 44 (2/2-3)					0*-*
LSRW MULD	Logical Shift Right of 16-Bit Accumulator Signed Multiply of Accumulator D and Memory Word	\$10 54 (2/2-3)	\$11 8F (A/28)	\$11 9F (3/29-30)	\$11 RF (A/30_31)	\$11 AF (3+/30+)	0 * - * **
NEGD	Negation (Twos-Complement) of Accumulator	\$10 40 (2/2-3)	\$1101 (4/20)	\$1131 (3/23-30)	ψ11 D1 (4/30-31)	\$11 At (5.750.)	* * * *
MIC	Logical OR of Immediate Value with Memory Byte			\$01 (3/6)	\$71 (4/7)	\$61 (3+/7+)	* * 0 -
ORD	Logically OR Accumulator D with Word from Memory		\$10 8A (4/4-5)	\$10 9A (3/5-7)	\$10 BA (4/6-8)	\$10 AA (3+/6+)	**0-
ORR PSHSW	Logically OR Source Register with Destination Register Push Accumulator W onto the Hardware Stack	\$10 38 (2/6)	\$10 35 (3/4)				**0-
SHUW	Push Accumulator W onto the User Stack	\$10 3A (2/6)					
PULSW	Pull Accumulator W from the Hardware Stack	\$10 39 (2/6)					
PULUW	Pull Accumulator W from the User Stack	\$10 3B (2/6)					
ROLD ROLW	Rotate 16-Bit Accumulator Left through Carry Rotate 16-Bit Accumulator Left through Carry	\$10 49 (2/2-3) \$10 59 (2/2-3)					* * * *
RORD	Rotate 16-Bit Accumulator Right through Carry	\$10 46 (2/2-3)					* * _ *
RORW	Rotate 16-Bit Accumulator Right through Carry	\$10 56 (2/2-3)					**-*
SBCD	Subtract Memory Word and Carry from Accumulator D		\$10 82 (4/4-5)	\$10 92 (3/5-7)	\$10 B2 (3+/6+)	\$10 A2 (3+/6+)	* * * *
SBCR	Subtract Source Register and Carry from Destination Register Sign Extend a 16-bit Value in W to a 32-bit Value in Q	\$14 (1/4)	\$10 33 (3/4)				* *
STE	Store 8-Bit Accumulator to Memory	÷ · · (1/-1)		\$11 97 (3/4-5)	\$11 B7 (4/5-6)	\$11 A7 (3+/5+)	**0-
STF	Store 8-Bit Accumulator to Memory			\$11 D7 (3/4-5)	\$11 F7 (4/5-6)	\$11 E7 (3+/5+)	**0-
STW STBT	Store 16-Bit Register to Memory Store value of a Register Bit into Memory			\$10 97 3/5-6) \$11 37 (4/7-8)	\$10 B7 (4/6-7)	\$10 A7 (3+/6+)	* * 0 -
STQ	Store Contents of Accumulator Q to Memory			\$11 37 (4/7-8) \$10 DD (3/7-8)	\$10 FD(4/8-9)	\$10 ED (3+/8+)	**0-
SUBE	Subtract from value in 8-Bit Accumulator		\$11 80 (3/3)	\$11 90 (3/4-5)	\$11 B0 (4/5-6)	\$11 A0 (4/5-6)	* _ * * *
SUBF	Subtract from value in 8-Bit Accumulator		\$11 C0 (3/3)	\$11 D0 (3/4-5)	\$11 F0 (4/5-6)	\$11 E0 (4/5-6)	* - * * * *
SUBW	Subtract from value in 16-Bit Accumulator Subtract Source Register from Destination Register		\$10 80 (4/4-5) \$10 32 (3/4)	\$10 90 (3/5-7)	\$10 B0 (4/6-8)	\$10 A0 (3+/6+)	* * * *
FM ++	Transfer Memory		\$10.32 (3/4)				
ГЕМ	Transfer Memory		\$11 39 (3/9+)				
	Transfer Memory		\$11 3A (3/9+)				
ΓFM +x			\$11 3B (3/9+)				
ΓFM +x ΓFM x+	Transfer Memory Bit Test Immediate Value with Memory Byte		, , , , , ,	\$0B (3/6)	\$7B (4/7)	\$6B (3+/7+)	* * ^ _
ΓFM +x	Transfer Memory Bit Test Immediate Value with Memory Byte Test Value in Accumulator	\$10 4D (2/2-3)		\$0B (3/6)	\$7B (4/7)	\$6B (3+/7+)	**0 - **0 -
TFM +x TFM x+ TIM	Bit Test Immediate Value with Memory Byte	\$10 4D (2/2-3) \$11 4D (2/2-3) \$11 5D (2/2-3)	, , ,	\$0B (3/6)	\$7B (4/7)	\$6B (3+/7+)	

8086				
Mnemonic	Description	Example	Valid Regs	Flags affected
AAA	ASCII Adjust for Addition. Treats AL as an unpacked binary coded decimal number	AAA		oszApC
AAD	ASCII Adjust for Division. AL=AL+(AH*10), AH=0.	AAD		oszapc
AAM	ASCII Adjust for Multiplication. We can use the normal MUL command then use AAM	AAM		o S Z a P c
AAS ADC dest,src	ASCII Adjust for Subtraction. This treats AL as an unpacked binary coded decimal number Add src and the carry flag to dest.	AAS ADC CX,1000h		0 S Z A P C
ADD dest,src	Add src to dest.	ADD CX,1000h		OSZAPC
AND dest,src	Logical AND of bits in dest with Accumulator scr.	AND AX,1100h		OSZAPC
CALL dest	Call Subroutine at address dest.	CALL 1000h		
CBW	Convert the 8 bit byte in AL into a 16 bit word in AX.	CBW		
CLC	Clear the Carry Flag. C flag will be set to Zero.	CLC		C
CLD	Clear the Direction Flag. D flag will be set to Zero. This is used for 'String functions'.	CLD		D
CLI	Clear the Interrupt enable flag. I flag will be set to 0. This disables maskable interrupts. Complement the Carry flag. If C=1 it will now be 0. If it was 0 it will now be 1.	CLI CMC		C
CMP dest,src	Compare the Byte or Word dest to src. This sets the flags the same as "SUB dest,src" would.	CMP AL,32		OSZAPC
CMPSB CMPSW	Compare DS:SI to ES:DI. This command can work in bytes or words. Sets flags like CMP	REPZ CMPSB		OSZAPC
CWD	Convert the 16 bit word in AX into a 32 bit doubleword in DX.AX. This 'Sign Extends' AX	CWD		
DAA	Decimal Adjust for Addition. This treats AL as a packed binary coded decimal number.	DAA		OSZAPC
DAS DEC Dest	Decimal Adjust for Subtraction. This treats AL as a packed binary coded decimal number. Decrement Dest by one.	DAS DEC AL		O S Z A P C O S Z A P -
DIV src	Divide Unsigned number AX or DX.AX by src. AL=AX/src (8 bit) or AX=DX.AX/src (16 bit)	DIV CX		oszapc
ESC #,src	This command is for working with multiple processors - it's not something you will need.	ESC 1,AH		
HLT	Stop the CPU until an interrupt occurs	HLT		
IDIV src	Divide Signed number AX or DX.AX by src. AL=AX / src (8 bit) or AX=DX.AX / src (16 bit)	IDIV CX		oszapc
IMUL src	Multiply Signed number AX or DX.AX by src. AX=AL*src (8 bit) or DX.AX=AX*src (16 bit)	IMUL CX		OszapC
IN dest,port	Read in an 8 bit byte or 16 bit word into dest (either AX, AL or AH). Use DX for 16 bit port num	IN AX,F0h		
INC Dest INT #	Increase Dest by one. This is faster than using ADD with a value of 1.	INC AL INT 33h		O S Z A P -
INT #	Causes software interrupt #. The flags are pushed onto the stack before call INTO will cause Interrupt 4 if the Overflow flag (O) is set, otherwise it will have no effect.	INTO		
IRET	Restore the flags from the stack and return from an Interrupt.	IRET		OSZAPC
Jcc addr	Jump to 8 bit offset addr if condition cc is true.	JO ErrorHandle		
JCXZ addr	Jump to 8 bit offset addr if CX=0.	JCXZ NoLoop		
JMP addr	Jump to address addr.	JMP BX		
LAHF	Load AH from the Flags. This only transfers the main flags: SZ-A-P-C	LAHF		
LDS reg,addr LEA reg,src	Load a full 32 bit pointer into DS segment register and register reg. Load the effective address src into reg.	LDS BX,TestPointer LEA CX,[BX+DI]	AX, BX, CX, DX, SI, D	
LES reg,addr	Load the effective address src into reg. Load a full 32 bit pointer into ES segment register and register reg.	LES AX, MyLabel	AX, BX, CX, DX,SI, DI. AX,BX,CX,DX,SI,DI	
LOCK	Enable the LOCK signal. This is for multiprocessor systems.	LOCK	AA,DA,CX,DX,SI,DI	
LODSB LODSW	Load from DS:SI into AX or AL. This command can work in bytes or words.	LODSB		
LOOP addr	Decrease CX and jump to label addr if CX is not zero.	LOOP LoopLabel		
LOOPNE addr	Decrease CX and jump to label addr if CX is not zero and the Zero flag is not set.	LOOPNZ LoopLabel		
LOOPNE addr LOOPZ addr LOOPE addr	Decrease CX and jump to label addr if CX is not zero and the Zero flag is set.	LOOPZ LoopLabel		
MOV dest,src	Move a value from source src to destination dest.	MOV AX,BX		
MOVSB MOVSW	Move a value from DS:SI to ES:DI. This command can be combined with repeat command REP, to repeat CX times.	REPZ MOVSB		
MUL src	Multiply unsigned number AX or DX.AX by src.AX=AL*src (8 bit) or DX.AX=AX*src (16 bit)	MUL CX		OszapC
NEG dest	Negate dest (Twos Complement of the number).	NEG AL		
NOP	No Operation. This command has no effect on any registers or memory.	NOP		
NOT dest	Invert/Flip all the bits of dest.	NOT dest		
OR dest,src	Logically ORs the src and dest parameter together.	OR AX,BX		OSZaPC
OUT port,src	Send an 8 bit byte or 16 bit word from src (either AX or AL) to hardware port number port.	OUT 100,AL		
POP reg	Pop a pair of bytes off the stack into 16 bit register reg.	POP ES POPF	AX, BX, CX, DX, SI, D	
PUSH reg	Pop a pair of bytes off the stack into the 16 bit Flags register. Push a pair of bytes from 16 bit register reg onto the top of the stack.	PUSH AX		0 D I T S Z A P C
PUSHF	Push a pair of bytes from 16 bit register reg onto the top of the stack. Push a pair of bytes off the stack into the 16 bit Flags register.	PUSHF		
RCL dest,count	Rotate bits in Destination dest to the Left by count bits, with the carry flag acting as an extra bit.	RCL AX,1		0 C
RCR dest,count		RCR AX,1		0 C
REP stringop	Repeat string operation stringop while CX>0. Decrease CX after each iteration	REP LODSW		
REPE stringop REPNE stringop	Repeat string operation stringon while the Z flag is set and CX>0. Decrease CX each time	REPZ CMPSB		
REPNE stringop REPNZ stringop		REPNZ CMPSB		
ROL dest count	Return from a subroutine. Rotate hits in Destination dest to the Left by count hits	RET ROL AX,1		C
·	Rotate bits in Destination dest to the Left by count bits Rotate bits in Destination dest to the Right by count bits	ROL AX,1 ROR AL,1		0 C
SAHF	Store AH to the Flags. This only transfers the main flags: SZ-A-P-C.	SAHF		- S Z A P C
	Shift the bits for Arithmetic in Destination dest to the Left by count bits.	SAL AX,1		0 C
SAR dest,count	•	SAR AX,1		0 C
SBB dest,src	Subtract src and the Borrow (carry flag) from dest.	SBB AL,BL		OSZAPC
SCASBSCASW	Scan ES:DI and compare to AX or AL. This command can work in bytes or words. (Like CMP)	REPZ SCASB		OSZAPC
SHL dest,count	Shift the bits logically Left in destination dest by count bits.	SHL AX,1		0 C 0 C
SHR dest,count	Shift the bits logically Right in destination dest by count bits. Set the Carry Flag. C flag will be set to 1.	SHR AX,1 STC		C
STD	Set the Direction Flag. D flag will be set to 1. This is used for 'String functions'.	STD		D
STI	Set the Interrupt enable flag. I flag will be set to 1. This is used to 1 of thing functions.	STI		I
STOSBSTOSW	Store AX or AL to ES:DI. This command can work in bytes or words.	REP STOSB		
SUB dest,src	Subtract src from dest.	SUB AX,BX		OSZAPC
TEST dest,src	Test dest, setting the flags in the same way a logical "AND src" would. Dest unchanged	TEST BX,64h		OSZAPC
WAIT	Wait until the busy pin of the CPU is inactive.	WAIT		OSZAPC
XCHG reg1,reg2	Exchange the contents of registers reg1 and reg2. Translate AL using lookup table DS:BX. AL is read from memory address [DS:BX+AL].	XCHG BH,AL XLAT		
XOR dest,src	Logical XOR (eXclusive OR) of bits in dest with src.	XOR AX,BX		OSZaPC
		,		
Command	Details Programme Control of the Con	Flans		

Command		Details	Flags
JA / JNBE	Above / Not Below or Equal	(For Unsigned Numbers)	C=0 AND Z=0
JBE / JNA	Below or Equal / Not Above	(For Unsigned Numbers)	C=1 OR Z=1

JC JB / JNAE	Carry Below / Not Above or Equal (For Unsigned Numbers)	C=1	
JE / JZ	Equal / zero	Z=1	
JG / JNLE	Greater / Not Less than or Equal (For Signed Numbers)	((S XOR O) OR Z)=0	
JGE / JNL	Greater or Equal / Not Less (For Signed Numbers)	(S XOR O)=0	
JLE / JNG	Less than or Equal / Not Greater (For Signed Numbers)	((S XOR O) OR Z)=1	
JL / JNGE	Less / Not Greater or Equal (For Signed Numbers)	(S XOR O)=1	
JNC JAE / JNB	No CarryAbove or Equal / Not Below (For Unsigned Numbers)	C=0	
JNE / JNZ	Not Equal / Not zero	Z=0	
JNO	Not Overflow	O=0	
JNP / JPO	Not Parity / Parity Odd	P=0	
JNS	Not Signed (not negative)	S=0	
JO	overflow	O=1	
JP / JPE	Parity / Parity Equal (bits 0-7 only)	P=1	
JS	Signed (is positive)	S=1	

PDP-11										
							ode (Octa			
Mnemonic HALT	Function Stop	Notes	NZVC	0	E D C	B A 9	8 7 6	5 4 3	2 1 0	
WAIT	Stop until interrupt			0		0	0	0	1 1	
RESET	Reset all IO devices			0	0	0	0	0	5	
NOP	NoOp Set to Toro		0100	0	0	0	2	4	0	
CLR{B} dest INC{B} dest	Set to zero Add 1		0100	0		5 5	0 2	D D	D D	
DEC{B} dest	Sub 1		* * * -	0	0	5	3	D	D	
ADC{B} dest	Add with carry		* * * *	0	1	5	5	D	D	
SBC{B} dest	Subtract with Carry		****	0	0	5 5	6 7	D D	D D	
TST{B} dest NEG{B} dest	Set Condition Codes Negate		* * * *	0		5	4	D	D	
COM{B} dest	Ones compliment		* * 0 1	0		5	1	D	D	
ROR(B) dest	Rotate Right (through Carry)		* * * *	0	0	6	0	D D	D D	
ROL{B} dest ASR{B} dest	Rotate Left (through Carry) Arithmatic shift Right		* * * *	0	0	6	2	D	D	
ASL{B} dest	Arithmatic shift Left		* * * *	0	0	6	3	D	D	
SWAB SXT	Swap Bytes in a word		* * * 0	0	0	0 6	3	D D	D D	
MUL s,d	Sign Extend Multiply (if even registers 32 bit (r0+r1)	else 16 (r1)	-*0- **0*	0	7	0	7 R	S	S	mul r0,r2 ;R2.R3=R0*R2
DIV src,dest	Divide (dest reg must be even - r1=res		* * * *	0	7	1	R	S	s	,
ASH n,reg	Arithmatic shift (by n bits) (n Positive=F	Right Neg=Left	****	0	7	2	R	S	S	
ASHC n,reg XOR src,dest	Arithmatic shift combined (32 bit pair) Flip bits of dest with src		* * * *	0	7	3 4	R R	S S	S	div r0,r2 ;R2=R2.R3/R0 Rmdr in R3
MOV{B} src,dest	Move src to dest			В	1	S	S	D	D	divio,iz ,itz itz.ito/ito itilia iii ito
ADD src,dest	Add src to dest			0	6	S	S	D	D	
SUB src,dest CMP{B} src,dest	Subtract s from d Compare (set flags like src-dest)			1 B	6 2	S S	S S	D D	D D	
BIS{B} src,dest	Bit Set (OR)			В	5	S	S	D	D	
BIC{B} src,dest	Bit Clear (for AND use with COM/ ^C to			В	4	s	S	D	D	
BIT{B} src,dest BR ofst	Bit Test (like AND but doesn't alter des Branch Always	t)		B 0	3	S 0	S 1 B B	D B	D B	
BNEofst	Branch Not Equal	Z=0		0	0	1	1 B B 0 B B	B	B B	
BEQ ofst	Branch Equal	Z=1		0	0	1	1 B B	В	В	
BPL ofst	Branch if plus	N=0		1	0	0	0 B B	В	В	
BMI ofst BVC ofst	Branch if minus Branch if Overflow Clear	N=1 V=0		1	0	0 2	1 B B 0 B B	B B	B B	
BVS ofst	Branch if Overflow Set	V=1		1	0	2	1 B B	В	B	
BHIS ofst	Branch if higher or same	C=0		1	0	3	0 B B	В	В	
BCC ofst	Branch if carry clear	C=0 C=1		1	0	3	0 B B 1 B B	B B	B B	
BLO ofst BCS ofst	Branch if lower Branch if carry set	C=1		1	0	3	1 B B	B	B B	
BGE ofst	Branch if greater than or equal to	N xor V=0		0	o	2	0 B B	В	B	
BLT ofst	Branch if less than	N xor V=1		0	0	2	1 B B	В	В	
BGT ofst BLE ofst	Branch if greater than Branch on less than or equal to	not $(N \times V)=0$ not $(N \times V)=1$		0	0	3	0 B B 1 B B	B B	B B	
BHI ofst	Branch on higher than	C not Z =0		1	0	1	0 B B	В	B	
BLOS ofst	Branch on lower than or same as	C not Z = 1		1	0	1	1 B B	В	В	
JMP dest SOB	Jump Subtract 1 and branch			0	7	0 7	1 R	A N	A N	
JSR reg,Label	Jump to subroutine, setting reg to retur	n address		0	ó	4	R	A	A	
RTS reg	Return from subroutine to address reg,		m the stack		0	0	2	0	R	
RTI	Return from interrupt/trap	T: 400		0	0	0	0	0	2	
TRAP BPT	Trap Breakpoint trap	T>=400		1 0	0	4	7 0	T 0	7 3	
IOT	I/O Trap			0	o	0	0	0	4	
EMT	Emulator Trap	T<=400		1	0	4	T	T	T	
RTT SPL	Return from trace trap Set priority level			0	0	0	0 2	0	6 N	
-	Clear Multiple			0	0	0	2	1 0 N		
CLC	Clear Carry flag	C=0		0	0	0	2	4	1	
CLV CLZ	Clear Overflow flag Clear Zero flag	V=0 Z=0		0	0	0	2 2	4	2 4	
CLN	Clear Negative flag	N=0		0	0	0	2	5	0	
ccc	Clear Condition codes	All=0		0	0	0	2	5	7	
- SEC	Set Multiple	C=1		0	0	0	2	1 1 N		
SEV	Set Carry flag Set Overflow flag	V=1		0	0	0	2 2	6	1 2	
SEZ	Set Zero flag	Z=1		0	0	0	2	6	4	
SEN	Set Negative flag	N=1		0	0	0	2	7	0 7	
SCC	Set condition codes	All=1			4 2 1	0 4 2 1	4 2 1	7 4 2 1	7 4 2 1	
.ASCII	Ascii									
.ASCIZ	Ascii followed by zero byte									
ALIGN ORG	Base address Base address									
.BYTE	Byte data									
.WORD	Word data									
.BLKW n .BLKB n	output n zero words output n zero bytes									
.END	end of source code									
INCLUDE	include another file					1	1	1		
CALL addr RETURN	Call subroutine – same as JSR PC,n Return from subroutine -same as RTS	PC								
MFPS r	Move from Processor status to register									
MTPS r	move to Processor Status from reg r									
{B}=Byte (0=word /	1 = Byte)									

TMS9900	Meaning	Bytes	Fmt	LA=CVPX	Details	Example
A S,D	Add	A000	1	*****	Dotallo	A @>100,R2
AB S,D	Add Bytes	B000	1	* * * * * * _		7. 6. 100,1.2
C S,D	Compare	8000	1	* * *		
CB S,D	Compare Bytes	9000	1	* * * * -		CB R1,R2
S S,D	Subtract	6000	1	* * * * *		02 111,112
SB S,D	Subtract Bytes	7000	1	* * * * * * _		
soc	Set ones Corresponding (OR)	E000	1	* * *		
SOCB	Set ones Corresponding Bytes (OR)	F000	1	* * * * -		
SZC	Set Zeros Corresponding (Reverse AND)	4000	1	* * *		
SZCB	Set Zeros Corresponding Bytes (Reverse AND)	5000	1	* * * * -		
MOV S,D	Move	C000	1	* * *		
MOVB S,D	Move Bytes	D000	1	* * * * -		
COC S,D	Compare Ones Corresponding	2000	3	*	ones in S also in D?	COC R10,RII
CZC S,D	Compare Zeros Corresponding	2400	3	*		
XOR S,D	Flip Bits	2800	3	* * *		
MPY S,D	Multiply s*d – result in d,d+1	3800	9			MPY R2,R3
DIV Ss,D	Divide d,d+1 by s, result in d,d+1	3C00	9	*		DIV @>FEOO,R5
XOP A,n	Extend Operation	2800	9	222222	Load new settings from address at vector	AXOP @>FFOO,4
B R	Branch to register R / @addr	0440	6		R->PC	B *R2
BL A	Branch and Link address A	0680	6		PC→WR11, SA→PC	
BLWP	Branch and Load Workspace Pointer	0400	6		(A)→WP (A+2)→PC ST→R15, PC→R14	, WP→R13 (addr is 2 pntrs)
CLR D	Clear Operand	04C0	6			, ,
SETO	Set To Ones	0700	6			
INV D	Invert	0540	6	* * *		
NEG D	Negative	0500	6	* * * * *		
ABS D	Absolute Value	0740	6	* * * * *		
SWPB D	Swap Bytes	06C0	6			
INC D	Increment	0580	6	* * * * *		
INCT D	Increment by 2	05C0	6	* * * * *		
DEC D	Decrement	0600	6	* * * * *		
DECT D	Decrement by 2	0640	6	* * * * *		
X D	Execute	0480	6	222222		
LDCR S,B	Load Communication Register	3000	4	* * * 1 -	Transfer B bits from S	
STCR S,B	Store Communication Register	3400	4	* * * 1 -	Transfer B bits from S	
SBO n	Set CRU Bit to 1	1D00	Χ			SBO 4
SBZ n	Set CRU Bit to 0	1E00	Χ			
TB n	Test CRU Bit	1F00	Χ	*		
JEQ n	Jump Equal	1300	2		Jump to offset n	JEQ \$+4
JGT	Jump Greater Than (Signed)	1500	2			
JH	Jump High	1B00	2			
JHE	Jump Higher or Equal	1400	2			
JL	Jump Lower	1A00	2			
JLE	Jump Lower or Equal	1200	2			
JLT	Jump Less Than (Signed)	1100	2			
JMP	Jump	1000	2			JMP \$
JNC	Jump No Carry	1800	2			
JNE	Jump Not Equals	1600	2			
JNO	Jump No Overflow	1900	2			
JOC	Jump On Carry	1800	2			
JOP	Jump Odd Parity	1C00	2			OLA DI C
SLA D,B	Shift Left Arithmatic	0A00	5	* * * * *	Shift D by B bits (0=use R0)	SLA RI,O
SRA D,B	Shift Right Arithmatic	0800	5	* * * *	Shift D by B bits (0=use R0)	SRA RI,2
SRC D,B	Shift Right Circular	0B00	5	* * * *	Circular shift D by B bits (0=use R0)	SRC R5,4
SRL D,B	Shift Right Logical	0900	5	* * * * *		
AI D,nn	And Immediate	0220	8	* * * * *	Add n to reg D	ALDO - EE
ANDI D,nn	And Immediate	0240	8	* * *		AI R2,>FF
CI D,nn	Compare Immediate	0280	8	* * * * * *	Compare D to n	CI R2,>10E
LI D,nn	Load Immediate	0200	8			
ORI	Or Immediate	0260	8	* * *		LWDL>ECOO
LWPI A	Load Workspace Pointer Immediate	02E0	X		A→WP	LWPI >FCOO
LIMI	Load Interrupt Mask	0300	X			
STST	Store Status Register	02C0	X			CTWD D2
STWP	Store Workspace Pointer	02A0	X	* * * * * *		STWP R2
RTWP	Return from Context Switch	0380	X		R13→WP, R14→PC, R15→ST	
IDLE	Idle Poset	0340	7			
RSET	Reset	0360	7			
CKON	User Defined	03C0	7			
CKON	User Defined	03A0	7			
LREX	User Defined	03E0	7			

M	IPS				
V	Instruction	Delay?	RISCV Example		FEDCBA9876543210FEDCBA9876543210
	LA dest,addr LB dest,addr	Load	LA	Load address Load byte	LUI\$at,>label ORIRd,\$at, <label< td=""></label<>
i	LBU dest,addr	R3000 Load R3000 Load	LB LBU	Load byte unsigned	100100sssstttt
	LH dest,addr	R3000 Load	LH	Load halfword	100001ssssttttiiiiiiiiiiiiiiiii
i	LHU dest,addr LW dest,addr	R3000 Load R3000 Load	LHU LW	Load halfword unsigned Load word	100101ssssstttt iiiiiiiiiiiiiiiiiiiiiiiiiiiii
	LWL dest,addr	R3000		Load word left (can Load partial data from unword aligned data)	100010sssstttt i i i
	LWR dest,addr LD dest,addr	R3000		Load word right (can Load partial data from unword aligned data) Load double	1 0 0 1 1 0 s s s s s s t t t t t i i i i i i i i i i i i i i
	ULH dest,addr				LB Rd,4(Rs) LBU \$at,3(Rs) SLL Rd,Rd,8 OR Rd,Rd,\$at
	ULHU dest,addr ULW dest,addr				LBU Rd,4(Rs) LBU \$at,3(Rs) SLL Rd,Rd,8 OR Rd,Rd,\$at LWL Rd, 6(Rs) LWR Rd,3(Rs)
	LI dest,expr		LI	Load Immediate	LUI \$at,>imm ORI Rd,\$at, <imm ori="" rt,\$0,imm<="" td=""></imm>
!	LUI dest,expr SB source,addr	R3000 R3000	LUI SB	Load Upper Immediate 0xFFFF Store Byte	0 0 1 1 1 1 0 0 0 0 0 0 t t t t t i i i i i i i i i i i i i i
	SD expr,dest	110000		Store Doubleword	
ı	SH expr,dest SWL expr,dest	R3000 R3000	SH	Store Halfword Store Word Left (can Store partial data from unword aligned data)	101001sssstttt
	SWR expr,dest	R3000		Store Word Right (can Store partial data from unword aligned data)	101110sssstttt iiiiiii
ı	SW expr,dest USH dest,expr	R3000	SW	Store Word Unaligned Store Half Word	1 0 1 0 1 1 s s s s s t t t t t i i i i i i i i i i
	USW dest,expr			Unaligned Store Word	SWL Rd,6(Rs) SWR Rd,3(Rs)
R	ADD rd, rs, rt ADDI rt,rs,imm	R3000 R3000	ADD ADDI	Add (Signed) Add Immediate	0 0 0 0 0 0 0 s s s s s s t t t t t d d d d d d 0 0 0 0 0 1 0 0 0 0 0 0 0 0
i	ADDIU dest,src1,imm	R3000	ADDI	Add Immediate Unsigned	001001ssssstttt iiiiiiiiiiiiiiiii
R R	ADDU dest,src1,src2 AND rd,rs,rt	R3000 R3000	AND		0 0 0 0 0 0 0 s s s s s t t t t t d d d d d 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
1	ANDI dest,src1,imm	R3000 R3000	ANDI	And Immediate	001100ssssstttt iiiiiiii
R R	DIV dest,src1,src2 DIVU dest,src1,src2	R3000	DIV		0 0 0 0 0 0 0 s s s s s t t t t t 0 0 0 0
R	XOR dest,src1,src2	R3000 R3000	SIVO	Exclusive OR	0 0 0 0 0 0 0 s s s s s t t t t t d d d d d 0 0 0 0 0 0 1 0 1 1 0
	XORI dest,src,imm MUL dest,src1,src2	R3000	MIII	Xor Immediate	001110ssssttttt iiiiiii
	MULO dest,src1,src2		MUL MULH	• • •	MULT Rs,Rt MFLO Rd MULT Rs,Rt MFHI \$at MFLO Rd SRA Rd,Rd,31 BEQ \$at,Rd,ok BREAK \$0 ok:MFLO Rd
_	MULOU dest,src1,src2		MULHU	Multiply with Overflow Unsigned	MULTU Rs,Rt MFHI \$at BEQ \$at, \$0, ok ok: BREAK \$0 MFLO Rd
R R	NOR dest,src1,src2 OR dest,src1,src2	R3000 R3000	OR		0 0 0 0 0 0 0 s s s s s s t t t t t d d d d d 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0
I	ORI dest,src1,imm	R3000	ORI	OR Immediate	001101sssstttt iiiiiiii
	SEQ dest,src1,src2 SGT dest,src1,src2				BEQ Rt,Rs,yes ORI Rd,\$0,0 BRQ \$0,\$0,skip yes:ORI Rd,\$0,1 skip: SLT Rd, Rt, Rs
	SGE dest,src1,src2			Set Greater/Equal	BNE Rt,Rs,yes ORI Rd,\$0,1 BEQ \$0,\$0,skip yes:SLT Rd,Rt,Rs skip:
	SGEU dest,src1,src2 SGTU dest,src1,src2				BNE Rt,Rs,yes ORI Rd,\$0,1 BEQ \$0,\$0,skip yes:SITU Rd,Rt,Rs skip: SLTU Rd, Rt, Rs
R	SLT dest,src1,src2 SLTI dest,src,imm	R3000			0 0 0 0 0 0 0 s s s s s t t t t t d d d d d 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0
i	SLTIU dest,src,imm	R3000 R3000		Set on Less Than Immediate Unsigned	001011ss ssstttt iiiiiiii iiiiiii
	SLE dest,src1,src2 SLEU dest,src1,src2				BNE Rt,Rs,yes ORI Rd,\$0,1 BEQ \$0,\$0,skip yes:SLT Rd, Rs, Rt skip: BNE Rt,Rs,yes ORI Rd,\$0,1 BEQ \$0,\$0,skip yes:SLTU Rd,Rs,Rt skip:
R	SLTU dest,src1,src2	R3000		Set Less Unsigned	0 0 0 0 0 0 0 s s s s s t t t t t d d d d d 0 0 0 0 0 1 0 1 0 1 0 1
R	SNE dest,src1,src2 SUB dest,src1,src2	R3000		Set Not Equal Subtract (With Overflow)	BRQ Rt,Rs,yes ORI Rd,\$0,1 BEQ \$0,\$0,\$kip yes:ORI Rd,\$0,0 skip:
R	SUBU dest,src1,src2	R3000		Subtract (Without Overflow)	0 0 0 0 0 0 0 s s s s s t t t t t d d d d d 0 0 0 0 0 1 0 0 0 1 1
	REMU dest,src1,src2 REMU dest,src1,src2		REMU REMU		BNE Rt,\$0,8 BREAK \$0 DIV Rs,Rt MFHI Rd BNE Rt,\$0,ok BREAK \$0 ok: DIVU Rs,Rt MFHI Rd
	ROL dest,src1,src2 ROR dest,src1,src2			Rotate Left (Reg or imm) Rotate Right (Reg or imm)	SUBU \$at,\$0,Rt SRLV \$at,Rs,\$at SLLV Rd,Rs,Rt OR Rd,Rd,\$a SRL \$at,Rs,32-sa SLL Rd,Rs,sa OR Rd,Rd,\$at SUBU \$at,\$0,Rt SLLV \$at,Rs,\$at SRLV Rd,Rs,Rt OR Rd,Rd,\$at SLL \$at,Rs,32-sa SRt Rd,Rs,sa OR Rd,Rd,\$at
	SRA dest,src1,imm	R3000	SRAI	Shift Right Arithmatic Immediate	000000000000tttttdddddiii ii000011
R	SRAV dest,src1,src2 SLL dest,src, imm	R3000 R3000	SRA SLLI		0 0 0 0 0 0 0 s s s s s s t t t t t d d d d d 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
R	SLLV dest,src1,src2	R3000	SLL	Shift Left Logical by Variable	000000ssssttttdddddd00000001000
R	SRL dest,src, imm SRLV dest,src1,src2	R3000 R3000	SRLI SRL	Shift Right Logical Immediate Shift Right Logical by Variable	0 0 0 0 0 0 0 0 0 0 0 0 t t t t t d d d d
	ABS dest,src NEG dest,src		NEG		ADDU Rd,\$0,Rs BGEZ Rs,1 SUB Rd,\$0,Rs SUB Rd,\$0,Rs
	NEGU dest,src			Negate (Unsigned)	SUBU Rd,\$0,Rs
	NOT dest,src MOVE dest,src		NOT MV		NOR Rd,Rs,\$0 ADDU Rd,\$0,Rs
R	MULT src1,src2	R3000		Multiply result in HI/LOW (leave 2 instructions before next Mult/Div)	000000sssstttt 000000000011000
R J	MULTU src1,src2 J addr	R3000 R3000	J	Jump	0 0 0 0 0 0 0 s s s s s t t t t t 0 0 0 0
J	JAL addr JALR return,reg	R3000	JAL	Jump and link	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
R	JR reg	R3000 R3000	JA:LR JR	Jump to address in register	0 0 0 0 0 0 0 s s s s s s 0 0 0 0 0 0 0
I	BEQ src1, src2, label BGT src1, src2, label	R3000 Branch Branch			0 0 0 1 0 0 s s s s s t t t t t i i i i i i i i i i
	BGE src1, src2, label	Branch	BGE	Branch on Greater/Equal	SLT \$at, rs,rt BEQ \$at,\$0,Label
	BGEU src1, src2, label BGTU src1, src2, label	Branch Branch			SLTU \$at, rs,rt BEQ \$at,\$0,Label SLTU \$at,Rs,Rt BNE \$at,\$0,Label
	BLT src1, src2, label	Branch	BLT	Branch on Less than	SLT \$at,Rs,Rt BNE \$at,\$0,Label
	BLE src1, src2, label BLEU src1, src2, label	Branch Branch			SLT \$at, Rt,Rs BEQ \$at,\$0,Label SLTU \$at, Rt,Rs BEQ \$at,\$0,Label
,	BLTU src1, src2, label	Branch	BLTU	Branch on Less Unsigned	SLTU \$at,Rs,Rt BNE \$at,\$0,Label
1	BNE src1, src2, label BEQZ src1, label	R3000 Branch	BNE		0 0 0 1 0 1 s s s s s s t t t t i i i i i i i i i i i i i i i
	BGEZ src1, label BGTZ src1, label	R3000			000001sssss00001
	BGEZAL src1, label	R3000		Branch on Greater or equal to zero	000001ssss10001
1	BLTZAL src1, label BLEZ src1, label	R3000		Branch on less than zero and link	0 0 0 0 0 0 1 s s s s s 1 0 0 0 0 0 i i i i i i i i i i i i i i
	BLTZ src1, label	R3000 R3000		Branch on less than zero	000001ss sss00000 iiiiiiii iiiiii
	BNEZ src1, label B label				BNE Rs,\$0,Label Bgez \$0,label
	BAL label			Branch and Link	00000100 00010001 111111111111111111111
	BREAK breakcode RFE	R3000			000000111111111111111111111111111111111
_	SYSCALL		ecall	System Call	000000000000000000001100
R R	MFHI register MTHI register	R3000 R3000			0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
R R	MFLO register MTLO register	R3000		Move from LOW to register	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 d d d d d d 0 0 0 0 0 0 1 0 0 1 0
K	LWCz dest,addr	R3000 R3000	LWC2 \$1,0(a0)	Load Word Coprocessor Z	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P	SWCz source,addr MFCz dest-gpr,source	R3000	SWC2 \$1,0(a0) MFC2 a0,\$1	Store Word Coprocessor z Move from Coprocessor z	1 1 1 0 0 1 b b b b f f f f f f i i i i i i i i i i
ĸ	MTCz src-gpr, destination	R3000 R3000	MFC2 a0,\$1 MTC2 a0,\$1	Move to Coprocessor z	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	CFCz dest-gpr, source COPz cofun	R3000 R3000	CFC2 a0,\$1 COP2 1	control from Coprocessor z Perform Coprocessor operation cofun on coprocessor z	
	CTCz src-gpr,destionation		CTC2 a0,\$1	move reg RT into control register RD of coprocessor z	
	NOP			No Operation	OR \$0,\$0,\$0
	BCzF label			Branch Coprocessor z false	
	BCzT label			Branch Coprocessor z true	
	Cz expr	Store		Coprocessor z operation	

