MC Assembler Specification

MC Microprocessor Development

Revision 0.1

December 30, 2022

Table of Contents

[Changelog 3](#_Toc123388688)

[Dependencies 4](#_Toc123388689)

[Installation 5](#_Toc123388690)

[Command Line Information 6](#_Toc123388691)

[MCAsm Command 6](#_Toc123388692)

[Options 6](#_Toc123388693)

[asm Sub-Command 6](#_Toc123388694)

[Options 6](#_Toc123388695)

[Arguments 6](#_Toc123388696)

[dis Sub-Command 6](#_Toc123388697)

[Options 6](#_Toc123388698)

[Arguments 6](#_Toc123388699)

[Instructions 7](#_Toc123388700)

[Instruction Format 7](#_Toc123388701)

[Opcode Table 7](#_Toc123388702)

[Instruction Set 7](#_Toc123388703)

[MC Assembly Language 9](#_Toc123388704)

[Statements 9](#_Toc123388705)

[Instructions 9](#_Toc123388706)

[Directives 9](#_Toc123388707)

[Labels 9](#_Toc123388708)

[Other Syntax Rules 9](#_Toc123388709)

[Comments 9](#_Toc123388710)

[Whitespace 9](#_Toc123388711)

[Multiline Statements 10](#_Toc123388712)

# Changelog

|  |  |
| --- | --- |
| **Version** | **Commentary** |
| 0.1 | Initial draft of specification |

# Dependencies

Java 17 or above - <https://www.oracle.com/java/technologies/downloads/>

PicoCLI – <https://www.picocli.info/> (added to lib automatically)

# Installation

The MC-Microprocessor series of programs can be installed via

> git clone https://github.com/JonathanUhler/MC-Microprocessor

Move the contents of MC-Assembler/bin to someplace in the system $PATH. Make sure the MCAsm script and lib directory are also included. Run MCAsm --help to get started.

# Command Line Information

## MCAsm Command

### Options

|  |  |
| --- | --- |
| **Option** | **Commentary** |
| -h, --help | Display help message and exit. |
| -V, --version | Display the version and exit. |

## asm Sub-Command

### Options

|  |  |
| --- | --- |
| **Option** | **Commentary** |
| -f, --force | Overwrites existing content in the output file, if -o is used. If not specified, content is appended to the output file. |
| -h, --help | Display help message and exit. |
| -o, --outfile <OUT\_FILE> | Specify the location of an output file. If not specified, output is printed to stdout. |

### Arguments

|  |  |
| --- | --- |
| **Argument** | **Commentary** |
| IN\_FILE | Input file with assembly instructions. |

## dis Sub-Command

### Options

|  |  |
| --- | --- |
| **Option** | **Commentary** |
| -f, --force | Overwrites existing content in the output file, if -o is used. If not specified, content is appended to the output file. |
| -h, --help | Display help message and exit. |
| -o, --outfile <OUT\_FILE> | Specify the location of an output file. If not specified, output is printed to stdout. |

### Arguments

|  |  |
| --- | --- |
| **Argument** | **Commentary** |
| IN\_FILE | Input file with assembled instructions. |

# Instructions

## Instruction Format

24 20 | 19 16 | 15 12 | 11 8 | 7 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| opcode | rw | ra | rb | imm |

opcode = bits[24:40] operation code for this instruction

rw= bits[19:16] register file address for writes

ra= bits[15:12] register file address for read on A bus

rb= bits[11:8] register file address for read on B bus

imm = bits[7:0] numeric literal value for specific instructions, otherwise ignored

## Opcode Table

**bits[24:23]**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **bits[22:20]** | | | | | | | | |
|  |  | **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
| **00** | OR | AND |  | NOT | CMP |  | ADD | SUB |
| **01** | ORI | ANDI |  | NOTI | CMPI |  | ADDI | SUBI |
| **10** | BRZ | BRO | MF |  |  |  |  |  |
| **11** | LD | ST |  |  |  |  |  | HALT |

Instructions in light gray are currently not supported for the following reasons:

* The HI and LO registers are not used, thus MFLO and MFHI are not used
* LD is not used because there is no memory to load from
* ST is not used because there is no memory to store to

