UNIVERSITY OF COLOMBO, SRI LANKA

UCSC

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

BACHELOR OF INFORMATION SYSTEMS

Academic Year 2014/2015 – First Year Examination – Semester I – 2015

IS1102 – Computer Systems

TWO (2) HOURS

To be completed by the candidate

Examination Index No:

Important Instructions to candidates:

1. The medium of instruction and questions is **English**.
2. If a page or a part of this question paper is not printed, please inform the supervisor immediately.
3. Note that questions appear on both sides of the paper. If a page is not printed, please inform the supervisor immediately.
4. Write your index number in each and every page of the question paper.
5. This paper has **4** questions and **14** pages.
6. Answer **ALL** questions. All questions carry equal marks (**25** marks).
7. Any electronic device capable of storing and retrieving text including electronic dictionaries and mobile phones are not allowed.
8. Calculators are not allowed.

For Examiner's use only	
Question No	Marks
1	
2	
3	
4	
Total	

1. **(a)** What is the decimal number equivalent of the 16-bit floating point number **0 10101 0101010101**? Assume that 16-bit floating point representation is with a sign bit, 5-bit exponent and 10-bit mantissa.

[6 Marks]

In Excess k, $k = 2^{N-1} - 1 = 2^{4} - 1 = 15$

Therefore Actual Exponent $= 21 - 15 = 6$

$\therefore$ Answer $= +1 \cdot 0101010101 \times 2^{+6}$

$= +1010101 \cdot 0101$

$= +85.3125$

(b) What is the loss of accuracy (round-off-error) when converting the decimal value **+511.875** to 16-bit floating point representation with a sign bit, 5-bit exponent and a 10-bit mantissa?

[6 Marks]

$+511.875 = +111111111 \cdot 111$

$= +1 \cdot 11111111111 \times 2^{8}$

In Excess k, Exponent $= 15 + 8 = 23$

$\therefore$ Therefore Representation,

0	1 0 1 1 1	1 1 1 1 1 1 1 1 1 1

1

$\therefore$ Round-off-error $= 0.125$

(c) What is the IEEE standard 32-bit floating point representation of the decimal number **+42.625**?

[6 Marks]

$+42,625 = +101010 \cdot 101$

$= +1 \cdot 0101010101 \times 2^{+5}$

In Excess k, k = 127

∴ In Excess k, Exponent = 127+5 = 132

∴ Representation.

0	10000100	010101010 . . . .

//

(d) What is the equivalent decimal number to the IEEE standard 32-bit floating point representation of **0 10000100 01010101000000000000000**

[7 Marks]

In Excess k, k = 127

∴ Actual Exponent = 132 − 127 = 5

∴ Answer $= +1 \cdot 01010101 \times 2^{+5}$

$= +101010 \cdot 101$

$= +42.625$ //

2. Consider the following **truth table** for a digital circuit.

A	B	C	D	F(A,B,C,D)
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**1**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**0**

(b) Write the **Boolean expression** (Sum of Products) that represents the logic function performed by the above circuit

[3 Marks]

$F(A, B, C, D) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D}$
$+ \overline{A}\,B\,C\,\overline{D} + A\,\overline{B}\,\overline{C}\,\overline{D} + A\,\overline{B}\,C\,\overline{D}$
$+ A\,B\,\overline{C}\,\overline{D}$

(c) Write the **Boolean expression** (Product of Sums) that represents the logic function performed by the above circuit

[3 Marks]

$F(A, B, C, D) = (A+B+C+\overline{D}) \cdot (A+B+\overline{C}+\overline{D}) \cdot (A+\overline{B}+C+\overline{D}) \cdot$
$(A+\overline{B}+\overline{C}+\overline{D}) \cdot (\overline{A}+B+C+\overline{D}) \cdot (\overline{A}+B+\overline{C}+\overline{D}) \cdot$
$(\overline{A}+\overline{B}+C+\overline{D}) \cdot (\overline{A}+\overline{B}+\overline{C}+D) \cdot (\overline{A}+\overline{B}+\overline{C}+\overline{D})$

(c) Get the most simplified **Sum of Products** (SOP) and **Product of Sums** (POS) expressions that represents the logic function of the above digital system.

