

Appendix G - Per-CPU Opcode Quick Reference

One table per CPU. Each table is a single-line summary of every mnemonic; the full encoding, addressing modes, flag effects, and timing notes are in the per-CPU chapter (Chapters 25-30). This appendix is for the reader who has the chapter open already and only needs to look up "is this mnemonic spelled this way" or "what does this opcode do at a glance".

The byte-entry crib under each CPU is deliberately small. It covers the encodings used by the chapter example so a reader can type, disassemble, and check the machine-code program directly in IE Mon. For IE64, IE Mon can also assemble one instruction at a time with `A addr`; the emitted bytes are still shown here because the bytes and the `d` listing are the permanent check.

The IE64 opcode names and byte values in this appendix are checked against the shared IE64 opcode table used by the CPU, monitor disassembler, in-machine assembler, and companion assembler and disassembler surfaces. That keeps the lookup table, byte-entry examples, and monitor listings on the same spelling.

G.1 IE64 (Chapter 25)

Fixed 8-byte instruction, 32 GPRs R0-R31, R0 = 0, R31 = SP. Compare-and-branch ISA; no separate flag register.

With IE64 as the focussed CPU, `A addr` accepts one mnemonic instruction and prints the 8 bytes it wrote. Use it for short IE64 programs when the mnemonic form is easier to type, then confirm with `d` just as Chapter 25 does.

Group	Mnemonics
ALU	ADD, SUB, MULU, MULS, DIVU, DIVS, MOD, MODS, AND, OR, EOR, NOT, NEG, LSL, LSR, ASR, ROL, ROR, CLZ, CTZ, POPCNT, BSWAP.
Load / store	LOAD.B/.W/.L/.Q, STORE.B/.W/.L/.Q. Address modes: register indirect, register plus displacement, and absolute.
Immediate	MOVE.Q rd, #imm32, MOV.T rd, #imm32, MOVEQ rd, rs, LEA rd, disp(rs).
Branch	BRA, JSR, RTS, JMP, BEQ, BNE, BLT, BLE, BGT, BGE, BHI, BLS. Conditional forms compare two GPRs and a target.
FPU	FLOAD, FSTORE, FADD, FSUB, FMUL, FDIV, FMOD, FNEG, FABS, FSQRT, FCMP, FCVTIF, FCVTFI, FSIN, FCOS, FTAN, FATAN, FLOG, FEXP, FPOW, DMOV, DLOAD, DSTORE, DADD, DSUB, DMUL, DDIV, DMOD, DABS, DNEG, DSQRT, DINT, DCOMP, DCVTIF, DCVTFI, FCVTS.D, FCVTD.S, DSIN, DCOS, DTAN, DATAN, DLOG, DEXP, DPOW.
System	SYSCALL, HALT, WAIT, MTCR, MFCR, ERET, TLBFLUSH, TLBINVAL, SMODE.

The in-machine assembler also accepts the IE64 source forms BEQZ, BNEZ, BLTZ, BGEZ, BGTZ, and BLEZ. They are not separate architectural opcodes. They encode the corresponding compare-and-branch operation with R0 as the second register.

Selected IE64 FP64 opcode bytes:

Opcode	Mnemonic	Notes
\$80	DMOV	Copy an FP64 register pair.
\$81	DLOAD	Load a 64-bit FP value from memory.
\$82	DSTORE	Store a 64-bit FP value to memory.

Opcode	Mnemonic	Notes
\$83-\$87	DADD ... DMOD	FP64 arithmetic.
\$88-\$8B	DABS ... DINT	FP64 unary operations.
\$8C	DCMP	FP64 compare.
\$8D-\$90	DCVTIF, DCVTFI, FCVTSD, FCVTDS	Integer and width conversions.
\$91	DSIN	FP64 sine.
\$92	DCOS	FP64 cosine.
\$93	DTAN	FP64 tangent.
\$94	DATAN	FP64 arctangent.
\$95	DLOG	FP64 natural logarithm.
\$96	DEXP	FP64 exponent.
\$97	DPOW	FP64 power.

Byte-entry crib for Chapter 25:

Bytes	Meaning	Hand-entry note
01 17 00 00 ii ii ii ii	MOVE.Q R2,#imm32	Byte 1 is (2 << 3) (3 << 1) 1; the immediate is little-endian.
01 0F 00 00 ii ii ii ii	MOVE.Q R1,#imm32	Byte 1 is (1 << 3) (3 << 1) 1; use this for small values before a store.
11 08 10 00 dd dd dd dd	STORE.B R1,disp(R2)	Byte 1 selects R1 and byte size; byte 2 selects base register R2; displacement is little-endian.
11 0C 10 00 dd dd dd dd	STORE.L R1,disp(R2)	Same register fields, size code L; used for 32-bit frequency or address registers.
40 06 00 00 dd dd dd dd	BRA disp32	The Chapter 25 example uses displacement 0, which branches back to the branch instruction itself.

G.2 IE32 (Chapter 26)

8-byte instruction, 16 named registers A, X, Y, Z, B, C, D, E, F, G, H, S, T, U, V, W, no MMU, no FPU.

