makefile

```
1 PROJECT_NAME=Lab_2_ARM_Cortex_M3
 2 CC=arm-none-eabi-
 3     CFLAGS=-mcpu=cortex-m3
 4 INCS=-I.
   LIBS=
 5
 6
    C_OBJ_SRC=$(wildcard *.c)
 7
    C_OBJ=$(C_OBJ_SRC:.c=.o)
 8
    S_OBJ_SRC=$(wildcard *.s)
      S_OBJ=$(S_OBJ_SRC:.s=.o)
 9
10
11
     build: $(PROJECT_NAME).bin
12
     13
     %.o: %.c
14
15
      $(CC)gcc.exe $(CFLAGS) -c $(INCS) $< -o $@
16
17
     %.o: %.s
        $(CC)as.exe $(CFLAGS) $< -o $@
18
19
20
    $(PROJECT NAME).elf: $(C OBJ) $(S OBJ)
21
         $(CC)ld.exe --script linker_script.ld $(LIBS) $(C_OBJ) $(S_OBJ) -o $(PROJECT_NAME)
22
23
     $(PROJECT_NAME).bin: $(PROJECT_NAME).elf
         $(CC)objcopy.exe -0 binary $< $@
24
25
26
     assembly:
27
         $(CC)objdump.exe -D $(PROJECT_NAME).elf > $(PROJECT_NAME).s
28
         @echo "======Generated Assembly from .elf file========="
29
30
     clean_all:
31
        rm *.o *.elf *.bin *.map $(PROJECT_NAME).s
     clean_exe:
33
        rm *.elf *.bin *.map
34
```

main.c

```
1 #include <stdint.h>
 2
 3
    #define RCC BASE
                             0x40021000
    #define PORTA BASE
                              0x40010800
 5
     #define RCC_APB2ENR
                              *(volatile uint32_t*)(RCC_BASE+0x18)
 7
     #define GPIOA CRH
                              *(volatile uint32_t*)(PORTA_BASE+0x04)
 8
     #define GPIOA ODR
                              *(volatile uint32_t*)(PORTA_BASE+0x0C)
 9
    int main(void)
10
11
12
         RCC\_APB2ENR = 1 << 2;
13
         GPIOA CRH &=0xFF0FFFFF;
         GPIOA_CRH |=0x00200000;
14
15
         while (1)
16
         {
             GPIOA ODR =1 << 13;
17
18
             for (int i=0; i<5000; i++);
19
             GPIOA ODR \&=\sim(1<<13);
20
             for (int i=0; i<5000; i++);
21
22
         return 0;
23
24
```

Part1: startup.s

```
1
     .section .vectors
 2
     .word
             0x20001000
 3
     .word
              reset
     .word vector_handler
.word vector_handler
 4
                                   /* 2 NMI */
                                   /* 3 Hard Fault */
 5
                                   /* 4 MM Fault */
     .word vector_handler
 6
            vector handler
                                   /* 5 Bus Fault */
 7
     .word
     .word vector_handler
                                 /* 6 Usage Fault */
 8
                                  /* 7 ..... */
             /* To IRQ67 */
9
10
11
     .section .text
12
    reset:
13
        bl main
14
        b .
15
16
   .thumb func
     vector handler:
17
        b _reset
18
```

linker script.ld

```
ENTRY(.vectors)
2
3
    MEMORY
4
5
        FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128k
6
        SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
7
8
9
     SECTIONS
10
11
         .text : {
                 *(.vectors*) /*Expected to be 7*4=28B=1C*/
12
                 *(.text*) /*Depend on code size in main.c and startup.s*/
13
14
                 *.rodata /*Expected to be 0B*/
15
         } >FLASH
16
         .data : {
17
                 *(.data) /*Expected to be 0B*/
18
19
            } >FLASH
20
21
                 *(.bss) /*Expected to be 0B*/
22
23
             } >SRAM
24
```

map file.map

```
2
           Memory Configuration
       3
       4
           Name
                           Origin
                                              Length
       5
           FLASH
                           0x08000000
                                              0x00020000
       6
           SRAM
                           0x20000000
                                              0x00005000
       7
           *default*
                           0x00000000
                                              0xffffffff
       8
       9
           Linker script and memory map
      10
      11
      12
                          0x08000000
           .text
                                           0xa0
           *(.vectors*)
      13
                                                =28B
            .vectors
      14
                          0x08000000
                                               startup.o
            *(.text*)
      15
      16
                          0x0800001c
                                           0x7c main.o
      17
                          0x0800001c
                                                   main
      18
            .text
                          0x08000098
                                           0x8 startup.o
      19
            *.rodata()
39
      .data
                         0x080000a0
                                               0x0
       *(.data)
40
41
       .data
                         0x080000a0
                                               0x0 main.o
       .data
                                               0x0 startup.o
42
                         0x080000a0
47
      .bss
                         0x20000000
                                               0x0
48
       *(.bss)
49
        .bss
                         0x20000000
                                               0x0 main.o
50
       .bss
                         0x20000000
                                               0x0 startup.o
```

