**1.**

**General**

The environment consists of several parts (the main folder for exercise was - C:\os-dev, code was at C:\os-dev\basic-os-16, all the tools at - C:\os-dev\env)

Source code, all scripts and compiling tools (tasm, tcc, tlink) reside on Win7 host machine.

Dosbox is used to compile and link the code to several separate binaries. DosBox has mounts to access compiling tools, source and compiling/linking script. The output is place on Win7 machine.

Ubuntu VM is used to create a single binary representing the contents of floppy or disk out of binaries produced by DosBox. The VM has access to the binaries on Win7 through shared folder. The output is place on Win7 machine.

DOS VM is used to run the code(any other VM that has floppy will work as well). It is using the output of Ubuntu VM on Win7 as a content of floppy.

Bochs is used to debug the code. It is using the output of Ubuntu VM on Win7 as a binary for debug.

**2.**

**Source code**

The code is placed to C:\os-dev\basic-os-16 (the src folder). Location is arbitrary.

**3.**

**Tools for compiling and linking**

Download tcc and tasm to C:\os-dev\env\tcasm. The folder should contain tasm and tc folders that have all the needed binaries. In this exercise I attach all those tools already in C:\os-dev\env\tcasm – no need to search on internet.

**4.**

**Dosbox – environment where the C and tasm code is compiled and linked**

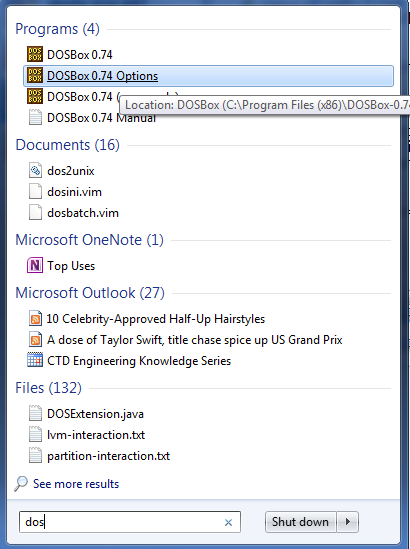
4.1

Download and install Dosbox on Windows (the following is used now DOSBox0.74-win32-installer.exe). Exists in C:\os-dev\env.

4.2

Configure Dosbox

Go to Start menu on Windows and select DosBox0.74 Options



Add the following to the end of the opened file ----

#####################################################

mount c: C:\os-dev\env\tcasm

mount d: C:\os-dev

mount s: C:\os-dev\basic-~1\src

set path=%path%;c:\tc\bin;c:\tasm\bin

D:

cd basic-~1

####################################################

So within the DosBox the drive c is mapped to C:\os-dev\env\tcasm on Windows. This folder on my local Windows contains all the needed tools for compiling and linking – tcc, tasm, tlink.

Drive d is mapped to C:\os-dev – the / basic-os-16/source subfolder of this directory contains all the source code. In configuration snippet above basic-~1 is used instead of basic-os-16 as basic-os-16 is too long for folder name per DosBox!

**5.**

**Compiling and linking the code**

In the DosBox (change one level up to in directory tree if needed) and run the script that

will do all the work ---

MYRUN.BAT

The result is placed in C:\os-dev\ basic-os-16\obj. The most important files are myboot.bin, KERN.BIN, SHELL.BIN, LS.BIN, HELLO.BIN, CALC.BIN.

**6.**

**Creating a final binary file**

The file will contain all the parts of the program – bootloader, kernel, shell, ls, hello, calc.

5.1

Prepare VirtualBox Ubuntu VM with C:\os-dev\ basic-os-16 shared folder. So this VM will have an access to all the code and scripts.

The shared folder should be auto mounted by VirtualBox under media

folder - /media/sf\_ basic-os-16. So change to /media/sf\_ basic-os-16 in Ubuntu and execute make command.