KIS	Risc-V								
		Instruction	MIPS		seudocode	F E D C B A 9 8 7 6 5 4			
U U			LUI	Load Upper Immediate (Top 20 bits of rd) rd		Imm [31:1 Imm [31:1	-	rd	0010111
UJ		AUIPC rd,offset JAL rd,offset	JAL	• •	← pc + offset ← pc + len(inst)··· pc ← pc + off	[20][10:1]	د] [11][19:12]	rd rd	0010111 1101111
I	RV32I	JALR rd.rs1.offset		•	\leftarrow pc + len(inst)pc \leftarrow pc + on \leftarrow pc + len(inst)pc \leftarrow (rs1+off) \land -2	Imm [11:0]	Rs1 000		1100111
SB		BEQ rs1,rs2,offset	BEQ		rs1 = rs2 then pc ← pc + offset	Imm [12][10:0] Rs2	Rs1 000		1100011
SB		BGE rs1,rs2,offset			rs1 ≥ rs2 then pc ← pc + offset	Imm [12][10:3] Rs2	Rs1 10	Im [11][4:1]	1100011
SB	RV32I	BGEU rs1,rs2,offset	BGEU	Branch Greater than Equal Unsigned if r	rs1 ≥ rs2 then pc ← pc + offset	Imm [12][10:5] Rs2	Rs1 11	Im [11][4:1]	1100011
SB		BLT rs1,rs2,offset	BLT		rs1 < rs2 then pc ← pc + offset	Imm [12][10:2] Rs2	Rs1 100		1100011
SB					rs1 < rs2 then pc ← pc + offset	Imm [12][10:4] Rs2	Rs1 110		1100011
SB	RV32I RV32I	BNE rs1,rs2,offset LB rd,offset(rs1)	BNE LB		rs1 ≠ rs2 then pc ← pc + offset ← s8[rs1 + offset]	Imm [12][10:1] Rs2 Imm [11:0]	Rs1 000		1100011 0000011
i	RV32I	LBU rd.offset(rs1)	LBU	•	← u8[rs1 + offset]	Imm [11:0]	Rs1 100		0000011
i		LH rd,offset(rs1)	LH		← s16[rs1 + offset]	Imm [11:0]	Rs1 00		0000011
I	RV32I	LHU rd,offset(rs1)	LHU	Load Half Unsigned rd	← u16[rs1 + offset]	Imm [11:0]	Rs1 10	1 rd	0000011
I	RV32I	LW rd,offset(rs1)	LW		← s32[rs1 + offset]	Imm [11:0]	Rs1 010		0000011
S		SB rs2,offset(rs1)	SB	•	[rs1 + offset] ← rs2	Imm [11:5] Rs2		0 Imm [4:0]	0100011
S S	RV32I RV32I	SH rs2,offset(rs1)	SH SW		6[rs1 + offset] ← rs2 2[rs1 + offset] ← rs2	Imm [11:5] Rs2 Imm [11:5] Rs2		1 Imm [4:0] 0 Imm [4:0]	0100011 0100011
ı		SW rs2,offset(rs1) ADDI rd,rs1,imm			← rs1 + sx(imm)	Imm [11:0]	Rs1 000		0010011
i		ANDI rd,rs1,imm			← ux(rs1) ∧ ux(imm)	Imm [11:0]	Rs1 11		0010011
I		ORI rd,rs1,imm			← ux(rs1) V ux(imm)	Imm [11:0]	Rs1 110		0010011
I	RV32I	SLTI rd,rs1,imm	SLTI	Set Less Than Immediate (rd=1/0) rd	← sx(rs1) < sx(imm)	Imm [11:0]	Rs1 010	0 rd	0010011
I	RV32I	SLTIU rd,rs1,imm	SLTIU	Set Less Than Immediate Unsigned $(rd=1rd$	$\leftarrow ux(rs1) < ux(imm)$	Imm [11:0]	Rs1 01	1 rd	0010011
I		XORI rd,rs1,imm	XORI		← ux(rs1) ⊕ ux(imm)	Imm [11:0]	Rs1 100		0010011
R	RV32I	SLLI rd,rs1,imm		_	← ux(rs1) ≪ ux(imm)	0000000 shamt	Rs1 00		0010011
R R		SRLI rd,rs1,imm		5 5	← ux(rs1) ≫ ux(imm) ← ox(rs1) ≫ ux(imm)	0000000 shamt 0100000 shamt	Rs1 10		0010011 0010011
R S		SRAI rd,rs1,imm ADD rd,rs1,rs2	ADD	-	← sx(rs1) ≫ ux(imm) ← sx(rs1) + sx(rs2)	0100000 shamt 0000000 shamt	Rs1 10		0010011
s S		AND rd,rs1,rs2	700		$\leftarrow sx(rs1) + sx(rs2)$ $\leftarrow ux(rs1) \wedge ux(rs2)$	0000000 shamt	Rs1 11		0110011
S		OR rd,rs1,rs2			← ux(rs1) V ux(rs2)	0000000 shamt	Rs1 110		0110011
S	RV32I	SUB rd,rs1,rs2			← sx(rs1) - sx(rs2)	0100000 shamt	Rs1 000	0 rd	0110011
S	RV32I	SLL rd,rs1,rs2	SLLV		← ux(rs1) ≪ rs2	0000000 shamt	Rs1 00	1 rd	0110011
S		SLT rd,rs1,rs2			← sx(rs1) < sx(rs2)	0000000 shamt	Rs1 010		0110011
S		SLTU rd,rs1,rs2		_	← ux(rs1) < ux(rs2)	0000000 shamt	Rs1 01		0110011
s s		SRL rd,rs1,rs2			← ux(rs1) ≫ rs2	0000000 shamt	Rs1 10		0110011
S		SRA rd,rs1,rs2		-	← sx(rs1) ≫ rs2 ← ux(rs1) ⊕ ux(rs2)	0100000 shamt 0000000 shamt	Rs1 10		0110011 0110011
s I	RV32I	XOR rd,rs1,rs2 FENCE pred,succ	XUK	Fence rd	← ux(rs1) ⊕ ux(rs2)	Fm Pred Suc	Rs1 000		1110011
i	RV32I	FENCE.I		Fence Instruction		Imm	Rs1 00		0001111
I	RV64I	LWU rd,offset(rs1)			← u32[rs1 + offset]	Imm [11:0]	Rs1 110		0000011
	RV64I	LD rd,offset(rs1)		Load Double rd	← u64[rs1 + offset]	Imm [11:0]	Rs1 01	1 rd	0000011
	RV64I	SD rs2,offset(rs1)		Store Double u64	4[rs1 + offset] ← rs2	Imm [11:5] Rs2	Rs1 01	1 Imm [4:0]	0100011
	RV64I	SLLI rd,rs1,imm	SLL		← ux(rs1) ≪ sx(imm)	000000 shamt	Rs1 00		0010011
	RV64I	SRLI rd,rs1,imm	SRL		← ux(rs1) ≫ sx(imm)	000000 shamt	Rs1 10		0010011
	RV64I	SRAI rd,rs1,imm	SRA		← sx(rs1) ≫ sx(imm)	010000 shamt	Rs1 10		0010011
	RV64I RV64I	ADDIW rd,rs1,imm SLLIW rd,rs1,imm			← s32(rs1) + imm ← s32(u32(rs1) ≪ imm)	Imm [11:0] 000000 shamt	Rs1 000		0011011 0011011
	RV64I	SRLIW rd,rs1,imm		<u> </u>	← s32(u32(rs1) ≫ imm)	000000 shamt	Rs1 10		0011011
	RV64I	SRAIW rd,rs1,imm			← s32(rs1) ≫ imm	010000 shamt	Rs1 10		0011011
		ADDW rd,rs1,rs2			← s32(rs1) + s32(rs2)	000000 Rs2	Rs1 000		0111011
	RV64I	SUBW rd,rs1,rs2			← s32(rs1) - s32(rs2)	010000 Rs2	Rs1 000	0 rd	0111011
	RV64I	SLLW rd,rs1,rs2		<u> </u>	← s32(u32(rs1) ≪ rs2)	000000 Rs2	Rs1 00		0111011
	RV64I	SRLW rd,rs1,rs2		5 5	← s32(u32(rs1) ≫ rs2)	000000 Rs2	Rs1 10		0111011
	RV64I	SRAW rd,rs1,rs2			← s32(rs1) ≫ rs2	010000 Rs2	Rs1 10		0111011
nn S	RV32M	MUL rd,rs1,rs2 MULH rd,rs1,rs2	MULI		$\leftarrow ux(rs1) \times ux(rs2)$ $\leftarrow (av(rs1) \times av(rs2)) \times v(ss2)$	0000001 Rs2 0000001 Rs2	Rs1 000		0110011 0110011
s S	RV32M RV32M	MULHSU rd,rs1,rs2			← (sx(rs1) × sx(rs2)) » xlen ← (sx(rs1) × ux(rs2)) » xlen	0000001 Rs2	Rs1 00		0110011
S	RV32M	MULHU rd,rs1,rs2			← (ux(rs1) × ux(rs2)) » xlen	0000001 Rs2	Rs1 01		0110011
S	RV32M	DIV rd,rs1,rs2	DIV		← sx(rs1) ÷ sx(rs2)	0000001 Rs2	Rs1 100		0110011
S		DIVU rd,rs1,rs2	DIVU		← ux(rs1) ÷ ux(rs2)	0000001 Rs2	Rs1 10		0110011
S	RV32M	REM rd,rs1,rs2	REM	Remainder Signed rd	← sx(rs1) mod sx(rs2)	0000001 Rs2	Rs1 110		0110011
S	RV32M	REMU rd,rs1,rs2	REMU		← ux(rs1) mod ux(rs2)	0000001 Rs2	Rs1 11		0111011
S	RV64M	MULW rd,rs1,rs2		•	$\leftarrow u32(rs1) \times u32(rs2)$	0000001 Rs2	Rs1 000		0111011
S S	RV64M RV64M	DIVW rd,rs1,rs2 DIVUW rd,rs1,rs2			$\leftarrow s32(rs1) \div s32(rs2)$ $\leftarrow u32(rs1) \div u32(rs2)$	0000001 Rs2 0000001 Rs2	Rs1 100 Rs1 100		0111011 0111011
S S	RV64M	REMW rd,rs1,rs2		_	← u32(rs1) ÷ u32(rs2) ← s32(rs1) mod s32(rs2)	0000001 Rs2	Rs1 10		0111011
S	RV64M	REMUW rd,rs1,rs2			← u32(rs1) mod u32(rs2)	0000001 Rs2	Rs1 11		0111011
		.2byte		16-bit comma separated words (unaligned)					
		.4byte		32-bit comma separated words (unaligned)					
	Directive	.8byte		64-bit comma separated words (unaligned)					
		.half		16-bit comma separated words (naturally alig					
		.word		32-bit comma separated words (naturally alig					
		.dword		64-bit comma separated words (naturally alig	gnea)				
		.byte .dtpreldword		8-bit comma separated words 64-bit thread local word					
		.dtprelword		32-bit thread local word					
	Directive	.sleb128 expression		signed little endian base 128, DWARF					
	Directive	.uleb128 expression		unsigned little endian base 128, DWARF					
				emit string (alias for .string)					
		.string		emit string					
		.incbin "filename"		emit the included file as a binary sequence of	of octets				
		.zero integer		zero bytes					
		.align integer .balign b,[pad_val=0]		align to power of 2 (alias for .p2align) byte align					
		.p2align p2,[pad_val=0],max		align to power of 2					
				emit symbol_name to symbol table (scope GL	LOBAL)				
		.local symbol_name		emit symbol_name to symbol table (scope LO					
	Directive	.equ name, value		constant definition					
		.text		emit .text section (if not present) and make of					
				emit .data section (if not present) and make					
		.rodata		emit .rodata section (if not present) and make					
	Directive	.DSS		emit .bss section (if not present) and make c	current				

irective	.comm sym_nam,sz,aln		emit common object to .bss section		
irective	.common sym_name,sz,aln		emit common object to .bss section		
irective	.section sect		emit section (if not present, default .text	[{.text,.data,.rodata,.bss}]	
irective	.option opt		RISC-V options	{rvc,norvc,pic,nopic,push,pop}	
irective	.macro name arg1 [, argn]		begin macro definition ¥argname to subst	itute	
	.endm		end macro definition		
	.file "filename"		emit filename FILE LOCAL symbol table		
irective	.ident "string"		accepted for source compatibility		
irective	.size symbol, symbol		accepted for source compatibility		
irective	.type symbol, @function		accepted for source compatibility		
suedo	NOP		No operation	addi zero,zero,0	
suedo	∐ rd, imm	LI	Load immediate	(several expansions) (LUA+ADDI)	
	LA rd, symbol	LA	Load address	(several expansions)	
suedo	MV rd, rs1	MOVE	Copy register	addi rd, rs, 0	
suedo	NOT rd, rs1	NOT	One's complement	xori rd, rs, -1	
suedo	NEG rd, rs1	NEG	Two's complement	sub rd, x0, rs	
	NEGW rd, rs1		Two's complement Word	subw rd, x0, rs	
suedo	SEXT.W rd, rs1		Sign extend Word	addiw rd, rs, 0	
suedo	SEQZ rd, rs1		Set if = zero	sltiu rd, rs, 1	
suedo	SNEZ rd, rs1		Set if ≠ zero	sltu rd, x0, rs	
suedo	SLTZ rd, rs1		Set if < zero	slt rd, rs, x0	
suedo	SGTZ rd, rs1		Set if > zero	slt rd, x0, rs	
suedo	FMV.S frd, frs1		Single-precision move	fsgnj.s frd, frs, frs	
suedo	FAB.S frd, frs1		Single-precision absolute value	fsgnjx.s frd, frs, frs	
suedo	FNEG.S frd, frs1		Single-precision negate	fsgnjn.s frd, frs, frs	
suedo	FMV.D frd, frs1		Double-precision move	fsgnj.d frd, frs, frs	
suedo	FABS.D frd, frs1		Double-precision absolute value	fsgnjx.d frd, frs, frs	
suedo	FNEG.D frd, frs1		Double-precision negate	fsgnjn.d frd, frs, frs	
suedo	BEQZ rs1, offset		Branch if = zero	beq rs, x0, offset	
suedo	BNEZ rs1, offset		Branch if ≠ zero	bne rs, x0, offset	
suedo	BLEZ rs1, offset		Branch if ≤ zero	bge x0, rs, offset	
suedo	BGEZ rs1, offset		Branch if ≥ zero	bge rs, x0, offset	
suedo	BLTZ rs1, offset		Branch if < zero	blt rs, x0, offset	
suedo	BGTZ rs1, offset		Branch if > zero	blt x0, rs, offset	
suedo	BGT rs, rt, offset		Branch if >	blt rt, rs, offset	
suedo	BLE rs, rt, offset		Branch if ≤	bge rt, rs, offset	
suedo	BGTU rs, rt, offset		Branch if >, unsigned	bltu rt, rs, offset	
suedo	BLEU rs, rt, offset		Branch if ≤, unsigned	bltu rt, rs, offset	
suedo	J offset	J	Jump	jal x0, offset	
suedo	JR offset	JR	Jump register	jal x1, offset	
suedo	RET		Return from subroutine	jalr x0, x1, 0	

ARM		
Mnemonic	Description	Example
ADCccS Rn, Rm, Op2	Add With Carry.	ADC R0,R0,#4
ADDccS Rn, Rm, Op2	Add Op2 to Rm and store the result in Rn.	ADD R0,R0,#4
ANDccS Rn, Rm, Op2	Logically AND Op2 with Rm and store the result in Rn.	AND R0,R0,#4
Bcc Label	Branch to a relative Label.	BEQ ConditionalJump
BICccS Rn, Rm, Op2	Logically Bit Clear Op2 with Rm and store the result in Rn.	BIC R0,R0,#4
BLcc Label	Branch and Link to a relative subroutine Label.	BL TestSub
CMNcc Rn, Op2	Compare Negative Rn to Op2. set the flags like"ADDS Rn,Op2"	CMN R0,#4
CMPcc Rn, Op2	Compare Rn to Op2. set the flags, the same as "SUBS Rn,Op2"	CMP R0,#4
EORccS Rn, Rm, Op2	Logically Exclusive OR Op2 with Rm and store the result in Rn.	EOR R0,R0,#4
LDMccadm Rn!, {Regs}	Transfer range of registers {Regs} to address in Rn. Like POP	LDMFD sp!,{r0,r1,r2}
LDRcc Rn, Flex		
LDRccB Rn, Flex	Load register Rn from address Flex	LDR R0,NearLabel
LDRccH Rn, Off		
LDRccSH Rn, Off	Half-Mand (4C hit) Cinnad Mand (4C Bit) and Cinnad Buta (6 Bit) land	I DDCD D0 (D4 # 055)
LDRccSB Rn, Off	HalfWord (16 bit), Signed Word (16 Bit) and Signed Byte (8 Bit) load	LDRSB R0,[R1,#-255]
MLAccS Rn, Rm, Ro, Rp	32 bit Multiplication and Add. Rn=(Rm*Ro)+ Rp	MLA R0,R1,R2,R3
MOVccS Rn, Op2 MRScc Rn,sr	Move value in Op2 into Rn.	MOV R0,#0xFF
•	Move sr (either CPSR or SPSR) to register Rn.	MRS R0,SPSR
MSRcc sr_f,# MSRcc sr_f,Rn	Move immediate # or register into flags f of sr (either CPSR or SPSR).	MSR CPSR F,#0
MULccS Rn, Rm, Ro	32 bit Multiplication. Rn=Rm*Ro.	MUL R0,R1,R2
MVNccS Rn, Op2	Move Not. Flip all the bits of Op2 and move result into Rn.	MVN R0,#0xFF
ORRccS Rn, Rm, Op2	Logically OR Op2 with Rm and store the result in Rn.	ORR R0,R0,#4
RSBccS Rn, Rm, Op2	Reverse Subtract. This performs the calculation Rn=Op2-Rm.	RSB R0,R0,#6
RSCccS Rn, Rm, Op2	Reverse Subtract with Carry. Rn=(Op2-Rm)-C.	RSC R0,R0,#6
SBCccS Rn, Rm, Op2	Reverse Subtract with Carry. Rn=(Op2-Rm)-C.	SBC R0,R0,#6
STMccadm Rn!, {Regs}	Transfer range of registers {Regs} to the address in Rn. Like PUSH	STMFD sp!,{r0,r1,r2}
STRcc Rn, Flex		
STRccB Rn, Flex	Store register Rn to address Flex.	STR r0,[r1,r2,asl #2]
STRccH Rn, Off		
STRccSH Rn, Off		
STRccSB Rn, Off	Half Word (16 bit), Signed half Word (16 Bit) and Signed Byte (8 Bit) store	STRSB R0,[R1,#-255]
SUBccS Rn, Rm, Op2	Subtract. This performs the calculation Rn=Rm-Op2.	SUB R0,R0,#6
SWIcc #	Software Interrupt.	SWI 3
SWPccB Rn, Rm, [Ro]	Swap a register and memory. Rn=[Ro], [Ro]=Rm.	SWPB R0,R1,[R2]
TEQcc Rn, Rm, Op2	Test for bitwise Equality. Set the flags like "EOR Rn,Rm,Op2"	TEQ R0,R0,#6
TSTcc Rn, Rm, Op2	Test bits. Set the flags like "AND Rn,Rm,Op2"	TST R0,R0,#6

ARM A	ddressing Modes			
Param	Mode	Format	Details	Example
Op2	Immediate	#n	Fixed value of n Can be any value made by an 8 bit immediate shifted by an even number of bits, eg 0xFF or 0xFF000000 are OK.	ADD R0,R0,#1
Op2	Register	Rn	value in register Rn	ADD R0,R0,R1
Op2	Register Shifted by Immediate	Rn, shft #n	Shift Register Rn by #n using shifter shft Options: LSL #n, LSR #n, ASR #n, ROR #n, RRX note: RRX can only shift 1 bit	MOV R0,R1,ROR #2
Op2	Register Shifted by Register	Rn, shft Rm	Shift Register by Rm using shifter shft Options: LSL Rm, LSR Rm, ASR Rm, ROR Rm	MOV R0,R1,ROR R2
Flex	Immediate offset Immediate pre-indexed	[Rn,#n] [Rn,#n]!	value from address in register Rn+n ! means Preindexed, set Rn=Rn+n	LDR R0,[R1] ;#n=0 LDR R0,[R1,#4] LDR R0,[R1,#-4]!
Flex	Register offset Register pre-indexed	[Rn,{-}Rm] [Rn,{-}Rm]!	value from address in register Rn+Rm ! means Preindexed, set Rn=Rn+Rm	LDR R0,[R1,-R2] LDR R0,[R1,R2]!
Flex	Scaled register offset Scaled register pre-indexed	[Rn, Rm,shft #n] [Rn, Rm,shft #n]!	value from address in register, if LSL then Rn+(Rm*#n) ! means Preindexed, set Rn=Rn+n	LDR R0,[R1,R2, LSL #2] LDR R0,[R1,R2, LSL #2]!
Flex	Immediate post-indexed	[Rn],#n	value from address in register Rn set Rn=Rn+n (No need for ! - as it's the only purpose of the command!)	LDR R0,[R1],#4
Flex	Register post-indexed	[Rn], {-}Rm	value from address in register Rn set Rn=Rn+Rm (No need for ! - as it's the only purpose of the command!)	LDR R0,[R1],R2 LDR R0,[R1],-R2
Flex	Scaled register post-indexed	[Rn], {-}Rm, shft #n	Shift Register by #n using shifter shft Options: LSL #n, LSR #n, ASR #n, ROR #n, RRX	LDR R0,[R1],R2,LSL #2 LDR R0,[R1],-R2,RRX

ARM Co	nditions	
cc code	Meaning	Flag
EQ	EQual	Z=1
NE	Not Equal	Z=0
CS	Carry Set	
HS	Higher or Same (Unsigned)	C=1
CC LO	Carry Clear LOwer (Unsigned)	C=0
MI	MInus (Negative)	N=1
PL	PLus (Positive)	N=0
VS	oVerflow Set	V=1
VC	oVerflow Clear	V=0
HI	HIgher (Unsigned)	C=1 and Z=0
LS	Lower or Same (Unsigned)	C=0 and Z=1
GE	Greater or Equal (Signed)	N=V
LT	Less Than (Signed)	N<>V
GT	Greater Than (Signed)	Z=0 and N=V
LE	Less than or Equal (Signed)	Z=1 or N<>V
AL	ALways	No condition

ARM Thumb				4.	Mattabassa
Command DB Pd (Pp #1	Detail Exam		Cycles Opco	de NZCV	ValidRegs
_ DR Rd,[Rn,#] _ DRB Rd,[Rn,#]		3,[r5,#0] 3 r3,[r5,#2]			R0-R7,#= 0 to 124 (Multiples of 4) R0-R7,#= 0 to 31 (Multiples of 1)
_DSB Rd,[Rn,#]		iB r3,[r5,#2]			R0-R7,#= 0 to 31 (Multiples of 1)
. DRH Rd,[Rn,#]		l r3,[r5,#4]			R0-R7,#= 0 to 61 (Multiples of 1)
. DSH Rd,[Rn,#]		6H r3,[r5,#4]			R0-R7,#= 0 to 62 (Multiples of 2)
TR Rd,[Rn,#]		3,[r5,#0]			R0-R7,#= 0 to 124 (Multiples of 4)
TRB Rd,[Rn,#]	Store Register (8 Bit) STRE	r3,[r5,#0]			R0-R7,#= 0 to 31 (Multiples of 1)
TRH Rd,[Rn,#]	Store Register (16 Bit) STRH	l r3,[r5,#0]			R0-R7,#= 0 to 62 (Multiples of 2)
OP {reglist}	Pop registers from the stack POP	{r1-r3,r5}			R0-R7,LR
USH {reglist}		l {r1-r3,r5}			R0-R7,PC
.DMIA Rn!,{reglist}	Load Multiple and increment afte LDMI				R0-R7
TMIA Rn!,{reglist}	Store Multiple and increment afte STMI				R0-R7
NDD Rd,Rn,Rm	Add	Rd=Rn+R		NZCV	R0-R7
ADD Rd,Rn,# ADD Rd,#	Add Add	Rd=Rn+# Rd=Rd+#		NZCV NZCV	R0-R7 #=0 to 7 R0-R7 #=0 to 255
SUB Rd,Rn,Rm	Subtract	Rd=Rd+# Rd=Rn-R		NZCV	R0-R7
SUB Rd,Rn,#	Subtract	Rd=Rn-#		NZCV	R0-R7 #=0 to 7
SUB Rd,#	Subtract	Rd=Rd-#		NZCV	R0-R7 #=0 to 255
DD Rd,Rm	Add Low/High Regs (Can't both be low		lm	NZCV	R0-R15,SP
ADD SP,#	Add to Stack Pointer	SP=SP+#		NZCV	#0 to 508 (Multiple of 4)
SUB SP,#	Add to Stack Pointer	SP=SP+#		NZCV	#0 to 508 (Multiple of 4)
ADD Rd,PC/SP,#	Add immediate to SP/PC	Rd=PC/S		NZCV	R0-R7, Rp=PC/SP #=0 to 1020 (Multiples of 4)
ADC Rd,Rm	Add with carry	Rd=Rd+R		NZCV	R0-R7
BBC Rd,Rm	Subtract with carry	Rd=Rd-(F	,	NZCV	R0-R7
IUL Rd,Rm	Multiply	Rd=Rd*R		N Z	R0-R7
ND Rd,Rm	Logical AND	Rd=Rd Al		N Z	R0-R7
ORR Rd,Rm	Logical OR	Rd=Rd O		N Z	R0-R7
OR Rd,Rm	Logical Bit Clear	Rd=Rd E0		N Z	R0-R7
BIC Rd,Rm ASR Rd,Rs	Logical Bit Clear Arithmetic Shift Right Rs bits	Rd=Rd Al Rd=Rd As	ND (NOT Rm) SR Rs	N Z N Z C -	R0-R7 R0-R7
ISR Rd,RS ISR Rd,#	Arithmetic Shift Right # bits	Rd=Rd As		NZC-	R0-R7, #1 to 32
.SR Rd,Rs	Logical Shift Right Rs bits	Rd=Rd LS		NZC-	R0-R7
.SR Rd,#	Logical Shift Right # bits	Rd=Rd LS		NZC-	R0-R7, #1 to 32
.SL Rd,Rs	Logical Shift Left Rs bits	Rd=Rd LS		NZC-	R0-R7
.SL Rd,#	Logical Shift Left # bits	Rd=Rd LS		NZC-	R0-R7, #0 to 31
ROR Rd,Rs	Rotate Right Rs bits	Rd=Rd R	OR Rs	NZC-	R0-R7
CMP Rn,Rm	Compare (Set flags like SUB)	Flags=Rn	-Rm	NZCV	R0-R15
CMP Rn,#	Compare (Set flags like SUB)	Flags=Rn	-#	NZCV	R0-R7, #0 to 255
CMN Rn,Rm	Compare Negative (Set flags like ADD)		+Rm	NZCV	R0-R7
//OV Rd,#	Move Immediate	Rd=#		NZCV	R0-R7, #0 to 255
IOV Rd,Rm	Move	Rd=Rm	_	NZCV	R0-R15 (Flags unchanged R8+)
IVN Rd,Rm	Move Not (Flip bits of Rm)	Rd=NOT	Rm	NZCV	R0-R7
NEG Rd,Rm	Negate	Rd=-Rm	AND Do	NZCV	R0-R7
「ST Rn,Rm 3 label	Test Masked (AND)	Flags= Rr	AND Rm	N Z	R0-R7
BEQ label	Branch to label Branch if Equal	Z=1			Label= -2048 to +2048 -252 to +258
BNE label	Branch if Not Equal	Z=0			-252 to +258
BCS label	Branch Carry Set	C=1			-252 to +258
BHS label	Branch if Higher or Same (Unsigned)	C=1			-252 to +258
BCC label	Branch if Carry Clear	C=0			-252 to +258
BLO label	Branch if Lower or Same (Unsigned)	C=0			-252 to +258
BMI label	Branch if Minus	N=1			-252 to +258
BPL label	Branch if Plus	N=0			-252 to +258
BVS label	Branch if oVerflow Set	V=1			-252 to +258
BVC label	Branch if oVerflow Clear	V=0			-252 to +258
BHI label	Branch if Higher (Unsigned)	C=1 and 2			-252 to +258
BLS label	Branch if Lower or Same (Unsigned)	C=0 or Z=	:1		-252 to +258
BGE label	Branch if Greater or Equal (Signed)	N=V			-252 to +258
BLT label	Branch if Creater than (Signed)	N<>V 7=0 N=V			-252 to +258
BGT label	Branch if Greater than (Signed) Branch if Less than or Equal (Signed)	Z=0 N=V Z=1 N<>\	,		-252 to +258
BLE label BL label	Branch and Link		R14/LR=Return Address		-252 to +258 -4mb to +4mb
SK Rm	Branch and Link Branch and Exchange to Rm	PC=label PC=Rm	IN 1-7/LIN-INCIUIII AUUI 655	T=Bit0	R0-R15, -4mb to +4mb
6WI #	Software Interrupt	, O=1411		1-Dito	#=0 to 255
SKPT#	Breakpoint (enter debug mode)				#=0 to 255
ADR Rn,addr	Load address into Rn	ADD Rn,F	PC,#		#=0 to 1024
IOP	No operation	MOV R8,I			
	LoaD Register (32 bit) LDR	3,[r5,r0]			R0-R7,#= 0 to 124 (Multiples of 4)
.DR Rd,[Rn.Rm]		r3,[r5,r0]			R0-R7,#= 0 to 31 (Multiples of 1)
	LoaD Register (8 bit) LDRE	B r3,[r5,r0]			R0-R7,#= 0 to 31 (Multiples of 1)
.DRB Rd,[Rn,Rm]					R0-R7,#= 0 to 62 (Multiples of 2)
.DRB Rd,[Rn,Rm] .DSB Rd,[Rn,Rm]	LoaD Register (Signed 8 bit) LDRS	l r3,[r5,r0]			
DRB Rd,[Rn,Rm] DSB Rd,[Rn,Rm] DRH Rd,[Rn,Rm]	LoaD Register (Signed 8 bit) LDRS LoaD Register (16 bit) LDRS				R0-R7,#= 0 to 62 (Multiples of 2)
DRB Rd,[Rn,Rm] DSB Rd,[Rn,Rm] DRH Rd,[Rn,Rm] DSH Rd,[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) LDRS	l r3,[r5,r0]			R0-R7,#= 0 to 62 (Multiples of 2) R0-R7,#= 0 to 124 (Multiples of 4)
.DR Rd,[Rn,Rm] .DRB Rd,[Rn,Rm] .DSB Rd,[Rn,Rm] .DRH Rd,[Rn,Rm] .DSH Rd,[Rn,Rm] .DSH Rd,[Rn,Rm] .TR Rd,[Rn,Rm] .TR Rd,[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) Store Register (32 Bit) Store Register (8 Bit) STRE	l r3,[r5,r0] sH r3,[r5,r0]			R0-R7,#= 0 to 124 (Multiples of 4) R0-R7,#= 0 to 31 (Multiples of 1)
.DRB Rd,[Rn,Rm] .DSB Rd,[Rn,Rm] .DRH Rd,[Rn,Rm] .DSH Rd,[Rn,Rm] .DSH Rd,[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) Store Register (32 Bit) Store Register (8 Bit) STRE	l r3,[r5,r0] 5H r3,[r5,r0] 3,[r5,r0]			R0-R7,#= 0 to 124 (Multiples of 4)
.DRB Rd.[Rn,Rm] .DSB Rd.[Rn,Rm] .DRH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .TTR Rd.[Rn,Rm] .TTR Rd.[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) Store Register (32 Bit) Store Register (8 Bit) Store Register (16 Bit) STRI	l r3,[r5,r0] H r3,[r5,r0] 3,[r5,r0] : r3,[r5,r0] I r3,[r5,r0]			R0-R7,#= 0 to 124 (Multiples of 4) R0-R7,#= 0 to 31 (Multiples of 1) R0-R7,#= 0 to 62 (Multiples of 2)
.DRB Rd.[Rn,Rm] .DSB Rd.[Rn,Rm] .DRH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .TRR Bd.[Rn,Rm] .TRB Rd.[Rn,Rm] .TRH Rd.[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) LoaD Register (32 Bit) Store Register (8 Bit) Store Register (16 Bit) STRE Store Register PC relative (32 bit LDR in the control of th	I r3,[r5,r0] :H r3,[r5,r0] :3,[r5,r0] : r3,[r5,r0] I r3,[r5,r0] 3,[pc,#4]			R0-R7,#= 0 to 124 (Multiples of 4) R0-R7,#= 0 to 31 (Multiples of 1) R0-R7,#= 0 to 62 (Multiples of 2) R0-R7,#= 0 to 124 (Multiples of 4)
.DRB Rd.[Rn,Rm] .DSB Rd.[Rn,Rm] .DRH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .DSH Rd.[Rn,Rm] .TTR Rd.[Rn,Rm] .TTR Rd.[Rn,Rm]	LoaD Register (Signed 8 bit) LoaD Register (16 bit) LoaD Register (Signed 16 bit) Store Register (32 Bit) Store Register (8 Bit) Store Register (16 Bit) STRI	I r3,[r5,r0] IH r3,[r5,r0] 3,[r5,r0] I r3,[r5,r0] I r3,[r5,r0] 3,[pc,#4] 3,[sp,#4]			R0-R7,#= 0 to 124 (Multiples of 4) R0-R7,#= 0 to 31 (Multiples of 1) R0-R7,#= 0 to 62 (Multiples of 2)

Command	Detail	Example	OP	Cycles	Opcode	Arch
ADCccS R0, R1, OP2	Add with Carry	ADC R0,R1,R2	R0 = R1+R2+C	1	0101	
ADDccS R0, R1, OP2	Add	ADD R0,R1,R2	R0 = R1+R2	1	0100	
ANDccS R0, R1, OP2	Bitwise AND	AND R0,R1,R2	R0 = R1 and R2	1	0000	
Bcc addr	Branch (JP)	B label	R15=addr	3		
BICccS R0, R1, OP2	Bit Clear		R0 = R1 and (CPL R2)	1	1110	
BLcc addr	Branch and Link (CALL)	BL label	R14=R15 R15=addr	3		
CDPcc #,e,Crd,Crn,Crm,e2 CDC	Return From Exception					2,5
CMNccP R1, OP2	Compare Negative (Set flags like ADD)		flags=R1+R2	1	1001	
CMPccP R1, OP2	Compare (Set flags like SUB)		flags=R1-R2	1	1010	
EORccS R0, R1, OP2	Exclusive OR (XOR)		R0 = R1 xor R2	1	0001	
LDCccLTN #,Crd,addr,L LDC2	Load Coprocessor					2,5
LDMccmm R0!,{R1,R2R3}	Load Multiple (POP)		Move R1,R2R3→(R0)	1+		
LDRccBT R0,addr,shft	LoaD Register (B=8 bit / T=access in user mode)		R0=(addr)	3+	psuedo	
LDRccH R0,addr,shft	LoaD Register (16 bit)		R0=(addr)	3+		4T+
LDRccSB R0,addr,shft	LoaD Register (8 bit signed)		R0=(addr)	3+		4
LDRccSH R0,addr,shft	LoaD Register (16 bit signed)		R0=(addr)	3+		4
MCRcc #,e,Rd,Crn,Crm,e2 MCR	2Move from registers to coprocessor					2,5,5Ed
MLAccS R0,R1,R2,R3	Multiply with Accumulate		R0=(R1*R2)+R3	16		2
MOVccS R0, R2 ,shft	Move		R0 = R2	1	1101	
MRCcc #,e,Rd,Crn,Crm,e2 ,MRC	Coprocessor Register transfer					2,5
MRScc R0,flags	Move from CPSR/SPSR to register MSR R0,CPS	R MRS R4, CPSR	=PSR			3
MSRcc fields,#n/R0	Move from register to CPSR	MSR CPSR, R4	PSR=Rm			3
MULccS R0, R1, R2	Multiply		R0=R1*R2	16		2
MVNccS R0, R2 ,shft	Move Not (Flip bits of R2)		R0 = -R2	1	1111	
ORRccS R0, R1, R2 ,shft	Inclusive Or		R0 = R1 or R2	1	1100	
RSBccS R0, R1, R2 ,shft	Reverse SuBtract		R0 = R2-R1	1	0011	
RSCccS R0, R1, R2 ,shft	Reverse Subtract with Carry		R0 = R2-R1+C-1	1	0111	
SBCccS R0, R1, R2 ,shft	Subtract with carry		R0 = R1-R2+C-1	1	0110	
STCccLTN #,Crd,addr,L STC2	Store to Coprosessor					2,5ExP
STMccmm R0,{R1,R2R3}!	Store Multiple (PUSH)		Restore (R0)-> R1,R2	2+		
STRccBT R0,(addr),shft	Store Register (32 Bit)		(addr)=R0	2+		
STRccH R0,(addr)	Store Register (16 bit)		(addr)=R0	2+ (H=4+)		4T+
SUBccS R0, R1, R2 ,shft	Subtract		R0 = R1-R2	1	0010	
SWIcc #n	Software Interrupt (RST)			3		
SWPccB r0,r1,[base]	Load r0 from [base],store r1 in [base]		Rd=Rn Rn=Rd			3
TEQccP R1, R2 ,shft	Test Inverted (EOR) (P=set flags)	teqp R4,#0	flags=R1 xor R2	1	1001	
TSTccP R1, R2 ,shft	Test Masked (AND) (P=set flags)		flags=R1 AND R2	1	1000	
ADRcc Rn,addr	Load relative address into register		R0=addr		psuedo	
ADRccL Rn,label	Load Long relative address into register				psuedo	
NOP	no operation				psuedo	
P - Alter Processor Flags	S – Set condition codes	cc - Condition Code		B – Byte		
H - 16 Bit D - 64 bit	T- Translation (User Privilages in Super mode)					

Arm 5+

ARM Complete

Command	Detail	Example	OP	Cycles	Opcode	Arch
BXJcc R0	Branch and change to Jazelle state					6
BKPT imm	Breakpoint					5
BLX addr,BLXcc R0	Branch , link and exchange					5Tb
BXcc R0	Branch and exchange		R15=Rn Tbit=Rn[0]			5tb
CLZcc R0, R1	Count Leading Zeros					5
CPSeeff #n	Change Processor state					6
CPYcc R0, R1	Copy one register to another		R0=R1			6
LDRccD R0,addr	LoaD Register (64 bit)		R0=(addr),R1=(addr+4)	3+		5TE
LDREXcc R0,R1	LoaD Register and set memory exclusive		R0=(R1)	3+		6
MAR	Mover from registers to 40 bit acc					Xscale
MCRRcc #,e,Rd,Rn,Crn,Crm,e2	Move from 2 registers to coprocessor					5TE,6
MIA,MIAPH,MIAxy	Multiply with internal 40 bit accumulate					Xscale
MRA	Multiply from 40 bit accumulator to registers					Xscale
MRRCcc #,e,Rd,Rn,Crm, MRC2	Move from coprocessor to 2 regs					5E
PKHBTcc R0, R1, R2 ,shft	Pack Halfword Bottom/Top (L from R1 / H from R2)		R0=R2H+R1L			6
PKHTBcc R0, R1, R2 ,shft	Pack Halfword Top/Bottom (H from R1 / L from R2)		R0=R1H+R2L			6
PLD mode	Cache Preload					5E
QADDcc R0, R1, R2	Saturating Arithmatic					5Exp
QADD16cc R0, R1, R2	Saturating Arithmatic (16 bit)					6
QADD8cc R0, R1, R2	Saturating Arithmatic (8 bit)					6
QADDSUBXcc R0, R1, R2	Saturating Add and Subtract with Exchange					6
QDADDcc R0, R1, R2	Saturating Double and Add					5TE
QDSUBcc R0, R1, R2	Saturating Double and Subtract					5TE
QSUBcc R0, R1, R2	Saturating Subtract					5TE
QSUB16cc R0, R1, R2	Saturating Subtract (16 bit)					6
QSUB8cc R0, R1, R2	Saturating Subtract (8 bit)					6
QSUBADDXcc R0, R1, R2	Saturating Add and Subtract with Exchange					6
REVcc R0, R1	reverses the byte order in a 32-bit register.					6
REV16cc R0, R1	reverses the byte order in a 16-bit register.					6
REVSHcc R0, R1	reverses the byte order in a 16-bit register, and sign e	extend				6
RFE <mode> R0!</mode>	Return From Exception					6
SADD16cc R0, R1, R2	Signed Add two 16 bit numbers					6
SADD8cc R0, R1, R2	Signed Add four 8-bit signed integer additions					6
SADDSUBXcc R0, R1, R2	Signed 16-bit Add and Subtract with Exchange					6
SEL cc R0, R1, R2	Select bytes from R1/R2 based on GE flags					6

