Name: _________________________________

Problem 1 (30 points): ______
Problem 2 (10 points): ______
Problem 3 (10 points): ______
Problem 4 (15 points): ______
Problem 5 (20 points): ______
Problem 6 (20 points): ______
Problem 7 (20 points): ______
Total (125 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!
Problem 1 (30 points)

Part a (4 points): A RAM cell stores one bit of information. That storage is accomplished

in an SRAM cell by

and in a DRAM cell by

Part b (5 points): Amdahl’s Law says that the maximum speedup I can get on a multiprocessor for an algorithm that is 95% vectorizable is

To achieve that speedup, I need

processors.

Part c (5 points): Recall that asynchronous I/O with central arbitration requires at least the following bus control signals:

\[ BR_i, BG_i \text{ (for all priority levels } i) \], \[ BBSY, SACK, MSYN, SSYN \]

What signal does the PAU receive, indicating that a device controller has accepted being bus master for the next bus cycle?

What does the PAU do as a result of receiving this signal?
Part d (6 points): Recall that Omega networks are better than buses with respect to and better than full crossbars with respect to.

Suppose $n_5$ wants to talk to $a_3$ and $n_7$ wants to talk to $a_4$ in the following Omega network in the same cycle. Can they do it? Show how by tracing the paths on the Omega network below, or explain why not in ten words or less in the box provided.
Problem 1 continued:

**Part e (5 points):** A Directory based Cache Coherency scheme has five processors, each with a private cache. Cache line X has directory information as follows: 100111. Processor 2 wants to read a value in this cache line, and takes a cache miss.

What can you tell me about the Cache Coherency logic? Please be specific.

**Part f (5 points):** Suppose a new floating point data type is invented, which follows all the insights of the IEEE Standard. The only difference is it has 11 fraction bits and 8 exponent bits. Show below the representation for the largest positive subnormal number, using this format.
Problem 2 (10 points):

We discussed multiplication as a sequence of shifts/adds in EE 316. We extended it to Booth’s algorithm (2 bits per iteration) in class. Now, on the exam, we wish to extend it to 3 bits per iteration.

A truth table to describe the combinational logic needed would consist of 4 inputs and 4 outputs. The 4 inputs are the 3 bits of the multiplier and the input borrow bit. The 4 outputs are a left shift of the multiplicand by $k_1$ bits, an Add or Subtract signal, a subsequent left shift of the multiplicand by $k_2$ bits, and the output borrow bit.

For the case where the 3 bits of the multiplier are 110 and the input borrow bit is 0, specify the 4 outputs listed below.

1. Left shift the multiplicand _____ bits to the left.

2. Perform the operation of [Add / Subtract (circle one)] in the ALU.

3. Left shift the multiplicand _____ bits to the left.

4. Set the Borrow bit to 0 / 1 (circle one).
Problem 3 (10 points)

Consider the following data flow graph:

Note that node $\text{Odd?}$ outputs True if its input is odd and False if its input is even.

What does the data flow graph above do?

Answer:
Problem 4 (15 points):

An array A consists of 1536 one-byte elements, stored in 1536 consecutive locations of memory, starting at address x4000. (Hint : 1536 = 1024 + 512)

A programmer wishes to add 10 to every 8th element, but does so by the very curious method of adding 1 to every 8th element, and then iterating that process ten times. The C program segment looks like this:

```c
for ( i = 0; i < 10; i++ )
{
    for ( index = 0; index < 1536; index = index + 8 )
}
```

Assume that the compiler allocates registers for the loop variables i and index. The processor has a 1 KB, 2-way set associative cache. You may assume that the cache is initially empty. You may leave the answers for the following questions as fractions. Please show your work.

Part a (5 points): For a cache line size of 16 bytes, calculate the miss ratio if the perfect LRU replacement policy is used.

Part b (10 points): Suppose the cache line is 4 bytes. All other parameters remain the same. Circle the replacement policy which would give the least number of misses. Also, calculate the miss ratio for your chosen replacement policy.

