Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 360N, Fall 2005
Yale Patt, Instructor
Aater Suleman, Linda Bigelow, Jose Joao, Veynu Narasiman, TAs
Exam 2, November 16, 2005

Name: Solution

Problem 1 (20 points):_____

Problem 2 (20 points):_____

Problem 3 (20 points):_____

Problem 4 (20 points):_____

Problem 5 (20 points):_____

Total (100 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

GOOD LUCK!

Name:_____

Problem 1 (20 points)

Part a (6 points): A 12 x 18 array of bytes is stored in row major order, starting at location 10,000 (decimal). Memory is byte-addressable. I wish to execute a vector load of the 15th column.

What value must first be put in the Vstride register? [18]

What value must first be put in the Vlength register? [12]

What is the starting address of the vector load? [10014]

Part b (5 points): A snapshot of the taken/not-taken branch behavior of a processor is

... T T T T T T N N T T N N T N N T

If the branch predictor used is the 2-bit saturating counter, how many of the last ten branches are predicted correctly?

ANSWER: [2]

Part c (6 points): The DMA mechanism allows information to travel between [memory] and [devices], thereby freeing up [the processor] for other activities.

Part d (3 points): Most interrupts and very few exceptions are [maskable], which means they may not be serviced as expeditiously as is usually the case, or they may not be serviced at all.

Name:_____

Problem 2 (20 points)

Initially, the register file of the Tomasulo-like machine shown below contains the following data in the registers and the reservation stations. Note that one instruction has been fetched, decoded, and issued, but has not been executed yet.

	V	TAG	VALUE
R0	1	π	17
R1	1	ρ	13
R2	1	σ	14
R3	1	β	16
R4	1	β	5
R5	1	χ	3
R6	1	α	7
R7	0	α	8

	V	TAG	VALUE	V	TAG	VALUE	
	1	–	5	1	–	7	α
							β
							χ
							δ

+

	V	TAG	VALUE	V	TAG	VALUE	
							π
							ρ
							σ
							τ

*

Four more instructions are then fetched, decoded and issued in program order, none are executed. At that point the register file and reservation stations are as shown below:

	V	TAG	VALUE
R0	1	π	17
R1	1	ρ	13
R2	1	σ	14
R3	1	β	16
R4	1	β	5
R5	0	ρ	3
R6	1	α	7
R7	0	π	8

	V	TAG	VALUE	V	TAG	VALUE	
	1	–	5	1	–	7	α
	1	–	14	0	α	–	β
	1	–	5	1	–	16	χ
							δ

+

	V	TAG	VALUE	V	TAG	VALUE	
	0	β	–	0	χ	–	π
	0	χ	–	0	π	–	ρ
							σ
							τ

*

Part a (14 points): Using only instructions of the form `ADD Ri, Rj, Rk` and `MUL Ri, Rj, Rk`, where `Ri` is the destination register and `Rj` and `Rk` are the source registers, show all five instructions in program order. (Hint: You might consider doing part b first and using that result to help you solve part a.)

NOTE: The instructions in this problem are unique, but two of them can be swapped. Both orderings will receive full credit.

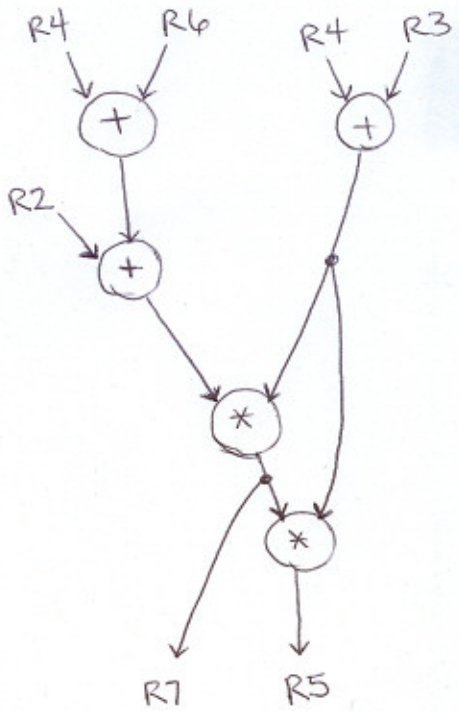
1	ADD R7, R4, R6
2	ADD R7, R2, R7
3	ADD R5, R4, R3
4	MUL R7, R7, R5
5	MUL R5, R5, R7

Instructions 2 and 3 may be swapped

Name:_____

Problem 2 continued

Part b (6 points): Show the data flow graph of the code in part a.



