NAME: _____ UID: _____

CS 4400: Computer Systems

Midterm Exam 2
Fall 2010

Please give your solutions in the space provided on the exam. If you choose to show your work on the exam, be sure to clearly indicate your final solution to each problem.

The exam is open-book, but closed-notes. *In addition, no laptops, calculators, cell phones, or other electronic devices are allowed.*

The point value of each question is clearly marked, so allocate your time wisely. The exam is worth a total of 75 points.

You must complete all work by 2:45pm, there are no exceptions.

Make sure that you have 13 numbered pages.

Problem 1	/ 14 points
Problem 2	/ 9 points
Problem 3	/ 12 points
Problem 4	/ 6 points
Problem 5	/ 12 points
Problem 6	/ 6 points
Problem 7	/ 8 points
Problem 8	/ 8 points
Total	/ 75 points

/ 14 points

1. One of your CS 4400 classmates has been given the job of improving the performance of a function that computes the factorial of its input parameter. Here is the original (unimproved) version of the function:

```
int factorial(int n) {
  int i;
  int result = 1;

  for(i = n; i > 0; i--)
    result *= i;

  return result;
}
```

Your classmate has attempted to unroll the loop by a factor of 2, producing this version of the function:

```
int factorial_u2(int n) {
  int i;
  int result = 1;

  for(i = n; i > 0; i -= 2)
    result = (result * i) * (i - 1);

  return result;
}
```

(a) For what values of `n` will the functions `factorial_u2` and `factorial` return different values?

(b) Provide a detailed description of how to correct `factorial_u2`, either in English or using C code.

(c) The performance of the original function `factorial` is 4.0 CPE. Even after correcting function `factorial_u2`, you find that the performance is still 4.0 CPE. Why doesn't unrolling by a factor of 2 reduce the CPE?

(Assume that the latency of integer addition is 1 cycle, with an issue time of 0.33 cycles. Also assume that the latency of integer multiplication is 4 cycles, with an issue time of 1 cycle.)

(d) It does improve the performance to change the loop body to:

```
result *= i * (i - 1);
```

Why is the CPE reduced and what is the new CPE?

(e) Suppose that another function `gcd` makes up 55% of a system, and you can produce new and improved version that speeds up `gcd` by a factor of 11.0.

You can improve yet another function `average` by a factor of 41.0. The `average` function makes up 41% of the system.

Improvement of which function, `gcd` or `average`, will result in the largest improvement of the overall system?

/ 9 points

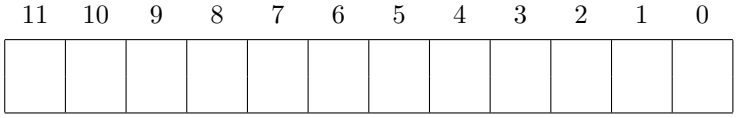
2. The following problem concerns basic cache lookups.

   - The memory is byte addressable.
   - Memory accesses are to **1-byte words** (not 4-byte words).
   - Physical addresses are 12 bits wide.
   - The cache is 4-way set associative, with a 2-byte block size and a capacity of 64 bytes.

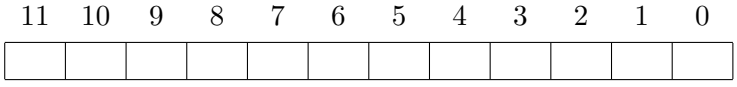
   The contents of the cache are as follows. Note that all numbers are given in hexadecimal and that the "V" column indicates the valid bit.