FIFO (First In First Out)  LIFO (Last In First Out)  LRU

Miss ratio:
Problem 5 (20 points)

An in-order vector processor with 11 cycle memory latency, 16-way interleaved memory and 8 vector registers of length 64 is used to execute the vector code resulting from compiling the following high-level language program segment:

```c
for ( i = 0; i < 40; i = i + 2 )
{
    C[i] = A[i] + B[i];
    C[i+1] = A[i+1];
}
```

Part a (5 points): Shown below is the vector code produce by the compiler with some information missing. Your job is to fill in the missing information.

```
MOVI VLEN, #
MOVI VSTR, #
VLD V0, A
VLD V1, B
VADD V2, V0, V1 
VLD

```

You have available the following vector instructions:

- `MOVI VLEN, #n ; (1 cycle)`
- `MOVI VSTR, #n ; (1 cycle)`
- `VADD Vi, Vj, Vk ; Vi is dest, Vj, Vk are sources (4 staged pipelined adder)`
- `VLD Vi, A + k ; Vi gets loaded with contents of memory, starting at A + k`
- `VST Vi, A + k ; Contents of Vi gets stored in memory, starting at A + k`
Problem 5 continued:

Part b (7 points): How many cycles does it take the vector code to execute? Assume that the memory has 2 load ports and 1 store port and the vector processor supports vector chaining. Please show your work.

Part c (8 points): A simple reordering of the instructions will result in improved execution time. Show the new order below which results in minimal execution time. How many cycles does the new order take? Please show your work.
Problem 6 (20 points)

Consider the following two-level virtual memory system for the LC-3b. Recall that the user page table is located in the system virtual space:

- Virtual Address Space: 64KB
- User Space Range: x0000 to x7FFF
- System Space Range: x8000 to xFFFF
- Number of Page Frames: 32
- Page Size: 256 bytes
- Page Table Entry Size: 2 bytes

The system does not include a Translation Lookaside Buffer and a cache. Each Page Table Entry (PTE) contains valid (V), Modified (M), Reference (R) bits and Page Frame Number (PFN). The PTE format is as follows:

```
<table>
<thead>
<tr>
<th>15</th>
<th>11</th>
<th>10</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>PFN</td>
</tr>
</tbody>
</table>
```

Part a (2 points): How many bits are needed for the address bus to the physical memory? Show the computation.

Answer: 20

Part b (18 points): The machine stopped at a breakpoint and the following state information was observed:

```
PC   SBR    UBR    R0    R1    R2    R3    R4    R5    R6    R7
x3010 x0000  xA000  x1234  x5000  x4000  x1233  x1299  xFEBC  xABCD
```

Note: SBR is the System Page Table Base Register and UBR is the User Page Table Base Register. Each points to the first entry of the corresponding page table.

After execution resumed, the machine issued the following successive seven physical memory requests, uninterrupted by any page faults, access control violations, or anything else. Note that each entry is incomplete. Your job: complete the seven entries.

<table>
<thead>
<tr>
<th>Access #</th>
<th>PA</th>
<th>Access</th>
<th>Data</th>
<th>Identity of Item Being Accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Read</td>
<td>x8C18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Read</td>
<td>x8410</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>x7E82</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>x8008</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>xABCD</td>
<td></td>
</tr>
</tbody>
</table>

Note: The last column should identify what is being read specifically. For example: "The instruction LDB R0,R1,#0," "The PTE for User Virtual Page 0," "PTE for System Virtual Page 0," etc.
### LC-3b Instruction Encodings

