1. **Two's Complement Arithmetic** Compute the following additions assuming the numbers are 8-bit 2's complement numbers. Show the result, and indicate which, if any, cause overflow. Also give the decimal equivalent of the answer in each case.

```
A)  01001010       B)    10010110       C) 10101101
  + 11110110           + 11001101         +11011110
  ----------           ----------         ---------
```

2. **Number Conversions** Fill in the blanks in the following table that shows numbers in binary, octal, and hexadecimal forms:

	Binary	Octal	Hexadecimal
A:	101010010111110000010		
B:		73433420	
C:			6A9CE

3. **RISC vs. CISC** The 80x86 architecture has many instructions that are quite complex. The modern vogue in computer architecture says that simpler instructions are better, but the 80x86 continues to thrive. List at least two advantages and two disadvantages of a CISC architecture like 80x86 relative to a RISC machine like MIPS.

4. State Machine Implementation

Given the following state diagram, and state encoding, implement the state machine using D flip flops.
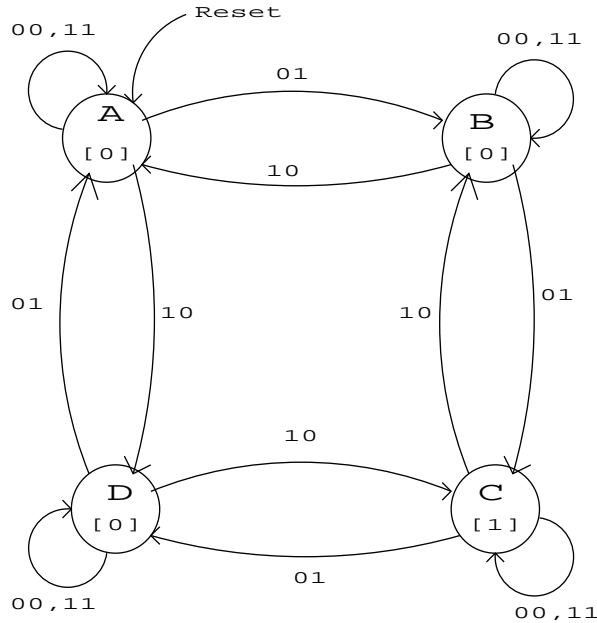

```
Inputs are I1,I0
Output is Z
State bits are P1,P0
```

(a) Fill in the state table including present state, next state, and output information.

(b) Fill in Karnaugh maps for the next-state input values for each flip flop, and the output. Circle the terms in the K-map and write the optimized boolean equations for each of the state variables and output values

(c) Write Verilog code to implement this state machine.

5. Circuit Timing

Consider a generic sequential circuit with combinational logic connected to an edge-triggered register, and the output of that register fed back to the inputs of the combinational logic.
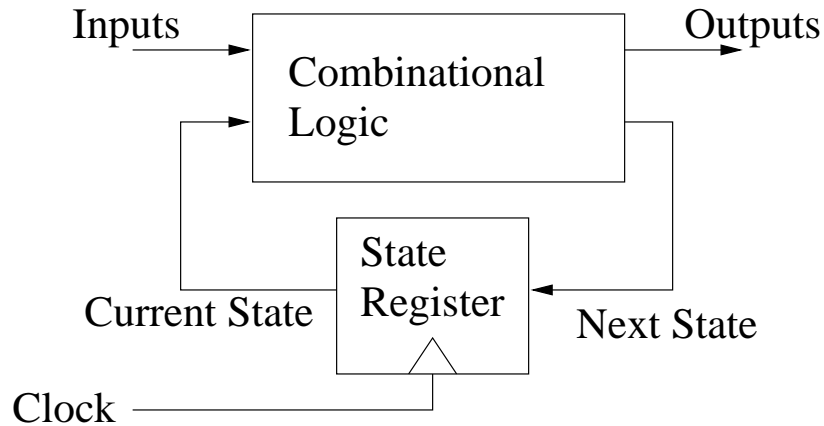
The specs are:

- Combinational logic propogation delay $T_{pd} = 100ns$
- Combinational logic contamination delay $T_{cd} = 30ns$ *Contamination delay is the minimum interval following an input change during which the previous outputs are guaranteed to remain valid*
- Register propogation delay $T_{pd} = 13ns$
- Register setup time $T_{su} = 20ns$
- Register hold time $T_h = 5ns$

Answer the following questions:

(a) What is the minimum clock period that will ensure correct operation of the circuit?

(b) By how long must any change in inputs preceed the next clock edge?

(c) There is a window in time with respect to the clock edge when the inputs are allowed to change without violating timing restrictions. Define this window in time with respect to the clock edge. That is, when can the inputs change with respect to the clock?

