Assembly Language (C, D, E)
Fall 2022
Assignment-3
Total Marks:90
Submission: Submit on google classroom. Submit each question as separate file. Name of the file will be your roll number and the question number e.g., if your roll number is 21L-1075 and your question number is 1, then your file name should be 21L1075-Q1.asm.
DONOT SHARE YOUR CODE WITH ANYONE.
DO NOT COPY ANYONES CODE.
DO NOT PLACE YOU CODE AT UNSECURE LOCATION.
Deadline 5th November 2022, till midnight. 25% will be deducted for late submissions.

Question 1: [15 marks]

We can represent 3 octal digits in a byte by applying 2–3–3 system. For instance, 11010110_2 is 11010110_2 or 326_8 . Bit 6 and 7 determine the left octal digit (which can never have a value greater than 3), bit 3, 4, and 5 the middle digit, and bit 0, 1, and 2 the right digit. The 16-bit binary value is represented in split octal by applying the 2–3–3 system to the high-order and low-order bytes separately.
Write a subroutine splitOctal which converts a 16-bit binary value, passed to it as parameter, into its equivalent splitOctal value as a string of exactly six characters (where each character is one splitOctal digit). The procedure will have two parameters, passed through stack:
a. The word value
b. the address of the destination string
Ascii code of 0-9 digits is 0x30-0x39. The stack should be emptied after the subroutine ends and none of the register values should be destroyed during the process.

Question 2: [15 marks]

Write a subroutine **minOfMany** which receives a variable number of parameters, and returns the minimum of them (all are positive) in ax register. The last parameter pushed to the subroutine is the number of parameters. Following is an example of how **minOfMany** may be called.
PUSH 5
PUSH 8
PUSH 2
PUSH 3; number of parameters before this one
call **minOfMany**
The subroutine should clear all the parameters off the stack, and also preserve the contents of all registers (except ax in which the minimum is returned).

Question 3: [15 marks]

Write a subroutine 'flip' that creates a flipped image of the upper half of the screen on the lower half such that the top left character appears as the bottom right character in the flipped image (see example below. Left image is original image while Right image is flipped one). Note that the subroutine will override the original lower half.

Abcdef ghijk qrstuv	Abcdef ghijk fedcbA

Question 4: [15 marks]

Write a subroutine ***Trimmer*** that trims all the occurrences of string2 from string1 printed on video screen. The subroutine takes the following parameters (in same order):

i. Row number where string 1 is printed on screen
ii. Column number where string 1 is printed on screen
iii. Length of string 1
iv. Offset (i.e. address) of string 2 in data segment

Use **ONLY** string instructions. No credit for doing the question without string instructions. You can safely assume that String2 length is less than String1 length.

Example:
String 1 printed on screen is:
The alignment-check exception can also be used by interpreters to flag some pointers as special by misaligning the pointer.
String2 db: 'align',0
Your subroutines updates string 1 as:
The ment-check exception can also be used by interpreters to flag some pointers as special by mising the pointer.

Question 5: [15 marks]

Write an assembly program that takes a 4-digit decimal number as input from user and calculates whether the number is a happy number or unhappy. If the number is a happy number, display 'Happy' on the screen else display 'UnHappy'. A happy number is defined by the following process:

Starting with the given number, replace that number by the sum of the squares of its digits. Repeat the process on the replaced number until the number either equals 1 within 256 iterations of the process. The number for which this process ends in 1 within 256 iterations is called a happy number, otherwise it is called an unhappy number.

Question 6: [15 marks]

Consider your video screen is divided in four equal parts, each part with 12 rows and 40 columns. (skip the last row of the screen) Whenever a key is hit on keyboard, memory area 1 is swapped with memory area 4. Write complete program using string instructions.

Memory area 1	Memory area 2
Memory area 3	Memory area 4