C ports of PL/M 80 and Fortran applications

This repository contains my ports of several PL/M 80 applications to C and my port of the old Fortran based PL/M80 compiler, again to C.

There is also a historical port of the ISIS-II PL/M 80 compiler to C++, directly done from the disassembled code, i.e. before I reversed engineered the compiler into PL/M 80 source.

Originally the code was distributed as part of my Intel80Tools repository, however I have refactored into its own repository as a cleaner option, as the number of ports increases. Prebuilt Windows 32bit binaries of these tools (except the historic C++ port) are still distributed as part of the Intel80Tools repository.

See the doc directory for information on each of the applications, including usage differences from the original applications.

Visual studio solution files are provided for all the tools, along Linux Makefiles.

Note if you get a warning message when building files, it may be because you are using an older or possibly newer version than I have set the project to. Visual Studio provides simple options to retarget to any version you have.

There are also a three of script files from my versionTools repository in the Scripts directory. Of these getVersion.cmd and getVersion.pl are the main ones that creates the version numbers. The other install.cmd is used to auto install Windows executables to your desired directory. This can be replaced with your own installation scripts. For Linux all the built files can be found in the subdirectory Linux/Install, from here you can install to your required directory.

Recent major changes

4-Jul-2023

Lib 2.1 has basically been rewritten, the major changes are

- Like link and locate it now supports long command lines.
- CREATE has been enhanced to allow the library to be created **with** a list of filenames and library modules to be included as part of the create. When this is done any old library file is overwritten unlike the basic CREATE command
- Lib supports a command line parameters with & at the end of the line allowing for additional input from stdin. In this case lib exits after the command
- When used without command line parameters, the utility enters an interactive mode where multiple commands can be issued. These also support long line and & continuation.
- A new HELP command has been added
- Under windows lib adds setargv.obj to support file name wild cards. See the notes below about a feature added to make use of this.

Other changes

- Restrictions on file names have been eased and improved for command line arguments

- When processing file names or text within parentheses, where a single item is expected, leading and trailing spaces are removed. Note, a file name cannot contain a) character.
- When processing file names or text, the string can be enclosed in single quotes. All text inside the quotes is accepted including leading and trailing spaces. Note a single quote cannot be included.
  Apart from asm80, the quoting model works for all non punctuation tokens, see examples below, however in most cases it isn't needed, even for file names. For asm80 the quotes only work for filenames.
- When entering text on the command line, it is recommended to include either the parentheses or quote options inside double quotes to ensure that the OS shell does not change embedded spaces.

  Examples of file name processing:

```
When expecting a single item
PRINT( my file's.map )       uses "my file's.map"
PRINT( 'my (new) file.map' ) uses "my (new) file.map"
from the command line
"PRINT(my file's map)"       uses "my file's.map"
-p "'my file's.map'"         uses "my file" and will likely error on
s.map
-p "'my (new) file.map'"     uses "my (new) file.map"

When not expecting a single item
LIST plm80.lib(@PRMSK,@PSMSK)   @PRMSK & @PSMSK processed as expected

Quoting of tokens
'PRINT'  (  'my file'  )  is ok
'PRINT' '(' 'my file' ')' is not as punctuation should not be quoted
```

- Note, embedded control characters are mapped to space, so they can still not be used, although non ascii characters should work.

- Module names are now checked, with underbar '_' being added to the valid characters allowed.
- For the command line only, for lib, link and locate, as an extension a single ! toggles between inserting spaces or commas between arguments. The principle reason for this is that entering a comma between multiple arguments prevents the use of wildcards on file names in lib. However it can make entering lists for link and locate easier. The switch to comma is after the next argument, the switch back to a space is immediate. For example

```
lib c libname with *.obj    - will fail with more than one object file
matching *.obj
lib c libname with ! *.obj  - will insert commas between the file names and
work
lib a ! *.obj ! to libname  - will only comma separate the *.obj files
```

