1. True/False
   Directions: Circle either True or False.

   a. (True/False) According to Flynn’s taxonomy, a multicore CPU is classified as MISD.

   b. For each claim about Warehouse Scale Computing, indicate whether it is true or false.
      i. (True/False) An energy-proportional server has energy usage increase quadratically with load.
      ii. (True/False) Improving the power efficiency of an individual compute node will also improve the Power Utilization Effectiveness (PUE) of a datacenter.
      iii. (True/False) The greatest part of monthly expenses for a datacenter are amortized Capital Expenditures (CAPEX).

   c. For each claim about the MapReduce programming model, indicate whether it is true or false.
      i. (True/False) MapReduce programs running on a single core are usually faster than a simple serial implementation
      ii. (True/False) MapReduce works well on clusters with hundreds or thousands of machines
      iii. (True/False) MapReduce is the only framework used for writing large distributed programs.
      iv. (True/False) MapReduce can sometimes give the wrong answer if a worker crashes.
      v. (True/False) A single Map task will usually have its map() method called many times.

   d. For each claim about Hadoop MapReduce, indicate whether it is true or false.
      i. (True/False) If you don’t define map() or reduce(), the framework will use the identity function.
      ii. (True/False) Each reducer’s output is sorted after the last call to reduce()
      iii. (True/False) The values associated with a given key are sorted.
      iv. (True/False) The Reducers can start copying and sorting map outputs before all maps have finished.
      v. (True/False) The map() function will be called exactly once for each input key-value pair

   - Minus 1 for each wrong answer
   (minimum of 0)
2. Number Systems and Floating Point
Creatures on Mars have three “fingers” on three “hands”, and consequently have a base 9 number system (basically nine weird symbols that translate into 0, 1, 2, 3, 4, 5, 6, 7, 8).

a. What is the number $188_{nine}$ in base 10?

b. What is the number $2A5_{hex}$ in Martian base 9?

\[ 832_{nine} \]

- partial credit if I saw stupid mistake in shown work

- 2 points each


c. Convert the following single-precision floating point numbers back to a decimal representation. Match each bit pattern with a single member of the given bank of decimal numbers by writing the letter of the matching real number into the blank.

<table>
<thead>
<tr>
<th>i.</th>
<th>10111010 010100000000000000000000</th>
<th>c. [ 1.25 \times 2^{24} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ii.</td>
<td>0 11111111 000000000000000000000000</td>
<td>b. infinity</td>
</tr>
<tr>
<td>iii.</td>
<td>1 11111111 010000001100110000101</td>
<td>d. NaN</td>
</tr>
<tr>
<td>iv.</td>
<td>0 00011000 010000000000000000000000</td>
<td>f. [-\infty ]</td>
</tr>
</tbody>
</table>

- 1 pt. each

d. Given the approximately 4 billion binary patterns in a 32-bit word, a single-precision floating-point word can exactly represent

- approximately $2^{30}$ (1 billion) real numbers between $-\infty$ (infinity) and -1,

- approximately $2^{30}$ (1 billion) real numbers between -1 and 0,

- approximately $2^{30}$ (1 billion) real numbers between 0 and +1, and

- approximately $2^{30}$ (1 billion) real numbers between +1 and $+\infty$ (infinity).

You can use English words or numbers in scientific notation in your answers.

- 2 pts each

- answers had to be $\sim 2^{30}$, not $2^{29}$ or $2^{31}$ for credit
3. Caches

a. The Average Memory Access Time equation (AMAT) has three components: hit time, miss rate, and miss penalty. For each of the following cache optimizations, indicate which component of the AMAT equation may be improved. Circle one.

- Using a second-level cache
- Using smaller blocks
- Using larger blocks
- Using a smaller first-level cache
- Using a larger first-level cache

b. Given a direct-mapped cache, initially empty, and the following memory access pattern (all byte addresses and 32-bit word accesses, 32-bit addresses)

| 8 | 0 | 4 | 32 | 36 | 8 | 0 | 4 | 16 | 0 |

What is the hit rate, miss rate, and what blocks are in the cache after these accesses if:

i. the cache has 8 32-bit blocks?

hit rate: 0.2 miss rate: 0.8

blocks at end (write the appropriate full byte addresses (NOT the tag) in the appropriate blocks, or "EMPTY" if the block is empty):

| 0 32 0 |
| 4 36 4 |
| 8 |
| EMPTY |
| 16 |
| EMPTY |

- 1: minor mistake, easily seen
- 2: fairly major mistake
- 3: we had no idea what was happening
The memory access pattern is repeated here for your convenience:
8 0 4 32 36 8 0 4 16 0

ii. the cache has 4 32-bit blocks?

