

Chapter 29 - The M68K MC68020-Class Processor

The M68K is the 16/32-bit processor that replaced the 6502 and the Z80 in the late 1980s machines: the Atari ST, the Commodore Amiga, the early Apple Macintosh, and the Sega Mega Drive. The Intuition Engine M68K is a full MC68020-class integer CPU. It has the original 68000 programmer's model plus the MC68020 extensions: bit-field operations, scaled indexing, full extension words, 32-bit multiply and divide forms, packed BCD helpers, compare-and-swap, bounds checking, module call/return, and the line-F coprocessor path used by 68881-style floating point.

It has eight 32-bit data registers, eight 32-bit address registers, a flat 32-bit address space, and a supervisor/user mode split. Unlike the smaller chips, it sees Intuition Engine's MMIO map directly without any banking.

29.1 User-mode programmer's model

Group	Registers
Data registers	D0-D7, each 32 bits
Address registers	A0-A7, each 32 bits (A7 is the stack pointer)
Program counter	PC, 32 bits
Condition code reg	CCR, low byte of the status register

Each data register holds an 8-bit, 16-bit, or 32-bit value (byte/word/long). The size is selected by a suffix on the mnemonic: `MOVE.B`, `MOVE.W`, `MOVE.L`. Operations on the smaller sizes leave the unused high bits of the register unchanged.

Address registers always hold a 32-bit value. Sign-extension from 16-bit operations fills the whole register. The unique A7 (also called SP) is the active stack pointer and switches between user and supervisor copies on mode change.

The CCR has five condition code bits arranged X N Z V C (low to high) within an 8-bit byte:

Bit	Name	Meaning
4	X	Extend - copy of C for arithmetic; preserved for some shifts
3	N	Negative - bit 7/15/31 of the result
2	Z	Zero - result was zero
1	V	Overflow - signed overflow occurred
0	C	Carry - unsigned overflow / borrow

29.2 Supervisor-mode model

In supervisor mode, the user model is augmented with:

- The full 16-bit status register SR, of which the low byte is the CCR. The high byte holds:
- bit 15 T1, bit 14 T0: trace mode selectors
- bit 13 S: supervisor flag (1 = supervisor)

- bit 12 M: master/interrupt stack select (MC68020-class)
- bits 10-8 I2/I1/I0: interrupt priority mask (0-7)
- A separate **supervisor stack pointer** A7' (or SSP). The user stack pointer is preserved as USP.
- The **vector base register** VBR. The exception vector table starts at the address in VBR. On reset VBR is 0.
- The source and destination function code registers SFC and DFC, used by the MOVES instruction.
- The cache control register CACR and cache address register CAAR.

User code that tries to access these registers raises a privilege violation exception (vector 8).

29.3 Memory model

Multi-byte values are stored **big-endian** in M68K main memory: the most-significant byte goes at the lowest address. A 32-bit write of \$12345678 to address \$1000 places \$12 at \$1000, \$34 at \$1001, \$56 at \$1002, \$78 at \$1003.

Intuition Engine's MMIO devices are little-endian, however; the bus normalises every MMIO access so an M68K `MOVE.L D0, ($F0800).L` delivers the same 32-bit value to the audio chip as an IE64 `STORE.L (R5), Rd`. Plain RAM accesses keep their natural big-endian order.

Word and long operands at odd addresses are valid on the MC68020 (the 68000 would have raised an address error); this chip follows the MC68020 rule. Instruction fetches still require word alignment.

The address space is flat 32-bit. The MMIO map of Chapter 24 is accessible directly. The sign-extended alias at \$FFFF0000 makes M68K's pre-decrement from zero (`-(SP)` with `SP = 0`) work.

When a flat M68K image is loaded, its bytes are deposited into RAM at the M68K entry point. The loader does not fire MMIO side effects from image bytes that overlap device apertures, and it leaves the stack guard hole untouched. Once the image is running, ordinary M68K loads and stores still reach MMIO normally.