SETEND <endian></endian>	Set Endian mode				6
SHADD16cc R0, R1, R2	Signed Halving Add (16 bit)				6
SHADD8cc R0, R1, R2	Signed Halving Add (8 bit)				6
SHADDSUBXcc R0, R1, R2	Signed Halving Add and Subtract with Exchange (16	bit)			6
SHSUB16cc R0, R1, R2	Signed Halving Subtract (16 bit)				6
SHSUB8cc R0, R1, R2	Signed Halving Subtract (8 bit)				6
SHSUBADDXcc R0, R1, R2	Signed Halving Subtract and Add with Exchange (16	bit)			6
SMLALxycc R0L, R1H, R2,R3	Signed Multiply-accumulate Long				5TE
SMLAxycc	Signed Multiply-accumulate				5TE
SMLADXcc	Signed Multiply-accumulate Dual				6
SMLALccS R0L, R1H, R2,R3	Signed Multiply-accumulate Long				6
SMLAWycc	Signed Multiply-accumulate Word B and T				5ExP
SMLSDXcc R0, R1, R2,R3	Signed Multiply Subtract accumulate Dual				6
SMLSLDXcc R0, R1, R2,R3	Signed Multiply Subtract accumulate LongDual				6
SMMLARcc R0, R1, R2,R3	Signed Most significant word Multiply Accumulate				6
SMMLSRcc R0, R1, R2,R3	Signed Most significant word Multiply Subtract				6
SMULLRcc R0, R1, R2	Signed Multiply (R=Round)				6
SMUADXcc R0, R1, R2	Signed Dual Multiply Add				6
SMULXYcc R0, R1, R2	Signed Multiply BB, BT, TB, or TT				ARMv5TE
SMULLcc R0L, R1H, R2,R3	Signed Multiply Long				ARMv5TE
SMULWYCC R0, R1, R2	Signed Multiply Word B and T				ARMv5TE
SMUSDXcc R0, R1, R2	Signed Dual Multiply Subtract				6
SRS <mode> #mode!</mode>	Store Return State				6
SSAT16cc R0,#n, R1,shft	Signed Saturate (16 bit)				6
SSATcc R0,#n, R1,shft	Signed Saturate				6
SSUB16cc R0, R1, R2	Signed Subtract (16 bit)				6
SSUB8cc R0, R1, R2	Signed Subtract (8 bit)				6
SSUBADDXcc R0, R1, R2	Signed Subtract and Add with Exchange (16 bit)				6
STRccD R0,(addr)	Store Register (64 bit)		(addr)=R0,(addr+4)=R1	2+	ARMv5TE
STREXCC R0,R1,R2	Store Register Exclusive		(addi) Tto,(addi +) Tti		6
SXTABcc R0,R1,R2,shft	Extract an 8 bit value, and sign extend				6
SXTAB16cc R0,R1,R2,shft	Extract two 8 bit value, and sign extend to 16 bits				6
SXTAHcc R0,R1,R2,shft	Extract a 16 bit value, and sign extend				6
SXTBcc R0,R1,shft	Take a 8-bit value from a register and sign extends it	to 32 hits			6
SXTB16cc R0,R1,shft	Take two 8-bit value from a register and sign extends it				6
SXTHcc R0,R1,shft	Take two 16-bit value from a register and sign extends				6
UADD16cc R0,R1,R2	Unsigned Add (16 bit)	10 32 bits			6
UADD8cc R0,R1,R2	Unsigned Add (8 bit)				6
UADDSUBXcc R0,R1,R2	Unsigned Add and Subtract with Exchange				6
					6
UHADD900 P0 P1 P2	Unsigned Halving Add (16 bit) Unsigned Halving Add (8 bit)				6
UHADD8cc R0,R1,R2					6
UHSUB16cc R0,R1,R2	Unsigned Halving Subtract (16 bit)				
UHSUB8cc R0,R1,R2	Unsigned Halving Subtract (8 bit)				6
USUBADDXcc R0,R1,R2	Unsigned Subtract and Add with Exchange				6
UMAALCCS ROL, R1H, R2,R3	Unsigned Multiply Accumulate Long				6
UMULLccS R0L, R1H, R2,R3	Unsigned Multiply Long				6
UQADD16cc R0,R1,R2	Unsigned Saturating Add (16 bit)				6
UQADD8cc R0,R1,R2	Unsigned Saturating Add (8 bit)				6
UQADDSUBXcc R0,R1,R2	Unsigned Saturating Add and Subtract with Exchange	9			6
UQSUB16cc R0,R1,R2	Unsigned Saturating Subtract (16 bit)				6
UQSUB8cc R0,R1,R2	Unsigned Saturating Subtract (8 bit)				6
UQSUBADDXcc R0,R1,R2	Unsigned Saturating Subtract and Add with Exchange	9			6
USAD8cc R0,R1,R2	Unsigned Sum of Absolute Differences				6
USADA8cc R0,R1,R2,R3	Unsigned Sum of Absolute Differences and Accumula	ate			6
USATcc R0,#n, R1,shft	Unsigned Saturate				6
USAT16cc R0,#n, R1,shft	Unsigned Saturate (16 bit)				6
USUB16cc R0,R1,R2	Unsigned Subtract (16 bit)				6
USUB8cc R0,R1,R2	Unsigned Subtract (8 bit)				6
USUBADDXcc R0,R1,R2	Unsigned Subtract and Add with Exchange				6
UXTABcc R0,R1,R2,shft	Extract an 8 bit value and Zero extend				6
UXTAB16cc R0,R1,R2,shft	Extract two 8 bit values and Zero extend				6
UXTAHcc R0,R1,R2,shft	Extract an 16 bit value and Zero extend				6
UXTBcc R0,R1,shft	Extract an 8 bit value and Zero extend				6
UXTB16cc R0,R1,shft	Extract two 8 bit values and Zero extend				6
UXTHcc R0,R1,shft	Extract a 16 bit value and Zero Extend				6
		_			
P - Alter Processor Flags	S – Set condition codes	cc - Condition Code		B – Byte	
H - 16 Bit D - 64 bit	T- Translation (User Privilages in Super mode)		_		

Power PC	Integer Arithmetic Instructions	Details .=updateCR o=overflow c=carry e=extended m=minus one z=zero as param (R0)	R0=0?
addi rD,rA,SIMM	Add Immediate	The sum (rAID) + SIMM is placed into register rD.	Υ
addis rD,rA,SIMM add rD,rA,rB	Add immediate shifted Add	The sum (rA 0) + (SIMM x'0000') is placed into rD. The sum (rA) + (rB) is placed into register rD.	- Y
subf rD,rA,rB	Subtract from	add. rD,rA,rB addo rD,rA,rB addo. rD,rA,rB The sum -, (rA) + (rB) + 1 is placed into rD	
addic rD,rA,SIMM	Add Immediate Carrying	subf. rD,rA,rB subfo rD,rA,rB subfo rD,rA,rB The sum (rA) + SIMM is placed into register rD.	
		addic. rD,rA,SIMM	-
addc rD,rA,SIMM addc rD,rA,rB	Subtract from Immediate Carrying Add Carrying	The sum (rA) + SIMM + 1 is placed into register rD. The sum (rA) + (rB) is placed into register rD.	-
subfc rD,rA,rB	Subtract from Carrying	addc. rD,rA,rB addco rD,rA,rB addco. rD,rA,rB The sum -, (rA) + (rB) + 1 is placed into register rD.	_
adde rD,rA,rB	Add Extended	subfc. rD,rA,rB subfco rD,rA,rB subfco. rD,rA,rB The sum (rA) + (rB) + XER(CA) is placed into register rD.	_
subfe rD,rA,rB	Subtract from Extended	adde. rD,rA,rB addeo rD,rA,rB addeo. rD,rA,rB The sum -,(rA) + (rB) + XER(CA) is placed into register rD.	
		subfe. rD,rA,rB subfeo rD,rA,rB subfeo. rD,rA,rB	
addme rD,rA	Add to Minus One Extended	The sum (rA) + XER(CA) + x'FFFFFFFF' is placed into register rD. addme. rD,rA addmeo rD,rA addmeo. rD,rA	
subfme rD,rA	Subtract from Minus One Extended	The sum, (rA) + XER(CA) + x'FFFFFFFF' is placed into register rD. subfme. rD,rA subfmeo rD,rA subfmeo. rD,rA subfmeo.	-
addze rD,rA	Add to Zero Extended	The sum (rA) + XER(CA) is placed into register rD addze. rD,rA addzeo rD,rA addzeo. rD,rA	-
subfze rD,rA	Subtract from Zero Extended	The sum, (rA) + XER(CA) is placed into register rD. subfze. rD,rA subfzeo rD,rA subfzeo. rD,rA	-
neg rD,rA	Negate	NEGate register rA, and store the results in register rD. This converts a positive to a negative and vice versa. neg. rD,rA nego rD,rA nego. rD,rA	-
Opcode mulli rD,rA,SIMM	Integer Compare Instructions Multiply Low Immediate	Details .=updateCCR o=Overflow	
mullw rD,rA,rB	Multiply Low	The low-order 32 bits of the 48-bit product (rA)*SIMM are placed into rD. Use with mulhw for full 64 bit The low-order 32 bits of the 64-bit product (rA)*(rB) are placed into rD. Use with mulhwu for full 64 bit	-
mulhw rD,rA,rB	Multiply High Word	mullw.rD,rA,rB mullworD,rA,rB mullwo	-
mulhwu rD,rA,rB	Multiply High Word Unsigned	mulhw. rD,rA,rB The contents of rA and rB are interpreted as 32-bit unsigned integers. This gives the top 32 bit of the 64-bit product.	_
divw rD,rA,rB	Divide Word	mulhwu.rD,rA,rB rD= rA (signed) / rB (signed). To Get Signed Remainder: divw rD,rA,rB rD = quotient mull rD,rD,rB rD = quotient*divisor	
	SIdo TTOIG	subfrD,rD,rA rD = remainder diww.rD,rA,rB diwwo.rD,rA,rB diwwo.rD,rA,rB	
divwu rD,rA,rB	Divide Word Unsigned	rD= rA (unsigned) / rB (unsigned). To Get Unsigned Remainder: divwu rD,rA,r8 rD = quotient mull rD,rD,r8 rD =	-
		quotient*divisor subf rD,rD,rA rD = remainder divwu. rD,rA,rB divwuo rD,rA,rB divwuo. rD,rA,rB	
Opcode cmp crfD,L,rA,rB	Integer Compare Instructions Compare (Signed)	Details i=immediate w=word CoMPare signed rA with signed rB, storing results in CR field crfD. L=Length (32/64 bits)	-
cmpl crfD,L,rA,rB	Compare Logical (unsigned)	cmpi crfD,L,rA,SIMM cmpw crfD,rA,rB (cmp crfD,O,rA,rB) cmpwi crfD,rA,SIMM (cmpi crfD,O,rA,SIMM) CoMPare unsigned rA with unsigned rB, storing results in CR field crfD. L=Length (32/64 bits)	
•		cmpli crfD,L,rĂ,UIMM cmplwi crfD,rA,UIMM (cmpli crfD,O,rA,UIMM) cmplw crfD,rA,rB (cmpl crfD,O,rA,rB)	-
Opcode and rA,rS,rB	Integer Logical Instructions AND	Details .=updateCR i=immediate s=shifted c=complement / b=byte h=half The contents of rS is ANDed with the contents of register rB and the result is placed into rA.	-
or rA,rS,rB	OR	and. rA,rS,rB andi. rA,rS,UIMM andis. rA,rS,UIMM and rA,rS,rB and c. rA,rS,rB The contents of rS is ORed with the contents of rB and the result is placed into rA.	-
xor rA,rS,rB	XOR	or, rA,rS,rB ori rA,rS,UIMM oris rA,rS,UIMM orc rA,rS,rB orc. rA,rS,rB The contents of rS is XORed with the contents of rB and the result is placed into register rA.	_
nand rA,rS,rB	NAND	xor. rA,rS,rB xori rA,rS,UIMM xoris rA,rS,UIMM The contents of rS is ANDed with the contents of rB and the one's complement of the result is placed into register rA.	
nana 17,10,15	MANU	NAND with rA=rB can be used to obtain the one's complement. nand. rA,rS,rB	
nor rA,rS,rB	NOR	The contents of rS is ORed with the contents of rB and the one's complement of the result is placed into register rA.	-
eqv rA,rS,rB	Equivalent	nor. rA,rS,rB The contents of rS is XORed with the contents of rB and the complemented result is placed into register rA.	-
extsb rA,rS	Extend Sign Byte / Halfword	eqv. rA,rS,rB Register r S[24-31] are placed into rA[24-31]. Bit 24 of rS is placed into rA[O-23].	-
extsh rA,rS cntlzw rA,rS	Count Leading Zeros Word	extsb. rA,rS extsh. rA,rS Count of leading zero bits of rS is placed into rA. This number ranges from 0 to 32, inclusive.	_
Opcode	Integer Rotate Instructions	cnttzw. rA,rS Details .=updateCR	
rlwinm rA,rS,SH,MB,ME	Rotate Left Word Immediate then AND	Rotate rS left by SH bits, and keep (AND) bits MB-ME, storing results in rA	-
	with Mask	EG: rlwinm r3,r4,8,24,31 will shift R4<<8 and keep bits 24-31 (0=MSB 31=LSB) rlwinm. rA,rS,SH,MB,ME	
rlwnm rA,rS,rB,MB,ME	Rotate Left Word then AND with Mask	Rotate rS left by rB bits, and keep (AND) bits MB-ME, storing results in rA EG: rlwinm r3,r4,8,24,31 will shift R4<<8 and keep bits 24-31 (0=MSB 31=LSB)	-
rlwimi rA,rS,SH,MB,ME	Rotate Left Word Immediate then Mask	rlwnm. rA,rS,rB,MB,ME Rotate rS left by SH bits, and transfer bits MB-ME into rA (leaving other bits unchanged)	_
	Insert	EG: rlwinmi r3,r4,8,24,31 will shift R4<<8 and transfer bits 24-31 (0=MSB 31=LSB) to R3, bits 0-23 of R3 are unchanged rlwimi. rA,rS,SH,MB,ME	
Opcode slw rA,rS,rB	Integer Shift Instructions Shift Left Word	Details =updateCR rS is shifted left rB bits. Bits shifted out are lost, new bits on the right are 0.	
		slw. rA,rS,rB	
srw rA,rS,rB	Shift Right Word (Unigned)	rS is shifted right rB bits. New bits on the left are 0. srw. rA,rS,rB	
srawi rA,rS,SH	Shift Right Algebraic Word Immediate (Signed)	rS is right SH bits. Bits shifted out are lost. New bits on the left retain the old top bit, keeping the sign. XER[CA] is set if r S contains a negative number and any 1-bits are shifted out of position 3, otherwise XER(CA) is cleared.	-
sraw rA,rS,rB	Shift Right Algebraic Word (Signed)	srawi. rA,rS,SH rS is shifted right rB[26-31] bits. New bits on the left retain the old top bit, keeping the sign. XER[CA] is set to 1 if rS	-
	,	contains a negative number and any 1-bits are shifted out of position 31; otherwise XER[CA] is cleared to O. A If rB[26]= 1, then rA is filled with 32 sign bits (bit 0) from rS. Condition register field CRO is set based on the value written into rA.	
Opcode	Integer Load Instructions	sraw. rA,rS,rB Details b=byte h=half w=word / z=zero extend (unsigned) a=algebraic (Sign extended) u=update x=indexed	
lbz rD,d(rA)	Load Byte and Zero	Load and zero extend an 8 bit byte from the effective address.	Υ
Ihz rD,d(rA)	Load Half Word and Zero	Ibzx rD,rA,rB Ibzu rD,d(rA) Ibzux rD,rA,rB	Υ
Iha rD,d(rA)	Load Half Word Algebraic	Ihzx rD,rA,rB Ihzu rD,d(rA) Ihzux rD,rA,rB Load and sign extend an 16 bit half word from the effective address.	Υ
lwz rD,d(rA)	Load Word and Zero	Ihax rD,rA,rB Ihau rD,d(rA) Ihaux rD,rA,rB The effective address is the sum (rAl0)+d. The word in memory addressed by the EA is loaded into register rD[0-31].	Υ
Opcode	Integer Store Instructions	Iwzx rD,rA,rB Iwzu rD,d(rA) Iwzux rD,rA,rB Details b=byte h=half w=word / x=indexed u=update	
stb rS,d(rA)	Store Byte	The effective address is the sum (rA 0) + d. Register rS[24–31] is stored into the byte in memory addressed by the EA. stbx rS,rA,rB stbu rS,d(rA) stbux rS,rA,rB	Υ
,	Store Half word	The effective address is the sum (rAIO)+d. rS[16-31] is stored into the half-word in memory addressed by the EA.	Υ
sth rS,d(rA)	Store Hall word	ethy rC rA rB ethu rC d(rA) ethuy rC rA rB	
	Store Word	sthx rS,rA,rB sthu rS,d(rA) sthux rS,rA,rB The effective address is the sum (rAIO)+d. Register rS is stored into the word in memory addressed by the EA.	Υ
sth rS,d(rA)		The effective address is the sum (rAIO)+d. Register rS is stored into the word in memory addressed by the EA. stwx rS,rA,rB stwu rS,d(rA) stwux rS,rA,rB	Y
sth rS,d(rA) stw rS,d(rA)	Store Word	The effective address is the sum (rAIO)+d. Register rS is stored into the word in memory addressed by the EA. stwx rS,rA,rB stwu rS,d(rA) stwux rS,rA,rB	Y

and A. S. Washington. Security of the Control of t	lis rD,value	Load Shifted Immediate	Load a 16-bit signed immediate value, shifted left by 16 bits, into rA (equivalent to addis rA,0,value)	-
and A. G. Compared Register Control (A. C. March Co. C.	la rD,SIMM(rA) la rD,v			-
Selected to an information of the common of	mr rA,rS			
Supplemental and supplemental productions and supplemental supplemental surplemental surplementa	sub rD,rA,rB	Complement Register		-
Bern 1 Au G. 1. Control of March 1 Au Contr	subi rD,rA,SIMM	Integer Load and Store with Ryte		-
Indeed CAUSE Country of the Company in Process of the Country of t		Reversal Instructions		
settles (A.A.) Size is left five Depthoses network in the company of STAC (A.A.) Yet is a comp		,		
Silvage Load and Stock Marges Vising Funds of Stock Marges Vising Funds Vising Funds of Stock Marges Vising Funds Vising	sthbrx rS,rA,rB	Store Half Word Byte-Reverse Indexed	Store a Endian-Reversed 16 bit half from rD into (rA+rB). Byte reversed version of STHX rD,rA,rB	Υ
The effective of the service of the				Y
there is 0,019. Store Murple Vive (Prior) The prior Store	Imw rD d(rA)		The affective address is the sum (rAIO)+d, n= 32-rD, n consecutive words starting at EA are loaded into GPRs rD	Y
Transport of State (1997) The Control of Sta		<u> </u>	through 31. EA should be 32 bit aligned. Used as stack POP	
Internation Church State Clause Street your foundementable from Church State Church State Church State Church State Church State Sta	, , ,	Store Multiple Word (PUSH)	through 31. EA should be 32 bit aligned. Used as stack PUSH	Y
Interest (CAA) RB Load Strip Word Indexed Market (CAA) RB Some Strip Word Indexed Market (CAA) RB S				Y
States of SA/AB State Starty Word immediate State No SA/AB State Starty Word immediate Starty Months of Sarty Absolute Starty Word immediate Starty Months of Sarty Months of Sarty Absolute Starty Word immediate Starty Months of Sarty Months of	Iswx rD,rA,rB	·	Load XER:ByteCount (low 6 bits) bytes into rD+ from (rA+rB)EG: lswx r5,r3,r4#; Load XER:ByteCount (low 6 bits) bytes	
Stews (PA-AB) Stews 1974 More Noted Stews (PA-AB) Stews 1974 More	stswi rS,rA,NB	Store String Word Immediate		Υ
Spread Particular Spread Program Spr	stswx rS,rA,rB		Load XER:ByteCount (low 6 bits) bytes into rD+ from (rA+rB)EG: stswx r5,r3,r4#, Store XER:ByteCount (low 6 bits)	Υ
Sa mm, addr Bonach Aboutube Branch Except Control to the standards and prices specified by an address of the standard and prices of the standards and prices of the standa	Opcode	Branch Instructions		
Minimipator Both Many address Both Associate New Link Both Many and Many	b imm_addr			-
Search conditionally to the address controlled as the sum of the immediate patients with a line the condition register (25) to be used. Search conditionally be the address controlled as the sum of the immediate patients with a line the condition register. The line is the condition register to be used. Search conditionally the conditional from this. But BO RIL Integral patients with a line condition register to the line register. The interaction address coloring is interaction in Expendent of the Conditional to Line Register. But Bornach Conditional to Line Register. The conditional to Line Register. The line advants a product on the conditional to the line does not line to Line Register. The line conditional to Line Register. The line conditional to Line Register. The line advants a product on the conditional to the line advants a line line Register. The line conditional to Line Register. The line advants a line line Register. The line advants a line line line line line line line line	bl imm_addr			-
restruction. The 80 personal specifies the six in the condition register (CR) to be used to be 100 Bit larged, and the service of the control of the 100 Bit larged, and the service of the control of the 100 Bit larged, and the service of the control of the 100 Bit larged and the service of the control of the 100 Bit larged and the service of the 100 Bit larged and the service of the 100 Bit larged and the 100	bla imm_addr			-
both 80.0 it injury and in the conditional both left conditional both and the second conditional both in the register. Branch Conditional to Link Register. Branch Link Graphic Conditional to Link Register. Branch Link Link Link Section Register. Branch Link Link Link Link Link Link Link Link			instruction. The BI operand specifies the bit in the condition register (CR) to be used	
instruction. The instruction actives following his instruction is placed into the first register. Link Conditional Scale of the Test Register of Stand Conditional Scale of Test Register. Bernard Conditional Scale Link Register. Link Conditional Scale Link Register for the Scale Scale Scale Scale Link Conditional Scale Link Register. Bernard Conditional Scale Link Register for the Scale Scale Link Conditional Scale Link Register. Bernard Conditional Scale Link Register for the Scale Scale Link Conditional Scale Link Register. Bernard Conditional Scale Link Register for the Scale Register. Bernard Conditional Scale Link Register. Bernard Conditio	bca BO,BI,target_addr bcl BO,BI,target_addr		•	-
the law register. The control of th			instruction. The instruction address following this instruction is placed into the link register.	
The Bit operand specifies the bit in the condition register to be used bett as the condition of the branch. Bit of the Conditional to Court Register the stand conditional to the branch to the stand conditional to Court Register. Branch Conditional to Court Register the stand conditional to Court Register. Branch Condition Register Policy Court Device Device Device			the link register.	
Decided Pound Decided Conditional to Link Register them Branch conditionally to the distress specified in the finit register.	bcir BO,BI	Branch Conditional to Link Register		-
Bearth Conditional to Count Register Bearth Count Page Bearth Conditional to Count Register Bearth Conditional to Count Register Bearth Conditional to Count Register Bearth Count Page	bciri BO,BI		Branch conditionally to the address specified in the link register.	-
Branch Conditional to Count Register Part Inc. Part Register Opposition From Conditional to Count Register Discontinuous Branch From Conditional to Count Register Discontinuous Branch From Conditional to Count Register Discontinuous Branch From Conditional Count Register Discontinuous Branch From Conditional State S	bcctr BO,BI		Branch conditionally to the address specified in the count register. The BI operand specifies the bit in the condition	-
then Link. Part Pa	bcctrl BO BI	Branch Conditional to Count Register		
Opcode Service Servi		then Link.	placed into the link register.	
bit tanget Branch if less than bit label bits addr bittr bitter bittl abel bits addr bittr bitter bittl abel bits addr bittr bitter bittl bits bits addr bittr bitter bits b				-
beq labed Branch if equal bega addr begin	blt target			-
big taget Branch if greater than big table big taget of bigter bigter big to be big taget big the staget Branch if not less than big labed bins addr bins bins bins bins bins bins bins bins	beq target			-
bill taget Branch if not less than bill label Dink addr Dink Dink	bge target			
bes larget Branch if not greater han begin label brag addr braght begin label brag addr braght best larget Branch if not summary overflow best label base addr both best best address best best best best best best best b	bnl target			-
bas target Branch if summary overflow bus label bas add brash and brash address to brast summary overflow bus label bas address based brash address brash br	bne target			
but target Branch if not unordered but label but an addr bun if about bun target Branch if not unordered but label but an addr bun if a bun target but label but an addr bun if a bun target but but label but an addr bun if about - Decode Berah if not unordered but label but an addr bun if a bun target but but - Decode Berah if Condition False but cond.target Branch if Condition False but bet of the target but bet and the target but bet of target but bet	bso target	Branch if summary overflow	bso label bsoa addr bsolr bsotr bsol label bsola addr bsolri bsotri	
binut target Branch if not unordered Opcode Sec PaudeOp Opticals anabsolute prize to lark Register CER, pranch if CER, pranch	bus target			-
bit condularget Branch if Condition False bit abit better bit Data bit bit Da	bnu target	Branch if not unordered	bnu label bnua addr bnuir bnuctr bnui label bnuia addr bnuiri bnuctri	-
bif briff biffet biff biff biffet biff biff biffet biff biff biffet biff biffet biff biffet biffet biff biffet bif				-
bdnzz cond,target Branch after Decrement if nonzero and Decrement Count Register CTR, branch if CTR nonzero AND condition frue (cond=LT/GT/EQ/SO/UN) - bdnzz cond,target Branch after Decrement if nonzero and bdnzz bdnztli bdnztl bdnztl bdnztla bdnztli bdn	bf cond,target	Branch if Condition False		-
buttar cond.taget Branch after Decrement if nonzero and buttar bu	bdnz target	Branch after Decrement if nonzero		-
bedraft cond,target branch after Decrement if nonzero and condition cond FALSE bdzt arget Branch after Decrement if zero bdzt bdzt bdzd bdzd bdzd bdzd bdzd bdzd	bdnzt cond,target			-
betz larget Branch after Decrement if zero Decrement Courin Register CTR, branch if CTR zero batz abdzi by datz bdzi by datz by datz bdzi by datz b	bdnzf cond,target	Branch after Decrement if nonzero and	Decrement Count Register CTR, branch if CTR nonzero AND condition false (cond=LT/GT/EQ/SO/UN)	-
bdatz cond,target Branch after Decrement if zero and condition cond TRUE bdatz cond,target Branch after Decrement if zero and condition cond TRUE bdatz cond,target Branch after Decrement if zero and condition cond TRUE bdatz cond,target Branch after Decrement if zero and condition cond FALSE Opcode Condition Register Logical Instructions crand crbD,crbA,crbB Condition Register AND Single bit crbA is ANDed with crbB and stored in crbD. cror crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and stored in crbD. crand crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and stored in crbD. crand crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and stored in crbD. crand crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register NAND Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crand crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. crbA is CrbB and stored in crbD. crbB and stored in crbD. crbB and stored	bdz target			-
condition cond TRUE but at botat but but at botat but but at botat but but but but but but but but but bu			bdza bdzir bdzi bdzia bdziri	
condition cond FALSE bdzfir bdzfil bd		condition cond TRUE	bdzta bdztir bdzti bdztia bdztiri	
Opcode Condition Register Logical Instructions crand crbD,crbA,crbB Condition Register AND single bit crbA is ANDed with crbB and stored in crbD. crox crbD,crbA,crbB Condition Register OR single bit crbA is ORed with crbB and stored in crbD. crox crbD,crbA,crbB Condition Register NAND single bit crbA is NCRed with crbB and stored in crbD. crox crbD,crbA,crbB Condition Register NAND single bit crbA is NCRed with crbB and stored in crbD. crox crbD,crbA,crbB Condition Register NAND single bit crbA is NCRed with crbB and the complement (NOT) is stored in crbD. creq crbD,crbA,crbB Condition Register NAND single bit crbA is NOBed with crbB and the complement (NOT) is stored in crbD. creq crbD,crbA,crbB Condition Register NAND single bit crbA is CORed with crbB and the complement (NOT) is stored in crbD. creque crbD,crbA,crbB Condition Register NAND with single bit crbA is ANDed with trobB and the complement (NOT) is stored in crbD. creque crbD,crbA,crbB Condition Register NaND with single bit crbA is ANDed with the complement (NOT) is stored in crbD. creque crbD,crbA,crbB Condition Register NaND with single bit crbA is ANDed with the complement (NOT) is stored in crbD. creque crbD,crbA,crbB Condition Register NaND with single bit crbA is ANDed with the compliment (bit flipped) of crbB and stored in crbD. creque crbD,crbA,crbB Condition Register NaND with Complement single bit crbA is ORed with the compliment (bit flipped) of crbB and stored in crbD. creque crbD,crbA,crbB Condition Register Field The contents of crfS are copied into crfD. No other condition register fields are changed. - Opcode System Linkage Instructions ref crfD,crfS Move Condition Register Field The contents of crfS are copied into crfD. No other condition register fields are changed. - Opcode Trap Instructions and Mnemonics ref crfD,crfS Move from Interrupt - Creq System Call Trap Word Immediate T	bdzf cond,target			
crand crbD,crbA,crbB Condition Register AND single bit crbA is ANDed with crbB and stored in crbD. - crror crbD,crbA,crbB Condition Register OR single bit crbA is ORed with crbB and stored in crbD. - crror crbD,crbA,crbB Condition Register NOR single bit crbA is XORed with crbB and stored in crbD. - crand crbD,crbA,crbB Condition Register NOR single bit crbA is ANDed with crbB and the complement (NOT) is stored in crbD. - creand crbD,crbA,crbB Condition Register NOR single bit crbA is ORed with crbB and the complement (NOT) is stored in crbD. - creand crbD,crbA,crbB Condition Register Equivalent single bit crbA is CORed with crbB and the complement (NOT) is stored in crbD. - creand crbD,crbA,crbB Condition Register Equivalent single bit crbA is CORed with crbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register AND with Complement single bit crbA is ANDed with the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register NOR single bit crbA is ORed with trbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register NOR single bit crbA is ORed with trbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register NOR single bit crbA is ANDed with the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register NOR single bit crbA is ANDed with the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register Field Single bit crbA is ORed with trbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register Field Single bit crbA is ANDed with the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register Field Single bit crbA is ANDed with trbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register Field Single bit crbA is ANDed with trbB and the complement (NOT) is stored in crbD. - crande crbD,crbA,crbB Condition Register Field Single bit crbA is ANDed with trbB and the comp	Opcode	Condition Register Logical		
croor ctrD,crtA,crbB Condition Register XOR single bit crbA is XORed with crbB and stored in crbD	crand crbD,crbA,crbB	Condition Register AND		-
crnand crbD,crbA,crbB	cror crbD,crbA,crbB			-
creqv crbD,crbA,crbB	crnand crbD,crbA,crbB	Condition Register NAND	single bit crbA is ANDed with crbB and the complement (NOT) is stored in crbD.	
crandc crbD,crbA,crbB Condition Register AND with Complement single bit crbA is ANDed with the compliment (bit flipped) of crbB and stored in crbD. crorc crbD,crbA,crbB Condition Register OR with Complement single bit crbA is ORed with the compliment (bit flipped) of crbB and stored in crbD. crorc crbD,crbA,crbB Condition Register Field The contents of crfS are copied into crfD. No other condition register fields are changed.	crnor crbD,crbA,crbB creav crbD,crbA,crbB			-
crorc crbD,crbA,crbB Condition Register OR with Complement single bit crbA is ORed with the compliment (bit flipped) of crbB and stored in crbD. - composed System Linkage Instructions Details - composed System Call - composed Register Field The contents of crfS are copied into crfD. No other condition register fields are changed. - composed System Call - composed Trap Instructions and Mnemonics Details - composed Trap Instructions and Mnemonics Details - twi TO,rA,SIMM Trap Word Immediate Trap if condition TO comparing rA to SIMM (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT) - composed Trap Word Immediate Trap if condition TO comparing rA to SIMM tweet rA,SIMM tweet rA,FB	crandc crbD,crbA,crbB	Condition Register AND with		-
mcrf crfD,crfS Move Condition Register Field The contents of crfS are copied into crfD. No other condition register fields are changed. - Opcode System Linkage Instructions - System Call - Copy of Return from Interrupt - Opcode Trap Instructions and Mnemonics Trap if condition TO comparing rA to SIMM (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT) - Trap Word Immediate Trap if condition TO comparing rA to SIMM twgei rA,SIMM twgei rA,FB twgei rA,rB twgei rA	crorc crbD,crbA,crbB		single bit crbA is ORed with the compliment (bit flipped) of crbB and stored in crbD.	- 1
Opcode System Linkage Instructions Details sc System Call rfi Return from Interrupt Opcode Trap Instructions and Mnemonics Details twi TO,rA,SIMM Trap Word Immediate Trap if condition TO comparing rA to SIMM (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT) twiti rA,SIMM twlei rA,SIMM tweei rA,SIMM twgei rA,SIMM twgei rA,SIMM twnei rA,SIMM twnei rA,SIMM twnei rA,SIMM twnei rA,SIMM twnei rA,SIMM twllti rA,SIMM twllei rA,SIMM twllei rA,SIMM twllei rA,SIMM twllei rA,SIMM twllei rA,SIMM twnei rA,SIMM twnei rA,SIMM twllei rA,SIMM twllei rA,SIMM twllei rA,SIMM twnei rA,SIMM twllei rA,SIMM twl				-
Fig. Return from Interrupt Opcode Trap Instructions and Mnemonics Trap Word Immediate Trap Word Immediate Trap if condition TO comparing rA to SIMM (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT) twit ir A, SIMM twier rA, SIMM tweet rA, SIMM twilet rA, SIMM twile	Opcode	System Linkage Instructions		
Trap Instructions and Mnemonics Details	rfi	,		-
twiti rA, SIMM twelir rA, SIMM tweer rA, SIM tweer rA, SIMM tweer rA, SIMM tweer rA, SIMM tweer rA, SIM tweer rA, SIM tweer rA, SIM tweer rA, SIM tweer rA,	Opcode	Trap Instructions and Mnemonics		
twnei rA, SIMM twilti rA, SIMM twigi rA, SIMM twilei rA, SIMM	twi TO,rA,SIMM	rap Word Immediate	twiti rA,SIMM twiei rA,SIMM tweqi rA,SIMM twgei rA,SIMM twgti rA,SIMM twnii rA,SIMM	-
tw TO,rA,rB Trap Word Trap if condition TO comparing rA to RB (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT) twlt rA,rB tweq rA,rB tweq rA,rB twg rA,rB twnl rA,rB twne rA,rB twle rA,rB twne r			twnei rA,SIMM twilti rA,SIMM twngi rA,SIMM twilti rA,SIMM twilei rA,SIMM twigei rA,SIMM	
twllt rA,rB twlle rA,rB twlg rA,rB twlg rA,rB twlng rA	tw TO,rA,rB	Trap Word	Trap if condition TO comparing rA to RB (TO=01/2/3/4=LT/GT/EQ/LogicalLT/LogicalGT)	-
Register Instructions Move to Condition Register Fields CRM=bits 0-7 eg 0xFF=all CRM fields mcrxr crfD Move to Condition Register Move from XER bits 0-3 (SO OV CA), Clear these bits in XER			twilt rA,rB twile rA,rB twige rA,rB twigt rA,rB twini rA,rB twing rA,rB	
mcrxr crfD Move to Condition Register Move from XER bits 0-3 (SO OV CA), Clear these bits in XER - mfcr rD Move from Condition Register - mtmsr rS Move to Machine State Register -	Opcode	Register Instructions		
mfcr rD Move from Condition Register - mtmsr rS Move to Machine State Register -	mtcrf CRM,rS mcrxr crfD			-
	mfcr rD	Move from Condition Register		-
	mtmsr rS mfmsr rD	Move to Machine State Register Move from Machine State Register		-

Opcode	Cache Management Supervisor-Level Instruction	Details	
dcbi rA,rB	Data Cache Block Invalidate		-
Opcode	User-Level Cache Instructions	Details	
dcbt rA,rB	Data Cache Block Touch		-
dcbtst rA,rB	Data Cache Block Touch for Store		-
clcs rD,rA	Cache Line Compute Size		-
dcbz rA,rB	Data Cache Block Set to Zero		-
dcbst rA,rB	Data Cache Block Store		-
dcbf rA,rB	Data Cache Block Flush		-
Opcode	Move to/from Special Purpose Register Instructions	Details	
mtspr SPR,rS	Move to Special Purpose Register	Transfer rS to SPR (MQ,XER,RTCU,RTCL,DEC,LR,CTR). Simplified versions exist:	-
		mtxer rA mtspr 1,rA mtlr rA mtspr 8,rA mtctr rA mtspr 9,rA	
mfspr rD,SPR	Move from Special Purpose Register	Transfer Special SPR to rD. (MQ,XER,RTCU,RTCL,DEC,LR,CTR). Simplified versions exist: mfxer rA mfspr rA,1 mflr rA mfspr rA,8 mfctr rA mfspr rA,9	-
Opcode	Segment Register Manipulation Instructions	Details	
mtsr SR,rS	Move to Segment Register	The contents of rS is placed into segment register specified by operand SR. (Supervisor instruction)	-
mtsrin rS,rB	Move to Segment Register Indirect	The contents of rS are copied to the segment register selected by bits 0–3 of rB. (Supervisor instruction)	-
mfsr rD,SR	Move from Segment Register	The contents of the segment register specified by operand SR are placed into rD. (Supervisor instruction)	-
mfsrin rD,rB	Move from Segment Register Indirect	The contents of the segment register selected by bits 0–3 of rB are copied into rD. (Supervisor instruction)	-
Opcode	Translation Lookaside Buffer Management Instruction	Details	
tlbie rB	Translation Lookaside Buffer Invalidate Entry		-
Opcode	External Control Instructions	Details	
eciwx rD,rA,rB	External Control Input Word Indexed		Υ
ecowx rS,rA,rB	External Control Output Word Indexed		Υ
Opcode	Memory Synchronization Instructions	Details	
eieio	Enforce In Order Execution of I/O		-
isync	Instruction Synchronize		-
Iwarx rD,rA,rB	Load Word and Reserve Indexed	<u> </u>	-
stwcx. rS,rA,rB	Store Word Conditional Indexed	<u> </u>	-
sync	Synchronize		-

Floating point i	nstructions		
Opcode	Floating-Point Arithmetic Instructions	Details .=updateCR	
fadd frD.frA.frB	Floating-Point Add	fadd. frD.frA.frB	
fadds frD,frA,frB	Floating-Point Add Single-Precision	fadds. frD,frA,frB	
fsub frD.frA.frB	Floating-Point Add Single-Precision Floating-Point Subtract	fsub. frD,frA,frB	
fsubs frD,frA,frB	Floating-Point Subtract Single-Precision		
fmul frD,frA,frC	Floating-Point Multiply	fmul. frD,frA,frC	
fmuls frD,frA,frC	Floating-Point Multiply Single-Precision		
fdiv frD,frA,frB	Floating-Point Divide	fdiv. frD,frA,frB	
fdivs frD,frA,frB	Floating-Point Divide Single-Precision	fdivs. frD,frA,frB	
Opcode	Floating-Point Multiply-Add Instructions	Details .=updateCR	
fmadd frD,frA,frC,frB	Floating-Point Multiply-Add	fmadd. frD,frA,frC,frB	
fmadds frD,frA,frC,frB	Floating-Point Multiply-Add Single- Precision	fmadds. frD,frA,frC,frB	
fmsub frD,frA,frC,frB	Floating-Point Multiply-Subtract	fmsub. frD,frA,frC,frB	
fmsubs frD,frA,frC,frB	Floating-Point Multiply-Subtract Single- Precision	fmsubs. frD,frA,frC,frB	Ī
fnmadd frD,frA,frC,frB	Floating-Point Negative Multiply-Add	fnmadd. frD,frA,frC,frB	
fnmadds frD,frA,frC,frB	Floating-Point Negative Multiply-Add Single- Precision	fnmadds. frD,frA,frC,frB	
fnmsub frD,frA,frC,frB	Floating-Point Negative Multiply-Subtrac	fnmsub. frD,frA,frC,frB	
fnmsubs frD,frA,frC,frB	Floating-Point Negative Multiply-Subtract Single- Precision	fnmsubs. frD,frA,frC,frB	
Opcode	Floating-Point Rounding and Conversion Instructions	Details	
frsp frD,frB	Floating-Point Round to Single-Precision	frsp. frD,frB	
fctiw frD,frB	Floating-Point Convert to Integer Word	fctiw. frD,frB	
fctiwz frD,frB	Floating-Point Convert to Int Word with Round Toward Zero	fctiwz. frD,frB	
Opcode	Floating-Point Compare Instructions	Details	
fcmpu crfD,frA,frB	Floating-Point Compare Unordered		
fcmpo crfD,frA,frB	Floating-Point Compare Ordered		
Opcode	Floating-Point Compare Instructions	Details	
mffs frD	Move from FPSCR	mffs. frD	
mcrfs crfD,crfS	Move to Condition Register from FPSCR	mtfsfi. crfD,IMM	
mtfsfi crfD,IMM	Move to FPSCR Field Immediate		
mtfsf FM,frB	Move to FPSCR Fields	mtfsf. FM,frB	
mtfsb0 crbD	Move to FPSCR Bit 0	mtfsb0. crbD	
mtfsb1 crbD	Move to FPSCR Bit 1	mtfsb1. crbD	

