

Xi'an Jiaotong-Liverpool University
西交利物浦大学

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CPT210	Jianjun Chen	Computer Science and Software Engineering	81889137

2nd SEMESTER 2020/21 RESIT EXAMINATION

Undergraduate – Year 3

Microprocessor Systems

TIME ALLOWED: 2 Hours

INSTRUCTIONS TO CANDIDATES

- 1、 This is a closed-book examination, which is to be written without books or notes.
- 2、 Total marks available are 100.
- 3、 Answer all questions.
- 4、 Answer should be written in the answer booklet(s) provided.
- 5、 Only English solutions are accepted.
- 6、 The university approved calculator - Casio FS82ES/83ES can be used.
- 7、 All materials must be returned to the exam supervisor upon completion of the exam. Failure to do so will be deemed academic misconduct and will be dealt with accordingly.

Question A (20 Marks)

1. Convert the following decimal numbers into their IEEE-754 single-precision (32-bit) representations. Give your answers in hexadecimal form. (4 marks each)
 - a. 7.75
 - b. 112.25
 - c. -1920.0
2. Convert the following IEEE 754 single-precision numbers in hexadecimal into their decimal values accurate to 5 significant figures. (4 marks each)
 - a. 0xc1e80000
 - b. 0x40fc0000

Question B (20 Marks)

1. Explain the situations where the N flag and the Z flag is set when the CMP instruction is used. Please provide examples of operands in your explanation and make sure the examples cover all possibilities of N and Z flags. E.g., when is the N flag set but the Z flag is not set. (16 marks)
2. Describe, using the LDR instruction as an example, the difference between the pre-indexed immediate offset and the post-index immediate offset. (4 marks)

Question C (30 Marks)

1. Write an ARM assembly language program to find the minimum of the absolute values of the numbers stored in “my_numbers”. Assume that the number of elements in “my_numbers” is always fixed to 10. (15 marks)

my_numbers DCD -5, 11, 21, -72, -2, 2, 9, -3, 91, -6

2. Write an ARM assembly language program that shifts the bits in the register R9 outward from the centre once. That is, the least significant 16 bits should be shifted to the right (logically, not arithmetically) and the most significant 16 bits should be shifted left. The result should be stored back to R9. (15 marks)

Example:

Original R9: 0x10000008

Shifted R9: 0x20000004

Question D (30 marks)

Given a piece of C code shown like below:

```
int x = 0;
while (x < 7)
{
    x++;
    printf("x is %d\n", x);
}
```

The equivalent java version is:

```
int x = 0;
while (x < 7)
{
    x++;
    System.out.printf("x is %d\n", x);
}
```

1.2.3.4.5.6.7

If you compile the code above without any optimisations, the following assembly code will be generated (Note that L4 contains the string literal "x is %d\n"):

```

        mov     r3, #0
        str     r3, [r11, #-8]
        b       L2

L3:
        ldr     r3, [r11, #-8]
        add     r3, r3, #1
        str     r3, [r11, #-8]
        ldr     r1, [r11, #-8]
        ldr     r0, L4
        bl      printf

L2:
        ldr     r3, [r11, #-8]
        cmp     r3, #6
        ble     L3

```

*int x=0 (x is r3)
; store x into stack
; call L2*

*; load x
; x+=1
; store x
; load x
; load the string
; print the string out*

*; load x
; compare x with 6
; as long as x ≤ 6, execute while loop*

But if you turn on compiler optimisation, the following assembly code will be generated instead (Note that L6 contains the string literal "x is %d\n"):

```

        mov     r4, #0
        ldr     r5, L6

L2:
        add     r4, r4, #1
        mov     r1, r4
        mov     r0, r5
        bl      printf
        cmp     r4, #7
        bne     L2

```

*; x is r4
; load the string on r5*

*; x++
; prepare to print x
; prepare to print string
; print them out
; compare r4 (x) with 7
; as long as not equal continue*

Your task:

1. Describe the processes of the two pieces of assembly code. In the description, map the elements of the C/Java code to the assembly code. For example, how is the variable x represented in these two pieces of assembly code? Where is the corresponding code for “x++” etc. (16 marks)
2. Work out the status of NZCV flags of the first piece of assembly code after execution. (4 marks)
3. Describe the optimisation applied in this example. Why does the second piece of assembly code run faster? Do not limit your discussion at the assembly level, you should also apply other knowledge of microprocessor systems you learned. (10 marks)

END OF FINAL EXAM