CSCI-UA.0201-003

Computer Systems Organization

Final Exam Fall 2013

Dec 17th, 2013 - (Total time: 90 minutes)

Last name: First name:

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.**
- **Show all your thinking/steps so you can get partial credits.**
- **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) Suppose we have a process with one global variable. Then we do the following (in that order): we fork another process then we spawn 2 threads from each process. How many instance do we end up having from that global variable?

a. 1 (b. 2) c.3 d. 4

(B) If we want to add two arrays each of length N together. What is the most efficient way to get some performance?

(a. threads) b. processes c. doesn't matter d. sequential

(C) Suppose we try to get some performance from a process by spawning 4 threads from it. How many descriptor tables will we have for that process?

(a. 1) b. 2 c. 4 d.5

(D) Suppose you type a shell command line that looks like this: $> *progname* How many processes will result from this?

a. 1 b. 2 c. 4 (d. We cannot tell.)

(E) If your program has a bug that results in segmentation fault, we call it:

a. syscall (b. exception) c. function call d. interrupt

2. (5 points) What is the IEEE 754 presentation of the decimal number: -35?

The single precision 754 has 32 bits, divided into S (1 bit), E (8 bits), M(23 bits).
$-35 = -100011 = -1.00011 * 2^5$
S = 1 (since the number is negative)
E: 127 + 5 = 132 → 10000100
M = 00011000…0 (remember the hidden one)
So the final answer is:
1 100000100 00011000000…0

3. (6 points) Consider a small 2-way set associative cache with a total of 32 blocks and a block size of 16 bytes. Show how the address generated by the processor is split into tag, index (i.e. set), and offset in a 32-bit machine.

The address is split into: TAG SET OFFSET
Block size is 16 bytes = 2^4 → 4 bits needed for offset
2-way set associative means each set has 2 blocks.
The cache has a total of 32 blocks → number of sets = 32/2 = 16 = 2^4 → 4 bits needed for SET
We have 32-bit machine → The address is 32 bits in length → TAG needs 32 – (4+4) = 24 bits

4. (8 points) We have studied that a memory access by the processor needs to go to different stages before required data can be delivered to the processor (i.e. translating from virtual to physical, accessing caches, etc.). Suppose we have a processor with 1 TLB, 2 levels of cache, and 1 level of page table.

a. What is the best case scenario (i.e. the scenario where data is provided to the processor as fast as possible)?

- TLB access → TLB hit → physical address is ready
- L1 cache is accessed with physical address → L1 hit

b. What is the worst case scenario?

- TLB access → TLB miss
- Page Table accessed → page fault
- A page needs to be replaced from memory to make room for the page coming from the disk → This victim page is dirty and needs to be written back to disk
- The new page is brought from disk to memory
- Page table updated
- TLB updated → Physical address is ready
- L1 cache is accessed → L1 miss
- L2 cache is accessed → L2 miss
- Memory is accessed and cache block is brought to L2, then to L1 (with possible victim blocks to be written back)

5. (6 points) For the assembly code on the left, fill in the blanks on the right. Note:

- You may only use the C variable names `n`, `a`, `i` and `x`, not register names.
- When accessing a, use the array notation (i.e. a[]) and not the pointer notation (for example, use a[i] and not *(a+i))

```
looper:
  pushl %ebp
  movl %esp,%ebp
  pushl %esi
  pushl %ebx
  movl 8(%ebp),%ebx
  movl 12(%ebp),%esi
  xorl %edx,%edx
  xorl %ecx,%ecx
  cmpl %ebx,%edx
  jge .L25
.L27:
  movl (%esi,%ecx,4),%eax
  cmpl %edx,%eax
  jle .L28
  movl %eax,%edx
.L28:
  incl %edx
  incl %ecx
  cmpl %ebx,%ecx
  jl .L27
.L25:
  movl %edx,%eax
  popl %ebx
  popl %esi
  movl %ebp,%esp
  popl %ebp
  ret
```

```
int looper(int n, int *a) {
  int i;
  int x = ____0____;

  for(i = ____0____;

        ____i < n____;

        i++) {

    if (____a[i] > x____)

      x = ____a[i]____;


      ____x++____;
  }

  return x;
}
```

Things will be clearer to you once you figure out the map between the registers and the variables:

ebx → n
esi → a
edx → x
ecx → i