29.4 Addressing modes

The MC68020 provides 12 base modes plus several extensions:

#	Mode	Notation
1	Data register direct	Dn
2	Address register direct	An
3	Address register indirect	(An)
4	Postincrement	(An)+
5	Predecrement	-(An)
6	Indirect with 16-bit displacement	(d16, An)
7	Indirect with index + 8-bit displacement	(d8, An, Xn)
8	Absolute short	(xxx).W
9	Absolute long	(xxx).L
10	PC with 16-bit displacement	(d16, PC)
11	PC with index + 8-bit displacement	(d8, PC, Xn)
12	Immediate	#<data>

#	Mode	Notation
13	MC68020 memory indirect pre-indexed	([bd, An, Xn] , od)
14	MC68020 memory indirect post-indexed	([bd, An] , Xn, od)
15	MC68020 PC-relative memory indirect	([bd, PC, Xn] , od)

Scaled indexing (*1, *2, *4, *8) is available on every mode that takes an Xn. The full-extension word adds an optional 32-bit base displacement and 32-bit outer displacement.

29.5 Instruction set

The full instruction set is in Appendix G. The groups below give a flavour of what is available; each mnemonic accepts a size suffix .B/.W/.L unless noted.

29.5.1 Data movement

MOVE, MOVEA (move into address register), MOVEQ (move quick - sign-extend an 8-bit immediate into a long), MOVEM (move multiple registers), EXG (exchange two registers), LEA (load effective address), PEA (push effective address), SWAP (swap halves of a long), LINK / UNLK (set up / tear down a stack frame), MOVES (move with function code).

29.5.2 Arithmetic

ADD, ADDA, ADDI, ADDQ, ADDX (add with extend); SUB, SUBA, SUBI, SUBQ, SUBX; MULU, MULS (32-bit forms on MC68020-class chips); DIVU, DIVS; NEG, NEGX; CLR; EXT and EXTB.L (sign-extend); CMP, CMPA, CMPI, CMPM, CMP2 (bounds-check compare). Decimal: ABCD, SBCD, NBCD, PACK, UNPK.

29.5.3 Logic

AND, ANDI, OR, ORI, EOR, EORI, NOT. The I-suffix forms accept an immediate as the source.

29.5.4 Shifts and rotates

ASL, ASR (arithmetic shifts); LSL, LSR (logical shifts); ROL, ROR (rotate); ROXL, ROXR (rotate through extend). Each can shift by a fixed 1-8 or by the count in a data register.

29.5.5 Bit operations and bit fields

BTST, BSET, BCLR, BCHG: test/set/clear/change one bit.

The MC68020 adds full bit-field operations on arbitrary-width fields at arbitrary offsets in memory or registers: BFTST, BFEXTU, BFEXTS, BFCHG, BFCLR, BFSET, BFINS, BFFF0 (find first one).

29.5.6 Branches and jumps

Mnemonic	Effect
Bcc label	Conditional branch (Bcc = BEQ, BNE, BCS, BCC, BMI, BPL, BVVS, BVC, BHI, BLS, BGT, BGE, BLT, BLE, BRA, BSR)
DBcc Dn, label	Decrement and branch if condition (Dn -= 1; branch if CC and Dn != -1)
Scc <ea>	Set byte to all-ones if condition true, else zero
JMP <ea>	Unconditional jump

Mnemonic	Effect
JSR <ea>	Jump to subroutine
BSR label	Branch to subroutine (PC-relative)
RTS	Return from subroutine
RTD #n	Return and add n to SP

29.5.7 System

Mnemonic	Effect
NOP	No operation
STOP #imm	Set SR, stop until interrupt or trace
RESET	Assert the external reset line (peripherals reset)
TRAP #n	Software interrupt (vector 32+n)
TRAPV	Trap if overflow flag set
CHK <ea>, Dn	Trap if Dn outside bounds
CHK2	Bounds-check against a pair of bounds
MOVE to/from SR	Read/write status register (privileged on 68010+)
ANDI to SR, ORI to SR, EORI to SR	Bit operations on SR (privileged)
RTE	Return from exception
MOVEC	Read/write control register (privileged)
CAS, CAS2	Atomic compare-and-swap
CALLM, RTM	Module call/return

29.5.8 Floating point

Line-F opcodes whose coprocessor ID is 1 enter the 68881-style floating-point decoder. Normal M68K CPU startup includes that FPU. If the FPU is absent, if the coprocessor ID is not 1, or if the FPU sub-operation is outside the accepted set, the instruction raises the line-F exception. The floating-point model has eight data registers FP0-FP7 plus the control registers FPCR, FPSR, and FPIAR.