[7 Marks]

AB	00	01	11	10
00	1	1	1	1
01				
11				
10	1	1		1

Sum of products $= \bar{A}\bar{D} + \bar{B}\bar{D} + \bar{C}\bar{D}$

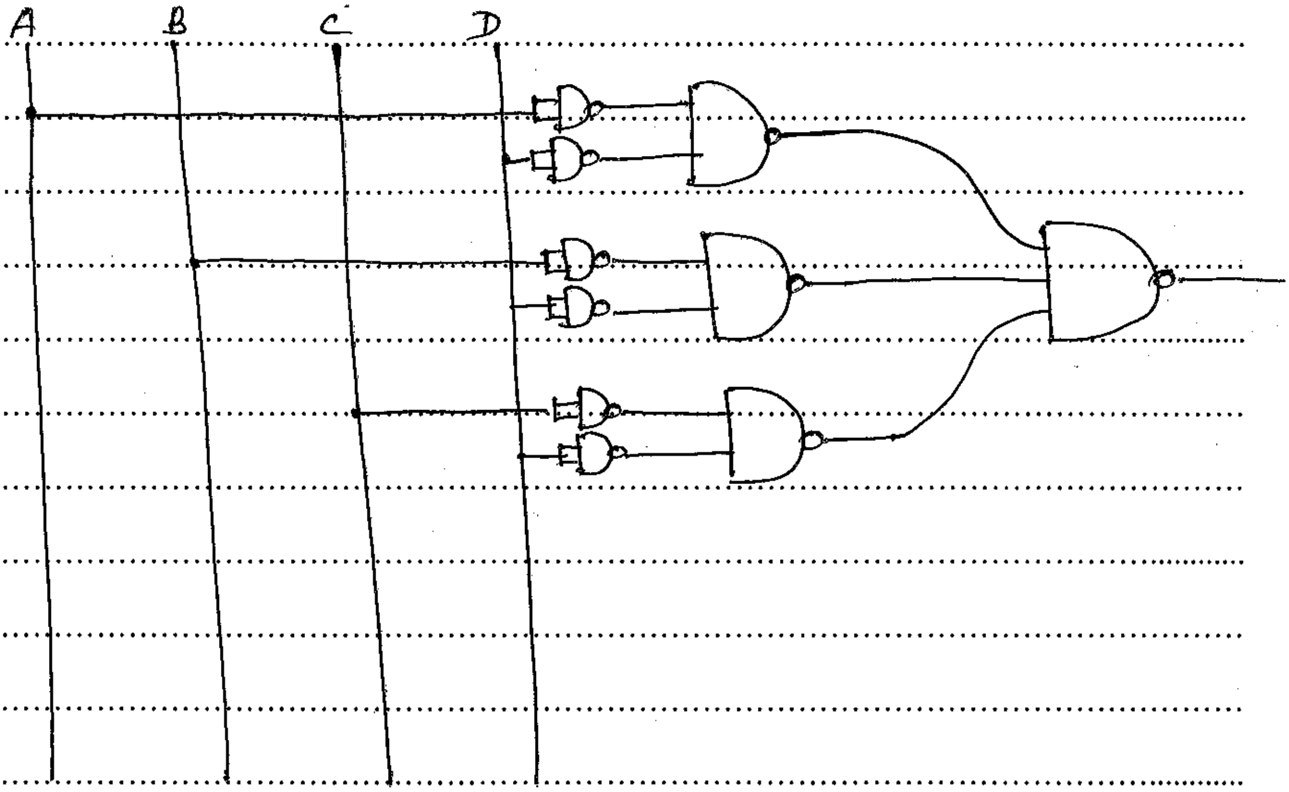
Product of Sums $= (\bar{D}) \cdot (\bar{A} + \bar{B} + \bar{C})$

(d) Design the logic circuit for the above **SOP** expression in **(c)** using only **NAND** gates. Clearly indicate the method and steps.

[6 Marks]

$$SOP = \overline{A}\,\overline{D} + \overline{B}\,\overline{D} + \overline{C}\,\overline{D}$$

$$= \overline{\left(\overline{\overline{A}\cdot\overline{D}}\right)\cdot\left(\overline{\overline{B}\cdot\overline{D}}\right)\cdot\left(\overline{\overline{C}\cdot\overline{D}}\right)}$$

