we have used the system call kernel2D copy(), which copies one 2-D float-ing point matrix to another using \_\_copy from user() and copy from user() kernel functions Firstly I have used VMware to setup Artix kernel and then connected it to an ide(VS CODE) then I have created a code in which a syscall\_define function call which is used to copy the data from a 3by3 matrix to another matrix, and that code is placed in the sys.c file,

for adding the syscall :-

commands to run on linux are:-

kernel/linux-6.XX.XX/arch/x86/entry/syscalls

there opening nano syscall\_64.tbl

there we need to edit the file as

451 common kernel2Dcopy sys\_ kernel2Dcopy

Now for implementing this we need to define a new function in the sys.c file

Commands to be written on the shell are:-

cd new\_kernel/linux-6.XX.XX/kernel

nano sys.c

we does the required edit over there

SYSCALL\_DEFINE4(kernel2Dcopy, float\*\*, mat1, float\*\*, mat2, int, row, int ,col)

Now we need to compile the kernel using commands as follows:-

then make cmd is called on the terminal, and when the process is completed commands are called on the terminal to install the make modules, and then a new kernel name and image is created by using the respective commands

make

make modules

make modules\_install

make bzImage

cp -v arch/x86/boot/bzImage /boot/vmlinuz-linux\_ kernel\_paras

cp /etc/mkinitcpio.d/linux.preset /etc/mkinitcpio.d/linux\_kernel\_paras.preset

nano /etc/mkinitcpio.d/linux\_kernel\_paras.preset

Now, we make the necessary changes to the file that got opened.

mkinitcpio -p linux\_ kernel\_paras

update-grub

reboot

now for

After this, we reboot and log in again to check that the kernel is completely set up and ready to run the test.c file

test.c file is created to check whether the matrix is being copied or not.

If it gets copied, then the tar file is copied to another file, and then the differences between the standard kernel and the kernel set up by us are stored in the div file, which is then downloaded from the ide