Disassembly of .elf file

```
5
     Disassembly of section .text:
 6
 7
     08000000 <main-0x1c>:
8
      8000000:
                 20001000
                             andcs r1, r0, r0
9
                             stmdaeq r0, {r3, r4, r7}
      8000004:
                 08000098
                             stmdaeq r0, {r0, r1, r2, r3, r4, r7}
10
      8000008:
                 0800009f
11
      800000c:
                 0800009f
                             stmdaeq r0, {r0, r1, r2, r3, r4, r7}
12
      8000010:
                 0800009f
                             stmdaeq r0, {r0, r1, r2, r3, r4, r7}
                 0800009f
                             stmdaeq r0, {r0, r1, r2, r3, r4, r7}
13
      8000014:
                 0800009f
                             stmdaeq r0, {r0, r1, r2, r3, r4, r7}
14
     8000018:
15
```

- 0x20001000 stored @0x08000000 in FLASH, Initialize SP @0x20001000.
- 0x080000098 stored @0x08000004 in FLASH, 0x080000098 is _reset handler address.
- From 0x080000008 to 0x080000018 which is 28B is our vector table, 0x0800009f is stored in all of them, 0x0800009f is vector_handler address, now we know that vector_handler is alias for 5 handlers.
- Now expected that at runtime CPU go to entry point at 0x08000000 and initialize SP then go to 0x08000004 which branch execution to _reset @0x08000098 and _reset will branch execution to main @0x0800001c.
- Now expected that main instructions start @0x0800001c and ends @0x08000094 then_reset starts @0x08000098 and ends @0x0800009c then vector handler starts.

- The question here is the address of vector_handler is 0x0800009e and the vector section stores 0x0800009f which is after one byte of 0x0800009e, and the instruction itself e7fb takes 2 bytes which supposed be 0x0800009fe and 0x0800009f
- I tried to debug using QEMU, but the debugger cannot show what happens since it don't stop at breakpoints unless complete the interrupt handler

```
16 ▼ 0800001c <main>:

      800001c:
      b480
      push {r7}

      800001e:
      b083
      sub sp, #12

      8000020:
      af00
      add r7, sp, #0

      8000022:
      4b1a
      ldr r3, [pc, #104] @ (800008c <main+0x70>)

      8000024:
      681b
      ldr r3, [r3, #0]

      8000026:
      4a19
      ldr r2, [pc, #100] @ (800008c <main+0x70>)

           800001c:
                                                     push {r7}
 17
 19
 20
  21
 22
 23
            8000028: f043 0304 orr.w r3, r3, #4
            800002c: 6013 str r3, [r2, #0]
            25
  26
  27
            8000034: f423 0370 bic.w r3, r3, #15728640 @ 0xf00000
  28
            8000038: 6013 str r3, [r2, #0]

800003a: 4b15 ldr r3, [pc, #84] @ (8000090 <main+0x74>)

800003c: 681b ldr r3, [r3, #0]

800003e: 4a14 ldr r2, [pc, #80] @ (8000090 <main+0x74>)

8000040: f443 1300 orr.w r3, r3, #2097152 @ 0x200000
  29
  30
 31
  32
  33
           34
  35
 36
  37
            800004c: f443 5300 orr.w r3, r3, #8192
8000050: 6013 str r3, [r2, #0]
8000052: 2300 movs r3, #0
  38
 39
 40
            8000052: 2300 movs r3, #0
8000054: 607b str r3, [r7, #4]
8000056: e002 b.n 800005e <main+0x42>
8000058: 687b ldr r3, [r7, #4]
800005a: 3301 adds r3, #1
800005c: 607b str r3, [r7, #4]
800005c: 687b ldr r3, [r7, #4]
 41
 42
 43
 44
 45
 46
            8000060: f241 3287 movw r2, #4999 @ 0x1387
           8000064: 4293 cmp r3, r2
8000066: ddf7 ble.n 8000058 <main+0x3c>
8000068: 4b0a ldr r3, [pc, #40] @ (8000094 <main+0x78>)
800006a: 681b ldr r3, [r3, #0]
 48
 49
 50
 51
            800006c: 4a09 ldr r2, [pc, #36] @ (8000094 <main+0x78>)
 52
            800006e: f423 5300 bic.w r3, r3, #8192 @ 0x2000
 53
           800006e: f423 5300 bic.w r3, r3, #8192 @ 0x2
8000072: 6013 str r3, [r2, #0]
8000074: 2300 movs r3, #0
8000076: 603b str r3, [r7, #0]
8000078: e002 b.n 8000080 <main+0x64>
800007a: 683b ldr r3, [r7, #0]
800007c: 3301 adds r3, #1
800007e: 603b str r3, [r7, #0]
8000080: 683b ldr r3, [r7, #0]
 54
 56
 57
 58
 59
 60
 61
            8000082: f241 3287 movw r2, #4999 @ 0x1387
 62
           8000086: 4293 cmp r3, r2

8000088: ddf7 ble.n 800007a <main+0x5e>

800008a: e7dc b.n 8000046 <main+0x2a>

800008c: 40021018 andmi r1, r2, r8, lsl r0

8000090: 40010804 andmi r0, r1, r4, lsl #16

8000094: 4001080c andmi r0, r1, ip, lsl #16
 64
 65
 66
 67
 68
 69
           08000098 <_reset>:
 71
           8000098: f7ff ffc0 bl 800001c <main>
 72
            800009c: e7fe b.n 800009c <_reset+0x4>
 73
         0800009e <vector handler>:
            800009e: e7fb b.n 8000098 <_reset>
```

