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 FINAL 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. This will count for 70% in the final assessment.**

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. 6.1328125
   b. 24.375
   c. 1703936.0

2. Convert the following IEEE 754 single-precision numbers in hexadecimal into their decimal values accurate to 5 significant figures. (4 marks each)
   a. 0xC0350000
   b. 0x40310000

Question B (20 Marks)

1. Explain the difference between the C flag and the V flag in terms of addition (ADD) overflows. Describe the situations where the carry flag is set or the overflow flag is set or both flags are set. Please provide examples of operands in your explanation and make sure the examples cover all possibilities of C flags and V flags. E.g., when is the C flag set but the V flag is not set. (16 marks)
2. Describe, using the STR 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 maximum value 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          -1, 22, 31, -22, -12, 21, 7, 8, 91, -6
   ```

2. Write an ARM assembly language program that shifts the bits in register R9 to the centre once. That is, the least significant 16 bits should be shifted to the left and the most significant 16 bits should be shifted right (logically, not arithmetically). Neither part should overwrite the bits of another (see the examples below). The result should be stored in R9. (15 marks)

   Example 1:
   Original R9: 0x20000004
   Shifted R9: 0x10000008

   Example 2:
   Original R9: 0x00099000
   Shifted R9: 0x00042000

Question D (30 marks)

Given a piece of C code shown like below:

```
int x = 0;

for (int i = 1; i < 8; i++)

{

     x++;

     printf("x is %d\n", x);

}
```

The equivalent java version is:

```
int x = 0;

for (int i = 1; i < 8; i++)

{

     x++;

     System.out.printf("x is %d\n", x);

}
```

If you compile the code 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]
      mov   r3, #1
      str   r3, [r11, #-12]
      b     L2
L3
      ldr   r3, [r11, #-8]
      add   r3, r3, #1
      str   r3, [r11, #-8]
      ldr   r1, [r11, #-8]
      ldr   r0, L4
      bl    printf
      ldr   r3, [r11, #-12]
      add   r3, r3, #1
      str   r3, [r11, #-12]
L2
      ldr   r3, [r11, #-12]
      cmp   r3, #7
      ble   L3
```

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
```

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 second 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