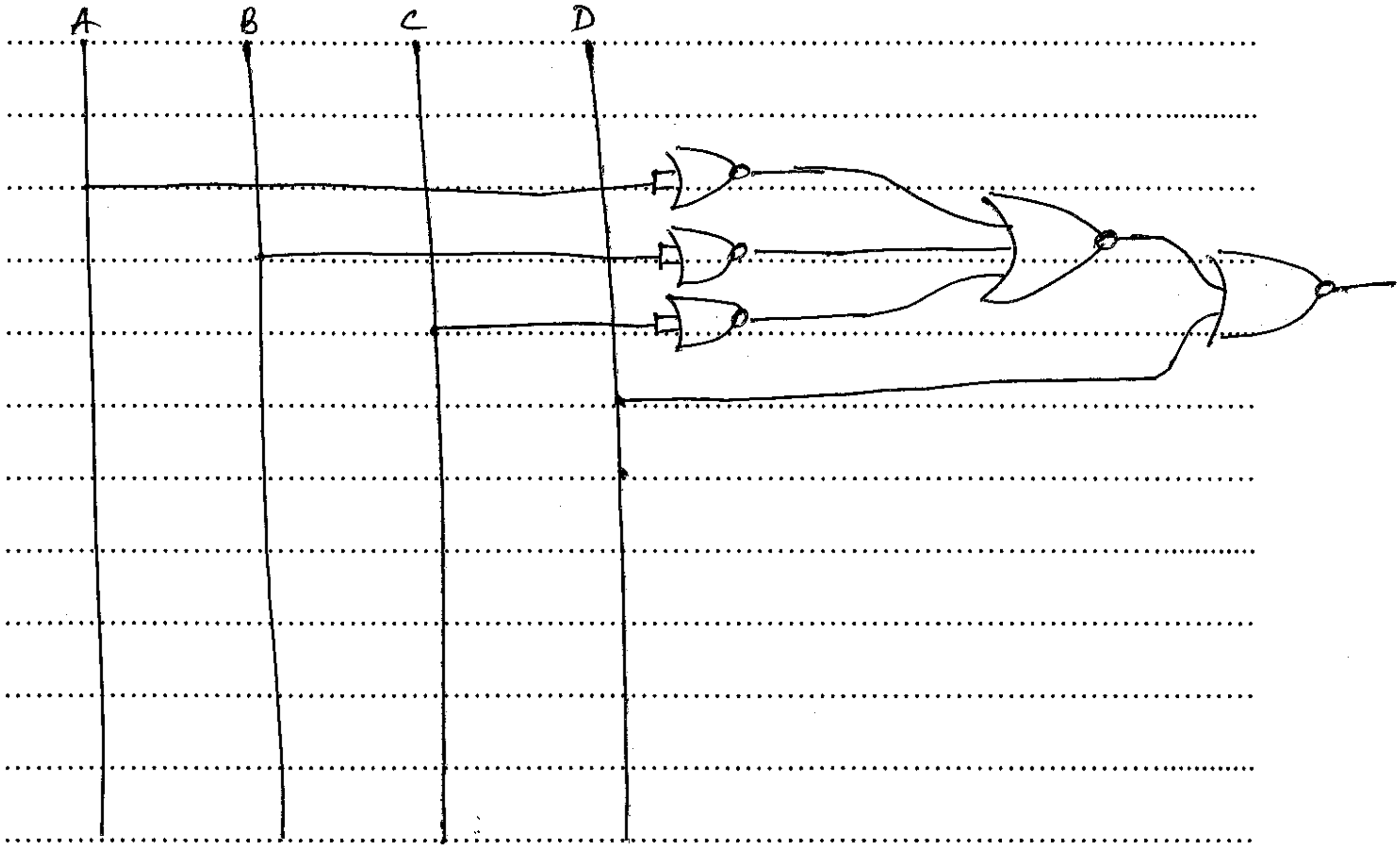


(e) Design the logic circuit for the above **POS** expression in **(c)** using only **NOR** gates. Clearly indicate the method and steps.

[6 Marks]

$$POS = (\bar{D}) \cdot (\bar{A} + \bar{B} + \bar{C})$$

$$= \overline{(D) + (\overline{\bar{A} + \bar{B} + \bar{C}})}$$



3. (a) Consider a machine with instruction format of the form **opcode R M** where **R** is a register address and **M** is a memory address. Instructions are 16 bits long and one of the instruction formats provides 4 bits for the op-code, 4 bits for the register and other 8 bits for the memory address of the operand. Assume that the word size of this machine is 8 bits (byte addressable).

Some of the op-codes of the above (a) processor is given below:

0001 – L R, A	**LOAD** the register **R** with the content of memory cell **A**
0010 – LI R, I	**LOAD** the register **R** with the value **I**
0011 – ST R, A	**STORE** the content of the register **R** to the memory cell whose address is **A**
0101 – ADD R0, R1, R2	**ADD** the numbers in registers **R1** and **R2** and place the result in register **R0**
1001 - XOR R0, R1, R2	**XOR** the bit patterns in **R1** and **R2** and place the result in **R0**
1000 - AND R0, R1, R2	**AND** the bit patterns in **R1** and **R2** and place the result in **R0**
1110 – JMP R, A	**JUMP t**o the instruction located in the memory cell **A** if the bit pattern in **R** is equal to the one in **R0**
1111 - HALT	**HALT** the execution

Write down the machine code instructions sequence to execute the following program statements.

```
void main( )
{
        int  A = 33 ;
        int  B = 11 ;
        A = A – B ;

        if ( A >= 0 )
                do
                {
                        A = A – B ;
                } while ( A < 0 ) ;
        else
                do
                {
                        A = A + B ;
                } while ( A >= 0 ) ;
}
```

Assume that **A** and **B** are variables refer the memory addresses **80**, **81** and the initial program counter (PC) as hexadecimal **30**.