5.2

The result of “make” is C:\os-dev\basic-os-16\obj\mydisk.img

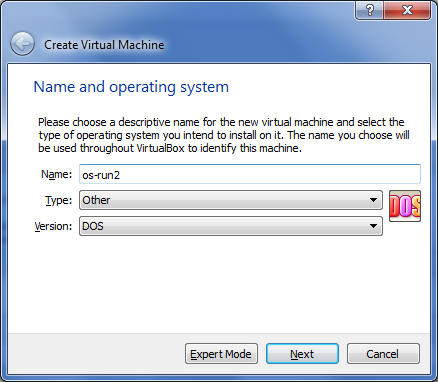
**7.**

**Running the code in DOS VM**

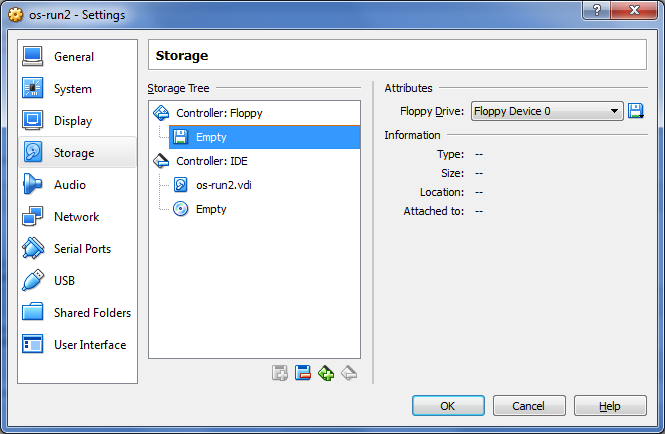
7.1

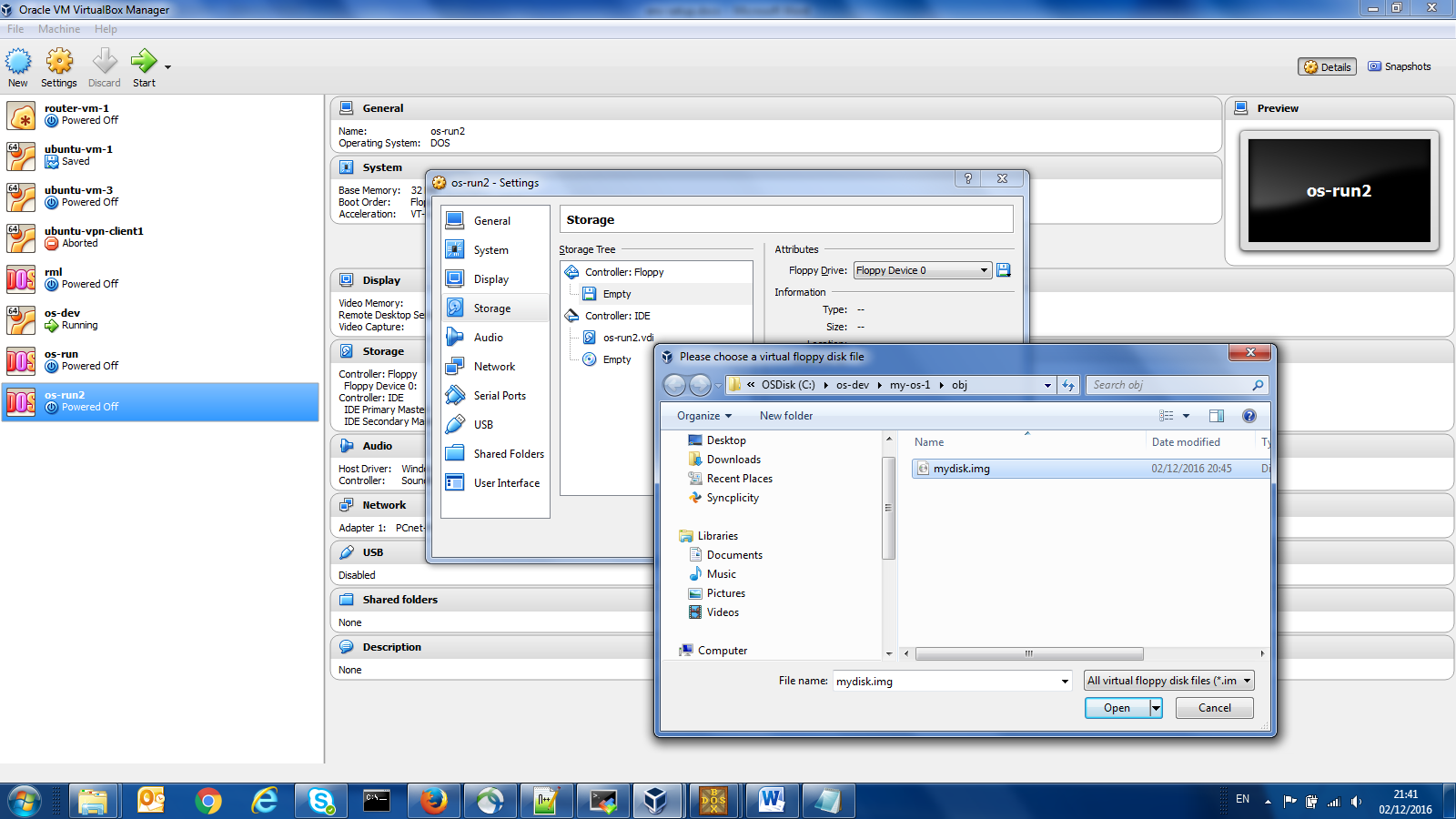
DosBox setup

Create the DOS VM – select version DOS like here when creating VM

