Z80CPM22.BAS
MMBASIC Z80 SIMULATOR

Contents

Quick Up and Running

1. Set the OPTION CONTROL 120 on the CMM2 to ensure all controls can be created.
2. Connect a terminal such as Tera Term to the CMM2. In Terra Term Set Up/Terminal menu ensure that the New Line settings for Receive and Transmit is set to CR (Not CR+LF).
3. From the unzipped folder copy the following files to the root of the CMM2 SD,
    a. Z80CPM22.bas
    b. Z80Sim_CSUB.INC
    c. BDOS.HEX
    d. BIOS.HEX
    e. CCP.HEX
    f. CPM_Disk_A.dat (the A: CP/M disk containing CP/M transient programs).
    g. CPM_Disk_B.dat (the B: CP/M disk containing some example programs)
4. Type run "Z80CPM22.bas" to start the simulator or use F2 in the file menu. This will automatically generate CPM_Disk_C.dat on the CMM2 CD card if it doesn't exist.
5. Select LOAD CP/M button in the Simulator display.
6. Select RUN button.
7. The terminal will display A> indicating that CP/M is running and the A drive is logged in.

8. Type DIR followed by RETURN key to show the A drive directory.
9. Enter B: to change to the B drive.
10. Type DIR followed by RETURN key to show the B drive directory.
11. Type CLOCK to run the clock program and an analogue clock will be displayed on the CMM2 video screen.
12. Press any key to exit the CLOCK and return back to CP/M.
13. Press CTRL Q to exit the program and return to the CMM2 MMBasic monitor.

Introduction

The MMBasic program, Z80CPM22.BAS, and its attached include file with the CFunction CSim simulates a Z80 microprocessor. Note this is a simulator and not an emulator so some of the hardware interfaces have not been implemented. The MMBASIC program can also load CP/M 2.2 and run it using a terminal connection such as Tera Term. The program can be run with or without a mouse, although a mouse is the easiest (see below).
Note that CP/M 2.2 source assembly files are in the public domain (see http://www.cpm.z80.de/license.html).

Overview

The simulated processor uses an integer array, Registers(11), for the Z80 registers and housekeeping memory and an integer array, Memory64k(8192), for the 64k of Z80 memory. Both Registers() and Memory64k() are parsed to the simulator CFunction. The MMBasic simulator program runs in a continuous DO loop to manage the screen buttons. The simulator can run in STEP mode, RUN TO mode or RUN mode. The CFunction also operates in a continuous loop when called and while active none of the buttons or key commands will be usable, refer to Simulator Operation below for exiting an assembly program. Entering CTRL Q exits the program and returns to the CMM2 MBasic monitor. Note that the CFunction is compiled with the option O0 set (no optimisation as the CFunction relies on the functions being at a set place in memory and not in-lined as part of optimisation).

Supplied Files

Refer to the Read Me document.

Compatible Programs

This simulator should run any CP/M 2.2 program if its terminal output is compatible with a program such as Tera Term. A program must be copied over to the CP/M disk, see below in Button Operation for details on how to do this. To run a program where you want to interface back to the CMM2 display then its best to assemble the program using the provided Windows Console assembler as that has an in-built mnemonic (MMB 0) to call CMM2 functions.

Exiting the Simulator

This can be done by:

1. Pressing the EXIT button (or its equivalent CTRL X) if not actually running an assembled program (or CTRL Q).

2. CTRL Q if running an assembled program. The CFunction checks for CTRL Q each millisecond in its continuous loop. Note CTRL C will NOT exit the program as CTRL C is reserved by CP/M for terminating a CP/M program

Simulator Operation

The Simulator consists of a MMBasic monitor program and a CFunction, CSim. The monitor program is the interface to the user and manages the Button/Keyboard presses and displaying the Z80 registers. The CFunction CSim manages the Z80 instruction fetch and their execution, Z80 assembly program counter incrementation, Z80 register structure update to reflect current instruction status, interrupt management and updating timing counters.

Note that interrupts (pin checks) are set OFF by default and will only be undertaken if the instruction MMB 0 is undertaken with Register A set to 91H (SET_INT_TIME) and HL register set to microseconds between interrupt checks (see CFunction and Assembly Interface below).