Caution: The location where commas and spaces are added is based on what the OS shell interprets as an argument, for example

```
link ! a.obj b.obj "my.lib(" MOD1 MOD2 ")" ! ... would be passed as
    link a.obj,b.obj,my.lib(,MOD1,MOD2,) ...       - probably not as intended
link ! a.obj b.obj "my.lib( MOD1, MOD2)" ! ... would be passed as
    link a.obj,b.obj,my.lib( MOD1, MOD2) ...       - as intended
link ! a.obj b.obj "my.lib(" ! MOD1 MOD2 ! ")" ... - would also work
Note the use of quotes around the my.lib( and ) is to avoid bash shell trying
to process the ( or ).
```

27-Jun-2023

Locate 3.0 has been updated, the major changes are

- Like link, locate now supports long command lines and simple response files. In addition several Windows/POSIX style options are now supported. The-h, option shows a summary of the options. The ORDER and segment address options remain as per Intel.
- File name handling and restrictions are as per link.
- Large chunks of code were removed as there is no longer a need to page partial located files to disk.
- Checking for COMMON names has been simplified by caching the information.
- Code common to Link and Locate has been refactored so that if bugs are discovered in the common code, both tools can benefit.
- Memory management uses malloc/realloc/strdup/free, with appropriate wrappers.
- For both Link and Locate, IoError now includes any file mapping.
- As with Link, the output file is deleted on error, to help with make builds.
- The code should now work on big endian processors.
- Two new options have been included NOEXTERN and NOOVERLAP. These cause unresolved externals and overlapping segments to be treated as errors, rather than just warnings. These are mainly to support make style builds

Note both link and locate have been successfully tested against the code in my intel80tools repository. Handling of COMMON hasn't yet been extensively tested, nor has the forced overlap of Segments.

PLM82 has had extensive changes to simplify and to facilitate the move to use structures rather than integer arrays for the symbol & information tables. Part of this was driven by a problem with the optimising compiler with very large functions. This may be a result of "undefined behaviour".

16-Jun-2023

Recent changes have been around link 3.0, major changes are

- Command line can now be any length, all control characters entered other than new line ('\n') are converted to a space. For invoke and errors long commands lines are formatted to a sensible width.
  As per the Intel code an ampersand '&' indicates additional command line text comes from stdin, which may be from a redirected file. For some OS, the '&' will need to be escaped on the command line, so as a convenience invoking link with no arguments starts input from

stdin without the need for the command line '&'. This allow support for a simple response file by using **link <file**.

- Some support for Windows/POSIX style options has been added. Use link -h for a summary. Note the -v, -V and -h options are only supported directly from the command line.
- An additional option -w (or WERROR) has been added to cause unresolved, multiply defined and COMMON length conflict warnings to be treated as errors. This is to help with make usage and would typically be used on the final link stage. Earlier partial links e.g., for overlays, should not use the option.
  Related to this is that the target file is deleted on error. Without this make can get confused as it sees a generated file, even though it is invalid.
  Note other possible warnings are given if 'MAP', if requested. These include overlaps, gaps and code that exceeds page limits. The listing file should be consulted to review these.
- As with asm80, native filenames (including directory prefixes) can be used and optionally prefixed by :Fx: where x is a digit. There are filename limitations, specifically:
  - filenames cannot contain a space, parenthesis, comma or ampersand. As with the asm80 port, path names may contain these characters providing the path is mapped to a :Fx: drive using environment variables.
  - filenames are not checked for legality, so may report the error when trying to open or create a file.
  - For windows the check for the output file name matching an input file name is done case insensitive. ***Warning the compare is only done on the entered name, not any mapped name, so aliases (e.g. absolute vs. relative paths) and links to the same file will not be detected.***
  - publics is not allowed as a file name as it conflicts with the publics(...) option. Note publics with an extension is allowed.
  - The checking that a filename is a disk file has been removed. It was previously only done on the :xx: device name. Failures to open or seek a file will be detected later.
    Whilst it would be possible to re-add file checking, using stat, it would need to handle the create file case where null devices and non-existent files are allowed, possibly checking that the parent directory exists. For now the benefit of adding this is seen as minimal.
- Module names can now contain the underbar '_' character.
- The use of an externals temporary file has been eliminated
- I/O now uses stdio functions rather than raw read/write.
- The library scan functions now cache the dictionary and offsets information, avoiding repeated reloads
- Input record processing has been simplified, as loading even long records is reasonable given modern memory limit.
- Memory management now uses malloc/realloc, except for the fixups, which are managed in a dynamically grown array.
- System error messages have been converted to use standard messages and application messages have been made more explicit, allowing for tailored messages rather than generic error codes.
- The code should now work on big endian processors.

