Lab 1 lecture 2 unit 3

Print string “learn-in-depth:ibrahim” on on QEMU Emulator

1. Create our app.c , UART.c and UART.h files and open it using editor “e.g. Sublime”

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ touch app.c uart.c uart.h

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ subl *
```

2. Write our UART driver

```c
/**
 * @name    : uart.c
 * @date    : sept 3, 2024
 * @author  : Eng.Ibrahim El-mursi
 * @brief   : this program is UART driver for versatilePB platform
 */
#include "uart.h"
#define UART0DR *((volatile uint32_t*)(uint32_t)(0x101f1000))

void uart_send_string(uint8_t* str)
{
    while(*str !='\0')
    {
        UART0DR=(uint32_t) (*str);
        str++;
    }
}
```

```c
/**
 * @name    : uart.h
 * @date    : sept 3, 2024
 * @author  : Eng.Ibrahim El-mursi
 * @brief   : this program is UART driver for versatilePB platform
 *
 */
#ifndef _UART_H
#define _UART_H
#include <stdint.h>
void uart_send_string(uint8_t*);
#endif
```

3. Write our app.c program

```c
/**
 * @name    : app.c
 * @date    : sept 3, 2024
 * @author  : Eng.Ibrahim El-mursi
 * @brief   : this program is used to Print string
 *            “learn-in-depth:ibrahim” on versatilePB platform
 *            using UART protocal
 */
#include "uart.h"
uint8_t str[] ="learn-in-depth:ibrahim";
const uint8_t str1[] ="learn-in-depth:ibrahim";
void main(void)
{
    uart_send_string(str);
}
```

4. Generate our object files (relocatable files) app.o , uart.o
   a. With debug sections

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s app.c -o app.o

Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s uart.c -o uart.o
```

   b. Without debug sections

5. Navigate the object files using arm binary utilities
   a. With debug sections

   Command:

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-objdump.exe -h app.o
```

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-objdump.exe -h uart.o
```

   Output:

```
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         00000018  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000064  2**0
                  ALLOC
  3 .rodata       00000018  00000000  00000000  00000064  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   000000cd  00000000  00000000  0000007c  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 0000008a  00000000  00000000  00000149  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    0000002c  00000000  00000000  000001d3  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000001ff  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   0000007c  00000000  00000000  0000021f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000000e6  00000000  00000000  0000029b  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      00000012  00000000  00000000  00000381  2**0
                  CONTENTS, READONLY
 11 .ARM.attributes 00000032  00000000  00000000  00000393  2**0
                  CONTENTS, READONLY
 12 .debug_frame  0000002c  00000000  00000000  000003c8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

```
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 .debug_info   000000b1  00000000  00000000  00000084  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000069  00000000  00000000  00000135  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  0000019e  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000001ca  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000084  00000000  00000000  000001ea  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000000ee  00000000  00000000  0000026e  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000035c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  0000036e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000028  00000000  00000000  000003a0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

   b. Without debug sections

   Command:

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s app.c -o app.o

Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s uart.c -o uart.o
```

Output:

```
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         00000018  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000064  2**0
                  ALLOC
  3 .rodata       00000018  00000000  00000000  00000064  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .comment      00000012  00000000  00000000  0000007c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000032  00000000  00000000  0000008e  2**0
                  CONTENTS, READONLY

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (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         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
```

6. Generate assemply file using arm binary utilities

   Command:

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-objdump.exe -D uart.o > uart.s

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-objdump.exe -D app.o > app.s
```

Output:

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


Disassembly of section .text:

00000000 <main>:
   0: e92d4800    push  {fp, lr}
   4: e28db004    add   fp, sp, #4
   8: e59f0004    ldr   r0, [pc, #4]   ; 14 <main+0x14>
   c: ebfffffe    bl 0 <uart_send_string>
  10: e8bd8800    pop   {fp, pc}
  14: 00000000    andeq r0, r0, r0

Disassembly of section .data:

00000000 <str>:
   0: 7261656c    rsbvc r6, r1, #108, 10  ; 0x1b000000
   4: 6e692d6e    cdpvs 13, 6, cr2, cr9, cr14, {3}
   8: 7065642d    rsbvc r6, r5, sp, lsr #8
   c: 693a6874    ldmdbvs  sl!, {r2, r4, r5, r6, fp, sp, lr}
  10: 68617262    stmdavs  r1!, {r1, r5, r6, r9, ip, sp, lr}^
  14: 00006d69    andeq r6, r0, r9, ror #26

Disassembly of section .rodata:

00000000 <str1>:
   0: 7261656c    rsbvc r6, r1, #108, 10  ; 0x1b000000
   4: 6e692d6e    cdpvs 13, 6, cr2, cr9, cr14, {3}
   8: 7065642d    rsbvc r6, r5, sp, lsr #8
   c: 693a6874    ldmdbvs  sl!, {r2, r4, r5, r6, fp, sp, lr}
  10: 68617262    stmdavs  r1!, {r1, r5, r6, r9, ip, sp, lr}^
  14: 00006d69    andeq r6, r0, r9, ror #26