Part2: startup.c

Without alias:

```
#include <stdint.h>
 1
 2
     #define stack top 0x20001000
 3
     extern int main(void);
 5
     //void Default_handler(void);
 7
     void Reset_handler(void);
 8
     void NMI_handler(void)__attribute__((weak));
9
     void HW_fault_handler(void)__attribute__((weak));
     void MM_fault_handler(void)__attribute__((weak));
     void Bus_fault_handler(void)__attribute__((weak));
11
     void Usage_fault_handler(void)__attribute__((weak));
12
13
     uint32_t vectors[]_attribute__((section(".vectors")))={
14
15
         stack_top,
16
          (uint32 t) &Reset handler,
17
          (uint32_t) &NMI_handler,
          (uint32_t) &HW_fault_handler,
18
19
          (uint32_t) &MM_fault_handler,
20
          (uint32_t) &Bus_fault_handler,
21
          (uint32_t) &Usage_fault_handler,
22
23
24
     void Reset_handler(void){
25
         main();
26
     void NMI_handler(void){Reset_handler();}
27
     void HW fault handler(void){Reset handler();}
29
     void MM_fault_handler(void){Reset_handler();}
30
     void Bus_fault_handler(void){Reset_handler();}
31
     void Usage_fault_handler(void){Reset_handler();}
32
33
     void Default_handler(void){
34
         Reset_handler();
35
     */
36
```

linker_script.ld

```
ENTRY(.vectors)
     MEMORY
         FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128k
6
         SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
     }
                 *(.vectors*) /<mark>*Expected to be 7*4=28B=1C</mark>*/
                 *(.text*) /*Depend on code size in main.c and startup.c*/
                *.rodata /*Expected to be 0B*/
         } >FLASH
         .data : { *(.data) /*Expected to be 0B*/
19
             } >FLASH
         } >SRAM
24
```

map file.map

```
1
2
     Memory Configuration
3
4
     Name
                      Origin
                                         Length
                                                            Attributes
5
     FLASH
                      0x08000000
                                         0x00020000
                                                            xr
     SRAM
                      0x20000000
                                         0x00005000
6
                                                            xrw
7
     *default*
                                         0xffffffff
                      0x00000000
8
9
     Linker script and memory map
10
11
12
     .text
                     0x08000000
                                      0xe0
      *(.vectors*)
13
14
      .vectors
                     0x08000000
                                      0x1c startup.o
15
                     0x08000000
                                               vectors
      *(.text*)
16
                     0x0800001c
17
      .text
                                      0x7c main.o
18
                     0x0800001c
19
      .text
                     0x08000098
                                      0x48 startup.o
20
                     0x08000098
                                               Reset handler
                      0x080000a4
21
                                               NMI handler
22
                     0x080000b0
                                               HW fault handler
                                               MM fault handler
23
24
                                               Bus fault handler
25
                                               Usage_fault_handler
     *.rodata()
26
        .data
                            0x080000e0
                                                  0x0
 47
        *(.data)
 48
         .data
                            0x080000e0
                                                  0x0 main.o
         .data
 49
                            0x080000e0
                                                  0x0 startup.o
 54
         .bss
                            0x20000000
                                                   0x0
          *(.bss)
 55
  56
          .bss
                             0x20000000
                                                   0x0 main.o
  57
          .bss
                             0x20000000
                                                   0x0 startup.o
```

• Each fault handler has its own address.

