

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

EE 360N, Fall 2005
Yalé Patt, Instructor
Aater Suleman, Linda Bigelow, Jose Joao, Veynu Narasiman, TAs
Exam 1, October 19, 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: _____

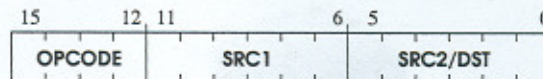
Problem 1 (20 points)

Part a (4 points): The VAX has an instruction `INSQUE ptr1, ptr2` which causes the node pointed to by `ptr1` to be inserted into a doubly linked list immediately after the node pointed to by `ptr2`. We say the doubly linked list is a

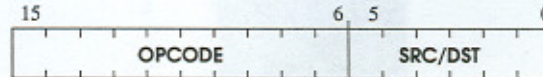
data type

supported in the VAX ISA.

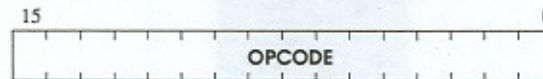
Part b (5 points): The PDP-11 had several two-operand instructions with the format



one-operand instructions with the format



and zero-operand instructions with the format



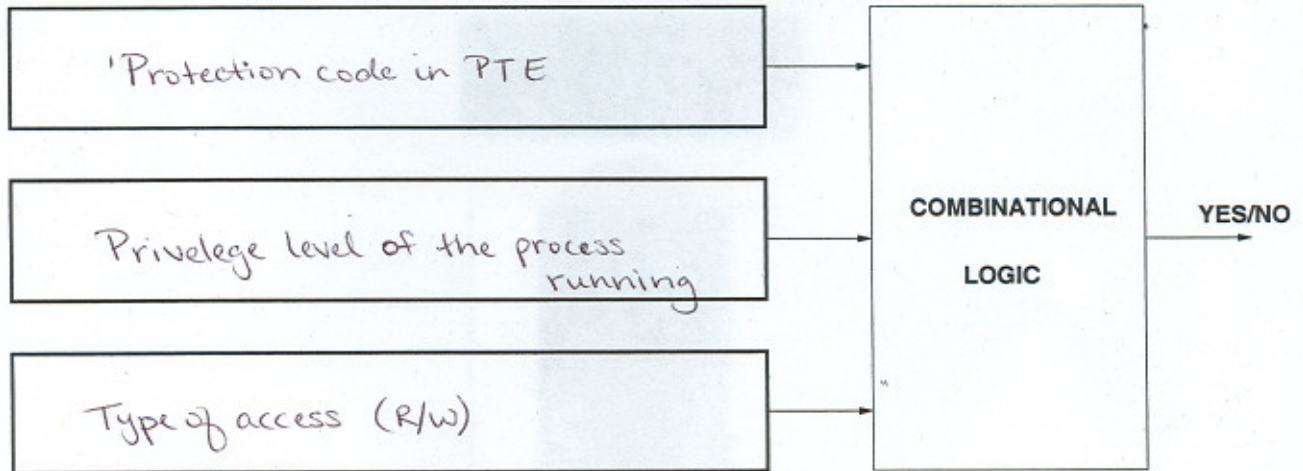
The `ADD` instruction had opcode `0110`. The **prefix property** guaranteed

No other opcode starts with 0110

Name: _____

Problem 1 continued

Part c (6 points): The virtual memory management system is called into play on every memory access instruction. In addition to translation, the right to access the information in the memory location is determined. The logic is as follows:



Identify the three elements that are input to the combinational logic required to make the yes/no access decision.

Part d (5 points): A page table has a base register and a length (aka limit) register. The purpose of the length register is

