# Unit 3 Embedded C
# Lecture 2
# Lab 1 Report

## Creating a BareMetal Application on

## ARM VersatilePB Board

---

*By: Yara Ashraf*

---

## Step 1] Creating source and header files

- Adding ARM Toolchain path to directory

```
MINGW32:/d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec 2 Lab Assignment

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 2 Lab Assignment (main)
$ touch app.c uart.c uart.h

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ export PATH=/d/Programs/Installed/ARM/bin/:$PATH
```

app.c | app.s | startup.s | Map_file.map | linker_script.ld | uart.h | uart.c

```c
#ifndef _UART_H_
#define _UART_H_

//UART APIs
void Uart_Send_String(unsigned char* P_tx_string);

#endif
```

app.c | app.s | startup.s | Map_file.map | linker_script.ld | uart.h | uart.c

```c
#include "uart.h"

//define UART0 Register
#define UART0DR *((volatile unsigned int* const)((unsigned int *)0x101f1000))

void Uart_Send_String(unsigned char *P_tx_string)
{
    while(*P_tx_string!='\0')
    {   //loop till end of string
        UART0DR=(unsigned int)(*P_tx_string); //transmit char
        P_tx_string++;
    }
}
```

app.c | app.s | startup.s | Map_file.map | linker_script.ld | uart.h | uart.c

```c
//For VersitilePB Board

#include "uart.h"

unsigned char string_buffer[100] = "Learn-in-depth:Yara";

void main(void)
{
    Uart_Send_String(string_buffer);
}
```

## Step 2] Compiling to get relocatable files (*.o)

- To specify the processor, use option "-mcpu=arm926ej-s"
- To include headers, use option "-I ."

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
signment (main)
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -I . app.c -o app.o

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
signment (main)
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -I . uart.c -o uart.o

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
signment (main)
$ ls *.o
app.o  uart.o
```

## Step 4] Analyzing object files headers section using binary utility **"objdump"**

- **VMA** and **LMA** values are all zeros as the addresses are not known yet.
- **.data** section size = (64) in hex = (01100100) in binary = (100) in decimal, equivalent to the below global array of 100 char (100 bytes)

```c
unsigned char string_buffer[100] = "Learn-in-depth:Yara";
```

- **.bss** section = 0 as there is no global uninitialized data
- **.rodata** section is not seen as there is no constant data

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec
signment (main)
$ 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         0000001c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000050  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .comment      0000007f  00000000  00000000  000000b4  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000133  2**0
                  CONTENTS, READONLY
```

## Step 5] Compiling c files and passing "-g" option to get debug details for relocatable files

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedde
signment (main)
$ arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s -I . app.c -o app.o

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedde
signment (main)
$ arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s -I . uart.c -o uart.o
```

- Debug section headers are seen below:

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit
signment (main)
$ 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         0000001c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000050  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .debug_info   00000066  00000000  00000000  000000b4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000005a  00000000  00000000  0000011a  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000020  00000000  00000000  00000174  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_line   00000035  00000000  00000000  00000194  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_str    000000ef  00000000  00000000  000001c9  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      0000007f  00000000  00000000  000002b8  2**0
                  CONTENTS, READONLY
  9 .debug_frame  0000002c  00000000  00000000  00000338  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 10 .ARM.attributes 00000032  00000000  00000000  00000364  2**0
                  CONTENTS, READONLY
```

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/U
signment (main)
$ 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         00000054  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000088  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000088  2**0
                  ALLOC
  3 .debug_info   00000057  00000000  00000000  00000088  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000051  00000000  00000000  000000df  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000020  00000000  00000000  00000130  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_line   00000039  00000000  00000000  00000150  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_str    000000ed  00000000  00000000  00000189  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      0000007f  00000000  00000000  00000276  2**0
                  CONTENTS, READONLY
  9 .debug_frame  00000030  00000000  00000000  000002f8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 10 .ARM.attributes 00000032  00000000  00000000  00000328  2**0
                  CONTENTS, READONLY
```

## Step 6] Adding a global const array of 100 characters in app.c

