Name: Chirag Sakhuja

Problem 1 (20 points):_____
Problem 2 (25 points):_____
Problem 3 (25 points):_____
Problem 4 (30 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.

Please read the following sentence, and if you agree, sign where requested: I have not given nor received any unauthorized help on this exam.

Signature:________________________________________________________

GOOD LUCK!
Problem 1 (20 points): Answer the following questions.

Part a (5 points): A 16KB, 2-way set associative, physically-indexed, physically-tagged cache has a line size of 64B. We wish to use it with x86 ISA, which has a page size of 4KB. Assuming no help from the Operating System, can we design the cache such that the TLB, Tag Store, and Data Store accesses can all be made at the same time?

Yes/No (Circle one).

Explain.

For this design to be possible, the index and offset fields must be a total of 13 bits, but only 12 bits are guaranteed not to change during translation.

Part b (5 points): A computer implements IEEE Floating Point, with the one exception that each data element is represented with 12 bits. Six bits are used for the exponent.

What is the smallest positive normalized number that can be represented exactly? Hint: Show result as power of 2.

\[ \frac{1}{2} \times 2^{6} = \frac{1}{2} \times 64 = 32 \]

Bias is in the middle, so \( 2^{5} - 1 = 31 \)

\[ 2^{-36} \]

What is the smallest positive number that can be represented exactly? Hint: Show result as power of 2.

\[ 0.00001 \times 2^{-30} \]

\[ 2^{-35} \]

Part c (5 points): The microarchitecture of the VAX-11/780 has a 32-bit register containing the value in hex: 0x66666666. Could this register be of any use in performing BCD arithmetic. Yes/No. Explain.

This value can be used to adjust the BCD numbers before performing an ADD.

Part d (5 points): Interrupts and Exceptions both interrupt the normal execution of a program, put the machine in a consistent state, and go to a service routine for handling. There are, however, many differences between interrupts and exceptions, for example when they are carried out, their priority level, the context within which they operate, etc., mostly due to the fact that interrupts are caused by events that are

external

while exceptions are caused by events that are

internal
**Problem 2 (25 points):** We have augmented the LC-3b with the memory hierarchy shown below.

Addr(A), Addr(B), and Addr(C) represent addresses that access respectively L1, L2, and Memory. Data(B) and Data(C) each transfer a full cache line.

The table below shows a sequence of five memory accesses from the LC-3b core. If ADDR(A) misses in L1, an access is required to L2. If ADDR(B) misses in L2, an access is required to Memory. Each of the five requests from the LC-3b core must complete before the next access from the LC-3b core is initiated.

<table>
<thead>
<tr>
<th>ADDR(A)</th>
<th>ADDR(B)</th>
<th>ADDR(C)</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3000</td>
<td>0x100</td>
<td>0x100</td>
<td>Read</td>
</tr>
<tr>
<td>0x3000</td>
<td>0x100</td>
<td>0x100</td>
<td>Read</td>
</tr>
<tr>
<td>0x3003</td>
<td>0x100</td>
<td>0x100</td>
<td>Write</td>
</tr>
<tr>
<td>0x3004</td>
<td>0x104</td>
<td>0x104</td>
<td>Read</td>
</tr>
<tr>
<td>0x3008</td>
<td>0x108</td>
<td>0x108</td>
<td>Read</td>
</tr>
<tr>
<td>0x3008</td>
<td>0x108</td>
<td>0x108</td>
<td>Read</td>
</tr>
<tr>
<td>0x8000</td>
<td>0x100</td>
<td>0x200</td>
<td>Write</td>
</tr>
<tr>
<td>0x8000</td>
<td>0x100</td>
<td>0x200</td>
<td>Read</td>
</tr>
<tr>
<td>0x8000</td>
<td>0x100</td>
<td>0x200</td>
<td>Read</td>
</tr>
</tbody>
</table>

You may make the following assumptions:

- Virtual addresses are 16 bits and the page size is 256B
- The TLB has 2 entries and is fully associative
- All accesses to the TLB are hits
- The L1 and L2 are both physically-indexed, physically-tagged
- The L1 contains 64 sets
- If a cache line is present in the L1, it will also be present in the L2 (although the contents of L2 may not be correct)
- The caches are initially empty
Part a (3 points): How many bytes are in an L1 cache line?

$\text{Bytes} = 4$

Part b (3 points): How many bytes are in an L2 cache line?

$\text{Bytes} = 8$

Part c (3 points): Fill in the VPNs and the PFNs of the two TLB entries.

<table>
<thead>
<tr>
<th>VPN</th>
<th>PFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3000</td>
<td>x1</td>
</tr>
<tr>
<td>0x8000</td>
<td>x2</td>
</tr>
</tbody>
</table>

$0x3000$ translates to $0x100$

$0x8000$ translates to $0x200$

Low 8 bits are page offset

Part d (4 points): Is the L1 cache write through or write back? (Circle one)

Write Through / Write Back

The write to $0x3003$ did not generate a write to the L2

Part e (4 points): Is the L2 cache write through or write back? (Circle one)

Write Through / Write Back

The write to $0x100$ did not generate a write to the memory

Part f (4 points): What is the associativity of the L1 cache?

$\text{Way(s)} = 1$

When $0x8000$ is brought into the L1, it evicts $0x3000$, which is the only other location in the same set (if it was more than 1 way, it would not be necessary to evict $0x3000$).