hit rate: 0.1 miss rate: 0.9

blocks at end (write the appropriate full byte addresses (NOT the tag) in the appropriate blocks, or "EMPTY" if the block is empty):

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMPTY</td>
</tr>
</tbody>
</table>

same as before

c. Consider a write-allocate, write-back, direct-mapped cache with 16 byte blocks and 64*2^{10} bytes of data bits. Assume a byte-addressed machine with 32-bit addresses.

i. Partition the following address and label each field with its name and size in bits.

```
31
| TAG | 16 | INDEX | 12 | OFFSET | 4 3 |
```

ii. Given the address DEADBEEF_{hex}, what is the value of the index, offset, and tag? (Write your answers in hexadecimal.)

index = \text{0xBEE}
offset = \text{0xF}
tag = \text{0xDEAD}

iii. How many cache management bits are there for each block? List them.

\text{TAG \hspace{0.5cm} VALID \hspace{0.5cm} DIRTY}

2 points -1 per mistake

iv. What is the total number of bits (data AND cache management) that comprise the cache?

\[ 2^{12} \text{ blocks} \times 18 \text{ management bits per block} + 2^{12} (128 + 18) \text{ bits} = 584 \text{ Kbits} \]
4. AMAT
Suppose a MIPS program executes on a machine with a single data cache and a single instruction cache, and
  • 20% of the executed instructions are loads or stores;
  • the data cache hit rate is 95%;
  • the instruction cache hit rate is 99.9%;
  • the instruction and data cache miss penalty is 10 cycles;
  • the instruction and data cache hit time is 1 cycle; (ideal $CPI = 1$, so overlapped).

a. How many memory references are there per executed instruction?

\[ 0.2 \]

1 POINT FOR 0.2

b. How many data cache misses are there per executed instruction?

\[ 20\% \times 5\% = 1\% \]

\[ 0.02 \times 0.05 = 0.01 \]

c. How many instruction cache misses are there per executed instruction?

\[ 0.1\% \]

\[ 0.001 \]

d. Assume that if there were no cache misses, the CPI would be 1. What is the CPI of the program given the cache miss rates above?

\[ 1 + 10\% \times 10 + 0.1\% \times 10 = 1.11 \]

\[ 1 + 0.01 \times 10 + 0.001 \times 10 = 1.11 \]

e. What is the average memory access time of the program?

HIT TIME + MISS RATE \times MISS PENALTY

\[ 1 + 0.011 \times 10 = 1.11 \text{ clock cycles} \]

1 POINT IF PROPERLY CALCULATE FOR INSTRUCTION CACHE MISSES OR DATA CACHE MISSES
5. Cache-optimized data structures
In a physically-based animation you have the following array of structures:
struct vertex {
    float x, y, z; /* position */
    float vx, vy, vz; /* velocity */
    float fx, fy, fz; /* force */
    float nx, ny, nz; /* normal */
    float tx, ty; /* texture */
} vertices[10000];
You want to reset all forces (fx, fy, fz) to zero. Assume a 64KB cache with 64-byte blocks,
write allocate, write back, direct mapped.
a. How many bytes do you need to set in the array in total?
   \[ 3 \times 4 \times 10000 = 120,000 \]
   \[ \text{MINOR MISCALCULATION}: -1 \]
   \[ 240,000 : 2 \text{ POINTS} \]
b. How many bytes will be read from memory when doing so?
   \[ 14 \times 4 \times 10000 = 560,000 \]
   \[ 640,000 : 2 \text{ POINTS} \]
   \[ \text{MINOR MISCALCULATION}: -1 \]
c. Suggest a change in the data structure that minimizes the memory traffic.

"Structure of Arrays" instead of
"Array of Structures":

struct vertices {
    float x [10000], y [10000], z [10000];
};

4 POINTS EACH.
6. MIPS, C, and Pointers
Given the following C definition for a node in a binary tree, and the following C source code that prints out the nodes in a tree post-order,

```c
struct node {
    struct node *left, *right;
    const char* value;
};
void print_postorder(const struct node* root) {
    if(root != NULL) {
        print_postorder(root->left);
        print_postorder(root->right);
        printf("The node at address %p has value %s\n", root, root->value);
    }
}
```

fill in the blanks in the following MIPS assembly code that implements the `print_postorder` function.

```assembly
.data
thestring: .asciiz "The node at address %p has value %s\n"
.text
print_postorder:
addiu $sp, $sp, -8
sw $ra, 0($sp)
sw $s0, 4($sp)
beq $a0, $0, out
move $s0, $s0
lw $a0, 0($a0)
jal print_postorder
lw $a0, 4($s0)
jal print_postorder
out:
lw $ra, 0($sp)
lw $s0, 4($sp)
addiu $sp, $sp, 8
jr $ra
```

1 POINT PER BLANK.