Name:_____

Problem 3 (20 points)

A vector processor with 11 cycle memory latency and 16-way interleaved memory, 8 vector registers of length 64, supporting vector chaining, is used to execute the vector code resulting from compiling the following high-level code:

```
for (i=0; i<10; i++) {
    D[i] = B[i] * C[i];
    E[i] = A[i] + D[i];
}
```

Part a (5 points): Write the vector code to accomplish this. Both results D[i] and E[i] have to be stored to memory. You have available the following vector instructions:

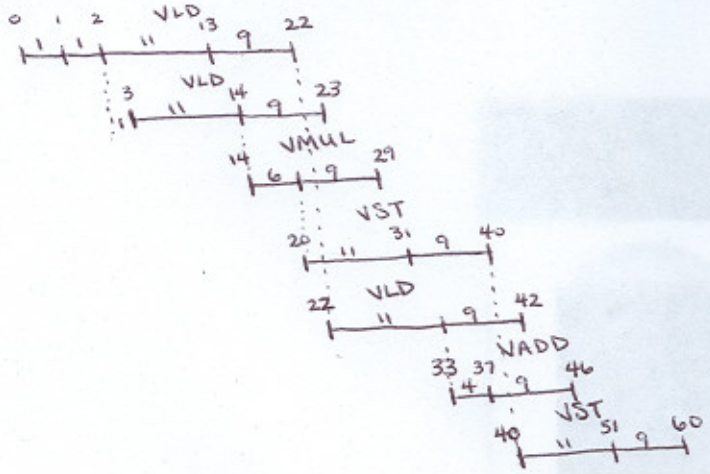
```
MOVI VLEN, #n   ; (1 cycle)
MOVI VSTR, #n   ; (1 cycle)
VADD Vi,Vj,Vk   ; Vj, Vk are sources, Vi is dest (4-stage pipelined adder)
VMUL Vi,Vj,Vk   ; (6-stage pipelined multiplier)
VLD  Vi,A       ; Vi gets loaded with contents of memory, starting at A
VST  Vi,A       ; Contents of Vi gets stored in memory, starting at A
```

```
MOVI VLEN,#10
MOVI VSTR, #1
VLD  V0, B
VLD  V1, C
VMUL V2, V1, V0
VST  V2, D
VLD  V3, A
VADD V4, V3, V2
VST  V4, E
```

Name:_____

Problem 3 continued

Part b (6 points): If the memory has 2 load ports and 1 store port, how many cycles does the program take to finish?



ANSWER: 60 cycles

Part c (9 points): If this program has to finish in fewer than 60 cycles,

the **minimum** number of load ports required is: 2.

the **minimum** number of store ports required is: 2.

With this configuration, the program takes 57 cycles to finish.

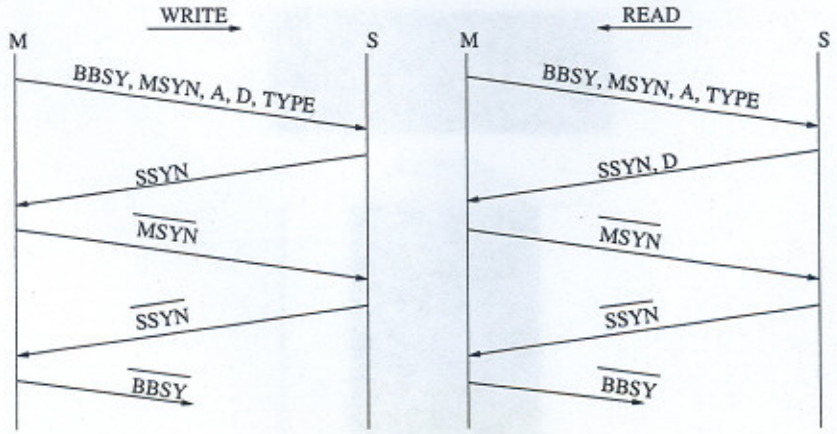
Same diagram as for part (b) except that the second VST instruction can start in cycle 37 (instead of 40) if there are 2 store ports

Name:_____

Problem 4 (20 points)

You are asked to implement the state machine for an asynchronous controller to handle both **arbitration and transaction** functions. The controller manages read and write traffic for the device attached to it.

