Steps of GDB integration:

1. Launch the given linux image file, go to the following link to download the bochs source file with gdb in it, and unzip it.

http://sourceforge.net/projects/bochs/files/bochs/2.6.8/

1. To ensure that there is no installment of bochs in your environment, open a terminal and execute the following command.

*sudo apt-get remove bochs*

1. Go to the directory of bochs source code. In that directory, we configure the bochs file with gdb stub with the following command. After this, we enabled that the bochs could be installed with gdb stub.

*sudo ./configure --enable-gdb-stub*

1. After the configuration, run the following command to compile the bochs

*sudo make*

1. After compilation, we install the bochs with the following command

*sudo make install*

(In my computer, I encountered an error related to x11 in this step, so I run [sudo apt-get install xorg-dev] in advance and then I repeated the step 1-5)

1. With the bochs-gdb environment setup, we now set the connection between our kernel project and the gdb debugger. We enter into the project directory MP1\_Sources and add the following line to **bochsrc.bxrc** file. This step aims to set port through which the gdb and bochs can communicate.

*gdbstub: enabled=1, port=1234, text\_base=0, data\_base=0, bss\_base=0*

1. In the MP1\_Sources project directory, we add “-g” to every gcc command to enable debugging. After the addition, we keep the terminal directory under the MP1\_Sources to use the following command to recompile our project files.

*make*

1. Now our terminal is the MP1\_Sources, and our bochs with gdb has been installed, and the connection between bochs and gdb has been created. Now we run the following command to boot the bochs, it would not boot immediately, it would wait for gdb connection on port 1234.

*Bochs -f bochsrc.bxrc*

1. To enable connection, we open another terminal to run gdb and connect it to bochs through port 1234. We direct the terminal to MP1\_Sources, and then run the following command.

*gdb –q kernel.o*

*target remote :1234*

Now we should be able to see on one the two terminals showing “Connected to 127.0.0.1”. This the end of integration.

