Unit3_Lesson(2)

1- write codes

I- App.c

```c
#include "Uart.h"

unsigned char string_uart[100]="learn-in-depth:<Hazem>";
void main(){

    Uart_u32GetString ( string_uart );

}
```

II- Uart.c

```c
#include  "Uart.h"


void Uart_u32GetString ( u8* Copy_pu32 ){
    while(*Copy_pu32!='\0'){
        UART_u32UART0DR= *Copy_pu32;
        Copy_pu32++;
    }
}
```

III-Uart.h

```c
#ifndef _UART_H_
#define _UART_H_

typedef unsigned int    u32;
typedef unsigned char   u8;

#define UART_u32UART0DR       *(volatile u32*)((u32*)(0x101f1000))


void Uart_u32GetString ( u8* Copy_pu32 );



#endif
```

IV-startup.s

```
.globl reset
reset:
    ldr sp, = StackTop
    bl main

stop: b stop
```

V-Linker_Script

```
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;
    StackTop = . ;
}
```

2-get obj_file form App.c Uart .c included Uart.h

I- App.o

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

II-Uart.o

arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s Uart.c -o Uart.o

III-startup.o

arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startUp.o

To show sections for object_file

App.o

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ 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
```

Uart.o

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ 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 .comment      0000007f  00000000  00000000  00000088  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000107  2**0
                  CONTENTS, READONLY
```

Startup.o

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ 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
```

To show symbol table for App.o Uart.o and startUp.o

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ arm-none-eabi-nm.exe  app.o
00000000 T main
00000000 D string_uart
         U Uart_u32GetString
```

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ arm-none-eabi-nm.exe  Uart.o
00000000 T Uart_u32GetString
```

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ arm-none-eabi-nm.exe  startUp.o
         U main
00000000 T reset
         U StackTop
00000008 t stop
```

3-use linker_script.ld to get executable_file (learn-in-depth.elf) and map_file.map

arm-none-eabi-ld.exe -T linker_script.ld startUp.o app.o Uart.o -o learn-in-depth.elf -Map=Map_file.map

To show sections for learn-in-depth.elf

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ 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  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000070  00010010  00010010  00010010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010080  00010080  00010080  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000100e4  2**0
                  CONTENTS, READONLY
  4 .comment      0000007e  00000000  00000000  00010112  2**0
                  CONTENTS, READONLY
```

To show symbol table for learn-in-depth.elf

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ arm-none-eabi-nm.exe  learn-in-depth.elf
00010064 T main
00010000 T reset
000110e4 D StackTop
00010008 t stop
00010080 D string_uart
00010010 T Uart_u32GetString
```

4-get binary file to use in burn

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

5- burn binary file on board using qemu

```
hp@DESKTOP-2VPJ56U MINGW32 /f/Assignment_L2
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
learn-in-depth:<Hazem>
```