Disassembly of .elf file

```
Disassembly of section .text:
5
6
 7 ▼ 08000000 <vectors>:
      8000000: 20001000
                             andcs r1, r0, r0
8
9
      8000004:
                              stmdaeq r0, {r0, r3, r4, r7}
      8000008:
                              stmdaeq r0, {r0, r2, r5, r7}
10
      800000c:
                              stmdaeq r0, {r0, r4, r5, r7}
                              stmdaeq r0, {r0, r2, r3, r4, r5, r7}
12
      8000010:
                              stmdaeq r0, {r0, r3, r6, r7}
13
      8000014:
14
      8000018:
                              stmdaeq r0, {r0, r2, r4, r6, r7}
15
```

```
0800001c <main>:
                                            {r7}
sp, #12
r7, sp, #0
r3, [pc, #104] @ (800008c <main+0x70>)
           800001c:
                       h489
           8000020:
                        af00
                                      add
           8000022
                        4b1a
                                      ldr
  20
21
22
23
24
25
26
27
28
                                             r3,
                                                 [r3, #0]
           8000024:
                        681b
                                      ldr
                                      ldr r2, [pc, #100] @ (800008c <main+0x70>)
orr.w r3, r3, #4
           8000026:
                        4a19
                                            r3, [r2, #0]
r3, [pc, #96] @ (8000090 <main+0x74>)
r3, [r3, #0]
           800002c:
                        6013
                                      str
           800002e:
8000030:
                        4b18
                                      ldr
ldr
                        681b
                                      ldr r2, [pc, #92] @ (8000090 <main+0x74>)
bic.w r3, r3, #15728640 @ 0xf00000
           8000032
                        4a17
                        f423 0370
                                            r3, [r2, #0]
r3, [pc, #84] @ (8000090 <main+0x74>)
r3, [r3, #0]
  29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
           8000038:
                        6013
                                      str
           800003a:
800003c:
                                      ldr
ldr
                        4b15
                        681b
                                     rs, [rs, #0] dr r2, [pc, #80] @ (8000090 <main+0x74>) orr.w r3, r3, #2097152 @ 0x200000 str r3, [r2, #0] dr r3, [pc, #76] @ (8000094 <main+0x78>) dr r3, [r3, #0]
           8000030
                        4a14
           8000040:
                        f443 1300
           8000044:
                        6013
           8000046:
8000048:
                        4b13
681b
                                     In r2, [r2, #0] (8000094 <main+0x78>)
orr.w r3, r3, #8192 @ 0x2000
str r3, [r2, #0]
movs r3, #0
           800004a:
800004c:
                        4a12
f443 5300
           8000050:
                        6013
           8000052:
                        2300
                                            r3, [r7, #4]
800005e <main+0x42>
r3, [r7, #4]
           8000054:
                        607b
                                      str
           8000056:
8000058:
                        e002
                                      b.n
ldr
                        687b
                                     1dr r3, [r7, m*]
adds r3, #1
str r3, [r7, #4]
ldr r3, [r7, #4]
movw r2, #4999 @ 0x1387
cmp r3, r2
           800005a:
                        3301
  45
46
47
48
                        607b
           800005e:
                        687b
           8000060:
                        f241 3287
           8000064:
                        4293
                                      ble.n 8000058 <main+0x3c>
  49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
           8000066:
                        ddf7
                                     800006a:
                        681b
           800006c:
                        4200
           800006e:
                        f423 5300
           8000072:
                        6013
           8000074:
8000076:
                       2300
603b
                                     str r3, [r7, #0]
b.n 8000080 <main+0x64>
ldr r3, [r7, #0]
adds r3, #1
str r3, [r7, #0]
ldr r3, [r7, #0]
movw r2, #4999 @ 0x1387
cmp r3, r2
           8000078:
800007a:
                        e002
683b
           800007c:
                        3301
           800007e:
                        603b
           8000080:
                        683b
           8000082:
8000086:
                        f241 3287
4293
                                      ble.n 800007a <main+0x5e>
           8000088:
                        ddf7
                                     b.n 8000046 (main+0x2a)
andmi r1, r2, r8, lsl r0
andmi r0, r1, r4, lsl #16
andmi r0, r1, ip, lsl #16
  65
66
67
68
                        e7dc
                        40021018
           800008c:
                       40010804
4001080c
           8000090:
           8000094:
         08000098 <Reset_handler>:
 70
 71
            8000098:
                              b580
                                                   push {r7, lr}
                                                   add r7, sp, #0
 72
            800009a:
                              af00
 73
                              f7ff ffbe
            800009c:
                                                  bl 800001c <main>
            80000a0:
 74
                              bf00
                                                   nop
 75
            80000a2:
                              bd80
                                                            {r7, pc}
                                                   pop
 76
 77
 78
            80000a4:
                              b580
                                                   push {r7, lr}
 79
                              af00
            80000a6:
                                                   add r7, sp, #0
 80
            80000a8:
                               f7ff fff6
                                                   bl 8000098 <Reset_handler>
 81
            80000ac:
                              bf00
                                                  nop
 82
            80000ae:
                              bd80
                                                            {r7, pc}
                                                   pop
 83
 84
          080000b0 <HW fault handler>
 85
                                                           {r7, lr}
            80000b0:
                              b580
                                                   push
 86
            80000b2:
                               af00
                                                   add
                                                             r7, sp, #0
 87
            80000b4:
                              f7ff fff0
                                                   bl 8000098 <Reset_handler>
                              bf00
 88
            80000b8:
                                                   nop
 89
            80000ba:
                              bd80
                                                   рор
                                                             {r7, pc}
 90
 91
 92
            80000bc:
                              b580
                                                   push
                                                            {r7, lr}
 93
                              af00
            80000be:
                                                            r7, sp, #0
                                                   add
                              f7ff ffea
                                                   bl 8000098 <Reset_handler>
            80000c0:
 95
            80000c4:
                              bf00
                                                   nop
 96
            80000c6:
                              bd80
                                                            {r7, pc}
                                                   pop
 97
 98
 99
            80000c8:
                              b580
                                                            {r7, lr}
                                                   push
                               af00
            80000ca:
                                                   add
                                                            r7, sp, #0
101
            80000cc:
                               f7ff ffe4
                                                   bl 8000098 <Reset_handler>
                              bf00
            80000d0:
                                                   nop
                              bd80
103
            80000d2:
                                                   pop
                                                             {r7, pc}
104
                              b580
            80000d4:
                                                   push
                                                             {r7, lr}
107
                              af00
            80000d6:
                                                   add
                                                             r7, sp, #0
                               f7ff ffde
            80000d8:
                                                   bl 8000098 <Reset_handler>
109
            80000dc:
                              bf00
                                                   nop
110
          80000de:
                              bd80
                                                   pop
                                                            {r7, pc}
```

