

Name: KEY

Login: cs61c-\_\_\_

1. True/False

Directions: Circle either True or False.

F  
F  
F  
T  
F  
T  
F  
T  
F  
F  
T  
F

- a. (True/False) According to Flynn's taxonomy, a multicore CPU is classified as MISD.
- b. For each claim about Warehouse Scale Computing, indicate whether it is true or false.
  - i. (True/False) An energy-proportional server has energy usage increase quadratically with load.
  - ii. (True/False) Improving the power efficiency of an individual compute node will also improve the Power Utilization Effectiveness (PUE) of a datacenter.
  - iii. (True/False) The greatest part of monthly expenses for a datacenter are amortized Capital Expenditures (CAPEX).
- c. For each claim about the MapReduce programming model, indicate whether it is true or false.
  - i. (True/False) MapReduce programs running on a single core are usually faster than a simple serial implementation
  - ii. (True/False) MapReduce works well on clusters with hundreds or thousands of machines
  - iii. (True/False) MapReduce is the only framework used for writing large distributed programs.
  - iv. (True/False) MapReduce can sometimes give the wrong answer if a worker crashes.
  - v. (True/False) A single Map task will usually have its map() method called many times.
- d. For each claim about Hadoop MapReduce, indicate whether it is true or false.
  - i. (True/False) If you don't define map() or reduce(), the framework will use the identity function.
  - ii. (True/False) Each reducer's output is sorted after the last call to reduce()
  - iii. (True/False) The values associated with a given key are sorted.
  - iv. (True/False) The Reducers can start copying and sorting map outputs before all maps have finished
  - v. (True/False) The map() function will be called exactly once for each input key-value pair

- Minus 1 for each wrong answer  
(minimum of 0)

Name: KEY

Login: cs61c-        

## 2. Number Systems and Floating Point

Creatures on Mars have three "fingers" on three "hands", and consequently have a base 9 number system (basically nine weird symbols that translate into 0, 1, 2, 3, 4, 5, 6, 7, 8).

a. What is the number  $188_{\text{nine}}$  in base 10?

161 ten

b. What is the number  $2A5_{\text{hex}}$  in Martian base 9?

832 nine

- 2 points each  
- partial credit if I saw stupid mistake in shown work

c. Convert the following single-precision floating point numbers back to a decimal representation. Match each bit pattern with a single member of the given bank of decimal numbers by writing the letter of the matching real number into the blank.

- i. ~~0~~ 10111010 010100000000000000000000 C  
ii. 0 11111111 000000000000000000000000 b  
iii. 1 11111111 01000000011001100000101 d  
iv. 0 00011000 010000000000000000000000 g

- a.  $1.25 * 2^{24}$   
b. infinity  
c.  $1.3125 * 2^{59}$   
d. NaN  
e.  $1.3125 * 2^{-103}$   
f. -infinity  
g.  $1.25 * 2^{-103}$   
h.  $1.25 * 2^{61}$

1 pt. each

d. Given the approximately 4 billion binary patterns in a 32-bit word, a single-precision floating-point word can exactly represent

- approximately  $2^{30}$  (1 billion) real numbers between  $-\infty$  (infinity) and  $-1$ ,
- approximately  $2^{30}$  (1 billion) real numbers between  $-1$  and  $0$ ,
- approximately  $2^{30}$  (1 billion) real numbers between  $0$  and  $+1$ , and
- approximately  $2^{30}$  (1 billion) real numbers between  $+1$  and  $+\infty$  (infinity).

You can use English words or numbers in scientific notation in your answers.

- 2 pts each

- answers had to be  $\sim 2^{30}$ , not  $2^{29}$  or  $2^{31}$   
for credit

Name: KEY

Login: cs61c-\_\_

L1

3. Caches

a. The Average Memory Access Time equation (AMAT) has three components: hit time, miss rate, and miss penalty. For each of the following cache optimizations, indicate which component of the AMAT equation may be improved. Circle one.

- Using a second-level cache      hit time    miss rate    miss penalty
- ~~• Using smaller blocks      hit time    miss rate    miss penalty~~
- Using larger blocks      hit time    miss rate    miss penalty
- Using a smaller first-level cache    hit time    miss rate    miss penalty
- Using a larger first-level cache    hit time    miss rate    miss penalty

1 pt each (max 4)

b. Given a direct-mapped cache, initially empty, and the following memory access pattern (all byte addresses and 32-bit word accesses, 32-bit addresses)

8    0    4    32    36    8    0    4    16    0

What is the hit rate, miss rate, and what blocks are in the cache after these accesses if

i. the cache has 8 32-bit blocks?

hit rate: 0.2    miss rate: 0.8

2 pts - correctness  
1 pt - miss (adds to 100%? is a rate?)

blocks at end (write the appropriate full byte addresses (NOT the tag) in the appropriate blocks, or "EMPTY" if the block is empty):

|                              |
|------------------------------|
| <del>0</del> <del>32</del> 0 |
| <del>4</del> <del>36</del> 4 |
| 8                            |
| EMPTY                        |
| 16                           |
| EMPTY                        |
|                              |
|                              |

3 pts total

- 1: minor mistake, easily seen
- 2: fairly major mistake
- 3: we had no idea what was happening

Name: KEY

Login: cs61c-    

The memory access pattern is repeated here for your convenience:

8 0 4 32 36 8 0 4 16 0

ii. the cache has 4 32-bit blocks?

hit rate: 0.1 miss rate: 0.9 *Same as before*

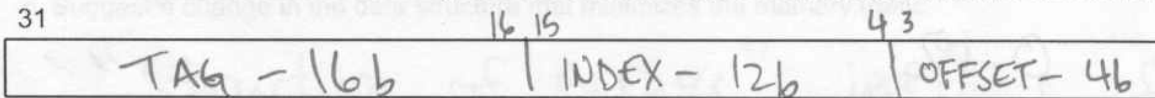
blocks at end (write the appropriate full byte addresses (NOT the tag) in the appropriate blocks, or "EMPTY" if the block is empty):

|       |
|-------|
| 0     |
| 4     |
| 8     |
| EMPTY |

*Same as before*

c. Consider a write-allocate, write-back, direct-mapped cache with 16 byte blocks and  $64 \times 2^{10}$  bytes of data bits. Assume a byte-addressed machine with 32-bit addresses.

i. Partition the following address and label each field with its name and size in bits.



