Problem 1: Short Answer Questions [23 points]

1 points for each answer. (E.g., questions with two parts are 2 points.) Please answer each question with a short answer of no more than two sentences.

1. What is the range of values that can be represented with a 6-bit, unsigned, fixed-point number?
   A: 111111. = 127 to .000001 = 1/64

2. Why does MIPS have multiple instruction formats?
   A: To provide different amounts of immediate (constant) information for register-register operations, register-immediate/address, and address (jump) operations.

3. Why is it important to have a standard for register usage for procedure calls?
   A: To enable code (libraries) to work together without knowing exactly what register the other code may use.

4. Serial multiplication of the two unsigned numbers 1100111 and 1011110 requires how many additions?
   A: 5. Each 0 skips an addition.

5. How are a) decoders and b) MUXes used in memory arrays?
   A: Decoders are used to select a row based on the input address.
   B: MUXes are used to select the correct columns based on the input address.

6. A program consists of 20% floating-point operations. A new processor speeds up floating point operations by 10x. a) How much faster does the program run, and b) what is the best you could achieve with infinite floating-point acceleration?
   A: The 20% goes 10x faster, so it takes 2%, which means you save 18% of the execution. So you are now running in 82% of the time or 1.2x faster.
   B: The best you could do would be to bring the 20% down to 0%, which would be 80% of the execution time or 1.25x faster.

7. A single-cycle processor has three types of instructions: A takes 1ns, B takes 2ns, and C takes 10ns. a) A program with an instruction mix of 40% A and 60% B will run at what clock speed? b) An ideal multi-cycle processor (as discussed in class) would run at what clock speed?
   A: 100MHz. The clock speed is limited by the slowest instruction regardless of what instructions the application uses.
   B: 1000MHz. The clock speed is ideally limited only by the fastest instruction.

8. What is the key to getting full performance from a pipelined processor?
   A: Keep the pipeline full.

9. What is the implication of having a single memory in a a) 5-stage pipeline? b) How is this typically addressed?
   A: The address and data accesses share it causing hazards on every memory access instruction.
   B: Using separate instruction and data caches to allow concurrent accesses.
10. Why is it beneficial to have forwarding busses from both MEM and WB in the 5-stage MIPS pipeline?
   A: Enables instructions that are 1 and 2 cycles apart to avoid data hazards due to results not being available.

11. Where do computers use a) SRAM and b) DRAM, and why?
   A: SRAM is used for small performance critical resources like caches and register files because it is much faster than DRAM.

   B: DRAM is used for large storages such as main memory because it is much cheaper than SRAM.

12. A write-through cache holds 1MB of data and has one valid bit per line, a line size of 64 bytes, and is used on a machine with a byte-addressable physical memory address size of 32 bits. What percentage of the cache’s storage is used to hold data?
   A: Each cache line of 64 bytes is 6 bits of address space for a byte-addressable machine. We need 32-6=26 bits of tag for each line, plus 1 bit for valid (no dirty bits on a write-through cache) = 27 bits of tag. So for each line we have 27 bits not holding data and 64*8 bits holding data = (64*8)/(27+64*8) = 95%

13. A processor has a CPI of 1.0, every other instruction loads data, and accessing memory takes 100 cycles. A 64kB unified cache has an access time of 1 cycle and a miss rate of 10%. How long (in cycles) does it take to execute 100 instructions a) with and b) without the cache?
   A: 1.5 memory accesses per instruction (1 for instruction 0.5 for data). 10% take 100 cycles, 90% take 1 cycle. So 100 instructions: 100 * (1 + 1.5 * (0.1*100 + .9*1)) = 1735

   B: 100*(1+1.5*100) = 15100

14. How does virtual memory make it a) easier and b) safer to run multiple programs at once?
   A: Easier: each application sees a full address space so it can allocate data independently of where it is placed in physical memory.

   B: Safer: pages are used to protect memory so applications can’t access each other’s data.

15. The memory hierarchy provides two key illusions to the programmer. What are they and why are they important?
   A: Size: virtual memory makes it look like you have huge amounts of memory, which simplifies programming by removing memory management from the user.

   B: Speed: caches make it look like memory is much faster than it is, which simplifies programming by removing data movement optimizations from the programmer.
Problem 2: False/True [15 points]
0 points for no answer, -1 point for an incorrect answer, +1 point for a correct answer. Circle either false or true or neither.

1. The bits from the I-format immediate field are directly added to the base register (PC or RS).
   a. True False — False – They are sign extended first

2. CISC processors require more instructions to do the same work than RISC processors.
   a. True False — False – CISC instructions are “complex” which means they can do more work per instruction, so such processors require fewer instructions

3. Modern x86 machines are RISC processors internally.
   a. True False — True – Modern x86 machines use CISC instructions but convert them to RISC-like instructions when processing them

4. A machine that executes 24 MIPS is faster than a 1GHz machine with a CPI of 50.
   a. True False — True. A CPI of 50 at 1GHz is 1000/50 = 20MIPS.

5. Karnaugh map groups may include both 1s and 0s to simplify logic.
   a. True False — False – Groups may only include 1s or 0s, but may include Xs to simplify the logic.

6. A single-cycle datapath needs only one regular memory.
   a. True False — False. A single-cycle datapath needs to access both the instruction and data in the same cycle, and therefore needs two memories.

7. A 100MHz processor is pipelined to 10 stages with 1ns latches. For optimal code, the speedup will be 10x.
   a. True False — False. The new processor will not run at 100MHz because the latch takes 1ns and will slow down the clock.

8. If a processor shares an ALU for both addition and multiplication, this causes a control hazard that prevents multiply and addition instructions from executing simultaneously.
   a. True False — False. This is a structural hazard.

