# MCS-4 Instruction Set

## INSTRUCTION FORMAT

There are 2 types of instructions:
- 1 word instruction with an 8-bit code
- 2 word instruction with 16-bit code

The first 4-bit code is called OPR, the second 4-bit code is called OPA.

## SYMBOLS AND ABBREVIATION

| | |
|---|---|
| ( ) | the content of |
| → | is transferred to |
| ACC | Accumulator (4-bit) |
| CY | Carry/link Flip-Flop |
| ACBR | Accumulator Buffer Register (4-bit) |
| RRRR | Index register address |
| RRR | Index register pair address |
| $P_L$ | Low order program counter Field (4-bit) |
| $P_M$ | Middle order program counter Field (4-bit) |
| $P_H$ | High order program counter Field (4-bit) |
| $a_i$ | Order i content of the accumulator |
| $CM_i$ | Order i content of the command register |
| M | RAM main character location |
| $M_{Si}$ | RAM status character i |
| DB (T) | Data bus content at time T |
| Stack | The 3 registers in the address register other than the program counter. |
| ~ | Complement |

## MACHINE INSTRUCTIONS

| MNEMONIC | 1ST BYTE OPR OPA | 2ND BYTE OPR OPA | DESCRIPTION |
|---|---|---|---|
| NOP | 0000 0000 | - | No operation. |
| FIM | 0010 RRR0 | $D_2D_2D_2D_2$ $D_1D_1D_1D_1$ | Fetch immediate (direct) from ROM Data DDDD DDDD to index register pair location RRR.<br>**$D_2D_2D_2D_2$ → RRR0**<br>**$D_1D_1D_1D_1$ → RRR1** |

| | | | |
|---|---|---|---|
| SRC | 0010 RRR1 | - | The 8 bit content of the designated index register pair is sent to the RAM address register at $X_2$ and $X_3$. A subsequent read, write, or I/O operation of the RAM will use address. Specifically, the first 2 bits of the address designate a RAM chip; the second 2 bits designate 1 out of 4 registers within the chip; the last 4 bits designate 1 out of 16 4-bit main memory characters within the register. This command is also used to designate a ROM for a subsequent ROM I/O port operation. The first 4 bits designate the ROM chip number to be selected. The address in ROM or RAM is not cleared until the next SRC instruction is executed. The 8 bit content of the index register is unaffected.<br><br>1) When referencing a Data RAM data character (WRM, SBM, ADM, RDM):<br>**XXYY ZZZZ**,<br>where **XX** - 1 of 4 Data RAM chips within the Data RAM bank, previously selected by a DCL instruction,<br>**YY** - 1 of 4 registers within the Data RAM chip,<br>**ZZZZ** - 1 of 16 4-bit data characters within the register<br>2) When referencing a Data RAM status character (WR0, WR1, WR2, WR3, RD0, RD1, RD2, RD3):<br>**XXYY ZZZZ**,<br>where **XX** - 1 of 4 Data RAM chips within the Data RAM bank, previously selected by a DCL instruction,<br>**YY** - 1 of 4 registers within the Data RAM chip,<br>**ZZZZ** - these bits are not relevant for this reference<br>3) When referencing a RAM output port (WMP):<br>**XXYY YYYY**,<br>where **XX** - the port associated with 1 of 4 Data RAM chips within the Data RAM bank previously selected by a DCL<br>**YY YYYY** - these bits are not relevant for this reference<br>4) When referencing a ROM input or output port (WRR, RDR):<br>**XXXX YYYY**,<br>where **XXXX** - the port associated with 1 of 16 ROM's<br>**YYYY** - these bits are not relevant for this reference<br>**(RRR0) → DB (X2)**<br>**(RRR1) → DB (X3)** |

