Introduction to Computing Systems (2022 Fall)

Mid-term Exam

Saturday, November 19th, 2022

Student_ID: _____________ Name: ____________

Organization	*Score*
Signature (1 point)	
A.Short Answers (24 points)	
B.Digital Logic Structures(29 points)	
C.Von Neumann Model (18 points)	
D.LC-3 Programming (28 points)	
Total (100 points)	

I will not cheat on this exam.

________________signature (1 point)

PART A: Short answers (24 points)

A1 (3 points): Add the two hexadecimal 2's complement integers below:

$$\begin{array}{r} \text{x7D85} \\ \underline{+\ \ \text{xF0A0}} \end{array}$$

A2 (5 points): Add two n bit 2's complement integers with the same sign(both positive and both negative), and we can get an n+1 bit result. Prove that the operation overflows if and only if the highest two digits of the result are different.

A3 (4 points): Write 32bit IEEE floating point representation of $(4.21875)_D$.Can this value be represented accurately?

A4 There are no floating-point instructions in current LC-3 ISA, so it's complicated to handle floating point calculations on LC-3. Now we propose a set of floating point instruction extension, so called LC-3f, to support floating point calculation on LC-3 computer. One problem is that there're only 16 bits in each general purpose register (GPR, R0 to R7) in LC-3 computer, so it's difficult for us to calculate floating number in 32 bits. One possible plan is called Float-16, which uses 16 bits instead of 32bits to represent floating numbers. It is similiar to the 32-bit floating point data type, while it has only 5 exponent bits.

1. (1 point) What is the biggest number that float-16 can represent? Give your answer in decimal.

2. (5 points) Suppose that there's one float-16 number stored in R0 (R0 cannot be NaN or inf), now your job is to write code in **LC-3 0/1 machine code** in no more than 20 instructions to determine its sign (positive, zero or negative).Your answer should be stored in R1. When the input is zero, positive or negative, the result is 0, 1, 2 respectively. Your program should start at memory location x3000, and all GPRs except R0 are set to zeros at begining and you can use all of them.
   Hint: you can add some comments and space to improve the readability of your program, and the program should end up with HALT.

3. (2 points) The binary value below are 32bit IEEE floating point numbers. Convert them to float-16 numbers. Give your answer in binary.
   (1) 0 10000111 10101111000000000000000
   (2) 0 11000001 10001000100000000000000

A5 (4 points): The current LC-3 memory contains a total of 1 Megabit (2^{20} bits). If we were to make the memory byte-addressable (a byte is 8 bits) without changing its size, and without changing the size of the instructions (16 bits), how many bits would be necessary in the following registers?

MAR $\boxed{}$ IR $\boxed{}$

PART B: Digital Logic Structures (29 points)

B1(12 points)

1. (9 points) Please implement a transistor-level circuit for the following truth table. A and B are input signals, Out_1 and Out_2 are output signals.

A	B	Out_1	Out_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

   Hint: you can first implement some parts of the circuit as sub-modules, and then build the entire circuit based on these sub-modules.

2. (3 points) What does this circuit do?

B2 (11 points)

A regular expression (shortened as regex or regexp) is a sequence of characters that specifies a search pattern in text. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

Here follows some basic concepts in regular expression.

<table>
<tr><th>metacharacter</th><th>description</th></tr>
<tr><td>( )</td><td>Defines a marked subexpression like in C language. A marked subexpression is also called a block or capturing group.</td></tr>
<tr><td>[ ]</td><td>A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c".</td></tr>
<tr><td>|</td><td>Boolean "or".</td></tr>
<tr><td>?</td><td>Matches the preceding element zero or one time.</td></tr>
<tr><td>+</td><td>Matches the preceding element one or more times.</td></tr>
<tr><td>*</td><td>Matches the preceding element zero or more times.</td></tr>
</table>

Tip: the subexpression and bracket expression would be regarded as an element when you use ``?``,``+``and ``*``.

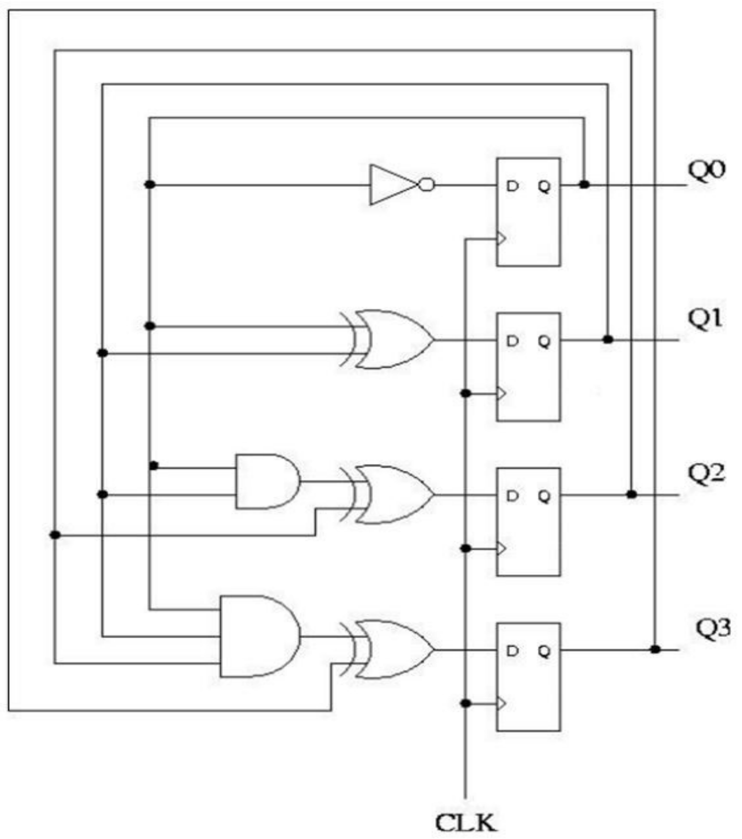
Examples:

a|b* denotes {"", "a", "b", "bb", "bbb", ...}
(a|b)* denotes the set of all strings with no symbols other than "a" and "b", including the empty string: {"", "a", "b", "aa", "ab", "ba", "bb", "aaa", ...}
ab+(c|d) denotes the set of strings starting with "a", then one or more "b"s and finally optionally a "c" or "d": {"ab", "abc", "abd", "abbc", "abbd", ...}
[0-9]+(a|c)?[0-9]* denotes the set of strings starting with at least one number from 0 to 9, then zero or one character in "a" or "c", and finally optionally numbers: {"3c", "202", "3c202", ...}
(0|(1(01*0)*1))* denotes the set of binary numbers that are multiples of 3: { "", "0", "00", "11", "000", "011", "110", "0000", "0011", "0110", "1001", "1100", "1111", "00000", ... }

Now please draw a finate state machine for regular expression **([1-5]+(a|b|c)?[de])*0[0-9]***, suppose that the input string only contains **abcde** and **0-9**.

Hint: to simplify your state diagram，you can use regular expression element to represent external inputs and outputs of your state machine, such as use "**[1-5]**" instead of "**1,2,3,4,5**". You can also use "**others**" to represent the inputs other than the inputs that you have clearly defined.

B3 (6 points) Describe what the following sequential circuit does in one sentence. Assume that the output $Q_3Q_2Q_1Q_0$ is initially 0000.



PART C: The Von Neumann Model (18 points)

For the following question, refer to the LC-3 Von Neumann Model below. The LC-3 is about to start the instruction cycle for the instruction at x3000.

The current memory state:

address	instructions / data	note
x3000	0010 000 011111111	R0 ←M[x3100]
x3001	1110 101 011111111	R5 ← x3101
x3002	0001 010 010 1 00010	R2 ← R2 + 2
x3003	0001 011 011 1 00001	R3 ← R3 + 1
x3004	0101 100 000 0 00 010	R4 ← R0 & R2
x3005	0000 010 000000001	…
x3006	0001 001 001 0 00 011	R1 ← R1 + R3
x3007	0001 011 011 0 00 011	R3 ← R3 + R3
x3008	0001 010 010 0 00 010	R2 ← R2 + R2
x3009	0000 101 111111010	…
x300a	0111 001 101 000000	Store R1 in x3101
x300b	1111 0000 00100101	HALT
…	…	…
x3100	0101 0101 0101 0101	**data**

The current register state:

R0	R1	R2	R3	R4	R5	R6	R7
0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000

C1 (12 points) Suppose after the execution of some instructions, the LC-3 is about to start the instruction cycle for the instruction at x300a, complete the table below.

IR	PC	MAR	MDR

C2 (4 points) Now the LC-3 finishes the execution for the instruction at x300a, complete the table below.

MAR	MDR

C3 (2 points) Explain what the code between x3000 and x300a does in one sentence.

PART D: LC-3 Programming (28 points)

D1 (16 points): The PC is initially loaded with x3000 and the instruction at address x3000 is executed. Actually, five more instructions are also executed. The table below contains the contents of some key registers at the end of execution for each of the six instructions.

Your job: complete the table

	PC	**MAR**	**MDR**	**IR**	**R0**	**R1**	**R2**	**R3**
Before execution	x3000	/	/	/	x0000	x0000	x0002	x0003
After the 1st execution			x0007	x2022				
After the 2nd execution				x52BF				
After the 3rd execution			x1_E_				x0003	
After the 4th execution			x1__1					x0005
After the 5th execution				x1_3_	x0006			
After the 6th execution				x07FB				

D2 (4 points): Assume the instruction at x3006 is xF025. What is the final value of R3? Denote the value at x3023 as N(N⩾0) and the final value of R3 as F(N). what is the expression of F(N)? You are allowed to define F(N) recursively(递归地).

D3 (8 points): You may notice that the initial values of R2 and R3 are 2 and 3. Now we **re-initialize** R2 and R3 to 0 (initial values of R0 and R1 are still 0). And we expect R3 = 2 when N = 0, R3 = 3 when N = 1, and R3 = F(N - 2) when N⩾2.

Your job is to Fill in the blank of the table below. **Remember that N is still stored at x3023.**

Note that the instructions between x3008 and x300D are identical to the instructions between x3001 and x3006.

PC	IR
x3000	x20__
x3001	x16__
x3002	x1__F
x3003	x0___
x3004	
x3005	
x3006	x0__
x3007	
x3008	x52BF
…	…
x300C	x07FB
x300D	xF025