VRIJE UNIVERSITEIT BRUSSEL - ETRO

Computersystems: Session 1

David Blinder, Jan G. Cornelis, Stijn Bettens, Tim Bruylants and Peter Schelkens

October 8, 2020

1 DOSBox

DOSBox is a DOS machine emulator. It can emulate an Intel® 80386 CPU in both real and protected modes, handles XMS and EMS memory, provides emulation of the common graphics adapters used in DOS machines (CGA, EGA, VGA and VESA) and is able to emulate sound cards such as SoundBlaster^{TM} and UltraSound^{TM} . On top of this, it provides a modernized, but compatible, DOS emulation layer for the operating system.

The assembly language exercises and project for this course will take place inside the DOSBox emulated machine environment. This helps you to focus on learning x86 assembly language and machine designs, without having to worry about the extra complexities of today's modern hardware and software environments. For this purpose, we created a package called "ASMBox", that contains all the necessary tools for the practicum course.

1. Download the ASMBox zip-file from the course page at `https://canvas.vub.be/` in the folder "Practica/ASMBox + examples" and extract it.
2. Find and execute ASMBox.bat (for Windows)
3. In DOSBox, try the commands: `help`, `dir`, `cd <directory>`, `cd ..`, ...
4. The `c_disk` folder contains the content that is visible in DOSBox as a node of a tree with `C:\` as the root. When making updates to that folder from outside DOSBox (e.g. adding/removing files with your OS file explorer), then you should use `CTRL-F4` in DOSBox to notify its cache about the changes. This does NOT happen automatically!

Note: *We recommend using a virtual machine running Windows when using other operating systems. Otherwise, DOSBox is available for a variety of operating systems, including Mac OS or Linux. Students who prefer to directly work on operating systems other than Windows should visit `http://www.dosbox.com/` for more information and downloads. Regardless of the host operating system, all students are advised to use the OS independent ASMbox ISO from the aforementioned zip-file on Canvas.*

2 80386 32-BIT PROTECTED MODE

This practicum involves learning the Intel® assembly language for the 80386 CPU in 32-bit protected mode. Protected mode – also called protected virtual address mode – is an operational mode of the 80386 CPUs in which 32-bit instructions are enabled, and memory is virtualized into a linear address space. The advantage is that your code will be able to make use of more than 640kB of RAM, and that extra more (and faster) instructions are available. Furthermore, it avoids having to use the 16-bit segment/offset memory addressing mode. The assembler tool will provide the required code to switch the CPU to protected mode when executing your programs (if correctly configured).

Note: *The 80386 CPU is backward compatible with the 8086, 8088, 80186 and 80286 CPUs, but be aware that older assembly code might not work in 32-bit protected mode. So, always look for "DOS 32-bit" assembler, when searching online. Moreover, the so-called "Win32" assembly is similar, but uses entirely different OS-callback mechanisms.*

3 ASSEMBLER AND LINKER

The assembler and linker tools used in this course are Watcom Make, Turbo Assembler, Watcom Assembler and Watcom Linker. All students are obliged to make use of Watcom Makefiles for all assignments and the project. Examples can be found in `C:\EXAMPLES\` in the ASMBox environment.

Assembly source code is just text. As such, you can edit the code with the text editor of your choice. Good examples of (cross-platform) text editors are *Sublime Text 3*, *Notepad++* or *Vim*. Once more, we remind you to press `CTRL-F4` in DOSBox after adding/removing files

All examples are provided with their `Makefile`. The makefile contains the building instructions that allow the assembler and linker tools to handle your source code and construct the DOS executable. The tool to invoke the build process is called WMake (command `wmake` in DOSBox).

4 EXERCISES

1. Build and run the simple "Hello World!" program found in `C:\EXERCISES\HELLO\`. It uses function `09h` (in `AH`) of `int 21h`. Reference: `http://stanislavs.org/helppc/int_21-9.html`.

2. Write an if-then-else construct. Print to the screen a message that depends on the value of `EAX`, which can be 1 (true), 0 (false) or neither. Reference: `https://en.wikibooks.org/wiki/X86_Disassembly/Branches`.

3. Write a program that prints 10 times "Hello World!". Make use of branching and the `ECX` register for counting. Reference: `https://en.wikibooks.org/wiki/X86_Disassembly/Loops`.

4. Make a program that draws a pyramid of `*` symbols, given a height in the `EBX` register (see Fig. 4.1). Reference: `http://stanislavs.org/helppc/int_21-2.html`.

```
    *
   ***
  *****
 *******
*********
```

Figure 4.1: A pyramid of asterisks for `EBX = 5`

5 Advice

The best way to learn any assembly language is by means of hands-on-experience. We recommend you to read `_AssemblyProgrammingCompendium.pdf` and `_Overzicht_practica_computersystemen.pdf` in the folder `Handleidingen` on Canvas. These documents are written for students who are new to assembly coding and they will provide you with the necessary information to start writing simple programs for 80386 CPUs. Experiment with the instructions and observe their behavior by means of the debugger or by generating some textual output. You will definitely make mistakes in the beginning so start with short programs that allow you to identify the bugs quickly with the help of the error messages that pop up. As soon as you have a notion of what the most common error messages mean, you will have to spend less effort searching for bugs.