Fall Semester 2015

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 then not completely clear. Be concise and precise in your answers and state clearly any assumption you may have made. You have 2 hours and 30 minutes to complete your exam. Be wise in managing your time. Good luck.

Make sure that you have ***14 pages.***

Question 1	/ 30
Question 2	/ 20
Question 3	/ 30
Question 4	/ 30
Question 5	/ 35
Question 6	/ 15
Question 7	/ 20
Total	/ 180

Name: Student ID:

1. Basic Concepts (30 points)

A. Explain the memory layout of a program (the role of each section) (5 points)

B. What is context switching and why is it useful? (5 points)

C. What is virtual memory and why is such a concept useful? (5 points)

D. What are temporal and spatial localities and why does locality make caching effective? (5 points)

E. What is “exception”? and what are 4 types of exceptions? (5 points)

F. What is a relocation record, and why are relocation records needed? (5 points)

Name: Student ID:

2. Stack (20 Points)

A. Below is a source code of stack. Fill in the blanks to complete the code. (10 points)

```
#define STACK_MAX 100

struct Stack {

        int     data[STACK_MAX];

        int     size;

};

typedef struct Stack Stack;


int Stack_Top(Stack *S)

{

        if (S->size == 0) {

                fprintf(stderr, "Error: stack empty\n");

                return -1;

        }

        return S->data[S->size-1];

}

void Stack_Init(Stack *S) {   S->size = 0; }

void Stack_Push(Stack *S, int d)
{
        if (_____________________)
           ___________________________________;
        else
            fprintf(stderr, "Error: stack full\n");
}
void Stack_Pop(Stack *S)
{
    if (_____________________)
        fprintf(stderr, "Error: stack empty\n");
    else
        S->size--;
}
```

Name: Student ID:

B. What is the resulting output to **stdout** (standard output) of the code below? (10 points)

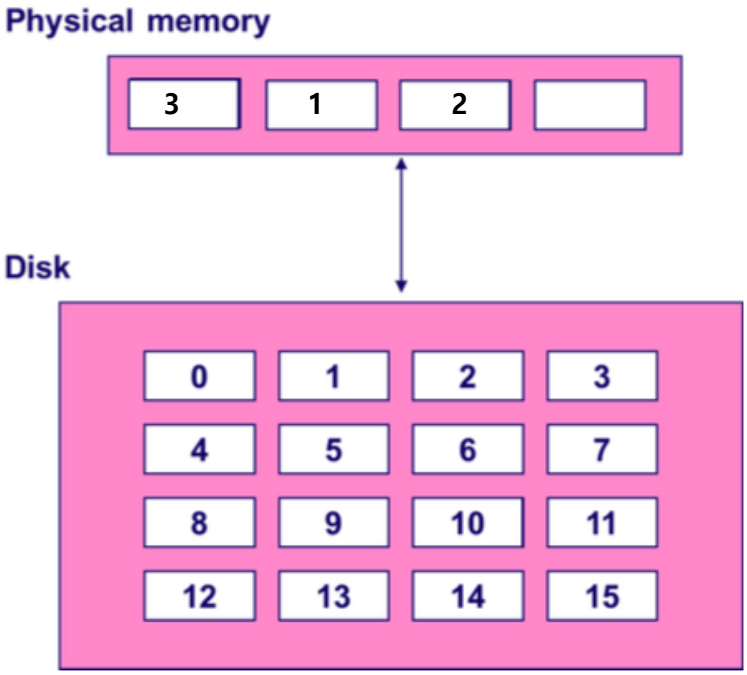
```
void main()
{
    Stack S;

    Stack_Init(&S);

     Stack_Push(&S, 30);
     Stack_Push(&S, 40);

     printf("Top: %d\n", Stack_Top(&S));

     Stack_Pop(&S);
     printf("Top: %d\n", Stack_Top(&S));

     Stack_Pop(&S);
     printf("Top: %d\n", Stack_Top(&S));
}
```

Answer)

Name: Student ID:

3. Memory Hierarchy and Cache Replacement (30 Points)

The figure below shows the memory hierarchy between physical memory and disk. The main memory can hold only four physical pages.



