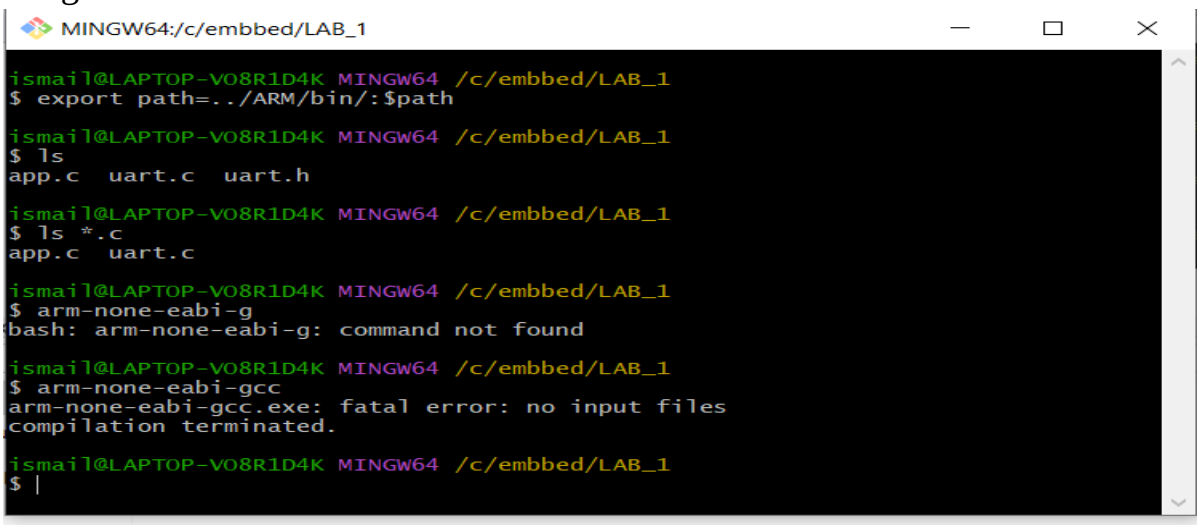
Booting sequence report

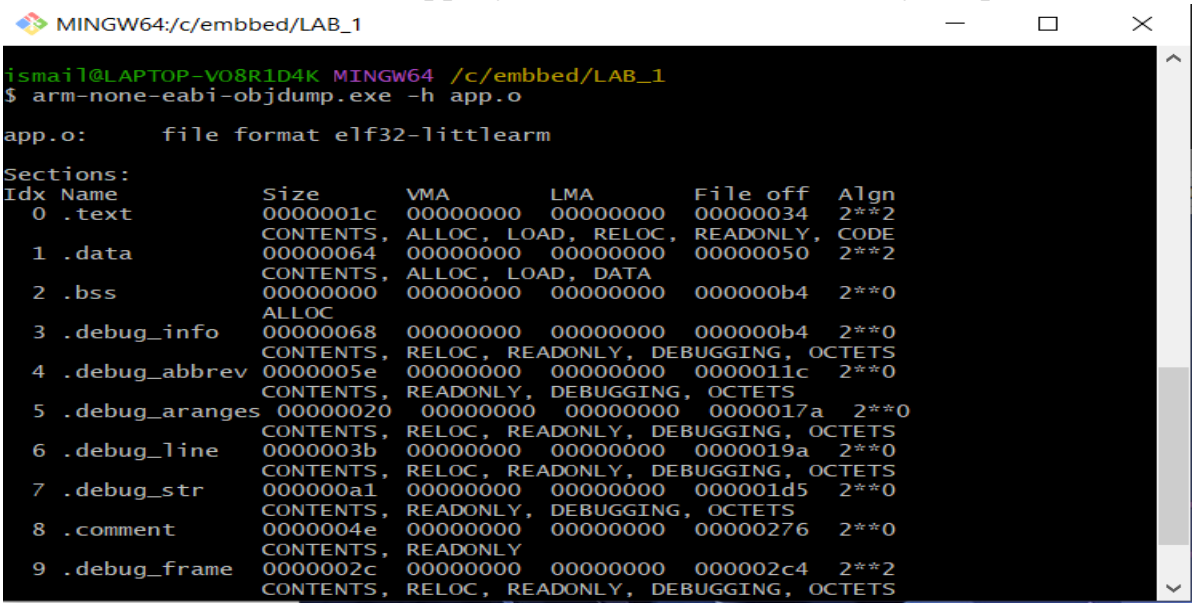
Export the PATH of ARM-Cross-toolchain “arm-none-eabi-gcc.exe” to make things ez