A R1,D2(X2,B2) AR R1,R2 VA VR1, VR3,RS2(RT2) AP D1(L1,B1),D2(L2,B2) AH R1,D2(X2,B2)	Description Add Add	RX RR	5A	CI	Inst (360 - Float) AD R1,D2(X2,B2)	Description Add Normalized (L)	RX	Op CI 6A c	Psuedo Op B R1 / BR R1	Description
VA VR1, VR3,RS2(RT2) AP D1(L1,B1),D2(L2,B2) AH R1,D2(X2,B2)		KK			ADD D4 D2					Branch to Register
AP D1(L1,B1),D2(L2,B2) AH R1,D2(X2,B2)	Add	VST	1A A420	C IM	ADR R1,R2 AE R1,D2(X2,B2)	Add Normalized (L) Add Normalized (S)	RR RX	2A c 7A c	BH R1 / BHR R1 BL R1 / BLR R1	Branch on A High (after compare) Branch on A Low (after compare)
	Add Decimal	SS	FA	С	AER R1,R2	Add Normalized (S)	RR	3А с	BE R1 / BER R1	Branch on A Equal to B (after compare)
ALA NI,NZ	Add Logical	RX RR	4A	C	AW R1,D2(X2,B2)	Add Unnormalized (L)	RX RR	6E c	BNH R1 / BNHR R1	Branch on A Not High (after compare) Branch on A Not Low (after compare)
	Add Logical Add Normalized (E)	RR	1E 36	C C	AWR R1,R2 AU R1,D2(X2.B2)	Add Unnormalized (L) Add Unnormalized (S)	RX	2E c 7E c	BNL R1 / BNLR R1 BNE R1 / BNER R1	Branch on A Not Low (after compare) Branch on A Equal to B (after compare)
'	Add Logical	RX	5E	С	AUR R1,R2	Add Unnormalized (S)	RR	3E c	BP R1 / BPR R1	Branch on Plus (after arithmetic)
	AND	RX	54	С	CD R1,D2(X2,B2)	Compare (L)	RX	69 c	BM R1 / BMR R1	Branch on Minus (after arithmetic)
	AND AND	SS SI	D4 94	C	CDR R1,R2 CE R1,D2(X2,B2)	Compare (L) Compare (S)	RR RX	29 c 79 c	BZ R1 / BZR R1 BO R1 / BOR R1	Branch on Zero (after arithmetic) Branch on Overflow (after arithmetic)
\ //	AND	RR	14	С	CER R1,R2	Compare (S)	RR	39 c	BNP R1 / BNPR R1	Branch on Not Plus (after arithmetic)
	Branch and Link	RX	45		DD R1,D2(X2,B2)	Divide (L)	RX	5D	BNM R1 / BNMR R1	Branch on Not Minus (after arithmetic)
BALR R1,R2 BAS R1,D2(X2.B2)	Branch and Link Branch and Save	RR RX	05 4D		DDR R1,R2 DE R1,D2(X2,B2)	Divide (L) Divide (S)	RR RX	2D 7D	BNZ R1 / BNZR R1 BNO R1 / BNOR R1	Branch on Not Zero (after arithmetic) Branch on Not Overflow (after arithmetic)
	Branch and Save	RR	0D		DER R1,R2	Divide (S)	RR	3D	BO R1 / BOR R1	Branch if Ones (after mask test)
1 1 1	Branch on Condition	RS	47		HDR R1,R2	Halve (L)	RR	24	BM R1 / BMR R1	Branch if Mixed (after mask test)
	Branch on Condition Branch on Count	RR RX	07 46		HER R1,R2 LD R1,D2(X2,B2)	Halve (S) Load (L)	RR RX	34 68	BZ R1 / BZR R1 BNO R1 / BNOR R1	Branch if Zero (after mask test) Branch if Not Ones (after mask test)
	Branch on Count	RR	06		LDR R1,R2	Load (L)	RR	28	BNM R1 / BNMR R1	Branch if Not Mixed (after mask test)
	Branch on Index High	RS	86		LE R1,D2(X2,B2)	Load (S)	RX	78	BNZ R1 / BNZR R1	Branch if Not Zero (after mask test)
	Branch on Index Low Compare	RS RX	87 59	С	LER R1,R2 LTDR R1,R2	Load (S) Load and Test (L)	RR RR	38 22 c	NOP WTO 'Text'	No OPeration Write To Operator
CP D1(L1,B1),D2(L2,B2)	Compare Decimal	SS	F9	С	LTER R1,R2	Load and Test (S)	RR	32 c		
	Compare Halfword	RX	49	С	LCDR R1,R2	Load Complement (L)	RR	23 c	Assembler Instruction	
	Compare Logical Compare Logical	RX SS	55 D5	C C	LCER R1,R2 LNDR R1,R2	Load Complement (S) Load Negative (L)	RR RR	33 c 21 c	DC DS	Define constant Define storage
	Compare Logical	SI	95	С	LNER R1,R2	Load Negative (S)	RR	31 c	ccw	Define channel command word
	Convert to Binary	RX	4F		LPDR R1,R2	Load Positive (L)	RR	20 c	CCMO	Define format-0 channel command word
CVD R1,D2(X2,B2) D R1,D2(X2,B2)	Convert to Decimal Divide	RX RX	4E 5D		LPER R1,R2 LRDR R1,R2	Load Positive (S) Load Rounded (E/L)	RR RR	30 c 25	CCW1** START	Define format-1 channel command word Start assembly
DR R1,R2	Divide	RR	1D		LRER R1,R2	Load Rounded (L/S)	RR	35	LOCTR**	Specify multiple location counters
DP D1(L1,B1),D2(L2,B2)	Divide Decimal	SS	FD		MXR R1,R2	Multiply (E)	RR	26	CSE CT	Identify control section
ED D1,LI,B1),D2(B2) EDMK D1(L,B1),D2(B2)	Edit Edit and Mark	SS SS	DE DF	C C	MD R1,D2(X2,B2) MDR R1,R2	Multiply (L) Multiply (L)	RX RR	6C 2C	DSECT DXD*	Identify dummy section Define external dummy section
X R1,D2(X2,B2)	Exclusive OR	RX	57	С	MXD R1,D2(X2,B2)	Multiply (L/E)	RX	67	CXD*	Cumulative length of external dummy section
XC D1(L,B1),D2(B2)	Exclusive OR	SS	D7	С	MXDR R1,R2	Multiply (L/E)	RR	27	COM	Identify blank common control section
	Exclusive OR Exclusive OR	SI RR	97 17	C	ME R1,D2(X2,B2) MER R1,R2	Multiply (S/L) Multiply (S/L)	RX RR	7C 3C	AMODE** RMODE**	Specify addressing mode Specify residence mode
	Execute	RX	44		STD R1,D2(X2,B2)	Store (L)	RX	60	ENTRY	Identify entry-point symbol
	Halt I/O	S	9E00	рс	STE R1,D2(X2,B2)	Store (S)	RX	70	EXTRN	Identify external symbol
	Insert Character Insert Storage Key	RX RR	43 09	р	SXR R1,R2 SD R1,D2(X2,B2)	Subtract Normalized (E) Subtract Normalized (L)	RR RX	37 6B c	WXTRN USING	Identify weak external symbol Use base address register
L R1,D2(X2,B2)	Load	RX	58	P	SDR R1,R2	Subtract Normalized (L)	RR	2B c	DROP	Drop base address register
LR R1,R2	Load	RR	18		SE R1,D2(X2,B2)	Subtract Normalized (S)	RX	7B c	TITLE	Identify assembly output
LA R1,D2(X2,B2) LTR R1,R2	Load Address Load and Test	RX RR	41 12	С	SER R1,R2 SW R1,D2(X2,B2)	Subtract Normalized (S) Subtract Unnormalized (L)	RR RX	3B c 6F	EJECT SPACE	Sta rt new page Space listing
	Load Complement	RR	13	С	SWR R1,R2	Subtract Unnormalized (L)	RR	2F	PRINT	Print optional data
LH R1,D2(X2,B2)	Load Halfword	RX	48		SU R1,D2(X2,B2)	Subtract Unnormalized (S)		7F	ICTL	Input format control
LM R1,R2,D2(B2) LNR R1,R2	Load Multiple (regs R1-R2) Load Negative	RS RR	98 11	С	SUR R1,R2	Subtract Unnormalized (S)	RR	3F	ISEQ PUNCH	Input sequence checking Punch a card
	Load Positive	RR	10	c					RE PRO	Reproduce following card
LPSW D2(B2)	Load PSW	S	82	pn					ORG	Set location counter
LRA R1,D2(X2,B2) MVC D1(L,B1),D2(B2)	Load Real Address Move Characters	RX SS	B1 D2	рс	c. Condition code set				EQU OPSYN*	Equate symbol Equate operation code
MVI D1(B1),L2	Move Immediate	SI	92		i. Interruptible instruc	tion.			PUSH*	Save current PRINT or USING status
MVN D1(L,B1),D2(B2)	Move Numerics	SS	D1		n. New condition code				POP*	Restore PRINT or USING status
	Move with Offset Move Zones	SS SS	F1 D3		 p. Privileged instruction q. Semiprivileged instruction 				LTORG CNOP	Begin literal pool Conditional no operation
M R1,D2(X2,B2)	Multiply	RX	5C			m state and supervisor state	differs	S.	COPY	Copy predefined source coding
MR R1,R2	Multiply	RR	1C		y. Condition code ma				END	End assembly
MP D1(L1,B1),D2(L2,B2) MH R1,D2(X2,B2)	Multiply Decimal Multiply Halfword	SS RX	FC 4C		Electing point operan	d longths: Notes:			MACRO MEXIT	Macro definition header Macro definition exit
O R1,D2(X2,B2)	OR OR	RX	56	С	Floating-point operan (E) Extended source				MEND	Macro definition trailer
OC D1(L,B1),D2(B2)	OR	SS	D6	С	(E/L) Extended source	e, long result.			AREAD**	Assign card to SETC symbol
OI D1(B11.12 OR R1,R2	OR OR	SI RR	96 16	C C	(L/E) Long source, ex (L) Long source and r				ACTR AGO	Conditional assembly loop counter Unconditional branch
	Pack	SS	F2	C	(L/S) Long source, sh				AIF	Conditional branch
RDD D1(B1).I2	Read Direct	SI	85	p	(S/L) Short source, lo	ng result.			ANOP	Assembly no operation
	Set Storage Key	RR RR	04 08	n	(S) Short source and	result.			GBLA GBLB	Define global SETA symbol
	Set Storage Key Set System Mask	S	80 80	p p	Class (for instructions	subject to vector-control bit	, CR 0	bit 14)	GBLC	Define global SETB symbol Define global SETC symbol
SLDA R1,D2(B2)	Shift Left Double	RS	8F	С	IC: Interruptible; IVCT	- VIX) elements processed.			LCLA	Define local SET A symbol
SLDL R1,D2(B2)	Shift Left Double Logical	RS	8D	•		r (bit count in a general regis			LCLB	Define local SETS symbol
	Shift Left Single Shift Left Single Logical	RS RS	8B 89	С		/IX) elements processed, whΓ - VIX) elements processed.				Define local SETC symbol Generate error message
SRDA R1,D2(B2)	Shift Right Double	RS	8E	С	IP: Interruptible; (part	ial-sum-number - VIX) eleme	nts pro		MHELP**	Trace macro flow
SRDL R1,D2(B2)	Shift Right Double Logical	RS	8C			ion-size) elements processe	d.		SETA	Set arithmetic variable symbol
	Shift Right Single Shift Right Single Logical	RS RS	8A 88	С		(VCT) elements processed. (section-size) elements proc	essed		SETB SETC	Set binary variable symbol Set character variable symbol
SIO D2(B2)	Start I/O	S	9C00	рс	NO: Not interruptible;	no elements processed (VS				Service Symbol
ST R1,D2(X2,B2)	Store Character	RX	50		N1: Not interruptible;	one element processed.				
STC R1,D2(X2,B2) STH R1,D2(X2,B2)	Store Character Store Halfword	RX RX	42 40							
STM R1,R3,D2(B2)	Store Multiple (regs R1-R2)		90			ciation with first, second, or	third o	operand		
	Subtract Pagimal	RX	5B	С	B1, B2: Base register					
	Subtract Decimal Subtract Halfword	SS RX	FB 4B	C	D1, D2: Displacemer GR2: Register design	nt field (12 Bit) ation field (general register)				
SL R1,D2(X2,B2)	Subtract Logical	RX	5F	C	I2: Immediate operan					
SLR R1,R2	Subtract Logical	RR	1F	С	L: Length field (8 bit)					
	Supervisor Call Test and Set	RR S	0A 93	С	L1, L2: Length field (4 bit) nation field (equivalent to GR	13 if ~	eneral rea	ister or FR3 if floating	point register)
	Test Channel	S	93 9F00	c pc	R1, R2, R3: Register		o ii g	jonerai 1 e g	iotor, or i-no ir iloating-f	oonic register /
TIO D2(B2)	Test I/O	S	9D00	рс	RS2: Register design	ation field (starting address of	of vector	or)		
	Test under Mask Translate	SI SS	91 DC	С		ation field (stride of vector) ster designation field (vector	regiot	er)		
	Translate and Test	SS	DD	С	X2: Index register des		rogisti	,		
	Unpack Zero and Add	SS SS	F3		· · · ·					
			F8	С						