*3 pts  
-1 for each*

ii. Given the address DEADBEEF<sub>hex</sub>, what is the value of the index, offset, and tag? (Write your answers in hexadecimal.) *mistake (tag wrong b/c of wrong index was ok)*

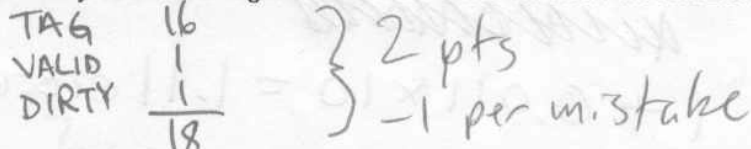
index = 0x BEE

offset = 0x F

tag = 0x DEAD

*1 pt, correctness based on above breakdown*

iii. How many cache management bits are there for each block? List them.



iv. What is the total number of bits (data **AND** cache management) that comprise the cache?

$2^{12}$  blocks  
18 management bits per block  
128 data bits per block

*2 pts  
-1 per mistake*

$2^{12} (128 + 18) \text{ bits} = 584 \text{ Kbits}$

Name: KEY

Login: cs61c-    

#### 4. AMAT

Suppose a MIPS program executes on a machine with a single data cache and a single instruction cache, and

- 20% of the executed instructions are loads or stores;
- the data cache hit rate is 95%;
- the instruction cache hit rate is 99.9%;
- the instruction and data cache miss penalty is 10 cycles;
- the instruction and data cache hit time is 1 cycle; (Ideal CPI = 1, so overlapped).

a. How many memory references are there per executed instruction?

1.2

1 POINT FOR .2

b. How many data cache misses are there per executed instruction?

$$20\% \times 5\% = 1\%$$

$$\rightarrow 0.2 \times 0.05 = 0.01$$

c. How many instruction cache misses are there per executed instruction?

0.1%

$$\rightarrow 0.001$$

d. Assume that if there were no cache misses, the CPI would be 1. What is the CPI of the program given the cache miss rates above?

$$1 + 1\% \times 10 + 0.1\% \times 10 = 1.11$$

$$1 + 0.01 \times 10 + 0.001 \times 10 = 1.11$$

1 POINT IF MISCALCULATE  
in a. and b. but properly  
multiply x10 here

e. What is the average memory access time of the program?

HIT TIME + MISS RATE  $\times$  MISS PENALTY

~~1 + 0.011 \times 10 = 1.11~~

$$1 + 0.011 \times 10 = 1.11 \text{ CLOCK CYCLES}$$

1 POINT IF PROPERLY  
CALCULATE FOR INSTRUCTION  
CACHE MISSES OR  
DATA CACHE MISSES



Name: KEY

Login: cs61c-    

### 5. Cache-optimized data structures

In a physically-based animation you have the following array of structures:

```
struct vertex {  
    float x, y, z; /* position */  
    float vx, vy, vz; /* velocity */  
    float fx, fy, fz; /* force */  
    float nx, ny, nz; /* normal */  
    float tx, ty; /* texture */  
} vertices[10000];
```

You want to reset all forces (fx, fy, fz) to zero. Assume a 64KB cache with 64-byte blocks, write allocate, write back, direct mapped.

a. How many bytes do you need to set in the array in total?

$$3 \times 4 \times 10000 = 120,000$$

240.000 : 2 POINTS  
MINOR MISCALCULATION: -1

b. How many bytes will be read from memory when doing so?

$$14 \times 4 \times 10000 = 560,000$$

640.000 : 2 POINTS  
MINOR MISCALCULATION: -1

c. Suggest a change in the data structure that minimizes the memory traffic.

"Structure of Arrays" instead of  
"Array of Structures":

```
struct vertices {  
    float x[10000], y[10000], z[10000];  
};
```

4 POINTS EACH.

Name: KEY

Login: cs61c-    

### 6. MIPS, C, and Pointers

Given the following C definition for a node in a binary tree, and the following C source code that prints out the nodes in a tree post-order,

```
struct node {
    struct node *left, *right;
    const char* value;
};

void print_postorder(const struct node* root) {
    if(root != NULL) {
        print_postorder(root->left);
        print_postorder(root->right);
        printf("The node at address %p has value %s\n", root, root->value);
    }
}
```

fill in the blanks in the following MIPS assembly code that implements the print\_postorder function.

```
.data
thestring: .asciiz "The node at address %p has value %s\n"
.text
```

print\_postorder:

addiu \$sp, \$sp, -8

sw \$ra, 0(\$sp)

sw \$s0, 4(\$sp)

beq \$a0, \$0, out

move \$s0, \$a0

lw \$a0, 0(\$a0)

jal print\_postorder

lw \$a0, 4(\$s0)

jal print\_postorder

la \$a0, thestring

move \$a1, \$s0

lw \$a2, 8(\$s0)

jal printf

out:

lw \$ra, 0(\$sp)

lw \$s0, 4(\$sp)

addiu \$sp, \$sp, 8

jr \$ra

1 POINT PER BLANK.