Name: CS520

Final Exam

13 May 2016, 120 minutes, 26 questions, 100 points

> The exam is closed book and notes.
>
> Please keep all electronic devices turned off and out of reach.
>
> Note that a question may require *multiple* checked boxes for a correct answer. Checking *some* but not *all* of the required boxes will result in a *partial* answer worth only 2 of the 4 points. Checking any box that shouldn't be checked results in an *incorrect* answer, worth zero.

1. [0 pts]
   - ◯ Return my exam to my Kingsbury mailbox.
   - ◯ Hold my exam in your office. I will pick it up prior to September 15.
   - √ Shred my exam. I never want to see it again.

2. Interpret `0xB0E9A56F` (shown in hexadecimal) as an IEEE single-precision floating-point value. Which of the following statements about this value are true? [4 pts]
   - √ It is negative.
   - √ Its stored exponent is `0x61`.
   - √ Its actual exponent is `-30` (in decimal).
   - ◯ It is a denormalized value.

3. In the goroutines assignment, which of the following items were stored in the Goroutine Control Block (GCB): [4 pts]
   - √ saved RSP register.
   - √ saved RDI register.
   - √ saved RSI register.
   - √ link to the next GCB in the ready list.

4. Decode the following UTF-8 sequence (shown as hexadecimal) to a Unicode character: `EC 83 93`. The answer represented in UTF-32 in hexadecimal is: [4 pts]
   - √ `0x0000C0D3`.
   - ◯ `0x0001C1D2`.
   - ◯ `0x0000D1B2`.
   - ◯ `0x0000C033`.
   - ◯ none of the above.

5. Consider the following Intel-64 instruction expressed in Linux assembly language: `addq (%rax),%rbx`. What does this instruction do? [4 pts]
   - ◯ adds the quadword in RBX to the quadword pointed to by RAX.
   - √ adds the quadword pointed to by RAX to the quadword in RBX.
   - ◯ adds the quadword in RAX to the quadword pointed to by RBX.
   - ◯ adds the quadword pointed to by RBX to the quadword in RAX.
   - ◯ none of the above.

6. Consider the following C function: [4 pts]

```
unsigned int f(void)
{
  int i = 0;
  return *(unsigned char *) &i;
}
```

On a machine with a byte-addressable memory, the function will:

- ◯ return 0 if the machine is little-endian and 1 otherwise.
- ◯ return 1 if the machine is little-endian and 0 otherwise.
- √ always return 0.
- ◯ none of the above.

7. When a goroutine trys to send to a full channel, it must *block*. This is done by: [4 pts]

- ◯ moving its GCB to the end of the ready list and yielding.
- √ moving its GCB to the end of a queue in the channel and yielding.
- ◯ setting the RIP register to the "start" function.
- ◯ popping the return address from the top of the runtime stack.
- ◯ none of the above.

8. Add the following two IEEE single-precision floating-point values shown in hexadecimal: `0x40000000` and `0xC0000003`. The result in hexadecimal is: [4 pts]

- ◯ `0xB5200000`.
- √ `0xB5400000`.
- ◯ `0xB5000003`.
- ◯ `0x45100000`.
- ◯ none of the above.

9. The encoding of this RISC-V instruction, `xori x3,x7,-7`, in hexadecimal is: [4 pts]

- ◯ `0x8073C193`.
- ◯ `0xFF934713`.
- √ `0xFF93C193`.
- ◯ `0x80734713`.
- ◯ none of the above.

10. In an implementation of an exception mechanism for C programs, the **cancelCatchException** function: [4 pts]

- ◯ pushs the current RBP, the saved RBP and the saved RIP onto the "snapshot" stack.
- ◯ clears the "catch" bit of the RAX register.
- √ pops the "snapshot" stack.
- ◯ none of the above.

11. A memory cache with a single set is known as a: [4 pts]

- ◯ direct-mapped cache.
- ◯ set-associative cache.
- √ fully-associative cache.
- ◯ translation lookaside buffer.
- ◯ none of the above.

12. The purpose of the sweep phase of a mark-and-sweep garbage collector is to: [4 pts]

- ◯ gather all the garbage blocks at one end of the heap.
- √ identify all the garbage blocks.
- ◯ identify all the reachable, allocated blocks.
- ◯ clear the mark bit on all blocks.
- ◯ none of the above.

13. Which of the following statements about Von Neumann's design for the IAS computer are true? [4 pts]

- √ It had an instruction that modified the address field of another instruction.
- √ It was a stored-program computer.
- √ It had a PC register.
- √ It packed two instructions per memory word.