4-way Set Associative Cache															
Index	Tag	V	Byte0	Byte1	Tag	V	Byte0	Byte1	Tag	V	Byte0	Byte1	Tag	V	Byte0	Byte1
0	29	0	34	29	87	0	39	AE	7D	1	68	F2	8B	1	64	38
1	F3	1	0D	8F	3D	1	0C	3A	4A	1	A4	DB	D9	1	A5	3C
2	A7	1	E2	04	AB	1	D2	04	E3	0	3C	A4	01	0	EE	05
3	3B	0	AC	1F	E0	0	B5	70	3B	1	66	95	37	1	49	F3
4	80	1	60	35	2B	0	19	57	49	1	8D	0E	00	0	70	AB
5	EA	1	B4	17	CC	1	67	DB	8A	0	DE	AA	18	1	2C	D3
6	1C	0	3F	A4	01	0	3A	C1	F0	0	20	13	7F	1	DF	05
7	0F	0	00	FF	AF	1	B1	5F	99	0	AC	96	3A	1	22	79

(a) The box below shows the format of a physical address. Indicate the fields that would be used to determine the block offset (label as *CO*), the cache index (label as *CI*), and the cache tag (label as *CT*).



(b) Binary expansion of the physical address `E34` (one bit per box)



(c) For the given physical address, indicate the cache entry accessed and the cache byte value returned **in hex**. Indicate whether a cache miss occurs. (If there is a cache miss, enter "–" for "Cache Byte Returned".)

Parameter	Value
Cache Offset (CO)	0x
Cache Index (CI)	0x
Cache Tag (CT)	0x
Cache Hit? (Y/N)	
Cache Byte Returned	0x

/ 12 points

3. After following the recent mid-term election you decide to start a business in developing software for electronic voting. The software will run on a machine with a 1024-byte direct-mapped cache whose block size is 32 bytes.

   You are implementing a prototype of your software that assumes that there are three candidates. The structure you are using is defined as follows.

```
struct vote {
  int candidates[3];
  int valid;
};

struct vote vote_array[16][16];
int i, j, k;
```

   You have to decide between two alternative implementations of the routine that initializes the array `vote_array`. You want to choose the one with the better cache performance.

   You can assume the following.

   - `sizeof(int) = 4`
   - `vote_array` begins at memory address 0.
   - The cache is initially empty.
   - The only memory accesses are to the entries of the array `vote_array`. Variables `i`, `j` and `k` are stored in registers.

   (a) What percentage of the writes in the following code will miss in the cache?

```
for(i=0; i<16; i++)
  for(j=0; j<16; j++)
    vote_array[i][j].valid = 0;

for(i=0; i<16; i++)
  for(j=0; j<16; j++)
    for(k=0; k<3; k++)
      vote_array[i][j].candidates[k] = 0;
```

   Total number of misses in the first loop: _____

   Total number of misses in the second loop: _____

   Overall miss rate for writes to `vote_array`: _____ %

(b) What percentage of the writes in the following code will miss in the cache?

```
for(i=0; i<16; i++)
  for(j=0; j<16; j++) {
    for(k=0; k<3; k++)
      vote_array[i][j].candidates[k] = 0;

    vote_array[i][j].valid = 0;
  }
```

Miss rate for writes to `vote_array`: _____ %

(c) What percentage of the writes in the following code will miss in the cache?

```
for(j=0; j<16; j++)
  for(i=0; i<16; i++) {
    for(k=0; k<3; k++)
      vote_array[i][j].candidates[k] = 0;

    vote_array[i][j].valid = 0;
  }
```

Miss rate for writes to `vote_array`: _____ %

/ 6 points

4. In this problem, let `REF(x.i)` → `DEF(x.k)` denote that the linker will associate any reference to symbol `x` in module `i` with the definition of `x` in module `k`.

   Use this notation to indicate how the linker would resolve references to the multiply defined symbol in each pair of modules given below. If there is a link-time error, write "ERROR". If the linker arbitrarily chooses one of the definitions, write "UNKNOWN".

