Q1. Evaluate the following (10 marks, 2 marks each),

(a) \( 90_{10} = X_{2}, \Rightarrow X = \)
(b) \( 23_{16} = Y_{5}, \Rightarrow Y = \)
(c) \( 0111_{2} + 111_{2} = \)
(d) \( 0101_{2} \times 0011_{2} = \)
(e) The C-language expression
   
   \[
   \text{printf(''%d\n'\',X); where signed char X = 182;}
   \]

Q2. MU0 Microprocessor (10 marks).

Consider the following MU0 machine language program.

\[
0004 \quad 0005 \quad 0006 \quad 7000 \quad 000A \quad 0001 \quad 0000
\]

Using the figures on page 2 answer the following.

(a) Describe in words what happens on each of the lines MU0 assembly language program (3 marks).
(b) What is the contents of address 0006 after execution of the program (2 marks).
(c) Draw a timing diagram showing the contents of the program counter (PC), the instruction register (IR) and the accumulator (ACC) over the two clock periods where the instruction 2005 is being executed. Label the timing diagrams with the trace values in hexadecimal (5 marks).

Q3. Logic Synthesis and Karnaugh Maps (10 marks, 2 marks each)

A function generator consists of a counter that drives a decoder. The counter counts up from 0-15 repetitively. The decoder inputs the 4 bit address from the counter and outputs a 2-bit ramp wave. For counter values (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) the decoder produces,

\[
(00,01,00,01,10,11,00,01,10,11,00,01,10,11,11,11,00,01)\]

The decoder can be described by a truth table with 16 \( \times 4 \) bit inputs and 16 \( \times 2 \) bit outputs. Perform the following.

(1) Draw the truth table of the decoder.
(2) Draw the Karnaugh maps from the truth table.
(3) On the Karnaugh maps loop out the logical adjacencies.
(4) Write down each of the minimised Boolean logic expressions describing the decoder.
(5) Draw the circuit schematic of the decoder.

Q4. VERILOG (10 marks)

(1) Explain the difference between blocking and non-blocking assignments (4 marks).
(2) Draw a timing diagram starting at \( t = 0 \) and running for four clock periods showing the values of \( X,Y,Z \) and \( clk \) produced by the following VERILOG. (6 marks)

```verilog
module bnn;

reg clk;
reg x;
reg y;
reg z;

initial begin
   x = 0;
   clk = 0;
   forever #1 clk = ~clk;
end

always @(posedge clk) begin
   y <= x;
   x <= ~x;
end

always @(*) begin
   z = x;
end
endmodule
```
ADD

FSM State Transition Table

| state | F[1:0] | Next state | B|En | PC|En | Acc|En | M[1:0] | Xsel | Ysel | Asel | Ren | Wen |
|-------|--------|-------------|----|----|----|----|----|-------|------|------|------|-----|-----|
| 0     | xxx    | 1           | 1  | 1  | 1  | 0  | 10 | 0     | x    | 0    | 1    | 0   |     |
| 1     | 000    | 0           | 0  | 0  | 0  | 1  | 00 | 0     | 1    | 1    | 1    | 1   | 0   |
| 1     | 001    | 0           | 0  | 0  | 0  | 0  | xx | 1     | x    | 1    | 0    | 0   | 1   |
| 1     | 010    | 0           | 0  | 0  | 0  | 1  | 01 | 1     | 1    | 1    | 1    | 1   | 0   |
| 1     | 011    | 0           | 0  | 0  | 0  | 1  | 11 | 1     | 1    | 1    | 1    | 1   | 0   |
| 1     | 100    | 0           | 0  | 1  | 0  | 0  | 00 | 0     | 0    | x    | 0    | 0   | 0   |
| 1     | 101    | 0           | 0  | N  | 0  | 0  | 00 | 0     | 0    | x    | 0    | 0   | 0   |
| 1     | 110    | 0           | 0  | Z  | 0  | 0  | 00 | 0     | 0    | x    | 0    | 0   | 0   |
| 1     | 111    | 1           | 0  | 0  | 0  | 0  | xx | 0     | x    | x    | 0    | 0   | 0   |

Notes:
- N and Z are the Negative and Zero state of the Accumulator, respectively.
- (used to reduce the size of the table, as drawn)
- If a value is not going to be latched it doesn’t matter what it is!
- (e.g. ALU output for STO)
- STP operates by remaining in its evaluation state.

MU0 Instruction Set

<table>
<thead>
<tr>
<th>F</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LDA</td>
<td>Acc := [S]</td>
</tr>
<tr>
<td>1</td>
<td>STO</td>
<td>[S] := Acc</td>
</tr>
<tr>
<td>2</td>
<td>ADD</td>
<td>Acc := Acc + [S]</td>
</tr>
<tr>
<td>3</td>
<td>SUB</td>
<td>Acc := Acc − [S]</td>
</tr>
<tr>
<td>4</td>
<td>JMP</td>
<td>PC := S</td>
</tr>
<tr>
<td>5</td>
<td>JGE</td>
<td>If Acc &gt;= 0, PC := S</td>
</tr>
<tr>
<td>6</td>
<td>JNE</td>
<td>If Acc ≠ 0, PC := S</td>
</tr>
<tr>
<td>7</td>
<td>STP</td>
<td>Stop</td>
</tr>
</tbody>
</table>