| | | | |
|---|---|---|---|
| JCN | 0001 $C_1C_2C_3C_4$ | $A_2A_2A_2A_2$ $A_1A_1A_1A_1$ | If the designated condition code is true, program control is transferred to the instruction located at the 8 bit address A2A2A2A2 A1A1A1A1 on the same page ROM, where JCN is located.<br>If the condition is not true the next instruction in sequence after JCN is executed.<br>**If $C_1C_2C_3C_4$ is true**<br>**$A_2A_2A_2A_2 \rightarrow P_M$,**<br>**$A_1A_1A_1A_1 \rightarrow P_L$,**<br>**$P_H$ unchanged**<br>**If $C_1C_2C_3C_4$ is false**<br>**$(P_H) \rightarrow P_H$,**<br>**$(P_M) \rightarrow P_M$,**<br>**$(P_L + 2) \rightarrow P_L$**<br>The condition bits are assigned as follows:<br>$C_1 = 0$ : Do not invert jump condition<br>$C_1 = 1$ : Invert jump condition<br>$C_2 = 1$ : Jump if the accumulator content is zero<br>$C_3 = 1$ : Jump if the carry/link content is 1<br>$C_4 = 1$ : Jump if the test signal is zero<br><br>**Jump = ~$C_1$*((ACC=0)*$C_2$+(CY=1)*$C_3$+~TEST*$C_4$) + $C_1$*~((ACC=0)*$C_2$+(CY=1)*$C_3$+~TEST*$C_4$)**<br><br>Exception: If JCN is located on words 0xFE and 0xFF of a ROM page, when JCN is executed and the condition is true, program control is transferred to the 8-bit address on the next page where JCN located. |
| FIN | 0011 RRR0 | - | Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR.<br>**($P_H$) (0000) (0001) → ROM address**<br>**(OPR) → RRR0**<br>**(OPA) → RRR1**<br><br>Exception: when FIN is located at address ($P_H$) 1111 1111 data will be fetched from the next page (ROM) in sequence and not from the same page (ROM) where the FIN instruction is located. That is, next address is ($P_H$+1) (0000) (0001) and not ($P_H$) (0000) (0001) |
| JIN | 0011 RRR1 | - | Jump indirect. Send contents of register pair RRR out as an address at AAAA AAAA.<br>**(RRR0) → $P_M$,**<br>**(RRR1) → $P_L$,**<br>**$P_H$ unchanged**<br><br>Exception: when JIN is located at the address ($P_H$) 1111 1111 program control is transferred to the next page in sequence and not to the same page where the JIN instruction is located. That is, the next address is ($P_H$+1) (RRR0) (RRR1) and not ($P_H$) (RRR0) (RRR1) |

| | | | |
|---|---|---|---|
| JUN | 0100 $A_3A_3A_3A_3$ | $A_2A_2A_2A_2$ $A_1A_1A_1A_1$ | Jump unconditional to ROM address $A_3A_3A_3A_3$ $A_2A_2A_2A_2$ $A_1A_1A_1A_1$.<br>**$A_1A_1A_1A_1 \rightarrow P_L$,**<br>**$A_2A_2A_2A_2 \rightarrow P_M$,**<br>**$A_3A_3A_3A_3 \rightarrow P_H$** |
| JMS | 0101 $A_3A_3A_3A_3$ | $A_2A_2A_2A_2$ $A_1A_1A_1A_1$ | Jump to subroutine.<br>The address of the next instruction in sequence following JMS (return address) is saved in the push down stack. Program control is transferred to the instruction located at the 12 bit address $A_3A_3A_3A_3$ $A_2A_2A_2A_2$ $A_1A_1A_1A_1$. Execution of a return instruction (BBL) will caused the saved address to be pulled out of the stack, therefore, program control is transferred to the next sequential instruction after the last JMS.<br>The push down stack has 4 registers. One of them is used as the program counter, therefore nesting of JMS can occur up to 3 levels.<br>**($P_H$, $P_M$, $P_L$) → Stack**<br>**$A_1A_1A_1A_1 \rightarrow P_L$,**<br>**$A_2A_2A_2A_2 \rightarrow P_M$,**<br>**$A_3A_3A_3A_3 \rightarrow P_H$** |
| INC | 0110 RRRR | - | Increment index register.<br>The 4 bit content of he designated index register is incremented by 1. The index register is set to 0 in case of overflow. The carry/link is unaffected.<br>**(RRRR) + 1 → RRRR** |
| ISZ | 0111 RRRR | $A_2A_2A_2A_2$ $A_1A_1A_1A_1$ | Increment index register, skip if zero.<br>Increment contents of register RRRR. Go to ROM address $A_2A_2A_2A_2$ $A_1A_1A_1A_1$ (within the same ROM that contains this ISZ instruction) if result ≠ 0, otherwise skip (go to the next instruction in sequence).<br>The accumulator and carry/link are unaffected.<br>**(RRRR) + 1 → RRRR,**<br>**If result = 0**<br>**($P_H$) → $P_H$,**<br>**($P_M$) → $P_M$,**<br>**($P_L$ + 2) → $P_L$;**<br>**If result ≠ 0**<br>**($P_H$) → $P_H$,**<br>**$A_2A_2A_2A_2 \rightarrow P_M$,**<br>**$A_1A_1A_1A_1 \rightarrow P_L$**<br><br>Exception: if ISZ is located of words 0xFE and 0xFF of a ROM page, when ISZ is executed and the result is not zero, program control is transferred to the 8-bit address located on the next page in sequence and not on the same page where ISZ is located. |