To specify the size of the image in pages

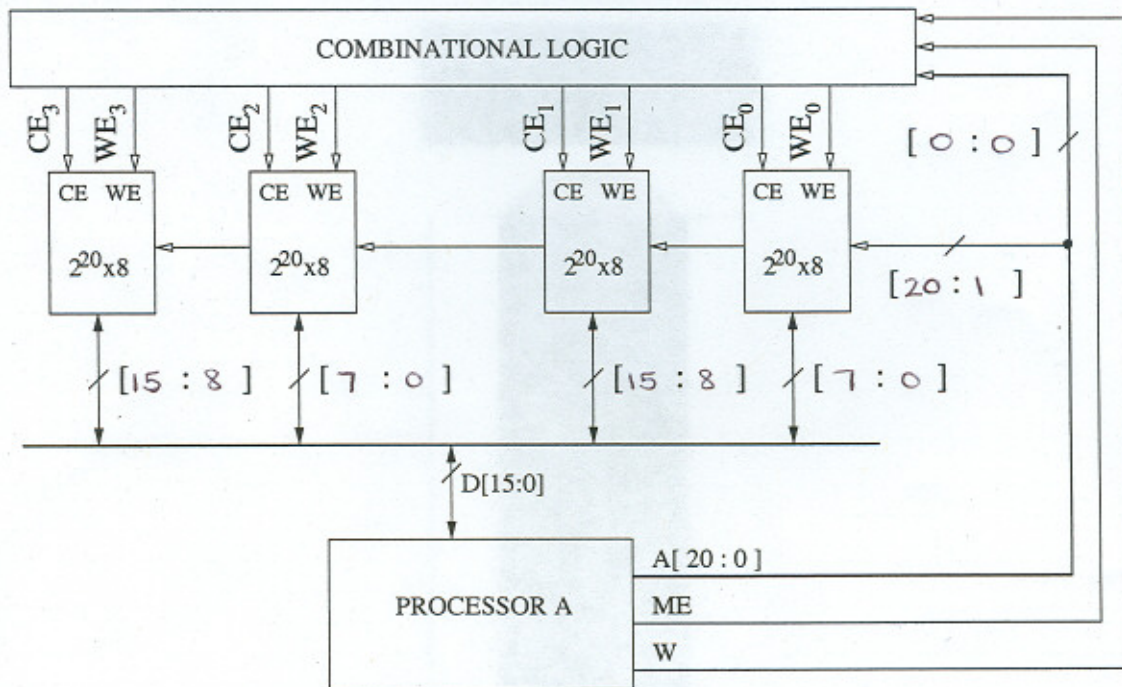
What would happen if we did not have one?

We could access memory beyond the page table, thinking it was a PTE, and compute the PA using bogus information

Name: _____

Problem 2 (20 points)

Part a (10 points): You are given four 1 MB, 8-bit wide, DRAM chips to build a 4 MB, 2-way interleaved, **word-addressable** main memory of a little-endian computer. Specify the bit fields indicated in the shaded regions for each unspecified signal.



Write the logic equations $CE_0, WE_0, \dots, CE_3, WE_3$ in terms of the inputs shown. Do not use more inputs than required to specify each output.

$CE_0 = \underline{ME \cdot \overline{A[0]}}$

$CE_1 = \underline{ME \cdot A[0]}$

$CE_2 = \underline{ME \cdot A[0]}$

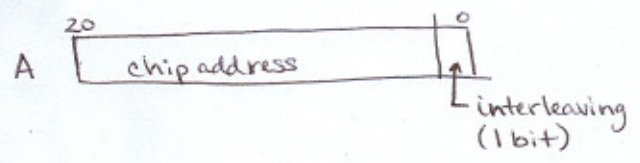
$CE_3 = \underline{ME \cdot \overline{A[0]}}$