The useful arithmetic and data-movement set includes FMOVE, FMOVEM, FMOVECR, FINT, FINTRZ, FSQRT, FABS, FNEG, FADD, FSUB, FMUL, FDIV, FMOD, FREM, FSCALE, FCMP, FTST, FLOGN, FLOG10, FLOG2, FETOX, FTWOTOX, FTENTOX, FSIN, FCOS, FTAN, FASIN, FACOS, FATAN, FSINH, FCOSH, FTANH, FGETEXP, and FGETMAN. Floating conditional forms FBcc, FDBcc, FScc, and FTRAPcc are decoded through the same condition-code register.

Immediate FPU operands are accepted for byte, word, long, single, double, and extended formats. Memory operands use the same formats except for packed decimal. Extended real values are stored as a 12-byte 68881-style memory field. FMOVEM can move multiple FP data registers in this extended format and can also move FPCR, FPSR, and FPIAR as longwords. Pre-decrement and post-increment addressing advance by 12 bytes per selected FP data register, or 4 bytes per selected control register.

Data-register direct FPU operands use the Dn register itself, not a memory cell at address zero. For FMOVE Dn, Fpm and the arithmetic forms that use Dn as the source, byte, word, long integer, and single-precision formats are accepted. For FMOVE

F_m, D_n, byte and word destinations update the low part of D_n, long integer replaces the whole register, and single precision stores the IEEE bits in D_n. Double and extended values need memory or immediate operands because one data register cannot hold them.

Single-precision and double-precision qualified opmodes are decoded as result-precision requests. A single-qualified operation rounds the destination result to single precision after the operation. A double-qualified operation leaves the result at the machine's double storage precision.

FSAVE writes an idle FPU state frame and FRESTORE consumes one. They are supervisor instructions. The idle frame starts with version byte \$1F and frame-size byte \$18; the remaining frame payload is zero because this FPU has no hidden pipeline state. A null restore frame resets FPCR, FPSR, and FPIAR to zero.

Packed-decimal FPU memory operands are not accepted. Use integer PACK/UNPK for decimal byte work, or convert values to a binary floating format before using the FPU.

29.6 Exception model

The exception vector table holds 256 four-byte entries at the address in VBR. After reset VBR = 0, so the table starts at address 0.

Selected vector assignments used by Intuition Engine:

Vector	Offset	Meaning
0	\$000	Reset: initial SSP
1	\$004	Reset: initial PC
2	\$008	Bus error
3	\$00C	Address error
4	\$010	Illegal instruction
5	\$014	Divide by zero
6	\$018	CHK / CHK2 instruction
7	\$01C	TRAPV instruction
8	\$020	Privilege violation
9	\$024	Trace
10	\$028	Line A trap (unimplemented A-line opcode)
11	\$02C	Line F trap (unimplemented F-line opcode)
24	\$060	Spurious interrupt
25-31	\$064-\$07C	Auto-vectored interrupt levels 1-7
32-47	\$080-\$0BC	TRAP #0 through TRAP #15
64+	\$100+	User-defined interrupts

When an exception fires, the CPU:

1. Computes a stack frame containing at least PC and SR.
2. Switches to supervisor mode if not already there.
3. Pushes the frame onto the supervisor stack.

4. Loads PC from $VBR + 4 * \text{vector}$.

RTE reverses the sequence and restores SR.

29.7 Interrupts on Intuition Engine

The Intuition Engine raises M68K interrupts at level 5 for video events (Copper, VBlank) and level 4 for audio engines, using auto-vector mode. A program that wants to respond to either installs a handler address in the corresponding auto-vector slot: level 5 uses vector offset \$74, and level 4 uses vector offset \$70. Store the handler address there as a big-endian longword, then set the interrupt mask in SR low enough to admit the level you want.

Setting I2:I1:I0 in SR to 4 permits both; to 5 permits only level 5. Level 7 is non-maskable.

The IRQ diagnostics block in Chapter 24 reports live counts of these auto-vectored interrupts and is useful when debugging a hung handler.

29.8 Bounded differences

The programming model is MC68020-class, with these documented boundaries:

- Address translation is not present. The PMMU instructions and page-table machinery associated with later family members are outside this chip.
- Cache state and cache timing are not modelled; CACR and CAAR are programmer-visible control registers, not a promise of cycle-accurate cache behaviour.
- Trace mode bits T1 and T0 are stored in SR, but they do not raise trace exceptions after each instruction.
- Dynamic bus sizing is not modelled. External transfers use the full 32-bit M68K client path on the Intuition Engine bus.
- Line-F coprocessor ID 1 is the 68881-style FPU path. Other coprocessor IDs, absent-FPU cases, and unsupported FPU sub-ops raise the line-F exception.
- FSAVE and FRESTORE are accepted only in supervisor mode.
- Packed-decimal FPU memory operands are not accepted.

