Name:

Note: (Late: -15% Penalty).

1) (15 pts) Write a brief essay paragraph explaining one breakthrough in the history of computing and the significance of the breakthrough to advancing Computer Science.

Acceptable answers include explanations of vacuum tubes, transistors, integrated circuits, VLSI, binary arithmetic, quantum computing, and parallel computing.

Ans. See backside for Work

2) (15 pts) Using the CRC polynomial 1101, compute the CRC code for the information word, 1 0110 1101

Ans:

See backside for Work

3) (10 pts) Convert the following decimal fractions to binary with a maximum of six places to the right of the binary point:

a. 26.78125

See backside for Work

b. 194.03125

See backside for Work

4) (15 pts) Define the following

a. Combination Logic:

Combination Logic takes inputs and runs them through logical gates.

b. Sequential Logic:

Sequential Logic takes inputs and loops some of them in a second or more times. So there is like a state which effects the output of the next round of inputs.

Name:

c. How are sequential circuits different than combinational circuits?

Sequential circuits depend on the last input. sort of like a state. It holds on to previous inputs and they affect curr Input.

5) (10 pts) Add the following 8-bit two's complement numbers (i.e. one sign bit and seven data bits) AND indicate "Overflow" if it occurs.

$-128 \leq r \leq 127$

a. 1111 0101
\+ 1101 0101
11001010

Answer: No overflow answer = 11001010

b. 0110 1011
\+ 0101 0101
11000000

0010101
16 + 4 + 1 = 21

= 64

32 + 8 + 2 + 1
0101011 = 43

Answer: There is overflow because we added two positive numbers and got a negative

6) (15 pts) Given the boolean equation: x'yz + x(yz)' + x'(y+z) + (xyz)'

a. Construct the truth table for the Boolean equation:

x	y	z	F(xyz)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

[redacted]

Name:

b. Construct the K-map for this circuit.

z \ xy	00	01	11	10
0	1	1	0	1
1	1	1	0	0

c. Write a reduced but equivalent Boolean equation. Use either the K-map or apply the Boolean postulates/theorem's to do your reduction.

'X + '(YZ)

7) (15 pts) Assume a 2^{20} byte memory:

a. What are the lowest and highest addresses if memory is byte-addressable?

In binary (20 bits) — in decimal

Lowest = 00000000000000000000 = 0000000

highest = 11111111111111111111 = 1048576

b. What are the lowest and highest addresses if memory is word-addressable, assuming a 16-bit word?

in binary (19 bits) — in decimal

Lowest = 0000000000000000000 = 000 000

highest = 1111111111111111111 = 524 288

c. What are the lowest and highest addresses if memory is word-addressable, assuming a 32-bit word?