Group	Mnemonics
ALU	ADD, SUB, MUL, DIV, MOD, AND, OR, XOR, NOT, NEG, SHL, SHR.
Load / store	LDA/LDX/LDY/LDZ, LDB-LDW, STA/STX/STY/STZ, STB-STW, plus generic LOAD/STORE.
Immediate	Immediate addressing mode on load and ALU instructions.
Branch	JMP, JSR, RTS, JZ, JNZ, JGT, JGE, JLT, JLE.
Stack	PUSH rA, POP rA.
Timing	WAIT n delays for approximately n microseconds during normal execution.

Byte-entry crib for Chapter 26:

Bytes	Meaning	Hand-entry note
20 00 00 00 vv vv vv vv	LDA A,#imm32	Opcode \$20, register A, immediate mode, little-endian value.
24 00 04 00 aa aa aa aa	STA A,M:addr32	Opcode \$24, register A, direct-memory mode, little-endian address.
06 00 00 00 aa aa aa aa	JMP addr32	Opcode \$06; the Chapter 26 example jumps to its own address to keep the tone alive.

G.3 6502 (Chapter 27)

A, X, Y, S, P, PC. Addressing modes: implied, accumulator, immediate, zero-page (+ X, + Y), absolute (+ X, + Y), indirect, (indirect,X), (indirect),Y, relative.

Group	Mnemonics
Load / store	LDA, LDX, LDY, STA, STX, STY.
Transfer	TAX, TAY, TXA, TYA, TSX, TXS.
Arith / log	ADC, SBC, AND, ORA, EOR, BIT, CMP, CPX, CPY.
Inc / dec	INC, DEC, INX, INY, DEX, DEY.
Shift	ASL, LSR, ROL, ROR.
Branch	BCC, BCS, BEQ, BNE, BMI, BPL, BVC, BVS.
Jump / sub	JMP, JSR, RTS, RTI.
Flag	CLC, SEC, CLD, SED, CLI, SEI, CLV.
Stack	PHA, PLA, PHP, PLP.
System	BRK, NOP.
Undoc	LAX, SAX, DCP, ISC, RLA, RRA, SLO, SRE, ANC, ARR, ASR, AXS, XAA.

Flag register P: bit 0 C, 1 Z, 2 I, 3 D, 4 B, 5 -, 6 V, 7 N (silicon convention used by this chip).

Byte-entry crib for Chapter 27:

Bytes	Meaning	Hand-entry note
A9 nn	LDA #nn	Immediate load into accumulator.
8D ll hh	STA \$hhll	Absolute store; address operand is low byte then high byte.
4C ll hh	JMP \$hhll	Absolute jump; the Chapter 27 example jumps to its own address to leave POKEY sounding.

G.4 Z80 (Chapter 28)

Main: A, F, B, C, D, E, H, L; alt: A', F', B', C', D', E', H', L'; index: IX, IY; refresh / interrupt: I, R; stack: SP. Prefixes: CB (bit / rotate), ED (extended), DD / FD (IX / IY).

Group	Mnemonics
8-bit load	LD r, r', LD r, n, LD r, (HL), LD r, (IX+d), LD A, (BC)/(DE)/(nn).

Group	Mnemonics
16-bit load	LD rp, nn, LD rp, (nn), LD (nn), rp, PUSH rp, POP rp.
Block	LDI, LDIR, LDD, LDDR, CPI, CPIR, CPD, CPDR.
Arith / log	ADD, ADC, SUB, SBC, AND, OR, XOR, CP, INC, DEC.
16-bit ALU	ADD HL, rp, ADC HL, rp, SBC HL, rp, ADD IX, rp, ADD IY, rp.
Rotate / shift	RLCA, RLA, RRCA, RRA, RLC, RL, RRC, RR, SLA, SRA, SRL, SLL (undoc).
Bit	BIT b, r, SET b, r, RES b, r.
Jump / call	JP, JR, CALL, RET, with condition codes NZ, Z, NC, C, PO, PE, P, M. DJNZ.
I/O	IN A, (n), IN r, (C), OUT (n), A, OUT (C), r, INI, OUTI, INIR, OTIR, IND, OUTD, INDR, OTDR.
Interrupt	EI, DI, IM 0, IM 1, IM 2, RETI, RETN.
Misc	NOP, HALT, EX, EXX, DAA, CPL, NEG, CCF, SCF.

Flag register: S, Z, Y (bit 5 undoc), H, X (bit 3 undoc), P/V, N, C.

Byte-entry crib for Chapter 28:

Bytes	Meaning	Hand-entry note
3E nn	LD A, nn	Immediate load into accumulator.
32 ll hh	LD (\$hhll), A	Direct memory store; address operand is low byte then high byte.
D3 pp	OUT (pp), A	Immediate port output; Chapter 28 uses PSG ports \$F0 and \$F1.
C3 ll hh	JP \$hhll	Absolute jump; address operand is low byte then high byte.