Disassembly of section .comment:

00000000 <.comment>:
   0: 43434700    movtmi   r4, #14080  ; 0x3700
   4: 4728203a          ; <UNDEFINED> instruction: 0x4728203a
   8: 2029554e    eorcs r5, r9, lr, asr #10
   c: 2e372e34    mrccs 14, 1, r2, cr7, cr4, {1}
  10: Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
   0: 00003141    andeq r3, r0, r1, asr #2
   4: 61656100    cmnvs r5, r0, lsl #2
   8: 01006962    tsteq r0, r2, ror #18
   c: 00000027    andeq r0, r0, r7, lsr #32
  10: 4d524105    ldfmie   f4, [r2, #-20] ; 0xfffffec
  14: 45363239    ldrmi r3, [r6, #-569]!  ; 0x239
  18: 00532d4a    subseq   r2, r3, sl, asr #26
  1c: 01080506    tsteq r8, r6, lsl #10
  20: 04120109    ldreq r0, [r2], #-265   ; 0x109
  24: 01150114    tsteq r5, r4, lsl r1
  28: 01180317    tsteq r8, r7, lsl r3
  2c: 011a0119    tsteq sl, r9, lsl r1
  30: Address 0x00000030 is out of bounds.
```

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


Disassembly of section .text:

00000000 <uart_send_string>:
   0: e52db004    push  {fp}     ; (str fp, [sp, #-4]!)
   4: e28db000    add   fp, sp, #0
   8: e24dd00c    sub   sp, sp, #12
   c: e50b0008    str   r0, [fp, #-8]
  10: ea000006    b  30 <uart_send_string+0x30>
  14: e59f3030    ldr   r3, [pc, #48]  ; 4c <uart_send_string+0x4c>
  18: e51b2008    ldr   r2, [fp, #-8]
  1c: e5d22000    ldrb  r2, [r2]
  20: e5832000    str   r2, [r3]
  24: e51b3008    ldr   r3, [fp, #-8]
  28: e2833001    add   r3, r3, #1
  2c: e50b3008    str   r3, [fp, #-8]
  30: e51b3008    ldr   r3, [fp, #-8]
  34: e5d33000    ldrb  r3, [r3]
  38: e3530000    cmp   r3, #0
  3c: 1afffff4    bne   14 <uart_send_string+0x14>
  40: e28bd000    add   sp, fp, #0
  44: e8bd0800    ldmfd sp!, {fp}
  48: e12fff1e    bx lr
  4c: 101f1000    andsne   r1, pc, r0

Disassembly of section .comment:

00000000 <.comment>:
   0: 43434700    movtmi   r4, #14080  ; 0x3700
   4: 4728203a          ; <UNDEFINED> instruction: 0x4728203a
   8: 2029554e    eorcs r5, r9, lr, asr #10
   c: 2e372e34    mrccs 14, 1, r2, cr7, cr4, {1}
  10: Address 0x00000010 is out of bounds.


Disassembly of section .ARM.attributes:

00000000 <.ARM.attributes>:
   0: 00003141    andeq r3, r0, r1, asr #2
   4: 61656100    cmnvs r5, r0, lsl #2
   8: 01006962    tsteq r0, r2, ror #18
   c: 00000027    andeq r0, r0, r7, lsr #32
  10: 4d524105    ldfmie   f4, [r2, #-20] ; 0xfffffec
  14: 45363239    ldrmi r3, [r6, #-569]!  ; 0x239
  18: 00532d4a    subseq   r2, r3, sl, asr #26
  1c: 01080506    tsteq r8, r6, lsl #10
  20: 04120109    ldreq r0, [r2], #-265   ; 0x109
  24: 01150114    tsteq r5, r4, lsl r1
  28: 01180317    tsteq r8, r7, lsl r3
  2c: 011a0119    tsteq sl, r9, lsl r1
  30: Address 0x00000030 is out of bounds.
```

7. Create startup.s file

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ touch startup.s

Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ subl startup.s
```

8. Write our startup code

```
1  .global reset
2  reset:
3      ldr sp, =stack_top
4      bl main
5  stop: b stop
6
```

9. Generate startup.o file and analyze it using arm binary utilities

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ 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
```

10. Create linker_script.ld file

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ touch linker_script.ld

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/unit_3/lesson_2/lab (main)
$ subl linker_script.ld
```

11. Write our linker script and define stack_top symbol

```
ENTRY(reset)

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

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

12. Read symbols for each object file before linking

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64
it_3/lesson_2/lab (main)
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D str
00000000 R str1
         U uart_send_string

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64
it_3/lesson_2/lab (main)
$ arm-none-eabi-nm.exe uart.o
00000000 T uart_send_string

Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64
it_3/lesson_2/lab (main)
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
00000008 t stop
```

13. Linking all files, generate executable file and map file

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-ld.exe -T linker_script.ld *.o -o app.elf -Map=map_file.map
```

14. Read symbols for executable file after linking

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-nm.exe app.elf
00010010 T main
00010000 T reset
000110a8 D stack_top
00010008 t stop
00010090 D str
00010078 R str1
00010028 T uart_send_string
```

15. Symbols in executable file

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLS0 MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-objdump.exe -h app.elf

app.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 .rodata       00000018  00010078  00010078  00008078  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .data         00000018  00010090  00010090  00008090  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .ARM.attributes 0000002e  00000000  00000000  000080a8  2**0
                  CONTENTS, READONLY
  5 .comment      00000011  00000000  00000000  000080d6  2**0
                  CONTENTS, READONLY
```

16. Generate binary file

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ arm-none-eabi-objcopy.exe -O binary app.elf app.bin
```

17. Check if QEMU support this machine

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ qemu-system-arm.exe -machine help
Supported machines are:
akita                Sharp SL-C1000 (Akita) PDA (PXA270)
ast2500-evb          Aspeed AST2500 EVB (ARM1176)
bast                 Simtec Electronics BAST (S3C2410A, ARM920T)
borzoi               Sharp SL-C3100 (Borzoi) PDA (PXA270)
canon-a1100          Canon PowerShot A1100 IS
cheetah              Palm Tungsten|E aka. Cheetah PDA (OMAP310)
collie               Sharp SL-5500 (Collie) PDA (SA-1110)
connex               Gumstix Connex (PXA255)
cubieboard           cubietech cubieboard
emcraft-sf2          SmartFusion2 SOM kit from Emcraft (M2S010)
highbank             Calxeda Highbank (ECX-1000)
imx25-pdk            ARM i.MX25 PDK board (ARM926)
integratorcp         ARM Integrator/CP (ARM926EJ-S)
kzm                  ARM KZM Emulation Baseboard (ARM1136)
lm3s6965evb          Stellaris LM3S6965EVB
lm3s811evb           Stellaris LM3S811EVB
mainstone            Mainstone II (PXA27x)
mcimx7d-sabre        Freescale i.MX7 DUAL SABRE (Cortex A7)
midway               Calxeda Midway (ECX-2000)
mps2-an385           ARM MPS2 with AN385 FPGA image for Cortex-M3
mps2-an505           ARM MPS2 with AN505 FPGA image for Cortex-M33
mps2-an511           ARM MPS2 with AN511 DesignStart FPGA image for Cortex-M3
musicpal             Marvell 88w8618 / MusicPal (ARM926EJ-S)
n800                 Nokia N800 tablet aka. RX-34 (OMAP2420)
n810                 Nokia N810 tablet aka. RX-44 (OMAP2420)
netduino2            Netduino 2 Machine
none                 empty machine
nuri                 Samsung NURI board (Exynos4210)
palmetto-bmc         OpenPOWER Palmetto BMC (ARM926EJ-S)
raspi2               Raspberry Pi 2
realview-eb          ARM RealView Emulation Baseboard (ARM926EJ-S)
realview-eb-mpcore   ARM RealView Emulation Baseboard (ARM11MPCore)
realview-pb-a8       ARM RealView Platform Baseboard for Cortex-A8
realview-pbx-a9      ARM RealView Platform Baseboard Explore for Cortex-A9
romulus-bmc          OpenPOWER Romulus BMC (ARM1176)
sabrelite            Freescale i.MX6 Quad SABRE Lite Board (Cortex A9)
smdk2443             smdk2443 (ARM920-T)
smdkc210             Samsung SMDKC210 board (Exynos4210)
spitz                Sharp SL-C3000 (Spitz) PDA (PXA270)
sx1                  Siemens SX1 (OMAP310) V2
sx1-v1               Siemens SX1 (OMAP310) V1
terrier              Sharp SL-C3200 (Terrier) PDA (PXA270)
tosa                 Sharp SL-6000 (Tosa) PDA (PXA255)
tt                   OpenTom (ARM920-T)
tt666                OpenTom (ARM920-T)
verdex               Gumstix Verdex (PXA270)
versatileab          ARM Versatile/AB (ARM926EJ-S)
versatilepb          ARM Versatile/PB (ARM926EJ-S)
vexpress-a15         ARM Versatile Express for Cortex-A15
vexpress-a9          ARM Versatile Express for Cortex-A9
```

18. Run the app.bin on the QEMU emulator

```
Eng.Ibrahim El-mursi@DESKTOP-II7CLSO MINGW64 /d/learn_in_depth/git/first_term/un
it_3/lesson_2/lab (main)
$ qemu-system-arm.exe -M versatilepb -m 128M -nographic -kernel app.bin
dsound: Could not initialize DirectSoundCapture
dsound: Reason: No sound driver is available for use, or the given GUID is not a
 valid DirectSound device ID
learn-in-depth:ibrahim
```