Lab 1

In this lab: create a BareMetal Software to send “learn-in-depth:<my_name>” using UART.

Requirements:

1- Cross tool chain
2- C code files
3- Startup.s
4- Linker script

Steps:

[1] Cross tool chain

Using ARM-Cross toolchain

[2] C code files [app.c uart.c uart.h]

uart.h

```h
#ifndef _UART_H_
#define  _UART_H_
void uart_send_string (unsigned char *p_tx_string);
#endif
```

uart.c

```c
#include "uart.h"
#define UART0DR  *((volatile unsigned int* const)((unsigned int *)0x101f1000))
void uart_send_string (unsigned char *p_tx_string)
{
    while (*p_tx_string != '\0')
    {
        UART0DR=(unsigned int )(*p_tx_string);
        p_tx_string++;
    }
}
```

app.c

```c
#include "uart.h"
unsigned char string_buffer[100]="Learn-in-depth:<yousef_mossad>";
void main (void)
{
    uart_send_string(string_buffer);
}
```

[3] Startup.s

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

[4] Linker script [linker_script. ld]

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

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

Using ARM-Cross toolchain:

Compiling the files and generating (app.o and uart.o)

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -I. app.c -o app.o

elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -I. uart.c -o uart.o
```

sections with debug for app.o:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_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         00000018  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b0  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  000000b0  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  000000c2  2**0
                  CONTENTS, READONLY
```

sections with debug for uart.o:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ 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, 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 00000032  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

assembling and getting startup.o file:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s -I. startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
```

sections with debug for startup.o:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/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 00000022  00000000  00000000  00000044  2**0
                  CONTENTS, READONLY
```

link all files together to get learn-in-depth.elf:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf
```

To out map_file:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf -Map=output.map
```

sections for learn-in-depth.elf:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_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         00000068  00010010  00010010  00008010  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010078  00010078  00008078  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000080dc  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  0000810a  2**0
                  CONTENTS, READONLY
```

symbols in app.o, uart.o and startup.o:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-nm.exe  app.o
00000000 T main
00000000 D string_buffer
         U uart_send_string

elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-nm.exe  uart.o
00000000 T uart_send_string

elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-nm.exe  startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

Symbols for learn-in-depth.elf:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-nm.exe  learn-in-depth.elf
00010010 T main
00010000 T reset
000110dc D stack_top
00010008 t stop
00010078 D string_buffer
00010028 T uart_send_string
```

To hexa file:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
```

To read the elf file to make sure that the entry point address is correct:

```
$ 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:          33124 (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
```

To run:

```
elbostan@DESKTOP-13N8QSR MINGW64 /d/Diploma Embeded system/units/unit 3/unit3_embeddedC_lesson2/lab_1
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn-in-depth.bin
Learn-in-depth:<yousef_mossad>
```