The CSim undertakes the above operations in an infinite loop with specific conditions allowing it to escape the loop and return to the MMBasic Simulator interface program. On each loop the following is undertaken by CSim:

1. Fetch the next instruction and increment the Program Counter.
2. If TRACE ON is activated command the MMBasic monitor to send out the Z80 status to the console (it is done prior to executing the Z80 instruction).
3. Carry out the Z80 instruction, including updating all necessary registers and memory.
4. Check if the EXIT_FLAG has been set, and if set return to the MMBasic Simulator interface program. The EXIT_FLAG can be set by the assembly program (MMB 0 with Register A set to 90H).
5. Check if RUN_TO and STEP flags set and if set return to the MMBasic monitor if the right conditions have been met (i.e. steps completed or address reached).
6. Check if a millisecond has elapsed and if so
    a. call the CMM2 RoutineChecks().
    b. check if CTRL Q entered in the keyboard and exit if needed.
    c. and update the millisecond counter.
7. Check any assembly program request flags to measure elapsed millisecond time.
8. Check if interrupt timer has elapsed and if elapsed check interrupt pins. Note the interrupt timer checks must be enabled, default is not to check.
9. Update the microsecond timer counter from the CMM2 microsecond counter (uSecTimer()).

The CSim CFunction is called by the MMBasic program in the following situations:

1. Initially at the start of the MMBasic program to set up the Z80 instruction pointer array to the instruction management C functions.
2. When any of the STEP, RUN TO and RUN modes are activated.

CSim will only exit its continuous loop and return to the MMBasic simulator interface in the following conditions:

1. If the required steps have been completed if running in STEP mode.
2. If the required address has been reached by the Program Counter if using RUN TO mode
3. An assembly program uses instruction MMB 0 with A Register set to 90H (EXIT_TO_MMBASIC).
4. If in CP/M and the Simulator program memory at location &HFFFF is set to &H44 ('D') and the CTRL C key is pressed. Note that CTRL C key (WARM BOOT) would normally re-load the CCP.HEX file and then return to the CP/M CCP monitor. If the MMBasic monitor detects the &H44 set correctly on a WARM BOOT it will force the Simulator into STEP mode and not load CCP, so once the required steps are completed it will return from CSim to the Simulator monitor.

Note that if an assembly program has a loop that never returns then the CSim will never return unless CTRL Q is pressed or power is removed.

Writing a Program

1. It is suggested to use the provided assembly source example files as a guide to understand how to write a program that can interface back to CMM2. An assembly program can be written in any text editor, for example Notepad++ is ideal and then assembled into Intel HEX format using the provided Windows console program *Z80 Assembler.exe* (or any other assembler). Refer to the included manual Z80MacAss.PDF for further details on how to use the assembler.
2. The assembly program can be written to run in the Z80 simulator if it starts at address 0 or as a CP/M program if its start address is 100 Hex (see below for details of how to create a CP/M COM file from a hexadecimal file).
3. Note that on exiting a program not running under CP/M do not use the standard ret (C9 code) as there is no assembly monitor, instead the last instructions should be as follows,

   When using the provided Z80 Assembler.exe:

   ```
   mvi a, 90H ; set A reg to hex 90, exit code
   MMB 0 ; will exit the CFunction and as such will return to
   ```

   MMBASIC loop.

   When using any other assembler:

   ```
   mvi a, 90H
   db D0H,00 ; same a 'MMB 0' machine code of D000
   ```

4. **NOTE**: Do not use the MMB 0 instruction with the A register set to 02 (CFunctions.h getConsole) as the CMM2 console buffer is checked for CTRL Q on each millisecond and if needed any chars found are placed in the Simulator buffer and as such there never will be a character returned directly from the CMM2 console buffer. Use the CP/M calls of CPM_CONST (console status) to see if any chars are in the Simulator buffer and then use CPM_CONIN to get it. See C Function and Assembly Interface.
5. The Simulator console buffer is only 12 chars long so to prevent wrap around and over writing extract the chars into your program as they are typed and don't rely on the buffer to store long lines.

Non-Mouse Operation
Buttons. The Title on each Button has a character that is underlined. If this character is pressed on the keyboard with the CTRL key it will perform the same operation as pressing the left mouse button whilst the cursor is hovering over the Button.
Display Input. Any display that can have data input to it will have one of its Title characters with a line above it (a 'hat'). Pressing the LEFT ALT key with a 'hat' character will be the same as left clicking the left mouse button mouse whilst the cursor is hovering over that display.
Hexi-decimal Key Pad. Pressing any keyboard number (0 to 9) or letter (A to F) whilst pressing down the RIGHT ALT key will be equivalent of left clicking the mouse whilst the cursor is hovering over that keypad Button.

Register Display Format
All register will display their hexadecimal value on the left side of the display and the bits (0 or 1) in the rest of the display. The PC (Program Counter) register will also display the Z80 assembly code and its mnemonic instruction at the current PC location.

