Spring 2022 Operating Systems/Architecture Qualifying Exam

Name: _____

Instructions: There are nine (9) questions with a total of 200 points. Answers must fit in the space provided. Blank pages provided at the rear of the exam can be used for scratch paper – these will not be graded. The exam is closed book, so no notes, calculators, phones, etc. There are 3 hours (180 minutes) total for the exam. Good luck!

Question	**Score**
Q1) Virtual Memory (30pts)	
Q2) Concurrency (30pts)	
Q3) Kernel (20pts)	
Q4) Storage (20pts)	
Q5) Potpourri (20pts)	
Q6) Performance Evaluation (20pts)	
Q7) Caches and Memory (20pts)	
Q8) Branches (20pts)	
Q9) Value Prediction (20pts)	
Total (200pts)	

Q1) Virtual Memory (30pts)

(6pts) Suppose you're using a computer with 16GB of main memory. Suppose 4GB is being used by a web browser, and most of the remaining memory (e.g., 11GB) is filled up by the page cache. What happens if you start another program that uses 4GB of memory? Pick **one** of the following choices.

- ○ The program runs out of memory and is killed
- ○ The program runs out of memory and starts returning NULL from malloc
- ○ The page cache is shrunk by evicting pages
- ○ The OS will swap some of the program or web browser's memory to the swap space
- ○ The OS will kill the web browser

(8pts) Pick **ALL** of the choices where the page fault handler is **always** triggered.

- ☐ Reading from a NULL pointer
- ☐ Writing to a NULL pointer
- ☐ Reading from malloc'd memory
- ☐ Writing to malloc'd memory
- ☐ Reading from free'd memory
- ☐ Writing to free'd memory
- ☐ Reading from swapped out memory
- ☐ Writing to swapped out memory
- ☐ Reading from a shared global variable
- ☐ Writing to a shared global variable
- ☐ Reading from a constant global variable
- ☐ Writing to a constant global variable
- ☐ Reading from a non-page aligned pointer
- ☐ Writing to a non-page aligned pointer
- ☐ Reading from Copy-On-Write (COW) memory
- ☐ Writing to Copy-On-Write (COW) memory

(6pts) Inter-Process Communication (IPC) is a mechanism for transferring data between processes on a computer. For example, network sockets can be used as an IPC mechanism for sending data between two processes on a computer. Suppose we have a root pointer to a balanced binary search tree in process A. **Explain** how we would transfer the tree to another process B assuming we have a send(p, n) function that can transfer n bytes starting from location p.

(10pts) Suppose we're using an **8-bit** system with 16-byte pages and a two-level page table. Assume the two-level split is such that each table contains the same number of entries. Suppose the following is a dump of **physical memory**:

00:	50	10	20	20	10	90	f0	50	e0	40	20	90	00	50	80	10
10:	90	00	c0	50	30	e0	30	00	f0	f0	60	20	10	d0	50	f0
20:	40	e0	40	f0	f0	00	f0	f0	a0	30	a0	f0	50	e0	40	f0
30:	50	00	f0	f0	90	e0	90	40	f0	10	d0	20	10	90	f0	50
40:	80	40	00	f0	40	90	e0	10	d0	90	30	00	f0	f0	90	e0
50:	60	40	f0	10	20	20	10	90	f0	30	b0	10	20	40	00	f0
60:	50	70	e0	50	f0	50	80	f0	60	20	90	d0	50	f0	80	40
70:	50	00	f0	d0	c0	f0	20	10	c0	f0	f0	60	60	30	50	40
80:	50	40	10	20	c0	50	f0	00	f0	f0	c0	90	20	30	f0	30
90:	70	50	f0	60	90	e0	90	00	60	20	50	50	00	00	70	c0
a0:	40	20	30	f0	20	e0	20	e0	50	00	10	40	40	30	30	50
b0:	60	00	20	10	40	f0	30	f0	40	50	00	f0	90	40	d0	f0
c0:	a0	50	20	50	70	90	30	40	50	20	40	d0	30	c0	f0	e0
d0:	60	40	10	20	c0	50	00	30	70	90	80	30	80	40	20	c0
e0:	00	60	10	30	40	f0	20	90	10	c0	70	90	60	f0	50	80
f0:	b0	d0	00	c0	50	00	30	40	40	90	e0	00	00	70	c0	90

Suppose we declare char* p = 0xba. What is *p, assuming the process's base page table register (e.g., cr3) contains 0x60?