```c
//For VersitilePB Board

#include "uart.h"

unsigned char string_buffer[100] = "Learn-in-depth:Yara";
unsigned char const string_buffer_2[100] = "to create a rodata section

void main(void)
{
    Uart_Send_String(string_buffer);
}
```

- **.rodata** section is seen for the const array data with size of 64 hex = 100Bytes

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Ur
Embedded C/Lec 2 Lab Assignment (main)
$ 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         0000001c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  00000050  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .rodata       00000064  00000000  00000000  000000b4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      0000007f  00000000  00000000  00000118  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  00000197  2**0
                  CONTENTS, READONLY

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Ur
Embedded C/Lec 2 Lab Assignment (main)
$
```

## Step 7] Disassembly of app.o

- Section addresses are zeros as physical addresses are not known yet so virtual addresses are assigned

- Per section, assembly instructions have their equivalent hex format and corresponding vitual addresses starting from 0 and incrementing by 4 Bytes per instruction

app.c | app.s | uart.h | uart.c

```
app.o:     file format elf32-littlearm


Disassembly of section .text:

00000000 <main>:
   0: e92d4800    push  {fp, lr}
   4: e28db004    add   fp, sp, #4
   8: e59f0008    ldr   r0, [pc, #8]   ; 18 <main+0x18>
   c: ebfffffe    bl 0 <Uart_Send_String>
  10: e1a00000    nop         ; (mov r0, r0)
  14: e8bd8800    pop   {fp, pc}
  18: 00000000    andeq r0, r0, r0

Disassembly of section .data:

00000000 <string_buffer>:
   0: 7261654c    rsbvc r6, r1, #76, 10   ; 0x13000000
   4: 6e692d6e    cdpvs 13, 6, cr2, cr9, cr14, {3}
   8: 7065642d    rsbvc r6, r5, sp, lsr #8
   c: 593a6874    ldmdbpl  sl!, {r2, r4, r5, r6, fp, sp, lr}
  10: 00617261    rsbeq r7, r1, r1, ror #4
   ...

Disassembly of section .rodata:

00000000 <string_buffer_2>:
   0: 63206f74          ; <UNDEFINED> instruction: 0x63206f74
   4: 74616572    strbtvc  r6, [r1], #-1394  ; 0xfffffa8e
   8: 20612065    rsbcs r2, r1, r5, rrx
   c: 61646f72    smcvs 18162 ; 0x46f2
  10: 73206174          ; <UNDEFINED> instruction: 0x73206174
  14: 69746365    ldmdbvs  r4!, {r0, r2, r5, r6, r8, r9, sp, lr}^
```

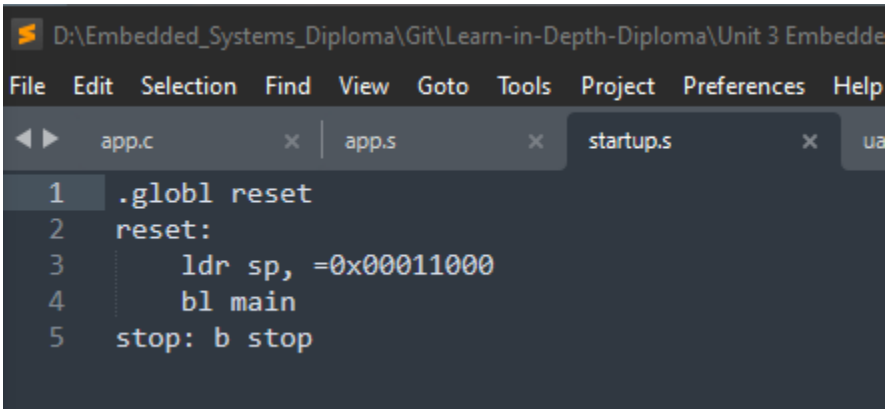
## Step 8] To display full sections content of app.o in hex format using “-s” option with “objdump” binary utility

- Includes any string constant.