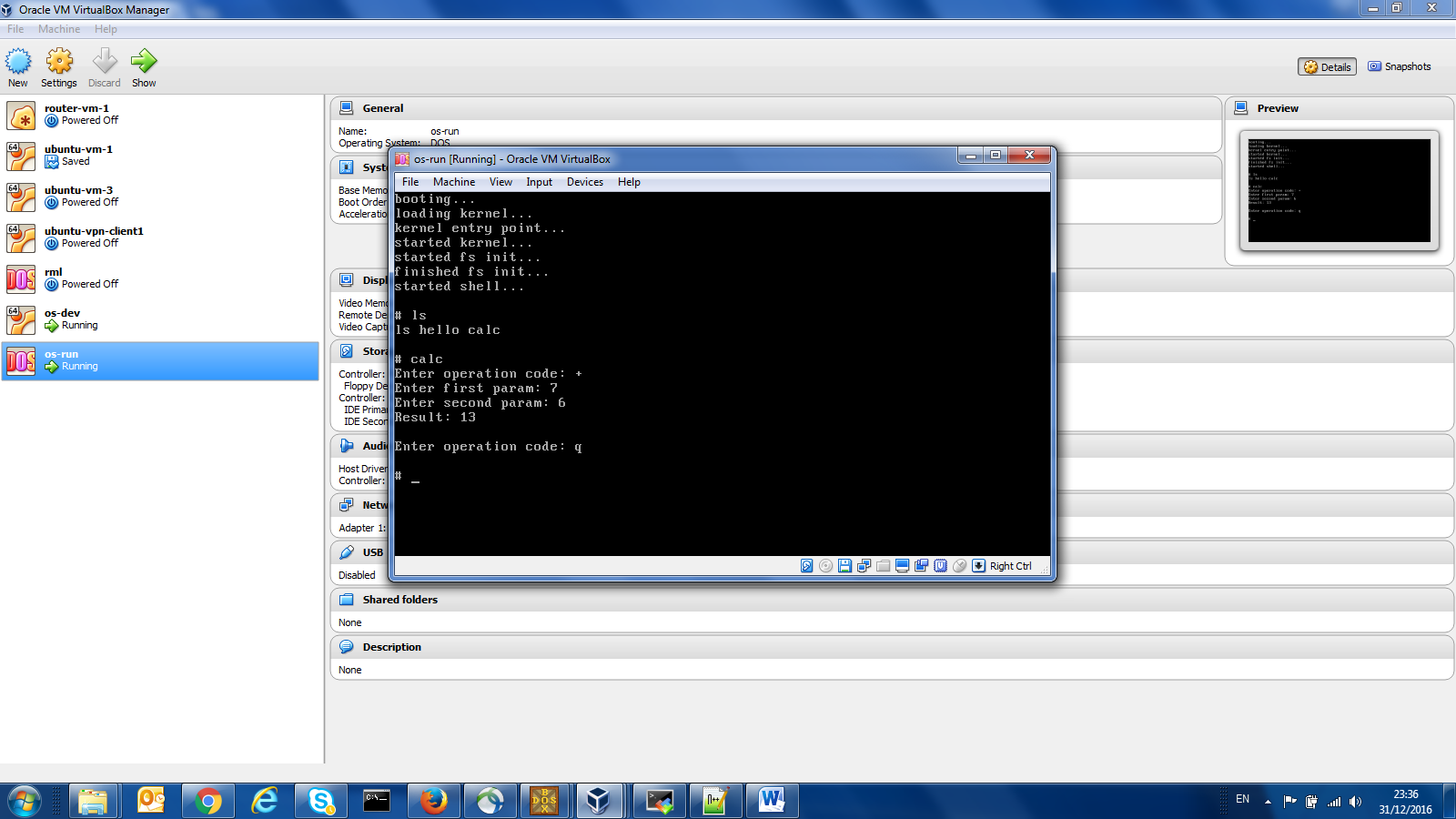


Accept all defaults to create a VM and then go to Storage setting and upload mydisk.img binary as an input for floppy controller.





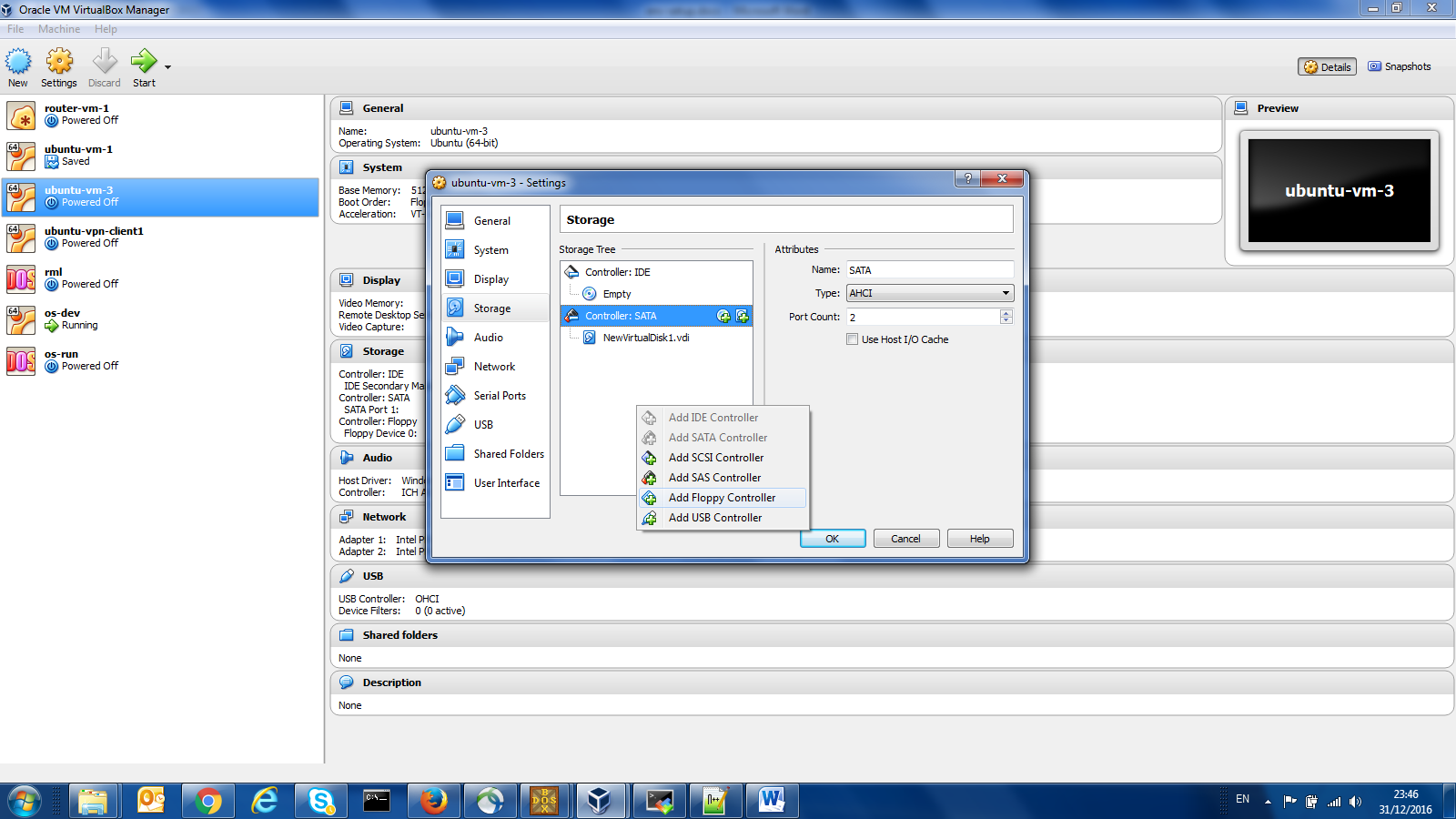
Now run the VM – you can see your OS loaded.