VACER VR1,VR2	Accumulate (S/L)	VV	A507	IM	VMXSD VR1,FR3.GR	Maximum Signed (L)	VR	A610IM
VAQ VR1,GR3,VR2	Add	QV	A5A0	IM	VMXSE VR1,FR3,GR2	Maximum Signed (S)	VR	A600IM
VAR VR1,VR3,VR2	Add	VV	A520	IM	VMNSD VR1,FR3,GR3 VMNSE VR1,FR3,GR3		VR	A611IM
VAS VR1,GR3,RS2(RT2) VADQ VR1,FR3,VR2	Add Add (L)	QST QV	A4A0 A490	IM IM	MC D1(B1),l2	Monitor Call	VR SI	A601IM AF
VADR VR1,VR3,VR2	Add (L)	VV	A510	IM	MVCIN D1(L,B1),D2(B		SS	E8
VADS VR1,FR3,RS2(RT2) VAD VR1,VR3,RS2(RT2)	Add (L) Add (L)	QST VST	A490 A410	IM IM	MVCL R1,R2 MVCP D1(R1,B1),D2(I	Move Long Move to Primary	RR SS	0E ic DA qc
VAE VR1,VR3,RS2(RT2)	Add (S)	VST	A400	IM	MVCS D1(R1,B1),D2(I		SS	DB qc
VAEQ VR1,FR3,VR2	Add (S)	QV	A580	IM	MVCK D1(R1,B1),D2(SS	D9 qc
VAER VR1, VR3,VR2 VAES VR1,FR3,RS2(RT2)	Add (S) Add (S)	VV QST	A500 A480	IM IM	VMQ VR1,GR3,VR2 VMR VR1,VR3,VR2		QV VV	A5A2IM A522IM
VN VR1,VR3,RS2(RT2)	AND	VST	A424	IM	VMS VR1,GR3,RS2(R			A5A2IM
VNQ VR1,GR3,VR2	AND	QV	A5A4	IM	VM VR1,VR3,RS2(RT			A522IM
VNR VR1,VR3,VR2 VNS VR1,GR3,RS2(RT2)	AND AND	VV QST	A524 A4A4	IM IM	VMD VR1,VR3,RS2(R VMDQ VR1,FR3,VR2		QV	A412IM A592IM
VNVM RS2	AND to VMR	VS.	A684	NC	VMDR VR1,VR3,VR2	Multiply (L)	٧٧	A512IM
CLRCH D2(B2)	Clear Channel	S	9F01	pc	VMDS VR1,FR3,RS2(A492IM
CLRIO D2(B2) VRCL D2(82)	Clear I/O Clear VR	S S	9D01 A6C5	pc IZ	VME VR1,VR3,RS2(R VMEQ VR1,FR3,VR2		VST QV	A402IM A582IM
CLM R1,M3,D2(B2)	Comp Logical Chars under	RS	BD	С	VMER VR1,VR3,VR2		VV	A502IM
VC M1,VR3,RS2(RT2)	Compare	VST	A428	IC	VMES VR1,FR3,RS2(I			A482IM
CR R1,R2 VCQ M1,GR3,VR2	Compare Compare	RR QV	19 A5A8	C IC		Multiply and Accumulate (L) Multiply and Accumulate (L)		A416IM A516IM
VCR M1.VR3,VR2	Compare	VV	A528	IC		Multiply and Accumulate (S		A406IM
VCS M1,GR3,RS2(RT2)	Compare (L)	QST	A4A8	IC		Multiply and Accumulate (Sa		A506IM
VCD M1,VR3,RS2(RT2) VCDQ M1,FR3,VR2	Compare (L) Compare (L)	VST QV	A418 A598	IC IC	VMAD VR1,VR3,RS2(VMADQ VR1,FR3,VR		VST QV	A414IM A594IM
VCDR M1,VR3,VR2	Compare (L)	VV	A518	IC	VMADS VR1,FR3,RS2	Multiply and Add (L)	OST	A494IM
VCDS M1,FR3,RS2(RT2)	Compare (L)		A498	IC		Multiply and Add (S/L)		A404IM
VCE M1,VR3,RS2(RT2) VCEQ M1,FR3,VR2	Compare (S) Compare (S)	VST QV	A408 A588	IC IC	VMAEQ VR1,FR3,VR2 VMAES VR1,FR3,RS2		QV OST	A584IM A484IM
VCER M1,VR3,VR2	Compare (S)	VV	A508	IC	VMSD VR1,VR3,RS2(Multiply and Subtract (L)		A415IM
VCES M1,FR3,RS2(RT2)	Compare and Swap	QST RS	A488 BA	IC		Multiply and Subtract (L)	QV OST	A595IM A495IM
CS R1,R3,D2,(B2) CDS R1,R3,D2(B2)	Compare and Swap Compare Double and Swap		BB BB	C C		Multiply and Subtract (L) Multiply and Subtract (S/L)		A495IM A405IM
CLR R1,R2	Compare Logical	RR	15	С	VMSEQ VR1,FR3,VR2	Multiply and Subtract (S/L)	QV	A585IM
CLCL R1,R2 VCVM	Compare Logical Long Complement VMR	RR RRE	0F A641	i c NC	VMSES VR1,FR3,RS2 VO VR1,VR3,RS2(RT	Multiply and Subtract (S/L)	QST VST	A485IM A425IM
CONCS D2(B2)	Connect Channel Set	S	B200	рс	BOQ VR1, GR3, VR2		QV	A5A5IM
VCZVM GR1	Count Left Zeros in VMR	RRE	A642	NC C	VOR VR1,VR3,VR2	OR	VV	A525IM
VCOVM GR1 Model-dependent	Count Ones in VMR Diagnose	RRE	A643 83	NC C PY	VOS VR1,GR3,RS2(R VOVM RS2	OR to VMR	QST VS	A5A5IM A685NC
DISCS D2(B2)	Disconnect Channel Set	s	83 B201	pc	PC D2(B2)	Program Call	vs S	B218q
VDD VR1, VR3,RS2(RT2)	Divide (L)	VST	A413	IM	PT R1,R2	Program Transfer	RRE	B228q
VDDQ VR1,FR3,VR2 VDDR VR1,VR3,VR2	Divide (L) Divide (L)	QV VV	A493 A513	IM IM	PTLB RRB D2(B2)	Purge TLB Reset Reference Bit	S S	B20Ep B213pc
VDDS VR1,FR3,RS2(RT2)	Divide (L)		A513 A493	IM	RABE R1,R2	Reset Reference Bit Extend		B213pc B22Apc
VDE VR1,VR3,RS2(RT2)	Divide (S)	VST	A403	IM	VACRS D2(B2)	Restore VAC	S	A6CENO F
VDEQ VR1,FR3,VR2 VDER VR1,VR3,VR2	Divide (S) Divide (S)	QV VV	A583 A503	IM IM	VMRRS D2(B2) VRRS GR1	Restore VMR Restore VR	S RRF	A6C3NZ A648IZ X0
VDES VR1,FR3,RS2(RT2)	Divide (S)	QST	A483	IM	VSRRS D2(B2)	Restore VSR	S	A6C2IZ X
VX VR1, VR3,RS2(RT2)	Exclusive OR		A426	IM	RIO D2(B2)	Resume I/O	S	9C02pc
VXQ VR1,GR3,VR2 VXR VR1,VR3,VR2	Exclusive OR Exclusive OR	QV VV	A5A6 A526	IM IM	VRSVC GR1 VACSV D2(B2)	Save Changed VR Save VAC	RRE S	A649IZ PO A6C/NO F
VXS VR1,GR3,RS2(RT2)	Exclusive OR	QST	A4A6	IM	VMRSV D2(B2)	Save VMR	S	A6C1NZ
VXVM RS2	Exclusive OR to VMR	VS	A686	NC	VRSV GR1	Save VR		A64AIZ C
VXEL VR1 ,GR3,GR2 VXELD VR1,FR3,GR2	Extract Element Extract Element (L)	VR VR	A629 A619	N1 N1	VS RSV D2(B2) SAC D2(B2)	Save VSR Set Address Space Control	S	A6C(NO X B219q
VXELE VR1,FR3,GR2	Extract Element (S)	VR	A609	N1	SCK D2(B2)	Set Clock	s	B204pc
EPAR R1	Extract Primary ASN	RRE	B226	q	SCKC D2(B2)	Set Clock Comparator	S	B206p
ESAR R1 VXVC GR1	Extract Secondary ASN Extract VCT	RRE RRE	B227 A644	q NO	SPT D2,(B2) SPX D2,(B2)	Set CPU Timer Set Prefix	S S	B208p B210p
VXVMM GR1	Extract Vector Mask Mode	RRE	A646	NO	SPKA D2(B2)	Set PSW Key from Address	S	B20Aq
HDV D2(B2) IAC R1	Halt Device	S	9E01 B224	pc	SSAR R1 SSKE R1,R2	Set Secondary ASN	RRE RRE	B225q B22Ep
ICM R1,M3,D2(B2)	Insert Address Space Conti Insert Characters under Ma		BS	db db	VSVMM D2(B2)	Set Storage Key Extended Set Vector Mask Mode	S	A6C(NO
IPK	Insert PSW Key	S	B20B	q		Shift and Round Decimal	SS	F0 c
ISKE R1,R2 IVSK R1,R2	Insert Storage Key Extende Insert Virtual Storage Key			p q		Shift Left Single Logical Shift Right Single Logical		E425IM E424IM
IPTE R1,R2	Invalidate Page Table Entry		B223 B221	p p	SIGP R1,R3,D2(B2)		RS	AE pc
VLR VR1,VR2	Load	VV	A509	ic	SIOF D2(B2)	Start I/O Fast Release	S	9C01pc
VLQ VR1,GR2	Load Load (L)	QV VST	A5A9	IC IC	VST VR1,RS2(RT2) VSTD VR1,RS2(RT2)	Store		A40EIC A41EIC
VID \/P1 DQ2/DT2\			Δ//10	10	VN1.R02(R12)			A+ ILIU
VLD VR1,RS2(RT2) VLDO VR1,FR2	Load (L)	QV	A419 A599	IC	VSTE VR1,RS2(RT2)			A40EIC
VLDO VR1,FR2 VLDR VR1,VR2	Load (L) Load (L)	QV VV	A599 A519	IC	VSTE VR1,RS2(RT2) STIDC D2,(B2)	Store (S) Store Channel ID	VST S	B203pc
VLDO VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2)	Load (L) Load (L) Load (S)	QV VV VST	A599 A519 A409	IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2)	Store (S) Store Channel ID Store Characters under Mas	VST S RS	B203pc BE
VLDO VR1,FR2 VLDR VR1,VR2	Load (L) Load (L)	QV VV	A599 A519	IC	VSTE VR1,RS2(RT2) STIDC D2,(B2)	Store (S) Store Channel ID	VST S	B203pc
VLDO VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1.FR2 VLER VR1,VR2 LASP D1(B1),D2(B2)	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran	QV VV VST QV VV nSSE	A599 A519 A409 A608 A589 E500	IC IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCKC D2,(B2) VSTK VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Compressed	VST S RS S S S VST	B203pc BE B205c B207p A40FIC
VLDO VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1.FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2)	Load (L) Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index	QV VV VST QV VV NSSE RSE	A599 A519 A409 A608 A589 E500 E428	IC IC IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCKC D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Compressed	VST S RS S S VST VST	B203pc BE B205c B207p A40FIC A41FIC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCDR VR1,VR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L)	QV VV VST QV VV NSSE RSE VV	A599 A519 A409 A608 A589 E500 E428 A562 A552	IC IC IC IC PC IG C IM	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCKC D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2 VSTKE VR1,RS2(RT2 STCTL R1,R3,D2(B2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control	VST S RS S S VST VST VST VST	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLBIX VR1,GR3,D2(B2) VLCDR VR1,VR2 VLCDR VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S)	QV VV VST QV VV NSSE RSE VV VV	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542	IC IC IC IC PC IG C IM IM	VSTE VR1,RS2(RT2) STIDC D2(,B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKE VR1,RS2(RT2 VSTKE VR1,RS2(RT2 STCTL R1,R3,D2(B2) STAP D2,(B2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Control Store CPU Address	VST S RS S VST VST VST TS S	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCDR VR1,VR2 VLCER VR1,VR2 LCTL R1,R3,D2(B2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Control	QV VV VST QV VV NSSE RSE VV VV VV	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7	IC IC IC IC IC IG C IM IM IM	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2 VSTKE VR1,RS2(RT2 STCTL R1,R3,D2(B2) STAP D2,(B2) STIDP D2,(B2)	Store (S) Store Channel ID Store Characters under Mat Store Clock Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store Control Store CPU Address Store CPU ID	VST S RS S S VST VST VST VST	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p B202p
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLBIX VR1,GR3,D2(B2) VLCDR VR1,VR2 VLCDR VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Control Load Element Load Element Load Element	QV VV VST QV VV SSE RSE VV VV VV RS VR VR	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7 A628 A618	IC IC IC IC IC IC IG IG IM	VSTE VR1,RS2(RT2) STIDE D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Conpressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU Timer Store Halfword	VST S RS S VST VST VST TS S S S VST	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p B202p B209p A42EIC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1.FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCDR VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2 VLEL VR1,GR3,GR2 VLELD VR1,FR3,GR2 VLELD VR1,FR3,GR2	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Control Load Element Load Element Load Element (L) Load Element (L) Load Element (L)	QV VV VST QV VV SSE RSE VV VV VV RS VR VR VR	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7 A628 A618 S608	IC IC IC IC IC IC IC IG IG IM IM IM IM IM P N1 N1	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STDP D2,(B2) STDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTI VR1,VR3,D2(B2)	Store (S) Store Channel ID Store Characters under Mat Store Clock Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU Timer Store Halfword Store Indirect	VST S RS S S VST VST VST TS S S S VST RSE	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p B202p B209p A42EIC E401IC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLCBLUT R1,R3,D2(B2) VLEL VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLY VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (S) Load Control Load Element Load Element (L) Load Element (S) Load Element (S) Load Espanded	QV VV VST QV VV SSE RSE VV VV VV RS VR VR VR VR VST	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7 A628 A618 S608 A40B	IC IC IC IC IC IC IC IG IG IM IN IN IN IN IN IN IN IC	VSTE VR1,RS2(RT2) STIDC D2(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STAP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU Timer Store Halfword Store Indirect Store Indirect Store Indirect Store Indirect Store Indirect	VST S RS S S VST VST TS S S VST TS S RSE RSE	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p B202p B202p B209p A42EIC E401IC R411IC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCBR VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLY VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYE VR1,RS2(RT2)	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Control Load Element Load Element (L) Load Espanded Load Expanded Load Expanded (L) Load Expanded (S)	QV VV VST QV VV INSSE RSE VV VV VV VR VR VR VR VST VST VST	A599 A519 A409 A608 A589 E500 E428 A562 A542 B7 A628 A618 S608 A40B A41B A40B	IC IC IC IC IC IG C IM IM IM IM P N1 N1 IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) STCTL R1,R3,D2(B2) STDT D2,(B2) STIDP D2,(B2) STPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTI VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTM VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Clock Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU ID Store CPU ID Store CPU Timer Store Halfword Store Indirect (L) Store Indirect (S) Store Matched	VST S RS S VST VST TS S S VST RSE RSE RSE VST	B203pc BE B205c B207p A40FiC A41FiC A40FiC B6 p B212p B202p B202p B42CiC E401lC R41 llC R40 llC A40EiC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLH VR1,RS2(RT2) VLH VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded (S) Load Halfword	QV VV VST QV VV INSSE RSE VV VV VV VR VR VR VST VST VST VST	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7 A628 A618 S608 A40B A41B A40B A429	IC I	VSTE VR1,RS2(RT2) STIDC D2.(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2.(B2) STCK D2.(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STAP D2.(B2) STAP D2.(B2) STPT D2.(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTWD VR1,RS3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU Timer Store Halfword Store Indirect Store Indirect (L) Store Indirect (S) Store Matched Store Matched Store Matched Store Matched	VST S RS S VST VST TS S S VST RSE RSE RSE VST VST	B203pc BE B205c B207p A40FIC A41FIC A40FIC B6 p B212p B202p B202p B209p A42CIC E40IIC R40IIC A40FIC A40FIC A40FIC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCDR VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLYDR,RS2(RT2) VLYDR,RS2(RT2) VLY VR1,RS2(RT2)	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Control Load Element Load Element (L) Load Espanded Load Expanded Load Expanded (L) Load Expanded (S)	QV VV VST QV VV ISSE RSE VV VV VV RS VR VR VR VST VST VST VST VST RSE	A599 A519 A409 A608 A589 E500 E428 A562 A542 B7 A628 A618 S608 A40B A40B A40B A429 E400	IC IC IC IC IC IG C IM IM IM IM P N1 N1 IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) STCTL R1,R3,D2(B2) STDT D2,(B2) STIDP D2,(B2) STPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTH VR1,RS2(RT2) VSTI VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTM VR1,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU Timer Store Halfword Store Indirect Store Indirect (L) Store Indirect (S) Store Matched Store Matched Store Matched Store Matched	VST S RS S VST VST TS S S VST RSE RSE RSE VST VST	B203pc BE B205c B207p A40FiC A41FiC A40FiC B6 p B212p B202p B202p B42CiC E401lC R41 llC R40 llC A40EiC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(82) VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLY VR1,RS2(RT2) VLI VR1,VR3,D2(B2) VLIE VR1,VR3,D2(B2) VLIE VR1,VR3,D2(B2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Halfword Load Indirect Load Indirect Load Indirect (L) Load Indirect (S)	QV VV VST QV VV VSST RSE VV VV VV VV RS VR VR VST VST VST VST RSE RSE RSE	A599 A519 A409 A608 A589 E500 E428 A562 A552 A542 B7 A628 A618 S608 A40B A41B A40B A429 E400 E400	IC IC IC IC IM IM IM IM IN IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STAP D2,(B2) STAP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STTMS D2,(B2) STPX D2,(B2) STPX D2,(B2) STNSM D1(B1),I2	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Control Store CPU Address Store CPU Timer Store Halfword Store Indirect Store Indirect (S) Store Indirect (S) Store Matched Store Matched (L) Store Matched (S) Store Matched (S) Store Prefix Store Then AND System Matched (S)	VST S RS S S VST TS S S VST RSE RSE RSE VST VST VST VST VST VST VST VST VST S S S S	B203pc BE B205c B207p A40FiC A40FiC A40FiC B6 p B212p B202p B202p B42EIC E401IC R401IC R401IC A40EiC
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLBIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCBR VR1,VR2 VLCER VR1,VR2 VLCER VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLY VR1,RS2(RT2) VL1 VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Control Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded Load Indirect Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (S)	QV VV VST QV VV VV NISSE RSE VST VST VST VST RSE RSE RSE VST	A599 A519 A409 A608 A589 E500 E428 A552 A542 B7 A628 A618 S608 A408 A40B A40B A429 E400 E410 A42A	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK PR1,RS,D2(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTM VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STSM D2,(B2) STNSM D1(B1),I2 STOSM D1(B1),I2	Store (S) Store Channel ID Store Chanacters under Mas Store Clock Store Clock Comparator Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU RAddress Store CPU ID Store CPU Timer Store Halfword Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (L) Store Matched (S) Store Matched (S) Store Prefix Store Indirect (S) Store Prefix Store Tren AND System Mas Store Then OR System Mas	VST S RS S S VST TS S S VST RSE RSE RSE VST VST VST VST VST VST VST VST S S S S	B203pc BE B205c B207p A40FIC A41FIC C A40FIC B6 p B212p B202p B202p B42EIC E401IC R401IC A40EIC A40EIC B211p AC p AD p
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(82) VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLY VR1,RS2(RT2) VLI VR1,VR3,D2(B2) VLIE VR1,VR3,D2(B2) VLIE VR1,VR3,D2(B2)	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Complement (S) Load Element Load Element Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (S) Load Halfword Load Indirect Load Indirect Load Indirect (S) Load Indirect (S) Load Integer Vector Load Integer Vector Load Matched	QV VV VV VV VV VV VV VV VV VST VST VST VS	A599 A519 A409 A608 A589 E500 E428 A562 B7 A628 A618 S608 A40B A440B A440B A440B A440A A440A A440A	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STAP D2,(B2) STAP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STTMS D2,(B2) STPX D2,(B2) STPX D2,(B2) STNSM D1(B1),I2	Store (S) Store Channel ID Store Chanacters under Mas Store Clock Store Clock Comparator Store Colock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU Timer Store Halfword Store Indirect Store Indirect (L) Store Indirect (S) Store Matched Store Matched (L) Store Matched (L) Store Prefix Store Prefix Store Then AND System Mas Store Then OR System Mas Store Vector Parameters	VST S RS S S VST TS S S VST RSE RSE RSE VST VST VST VST VST VST VST VST VST S S S S	B203pc BE B205c B207p A40FiC A40FiC A40FiC B6 p B212p B202p B203p A42EiC E401iC R411iC R411iC R40FiC A40EiC A40EiC A41EiC A40EiC B211p AC p A6CENO
VLD0 VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(82) VLGR VR1,VR2 VLCR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLELU VR1,FR3,GR2 VLELU VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VL1 VR1,VR3,D2(82) VLID VR1,VR3,D2(82) VLID VR1,VR3,D2(82) VLID VR1,RS2(RT2) VLIEVR1,VR3,D2(82) VLID VR1,RS2(RT2) VLIVR1,VR3,D2(82) VLID VR1,RS2(RT2) VLLU VR1,RS2(RT2) VLMQ VR1,GR2 VLMQ VR1,GR2 VLMQ VR1,GR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded Load Indirect Load Indirect Load Indirect (S) Load Matched Load Matched Load Matched	QV VV VST QV VV NSSE RSE VV VV VR VR VR VST VST VST VST VST VST VST VST VST VST	A599 A519 A409 A409 A608 A589 E500 A562 A542 B7 A628 A618 S608 A40B A40B A440B A440B A440B A440A A40A A	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STCTL R1,R3,D2(B2) STIDP D2,(B2) STIPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STSMD D1(B1),I2 VSTVP D2(B2) VSTVM D2(B2) VSTVM RS2 SR R1,R2	Store (S) Store Channel ID Store Channel ID Store Characters under Mas Store Clock Store Clock Comparator Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU ID Store CPU ID Store CPU Timer Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (L) Store Matched (S) Store Puffix Store Indirect (S) Store Matched (S) Store Prefix Store Matched (S) Store Tren AND System Mas Store Vector Parameters Store Vector Parameters Store VMR Subtract	VST S RS S S VST VST VST VST VST VST VST VST VS	B203pc BE B205c B207p A40FiC A40FiC B6 p B212p B202p B202p B202p B202p B402iC E401iC R440iiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC B211p AC p AC p A682NC 1B c
VLD0 VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLGR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCLE VR1,RS3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLY VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLIV VR1,RS2(RT2) VLIV VR1,RS2(RT2) VLIV VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLM VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Control Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Expanded (S) Load Expanded (L) Load Expanded (S) Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (L) Load Matched	QV VV VST QV VV VST QV VV VST QV VV VST	A599 A5199 A409 A608 A589 E500 E428 A562 A562 A542 B7 A628 A618 S608 A40B A40B A41B A40B A44B A40A A40A A50A A50A A41A	IC I	VSTE VR1, RS2(RT2) STIDC D2,(B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKD VR1, RS2(RT2) STCTL R1, R3, D2(B2) STAP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1, RS2(RT2) VSTID VR1, VR3, D2(B2) VSTID VR1, VR3, D2(B2) VSTM VR1, RS2(RT2) VSTMD D1(B1),12 STOSM D1(B1),12 VSTVP D2(B2) VSTVM RS2 SR R1,R2 VSQ VR1, GR3, VR2	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU Timer Store Halfword Store Indirect (L) Store Indirect (S) Store Matched (S) Store Matched (S) Store Matched (S) Store Then AND System Ms Store Then OR System Ms Store VMR Store VMR Store VMR Store VMR	VST S RSS S VST VST TS S S VST VST VST VST VST	B203pc BE B205c B207p A40FIC A40FIC A40FIC B6 p B212p B202p B203p A42EIC E401IC R411IC R411IC R40FIC A40EIC A40EIC A40EIC A40EIC B211p AC p A682NC B1 c B c A5A1IM
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLY VR1,RS2(RT2) VLLY VR1,RS2(RT2) VLLY VR1,RS2(RT2) VLLY VR1,RS2(RT2) VLLY VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMD VR1,RS2(RT2) VLMD VR1,RS2(RT2) VLMD VR1,RS2(RT2) VLMD VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (S) Load Complement (S) Load Element Load Element Load Element (S) Load Espanded Load Expanded Load Expanded (L) Load Expanded (L) Load Halfword Load Indirect Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (S) Load Indirect (S) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L)	QV VV VST QV VV NSSE RSE VV VV VR VR VR VST VST VST VST VST VST VST VST VST VST	A599 A519 A608 A608 A589 E500 A542 A542 A542 A618 A608 A408 A408 A408 A408 A400 E410 E400 A440 A40A A50A A40A A50A A40A A50A	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STAP D2,(B2) STPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD D1(B1),I2 STOSM D1(B1),I2 STOSM D1(B1),I2 STOSM D1(B1),I2 STOSM D1(B1),I2 VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSR VR1,VR3,VR2	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU Timer Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (L) Store Matched (S) Store Then AND System Mas Store Then OR System Mas Store Vector Parameters Store VMR Subtract Subtract Subtract	VST S RSS S VST VST TS S S S VST VST VST VST VS	B203pc BE B205c B207p A40FIC A40FIC A40FIC B6 p B212p B202p B202p B202p A42EIC E401IC R401IC R401IC R401IC A40EIC B211p AC p A0 p A62NC A682NC LB c A5A1IM A521IM
VLD0 VR1,FR2 VLD0 VR1,VR2 VLE VR1,RS2(RT2) VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLGR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLELE VR1,RS3,GR2 VLELD VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLIV VR1,RS2(RT2) VLIV VR1,RS2(RT2) VLIV VR1,RS2(RT2) VLIV VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLMQ VR1,RS2(RT2) VLMQ VR1,GR2 VLMD VR1,RS2(RT2) VLMD VR1,RS2(RT2) VLMD VR1,RS2(RT2) VLMD VR1,FR2 VLMD VR1,FR2 VLMD VR1,FR2 VLMD VR1,FR2 VLMD VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Indirect Load Indirect Load Indirect (S) Load Integer Vector Load Matched Load Matched Load Matched Load Matched Load Matched (L)	QV VV VST QV VV VV RS VV VV VST RSE RSE VST QV VST QV VST	A599 A519 A608 A608 A589 E500 E428 A562 A552 A618 A618 A40B A41B A40A A40A A40A A50A A41A A50A A41A A40A	IC IC IC IC IC IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STIDP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTM VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD D1(B1),12 STOSM D1(B1),12 STOSM D1(B1),12 STOSM D1(B1),12 VSTVP D2(B2) VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSR VR1,VR3,RS2(RT2 VSR VR1,RS,RS2(RT2 VSR VR1,RS1,RS2(RT2 VSR VR1,RS1,RS2	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU Timer Store Halfword Store Indirect Store Indirect Store Indirect (S) Store Matched Store Matched (L) Store Matched (S) Store Then AND System Mas Store Then OR System Mas Store VMR Store VMR Subtract Subtract Subtract Subtract Subtract Subtract Subtract Subtract Subtract	VST S RSS S VST VST TS S S VST VST TS S S VST VST	B203pc BE B205c B207p A40FiC A40FiC A40FiC B6 p B212p B202p B203p A42EiC E401lC R411lC R401lC A40EiC A40EiC A40EiC A40EiC A40EiC B211p AC p A682NC 1B c A55A1lM A521lM A421IM A441lM
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLSIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 LCTL R1,R3,D2(B2) VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMQ VR1,GR2 VLMD VR1,RS2(RT2) VLMDQ VR1,FR2 VLME VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Halfword Load Indirect Load Indirect Load Indirect (S) Load Indirect (S) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (L) Load Matched (S) Load Matched (S) Load Matched (S)	QV VV VST QV VV RS VR VR VST VST VST QV VV VST QV VV VST QV VST QV VST QV	A599 A6199 A608 A608 A589 E500 E428 A562 A562 B7 A628 S608 A40B A429 E400 E400 A42A A50A A45A A45A A45A A45A A45A A45A A45	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STCTL R1,R3,D2(B2) STPT D2,(B2) STPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMM P1(B1),I2 STOSM D1(B1),I2 STOSM D1(B1),I2 VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VS VR1,VR3,RS2(RT2) VSS VR1,VR3,RS2(RT2) VSS VR1,VR3,RS2(RT2) VSS VR1,VR3,RS2(RT2) VSS VR1,VR3,RS2(RT2) VSS VR1,VR3,RS2(RT2) VSD VR1,VR3,RS2(RT2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU ITimer Store Halfword Store Indirect (S) Store Indirect (S) Store Indirect (S) Store Matched (L) Store Indirect (S) Store Matched (S) Store Matched (S) Store Then AND System Mas Store Then OR System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract	VST S RSS S VST TS S S VST RSE RSE VST	B203pc BE B205c B207p A40FlC B6 p B212p B202p B202p B202p B202p B202p B402lC E401lC R401lC R401lC R401lC A40ElC A40ElC A40ElC B211p A6CRNO A682NC B c A55A1lM A421IM A441IM
VLDO VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(82) VLISIX VR1,GR3,D2(82) VLCR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VL1 VR1,RS2(RT2) VLL VR1,RS2(RT2) VLL VR1,RS2(RT2) VLL VR1,RS2(RT2) VLMQ VR1,RS2(RT2) VLMQ VR1,RS2(RT2) VLMQ VR1,RS2 VLMDQ VR1,RS2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element (L) Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded Load Expanded (S) Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (S) Load Indirect (S) Load Indirect (S) Load Indirect (L) Load Indirect (S) Load Indirect (L) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S)	QV VV VST QV VV VV VV VV VV VST VST VST VST VST VST	A599 A6199 A608 A608 A589 E500 E428 A562 B7 A628 S608 A40B A418 E400 E400 A42A A50A A50A A50A A50A A51A A50A A551A A40B A450A A55A A55A A55A A55A A55A A55A A55	IC IC IC IC IC IC IC IC	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKP D2,(B2) STIDP D2,(B2) STIDP D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD D2,(B2) STSMS D1(B1),I2 VSTVM D2,(B2) VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,RS2(RT2) VSD VR1,FR3,RS2(RT2) VSD VR1,FR3,VR2	Store (S) Store Channel ID Store Chanacters under Mas Store Clock Store Clock Comparator Store Clock Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Compressed (S) Store Compressed (S) Store CPU ID Store CPU ID Store CPU Inder Store Halfword Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (S) Store Matched (S) Store Prefix Store Matched (S) Store Prefix Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract Subtract Subtract Subtract (L) Subtract (L)	VST S RRS S VST TS S S VST VST VST VST VST VST V	B203pc BE B205c B207p A40FIC A40FIC A40FIC B6 p B212p B202p B203p B201p
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLSIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 LCTL R1,R3,D2(B2) VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELV R1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLH VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLIM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMD VR1,RS2(RT2) VLME VR1,VR2 VLME VR1,VR2 VLME VR1,VR2 VLME VR1,VR2 VLMR VR1,VR2 VLNR VR1,VR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (S) Load Halfword Load Indirect Load Indirect Load Indirect (S) Load Indirect (S) Load Indirect (S) Load Indirect (L) Load Indirect (L) Load Indirect (L) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Megative Load Negative	QV VV VST QV VV RS VR VST VST VST QV VV VST QV VV VV VST QV VV	A599 A608 A569 A608 A589 E428 A562 A562 A542 B7 A608 A41B A40B A40B A40A A50A A50A A51A A55A A55A A55A A55A A5	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2, (B2) STCK D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STDP D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTH VR1, VR3, D2(B2) VSTH VR1, VR3, D2(B2) VSTH VR1, RS2(RT2) VSTMD D2(B2) VSTVM RS2 STR3, D2(B2) VSTVM RS2 VSTVM RS2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, RS2(RT2) VSV VR1, VR3, RS2(RT2) VSV VR1, VR3, RS2(RT2) VSD VR1, RR3, RS2(RT2) VSD VR1, RR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, FR3, RS2(RT2) VSDR VR1, FR3, RS2(RT2) VSDR VR1, FR3, RS2(RT3)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Control Store CPU Address Store CPU ITIME Store Lalfword Store Indirect Store Indirect (S) Store Indirect (S) Store Matched (L) Store Indirect (S) Store Matched (S) Store Matched (S) Store Then AND System Mas Store Then AND System Mas Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract Subtract (L) Subtract (L) Subtract (L)	VST S B RS S S VST TS S S VST RSE RSE RSE RSE VST	B203pc BE B205c B207p A40FIC A40FIC B6 p B212p B202p B202p B202p B202p B42EIC E401IC R40IIC R40IIC R40IIC A40EIC A40EIC A40EIC B211p A6CENO A682NC B c A5A1IM A421IM A441IM A491IM A491IM
VLD0 VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2) VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,FR3,GR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYU VR1,RS2(RT2) VLYQ VR1,RS2(RT2) VLYQ VR1,RS2(RT2) VLYQ VR1,RS2(RT2) VLYQ VR1,RS2(RT2) VLIW VR1,RS2(RT2) VLIW VR1,RS2(RT2) VLIW VR1,RS2(RT2) VLMVR1,RS2(RT2) VLMVR1,RS2 VLNVR1,VR2 VLNVR1,VR2 VLNVR1,VR2 VLNVR1,VR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded Load Indirect Load Indirect Load Indirect (S) Load Indirect (S) Load Indirect (S) Load Indirect (L) Load Indirect (S) Load Indirect (L) Load Indirect (S) Load Indirect (S) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Negative Load Negative Load Negative (L)	QV VV VST VST VST VST VST VST VST VST VST	A599 A519 A608 A589 E500 E428 A562 A542 B7 A608 A408 A408 A408 A418 A408 A400 A42A A50A A56A A55A A55A A55A A55A A55A A55	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK D VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) STCTL R1,R3,D2(B2) STIDP D2,(B2) STIPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD D2(B2) VSTMD M1,RS2(RT2) VSTMP D2(B2) VSTVM RS2 STRSM D1(B1),I2 VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,RS2(RT2) VSQ VR1,GR3,RS2(RT2) VSQ VR1,GR3,RS2(RT2) VSDQ VR1,FR3,VR2 VSDQ VR1,FR3,VR2 VSDQ VR1,FR3,VR2 VSDQ VR1,FR3,VR2 VSDQ VR1,FR3,RS2(RT2)	Store (S) Store Channel ID Store Chanacters under Mas Store Clock Store Clock Comparator Store Clock Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Compressed (S) Store Compressed (S) Store CPU ID Store CPU ID Store CPU Inter Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (S) Store Matched (S) Store Prefix Store Indirect (S) Store Matched (S) Store Matched (S) Store Matched (S) Store Matched (S) Store Prefix Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract Subtract (L) Subtract (S)	VST S B RS S VST TS S S VST TS S S VST RSE	B203pc BE B205c B207p A40FIC A40FIC A40FIC B6 p B212p B202p B203p B201p B201p B201p B201p B201p A40EIC A40EIC A40EIC A41EIC B211p AC B211p AC B211p AC B211p AC B211p AGENO A682NC A5A1IM A421IM A441IM A451IM A451IM A451IM A491IM A491IM A491IM
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLSIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 LCTL R1,R3,D2(B2) VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLELV R1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLH VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLIM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMD VR1,RS2(RT2) VLME VR1,VR2 VLME VR1,VR2 VLME VR1,VR2 VLME VR1,VR2 VLMR VR1,VR2 VLNR VR1,VR2	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Indirect Load Indirect Load Indirect Load Indirect Load Indirect (S) Load Indirect (L) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Negative Load Negative Load Negative (L) Load Negative (S) Load Positive	QV VV VST QV VV RS VR VST VST VST QV VV VST QV VV VV VST QV VV	A599 A619 A608 A589 E500 E428 A562 A542 A618 A618 A40B A440B A440B A440B A440A A5AA A50A A50A A50A A50A A50A A50	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2, (B2) STCK D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STDP D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTH VR1, VR3, D2(B2) VSTH VR1, VR3, D2(B2) VSTH VR1, RS2(RT2) VSTMD D2(B2) VSTVM RS2 STR3, D2(B2) VSTVM RS2 VSTVM RS2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, RS2(RT2) VSV VR1, VR3, RS2(RT2) VSV VR1, VR3, RS2(RT2) VSD VR1, RR3, RS2(RT2) VSD VR1, RR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSDR VR1, FR3, RS2(RT2) VSDR VR1, FR3, RS2(RT2) VSDR VR1, FR3, RS2(RT3)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store CPU Timer Store Halfword Store CPU Timer Store Indirect Store Indirect Store Indirect Store Matched Store Matched Store Matched Store Matched Store Matched Store Then AND System Mas Store Then OR System Mas Store VMR Subtract Sub	VST S B RS S S VST TS S S VST RSE RSE RSE RSE VST	B203pc BE B205c B207p A40FiC A40FiC A40FiC B6 p B212p B202p B203p A42CiC E401lC R411lC R401lC A40EiC A40EiC A40EiC A40EiC A40EiC A41EiC A40EiC B211p AC p A62RNC B1 c B c A541lM A521lM A441lM A511lM A591lM A491lM A591lM
VLD0 VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(B1),D2(B2) VLBIX VR1,GR3,D2(B2) VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,FR3,GR2 VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLYEU VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYD VR1,RS2(RT2) VLYV VR1,RS2(RT2) VLYVR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,KR2(RT2) VLMV R1,RS2(RT2) VLMVR1,RS2(RT2) VLMVR1,RS2 VLMVR1,RS2 VLNVR1,RS2	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Complement (S) Load Element Load Element Load Element (S) Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (L) Load Indirect (S) Load Indirect (S) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Megative (L) Load Positive (L)	QV VV VST QV VST QV VST RSE RSE VV VV VST VST VST VST VST VST VST VST V	A599 A519 A608 A589 E500 E428 A562 A542 B7 A608 A418 A408 A408 A418 A408 A400 A42A A50A A51A A40A A50A A51A A50A A551 A551 A550 A554	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTK VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTKE VR1,RS2(RT2) VSTE VR1,RS2(RT2) VSTPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTH VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,VR3,D2(B2) VSTID VR1,RS2(RT2) VSTID VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2(RT2) VSTMP D2(B2) VSTVM RS2 SR R1,R2 VSQ VR1,GR3,VR2 VSSV VR1,GR3,RS2(RT2) VSV VR1,GR3,RS2(RT2) VSD VR1,VR3,RS2(RT2) VSD VR1,VR3,RS2(RT2) VSD VR1,VR3,RS2(RT2) VSD VR1,FR3,RS2(RT2) VSD VR1,FR3,RS2(RT2) VSD VR1,FR3,RS2(RT2) VSSE VR1,VR3,RS2(RT2) VSSE VR1,VR3,RS2(RT2) VSSE VR1,VR3,RS2(RT2) VSSE VR1,FR3,RS2(RT2)	Store (S) Store Channel ID Store Chanacters under Mas Store Clock Store Clock Comparator Store Clock Compressed Store Compressed (L) Store Compressed (S) Store CPU ID Store CPU ID Store CPU Inter Store Halfword Store Indirect (L) Store Indirect (S) Store Matched Store Matched (S) Store Prefix Store Indirect (S) Store Matched (S) Store Prefix Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract Subtract (L) Subtract (L) Subtract (L) Subtract (S)	VST	B203pc BE B205c B207p A40FIC A40FIC A40FIC B6 p B212p B202p B202p B202p B202p B203p A42CIC E401IC R411IC A40EIC A40EIC A40EIC A40EIC A40EIC B211p AC p A62ENC A5A1IM A42IIM A441IM A451IIM A451IIM A491IM A451IIM A451IIM A441IM A451IIM A441IM A451IIM A441IIM A451IIM A441IIM A451IIM A441IIM A451IIM A441IIM A451IIM A441IIM A451IIM A451IIM A441IIM A451IIM
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 VLES VR1,VR2 VLGN VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,VR2 VLCDR VR1,RS3,GR2 VLELD VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLY VR1,RS2(RT2) VLID VR1,VR3,DZ(B2) VLID VR1,VR3,DZ(B2) VLID VR1,VR3,DZ(B2) VLID VR1,VR3,DZ(B2) VLIM VR1,RS2(RT2) VLMQ VR1,GR2 VLMW VR1,RS2(RT2) VLMQ VR1,GR2 VLMW VR1,VR2 VLMD VR1,RS2(RT2) VLMQ VR1,GR2 VLMW VR1,VR2 VLMD VR1,RS2(RT2) VLMQ VR1,FR2 VLMD VR1,RS2(RT2) VLMQ VR1,RS2 VLMR VR1,VR2 VLNEW VR1,VR2 VLNEW VR1,VR2 VLNEW VR1,VR2 VLNEW VR1,VR2 VLPEW VR1,VR2 VLVCU GR1	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element (L) Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Indirect Load Indirect Load Indirect Load Indirect Load Indirect (L) Load Indirect (L) Load Matched Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (L) Load Matched (S) Load Negative Load Negative Load Negative Load Positive Load Positive (L) Load Positive (S) Load Positive (S) Load VCT and Update	QV VV VST QV VV RS RSE VV VV RS VST QV VV VST QV VV	A599 A619 A608 A589 E500 E428 A562 A542 B7 A628 A618 A40B A440B A440B A440A A5AA A50A A50A A55A A55A A55A A55	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCM C D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTK VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STDP D2, (B2) STPT D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTM VR1, RS2(RT2) VSTM VR1, RS2(RT2) VSTMD VR1, RS2(RT2) VSTVP D2(B2) VSTVM RS2 SR R1, R2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, RS2(RT2) VSS VR1, VR3, RS2 VSS VR1, VR3, RS2 VSS VR1, VR3, RS2 VSS VR1, VR3, VR2 VSS VR1, VR3, VR2 VSS VR1, VR3, VR2 VSS VR1, VR3, RS2(RT2) VSS VR1, VR3, VR2 VSS VR1, VR3, RS2(RT2) VSE VR1, VR3, RS2(RT2) VSE VR1, VR3, RS2(RT2) VSE VR1, VR3, RS2(RT2) VSE VR1, VR3, VR2 VSEQ VR1, FR3, RS2(F2) VSES VR1, FR3, RS2(F2)	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU II Store CPU Timer Store Halfword Store Indirect (S) Store Indirect (S) Store Matched Store Matched Store Matched Store Matched Store Matched Store Then OR System Mas Store Then OR System Mas Store VMR Subtract Subtract Subtract Subtract Subtract (L) Subtract (S)	VST S S RS S VST VST VST VST TTS S S VST RSE RSE RSE VST	B203pc BE B205c B207p A40FiC A40FiC A40FiC B6 p B212p B202p B203p A42CiC E401lC R411lC R401lC A40EiC A40EiC A40EiC A40EiC A40EiC A41EiC A40EiC B211p AC p A62RNC B1 c B c A5A1IM A521IM A441IM A511IM A591IM
VLDO VR1,FR2 VLDR VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLISIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 LCTL R1,R3,D2(B2) VLEL VR1,GR3,GR2 VLEL VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLELE VR1,FR3,GR2 VLYVR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLHV R1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLIM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,FR2 VLMQ VR1,FR2 VLMDQ VR1,FR2 VLMDQ VR1,FR2 VLMDQ VR1,FR2 VLMDQ VR1,FR2 VLMB VR1,VR2 VLME VR1,VR2 VLME VR1,VR2 VLNE VR1,VR2 VLNE VR1,VR2 VLNE VR1,VR2 VLNE VR1,VR2 VLPDR VR1,VR2 VLVCU GR1	Load (L) Load (L) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Complement (S) Load Element Load Element Load Element (S) Load Expanded Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Indirect Load Indirect (L) Load Indirect (S) Load Indirect (L) Load Indirect (S) Load Indirect (S) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Megative (L) Load Positive (L)	QV VVST QV VST QV VST RSE RSE VV VST VST VST VST VST VST VST VST VST	A599 A519 A608 A589 E500 E428 A562 A542 B7 A608 A418 A408 A408 A418 A408 A40A A50A A50A A51A A55A A55A A55A A55A A5	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2, (B2) STCK D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STDP D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTH VR1, VR3, D2(B2) VSTH VR1, VR3, D2(B2) VSTH VR1, VR3, D2(B2) VSTMD VR1, RS2(RT2) VSTVM RS2 STR3, RS2(RT2) VSTVM RS2 SR R1, R2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, FR3, RS2(RT2) VSDR VR1, VR3, RS2(RT2) VSPQ VR1, FR3, RS2(RT2) VSEQ VR1, VR3, VR2 VSEQ VR1, FR3, RS2(RT2) VSPSD VR1, FR3, RS2(RT2) VSPSD VR1, FR3, RS2(RT2) VSPSD VR1, FR3, RS2(RT3) VSPSD VR1, FR3	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ITIME Store Lalfword Store Indirect Store Indirect (S) Store Indirect (S) Store Indirect (S) Store Matched (L) Store Indirect (S) Store Matched (S) Store Matched (S) Store Then AND System Mas Store Then AND System Mas Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract (L) Subtract (L) Subtract (L) Subtract (L) Subtract (S) Sum Partial Sums (L) Test Block	VST	B203pc BE B205c B207p A40FIC A40FIC B6 p B212p B202p B202p B202p B202p B42EIC E401IC R40IIC R40IIC R40IIC A40EIC A40EIC B211p AC p A0 p A6CENO A682NC B c A5A1IM A421IM A441IM A591IM A441IM A451IM A461IM A461IM A481IM A481IM A61Aipc B22Cipc
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 VLES VR1,VR2 VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLELE VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLY VR1,RS2(RT2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLIM VR1,RS2(RT2) VLMQ VR1,GR2 VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLVU GR1 VLVG D2(B2) VLVM RS2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Indirect Load Indirect Load Indirect Load Indirect Load Indirect Load Matched Load Matched Load Matched Load Matched (L) Load Matched (S) Load Negative Load Negative Load Positive Load Positive Load Positive (L) Load Positive (S) Load VCT and Update Load VMR Load VMR Complement	QV VV TO QV	A599 A519 A608 A589 E580 E428 A562 A542 B7 A628 A618 A40B A440B A440B A440B A440B A450A A564 A564 A564 A564 A564 A564 A564 A6664 A6681	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTK D VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STPT D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTM VR1, RS2(RT2) VSTM VR1, RS2(RT2) VSTMD VR1, RS2(RT2) VSTMP US, RS2(RT2) VSTVP D2(B2) VSTVM RS2 SR R1, R2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, RS2(RT2) VSS VR1, VR3, VR2 V	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU IImer Store Halfword Store Indirect (S) Store Indirect (S) Store Matched Store Watched Store VMR Store Then OR System Mas Store Verbra Stor	VST	B203pc BE B205c B207p A40FiC A40FiC B6 p B212p B202p B202p B202p B202p A42EiC E401lC R401lC R401lC R401lC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC B211p AC p AC p AD p A62RNC A1B c A62RNC A41IM A42IIM A43IIM A441IM A49IIM A49IIM A40IIM A459IIM A40IIM A459IIM A40IIM A459IIM A40IIM A50IIM A40IIM A50IIM A40IIM A61Pipc B22Cipc E501pc E501pc C64640NC
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 LASP D1(81),D2(B2) VLISIX VR1,GR3,D2(B2) VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 LCTL R1,R3,D2(B2) VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLEL VR1,FR3,GR2 VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLY VR1,RS2(RT2) VLHV R1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,RS2(RT2) VLM VR1,RS2(RT2) VLMQ VR1,FR2 VLMG VR1,FR2 VLMG VR1,FR2 VLMG VR1,FR2 VLMG VR1,FR2 VLMG VR1,FR2 VLMG VR1,VR2 VLMG VR1,VR2 VLMG VR1,VR2 VLMG VR1,VR2 VLNG VR1,VR2 VLPG VR1,VR2 VLVCU GR1 VLVCA D2(B2) VLVM RS2 VL VR1,RS2(RT2)	Load (L) Load (S) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement (L) Load Complement (S) Load Complement (S) Load Element Load Element (L) Load Element (S) Load Expanded (L) Load Expanded (L) Load Expanded (L) Load Expanded (S) Load Halfword Load Indirect Load Indirect Load Indirect (L) Load Matched Load Matched Load Matched Load Matched (L) Load Matched (L) Load Matched (S) Load Positive (L) Load Positive (L) Load Positive (L) Load Positive (L) Load VMR Complement Load VMR Complement	QV VVT QV VST QV VV RS VST VST QV VVT VST QV VVT VST QV VVT VST QV QV VST QV VST QV	A599 A519 A608 A589 E500 E428 A562 A542 B7 A608 A418 A408 A408 A440A A50A A50A A51A A50A A55A A55A A56A A56A A56A A56A A56	IC I	VSTE VR1,RS2(RT2) STIDC D2,(B2) STCM R1,M3,D2(B2) STCM R1,M3,D2(B2) STCK D2,(B2) STCK D2,(B2) STCK D2,(B2) VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKD VR1,RS2(RT2) VSTKE VR1,RS2(RT2) STDP D2,(B2) STPT D2,(B2) STPT D2,(B2) VSTH VR1,RS2(RT2) VSTH VR1,RS2,D(B2) VSTH VR1,RS2,D(B2) VSTH VR1,RS2,D(B2) VSTMD VR1,RS2(RT2) VSTMD VR1,RS2,RT2 VSTMR VR1,RS2,RT2 VSTMR VR1,RS2,RT2 VSTVM RS2 SR R1,R2 VSTVM RS2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,VR2 VSQ VR1,GR3,RS2(RT2) VSD VR1,VR3,RS2(RT2) VSD VR1,VR3,RS2(RT2) VSDR VR1,VR3,RS2 VSD VR1,RR3,RS2(RT2) VSDR VR1,VR3,RS2 VSDR VR1,VR3,VR2 VSDR VR1,VR3,RS2 VSER VR1,VR3,VR2 VSER VR1,VR3,VR2 VSER VR1,VR3,VR2 VSER VR1,VR3,RS2(RT2) VSPSD VR1,FR3,RS2(RT2) VSPSD VR1,FR3	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Compressed (S) Store Compressed (S) Store Control Store CPU Address Store CPU ITIME Store Indirect Store Indirect Store Indirect (S) Store Indirect (S) Store Indirect (S) Store Matched (L) Store Indirect (S) Store Matched (S) Store Matched (S) Store Then AND System Mas Store Then AND System Mas Store Then AND System Mas Store Vector Parameters Store Vector Parameters Store Vector Subtract Subtract Subtract Subtract Subtract (L) Subtract (L) Subtract (L) Subtract (S)	VST	B203pc BE B205c B207p A40FIC A40FIC B6 p B212p B202p B202p B202p B202p B42EIC E401IC R40IIC R40IIC R40IIC A40EIC A40EIC A40EIC B211p A6CRNO A682NC 1B c A5A1IM A421IM A441IM A441IM A441IM A451IM A551IM A451IM A551IM A451IM A551IM A551IM A658NC A668NC A668NC A668NC A661Aipc E501pc E501pc E501pc E601pc 84 P
VLDO VR1,FR2 VLDV VR1,VR2 VLE VR1,RS2(RT2) VLEQ VR1,FR2 VLER VR1,VR2 VLES VR1,VR2 VLGR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCR VR1,VR2 VLCBR VR1,VR2 VLCBR VR1,VR2 VLELE VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLELD VR1,FR3,GR2 VLY VR1,RS2(RT2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLID VR1,VR3,D2(B2) VLIM VR1,RS2(RT2) VLMQ VR1,GR2 VLM VR1,RS2(RT2) VLMQ VR1,GR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLMR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLNR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLPR VR1,VR2 VLVU GR1 VLVG D2(B2) VLVM RS2	Load (L) Load (S) Load (S) Load (S) Load (S) Load Address Space Paran Load Bit Index Load Complement Load Complement (L) Load Complement (S) Load Element Load Element Load Element Load Element (S) Load Expanded Load Expanded Load Expanded Load Expanded Load Expanded (L) Load Indirect Load Indirect Load Indirect Load Indirect Load Indirect Load Matched Load Matched Load Matched Load Matched (L) Load Matched (S) Load Negative Load Negative Load Positive Load Positive Load Positive (L) Load Positive (S) Load VCT and Update Load VMR Load VMR Complement	QV VV TO QV	A599 A519 A608 A589 E580 E428 A562 A542 B7 A628 A618 A40B A440B A440B A440B A440B A450A A564 A564 A564 A564 A564 A564 A564 A6664 A6681	IC I	VSTE VR1, RS2(RT2) STIDC D2, (B2) STCM R1, M3, D2(B2) STCM R1, M3, D2(B2) STCK D2, (B2) STCK D2, (B2) VSTK VR1, RS2(RT2) VSTK D VR1, RS2(RT2) VSTKD VR1, RS2(RT2) VSTKE VR1, RS2(RT2) STDP D2, (B2) STPT D2, (B2) STPT D2, (B2) VSTH VR1, RS2(RT2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTI VR1, VR3, D2(B2) VSTM VR1, RS2(RT2) VSTM VR1, RS2(RT2) VSTMD VR1, RS2(RT2) VSTMP US, RS2(RT2) VSTVP D2(B2) VSTVM RS2 SR R1, R2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, VR2 VSQ VR1, GR3, RS2(RT2) VSS VR1, VR3, VR2 V	Store (S) Store Channel ID Store Characters under Mas Store Clock Store Clock Compressed Store Compressed Store Compressed Store Compressed (L) Store Compressed (S) Store Control Store CPU Address Store CPU ID Store CPU IImer Store Halfword Store Indirect (S) Store Indirect (S) Store Matched Store Watched Store VMR Store Then OR System Mas Store Verbra Stor	VST	B203pc BE B205c B207p A40FiC A40FiC B6 p B212p B202p B202p B202p B202p A42EiC E401lC R401lC R401lC R401lC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC A40EiC B211p AC p AC p AD p A62RNC A1B c A62RNC A41IM A42IIM A43IIM A441IM A49IIM A49IIM A40IIM A459IIM A40IIM A459IIM A40IIM A459IIM A40IIM A50IIM A40IIM A50IIM A40IIM A61Pipc B22Cipc E501pc E501pc C64640NC