Altering a Register or Display Box
Click on the required register or display box (or use the equivalent keyboard input as detailed in Non-mouse Operation) and its label will turn from Yellow to Green, with Green indicating it is selected for input. Using the mouse press the required value on the hexadecimal key pad, the USER INPUT display will reflect what the user has entered. When done press the OK button and the selected register or display box will be updated. Note the SP (Stack Pointer) register and PC register cannot be changed in this manual way. Pressing the RST Button will reset the PC register to 0000.

Entering A File Name
Either press the left mouse button whilst hovering over the FILE NAME box or press the left ALT plus N key. Once selected for input a green character prompt will flash in the file name box. Type in the file name and press enter. The file name will change colour from white to red indicating the file name has been accepted. Note that the ONLY edit character that is accepted is the back space key which will erase the last character typed in. The only file name characters accepted are alpha characters, numeric characters, underscore ('-') and file type delimiter period ('.').

Loading & Running a Program
Normally assembled programs are in Intel hex format and must be residing in the root directory of the CMM2 SD card.

1. Select the file name Text Box with the mouse/keyboard and enter the file name to load.
2. Press the LOAD HEX File and the file will be loaded into its correct memory location, normally this would start at address 0000 (org 000 in assembly).
3. The program can be run by the CFunction by selecting:

a. **STEP** Button. This will undertake a set number of instructions as specified in the STEP display (default is one). The required number of instructions to be undertaken can be changed by selecting the STEP display as specified in Altering a Register/Display Box. The program returns to the MMBasic program input loop on completion of the step(s).
b. **RUN TO** Button. This will run the program until the program counter (PC) matches the STOP ADDRESS. The STOP ADDRESS can be altered as per Altering a Register/Display Box. The program returns to the MMBasic program input loop once the stop address is reached.
c. **RUN** Button. Will commence running the loaded hex file from 0000 (unless CP/M was loaded then the PC register is set to FA00, which is the cold boot location). On program completion it will return to the MMBasic simulator interface program loop (assuming the correct exit code is in the assembly program, see Writing a Program section).

Notes

1. If doing any graphics in a program running directly from the simulator interface panel (i.e. ORG at 0 and not under CP/M) then start the program with the CTRL R key as pressing the RUN Button will result in the RUN Button being displayed in amongst your graphics. Running the MMB_CLR at the start of an assembly program should also erase any errant buttons being displayed.

Buttons Operation

LOAD CPM

This will load the three hexadecimal files that make up the CP/M operating system into the Z80 memory. A cold loader that is normally used to load CP/M into memory is not required. The files are:

- CCP.hex at address E400, (Console Command Processor).
- BDOS.hex at address EC06, (Basic Disk Operating System).
- BIOS.hex at address FA00, (Basic I/O System), tailored for CMM2.

Note that the CCP.hex and BDOS.hex are the exact same CP/M Version 2.2 assembly files released by Digital Research in January and February 1980. BIOS.hex is based on the Skeletal CBIOS supplied by Digital Research. These files where put in the public domain in 2001 and reconfirmed in July 2022, see http://www.cpm.z80.de/license.html. At the end of loading the PC (program counter) will be set to FA00 and so pressing RUN will start CP/M (or if is needed STEP, RUN TO). It is recommended that a console terminal like Tera Term is used to interface with CP/M. Note setting the STOP ADDRESS to 0000 and pressing the RUN TO button will exit the CFunction whenever CTRL C is pressed in CP/M or alternately setting memory location 0FFFFh to 44h will do the same but not reload CCP.hex. As part of loading CP/M the RST and interrupt vectors are reset to 0.

LOAD FILE

This will load a file into memory starting at &H100. It is normally used to transfer any file (e.g. TXT, DAT, COM, HEX or BAS) to the CP/M disk system. Its use is as follows,

1. Select the file name Text Box and enter the file name to load.

2. Press the LOAD FILE and the file will be loaded starting at memory location 0x100.
3. Press LOAD CPM button.
4. PRESS RUN button. The system will be running CP/M with the A> prompt.
5. Type SAVE XX Filename to save the file onto the CP/M drive, where XX is the file size in pages plus one (a page is 256 bytes (, i.e. Pages = 1+ (File Size/256)). A minimum of 2 pages is required (refer to the Digital Research CP/M Operating System Manual). The file will be saved on the CP/M disk. Note that the Digital Research Manual states that File Size/256 is the correct number of pages but on rare occasions some files I found would not transfer all bytes but adding one fixed this issue.

LOAD HEX

Use similarly to the LOAD FILE button but will load the binary data in the hexadecimal file at the memory locations as specified in the hexadecimal address field.

ZERO MEM

This will zero all the Z80 Registers and Memory.

RST

Reset the Program Counter back to 0000, clear the Simulator keyboard buffer and set all registers to 0.

