ASSEMBLY LANGUAGE PROJECTS 2018

Contents

Common Requirements

- Your program must be divided into **PROCs**, each of which is responsible for one and only one functionality. For example,
    - ReadArray: reads an array
    - SortArray: sorts a given array.
    - ViewImage: views an image.
- Your PROCs should be transparent for input parameters, do not forget the **USES** operator.
- Don't use any hardcoded values, instead use **constants** and operators
    - For example: Fetch array length by LENGTHOF operator… etc.
- The bonus items will not be counted unless the original project is complete.
- If the GUI is necessary for your project; **here** you are a link to a simple tutorial for how to link Assembly code with high-level language.
- Keep your code **clean**, follow a specific **convention**, and choose **meaningful** identifiers (variables and procedure names).
- All projects must be submitted with a **printed documentation**. In your documentation, draw the **flow chart** of your logic and the **procedures hierarchy** (refer to chapter 5).
- [Code like a professional]: Your code **must** be well **documented**; you should use commenting style like this:

```
;-----------------------------------------------
;Calculates: Sum of an integer array
;Receives: ESI Contains the offset of the Array
;   ECX Contains the length of the Array
;Returns:  EAX contains Array Sum
;-----------------------------------------------
SumArr PROC
```

1. Geometric Drawings

Write an Assembly program that reads lines coordinates from a file, your program should search among these lines for “Rectangles”, “Squares” and “Triangles”. For each detected shape, your program shall draw it on the console given the coordinates.

- **Details**
  Given a set of line coordinates, you should check if the construct one of the geometric shapes (Rectangles, squares, triangles). You can determine a shape only by intersected points. For triangle example given three lines, they construct tringle if and only if,
  1. line1 intersects with line2
  2. line2 intersects with line3
  3. line3 intersects with line1
- **Input**
  txt file contains several lines, each line gives 2 points for a geometric line, check **lines.txt**
- **Output**

Your program has 2 modes (text mode and drawing mode)

1. **Text mode**, for each shape type you should display
   - The number of that shape found.
   - Coordinates of each shape.

   For example:

   Triangles
   - *1 object detected*
   - *Coordinates:*
   - *(3,5) (2,2) (4,2)*

   Squares
   - *2 objects detected*
   - *Coordinates:*
   - ….
2. **Drawing mode**, for each shape you should draw its boundaries using * character

```
   *                    * * * * *
  * *                   *     *
 *   *                  * * * * *
* * * *
```

- **Bonus**
  Support GUI
- **Hints**
  You can read files and go to specific coordinates on the console using built-in functions in Irvine.
  Check the book, sections 5.4 and 11.1.
- **Team members:** 4 – 5 members.

2. Hospital Management System

Write an assembly program that handles the basic tasks of a Hospital Management System.

This project mainly uses file handling to perform basic operations like add, edit, search and delete records where each record will be stored in a separate line and columns are separated by a delimiter.

Functionality:

You're asked to implement the following functions in assembly:

Function	Parameters
Add_new_patient_record	Name, age, gender, disease description.
Search_patient_record	Full Name
Edit_patient_record	Full Name
List_of_patients	
SaveDatabase	File Name, DB Key
OpenDatabase	File Name, DB Key
Delete_Patient	Full Name

Functions Description

1. **Add_new_patient_record:** In this feature, user can add a new patient record choosing between O.P.D service and Emergency service. The information given for each service should be stored in a separate file. The information required are the same for both emergency and O.P.D service.

2. **Search_patient_record:** User can search the DB via the patient's full name. All the information corresponding to the respective patient should be displayed. These include the ones provided while adding a new patient record. If the patient's full name does not exist, the program should display a message saying that no records are available.

3. **List_of_patients:** users can list patient records by choosing any of the three options listed below:
    - Records of patients in alphabetical order.
    - Records of Emergency patients
    - Records of O.P.D. patients
4. **SaveDatabase:** When saving a database, each data item should be written encrypted with a user key (a single byte). Hint: Use XOR encryption which xor each byte of the data item with the user key.

5. **OpenDatabase:** When opening a database**,** the user can work on data previously saved and do all other functions on them. The user should enter the database key to be able to decrypt & view its content.

6. **Delete_Patient:** this function should delete patient's record from file.

Input

The application should display a menu asking the user for the operation he wants to perform, asks for data then call the corresponding function in the assembly .dll In case of error it should notify the user.