| | | | |
|---|---|---|---|
| ADD | 1000 RRRR | - | Add contents of register RRRR to accumulator with carry.<br>The 4 bit content of the designated index register is added to the content of the accumulator with carry. The result is stored in the accumulator. The carry/link is set to 1 if a sum greater then 0xF was generated to indicate a carry out; otherwise, the carry/link is set to 0. The 4 bit content of the index register is unaffected.<br>**(RRRR) + (ACC) + (CY) → ACC, CY** |
| SUB | 1001 RRRR | - | Subtract contents of register RRRR to accumulator with borrow.<br>The 4 bit content of the designated register is complemented (ones complement) and added to content of the accumulator with borrow and the result is stored in the accumulator. If a borrow is generated, the carry bit is set to 0; otherwise, it is set to 1. The 4 bit content of the index register is unaffected.<br>**(ACC) + ~(RRRR) + (CY) → ACC, CY** |
| LD | 1010 RRRR | - | Load contents of register RRRR to accumulator.<br>The previous contents of the accumulator are lost. The 4 bit content of the index register and the carry/link bit are unaffected.<br>**(RRRR) → ACC** |
| XCH | 1011 RRRR | - | Exchange contents of index register RRRR and accumulator. The carry/link bit is unaffected.<br>**(ACC) → ACBR,**<br>**(RRRR) → ACC,**<br>**(ACBR) → RRRR** |
| BBL | 1100 DDDD | - | Branch back (down 1 level in stack) and load data DDDD to accumulator.<br>The program counter (address stack) is pushed down one level. Program control transfers to the next instruction following the last jump to subroutine (JMS) instruction.<br>The 4 bits of data DDDD stored in the OPA portion of the instruction are loaded to the accumulator.<br>BBL is used to return from subroutine to main program.<br>**(Stack) → $P_L$, $P_M$, $P_H$;**<br>**DDDD → ACC** |
| LDM | 1101 DDDD | - | Load data DDDD to accumulator. The previous contents of accumulator are lost. The carry/link bit is unaffected.<br>**DDDD → ACC** |

# INPUT/OUTPUT AND RAM INSTRUCTIONS

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| WRM | 1110 0000 | - | Write the contents of the accumulator into the previously selected RAM main memory character. The accumulator and carry/link are unaffected.<br>**(ACC) → M** |

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| WMP | 1110 0001 | - | Write the contents of the accumulator into the previously selected RAM output ports. (Output lines)<br>The data is available on the output pins until a new WMP is executed on the same RAM chip. The content of the accumulator and the carry/link are unaffected.<br>**(ACC) → RAM output register** |
| WRR | 1110 0010 | - | Write the contents of the accumulator into the previously selected ROM output ports. (I/O lines)<br>The data is available on the output pins until a new WRR is executed on the same chip. The accumulator content and carry/link are unaffected.<br>**(ACC) → ROM output lines** |
| WR0 | 1110 0100 | - | Write the contents of the accumulator into the previously selected RAM status character 0.<br>The accumulator content and carry/link are unaffected.<br>**(ACC) → $M_{S0}$** |
| WR1 | 1110 0101 | - | Write the contents of the accumulator into the previously selected RAM status character 1.<br>The accumulator content and carry/link are unaffected.<br>**(ACC) → $M_{S1}$** |
| WR2 | 1110 0110 | - | Write the contents of the accumulator into the previously selected RAM status character 2.<br>The accumulator content and carry/link are unaffected.<br>**(ACC) → $M_{S2}$** |
| WR3 | 1110 0111 | - | Write the contents of the accumulator into the previously selected RAM status character 3.<br>The accumulator content and carry/link are unaffected.<br>**(ACC) → $M_{S3}$** |
| SBM | 1110 1000 | - | Subtract the previously selected RAM main memory character from accumulator with borrow.<br>The RAM character is unaffected.<br>**~(M) + (ACC) + ~(CY) → ACC, CY** |
| RDM | 1110 1001 | - | Read the previously selected RAM main memory character into the accumulator.<br>The carry/link is unaffected. The 4-bit data in memory is unaffected.<br>**(M) → ACC** |
| RDR | 1110 1010 | - | Read the contents of the previously selected ROM input port into the accumulator. (I/O lines)<br>The carry/link is unaffected.<br>**(ROM input lines) → ACC** |
| ADM | 1110 1011 | - | Add the previously selected RAM main memory character to accumulator with carry. The RAM character is unaffected.<br>**(M) + (ACC) + (CY) → ACC, CY** |
| RD0 | 1110 1100 | - | Read the previously selected RAM status character 0 into accumulator. The carry/link and the status character are unaffected.<br>**($M_{S0}$) → ACC** |