$WE_0 = \underline{W}$

$WE_1 = \underline{W}$

$WE_2 = \underline{W}$

$WE_3 = \underline{W}$

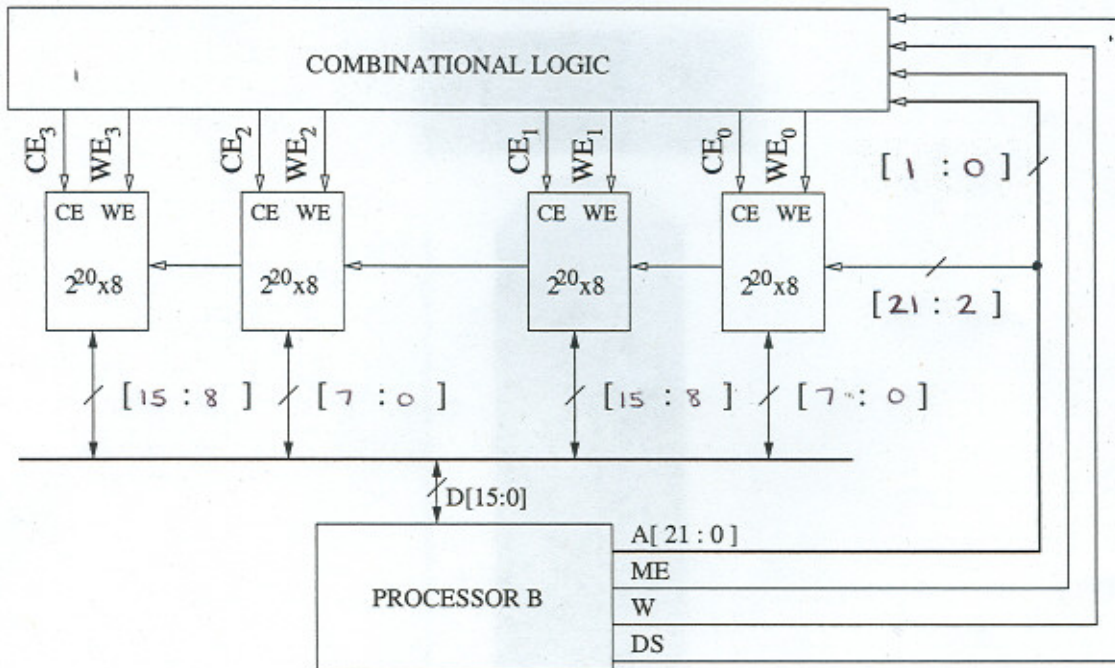


no byte-on-bus bits
because it is word addressable
and the bus is 16 bits

Name: _____

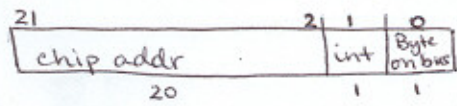
Problem 2 continued:

Part b (10 points): Suppose the memory system of Part a was changed to **byte-addressable**. That is, the four 1 MB, 8-bit wide DRAM chips form a 4 MB, 2-way interleaved, byte-addressable main memory. Specify the bit fields indicated in the shaded regions for each unspecified signal.



The ISA for Processor B supports LDB, LDW, STB, and STW instructions, but does not support unaligned accesses. Note the additional control signal DS (data size), where 1 designates word access, and 0 designates byte access. Write the logic equations for $CE_0, WE_0, \dots, CE_3, WE_3$ in terms of the inputs shown. Do not use more inputs than required to specify each output.

$CE_0 = \underline{ME \cdot \overline{A[1]}}$



$CE_1 = \underline{ME \cdot A[1]}$

$CE_2 = \underline{ME \cdot A[1]}$

$CE_3 = \underline{ME \cdot A[1]}$

$WE_0 = \underline{W \cdot (DS + \overline{DS} \cdot \overline{A[0]})} = W(DS + \overline{A[0]})$ or $W \cdot \overline{A[0]}$

$WE_1 = \underline{W \cdot (DS + \overline{DS} \cdot A[0])} = W(DS + A[0])$

$WE_2 = \underline{W \cdot (DS + \overline{DS} \cdot \overline{A[0]})} = W(DS + \overline{A[0]})$ or $W \cdot \overline{A[0]}$

$WE_3 = \underline{W \cdot (DS + \overline{DS} \cdot A[0])} = W(DS + A[0])$

Name: _____

Problem 3 (20 points):

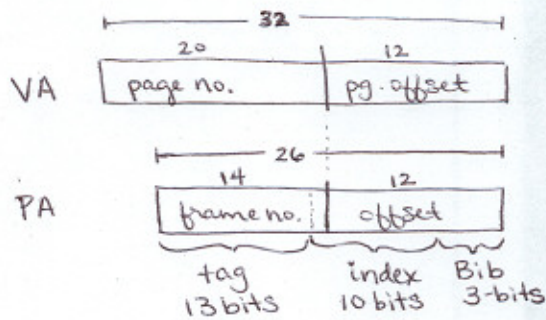
A physically addressed cache is one in which the physical memory locations of each cache line can be inferred from its tag store information.