G.5 M68K (MC68020-Class, Chapter 29)

D0-D7, A0-A7, PC, SR. CCR low byte: X N Z V C. Big-endian, byte-addressable, full 32-bit client path on the Intuition Engine bus.

Group	Mnemonics
Move	MOVE, MOVEA, MOVEM, MOVEP, MOVEQ, EXG, SWAP, LEA, PEA.
Arith	ADD, ADDA, ADDI, ADDQ, ADDX, SUB, SUBA, SUBI, SUBQ, SUBX, NEG, NEGX, CMP, CMPA, CMPI, CMPM, CLR, EXT, EXTB.
Mul / div	MULU, MULS, DIVU, DIVS; MC68020 long forms MULU.L, MULS.L, DIVUL, DIVSL.
Logical	AND, ANDI, OR, ORI, EOR, EORI, NOT.
Shift / rot	ASL, ASR, LSL, LSR, ROL, ROR, ROXL, ROXR.
Bit	BTST, BSET, BCLR, BCHG.
Bit field	BFTST, BFSET, BFCLR, BFCHG, BFEXTS, BFEXTU, BFINS, BFFF0.
Branch	BRA, BSR, Bcc (HI, LS, CC, CS, NE, EQ, VC, VS, PL, MI, GE, LT, GT, LE), DBcc, Scc, JMP, JSR, RTS, RTR, RTD.
BCD	ABCD, SBCD, NBCD.
Multiprocessor	TAS, CAS, CAS2.

Group	Mnemonics
System	TRAP #n, TRAPV, CHK, CHK2, STOP, RESET, ILLEGAL, NOP, LINK, UNLK, MOVE USP, MOVEC, MOVES, BKPT, RTE.
FPU	68881-style FMOVE, FADD, FSUB, FMUL, FDIV, FMOD, FREM, FSCALE, FSQRT, FABS, FNEG, FCMP, FTST, transcendental functions, FBcc, FDBcc, FScc, FTRAPcc, FMOVECR, FMOVEM, control-register moves, FSAVE, and FRESTORE.
Line A/F	unassigned opcode trap.

FPU data-register direct operands use Dn directly for byte, word, long integer, and single-precision formats. Double and extended forms use memory or immediate operands; packed decimal is outside the accepted FPU memory formats.

Byte-entry crib for Chapter 29:

Bytes	Meaning	Hand-entry note
13 FC 00 vv aa aa aa aa	MOVE.B #vv, \$aaaaaaaa	Opcode and extension words are big-endian; the byte immediate sits in the low byte of the \$00vv extension word.
60 00 dd dd	BRA.W disp16	Signed big-endian displacement from the following word; Chapter 29 uses \$FFFE for a self-loop.

G.6 x86 (Chapter 30, 8086 base + 386 and selected later extensions)

EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, plus 16-bit and 8-bit subregisters. Segments CS, DS, ES, FS, GS, SS. Real-mode only.

Group	Mnemonics
Move	MOV, MOVZX, MOVSX, CMOVcc, LEA, XCHG, XLAT, PUSH, POP, PUSHA, POPA, PUSHAD, POPAD, PUSHF, POPF, PUSHFD, POPFD.
Arith	ADD, ADC, SUB, SBB, INC, DEC, CMP, NEG, MUL, IMUL, DIV, IDIV, DAA, DAS, AAA, AAS, AAM, AAD, CBW, CWD, CWDE, CDQ.
Logical	AND, OR, XOR, NOT, TEST.
Shift / rot	SHL, SHR, SAR, SAL, ROL, ROR, RCL, RCR, SHLD, SHRD.
Bit	BT, BTS, BTR, BTC, BSF, BSR, SETcc.
Branch	JMP, Jcc (JE, JNE, JL, JG, JLE, JGE, JB, JA, JBE, JAE, JO, JNO, JS, JNS, JP, JNP, JCXZ, JECXZ), CALL, RET, RETF, LOOP, LOOPE, LOOPNE, INT n, INTO, IRET.
String	MOVS, CMPS, SCAS, LODS, STOS, INS, OUTS, prefixes REP, REPE, REPNE.
I/O	IN, OUT.
Flag	STC, CLC, CMC, STD, CLD, STI, CLI, LAHF, SAHF.
Segment	LDS, LES, LFS, LGS, LSS.
System	HLT, WAIT, NOP, ESC, LOCK.
Flat-mode extras	BSWAP, CMOVcc, MOVSX, MOVZX, dword forms of 16-bit ops via 66h / 67h prefixes.

Omitted (Chapter 30): all protected-mode opcodes (LGDT, LIDT, LLDT, LTR, LMSW, SMSW, ARPL, LAR, LSL, VERR, VERW, STR, SLDT), CR and DR register moves, INVLPG, WBINVD, INVD.

Byte-entry crib for Chapter 30:

Bytes	Meaning	Hand-entry note
B0 nn	MOV AL, nn	Immediate load into AL.
A2 aa aa aa aa	MOV [addr32], AL	Absolute store from AL; address operand is little-endian.
EB dd	JMP SHORT disp8	Signed displacement from the next byte; \$FE loops to the jump instruction itself.