Cowgol, C and Z80 assembler development environment

hosted on Z80 computers running CP/M 2.2

User's manual

Szilagyi Ladislau, July 2025

Introduction

This development environment for Z80 computers running CP/M 2.2 allows the construction of CP/M applications, starting from source files written in the Cowgol, C and Z80 assembler programming languages.

Components taken from the HiTech C compiler v3.09 are used, plus the Cowgol compiler built by David Given.

The object code format is compatible with the HiTech C compiler's object code format, thus facilitating the possibility of mixing code written in Cowgol, C and Z80 assembler.

Operating Details

Under CP/M 2.2, a single command will compile, assemble and link into a CP/M 2.2 executable file several Cowgol and optionally, C and Z80 assembler source files.

The syntax of the command is as follows:

>cowgol [options] files

The options are zero or more options, each consisting of a dash ('-'), a single key letter, and possibly an argument, following the key letter with no intervening space.

The files are one or more Cowgol, C source files, Z80 assembler source files, or Cowgol, C or assembler object files.

The command will, as determined by the specified options, compile any Cowgol, C source files given, assemble them into object code unless requested otherwise, assemble any Z80 assembler source files specified, and then link the results of the assemblies with any object or library files specified.

Upper and lower case can be used to specify the options or file names, the command does not distinguish between cases.

The options recognized are as follows:

-C The files will be only compiled/assembled.

-Mfile Builds the "file.map" memory map file for the executable.

-Lfile Adds the file "file.lib" to the list of files used as input by the LINK linker.

-O Z80 assembler code optimizations will be performed

-Tfile Build a table of symbols named "file.sym", to be used when debugging the executable

-S Type to the console a map of the Cowgol subroutines and variables used in these subroutines, with their exact addresses. This is very valuable when debugging the executable.

-X Reserve a larger stack (1KB) for the executable (the usual stack size is 128 bytes). This might be important if re-entrant C code is mixed into the project.

If the option -C is not specified, the files will be first compiled/assembled, then linked into a CP/M executable (named after the first Cowgol file name in the list of the source files).

More than one source file may be specified (with extensions: .cow = cowgol source file, .c = C source file, .as = assembler source file)

Also, C or AS object files (with extension .obj) and Cowgol object files (with extension .coo) may be specified.

The first source file must be a Cowgol source file. If more Cowgol source files are used, the first one will give the name of the executable being built, but the last one must contain the 'main' routine. C and assembler routines may be called from Cowgol source files.

The following executables are needed:

- $EXEC.COM , the "batch processor" from the HiTech's C compiler, who launches all the subsequent executables from the Cowgol toolchain
- COWGOL.COM (a modified variant of the HiTech's C.COM), the component that interprets the command line and feeds into $EXEC run requests for the subsequent executables from the Cowgol toolchain
- COWFE.COM , the "cowgol front end", who parses the source file, part of the Cowgol compiler, optimized
- COWBE.COM , the "cowgol back end", who builds the "cowgol object file", part of the Cowgol compiler, optimized
- COWLINK.COM , the "cowgol linker", who binds all the "cowgol object files" and outputs a Z80 assembler file, part of the Cowgol compiler, optimized
- COWFIX.COM , interface to Z80AS , who performs also code optimizations
- Z80AS.COM, the Z80 assembler (see https://github.com/Laci1953/Z80AS), who assembles the output of Cowfix and any Z80 assembler source file included in the command line, producing HiTech compatible object files
- LINK.COM , the HiTech's linker, who builds the final executable, using as input the object files and, if requested, the library file and producing the final executable
- CPP.COM , the HiTech's C pre-processor (needed only when C source files will be compiled), modified to accept // - style comments
- P1.COM , the HiTech's C compiler pass 1 (needed only when C source files will be compiled)
- CGEN.COM , the HiTech's C compiler pass 2 (needed only when C source files will be compiled)
- OPTIM.COM , the HiTech's C compiler optimizer (needed only when C source files will be compiled)
- LIBR.COM , the HiTech's librarian (needed only to build and manage .lib libraries)

Also, the Cowgol object file "cowgol.coo", containing basic Cowgol run-time support routines must be present on the disk.

The sequence of execution of these components is:

Cowgol source file >> COWFE >> COWBE >> COWLINK >> COWFIX >> Z80AS >> .OBJ file
or *C source file >> CPP >> P1 >> CGEN >> OPTIM >>Z80AS >> .OBJ file*
or *Z80 assembler source file >> Z80AS >> .OBJ file*
then *.OBJ object files & .LIB library files >> LINK >> .COM file*

Some examples of compilation commands:

1. >cowgol -c hexdump.cow

The Cowgol "hexdump.cow" source file will be compiled to the Cowgol object file "hexdump.coo".

2. >cowgol testas.cow rand.as

The Cowgol "testas.cow" source file will be compiled, the Z80 assembler "rand.as" source file will be assembled, then the resulting object code files will be linked to the "testas.com" executable.

3. >cowgol -Llibc -O dynmsort.cow merges.c rand.as

The Cowgol "dynmsort.cow" source file will be compiled, the C source file "merges.c" will be compiled, the Z80 assembler file "rand.as" will be assembled, all with code optimization, then the resulting object code files and parts of the "libc.lib" C runtime library will be linked to the "dynmsort.com" executable.

4. >cowgol -o -s -tsymbols alloc.coo testmem.cow xrnd.obj

The Cowgol source file "testmem.cow" will be compiled, with code optimization, then the resulting object code will be linked with the "alloc.coo" and "xrnd.obj" to the "testmem.com" executable. The Cowgol subroutines and variables will be typed to the console, and a file named "symbols.sym" will be created, to facilitate the debugging of the executable.

5. >cowgol -o misc.coo string.coo seqfile.coo startrek.cow

The Cowgol source file "startrek.cow" will be compiled, with code optimization, then linked to the "misc.coo", "string.coo" and "seqfile.coo" to the "strartrek.com" executable.

6. >cowgol -o misc.coo string.coo ranfile.coo advent.cow advtrav.cow advmain.cow

The Cowgol source files "advent.cow", "advtrav.cow" and "advmain.cow" will be compiled, with code optimization, then linked to the "misc.coo", "string.coo" and "ranfile.coo" to the "advent.com" executable.

Compiling large Cowgol source files

Because of the limited TPA's size, on Z80 computers provided with 64KB RAM, some large Cowgol source files fail to compile ("not enough memory").

For Z80 computers provided with 128/512 (or more) KB RAM, including RomWBW systems, to compile these large Cowgol source files, an enhanced COWFE.COM is provided, able to allocate part of the structures used by the compiler to the extra RAM space, beyond the usual 64KB (see https://github.com/Laci1953/Cowgol_on_CP_M/tree/main/128_512_KB).