82

B.Sc. Engg. CSE 6th Semester 16 February 2023 (Morning)

# ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

**ORGANISATION OF ISLAMIC COOPERATION (OIC)**

**Department of Computer Science and Engineering (CSE)**

**MID SEMESTER EXAMINATION** **SUMMER SEMESTER, 2021-2022**

**DURATION: 1 HOUR 30 MINUTES** **FULL MARKS: 75**

## CSE 4649: Systems Programming

**Programmable calculators are not allowed. Do not write anything on the question paper.**

Answer **all 3 (three)** questions. Figures in the right margin indicate full marks of questions whereas corresponding CO and PO are written within parentheses.

1. a) First 4-bytes of a file as output by the xxd command is shown in Figure 1. 6 (CO1) (PO1)

```
00000000:    7f45 4c46    .ELF
```

**Figure 1:** xxd output for question 1.a)

Explain what would happen if you tried to open this file using a photo viewer. Provide proper reasoning mentioning the cause of observed behavior of the photo viewer.

b) The expected output of the incomplete program shown in Code Snippet 1 is given in Figure 2. Determine the contents of the 6 '___' in Code Snippet 1 to get the expected output. 6 (CO1) (PO1)

```
typedef unsigned char *byte_pointer;
void show_bytes(byte_pointer start, int begin, int end) {
    int i;
    for (i = begin; i < end; i++)
        printf(" %.2x", start[i]);
    printf("\n");
}
int main() {
     long lval = 0x1600410481569;
     char cval = 0x07;
     int ival = 0x69007;
     byte_pointer lvalp = (byte_pointer) &lval;
     byte_pointer cvalp = (byte_pointer) &cval;
     byte_pointer ivalp = (byte_pointer) &ival;
     show_bytes(lvalp, ___, ___); // ①
     show_bytes(cvalp, ___, ___); // ②
     show_bytes(ivalp, ___, ___); // ③
}
```

**Code Snippet 1:** Incomplete code snippet for question 1.b)

```
10 04 60 01
    07
  06 00
```

**Figure 2:** Output of Code Snippet 1 for Question 1. b)

c) A programmer has written the code given in Code Snippet 2 to calculate the sum of all the elements of an array.

```
void fun(int arr[]){
       int sum = 0;
       int len = sizeof(arr) / sizeof(arr[0]);
       for (int i = 0; i < len; i++)
          sum += arr[i];
       printf("%d\n", sum);
}
int main(){
       int arr[] = {1, 2, 3};
       fun(arr);
}
```

**Code Snippet 2:** Code for printing the sum of elements of an array question 1.c)

i. Determine the output of this code. 3 (CO1) (PO1)

ii. The code is buggy. Explain the source of this bug. Fix the code to eliminate the bug so that the code works as expected. You cannot add new lines of code. You can only modify/remove existing lines of code. 10 (CO3) (PO2)

2. a) Consider the program shown in Code Snippet 3. 12 (CO1) (PO1)

```
int main()
{
       unsigned int ui = 1569;
       long int li = -1;
       short int si = -69;
       unsigned char uc = 110;
       if (ui > li){
            printf("Hello\n");
            if (si < ui)
                 printf("World\n");
       }
       if (uc < si)
            printf("!!!\n");
       else
            printf("???\n");
}
```

**Code Snippet 3:** A program for Question 2.a)

What will be the output of the program? Determine the resulting types and values of each of the conditional expressions with proper explanations.

b) The x86-64 LEAVE instruction is used for deallocating stack space before a function returns to its callee. When the instruction executes, it first saves the value of %rbp in %rsp and then restores %rbp from stack. Write a different set of x86-64 assembly instructions that is equivalent to LEAVE. 3 (CO1) (PO1)

c) Write the corresponding x86-64 assembly instructions for the C code given in Code Snippet 4. 6 (CO1) (PO1)

```
long fun(int *ip, long *lp, long b) {
        *lp = *ip;
        b = *lp;
        return b;
}
```

**Code Snippet 4:** C program for Question 2.c)

You must follow function parameter passing convention.

d) Consider variables x and y of type *int* and variable ux of type *unsigned int*. Now, determine whether the following expressions *always* yield 1 or not. 2 × 2 (CO2) (PO2)

i. $\sim x + \sim y + 1 == \sim(x + y)$

ii. $(x >= 0) \,||\, (x < ux)$

3. a) Based on the assembly code showed in the right column of Table 1, fill in the blanks of C source code in the left column of Table 1. 7 (CO1) (PO1)

**Table 1:** C code and corresponding x86-64 assembly

| | | |
|---|---|---|
| `int fun(int arr[], int *a, int *sum,` | `movl` | `4(%rdi), %eax` |
| `int *sub)` | `movl` | `%eax, (%rsi)` |
| `{` | `addl` | `(%rdi), %eax` |
| `    *a = ______;` | `movl` | `%eax, (%rdx)` |
| `    *sum = __________;` | `movl` | `8(%rdi), %esi` |
| `    *sub = ________;` | `subl` | `%eax, %esi` |
| `    return _____;` | `movl` | `%esi, (%rcx)` |
| `}` | `movl` | `(%rdx), %eax` |
| | `ret` | |

b) A lot of bugs arise due to mismatch between signed and unsigned data types. To avoid these bugs, JAVA does not have any unsigned type. But many programming languages (e.g. C) still have support for it. Explain why unsigned type is favored in many cases with an appropriate example in C. 8 (CO1) (PO1)

c) Explain integer conversion rules with code examples in C. You can write multiple code examples if needed. The code examples must demonstrate all the rules in action. 10 (CO1) (PO1)