IBM 370 Instruction	Description	Fmt	Ор	CI	Instruction	Description	Fmt	Ор	CI	Instruction	Description	Fmt	Op	CI	Psuedo Op	Description
A R1,D2(X2,B2) AD R1,D2(X2,B2)	Add Add Normalized (L)	RX	5A	c f	SD R1,D2(X2,B2) SDR R1,R2	Subtract Normalized (L) Subtract Normalized (L)	RX	6B 2B	C C	VLVCU GR1 VLVM RS2	Load VCT and Update	RRE VS	A645 A680	NO C NC	B R1 / BR R1	Branch to Register Branch on A High (after compare)
ADR R1,R2	Add Normalized (L)	RR	2A	c f	SE R1,D2(X2,B2)	Subtract Normalized (S)	RX	7B	С	VLY VR1,RS2(RT2)	Load Expanded	VST	A40B	IC	BL R1 / BLR R1	Branch on A Low (after compare)
AE R1,D2(X2,B2) AER R1,R2	Add Normalized (S) Add Normalized (S)	RR	3A	c f	SER R1,R2 SH R1,D2(X2,B2)	Subtract Normalized (S) Subtract Halfword	RX	3B 4B	C C	VLYD VR1,RS2(RT2) VLYE VR1,RS2(RT2)	Load Expanded (L) Load Expanded (S)	VST VST	A41B A40B	IC IC	BNH R1 / BNHR R1	Branch on A Equal to B (after compare) Branch on A Not High (after compare)
AH R1,D2(X2,B2) AL R1,D2(X2,B2)	Add Halfword Add Logical		4A 5E	c c i	SIGP R1,R3,D2(B2) SIO D2(B2)	Signal Processor Start I/O	RS S	AE 9C00	pc pc	VLZDR VR1 VLZER VR1	Load Zero (L) Load Zero (S)	VV	A51B A50B	IC	BNL R1 / BNLR R1 BNE R1 / BNER R1	Branch on A Not Low (after compare) Branch on A Equal to B (after compare)
ALR R1,R2 AP D1(L1.B1).D2(L2.B2)	Add Logical Add Decimal	RR SS		c c i	SIOF D2(B2) SL R1.D2(X2.B2)	Start I/O Fast Release Subtract Logical	S RX	9C01 5F	pc c	VLZR VR1 VM VR1,VR3,RS2(RT2)	Load Zero Multiply	VV VST	A50B A522	IC IM	BP R1 / BPR R1 BM R1 / BMR R1	Branch on Plus (after arithmetic) Branch on Minus (after arithmetic)
AR R1,R2 AU R1,D2(X2,B2)	Add Add Unnormalized (S)	RR RX	1A	c i	SLA R1,D2(B2) SLDA R1 D2(B2)	Shift Left Single Shift Left Double	RS RS	8B 8F	C C	VMAD VR1,VR3,RS2(RT2) VMADQ VR1,FR3,VR2	Multiply and Add (L) Multiply and Add (L)	VST	A414 A594	IM IM	BZ R1 / BZR R1 BO R1 / BOR R1	Branch on Zero (after arithmetic) Branch on Overflow (after arithmetic)
AUR R1,R2	Add Unnormalized (S)	RR	3E	c i	SLDL R1,D2(B2)	Shift Left Double Logical	RS	8D	C	VMADS VR1,FR3,RS2(RT2)	Multiply and Add (L)	OST	A494	IM	BNP R1 / BNPR R1	Branch on Not Plus (after arithmetic)
AW R1,D2(X2,B2) AWR R1,R2	Add Unnormalized (L) Add Unnormalized (L)	RR	2E	c i	SLL R1,D2(B2) SLR R1,R2	Shift Left Single Logical Subtract Logical	RS RR		С	VMAE VR1,VR3,RS2(RT2) VMAEQ VR1,FR3,VR2	Multiply and Add (S/L) Multiply and Add (S/L)	VST QV	A404 A584	IM IM	BNZ R1 / BNZR R1	Branch on Not Minus (after arithmetic) Branch on Not Zero (after arithmetic)
AXR R1,R2 BAL R1,D2(X1,B2)	Add Normalized (E) Branch and Link		36 45	c i	SP D1(L1,B1),D2(L2,B2) SPKA D2(B2)	Subtract Decimal Set PSW Key from Address	SS S		c q	VMAES VR1,FR3,RS2(RT2) VMCD VR1,VR3,RS2(RT2)		QST	A484 A416	IM IM	BNO R1 / BNOR R1 BO R1 / BOR R1	Branch on Not Overflow (after arithmetic) Branch if Ones (after mask test)
BALR R1,R2 BAS R1.D2(X2.B2)	Branch and Link Branch and Save		05 4D	i	SPM R1 SPT D2 (B2)	Set Program Mask Set CPU Timer	RR S	04	n p	VMCDR VR1,VR3,VR2 VMCE VR1,VR3,RS2(RT2)	Multiply and Accumulate (L)	VV VST	A516 A406	IM IM	BM R1 / BMR R1 BZ R1 / BZR R1	Branch if Mixed (after mask test) Branch if Zero (after mask test)
BASR R1,R2 BC M1.D2(X2.B2)	Branch and Save Branch on Condition	RR	0D 47		SPX D2,(B2) SR R1.R2	Set Prefix Subtract	S RR	B210 1B	p C	VMCER VR1,VR3,VR2 VMD VR1 VR3 RS2(RT2)		VV VST	A506 A412	IM IM	BNO R1 / BNOR R1	Branch if Not Ones (after mask test) Branch if Not Mixed (after mask test)
BCR M1,R2	Branch on Condition	RR	07		SRA R1,D2(B2)	Shift Right Single	RS	8A	С	VMDQ VR1,FR3,VR2	Multiply (L)	QV	A592	IM	BNZ R1 / BNZR R1	Branch if Not Zero (after mask test)
BCT R1,D2(X2.B2) BCTR R1,R2	Branch on Count Branch on Count	RR	46 06	i	SRDA R1,D2(B2) SRDL R1,D2(B2)	Shift Right Double Shift Right Double Logical	RS RS	8E 8C	С	VMDR VR1,VR3,VR2 VMDS VR1,FR3,RS2(RT2)	Multiply (L) Multiply (L)	VV QST	A512 A492	IM IM	NOP WTO 'Text'	No OPeration Write To Operator
BXH R1,R3,D2(B2) BXLE R1,R3,D2(B2)	Branch on Index High Branch on Index Low	RS RS	86 87	ì	SRL R1,D2(B2) SRP D1(L1.B1),D2(B2),I3	Shift Right Single Logical Shift and Round Decimal	RS SS	88 F0	С	VME VR1,VR3,RS2(RT2) VMEQ VR1,FR3,VR2	Multiply (S/L) Multiply (S/L)	VST	A402 A582	IM IM	Assembler Instruction	Description
C R1,D2(X2,B2) CD R1,D2(X2,B2)	Compare Compare (L)	RX	59	c c i	SSAR R1 SSK R1.R2	Set Secondary ASN Set Storage Key	RRE RR	B225	q p	VMER VR1,VR3,VR2 VMES VR1,FR3,RS2(RT2)	Multiply (S/L) Multiply (S/L)	VV QST	A502 A482	IM IM	DC DS	Define constant Define storage
CDR R1,R2 CDS R1,R3,D2(B2)	Compare (L) Compare Double and Swap	RR	29	c c i	SSKE R1,R2 SSM D2,(B2)	Set Storage Key Extended Set System Mask			p	VMNSD VR1,FR3,GR2 VMNSE VR1,FR3,GR2	Minimum Signed (L) Minimum Signed (S)	VR VR	A611 A601	IM IM	CCM	Define channel command word Define format-0 channel command word
CE R1,D2(X2,B2)	Compare (S)	RX	79	c i	ST R1,D2(X2,B2)	Store	RX	50	р	VMQ VR1,GR3,VR2	Multiply	QV	A5A2	IM	CCW1**	Define format-1 channel command word
CER R1,R2 CH R1,D2(X2,B2)	Compare (S) Compare Halfword	RX	49	c c i	STAP D2,(B2) STC R1,D2(X2,B2)	Store CPU Address Store Character	S RX	42	Р	VMR VR1,VR3,VR2 VMRRS D2(B2)	Multiply Restore VMR	VV S	A522 A6C3	IM NZ	START LOCTR**	Start assembly Specify multiple location counters
CL R1,D2(X2,B2) CLC D1,(L,B1),D2(B2)	Compare Logical Compare Logical	RX SS		c c	STCK D2,(B2) STCKC D2,(B2)	Store Clock Store Clock Comparator	S		C P	VMRSV D2(B2) VMS VR1,GR3,RS2(RT2)	Save VMR Multiply	S QST	A6C1 A5A2	NZ IM	CSE CT DSECT	Identify control section Identify dummy section
CLCL R1,R2 CLI D1(B1),I2	Compare Logical Long Compare Logical	RR SI	0F	i c	STCM R1,M3,D2(B2) STCTL R1.R3.D2(B2)	Store Characters under Mask Store Control	RS TS	BE B6	p	VMSD VR1,VR3,RS2(RT2) VMSDQ VR1,FR3,VR2	Multiply and Subtract (L) Multiply and Subtract (L)	VST	A415 A595	IM IM	DXD*	Define external dummy section Cumulative length of external dummy section
CLM R1,M3,D2(B2)	Comp Logical Chars under Mask	RS	BD	c f	STD R1,D2(X2,B2)	Store (L)	RX	60	Р	VMSDS VR1,FR3,RS2(RT2)	Multiply and Subtract (L)	QST	A495	IM	COM	Identify blank common control section
CLR R1,R2 CLRCH D2(B2)	Compare Logical Clear Channel	RR S	9F01	c f	STE R1,D2(X2,B2) STH R1,D2(X2,B2)	Store (S) Store Halfword	RX RX	70 40		VMSE VR1,VR3,RS2(RT2) VMSEQ VR1,FR3,VR2	Multiply and Subtract (S/L) Multiply and Subtract (S/L)	QV	A405 A585	IM IM	AMODE**	Specify addressing mode Specify residence mode
CLRIO D2(B2) CONCS D2(B2)	Clear I/O Connect Channel Set	S S	B200	pc pc	STIDC D2,(B2) STIDP D2,(B2)	Store Channel ID Store CPU ID	S S	B203 B202	pc p	VMSES VR1,FR3,RS2(RT2) VMXAD VR1,FR3,GR2	Maximum Absolute (L)	QST VR	A485 A612	IM IM	ENTRY EXTRN	Identify entry-point symbol Identify external symbol
CP D1(L1,B1),D2(L2,B2) CR R1,R2	Compare Decimal Compare	SS RR	F9	c i	STM R1,R3,D2(B2) STNSM D1(B1),I2	Store Multiple Store Then AND System Mask	RS	90 AC	р	VMXAE VR1,FR3,GR2 VMXSD VR1,FR3.GR2	Maximum Absolute (S) Maximum Signed (L)	VR VR	A602 A610	IM IM	WXTRN USING	Identify weak external symbol Use base address register
CS R1,R3,D2,(B2) CVB R1,D2(X2.B2)	Compare and Swap Convert to Binary	RS		С	STOSM D1(B1),I2 STPT D2,(B2)	Store Then OR System Mask Store CPU Timer	SI S	AD B209	p	VMXSE VR1,FR3,GR2 VN VR1,VR3,RS2(RT2)	Maximum Signed (S) AND	VR VST	A600 A424	IM IM	DROP TITLE	Drop base address register Identify assembly output
CVD R1,D2(X2,B2)	Convert to Decimal	RX	4E		STPX D2,(B2)	Store CPU Timer Store Prefix Subtract Unnormalized (S)	S	B211	p p	VNQ VR1,GR3,VR2	AND	QV VV	A5A4	IM	EJECT	Sta rt new page
D R1,D2(X2,B2) DD R1,D2(X2,B2)	Divide Divide (L)	RX	5D 5D	f	SU R1,D2(X2,B2) SUR R1,R2	Subtract Unnormalized (S)	RX RR	7F 3F		VNR VR1,VR3,VR2 VNS VR1,GR3,RS2(RT2)	AND AND	QST	A524 A4A4	IM IM	SPACE PRINT	Space listing Print optionaJ data
DDR R1,R2 DE R1,D2(X2,B2)	Divide (L) Divide (S)	RR RX	2D 7D	i	SVC I SW R1,D2(X2,B2)	Supervisor Call Subtract Unnormalized (L)	RR RX	0A 6F		VNVM RS2 VO VR1,VR3,RS2(RT2)	AND to VMR OR	VS VST	A684 A425	NC IM	ICTL ISEQ	Input format control Input sequence checking
DER R1,R2 DISCS D2(B2)	Divide (S) Disconnect Channel Set	RR S	3D	pc f	SWR R1,R2 SXR R1,R2	Subtract Unnormalized (L) Subtract Normalized (E)	RR RR	2F 37		BOQ VR1,GR3,VR2 VOR VR1,VR3,VR2	OR OR	QV VV	A5A5 A525	IM IM	PUNCH RE PRO	Punch a card Reproduce following card
DP D1(L1,B1),D2(L2,B2) DR R1,R2	Divide Decimal Divide	SS	FD 1D		TB R1,R2 TCH D2(B2)	Test Block Test Channel	RRE	B22C	ipc pc	VOS VR1, VR3, VR2 VOS VR1, GR3, RS2(RT2) VOVM RS2	OR OR to VMR	QST VS	A5A5 A685	IM NC	ORG EQU	Set location counter Equate symbol
ED D1,LI,B1),D2(B2)	Edit	SS	DE	c	TIO D2(B2)	Test I/O	S	9D00	pc pc	VRCL D2(82)	Clear VR	S	A6C5	IZ	OPSYN*	Equate operation code
EDMK D1(L,B1),D2(B2) EPAR R1	Edit and Mark Extract Primary ASN		B226	c i q	TM D1(B1),I2 TPROT D1(B1),D2(B2)	Test under Mask Test Protection	SI SSE	91 E501	pc pc	VRRS GR1 VRSV GR1	Restore VR Save VR	RRE RRE	A648 A64A	IZ XC IZ C	PUSH* POP*	Save current PRINT or USING status Restore PRINT or USING status
ESAR R1 EX R1.D2(X2.B2)	Extract Secondary ASN Execute		B227 44	q į	TR D1(L,B1),D2(B2) TRT D1(L,B1),D2(B2)	Translate Translate and Test	SS	DC DD	С	VRSVC GR1 VS VR1,VR3,RS2(RT2)	Save Changed VR Subtract	RRE	A649 A421	IZ PC	LTORG	Begin literal pool Conditional no operation
HDR R1,R2 HDV D2(B2)	Halve (L) Halt Device		24 9E01		TS D2(B2) UNPK D1(L1,B1),D2(L2,B2)	Test and Set	S SS	93 F3	С	VSD VR1,VR3,RS2(RT2) VSDQ VR1,FR3,VR2	Subtract (L) Subtract (L)	VST	A411 A591	IM IM	COPY	Copy predefined source coding End assembly
HER R1,R2 HIO D2(B2)	Halve (S)	RR	34	- i	VA VR1, VR3,RS2(RT2)	Add	VST	A420	IM IM	VSDR VR1,VR3,VR2	Subtract (L)	VV	A511	IM IM	MACRO	Macro definition header
IAC R1	Halt I/O Insert Address Space Control		B224	qp	VACD VR1,RS2(RT2) VACDR VR1,VR2	Accumulate (L) Accumulate (L)	VV	A517	IM	VSDS VR1,FR3,RS2(RT2) VSE VR1,VR3,RS2(RT2)	Subtract (S)	VST	A491 A401	IM	MEND	Macro definition exit Macro definition trailer
IC R1,D2(X2,B2) ICM R1,M3,D2(B2)	Insert Character Insert Characters under Mask		43 BS	С	VACE VR1,RS2(RT2) VACER VR1,VR2	Accumulate (S/L) Accumulate (S/L)	VST	A407 A507	IM IM	VSEQ VR1,FR3,VR2 VSER VR1,VR3,VR2	Subtract (S) Subtract (S)	QV VV	A581 A501	IM IM	AREAD** ACTR	Assign card to SETC symbol Conditional assembly loop counter
IPK IPTE R1 R2	Insert PSW Key Invalidate Page Table Entry			q p	VACRS D2(B2) VACSV D2(B2)	Restore VAC Save VAC	S	A6CB A6CA	NO P	VSES VR1,FR3,RS2(RT2) VSLL VR1, VR3,D2(B2)	Subtract (S) Shift Left Single Logical	QST RSE	A481 F425	IM IM	AGO AIF	Unconditional branch Conditional branch
ISK R1,R2 ISKE R1.R2	Insert Storage Key	RR	09	р	VAD VR1,VR3,RS2(RT2) VADQ VR1,FR3,VR2	Add (L) Add (L)	VST	A410 A490	IM IM	VSPSD VR1,FR2 VSQ VR1,GR3,VR2	Sum Partial Sums (L) Subtract	VR	A61A A5A1	ipc IM	ANOP GBLA	Assembly no operation Define global SETA symbol
IVSK R1,R2	Insert Storage Key Extended Insert Virtual Storage Key	RRE	B223	p q	VADR VR1,VR3,VR2	Add (L)	VV	A510	IM	VSR VR1,VR3.VR2	Subtract	VV	A521	IM	GBLB	Define global SETB symbol
L R1,D2(X2,B2) LA R1,D2(X2,B2)	Load Load Address	RX	58 41		VADS VR1,FR3,RS2(RT2) VAE VR1,VR3,RS2(RT2)	Add (L) Add (S)	QST VST	A490 A400	IM IM	VSRL VR1,VRa,D2(B2) VSRRS D2(B2)	Shift Right Single Logical Restore VSR	RSE S	E424 A6C2	IM IZ X	GBLC LCLA	Define global SETC symbol Define local SET A symbol
LASP D1(B1),D2(B2) LCDR R1,R2	Load Address Space Parameters Load Complement (L)			pc c	VAEQ VR1,FR3,VR2 VAER VR1, VR3,VR2	Add (S) Add (S)	QV VV	A580 A500	IM IM	VS RSV D2(B2) VSS VR1,GR3,RS2(RT2)	Save VSR Subtract	S QST	A6C0 A4A1	NO X IM	LCLB LCLC	Define local SETB symbol Define local SETC symbol
LCER R1,R2 LCR R1.R2	Load Complement (S) Load Complement			c c	VAES VR1,FR3,RS2(RT2) VAQ VR1.GR3.VR2	Add (S) Add	QST QV	A480 A5A0	IM IM	VST VR1,RS2(RT2) VSTD VR1,RS2(RT2)	Store Store (L)	VST	A40D A41D	IC IC	MNOTE MHELP**	Generate error message Trace macro flow
LCTL R1,R3,D2(B2) LD R1.D2(X2.B2)	Load Control Load (L)	RS		p	VAR VR1,VR3,VR2 VAS VR1,GR3,RS2(RT2)	Add Add	VV	A520 A4A0	IM IM	VSTE VR1,RS2(RT2) VSTH VR1,RS2(RT2)	Store (S) Store Halfword	VST	A40D A42D	IC IC	SETA SETB	Set arithmetic variable symbol Set binary variable symbol
LDR R1,R2	Load (L)	RR	28		VC M1,VR3,RS2(RT2)	Compare	VST	A428	IC	VSTI VR1,VR3,D2(B2)	Store Indirect	RSE	E401	IC	SETC	Set character variable symbol
LE R1,D2(X2,B2) LER R1,R2	Load (S) Load (S)	RR	78 38		VCD M1,VR3,RS2(RT2) VCDQ M1,FR3,VR2	Compare (L) Compare (L)	VST QV	A418 A598	IC IC	VSTID VR1,VR3,D2(B2) VSTIE VR1,VR3,D2(B2)	Store Indirect (L) Store Indirect (S)	RSE RSE	R411 R401	IC IC		
LH R1,D2(X2,B2) LM R1,R2,D2(B2)	Load Halfword Load Multiple		48 98		VCDR M1,VR3,VR2 VCDS M1,FR3,RS2(RT2)	Compare (L) Compare (L)	VV QST	A518 A498	IC IC	VSTK VR1,RS2(RT2) VSTKD VR1,RS2(RT2)	Store Compressed Store Compressed (L)	VST	A40F A41F	IC IC		
LNDR R1,R2 LNER R1.R2	Load Negative (L) Load Negative (S)			c c	VCE M1,VR3,RS2(RT2) VCEO M1 FR3 VR2	Compare (S)	VST	A408 A588	IC IC	VSTKE VR1,RS2(RT2) VSTM VR1.RS2(RT2)	Store Compressed (S) Store Matched	VST	A40F A40E	IC IC		
LNR R1,R2 LPDR R1,R2	Load Negative Load Positive (L)	RR	11	c	VCER M1, VR3, VR2 VCES M1, FR3, RS2(RT2)	Compare (S) Compare (S)	VV	A508 A488	IC IC	VSTMD VR1,RS2(RT2) VSTME VR1,RS2(RT2)	Store Matched (L) Store Matched (S)	VST	A41E A40E	IC IC		
LPER R1,R2 LPER R1 R2	Load Positive (S)	RR RR	30	С	VCOVM GR1 VCQ M1.GR3.VR2	Count Ones in VMR	RRE		NC C	VSTVM RS2 VSTVP D2(B2)	Store VMR	VS	A682	NC		
LPSW D2(B2)	Load Positive Load PSW	S	82	c pn	VCR M1.VR3,VR2	Compare Compare	VV	A528	IC IC	VSVMM D2(B2)	Set Vector Mask Mode	S S	A6C8 A6C6	NO NO		
LR R1,R2 LRA R1,D2(X2,B2)	Load Load Real Address	RX		рс	VCS M1,GR3,RS2(RT2) VCVM	Compare Complement VMR			IC NC	VTVM VX VR1, VR3,RS2(RT2)	Test VMR Exclusive OR	RRE VST	A640 A426	NC C IM		
LRDR R1,R2 LRER R1,R2	Load Rounded (E/L) Load Rounded (L/S)	RR	25 35		VCZVM GR1 VDD VR1, VR3,RS2(RT2)	Count Left Zeros in VMR Divide (L)		A642 A413	NC C IM	VXEL VR1 ,GR3,GR2 VXELD VR1,FR3,GR2	Extract Element Extract Element (L)	VR VR	A629 A619	N1 N1		
LTDR R1,R2 LTER R1,R2	Load and Test (L) Load and Test (S)	RR	22	c c	VDDQ VR1,FR3,VR2 VDDR VR1,VR3,VR2	Divide (L) Divide (L)	QV VV	A493	IM IM	VXELE VR1,FR3,GR2 VXQ VR1,GR3,VR2	Extract Element (S) Exclusive OR	VR QV	A609	N1 IM		
LTR R1,R2	Load and Test	RR	12	c	VDDS VR1,FR3,RS2(RT2)	Divide (L)	QST	A493	IM	VXR VR1,VR3,VR2	Exclusive OR	VV	A526	IM		
M R1,D2(X2,B2) MC D1(B1),I2	Multiply Monitor Call	SI	5C AF		VDE VR1,VR3,RS2(RT2) VDEQ VR1,FR3,VR2	Divide (S) Divide (S)	QV	A583	IM IM	VXS VR1,GR3,RS2(RT2) VXVC GR1	Exclusive OR Extract VCT	QST RRE		NO NO		
MD R1,D2(X2,B2) MDR R1,R2	Multiply (L) Multiply (L)	RR	6C 2C		VDER VR1,VR3,VR2 VDES VR1,FR3,RS2(RT2)	Divide (S) Divide (S)	VV QST	A483	IM IM	VXVM RS2 VXVMM GR1	Exclusive OR to VMR Extract Vector Mask Mode	VS RRE	A686 A646	NC NO		
ME R1,D2(X2,B2) MER R1,R2	Multiply (S/L) Multiply (S/L)	RX	7C 3C		VL VR1,RS2(RT2) VLBIX VR1,GR3,D2(B2)	Load VST Load Bit Index	VST	A409 E428	IC IG C	VZPSO VR1 WRD D1(B1),I2	Zero Partial Sums (L) Write Direct	VR SI	A61B 84	IP P		
MH R1,D2(X2,B2) MP D1(L1,B1),D2(L2,B2)	Multiply Halfword Multiply Decimal	RX	4C FC		VLCDR VR1,VR2 VLCER VR1 VR2	Load Complement (L) Load Complement (S)	VV VV	A552		X R1,D2(X2,B2) i XC D1(L,B1),D2(B2)	Exclusive OR Exclusive OR	RX SS	57 D7	C C		
MR R1,R2	Multiply	RR	1C		VLCR VR1,VR2	Load Complement	VV	A562	IM	XI D1(B1),I2	Exclusive OR	SI	97	С		
MVC D1(L,B1),D2(B2) MVCIN D1(L,B1),D2(B2)	Move Characters Move Inverse	SS	D2 E8		VLCVM RS2 VLD VR1,RS2(RT2)	Load VMR Complement Load (L)	VS VST	A419	NC IC	i XR R1,R2 i ZAP D1(L1,B1),D2(L2,B2)	Exclusive OR Zero and Add	RR SS	17 F8	C		
MVCK D1(R1,B1),D2(B2),R3 MVCL R1,R2	Move with Key Move Long	SS		qc i c	VLDO VR1,FR2 VLDR VR1,VR2	Load (L) Load (L)	QV VV		IC IC	Model-dependent	Diagnose	-	83	PY		
MVCP D1(R1,B1),D2(B2),R3 MVCS D1(R1,B1),D2(B2),R3	Move to Primary	SS	DA	qc qc	VLE VR1,RS2(RT2) VLEL VR1,GR3,GR2	Load (S) Load Element	VST VR		IC	c. Condition code set. i. Interruptible instruction.						
MVI D1(B1),L2	Move Immediate	SI	92	₹~	VLELD VR1,GR3,GR2 VLELD VR1,FR3,GR2 VLELE VR1.FR3.GR2	Load Element (L)	VR	A618	N1	n. New condition code loader	d.					
MVN D1(L,B1),D2(B2) MVO D1(L1,B1),D2(L2,B2)	Move Numerics Move with Offset	SS	D1 F1		VLEQ VR1.FR2	Load Element (S) Load (S)	VR QV	A608	N1 IC	 p. Privileged instruction. q. Semiprivileged instruction. 	:					
MVZ D1(L,B1),D2(B2) MXD R1,D2(X2,B2)	Move Zones Multiply (L/E)	RX	D3 67		VLER VR1,VR2 VLH VR1,RS2(RT2)	Load (S) Load Halfword	VV VST		IC IC	 x. Execution in problem state y. Condition code may be se 						
MXDR R1,R2 MXR R1 R2	Multiply (L/E) Multiply (E)	RR	27 26		VLI VR1, VR3,D2(B2) VLID VR1,VR3,D2(B2)	Load Indirect Load Indirect (L)	RSE	E400	IC IC	Floating-point operand length	ns: Notes:					
N R1,D2(X2,B2) NC D1(L,B1),D2(B2)	AND AND	RX	54 D4	С	VLIE VR1, VR3,D2(B2) VLINT VR1,RS2(RT2)	Load Indirect (S) Load Integer Vector		E400 A42A	IC	(E) Extended source and res (E/L) Extended source, long	ult.					
NI D1(B1),I2	AND	SI	94	c	VLM VR1,RS2(RT2)	Load Matched	VST	A40A	IC	(L/E) Long source, extended						
NR R1,R2 O R1,D2(X2,B2)	AND OR		14 56	C	VLMD VR1,RS2(RT2) VLMDQ VR1,FR2	Load Matched (L) Load Matched (L)	VST QV	A41A A59A	IC	(L) Long source and result. (L/S) Long source, short result.						
OC D1(L,B1),D2(B2) OI D1(B11.12	OR OR	SS SI	D6 96	C C	VLMDR VR1,VR2 VLME VR1,RS2(RT2)	Load Matched (L) Load Matched (S)	VV VST	A51A A40A		(S/L) Short source, long resu (S) Short source and result.	ılt.					
OR R1,R2 PACK D1(L1,B1),D2(L2,B2)	OR Pack	RR	16 F2	С	VLMEQ VR1,FR2 VLMER VR1.VR2	Load Matched (S) Load Matched (S)	QV	A58A A50A	IC		at to vector-control bit, CR 0 bit 1	4)				
PC D2(B2)	Program Call	S	B218	q	VLMQ VR1,GR2	Load Matched	QV	A5AA	IC	IC: Interruptible; IVCT - VIX)	elements processed.					
PT R1.R2	Program Transfer Purge TLB	S	B20D	q p	VLMR VR1,VR2 VLNDR VR1,VR2	Load Matched Load Negative (L)	VV		IM	or (section-size - VIX) ele	ount in a general register) elemen ments processed, whichever is f	ewer.				
	Read Direct	SI	85	p pc	VLNER VR1,VR2 VLNR VR1 VR2	Load Negative (S) Load Negative	VV VV	A541 A561	IM IM	IM: Interruptible; (VCT - VIX)	elements processed, vector-ma -number - VIX) elements process	sk mo	de.			
RDD D1(B1).I2	Resume I/O	S														
RDD D1(B1).I2 RIO D2(B2) RRB D2(B2)	Reset Reference Bit	s	B213	DC	VLPDR VR1,VR2	Load Positive (L)	VV		IM	IZ: Interruptible; (section-size						
RDD D1(B1).I2 RIO D2(B2) RRB D2(B2) RABE R1,R2 S R1,D2(X2,B2)	Reset Reference Bit Reset Reference Bit Extended Subtract	S RRE RX	B213 B22A 5B	pc pc	VLPER VR1,VR2 VLPR VR1,VR2	Load Positive (S) Load Positive	VV	A540 A560	IM IM	NC: Not interruptible; (VCT) NZ: Not interruptible; (section	elements processed. n-size) elements processed.	nol	nine\			
PTLB RDD D1(B1).12 RDD D2(B2) RRB D2(B2) RABE R1,R2 SR1,D2(X2,B2) SAC D2(B2) SCK D2(B2) SCKC D2(B2)	Reset Reference Bit Reset Reference Bit Extended	S RRE	B213 B22A 5B B219	pc c q pc	VLPER VR1,VR2	Load Positive (S)	VV	A540	IM IM IC IC	NC: Not interruptible; (VCT) NZ: Not interruptible; (section	elements processed. n-size) elements processed. nents processed (VSRIVAC hou	sekeej	oing).			

Sex. Cu(2c) set under C

IBM Bit order

Normal IBM order Bit Value

	Mos	t Sig	nifica	ant B	ts																							Lea	ast S	ignifi	cant	Bits
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
•	2G	1G	512M	256M	128M	64M	32M	16M	8M	4M	2M	1M	512K	256K	128K	64K	32K	16K	8K	4K	2K	1K	512	256	128	64	32	16	8	4	2	1

Bit order on the PowerPC and IBM370 are the opposite of most systems!

= :	DIC	2														
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F
0x	NUL	SOH	STX	ETX	SEL	HT	RNL	DEL	GE	SPS	RPT	VT	FF	CR	so	SI
1x	DLE	DC1	DC2	DC3	ES/EN	NL	BS	POC	CAN	EM	UBS	CU1	IFS	IGS	IRS	US/ITB
2x	DS	sos	FS	WUS	YP/INF	LF	ETB	ESC	SA	SFE	SM/SW	CSP	MFA	ENQ	ACK	BEL
3x			SYN	IR	PP	TRN	NBS	EOT	SBS	IT	RFF	CU3	DC4	NAK		SUB
4x	SP										¢		<	(+	
5x	&										!	\$	*)	;	7
6x	-	1										,	%	_	>	?
7x										`	:	#	@	•	=	"
8x		а	b	С	d	е	f	g	h	i						±
9x		j	k	- 1	m	n	0	р	q	r						
Ax		~	s	t	u	٧	w	Х	у	z						
Bx	٨										[]				
Cx	{	Α	В	С	D	Е	F	G	Н	1						
Dx	}	J	K	L	М	Ν	0	Р	Q	R						
Ex	١		S	Т	U	٧	W	Х	Υ	Ζ						
Fx	0	1	2	3	4	5	6	7	8	9						EO

- 63 62 0

- 1 62 2 61 4 59 5 58 6 57 7 56

- 8 55 11 52
- 12 51
- 15 48
- 16 47 17 46

- 18 45 19 44 20 43 23 40

- 24 39 31 32 32 31 33 30 63 0

SUPER-H Opcode ADD Rm,Rn ADD #mm,Rn ADDC Rm,Rn ADDV Rm,Rn AND Rm,Rn AND Rmm,R0 AND.B #imm,@(R0,GBR) BF label BRA label BRA label BRAF @Rn BSR label BSRF @Rn	Instruction ADD Binary ADD Binary ADD with Carry ADD with V Flag Overflow Check AND Logical AND Logical AND Logical Branch if False	Adds general register Rn data to Rm data, and stores the result in Rn	Function Rm + Rn — Rn Rn + 8mm — Rn Rn + 8m + T — Rn cary — T Rn + Rm — Rn ownflow — T Rn & Rm — Rn Rn So & mm — RD	Code 0011nnnmmmm1100 0111nnnniiiiiiii 0011nnnnmmmm1110 0011nnnnmmmm1111	Del Slot Tbit Carry Ovfw	1 1 1	Example ADD R0.R1 ADD #H*01,R2 ADDC R3,R1 ADDV R0,R1
ADD #imm,Rn ADDV Rm,Rn ADDV Rm,Rn AND Rm,Rn AND #imm,R0 AND.B #imm,@(R0,GBR) BF label BF/S label BRAF @Rn BRAF @Rn BSR label	ADD Binary ADD with Carry ADD with V Flag Overflow Check AND Logical AND Logical AND Logical	8-bit immediate data can be added instead of Rm data. Since the 8-bit immediate data is sign-extended to 32 bits, this instruction can add and subtract immediate data. Adds Rm data and the 1'bit to general register Rm data, and stores the result in Rn. The T bit changes accounting to the result. Adds general register Rm data to Rm data, and stores the result in Rn. If an overflow occurs, the T bit is set of the Logically ANDs general registers Rn and Rm, and stores the result in Rn.	$\begin{array}{l} Rn + \#imm \to Rn \\ \\ Rn + Rm + T \to Rn, \\ carry \to T \\ Rn + Rm \to Rn, \\ overflow \to T \\ \\ Rn \& Rm \to Rn \end{array}$	0111nnnniiiiiii 0011nnnnmmmm1110 0011nnnnmmmm1111 0010nnnnmmmm1001	- Carry Ovfw	1 1 1	ADD #H'01,R2 ADDC R3,R1
ADDC Rm,Rn ADDV Rm,Rn AND Rm,Rn AND #imm,R0 AND.B #imm,@(R0,GBR) 3F label 3FA label 3RA label 3RAF @Rn 3SR label	ADD with Carry ADD with V Flag Overflow Check AND Logical AND Logical AND Logical	bits, this instruction can add and subtract immediate data. Adds Rm data and the T bit to general register Rn data, and stores the result in Rn. The T bit changes according to the result. Adds general register Rn data to Rm data, and stores the result in Rn. If an overflow occurs, the T bit is set to 1. Logically ANDs general registers Rn and Rm, and stores the result in Rn.	$\begin{aligned} &Rn + Rm + T \rightarrow Rn, \\ &carry \rightarrow T \\ &Rn + Rm \rightarrow Rn, \\ &overflow \rightarrow T \\ &Rn \triangleq Rm \rightarrow Rn \end{aligned}$	0011nnnnmmmm1110 0011nnnnmmmm1111 0010nnnnmmmm1001	Ovfw	1	ADDC R3,R1
ADDV Rm,Rn AND Rm,Rn AND mmm,R0 AND.B #imm,@(R0,GBR) BF label BF/S label BRA label BRAF @Rn BSR label	ADD with V Flag Overflow Check AND Logical AND Logical AND Logical	according to the result. Adds general register Rn data to Rm data, and stores the result in Rn. If an overflow occurs, the T bit is set to 1. Clogically AMDs general registers Rn and Rm, and stores the result in Rn.	$Rn + Rm \rightarrow Rn$, overflow $\rightarrow T$ $Rn \& Rm \rightarrow Rn$	0011nnnnmmmm1111 0010nnnnmmmm1001	Ovfw	1	
ND Rm,Rn ND #mm,R0 ND #mm,Q(R0,GBR) IF label BF/S label BRA label SRAF @Rn SISR label	AND Logical AND Logical AND Logical	Logically ANDs general registers Rn and Rm, and stores the result in Rn.	$Rn \& Rm \rightarrow Rn$				
IND.B #imm,@(R0,GBR) IF label IF/S label IRA label IRA label IRA label IRA label	AND Logical	The contents of general register R0 can be ANDed with zero-extended 8-bit immediate	P0 6 imm - P0			1	AND R0,R1
BF label BF/S label BRA label BRAF @Rn BSR label		8-bit memory data pointed to by GBR relative addressing can be ANDed with 8-bit immediate data.	(R0 + GBR) & imm	11001001iiiiiiii 11001101iiiiiiii	-	1	AND #H'0F,R0 AND.B #H'80,@(R0,GBR)
BRA label BRAF @Rn BSR label		Reads the T bit, and conditionally branches. If T = 0, it branches to the branch destination address. If T = 1, BF	→ (R0 + GBR) When T = 0, disp × 2 + PC → PC; When T = 1, nop	10001101111111111111111111111111111111		3/1	BF TRGET_F
BRA label BRAF @Rn BSR label	December 5 Februarith Delevi Olet	executes the next instruction. The branch destination is an address specified by PC + displacement. Reads the T bit and conditionally branches. If T = 0, it branches after executing the next instruction. If T = 1,		10001111dddddddd	,,	2/1	BF/S TRGET_F
BRAF @Rn 3SR label	Branch if False with Delay Slot	BF/S executes the next instruction. The branch destination is an address specified by PC + displacement.	When T = 0, disp × 2+ PC \rightarrow PC; When T = 1, nop	10001111aaaaaaaa	Y -	2/1	BF/S IRGEI_F
BSR label	Branch	Branches unconditionally after executing the instruction following this BRA instruction. The branch destination is an address specified by PC + displacement However, in this case it is	$disp \times 2 + PC \to PC$	1010dddddddddddd	Υ -	2	BRA TRGET
	Branch Far	used for address calculation. Branches unconditionally. The branch destination is PC + the 32-bit contents of the	$Rn + PC \rightarrow PC$	0000mmmm00100011	Υ -	2	
DODE @D-	Branch to Subroutine	general register Rm. Branches to the subroutine procedure at a specified address. The PC value is stored	$PC \to PR, disp \times 2 + PC \to PC$	1011dddddddddddd	Υ -	2	BSR TRGET
	Branch to Subroutine Far	in the PR, and the program branches to an address specified by PC + displacement However, in this case it is used for address calculation. Branches to the subroutine procedure at a specified address after executing the	PC → PR. Rn + PC → PC	0000mmmm00000011	Y -	2	BRSF R0
ST label	Branch if True	instruction following this BSRF instruction. The PC value is stored in the PR.	When T = 1, disp × 2 + PC → PC; When T = 0, nop	10001001dddddddd	1 -	3/1	BT TRGET_T
BT/S label	Branch if True with Delay Slot	Reads the T bit and conditionally branches. If T = 1, BT/S branches after the	When T = 1,disp × 2 + PC \rightarrow PC;	100011011dddddddd	Y -	2/1	BT/S TARGET_T
		following instruction executes. If T = 0, BT/S executes the next instruction. The branch destination is an address specified by PC + displacement	When T = 0, nop				
CLRMAC CLRT	Clear MAC Register Clear T Bit	Clear the MACI rand MACE Register.	0 → MACH, MACL 0 → T	000000000101000 0000000000001000	-	1	CLRMAC CLRT
CMP/EQ Rm,Rn CMP/GE Rm.Rn	Compare Equal Compare Greater or Equal (signed)	1141 1411, 1	When Rn = Rm,1 \rightarrow T When signed and Rn • Rm, 1 \rightarrow T	0011nnnnmmmm0000 0011nnnnmmmm0011		1	CMP/GE R0,R1
CMP/GT Rm,Rn	Compare Greater Than (Signed)	If Rn > Rm with signed data, T = 1	When signed and Rn > Rm, 1 \rightarrow T	0011nnnnmmmm0111	resit	1	GWII 7GE IXO,IXI
CMP/HI Rm,Rn CMP/HS Rm,Rn	Compare Higher (Unsigned) Compare Higher or Same (Unsigned)	in ture run mur droighod data, r	When unsigned and Rn > Rm, $1 \rightarrow T$ When unsigned and Rn • Rm, $1 \rightarrow T$	0011nnnnmmmm0110 0011nnnnmmmm0010		1	CMP/HS R0,R1
CMP/PL Rn	Compare if Plus Compare if Plus or Zero	If Rn > 0, T = 1	When Rn > 0, 1 \rightarrow T When Rn • 0, 1 \rightarrow T	0100nnnn00010101	resit	1	
CMP/PZ Rn CMP/STR Rm,Rn	Compare String	If a byte in Rn equals a byte in Rm, T = 1 (Any of 4 bytes, must be same position in Rm+Rn)	When byte in Rn = byte in Rm, $1 \rightarrow T$	0100nnnn00010001 0010nnnnmmmm1100	resit resit	1	CMP/STR R2,R3
CMP/EQ #imm,R0 DIV0S Rm,Rn	Compare Equal Immediate Divide Step 0 as Signed	DIVOS is an initialization instruction for signed division. It finds the quotient by	When R0 = imm, $1 \rightarrow T$ MSB of Rn \rightarrow Q, MSB of Rm \rightarrow M,M^Q \rightarrow T	10001000iiiiiiii 0010nnnnmmmm0111		1	DIV0S R0,R1
		instruction.					
DIV0U	Divide Step 0 as Unsigned	repeatedly dividing in combination with the DIV1 or another instruction that divides for each bit after this	$0 \rightarrow M/Q/T$	000000000011001	0	1	DIVOU
DIV1 Rm,Rn	Divide 1 Step	instruction. Uses single-step division to divide one bit of the 32-bit data in general register Rn (divided by De data (divisor) It finds a guallant through condition of the independently or used in	1 step division (Rn ÷ Rm)	0011nnnnmmmm0100	resit	1	DIV1 R0,R1
		(dividend) by Rm data (divisor). It finds a quotient through repetition either independently or used in combination with other instructions. During this repetition, do not rewrite the specified register or the M, Q, and T bits.					
DMULS.L Rm,Rn	Double-Length Multiply as Signed	Performs Signed multiplication of 32-bit Rn and Rm, and stores the 64-bit results in MACH:MACL	With sign,Rn × Rm →MACH, MACL	0011nnnnmmmm1101	-	2-4	DMULS.L R0,R1
OMULU.L Rm,Rn OT Rn	Double-Length Multiply as Unsigned Decrement and Test (DJNZ)	Performs Unsilgned multiplication of 32-bit Rn and Rm, and stores the 64-bit results in MACH:MACL The contents of general register Rn are decremented by 1 and the result compared to	Without sign,Rn × Rm →MACH, MACL Rn – 1 → Rn;When Rn is 0,1 → T,	0011nnnnmmmm0101 0100nnnn00010000	- resit	2-4	DMULU.L R0,R1 DT R5
EXTS.B Rm,Rn	Extend as Signed	O (zero). When the result is 0, the T bit is set to 1. When the result is not zero, the T bit is set to 0. Sign-extends general register Rm data, and stores the result in Rn	when Rn is nonzero, $0 \rightarrow T$ Sign-extend Rm from byte \rightarrow Rn	0110nnnnmmmm1110	-	1	EXTS.B R0,R1
EXTS.W Rm,Rn	Extend as Signed	Sign-extends general register Rm data, and stores the result in Rn	Sign-extend Rm from word → Rn Zero-extend Rm from byte → Rn	0110nnnnmmmm1111		1	EXTS.W R0,R1
EXTU.B Rm,Rn EXTU.W Rm,Rn	Extend as Unsigned Extend as Unsigned	Zero-extends general register Rm data, and stores the result in Rn. Zero-extends general register Rm data, and stores the result in Rn.	Zero-extend Rm from byte → Rn Zero-extend Rm from word → Rn	0110nnnnmmmm1100 0110nnnnmmmm1101	1	1	EXTU.B R0,R1 EXTU.W R0,R1
JMP @Rn	Jump	Branches unconditionally to the address specified by register indirect addressing. The branch destination is an address specified by the 32-bit data in general register Rm.		0100mmmm00101011	Υ -	2	JMP @R0
JSR @Rn	Jump to Subroutine	Branches to the subroutine procedure at the address specified by register indirect addressing. The PC value is stored in the PR. The jump destination is an address specified by the 32-bit data	$PC \rightarrow PR, Rn \rightarrow PC$	0100mmmm00001011	Υ -	2	JSR @R0
LDC Rm,SR	Load to Control Register	in general register Rm. The stored/saved PC is the address four bytes after this instruction. Store the source operand into control register	$Rm \rightarrow SR$	0100mmmm00001110	LSB	1	LDC R0,SR
LDC Rm,GBR LDC Rm,VBR	Load to Control Register Load to Control Register	Store the source operand into control register Store the source operand into control register	$Rm \rightarrow GBR$ $Rm \rightarrow VBR$	0100mmmm00011110 0100mmmm00101110		1	
LDC.L @Rm+,SR	Load to Control Register	Store the source operand into control register	$(Rm) \rightarrow SR, Rm + 4 \rightarrow Rm$ $(Rm) \rightarrow GBR, Rm + 4 \rightarrow Rm$	0100mmmm00000111	LSB	3	
LDC.L @Rm+,GBR LDC.L @Rm+,VBR	Load to Control Register Load to Control Register	Store the source operand into control register Store the source operand into control register	$(Rm) \rightarrow GBR, Rm + 4 \rightarrow Rm$ $(Rm) \rightarrow VBR, Rm + 4 \rightarrow Rm$	0100mmmm00010111 0100mmmm00100111		3	LDC.L @R15+,GBR
LDS Rm,MACH LDS Rm,MACL	Load to System Register Load to System Register	Store the source operand into the system register MACH, MACL, or PR or the DSP	$Rm \rightarrow MACH$ $Rm \rightarrow MACL$	0100mmmm00001010 0100mmmm00011010	-	1	
LDS Rm,PR	Load to System Register	MSB of the data is copied into A0G.	$Rm \to PR$	0100mmmm00101010	-	1	LDS R0,PR
LDS.L @Rm+,MACH LDS.L @Rm+,MACL	Load to System Register Load to System Register	clore the course operand into the cyclon register in tert, in tez, or river the Ber	$(Rm) \rightarrow MACH,Rm + 4 \rightarrow Rm$ $(Rm) \rightarrow MACL,Rm + 4 \rightarrow Rm$	0100mmmm00000110 0100mmmm00010110	-	1	LDS.L @R15+,MACL
LDS.L @Rm+,PR	Load to System Register	MSB of the data is copied into A0G.	(Rm) → PR,Rm + 4 → Rm	0100mmmm00100110	-	1	
MAC.L @Rm+,@Rn+	Multiply and Accumulate Calculation Long	registers Rm and Rn as addresses. The 64-bit result is added to contents of the MAC	Signed operation (Rn) × (Rm) + MAC→ MAC	0000nnnnmmm1111	-	3/2-4	MAC.L @R0+,@R1+
		register, and the final result is stored in the MAC register. Every time an operand is read, they increment Rm and Rn by four.					
MAC.W @Rm+,@Rn+ MAC @Rm+,@Rn+	Multiply and Accumulate Calculation Word	Performs 16-bit multiplication of signed @Rn and @Rm, with post increment, and Adds the 32-bit result in MACL. MACH is any overflow (1 bit)	With sign, (Rn) × (Rm) + MAC → MAC	0100nnnnmmmm1111	-	3/2-4	MAC.W @R0+,@R1+
MOV Rm,Rn MOV.B Rm,@Rn	Move Data Move Data	Transfers the source operand to the destination. When the operand is stored in memory, the transferred data can be a byte, word, or longword. Loaded data from	$Rm \rightarrow Rn$ $Rm \rightarrow (Rn)$	0110nnnnmmmm0011 0010nnnnmmmm0000	-	1	MOV R0,R1
MOV.W Rm,@Rn	Move Data	memory is stored in a register after it is sign-extended to a longword.	$Rm \rightarrow (Rn)$	0010nnnnmmmm0001	-	1	MOV.W R0,@R1
MOV.L Rm,@Rn MOV.B @Rm,Rn	Move Data Move Data		$Rm \rightarrow (Rn)$ $(Rm) \rightarrow sign extension \rightarrow Rn$	0010nnnnmmmm0010 0110nnnnmmmm0000		1	
MOV.W @Rm,Rn	Move Data		(Rm) → sign extension → Rn (Rm) → Rn	0110nnnnmmmm0001 0110nnnnmmmm0010	-	1	
MOV.L @Rm,Rn MOV.B Rm,@–Rn	Move Data Move Data		$Rn - 1 \rightarrow Rn, Rm \rightarrow (Rn)$	0010nnnnmmmm0100		1	
MOV.W Rm,@–Rn MOV.L Rm,@–Rn	Move Data Move Data		$Rn - 2 \rightarrow Rn, Rm \rightarrow (Rn)$ $Rn - 4 \rightarrow Rn, Rm \rightarrow (Rn)$	0010nnnnmmmm0101 0010nnnnmmmm0110	-	1	MOV.W R0,@-R1
MOV.B @Rm+,Rn	Move Data		$(Rm) \rightarrow sign ext \rightarrow Rn, Rm + 1 \rightarrow Rm$ $(Rm) \rightarrow sign ext \rightarrow Rn, Rm + 2 \rightarrow Rm$	0110nnnnmmmm0100	-	1	MOV.B @R0,R1
MOV.W @Rm+,Rn MOV.L @Rm+,Rn	Move Data Move Data		$(Rm) \rightarrow Sign ext \rightarrow Rm, Rm + 2 \rightarrow Rm$ $(Rm) \rightarrow Rn, Rm + 4 \rightarrow Rm$	0110nnnnmmmm0101 0110nnnnmmmm0110	1	1	MOV.L @R0+,R1
MOV.B Rm,@(R0,Rn) MOV.W Rm,@(R0,Rn)	Move Data Move Data		$Rm \rightarrow (R0 + Rn)$ $Rm \rightarrow (R0 + Rn)$	0000nnnnmmmm0100 0000nnnnmmmm0101	-	1	MOV.B R1,@(R0,R2)
MOV.L Rm,@(R0,Rn)	Move Data		$Rm \rightarrow (R0 + Rn)$	0000nnnnmmmm0110	-	1	
MOV.B @(R0,Rm),Rn MOV.W @(R0,Rm),Rn	Move Data Move Data		(R0 + Rm) → sign extension → Rn (R0 + Rm) → sign extension → Rn	0000nnnnmmmm1100 0000nnnnmmmm1101	-	1	MOV.W @(R0,R2),R1
MOV.L @(R0,Rm),Rn MOV #imm,Rn	Move Data Move Immediate Data	Stores immediate data, sign-extended to a longword, into general register Rn.	$(R0 + Rm) \rightarrow Rn$ $imm \rightarrow sign extension \rightarrow Rn$	0000nnnnmmmm1110 1110nnnniiiiiii	-	1	MOV #H'80,R1
MOV.W @(disp,PC),Rn	Move Immediate Data	Stores immediate data, sign-extended to a longword, into general register Rn.	$(disp \times 2 + PC) \to sign\;ext \to Rn$	1001nnnndddddddd	-	1	MOV.W IMM,R2
MOV.L @(disp,PC),Rn MOV.B @(disp,GBR),R0	Move Immediate Data Move Peripheral Data	Stores immediate data, sign-extended to a longword, into general register Rn. Transfers the source operand to the dest. Designed for the peripheral module area.	$(disp \times 4 + PC) \rightarrow Rn$ $(disp + GBR) \rightarrow sign ext \rightarrow R0$	1101nnnndddddddd 11000100dddddddd		1	MOV.L @(4,PC),R3
MOV.W @(disp,GBR),R0	Move Peripheral Data	Transfers the source operand to the dest. Designed for the peripheral module area.	$(disp \times 2 + GBR) \rightarrow sign ext \rightarrow R0$ $(disp \times 4 + GBR) \rightarrow R0$	11000101dddddddd	-	1	MOVI @(2.CPD) D2
MOV.L @(disp,GBR),R0 MOV.B R0,@(disp,GBR)	Move Peripheral Data Move Peripheral Data	Transfers the source operand to the dest. Designed for the peripheral module area.	R0 → (disp + GBR)	11000110dddddddd 11000000dddddddd		1	MOV.L @(2,GBR),R0 MOV.B R0,@(1,GBR)
MOV.W R0,@(disp,GBR) MOV.L R0,@(disp,GBR)	Move Peripheral Data Move Peripheral Data	Transfers the source operand to the dest. Designed for the peripheral module area.	$R0 \rightarrow (disp \times 2 + GBR)$ $R0 \rightarrow (disp \times 4 + GBR)$	11000001dddddddd 11000010dddddddd	-	1	
MOV.B R0,@(disp,Rn)	Move Structure Data	Transfers the source operand to the dest. Designed for structure or a stack.	$R0 \rightarrow (disp + Rn)$	10000000nnnndddd	-	1	
MOV.W R0,@(disp,Rn) MOV.L Rm,@(disp,Rn)	Move Structure Data Move Structure Data	Transfers the source operand to the dest. Designed for structure or a stack. Transfers the source operand to the dest. Designed for structure or a stack.	$R0 \rightarrow (disp \times 2 + Rn)$ $Rm \rightarrow (disp \times 4 + Rn)$	10000001nnnndddd 0001nnnnmmmmdddd		1	MOV.L R0,@(HF,R1)
MOV.B @(disp,Rn),R0 MOV.W @(disp,Rn),R0	Move Structure Data Move Structure Data	Transfers the source operand to the dest. Designed for structure or a stack. Transfers the source operand to the dest. Designed for structure or a stack.	$(disp + Rn) \rightarrow sign \ extension \rightarrow R0$ $(disp \times 2 + Rn) \rightarrow sign \ extension \rightarrow R0$	10000100mmmmdddd 10000101mmmmdddd	-	1	
MOV.L @(disp,Rm),Rn	Move Structure Data	Transfers the source operand to the dest. Designed for structure or a stack.	$(disp \times 4 + Rm) \to Rn$	0101nnnnmmmmdddd		1	MOV.L @(2,R0),R1
	Move Effective Address	Stores the effective address of the source operand into general register R0. The 8-bit displacement is zero-extended and quadrupled	$disp \times 4 + PC \rightarrow R0$ $T \rightarrow Rn$	11000111dddddddd	-	1	MOVA @(0,PC),R0
MOVT Rn	Move T Bit	when T = 0, 0 is stored in Rn.		0000nnnn00101001		1	MOVT R0 MULL R0,R1
MUL.L Rm,Rn MUL Rm,Rn	Multiply Long	Performs 32-bit multiplication of Rn and Rm (Signed or unsigned), and stores the bottom 32 bits of the result in the MACL register. MACH is unchanged		0000nnnnmmmm0111		2-4	
MULS.W Rm,Rn MULS Rm,Rn	Multiply as Signed Word	Performs 16-bit multiplication of signed Rn and Rm, and stores the 32-bit result in MACL. MACH is unchanged		0010nnnnmmmm1111	-	1-3	MULS R0,R1
MULU.W Rm,Rn MULU Rm,Rn	Multiply as Unsigned Word	Performs 16-bit multiplication of unsigned Rn and Rm, and stores the 32-bit result in MACL. MACH is unchanged	Unsigned, Rn × Rm → MACL	0010nnnnmmm1110	-	1-3	MULU R0,R1
	Negate	Takes the two's complement of data in general register Rm, and stores the result in Rn. This effectively subtracts Rm data from 0, and stores the result in Rn.	$0 - Rm \rightarrow Rn$	0110nnnnmmmm1011	-	1	NEG R0,R1
NEG Rm,Rn	Negate with Carry	Subtracts general register Rm data and the T bit from 0, and stores the result in Rn. If a borrow is generated. T bit changes accordingly. This instruction is used for inverting the sign of a value that	$0-Rm-T\to Rn,Borrow\to T$	0110nnnnmmmm1010	-	1	NEGC R1,R1
	No operation	has more than 32 bits. Increments the PC to execute the next instruction.	No operation	0000000000001001	-	1	NOP
NEGC Rm,Rn		Takes the one's complement of general register Rm data, and stores the result in Rn. This effectively inverts each bit of Rm data and stores the result in Rn.	\sim Rm → Rn	0110nnnnmmmm0111	-	1	NOT R0,R1
NEGC Rm,Rn	NOT—Logical Complement					4	OR R0.R1
NEGC Rm,Rn NOP NOT Rm,Rn DR Rm,Rn	OR Logical	Logically ORs the contents of general registers Rn and Rm, and stores the result in	$Rn \mid Rm \rightarrow Rn$ $R0 \mid imm \rightarrow R0$	0010nnnnmmmm1011		1	OR #H'F0.R0
NEGC Rm,Rn NOP NOT Rm,Rn DR Rm,Rn DR #imm,R0 DR.B #imm,@(R0,GBR)	OR Logical OR Logical OR Logical	Logically ORs the contents of general registers Rn and Rm, and stores the result in Rn. The contents of general register R0 can also be ORed with zero-extended 8-bit immediate data, or 8-bit memory data accessed by using indirect indexed GBR addressing can be ORed with 8-bit immediate data.	R0 imm \rightarrow R0 (R0 + GBR) imm \rightarrow (R0 + GBR)	11001011iiiiiii 11001111iiiiiiii	-	1 3	OR #H'F0,R0 OR.B #H'50,@(R0,GBR)
NEGC Rm,Rn NOP NOT Rm,Rn DR Rm,Rn DR #imm,R0 DR.B #imm,@(R0,GBR) ROTCL Rn	OR Logical OR Logical OR Logical Rotate with Carry Left	Logically ORs the contents of general registers Rn and Rm, and stores the result in Rn. The contents of general register Ro can also be ORed with zero-extended 8-bit immediate data, or 8-bit memory data accessed by using indirect indexed GBR addressing can be ORed with 8-bit immediate data. Rolates the contents of general register Rn and the T bit to the left by one bit, and stores the result in Rn. The bit that is shifted out is storakered to the T bit.	R0 imm → R0 (R0 + GBR) imm → (R0 + GBR) T ← Rn ← T	11001011iiiiiii 110011111iiiiiii 0100nnnn00100100		1 3 1	OR #H'F0,R0 OR.B #H'50,@(R0,GBR) ROTCL R0
NEG Rm,Rn NOP NOT Rm,Rn OR Rm,Rn OR #imm,R0 OR.B #imm,@(R0,GBR) ROTCL Rn ROTCR	OR Logical OR Logical OR Logical Rotate with Carry Left Rotate with Carry Right	Logically ORs the contents of general registers Rn and Rm, and stores the result in mediate data, or 8-bit memory data accessed by using indirect indexed GBR addressing can be ORed with 8-re-ventioned 8-bit immediate data. Rotates the contents of general register Rn and the T bit to the left by one bit, and stores the result in Rn. The bit that is shifted out is transferred to the T bit. Rotates the contents of general register Rn and the T bit to the left by one bit, and stores the result in Rn. The bit that is shifted out is transferred to the T bit.	R0 imm \rightarrow R0 (R0 + GBR) imm \rightarrow (R0 + GBR)	11001011iiiiiii 11001111iiiiiii 0100nnnn00100100 0100nnnn00100101	LSB	1	OR #H*F0,R0 OR.B #H*50,@(R0,GBR) ROTCL R0 ROTCR R0
NEGC Rm,Rn NOP NOT Rm,Rn DR Rm,Rn DR #imm,R0 DR.B #imm,@(R0,GBR) ROTCL Rn	OR Logical OR Logical OR Logical Rotate with Carry Left	Logically ORs the contents of general registers Rn and Rm, and stores the result in Rn. The contents of general register Ro can also be ORed with zero-extended 8-bit immediate data, or 8-bit memory data accessed by using indirect indexed GBR addressing can be ORed with 8-bit immediate data. Rolates the contents of general register Rn and the T bit to the left by one bit, and stores the result in Rn. The bit that is shifted out is storakered to the T bit.	R0 $ \text{Imm} \rightarrow \text{R0}$ $(\text{R0} + \text{GBR}) \text{Imm} \rightarrow (\text{R0} + \text{GBR})$ $T \leftarrow \text{Rn} \leftarrow T$ $T \rightarrow \text{Rn} \rightarrow T$	11001011iiiiiii 110011111iiiiiii 0100nnnn00100100	LSB MSB	1	OR #H'F0,R0 OR.B #H'50,@(R0,GBR) ROTCL R0

