Reverse engineering tools

The notes below document the tools I have developed for reverse engineering.

Note with the exception of disintelLib, ngenpex, plm81 and plm82 all of the source code is also included in the repository.

These tools are located in the tools directory and for non script files, the source is in toolsrc unless otherwise noted in parenthesis.

Note most tools now support being invoked with -v and -V. The lower case version shows simple version information, the uppercase version provides additional git information to help identify the version. Both of these should be the only option on the command line.

aomf2bin.exe

This utility take an absolute omf85, omf86 or omf286 file and creates binary images suitable for a prom programmer. There is an ability to set the base address of the prom and, whether to pad to a prom boundary, with 0 or 0xff. Optionally separate files can be created for odd and even bytes

```
usage: aomf2bin -v | -V | option* infile [outfile | -o odd_outfile | -e
even_outfile]+
   supported options are
   -b address - sets base address of rom
   -p         - pads image to eprom boundary
   -v / -V    - show version info and exit
   -z         - sets uninitialsed data to 0 instead of 0xff
```

asm80.exe, locate.exe, lib.exe, link.exe, plm80.exe (c-ports)

C port of Intel's ISIS tool chain. The usage is as per Intel's documentation, see Cports.md for more information, including how directories are mapped to drives.

asm80x.pl

This is a perl wrapper script that provides long label name support to Intel's ASM80. It takes as input a file with extension .asmx along with the usual asm80 options. It updates the listing and object files to reflect the long names. Since it uses thames and asm80 v4.1, the special features of thames are supported. See thames.md

In addition to long names _ and $ are ignored in names, this allows names to reflect PL/M conventions. However a label beginning with $ is passed through unchanged and as is a single _ . If these are invalid asm80 will report the error.

Note names excluding the _ and $ can be up to 31 characters and are converted to uppercase.

Due to the implementation there are a small number of limitations and differences in behaviour

- For MACRO, IPR and IPRC long names are not supported in the text after the key word, however a long label before the opcode is supported
- For MACRO attempts to create a label of the form @nnnnn, where nnnnn is a decimal number, using concatenation features is likely to cause problems unless nnnnn is a large number.

- Include files are process before submission to asm80 since they may contain long label names.
- As a bonus, by using a label beginning _n where n is digit, it is possible to create label names in the that begin with a digit. These will appear in the object file if they are public or local symbols are requested.
- The symbol table in the listing file is reformatted to accommodate the longer file names. This may cause the last page to be longer than the standard page length. Also any set page width is ignored.

asmx.pl

An experimental perl wrapper adding long label names and structure support to asm80. This is not yet robust.

binobj.exe (c-ports)

A port of Intel's binobj utility. This one supports windows/unix filenames and does not support ISIS drive mapping.

```
usage: binobj -v | -V | binfile [to] objfile
```

delib.pl

Simple perl script to extract the contents of an Intel OMF85 or OMF86 library file into separate files.

```
usage: delib.pl library targetdir
```

disIntelLib.exe (source currently private)

A homegrown utility to auto disassemble an Intel omf85 library into individual files. During the disassembly, whether the original code was PL/M or ASM is noted and the extension named accordingly.

```
Usage: disIntelLib infile
```

dumpintel.exe

Dumps the detail of the content of omf85, omf51, omf96 and omf86 files. omf96 currently only shows the record types as I have no samples to verify. The others decode as per the intel specifications with some of the none Intel additions for omf86

```
usage: dumpintel -v | -V | objfile [outputfile]
```

fixomf.pl

Fixes checksums for omf files and optionally removed @Pnnnn publics from the file. Used to synthesise old unavailable plm compilers.

```
usage: fixomf.pl [-p] infile [outfile]
where  -p removes @Pnnnn publics
```

genpatch.exe

This is used to auto generate the patch files for obj2bin. They take as input the generated omf85 object file and the target binary file and generates the specified patchfile

```
usage: genpatch [-i] [-s] [-z] objFile binfile patchfile
where -i    indicates to interpret the bin file as an Intel .BIN file
      -s    reserved for future use to show strings as a trailing comment
      -z    by default obj2bin auto fills uninitialised data to 0 and the
patchfile
            assumes this. The option forces the 0 initialisation to be in
patchfile
```

hexobj.exe (c-ports)

