

Instruction Set

The instruction set of the microcontroller consists of a few categories of instructions.

1. Data movement

- CPU & Memory (**Load / Store**)
- Register to register inside CPU (**Transfer / Exchange**)
- Memory to memory, definitely through CPU (**Move**)

2. Modify Data

- Arithmetic (**Add / Subtract, Multiply / Divide, ...**)
- Logical (**AND, OR, XOR, NOT, ...**)

3. Decision Making (**Test bits or counters,**)

4. Flow Control (**Branch, Jump, Subroutine call, ...**)

5. Miscellaneous

1. Data Movement (1 of 3)

CPU ↔ Memory/Address (Load / Store / Load effective address)

| Transfer from memory to CPU | | Store from CPU to memory | |
|--|--------------------------------|--|----------|
| Instruction | Meaning | Instruction | Meaning |
| LDAA <addr> | Load A | STAA <addr> | Store A |
| LDAB <addr> | Load B | STAB <addr> | Store B |
| LDD <addr> | Load D | STD <addr> | Store D |
| LDS <addr> | Load SP | STS <addr> | Store SP |
| LDX <addr> | Load index register X | STX <addr> | Store X |
| LDY <addr> | Load index register Y | STY <addr> | Store Y |
| All addressing modes are used | | All addressing modes except IMM | |
| Affect Z and N flags and resets V flag | | | |
| LEAS <addr> | Load effective address into SP | | |
| LEAX <addr> | Load effective address into X | | |
| LEAY <addr> | Load effective address into Y | | |
| Only indexed addressing is allowed | | No effect on flags | |

Data Movement (2 of 3)

CPU ↔ CPU (Transfer / Exchange)

| Transfer from register to register (Not PC) | | Exchange between two registers (Not PC) | |
|--|---|--|---|
| Instruction | Meaning | Instruction | Meaning |
| TAB / TBA | Transfer A to B / B to A | EXG reg, reg | Exchange register to register (A, B, CCR, D, X, Y, or SP) |
| TAP / TPA | Transfer A to CCR / CCR | XGDY / XGDY | Exchange D with X / Y |
| TFR reg, reg | Transfer register to register (A, B, CCR, D, X, Y, or SP) | | |
| TSX / TXS | Transfer S to X / X to S | | |
| TSY / TYS | Transfer S to Y / Y to S | | |
| Zero extension if 8-bit to 16-bit operands, or use SEX (sign-extend instruction) | | | |
| Only INH addressing mode | | | |
| No effect on flags except: •TAP affects NZVC (definitely) •TAB & TBA affects NZ and resets V | | No effect on flags | |

Data Movement (3 of 3)

Memory ↔ Memory (Move byte/word)

| Move byte or word from memory to memory | |
|--|--------------------------------------|
| Instruction | Meaning |
| MOVB <source addr>,<destination addr> MOVW <source addr>,<destination addr> | Move byte/Word from memory to memory |
| Source address can be IMM, EXT, or IDX Destination address can be EXT, or IDX | |
| No effect on flags | |

2. Modify Data

(1 of 6)

Arithmetic Add/Subtract

| Add register to register or to memory | | Subtract register from register or from memory | |
|--|----------------------------------|---|--|
| Instruction | Meaning | Instruction | Meaning |
| ABA ABX ABY | Add B to A, X, or Y | SBA | Subtract B from A |
| Inherent addressing | | | |
| ADDA <addr> ADDB <addr> ADDD <addr> | Add memory to A, B, or D. | SUBA <addr> SUBB <addr> SUBD <addr> | Subtract memory from A, B, or D |
| ADCA <addr> ADCB <addr> | Add with carry memory to A or B. | SBCA <addr> SBCB <addr> | Subtract with carry memory from A or B |
| All addressing modes are used | | | |
| NZVC | | | |

2. Modify Data

(2 of 6)

