Question 1 (12 Points)

a. Write down the binary equivalent of the decimal number 12.

b. How many outputs does an N-input decoder have?

c. How many select lines does an 8 x 1 Mux have?

d. If register R2 contains # 4, what decimal value will it have after the following instruction is executed?

\[ \text{ADD R2, R2, R2} \]

e. A memory has 8 address lines and each location stores 4 bits of information. What are its address space and addressability?

f. Construct a 4-input AND gate with inputs A, B, C and D, using only 2-input AND gates. Assume you only have three 2-input AND gates.

Question 2 (8 Points)

a. Write down the result of bit-wise Exclusive-OR operation on the following two numbers.

1010111 and 0101111

b. Write down your answer to part (a) in hexadecinal

c. Convert decimal number 13.75 to binary.

d. Represent the decimal number 13.75 in the IEEE floating point format.
Question 3 (15 points)

a. Consider 2-bit numbers A \([A1 : A0]\) and B \([B1 : B0]\). Construct a truth table for the multiplication function of these numbers. (Hint: You need to first figure out how many output bits are required to represent the result of a two bit multiplier.)

b. Write the Boolean expression for each output bit of the truth table in part (a)

\[
\text{OR}
\]

b. Implement the truth table in part (a) using logic gates.

Question 4 (10 points)

Assume register R2 contains the value xAAAA before the execution of the following seven instructions.

<table>
<thead>
<tr>
<th>Address</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>30F6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30F7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30F8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30F9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30FA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>30FB</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30FC</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

a. How many different values are contained in R2 during the execution of the seven instructions.

b. What are they?

Question 5 (5 points)

Draw transistor level diagrams for a 3-input AND gate and a 3-input OR gate.
Question 6 (10 Points)

The following program multiplies two numbers A and B, which are single digit positive decimal numbers provided during runtime. The main program prompts the user to enter the numbers A and B, which are to be stored at memory locations x3100 and x3101 respectively. The program uses the subroutine MUL to multiply these two numbers. The subroutine assumes that the numbers A and B are in R1 and R2 respectively and it stores the result of the multiplication in R3. The final result needs to be stored at memory location x3102.

```
01   .ORIG    x3000
02   LD    R0, PROMPT1
03   TRAP    x22
04   TRAP     x20
05   TRAP     x21
06   ST     R0, FIRST_NO
07   ADD    R1, R0, #0
08   LD    R0, PROMPT2
09   TRAP    x22
10   TRAP     x20
11   TRAP     x21
12   ST     R0, SECOND_NO
13   ADD    R2, R0, #0
14   JMP    MUL
15   ST    R3, OUTPUT
17  PROMPT1 .STRINGZ "Input A (1-9)"
18  PROMPT2 .STRINGZ "Input B (1-9)"
19  FIRST_NO .FILL    x3100
20  SECOND_NO .FILL    x3100
21  OUTPUT .FILL    x3100
22  MUL  ------------------------------- ; this subroutine multiplies
-  ------------------------------- ; the contents of R1 and R1 and
-  ------------------------------- ; stores the result in R3
-    RET
-  DIV  ------------------------------- ; this subroutine divides the
-  ------------------------------- ; contents of R1 by those of R2
-  ------------------------------- ; and stores the quotient in R3
-  ------------------------------- ; and remainder in r4
-    RET
```

a. The above program has a few errors. Identify these errors and correct them. Write the corrected instruction and corresponding line numbers in your answers.
b. Explain the purpose of line 05 to 11. If the result in R3 was to be displayed how would you incorporate that in the program after line 15? Write down your instructions. (Hint: You may have to output more than one character. Use the DIV subroutine for this part of the question.)

**Question 7 (10 points)**

a. Name the phases of an instruction cycle and describe (in less than 15 words) what operations occur in each phase. You will lose points if your answers exceed 15 words.

b. For the following instructions write what operations occur in each phase of the instruction cycle.

i. ADD R1, R2, R3  
ii. LD R1, LABEL  
iii. NOT R1, R1

**Question 8 (10 points)**

The following table represents a small memory. Refer to this table for the following questions.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0001 1110 0100 0011</td>
</tr>
<tr>
<td>0001</td>
<td>1111 0000 0010 0101</td>
</tr>
<tr>
<td>0010</td>
<td>0110 1111 0000 0001</td>
</tr>
<tr>
<td>0011</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>0100</td>
<td>0000 0000 0110 0101</td>
</tr>
<tr>
<td>0101</td>
<td>0000 0000 0000 0110</td>
</tr>
<tr>
<td>0110</td>
<td>1111 1110 1101 0011</td>
</tr>
<tr>
<td>0111</td>
<td>0000 0110 1101 1001</td>
</tr>
</tbody>
</table>

The binary value within each location can be interpreted in many ways. Binary values may represent unsigned numbers, 2’s complement numbers, instructions and so forth.

a. Interpret the contents of locations 0 and 1 and 2’s complement integers.

b. Interpret the contents of location 4 as an ASCII value.
c. Interpret the contents of locations 0 and 1 and unsigned integers.

d. If the binary pattern in location 0 were interpreted as an instruction, what instruction would it represent?

e. Say the value stored in location 5 is a memory address. To which location does it refer? What hexadecimal value does that location (the one referred to by location 5) contain?

**Question 9 (10 points)**

Which addressing mode makes the most sense to use under the following conditions? There may be more than one correct answer (though you are expected to give just one), so justify your answer with a very brief explanation.

a. If you want to load one value from the current page.

b. If you want to load one value from a page other than the current page.

c. If you want to load an array of sequential addresses on the current page.
**Question 10 (10 points)**

Consider a simple implementation of a stack to store 16-bit numbers. Assume that register R0 functions as the top-of-stack pointer. Subroutine PUSH1 pushes the data in R1 on to the stack while POP1 pops data off the stack.

- **PUSH1**
  
  ```
  ADD R0, R0, #1
  STR R1, R0, #0
  RET
  ```

- **POP1**
  
  ```
  LDR R1, R0, #0
  ADD R0, R0, #-1
  RET
  ```

Now say we want to use this stack to store 32-bit numbers (each number will require two consecutive memory locations). We use the subroutines PUSH2 and POP2 to push and pop 32-bit numbers onto and off the stack. PUSH2 calls PUSH1 twice and POP2 calls POP1 twice. Assume that the lower-16 bits are in R2 while the higher 16-bits are in R3. Complete the subroutines PUSH2 and POP2 by filling in the missing instructions. You should maintain the order of bits, is, after you POP2 a number, the lower 16-bits should be in R2 and the higher 16-bits should be in R3. Keep in mind that the return address for a subroutine call is stored in R7.

- **PUSH2**
  
  ```
  ______________________________
  JSR PUSH1
  ______________________________
  JSR PUSH1
  ______________________________
  RET
  ```

- **POP2**
  
  ```
  ______________________________
  JSR POP1
  ______________________________
  ______________________________
  ______________________________
  ______________________________
  RET
  ```