With alias:

```
#include <stdint.h>
 1
     #define stack top 0x20001000
 3
 4
     extern int main(void);
 5
     void Default handler(void);
 6
 7
     void Reset handler(void);
8 void NMI_handler(void)__attribute__((weak,alias("Default_handler")));
     void HW_fault_handler(void)__attribute__((weak,alias("Default_handler")));
 9
     void MM_fault_handler(void)__attribute__((weak,alias("Default_handler")));
10
     void Bus_fault_handler(void)__attribute__((weak,alias("Default_handler")));
11
     void Usage_fault_handler(void)__attribute__((weak,alias("Default_handler")));
12
13
     uint32 t vectors[] attribute ((section(".vectors")))={
14
15
         stack_top,
16
         (uint32_t) &Reset_handler,
17
         (uint32_t) &NMI_handler,
         (uint32 t) &HW fault handler,
18
19
         (uint32_t) &MM_fault_handler,
20
         (uint32_t) &Bus_fault_handler,
         (uint32 t) &Usage fault handler,
21
22
     };
23
24
     void Reset_handler(void){
25
         main();
26
27
     void Default handler(void){
28
         Reset_handler();
29
30
```

linker_script.ld

```
ENTRY(.vectors)
 1
 2
 3
     MEMORY
         FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128k
 6
         SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
 7
 8
9
     SECTIONS
10
                 *(.vectors*) /*Expected to be 7*4=28B=1C*/
12
13
                 *(.text*) /*Depend on code size in main.c and startup.c*/
                 *.rodata /*Expected to be 0B*/
14
         } >FLASH
15
16
         .data : {
    *(.data) /*Expected to be 0B*/
17
18
19
             } >FLASH
20
         23
             } >SRAM
24
```

map file.map

```
1
 2
      Memory Configuration
 3
 4
      Name
                       Origin
                                           Length
                                                               Attributes
 5
      FLASH
                       0x08000000
                                           0x00020000
                                                               xr
      SRAM
                                           0x00005000
 6
                       0x20000000
                                                               xrw
 7
      *default*
                                           0xffffffff
                       0x00000000
 8
 9
      Linker script and memory map
10
11
12
      .text
                      0x08000000
                                        0xb0
       *(.vectors*)
13
14
       .vectors
                      0x08000000
                                        0x1c startup.o
15
                      0x08000000
                                                 vectors
16
       *(.text*)
17
       .text
                      0x0800001c
                                        0x7c main.o
18
                      0x0800001c
19
       .text
                      0x08000098
                                        0x18 startup.o
20
                      0x08000098
                                                 Reset handler
21
                      0x080000a4
                                                 Bus fault handler
22
                      0x080000a4
                                                 HW_fault_handler
23
                      0x080000a4
                                                 Usage_fault_handler
24
                      0x080000a4
                                                 MM_fault_handler
25
                       0x080000a4
                                                 Default_handler
26
                      0x080000a4
                                                 NMI_handler
27
       *.rodata()
 47
         .data
                             0x080000b0
                                                    0x0
          *(.data)
  48
          .data
                                                    0x0 main.o
  49
                             0x080000b0
  50
          .data
                             0x080000b0
                                                    0x0 startup.o
  55
         .bss
                              0x20000000
                                                     0x0
          *(.bss)
  56
  57
          .bss
                              0x20000000
                                                     0x0 main.o
  58
          .bss
                              0x20000000
                                                     0x0 startup.o
```