STEP

Will undertake a single instruction (default) at the current Program Counter or how many instructions set in the STEP display Maximum of 255 (&HFF). **Warning** if your Z80 program changes Pages then comment it out if in STEP or RUN TO mode or you will not be able to step once the Page is changed.

RUN TO

Will run a program from the current Program Counter until the Program Counter matches the value set in the STOP ADDRESS display.

EXIT

Exit this Simulator program and return to the MMBasic monitor.

KEY PAD (0 to F)

For entering values into the various displays.

INC & DEC

Will increment or decrement the ADDRESS display value and cause the value in the MEMORY display to reflect the value at the indicated address.

OK

Will transfer the value in the USER INPUT display to the selected display (the one with the green coloured title).

TRACE

When activated, (Red LED Lit after pressing TRACE Button or CTRL A), this will cause the Z80 status to be displayed on the console display for each instruction when in STEP or RUN TO mode. Note the status displayed is <u>prior</u> to actually undertaking the instruction. The status is laid out on a single line and consists of,
Program Counter: - Instruction Code - Instruction mnemonic – Next 2 bytes after instruction – Flag Register – A Reg – BC Reg – DE Reg -HL Reg IX Reg – IY Reg – Stack Pointer Reg – A bracketed Byte that the BC Reg, DE Reg and HL Reg would point to if they were pointers.

Note:

1. Flag Reg. + indicated positive value in A Reg (bit 7), a small n indicates the N flag is set (bit 1). Using small n makes it easier to see when carry bit is set to no carry (i.e. nNC is bit 1 set and bit 0 not set).
2. Displaying the next 2 bytes after the instruction (appears directly after closing bracket ‘]’ of instruction mnemonic can assist in debugging where the instruction is undertaking a direct load, compare add etc.
3. Running in trace mode will significantly slow the simulator down, but if the program is a long one then use the RUN TO button to go to the place of interest then turn the TRACE ON and step through the program. Having STEP set to FF (255) has been found to be ideal in sorting out a program issue if the program is long.

CP/M Disks

The CP/M emulator can have up to 4 disks, each 256,256 bytes in size. CP/M 2.2 uses a disk file system that is not compatible with the CMM2 file system and as such a simulated disk system is used that is fully compatible with CP/M 2.2. This uses up to four 256,256 byte size files on the CMM2 SD card, i.e CPM_Disk_A.dat to CPM_Disk_D.dat. This CP/M file system is based on 77 tracks with each track having 26 sectors and each sector is 128 bytes. Note that the granularity of the file is based on a block. A block is 8 sectors or 1024 bytes, so the smallest CP/M file size is 1024 bytes. Directory Entries (quantity 64, each of 32 byte length or a total of 2 Blocks) are at track 0, sector 1 to 16. The CP/M system will manage all the interaction of file management through the call back to the MMBasic simulator program subroutine ‘MMCB’.
CPM_Disk_A.dat supplied in the zip file comes preloaded with the following files.
Transient programs supplied by Digital Research,

1. STAT.COM
2. ED.COM
3. DDT.com
4. DUMP.COM
5. LOAD.COM
6. PIP.COM
7. SUBMIT.COM
8. XSUB.COM

CPM_Disk_B.dat supplied in the zip file comes preloaded with some test files

Tips on Using CP/M

1. Refer to the Digital Research Manual under CP/M 2.X at this web site to get an understanding on how to use CP/M 2.2.
2. Terra Term connected to the console is ideal to interface to CP/M, however in Terra Term Set Up/Terminal menu ensure that the New Line settings for Receive and Transmit is set to CR (Not CR+LF).
3. If memory location &HFFFF is set to &H44 then will it cause CTRL C in CP/M to come back to the MMBASIC simulator panel program instead of CP/M normal warm boot of reloading CCP.hex. This can be useful for debugging.