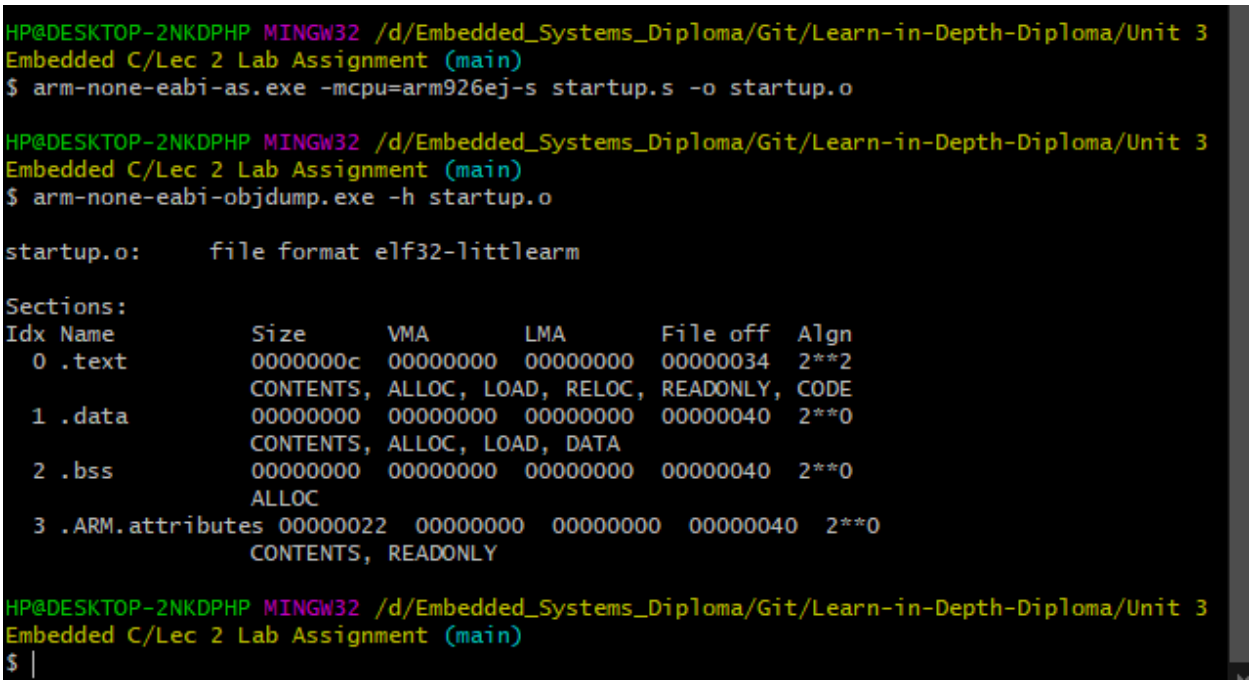
```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3
Embedded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-objdump.exe -s app.o

app.o:     file format elf32-littlearm

Contents of section .text:
 0000 00482de9 04b08de2 08009fe5 feffffeb  .H-.............
 0010 0000a0e1 0088bde8 00000000           ............
Contents of section .data:
 0000 4c656172 6e2d696e 2d646570 74683a59  Learn-in-depth:Y
 0010 61726100 00000000 00000000 00000000  ara.............
 0020 00000000 00000000 00000000 00000000  ................
 0030 00000000 00000000 00000000 00000000  ................
 0040 00000000 00000000 00000000 00000000  ................
 0050 00000000 00000000 00000000 00000000  ................
 0060 00000000                             ....
Contents of section .rodata:
 0000 746f2063 72656174 65206120 726f6461  to create a roda
 0010 74612073 65637469 6f6e0000 00000000  ta section......
 0020 00000000 00000000 00000000 00000000  ................
 0030 00000000 00000000 00000000 00000000  ................
 0040 00000000 00000000 00000000 00000000  ................
 0050 00000000 00000000 00000000 00000000  ................
 0060 00000000                             ....
Contents of section .comment:
 0000 00474343 3a202847 4e552054 6f6f6c73  .GCC: (GNU Tools
 0010 20666f72 2041726d 20456d62 65646465   for Arm Embedde
 0020 64205072 6f636573 736f7273 20372d32  d Processors 7-2
 0030 3031372d 71342d6d 616a6f72 2920372e  017-q4-major) 7.
 0040 322e3120 32303137 30393034 20287265  2.1 20170904 (re
 0050 6c656173 6529205b 41524d2f 656d6265  lease) [ARM/embe
 0060 64646564 2d372d62 72616e63 68207265  dded-7-branch re
 0070 76697369 6f6e2032 35353230 345d00    vision 255204].
Contents of section .ARM.attributes:
 0000 41310000 00616561 62690001 27000000  A1...aeabi..'...
 0010 0541524d 39323645 4a2d5300 06050801  .ARM926EJ-S.....
 0020 09011204 14011501 17031801 19011a01  ................
 0030 1e06                                 ..
```

## Step 9] Startup file

- **.globl** is used to make reset section global to other files (used later in linker_script.ld)
- **Stack pointer** is assigned an address (will be modified when actual address is calculated in linker script)
- In **stop** section, we branch to create a loop in case there is no while (1) loop in app code so program doesn't end.