31-May-2023

- Modified the listing files to include the date and time of the assembly in the header, if there is room, which will usually be the case. The default header width is now 80 unless the page width is smaller. The date is shown in the format [yyyy-mm-dd hh:mm].
- The Symbol Cross Reference now uses the common new page functionality so page numbers don't restart from 1
- Symbol and Cross Reference information that cross page boundaries how add a sub heading to indicate that the information is continued.
- Fixed a bug in the column and row variables which were still left as byte, i.e. up to 255, even though the page width and length could be larger

30-May-2023

ASM80

- The macro file is now simulated in memory, removing the external file.
- The underbar character is now accepted within labels
- Macro and Line buffers now auto grow as required.
- The combined length of nested macro arguments has been increased to 4096 and the arguments now grow upwards in memory rather than down. Changing to support dynamic growth would require many changes so the simpler option is to allocate a large array.
- Code has been added to replicate the format of the object file to match the original assembler, provided no label > 6 chars is used.
- The header lines have been modified to reflect that they are no longer ISIS-II and to enable long module names to be centred in the available space.
- The optional MACROFILE drive can be entered but it is not checked and is ignored.
- Some redundant functions and variables have been removed. Additionally some previously defined global variables have been made static, or function level variables.
- I have increased the size of the token stack to 20, this will allow longer db/dw lines
- The checking of whether a variable is local has been improved. Now only labels of the form ?? nnnn, where nnnn is an auto generated four digit decimal number, are considered local.
- PageLength and PageWidth can now be set up to a value of 65535, with 0 an alias for 65535. Existing lower limits still apply.

Other changes

Some of the Windows build has been fixed. Certain configurations broke when I added Linux support.

19-May-2023

- Labels up to 31 characters are now supported and as per PL/M embedded $ can be used in labels and numbers to make them easier to read. The $ is excluded from the name. Symbol listings, xref listings and object files all support the longer length variables.

- In order to support the changes the symbol table now holds a pointer to a C string rather than a packed double word. Related to this the symbol table memory management has been modified.
    - A maximum of 8192 symbols are supported.
    - ~~The maximum combined length of nested parameters is currently 1024 characters (+ length bytes)~~
    - ~~The maximum line length of macro body is currently 512 bytes~~
    - The keyword table has been modified to use a hash table generated by gperf.

      If you get table errors let me know.
- ~~Due to the longer labels, I have had to make some changes to the checks on when to generate a new OMF85 record. The original made an assumption that the names were up to 6 characters. A consequence of this is that symbol splits across records may be different. The generated object files will be equivalent.~~
- With the extended label name, it is possible that some old code will generate undefined errors. This is because in the original the names were truncated to 6 characters instead of 31, e.g. RESTART and RESTAR will be the same on the original assembler but flag an error now.
- I have re-enabled MACROFILE, which defaults to on. To turn off use the NOMACROFILE option. This will allow reserved names to be used for older code.
  ~~Note support for the optional (drive), has been removed and if used will generate an error.~~

16-May-2023

I have done the first part of a major rework of asm80, the key changes are noted below. If you find any problems please let me know.