 All of fault handlers have the same address of Default_handler, since all of them are alias for Default_handler

Disassembly of .elf file

```
5 Disassembly of section .text:
 7
      08000000 <vectors>:
       8000000:
                  20001000
 8
                               andcs r1, r0, r0
 9
       8000004:
                               stmdaeq r0, {r0, r3, r4, r7}
10
       8000008:
                  080000a5
                               stmdaeq
                                       r0, {r0, r2, r5, r7}
11
       800000c:
                  080000a5
                               stmdaeq r0, {r0, r2, r5, r7}
                               stmdaeq r0, {r0, r2, r5, r7}
12
       8000010:
                  080000a5
13
       8000014:
                  080000a5
                               stmdaeq r0, {r0, r2, r5, r7}
14
       8000018:
                  080000a5
                               stmdaeq r0, {r0, r2, r5, r7}
15
```

```
0800001c <main>:
17
      800001c:
                              push {r7}
       800001e:
                  h083
                              sub
18
                                    sp, #12
19
       8000020:
                  af00
                              add
                                    r7, sp, #0
                                    r3, [pc, #104] @ (800008c <main+0x70>)
      8000022:
                  4b1a
                              ldr
21
       8000024:
                  681b
                              ldr
                                    r3, [r3, #0]
                                   r2, [pc, #100] @ (800008c <main+0x70>)
22
       8000026:
                              ldr
                  4a19
23
       8000028:
                  f043 0304
                             orr.w r3, r3, #4
24
       800002c:
                  6013
                              str
                                  r3, [r2, #0]
                  4b18
25
       800002e:
                              ldr
                                    r3, [pc, #96] @ (8000090 <main+0x74>)
26
       8000030:
                  681b
                              ldr
                                    r3, [r3, #0]
                                   r2, [pc, #92] @ (8000090 <main+0x74>)
       8000032:
                  4a17
                              1dr
28
       8000034:
                  f423 0370
                             bic.w r3, r3, #15728640 @ 0xf00000
                              str
       8000038:
                                   r3, [r2, #0]
                  6013
30
       800003a:
                  4b15
                              ldr
                                    r3, [pc, #84]
                                                   @ (8000090 <main+0x74>)
31
       800003c:
                  681b
                              ldr
                                   r3, [r3, #0]
       800003e:
                  4a14
                              ldr
                                    r2, [pc, #80]
                                                  @ (8000090 <main+0x74>)
33
       8000040:
                  f443 1300
                             orr.w r3, r3, #2097152 @ 0x200000
       8000044:
                  6013
                              str
                                  r3, [r2, #0]
       8000046:
                                   r3, [pc, #76]
                                                  @ (8000094 <main+0x78>)
                  4b13
                              ldr
       8000048:
                                   r3, [r3, #0]
36
                  681b
                              ldr
37
       800004a:
                              ldr
                                   r2, [pc, #72]
                                                   @ (8000094 <main+0x78>)
                  4a12
38
       800004c:
                  f443 5300
                             orr.w r3, r3, #8192
                                                  @ 0x2000
       8000050:
                  6013
                              str r3, [r2, #0]
40
       8000052:
                  2300
                              movs r3, #0
41
       8000054:
                  607b
                              str
                                   r3, [r7, #4]
                                    800005e <main+0x42>
42
       8000056:
                  e002
                              b.n
       8000058:
43
                  687b
                              ldr
                                    r3, [r7, #4]
44
       800005a:
                  3301
                              adds
                                   r3, #1
       800005c:
                              str
                                   r3, [r7, #4]
45
                  607b
46
       800005e:
                  687b
                              ldr
                                    r3, [r7, #4]
                                                @ 0x1387
47
       8000060:
                  f241 3287
                             movw r2, #4999
48
       8000064:
                  4293
                              cmp
                                   r3, r2
49
       8000066:
                  ddf7
                              ble.n 8000058 <main+0x3c>
       8000068:
                  4b0a
                              1dr
                                    r3, [pc, #40] @ (8000094 <main+0x78>)
       800006a:
                  681b
                              ldr
                                    r3, [r3, #0]
       800006c:
                                                  @ (8000094 <main+0x78>)
                  4a09
                              1dr
                                   r2, [pc, #36]
       800006e:
                  f423 5300
                              bic.w r3, r3, #8192 @ 0x2000
54
       8000072:
                  6013
                              str r3, [r2, #0]
       8000074:
                  2300
                              movs r3, #0
56
       8000076:
                                   r3, [r7, #0]
                  603b
                              str
57
       8000078:
                  e002
                              b.n
                                   8000080 <main+0x64>
58
       800007a:
                  683b
                              ldr
                                   r3, [r7, #0]
                              adds r3, #1
       800007c:
                  3301
       800007e:
                              str r3, [r7, #0]
                  603b
                              ldr
61
       8000080:
                  683b
                                   r3, [r7, #0]
       8000082:
                  f241 3287
                              movw r2, #4999
                                               @ 0x1387
63
       8000086:
                  4293
                              cmp r3, r2
64
       8000088:
                  ddf7
                              ble.n 800007a <main+0x5e>
65
       800008a:
                  e7dc
                              b.n 8000046 <main+0x2a>
       800008c:
                  40021018
                              andmi r1, r2, r8, lsl r0
67
       8000090:
                  40010804
                              andmi r0, r1, r4, lsl #16
       8000094:
                  4001080c
                              andmi r0, r1, ip, lsl #16
69
  70
  71
          8000098:
                        b580
                                               {r7, lr}
                                       push
  72
          800009a:
                        af00
                                               r7, sp, #0
                                       add
  73
                        f7ff ffbe
                                       bl 800001c <main>
          800009c:
  74
                        bf00
          80000a0:
                                       nop
  75
          80000a2:
                        bd80
                                               {r7, pc}
                                       pop
  76
  77
        080000a4 <Default handler>
  78
          80000a4:
                        b580
                                               {r7, lr}
  79
                        af00
          80000a6:
                                       add
                                               r7, sp, #0
  80
          80000a8:
                        f7ff fff6
                                       bl 8000098 <Reset_handler>
  81
                        bf00
          80000ac:
                                       nop
  82
          80000ae:
                        bd80
                                       pop
                                               {r7, pc}
```

