**Commands Description:**

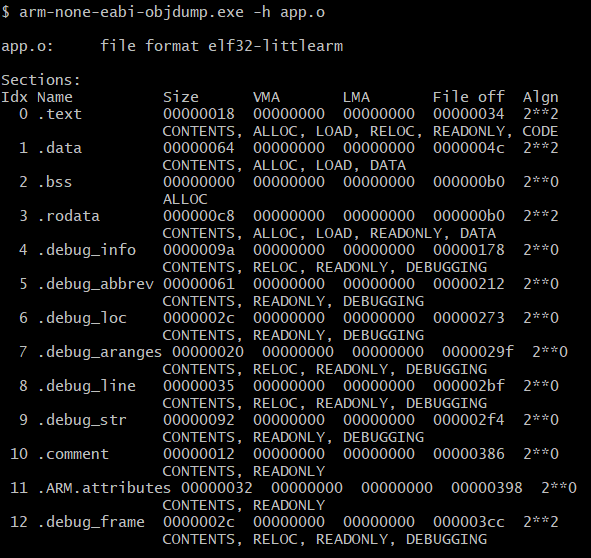
* Create file
  + touch filename

arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s -o0 -I . app.c -o app.o

|  |  |
| --- | --- |
| -c | option to compile and assemble but don’t link |
| -g | include debugging information |
| -mcpu= | choose CPU |
| -o0 | optimization level zero |
| -I . | Include header + directory of the used header (here ‘.’ Means include the header in the current directory). |
| -o | Get output |

**Display the contents of the section headers:**

arm-none-eabi-objdump.exe -h app.o

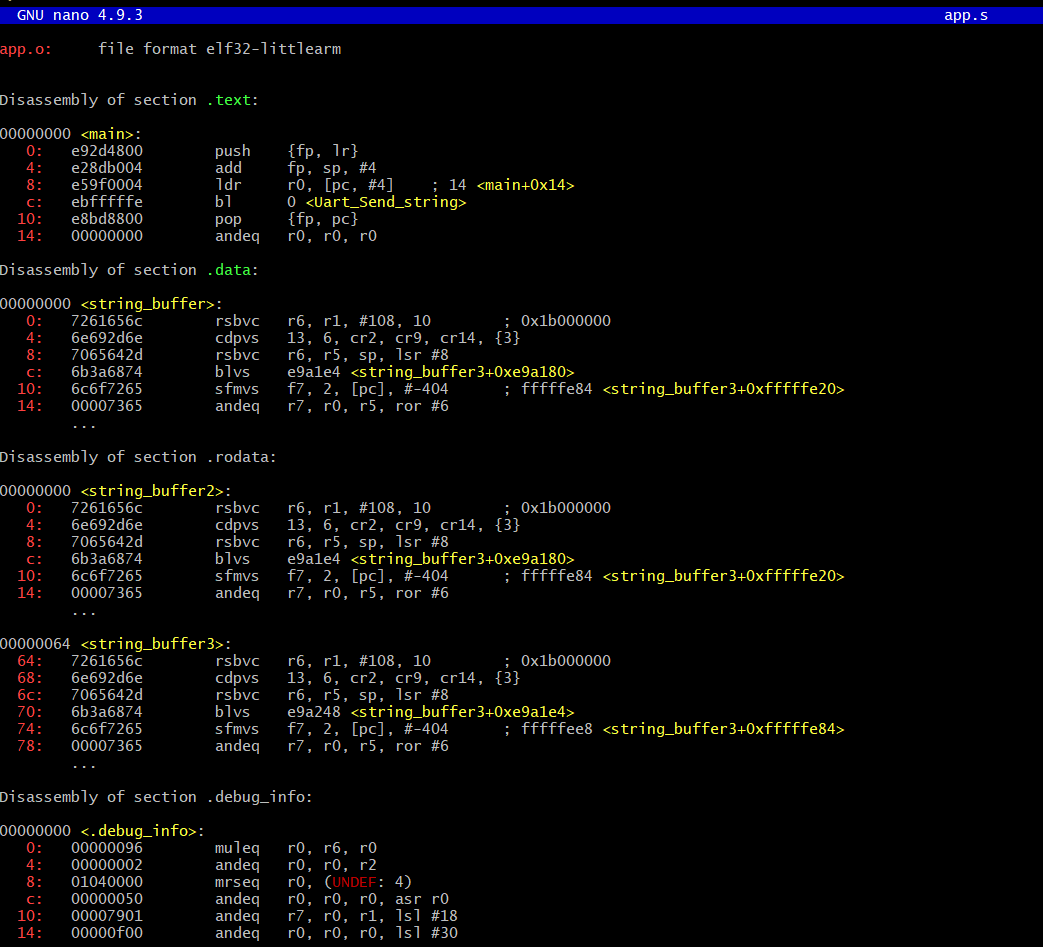


Display assembler content:

-D all sections

-d executable sections

arm-none-eabi-objdump.exe -D app.o >> app.s



**Linking the .o files and get the .elf file with the linker script**

arm-none-eabi-ld.exe -T linker\_script.ld app.o uart.o startup.o -o learn-in-depth.elf

to get the map file

arm-none-eabi-ld.exe -T linker\_script.ld app.o uart.o startup.o -o learn-in-depth.elf -Map=Map\_file.map

**To see the symbols in the obj file**

arm-none-eabi-nm.exe app.o

**To get the hex/bin file**

arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin

**Run the elf file using qemu:**

qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin

-M “machine name” ->versatilepb

-m memory size -> 128M

-nographic -> diable the display (this board has display)

-kernel “binary file” -> learn-in-depth.bin

