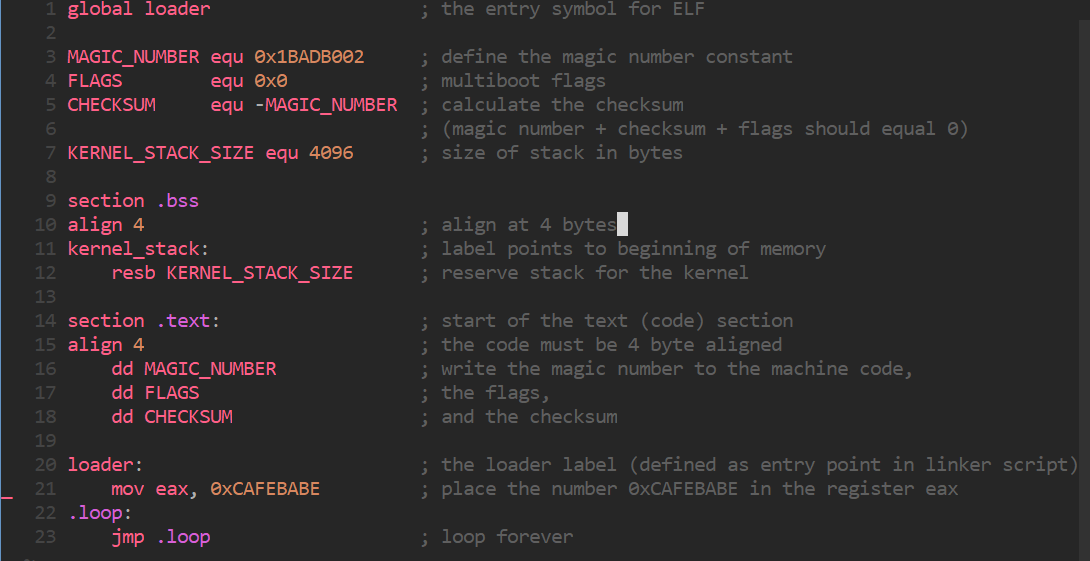
Robert Kats - James Anderson - Patrick Star

We decide to run the little Os using the windows linux subsystem with ubuntu 16 (Rob) and 18 (James) and 18(Pat). We then used an xServer forwarding application( [xMing](https://sourceforge.net/projects/xming/) ) to allows forwarding of the gui applications to run on windows.

We then installed all the packages required running the command

**sudo** apt-get install build-essential nasm genisoimage bochs bochs-sdl

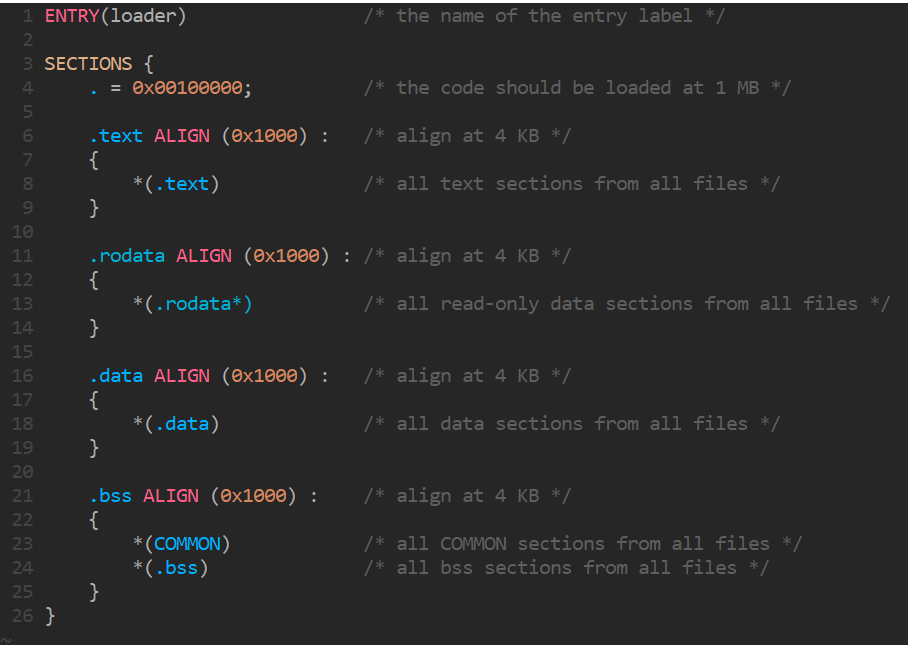
Then we created a loader.s file and added the following code:



