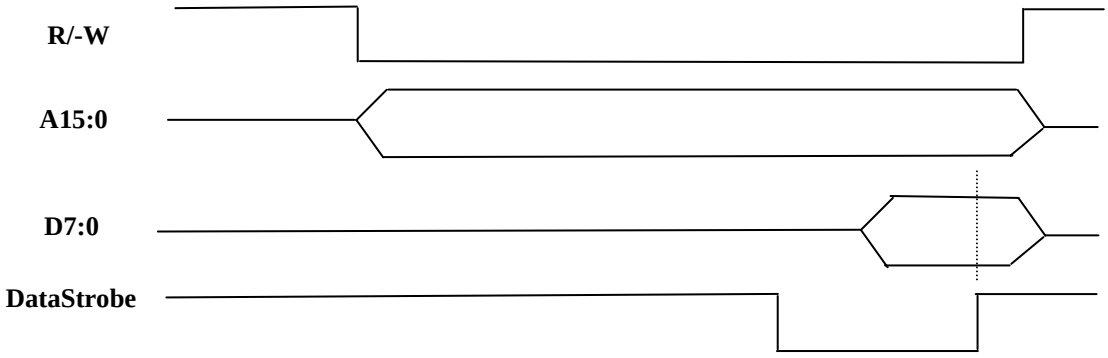
Ph.D Qualifying Exam (Microprocessor Applications) – January 2008

1. Using 16kx4 SRAM devices, show how to create a 32 KB memory. Assume a 16-bit address bus (A15:0), an 8-bit data bus (D7:0), an active high chip enable (CE), an active high output enable (OE), a read/write input (1=read, 0=write), and an active low $\overline{\text{DataStrobe}}$ (where a rising edge represents valid data). Show the decode circuitry (gate or register-transfer level) required to place the 32 KB memory at 3000 (hex) using full address decoding. Specify the address range for the 32 KB memory. 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 $\overline{\text{DataStrobe}}$.

2. Given a memory placed at memory address 4000 (hex), with a 16-bit address bus and a 16-bit data bus, write an assembly program using the following ISA that initially clears the first 1000 bytes of the memory, waits until a button mapped to port 2000 (hex) is pressed, and then writes 1000 samples from a sensor mapped to port 3000 (hex) to consecutive memory locations, starting at address 4000 (hex). To detect the button press, use polling on port 2000 (hex) bit 0, assuming false (button not pressed) when high and true (button pressed) when low. Assume the button does not require debouncing. Include comments.

Microprocessor ISA:

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	AND reg5, reg6, reg12
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
BLE	regS1, regS2, label	Branch to label if regS1 <= regS2	BLE reg2, reg3, targetLabel
BGT	regS1, regS2, label	Branch to label if regS1 > regS2	BGT reg2, reg3, targetLabel
BGE	regS1, regS2, label	Branch to label if regS1 >= regS2	BGE reg2, reg3, targetLabel
J	Label	Jump to label	J targetLabel