A port of Intel's hexobj utility. This one supports windows/unix filenames and does not support ISIS drive mapping.

```
Usage: hexobj -v | -V | hexfile objfile [startaddr]
Or:    hexobj hexfile TO objfile [$] [START ( startaddr ) ]
startaddr replaces the start address specified in the hexfile.
Intel style address formats are supported
```

install.cmd

This is a windows batch file that is mainly used as part of the visual studio build process to auto copy compiled code to target directories. The master repository for this tool is my github repository versionTools

```
usage: install.cmd file_with_path installRoot [configFile]
       configFile defaults to installRoot\install.cfg

install.cfg contains lines of the form type,dir[,suffix]
  where type is the closest parent directory ending in debug or release on the
path
  to the name of the file to copy. The test is case insenitive.
  dir is the directory to install to; a leading + is replaced by installRoot
  suffix is inserted into the installed filename just before the .exe extension
  In both dir & suffix a $d is replaced by the current local date string in
format yyyymmdd
  and a $t is replaced by the current local time string in format hhmmss
  All lines where type matches the input file's directory name are processed

Example with install.cfg in the current directory containing the line
x86-Release,+prebuilt
x86-Release,d:\bin,_32

install . path\x86-Release\myfile.exe
copies myfile to .\prebuilt\myfile.exe and d:\bin\myfile_32.exe

Control lines are also supported and they change what files the control lines
apply to
Each control line's impact continue until the next control line
A control line starting with a + enables processing only for the list of files
after the +
One starting  with a - only enables processing for files not in the list
a file name of * matches all files so +* renables processing for all files
-* stops all processing until the next control line (of limited use)
```

isisc.exe [depreciated]

Compares files using Intel's BIN format. This is now depreciated and the equivalent capability can be achieved in one of two ways

1. Convert the files to OMF85 format using binobj, then using omfcmp to check for differences
2. Use obj2bin with a patch file to add the junk data at the end of the file and use omfcmp or fc /b to compare

```
Usage: isisc -v | -V | file1 file2
```

isisu.exe

This utility dumps the block ranges and start address from an Intel BIN format file. This is now of limited use as I tend to convert the BIN files to OMF and load directly into my disassembler, preserving the uninitialised data information.

```
Usage: %s -v | -V | file
```

### makedepend.pl [depreciated]

Creates a dependency include file for gnu make for plm or asm source. This is now depreciated since thames allows this to be created as part of the normal assembly / compilation similar to modern C compilers

```
usage: makedepend.pl target source
where target is the object file and source is the source file. The environment
variables ISIS_Fn are used to map drives :Fn: to directories.
The source file is scanned for include files.
```

mkisisdir.pl

Simple utility to create an ISIS.DIR file to test ixref. It fills ISIS.DIR with all files in the current directory matching the pattern ??????.???. It only includes enough information to allow ixref to work and does not fill in all of the normal ISIS.DIR content.

```
mkisisdir.pl
```

mkmake.pl [depreciated]

Simple perl script that does a partial job of translating an Intel .CSD build file into a makefile. It is depreciated since the makefile support has been significantly enhanced since this was written.

```
mkmake.pl
```

It processes all .CSD files in the current directory.

ml80.exe, l81.exe, l82.exe, l83.exe (c-ports)

C ports of the CP/M high level assembler ml80 which is available under the cpmsrc/ml80

```
ml80 file[.ext]             .ext defaults to .m80
l81  file
l82  file
l83  file
```

ngenpex.exe (source currently private)

My own implementation of the intel software tools utility genpex. This fixes a number of issues with the original which I have included in the itools directory along with the original documentation.