We want to design a physically addressed, 4-way set-associative, write back cache with a random replacement policy. Line size is 8 bytes. Virtual memory is byte-addressable, with a 32-bit address space, and the page size is 4kB. Physical memory has a 26-bit address space. The memory management system restricts even numbered virtual pages to even numbered physical frames and odd numbered virtual pages to odd numbered physical frames.

We want to be able to access the TLB, tagstore, and data store of this cache concurrently.

So we do not have to worry about back-translation tables, restrictions on shared frames, or invalidating the cache on a context switch, we insist that a line of physical memory cannot be present in more than one slot in the cache at the same time.

Part a (10 points): What is the largest size (in bytes) of the data store of the cache that will satisfy all of the above?



page size = 4kB \Rightarrow 12-bit pg offset
block size = 8B \Rightarrow 3-bit Byte in block

$$\begin{aligned} \text{Data store size} &= \# \text{ rows} \times \text{associativity} \times \text{block size} \\ &= 2^{10} \times 4 \times 8 \text{ B} \\ &= 2^{15} \text{ B (or 32 kB)} \end{aligned}$$

To satisfy all the requirements, 1 bit of the index must come from the page number of the VA

2^{15} Bytes

Part b (10 points): What is the minimum size (in bits) of the corresponding tag store?

Each entry in the tag store needs

- Tag (13 bits)
- Valid bit (1 bit)
- Dirty bit (1 bit)
- No replacement bits

= 15 bits per entry

Tag store size = #rows \times #entries per row \times #bits per entry

$$\begin{aligned} &= 2^{10} \times 4 \times 15 \text{ bits} \\ &= 2^{10} \times 60 \text{ bits} \\ &= 60 \text{ k bits} \end{aligned}$$

60k bits

Name: _____

Problem 4 (20 points)

A machine has a two-level virtual address translation mechanism similar to the VAX.

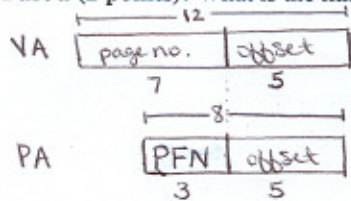
The machine's memory system is defined as follows:

- Virtual Memory Size: 4KB
- Physical Memory Size: 256 Bytes
- Page Size: 32 Bytes
- Memory is byte-addressable

Virtual address space is partitioned into P0 space, P1 space, system space and reserved space. The space a virtual address belongs to is specified by the most significant two bits of the virtual address, with 00 indicating P0 space, 01 indicating P1 space, and 10 indicating system space.

A PTE consists of a Valid bit, 3 bits for protection, a Reference bit, a Modified bit, and the Page Frame Number (PFN).

Part a (2 points): What is the minimum size of a PTE in Bytes?



$$\begin{aligned} \text{PTE} &= \text{PFN} + \text{Valid bit} + \text{protection bits} + \text{reference bit} + \text{modify bit} \\ &= 3 + 1 + 3 + 1 + 1 \\ &= 9 \text{ bits} \\ &\Rightarrow \text{each PTE is 2 bytes} \end{aligned}$$

2

 Bytes

Part b (2 points): Using the PTE size from part a, what is the maximum size of a process' P0 page table in Bytes?

$$\begin{aligned} \# \text{ of pages in } P_0 \text{ space} &= \frac{\left(\frac{1}{4}\right)(2^{12})}{2^5} \quad (\# \text{ of bytes in } P_0 \text{ space}) \\ & \quad (\text{size of a pg, in bytes}) \\ &= 2^5 \text{ pages} \end{aligned}$$

Each page of P0 space needs a PTE in the page table

64

 Bytes

$$\begin{aligned} \text{page table} &= \# \text{ entries} \times \text{Size of PTE} \\ &= 2^5 \times 2 \text{ Bytes} \\ &= 64 \text{ Bytes} \end{aligned}$$