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD*</td>
<td>0001 DR SR1 0 00 SR2</td>
<td>Add register to register</td>
</tr>
<tr>
<td>ADD*</td>
<td>0001 DR SR1 1 imm5</td>
<td>Add register to immediate</td>
</tr>
<tr>
<td>AND*</td>
<td>0101 DR SR1 0 00 SR2</td>
<td>AND register to register</td>
</tr>
<tr>
<td>AND*</td>
<td>0101 DR SR1 1 imm5</td>
<td>AND register to immediate</td>
</tr>
<tr>
<td>BR</td>
<td>0000 n z p PCoffset9</td>
<td>Branch to offset</td>
</tr>
<tr>
<td>JMP</td>
<td>1100 00 BaseR 000000</td>
<td>Jump to label</td>
</tr>
<tr>
<td>JSR</td>
<td>0100 1 PCoffset11</td>
<td>Jump to subroutine</td>
</tr>
<tr>
<td>JSRR</td>
<td>0100 0 00 BaseR 000000</td>
<td>Jump to subroutine with register as target</td>
</tr>
<tr>
<td>LDB*</td>
<td>0010 DR BaseR offset6</td>
<td>Load byte from memory</td>
</tr>
<tr>
<td>LDW*</td>
<td>0110 DR BaseR offset6</td>
<td>Load word from memory</td>
</tr>
<tr>
<td>LEA*</td>
<td>1110 DR PCoffset9</td>
<td>Load effective address</td>
</tr>
<tr>
<td>NOT*</td>
<td>1001 DR SR 1 1111</td>
<td>Not register</td>
</tr>
<tr>
<td>RET</td>
<td>1100 00 111 000000</td>
<td>Return from subroutine</td>
</tr>
<tr>
<td>RTI</td>
<td>1000 000000000000</td>
<td>Return from interrupt</td>
</tr>
<tr>
<td>LSHF*</td>
<td>1101 DR SR 0 0 amount4</td>
<td>Logical shift left with fixed amount</td>
</tr>
<tr>
<td>RSHFL*</td>
<td>1101 DR SR 0 1 amount4</td>
<td>Logical rotate shift left with fixed amount</td>
</tr>
<tr>
<td>RSHFA*</td>
<td>1101 DR SR 1 1 amount4</td>
<td>Logical rotate shift left with fixed amount</td>
</tr>
<tr>
<td>STB</td>
<td>0011 SR BaseR offset6</td>
<td>Store byte to memory</td>
</tr>
<tr>
<td>STW</td>
<td>0111 SR BaseR offset6</td>
<td>Store word to memory</td>
</tr>
<tr>
<td>TRAP</td>
<td>1111 0000 trapvect8</td>
<td>Trap to exception</td>
</tr>
<tr>
<td>XOR*</td>
<td>1001 DR SR1 0 00 SR2</td>
<td>XOR register to register</td>
</tr>
<tr>
<td>XOR*</td>
<td>1001 DR SR 1 imm5</td>
<td>XOR register to immediate</td>
</tr>
</tbody>
</table>

Figure 1: LC-3b Instruction Encodings
Problem 7 (20 points)

We are designing a 42-bit floating point format similar to the IEEE standard discussed in class. We have decided on the sizes of the exponent and fraction fields as shown below:

<table>
<thead>
<tr>
<th>S</th>
<th>Exponent</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>31</td>
</tr>
</tbody>
</table>

However, we have not yet decided on the bias of the exponent. Please use $N$ for the bias in your answers to parts a, b, and c below.

Part a (5 points): What is the smallest positive number represented by this format ($Min$)?

$$Min = \boxed{0} \times 2^{12}$$

Part b (5 points): What is the largest positive finite number represented by this format ($Max$)?

$$Max = \boxed{1} \times 2^{12}$$
Problem 7 continued:

Part c (5 points): In class, we defined the range of real values rounded to $Min$ if the round to nearest rounding mode is used. The shaded range in the figure below represents the real values rounded to $Max$. Any number to the right of that range is rounded to positive infinity.

![Diagram showing real values rounded to Max]

We want to choose the bias ($N$) so that for any positive finite number $x$ represented exactly by our floating point format, its reciprocal ($1/x$) can also be represented (after rounding to nearest) as a positive finite number.

That means $\frac{1}{Min}$ must be less than $\times 2$

and, $\frac{1}{Max}$ must be greater than $\times 2$

Part d (5 points): Compute the bias that will allow the results in part c to hold. NOTE: In part c you have created two inequalities in $N$. Solve those 2 inequalities.

Answer: [box]