Total Time Allowed: **120 Minutes** Maximum Marks: **60**

i. The examination has 5 pages and 5 questions. Answer all the questions. State any **assumptions**.
ii. To earn maximum credit, your answer must be concise, to the point and in the given space
iii. All questions are not of the same difficulty and value. Consider this when allocating time for their solutions.

34/60

1. (a) Name and describe the four classifications for real-time systems. Given an example of each.

MARKS: 8

Hard Real time → military missile, once fired cannot stop till completed mission.
Soft Real time → microwave, can be stopped at any time without completion.
Real real time → digital clock
Firm Real time → car engines could have missing beats pistons but will still work.

1/8

this is mechanical not electro

→ where are the descriptions?

b) List and briefly explain the three requirements of a Real Time Operating System (RTOS), which distinctly sets it apart from a standard OS.

MARKS: 6

* Conditional execution allows for ITE functions
* Use of NVIC for better interupt management.
* Can use one of the 4 classifications of Real time system therefore allowing for more specific operations.

0/6

2. Indicate (in the space provided) whether the following statements are TRUE or FALSE. To obtain full marks for each question, include SHORT comments in support of your answer.

MARKS: 12 (2 each)

a) The tail chaining feature in the ARM Cortex-M3 is used to improve context switching.
TRUE x or FALSE ___?

tail chaining is part of the NVIC. ? OK

b) The instruction *BL <addr>* moves the PC address found in R14 to the Link Register R15, and then branches to the location specified in <addr>.
TRUE ___ or FALSE x ?

PC is R15 Link Register is R14

c) The NVIC system in the Cortex-M3 supports vectored interrupts and dynamic priority changes.
TRUE x or FALSE ___?

It also supports tail-chaining and dynamic interrupts

d) ARM's R-Series line consist of CPUs dedicated to Real-Time applications.
TRUE x or FALSE ___?

A-Series are for High-performance, R-Series for Real-Time, and M-series. ?

e) A CISC-type instruction set may not be implemented as a Von Neumann model.
TRUE ___ or FALSE x ?

CISC-type instructions are 16 bit therefore can be implemented as a Von Neumann model. No!

f) The instruction `ADDS.W R2,#1` belongs to ARM Thumb2's 16-bit instruction set.
TRUE ___ or FALSE X ?

W is for word which is 32-Bits

3. Consider the following equation:

$$f = \frac{\sum_{i=0}^{50} z}{\sum_{i=0}^{30} x * y}$$

Write a non-preemptive multitasking application needed to calculate the equation above, in C. Assume $x = 2$, $y = 3$, $z = 10$, and that a minimum of 3 tasks must be employed. Assign task priorities as required, and specify any parameters that must be enabled (or disabled) in the RTX kernel (i.e. RTX_Conf_CM).

MARKS: 14

```
#include <stdio.h>
#include "LPC17xx.h"
#include <RTL.h>

OS_TID id1, id2;
int x, y, z, sum, f, countA, countB;
__task void taskA (void);
__task void taskB (void);
__task void taskC (void);

__task void taskA (void) {
    if (countA <= 50)
    {
        z += 2;
    }
    else
    {
        os_evt_set (0x0001, id1);
        os_tsk_delete_self();
    }
}

__task void taskB(void) {
    id1 = os_tsk_self();
    if (os_evt_wait_and(0x0001, 0xFFFF)
    {
        if (countB <=30) {
            tempVar = x * y;
            sum += tempVar;
        }
    }
}

__task void taskC (void) {
    id2 = os_tsk_self();
    os_evt_wait_and (0x0002, 0xFFFF);
    f = z / sum;
    os_tsk_delete_self();
}

int main(void) {
    SystemInit();
    os_tsk_create (taskA, 0x03);
    os_tsk_create (taskB, 0x01);
    os_tsk_create (taskC, 0x01);
    os_tsk_sss_init (taskA);
    os_tsk_delete_self();
}
```

10/2

1/1

FINITE!

2/2

3/3

2 1/2

5.5 Not in

6

4. (a) Consider the following C code snippet:

```
r1 = r2 + (r3/2);
r5 = r1 + (r4*8);
```

Convert the code above to ARMv7 assembly. Assume that the compiler flag -O3 has been set, and that there is no dead code elimination.

MARKS: 5

```
ADD R1 R2 R3 LSR #2
ADD R5 R1 R4 LSL #3
```

4
5

4. (b) Consider the following C code snippet:

```
if(a >= b){
        a = c - b;
        c = 0;
}else{
        a = c + b;
        c = 5;
}
```

Convert the code above to ARMv7 assembly. Assume that the compiler flag -O3 has been set, and that there is no dead code elimination.

MARKS: 5

```
CMP
ITTEE GE R1, R2
SUBGE R1 R2 R3
MOVGE R3 #0
ADDLT R1 R3 R4
MOVLT R3, #5
```

4.5
5

5. (a) Consider the SRAM address 0x40080087. Assuming the ARM Cortex-M3 system architecture, calculate the Bit Band address needed to directly access bit 7 of the specified peripheral address. Show all your calculations.

MARKS: 5

0x40080087 − 0x40000000 = 0x00080087 ?

BB = 0x42000000 + (0x00080087 × 0x00000010) + (7×4)

BB = 0x42000000 + (0x00080087 × 0x00000010) + 1C

BB = 0x42000000 + 0x010010E0 + 1C

BB = 0x430010FC ✓

? what are all these ts? funds??

3.5/5

5. (b) Integrate the bit banding address calculated in (a) into the C code below. The program should setup the address as a variable to be used in main. The main function should set the bit band variable to 1, delay for 50 ticks using os_delay(), and then clear the same variable (i.e. to 0).

MARKS: 5

```
#include <stdio.h>
#include "LPC17xx.h"

SET R* = 0x430010FC ?

~~R1~~

int R1* = R; ?
```

define ——————— 4/5

```
int main(void){

R1 = 1; ✓
  os_delay(50); ✓
R1 = 0; ✓

}
```