4. Exporting Files from CP/M Disks. To Export a file off a CP/M disk to the CMM2 SD (SDA) card use the program EXPORT.COM. This creates or overwrites CPM_EXPORT file on the CMM2 SD card. This will only export a file off the currently selected disk drive. If for example EXPORT.COM and TEST1.TXT are on the A: drive then the below command will export TEST1.TXT,
   EXPORT SDA: TEST1.TXT
   If TEST1.TXT is on the B: drive then select B: drive and use the command,
   A:EXPORT SDA: TEST1.TXT (assumes EXPORT.COM is on A: drive)
   Note the MMBasic Simulator Interface program treats text file and binary files differently in how they are stored. If the CP/M file is type TXT, BAS, PRN, ASM or HEX the file gets exported as a text file else it will be exported as a binary file.
5. If using Microsoft MBASIC.COM ensure you have Caps Lock on as it directly reads the console in and won't recognise lower case file names. Refer to the examples on how to interface back into the CMM2 hardware using assembly calls. I'm unsure of the licence details on CP/M 2.2 Microsoft Basic, however it can be downloaded from this site.
6. If using Microsoft Basic compiler (BASCOM.COM) and Linker L80.COM ensure Caps Lock is on. Both of these programs can take a while if the Basic program is of moderate size. Note you cannot use CALL if using the compiler as it would be expecting the assembly file to be a .REL file to correctly link it. To get the millisecond time stamps set the Count Flag at address &HFFFE to 1 to get now time and to 2 to get elapsed time. The time is stored at &HFFF9 to &HFFFC (little endian and equivalent to 32 bits). Refer to the example. An example of the commands to compile and link a program in CP/M assuming Microsoft compiler is on the C: drive, the Basic source programs are on the B: drive and the user is logged into the C: drive are:
   a. BASCOM
      * B:MBCOMP.REL=B:MBCOMP.BAS/O
   b. L80 B:MBCOMP,B:MBCOMP/N/E

Simulator Speed Testing

A number of speed tests were undertaken that also demonstrate how to make calls into the CFunction routines from assembly or Microsoft Basic.

SpeedTst.asm. The assembly program SpeedTst.asm can be run from the main Simulator interface panel using LOAD HEX Button. It displays the elapsed time to undertake 50,000 loops of a program that uses 54 different instructions and then displays the time taken. This only checks out an indicative 54 of the possible 694 legitimate combinations of opcodes and operands. Comparing the times for a 4MHz Z80 in the Zilog Z80 User Manual then the simulator runs a bit over 2.5 times faster or approximately an average of a little over one microsecond per instruction.

MBCALL.BAS and MBUSR.BAS are Basic programs that can be run in CP/M using Microsoft Basic (MBASIC.COM). The programs are based on the TBS BASIC speed test.

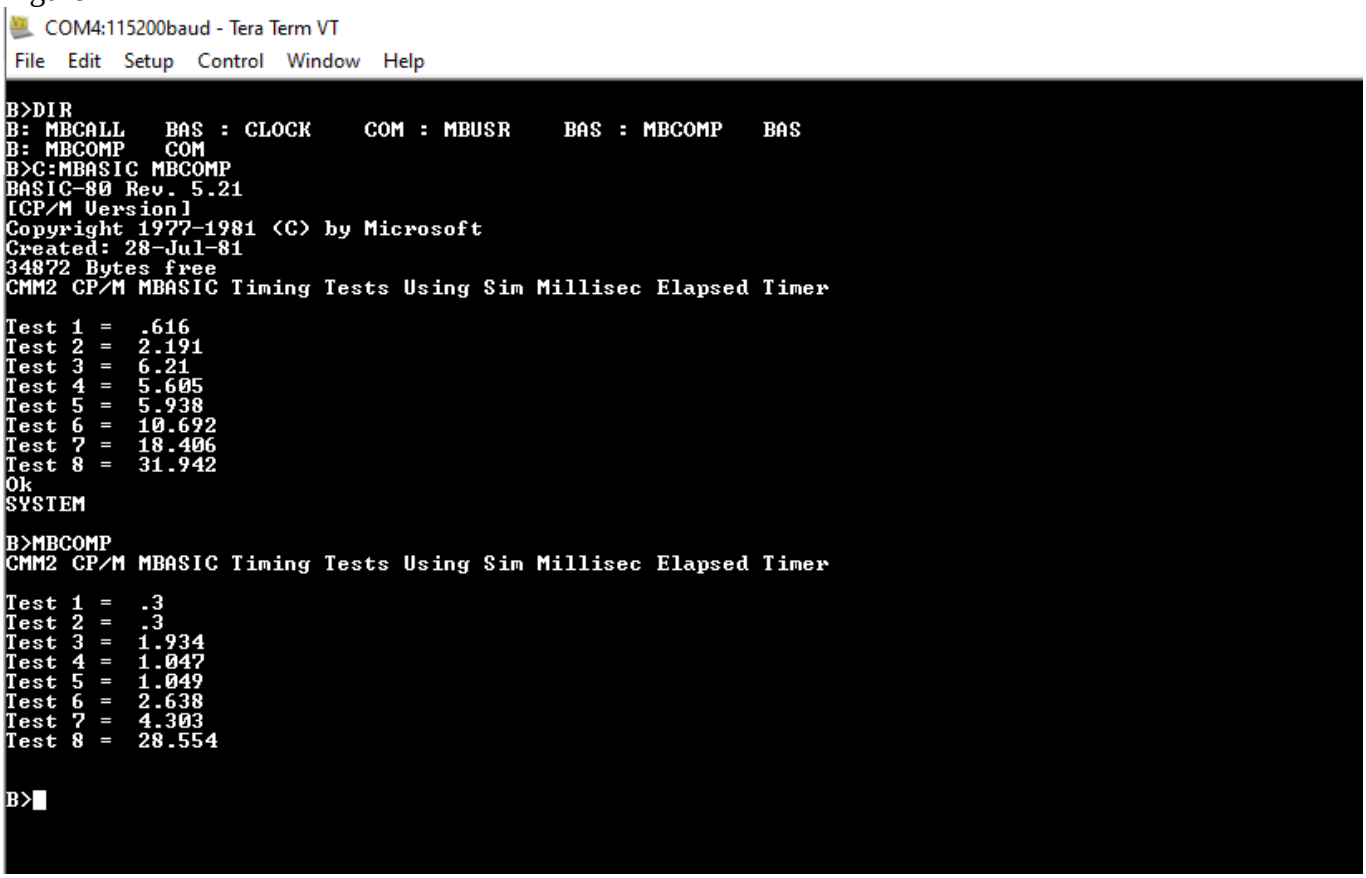
MBCOMP.BAS is a Basic program based on the TBS BASIC speed test that can be run from MBASIC.COM or compiled and linked using Microsoft BASCOM.COM and