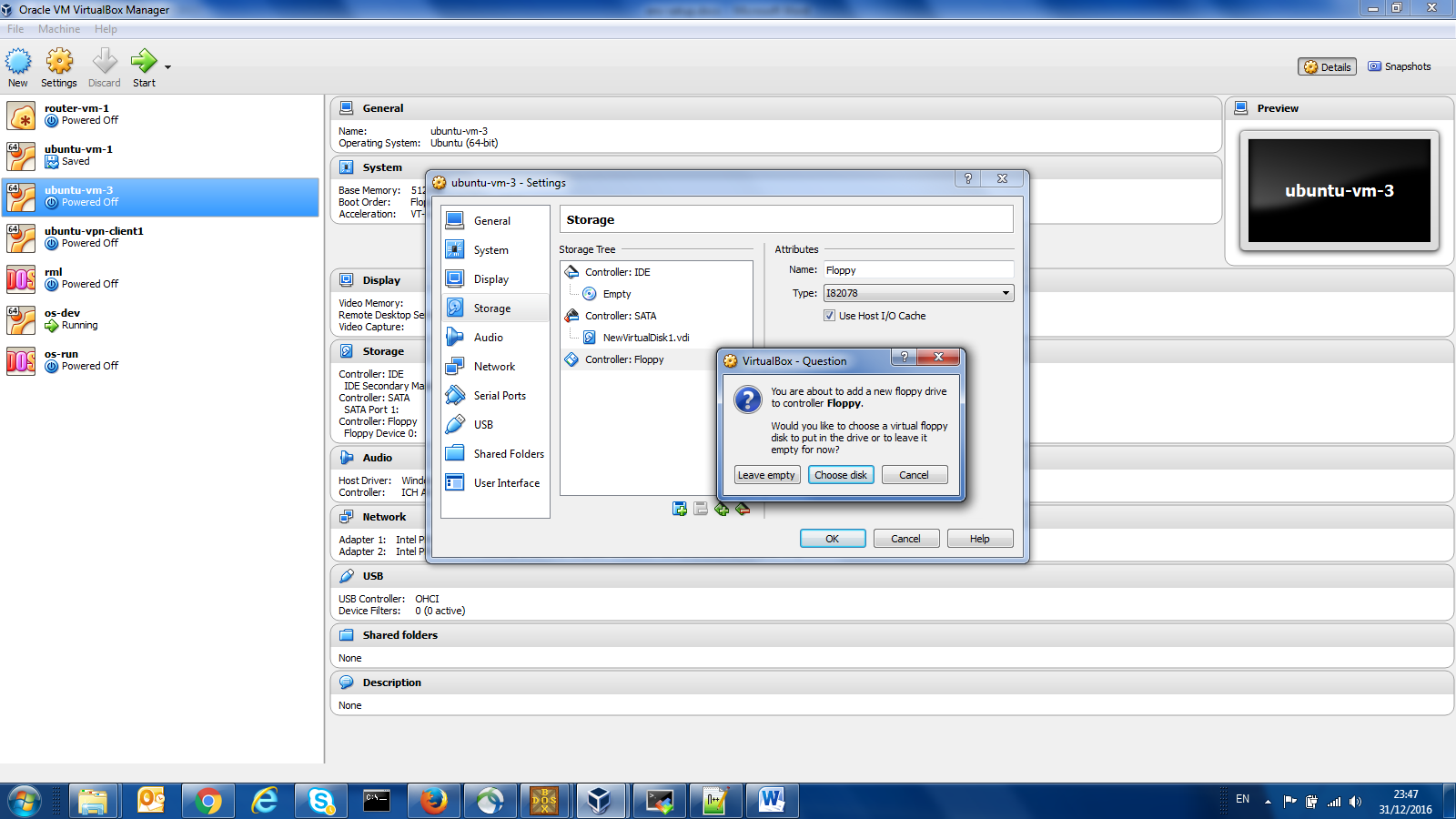
7.2

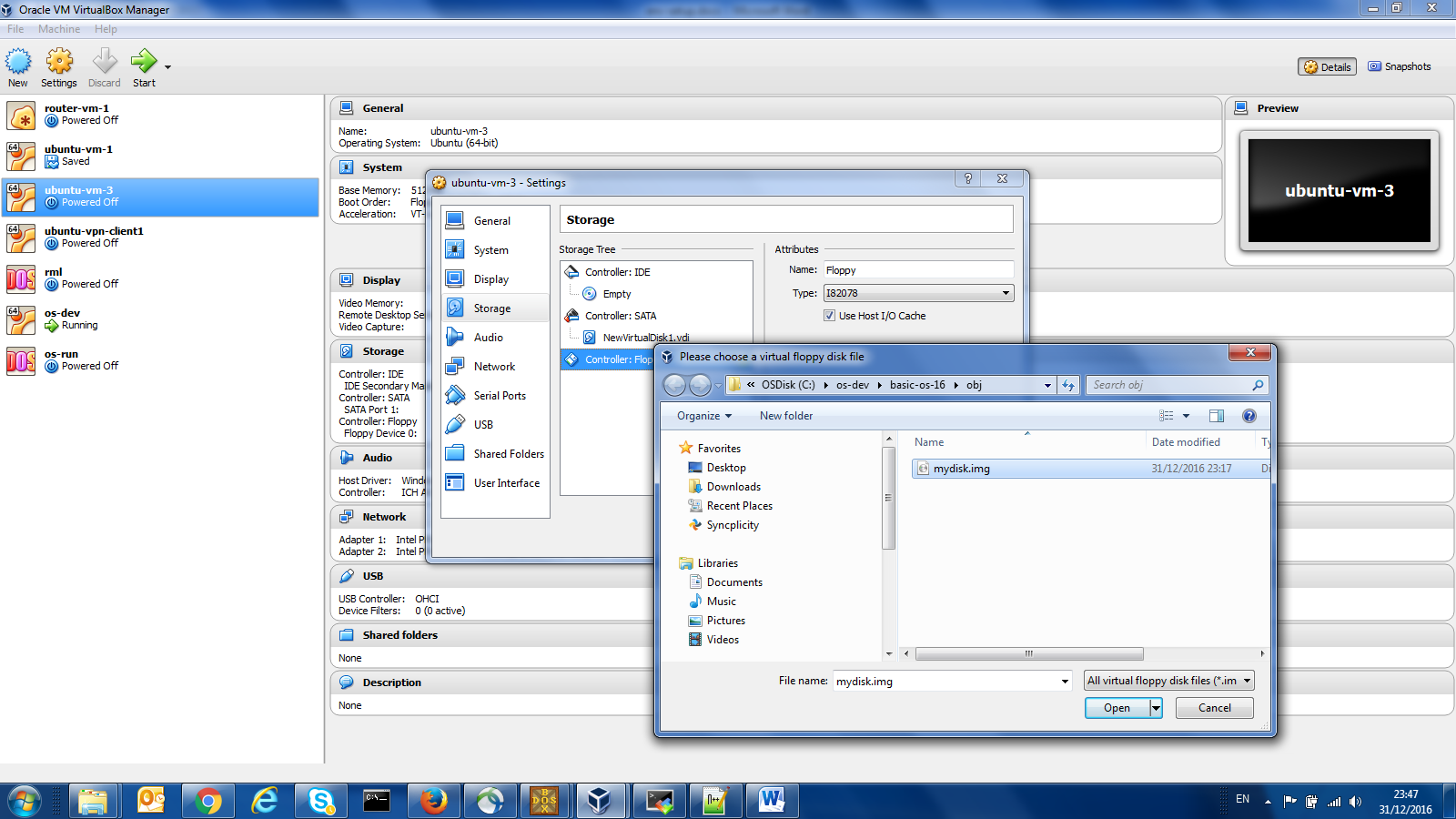
Any other VM setup (say, Ubuntu 64 VM)

