**kernel.c (Source code)**

/\* kernel.c

\* Bina Maria (00366107)

\* MS (CS) Fall - 17

\* Kernel program to print "Hello World" at the top-left corner of the screen.

\*/

/\* Declaring Function \*/

void showHelloWorld();

/\* Declaring Variables \*/

int color = 0x7;

int segmentBaseAddress = 0xB000;

int memoryBaseAddress = 0x8000;

/\* The entry point of the program \*/

int main()

{

showHelloWorld();

/\* Infinite Loop \*/

while (1==1)

{

}

}

/\* Prints "Hello World!" at the top-left corner of the screen \*/

void showHelloWorld()

{

putInMemory(segmentBaseAddress, memoryBaseAddress, 'H');

putInMemory(segmentBaseAddress, memoryBaseAddress + 1, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 2, 'e');

putInMemory(segmentBaseAddress, memoryBaseAddress + 3, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 4, 'l');

putInMemory(segmentBaseAddress, memoryBaseAddress + 5, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 6, 'l');

putInMemory(segmentBaseAddress, memoryBaseAddress + 7, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 8, 'o');

putInMemory(segmentBaseAddress, memoryBaseAddress + 9, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 10, ' ');

putInMemory(segmentBaseAddress, memoryBaseAddress + 11, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 12, 'W');

putInMemory(segmentBaseAddress, memoryBaseAddress + 13, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 14, 'o');

putInMemory(segmentBaseAddress, memoryBaseAddress + 15, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 16, 'r');

putInMemory(segmentBaseAddress, memoryBaseAddress + 17, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 18, 'l');

putInMemory(segmentBaseAddress, memoryBaseAddress + 19, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 20, 'd');

putInMemory(segmentBaseAddress, memoryBaseAddress + 21, color);

putInMemory(segmentBaseAddress, memoryBaseAddress + 22, '!');

putInMemory(segmentBaseAddress, memoryBaseAddress + 23, color);

}

**compileOS.sh (Source code)**

#compileOS.sh

#Bina Maria (00366107)

#MS (CS) Fall - 17

#Shell script to produce floppya.img file.

nasm bootload.asm #To assemble bootloader using NASM assembler!

dd if=/dev/zero of=floppya.img bs=512 count=2880 #To make image file of floppy disk that is filled with zeros!

dd if=bootload of=floppya.img bs=512 count=1 conv=notrunc #To copy bootload to the beginning of floppya.img!

bcc -ansi -c -o kernel\_c.o kernel.c #To compile your kernel!

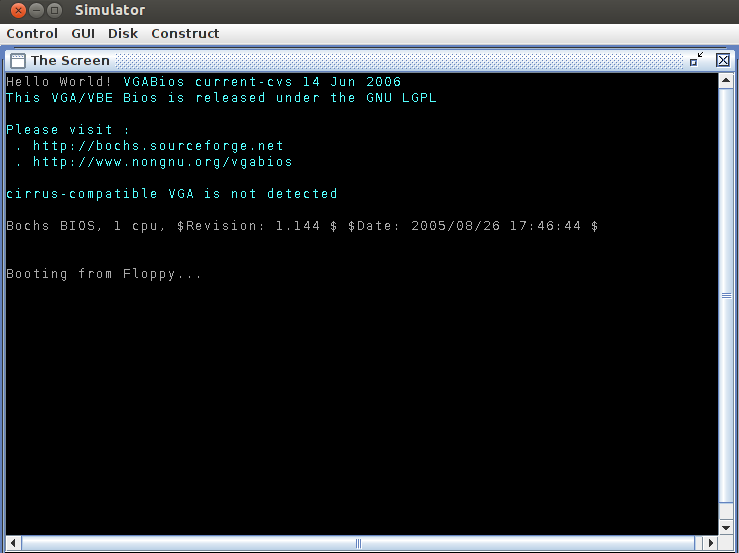
as86 kernel.asm -o kernel\_asm.o #To assemble kernel.asm!

ld86 -o kernel -d kernel\_c.o kernel\_asm.o #To link and produce kernel!

dd if=kernel of=floppya.img bs=512 conv=notrunc seek=3 #To copy kernel to the third sector of floppya.img!

**Output Screenshots**

Testcase 1: When tried running Emumaker86 after producing floppya.img file manually without using shell script.



2. Testcase 2: When tried running Emumaker86 after producing floppya.img file using shell script.