L80.COM. Below (Figure 1) shows the times for the 8 tests running in MBASIC.COM and as a compiled program (MBCOMP.COM).

Useful Links

1. Micro Vibe.
2. The Unofficial CP/M Web site.
3. CP/M Internals.
4. John Elliotts's CP/m Main Page
5. Tim Olmstead Memorial Digital Research CP/M Library
6. Z80 Instruction Set.
7. Digital Research Background

Figure 1



CFunction and Assembly Interface

Assembly source can interface with most of the CMM2 functions as defined in ARMCFunctions.h and some additional routines that make the CP/M work. In all cases the A Register must be set to the routine number prior to using the special instruction mnemonic of MMB 0 if using the accompanying assembler (or the instruction byte of DB D0,00 if using other assemblers).

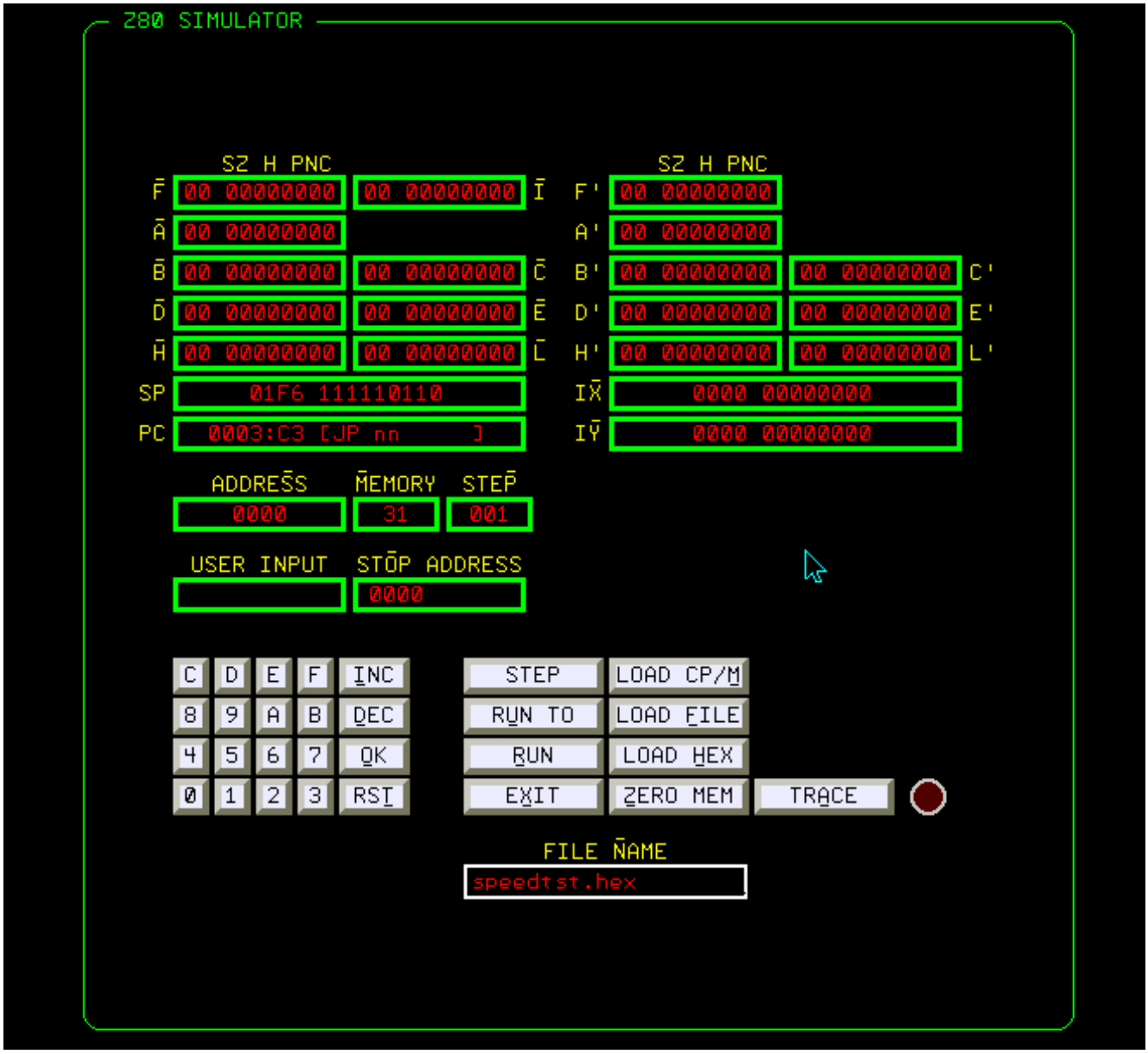
The routine number are:

