Andrew login ID: _____

Full Name: _____

Recitation Section: _____

CS 15-213, Spring 2008

Exam 2

Thu. April 3, 2008

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, Andrew login ID, and recitation section (A–H) on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 59 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. No calculators or other electronic devices are allowed.
- Good luck!

1 (8):	
2 (9):	
3 (10):	
4 (12):	
5 (6):	
6 (8):	
7 (6):	
TOTAL (59):	

Problem 1. (8 points):

Consider the following C function:

```
data_t psum(data_t a[], data_t b[], data_t c[], int cnt)
{
    data_t r = 0;
    int i;
    for (i = 0; i < cnt; i++) {
        /* Inner loop expression */
        r =   r  +  b[i]  *  b[i]  +  a[i] * c[i] ;
    }
    return r;
}
```

In this code, data type `data_t` can be defined to different types using a `typedef` declaration.

According to the C rules for operator precedence and associativity, the line labeled "Inner loop expression" will be computed according to the following parenthesization:

```
r =  (r  + (b[i]  *  b[i]))  +  (a[i]  *  c[i]) ;
```

Imagine we run this code on a machine in which multiplication requires 5 cycles, while addition requires 3. Assume that these latencies are the only factors constraining the performance of the program. Don't worry about the cost of memory references, the cost of operations that compute and check the loop index, resource limitations, etc.

We list 4 different parenthesizations for the "Inner loop expression" below; they will not all compute the same result. For each parenthesization, write down the CPE (cycles per element) that the function would achieve. **Hint:** All of your answers will be in the set $\{3, 5, 6, 8, 10, 9, 11, 13, 15\}$.

```
// P1.  CPE =
r =  (r  + (b[i]  *  b[i]))  +  (a[i]  *  c[i]) ;

// P2.  CPE =
r =  ((r  +  b[i])  *  b[i])  +  (a[i]  *  c[i]) ;

// P3.  CPE =
r =  r  +  ((b[i]  *  b[i])  +  (a[i]  *  c[i])) ;

// P4.  CPE =
r =  (r  +  b[i])  *  (b[i]  +  (a[i]  *  c[i])) ;
```

Problem 2. (9 points):

Consider the following C function to sum all the elements of a 3×3 matrix. Note that it is iterating over the matrix **column-wise**.

```
char sum_matrix(char matrix[3][3]) {
  int row, col;
  char sum = 0;
  for (col = 0; col < 3; col++) {
    for (row = 0; row < 3; row++) {
      sum += matrix[row][col];
    }
  }
  return sum;
}
```

Suppose we run this code on a machine whose memory system has the following characteristics:

- Memory is byte-addressable.
- There are registers, an L1 cache, and main memory.
- Char's are 8 bits wide.
- The cache is direct-mapped, with 4 sets and 4-byte blocks.

You should also assume:

- `matrix` begins at address 0.
- `sum`, `row` and `col` are in registers; that is, the only memory accesses during the execution of this function are to `matrix`.
- The cache is initially cold and the array has been initialized elsewhere.

Fill in the table below. In each cell, write "**h**" if there is a cache hit when accessing the corresponding element of the matrix, or "**m**" if there is a cache miss.

	0	1	2
0			
1			
2			

Problem 3. (10 points):

Consider the following C function that calculates a value analogous to a matrix version of the dot-product. Note the different access patterns of A and B. Assume $X = 64$, which implies the total number of accesses is $2 * X^2 = 8192$.

```
int A[X][X], B[X][X];

int matrix_dot_product()
{
  int i, j, s = 0;
  for (i = 0; i < X; i++)
    for (j = 0; j < X; j++)
      s += A[i][j] * B[j][i];

  return s;
}
```

Assume the following:

- Memory consists of registers, an L1 cache, and main memory.
- The cache is cold when the function is called and the arrays have been initialized elsewhere.
- Variables i, j, s, and pointers to A and B are all stored in registers.
- The arrays A and B are aligned such $A[0][0]$ begins a cache block, and $B[0][0]$ comes immediately following $A[X-1][X-1]$. Assume in the above code that $A[i][j]$ is accessed before $B[j][i]$ for each value of i and j.