Bonus:

- GUI implementation instead of using console application.
- Adding any non-trivial Functions.

Hint

You can read and write files using built-in functions in Irvine. Check the book, section 11.1.

Team Members: 4 – 5 members

3. Image Processing Package

Our Image Processing Package includes two tasks:

1. Sobel Edge Detector.
2. Histogram Equalization.

What is a computer image?

A computer image is a digitized version of a picture taken with a capturing device like the camera or it is the electronic visual representation of pictures which is stored in a computer. As we all know that the computer understands 1's and 0's only, so everything you see on the computer screen is mapped to binary number(s) and of course a computer image is mapped to some binary numbers. A computer image stored as a 2D array of pixels, each pixel consists of three color channels Red, Green and Blue (RGB) each color channel takes a value between 0 – 255 inclusive. Mixing decimal values of these color channels gives us the different color that we see on the screen.

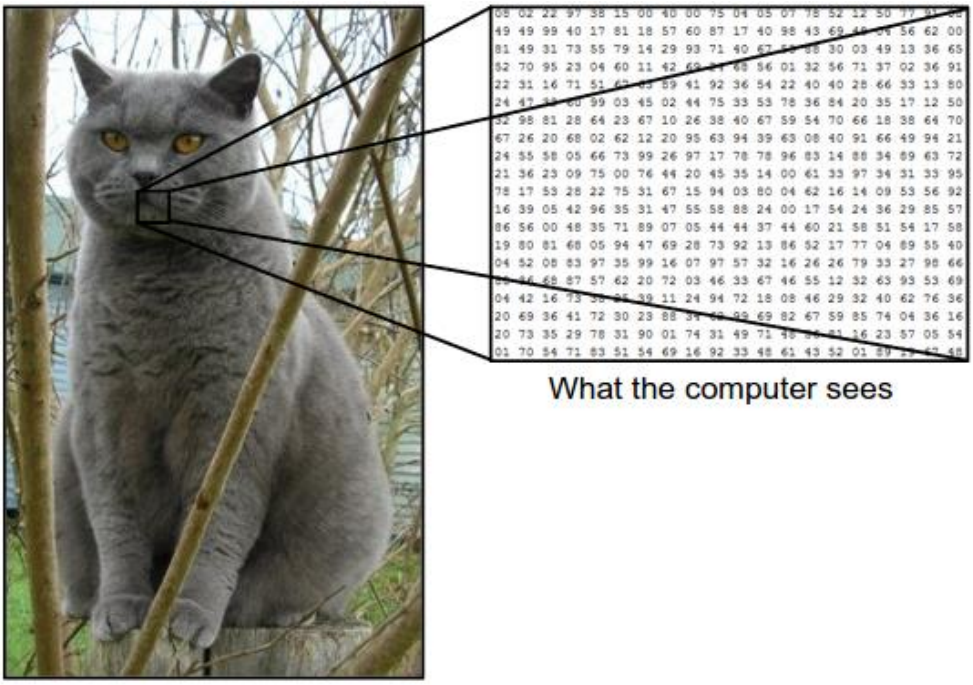


Figure 3-1 A portion of an image showing what the computer sees

1. Sobel Edge Detector

Sobel operator is used to extract edges from a given image. Extracting edges from an image is useful to extract some important features that can be used later in many computer vision related tasks such as: detecting license plate, face detection, object detection and so on. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. If we define A as the source image, and G_x and G_y are two images which at each point contain the vertical and horizontal derivative approximations respectively, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

where * here denotes the 2-dimensional signal processing convolution operation.

