GCC 8080/Z80/64180 Toolset

Contents

Purpose

This document describes how to setup a tool set that will enable you to assemble and link 8080/Z80/64180 source code. This is necessary when working on some very old boards, and since these processors are now obsolete, it is not very easy to find a good tool set.

Where to Get the Tool Set

The assembler tools I found from this page:

http://www.z80.info/z80sdt.htm

Specifically, I ended up using ZCC compiler package (from this page also) – which also includes an assembler and linker - which can be directly downloaded here:

http://www.z80.info/zip/zcc096.zip

I also saved a copy on my server in case the original server is no longer accessible:

zcc096.zip

Limitations

This has many downfalls, but does work.

Some limitations are the linker will only link a few files – go over about 10 or so and it stops working. But it comes with source code so you can fix it.

The executables that come with it will NOT work on a 64bit PC.

I originally setup a Windows XP virtual machine to run the tools (which was a pain to do) but it later dawned on me that I could re-build the executables in Visual Studio. This was relatively simple once I looked into it. I've listed the issues below, otherwise it is pretty straightforward.

1. Need to lower the compiler warning level to '2', otherwise it flags all the 'old' style function definitions and safety violations.
2. Need to modify a bunch of files that include alloc.h (the old Borland version.) to malloc.h.

That's it! Now you can use the tools on 64bit Windows.

The tool itself does not technically support 8080 targets – other than by the fact that Z80 opcodes are backward compatible with 8080 opcodes. However, because the tool is a Z80 assembler it will not flag any issues if you use a Z80 opcode that is not supported in 8080 targets – this can be incredibly hard to debug!

Port to Support 8080 Targets

I modified the AS source code to:

- better support 8080 targets.
- Support '.warning' pre-processor option (it was missing)
- Extended input line size from 128 to 250. Any text past this point (like long comments) is treated as a new line which caused them to annoyingly show up as mysterious errors.

Basically, the 8080 support implements a new assembler command (.i8080 – seen at the top below) which causes any Z80 instructions that are not compatible with an 8080 target to be flagged (with 't' at left) and causes the assembly to fail with errors.

```
                            1         .title  Test of Z80 / HD64180 assembler in 8080 mode
                            2
     0000                   3       .i8080
                            4
     0055                   5         offset  =       0x55            ;arbitrary constants
     0020                   6         n       =       0x20
     0584                   7         nn      =       0x0584
                            8
                            9         ; notes:
                           10         ;       Leading 'a' operand is optional.
                           11         ;       If offset is ommitted 0 is assumed.
                           12
                           13         ;*************************************************************
                           14         ;add with carry to 'a'
     0000 8E               15         adc     a,(hl)                  ; 8E
t    0001                  16         adc     a,offset(ix)            ; (t) INVALID - DD 8E 55
t    0001                  17         adc     a,offset(iy)            ; (t) INVALID - FD 8E 55
     0001 8F               18         adc     a,a                     ; 8F
     0002 88               19         adc     a,b                     ; 88
```

This prevents you from accidentally using a non-8080 compatible Z80 mnemonic and wasting hours of time trying to find out why it doesn't work.

Note that the assembler still uses the Z80 mnemonic form – rather than the 8080 format instructions. I left it this way because who needs to remember TWO sets of obsolete assembler mnemonics.

This new modified AS assembler will still work with the rest of the tool set – including the linker.

Get the Latest Tool Set Here

Here is a zip file of the fully modified tool set development directory structure – including VS2010 build configurations. This includes all of the changes described above.

ZCC Development Folder

Example Batch File Implementation

Here is an example batch file that will allow you to build a basic assembler project:

```
set path=%path%;c:\z80\zcc\exewin32;.\Tools
echo on

asz80 -l powerup.asm
asz80 -l keyboard.asm
asz80 -l pic.asm
asz80 -l pit.asm
asz80 -l video.asm
aslink -i -m -x -b CODE=0x0068 -b RAM=0x2000 powerup.rel keyboard.rel pic.rel pit.rel video.rel
del bootrom.hex
ren a.ihx bootrom.hex
del bootrom.map
ren a.map bootrom.map
srec_cat bootrom.hex -Intel -o bootrom.bin -Binary
pause
```

This assumes that:

1. All your assembler files are in the same directory as the batch file.
2. That the assembler tool set (described in this document) is located in “c:\z80\zcc\exewin32”
3. That you have placed the srec.cat.exe (srecord-win32.exe) – and it’s associated DLLs - hex to binary tool files in the “.\Tools” folder.

The ‘pause’ at the end is so that (if invoked by double clicking on the icon) the user can see any error messages before the window disappears.

This generates:

1. Bootrom.hex – which can be burned into an EPROM.
2. Bootrom.bin – which can be downloaded into a PicoROM ROM emulator.