Recall the **transaction** timing diagrams as described in class for a bus with separate address and data lines.

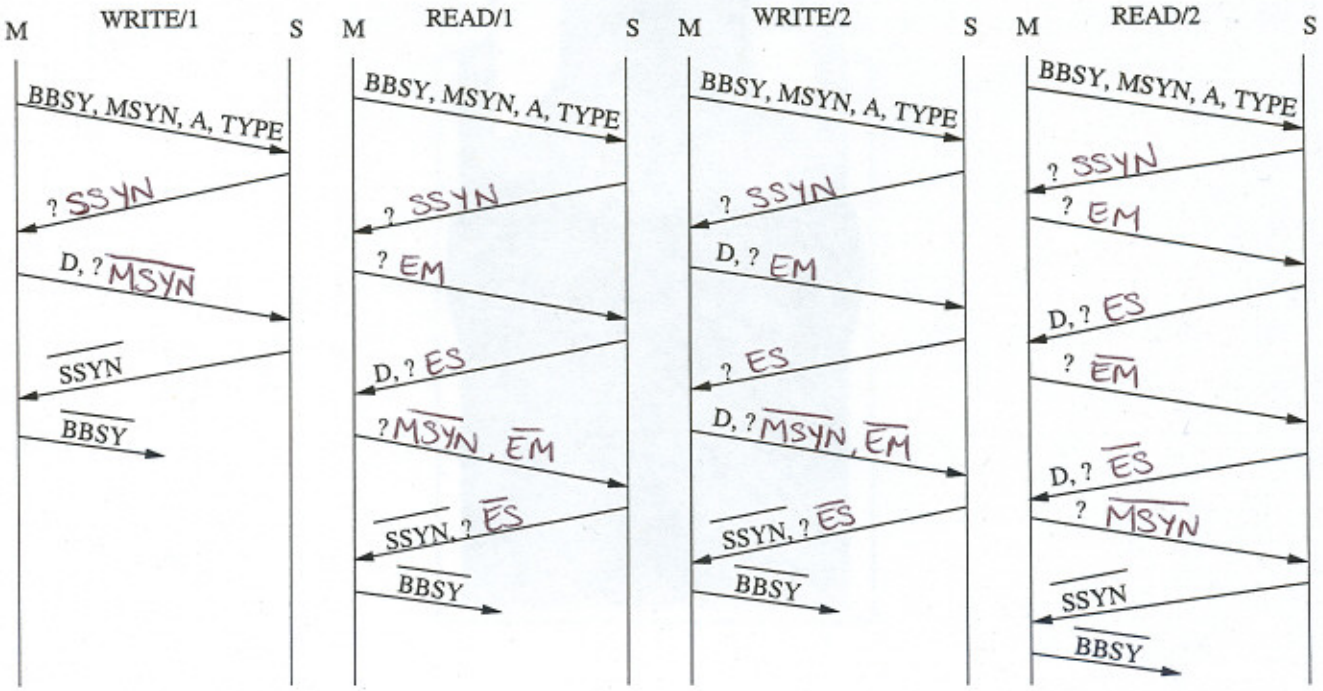


In this problem, we have a **multiplexed** address/data bus. Furthermore, **transfers can be one bus-width or two.**

The bus master knows which of the four transactions is required and asserts a 2-bit signal TYPE in conjunction with MSYN at the start of the bus cycle to notify the slave which of the four transactions (write of width one, read of width one, write of width two, and read of width two) is about to occur.

The master and slave will each need one extra control signal to perform these transactions. Call them Extra-M (EM) and Extra-S (ES).

