CSCI-UA.0201

Computer Systems Organization

Final Exam Spring 2015

(Duration: 75 minutes)

Last name: First name:

Net ID:

Notes:

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly**
- **Questions vary in difficulty; it is strongly recommended that you do not spend too much time on any one question.**
- **This exam is open book/notes.**

---

1. (5 points) Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

(A) Which of the following has does not cause the machine to move to kernel mode (i.e. stops current user-level program and OS takes over)?

1. divide by a negative number 2. divide by zero
3. system call 4. interrupt

(B) If a process calls open (i.e. opening a file) 3 times, how many descriptor tables will we end up having for this process?

1. one 2. two 3. three 4.four

(C) The memory region from the bottom of heap to the top of the heap is always 100% used.

1. The above statement is true
2. The above statement is false
3. It depends on whether the machine is 32-bit or 64-bit
4. It depends on whether the machine is big-endian or little endian.

(D) The length of the virtual address must always be larger than the length of the physical address.

1. Statement is true. 2. Statement is false. 3. Depends on OS.

(E) Which of the following has a larger size (in terms of bytes)?

1. pointer in a 32-bit machine 2. pointer in a 64-bit machine
3. pointer in a 32-bit machine pointing to an array of 100 integers

2. (1 pt) In many systems, a cache block size is 64 byte and a page size is 4KB. Why cache block size is always smaller than page size?

Because the difference in speed between cache and main memory (i.e SRAM vs DRAM) is smaller than between main memory and traditional disk. The speed includes everything (technology, bus frequency, ...).

3. Let's assume we have the following C code.

```
int nothing(int n)
        {
                int v;
                v = n * 2 + 1;
                return v;
        }
```

The compiler generated the following assembly code for the above C code.

```
pushl  %ebp
movl   %esp, %ebp
subl   $16, %esp
pushl  %ebx
movl   8(%ebp), %ebx
addl   %ebx, %ebx
addl   $1, %ebx
movl  %ebx, -4(%ebp)
movl  -4(%ebp), %eax
movl -20(%ebp), %ebx
movl %ebp, %esp
popl %ebp
ret
```

a. (1 pt) Where is v stored, relative to ebp?

b. (1 pt) Where is n stored, relative to ebp?

c. (1 pt) Is there anything stored in -20(%ebp)? If so, what is that?

d. (1 pt) Is there anything stored at -12(%ebp)? If so, what is that?

4. Consider the following C code:

```
main() {
        if (fork() == 0) {
                if (fork() == 0) {
                        printf("3");
                }
                else {
                        pid_t pid;     int status;
                        if ((pid = wait(&status)) > 0) {
                                printf("4");
                        }
                    }
                }
        else {
                if (fork() == 0) {
                        printf("1");
                        exit(0);
                }
                printf("2");
         }
        printf("0");
        return 0;
}
```

a. (1 pt) How many processes do we have after all fork() have been executed?
4

b.(2 pts) Out of the 5 outputs below, which ones correspond to possible output of the program (circle all possible ones).

A. 1234000 B. 2034012 C. 2300140 D. 2030401 E. 3200410

c.(1 pt) If the system on which the above program runs uses one level page table. How many page tables we end up having after all forks have been executed? Justify.
4 because there must be a page table per process.

d. (1 pt) How many processes execute the instruction “return 0”?
3

5. Consider a computer system that has a cache with 256 blocks. Each block can store 16 bytes. What will be the value stored in the <u>TAG field</u> of the cache block that holds the memory block containing the address 0x3CFBCF? As you can see, the address is 24 bits in length.

a.(2 pts) if the cache is a direct-mapped (i.e. 1-way set associative)

block = 16 bytes → 2^4 → offset is 4 bits
direct mapped → num sets = number of blocks = 256 = 2^8 → set is 8 bits
0x3CFBCF → TAG = 3CF SET = BC Offset = F

b. (2 pts)if the cache is 16-way set-asscociative

block = 16 bytes → 2^4 → offset is 4 bits
16-way → num sets = number of blocks/16 = 256/16 = 2^4 → set is 4 bits
0x3CFBCF → TAG = 3CFB SET = C Offset = F

c. (1 pt) Are caches accessed using virtual address or physical address?

physical address