Arithmetic Increment/Decrement & Negate/Clear

| INC/DEC register or memory | | | Negate register or memory | | |
|---|---|------------------------------|------------------------------|------------------------------------|------------------------------|
| Instruction | Meaning | Flags affected | Instruction | Meaning | Flags affected |
| INCA/DECA INCB/DECB INS/DES INX/DEX INY/DEY | Increment/ Decrement A, B, SP, X, or Y | NZV NZV None Z Z | NEGA/NEGB CLRA/CLRB | 2's comp. A/B Clear A/B | NZVC Z=1 N=0 VC |
| Inherent addressing | | | | | |
| INC/DEC <addr> | Increment/ Decrement memory | NZV | NEG <addr> CLR <addr> | 2's comp. mem. Clear memory | NZVC Z=1 N=0 VC |
| EXT and IDX addressing | | | | | |

2. Modify Data

(3 of 6)

Multiply & Divide

| Multiply registers | | | Divide registers | | |
|--------------------|--|-------|------------------|--|---------------|
| Instruction | Meaning | Flags | Instruction | Meaning | Flags |
| MUL | 8-bit $(A)X(B) \rightarrow D$ | C | IDIV/IDIVS | 16-bit unsigned/signed $(D) \div (X) \rightarrow X$ Remainder $\rightarrow D$ | ZC and V=0 |
| EMUL/EMULS | 16-bit unsigned/signed $(D)X(Y) \rightarrow Y:D$ | NZC | EDIV/EDIVS | 32-bit by 16-bit unsigned/signed $(Y:D) \div (X) \rightarrow Y$ Remainder $\rightarrow D$ | NZVC |
| | | | FDIV | 16-bit fractional div $(D) \div (X) \rightarrow X$ Remainder $\rightarrow D$ | ZVC |

INH addressing

Multiply and Accumulate

| Instruction | Meaning | Flags |
|---------------------|---|-------|
| EMACS <16-bit addr> | Multiply two 16-bit memory operands and add the result to a third 32-bit sum in memory. | NZVC |

The two operands are pointed to by X & Y, and the sum by EXTended <16-bit addr>

2. Modify Data

(4 of 6)

Boolean Logic (AND/OR/XOR/NOT)

| AND/OR/XOR register to memory | | Complement register or from memory | |
|--|----------------------------------|---|------------------------------|
| Instruction | Meaning | Instruction | Meaning |
| ANDA <addr> ANDB <addr> ANDCC <addr> | Add A, B, or CCR with memory | COMA COMB | Complement A Complement B |
| ORAA <addr> ORAB <addr> ORCC <addr> | OR A, B, or CCR to memory | COM <addr> | Complement memory |
| EORA <addr> EORB <addr> | Exclusive OR A or B with memory. | | |
| All addressing modes are used | | | |
| NZ, and V=0 (ANDCC/ORCC affect all flags) (COM resets C) | | | |

2. Modify Data

(5 of 6)

Bit test and manipulation

| Bit Test accumulator AND memory | | Bit Set/Clear register or from memory | |
|---------------------------------|---------------------------------------|--|--|
| Instruction | Meaning | Instruction | Meaning |
| BITA <addr> BITB <addr> | Fictitious AND of A/B, with memory | SEC/CLC SEI/CLI SEV/CLV | Set/Clear C Set/Clear I Set/Clear V |
| | | Inherent addressing | |
| | | BSET/BCLR <addr>,mask | Set/Clear selected bits in memory, according to mask |
| All addressing modes are used | | Direct, extended, and indexed addressing modes are used | |
| NZ and V=0 | | | |

2. Modify Data

(6 of 6)

Shift & Rotate

| Shift (Logical/Arithmetic) | | Rotate (Logical/Arithmetic) | |
|--|---|-----------------------------|-----------------------------|
| Instruction | Meaning | Instruction | Meaning |
| LSL <addr> LSLA/LSLB LSLD | Logical shift left memory/A/B/D | ROL ROLA/ROLB | Rotate left memory/A/B |
| LSR <addr> LSRA/LSRB LSRD (N is set to 0) | Logical shift right memory/A/B/D | ROR RORA/RORB | Rotate right memory/A/B |
| ASL <addr> ASLA/ASLB ASLD | Arithmetic shift left instructions are the same as logical shift left memory/A/B/D | | |
| ASR <addr> ASRA/ASRB | Logical shift right memory/A/B | | |
| INH, EXT, IDX addressing | | | |
| Flags affected NZVC | | | |