The tool takes a master database of public variables, literals, procedures, based and label declaration and a source plm file. It generates an included file with required external and literal definitions for the plm file to compile.

See the genpex.txt file in the itools directory for the main details. My changes are:

1. Text in simple strings are ignored e.g. 'ERROR x' does not cause the error procedure to be defined as an external
2. The pex file allows a minus prefix to force prevent a match
3. Procedure definitions optionally allow parameter names to be included by following the type indicator with a space, the name and a comma or trailing)
4. In genpex a trailing S indicates an array variable and a (1) size is written to the ipx file. ngenpex optionally allows a (size) where size can be number or a "literal", this is written to the ipx file. This allows the plm size, length and last functions to be used
5. Local variable names do not trigger external definitions
6. Literal or variable declarations before the $include line for the generated .ipx file take precedence
7. Optionally ngenpex will emit public definitions in the plm file into a .pub file to allow an initial pex database to be created. Note it will also emit externals with a # prefix. This allow you to cross check for missing / incorrect usage

   ```
   usage: ngenpex pexfile sourcefile [-p]
   where the -p is optional and generates the .pub file noted above
   ```

obj2bin.exe

This utility is designed to support the creation of .COM, .T0 and .BIN files and includes the ability to patch the resultant file. Patching is potentially needed for two reasons.

1. Intel's tools support uninitialised data but the .COM and .T0 are pure memory images. By default obj2bin will set these locations to 0, however the original binary production is likely to have used data in memory at the time of creation. Patching allows this random data to be matched
2. It is possible that the binary images have data after the end of the program to align with sector boundaries, The patch capability allows this to be added at the end of the file.

```
Usage: obj2bin -v | -V |  [-i] infile [patchfile] outfile
```

```
Where -v/-V provide version information
and   -i    produces Intel formast .BIN files

The patch file, if used  has the following format and operates in one of two
modes
PATCH the initial mode and APPEND which starts after the key word APPEND is seen
at the start of a line, the keyword is case insensitive. The two modes are
required
since for .BIN files in particular the extra data needs to occur after the start
record
Note for all lines leading space is ignored and all values are in hex.
The address used to apply the patch is determined as follows
PATCH mode:
    Each line starts with a hex address to start the patch, anything other than a
hex       value is seen the line is ignored

APPEND mode:
    Initially the address is set to the current highest used location when APPEND
is        seen there after the address increments for each new value appended

Once the patch address is known all other data is a set of space separated values
specifiers in one of the following
    value ['x' repeatCnt]
        where value can one of
        hexvalue
        'string'                 note string supports \n \r \t \' and \\ escapes
        -                        set to uninitialised
        =                        leave unchanged i.e. skip the bytes
Note in APPEND mode, - and = are teated as 0
If the value is not a valid hex value, string, - or = the rest of the line is
skipped
if the x is present and repeatCnt is invalid an error is reported

The above noted, it is safer to use a semicolon to start a comment as this is
treated
as the end of line
```

objhex.exe (c-ports)

A port of Intel's objhex utility. This one supports windows/unix filenames and does not support ISIS drive mapping.

```
usage: objhex -v | -V | objfile [to] hexfile
```

omfcmp.exe

This tool is designed to intelligently compare intel OMF85 files, however it will revert to comparing binary files.

```
Usage: omfcmp -v | -V | file1 file2
```

pack.pl

Manage a packed source file

```
usage: pack.pl [-h] [-a pattern | -c pattern | -d pattern | -l | -u] [-f] [file]

where -h            prints simple help and exits
      -a pattern    add text files matching pattern - also updates changed files
      -c pattern    create new packed file from text files matching pattern
      -d pattern    remove text files matching pattern
      -l            list names of included files
      -u            update files in existing packed file
      -f            files only no directories
      file          an optional target file - default is {curdir}_all.src

      default operation is -c "*"

      patterns are case insensitive ? matches any char * matches any number of
chars
      multiple patterns are separated by |
      [..] matches ranges of chars and spaces should not be escaped
      e.g. to match a file name with a space use "* *"
```

patchbin.exe [depreciated]

