Unit3_Lesson(2)

write codes

App.c

```c
#include "Uart.h"

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

    Uart_u32GetString ( string_uart );

}
```

Uart.c

```c

#include  "Uart.h"


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

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
```

startup.s

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

stop: b stop
```

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 = . ;
}
```

Makefile

```
CC=arm-none-eabi-
CFLAGS=-g -mcpu=arm926ej-s
INCS= -I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As_s=$(wildcard *.s)
As_o=$(As_s:.s=.o)
project_name= learn-in-depth

all:$(project_name).bin

%.o: %.s
    $(CC)as.exe $(CFLAGS) $< -o $@

%.o: %.c
    $(CC)gcc.exe -c $(CFLAGS) $(INCS)  $< -o $@

$(project_name).elf: $(OBJ) $(As_o)
    $(CC)ld.exe -T linker_script.ld  $(LIBS) startUp.o app.o Uart.o  -o  $@  -Map=Map_file.map

$(project_name).bin: $(project_name).elf
    $(CC)objcopy.exe -O binary $<  $@

clear_all:
    rm *.o *.elf *.bin
```

App.o

mingw32-make.exe app.o

Uart.o

mingw32-make.exe Uart.o

startup.o

mingw32-make.exe 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
```

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

mingw32-make-exe learn-in-depth.elf

To show sections for App.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 App.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
```

Get binary file to use in burn

mingw32-make-exe all

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>
```

For debugging using qemu

```
$ qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
```

to connect the board on qemu using terminal

```
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 8.0.50.20171128-
git
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startUp.s:5
5               ldr sp, = StackTop
(gdb)
```

to display 3 lines assembly and show PC

```
(gdb) x/3i $pc
=> 0x10000 <reset>:     ldr     sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:   bl      0x10010 <main>
   0x10008 <stop>:      b       0x10008 <stop>
(gdb)
```

put breakpoints

```
(gdb) b reset
Breakpoint 1 at 0x10000: file startUp.s, line 5.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 5.
(gdb) b main
Breakpoint 3 at 0x10018: file app.c, line 7.
(gdb) b Uart_u32GetString
Breakpoint 4 at 0x1003c: file Uart.c, line 7.
```

Know the current breakpoints

```
(gdb) info breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x00010000 startUp.s:5
2       breakpoint     keep y   0x00010010 in main at app.c:5
3       breakpoint     keep y   0x00010018 in main at app.c:7
4       breakpoint     keep y   0x0001003c in Uart_u32GetString at Uart.c:7
```

continue to the next breakpoint

```
(gdb) c
Continuing.

Breakpoint 2, main () at app.c:5
5       void main(){
```

see the current file

```
(gdb) l
1
2       #include "Uart.h"
3
4       unsigned char string_uart[100]="learn-in-depth:<Hazem>";
5       void main(){
6
7               Uart_u32GetString ( string_uart );
8
9       }(gdb)
```

move one instruction

```
9       }(gdb) si
0x00010014      5       void main(){
```

move on C line

```
(gdb) s

Breakpoint 3, main () at app.c:7
7               Uart_u32GetString ( string_uart );
```

print a variable's value

```
(gdb) print string_uart[0]
$1 = 108 'l'
```

Watch a variable

```
(gdb) watch string_uart[0]
Hardware watchpoint 5: string_uart[0]
```

see where pc is

```
(gdb) where
#0  main () at app.c:7
```