Computer Organization and Operating System Design (CSE 500)

Final Exam
January 2020 Term
Syracuse University
Professor: William Katsak

Standard Score: 100
Points Available: 120

Name: JACK IMBURGIA

Multiple Choice (36 pts)

1. On MIPS, how are function arguments passed? (2 pts)
   (a) On the stack
   (b) In registers
   (c) By osmosis
   (d) Via telepathy
   (e) None of the above
2. MIPS register conventions reserve which set of registers for use by interrupt handlers? (2 pts)
   (a) `$K`-registers
   (b) `$I`-registers
   (c) `$Z`-registers
   (d) `$S`-registers
   (e) None of the above
3. Which "law" can be used to calculate speedup of parallel programs? (2 pts)
   (a) Amdahl's Law
   (b) Moore's Law
   (c) Murphy's Law
   (d) Faraday's Law
   (e) None of the above
4. Modern computer processors use which numbering system internally? (2 pts)
   (a) Ternary (Base-3)
   (b) Binary (Base-2)
   (c) Hexadecimal (Base-16)
   (d) Octal (Base-8)
   (e) None of the above
5. What is the private work space dedicated to a function called? (2 pts)
   (a) Stack Frame
   (b) Heap
   (c) Reserve
   (d) Allocation
   (e) None of the above
6. On MIPS, in the case of nested function calls, where are the return addresses of previous functions (i.e. not the current function) stored? (2 pts)
   (a) Heap
   (b) OS Memory
   (c) Registers
   (d) In the MMU
   (e) None of the above

7. _____ allow a program to have multiple execution contexts that share memory. (2 pts)

   (a) Processes
   (b) Tasks
   (c) Strings
   (d) Threads
   (e) None of the above

8. Paging (for memory) and the inode/block pointer scheme used in the UNIX filesystem are examples of _____ allocation schemes. (2 pts)

   (a) Contiguous
   (b) Segmented
   (c) Naive
   (d) Indexed
   (e) None of the above

9. Which of the following IS an operational mode on a MIPS processor (2 pts)

   (a) User Mode
   (b) Network Mode
   (c) Secure Mode
   (d) Superuser Mode
   (e) None of the above

10. The piece of software that transforms code written in a high level language (like C) to a lower level language (like assembly) is called what? (2 pts)

    (a) Word Processor
    (b) Assembler
    (c) Operating System
    (d) Compiler
    (e) None of the above

11. Which of the following is NOT a processor Instruction Set Architecture (ISA)? (2 pts)

    (a) x86
    (b) x86-64
    (c) ARM
    (d) MIPS
    (e) None of the above

12. On MIPS, where is the current function's return address stored? (2 pts)

    (a) In the cloud
    (b) On the stack
    (c) On the heap
    (d) In a register
    (e) None of the above

13. Which synchronization scheme makes it possible to sleep inside a critical section? (2 pts)

(a) Monitors
(b) Locks
(c) Semaphores
(d) Flags
(e) None of the above

14. What is the length of MIPS instructions? (2 pts)

(a) 1 byte
(b) 4 bytes
(c) Variable
(d) 32 bytes
(e) None of the above

15. Which "law" refers to the observation that the number of transistors per area of a microchip die doubles roughly every 18 months? (2 pts)

(a) Amdahl's Law
(b) Moore's Law
(c) Murphy's Law
(d) Faraday's Law
(e) None of the above

16. Contiguous memory allocation schemes tend to lead to _____ fragmentation. (2 pts)

(a) Internal
(b) Extreme
(c) External
(d) Disk
(e) None of the above

17. A single UNIX-style pipe provides what mode of communication? (2 pts)

(a) Telepathic
(b) Full-duplex
(c) Half-duplex
(d) Quantum
(e) None of the above

18. Where is the filename stored in the UNIX filesystem? (2 pts)

(a) Inode
(b) Directory Block
(c) Data Block
(d) Superblock
(e) None of the above

Binary Numbers (14 pts)

For this section, perform the following bitwise operations:
Assume the following:

- All operations are taking place on an imaginary 8-bit machine.

1. Convert each number from binary to decimal using both unsigned and 2's complement.

   (a) 1100 0111_2 (2 pts) 1 1 0 0 0 1 1 1

   $-2^7 + 2^6$ $+ 2^2 + 2^1 + 2^0$

   $-128 + 64$ $+ 4 + 2 + 1 =$ -57

   (b) 0100 0111_2 (2 pts) 0 1 0 0 0 1 1 1

   26 $+ 2^2 + 2^1 + 2^0$

   64 $+ 4 + 2 + 1 =$ 71

2. Perform the following bitwise operations:

   (a) 0101 1010 & 1001 0110 (2 pts)

   AND 0101 1010
   1001 0110
   0001 0010

   (b) 0101 1010 | 1001 0110 (2 pts)

   OR 0101 1010
   1001 0110
   1101 1110

   (c) 0101 1010 XOR 1001 0110 (2 pts)

   1 = DIFFERENT
   0 = SAME

   0101 1010
   1001 0110
   1100 1100

   (d) NOT 1111 0000 (2 pts)

   NOT 1111 0000
   0000 1111

   (e) 1001 1001 << 4 >> 4 (2 pts)

   1001 1001
   <<4 1001 0000
   >>4 0000 1001

Short Answer (20 pts)

1. On MIPS, is a function caller or the function itself (callee) responsible for saving `$sX` register values? (4 pts)

CALLEE

2. Give an example of a situation in which a lock (mutex) would be required. (4 pts)

WHEN IT IS REQUIRED THAT ONLY ONE THREAD AT A TIME CAN ACCESS A CRITICAL SECTION OF CODE.