9. The performance of an I/O device is determined by its throughput.
   a. True False — False. Latency is also very important.

10. Having a larger cache directly improves spatial locality.
    a. True False — False. A larger cache improves temporal locality. A larger block (or cache line) will improve spatial locality. A larger cache may indirectly improve spatial locality via temporal locality.

11. Set-associative caches can be as good as fully-associative caches.
    a. True False — True. Set-associative caches may be just as good as fully-associative caches if the associativity is high enough for the workload.

12. A machine with a higher CPU clock frequency benefits more from a cache than a machine with a lower CPU clock frequency.
    a. True False — True. Higher clock frequency means more cycles spent waiting on memory accesses, so a cache is more beneficial. (Memory speed does not scale with processor speed.)

13. Virtual memory allows a program to use 8GB of memory on a 32-bit processor with only 1GB of physical RAM.
    a. True False — False. A 32-bit processor can only address up to 2^32 (4G) bytes of memory. Virtual memory allows each program to use up to that limit regardless of the amount of physical RAM, but does not allow any individual program to exceed that limit.

14. Page tables are stored exclusively in dedicated cache hardware to speed up memory accesses.
    a. True False — False. The TLB is a cache that stores the most recently used pages. The rest are stored in main memory.
15. **Virtual memory prevents applications from sharing data for security.**
   a. True  False — **False**. Applications can share data through virtual memory by having their page table entries map to the same physical page.
Problem 3: Caches [23 points]

Three, 4-entry, LRU caches are shown below, with the most recently used entry at the top of each set. They are fully-associative (FA), direct-mapped (DM), and 2-way set-associative (SA). Each cache has a block/line size of 1 byte/address. The DM and SA caches use the modulo (LSB-based) indexing function discussed in class. Show how the contents of the caches change with the access patterns below by writing the memory address in each location of the cache after each access at each time step and circling whether the access is a hit (H) or miss (M).

A): 0, 8, 4, 8, 4, 0 [6 points]

<table>
<thead>
<tr>
<th>Cycle 0—0</th>
<th>Cycle 1—8</th>
<th>Cycle 2—4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA DM SA</td>
<td>FA DM SA</td>
<td>FA DM SA</td>
</tr>
<tr>
<td>0 0 0</td>
<td>8 8 8</td>
<td>4 4 4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M M M</td>
<td>M M M</td>
<td>M M M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle 3—8</th>
<th>Cycle 4—4</th>
<th>Cycle 5—0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA DM SA</td>
<td>FA DM SA</td>
<td>FA DM SA</td>
</tr>
<tr>
<td>8 8 8</td>
<td>4 4 4</td>
<td>0 0 0</td>
</tr>
<tr>
<td>4 4</td>
<td>8 8</td>
<td>4 4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>H M H</td>
<td>H M H</td>
<td>H M M</td>
</tr>
</tbody>
</table>

B) Hit ratios: (ratio of hits to total accesses) [5 points]

FA: 50 % DM: 0 % SA: 33 %

Briefly explain the differences (or lack thereof) in miss ratios:
DM: all accesses map to the same line so they always miss; SA is more flexible because it can hold two conflicting addresses; FA can hold all of them.

C): 0, 1, 2, 1, 2, 0 [6 points]

<table>
<thead>
<tr>
<th>Cycle 0—0</th>
<th>Cycle 1—1</th>
<th>Cycle 2—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA DM SA</td>
<td>FA DM SA</td>
<td>FA DM SA</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1 0 0</td>
<td>2 0 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M M M</td>
<td>M M M</td>
<td>M M M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle 3—1</th>
<th>Cycle 4—2</th>
<th>Cycle 5—0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA DM SA</td>
<td>FA DM SA</td>
<td>FA DM SA</td>
</tr>
<tr>
<td>1 0 2</td>
<td>2 0 2</td>
<td>0 0 0</td>
</tr>
<tr>
<td>2 1 1</td>
<td>1 1 0</td>
<td>2 1 2</td>
</tr>
<tr>
<td>0 2 1</td>
<td>0 2 1</td>
<td>1 2 1</td>
</tr>
<tr>
<td>H H H</td>
<td>H H H</td>
<td>H H H</td>
</tr>
</tbody>
</table>

D) Hit ratios: (ratio of hits to total accesses) [5 points]

FA: 50 % DM: 50 % SA: 50 %

Briefly explain the differences (or lack thereof) in miss ratios:
The addresses map nicely for all three caches so they behave the same.

E) Explain why B and D show (or do not show) different hit ratios for the caches: [1 point]
The address streams are different so the caches behave differently.
Problem 4: Virtual Memory and Caches [8 points]

1. Virtually-indexed, Virtually-tagged cache:
   a. Why is this approach appealing?
   Speed: don’t have to wait for a TLB lookup to do a cache lookup.
   b. What’s the problem?
   Security: Can’t tell if a process should have access based on the virtual address since all programs share the same virtual address space.
   c. How can we solve this problem?
   Flush the cache on context switches or append process ID bits to each cache line.

2. Virtually-indexed, Physically-tagged cache:
   a. Why do you need the TLB lookup?
   To verify the tag on the cache line is for the right process
   b. If you do a parallel cache and TLB lookup, what are the limitations this imposes?
   Need to make sure that the bits used for the cache index do not contain any of the page number bits. Limits the size of the cache.

3. Physically-indexed, Physically-tagged cache:
   a. What performance problem does this cache impose?
   Have to wait for a TLB lookup before indexing the cache (slow).
   b. What extra information is needed to determine if a process is allowed to access the data?
   None. Physical addresses are unique so they identify the process.

4. Physically-indexed, Virtually-tagged cache:
   a. List two problems with such a design:
   Have to wait for a TLB lookup to index the cache (slow) and no way to determine if a line belongs to a process from the virtual tag.