Operating Systems/Computer Architecture Candidacy Exam Spring 2023

Student's Name:
Student's PSU email:

Instructions: This exam has two sections: operating systems and computer architecture. Each section has 100 points worth of questions. You have 3 hours to answer 50 points worth of questions from each section. Among 100 points worth of questions for each section, **explicitly <u>select *only* 50 points worth of questions</u> by <u>circling their question number</u>**. **Only those which are selected will be graded**. Good luck!

Interrupts, Traps, and System Calls (25pts)

1. (5 pts) How does the CPU know the addresses of the operating system's interrupt and trap handlers?

2. (5 pts) What is a system call? How does it differ from a function call?

3. (5 pts) What is a context switch? List one cause of a context switch.

4. (5 pts) List two events that generate interrupts.

5. (5 pts) List two events that generate traps.

Virtual Memory (25 pts)

1. (5 pts) What is the purpose of a page table?

2. (5 pts) True or False: The operating system installs a page table upon boot and that single page table is shared by all processes loaded into memory.

3. (5 pts) Why are page tables multi-level? Why not have one flat page table mapping the whole virtual address space?

4. (5 pts) What is TLB? What purpose does it serve?

5. (5 pts) Describe one pathological case where LRU page replacement policy is not better than choosing a random page to evict.

Concurrency (25 pts)

1. (5 pts) True or False: The thread that tries to acquire a spinlock that is already held by another thread busy-waits until the lock becomes available.

2. (5 pts) What is a condition variable? What problem does it solve?

3. (5 pts) The following code tries to ensure that thread_first runs before thread_second. Assume the mutex m and the condition variable c are defined globally, and fill in the body of thread_first.

```
void thread_second() {
     pthread_mutex_lock(&m);
     while (done == 0)
          pthread_cond_wait(&c, &m);
     pthread_mutex_unlock(&m);
}


void thread_first() {




}
```

4. (5 pts) Explain why the following code does not always print out 2000000. (Illustrate at least one case with a scheduling timeline.)

```
int i = 0;
void* run(void* _) {
     for (int j = 0; j < 1000000; j++) i++;
}

void main() {
     pthread_t t1, t2;
     pthread_create(&t1, NULL, run, NULL);
     pthread_create(&t2, NULL, run, NULL);
     pthread_join(t1, NULL);
     pthread_join(t2, NULL);
     printf("%d\n", i);
}
```

5. (5 pts) What is a readers-writer lock? In what scenario is it useful?

Storage (25 pts)

1. (5 pts) True or False: RAID-5 can withstand two disk failures.

2. (5 pts) What is a journaling file system?

3. (5 pts) Most file systems provide metadata consistency by default. What is meant by metadata consistency?

4. (5 pts) The ext4 file system can provide data consistency in addition to metadata consistency, but data consistency is not enabled by default. Why do you think this is?

5. (5 pts) You boot your computer and open a large file in your editor and it takes a few seconds to load. You close your editor, go for lunch, and after you come back you reopen the same file and it loads almost instantly. Why?

Datapaths and Dependencies (30 pts)

1. (5 pts) Read the below MIPS instructions. Find all the RAW, WAW, and WAR dependencies. Indicate them with an arrow and label them clearly whether they are RAW, WAW, or WAR.

```
add $1, $2, $3
lw $2, 0($1)
add $2, $2, $3
sw $2, 0($1)
lw $4, 0($2)
add $4, $2, $3
```

   Hint: The instruction formats for MIPS are as the following:
   - add $rd, $rs, $rt: REG[$rd] ← REG[$rs] + REG[$rt]
   - lw $rt, imm($rs): REG[$rt] ← MEM[REG[$rs] + imm]
   - sw $rt, imm($rs): MEM[REG[$rs] + imm] ← REG[$rt]

2. (5 pts) Between RAW, WAW, and WAR, which can be resolved with register renaming? Explain your answer briefly.

3. (5 pts) Explain who does the register renaming in a (1) VLIW processor, and a (2) dynamic superscalar (OoO) processor, respectively. Explain the pros and cons for each approach.

4. (10 pts) Consider a 5-stage MIPS pipeline of Fetch (F), Decode (D), Execute (E), Memory (M), and Writeback (W). In the table below, for each cycle, indicate the cycle an instruction computes a stage with a capital letter of {F, D, E, M, W}. Indicate stalls with a lowercase letter of {f, d, e, m, w}. Assume all stages, including M, finish in 1 cycle and there are no exceptions. Assume **full forwarding support**.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add $1, $2, $3	F	D	E	M	W									
lw $2, 0($1)														
add $2, $2, $3														
sw $2, 0($1)														
lw $4, 0($2)														
add $4, $2, $3														

5. (5 pts) Explain what structural hazard, data hazard, and control hazard are, respectively. Provide an example for each hazard.

Caches and memory (30 pts)

1. (5 pts) Explain the difference between a write-through cache and a write-back cache. State at least one pro and con for each design.

2. (5 pts) Consider a cache with 32KB of data capacity. Assume the cache is 4-way set associative with each cache block being 4 words. How many bits are needed for byte offset, block offset, index, and tag? Assume a 32-bit address space.

3. (5 pts) Assume the above cache has 1 bit for valid bit and 1 bit for dirty bit for each cache block. How many total bits are needed for the entire cache to be implemented (valid bit + dirty bit + tag + cache blocks)? You can leave you answer in a form of, e.g., 4 * (1 + 20 + 256) Kb.

4. (5 pts) Explain what compulsory miss, capacity miss, and conflict miss are in cache, respectively. Explain how each miss can be eased with modifications to hardware.

5. (5 pts) Briefly explain what temporal locality and spatial locality each means, and how cache can improve each locality. Provide a simple example code for each answer.

6. (5 pts) Many CPUs have multiple levels of caches instead of one giant cache. Explain why having multiple levels of caches can improve performance. If you decide to design a two-layer cache structure (L1 and L2), explain how the two designs should differ.

Performance Analysis (25 pts)

1. (5 pts) Machine A has a clock cycle time (CCT) of 200ps. Instructions take the following number of cycles to finish: Branch (3 cycle), R-type (4 cycles), Load (5 cycles), Store (4 cycles). Assume the machine is executing 10,000 instructions in total, where 20% is Branch, 50% is R-type, 20% is Load, and 10% is Store. What is the cycle-per-instruction (CPI) of machine A? The machine is not pipelined.

2. (5 pts) What is the execution time of running 10,000 instructions on machine A?

3. (5 pts) Assume you changed the architecture of machine A and made R-type instructions to be 2 times faster than before. How much speedup do you get when you run the same program?

4. (5 pts) Machine B is a single-cycle processor of clock cycle time (CCT) of 1000ps, which always executes any instruction in a single cycle.What is the CPI of machine B?

5. (5 pts) If you compare machine B with machine A from Q1 (not after the change made in Q3), which machine is faster? Will your answer change if you run a different program? Answer both questions and justify your answer briefly.

Concurrency and Parallelism (15 pts)

1. (5 pts) Write a simple code that runs on two different threads, whose possible outcome can be different on a processor with sequential consistency (SC) and a processor with a more relaxed consistency model (e.g., TSO, PC). Briefly explain how and why the outcomes will be different.

2. (5 pts) MSI cache coherence protocol has M, S, and I state. Explain briefly the meaning of each state. Also, can two processors be in the following states: (M, M), (M, S), (S, S), (M, I), (S, I)? Indicate which combinations are disallowed and briefly explain why.

3. (5 pts) Briefly explain why instructions like load-linked, store-conditional, and atomic exchange are needed. When are they useful?

(end)