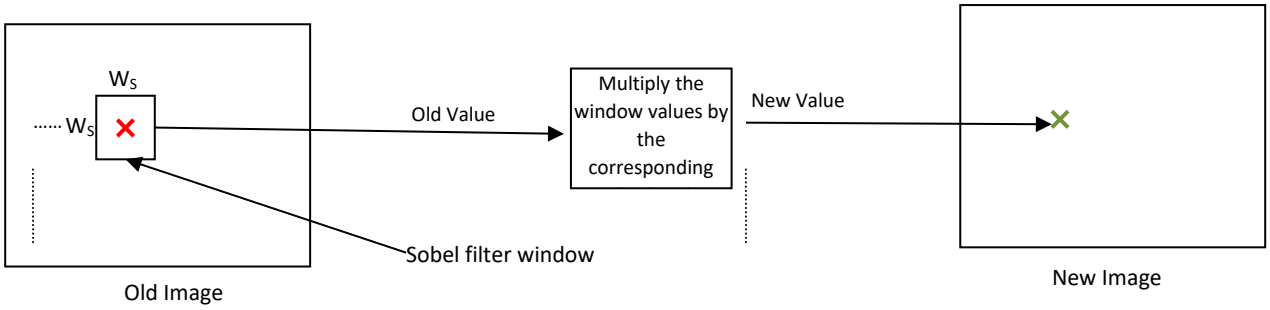


Figure 1: demonstration of the convolution process

You are required to implement Sobel operator filter in both G_x *and* G_y directions.

Sobel operator works as follows:

1. Given a grayscale image, pad the borders of that image with zeros. Check this link.
2. Filter the image using a sliding window contains some constant values.
3. For more information surf the internet and use your self-study skills (check the references section below).

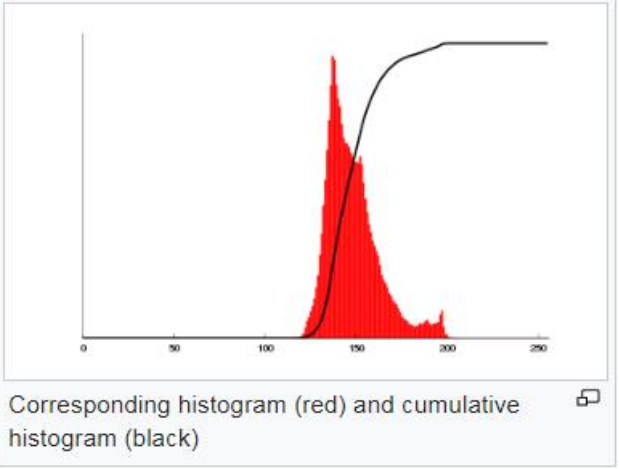
2. Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

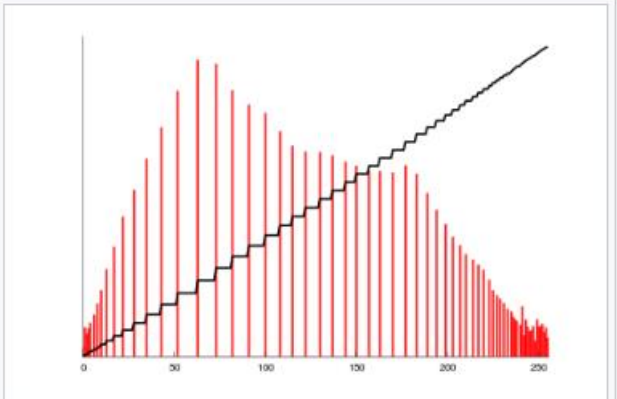
In scientific imaging where spatial correlation is more important than intensity of signal (such as separating DNA fragments of quantized length), the small signal to noise ratio usually hampers visual detection.

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Steps:

1. Compute the histogram of the image and store the result for each color channel (R, G and B) in an array. (click to view reference).
2. Find the cumulative sum for each of the 3 resultant arrays from the previous step and round the values. The new array after round will be computed according to the following steps:
    a. Call the original array before rounding X and the after rounding Y.
    b. Initialize the first element of X: Y[0] = X[0]
    c. For all elements of Y except Y[0], calculate their values according to this formula:

$$Y[i] = \frac{X[i] + X[i-1]}{M * N} * 255$$

$$Where\ i: 0 \rightarrow 255, M\ and\ N\ are\ the\ width\ and\ height\ of\ the\ image\ respectively$$

d. Return Y.

3. Update each the color intensity for each color channel with the corresponding value from the previous step.

Implementation Notes

Unfortunately, **Irvine** library doesn't support images, so we have to find a method to read pixel values from an image. We will integrate our assembly project's DLL in a C# project. The C# project will read an image and dump it into a decimal array (1D Array), then it will pass the array to the assembly functions to manipulate these pixel values.

You'll deliver two modules (C# & Assembly Project). The first module is a GUI desktop application written in C#, which has the following functionalities:

1- Open a bitmap image from the disk.
2- Save a bitmap to disk.
3- Convert an array of pixel values of an image to bitmap then view it.
4- Convert a bitmap image into an integer array.
5- The required GUI components to find the Sobel edge image in G_x.
6- The required GUI components to find the Sobel edge image in G_y.
7- The required GUI components to apply histogram equalization to the given image, then display it.
8- The template contains an example function implemented for you in both assembly and C# (Invert image function). This function is just an example to demonstrate how images are sent and received between C# and assembly. The inverted image will look like something like this image.