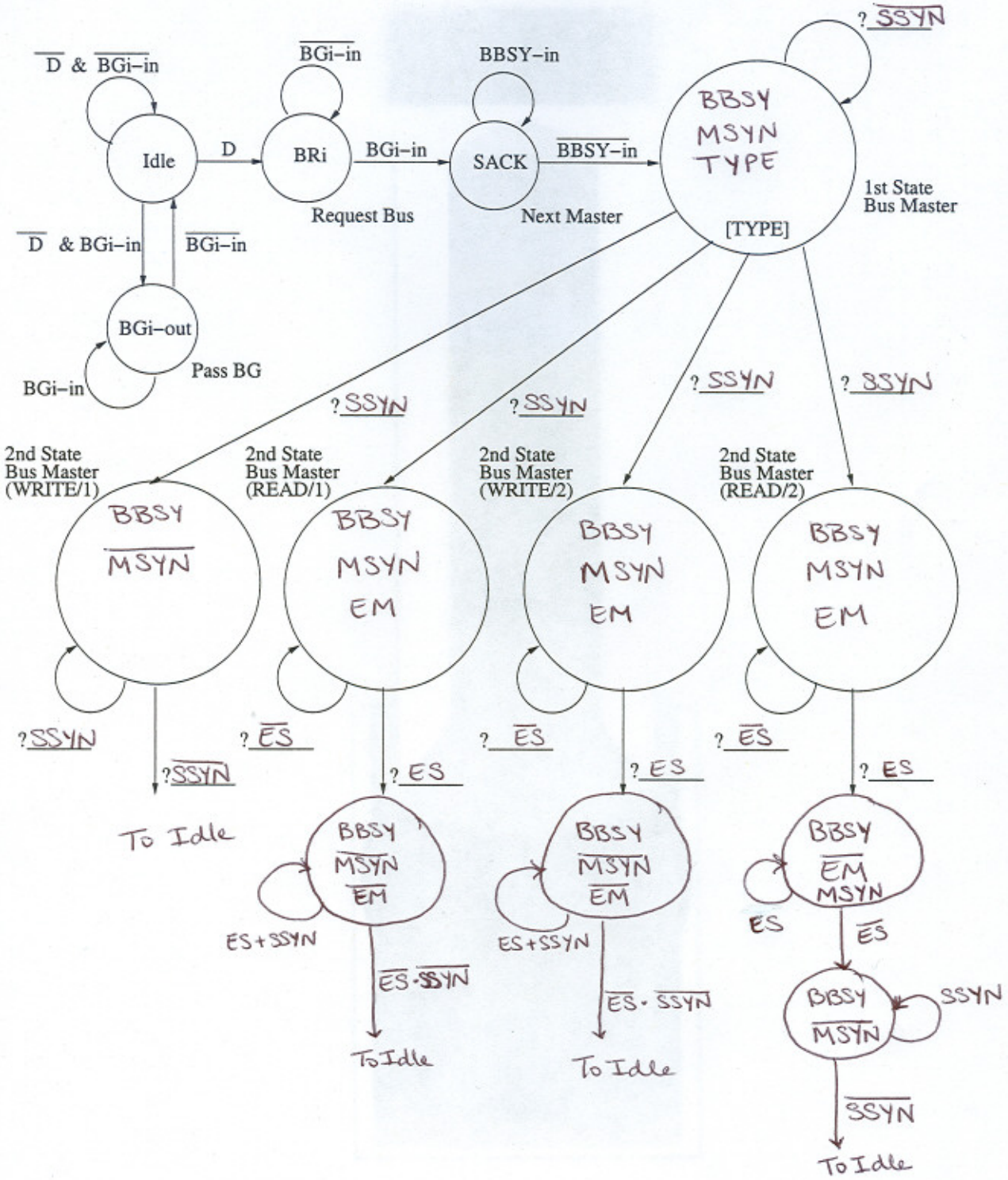
Part a (12 points): Complete the equivalent timing diagrams for the four cases by specifying the relevant control signals on every line where there is a "?". (Note: one "?" may imply more than one missing signal).



Name:_____

Problem 4 continued

Part b (8 points): Complete the state machine for the asynchronous controller to handle **both arbitration and transaction** functions. Show all relevant input signals that cause transitions and output signals according to the Moore machine convention. You **only** need to specify the control signals BUT you do need to specify ALL relevant control signals.

