Typesetting Conventions

To make the reading of this document easier, the following conventions are used.

- Example text is written in `monospaced`.
- <angle brackets and the text within> will be replaced with appropriate text.

Asm80 Assembler

How To Run

Execute the following in a command prompt.

```
asm80 <source file name>
```

How to Write

Comments

Everything between a semicolon and line break will be treated as a comment.

Literals

Decimal	Any numeral.
Hexadecimal	Numerals and letters A through F. Additionally, must either: 1. Start with a numeral and end in H. 2. Start with a dollar sign.
Character	Enclosed in either single or double quotation marks.

Symbols

The usable characters for symbols are letters, underscore, fullstop, question mark, at sign, and—after the first character only—numerals.

Operators

Types

Unary

`+`	Positive
`-`	Negative
`HIGH`	High byte
`LOW`	Low byte
`NOT`	Inversion (one's complement)

Binary

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`MOD`	Modulo
`SHL`	Shift left
`SHR`	Shift right
`AND`	Logical conjunction
`OR`	Inclusive disjunction
`XOR`	Exclusive disjunction

Order of Operations

Priority	Operator

1	`*`
	`/`
	`MOD`

2	`+` `-`
3	`SHL` `SHR`
4	`AND`
5	`OR` `XOR`

Directives

`INCLUDE`	Inserts a file.
`PUBLIC`	Makes referencing a symbol from another module possible.
`EXTERN` `EXT`	References a symbol of another module.
`CSEG`	Mark all after this directive as a code segment.
`DSEG`	Mark all after this directive as a data segment.
`EQU`	Defines a constant expression.
`DEFB` `DB`	Reserves constant byte(s).
`DEFW` `DW`	Reserves constant word(s).
`DEFS` `DS`	Secures a region of memory.

Control Flow Structures

IF-ELSE-ENDIF

Code	Function

`IF` <condition>
  <block 1>
`ELSE`
  <block 2>
`ENDIF`
	Executes <block 1> if <condition> is true、<block 2> if false. The ELSE block can be omitted.

DO-WHILE-WEND

Code	Function

`DO`
  <block 1>
`WHILE` <condition>
  <block 2>
`WEND`
	<block 1> executes, and then <block 2> executes and repeats as long as <condition> holds true.

DO-DWNZ

Code	Function

`DO`
  <block>
`DWNZ`
	Executes the <block> until the result of decrementing the B register is zero.

Peculiarities

Automatic Relative Jump Replacement

When the relative jump destination is out of bounds, it is replaced by an absolute jump.

Multiple Instructions

Writing multiple instructions per line is done by separating each with a vertical bar (|).

LinkLE Linker

How to Run

At the command prompt, type the following.

`linkle` <output file> <code segment address> <data segment address> <object filename(s)>

Segment Address

Multiple ranges can be specified for segment addresses, like `4000-4fff`, `2000`, etc.

Output File

By adding an extension, the format of the output file will change based on the following tables.

Formats Great For Emulators

Extension	Format
`CMT`	PC-8001
`P6`	PC-6001 (including loader)
`MZT`	MZ
`CAS`	MSX
`RAM`	PASOPIA (RAMPAK2)
`PRG`	JR-100
`CJR`	JR-200
`L3`	Basic Master 3
`CAS`	MSX
`T64`	Commodore64, VIC-20

Others

Extension	Format
`HEX`	Intel HEX
`S`	Motorola S record
Other	Contents of code segment as-is (without address information)

Authorship

Translated from the May 3, 2022 version of Inufuto's (https://github.com/inufuto/) original manual.

Translation and edits done by Christen Gottschlich (christen.got@gmail.com) and finished on August 18, 2023. Edits were added for clarity.