- Startup is assembled only as its written in assembly code to make initial configuration for the SoC.



## Step 10] Linker Script

- **ENTRY** command used to let debugger know to start at reset section
- **MEMORY** command used to specify name, starting address and memory size in bytes
- (**rwx**) for read/write and executable code sections
- **SECTIONS** command to name the output sections for the executable file
-

app.c | app.s | startup.s | linker_script.ld | uart.h

```
ENTRY(reset)

MEMORY
{
    Mem (rwx) : ORIGIN = 0x00000000, LENGTH = 64M
}

SECTIONS
{
    .startup :
    {
     startup.o(.text)
    }
    .text :
    {
        *(.text)
    }
    .data :
    {
        *(.data)
    }
    .bss :
    {
        *(.bss)
    }
}
```

- To link we pass “-T” option to linker to read the linker script and link the relocatable files

- Analyzing the output executable **.elf file**, the locator in linker used the linker script to map the virtual to the actual addresses.

- We can see both VMA and LMA values are set

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o learn-in-depth.elf

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ 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      0000000c  00000000  00000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000068  0000000c  0000000c  0000800c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .rodata       00000064  00000074  00000074  00008074  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .data         00000064  000000d8  000000d8  000080d8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .ARM.attributes 0000002e  00000000  00000000  0000813c  2**0
                  CONTENTS, READONLY
  5 .comment      00000011  00000000  00000000  0000816a  2**0
                  CONTENTS, READONLY

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ |
```

- We adjusted the linker script to merge input sections into one output section
- Merging (.rodata ) with all *(.data) sections and output to (.data)
- *(COMMON) is used to include any data that don't belong to specific section

D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 2 Lab As...

File Edit Selection Find View Goto Tools Project Preferences Help

app.c | app.s | startup.s | linker_script.ld | uart.h

```
ENTRY(reset)

MEMORY
{
    Mem (rwx) : ORIGIN = 0x00000000, LENGTH = 64M
}

SECTIONS
{
    .startup :
    {
     startup.o(.text)
    }
    .text :
    {
        *(.text)    *(.rodata)
    }
    .data :
    {
        *(.data)
    }
    .bss :
    {
        *(.bss)     *(COMMON)
    }
}
```

- After linking, we can see that the **.rodata** section is no longer visible as it was merged and output to the **.data** of the **.elf file**

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o learn-in-depth.elf

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ 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      0000000c  00000000  00000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         000000cc  0000000c  0000000c  0000800c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000000d8  000000d8  000080d8  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  0000813c  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000816a  2**0
                  CONTENTS, READONLY

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$
```

- **Using Location Counter**
- We set startup at 0x10000 and then the size of each section is incremented to location counter
- We create a **stack_top** symbol and assign it to:
    - ✓ The last location counter value + the stack size 0x1000 = 4KB

File Edit Selection Find View Goto Tools Project Preferences Help

app.c | app.s | startup.s | linker_script.ld | uart.h

```
ENTRY(reset)

MEMORY
{
    Mem (rwx) : ORIGIN = 0x00000000, LENGTH = 64M
}

SECTIONS
{
    . = 0x10000;
    .startup . :
    {
     startup.o(.text)
    }> Mem
    .text :
    {
        *(.text)    *(.rodata)
    }> Mem
    .data :
    {
        *(.data)
    }> Mem
    .bss :
    {
        *(.bss)     *(COMMON)
    }> Mem
    . = . + 0x1000;
    stack_top = . ;
}
```

- SP register is set to Stack_top symbol in startup.s

```
.globl reset
reset:
    ldr sp, =stack_top
    bl main
stop:
    b stop
```