[25 Marks]

```
30 Load_i   R1      21                 } A = 33
32 Store    R1      80(A)              }

34 Load_i   R2      0B                 } B = 11
36 Store    R2      81 (B)             }

38 Load_i   R3      FF                 }
3A XOR      R4, R2, R3                 } A = A - B
3C Load_i   R5      01                 }
3E ADD      R6, R4, R5 (- B)           }
40 ADD      R1, R1, R6                 }
42 Store    R1      80(A)              }

44 Load     R0      80                         }
46 AND      R7, R0, R1                         } A < 0
→48 JMP     R7      54     if (R0 == R7)       }

→4A ADD     R1, R1, R6  } A = A - B     } do {
4C Store    R1      80(A)               }   A = A - B
4E AND      R8, R0, R1                  } } while (A < 0)
→50 JMP     R8      4A     if (R0 == R7)   }

52 JMP      R0      60
```

```
54  ADD     R1 , R1 , R2   } A = A+B       }
56  Store   R1   80(A)     }               } do {
58  Load_i  R8   80                        }   A = A+B
5A  AND     R9 , R1 , R8                   } } while (A>=0)
5C  Load_i  R0   00                        }
5E  JMP     R9   [54]      } if (R0==R9)   }
[60] HALT
```

4. (a) A computer has 16 pages of virtual address space, but only 4 page frames. Initially, memory is empty. A program references the virtual pages in the following order: **0, 1, 2, 1, 2, 4, 3, 6, 0, 5, 7, 8, 7, 8, 2, 1, 3, 7, 3, and 4.**

For each memory reference, write the virtual page stored in each page frame under the **Least Recently Used (LRU)** and **First-In-First-Out (FIFO)** page replacement policies (clearly show the page frames in memory for each reference).

LRU

[10 Marks]

0	1	2	1	2
0	0	0	0	0
	1	1	1	1
		2	2	2

4	3	6	0	5
0	3	3	3	3
1	1	6	6	6
2	2	2	0	0
4	4	4	4	5

7	8	7	8	2
7	7	7	7	7
6	8	8	8	8
0	0	0	0	2
5	5	5	5	5

1	3	7	3	4
7	3	3	3	3
8	8	7	7	7
2	2	2	2	4
1	1	1	1	1

FIFO

0	1	2	1	2
0	0	0	0	0
	1	1	1	1
		2	2	2

4	3	6	0	5
0	3	3	3	3
1	1	6	6	6
2	2	2	0	0
4	4	4	4	5

7	8	7	8	2
7	7	7	7	7
6	8	8	8	8
0	0	0	0	2
5	5	5	5	5

1	3	7	3	4
7	3	3	3	3
8	8	7	7	7
2	2	2	2	4
1	1	1	1	1

(b) Suppose you have **4GB** of virtual memory, **32MB** of physical memory, and the page size of **16KB** (2^{14}) bytes.

(i). How many pages are there in virtual memory and in physical memory?

[5 Marks]

No. of Pages in Virtual Memory = 4GB / 16KB

= 2^{18} pages

No. of Page frames in Physical Memory = 32MB / 16KB

= 2^{11} page frames.

(ii). Assume that the relevant portion of your Page Table is as follows:

Page	Frame #	Valid Bit
0	2	1
1	3	0
4	8	0
8	0	1
11	5	1
…….	…….	…….
15	7	1
…….	…….	…….
17	4	1
…….	…….	…….
20	10	1
…….	…….	…….

Calculate the referenced address in main memory for the following virtual addresses: **0x0002D910** and **0x00053808**.

[10 Marks]

0x 0002D910

$$\underbrace{0000\ 0000\ 0000\ 0010\ 11}_{\text{page NO.}}\underbrace{01\ 1001\ 0001\ 0000}_{\text{offset}}$$

∴ page NO = 11 ⇒ page frame = 5

∴ Referend Address = 5 × 16KB + 6,416
= 5 × 16,384 + 6,416

= 88,336 //

0x 00053808

$$\underbrace{0000\ 0000\ 0000\ 0101\ 00}_{\text{page NO.}}\underbrace{11\ 1000\ 0000\ 1000}_{\text{offset}}$$

∴ page NO = 20 ⇒ page frame = 10

∴ Referened Address = 10 × 16,384 + 14,344

= 178,184 //

****