## Instruction Set

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Operation** | **Assembler Format** | **Commentary** |
| OR | rw <-- ra | rb | or rw, ra, rb | Bitwise or of ra and rb |
| AND | rw <-- ra & rb | and rw, ra, rb | Bitwise and of ra and rb |
| NOT | rw <-- ~rb | not rw, rb | Bitwise not (invert) of rb |
| CMP | rw <-- ra == rb | cmp rw, ra, rb | Store 1 IF ra == rb, otherwise 0 |
| ADD | rw <-- ra + rb | add rw, ra, rb | Sum of ra and rb |
| SUB | rw <-- ra - rb | sub rw, ra, rb | Difference of ra and rb |
| ORI | rw <-- ra | imm | ori rw, ra, imm | Bitwise or of ra and imm |
| ANDI | rw <--ra & imm | andi rw, ra, imm | Bitwise and of ra and imm |
| NOTI | rw <-- ~imm | noti rw, imm | Bitwise not (invert) of imm |
| CMPI | rw <-- ra == imm | cmpi rw, ra, imm | Store 1 IF ra == imm, otherwise 0 |
| ADDI | rw <-- ra + imm | addi rw, ra, imm | Sum of ra and imm |
| SUBI | rw <-- ra - imm | subi rw, ra, imm | Difference of ra and imm |
| BRZ | IF rb[0] == 0, PC <-- imm | brz rb, imm | Branch to imm if bit[0] of register rb is 0; execute the next instruction in sequence before taking the branch |
| BRO | IF rb[1] == 1, PC <-- imm | bro rb, imm | Branch to imm if bit[0] of register rb is 1; execute the next instruction in sequence before taking the branch |
| MFHI | rw <-- RF[17] | mfhi rw | Store the value of register 17 in rw |
| MFLO | rw <-- RF[16] | mflo rw | Store the value of register 16 in rw |
| LD | *Not implemented* | *-* | *-* |
| ST | *Not implemented* | *-* | *-* |
| HALT | RETURN imm | halt imm | Halt the CPU with exit code imm |
| NOP | PASS | nop | No operation, equivalent to  or r0, r0, r0 |
| LI | rw <-- imm | li rw, imm | Load imm to rw, equivalent to  ori rw, imm |
| MOV | rw <-- ra | mov rw, ra | Copies the value in ra to rw, equivalent to ori rw, ra, 0 |
| BR | PC <-- imm | br imm | Unconditional branch to imm, equivalent to brz r0, imm |

# MC Assembly Language

## Statements

### Instructions

See “Instructions > Instruction Set” above for more information on syntax and arguments.

### Directives

Directives begin with the “.” identifier. The list of valid directives is as follows:

|  |  |
| --- | --- |
| **Directive** | **Commentary** |
| .loc <pc> | Short for “location,” sets the program counter to pc. Only used by the assembler. |
| .pragma imm <hex|dec|default> | Sets the disassembled type for immediate values. Should be one of the string literals {hex, dec, default}. Only used by the disassembler. Default values are usually hex, with the exception of {ADDI, SUBI}. |
| .label <name> <pc> | Defines a label with name name at pc pc. Only used by the disassembler. See “MC Assembly Language > Statements > Labels” for more information on the use of labels in the assembler. |

### Labels

In the context of the assembler, labels appear in the form “name: ”, which defines a label with name name beginning at the line it is defined on. Label names should end with the “:” identifier. Because this delimiter exists, a single instruction can follow a label’s definition in the form “name: instruction”.

## Other Syntax Rules

### Comments

Comments are allowed by MC-Assembler. A comment is defined by a string beginning with //. The leading // and any following characters are removed and ignored.

### Whitespace

All trailing and leading whitespace is ignored by MC-Assembler. Line indentation does not matter, and any mix of tabs, spaces, and other whitespace characters can be used to the same effect. However, each statement must be on a separate line, with the exception of labels.

### Multiline Statements

With the exception of labels (which can be on the same line as their first instruction), each statement **must** be on a separate line. Because multiline statements are not allowed, no line-ending character is needed.