Name: _____

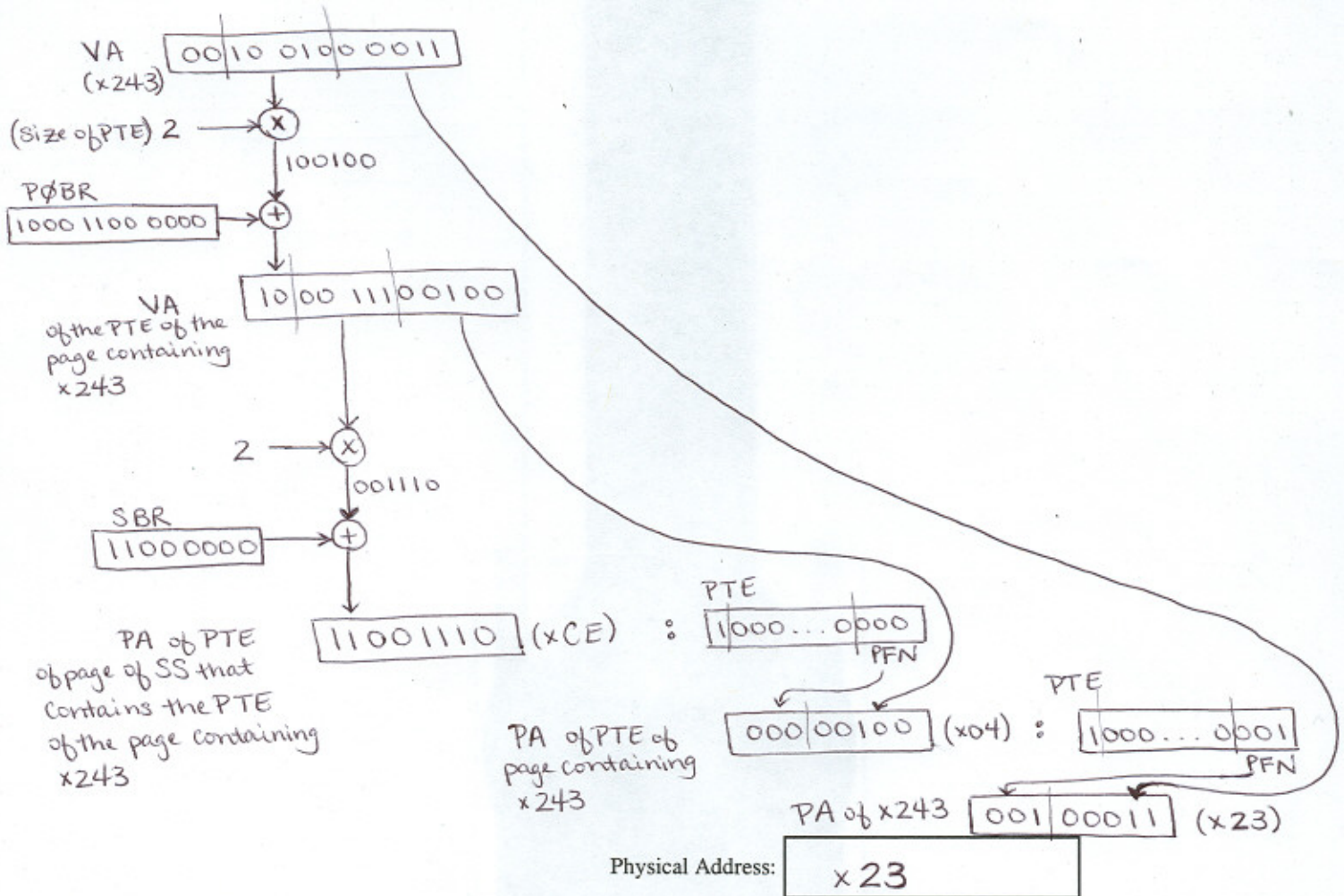
Problem 4 continued

Part c (8 points): If POBR is x8C0 and the SBR is xC0, what is the physical address corresponding to virtual address x243? Assume that the valid bit of a PTE is its most significant bit and the PFN is stored in its least significant bits.

The following blocks are the contents of a selected areas of physical memory that you may need to solve this problem:

xCE	x8000	xE0	x8004	x00	x0002	x80	x0438
xD0	x0007	xE2	x00A0	x02	x8002	x82	x0000
xD2	x8003	xE4	x0003	x04	x8001	x84	x8005
xD4	x0000	xE6	x0007	x06	x0134	x86	x0002

Note: There may be more information contained here than needed to solve the problem.



