CSC205 section 1 Spring 2015
Homework 6

How to Submit:
Please submit your solutions (parts A and B separately) through Blackboard.
Remember to put your name and homework number on all the documents that you
submit as attachments.

Total possible points in this homework: 0 for Part B (3 bonus pts)
(You receive 1 bonus point towards Part A if you get all Part A questions correct.)

Part A

Consider the following program written in MARIE's assembly language.

Address	Instruction
000	Load 004
001	Jump 002
002	Halt
003	00E1
004	00F2

1. Use a sequence of RTL's to describe what happens in the CPU from the start to the end of the execution of this program. Indicate the start of each new fetch-decode-execute cycle.
   (Hint: Every cycle begins with the same RTLs to fetch the instruction. Cycle 2 starts at step 7. Cycle 3 starts at step 12.)

Time step	RTL sequence performed in the CPU	Cycle
1	MAR <- PC	Start C1
2	IR <- M[MAR]	
3	PC <- PC+1	
4	MAR <- IR[11-0], decode	
5	MBR <- M[MAR]	
6	AC <- MBR	
7	MAR <- PC	Start C2
8	IR <- M[MAR]	
9	PC <- PC+1	
10	MAR <- IR[11-0], decode	
11	PC <- IR[11-0]	
12	MAR <- PC	Start C3
13	IR <- M[MAR]	
14	PC <- PC+1	
15	MAR <- IR[11-0], decode	
16	Nop	

2. Assume that the computer has just loaded the program shown in Q1, and is going to execute **Load 004**. Perform a trace of the execution of the instruction. The initial values of the registers are shown below.

Time step	RTL	PC	IR	MAR	MBR	AC
start		000	0000	000	0000	0000
1	MAR <- PC	000	0000	000	0000	0000
2	IR <- M[MAR]	000	1004	000	0000	0000
3	PC <- PC+1	001	1004	000	0000	0000
4	MAR <- IR[11-0], decode	001	1004	004	0000	0000
5	MBR <- M[MAR]	001	1004	004	00F2	0000
6	AC <- MBR	001	1004	004	00F2	00F2

3. Give the complete sequence of signal patterns, sent by the control to implement the custom part of the RTL description of the **AddI X** instruction described below:

	RTL
Default part of the fetch-decode-execute cycle	MAR <- X
Custom part of instruction	MBR <- M[MAR] MAR <- MBR MBR <- M[MAR] AC <- AC + MBR

Clock	P_5	P_4	P_3	P_2	P_1	P_0	A_1	A_0	C_r	Micro-operations
T_0	0	1	1	0	0	0	0	0	0	MBR <- M[MAR]
T_1	0	0	1	0	1	1	0	0	0	MAR <- MBR
T_2	0	1	1	0	0	0	0	0	0	MBR <- M[MAR]
T_3	1	0	0	0	1	1	0	1	0	AC <- AC + MBR
T_4	0	0	0	0	0	0	0	0	1	[reset counter]

4. Implement the hardwired control of **Jump X** using combinational logic.

Clock	P5	P4	P3	P2	P1	P0	A1	A0	Cr	Micro-operations
T0	0	1	0	1	1	1	0	0	0	PC <- IR[11-0]
T1	0	0	0	0	0	0	0	0	1	[reset counter]