Add floppy controller

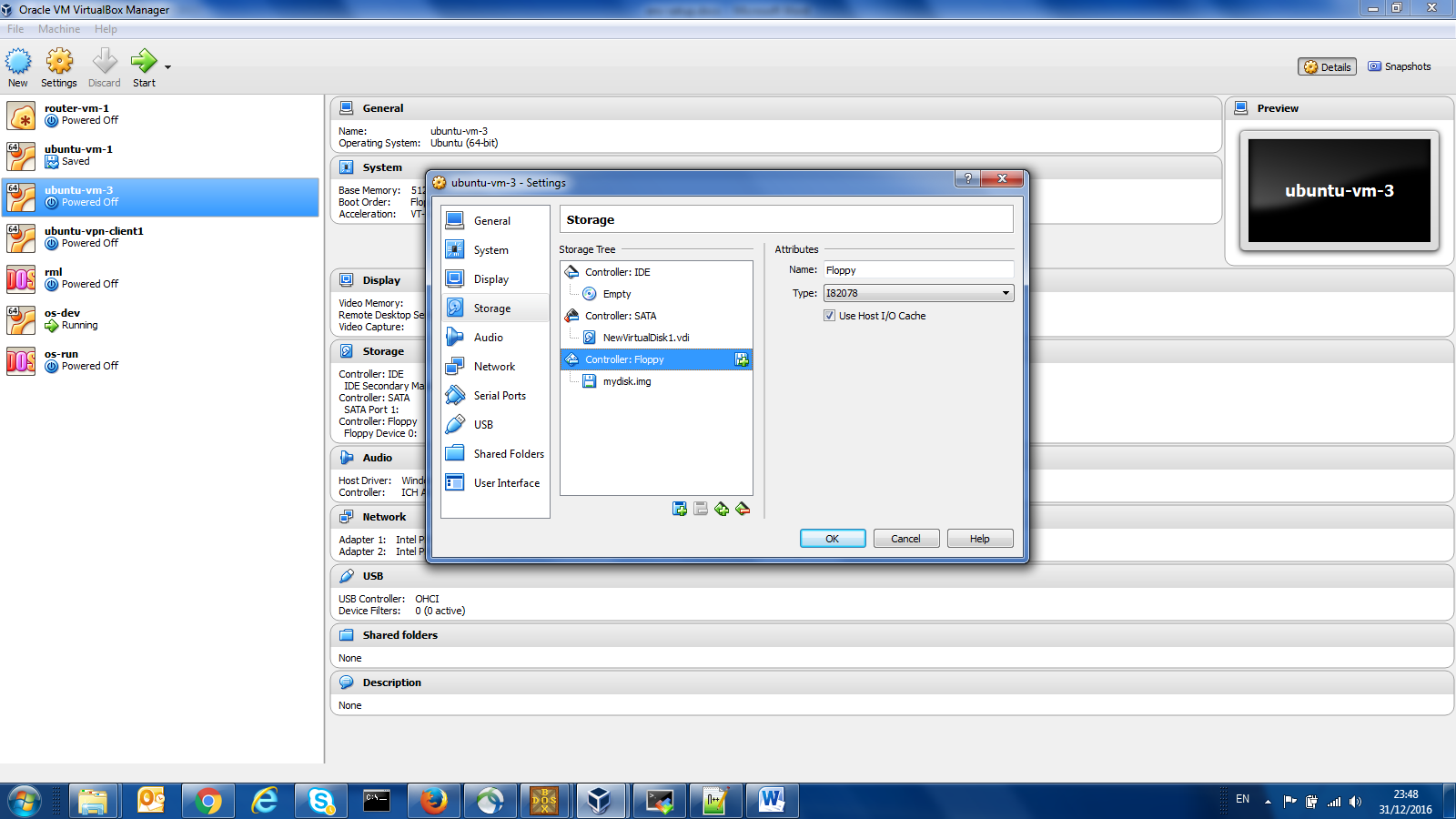


Choose mydisk.img as input for floppy controller

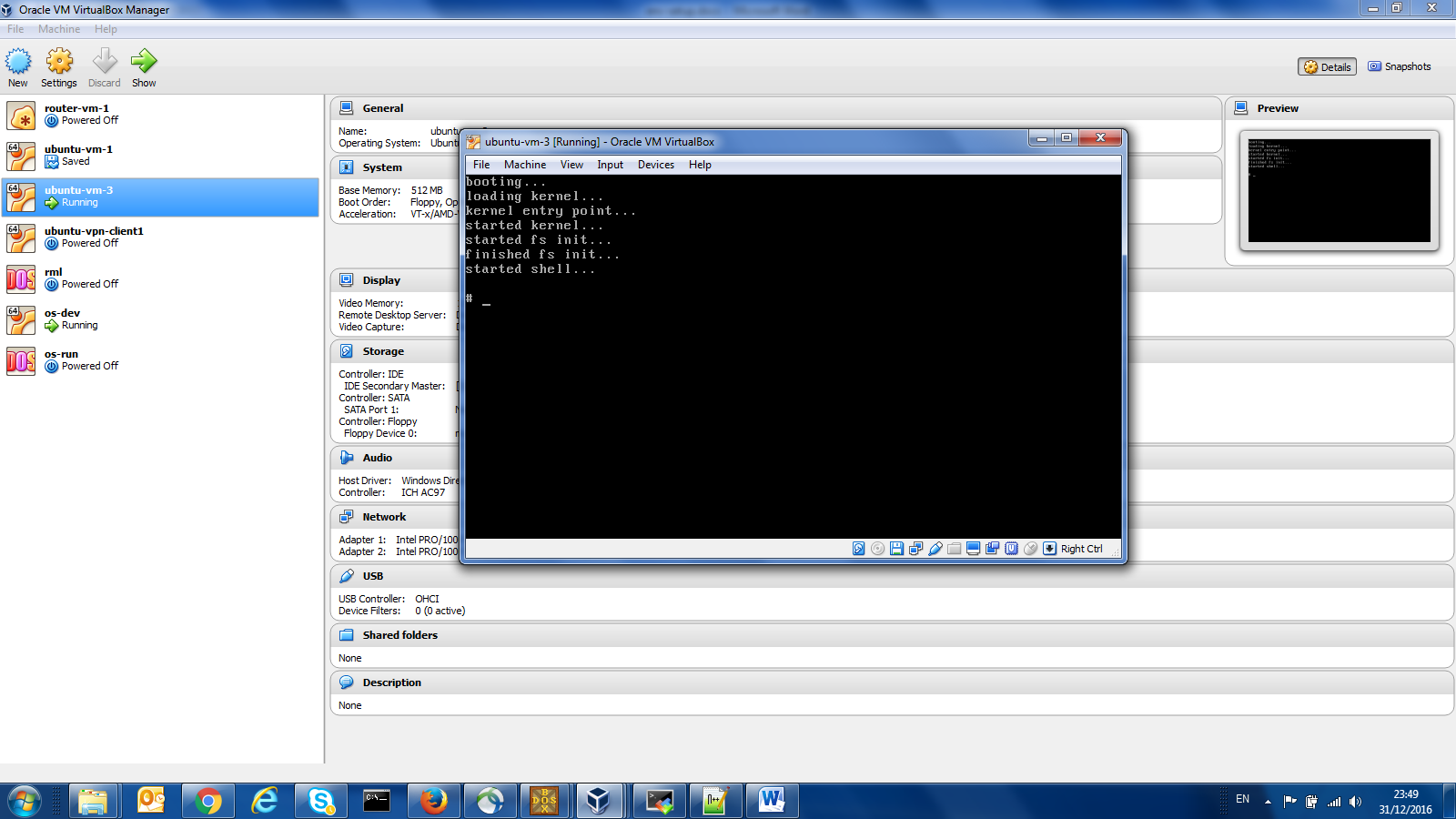




Here is the result



Now run the VM as usual



7.3

It is also possible to create CD image and provide it as input for disk controller.

**8.**

**Run the code on real machine from USB.**

Follow all the process of creating a new mydisk.img. Then insert USB to your Win7 machine and make it accessible in your Ubuntu VM from which your run “make” command (you have Vbox USB docs – check them if needed).

Assuming that USB appears as sdb, execute the following to copy your OS to USB

dd if=/dev/zero of=/dev/sdb bs=1440K count=1

dd if=./obj/mydisk.img of=/dev/sdb conv=notrunc bs=512 seek=0 count=126

Alternatively, from the same folder where you run “make”, run my script like this --- ./copy-to-usb.sh

Now shutdown your Win7 machine, start it again , enter BIOS setup (Enter then F1 on my Lenovo machine), change boot order to start from USB. Restart the machine and see OS loaded.

**9.**

**Debugging with Bochs**

9.1

Download and install Bochs-2.6.8.exe – exists in C:\os-dev\env.

9.2

The following file has the required configuration to run and debug the prepared mydisk.img - C:\os-dev\basic-os-16\bochsrc2.bxrc.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The file content specifies the location of the binary, size, log etc.. – here it is