The second module is the assembly library (DLL) with the following functionalities:

1- A function that takes a 1D integer array of pixel values, then returns the same array after filtering it using Sobel operator in a certain direction.
2- A function that takes a 1D integer array of pixel values, then returns the same array after equalizing its histogram.
3- Integrate that module with the C# desktop application project and use the required functionalities by calling the procedures from the DLL class library.

Bonus:

To be announced

Team Members: 4 – 5 members

References

1- **Project template is available here. You can either clone it or download it as a zip file. If you have a github account, do not fork this project to your account as everyone else will be able to watch your code!.**
2- **Edge detection and convolution process tutorials.**
3- **Best tutorials on image kernels.**
4- **Image convolution.**

4. ASM-Networking

Our ASM-Networking includes three main components:

1) ASM DLL, an assembly program, is responsible for encrypting and decrypting a message using AES algorithm.
2) Client Side which is responsible for sending an encrypted message and key to server side.
3) Server Side which is responsible for receiving an encrypted message and key.

This project reads a text and an encryption key from the user (using GUI in C#), sends that text and key to your assembly program to encrypt it using AES algorithm then send it to server side. The server side receives the encrypted message and key and sends them to your assembly program to decrypt it. Then, it will show the decrypted text for the end user (using GUI in C#).

Details

The **Advanced Encryption Standard (AES),** also known by its original name Rijndael is a specification for the encryption of electronic data. AES operates on a 4 × 4 column-major order array of bytes (128-bit) as a plain text. Encryption consists of 10 rounds of processing for 128-bit keys. The key is a 4 x 4 column major order of bytes. Except for the last round in each case, all other rounds are identical as shown in the figure 1. Each round includes 4 steps that will be discussed in details.

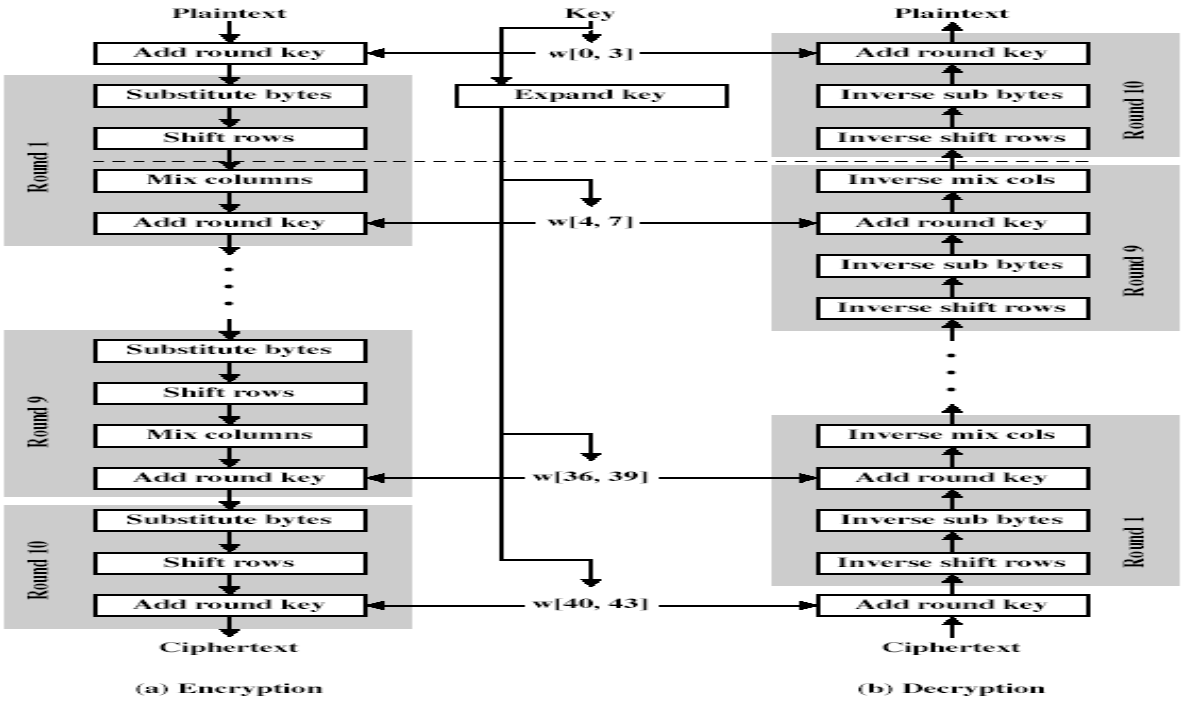


Figure 1. AES Algorithm

Input: 1 byte array (4 X 4 plain text matrix).