Then we ran the command nasm -f elf32 loader.s

Section 2.3.2 :

How we have to link to the kernel. Adding the following code to the file link.ld:



Now to link this and make the kernel we run the command: ld -T link.ld -melf\_i386 loader.o -o kernel.elf

Then we downloaded Grub to help us boot the kernel <ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz> Or

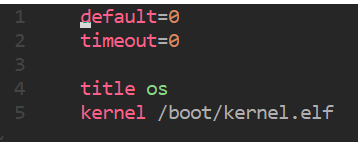
<http://littleosbook.github.com/files/stage2_eltorito>.

How we have to build the iso file by running the following commands:

mkdir -p iso/boot/grub && cp stage2\_eltorito iso/boot/grub/ && cp stage2\_eltorito iso/boot/grub/

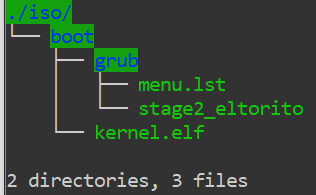
The first command creates a new directory and the other two commands copy the files into that new folder.

Then we made a menu for grub the file called menu.lst has the following code:



This is how grub will know to start our os.

This is the tree view of the iso folder:



After Running we will have an iso to run!

genisoimage -R \

-b boot/grub/stage2\_eltorito \

-no-emul-boot \

-boot-load-size 4 \

-A os \

-input-charset utf8 \

-quiet \

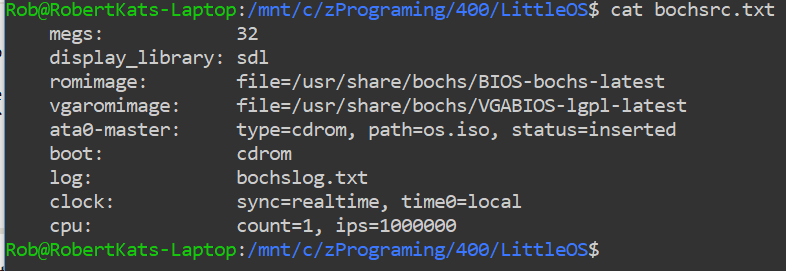
-boot-info-table \

-o os.iso \

iso

---2.3.5 time to run the iso

Now we need to run the iso file we made in side an emulator called Bochs and this emulator needs to be configured with the flowing code:



Then we turn it with the following command:

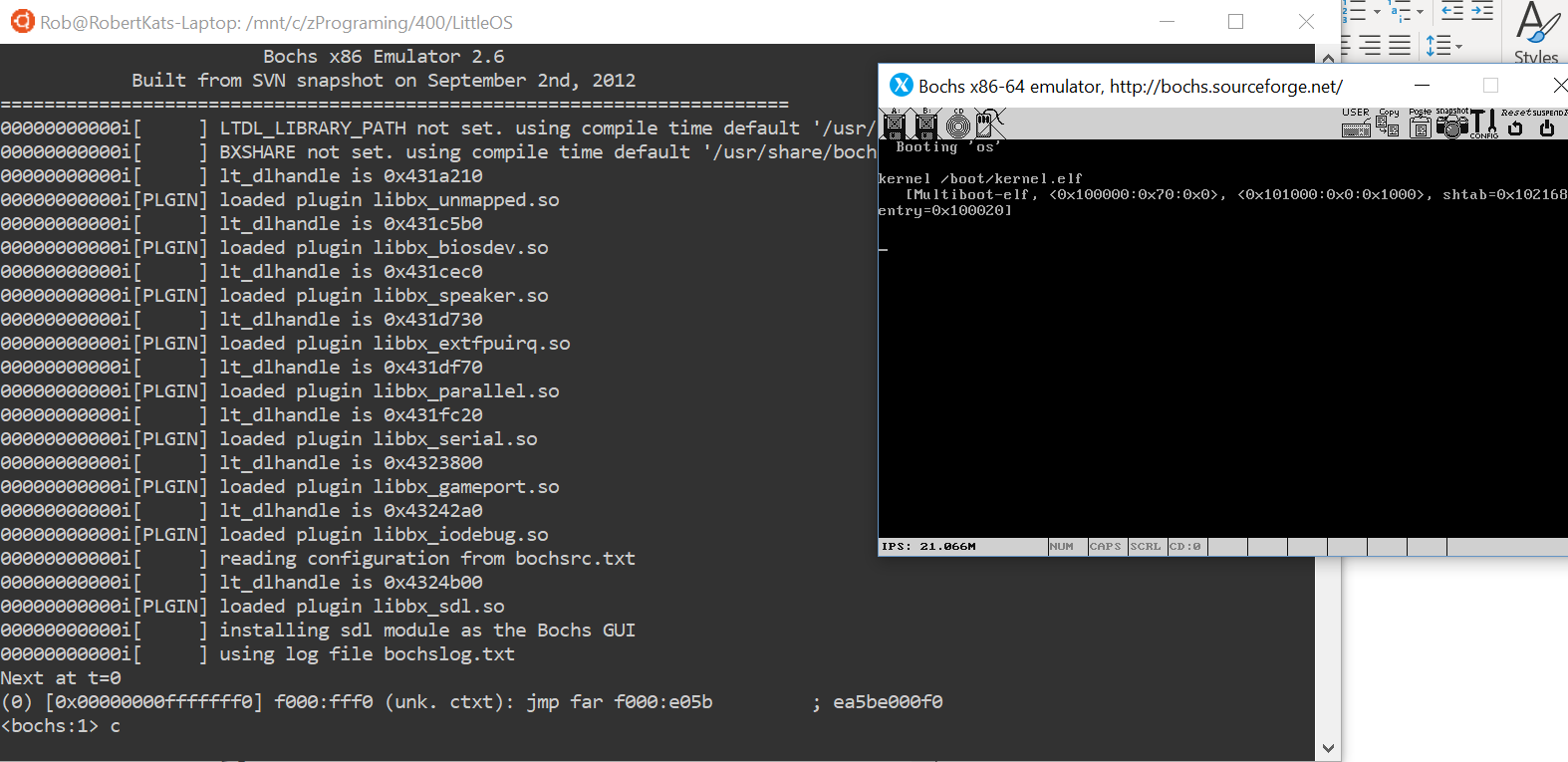
Bochs -f bochsrc.txt -q

Now after running we hit c to continue in the bochs and run the iso.

This actually caused errors before because we didn’t know we had to hit c

We found it on the github page of the little io

<https://github.com/littleosbook/littleosbook/issues/94#issuecomment-390510749>

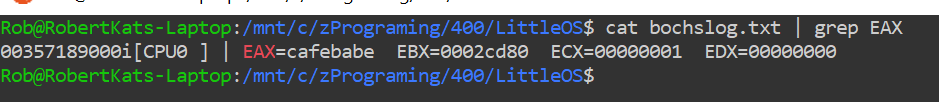


Now we look for cafebabe in the register we put it in.

With the following line the we liked to use better:

cat bochslog.txt | grep EAX

We take the ouput of the file and the grep it with the register we are looking for:

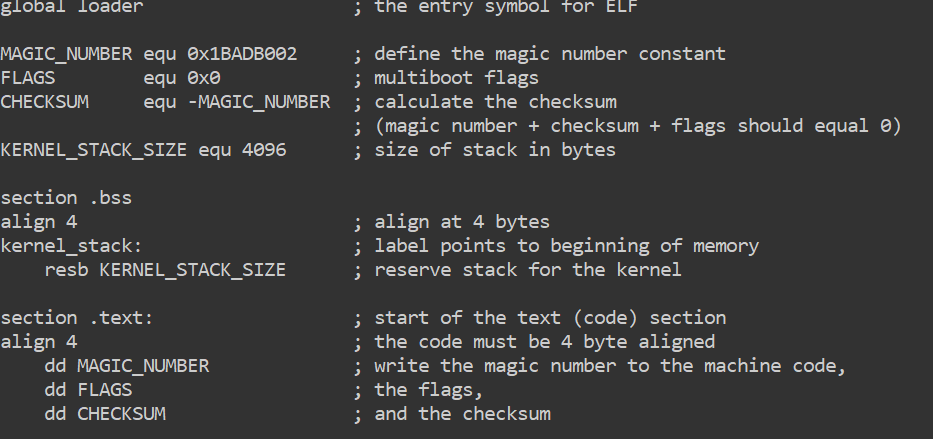


It worked !!

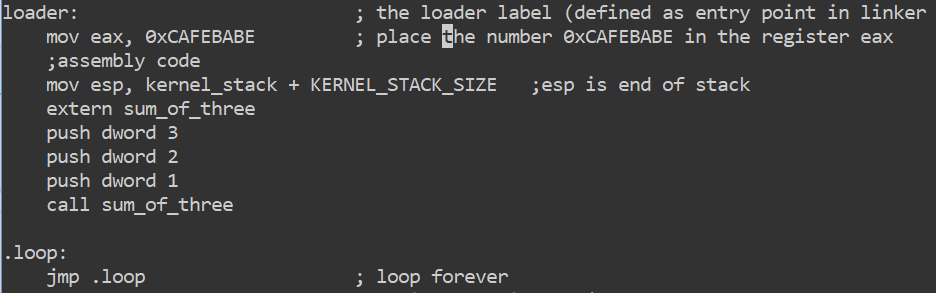
Chapter 3:

Now we get to the coding bit.

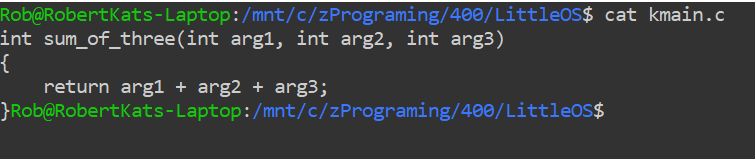
We lead into the loader file and we add a stack. We reserve memory so we don’t overwrite other data. How we have a stack and make the stackpointer using the esp register.



Now for 3.2 we have to call some c code from Assembly. This works because c complies down to assembly when using cdecl convention as the book puts it. We added this to the loader.s file:

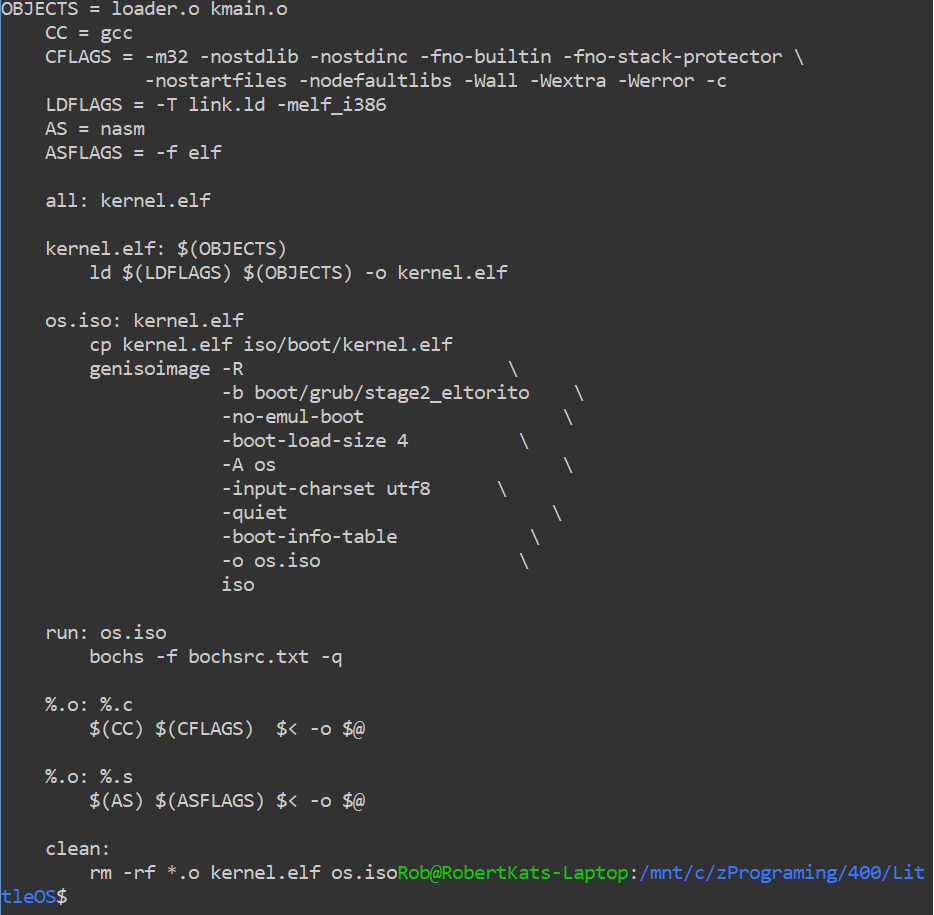


Then added this to the kmain.c:

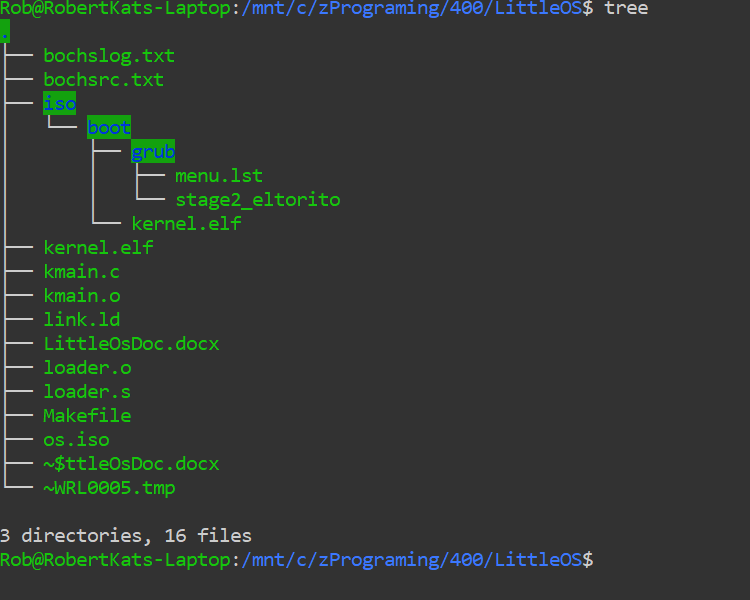


For 3.3 and 3.4 we now create a make file to compile the code and quickly run the os. This helps greatly with running and testing code as now all we have to do to run the os is type make run and it will compile and move all the files then will also make a iso file as well. After that I will run bosh for us all in one go.