0h: CFunctions.h uSec. HL points to the required 16 bit delay period.

1h: CFunctions.h putConsole. Register C contains the character

2h: CFunctions.h getConsole. Returns Register A with character. **WARNING, DO NOT CALL DIRECTLY FROM ANY ASSEMBLY CODE**. After each elapsed millisecond a check for CTRL Q is undertaken and hence there should never be a char in the CMM2 buffer as any char found is automatically checked for CTRL Q then placed in the Simulator buffer. Use the CP/M calls of CPM_CONST to see if any chars in Simulator buffer and then use CPM_CONIN to get the character.
3h: CFunctions.h ExtCfg. HL points to the first 16 bit parameter, then second parameter is next word (2 byte) in memory and third parameter is next word in memory.
4h: CFunctions.h ExtSet. HL points to the first 16 bit parameter, then second parameter is next word (2 byte) in memory.
5h: CFunctions.h ExtInp. C has pin number, Returns with HL containing input data.
6h: CFunctions.h PinSetBit. Register C has pin number, Register B contains second parameter (second bit, refer to ARMCFunctions.h).
7h: CFunctions.h PinRead. C has pin number, Returns with HL containing input data.
8h: CFunctions.h MMPrintString. Register DE pointing to zero terminated string
9h: CFunctions.h IntToStr. DE points to memory string location to be filled in pMemory, If C reg = 0 then HL is byte offset into registers where number stored, if C=1 then HL points to CP/M pMemory where number stored. B is second parameter, base.
Ah: CFunctions.h CheckAbort.
Bh: CFunctions.h DrawRectangle, HL points to X1, Y1, X2, Y2, Colour, all params 16 bit except colour which is 24 bits.
Ch: CFunctions.h DrawLine. HL points to a memory location laid out as 16 bit words of x1,y1,x2,y2,width,Color (16bits), Colour 8 bits , total use 13 bytes.
Dh: CFunctions.h HRes returned in HL.
Eh: CFunctions.h VRes returned in HL.
Fh: CFunctions.h. RunBasicSub. DE points to string with name (two 0's at end). This is fixed as 'MMCB\0\0'
10h: CFunctions.h. DrawPixel. HL points to parameter data, HL points to a memory location laid out as 16 bit words of x,y and colour and colour (byte). Total of 7 bytes.
11h: CFunctions.h uSecTimer. time value placed where HL pointing, note its 64 bit (8 bytes).
12h: CFunctions.h FastTimer. places value where HL pointing, note its 64 bit (8 bytes)
13h: CFunctions.h TicksPerUsec. places value where HL pointing, note its 64 bit (8 bytes).
14h: CFunctions.h. DrawCircle. HL pointing to start of parameters as
x, y, radius, w, c, fill, all params except colours 16 bit, colour 3 bytes. The aspect parameter is fixed as 1.00 so does not need to be set.
15h: CFunctions.h. DrawTriangle. HL pointing to start of parameters as,
x0, y0, x1, y1, x2, y2, c, fill, all params except colours 16 bit, colour 3 bytes.
21h: CPM_WARM_BOOT, called on CTRL C in CP/M. Reloads CCP.Hex or if 0xffff = 0x44 then just reverts to STEP MODE.
22h: CPM_CONST, Key board (console) status, returns FF in Register A if char in Simulator buffer else A Register is 0 if buffer empty.
23h: CPM_CONIN, The next console character is read into register A, if no char will wait so can hang until char entered.
24h: CPM_CONOUT, The character in register C sent to the console output device.