Case 1: Assume the cache is a 16KB direct-mapped data cache with 8-byte blocks. Calculate the cache miss rate by giving the number of **cache misses**. For partial credit briefly explain (one to two sentences) where collisions in the cache occur w.r.t. this access pattern (using locations of the arrays as a guide).

```
miss rate = __________ / 8192 (3 pts)
```

For your reference, $X = 64$ and here's the code snippet again:

```
int A[X][X], B[X][X];

int matrix_dot_product()
{
  int i, j, s = 0;
  for (i = 0; i < X; i++)
    for (j = 0; j < X; j++)
      s += A[i][j] * B[j][i];

  return s;
}
```

Case 2: Assume the cache has as many sets as in case 1, but is two-way set associative using an LRU replacement policy with 8-byte blocks. Calculate the cache miss rate by giving the number of **cache misses**.

miss rate = _____ / 8192 (3 pts)

1. Would a larger cache size, with the same cache-line size, reduce the miss rate? (Yes / No)

Case 1: _____ Case 2: _____ (2 pts)

2. Would a larger cache-line, with the same cache size, reduce the miss rate? (Yes / No)

Case 1: _____ Case 2: _____ (2 pts)

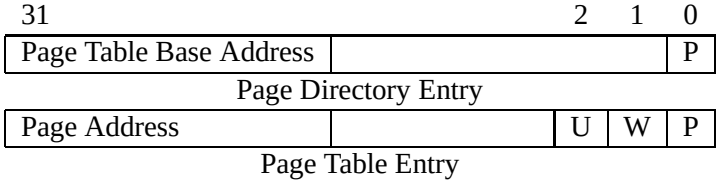
Problem 4. (12 points):

The following problem concerns virtual memory and the way virtual addresses are translated into physical addresses. Below are the specifications of the system on which the translation occurs.

- The system is a 32-bit machine - words are 4 bytes.
- Memory is byte addressable.
- The maximum size of a virtual address space is 4GB.
- The system is configured with 64MB of physical memory.
- The page size is 4KB.
- The system uses a two-level page tables. Tables at both levels are 4096 bytes (1 page) and entries in both tables are 4 bytes as shown below.

In this problem, you are given parts of a memory dump of this system running 2 processes. In each part of this question, one of the processes will issue a single memory operation (read or write of one byte) to a single virtual address (as indicated in each part). Your job is to figure out which physical addresses are accessed by the process if any, or determine if an error is encountered.

Entries in the first and second level tables have in their low-order bits flags denoting various access permissions.



- P = 1 $\Rightarrow$ Present
- W = 1 $\Rightarrow$ Writable
- U = 1 $\Rightarrow$ User-mode

The contents of relevant sections of memory is shown on the next page. All numbers are given in **hexadecimal**.

Address	Contents
001AC021	07693003
001AC084	00142003
0021A020	0481C001
0021A080	04A95001
0021A2FF	06128001
0021A300	05711001
0021ABFC	05176001
0021AC00	001AC001
0021B020	01FAC9DA
0021B080	052DB001
0021B2C0	0B2B36C2
0021B2FF	05A11001
0021B300	01FCF001
0021BBFC	06213001
0021BC00	001AC001
01FCF021	00382003
0481C048	0523A005
04A95048	048B8005
04A95120	07D6A005
051760F0	0E33F007
051763C0	08BF1007
052DB04A	09A62006
052DB128	0D718006
05711021	00113003
05A110F0	01133007
061280F0	0A114007
0614504A	0B183006
062133C0	052F1007

For the purposes of this problem, omitted entries have contents = 0.

Process 1 is a process in **user** mode (e.g. executing part of `main()`) and has page directory base address `0x0021A000`.
Process 2 is a process in **kernel** mode (e.g. executing a `read()` system call) and has page directory base address `0x0021B000`.