Name: _____

Problem 4 continued

Part d (3 points): Let's say that during the translation of virtual address x243, the valid bit of the 2nd PTE encountered was a 0, resulting in a page fault. What range of virtual addresses in P0 space would also result in this page fault?

The 2nd PTE is the PTE corresponding to the original VA (x243). A PTE maps one page of virtual memory, so all VAs on the same page as x243 will also result in a page fault. To find that range, take x243 and make the low 5 bits 0 (for the lower bound) and 1 (for the upper bound).

$$x243 = 0010\ 0100\ 0011$$

$$x240 = 0010\ 0100\ 0000$$

$$x25F = 0010\ 0100\ 1111$$

From

x240

to

x25F

Part e (5 points): Let's say that during the translation of virtual address x243, the valid bit of the 1st PTE encountered was a 0, resulting in a page fault. What range of virtual addresses in P0 space would also result in this page fault?

The 1st PTE is the PTE corresponding to the VA that points to the P₀ page table, which resides in System Space. This entry being invalid means that an entire page of the P₀ page table is not in physical memory. Since 1 page is 32B and each PTE is 2B, 1 page contains 16 PTEs and, therefore, maps 16 pages of P₀ space. Any VA in these 16 pages would cause a page fault.

Since 16 pages = 512B = 2⁹B, the lower bound of the range will be x243 with the low 9 bits all 0, and the upper bound will be x243 with the low 9 bits all 1.

$$x243 = 0010\ 0100\ 0011$$

$$x200 = 0010\ 0000\ 0000$$

$$x3FF = 0011\ 1111\ 1111$$

From

x200

to

x3FF

Name: _____

Problem 5 (20 points):

We wish to use the unused opcode 1010 to implement a new instruction **FETCH&OP**, which loads a word from memory, performs an operation (OP) on it, and stores the result back to the same memory location. The original value which was loaded from memory is saved in R7, and the condition codes are set based on that value. OP can be ADD, AND, or XOR. The second operand can be either the contents of a register or an immediate value. The specification of this instruction is as follows:

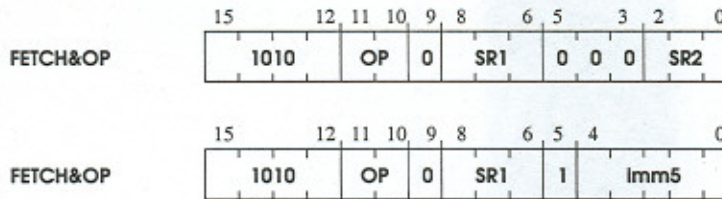
Assembler Formats

FETCH&ADD SR1, SR2
FETCH&ADD SR1, imm5

FETCH&AND SR1, SR2
FETCH&AND SR1, imm5

FETCH&XOR SR1, SR2
FETCH&XOR SR1, imm5

Encodings



Encoding of the 2-bit value OP is as follows:

ADD/00

AND/01

XOR/10

The value 11 is reserved

Operation

```
TEMP = MEM[SR1];  
if ( bit[5] == 0 )  
    MEM[SR1] = TEMP OP SR2;  
else  
    MEM[SR1] = TEMP OP SEXT(imm5);  
R7 = TEMP;  
setcc();
```


Name: _____

Problem 5 continued:

To implement FETCH&OP, we have added the following structures to the LC-3b data path:

1. A 16-bit register called TEMP
This requires a new control field, LD.TEMP, which is the load enable signal for the TEMP register.
2. A two-input mux, AMUX, to the A input of the ALU.
This requires a new control field, AMUX, specified as follows:
SR1/0, the SR1 source from the register file
TEMP/1, the value of the TEMP register
3. A two-input mux, ALUMUX, to the control input of the ALU.
This requires a new control field, ALUMUX, specified as follows:
ALUK/0, the ALUK bits from the control store
X/1, the 2-bit signal X