25h: CPM_LIST_OUT: The character in register C is sent to the console output device.
26h: CPM_PUNCH_OUT: Not Supported.
27h: CPM_READER: The next reader character (Console) is read into register A, if no char will wait so can hang.
28h: CPM_HOME: Set Disk to start of disk file.
29h: CPM_SEL_DISK: Select the disk, C Reg contains the disk number, A:=0, B: =1, C=3, D=4 (max 4 disks). If not available will return HL Register set to 0000 else return FFFF
2Ah: CPM_MOVE_TO_TRACK: Reg C contain the track.
2Bh: CPM_SETSEC, C Reg contains sector.
2Ch CPM_READ_DISK: A Reg = 0 if OK else 1. Data stored in DMA as set by CP/M.
2Dh CPM_WRITE_DISK: A Reg = 0 if OK else 1. Data sent from DMA as set by CP/M.
2Eh CPM_SET_DMA, HL contains address of DMA
40h Z80_SET_MODE_0_COMMAND Only used by Z80 to set IntMode_0_Code[3]
80F CPM_OPEN_EXPORT Opens MMBasic file CPM_EXPORT ready for writing.
81h CPM_TRANSFER_EXPORT Writes 128 bytes to CPM_EXPORT file from DMA at 80H.
82h CPM_CLOSE_EXPORT Closes CPM_EXPORT.
90h EXIT_TO_MMBASIC Return back to the simulator monitor.
91h SET_INT_TIME . Set interrupt time, HL = required microseconds, By default interrupts are not checked, this can be changed using this call.
92h START_TIMER Captures uSecTimer and stores in nElapseTimer of Registers (64 bit)
93h STOP_TIMER Captures uSecTimer and subtracts nElapseTimer to give elapsed time since Start Timer call.
94h PC_DEBUG_CNT. Only used for debugging this program to ensure PC counter is matches the compiled address, see switch case for details.
95h. MMB_PAGE_DISPLAY Calls CMM2 PAGE DISPLAY n with C Reg set to required PAGE
96h MMB_PAGE_WRITE Calls CMM2 PAGE WRITE n with C Reg set to required PAGE
97h MMB_CLR Calls MCMM2 CLS
98h MMB_TEXT_OUT Calls CMM2 TEXT, 'HL pointing to memory location laid out as,
Byte 0 & 1 = x, Byte 2 & 3 = y, Byte 4 = Alignment ('L', 'C', 'R') , Byte 5 = 'T', 'M', 'B', Byte 6 = orientation ('N', 'V', 'I','U','D', Byte 7 = Font, Byte 8 = Scale, Next 3 bytes for Colour and next 3 bytes for background colour. Register DE points to a String terminated in 0
99h MMB_GET_TIME, Calls CMM2 TIME$ but returns with C Reg = seconds, B Reg = Mins, E Reg = Hrs
9Ah DEBUG_PRINT_CREG. Can be called by assembly to print C Register as "C= XX"
9Bh DEBUG_PRINT_BCREG. Can be called by assembly to print BC Register as "BC= XX"
9Ch MMB_TRACE_REG_STATUS . Send the Z80 register status to console.
9Dh MMB_SAVE_IMAGE. Performs a SAVE IMAGE to ‘Z80Prog.bmp’.


Z80 SIMULATOR
SZ H PNC
SZ H PNC
F 00 00000000 00 00000000 I
F' 00 00000000
A 00 00000000
A' 00 00000000
B 00 00000000 00 00000000 C
B' 00 00000000 00 00000000 C'
D 00 00000000 00 00000000 E
D' 00 00000000 00 00000000 E'
H 00 00000000 00 00000000 L
H' 00 00000000 00 00000000 L'
SP 01F6 111110110
IX 0000 00000000
PC 0003:C3 [JP nn]
IY 0000 00000000
ADDRESS
MEMORY
STEP
0000
31
001
USER INPUT
STOP ADDRESS
0000
C D E F INC
8 9 A B DEC
4 5 6 7 OK
0 1 2 3 RST
STEP
LOAD CP/M
RUN TO
LOAD FILE
RUN
LOAD HEX
EXIT
ZERO MEM
TRACE
FILE NAME
speedtst.hex

CMM2 Screen