1 byte array (4 X 4 key matrix).

Output: 1 byte array (4 X 4 cipher text matrix).

Add Round Key step:

It's a simple XOR between the plain text and the round key.

It's the same step in encryption and decryption.

Substitute Bytes step:

Each byte is replaced with another according to a lookup table as follows.

Use the following table in encryption.

AES S-Box. The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value 0x9a is converted into 0xb8.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

And the following table for decryption.

Inverse S-Box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Shift Rows Step:

It's a transposition step where the last three rows of the state (matrix) are rotated left in encryption and right in decryption a certain number of **bytes** as shown in figure 2.

Rotate Left each row based on row index

Row Index	Number of Rotate Left
Row Index 0	0 Rotate left
Row Index 1	1 Rotate left
Row Index 2	2 Rotate left
Row Index 3	3 Rotate left

Figure 2. Shift rows

Mix Columns Step:

It's a multiplication step, multiplying the output matrix from shift rows by a fixed matrix using advanced method for multiplication. Use the following matrix in encryption.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Figure 3. Multiplication matrix for mix columns encryption

And use the following matrix in decryption.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

Figure 4. Multiplication matrix for mix columns decryption

Advanced Multiplication:

- It depends on **shifting** and **XOR** steps.
- The **addition** is done using **XOR**.
- The details of the steps are described as follows:
  Generally, after shifting the byte to left by one, check the output bit if it's one then XOR with 0x1B then use this value as the output of multiplication. If it's zero then use this shifted value as the output of multiplication.

  - Multiply by **01**, means use the same value of byte.
  - Multiply by **02**, means shift left one time and check the CF then use the output value as result of multiplication.

  - Multiply by **03**, is divided into two multiplication:
    - Multiply by one and keep the value in X0.
    - Take results and multiply it by two. To multiply by two use the above description and keep value in X1.
    - XOR the result of multiplication by one and result of multiplication by two (XOR X0 and X1) and this is the output of multiplication..

  - Multiply by **09**, is done as following:
    - Multiply by 1 and keep the value in X0.
    - Multiply X0 by 02 and keep value in X1.
    - Multiply X1 by 02 and keep value in X2.
    - Multiply X2 by 02 and keep value in X3.

- XOR it X0 with X3 and this is the output of multiplication.

➢ Multiply by **0B**, is done as following:
- Multiply by 1 and keep the value in X0.
- Multiply X0 by 02 and keep the value in X1.
- Multiply X1 by 02 and keep the value in X2.
- Multiply X2 by 02 and keep the value in X3.
- XOR (X3 , X1, X0) and this is the output of multiplication.

➢ Multiply by **0D**, is done as following:
- Multiply by 1 and keep the value in X0.
- Multiply X0 by 02 and keep the value in X1.
- Multiply X1 by 02 and keep the value in X2.
- Multiply X2 by 02 and keep the value in X3.
- XOR (X3, X2, X0) and this is the output of multiplication.

➢ Multiply by **0E**, is done as following:
- Multiply by 1 and keep the value in X0.
- Multiply X0 by 02 and keep the value in X1.
- Multiply X1 by 02 and keep the value in X2.
- Multiply X2 by 02 and keep the value in X3.
- XOR (X3, X2, X1) and this is the output of multiplication.