RTS	Return from Subroutine		Delayed branch, PR → PC	0000000000001011	Υ -	2	RTS
OFTT	C-4 T D#	the program from a subroutine program called by a BSR, BSRF, or JSR instruction.	1→T	0000000000011000			SETT
SETT SHAL Rn	Set T Bit	Sets the T bit to 1. Arithmetically shifts the contents of general register Rn to the left by one bit, and stores the result in Rn. The bit		0000000000011000	1	1	SHAL RO
SHAL RII	Shift Arithmetic Left 1 Bit with carry	that is shifted out of the operand is transferred to the T bit		0100nnnn00100000	MSB	1	STIPLE NO
SHAR Rn	Shift Arithmetic Right 1 Bit with carry	Arithmetically shifts the contents of general register Rn to the right by one bit, and stores the result in Rn. The bit that is shifted out is transferred to the T bit	$MSB \rightarrow Rn \rightarrow T$	0100nnnn00100001	LSB	1	SHAR R0
SHLL Rn	Shift Logical Left 1 Bit with carry	Logically shifts the contents of general register Rn to the left by one bit, and stores the result in Rn. The bit that is shifted out of the operand is transferred to the T bit	$T \leftarrow Rn \leftarrow 0$	0100nnnn00000000	MSB	1	SHLL R0
SHLL2 Rn	Shift Logical Left 2 Bits	Logically shifts the contents of general register Rn to the left by 2, 8, or 16 bits, and stores the result in Rn. Bits that are shifted out of the operand are not stored	$Rn \ll 2 \rightarrow Rn$	0100nnnn00001000	-	1	SHLL2 R0
SHLL8 Rn	Shift Logical Left 8 Bits	stores the result in Kir. Bits that are shifted out of the operand are not stored	$Rn \ll 8 \rightarrow Rn$	0100nnnn00011000	-	1	SHLL8 R0
SHLL16 Rn	Shift Logical Left 16 Bits		Rn << 16 → Rn	0100nnnn00101000	-	1	SHLL16 R0
SHLR Rn	Shift Logical Right 1 Bit with carry	Logically shifts the contents of general register Rn to the right by one bit, and stores the result in Rn. The bit that is shifted out of the operand is transferred to the T bit	$0 \to Rn \to T$	0100nnnn00000001	LSB	1	SHLR R0
SHLR2 Rn	Shift Logical Right 2 Bits	Logically shifts the contents of general register Rn to the right by 2, 8, or 16 bits, and stores the result in Rn. Bits that are shifted out of the operand are not stored	Rn>>2 → Rn	0100nnnn00001001	-	1	SHLR2 R0
SHLR8 Rn	Shift Logical Right 16 Bits	bits that are shifted out of the operand are not stored	Rn>>8 → Rn	0100nnnn00011001	-	1	SHLR8 R0
SHLR16 Rn	Shift Logical Right 16 Bits		Rn>>16 → Rn	0100nnnn00101001	-	1	SHLR16 R0
SLEEP	Sleep	Sets the CPU into power-down mode. CPU waits for an interrupt request.	Sleep	0000000000011011	-	3	SLEEP
STC SR,Rn	Store Control Register	Stores control register into a specified destination.	SR → Rn	0000nnnn00000010	-	1	STC SR,R0
STC GBR,Rn	Store Control Register	Stores control register into a specified destination.	GBR → Rn	0000nnnn00010010	-	1	
STC VBR,Rn	Store Control Register	Stores control register into a specified destination.	VBR → Rn	0000nnnn00100010	-	1	
STC.L SR,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn, SR \rightarrow (Rn)$	0100nnnn00000011	-	2	
STC.L GBR,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn, GBR \rightarrow (Rn)$	0100nnnn00010011	-	2	STC.L GBR,@-R15
STC.L VBR,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn$, $VBR \rightarrow (Rn)$	0100nnnn00100011	-	2	
STS MACH,Rn	Store System Register	Stores data from system register into a specified destination	MACH → Rn	0000nnnn00001010	-	1	STS MACH,R0
STS MACL,Rn	Store System Register	Stores data from system register into a specified destination	MACL → Rn	0000nnnn00011010	-	1	
STS PR,Rn	Store System Register	Stores data from system register into a specified destination	PR → Rn	0000nnnn00101010	-	1	
STS.L MACH,@-Rn	Store System Register	Stores data from system register into a specified destination	$Rn - 4 \rightarrow Rn,MACH \rightarrow (Rn)$	0100nnnn00000010	-	1	
STS.L MACL,@-Rn	Store System Register	Stores data from system register into a specified destination	$Rn - 4 \rightarrow Rn,MACL \rightarrow (Rn)$	0100nnnn00010010	-	1	
STS.L PR,@-Rn	Store System Register	Stores data from system register into a specified destination	$Rn = 4 \rightarrow Rn,PR \rightarrow (Rn)$	0100nnnn00100010	-	1	STS.L PR,@-R15
SUB Rm,Rn	Subtract Binary	Subtracts general register Rm data from Rn data, and stores the result in Rn. To subtract immediate data, use ADD #imm,Rn.	$Rn - Rm \rightarrow Rn$	0011nnnnmmmm1000	-	1	SUB R0,R1
SUBC Rm,Rn	Subtract with Carry	Subtracts Rm data and the T bit value from Rn data, and stores the result in Rn. The T bit changes according to the result.	$Rn - Rm - T \rightarrow Rn$, $Borrow \rightarrow T$	0011nnnnmmmm1010	-	1	SUBC R3,R1
SUBV Rm,Rn	Subtract with V Flag Underflow Check	Subtracts Rm data from general register Rn data, and stores the result in Rn. If an underflow occurs, the T bit is set to 1.	$Rn - Rm \rightarrow Rn$, underflow $\rightarrow T$	0011nnnnmmmm1011	Unde Flow		SUBV R0,R1
SWAP.B Rm,Rn	Swap Register Halves		Rm → Swap upper and lower halves of lower 2 bytes → Rn	0110nnnnmmmm1000	-	1	SWAP.B R0,R1
SWAP.W Rm,Rn	Swap Register Halves	transferred to the upper 16 bits of Rn. If a word is specified, bits 0 to 15 of Rm are swapped for bits 16 to 31.	Rm → Swap upper and lower word → Rn	0110nnnnmmmm1001	-	1	SWAP.W R0,R1
TAS.B @Rn	Test and Set	Reads byte data from address Rn, and sets the T bit to 1 if the data is 0, or clears the T bit to 0 otherwise. Data bit 7 of (Rn) is then set to 1 (Whatever happened to T). During this operation, the bus is not released.	When (Rn) is 0, 1 \rightarrow T, 1 \rightarrow MSB of (Rn)	0100nnnn00011011	resit	4	TAS.B @R7
TRAPA #imm	Trap Always	Execute Trap imm from vector table specified by VBR (offset=imm*4) The PC and SR values are stored on the stack	(imm × 4 + VBR) → PC	11000011iiiiiii	-	8	TRAPA #H'20
TST Rm,Rn	Test Logical	Logically ANDs the contents of general registers Rn and Rm, and sets the T bit to 1	Rn & Rm, when result is 0, 1 \rightarrow T	0010nnnnmmmm1000	resit	1	TST R0,R0
TST #imm,R0	Test Logical	if the result is 0 or clears the T bit to 0 if the result is not 0. The Rn data does not change. The contents of general register R0 can also be ANDed with zero-extended 8-bit immediate data, or the contents of 8-bit	R0 & imm, when result is 0, 1 \rightarrow T	11001000iiiiiiii	resit	1	TST #H*80,R0
TST.B #imm, @(R0,GBR)	Test Logical	memory accessed by indirect indexed GBR addressing can be ANDed with 8- bit immediate data. The R0 and memory data do not change.	(R0 + GBR) & imm, when result is 0, 1 → T	11001100iiiiiiii	resit	1	TST.B #H'A5,@(R0,GBR)
XOR Rm.Rn	Exclusive OR Logical	Exclusive ORs the contents of general registers Rn and Rm, and stores the result in	$Rn \wedge Rm \rightarrow Rn$	0010nnnnmmmm1010	-	1	XOR R0,R1
XOR #imm.R0	Exclusive OR Logical	Rn. The contents of general register R0 can also be exclusive ORed with zero-extended 8-bit immediate data,	R0 ^ imm → R0	1100101011111111	-	1	XOR #H'F0,R0
XOR.B #imm,@(R0,GBR)		or 8-bit memory accessed by indirect indexed GBR addressing can be exclusive ORed with 8-bit immediate data.	$(R0 + GBR) \land imm \rightarrow (R0 + GBR)$	11001110iiiiiiii	-	3	XOR.B #H'A5,@(R0,GBR)
XTRCT Rm,Rn	Extract	Extracts the middle 32 bits from the 64 bits of coupled general registers Rm and Rn,	Rm: Center 32 bits of Rn → Rn	0010nnnnmmmm1101	-	1	XTRCT R0,R1
		and stores the 32 bits in Rn				-	

SH-DSP only							
Opcode	Instruction	Description	Function	Code	Delay Tbit Slot	Cycl	Example
.DC Rm,MOD	Load to Control Register	Store the source operand into control register	$Rm \rightarrow MOD$	0100mmmm01011110	-	1	
.DC Rm,RE	Load to Control Register	Store the source operand into control register	$Rm \rightarrow RE$	0100mmmm01111110	-	1	
.DC Rm,RS	Load to Control Register	Store the source operand into control register	$Rm \rightarrow RS$	0100mmmm01101110	-	1	
.DC.L @Rm+,MOD	Load to Control Register	Store the source operand into control register	$(Rm) \rightarrow MOD, Rm + 4 \rightarrow Rm$	0100mmmm01010111	-	3	
.DC.L @Rm+,RE	Load to Control Register	Store the source operand into control register	$(Rm) \rightarrow RE, Rm + 4 \rightarrow Rm$	0100mmmm01110111	-	3	
.DC.L @Rm+,RS	Load to Control Register	Store the source operand into control register	$(Rm) \rightarrow RS, Rm + 4 \rightarrow Rm$	0100mmmm01100111	-	3	
.DRE @(disp,PC)	Load Effective Address to RE Register	Stores the effective address of the source operand in the repeat end register RE. The effective address is an address specified by PC + displacement. use with SETRC	disp × 2 + PC → RE	10001110dddddddd	-	1	LDRE END
.DRS @(disp,PC)	Load Effective Address to RS Register	Stores the effective address of the source operand in the repeat start register RS. The effective address is an address specified by PC + displacement. use with SETRC	disp × 2 + PC→ RS	10001100dddddddd	-	1	LDRS STA
DS Rm,PR	Load to System Register		$Rm \rightarrow PR$	0100mmmm00101010	-	1	LDS R0,PR
.DS Rm,DSR	Load to System Register		$Rm \rightarrow DSR$	0100mmmm01101010	-	1	
DS Rm,A0	Load to System Register		$Rm \rightarrow A0$	0100mmmm01111010	-	1	
DS Rm,X0	Load to System Register		$Rm \rightarrow X0$	0100mmmm10001010	-	1	
DS Rm,X1	Load to System Register		$Rm \rightarrow X1$	0100mmmm10011010	-	1	
DS Rm,Y0	Load to System Register		Rm → Y0	0100mmmm10101010	-	1	
DS Rm,Y1	Load to System Register		Rm → Y1	0100mmmm10111010	-	1	
DS.L @Rm+,DSR	Load to System Register	Store the source operand into the system register MACH, MACL, or PR or the DSP	$(Rm) \rightarrow DSR,Rm + 4 \rightarrow Rm$	0100mmmm01100110	-	1	
DS.L @Rm+,A0	Load to System Register	register DSR, A0, X0, X1, Y0, or Y1. When A0 is designated as the destination, the	(Rm) → A0,Rm + 4 → Rm	0100mmmm01110110	-	1	
DS.L @Rm+,X0	Load to System Register	MSB of the data is copied into A0G.	(Rm) → X0,Rm+4 → Rm	0100nnnn10000110	-	1	
DS.L @Rm+,X1	Load to System Register		(Rm) → X1,Rm+4 → Rm	0100nnnn10010110	-	1	
.DS.L @Rm+,Y0	Load to System Register		(Rm) → Y0,Rm+4 → Rm	0100nnnn10100110	-	1	
DS.L @Rm+,Y1	Load to System Register		(Rm) → Y1, Rm+4 → Rm	0100nnnn10110110	-	1	
SETRC Rm SETRC #imm	Set Repeat Count to RC	Sets the repeat count to the SR register's RC counter. When the operand is a register, the bottom 12 bits are used as the repeat count. When the operand is an immediate data value, 8 bits are used as the repeat count. Set repeat control flags to RF1, RF0 bits of the SR register. Use of the SETRC instruction is subject to any	Rm[11:0] RCCSR[27:16] Repeat control flag \rightarrow RF1, RF0 imm \rightarrow RC [23:26] zeros \rightarrow SR[27:24],	0100mmmm00010100 10000010iiiiiii	-	1	SETRC #32
		Set repeat control liags to RF1, RF0 bits of the SR register. Use of the SE1RC instruction is subject to any limitations.	Repeat control flag → RF1, RF0		-	'	SETING #02
TC RE,Rn	Store Control Register	Stores control register into a specified destination.	$RE \rightarrow Rn$	0000nnnn01110010	-	1	
TC RS,Rn	Store Control Register	Stores control register into a specified destination.	$RS \rightarrow Rn$	0000nnnn01100010	-	1	
STC.L MOD,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn,MOD \rightarrow (Rn)$	0100nnnn01010011	-	2	
TC.L RE,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn, RE \rightarrow (Rn)$	0100nnnn01110011	-	2	
TC.L RS,@-Rn	Store Control Register	Stores control register into a specified destination.	$Rn - 4 \rightarrow Rn, RS \rightarrow (Rn)$	0100nnnn01100011	-	2	
TS DSR,Rn	Store System Register	Stores data from system register into a specified destination	DSR → Rn	0000nnnn01101010	-	1	
TS A0,Rn	Store System Register	Stores data from system register into a specified destination	A0 → Rn	0000nnnn01111010	-	1	
TS X0,Rn	Store System Register	Stores data from system register into a specified destination	X0→Rn	0000nnnn10001010	-	1	
TS X1,Rn	Store System Register	Stores data from system register into a specified destination	X1→Rn	0000nnnn10011010	-	1	
TS Y0,Rn	Store System Register	Stores data from system register into a specified destination	Y0→Rn	0000nnnn10101010	-	1	
TS Y1,Rn	Store System Register	Stores data from system register into a specified destination	Y1→Rn	0000nnnn10111010	-	1	
TS.L DSR,@-Rn	Store System Register	Stores data from system register into a specified destination	$Rn - 4 \rightarrow Rn, DSR \rightarrow (Rn)$	0100nnnn01100010	-	1	
TS.L A0,@–Rn	Store System Register	Stores data from system register into a specified destination	Rn – 4 → Rn, A0 → (Rn)	0100nnnn01100010	-	1	
TS.L X0,@-Rn	Store System Register	Stores data from system register into a specified destination	Rn-4→Rn,X0→(Rn)	0100nnnn10000010	-	1	
STS.L X1,@-Rn	Store System Register	Stores data from system register into a specified destination	Rn-4→Rn,X1→(Rn)	0100nnnn10010010	-	1	
TS.L Y0,@-Rn	Store System Register	Stores data from system register into a specified destination	Rn-4→Rn,Y0→(Rn)	0100nnnn10100010	-	1	
STS.L Y1,@-Rn	Store System Register	Stores data from system register into a specified destination	Rn–4→Rn,Y1→(Rn)	0100nnnn10110010	-	1	

Ctrl	Dec	Neg	Hex	Oct	Binary	Asc	Ebd	Dec	Neg	Hex	Oct	Binary	Asc	Ebd	Dec	Neg	Hex	Oct	Binary	Asc	Ebd	Dec	Neg	Hex	Oct	Binary	Asc	Ebd
NULL	0	0	00	0	00000000		NUL	32	-224	20	040	00100000		DS	64	-192	40	100	01000000	6	SP	96	###	60	140	01100000	•	-
SOH	1	-255	01	1	0000001	⊌	SOH	33	-223	21	041	00100001	•	sos	65	-191	41	101	01000001	A		97	###	61	141	01100001	a	/
STX	2	-254	02	2	00000010	8	STX	34	-222	22	042	00100010	••	FS	66	-190	42	102	01000010	В		98	###	62	142	01100010	Ъ	
ETX	3	-253	03	3	00000011	٠	ETX	35	-221	23	043	00100011	#	wus	67	-189	43	103	01000011	С		99	###	63	143	01100011	C	
EOT	4	-252	04	4	00000100	٠	SEL	36	-220	24	044	00100100	\$	BYP	68	-188	44	104	01000100	D		100	###	64	144	01100100	d	
ENQ	5	-251	05	5	00000101	÷	HT	37	-219	25	045	00100101	7.	LF	69	-187	45	105	01000101	E		101	###	65	145	01100101	е	
ACK	6	-250	06	6	00000110	÷	RNL	38	-218	26	046	00100110	æ	ETB	70	-186	46	106	01000110	F		102	###	66	146	01100110	£	
BEL	7	-249	07	7	00000111	٠	DEL	39	-217	27	047	00100111	•	ESC	71	-185	47	107	01000111	G		103	###	67	147	01100111	g	
BS	8	-248	80	10	00001000		GE	40	-216	28	050	00101000	(SA	72	-184	48	110	01001000	Н		104	###	68	150	01101000	h	
TAB	9	-247	09	11	00001001	0	SPS	41	-215	29	051	00101001)	SFE	73	-183	49	111	01001001	Ι		105	###	69	151	01101001	i	
LF	10	-246	0A	12	00001010	0	RPT	42	-214	2A	052	00101010	×	SM	74	-182	4A	112	01001010	J		106	###	6A	152	01101010	j	:
VT	11	-245	0B	13	00001011	ď	VT	43	-213	2B	053	00101011	+	CSP	75	-181	4B	113	01001011	K		107	###	6B	153	01101011	k	,
FF	12	-244	0C	14	00001100	Q	FF	44	-212	2C	054	00101100	,	MFA	76	-180	4C	114	01001100	L		108	###	6C	154	01101100	1	×
CR	13	-243	0D	15	00001101	F	CR	45	-211	2D	055	00101101	_	ENQ	77	-179	4D	115	01001101	М		109	###	6D	155	01101101	м	_
so	14	-242	0E	16	00001110	Ħ	SOH	46	-210	2E	056	00101110	•	ACK	78	-178	4E	116	01001110	Ν		110	###	6E	156	01101110	n	 >
SI	15	-241	0F	17	00001111	•	SI	47	-209	2F	057	00101111		BEL	79	-177	4F	117	01001111	0		111	###	6F	157	01101111	0	?
DLE	16	-240	10	20	00010000		DLE	48	-208	30	060	00110000	0		80	-176	50	120	01010000	P	&	112	###	70	160	01110000	P	
DC1	17	-239	11	21	00010001	-	DC1	49	-207	31	061	00110001	1		81	-175	51	121	01010001	Q		113	###	71	161	01110001	q	
DC2	18	-238	12	22	00010010	#	DC2	50	-206	32	062	00110010	2	SYN	82	-174	52	122	01010010			114	###	72	162	01110010	r	
DC3	19	-237	13	23	00010011	!!	DC3	51	-205	33	063	00110011	3	IR	83	-173	53	123	01010011	S		115	###	73	163	01110011	5	
DC4	20	-236	14	24	00010100	ЯI	RES	52	-204	34	064	00110100	4	PP	84	-172	54	124	01010100	T		116	###			01110100	-	
NAK	21	-235		25	00010101	8	NL	53	-203	35	065	00110101	5	TRN	85				01010101			117	###			01110101		
SYN	22	-234	16	26	00010110	-	BS	54	-202	36	066	00110110	6	NBS	86	-170	56	126	01010110	V		118	###	76	166	01110110	V	
ETB	23	-233		27	00010111	<u> </u>	POC	55	-201			00110111	7	EOT	87	-169			01010111	М		119	###			01110111		
CAN	24	-232		30	00011000	Ť	CAN	56	-200	38	070	00111000	8	SBS	88	-168			01011000			120	###			01111000		
EN	25	-231		31	00011001	Ŧ	EM	57	-199	39	071	00111001	9	IT	89	-167			01011001	Y	_	121	###	79	171	01111001	9	1
SUB	26	-230		32	00011010	→	UBS	58	-198			00111010	:	RFF	90	-166			01011010	z	•	122	###			01111010		:
ESC	27	-229		33	00011011		CU1	59	-197			00111011	÷	CU3	91				01011011	E	\$	123	###			01111011		#
DS	28	-228		34	00011100		IFS	60	-196			00111100	<	DC4	92				01011100		*	124	###			01111100		e
GS	29	-227	1D	35	00011101		IGS	61	-195			00111101	=	NAK	93				01011101])	125	###			01111101		,
RS	30	-226		36	00011110		IRS	62	-194			00111110	>		94	-162			01011110	^	;	126	###			01111110		=
US	31	-225	1F	37	00011111	•	IUS	63	-193	3F	077	00111111	?	SUB	95	-161	5F	137	01011111		¬	127	###	7F	177	01111111	Δ	"

١.	30	-226	TE	36	00011110	_	IRS	02	-194	35	076	00111110			94	-162	5E	136	01011110			120	###	/ E	1/6	01111110		
:	31	-225	1F	37	00011111	▼	IUS	63	-193	3F	077	00111111	?	SUB	95	-161	5F	137	01011111	_	-	127	###	7F	177	0111111	Δ	••
				-																								
	Dec	Neg	Hex	Oct	Binary	Asc	Ebd	Dec	Neg	He	Oct	Binary	Asc	Ebd	Dec	Neg	Hex	Oct	Binary	Asc	Ebd	Dec	Neg	Hex	Oct	Binary	Asc	Ebd
_	128	-128				_		160	_	_		10100000			192	-64	_		11000000	L	-{	224				11100000		$\overline{}$
	129	-127		201	10000001	ü	a	161	-95			10100001		~	193	-63			11000001	_	À	225				11100001		'
	130	-126	82	202	10000010	_	ь	162				10100010		5	194	-62			11000010		в	226				11100010	•	s
	131	-125		203	10000010	ā	C C	163				10100011	_	t	195	-61			11000010	Τ	Č.	227				11100011	-	T
	132	-124		204			ă	164				10100011	_	-	196	-60			11000111	_	Ď	228				11100100	Σ	ů
	133	-124		204	10000100	ā		165				10100100	==	u		-59			11000100	_	E	229				11100100	_	Ü
							e							V	197					t	F						_	-
	134	-122		206			f	166				10100110	_	W	198	-58			11000110	F	1 - 1	230				11100110	-	M
	135	-121		207	10000111	5	. g	167				10100111		×	199	-57			11000111	!!	G	231				11100111	_	×
	136	-120		210	10001000		h	168				10101000	_	y	200	-56			11001000	Ŀ	 	232				11101000	_	Y
	137	-119			10001001		1	169				10101001		z	201	-55			11001001	ĪĒ	I	233				11101001	Ξ	z
	138	-118	8A		10001010	e		170	-86			10101010			202	-54			11001010	11		234				11101010		
	139	-117	8B	213	10001011	ï		171	-85	AB	253	10101011	12		203	-53	CB	313	11001011	ŦĒ		235	-21	EB	353	11101011	δ	
	140	-116	8C	214	10001100	ĩ		172	-84	AC	254	10101100	1		204	-52	CC	314	11001100	li		236	-20	EC	354	11101100	œ	
	141	-115	8D	215	10001101	ī		173	-83	AD	255	10101101	i		205	-51	CD	315	11001101	=		237	-19	ED	355	11101101	ρĎ	
	142	-114	8E	216	10001110	Ä		174	-82	AE	256	10101110	~~		206	-50	CE	316	11001110	#		238	-18	EE	356	11101110	€	
	143	-113	8F	217	10001111	À	±	175	-81	AF	257	10101111	>>		207	-49	CF	317	11001111	=		239	-17	EF	357	11101111	Π	
	144	-112	90	220	10010000	Ē		176	-80	B0	260	10110000		^	208	-48	D0	320	11010000	щ)	240	-16	F0	360	11110000	≡	0
	145	-111	91	221	10010001	æ	J J	177	-79	B1	261	10110001	***		209	-47	D1	321	11010001	₹	J	241	-15	F1	361	11110001	±	1
	146	-110	92	222	10010010	Æ	k	178	-78	B2	262	10110010	25		210	-46	D2	322	11010010	т	K	242	-14	F2	362	11110010	<u>></u>	2
	147	-109	93	223	10010011	6	1	179	-77	в3	263	10110011	ī		211	-45	D3	323	11010011	Щ	L	243	-13	F3	363	11110011	≤	3
	148	-108	94	224	10010100	ö	м	180	-76	В4	264	10110100	-Á		212	-44	D4	324	11010100	E	M	244	-12	F4	364	11110100	ſ	4
	149	-107	95	225	10010101	ō	ln l	181	-75	В5	265	10110101	4		213	-43	D5	325	11010101	F	N	245	-11	F5	365	11110101	.i	5
	150	-106	96	226	10010110	ũ	0	182	-74	В6	266	10110110	41		214	-42	D6	326	11010110	п	0	246	-10	F6	366	11110110	÷	6
	151	-105	97	227	10010111	ū	P	183	-73	в7	267	10110111	70		215	-41	D7	327	11010111	#	P	247	-9	F7	367	11110111	æ	7
	152	-104	98	230	10011000	ÿ	9	184	-72	в8	270	10111000			216	-40	D8	330	11011000	#	Q	248	-8	F8	370	11111000	•	8
	153	-103	99	231	10011001		r	185	-71	В9	271	10111001			217	-39	р9	331	11011001	j	R	249	-7	F9	371	11111001		9
	154	-102	9A	232	10011010	ü	-	186	-70	BA	272	10111010	ii	Г	218	-38	DA	332	11011010	_		250	-6	FA	372	11111010	-	
	155	-101	9B	233	10011011	ē		187	-69	вв	273	10111011		ī	219	-37	DB	333	11011011	Ė		251	-5	FB	373	11111011	J	
	156	-100	9C	234		-		188				10111100	•••	-	220	-36			11011100	=		252	-4			11111100	ň	
	157	-99	9D		10011101			189				10111101			221	-35			11011101	T		253	-3			11111101	2	
	158	-98	9E			•		190				10111110	_		222	-34			11011110	-		254	-2			11111110		
	159	-97	9F		10011111	æ		191				10111111	-		223	-33			11011111	_		255				11111111		ΕO
L		7,	JE	_5,				171	00	JE			1_		223	33	DE	-5,				233						

Bases 2

Dec	Neg	Hex	Oct	Binary
0	0	0000	000000	00000000 00000000
1	-65535	0001	000001	00000000 00000001
2	-65534	0002	000002	00000000 00000010
4	-65532	0004	000004	00000000 00000100
8	-65528	8000	000010	00000000 00001000
16	-65520	0010	000020	00000000 00010000
32	-65504	0020	000040	00000000 00100000
64	-65472	0040	000100	00000000 01000000
128	-65408	0800	000200	00000000 10000000
192	-65344	00C0	000300	00000000 11000000
256	-65280	0100	000400	0000001 00000000
320	-65216	0140	000500	00000001 01000000
384	-65152	0180	000600	00000001 10000000
448	-65088	01C0	000700	00000001 11000000
512	-65024	0200	001000	00000010 00000000
576	-64960	0240	001100	00000010 01000000
640	-64896	0280	001200	00000010 10000000
704	-64832	02C0	001300	00000010 11000000
768	-64768	0300	001400	00000011 00000000
832	-64704	0340	001500	00000011 01000000
896	-64640	0380	001600	00000011 10000000
960	-64576	03C0	001700	00000011 11000000
1,024	-64512	0400	002000	00000100 00000000
1,088	-64448	0440	002100	00000100 01000000
1,152	-64384	0480	002200	00000100 10000000
1,216	-64320	04C0	002300	00000100 11000000
1,280	-64256	0500	002400	00000101 00000000
1,344	-64192	0540	002500	00000101 01000000
1,408	-64128	0580	002600	00000101 10000000
1,472	-64064	05C0	002700	00000101 11000000
1,536	-64000	0600	003000	00000110 00000000
1,600	-63936	0640	003100	00000110 01000000
1,664	-63872	0680	003200	00000110 10000000
1,728	-63808	06C0	003300	00000110 11000000
1,792	-63744	0700	003400	00000111 00000000
1,856	-63680	0740	003500	00000111 01000000
1,920	-63616	0780	003600	00000111 10000000
1,984	-63552	07C0	003700	00000111 11000000
2,048	-63488	0800	004000	00001000 00000000

Dec	Neg	Hex	Oct	Binary
0	0	0000	000000	00000000 00000000
2,048	-63488	0800	004000	00001000 00000000
4,096	-61440	1000	010000	00010000 00000000
6,144	-59392	1800	014000	00011000 00000000
8,192	-57344	2000	020000	00100000 00000000
10,240	-55296	2800	024000	00101000 00000000
12,288	-53248	3000	030000	00110000 00000000
14,336	-51200	3800	034000	00111000 00000000
16,384	-49152	4000	040000	01000000 00000000
18,432	-47104	4800	044000	01001000 00000000
20,480	-45056	5000	050000	01010000 00000000
22,528	-43008	5800	054000	01011000 00000000
24,576	-40960	6000	060000	01100000 00000000
26,624	-38912	6800	064000	01101000 00000000
28,672	-36864	7000	070000	01110000 00000000
30,720	-34816	7800	074000	01111000 00000000
32,768	-32768	8000	100000	10000000 00000000
34,816	-30720	8800	104000	10001000 00000000
36,864	-28672	9000	110000	10010000 00000000
38,912	-26624	9800	114000	10011000 00000000
40,960	-24576	A000	120000	10100000 00000000
43,008	-22528	A800	124000	10101000 00000000
45,056	-20480	B000	130000	10110000 00000000
47,104	-18432	B800	134000	10111000 00000000
49,152	-16384	C000	140000	11000000 00000000
51,200	-14336	C800	144000	11001000 00000000
53,248	-12288	D000	150000	11010000 00000000
55,296	-10240	D800	154000	11011000 00000000
57,344	-8192	E000	160000	11100000 00000000
59,392	-6144	E800	164000	11101000 00000000
61,440	-4096	F000	170000	11110000 00000000
63,488	-2048	F800	174000	11111000 00000000
65,535	-1	FFFF	177777	11111111 11111111

16,777,215	FFFFFF	7777777	24 Bit
4,294,967,295	FFFFFFF	3.7778E+10	32 Bit

Colors & Resolutions

Screen	Bytes	Hex	Size (K)
256x192 256 color	49,152	C000	48K
256x192 16 color	24,576	6000	24K
256x192 8 color	18,432	4800	18K
256x192 4 color	12,288	3000	12K
256x192 2 color	6,144	1800	6K

Screen	Bytes	Hex	Size (K)
320x200 256 color	64,000	FA00	64K
320x200 16 color	32,000	7D00	32K
320x200 8 color	24,000	5DC0	24K
320x200 4 color	16,000	3E80	16K
320x200 2 color	8,000	1F40	8K

Screen	Bytes	Hex	Size (K)
32x32 Word Tiles	2.048	0800	2K
32x24 Word Tiles	1.536	0600	1.5K
32x32 Byte Tiles	1.024	0400	1K
32x24 Byte Tiles	768	0300	0.7K

Tile/Sprite	Bytes	Hex
8x8 2 color	8	08
8x8 4 color	16	10
8x8 8 color	24	18
8x8 16 color	32	20
8x8 256 color	64	40
16x16 2 color	32	20
16x16 4 color	64	40
16x16 8 color	96	60
16x16 16 color	128	80
16x16 256 color	256	100