(d) What is the smallest time after the clock edge that outputs can be expected to be valid?

(e) What is the smallest time after a clock edge that you expect the outputs to start changing?

6. Assembly Code I

For scientific computing applications, floating point performance is often a major factor that decides the purchase of one machine over another. To measure performance, the Linpack benchmark is often used to compare machines. This benchmark involves a lot of matrix operations, such as matrix multiply and gaussian elimination. Below is an assembler routine for multiplying two nxn matrices together; it is the basis for the next questions. Note that it was not written for any particular processor and it has not been tested; don't worry about details of the code. It implements a triply nested loop, each of which executes n times.

```
;  Matrix multiply
;  Calculates C = A x B where A and B are n x n matrices
;  Recall:  C[i,j] = Sum(A[i,k]*B[k,j])    0 <= k <= n-1
;
;  Input:  R1 = Address of matrix A        NOTE: The code simply implements
;          R2 = Address of matrix B              a triply nested loop, each
;          R3 = Address of matrix C              of which is of the form:
;          R4 = Size of matrix N                 for( x = 0; x < n; x++ )
;
;  Output:  C = A x B

matrix_multiply :

        ADD  R10,R3,R0       ; R10 = Address of C[0,0]
        ADD  R11,R1,R0       ; R11 = Address of A[0,0]
        SLI  R19,R4,#2       ; R19 = 4*N (row address increment)
        ADD  R5,R4,R0        ; R5 = Loop Counter for i
loopi:                       ; for (i = 0; i < N; i++) {
        ADD R6,R4,R0         ; R6 = Loop counter for j
        ADD  R12,R2,R0       ; R12 = Address of B[0,0]
loopj:                       ; for (j = 0; j < N; j++) {
        ADD R7,R4,R0         ; R7 = Loop counter for k
        ADD R20,R11,R0       ; Move address of A[i,0] to R20
        ADD R21,R12,R0       ; Move address of B[0,j] to R21
        FSUB F2,F2,F2        ; Clear floating point register F2 = 0
loopk:                       ; for (k = 0; k < N; k++) {
        LW  F4,0(R20)        ; F4 = A[i,k]
        LW  F5,0(R21)        ; F6 = B[k,j]
        FMUL F3,F4,F5        ; F3 = A[i,k]*B[k,j]
        FADD F2,F2,F3        ; F2 += F3   (add to sum)
        ADDI R20,R20,#4      ; R20 = &A[i,k+1]
        ADDI R21,R21,R19     ; R19 = &B[k+1,j]  R19 contains row increment
        SUBI R7,R7,#1        ; Decrement k loop counter
        BNZ  loopk           ; Loop if R7 != 0
endk:                        ; }
        SW  0(R10), F2       ; Save C[i,j]
        ADDI R10,R10,#4      ; Move to next C[i,j]
        ADD  R12,R12,#4      ; R12 = &B[0,++j]
        SUBI R6,R6,#1        ; Decrement j loop counter
        BNZ  loopj           ; Loop if R6 != 0
endj:                        : }
        ADD  R11,R11,R19     ; R11 = &A[++i,0]
        SUBI R5,R5,#1        ; Decrement i loop counter
        BNZ  loopi           ; Loop if R5 != 0
endi:                        ; }
        RET
```

Evaluation

Suppose that matrix multiply is to be run on a machine which has the following CPI (Cycles per Instruction) for different types of instructions:

All integer operations (ADD, BNZ, LW,etc..)	1 cycle
Floating point add/subtract (FADD,FSUB)	5 cycles
Floating point multiply (FMUL)	10 cycles

(a) Suppose that a matrix multiply is performed with the above code for N=100. Calculate the total number of clock cycles spent executing the program.

(b) Suppose the clock rate is 100 Mhz. What is the execution time in seconds of the matrix multiply for N=100?

(c) What is the Mflops rating of the program?

(d) The vendor of the machine claims that the peak floating point performance is 20 Mflops. How does your answer in part (iii) compare to this and how do you explain any discrepancy?

7. Assembly Code II

   (a) Load/Store Code Write an assembly language instruction sequence that is equivalent to the following fragment of C code. Assume the variables are initially in memory, but not in registers. (i and j have been declared as word variables and have been initialized) Also, your code must update the variables in memory after the loop has been executed. You may use any registers that you need, and you may assume that no overflow results from the multiplication. You may use any type of load/store assembly language that you like. For example, you might use MIPS assembly code, or the same style of assembly code as in the previous question. Make sure to comment your code though!

```
while ( i > j + 2 )
    { i = i - 1;
      j = j * 2;
    }
```