3. Which MIPS instruction format is used for the `jal` instruction? (4 pts)

R FORMAT

4. Give the three operations provided by a condition variable, and briefly describe what they do: (4 pts)

WAIT - RELEASES LOCK AND GOES TO SLEEP AND REACQUIRES ON RETURN

SIGNAL - WAKES UP ONE WAITING THREAD

BROADCAST - WAKES UP ALL WAITING THREADS

5. On MIPS, what are the `$aX` registers used for? (4 pts)

PASS ARGUMENTS INTO FUNCTIONS

MIPS to C (20 pts)

1. Convert the following MIPS assembly into equivalent C code.

```
    addi $s0, $0, 0
    addi $s1, $0, 0
label1:
    slti $t0, $s0, 10
    beq $t0, $0, exit
    addi $s0, $s0, 1
    addi $t0, $0, 5
    slt $t1, $t0, $s0
    bne $t1, $0, label2
    j label1
label2:
    addi $s1, $s1, 1
    j label1
exit:
```

Assume two variables, `x` and `y`, stored in `$s0` and `$s1`, respectively.

```c
#include <stdio.h>

main() {
    int x = 0;
    int y = 0;

    while(x < 10){
        x = x + 1;
        if (5 < x){
            y = y + 1;
        }
    }

    // printf("x = %d, y = %d", x, y);
}

```

C to MIPS (10 pts)

1. Convert the following C program into MIPS assembly:

```
int add(int, int);

int main()
{
    return add(2, 2);
}

int add(int a, int b)
{
    // Note that we explicitly do use a local variable here.
    // You MUST reserve an s-register to store the value.
    int r = a + b;
    return r;
}
```

Please try to make your program as complete as possible. You can start with the following assembly:

```
.data
    # Any globals here

.text
.globl main
main:
    # Your code here

    # Terminate program run
    # syscall 17 is exit2, which takes a return value.
    # The return value should be loaded into $a0.
    li $v0, 17
    syscall

.globl add
add:
    # Your code here
```

Note that `main()` also returns a value.

Please attach your solution to this problem as an additional file named:
`Lastname-Firstname-Final.asm`.

Condition Variables (10 pts)

1. Assuming access to the following variables and methods:

```
// An initialized mutex.
Mutex mutex;

// An initialized condition variable, already associated with mutex.
ConditionVariable condition;

// A queue object.
Queue queue;

// Methods available
mutex.lock()            // Lock mutex
mutex.unlock()          // Unlock mnutex
condition.wait()        // Wait on condition
condition.signal()      // Signal condition
queue.push()            // Push to tail of queue
queue.pop()             // Pop from head of queue
queue.empty()           // Return true if the queue is empty
```

Provide pseudocode to complete the following program, using the monitor abstraction correctly.

```
void *producer() {
    while true {
        mutex.lock();
        queue.push(data);
        // Here, insert code to notify the consumer that data is ready.
        CONDITION.SIGNAL();

        mutex.unlock();
    }
}

void *consumer() {
    while true {
        mutex.lock();
        // Here, insert code to wait until data is available on the queue.
        // Assume that spurious wakeups are possible on your system.

        WHILE (QUEUE.LENGTH() = 0) {
            CONDITION.WAIT();
        }

        queue.pop();
        mutex.unlock();
    }
}
```

File Systems (10 pts)

1. Given a UNIX-style filesystem, compute the maximum supported file size if the disk block size is **32 KB**, disk block pointers are **8 bytes**, and the inode provides **12** direct block pointers, **2** single-indirect block pointers, and **1** double-indirect block pointer. (10 pts)

   Remember, in this scheme, each direct pointer holds the address of a disk block (to be used for data, also known as data blocks), each single-indirect pointer holds the location of a disk block (known as an indirect block) which is filled only with pointers to data blocks, and each double-indirect pointer holds the location of an indirect block which is filled with pointers to other indirect blocks, each of which contain pointers to actual data blocks.

DISK BLOCK SIZE = 32 KB

DISK BLOCK POINTERS = 8 BYTES

INODE

12 DIRECT BLOCK POINTERS

2 SINGLE-INDIRECT BLOCK POINTERS

1 DOUBLE INDIRECT BLOCK POINTER

I THINK IT'S 8 MB BUT I CAN'T REMEMBER THE MATH...