## Step 11] Analyze Symbols in Relocatable and Executable Files

- Addresses are not allocated yet (all 00000000 for symbols that are defined within the object file)
  - ✓ T > .text
  - ✓ D >.data
  - ✓ R >.rodata
  - ✓ U > unresolved symbols
    - startup. o >> main
    - app.o >> Uart_Send_String

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D string_buffer
00000000 R string_buffer_2
         U Uart_Send_String

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-nm.exe uart.o
00000000 T Uart_Send_String
```

- After linking relocatable files to >> **learn-in-depth.elf ,** all the symbols are resolved to their corresponding addresses:
    - ✓ reset symbol = 00010000
    - ✓ stack_top symbol = 0001113c

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in
ded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-nm.exe learn-in-depth.elf
0001000c T main
00010000 T reset
0001113c D stack_top
00010008 t stop
000100d8 D string_buffer
00010074 T string_buffer_2
00010024 T Uart_Send_String
```

- Checking the section headers:
- VMA (Virtual Memory Address) and LMA (Load Memory Address) are set

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplo
$ 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         000000cc  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000100dc  000100dc  000080dc  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  00008140  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000816e  2**0
                  CONTENTS, READONLY
```

## Step 12] Map File

- (-Map=file.map) is passed as an option to the linker to output the map file that includes the results of the linking process to verify:
  - ✓ section names and lengths
  - ✓ symbol locations.

```
MINGW32:/d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec 2 Lab Assignment

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diplom
a/Unit 3 Embedded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-ld.exe -T linker_script.ld -Map=Map_file.map startup.o app.o uart.o -o learn-in-depth.elf
```

- As seen, the below map file results match the instructions given to the linker script:
  - ✓ Memory starting address is 0x00000000
  - ✓ Memory size = 0x04000000 >> 64M Bytes
  - ✓ reset symbol in .startup is at address = 0x0010000

```
D:\Embedded_Systems_Diploma\Git\Learn-in-Depth-Diploma\Unit 3 Embedded C\Lec 2 Lab As...
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help
app.c | app.s | startup.s | Map_file.map | linker_script.ld | uart.h | uart.c

1
2   Memory Configuration
3
4   Name             Origin             Length             Attributes
5   Mem              0x00000000         0x04000000         xrw
6   *default*        0x00000000         0xffffffff
7
8   Linker script and memory map
9
10                   0x00010000                 . = 0x10000
11
12  .startup         0x00010000         0x10
13   startup.o(.text)
14   .text           0x00010000         0x10 startup.o
15                   0x00010000                  reset
16
17  .text            0x00010010         0xcc
18   *(.text)
19   .text           0x00010010         0x18 app.o
20                   0x00010010                  main
21   .text           0x00010028         0x50 uart.o
22                   0x00010028                  Uart_Send_String
23   *(.rodata)
24   .rodata         0x00010078         0x64 app.o
25                   0x00010078                  string_buffer_2
26
```

## Step 13] Using “readelf” binary utility to verify the Entry Point

- Entry point is set to 0x10000 as per linker script.

- Section headers show starting address of each section:
  - ✓ .startup = 00010000
  - ✓ .text = 00010010
  - ✓ .data = 000100dc

- Section sizes:
  - ✓ .startup = 000010
  - ✓ .text = 0000cc
  - ✓ .data = 000064

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3
$ arm-none-eabi-readelf.exe -a learn-in-depth.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x10000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          33224 (bytes into file)
  Flags:                             0x5000002, has entry point, Version5 EABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         1
  Size of section headers:           40 (bytes)
  Number of section headers:         9
  Section header string table index: 6

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .startup          PROGBITS        00010000 008000 000010 00  AX  0   0  4
  [ 2] .text             PROGBITS        00010010 008010 0000cc 00  AX  0   0  4
  [ 3] .data             PROGBITS        000100dc 0080dc 000064 00  WA  0   0  4
  [ 4] .ARM.attributes   ARM_ATTRIBUTES  00000000 008140 00002e 00      0   0  1
  [ 5] .comment          PROGBITS        00000000 00816e 000011 01  MS  0   0  1
```

## Step 14] Output binary file to run on QEMU Emulator

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

- Emulator Output: UART sent string **"Learn-in-depth:Yara"**

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embed
ded C/Lec 2 Lab Assignment (main)
$ /d/Programs/Installed/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-i
n-depth.bin
Learn-in-depth:Yara
```