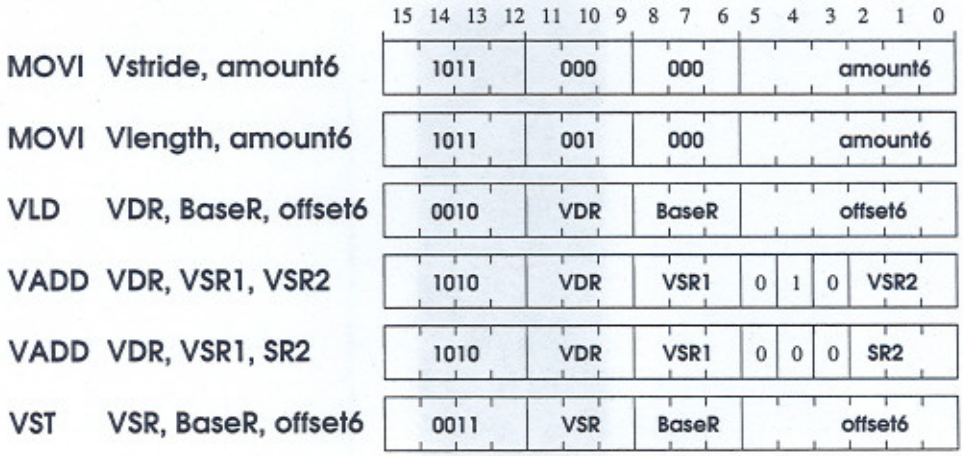


Name:_____

Problem 5 (20 points)

Little Computer Inc. is extending the LmmVC-3 (Little "mickey mouse" Vector Computer 3) that you implemented in Problem Set 5. For your convenience, the modified data path that you constructed in Problem Set 5 is shown on page 12. **The description on the remainder of this page repeats EXACTLY the specification given in Problem Set 5.**

Little Computer Inc. is now planning to build a new computer that is more suited for scientific applications. LC-3b can be modified for such applications by replacing the data type Byte with Vector. The new computer will be called LmmVC-3 (Little "mickey mouse" Vector Computer 3). LmmVC-3 ISA will support all the scalar operations that LC-3b currently supports except the LDB and STB will be replaced with VLD and VST respectively. Our data path will need to support the following new instructions:



Note: VDR = Vector Destination Register, VSR = Vector Source Register

MOVI
If IR[11:9] = 000, MOVI moves the unsigned quantity amount6 to Vector Stride Register (Vstride).
If IR[11:9] = 001, MOVI moves the unsigned quantity amount6 to Vector Length Register (Vlength).
This instruction has already been implemented for you.

VLD
VLD loads a vector of length Vlength from memory into VDR. VLD uses the opcode previously used by LDB. The starting address of the vector is computed by adding the LSHF1(SEXT(offset6)) to BaseR. Subsequent addresses are obtained by adding LSHF1(ZEXT(Vstride)) to the address of the preceding vector element.

VST
VST writes the contents of VSR into memory. VST uses the opcode previously used by STB. Address calculation is done in the same way as for VLD.

VADD
If IR[4] is a 1, VADD adds two vector registers (VSR1 and VSR2) and stores the result in VDR.
If IR[4] is a 0, VADD adds a scalar register (SR2) to every element of VSR and stores the result in VDR.

VLD, VST, and VADD do not modify the content of Vstride and Vlength registers.

Name:_____

Problem 5 continued

Since vector instructions take many cycles to complete, we want to be able to service interrupts in the middle of a vector instruction's execution. After the interrupt is serviced, the vector instruction should resume execution, but should not recompute any values that were computed prior to the interrupt.

Part a (7 points): In order to support this new feature, the interrupt initiation sequence needs to save three additional registers on the supervisor stack in addition to the PSR and PC. What are they?

Vindex

Vlength

Vstride

Part b (3 points): Describe any changes you will need to make to the RTI instruction (in fewer than 20 words).

RTI needs to POP Vindex, Vlength, and Vstride off the stack and restore them.

Part c (3 points): Assuming that the data path already includes all the support required to handle interrupts, describe the changes that you have to make to the data path shown on page 12 to implement interruptable vector instructions.