Example of Mix Columns:

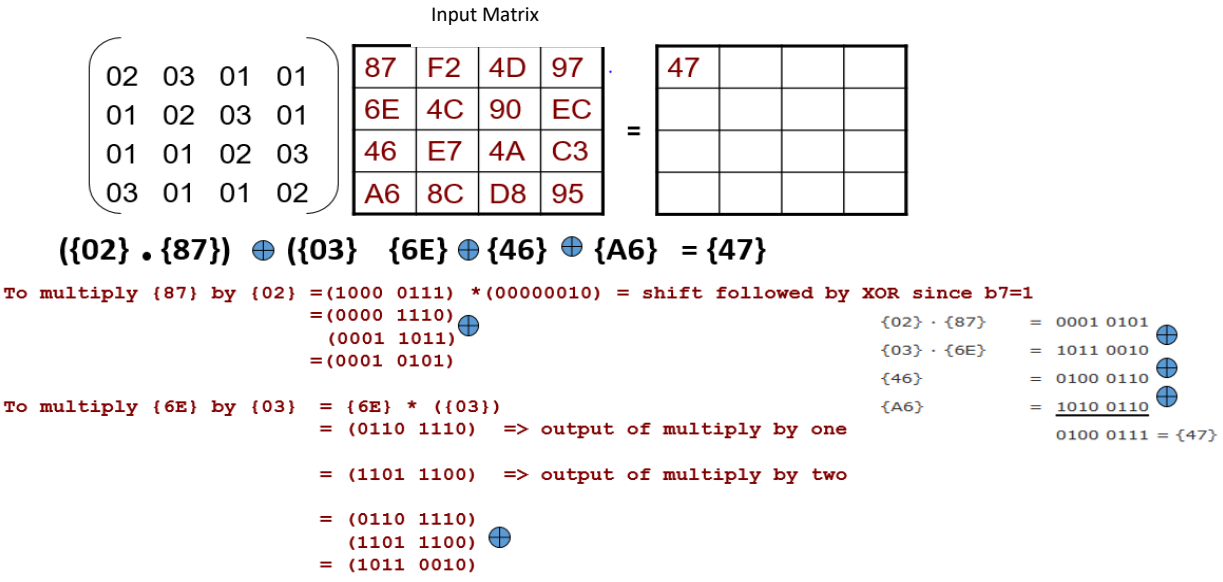


Figure 5. Example of Mix Columns

Generation of Key:

Given matrix of input key, you'll generate 10 keys for the 10 rounds. Each Round key depends the previous round key which means key for round 1 depends on the input key and key for round 2 depends on the key of round 1 and so on. The details of generation will be discussed in the following steps:

Each column W_i in the new key matrix is output of XORing W_{i-1} and W_{i-4} where W_{i-1} is the previous column in same key matrix, and W_{i-4} is the same column in previous key matrix.

Except the first column, as the previous column is in the previous round key. It will pass by a function that contains the following steps:

1) Rotate one byte up to down.
2) Substitute from S-box.
3) XOR with round constant column (choose column based on which round you're generating the key for).
4) XOR with the same column in the previous key.

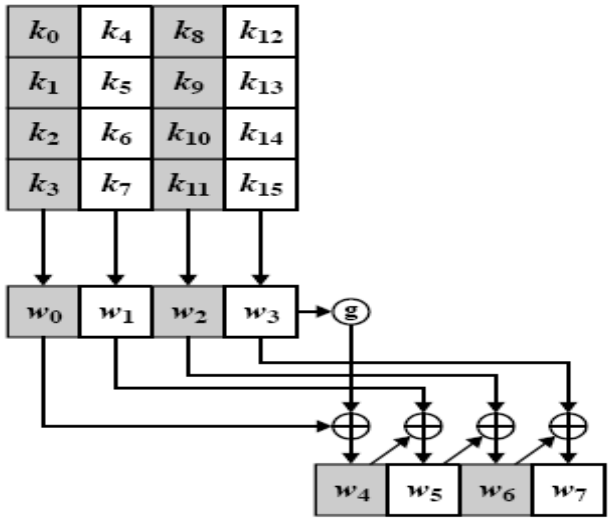


Figure 6. Generation of round key

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Figure 7. Round constant table

Example to generate W_4 as shown in the previous figure:

1) Rotate W_3 one byte from up to down.
2) Substitute from S-box.
3) XOR with first column in round constant as we generate the key for first round.
4) XOR with W_0

To use the Template:

1) Call your assembly function instead of the not implemented exception in client and server forms.
2) Run the server side first, by right click on the project, select debug, select run new instance.
3) On server side form, click the start button.
4) Run the client side, by right click on the project, select debug, select run new instance and enter new message and key.
5) The message and key should be 16 character each and click send.
6) Check the output in Server side form.

Bonus

- Support Socket programming using Assembly only.
- Implement steganography for embedding the encrypted message in an image and extract an embedded text from image using Assembly only, the image would be send and received through the C# application.