The memory **initially** contains page 3, 1, and 2, where **one slot is empty (see the figure)**. While the application starts to run, it accesses memory pages in the following sequence:

3 0 3 3 0 1 5 3 0 1 5 7 3

Note that "access" can be "read" or "write". Here, if access is "write," it changes the content of a page. Suppose that the system uses LRU (Least Recently Used) or LFU (Least Frequently Used) for cache replacement policy. Thus, we can consider the following 4 cases, depending on the cache replacement policy and the type of access.

Name: Student ID:

A. Access type: **Read**. How many disk reads occur while this program is running? Which page needs to be loaded from disk to memory during the program execution? (12 points)

A.1 Replacement policy: LRU

Disk read occurs _____ time(s) (3 points).

Explanation:

Disk read sequence: _____ (write down the page number) (3 points).

Explanation:

A.2 Replacement policy: LFU

Disk read occurs _____ time(s) (3 points).

Explanation:

Disk read sequence: _____ (write down the page number) (3 points).

Explanation:

B. Access type: **Write**. This causes the memory pages to be dirty when the application process performs writes. Here, "dirty" means that the content of the main memory and the disk in a same page is not consistent. How many disk writes does it have to perform under LRU and LFU? (18 points)
(Hint: When do you have to write a dirty page back to the disk?)

B.1 Replacement policy: LRU

Disk write occurs _____ time(s) (5 points).

Explanation:

Disk write sequence: _____ (write down the page number) (4 points).

Explanation:

B.2 Replacement policy: LFU

Disk write occurs _____ time(s) (5 points).

Explanation:

Disk write sequence: _____ (write down the page number) (4 points).

Explanation:

Name: Student ID:

4. Assembly Language: Code Reading (30 points)

The following assembly code was automatically generated by compiling a very short C function that takes a single parameter of type unsigned int and has a return type of type int. You may assume the formal parameter passed in is always in the range of zero to one hundred.

```
       .file   ''wipe.c''
       .text
.globl wipe
      .type wipe, @function
wipe:
      pushl %ebp
      movl %esp %ebp
      subl $16, %esp
      movl $0, -4(%ebp)
      jmp .L2
.L3:
      movl 8(%ebp), %eax
      andl $1, %eax
      testb %al, %al
      je .L4
      movl -4(%ebp), %eax
      addl 8(%ebp), %eax
      movl %eax, -4(%ebp)
.L4:
      subl   $1, 8(%ebp)
.L2:
      cmpl $0, 8(%ebp)
      jne .L3
      movl -4(%ebp), %eax
      leave
      ret
      .size  wipe, .-wipe
      .ident ''GCC: (GNU) 4.1.2''
      .section .note.GNU-stack,'''',@progbits
```

Here are some hints to help understand this code:
(a) "andl" is the bitwise AND operation.
(b) Register "al" is just the low-order byte of the A register.
(c) "testb" performs a bitwise AND of the two operands, sets the ZF flag to either 1 if the result of the AND is zero, or 0 otherwise, and discards the result, and discards the result. Note that the right next "je" takes jump if the ZF flag is 1.
(d) "leave" releases the stack frame, copying EBP into ESP.

Name: Student ID:

For the following questions, give answers in **"plain"** English, in brief, **"high-level"** descriptions.

A. What is stored in 8(%ebp)? (5 points)

B. What is stored in -4(%ebp)? (5 points)

C. The two instructions right after L2 perform one task. What is it? (5 points)

D. Describe the conditions necessary for the "je" after L3 to fail. (5 points)

E. When "je" fails and the three instructions after it are executed, what is happening? (5 points)

F. Give a high-level summary of what this code does. You should be able to state it in one or two sentences. (5 points)

Name: Student ID:

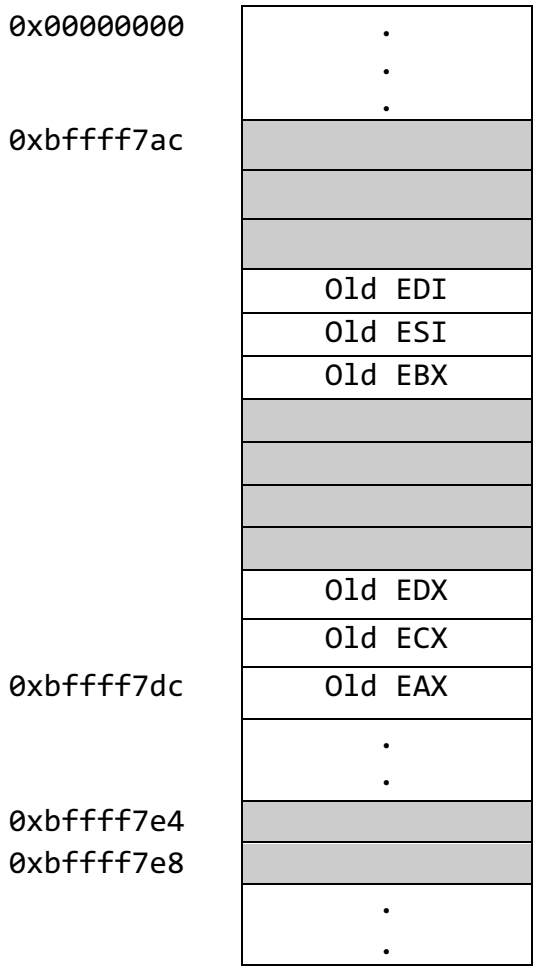
5. Assembly Language: Function Call (35 points)

The following C code describes `swap()` function and the code strip which calls this function.

```
void swap(int *a, int *b)
{
  int tmp;
  int *buf[2];
  tmp = *a;
  *a = *b;
  *b = tmp;
  buf[0] = a + 1;
  buf[1] = b + 5;
                 ←-------
}
```

```
.
.
int a = 3;
int b = 4;
swap(&a, &b);
.
```

Fill in the contents of the shaded area in the stack frame when `swap(&a, &b)` is called and paused at the arrow (←----) above. Write down the exact value using the given information, if possible. Mark the locations of `EBP` and `ESP` as well using arrows.



Name: Student ID:

6. Virtual Memory: Page Fault (15 points)

A. What is page fault? (5 points)

B. Why does implementing malloc() and free() with a single free list (with free blocks of different sizes) lead to a lot of virtual-memory *page faults* (10 points)?

Name: Student ID:

7. Function Pointer (20 points)

Please fill in three blanks so that the resulting out is shown at the end of this problem.

(a) (5 points), (b) (10 points), (c) (5 points)

Note that the function prototype of cos is: `double cos (double)`, which is the same for `tan` and `sin`.

```
#include <math.h>
#include <stdio.h>

void tabulate((a)_____________________________________________________);

int main(void)
{
  double final, increment, initial;
  initial = 0.0; final = 0.5; increment = 0.1;

 (b)_____________________________________ = {cos, sin, tan};

  printf("\n     x       cos(x)
         \n   -------    -------\n");
  tabulate(trigonometric_func[0], initial, final, increment);

  printf("\n     x       sin(x)
         \n   -------    -------\n");
  tabulate(trigonometric_func[1], initial, final, increment);

  printf("\n     x       tan(x)
         \n   -------    -------\n");
  tabulate(trigonometric_func[2], initial, final, increment);

  return 0;
}

void tabulate((c)_____________________________________________________)
{
  double x;
  int i, num_intervals;

  num_intervals = ceil((last - first) / incr);
  for (i = 0; i <= num_intervals; i++) {
    x = first + i * incr;
    printf("%10.5f %10.5f\n", x, (*f)(x));
  }
}
```

Name: Student ID:

Output

x	cos(x)
0.00000	1.00000
0.10000	0.99500
0.20000	0.98007
0.30000	0.95534
0.40000	0.92106
0.50000	0.87758

x	sin(x)
0.00000	0.00000
0.10000	0.09983
0.20000	0.19867
0.30000	0.29552
0.40000	0.38942
0.50000	0.47943

x	tan(x)
0.00000	0.00000
0.10000	0.10033
0.20000	0.20271
0.30000	0.30934
0.40000	0.42279
0.50000	0.54630