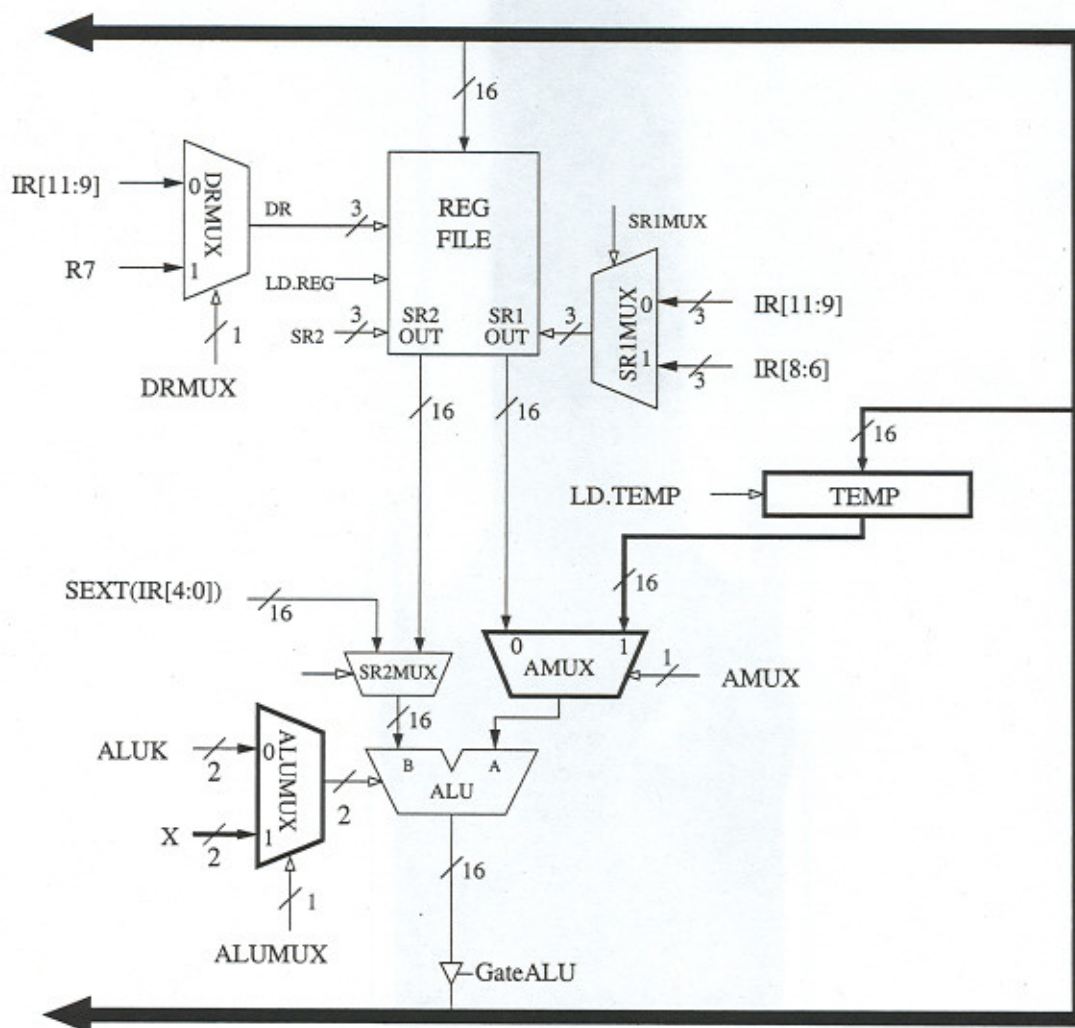


Figure 1: Modified datapath to support FETCH&OP instruction

Name: _____

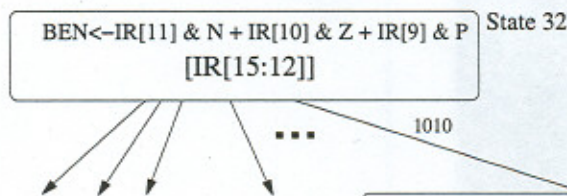
Problem 5 continued:

Part a (2 points): Identify the 2-bit signal X shown on the data path diagram:

