Fall Semester 2020

KAIST EE209

Programming Structures for Electrical Engineering

Final Exam

Name: ______________________________

Student ID: ______________________________

This exam is closed book and notes. Read the questions carefully and focus your answers on what has been asked. You are allowed to ask the instructor/TAs for help only in understanding the questions, in case you find them not completely clear. Be concise and precise in your answers and state clearly any assumption you may have made. You have 165 minutes (9:00 AM – 11:45 AM) to complete your exam. Be wise in managing your time. Good luck.

Question 1	______/______
Question 2	______/______
Question 3	______/______
Question 4	______/______
Question 5	______/______
Total	______/______

Name: Student ID:

1. (22 points) **Simple question**

The scores of (2) and (6) are 3, and all others' marks are 2.

(1) Choose all condition flags that become 1 as the result of "cmpl $20,$10"
(CF (Carry Flag) becomes 1.)

(a) ZF (Zero Flag) (b) SF (Sign Flag) (c) OF (Overflow Flag)

Ans) (b)

(2) Below function checks which aspect of the system? Explain.

```
int ee209p12(void){
   int x = 0x01020304;
   char *p = (char *)&x;
   return (p[0] == 0x01)
}
```

Ans) If machine is big endian, it returns 1, otherwise it returns 0

(3) Assuming no errors, which one of the following statements about fork is true?

(a) Called once, returns once.
(b) Called once, returns twice.
(c) Called once, returns never.
(d) None of the above

Ans) (b)

(4) Explain **virtual memory**

Ans) It refers to the virtual address space of a process that is mapped to physical memory space by address translation. Virtual memory gives the illusion that each process owns its independent memory space while actual physical memory is shared by multiple processes.

(5) Explain **Page table**

Ans) It refers to per-process table that maps a virtual memory page to a physical memory page. It is maintained by the kernel to support virtual memory per process.

(6) Below function checks which aspect of the system? Explain.

```
void *dummy (void){
     int i;
     return (void *)&i;   }
int ee209p16(void){
    int i;
    return (dummy() - (void *)&i < 0) ? 1 : 0; }
```

Ans) If stack grows from high to low address, return 1 otherwise it return 0

(7) In **32-bit** system with **8KB** page size, how many bits are needed to represent page offset? How many virtual pages can a process have?

Ans) 13 bits are needed. Process can have 2^{19} = 512 K pages

(8) Explain temporal locality and spatial locality

Ans) Temporal locality means that the recently accessed memory location is likely to be accessed again in the near future (1pt) while spatial locality means that the memory location nearby the recently accessed memory is likely to be used in the near future.

(9) Explain **First fit allocation**

Ans) "First fit" refers to the allocation policy that handles the memory allocation request as soon as the system finds a big enough free memory chunk in the free list that satisfies the request. It is simple and fast in terms of allocation, but it could lead to severe external fragmentation.

(10) What is a fundamental idea of the memory hierarchy?

(a) To create a large amount of storage that is expensive and fast.
(b) To create a small amount of storage that is expensive and slow.
(c) Smaller, faster devices serve as caches for larger, slower devices.
(d) Larger, slower devices serve as caches for smaller, faster devices.

Ans) (c)

2. (18 points) **Assembly Code**

The scores of (1) and (2) are 10 and 8, respectively.

Given the following assembly instruction for the function 'ee209', on a IA32 32-bit architecture derive the C code for the function (You can assume all values are signed integers)

```
ee209p2:
     pushl   %ebp
     movl    %esp, %ebp
     movl    8(%ebp), %eax
     addl    %eax, %eax
     addl    8(%ebp), %eax
     addl    $2, %eax
     subl    12(%ebp), %eax
     popl    %ebp
     ret
```

(1) Please write the code for the function below (make sure to include return and parameter types; the body of the function should only need to be a few C statements at most) and add a comment to each statement you write.

```
int ee209p2(int a, int b) {
int c = a                     // temp variable c initially a
                              // read first argument at 8(%ebp)
 c = c + c;                   // c = 2*a
                              // first add instruction
 c = c + a;                   // c = 3*a
                              // second add instruction
 c = c + 2;                   // c = 3*a + 2
                              // third add instruction
 c = c – b;                   // c = 3*a + 2 – b
                              // subtract 2nd argument at 12(%ebp)
 return c;
}
or, more simply,
int ee209p2(int a, int b)
{ return (3*a + 2 – b); }
```

Name: Student ID:

(2) Consider the following assembly program:

```
1   .section ".rodata"
2   msg:
3        .asciz  "Hi\n"
4        .section ".text"
5        .globl  main
6   main:
7        movl    %esp, %ebp
8        call    getchar
9        cmpl    $'A', %eax
10       jne     skip
11       pushl   $msg
12       call    printf
13       addl    $4, %esp
14  skip:
15      movl    $0, %eax
16      movl    %ebp, %esp
17      ret
```

Choose all the assembly instruction's line number to be relocated by the **linker.** (use line number left of code)

(a) ____________ Ans) Line 8

(b) ____________ Ans) Line 11

(c) ____________ Ans) Line 12

3. (20 points) **Process**

The scores of (1) is 4, and that of (2) and (3) is 8.