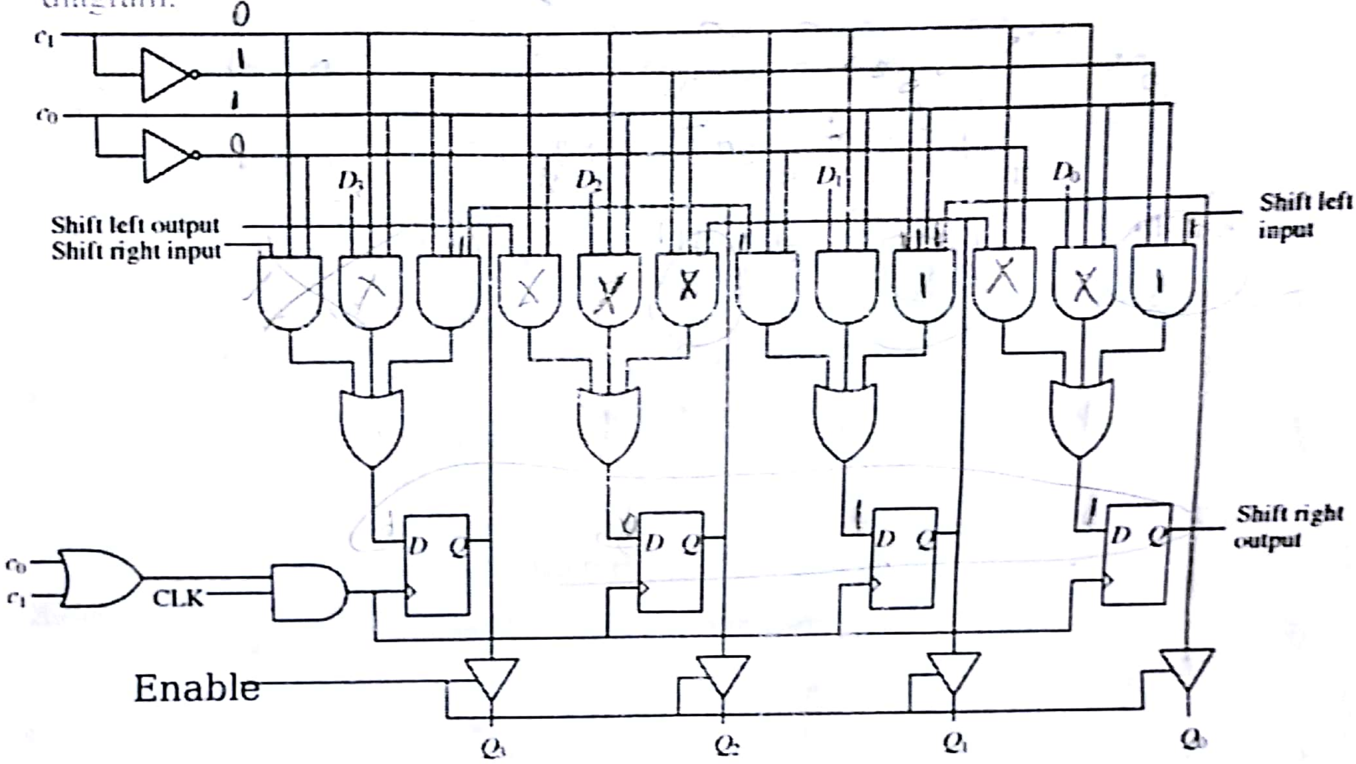
in binary (18 bits) — in decimal

Lowest = 000000000000000000 = 000000

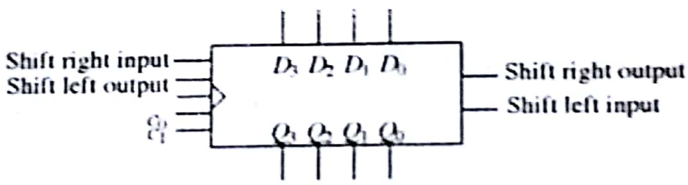
highest = 111111111111111111 = 262 144

Name:

8) (5 pts) Given the left-right shift register shown below answer the following questions. Place an 'x' for don't care conditions on the non-relevant outputs.
Note: Each Control input 'c0 and c1' as listed in the function table is shown twice in the diagram.



Control c_1 c_0	Function
0 0	No change
0 1	Shift left
1 0	Shift right
1 1	Parallel load



a. Fill in the state output of Q_0-to-Q_3 on clock Clk_4 given the input 'Enable=1' AND the following Input serial load of the shift-register in clocks Clk_0-to-Clk_3

Clk_0-to-Clk_3

	Clk_0	Clk_1	Clk_2	Clk_3	C_1	C_0
Shift Right Input	0	1	0	0	0	1
Shift Left Input	1	0	1	1		

Clk_4

Q_3	Q_2	Q_1	Q_0	C_1	C_0	Enable
1	0	1	1	0	0	1

ICS 232 Computer Organization & Architecture
Homework #3 Chapters 1- 5 150 points
Due Date: 03/17/2019

Name:

9) (15 pts) Using the MarieSim emulator - Write working assembly code that executes the logical programming construct:
Turing machines and Finite State Machines will continue until told to 'halt'. Please halt your code upon completion of the assigned task. Include comments to your code!

```
Initialize A=5; B=9;
UNTIL (A >=B)        ! Until A is greater than or equal to B
DO
      C = A + B      ! Add A to B
      STORE C        ! store in C
      A=A-1          ! Subtract 1 from A
      B=B-2          ! Subtract 2 from B
DONE
```

Answer: See Backside For Work

10) (15 pts) Using the MarieSim emulator - Write a working assembly code that executes the logical programming construct:
Turing machines and Finite State Machines will continue until told to 'halt'. Please halt your code upon completion of the assigned task. Include comments to your code!

```
IF (A = B)
THEN
      C = A + B
ELSE
      C = A – B
```

See Backside For Work

Answer:

11) (15 pts) Using the MarieSim emulator - Write working assembly code that executes the logical programming construct:
Include comments to your code!

Name:

Write a working assembly code that uses **stack parameter passing linkage** to call a subroutine labeled "**sub_1**" to subtract two numbers and return the results to the main routine. t = (r – s)

Your main routine must:

a. load your two parameters (r,s) from memory and push them onto your stack.
b. call sub_1
c. pop the result from your stack and store it into memory label 't: '.

Your sub-routine must:

a. pop your two parameters from your stack
b. subtract the parameters as shown in the problems formula
c. push the result onto the stack
d. return to your 'main' routine

Please initialize 't: 0' to start your code. t = (r – s)

Use care not to let your memory stack step on your code.