megs: 32

floppya: 1\_44=.\obj\mydisk.img, status=inserted

boot: a

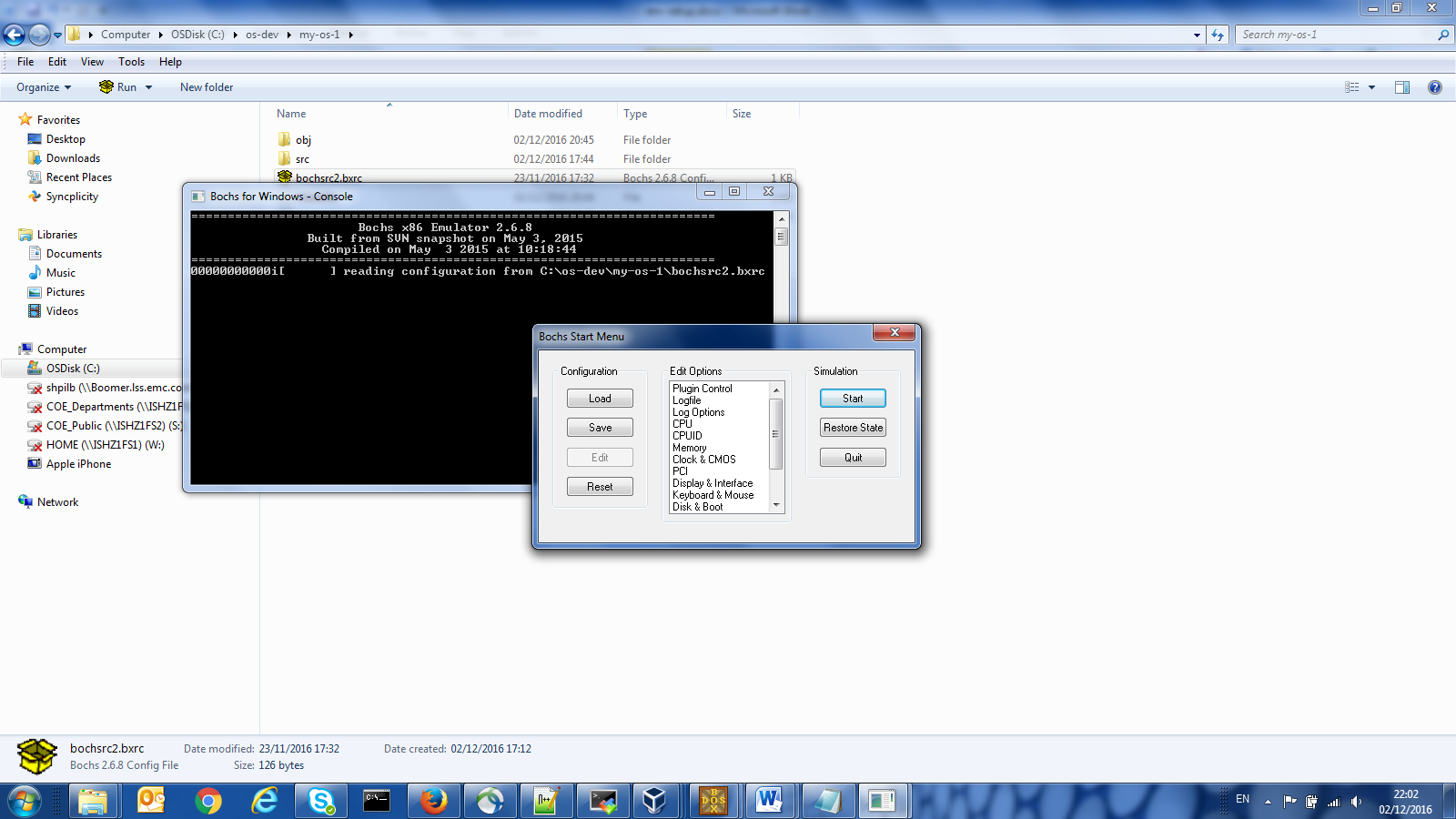
log: c:\bochsout.txt

mouse: enabled=0

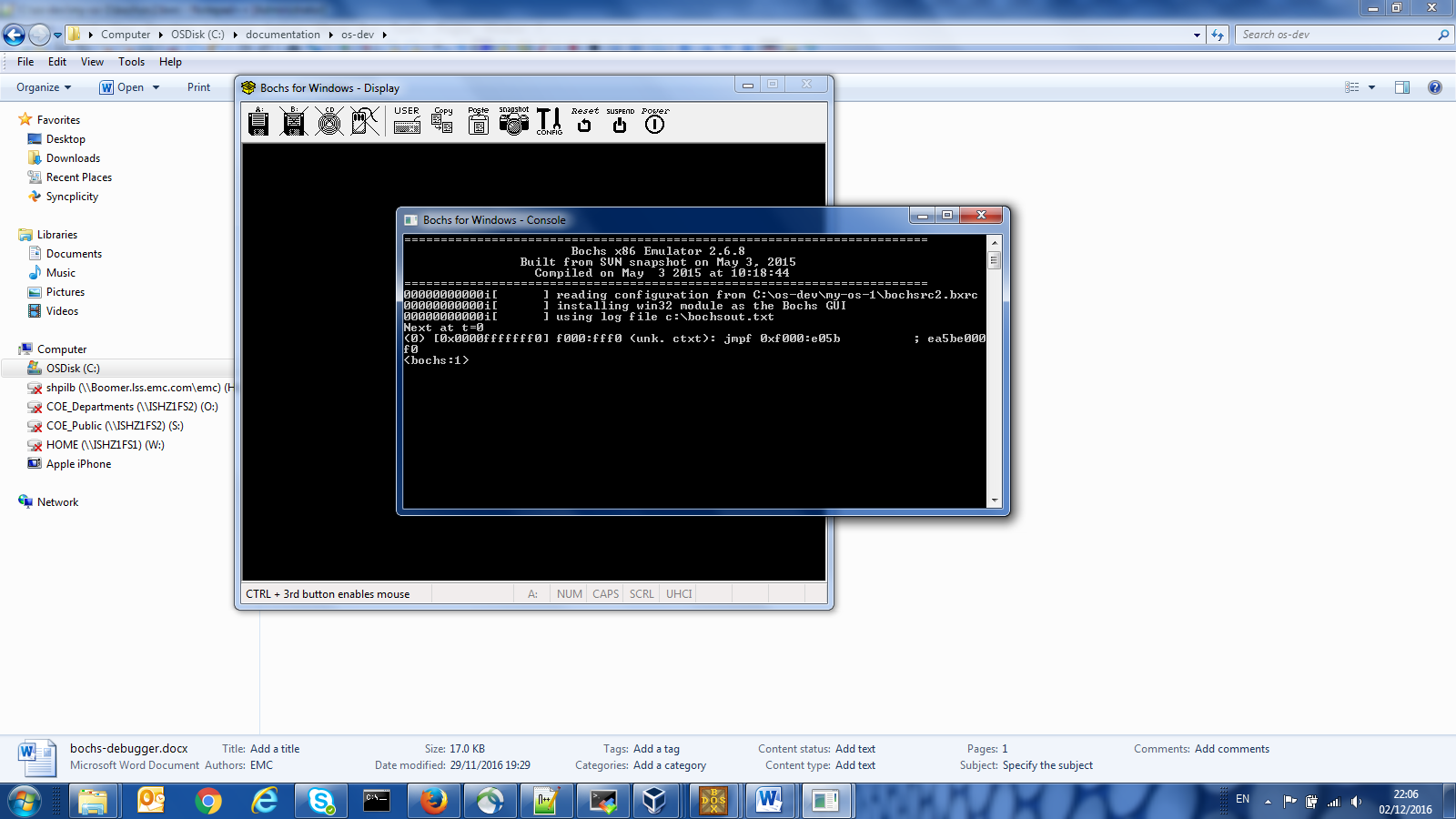
display\_library: win32

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Just right click it and start the debugger. Then press on start on next and the debugger will get started.



Here is how it should look



Now you can use the following things in debugging session

vb 0x7C0:0x200 //insert breakpoint

c //run to breakpoint

s //step

r //show registers

xp /20bx 0x7C0:0x023e //show 20 bytes in hex format at address 0x7C0:0x023e

u 0x7c0:0x200 0x7c0:0x300 // disassemble and show instruction linear addresses on left

lb 0x7e22 // set breakpoint on linear address