(1) How many lines of output does the following function.

(**Give your answer as function of *n***. Assume n≥1.)

```
void foo(int n) {
    int i;
    for (i = 0; i < n; i++) {
        fork();
    }
    printf("Hi there!\n");
}
```

Number of lines of output : ______________________

Ans) 2^n

(2) What are the possible output sequences from the following program.

```
int main() {
   if (fork() == 0) {
      printf("a");
      exit(0);
   }
   else {
      printf("b");
      wait(NULL);
   }
   printf("c");
   exit(0);
}
```

(a) Possible output sequence

Ans) abc, bac

(b) If you comment Line 8 wait(NULL), what are the possible sequence

Ans) abc, bac, bca

Name: Student ID:

(3) What is the output of the following program.

```
pid_t pid;
int counter = 2;
void handler1(int sig) {
    counter = counter - 1;
    printf("%d", counter);
    fflush(stdout);
    exit(0);
}
int main() {
    signal(SIGUSR1, handler1);
    printf("%d", counter);
    fflush(stdout);
    if ((pid = fork()) == 0) {
    while(1) {};
    }
    kill(pid, SIGUSR1);
    wait(NULL);
    counter = counter + 1;
    printf("%d", counter);
    exit(0);
}
```

Output: ______________

Ans) 213

Name: Student ID:

4. (20 points) **Signals**

Each of subproblems takes 5 credits.

Consider the following C program for **printing user inputs**.

```
1   int counter = 0;
2   void handler (int sig) {
3       counter++;
4   }
5
6   int main() {
7      signal(SIGUSR1, handler);
8      signal(SIGUSR2, handler);
9      int parent = getpid();
10     int child = fork();
11     if (child == 0) {
12
13        /* insert code here */
14
15        exit(0);
16     }
17     sleep(1);
18     waitpid(child, NULL, 0); /* wait for the completion of child; you
       don't need to take care of the 2nd and 3rd parameters here*/
19     printf("Received %d USR{1,2} signals\n", counter);
20     return 0;
21  }
```

For each of the following four versions of the above code, list "**ALL the possible outputs**" of this program (codes of each version will be inserted at **Line 13**), assuming that all function and system calls succeed and exit without error. You my also assume no externally issued signals are sent to either process.

(1)

kill (parent, SIGUSR1);
kill (parent, SIGUSR1);

Output: ______________

Ans) 1, 2

(2)

kill (parent, SIGUSR1);
kill (parent, SIGUSR1);
kill (parent, SIGUSR1);

Output: ________________

Ans) 1, 2, 3

(3)

```
kill (parent, SIGUSR1);
kill (parent, SIGUSR2);
```

Output: ________________

Ans) 1, 2

(4)

```
kill (parent, SIGUSR1);
kill (parent, SIGUSR2);
kill (parent, SIGUSR1);
kill (parent, SIGUSR2);
```

Output: _________________

Ans) 1, 2, 3, 4

5. (20 points) **IO management**

Each of subproblems takes 10 credits.

The following problems refer to a file called ***numbers.txt***, with contents the ASCII string ***0123456789***. You may assume calls to read() are atomic with respect to each other. The following file, read_and_print_one.h is compiled with each of the following code files.

```
/* read_and_print_one.h */

#ifndef READ_AND_PRINT_ONE
#define READ_AND_PRINT_ONE
#include <stdio.h>
#include <unistd.h>
static inline void read_and_print_one(int fd) {
    char c;
    read(fd, &c, 1);
    printf("%c", c);
    fflush(stdout);
}
#endif
```

(1) Consider the following code, and show the output.

```
#include "read_and_print_one.h"
#include <stdlib.h>
#include <fcntl.h>
int main() {
   int file1 = open("numbers.txt", O_RDONLY);
   int file2;
   int file3 = open("numbers.txt", O_RDONLY);
   file2 = dup2(file3, file2);
   read_and_print_one(file1);
   read_and_print_one(file2);
   read_and_print_one(file3);
   read_and_print_one(file2);
   read_and_print_one(file1);
   read_and_print_one(file3);
return 0;
}
```

(2) Consider the following code, and show the output as well.

```
#include "read_and_print_one.h"
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
   int file1;
   int file2;
```

```
    int file3;
    int pid;
    file1 = open("numbers.txt", O_RDONLY);
    file3 = open("numbers.txt", O_RDONLY);
    file2 = dup2(file3, file2);
    read_and_print_one(file1);
    read_and_print_one(file2);
    pid = fork();
    if (!pid) {
       read_and_print_one(file3);
       close(file3);
       file3 = open("numbers.txt", O_RDONLY);
       read_and_print_one(file3);
    } else {
       wait(NULL);
       read_and_print_one(file3);
       read_and_print_one(file2);
       read_and_print_one(file1);
    }
    read_and_print_one(file3);
    return 0;
 }
```

Ans) (1) 001213 (2) 001012314.