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| RD1 | 1110 1101 | - | Read the previously selected RAM status character 1 into accumulator. The carry/link and the status character are unaffected.<br>**$(M_{S1}) \rightarrow ACC$** |
| RD2 | 1110 1110 | - | Read the previously selected RAM status character 2 into accumulator. The carry/link and the status character are unaffected.<br>**$(M_{S2}) \rightarrow ACC$** |
| RD3 | 1110 1111 | - | Read the previously selected RAM status character 3 into accumulator. The carry/link and the status character are unaffected.<br>**$(M_{S3}) \rightarrow ACC$** |

# ACCUMULATOR GROUP INSTRUCTIONS

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| CLB | 1111 0000 | - | Clear both. (Accumulator and carry)<br>**$0 \rightarrow ACC$,**<br>**$0 \rightarrow CY$** |
| CLC | 1111 0001 | - | Clear carry.<br>**$0 \rightarrow CY$** |
| IAC | 1111 0010 | - | Increment accumulator.<br>No overflow sets the carry/link to 0; overflow sets the carry/link to a 1.<br>**$(ACC) + 1 \rightarrow ACC$** |
| CMC | 1111 0011 | - | Complement carry.<br>**$\sim(CY) \rightarrow CY$** |
| CMA | 1111 0100 | - | Complement accumulator.<br>The carry link is unaffected.<br>**$\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0 \rightarrow ACC$** |
| RAL | 1111 0101 | - | Rotate left. (Accumulator and carry)<br>The content of the accumulator and carry/link are rotated left.<br>**$CY \rightarrow A_0$,**<br>**$A_i \rightarrow A_{i+1}$,**<br>**$A_3 \rightarrow CY$** |
| RAR | 1111 0110 | - | Rotate right. (Accumulator and carry)<br>The content of the accumulator and carry/link are rotated right.<br>**$A_0 \rightarrow CY$,**<br>**$A_i \rightarrow A_{i-1}$,**<br>**$CY \rightarrow A_3$** |
| TCC | 1111 0111 | - | Transmit carry to accumulator and clear carry.<br>The accumulator is cleared. The least significant position of the accumulator is set to the value of the carry/link. The carry/link is set to 0.<br>**$0 \rightarrow ACC$,**<br>**$(CY) \rightarrow A_0$,**<br>**$0 \rightarrow CY$** |

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| DAC | 1111 1000 | - | Decrement accumulator.<br>A borrow sets the carry/link to 0; no borrow sets the carry/link to a 1.<br>**(ACC) - 1 → ACC** |
| TCS | 1111 1001 | - | Transfer carry subtract and clear carry.<br>The accumulator is set to 9 if the carry/link is 0.<br>The accumulator is set to 10 if the carry/link is 1.<br>The carry/link is set to 0.<br>**9 → ACC if (CY) = 0**<br>**10 → ACC if (CY) = 1**<br>**0 → CY** |
| STC | 1111 1010 | - | Set carry.<br>**1 → CY** |
| DAA | 1111 1011 | - | Decimal adjust accumulator.<br>The accumulator is incremented by 6 if either the carry/link is 1 or if the accumulator content is greater than 9. The carry/link is set a 1 if the result generates a carry, otherwise it is unaffected (it is not reset).<br>This instruction is used when adding decimal numbers.<br>**If (ACC) > 9 or CY = 1**<br>**(ACC) + 6 → ACC**<br>**If result produces carry**<br>**1 → CY**<br>**Otherwise**<br>**CY unaffected**<br>**Otherwise**<br>**ACC unaffected** |
| KBP | 1111 1100 | - | Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code. The carry/link is unaffected. The conversation table is shown below:<br>**If (ACC) = 0b0000**<br>**0b0000 → ACC**<br>**Else if (ACC) = 0b0001**<br>**0b0001 → ACC**<br>**Else if (ACC) = 0b0010**<br>**0b0010 → ACC**<br>**Else if (ACC) = 0b0100**<br>**0b0011 → ACC**<br>**Else if (ACC) = 0b1000**<br>**0b0100 → ACC**<br>**Else**<br>**0b1111 → ACC** |

| MNEMONIC | 1ST BYTE | 2ND BYTE | DESCRIPTION |
|---|---|---|---|
| DCL | 1111 1101 | - | Designate command line.<br>The content of the three least significant accumulator bits is transferred to the command control register within CPU.<br>This instruction provides RAM bank selection when multiple RAM banks are used. (if no DCL instruction is sent out, RAM Bank number 0 is automatically selected). DCL remains latched until is changed.<br>The selection is made according to the following truth table.<br>**(ACC) Bank No.**<br>**X000 0**<br>**X001 1**<br>**X010 2**<br>**X011 3**<br>**X100 4**<br>**X101 5**<br>**X110 6**<br>**x111 7** |