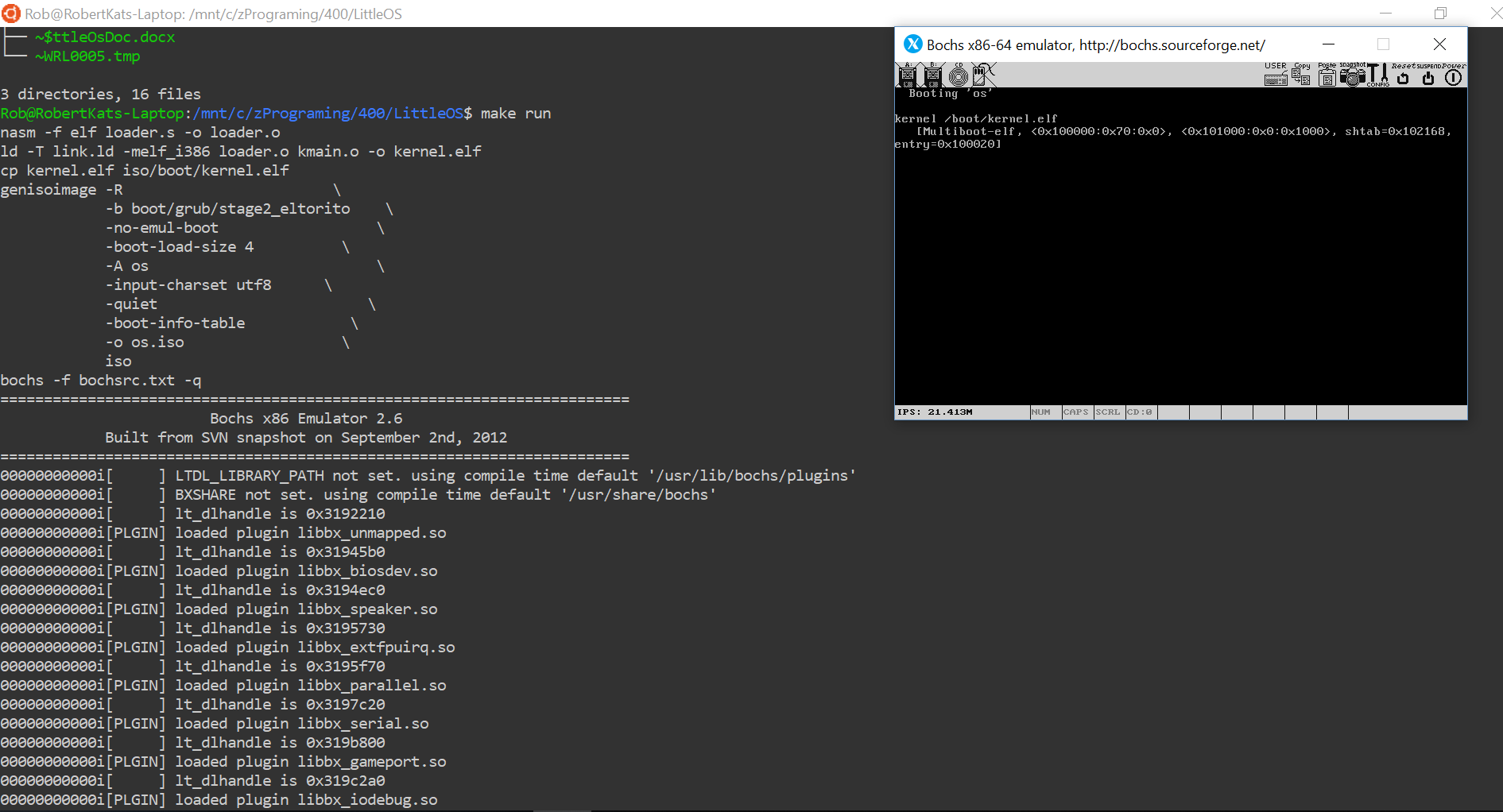
This is the make file:



And the tree of our project that is git staged:



It Runs!



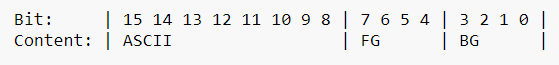
Chapter 4: Output

Now we are trying to output text to be displayed on the console and write data to the serial port. We need to create a layer between the kernel and hardware.

The method FBWriteCell was created to write a single character to a specified cell index with a specific foreground and background colors.

(talk more bout 4.2.1)

FB Write Cell Builds that 16 bit thing



4.2.2 Moving the cursor:

First in io.s (for creates the assembly that c can call as a function) Outb we called SendDataToPort for readability as well as GetDataFromPort READ BOUT IT

Then io.h is created to prototype for c the functions sendData and getData

Send data will take ushort that is the identifier for the port and uchar data witch is the data to be sent to the port.

char function takes the argemnt of the port and returns the unsigned char in that port

in kmain we include io.h (added defines use screen shots)

then we wrote the method FBMove curser takes a porsiton makes the appority senddatatoPort calls to move the curser to the position.

4.2.3