CS 61C Summer 2016
Guerrilla Section 7: Virtual Memory, I/O, ECC

Question 1:

The system in question has 1MiB of physical memory, 32-bit virtual addresses, and 256 physical pages. The memory management system uses a fully associative TLB with 128 entries and an LRU replacement scheme.

a. What is the size of the physical pages in bytes?

b. What is the size of the virtual pages in bytes?

c. What is the maximum number of pages a process can use?

d. What is the minimum number of bits required for the page table base address register?

Everybody Got Choices

e. Answer "Yup!" (True) or "Nope!" (False) to the following questions

i.	The page table is stored in main memory	**Yup!** **Nope!**
ii.	Every virtual page is mapped to a physical page	**Yup!** **Nope!**
iii.	The TLB is checked before the page table	**Yup!** **Nope!**
iv.	The penalty for a page fault is about the same as the penalty for a cache miss	**Yup!** **Nope!**
v.	A linear page table takes up more memory as the process uses more memory	**Yup!** **Nope!**

Question 2:

F2) *V(I/O)rtual Potpourri* (23 pts, 30 mins)

For the following questions, assume the following:

- 16-bit virtual addresses
- 4 KiB page size
- 16 KiB of physical memory with LRU page replacement policy
- Fully associative TLB with 4 entries and an LRU replacement policy

a) What is the maximum number of virtual pages per process? _____

b) How many bits wide is the *page table base register*? _____

For questions (c) and (d), assume that:

- Only the code and the two arrays take up memory
- The arrays are both page-aligned (starts on page boundary)
- The arrays are the same size and do not overlap
- ALL of the code fits in a single page and this is the only process running

```
void scale_n_copy(int32_t *base, int32_t *copy, uint32_t num_entries,
     int32_t scalar)
{
    for (uint32_t i=0; i < num_entries; i++)
        copy[i] = scalar * base[i];
}
```

c) If **`scale_n_copy`** were called on an array with **`N`** entries, where **`N`** is a multiple of the page size, how many page faults can occur in the worst-case scenario?

Answer: _____

d) In the best-case scenario, how many iterations of the loop can occur before a TLB miss?

Answer: _____

Question 3:

For the following questions, assume the following (IEC prefixes are on your green sheet):

- **You can ignore any accesses to instruction memory (code)**
- 16 EiB virtual address space per process
- 256 MiB page size
- 4 GiB of physical address space
- Fully associative TLB with 5 entries and an LRU replacement policy
- All arrays of doubles are **page-aligned** (start on a page boundary) and do **not** overlap
- All arrays are of a size equivalent to some nonzero integer multiple of 256 MiB
- All structs are tightly packed (fields are stored contiguously)
- All accesses to structs and arrays go out to caches/memory (there is no optimization by reusing values loaded into registers)

```
typedef struct { *double dbl; double fun;} doubleFun

void dblCpy(doubleFun* measurer, double* dblsToCpy) {
     measurer->fun = 0;
     for (uint32_t i = 0; i < ARRAY_SIZE; i+=4) {
          measurer->dbl[i] += dblsToCpy[i];
          measurer->fun += dblsToCpy[i];
     }
     measurer->fun /= ARRAY_SIZE;
}
...
/* Now, the code goes on to call the function dblCpy. Assume that space
for the array pointed to by measurer->dbl was allocated at some time in
the past and that all elements in the array were set to 0. The arrays
dblsToCpy and measurer->dbl are each of length ARRAY_SIZE.*/
... // dblCpy function call here
```

a) Fill the following table:

Virtual Page Number Bits:	Virtual Address Offset Bits:

Physical Page Number Bits:	Physical Address Offset Bits:

b) Assume the TLB has just been flushed. What TLB hit to miss ratio would be encountered if sizeof(double) * ARRAY_SIZE = 256 MiB and we run the above code? Show your work.

c) In the best-case scenario, how many iterations can be executed with no TLB misses? Use IEC prefixes when reporting your answer. Show your work.

Question 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?

2) Assume that we have an encoded value, 1001110_{two} with a single-bit error. Indicate below each parity bit if it has an error:

Parity Bit	P1	P2	P4
OK/ERROR			

Incorrect bit position: _____

Correct data: _____

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 _____

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

3. I/0

How should the following devices share information with a computer?
Pick between **P**olling, **I**nterrupt

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

d) Network Cards _____ e) Keyboard _____