(a)

```
/* Module 1 */
#include <stdio.h>
double d = 1.23;
int foo();
void main() {
  printf("%f, %i\n", d, foo(1));
}
```

```
/* Module 2 */
static int d = 45;
int foo(int a) {
  return a + d;
}
```

i. `REF(d.1)` → `DEF(_____)`

ii. `REF(d.2)` → `DEF(_____)`

(b)

```
/* Module 1 */
#include <stdio.h>
void foo();
char *ptr = "Hello world!";
void main() {
  foo();
  printf("%s\n", ptr);
}
```

```
/* Module 2 */
int x;
int *ptr = &x;
void foo() {
  (*ptr)++;
}
```

i. `REF(ptr.1)` → `DEF(_____)`

ii. `REF(ptr.2)` → `DEF(_____)`

(c)

```
/* Module 1 */
#include <stdio.h>
int foo;
void main() {
  printf("%d\n", ++foo);
}
```

```
/* Module 2 */
int foo() {
  return 100;
}
```

i. `REF(foo.1)` → `DEF(_____)`

ii. `REF(foo.2)` → `DEF(_____)`

/ 12 points

5. Consider the following two modules of a program.

```
/* main.c */
#include <stdio.h>
#include <stdlib.h>

void find_max(int, int);

extern int max;

static int input_a = 10000;

static void test_find_max() {
  int input_b = rand();

  find_max(input_a, input_b);

  printf("The maximum of %d and %d is %d.\n", input_a, input_b, max);
}

int main() {

  test_find_max();

  return 0;
}
```

```
/* max.c */

int max;

void find_max(int a, int b) {
  if(a > b)
    max = a;
  else
    max = b;
}
```

Fill in the following table about the symbol references in `main.c`. Pay close attention to the choices for each entry.

- *Entry* in `main.o .symtab`? ["yes" or "no"]
- *Type* ["extern", "global", or "local"]
- *Module* where defined ["main.o", "max.o", or a module of "libc.a"]
- *Section* where defined ["`.text`", "`.data`", or "`.bss`"]

If you answer that there is no entry in the `.symtab` segment of relocatable object file `main.o` for a certain symbol, fill in the remaining entries for that symbol with "–".

Symbol	*Entry*	*Type*	*Module*	*Section*
`find_max`				
`input_a`				
`input_b`				
`max`				
`rand`				
`test_find_max`				

/ 6 points

6. Consider the C program below. For space reasons, we are not checking error return codes. Assume that all functions return normally and that the proper header files have been included.

```
int main() {
   pid_t pid1, pid2, pid3;

   pid1 = getpid();
   pid2 = fork();
   pid3 = getppid();

   if(pid1 == pid2)
      printf("X");
   if(pid2 == pid3)
      printf("Y");
   if(pid1 == pid3)
      printf("Z");

   sleep (1);

   return 0;
}
```

Give just one possible output of the program.

/ 8 points

7. Consider the C program below. For space reasons, we are not checking error return codes. Assume that all functions return normally and that the proper header files have been included.

```
int main() {
   if(fork() == 0)
      if(fork() == 0)
         printf("3");
      else {
         pid_t pid;
         int status;
         if((pid = wait(&status)) > 0) {
            printf("4");
            exit(0);
         }
      }
   else {
     printf("2");
     exit(0);
   }
   printf("1");
   return 0;
}
```

(a) Draw a diagram that illustrates the processes at run-time.

(b) Give all possible outputs of this program.

/ 8 points

8. Consider the C program below. For space reasons, we are not checking error return codes. Assume that all functions return normally and that the proper header files have been included.

```
void handler1(int sig) {
   printf("child got SIGUSR1\n");
}
void handler2(int sig) {
   printf("parent got SIGUSR1\n");
}
int main() {
   pid_t pid;
   if((pid = fork()) == 0) {
      signal(SIGUSR1, handler1);
      kill(getppid(), SIGUSR1);
      sleep(1);
      exit(0);
   }
   else {
      signal(SIGUSR1, handler2);
      kill(pid, SIGUSR1);
      waitpid(pid, NULL, 0);
   }
   return 0;
}
```

The programmer who created this program would like it to always print these two lines:

```
parent got SIGUSR1
child got SIGUSR1
```

These lines could appear in the reverse order in different executions because the scheduler decides whether the parent or child runs first following a fork. However, this program is flawed—it does not print the two lines the programmer expects it to print.

Describe the error.