Part3: startup.c (alias & .data & .bss)
main.c

```
1
      #include <stdint.h>
 2
 3
      #define RCC BASE
                                0x40021000
 4
      #define PORTA_BASE
                                0x40010800
 5
                                *(volatile uint32_t*)(RCC_BASE+0x18)
 6
      #define RCC APB2ENR
                                *(volatile uint32_t*)(PORTA_BASE+0x04)
*(volatile uint32_t*)(PORTA_BASE+0x0C)
      #define GPIOA CRH
      #define GPIOA_ODR
 8
 Q
10
      volatile char DataArr[]={1,2,3}; //for 3B data section
11
      volatile struct A {
12
          char ch;
13
          uint32_t var;
14
      } BssStruct; //for 8B bss section
15
16
17
18
      int main(void)
19
           RCC APB2ENR =1<<2;
20
21
           GPIOA_CRH &=0xFF0FFFFF;
22
           GPIOA_CRH |=0x00200000;
23
          while (1)
24
25
               GPIOA_ODR |=1<<13;
               for (int i=0; i<5000; i++);
26
27
               GPIOA ODR \&=\sim(1<<13);
               for (int i=0; i<5000; i++);
29
30
           return 0;
31
```

startup.c

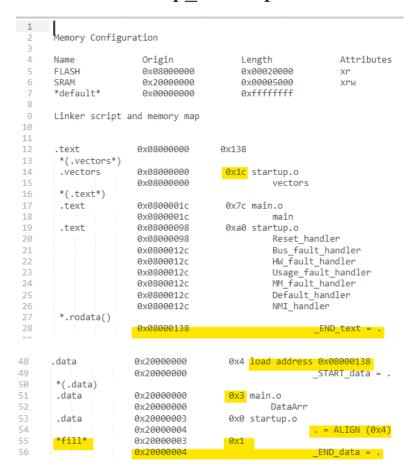
```
#include <stdint.h>
 1
 2
      #define NULL (void*)0
 3
 4
      extern int main(void);
      void Default_handler(void);
 5
 6
 7
      extern uint32 t stack top;
 8
      void Reset_handler(void);
9
      void NMI_handler(void)__attribute__((weak,alias("Default_handler")));
      void HW_fault_handler(void)__attribute__((weak,alias("Default_handler")));
10
      void MM_fault_handler(void)__attribute__((weak,alias("Default_handler")));
void Bus_fault_handler(void)__attribute__((weak,alias("Default_handler")));
11
12
13
      void Usage_fault_handler(void)__attribute__((weak,alias("Default_handler")));
14
15
      uint32_t vectors[]_attribute__((section(".vectors")))={
16
           (uint32_t) &stack_top,
17
           (uint32_t) &Reset_handler,
18
           (uint32 t) &NMI handler,
19
           (uint32_t) &HW_fault_handler,
20
           (uint32_t) &MM_fault_handler,
21
           (uint32 t) &Bus fault handler,
22
           (uint32 t) &Usage fault handler,
23
24
25
      extern uint32_t _END_text;
26
      extern uint32_t _START_data;
27
      extern uint32_t _END_data;
      extern uint32_t _START_bss;
28
29
      extern uint32 t END bss;
30
```

```
31
      void Reset_handler(void){
32
33
          uint32 t section size;
34
          uint8_t *p_src, *p_dist;
35
36
          //copy .data to SRAM
37
          section_size=(uint8_t*)&_END_data - (uint8_t*)&_START_data;
38
          p src=(uint8 t*)& END text;
39
          p dist=(uint8 t*)& START data;
          for (int i = 0; i < section_size; i++, p_dist++, p_src++)
40
41
42
              *p dist=*p src;
43
44
45
          //create .bss in SRAM with 0
          section_size=(uint8_t*)&_END_bss - (uint8_t*)&_START_bss;
47
          p_src=NULL;
48
          p dist=(uint8 t*)& START bss;
          for (int i = 0; i < section_size; i++, p_dist++, p_src++)
49
50
51
               *p_dist=0;
52
53
54
          //branch to main
          main();
55
56
57
      void Default handler(void){
58
59
          Reset handler();
60
```

linker_script.ld

```
ENTRY(vectors)
1
     MEMORY
4 ▼
          FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 128k
5
          SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
6
     }
8
9
      SECTIONS
10 ▼
11 ▼
          .text : {
                  *(.vectors*) /*Expected to be 7*4=28B=1C*/
12
13
                  *(.text*) /*Depend on code size in main.c and startup.c*/
                  *.rodata /*Expected to be 0B*/
15
                   _END_text = . ;
