Politecnico di Milano
SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Advanced Operating Systems

A.A. 2017-2018 – Exam date: February, 13th 2018

Prof. William FORNACIARI

Surname (readable)....................................... Name (readable)...........................

Matr.. Signature.......................................

Q1	Q2	TOT

NOTES

It is forbidden to refer to texts or notes of any kind as well as interact with their neighbors. Anyone found in possession of documents relating to the course, although not directly relevant to the subject of the examination will cancel the test. It is not allowed to leave during the first half hour, the task must still be returned, even if it is withdrawn. The presence of the writing (not delivered) implies the renunciation of any previous ratings.

Question Q1 (10 points)

Describe the main aspects of a RTOS, with particular emphasis on the characteristics of the computation, time constraints and main scheduling algorithms.

Question Q2 (13 points)

Consider the following application designed for a microcontroller with 256KByte of on-chip FLASH placed in the memory map at address 0, and 64KB of on-chip RAM placed at address 0x30000000.

File: linker.ld

```
ENTRY(Reset_Handler)




_stack_top = 0x30000000+64*1024;
SECTIONS {
    . = 0;
    .text : {
        KEEP(*(.isr_vector))
        *(.text)
        . = ALIGN(4);
        *(.rodata)
        . = ALIGN(4);
        __init_array_start = .;
        KEEP (*(.init_array))
        __init_array_end = .;
    } > flash
    . = ALIGN(8);
    _etext = .;
    .data : {
        _data = .;
        *(.data)
        . = ALIGN(8);
        _edata = .;
    } > ram AT > flash






    _end = .;
}
```

File: main.cpp

```
#include <stdio.h>

int x;
int y=15;
int z;

class Foo { public: Foo() { z++; } };

Foo a,b;

int main() {
    initialize_serial(); //From now on assume printf works
    Printf("%d %d %d\n",x,y,z);
}
```

File: startup.s

```
            .syntax unified
            .cpu cortex-m4
            .thumb
            .section .text
            .global Reset_Handler
            .type  Reset_Handler, %function
Reset_Handler:
            ldr  r0, =_data
            ldr  r1, =_edata
            ldr  r2, =_etext
            cmp  r0, r1
            bne  nodata
            subs r2, r2, #4
dataloop:ldr  r3, [r2, #4]!
            str  r3, [r0], #4
            cmp  r1, r0
            bne  dataloop
nodata:     ldr  r0, =_bss_start
            ldr  r1, =_bss_end
            cmp  r0, r1
            beq  nobss
            movs r3, #0
bssloop:    str  r3, [r0], #4
            cmp  r1, r0
            bne  bssloop
nobss:      ldr  r4, =__init_array_start
            ldr  r5, =__init_array_end
            cmp  r4, r5
            beq  noctor
ctorloop:ldr  r3, [r4], #4
            blx  r3
            blx  r3
            cmp  r5, r4
            bne  ctorloop
noctor:     bl   main
loop:       b    loop

            .section .isr_vector
            .global __Vectors
__Vectors:
            .word  _stack_top
            .word  Reset_Handler
```

1. Fill in the missing parts of linker.ld
2. What does the program print?
3. Fix startup.s
4. What does the fixed program print?

File: linker.ld

```
ENTRY(Reset_Handler)
MEMORY {
   flash(rx) : ORIGIN = 0x00000000, LENGTH = 256K
   ram(wx) : ORIGIN = 0x30000000, LENGTH =   64K
}
_stack_top = 0x30000000+64*1024;
SECTIONS {
   . = 0;
   .text : {
      KEEP(*(.isr_vector))
      *(.text)
      . = ALIGN(4);
      *(.rodata)
      . = ALIGN(4);
      __init_array_start = .;
      KEEP (*(.init_array))
      __init_array_end = .;
   } > flash
   . = ALIGN(8);
   _etext = .;
   .data : {
      _data = .;
      *(.data)
      . = ALIGN(8);
      _edata = .;
   } > ram AT > flash
   _bss_start = .;
   .bss : {
      *(.bss)
      . = ALIGN(8);
   } > ram
   _bss_end = .;
   _end = .;
}
```

File: main.cpp

```
#include <stdio.h>

int x;
int y=15;
int z;

class Foo { public: Foo() { z++; } };

Foo a,b;

int main() {
   initialize_serial(); //From now on assume printf works
   Printf("%d %d %d\n",x,y,z);
}
```

File: startup.s

```
            .syntax unified
            .cpu cortex-m4
            .thumb
            .section .text
            .global Reset_Handler
            .type  Reset_Handler, %function
Reset_Handler:
            ldr  r0, =_data
            ldr  r1, =_edata
            ldr  r2, =_etext
            cmp  r0, r1
            beq  nodata
            subs r2, r2, #4
dataloop:ldr  r3, [r2, #4]!
            str  r3, [r0], #4
            cmp  r1, r0
            bne  dataloop
nodata:  ldr  r0, =_bss_start
            ldr  r1, =_bss_end
            cmp  r0, r1
            beq  nobss
            movs r3, #0
bssloop: str  r3, [r0], #4
            cmp  r1, r0
            bne  bssloop
nobss:   ldr  r4, =__init_array_start
            ldr  r5, =__init_array_end
            cmp  r4, r5
            beq  noctor
ctorloop:ldr  r3, [r4], #4
            blx  r3
            cmp  r5, r4
            bne  ctorloop
noctor:  bl   main
loop:    b    loop

            .section .isr_vector
            .global __Vectors
__Vectors:
            .word  _stack_top
            .word  Reset_Handler
```

5. Fill in the missing parts of linker.ld
6. What does the program print?   0 a random value 4
7. Fix startup.s
8. What does the fixed program print? 0 15 2