For each of the following memory accesses, first calculate and fill in the address of the page directory entry and the page table entry. Then, if the lookup is successful, give the physical address accessed. Otherwise, circle the reason for the failure and give the address of the table entry causing the failure. You may use the 32-bit breakdown table if you wish, but you are not required to fill it in.

1. Process 1 writes to `0xBFCF0145`.
   Scratch space:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(a) Address of PDE: 0x

(b) Address of PTE: 0x

(c) (SUCCESS) The address accessed is: 0x

OR

(FAILURE) The address of the table entry causing the failure is: 0x

The access failed because (circle one):
page not present \ page not writable \ illegal non-supervisor access

2. Process 2 writes to `0x0804A1F0`. Scratch space:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

(a) Address of PDE: 0x

(b) Address of PTE: 0x

(c) (SUCCESS) The address accessed is: 0x

OR

(FAILURE) The address of the table entry causing the failure is: 0x

The access failed because (circle one):
page not present \ page not writable \ illegal non-supervisor access

Problem 5. (6 points):

Pink Bowler Software

A. Harry Bovik, a programmer at Pink Bowler software, has written some code that uses the following two globals and a function called tophat(). In which sections of the ELF file would they be stored? Please just write the name of the section.

(a) `int valid;`

(b) `int last_pid = -1;`

(c) `int tophat(int size, double brim_width) { .... }`

B. While working one day, Harry Bovik notices that one of his scurrilous employees has made an important project print out "Bovik couldn't make it through 213." Bovik traces the offensive string to a `.h` file of string literals. He plans to overwrite this string in memory while the program is running. Can he do it? Why or not?

C. The following shows excerpts from three different files.

```
file0.c
     ...
     static int var;
     ...
file1.c
     ...
     int var = 0;
     double b;
     ...
file2.c
     int var;
     int foo(void) {
         int c;
         int error = printf("How's your exam going?\n");
         return error;
     }
     ...
```

These files are compiled with the command line

```
gcc -o code file0.c file1.c file2.c
```

(a) Would the linker report an error resolving the symbol '`var`'?

(b) If so, explain why. If not, describe how the three declarations of this symbol would be resolved. (A few sentences will suffice.)

D. In the above code, where would the string "`How's your exam going?\n`" appear?

( ) `.bss`
( ) `.rodata`
( ) `.text`
( ) `.data`

Problem 6. (8 points):

Read over the following program:

```
int main(int argc, char *argv[]) {
    printf("PB ");
    pid_t pid;
    if (pid = fork()) {
        printf("PS ");
        printf("PR ");
        kill(pid, SIGCONT);
    } else {
        printf("CS ");
        kill(getpid(), SIGSTOP);
        printf("CR ");
    }
    return 0;
}
```

Fill in the blanks below as indicated, with the following assumptions:

- No system calls will fail.
- No other processes in the system will send signals to either the parent or the child.
- `printf()` is atomic, and will call `fflush(stdout)` after printing its argument(s) but before returning.

Write all possible outputs of this program in the following blanks: (you might not use all of them)

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Problem 7. (6 points):

A. Given:

```
char *foo() {
    char *copy;
    char *mystring = "Pink > Red";
    copy = malloc(strlen(mystring));
    return strcpy(copy, mystring);
}
```

Assume the call to `malloc` is successful. Is this a correct snippet of code? Why or why not?

B. Harry Bovik has written the following function. He'd like you to look it over before he commits it.

```
#define SUCCESS 0
#define ERROR (-1)
#define UNLOCKED 0

typedef struct {
    int locked;
} mutex_t;

int mutex_init(mutex_t *p) {
    if ((p = malloc(sizeof(mutex_t))) == NULL)
        return ERROR;
    p->locked = UNLOCKED;
    return SUCCESS;
}

mutex_t *new_mutex() {
    mutex_t *new = malloc(sizeof(mutex_t));
    if ((new == NULL) ||(mutex_init(new) != SUCCESS))
        return NULL;
    return new;
}
```

Describe the two most serious problems this code has. 1–2 sentences for each should suffice.