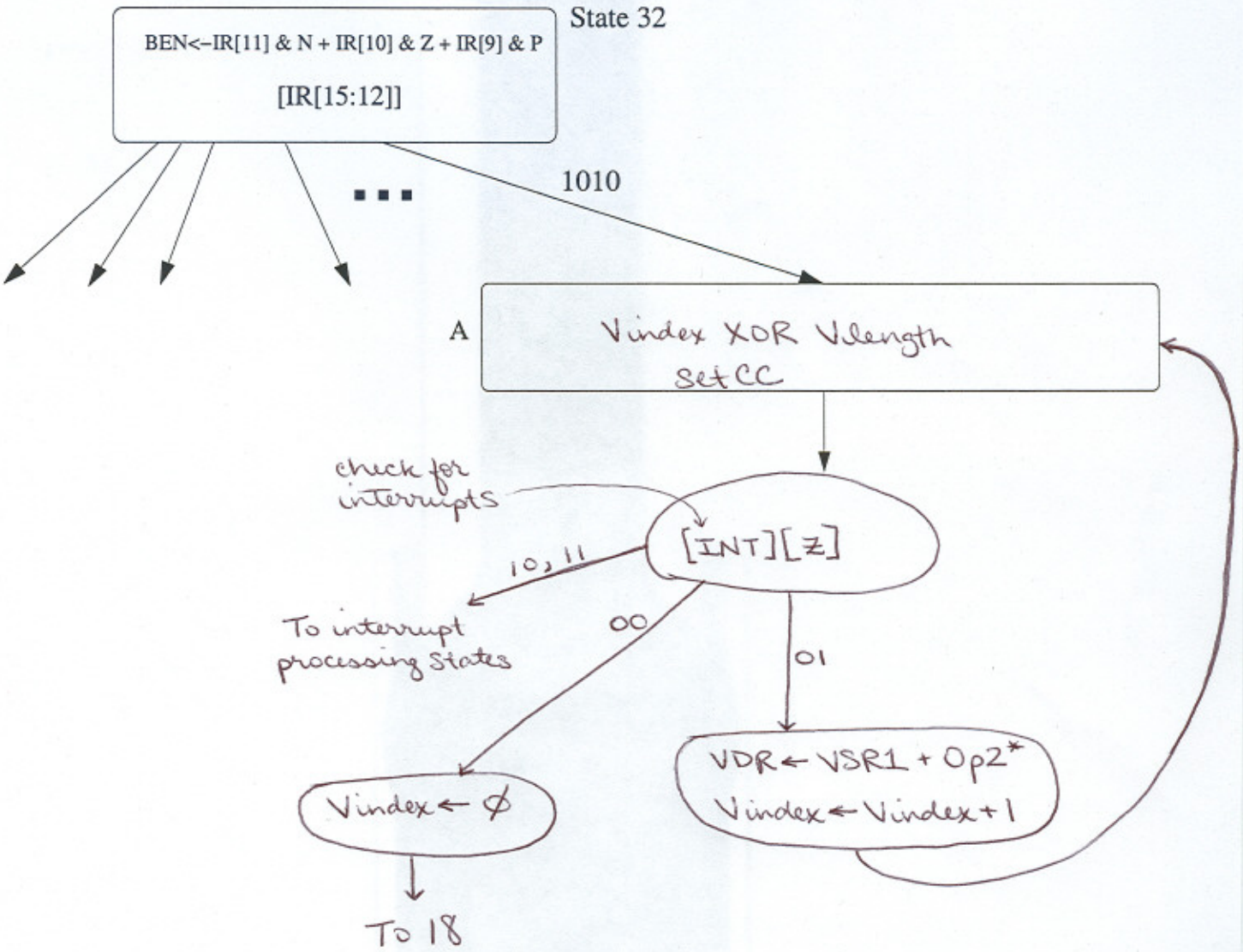
- Vindex needs a load signal and needs to source the bus
- Vlength needs a tristate device onto the bus and a gate signal
- Vstride needs a tristate device onto the bus and a gate signal

Name:_____

Problem 5 continued

Part d (7 points): We show the beginning of the state diagram necessary to implement VADD. Using the notation of the LC-3b State Diagram, add the states you need to implement VADD. Inside each state describe what happens in that state. You can assume that you are allowed to make any changes to the data path and microsequencer that you find necessary. You do not have to make/show these changes. You will need at least four states to implement VADD.