*p = 0x_____

Q2) Concurrency (30pts)

(10pts) Consider the following multi-threaded C code:

```
void* remove() {
  pthread_mutex_lock(&mutex);
  if (queueEmpty(&q)) {
    pthread_cond_wait(&cv, &mutex);
  }
  data = queuePop(&q);
  pthread_mutex_unlock(&mutex);
  return data;
}
void add(void* data) {
  pthread_mutex_lock(&mutex);
  queuePush(&q, data);
  if (queueSize(&q) == 1) {
    pthread_cond_signal(&cv);
  }
  pthread_mutex_unlock(&mutex);
}
```

Describe and **fix** any bug(s) you see, if any.

(10pts) Consider the following multi-threaded C code using a reader-writer lock:

```
Object* obj = NULL;
Object* getObj() {
  pthread_rwlock_rdlock(&rwlock);
  if (obj == NULL) {
    pthread_rwlock_unlock(&rwlock);
    pthread_rwlock_wrlock(&rwlock);
    if (obj == NULL) {
      obj = malloc(sizeof(Object));
      initObj(obj);
    }
    pthread_rwlock_unlock();
  } else {
    pthread_rwlock_unlock();
  }
  return obj;
}
```

Describe and **fix** any bug(s) you see, if any.

(10pts) Consider the following multi-threaded C code:

```
int joinQueue(server_t* servers, int numServers, job_t* j) {
  int r1 = rand() % numServers;
  int r2 = (r1 + 1 + (rand() % (numServers-1)) % numServers;
  return joinBestQueue(&servers[r1], &servers[r2], j);
}
int joinBestQueue(server_t* s1, server_t* s2, job_t* j) {
  pthread_mutex_lock(&s1->mutex);
  pthread_mutex_lock(&s2->mutex);
  server_t* best = (s1->numJobs < s2->numJobs) ? s1 : s2;
  queuePush(best->queue, j);
  int numJobs = ++best->numJobs;
  pthread_mutex_unlock(&s1->mutex);
  pthread_mutex_unlock(&s2->mutex);
  return numJobs;
}
```

Describe and **fix** any bug(s) you see, if any.

Q3) Kernel (20pts)

(8pts) Pick **ALL** of the choices that occur when context switching from thread A to thread B in a different process.

- ☐ Thread A's registers are saved
- ☐ Thread A's registers are restored
- ☐ Thread B's registers are saved
- ☐ Thread B's registers are restored
- ☐ The user stacks of the threads are swapped in memory
- ☐ Thread B's stack is loaded into memory
- ☐ The base page table register is switched
- ☐ The TLB is flushed
- ☐ The kernel stack register is switched
- ☐ The stack register is switched

(6pts) When performing a syscall, the computer will switch from the user stack to the kernel stack. How does the computer know where the kernel stack is? Pick **one** of the following choices.

- ○ Determined based on thread id
- ○ Specified in kernel stack register
- ○ Always starts at a predefined location
- ○ Location is configured during bootup
- ○ Looked up in a kernel data structure

(6pts) **Explain** what is different when context switching between threads in the same process vs. threads in different processes.

Q4) Storage (20pts)

(6pts) What is the effect of **increasing** a hard drive's rotation speed for:

Seek latency (Pick **one** of the following choices)

- ○ Increased seek latency
- ○ Decreased seek latency
- ○ Little to no effect on seek latency

Rotation latency (Pick **one** of the following choices)

- ○ Increased rotation latency
- ○ Decreased rotation latency
- ○ Little to no effect on rotation latency

Transfer latency (Pick **one** of the following choices)

- ○ Increased transfer latency
- ○ Decreased transfer latency
- ○ Little to no effect on transfer latency

(4pts) Suppose it takes 10ms to delete a 1MB file. About how much time would it take to delete a 100MB file? Pick **one** of the following choices.

- ○ About 10ms
- ○ Between 10ms-1000ms
- ○ About 1000ms
- ○ Over 1000ms

(4pts) **List two (2)** fields contained within an inode.

1. _____
2. _____

(6pts) **Explain** what is filesystem journaling. **Give one (1) example** where filesystem journaling is beneficial.

Q5) Potpourri (20pts)

(5pts) Explain why we need 'pipeline registers' (also known as 'state registers') in a pipelined datapath.

(5pts) Define the following terms:

a) Branch history table (BHT)

b) VLIW

c) Load-use data hazard

d) Out-of-order execution

e) Reorder buffer (ROB)

(5pts) Explain how Tomasulo's algorithm works and what it tries to achieve.

(5pts) Explain, via an example, why loop unrolling is used.