Patch a binary .COM file. The original reason for creating this is now manageable with obj2bin

```
Usage: patchbin -v | -V | patchfile filetopatch
where patch file has lines in the format
address value*
both address and value are in hex and the filetopatch is assumed to be load ax
100H
The file is extended if necessary
```

plmpp.exe

Only PL/M v4 supports a pre-processor. This utility provides a pre-processor for older versions of PL/M.

```
usage: plmpp -v | -V |  [-f] [-F] [-sVAR[=val]] [-rVAR] [-o outfile] srcfile
Where -f                - expands a level of include files, each -f does another
level
      -F                - expands all include files regardless of depth
      -sVAR[=val]       - same as PL/M's SET(VAR[=val])
      -rVAR             - same as PL/M's RESET(VAR)
      -o outfile        - specifies the output file, otherwise outputs to stdout
```

pretty.pl

plm81/plm82 compiler listing format does not interleave source and generated code. This utility processes the list file to interleave the source and code. It is not perfect because the interleave information is not 100% accurate. It is however reasonable.

```
usage: pretty.pl infile outfile
```

rebase.pl

Updates a listing file from ASM80 or PLM80 to remap code address locations. It is largely replaced by relst.pl. but it may have some use in partial builds scenarios.

```
usage: rebase.pl lstfile lstaddr realaddr
lstaddr is an address in lst file and realaddr is the real address
all addesses are adjusted to reflect the offset
```

relst-simple.pl (depreciated)

Takes a mapfile and a set of .lst files from a build and generated .prn files with the listing files updated to reflect the located addresses. relst.pl is the natural replacement

```
usage: relst.pl mapfile lstfiles
use the mapfile to adjust all of the specified listing files
the output files have the same name as each lstfile with a .prn extension added
```

relst.pl

This utility takes listing files generated as part of the build along with the located file and the map file and created a new set of files with both the location addresses and code bytes updated to reflect the located locations.

```
usage: relst.pl locfile mapfile lstfiles
where  locfile is the located application
       mapfile is the map file created during the build
       lstfiles are the .lst files created during the build (ASM and PLM)
The generated files have the same name as the lstfile but with .lst replaced bu
.prn
```

repack.pl

Updated the packed source file (*directory*_all.src) in the current directory with changed file content.

```
repack.pl
```

revisions.cmd

Shows revisions of not executable files with respect to the current repository.

Note external tools copies in from other repositories are likely to have different revisions numbers e.g. revisions.cmd, install.cmd and version.cmd come from the versionTools repository

unpack.exe, unpack.pl

These two files support extracting files form a packed source file. The perl variant will not extract the file if the contents are unchanged, this helps with makefiles as it reduces the number of rebuilds.

```
Usage: unpack -v | -V | [-r] [file]
       unpack.pl [-r] [file]
if file is not specified the default file is directory_all.src
where directory is current directory name
-r does a recursive unpack
```

version.cmd

This is used to generate version information from a git repository for visual studio builds. The master repository for this tool is my github repository versionTools

```
usage: version [-h] | [-q] [-f] [-a appid] [CACHE_PATH OUT_FILE]

 When called without arguments version information writes to console

 -h          - displays this output

 -q          - Suppress console output
 -f          - Ignore cached version information
 -a appid    - set appid. An appid of . is replaced by parent directory name
 CACHE_PATH  - Path for non-tracked file to store git version info used
 OUT_FILE    - Path to writable file where the generated information is saved

 Example pre-build event:
 CALL $(SolutionDir)scripts\version.cmd "Generated" "Generated\version.h"

 Note if the OUT_FILE ends in .cs an C# version information file is created
otherwise
 a C/C++ header file is generated.
```

```
Updated by Mark Ogden 10-Oct-2020
```