NOTES:
1. Your implementation must support servicing interrupts in the middle of VADD.
2. Clearly indicate in which state you check for interrupts.
3. Assume that the value of the Vindex register is 0 before the first vector instruction is ever executed.
4. As in the problem set, you are allowed to clobber the condition codes.
5. Make sure that your implementation works for Vlength = 0.



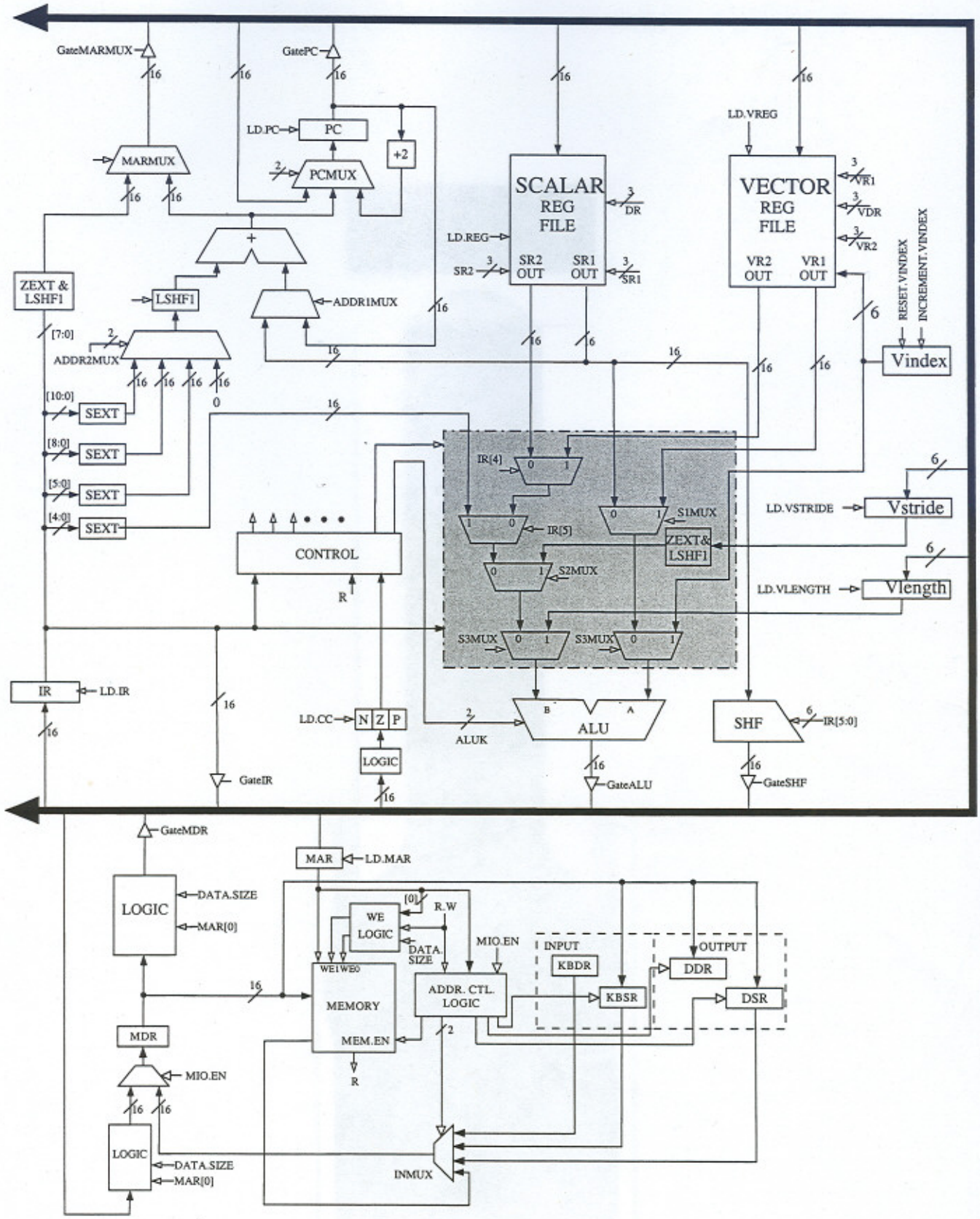


Figure 1: Modified data path to implement vector instructions