Q6) Performance Evaluation (20pts)

(6pts) The following measurements have been made using an architecture simulator for a processor design that is projected to have a clock rate of 1 GHz. What is the design's CPI?

Instruction Class	CPI	Frequency of Occurrence
R-type	1	38%
Load	5	24%
Branch	4	22%
Store	2	16%

CPI = _____

(7pts) How does this compare to an alternate design that shaves 2 cycles off the branch latency, but also increasing the number of cycles load and store operations take by 1 cycle?

CPI = _____

(7pts) Assume that an optimized version of the first design has been implemented which increases the clock rate to 2 GHz but also doubles the CPI of each instruction class (e.g., load now takes 10 cycles). Which design is faster, the 1 GHz version or the 2 GHz version, and by how much? Justify your answer.

Q7) Caches and Memory (20pts)

Suppose you had a computer that, on average, exhibited the following properties on the programs that you run:

Instruction miss rate: 2.5%
Data miss rate: 5.0%
Percentage of memory load/store instructions: 36%
Miss penalty: 110 cycles

There is no penalty for a cache hit (i.e., the cache can supply the data as fast as the processor can consume it). Assume also that a cache-block is one word.

You want to upgrade the computer, and your budget will allow one of the following options:

i. Get a new processor that is twice as fast as your current computer. The new processor's cache is twice as fast too (so it can keep up with the processor)
ii. Get a new memory that is twice as fast.

Which is a better choice? Explain fully, showing a quantitative comparison between the two options (i and ii).

Q8) Branches (20pts)

(5pts) Explain how stalling a processor pipeline to handle control hazards is different from stalling a pipeline to handle data hazards.

(5pts) Explain two strategies to handle control hazards.

(5pts) Consider a branch that has the following outcome pattern (T for taken, N for not taken):

T, T, N, N, N, N, T, T, N, T, N, N, T, T, T

How many branches are predicted correctly with a static (0-bit) always-taken branch predictor for this branch outcome pattern.

correct predictions = _____

(5pts) Using the same sequence, how many branches are predicted correctly with a dynamic 1-bit predictor where the initial state is Taken (T) for this branch outcome pattern?

correct predictions = _____

Q9) Value Prediction (20pts)

Data dependences can prevent hardware from changing the execution order of instructions and from executing instructions in parallel. While anti- and output-dependences can be eliminated via renaming, flow-dependences are much more challenging (as they indicate actual data transfer, as opposed to simple name reuse).

Value prediction is a method to handle data dependences. In value prediction, hardware predicts the value to be returned by a load instruction before the instruction accesses the cache/memory. The execution continues with the predicted value. In the background, the cache/memory is accessed and, if the value read (i.e., actual/correct value) is different the one predicted, the pipeline is flushed and the execution re-starts from the instruction that comes after the load using the actual value.

One method of value prediction for an instruction is "last-time prediction." The idea is to predict the value to be produced by the load instruction as the value produced by the same instruction the last time the instruction was executed. If the instruction was never executed before, the predictor predicts the value to be 1. Value prediction "accuracy" of an instruction refers to the fraction of times the value of an instruction is correctly predicted out of all times the instruction is executed.

Assume the following piece of code, which has four load instructions in each loop iteration, loads to arrays x, y, z, t:

```
// initialize integer array x: x consists of all 1's
// initialize integer array y: y consists of alternating 2's and 3's
// initialize integer array z: z consists of random values
// initialize integer array t: t consists of 0, 1, 2, 3, 4, ..., 99
int c = 0;
for (int i = 0; i < 100; i++)
    c += (x[i] * y[i]) – (z[i] * t[i]);
```

(5pts) What is the value prediction accuracy of the aforementioned predictor for the four load instructions in the program?

Prediction accuracy for load of x[i] = _____
Prediction accuracy for load of y[i] = _____
Prediction accuracy for load of z[i] = _____
Prediction accuracy for load of t[i] = _____

(5pts) Can you design a predictor that can achieve higher accuracy for the predictions of x[i], y[i], z[i] and t[i]? If so, explain your design for each load operation separately.

(5pts) Discuss (and illustrate on an archetypical 5-stage [FDEMW] RISC-paradigm pipeline) the changes/additions necessary to implement last value prediction.

(5pts) What are the similarities and differences between value prediction and branch prediction? Explain.

(SCRATCH PAPER – **NOT GRADED)**

(SCRATCH PAPER – **NOT GRADED)**

(SCRATCH PAPER – **NOT GRADED)**