YUV Formulas
Y = ((66*R + 129*G + 25*B + 128) >>8) + 16
U = ((-38*R - 74*G + 112*B + 128) >>8) + 128
V = ((112*R - 94*G - 18*B + 128) >>8) + 128
c = Y-16
d = U-128
e = V-128
R = (298*c + 409*e + 128) >> 8 , Limit 0-255
G = (298*c - 100*d - 208*e + 128) >>8 , Limit 0-255
B = (298*c + 516*d + 128)>>8 , Limit 0-255

Common Color combinations

Color	-RGB	
Red	-F00	
Green	-0F0	
Blue	-00F	
	·	
Cyan	-FF0	
Magenta	-F0F	
Yellow	-FF0	
		•
Black	-000	
Grey	-888	
White	-FFF	
Orange	-F80	
Pink	-F88	
Lilac	-88F	
Sky Blue	-08F	

-80F

Purple

Systems such as the ZX Spectrum, Camputers Lynx Fujitsu FM-7 and Sinclair QL use a palette based on Combinations of 3 primary color channel bitplanes:

Color	Number	-GRB	
Black	0	-000	
Blue	1	-001	
Red	2	-010	
Magenta	3	-011	
Green	4	-100	
Cyan	5	-101	
Yellow	6	-110	
White	7	-111	

Trigonometry

Deg	Rac	lians	SIN	cos	TAN
0	0	0.000	0.000	1.000	0.000
15		0.262	0.259	0.966	0.268
30	π/6	0.524	0.500	0.866	0.577
45	π/4	0.785	0.707	0.707	1.000
60	π/3	1.047	0.866	0.500	1.732
75		1.309	0.966	0.259	3.732
90	π/2	1.571	1.000	0.000	###
105		1.833	0.966	-0.259	-3.732
120	2π/3	2.094	0.866	-0.500	-1.732
135	3π/4	2.356	0.707	-0.707	-1.000
150	5π/6	2.618	0.500	-0.866	-0.577
165		2.880	0.259	-0.966	-0.268
180	π	3.142	0.000	-1.000	0.000
195		3.403	-0.259	-0.966	0.268
210		3.665	-0.500	-0.866	0.577
225		3.927	-0.707	-0.707	1.000
240		4.189	-0.866	-0.500	1.732
255		4.451	-0.966	-0.259	3.732
270	3π/2	4.712	-1.000	0.000	###
285		4.974	-0.966	0.259	-3.732
300		5.236	-0.866	0.500	-1.732
315		5.498	-0.707	0.707	-1.000
330		5.760	-0.500	0.866	-0.577
345		6.021	-0.259	0.966	-0.268
360	2π	6.283	0.000	1.000	0.000
375		6.5450	0.2588	0.9659	0.2679
390		6.8068	0.5000	0.8660	0.5774

Value	ASIN	ACOS	ATAN	CSC	SEC	СОТ
0.000	0.000	1.571	0.000	#NUM!	1.000	#NUM!
0.259	0.262	1.309	0.253	3.864	1.035	3.732
0.500	0.524	1.047	0.464	2.000	1.155	1.732
0.707	0.785	0.785	0.615	1.414	1.414	1.000
0.866	1.047	0.524	0.714	1.155	2.000	0.577
0.966	1.309	0.262	0.768	1.035	3.864	0.268
1.000	1.571	0.000	0.785	1.000	###	0.000
0.966	1.309	0.262	0.768	1.035	-3.864	-0.268
0.866	1.047	0.524	0.714	1.155	-2.000	-0.577
0.707	0.785	0.785	0.615	1.414	-1.414	-1.000
0.500	0.524	1.047	0.464	2.000	-1.155	-1.732
0.259	0.262	1.309	0.253	3.864	-1.035	-3.732
0.000	0.000	1.571	0.000	###	-1.000	###
-0.259	-0.262	1.833	-0.253	-3.864	-1.035	3.732
-0.500	-0.524	2.094	-0.464	-2.000	-1.155	1.732
-0.707	-0.785	2.356	-0.615	-1.414	-1.414	1.000
-0.866	-1.047	2.618	-0.714	-1.155	-2.000	0.577
-0.966	-1.309	2.880	-0.768	-1.035	-3.864	0.268
-1.000	-1.571	3.142	-0.785	-1.000	###	0.000
-0.966	-1.309	2.880	-0.768	-1.035	3.864	-0.268
-0.866	-1.047	2.618	-0.714	-1.155	2.000	-0.577
-0.707	-0.785	2.356	-0.615	-1.414	1.414	-1.000
-0.500	-0.524	2.094	-0.464	-2.000	1.155	-1.732
-0.259	-0.262	1.833	-0.253	-3.864	1.035	-3.732
0.000	0.000	1.571	0.000	###	1.000	###
0.2588	0.2618	1.3090	0.2533			

Byte Byte Rad Sin 0 0 11 33 21 64 32 90 43 110 53 123 64 127 75 123 85 110 96 90 107 64
11 33 21 64 32 90 43 110 53 123 64 127 75 123 85 110 96 90 107 64
21 64 32 90 43 110 53 123 64 127 75 123 85 110 96 90 107 64
32 90 43 110 53 123 64 127 75 123 85 110 96 90 107 64
43 110 53 123 64 127 75 123 85 110 96 90 107 64
53 123 64 127 75 123 85 110 96 90 107 64
53 123 64 127 75 123 85 110 96 90 107 64
75 123 85 110 96 90 107 64
85 110 96 90 107 64
96 90 107 64
107 64
117 33
128 (
139 -33
149 -64
160 -90
171 -110
181 -123
192 -127
203 -123
213 -110
224 -90
235 -64
245 -33
256

390 6.8068 0.5000 0.8660 0.5774 0.5000 0.5236 1.0472 0.4636

Csc(X) = 1/Sin(X)

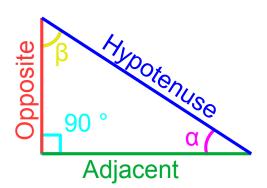
Sec (X) = 1/Cos(X)

degrees = radians × 180° / π radians = degrees × π / 180°

Cos (X) = Sin ($X + 90^{\circ}$)

Sin (X) = Cos ($X - 90^{\circ}$)

90 Degrees = $\pi/2$ rad



Cot (X) = Cos (X) / Sin (X)

Small Angle Approximation

Cot (X) = 1/Tan(X)

Small Angle Approximations						
Sin a ~= a						
Cos a ~= 1 - (a * a) / 2 ~= 1						
Tan a ~= a						

Trigonometry $Sin(\alpha) = Opposite / Hypotenuse$ Trigonometry $Cos(\alpha) = Adjacent / Hypotenuse$ Trigonometry $Tan(\alpha) = Opposite / Adjacent$ Trigonometry $Csc(\alpha) = Hypotenuse / Opposite$ Trigonometry $Sec(\alpha) = Hypotenuse / Adjacent$ Trigonometry $Cot(\alpha) = Adjacent / Opposite$

180 Rule $90 + \alpha + \beta = 180^{\circ}$ Pythagoras $H^2 = A^2 + O^2$

	Adj =	Opp =	Hyp =	$\alpha =$
Trigonometry	$Cos(\alpha)*H$	Sin (α) * H	O / Sin (α)	ATan(O / A)
Trigonometry	<mark>Ο</mark> / Tan (α)	Tan (α) * A	A / Cos (α)	ACos (A/H)
Trigonometry				ASin (0 / H)
Pythagoras	$\sqrt{(H^2-O^2)}$	$\sqrt{(\mathbf{H^2} - \mathbf{A}^2)}$	$\sqrt{(\mathbf{A}^2 + \mathbf{O}^2)}$	
180 Rule				180 - (90 + <i>\beta</i>)

-	ъ.	•	OTAL	000	TAN		\/ I	AOTH	4000	ATAN	000	050	OOT			'VO'
Deg		ians	SIN	COS	TAN		Value 0.0000	ASIN 0.0000	ACOS 1.5708	0.0000	#NUM!	SEC 1.0000	COT #NUM		HexRad HE	XSIN 0
15	0	0.0000 0.2618	0.0000 0.2588	1.0000 0.9659	0.0000 0.2679		0.0000	0.0000	1.3090	0.0000	3.8637	1.0353	3.7321		11	33
30	π/6	0.5236	0.5000	0.8660	0.2073		0.5000	0.5236	1.0472	0.4636	2.0000	1.1547	1.7321	1 1	21	64
45	π/4	0.7854	0.7071	0.7071	1.0000		0.7071	0.7854	0.7854	0.4050	1.4142	1.4142	1.0000	1	32	90
60	π/3	1.0472	0.8660	0.5000	1.7321		0.8660	1.0472	0.5236	0.7137	1.1547	2.0000	0.5774	→	43	110
75	0	1.3090	0.9659	0.2588	3.7321		0.9659	1.3090	0.2618	0.7681	1.0353	3.8637	0.2679		53	123
90	π/2	1.5708	1.0000	0.0000	###		1.0000	1.5708	0.0000	0.7854	1.0000	###	0.0000	⊣ ⊦	64	127
105		1.8326	0.9659	-0.2588	-3.7321		0.9659	1.3090	0.2618	0.7681	1.0353	-3.8637	-0.2679	1 1	75	123
120	2π/3	2.0944	0.8660	-0.5000	-1.7321		0.8660	1.0472	0.5236	0.7137	1.1547	-2.0000	-0.5774	1	85	110
135	3π/4	2.3562	0.7071	-0.7071	-1.0000		0.7071	0.7854	0.7854	0.6155	1.4142	-1.4142	-1.0000		96	90
150	5π/6	2.6180	0.5000	-0.8660	-0.5774		0.5000	0.5236	1.0472	0.4636	2.0000	-1.1547	-1.7321	1	107	64
165		2.8798	0.2588	-0.9659	-0.2679		0.2588	0.2618	1.3090	0.2533	3.8637	-1.0353	-3.7321	1 1	117	33
180	π	3.1416	0.0000	-1.0000	-0.0000		0.0000	0.0000	1.5708	0.0000	###	-1.0000	###	<u> </u>	128	0
195		3.4034	-0.2588	-0.9659	0.2679		-0.2588	-0.2618	1.8326	-0.2533	-3.8637	-1.0353	3.7321	1 1	139	-33
210		3.6652	-0.5000	-0.8660	0.5774		-0.5000	-0.5236	2.0944	-0.4636	-2.0000	-1.1547	1.7321	1 1	149	-64
225		3.9270	-0.7071	-0.7071	1.0000		-0.7071	-0.7854	2.3562	-0.6155	-1.4142	-1.4142	1.0000	1 1	160	-90
240		4.1888	-0.8660	-0.5000	1.7321		-0.8660	-1.0472	2.6180	-0.7137	-1.1547	-2.0000	0.5774	⊣ ⊦	171	-110
255		4.4506	-0.9659	-0.2588	3.7321		-0.9659	-1.3090	2.8798	-0.7681	-1.0353	-3.8637	0.2679	⊣ ⊦	181	-123
270	3π/2	4.7124	-1.0000	-0.0000	###		-1.0000	-1.5708	3.1416	-0.7854	-1.0000	###	0.0000		192	-127
285	02	4.9742	-0.9659	0.2588	-3.7321		-0.9659	-1.3090	2.8798	-0.7681	-1.0353	3.8637	-0.2679	-	203	-123
300		5.2360	-0.8660	0.5000	-1.7321		-0.8660	-1.0472	2.6180	-0.7137	-1.1547	2.0000	-0.5774	⊣ ⊦	213	-110
315		5.4978	-0.7071	0.7071	-1.0000		-0.7071	-0.7854	2.3562	-0.6155	-1.4142	1.4142	-1.0000	⊣ ⊦	224	-90
330		5.7596	-0.5000	0.8660	-0.5774		-0.5000	-0.5236	2.0944	-0.4636	-2.0000	1.1547	-1.7321	1	235	-64
345		6.0214	-0.2588	0.9659	-0.2679		-0.2588	-0.2618	1.8326	-0.2533	-3.8637	1.0353	-3.7321	1	245	-33
360	2π	6.2832	-0.2366	1.0000	-0.2079		-0.2366	-0.2018	1.5708	-0.2555	-3.003 <i>1</i> ###	1.0000	-3.7321 ###	<u>.</u>	256	-0
	<u> </u>										11111	1.0000	****	ן נ	200	-0
375		6.5450	0.2588	0.9659	0.2679		0.2588	0.2618	1.3090	0.2533						
390	ALA BER	6.8068		0.8660	0.5774	M :	0.5000	0.5236	1.0472	0.4636	A.A. + 88 -	a.n		Consti	alo Ama	
	ixA * Mat					Matrix B			1		ixA + Mat			Small Ar	igle Approxii	mations
A Rows	s Must =	R Cols		A	В	С	D	E			Be Same				Sin a ~= a	
_				F	G	Н	I	J		Α	В	С		Cos a ~	-= 1 - (a * a) /	2 ~= 1
	٨	K	L	KA+LF	KB+LG	KC+LH	KD+LI	KE+LJ		D	E	F			Tan a ~= a	
	Matrix A	М	N	MA+NF	MB+NG	MC+NH	MD+NI	ME+NJ		G	Н	I				
	atr	0	Р	OA+PF	OB+PG	OC+PH	OD+PI	0E+PJ	· '		+			Cos(X) =	Sin(X+ 90deg	rees)
	Ž	Q	R	QA+RF	QB+RG	QC+RH	QD+RI	QE+RJ		J	K	L			radians × 18	
•		· ·	- 11	Q/X-IXI	QD-ITG	QO-IIII	QD TU	QL-110	1	M	N	ō		-	degrees × π	
		Х	Υ	Z	W					P	Q	R		raulalis –	uegrees ^ ii	/ 100
ø	v		0	0	0		V - A	_	Į.	Р	=	К		000 - 4/0	NINI (A)	
Space	X	1					X = Across	5	1	A . I		0.1		CSC = 1/S		
ري ا	Y	0	1	0	0		Y = Up			A+J	B+K	C+L		SEC = 1/C		
3D	Z	0	0	1	0		Z = In			D+M	E+N	F+O		COT = 1/T	AN (A)	
	W	0	0	0	1		W = Trans	formation		G+P	H+Q	I+R				
											_					
4) 7 3		Х	Y	Z			Х	Y	Z			X	Υ	Z		
Rotate around X	Х	1	0	0	Rotate around Y	X	cos (A)	0	-sin (A)	Rotate around Z	Х	cos (A)	sin (A)	0		
^ ig st	Υ	0	cos (A)	sin (A)	3 of	Υ	0	1	0	30 3d	Υ	-sin (A)	cos (A)	0		
_ 10	Z	0	-sin (A)	cos (A)	_ 10	Z	sin (A)	0	cos (A)	10	Z	0	0	1		
	X2 = X					X2 = Z *	sin (A) + X	(* cos (A)	, i	X2 = X * c	cos (A) -	Y * sin (<i>A</i>	A)		
,	Y2 = Y *	cos (A) -	7 * sin (A)		Y2 = Y					Y2 = X * s	sin (A) + Y	* cos (A	A)		
-		sin (A) + Z		•			cos (A) -)	X * sin (A)		Z2 = Z		(.	7		
		/	000 (/ (•/			,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	(· OIII (/ (,							
Opposite	R	Mr.			Trigon	metry	Sin (a) =	Onnosite	/ Hypote	nuse	180 Rule		90 + W +	⊦ <u>β</u> = 180°		
: <u>:</u>		VP _O) x		Trigon	-	$Cos(\alpha)$		• • •	aoo	.co .ta.o		. u	, 100		
\sim	l		(O)		Trigon	-				anuca						
ŏ			> // ₀		ingon			Onnocite			Pythagor:	ie.	H2 = A2 +	O ²		
d				_	Trigon	-			o / Adjace	ent	Pythagora	ıs	H ² = A ² +	O ²		
		0	1.0	9	Trigon	ometry	Csc (a) =	= Hypoten	e / Adjace luse / Opp	ent oosite				O ²		
	90	0	tenus	9	Trigon	ometry ometry	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha)$	= Hypoten = Hypoten	e / Adjace luse / Opp luse / Adj	ent oosite acent	Cos (X) =	Sin (X + 9	90°)	O ²		
O	– 90	0	α	9	_	ometry ometry	Csc (a) =	= Hypoten = Hypoten	e / Adjace luse / Opp luse / Adj	ent oosite acent ite	Cos (X) = Sin (X) = (Sin (X + 9 Cos (X - 9	90°) 10°)	O^2		
O			α	2	Trigon	ometry ometry	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha)$	= Hypoten = Hypoten	e / Adjace luse / Opp luse / Adj	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X)	90°) 10°)	O ²		
O		Adjace	α	2	Trigon	ometry ometry	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha)$	= Hypoten = Hypoten	e / Adjace luse / Opp luse / Adj	ent posite acent ite	Cos (X) = Sin (X) = CSc (X) = Sec (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X)	90°) 10°)	O ²		
O		Adjace	a ent	<u> </u>	Trigono Trigono	ometry ometry ometry	$Csc(\alpha) = Sec(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha)$	= Hypoten = Hypoten = Adjacent	e / Adjace nuse / Opp nuse / Adj t / Opposi	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X)	90°) 90°)	O²		
	A	Adjace	α ent j=	Op	Trigono Trigono	ometry ometry ometry Hy	$Csc(\alpha) = Sec(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha)$	= Hypoten = Hypoten = Adjacent	e / Adjace nuse / Opp nuse / Adj t / Opposi	ent posite acent ite	Cos (X) = Sin (X) = CSc (X) = Sec (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X)	90°) 90°)	O^2		
Trigono	A cometry	Adjace Ad Cos (α ent j = α) * H	Op Sin (d	Trigono Trigono p = // * H	ometry ometry ometry Hy	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha) = $ $Cot(\alpha) = $ $p = $ $in(\alpha)$	= Hypoten = Hypoten = Adjacent	e / Adjace nuse / Opp nuse / Adj t / Opposi	ent losite acent ite	Cos (X) = Sin (X) = CSc (X) = Sec (X) = Cot (X	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9	90°) 90°) Sin (X)	O^2		
Trigono Trigono	cometry cometry	Adjace	α ent j = α) * H	Op	Trigono Trigono p = // * H	ometry ometry ometry Hy	$Csc(\alpha) = Sec(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha) = Cot(\alpha)$	Hypoten Hypoten Adjacent ATan (ACos (e / Adjace nuse / Opp nuse / Adj t / Opposi	ent losite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9	90°) 90°) Sin (X)	O^2		
Trigono Trigono Trigono	ometry ometry ometry	Adjace Cos (CO) / Ta	ent = \alpha * H \alpha (\alpha)	Op Sin (d Tan (d	Trigono Trigono P = () * H (x) * A	ometry ometry ometry of the control	$Csc(\alpha) = Sec(\alpha) = S$	= Hypoten = Hypoten = Adjacent	e / Adjace nuse / Opp nuse / Adj t / Opposi	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Cot (X) = Cot (X) = Sec (Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = π/2 ra	90°) 10°) Sin (X)	O^2		
Trigono Trigono Trigono Pythaç	ometry ometry ometry ometry goras	Adjace Ad Cos (ent = \alpha * H \alpha (\alpha)	Op Sin (d	Trigono Trigono P = () * H (x) * A	ometry ometry ometry of the control	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha) = $ $Cot(\alpha) = $ $p = $ $in(\alpha)$	ATan (ACos (ASin (e / Adjace nuse / Opp nuse / Adjace t / Opposi = 0 / A) A / H) O / H)	ent posite acent ite	Cos (X) = Sin (X) = Cos (X) = Sec (X) = Cot (X) = Cot (X) = Ot (X) = Cot (X)	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = $\pi/2$ ration	90°) 10°) Sin (X) ad	O^2		
Trigono Trigono Trigono	ometry ometry ometry ometry goras	Adjace Cos (CO) / Ta	ent = \alpha * H \alpha (\alpha)	Op Sin (d Tan (d	Trigono Trigono P = () * H (x) * A	ometry ometry ometry of the control	$Csc(\alpha) = Sec(\alpha) = S$	Hypoten Hypoten Adjacent ATan (ACos (e / Adjace nuse / Opp nuse / Adjace t / Opposi = 0 / A) A / H) O / H)	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Cot (X) = Cot (X) = Sec (Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = $\pi/2$ ration	90°) 10°) Sin (X) ad	O^2		
Trigono Trigono Trigono Pythaç	ometry ometry ometry ometry goras	Adjace Cos (CO) / Ta	ent $ \begin{vmatrix} \alpha \\ \beta \\ \beta \\ \alpha \end{vmatrix} * H $ $ an(\alpha) $ $ -0^{2} $	Op Sin (∂ Tan (∂ √(H²	Trigono Trigono P = () * H (x) * A	ometry ometry ometry of the control	$Csc(\alpha) = Sec(\alpha) = Sec(\alpha) = Cot(\alpha) = Sec(\alpha) = S$	ATan (ACos (ASin (e / Adjace nuse / Opp nuse / Adjace t / Opposi = 0 / A) A / H) O / H)	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Cot (X) = 90 Degree Radians = Radians = Sin (X) = Sin (Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = π/2 ra rad × 180 deg × π /	90°) 10°) Sin (X) ad	O^2		
Trigono Trigono Trigono Pythaç	ometry ometry ometry goras Rule	Adjace Cos (CO) / Ta	α ent	Op Sin (α Tan (α √(H²	Trigono Trigono P = () * H (x) * A	Hy O / S A / Co	Csc (α) = Sec (α) = Cot (α) =	ATan (ACos (ASin (180 - (tch β	e / Adjace use / Opp use / Adj t / Opposi	ent posite acent ite	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Degree Radians = Z Y	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 181 er deg × π /	90°) 10°) Sin (X) ad 0 / π / 180	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (CO) / Ta	α ent	Sin (α Tan (α $\sqrt{\frac{1}{2}}$ Roll α	Trigono Trigono P = () * H (x) * A	ometry ometry ometry O / S A / Cc √(A^2	Csc (α) = Sec (α) = Cot (α) =	Hypotene Hypotene Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz	= / Adjace suse / Opposite / Opp	ent posite acent ite cos(rz)*s	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Cot (X) = Pot (Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 181 ed deg × π / aw γ rx) + sin(rz rx) + sin(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (CO) / Ta	α ent	Op Sin (α Tan (α √(H²	Trigono Trigono P = () * H (x) * A	ometry ometry ometry O / S A / Cc √(A^2	Csc (α) = Sec (α) = Cot (α) =	Hypotene Hypotene Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz	= / Adjace suse / Opposite / Opp	ent posite acent ite cos(rz)*s	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Degree Radians = Z Y	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 181 deg × π / aw γ rx) + sin(ra rx) + sin(ra	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (CO) / Ta	α (α) = (α) * H = (α) * H = (α) = (α) = (α) X F cos(rz) * (sin(rz) * (sin(rz	Sin (α Tan (α $\sqrt{\frac{1}{2}}$ Roll α	Trigono Trigono P = () * H (x) * A	ometry ometry ometry O / S A / Cc √(A^2	Csc (α) = Sec (α) = Cot (α) =	Hypoten Hypoten Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz x) + cos(rz x) + cos(rz	= / Adjace suse / Opposite / Opp	ent posite acent ite cos(rz)*s	Cos (X) = Sin (X) = (Csc (X) = Sec (X) = Cot (X) = Cot (X) = Cot (X) = Pot (Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 18 ed deg × π / π / rx) + sin(rz rx) - cos(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythaç	ometry ometry ometry goras Rule	Adjace Cos (CO) / Ta	α (α) Ent (i) = (α) * H (α) (α) - (α) X F cos(rz) * (sin(rz) * (si	Sin (α Tan (α * cos(ry) * cos(ry) * (ry)	Trigono Trigono () * H () * A - A ²)	Hy O / S A / Co	Csc (α) = Sec (α) = Cot (α) = P = in (α) in (α) in (α) in (γ)*sin(α) cos(γ)*	Hypoten Hypoten Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz x) + cos(rz x) + cos(rz	= / Adjace suse / Opposite / Opp	ent posite acent ite cos(rz)*s	Cos (X) = Sin (X) = CSin (X) = Sin (X) = Sin (X) = Sin (Y) * COs (X) *	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 18 ed deg × π / π / rx) + sin(rz rx) - cos(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace	α (cosz*cos	Sin (α Tan (α √(H* cos(ry) cos(ry) (ry) y) + y1* (α	Trigono Trigono Trigono y) * H y) * A - A²)	Hypometry O / S A / Cc \[\sqrt{A} \] \[\cos(rz)^* \sin(rz)^* \sin(rz)^* \]	Csc (α) = Sec (α) = Cot (α) = P = in (α) in (α) in (α) in (γ)*sin(α) cos(γ)*	Hypoten Hypoten Adjacent ATan (ACos (ASin (= / Adjace suse / Opposite / Opp	cos(rz)*sin(rz)*si	Cos (X) = Sin (X) = CSin (Y)*COS(IN (Y))*COS(IN (Y))*COS(IN (Y))*COS(IN (Y))*CSIN (Y) * CSIN (Y) *	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 18 ed deg × π / π / rx) + sin(rz rx) - cos(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (0 O / Ta √(H ² x2 = x1* (y2	α ent j = α * H α α α α α α α α α	Op Sin (0 Tan (0 √(H*) * cos(ry) * cos(ry) *(ry) *y) + y1* ((/*sinx-sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = in (α) in (α) in (α) in (γ)*sin(γ) cos(γ) **sin(γ)*sin(γ) **sin(γ)*sin(γ) **sin(γ)*sin(γ)	Hypoten Hypoten Hypoten ATan (ACos (ASin (= / Adjace suse / Opposite / Opp	cos(rz)*sin(rz)*sin(cosy*sin.	Cos (X) = Sin (X) = CSin (Y)*cos(n(Y)*cos(n(Y)*cos(ros(Y))*x)	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 18 ed deg × π / π / rx) + sin(rz rx) - cos(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (0 O / Ta √(H ² x2 = x1* (y2	α ent j = α * H α α α α α α α α α	Op Sin (0 Tan (0 √(H*) * cos(ry) * cos(ry) *(ry) *y) + y1* ((/*sinx-sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = P = in (α) i	Hypoten Hypoten Hypoten ATan (ACos (ASin (= / Adjace suse / Opposite / Opp	cos(rz)*sin(rz)*sin(cosy*sin.	Cos (X) = Sin (X) = CSin (Y)*cos(n(Y)*cos(n(Y)*cos(ros(Y))*x)	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 5 es = π /2 ra er rad × 18 ed deg × π / π / rx) + sin(rz rx) - cos(rz	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Trigono Trigono Trigono Pythao 180 F	ometry ometry ometry goras Rule	Adjace Cos (0 O / Ta √(H ² x2 = x1* (y2	α ent j = α * H α α α α α α α α α	Op Sin (0 Tan (0 √(H*) * cos(ry) * cos(ry) *(ry) *y) + y1* ((/*sinx-sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = in (α) in (α) in (α) in (γ)*sin(γ) cos(γ) **sin(γ)*sin(γ) **sin(γ)*sin(γ) **sin(γ)*sin(γ)	Hypoten Hypoten Hypoten ATan (ACos (ASin (= / Adjace suse / Opposite / Opp	cos(rz)*sin(rz)*sin(cosy*sin.	Cos (X) = Sin (X) = CSin (Y)*COS(IN (Y)*COS(Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Sin (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = π/2 rad × 18t e deg × π / aw γ (x) + sin(ra x) - cos(rz)	90°) 10°) Sin (X) ad 0 / π / 180 z)*sin(rx)	Į		
Rotate around AXX X X X X X X	ometry ometry ometry goras Rule	Adjace Ad Cos (0 O / Ta √ (H') x2 = x1* (y2 = x1* (z2 = x1* (α (x) + H (α) + H (α) - 0°) X F (α) - 0° (α)	Sin (a) Tan (a) Tan (b) √ (H² cos(ry) cos(ry) y) + y1* (y*sinx—sin y*cosx+sir	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = P = in (α) cs (α) Y Pit sin(ry)*sin(ry) cos(ry) * ny *siny*sinx *siny*cos	Hypoten Hypoten Hypoten ATan (ACos (ASin (180 - (tch \(\beta\) Fix) - sin(rz) * sin(rx) +cosz*co x-cosz*sin X	= / Adjace suse / Opposite / Opp	cos(rz)*s cos(rz)*s sin(rz)*si (cosy*sin. (cosy*cosy*cosy*cosy*cosy*cosy*cosy*cosy*	Cos (X) = Sin (X) = CSin (Y)*COS(IN (Y)*COS(Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 es = π/2 ra er rad × 180 ed eg × π / aw γ rx) + sin(ra x) - cos(ra cos(ra)	90°) 90°) Sin (X) ad 0 / π 180 2)*sin(rx)		uning in Resis	
Rotate around AXX X X X X X X	ometry ometry ometry goras Rule	Adjace Adjace Cos (0 O Ta V H' x2 = x1* (1 y2 = x1* (2 x2 = x1* (3 x2 = x1* (4 x2 = x1*	α ent j = α * H α * H α α α - 0¹ X F cos(rz) * sin(rz) * -sin(rz) * (cosz*cos * (cosz*sin) * (cosz*sin) * α α α α α α α α α α	Sin (a) Tan (a) √(H* cos(ry) cos(ry) y) + y1* (y*sinx-sin z 0	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = Fraction (α) = P = in (α)	ATan (ACos (ASin (180 - (= / Adjace suse / Oppose = O / A) t / Oppose - O / A) A / H) O / H) - O /	cos(rz)*s cos(rz)*s sin(rz)*si (cosy*sin. (cosy*cos	Cos (X) = Sin (X) = Cos (X) = Sin (X) = Cos (X) = Cot (X) = Cot (X) = Sin (X) = Sin (X) = Sin (Y) = Sin (Y	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 E = rad × 186 E deg × π / E = rad × 186 E = rad	90°) 00°) Sin (X) ad 0 / π 180 2)*sin(rx) *sin(rx)	3dGraphicProgramma		
Rotate around AXX X X X X X X	ometry ometry ometry goras Rule X Y Z	Adjace Cos (α O / Ta √ (H² x2 = x1* (y2 = x1* (x2 = x1* (x2 = x1* (x3 = x1* (x4 = x1* (α (α) = (α) * H (α)	Op Sin (α Tan (α * cos(ry) * cos(ry) * (ry) * (ry) * y*sinx-sin * cosx+sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hy O / S	Csc (α) = Sec (α) = Cot (α) = P = in (α) cs (α) Y Pit sin(ry)*sin(ry) cos(ry) * ny *siny*sinx *siny*cos	Hypoten Hypoten Hypoten ATan (ACos (ASin (180 - (1	= / Adjace suse / Oppose = O / A) t / Oppose 0 / A) A / H) O / H) 90 + β) *cos(rx) sx) + z1* Y α 1	cos(rz)*s cos(rz)*s sin(rz)*si (cosy*sin. (cosy*cos	Cos (X) = Sin (X) = Cos (X) = Sin (X) = Cos (X) = Cot (X) = Cot (X) = Sin (X) = Sin (X) = Sin (Y) = Sin (Y	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 E = rad × 186 E deg × π / E = rad × 186 E = rad	90°) 10°) Sin (X) ad 0 / π 180 2)*sin(rx) *sin(rx) 2/details/Amiga.	3dGraphicProgram 3T-3D_Graphics_P	rogramming	
Small Rotate Angle around Angle XYX XYZ	ometry cometry cometry goras Rule X Y Z	Adjace Cos (α O / Ta √ (H' x2 = x1* (y2 = x1* (z2 = x1* (x3 = x4 =	α ent j = α * H α * H α α α - 0¹ X F cos(rz) * sin(rz) * -sin(rz) * (cosz*cos * (cosz*sin) * (cosz*sin) * α α α α α α α α α α	Sin (a) Tan (a) √(H* cos(ry) cos(ry) y) + y1* (y*sinx-sin z 0	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hypometry O/S A/Co Cos(rz)*s sin(rz)*si y1* (sinz: y1* (sinz: y1* (sinz:	Csc (α) = Sec (α) = Cot (α) = in (α) is (α) Y Pit sin(ry)*sin(ry) cos(ry)* **siny*sinx* **siny*cos X Y Z	Hypotene Hypotene Hypotene Adjacent ATan (ACos (ASin () 180 - () tch β rx) - sin(rz x) + cos(rz * sin(rx) +cosz*co x-cosz*sin X 1 -α -αβ	= / Adjace suse / Oppose = O / A) A / H) O / H) **cos(rx) **cos(rx) **xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	cos(rz)*s cos(rz)*s sin(rz)*si (cosy*sin. (cosy*cos	Cos (X) = Sin (X) = CSin (X) = Sin (X) =	Sin ($X + 9$ Cos ($X - 9$ 1/Sin (X) 1/Cos ($X - 9$ 1/Sin (X) 1/Cos ($X - 9$ 1/Sin ($X - 9$) 1/Tan ($X - 9$) 2 Fa = $x = x = x = x = x = x = x = x = x = $	90°) 10°) Sin (X) ad 0 / π / 180 2)*sin(rx) 2)*sin(rx) g/details/Amiga:	3dGraphicProgram 5T-3D_Graphics_P th-primer-for-graph		ment-2e
Small Rotate Angle around Approx XYZ	ometry ometry ometry goras Rule X Y Z X = X + 4	Adjace Cos (α O / Ta √ (H' x2 = x1* (y2 = x1* (z2 = x1* (α -αβ x * Y	C(ent J = (α) * H an (α) - O²) X F cos(rz) * - sin (cosz*cos (cosz*sin) (cosz*sin) γ α 1 β	Op Sin (α Tan (α * cos(ry) * cos(ry) * (ry) * (ry) * y*sinx-sin * cosx+sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hypometry O / S A / Co Cos(rz)*s sin(rz)*si y1* (sinz: y1* (sinz: y1* (sinz:	Csc (α) = Sec (α) = Cot (α) = P = in (α) i	= Hypoten = Hypoten = Hypoten = Adjacent ATan (ACos (ASin () 180 - () tch β xx) - sin(rz x) + cos(rz * sin(rx) - α - αβ x * () - α - αβ x * () - α	= / Adjace suse / Oppose = O / A) A / H) O / H) **Cos(rx) **Cos(rx) **sx) + z1** **rx) + z1** β **x * X)	cos(rz)*s cos(rz)*s sin(rz)*si (cosy*sin. (cosy*cos	Cos (X) = Sin (X) = CSin (X) = Sin (X) =	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Cos (X) 1/Cos (X) 1/Tan (X) Cos (X) / 9 E = rad × 186 E deg × rr AW Y TXX) + sin(rz X) - cos(rz Cos(rx) Sources: https://archive.on	90°) 10°) Sin (X) ad 0 / π / 180 2)*sin(rx) 2)*sin(rx) g/details/Amiga:	3dGraphicProgram 5T-3D_Graphics_P th-primer-for-graph	rogramming	ment-2e
Small Rotate Angle around Approx XYZ XPD Approx XYZ	ometry ometry ometry goras Rule X Y Z X = X + c Y = Y - c	Adjace Ad Cos (α O / Ta $\sqrt{(H')}$ $x^2 = x^{1*}(y^2 = x^{1*}(y^2 = x^{1*}(y^2 = x^2))$ $x^2 = x^{1*}(y^2 = x^2)$ $x^2 = x^2$	C(Op Sin (α Tan (α * cos(ry) * cos(ry) * (ry) * (ry) * y*sinx-sin * cosx+sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hypometry O/S A/Co Cos(rz)*si -z1* sin(rz)*si -z1* siny1* (sinz: y1* (sinz: y1* (sinz:	Csc (α) = Sec (α) = Cot (α) = Cot (α) = α =	= Hypoten = Hypoten = Hypoten = Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz x) + cos(rz * sin(rx) * cosz*si X 1 -α -αβ x * (Y - α x * X - β	= / Adjace suse / Opposition = / Opposition 0 / A) A / H) O / H) ** ** ** ** ** ** ** ** **	cos(rz)*s cos(rz	Cos (X) = Sin (X) = CSin (X) = CSin (X) = CSin (X) = Sin (X) = Sin (X) = Cot (X) = CSin (X) = Sin (X) = Si	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Sin (X) 1/Sin (X) 1/Tan	90°) 10°) Sin (X) ad 0 / π 180 2)*sin(rx) 2)*sin(rx) 2)/details/Amiga. 2/details/Arar_s 2/	3dGraphicProgram 5T-3D_Graphics_P th-primer-for-graph peebasm ution_matrix	rogramming ics-and-game-develop	ment-2e
Small Rotate Angle around Approx XYZ XPD Approx XYZ	ometry ometry ometry goras Rule X Y Z X = X + c Y = Y - c	Adjace Cos (α O / Ta √ (H' x2 = x1* (y2 = x1* (z2 = x1* (α -αβ x * Y	C(Op Sin (α Tan (α * cos(ry) * cos(ry) * (ry) * (ry) * y*sinx-sin * cosx+sin	Trigone Trigone 7) * H 1) * A - A²) sinz*cosy, z*cosy) +	Hypometry O/S A/Co Cos(rz)*si -z1* sin(rz)*si -z1* siny1* (sinz: y1* (sinz: y1* (sinz:	Csc (α) = Sec (α) = Cot (α) = P = in (α) i	= Hypoten = Hypoten = Hypoten = Adjacent ATan (ACos (ASin (180 - (tch β rx) - sin(rz x) + cos(rz * sin(rx) * cosz*si X 1 -α -αβ x * (Y - α x * X - β	= / Adjace suse / Opposition = / Opposition 0 / A) A / H) O / H) ** ** ** ** ** ** ** ** **	cos(rz)*s cos(rz	Cos (X) = Sin (X) = CSin (X) = CSin (X) = CSin (X) = Sin (X) = Sin (X) = Cot (X) = CSin (X) = Sin (X) = Si	Sin (X + 9 Cos (X - 9 1/Sin (X) 1/Sin (X) 1/Sin (X) 1/Tan	90°) 10°) Sin (X) ad 0 / π 180 2)*sin(rx) 2)*sin(rx) 2)/details/Amiga. 2/details/Arar_s 2/	3dGraphicProgram ST-3D_Graphics_P th-primer-for-graph seebasm	rogramming ics-and-game-develop	ment-2e

	Note	С	C#/Db	D	D ♯ /E ♭	E	F	F ♯ /G ♭	G	G#/A>	А	A ♯ / B ♭	В
	Octave 0	16.3516	17.3239	18.3541	19.4454	20.6017	21.8268	23.1247	24.4997	25.9565	27.5000	29.1352	30.8677
	Octave 1	32.7032	34.6478	36.7081	38.8909	41.2034	43.6535	46.2493	48.9994	51.9131	55.0000	58.2705	61.7354
>	Octave 2	65.4064	69.2957	73.4162	77.7818	82.4069	87.3071	92.4986	97.9989	103.8262	110.0000	116.5409	123.4708
ency	Octave 3	130.813	138.591	146.832	155.564	164.814	174.614	184.997	195.998	207.652	220.000	233.082	246.942
dne	Octave 4	261.626	277.183	293.665	311.127	329.628	349.228	369.994	391.995	415.305	440.000	466.164	493.883
Frec	Octave 5	523.251	554.365	587.330	622.254	659.255	698.457	739.989	783.991	830.609	880.000	932.328	987.767
ш	Octave 6	1046.50	1108.73	1174.66	1244.51	1318.51	1396.91	1479.98	1567.98	1661.22	1760.00	1864.66	1975.53
	Octave 7	2093.01	2217.46	2349.32	2489.02	2637.02	2793.83	2959.96	3135.96	3322.44	3520.00	3729.31	3951.07
	Octave 8	4186.01	4434.92	4698.64	4978.03	5274.04	5587.65	5919.91	6271.93	6644.88	7040.00	7458.62	7902.13



Floating Point	16-bit Signed	8bit signed	8 Bit Unsigned	4 Bit Unsigned
1	32767	127	255	15
	•••	•••	•••	•••
			•••	•••
		•••	•••	•••
0	0	0	128	8
	•••	•••		•••
				•••
	•••			•••
-1	-32768	-128	0	0