3. Decision Making

Compare & Test ... Minimum & Maximum

“Subtraction”: Used to establish conditions for branch instructions

| Compare register to register/memory | | Test register/memory against \$00 | |
|--|--|--|--|
| Instruction | Meaning | Instruction | Meaning |
| CBA | Compare A to B | TST <addr> TSTA/TSTB | Compare memory to \$00 Compare A /B to \$00 |
| CMPA <addr> CMPB <addr> CPD <addr> CPS <addr> CPX <addr> CPY <addr> | Compare A/B/D/SP/X/Y to memory “(r) – (mem)” | | |
| All addressing modes are used | | | |

| Minimum of register and memory (unsigned) | | Maximum of register and memory (unsigned) | |
|--|---|--|---|
| Instruction | Meaning | Instruction | Meaning |
| MINA/MNM <addr> | Minimum of A and mem, result → A/mem | MAXA/ MAXM <addr> | Maximum of A and mem, result → A/mem |
| EMIND/EMINM <addr> | Minimum of D and mem, result → D/mem | EMAXD/EMAXM <addr> | Maximum of D and mem, result → D/mem |
| IDX addressing | | | |
| NZVC | | | |

4. Flow Control

Branch instructions (condition from CCR bits) / Jump Instruction

| Simple Condition Branch | | Composite Condition Branch | |
|--|--|--|---|
| Instruction | Meaning | Instruction | Meaning |
| BRA/BRN <8-bit offset> | Unconditional branch: Always/Never | Unsigned Branches: BHI <8-bit offset> | Branch if: Higher (R > M) C+Z = 0 |
| •BCC/BCS <8-bit offset> | Branch if: •C (C=0/1) | BLO <8-bit offset> | Lower (R < M) C = 1 |
| •BEQ/BNE <8-bit offset> | •Equal/Not Equal (Z=1/0) | BHS <8-bit offset> | Higher or Same (R ≥ M) C = 0 |
| •BPL/BMI <8-bit offset> | •Plus/Minus (N=0/1) | BLS <8-bit offset> | Lower or Same (R ≤ M) C+Z= 1 |
| •BVC/BVS<8-bit offset> | •V Clear/Set (V=0/1) | Signed Branches: BGT <8-bit offset> | Branch if: Higher (R > M) NV = 0 |
| REL addressing (8-bit: -128→ +127) | | BLT <8-bit offset> | Lower (R < M) |
| Long branches are available(16-bit -32,768 to +32,767) | | BGE <8-bit offset> | Higher or Same (R ≥ M) |
| Jump instruction | | BLE <8-bit offset> | Lower or Same (R ≤ M) |
| Instruction | Meaning | | |
| JMP <addr> | Effective address → PC | | |
| EXT or INX addressing | | REL addressing (8-bit: -128→ +127) | |
| | | Long branches are available(16-bit -32,768 to +32,767) | |
| No flag affected | | | |

4. Flow Control

Loop primitive instructions

Counter branches

Branch condition: zero/nonzero value in a counter (A, B, D, X, Y, or SP)

Pre-decrement, pre-increment, or test-only

| Test and branch | | Decrement/Increment and branch | |
|--|---|--|--|
| Instruction | Meaning | Instruction | Meaning |
| TBEQ reg,<9-bit offset> TBNE reg,<9-bit offset> | Test counter and branch if zero/nonzero | DBEQ reg,<9-bit offset> DBNE reg,<9-bit offset> IBEQ reg,<9-bit offset> IBNE reg,<9-bit offset> | Decrement counter in <i>abdxys</i> and branch if zero/nonzero Increment counter in <i>abdxys</i> and branch if zero/nonzero |
| REL addressing (9-bit: -256 → +255) | | | |
| No flag affected | | | |