Part g (4 points): What is the minimum possible associativity of the L2 cache?

$\text{Way(s)} = 1$

It may seem like the same concept as part f except with locations $0x100$ and $0x200$. However, we do not have enough information to know whether those addresses map to the same set or not, so the minimum possible associativity is still 1.
**Problem 3 (25 points):** Let us use one of the unused opcodes to add an instruction DOTPRODUCT (i.e., dot product) to the LC-3 ISA. Its format will be

\[
\begin{array}{c}
1010 \\
\text{DR/SRA} \\
\text{SRB} \\
vlen5
\end{array}
\]

The DOTPRODUCT of two vectors is computed as shown below:

\[
\sum_{i=0}^{n-1} A[i] \times B[i]
\]

The two vectors are stored in memory. Their starting addresses are contained in SRA and SRB, and their length is specified as an immediate 5-bit value (vlen5). The instruction stores the result of the dot product in the register specified by DR. Assume vlen5 is not zero.

For this problem, you can assume no overflow will occur. Note: execution of this instruction will destroy the initial contents of SRA and SRB.

Your job: augment the LC-3 state machine, data path and microsequencer shown on the next three pages to add DOTPRODUCT to the LC-3 ISA.
**Part a, The state machine (12 points):** From decode (state 32), ten states are needed to complete the execution of DOTPRODUCT. One of the states (state 44) has been partially specified. Your job is to complete the specifications of all the states and add the missing state numbers.

```
Name:________________________

From state 32

CTR <- vien5
CLR, SUM

MAR <- SRA

MDR <- M[MAR]

TEMP <- MDR

MAR <- SRB
CTR <- CTR-1

MDR <- M[MAR]  
sc+cc

SUM <- MAC_OUT [Z]

DR <- SUM

SRA <- SRA - 1

SRB <- SRB - 1

To state 18

state 10

state 40

state 41

state 43

state 44

state 36

state 38

state 47

state 39

state 46
```
Part b, The data path (10 points): We have added GateIR[4:0], ALUMUX, made changes to DRMUX, provided registers for CTR (with built-in decrement functionality), TEMP, and SUM as well as a multiply-and-accumulate (MAC) unit. The MAC computes $M1 \times M2 + A1$. Your job is to implement the changes you made in Part a by connecting the necessary structures to the LC-3b datapath. You are free to add control signals and tri-state buffers as needed.
Part c, The microsequencer (3 points): To make this work, we need to add a COND2 control signal to the microsequencer. The only thing missing to complete the change to the microsequencer is the box labeled A. Your job: fill in the box labeled A.
Problem 4 (30 points): Suppose the LC-3b ISA had a 12-bit, byte-addressable, virtual address space with two levels of virtual to physical translation, similar to the VAX.

A PTE is shown below:

<table>
<thead>
<tr>
<th>V</th>
<th>M</th>
<th>ACC</th>
<th>0...0</th>
<th>PFN</th>
</tr>
</thead>
</table>

It includes a Valid bit, a Modify bit, a 2 bit Access Control field, some number of unused bits (i.e., 0..0), and the PFN. The low bits of the PTE are used for the PFN.

The access control bits are defined as follows:
- 00: none
- 01: read-only
- 10: read-write
- 11: —

The virtual address space is divided evenly into two regions. The first half is user space, the second half is system space.

The user space page table starts at the beginning of a page. The system space page table starts at the beginning of a frame. We require 1/4 of physical memory to store the entire system page table.

A user program fetches and executes one LC-3b instruction, resulting in six accesses to physical memory, as shown by the following table:

<table>
<thead>
<tr>
<th>VA</th>
<th>PA</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>x00EA</td>
<td>x9...4</td>
</tr>
<tr>
<td>x08B0</td>
<td>x90</td>
<td>x9...2</td>
</tr>
<tr>
<td>x0100</td>
<td>x40</td>
<td>x2862</td>
</tr>
<tr>
<td>—</td>
<td>x00EE</td>
<td>x9...5</td>
</tr>
<tr>
<td>x08F6</td>
<td>xB6</td>
<td>x9...1</td>
</tr>
<tr>
<td>x0572</td>
<td>x32</td>
<td>x10A0</td>
</tr>
</tbody>
</table>

Note: Since the size of the PTE has not been given, entire PTEs are not shown in the above table.

Part a (17 points): Fill in the entries for the following:

\[
\begin{align*}
\text{VA} &= \text{x0572} \\
\text{PA} &= \text{x0100} \\
\text{PTESIZE} &= 2 \\
\text{SBR} &= \text{x00E0} \\
\text{UBR} &= \text{x8C6} \\
\end{align*}
\]
Part b (3 points): After the instruction is executed, the register file is as shown:

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>x0550</td>
</tr>
<tr>
<td>R1</td>
<td>x0590</td>
</tr>
<tr>
<td>R2</td>
<td>x00A0</td>
</tr>
<tr>
<td>R3</td>
<td>x0200</td>
</tr>
<tr>
<td>R4</td>
<td>xFFA0</td>
</tr>
<tr>
<td>R5</td>
<td>x000C</td>
</tr>
<tr>
<td>R6</td>
<td>x0010</td>
</tr>
<tr>
<td>R7</td>
<td>x0100</td>
</tr>
</tbody>
</table>

What LC-3b instruction was executed?

\[
x \times 1E = 011110
- x \times 1E = 100010
\]

Part c (10 points): Complete the entries in the memory access table shown on the previous page.