16
          } >FLASH
17
18
19 ▼
          .data : ALIGN(4) {
                   _START_data = . ;
20
21
                  *(.data) /*Expected to be 0B*/
                  . = ALIGN(4);
                   END_data = . ;
              }> SRAM AT>FLASH
24
25
26 ▼
          .bss : ALIGN(4) {
                   START_bss = . ;
                  *(.bss) /*Expected to be 0B*/
                 . = ALIGN(4);
_END_bss = . ;
29
              } >SRAM
31
32
              . = . + 0x1000; /*not inside bss because this assignment change location counter .*/
33
              stack_top = . ; /*if inside bss the size of bss will increase 1000B*/
34
```

map_file.map



• *fill* of 1B is for Alignment, the size of global initialized variable is 3B.

```
0x20000004
61
                                          0x8 load address 0x0800013c
      .bss
62
                       0x20000004
                                                           START bss =
63
       *(.bss)
64
                       0x20000004
                                         0x8 main.o
       .bss
65
                       0x20000004
                                                  BssStruct
                       0x2000000c
66
                                          0x0 startup.o
                                                            = ALIGN (0x4)
67
                       0x2000000c
                                                           END_bss =
                       0x20000000
                                                           . = (. + 0x1000)
                       0x2000100c
69
                                                          stack_top = .
```

• The size of initialized struct taken by the 8 because of auto alignment in C structures.

sections of .elf file

```
arm-none-eabi-objdump -h Lab_2_ARM_Cortex_M3.elf
Lab_2_ARM_Cortex_M3.elf:
                             file format elf32-littlearm
Sections:
Idx Name
                  Size
                                       LMA
                                                 File off
                                                           Algn
                  00000138
                            08000000
                                      08000000
                                                           2**2
 0 .text
                                                 00001000
                  CONTENTS,
                            ALLOC, LOAD, READONLY, CODE
                                                           2**2
                  00000004
                            20000000 08000138
  1 .data
                                                 00002000
                  CONTENTS,
                            ALLOC, LOAD, DATA
  2 .bss
                  80000008
                            20000004
                                      0800013c
                                                 00002004
                                                           2**2
                  ALLOC
                  00000043
                                      00000000
  3 .comment
                            00000000
                                                 00002004
                                                           2**0
                  CONTENTS,
                            READONLY
  4 .ARM.attributes 0000002d 00000000
                                        00000000
                                                   00002047
                                                             2**0
                  CONTENTS, READONLY
```

Symbols of .elf file