- The I/O has been significantly reworked to use stdio buffering including adding support for Linux/Unix files. This has allowed most of the internal buffering code used in the original PL/M to be removed. For now some buffering around macros remains, but the underlying I/O has been modified. The changes also considerably simplify the handling of nested include files
- Related to the above, most of the error messages now use native error messages and write to stderr.
- The intermediate file required for xref support has been eliminated.
- The intermediate file currently still used for macro processing, now uses a system allocated temporary file, enabling the support for parallel invocation of asm80.
- Limits on the command line length have been removed as have limits on the source line length, however for listing purposes very long lines are truncated.
- Native filenames, including directory paths are now supported up to a limit of 260 characters. The change also supports the use of a :Fx: prefix (x is 1-9), which is replaced with the value of the associated environment variable :Fx: if it exists, otherwise the empty string is used. Note filenames including the directory path cannot include spaces or the ')' character. By using the :Fx: prefix it is possible to work around this for the directory path. The 260 character limit can also be resolved using the same technique if the OS supports longer paths.
- Support for ISIS devices e.g. :CO: and :BB: has been removed. Native equivalents can be used or for :BB: by using the NO option prefix.

- For the auto generated default .lst and .obj files a check is made on whether the file name part is upper case only. If it is then uppercase extensions .LST and .OBJ are used instead.
- The MACROFILE option is disabled as it is no longer needed. All assembly is done with macros enabled and the use of a system temporary file.
  Note, although previously undocumented, as the assembler always uses the macro version the key words macro and local, cannot be used as standard labels.
- Option MOD85 is now set by default. |The NO prefix for MOD85 is now supported to allow 8080 mode to be set.
- Initial work has been done to remove dependencies on packed structure support and little endian data formats. Further checking is required to check for missed changes.
- A number of preliminary changes have been made, pending support for names up to 31 characters to match PL/M.

6-May-2023

- Fixed sequence point issue with asm80. The compiler legally swapped lhs and rhs for an & operation, but this had unexpected side effects.
- asm80 & plm80c now support le endian processors
  Note as the old C++ port of pl/m is depreciated, I do not intend to make it portable across non le endian processors. It does however now support build on C++17 and later.

4-May-2023

- I have replaced the versioning model with one that is simpler to generate. A description can be found in the document Scripts/versionTools.pdf.
- Related to the above I have standardised the inclusion of supplementary version information and it's display. Details can be found in Scripts/showVersion.pdf
- I have modified the licence information to reflect original owners and that the code is released for hobbyist use and academic interest.
- Linux build support has been added. To avoid source tree pollution, a separate directory Linux has been added. Within this are subdirectories one for each application. Whilst each application can be built individually, the Makefile in the Linux directory can be used to build all the applications.
  The Makefiles assume gnu make is being used and the use of the -j (parallel builds) is supported.
  As I test using WSL on Windows, due to performance issues of git access from WSL to Windows, the default build does not force a version check or timestamp. In development this is ok, for release the publish target can be used to force the check.
- A number of bugs were identified by Andrey Hlus and have been fixed. Thanks Andrey.
- The code has been significantly modified to allow clean builds with gcc -Wall and cl /W3.
  Note using cl /W4 creates lots of additional minor warnings which I have checked. Most of the warnings are about assigning a larger integer to a smaller one, adding the explicit casts would make the code much harder to read.
  Also note cl /Wall generates lots of additional information, e.g. which functions are in-lined.
- binobj, hexobj, lib, link, locate and objhex should now work on non little endian processors, let me know if you find problems. The work on fixing asm80 and plm80c has started.

- tex has been modified to work on Linux, although care is needed to ensure file names are the correct case on the command line and within tex input. The changes which impact Windows and Linux are
    - On the command line only a - can also be used as the parameter marker instead of $. This avoids shell expansion under Linux. I suggest avoiding filenames starting with $ or - to avoid any confusion.
    - getch, which is used during paging, now only enters raw input whilst getting the single character. This hopefully will allow console input during tex processing to use standard line handling.
    - kbhit has been stubbed. Both operating systems support control C to abort a program. Although this does not give the opportunity to continue, entering control C is unlikely to be accidental.
      A limitation on this is when using a key hit to pause output to a printer to say handle a paper jam. As modern printer interfaces are usually spooled, direct printer output will be uncommon. If it does occur, run tex with the $I command to start output from a given page.

```
Updated by Mark Ogden 4-Jul-2023
```