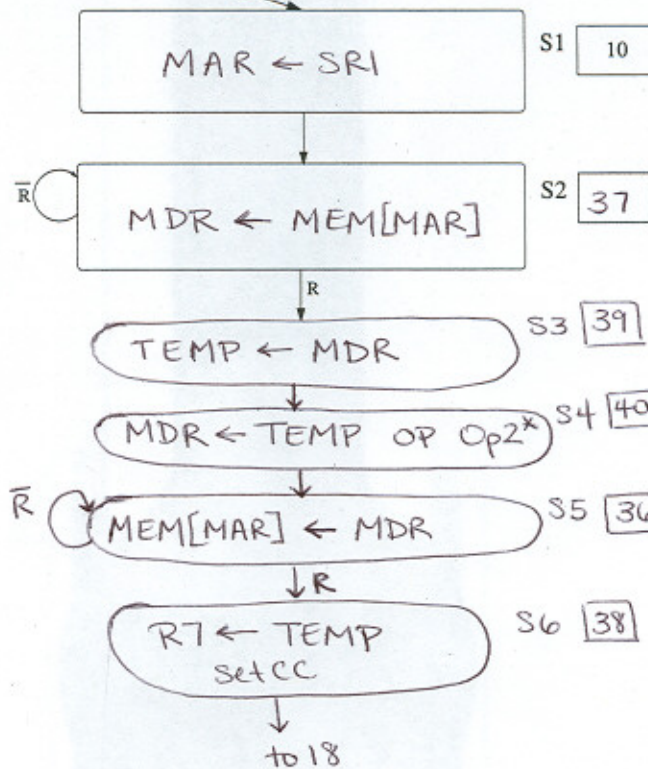
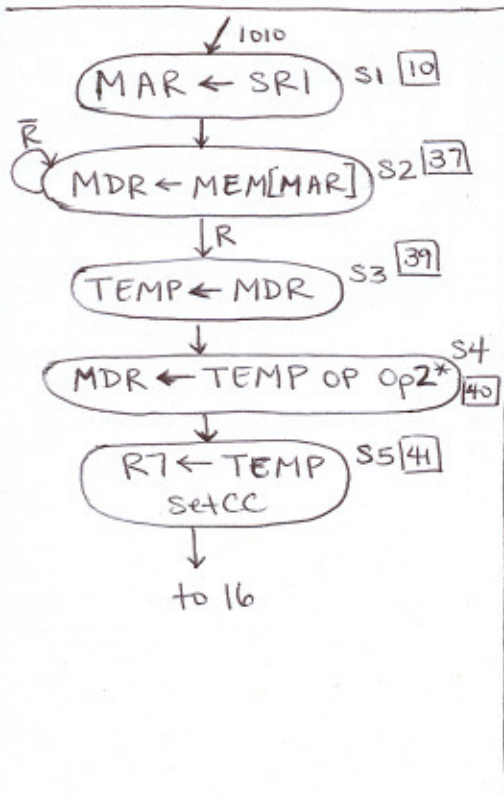
X: IR[11:10]

Part b (10 points): We show the beginning of the state diagram necessary to implement FETCH&OP. Using the notation of the LC-3b State Diagram, add the bubbles you need to implement the FETCH&OP instruction. Your job is to **describe** inside each bubble what happens in each state and **assign** each state an appropriate state number. You should be able to implement this state diagram using 5 or 6 new states.

NOTE: Your solution must work for instructions where SR2 is R7 e.g. FETCH&ADD R1, R7



Two possible solutions are shown



*Op2 can be SR2 or SEXT(imms)

Name: _____

Problem 5 continued:

Part c (8 points): The processing in each state you just added is controlled by asserting or negating each control signal. Enter a 1 or a 0 as appropriate for the microinstructions corresponding to the states you have added.

NOTE: Please use the encodings specified on the next page for all the signals.

	LD.REG	LD.CC	LD.TEMP	LD.MAR	LD.MDR	AMUX	SRIMUX	DRMUX	ALUK[1:0]	ALUMUX	Gate.ALU	Gate.MDR	MIO.EN	R.W.	DATA.SIZE
S1	0	0	0	1	0	0	1	0	1 1	0	1	0	0	0	0
S2	0	0	0	0	1	0	0	0	0 0	0	0	0	1	0	1
S3	0	0	1	0	0	0	0	0	0 0	0	0	1	0	0	1
S4	0	0	0	0	1	1	0	0	0 0	1	1	0	0	0	0
S5	0	0	0	0	0	0	0	0	0 0	0	0	0	1	1	1
S6	1	1	0	0	0	1	0	1	1 1	0	1	0	0	0	0

These control signals correspond to the 6 state solution on the previous page.