CS 61C Summer 2015
Guerrilla Section 5: VM, I/O & ECC

Problem 1: VM (adapted from Sp07 Final):

You run the following code on a system with the following parameter:

- 4 GiB virtual address space with 2 KiB page size, 1 GiB physical address space
- 8-entry TLB using LRU replacement
- 32 KiB data cache with 8B blocks and 4-way associative using LRU replacement

Assume that `char A[]` is both block-aligned and page-aligned, and that the code takes 1 page.

```
#define ARRAY_SIZE 33554432  // equal to 2^25
main() {
    int sum = 0, prod = 0;
    char *A = (char *) malloc(ARRAY_SIZE * sizeof(char));
    for (int i = 0; i < ARRAY_SIZE / STRETCH; i++) {
        for (j = 0; j < STRETCH; j++)      sum += A[i * STRETCH + j];
        for (j = STRETCH – 1; j >= 0; j--) prod *= A[i * STRETCH + j];
    }
}
```

a. What is the number of VPN bits? PPN bits? VPN bits: _____ PPN bits: _____

b. As we double STRETCH from 1 to 2 to 4 (…etc), we notice the number of cache misses doesn't change! What's the largest value of STRETCH before cache misses changes?

c. As we double STRETCH from 1 to 2 to 4 (…etc), we notice the number of TLB misses doesn't change! What is the largest value of STRETCH before TLB misses changes?

d. For any value of STRETCH, what is the fewest number of page faults we could ever generate? Round to the nearest power of two.

e. Now suppose instead of 2 KiB pages, we had 256 B pages. All other specified parameters remain the same. If our code is 64 instructions long, how would performance change if we loop unrolled by a factor of 8? Explain.

Problem 2: More VM (Su12 Final):

Consider the following OpenMP snippet:

```
int values[size]; #pragma omp parallel {
      int i = omp_get_thread_num();       int n = omp_get_num_threads();
for(int j = i * (size / n); j < (i + 1) * (size / n); j++)       {
            values[j] = j;       }
}
```

All cores share the same physical memory and we are running 2 threads. This is the sole process running. Each page is 1 KiB, and you have 2 pages of physical memory. The code snippet above starts at virtual address 0x400, and the values array starts at 0x800. The size, n, i, and j variables are all stored in registers. The functions omp_get_thread_num and omp_get_num_threads are stored in virtual addresses 0x440 to 0x480. The replacement policy for the page table is Least Recently Used.

At the start of the pragma omp parallel call the page table looks as follows:

Virtual Page Number	Valid	Dirty	Physical Page Number
0	0	0	0
1	1	0	0
2	1	1	1
3	0	0	1

a) How many page faults will occur if size = 0x080? _____

b) What is the minimum number of page faults that will occur if size = 0x200? _____

What is the maximum? _____

c) How could you reduce the maximum page faults for part (b)? Choose the valid options amongst the following:

___ increase virtual address space

___ increase physical address space

___ use SIMD instructions

___ change page table replacement policy to Random

___ decrease number of threads

___ increase number of threads

___ add a 4-entry TLB

Problem 3: Even More VM (from the Sp '04)

This is for questions (a) and (b). The first-level data cache for a certain processor can cache 64 KB of physical memory. Assume that the word size is 32 bits, the block size is 64 bytes, the size of the physical memory is 2 GB, and the cache is 4-way set associative.

a) How many bits are needed for the following? Show your work on the right. (3 points)

Tag	Index	Offset

b) For each of the following changes to the initial conditions above, indicate how these bits (i.e., the width of these fields) shift around. E.g., if a bit field stays the same, write "0", if a bit field *increases* by 5, write "+5", if a bit field *decreases* by 1, write "-1". (6 points)

Change	**Tag**	**Index**	**Offset**
Double the cache size (from 64 KB to 128 KB)			
Double the word size (from 32 bits to 64 bits)			
Change the associativity to fully associative			

c) A rich student who didn't take CS61C decides to splurge and buy 4 GB of rocket-fast RAM for their 32-bit MIPS system. They think to themselves: "Why should I turn on Virtual Memory?". What's the strongest argument (one sentence max) for turning it on? (3 pts)

d) Suppose a computer has a 32-bit virtual address space, 64 MB physical memory and a 4 KB page size. Based on this information, answer the following questions. Show your work. (6 pts)

How many virtual pages are there?	
How many physical pages are there?	
Assuming a one-level page-table design with each page table entry consuming 1 word, what is the size of the page table, in bytes?	

Problem 4: Potpourri! ECC, DMA, I/O

1. Hamming Codes

a) Given an N bit field, how many of those bits will be parity bits?

b) How long should a field be to store 15 bits of data with Hamming ECC for single error correction?

c) Come up with a generalized equation for the number of parity bits you need to store N data bits.

d) You are given a 21 bit number: 0x0a8c3c, which is storing some information using hamming encoding.

i) Is there no error, one error, or two?

ii) Fix the error. Now what do the entire 21 bits read?

iii) What information is being stored?

2. RAID (Partially from the Su '12 Final)

1) Which type(s) of RAID (1, 3, 4 or 5) would be best to fit the following needs?

a) Many fast reads _____ b) Many fast writes _____

c) Fast reads of critical information _____ d) Fast reads of small, byte-size data _____

2) There's a single disk failure in a disk array (you know which disk failed) and you want to read a single page. Assume that for block-striped arrays, the page is contained within a single block.

a) What's the fewest number of disks you have to read from if the array were:
i. RAID 1 with 2 disks _____ ii. RAID 5 with 4 disks _____

b) (2 point) What's the greatest number of disks you have to read from if the array were:
i. RAID 4 with 4 disks (including parity) _____ ii. RAID 5 with 4 disks _____

3. I/0

How should the following devices share information with a computer?
Pick between **P**olling, **I**nterrupt and **D**irect Memory Access.

a) Mouse _____ b) Hard drive _____ c) GPU _____

d) Network Cards _____ e) Keyboard _____

4. Interrupt (sp ’99, ’98, ‘94)

1) Circle T or F:

a) T or F: When an exception or I/O interrupt occurs, interrupts are disabled while vital information is being saved.

b) T or F: Both add and addu can overflow. The only difference is that addu doesn't trigger an exception.

c) T or F: The instructions, lw and sw, do not trigger exceptions or I/O interrupts.

d) T or F: In order for an I/O interrupt to occur (for example when a key on the keyboard is pressed), the only bit that needs to be set is the device's I.E., found in one of its registers.

2) A network has a latency of 100 ms, and a bandwidth of 10MB/s. How long will it take to transmit 1 byte across the network?

How long will it take to transmit 100 MB of data?

3a) When an exception occurs, the MIPS processor does all of the following except:

a. reads the Cause register
b. runs the code starting at location 0x80000080
c. switches to kernel mode and disables interrupts
d. saves the address of the instruction that raised the exception

3b) The main advantage of using interrupts is:

a. allows the processor to do other useful tasks while waiting for slow I/O
b. allows centralized error handling
c. allows the processor to switch to kernel mode
d. allows a user program to have access to I/O devices