Ph.D Qualifying Exam (Microprocessor Applications) – January 2013

For the first question, use the following microprocessor instruction set:

16 bit Address Bus & 16-bit Data Bus
16 16-bit registers (reg0, reg1, ..., reg15)

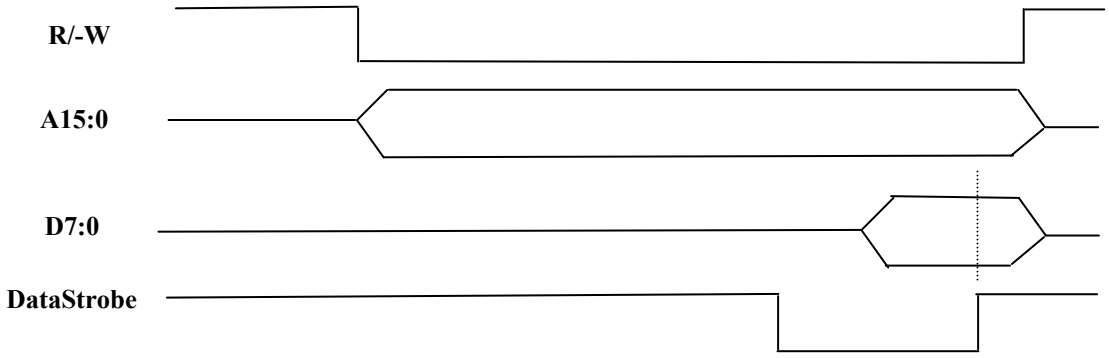
Instruction	Operands	Description	Example
LD	regD, immediate(regA)	regD = memory[immediate + regA]	LD reg3, 1000(reg5)
LDI	regD, immediate	regD = immediate	LDI reg7, 0
ST	regD, immediate(regA)	memory[immediate + regA] = regD	ST reg8, 3000(reg2)
ADD	regD, regS1, regS2	regD = regS1 + regS2	ADD reg5, reg6, reg12
SUB	regD, regS1, regS2	regD = regS1 - regS2	SUB reg5, reg6, reg12
AND	regD, regS1, regS2	regD = regS1 and regS2 (bitwise)	AND reg5, reg6, reg12
OR	regD, regS1, regS2	regD = regS1 or regS2 (bitwise)	OR reg5, reg6, reg12
INV	regD, regS1	regD = not regS1 (bitwise)	INV reg5, reg6
XOR	regD, regS1, regS2	regD = regS1 xor regS2 (bitwise)	OR reg5, reg6, reg12
SHL	regD, regS1, regS2	regD = regS1 << regS2 (shift left)	SHL reg5, reg5, reg1
BEQ	regS1, regS2, label	Branch to label if regS1 = regS2	BEQ reg2, reg3, targetLabel
BLT	regS1, regS2, label	Branch to label if regS1 < regS2	BLT reg2, reg3, targetLabel
BNE	regS1, regS2, label	Branch to label if regS1 != regS2	BNE reg2, reg3, targetLabel
J	Label	Jump to label	J targetLabel
CALL	Label	Call function “label”	CALL button_pressed
RET		Return from function	RET

1. You are given the following:
   - 1 active-low button connected to the lowest bit of port 1, which is mapped to address 2000 hex.
   - A timer that has *already* been configured to trigger an interrupt service routine (ISR) every millisecond. For simplicity, ignore any architecture-specific flags or registers that are typically cleared/set.

Write code for an ISR that debounces the button connected to port 1. The ISR should ignore any button press until a stable value of 0 occurs for 10 milliseconds. In order for software to access the state of the debounced button, the ISR should store the state to memory address 3000 hex. For example, memory address 3000 hex should contain a 1 whenever the button is not pressed and a 0 when the pressed button has been stable for 10 or more milliseconds (i.e., has been debounced).

Show all code for the ISR. Clearly define any memory locations used for variables. List any assumptions, such as memory locations that are initialized at the beginning of execution. Comment the code or show high-level pseudo-code for partial credit.

2. You are given a microprocessor with a 16-bit address bus (A15:0), an 8-bit data bus (D7:0), a read/write input (1=read, 0=write), and an active low $\overline{\text{DataStrobe}}$ (where a rising edge represents valid data). You also are a given a rising-edge triggered D flip-flop with a tri-state output that is controlled by an active-high output enable (OE). For this flip flop, the Q output is equal to the D input whenever OE = 1, otherwise Q is high Z. A timing diagram for a write is given below to illustrate the $\overline{\text{DataStrobe}}$. For a read, assume the microprocessor reads data on the rising edge of $\overline{\text{DataStrobe}}$.



(a – 40%) Create an 8-bit output port that is redundantly mapped using partial address decoding to addresses 5000 (hex) to 5fff (hex) using the provided D flip-flops and any other logic that you think is necessary.

(b – 50%) Using 4kx4 SRAM devices (i.e., 4k 4-bit words), show the devices along with the gate-level decode circuitry required to place 8k 8-bit words of memory at 6000 (hex) using full address decoding. Assume the SRAM devices have an address input, a read/write input (1=read, 0=write), an active-low chip enable (CE) input, and a 4-bit data port.

(c – 10%) Specify the address range for the 8kx8 memory in part b.