Answer:

12) (15 pts) Using the MarieSim emulator - Write working assembly code that executes the array initialization C code shown below:

Include comments to your code!

```
#include <stdio.h>

int main () {

  int n[ 10 ]; /* n is an array of 10 integers */
  int i;

  /* initialize elements of array n to 0 */
  for ( i = 0; i < 10; i++ ) {
    n[ i ] = i + 100; /* set element at location i to i + 100 */
  }

 return 0;
}
```

See Backside For Work

#1.) Time line of Mechnical Relays, Vacuum Tubes and Transistors

Mechinical relays were used to transmit data in pre 1940's era. They were implemented by sending electric current through a coil which in return closed a circuit wire switch via attraction by a magnetic field. Open circuit meant 1 and closed meant 0. The gate was an actual tiny metal rod with mass so that made it difficult for it to swing back and fourth at high rates. The engineers needed it to swing fast inorder for it to send bits at faster rates. Mechinical relays had a frequency of about 50Hz which was not fast enough.

Vacuum Tubes were a huge improvement over mechnical relays because they didn't have any swinging parts and were much faster. Vacuum tubes could reach +5000Hz frequencies. They were used in the first general purpose computer the ENIAC in 1945 designed by John Mauchly and colleague J. Presper Eckert. There was also Konrad Zuse an engineer with the same idea but he couldn't use Vacuum tubes because his funding wasn't adequate. Downside to Vacuum tubes was they were too expensive and hard to assemble in comparison to their predecesor the Transistor.

Transistors were much smaller (sometimes a million times smaller) and much more durable. They are solid components which means no complex inner assembly, could be smaller and also much faster than vacuum tubes. They are good enough to be still in use today and their small size is what made it possible for todays computers to be so compact and small.

#2.)

```
1101 | 101101101000
       1101
       01100
        1101
        0001110
           1101
           001110
             1101
             001100
                1101
                0001
```

CRC code = 001

#3.) a.) 26.78125

26 . 78125

26/2	=	13	r	0
13/2	=	6	r	1
6/2	=	3	r	0
3/2	=	1	r	1
1/2	=	0	r	1
0				

0.78125 × 2	=	1.5625
0.5625 × 2	=	1.1250
0.1250 × 2	=	0.2500
0.2500 × 2	=	0.5000
0.5000 × 2	=	1.0000
0.0000 × 2	=	0.0000

11010.110010

b.)

194 . 03125

194/2	=	97	r	0
97/2	=	48	r	1
48/2	=	24	r	0
24/2	=	12	r	0
12/2	=	6	r	0
6/2	=	3	r	0
3/2	=	1	r	1
1/2	=	0	r	1
0				

0.03125 × 2	=	0.0625
0.0625 × 2	=	0.1250
0.1250 × 2	=	0.2500
0.2500 × 2	=	0.5000
0.5000 × 2	=	1.0000
0.0000 × 2	=	0.0000

11000010, 000010

#9.)

```
        ORG     100 //

loop,   LOAD    A        //beginning of loop and load A in the accumulator
        ADD     B        //Add A + B (A is inside accumulator)
        STORE   C        //Store the sum in C
        LOAD    A        //Load A back in to the accumulator
        SUBT    one      //A - 1 is now in the accumulator
        STORE   A        //A = accumulator value
        LOAD    B        //Load B back in to the accumulator
        SUBT    two      //B - 2 is now in the accumulator
        STORE   B        //B = accumulator vaue
        LOAD    A        //LOAD A back into the accumulator
        SUBT    B        //Subtract acc from B and put result in accu
        SKIPCOND     800 //if A - B > 0 that means A >= B so stop(skip) loop
        JUMP         loop  //restart from  the beginning of the loop
        HALT             //Stop the Program

        A,        DEC 5   // A = 5
        B,        DEC 9   // B = 9
        C,        DEC 0   // C = 0
        one,      DEC 1   // one = 1
        two,      DEC 2   // two = 2
```

#10.)

```
ORG 100             //start from origin

LOAD A              //load value A into accumulator
SUBT B              //subtract B from the accumulator
SKIPCOND 400        //If A - B = 0 skip the else jump
JUMP  else

//THEN CLAUSE STARTS HERE
LOAD   A            //A goes into the accumulator
ADD    B            //B is added to A
STORE C             //Store the result in B
HALT                //Stop the program
//THEN CLAUSE ENDS HERE

else,LOAD   A       //load A into the accumulator
     SUBT   B       //subtract B from the accumulator
     STORE C        //Store the result in c
     HALT           //Stop the Program

A,    DEC 5// A = 5
B,    DEC 9// B = 5
C,    DEC 0// C = 0
```

#12)

```
            ORG     100

forloop,    LOAD        count       //count starts at 0
            ADD         oneHun      //Add 100 to the count
            STOREI      index       //STORE it in Array[index]
            LOAD        index       //LOAD index
            ADD         one         //index + 1
            STORE       index       //index = index + 1
            CLEAR                   //clear the accumulator
            LOAD        size        //load the size(10) in accumulator
            SUBT        one         //size - 1
            STORE       size        //size = size -1
            SKIPCOND    400         //if size – 1 = 0 break for loop
            JUMP        forloop
            HALT

 oneHun,     DEC        100         //oneHun = 100
 one,        DEC        1           //one = 1
 size,       DEC        10          //initial size = 10
 index,      HEX        112         //start of array @index=112 in hex
```