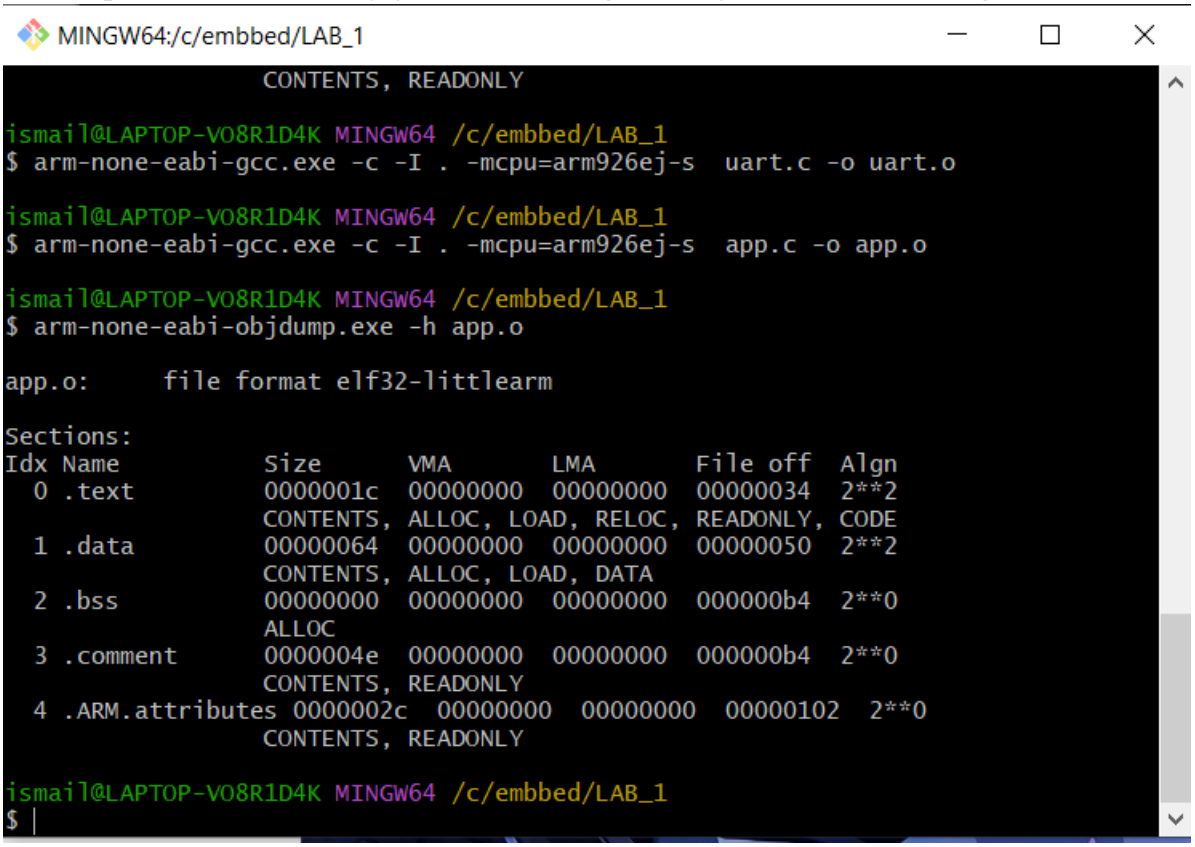
Then generate out file using touch command then use the command “arm-none-eabi-gcc.exe --help” to see what u want the output to be or what at what stage do you want to stop and give it the architecture of the MCP.

Object file is a relocatable Binary which means that it has not a physical address on the system bus, and it can not to run on processor.

- To know the Sections of App.c just use “arm-none-eabi-objdump.exe”.



If u don’t want the debug section to be created because the debug section takes more space from memory just in creating the object file remove -g.



You will need a startup file to know the initialize the stack pointer to reserve the Area from stack in RAM and to know the start point that the processor will execute it first thing when it's on.

```
ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s  startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted

ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000010  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000044  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000044  2**0
                  ALLOC
  3 .ARM.attributes 0000001c  00000000  00000000  00000044  2**0
                  CONTENTS, READONLY

ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$
```

Finally, you will need a linker script to define the volume of RAM and ROM and pass this information to the linker to link the different object files and give the variable we used a physical address in Memory and the beginning of the reset Section According to the specs of CPU entry point then put it at this address and we can use it by this command "arm-none-eabi-ld.exe" and then we need the binary file that will run on the specific MCP so we use this command "arm-none-eabi-objcopy.exe".

```
ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o -o learn-in-depth.elf

ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

To run the code making sure that it works on the target we use qemu.

```
ismail@LAPTOP-VO8R1D4K MINGW64 /c/embbed/LAB_1
$ ../qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-dep
th.bin
learn-in-depth:ismail
```