By Alfred Fayez

LAB 1 Report

Creating Baremetal SW with ARM cross toolchain & Simulating it on VersatilePB virtual board in QEMU

Contents

Table of Figures

By Alfred Fayez

1. Write source files and extracting object files then analyzing them

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-gcc.exe  -c -g -I . -mcpu=arm926ej-s UART.c -o UART.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ ls *.o
UART.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-obj
arm-none-eabi-objcopy.exe  arm-none-eabi-objdump.exe

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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 .debug_info   0000005c  00000000  00000000  00000084  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000051  00000000  00000000  000000e0  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000131  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000015d  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000003d  00000000  00000000  0000017d  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000006a  00000000  00000000  000001ba  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  00000224  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  00000236  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000028  00000000  00000000  00000268  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

Snippet 1 uart.o sections with debug

Note: since the linker script isn't linked with these object files, all the addresses (VMA/LMA) aren't correct yet.

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s app.c -o app.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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 .debug_info   0000006b  00000000  00000000  000000b0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 00000058  00000000  00000000  0000011b  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000002c  00000000  00000000  00000173  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  0000019f  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000035  00000000  00000000  000001bf  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000068  00000000  00000000  000001f4  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000012  00000000  00000000  0000025c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000032  00000000  00000000  0000026e  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000002c  00000000  00000000  000002a0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

Snippet 2 app.o sections with debug

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s app.c -o app.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-gcc.exe -c -I . -mcpu=arm926ej-s UART.c -o UART.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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
```

Snippet 3 uart.o & app.o sections without debug

2. Writing Startup Code and Extracting its Object File and Analyzing it

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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         0000000c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000040  2**0
                  ALLOC
  3 .ARM.attributes 00000022  00000000  00000000  00000040  2**0
                  CONTENTS, READONLY
```

Snippet 4 startup.o sections

3. Write Linker_Script And Linking It With Other Object Files To Get .elf File And Analizing it Using Binary Utilities (objdump)

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-ld.exe   -T linker_script.ld  startup.o app.o UART.o -o learn-in-depth.elf -Map=Map_file.map

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-nm.exe  learn-in-depth.elf
0001000c T main
00010000 T reset
00001000 D stack_top
00010008 t stop
000100d8 D string_buffer
00010074 T string_buffer2
00010024 T UART_Send_string

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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  00010000  00010000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         000000cc  0001000c  0001000c  0000800c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  000100d8  000100d8  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
```

Snippet 5 .elf file sections

Note: Here we can see that the addresses are corrected (Entry point is 0x00010000) to the physical addresses after linking with the linker script

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ 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:          33220 (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 00000c 00  AX  0   0  4
  [ 2] .text             PROGBITS        0001000c 00800c 0000cc 00  AX  0   0  4
  [ 3] .data             PROGBITS        000100d8 0080d8 000064 00  WA  0   0  4
  [ 4] .ARM.attributes   ARM_ATTRIBUTES  00000000 00813c 00002e 00      0   0  1
  [ 5] .comment          PROGBITS        00000000 00816a 000011 01  MS  0   0  1
  [ 6] .shstrtab         STRTAB          00000000 00817b 000049 00      0   0  1
  [ 7] .symtab           SYMTAB          00000000 00832c 000180 10      8  18  4
  [ 8] .strtab           STRTAB          00000000 0084ac 000066 00      0   0  1
```

Snippet 6 reading information about the elf file

4. Symbol Table Of The Object Files & The .elf File

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-nm.exe app.o
00000000 T main
00000000 D string_buffer
00000000 R string_buffer2
         U UART_Send_string

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-nm.exe UART.o
00000000 T UART_Send_string

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
00000008 t stop

Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ arm-none-eabi-nm.exe learn-in-depth.elf
0001000c T main
00010000 T reset
00001000 D stack_top
00010008 t stop
000100d8 D string_buffer
00010074 T string_buffer2
00010024 T UART_Send_string
```

Snippet 7 Symbol tables

Note: some of the symbols in object files are Unresolved, but all the symbols in the .elf file are linked and resolved.

5. Simulation on QEMU

```
Alfred@DESKTOP-2EK343U MINGW64 /e/Embedded/Unit_3/lesson_2/assignment/lab1
$ qemu-system-arm -M versatilepb -m 128m -nographic -kernel learn-in-depth.bin
Learn-in-depth: Alfred Fayez
```

Snippet 8 QEMU simulation