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.

43.5/60

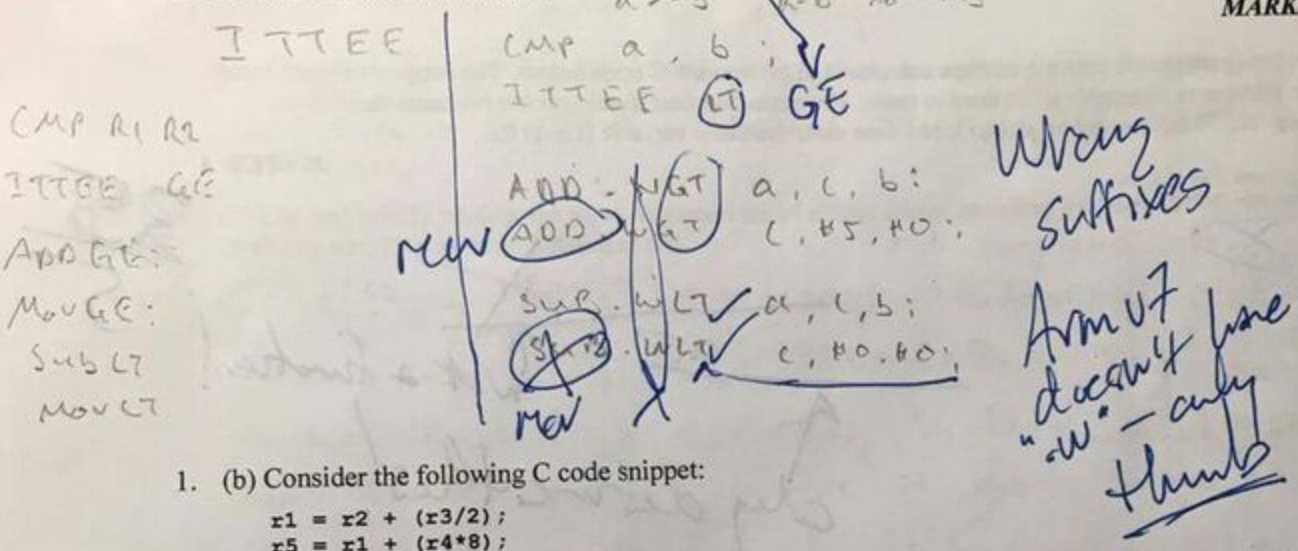
1. (a) Consider the following C code snippet:

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

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

2.5/5

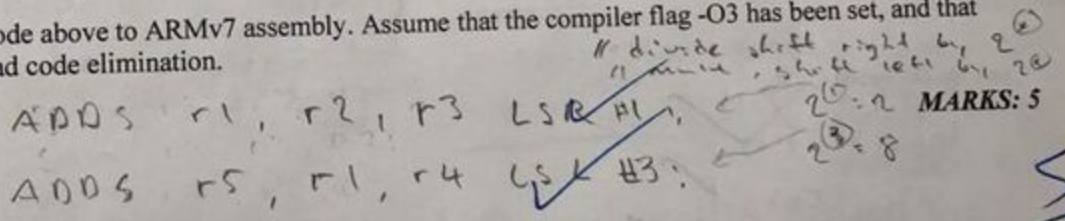


1. (b) Consider the following C code snippet:

```
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



5/5

7.5

2. (a) Consider the SRAM address 0x20080087. Assuming the ARM Cortex-M3 system architecture, calculate the Bit Band address needed to directly access bit 7 of the specified SRAM address. Show all your calculations. **MARKS: 5**

bit # : 7 — what is this??

BB add = 0x22000000 + (Byte offset * 32) + (1(7))

Byte offset : 0x20080087 − 0x20000000

= 0x 80087

Bit band address = 0x2300107F

4.5/5

2. (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"
volatile unsigned bit
#define bitband (0x22000000 + (0x20 * 0x80087) - (0x4 * 0x7))
#define led_1 (*(volatile unsigned int *) 0x2300107F)

// delay function

int main(void){
    led_1 = 1;
    os_delay(50);
    led_1 = 0;
}
```

3.5/5

volatile

Wat a function!

why define this!

#define BB_ADR = (i' (volatile unsigned
0x

8

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

MARKS: 8

The following below are the four classifications for real-time systems.

1) Soft - Real Time System: A system where if it misses an output once in a while it is still acceptable. For example, an mp3 player playing a song. OK

2) Firm - Real Time System: A soft-real time system where if it misses an output often enough, it can cause harm. "malfunction" For example, a pacemaker.

3) Hard - Real Time System: A system where the whole system will fail if output does not work 100% of the time. For example, car breaks.

4) Real - Real Time System: A hard-real time system with a very fast response time. For example, launching a missile.

8/8

3. (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

A standard OS must assign tasks, perform task switching, and communicate and synchronize tasks. Real Time Operating Systems are catered to real time systems where it must know what to expect in terms of delays. The RTX kernel must be implemented/used. The RTOS connects the USER to the CMSIS as well. X Nothing to do w/ general RTOS

Kernel must be used in OS as well ⊗

1) Timing behaviour must be predictable

2) RTOS must manage tasks

3) Be fast

2/6

10

4. Consider the following equation:

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

Write a non-preemptive multitasking application needed to calculate the equation above, in C. Assume $x = 3$, $y = 10$, $z = 2$, 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**

```
// in RTX_Conf_CM Round Robin box must be checked off
#include <stdio.h>
#include "LPC17xx.h"
#include <RTL.h>
double answer, result1, result2;
int x=3, y=10, z=2, i=0;

__task void task1 (void);
__task void task2 (void);
__task void task3 (void);

__task void task1 (void) {   // 1 task for numerator, 1 for denominator
                             // 1 for division
  for (i=0; i<=50; i++){
    result1 += x * y;
  }
  os_delete_self();
}

__task void task2 (void) {
  for (i=0; i<=30; i++){
    result2 += y;
  }
  os_delete_self();
}

__task void task3 (void) {   // division
  answer = result1 / result2;

  os_delete_self();
}

int main (void) {
  SystemInit();
  os_create_task (task1, 5);   // non preemptive cannot
  os_create_task (task2, 4);   // be interrupted
  os_create_task (task3, 4);
  os_delete_self ();
  os_sys_init(increase)
}
```

1/2 OK & off timers?

11/14

1/1

1.5/2

1.5/2

2/2

2/3 2/2 So set diff priorities!

OK

11

5. 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)

False

last sed has nothing to do with CPU chip

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

Von Neumann only has 1 memory for instructions and data. CISC-type has complex large instructions that take multiple clock cycles. CISC-type instruction sets are implemented in Harvard requires hardware that understands instructions are multiple seq

False

.W means it's wide so 32 bits

b) The instruction `ADDS.W R0,#1` belongs to ARM Thumb2's 16-bit instruction set.
TRUE ✓ or FALSE ___?

W = 32b

ARM Thumb 2 uses 16-bit instruction set so it allows 2 instructions per fetch. Suffix .s is apart of it. It's in the ARM Thumb 2 instruction set referred as well.

Context refers to tasks not interrupts

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

Tail chaining is used to improve nested interrupt support by preventing ISRs interrupting each other (late arrival, early, same time)

The link register is 14 & PC is 15

d) 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 ✓?

Initially, you move address from PC to LR, when you BL you move from LR to PC. LR has next instruction where PC is this instruction

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

NVIC supports vectored interrupt controllers, nested interrupt support, dynamic priority, and tail-chaining

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

ARM's A series is for high performance, R for RT applications, and M for low power consumption non RT.

7