29.9 A small example

This M68K byte-entry program starts SID voice 1 as a sawtooth tone. The M68K instruction stream is big-endian, but the byte writes land on the SID's byte-wide registers exactly as shown:

```

(m68k)> w 1000 13 FC 00 01 00 0F 08 00 13 FC 00 0F 00 0F 0E 18
(m68k)> w 1010 13 FC 00 C3 00 0F 0E 00 13 FC 00 10 00 0F 0E 01
(m68k)> w 1020 13 FC 00 00 00 0F 0E 05 13 FC 00 F0 00 0F 0E 06
(m68k)> w 1030 13 FC 00 21 00 0F 0E 04 60 00 FF FE
(m68k)> r pc 1000
(m68k)> d 1000 8
> 001000: 13FC 0001 000F 0800      MOVE.B #$01, $000F0800
   001008: 13FC 000F 000F 0E18      MOVE.B #$0F, $000F0E18
   001010: 13FC 00C3 000F 0E00      MOVE.B #$C3, $000F0E00
   001018: 13FC 0010 000F 0E01      MOVE.B #$10, $000F0E01
   001020: 13FC 0000 000F 0E05      MOVE.B #$00, $000F0E05
   001028: 13FC 00F0 000F 0E06      MOVE.B #$F0, $000F0E06
   001030: 13FC 0021 000F 0E04      MOVE.B #$21, $000F0E04
T 001038: 6000 FFFE      BRA.W $00001038
(m68k)> b 1038
(m68k)> g
(m68k)> m F0E00 1
000F0E00: C3 10 00 00 21 00 F0 00 00 00 00 00 00 00 00 00 .....!.....
(m68k)> m F0E18 1
000F0E18: 0F 00 00 7F 00 00 00 00 00 00 00 00 00 00 00 00 .....
(m68k)> bc 1038

```

M68K opcodes and extension words are entered in big-endian order:

Address	Bytes	Meaning
\$1000	13 FC 00 01 00 0F 08 00	MOVE.B #\$01, \$000F0800. Opcode word \$13FC is byte immediate to absolute-long; immediate word \$0001 supplies byte \$01; absolute address \$000F0800 is AUDIO_CTRL.
\$1008	13 FC 00 0F 00 0F 0E 18	Writes SID MODE_VOL at \$F0E18 to \$0F, maximum master volume.
\$1010	13 FC 00 C3 00 0F 0E 00	Writes frequency low byte \$C3 to SID voice 1 frequency low.
\$1018	13 FC 00 10 00 0F 0E 01	Writes frequency high byte \$10 to SID voice 1 frequency high.
\$1020	13 FC 00 00 00 0F 0E 05	Writes attack/decay \$00 for a fast start.
\$1028	13 FC 00 F0 00 0F 0E 06	Writes sustain/release \$F0, full sustain and quick release.
\$1030	13 FC 00 21 00 0F 0E 04	Writes control \$21, gate plus sawtooth.
\$1038	60 00 FF FE	BRA.W \$1038; extension word \$FFFE branches back to the branch instruction.

\$F0800 enables audio. \$F0E18 is MODE_VOL, so \$0F sets the SID master volume to maximum. The two frequency bytes \$C3 \$10 form SID frequency word \$10C3, close to concert A in the C64 tuning convention. AD=\$00 gives a fast attack/decay, SR=\$F0 sets full sustain, and CTRL=\$21 combines gate (\$01) with the sawtooth waveform bit (\$20). BRA.W uses extension word \$FFFE, which branches back to the branch instruction itself.

29.10 Voodoo triangle example

The M68K can reach the Voodoo card directly because it has a flat 32-bit address space. This example uses the regular `MOVE.L`

value,\$address` form to clear the Voodoo framebuffer to blue,

submit a flat-shaded triangle, and swap the buffer so the triangle appears. No assembler is involved; each register write is one 10-byte instruction.