14. Consider how the following two C loops would be accessed by a direct-mapped memory system with 16 sets and a block size of 2 ints. [4 pts]

```
for (i = 0; i < 256; i++)
  a[i] = i;
sum = 0;
for (i = 0; i < 256; i++)
  sum += a[i];
```

Assume only the array is in the cache and the first word of the array is at address zero. If the cache is initially empty, how many cache hits would there be for writes and how many cache hits would there be for reads?

- ◯ 0 read hits and 0 write hits.
- ◯ 64 read hits and 256 write hits.
- ◯ 128 read hits and 256 write hits.
- √ 128 read hits and 128 write hits.
- ◯ none of the above.

15. Consider how -29 (base 10) would be represented in the memory of a Little Endian machine as a 16-bit 2's complement integer. The two bytes, shown left to right in increasing memory address order, would be: [4 pts]

- ◯ `0xFF 0x23`.
- ◯ `0x23 0xFF`
- ◯ `0x22 0xFF`
- ◯ `0xFF 0x22`.
- ◯ `0x80 0x1D`.
- √ none of the above.

16. Consider this C declaration: [4 pts]

```
int *((*f)(int, long));
```

This declares a:

- √ pointer to a function that takes an int and a long, and returns a pointer to an int.
- ◯ pointer to a pointer to a function that takes an int and a long, and returns an int.
- ◯ function that takes an int and a long, and returns a pointer to a pointer to an int.
- ◯ none of the above.

17. The UTF-16 sequence (shown in hexadecimal) `0xD980 0xDC11` is represented in UTF-32 (in hexadecimal) as: [4 pts]

- ◯ two Unicode characters, `0x0000D980` and `0x0000DC11`.
- √ a single Unicode character, `0x00070011`.
- ◯ two Unicode characters, `0x00000007` and `0x00000011`.
- ◯ a single Unicode character, `0x00060011`.
- ◯ none of the above.

18. We used a POSIX mutex to implement a reentrant lock in order to: [4 pts]

- ◯ block a thread that needs to wait for the lock to be available.
- √ protect against concurrent access to the struct maintaining the state of the reentrant lock.
- ◯ enable condition variables to be associated with the lock.
- ◯ none of the above.

19. Which of the following statements are true? [4 pts]

- √ Insymbols in an object file are symbols defined in the file and made available to the linker.
- √ Outsymbols in an object file are symbols referenced in the file but not defined in the file.
- √ The insymbol section of an object file contains each insymbol and its address.
- √ The outsymbol section of an object file contains each outsymbol and the address of every reference to the outsymbol.

20. Which of the following are examples of a program exhibiting spatial locality: [4 pts]

- √ iterating through all members of an array in order.
- ◯ repeatedly incrementing a loop counter variable inside a loop.
- √ a sequence of instructions being executed in order without any branch or call instructions.
- ◯ putting the function return address in the RAX register.

21. Add together the following two 8-bit 2's complement integers (shown in hexadecimal): `0x81` and `0xFF`. Which of the following are true statements about the result? [4 pts]

- √ The result (in hexadecimal) is `0x80`.
- √ The result is negative.
- ◯ The result overflows.
- √ The result in decimal is -128.
- ◯ The result in decimal is 128.

22. Which of the following statements about calling `pthread_cond_wait` are true? [4 pts]

- ◯ The calling thread will only block if another thread is already waiting on the condition variable.
- √ The calling thread must first have locked the mutex that is passed as a parameter.
- √ The calling thread's lock of the mutex passed as a parameter will be released while it waits for the condition variable.
- ◯ The calling thread will first wait to acquire the lock on the mutex passed as a parameter and then it will wait on the condition variable.

23. Which of the following statements are true about the RISC-V ELF files that we processed in Program 3? [4 pts]

- √ The code is stored in the ".text" section.
- √ They contain a section header table.
- ◯ They are human-readable (contain only ASCII characters).
- ◯ They are always the same length.

24. The *capacity* of a channel: [4 pts]

- √ determines the size of the buffer in the channel.
- ◯ determines its ability to accept messages of different lengths.
- ◯ is the number of messages currently buffered in the channel.
- ◯ none of the above.

25. On the Intel 64 machine, the CMPXCHG instruction is typically used with the LOCK prefix: [4 pts]

- √ because its two memory accesses need to be atomic.
- ◯ to automatically lock a Pthread mutex.
- ◯ to inform the hardware that a low-level lock is being implemented.
- ◯ none of the above.

26. In Intel 64 stack frames, the saved RBP registers: [4 pts]

- ◯ store the return address of the caller for each active function call.
- ◯ stores the base address of the stack for each active thread.
- √ implement a linked list of the frames for all active function calls.
- ◯ none of the above.