4. Flow Control

Subroutine Jump & Return

| Branch to Subroutine | | Jump to Subroutine | |
|--------------------------------------|---|--|---|
| Instruction | Meaning | Instruction | Meaning |
| BSR <8-bit offset> | <ul style="list-style-type: none"> •Effective address → PC •Return address→ stack | JSR <addr> | <ul style="list-style-type: none"> •Effective address → PC •Return address→ stack |
| REL addressing | | DIR, EXT, INX addressing | |
| Return from Subroutine | | | |
| Instruction | Meaning | | |
| RTS | Top of stack (2 bytes) → PC | | |
| Call Subroutine (in extended memory) | | Program Page register (PPAGE) included | |
| Instruction | Meaning | | |
| CALL <addr/page> | <ul style="list-style-type: none"> •Return address & PPAGE→ stack (3 bytes) •Effective address→ PC •page→ PPAGE register | | |
| EXT , INX addressing | | | |
| Return from Call | | | |
| Instruction | Meaning | | |
| RTC | Top of stack (3 bytes)→ PPAGE & PC | | |
| No flag affected | | | |

5. Miscellaneous

(1 of 4)

Stacking

| PUSH | | PULL | |
|--|--|--|--|
| Instruction | Meaning | Instruction | Meaning |
| PSHA PSHB PSHD PSHC PSHX PSHY | <ul style="list-style-type: none">•Decrement SP accordingly by 1 or 2•register content→ stack | PULA PULB PULD PULC PULX PULY | <ul style="list-style-type: none">•stack→ register•Increment SP accordingly by 1 or 2 |
| INH addressing | | | |
| No effect on flags except for: PULC which affects all flags S X HI NZVC, BUT X flag is not cleared by operation | | | |

5. Miscellaneous

(2 of 4)

CC Instructions

| Set Flag(s) | | Clear Flag(s) | |
|-------------|---------|---------------|---------|
| Instruction | Meaning | Instruction | Meaning |
| SEC | 1 → C | CLC | 0 → C |
| SEV | 1 → V | CLV | 0 → V |
| SEI | 1 → I | CLI | 0 → I |

INH addressing

Remember from logical instructions:

- ORCC <IMM> to set some bits in CCR
- ANDCC <IMM> to reset some bits in CCR

Remember from data movement instructions:

- TAP to transfer (A) to CCR

Remember from stacking instructions:

- PULC to pull top of stack to CCR

5. Miscellaneous

(3 of 4)

Interrupt instructions ... Stop & Wait

| Software Interrupt | | Return from interrupt | |
|---------------------------------------|---|-----------------------|--|
| Instruction | Meaning | Instruction | Meaning |
| SWI | <ul style="list-style-type: none"> •Stack Y, return address, X, D, and CCR, update SP accordingly. •1 → I flag •(SWI vector) → PC | RTI | Retrieve from stack CCR, D, X, return address, and Y, update SP accordingly. |
| TRAP | <ul style="list-style-type: none"> •Stack return address, Y, X, D, and CCR, update SP accordingly. •1 → I flag •(TRAP vector) → PC | | |
| SHI NZVC flags affected 0 → X flag | | | |
| INH addressing | | | |
| Stop | | Wait for interrupt | |
| Instruction | Meaning | Instruction | Meaning |
| STOP | If S = 0: <ul style="list-style-type: none"> •Stack return address, Y, X, D, and CCR, update SP accordingly. •Stop all clocks. •Wait for interrupt or reset. | WAI | <ul style="list-style-type: none"> •Stack return address, Y, X, D, and CCR, update SP accordingly. •Wait for interrupt or reset. |

5. Miscellaneous

(4 of 4)

Background & Null operations

| Instruction | Meaning |
|--|---|
| BGND | If BDM enabled, enter BDM; else resume normal processing |
| NOP | Null operation |
| INH addressing | |
| BRN <8-bit addr> LBRN <16-bit addr> | Branch never Long Branch never |
| REL addressing | |
| No effect on flags | |