Voodoo coordinates use 12.4 fixed point, so 320 is entered as `$00001400`. The colour channels use 12.12 fixed point, so `$1000` means 1.0, `$0800` means 0.5, and `$0200` means 0.125.

```

(m68k)> w 1100 23 FC 00 00 00 01 00 0F 80 04
(m68k)> w 110A 23 FC FF 00 00 FF 00 0F 81 D8
(m68k)> w 1114 23 FC 00 00 00 00 00 0F 81 24
(m68k)> w 111E 23 FC 00 00 14 00 00 0F 80 08
(m68k)> w 1128 23 FC 00 00 06 40 00 0F 80 0C
(m68k)> w 1132 23 FC 00 00 1F 40 00 0F 80 10
(m68k)> w 113C 23 FC 00 00 17 C0 00 0F 80 14
(m68k)> w 1146 23 FC 00 00 08 C0 00 0F 80 18
(m68k)> w 1150 23 FC 00 00 17 C0 00 0F 80 1C
(m68k)> w 115A 23 FC 00 00 10 00 00 0F 80 20
(m68k)> w 1164 23 FC 00 00 08 00 00 0F 80 24
(m68k)> w 116E 23 FC 00 00 02 00 00 0F 80 28
(m68k)> w 1178 23 FC 00 00 10 00 00 0F 80 30
(m68k)> w 1182 23 FC 00 00 00 00 00 0F 80 80
(m68k)> w 118C 23 FC 00 00 00 01 00 0F 81 28
(m68k)> w 1196 60 00 FF FE
(m68k)> r pc 1100
(m68k)> d 1100 #16
> 001100: 23FC 0000 0001 000F 8004 MOVE.L #$00000001, $000F8004
  00110A: 23FC FF00 00FF 000F 81D8 MOVE.L #$FF0000FF, $000F81D8
  001114: 23FC 0000 0000 000F 8124 MOVE.L #$00000000, $000F8124
  00111E: 23FC 0000 1400 000F 8008 MOVE.L #$00001400, $000F8008
  001128: 23FC 0000 0640 000F 800C MOVE.L #$00000640, $000F800C
  001132: 23FC 0000 1F40 000F 8010 MOVE.L #$00001F40, $000F8010
  00113C: 23FC 0000 17C0 000F 8014 MOVE.L #$000017C0, $000F8014
  001146: 23FC 0000 08C0 000F 8018 MOVE.L #$000008C0, $000F8018
  001150: 23FC 0000 17C0 000F 801C MOVE.L #$000017C0, $000F801C
  00115A: 23FC 0000 1000 000F 8020 MOVE.L #$00001000, $000F8020
  001164: 23FC 0000 0800 000F 8024 MOVE.L #$00000800, $000F8024
  00116E: 23FC 0000 0200 000F 8028 MOVE.L #$00000200, $000F8028
  001178: 23FC 0000 1000 000F 8030 MOVE.L #$00001000, $000F8030
  001182: 23FC 0000 0000 000F 8080 MOVE.L #$00000000, $000F8080
  00118C: 23FC 0000 0001 000F 8128 MOVE.L #$00000001, $000F8128
T 001196: 6000 FFFE          BRA.W $00001196
(m68k)> b 1196
(m68k)> g
(m68k)> m F8008 2
000F8008: 00 14 00 00 40 06 00 00 40 1F 00 00 C0 17 00 00  ....@...@.....
000F8018: C0 08 00 00 C0 17 00 00 00 10 00 00 00 08 00 00  ....
(m68k)> m F8128 1
000F8128: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
(m68k)> bc 1196

```

The first three writes enable Voodoo, set COLOR0 to blue, and issue FAST_FILL_CMD to clear the framebuffer. The next six writes set triangle vertices A, B, and C:

Vertex	Register values	Screen position
A	\$00001400, \$00000640	(320, 100)
B	\$00001F40, \$000017C0	(500, 380)
C	\$000008C0, \$000017C0	(140, 380)

The four colour writes set red to 1.0, green to 0.5, blue to 0.125, and alpha to 1.0, giving the triangle a warm orange shade. Writing any value to TRIANGLE_CMD submits the current triangle. Writing 1 to SWAP_BUFFER_CMD presents it. The

monitor dump at \$F8008 shows the vertex registers in the Voodoo's little-endian register view, while the instruction bytes above remain ordinary big-endian M68K opcodes.

When the breakpoint is reached, the visible Voodoo layer has a blue background and a large orange triangle.

29.11 What comes next

Chapter 30 covers the last of the heritage CPUs: x86, the family that originated as the 8086 in 1978 and went on to dominate the PC industry. Intuition Engine implements a 32-bit x86 processor with full 8086/8088 semantics plus the 386-era 32-bit register and operand extensions, run in a permanent flat real-address mode.