Using VersatilePB virtual board in QEMU and ARM toolchain

1- Getting app.o uart.o files without debugging information:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson
 2/LAP 1
$ arm-none-eabi-gcc.exe -c app.c -o app.o

My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson
 2/LAP 1
$ arm-none-eabi-gcc.exe -c uart.c -o uart.o
```

-Sections:

App.o sections:

```
 2/LAP 1
$ arm-none-eabi-objdump.exe -h app.o

app.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000020  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000054  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b8  2**0
                  ALLOC
  3 .rodata       00000064  00000000  00000000  000000b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  0000011c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000030  00000000  00000000  0000012e  2**0
                  CONTENTS, READONLY
```

Uart.o sections:

```
$ arm-none-eabi-objdump.exe -h uart.o

uart.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000084  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000030  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

2-Getting startup.o file :

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-as.exe -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
```

-Sections:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 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 00000022  00000000  00000000  00000044  2**0
                  CONTENTS, READONLY
```

3- Symbols of object files:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D string_buffer
00000000 R string_buffer2
         U UART_Send_String
00000004 C x

My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-nm.exe uart.o
00000000 T UART_Send_String

My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop

My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$
```

4-Use linker_script to get executable file:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn_in_depth.elf
```

-Sections:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-objdump.exe -h learn_in_depth.elf

learn_in_depth.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .startup      00000010  00010000  00010000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000070  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .rodata       00000064  00010080  00010080  00008080  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .data         00000064  000100e4  000100e4  000080e4  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .bss          00000004  00010148  00010148  00008148  2**2
                  ALLOC
  5 .ARM.attributes 0000002e  00000000  00000000  00008148  2**0
                  CONTENTS, READONLY
  6 .comment      00000011  00000000  00000000  00008176  2**0
                  CONTENTS, READONLY
```

-Symbols:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-nm.exe learn_in_depth.elf
00010010 T main
00010000 T reset
0001114c B stack_top
00010008 t stop
000100e4 D string_buffer
00010080 R string_buffer2
00010030 T UART_Send_String
00010148 B x
```

5-Get the map file:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=output.map app.o startup.o uart.o
```

6-Get bin file and burn it on board by qemu:

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ arm-none-eabi-objcopy.exe -O binary learn_in_depth.elf learn_in_depth.bin
```

```
My Content@Abdelrhman MINGW64 /e/Embedded systems/keroles/Unit 3/Lectures/lesson 2/LAP 1
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn_in_depth.bin
learn-in-depth:Abdelrhman
```