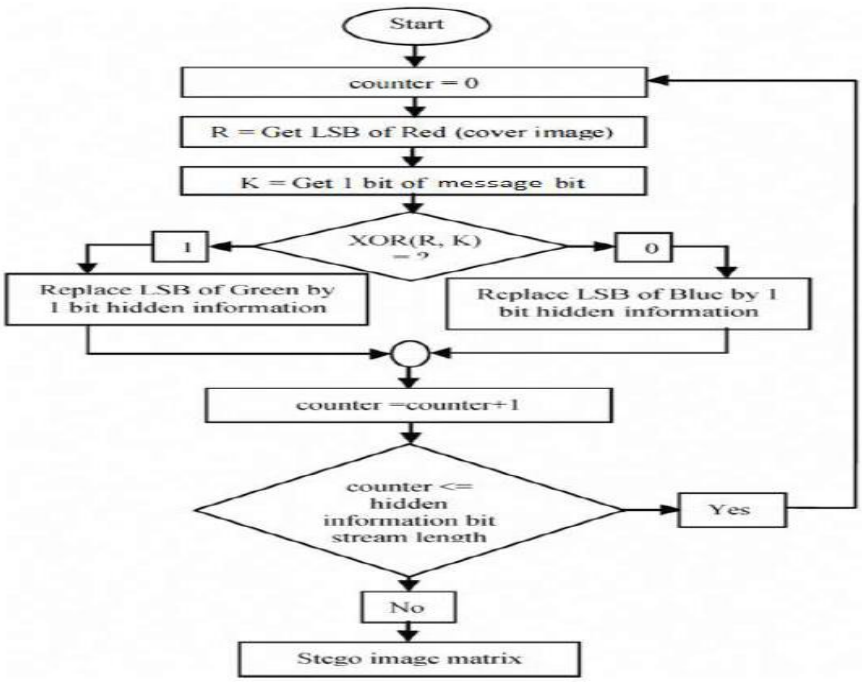


Figure 8. Flowchart of steganography

- **Hints**
  NASM supports socket programming on Linux.
  Substitution in key generation is done using S-box only not Inverse S-box in encryption and decryption.

- **Team members:** 5 members.

5. Text Editor

Write assembly program to design text editor where one can write , edit text and saving text into file. It supports also the following features:

1) Changing text color
2) Search for a word into text (with search options: “match case”, “whole word”) and outputs a list of lines where the word is found. Otherwise an error message should appear
3) Read from file “file path” and browse file in command prompt

Details

Using windows API, call the functions that would create a window and create a text inside it and respond to mouse and key events.

Refer to sections 11.1 and 11.2 in text book

Ouput

Text editor in MS windows with the mentioned features

Bonus

Support GUI (open file from windows)

Team Members: *4-5 members*

6. E-Payment System

An e-payment system is a way of making transactions or paying for goods and services through an electronic medium, without the use of checks or cash. It's also called an electronic payment system or online payment system. We will be focusing on E-payment by credit card.

Requirements:

You will write an assembly program that takes credit card info and payment amount and checks if credit card number and its info are valid and if the payment amount is available in balance it goes through with the transaction.

Input/Output:

Prompt the user with the following choices:

1. Add new credit card to the database:
    a. Input:
        i. Credit Card Number (number must have between 13 and 16 digits. It must start with: 4 for Visa cards and 5 for Master cards Ex: 4388576018402626)
        ii. Expiry date (in the form MM/YY EX: 10/18)
        iii. CVV (3 digit number)
        iv. Card Holder Name
        v. Balance
    b. Output:
        i. If credit card number is valid and name is not already in database, print the message "Credit card successfully added".
        ii. If credit card number is not valid or name is not already in database, print the message "Adding Credit card Failed" mentioning the failure reason.
2. Delete credit card from database:
    a. Input:
        i. User chooses to search by credit card number or card holder name.
    b. Output:
        i. If credit card found, print the message "Credit Card Deleted Successfully".
        ii. If credit card not found, print the message "Credit Card not fount".
3. Top-up credit card balance in the database:
    a. Input:
        i. User chooses to search by credit card number or card holder name.
        ii. Top-up amount.
    b. Output:
        i. If credit card found, print the message "Credit Card balance Toped-up successfully".
        ii. If credit card not found, print the message "Credit Card not fount".

4- Pay by credit card:

- Input:
    iii. Credit Card Number (number must have between 13 and 16 digits. It must start with: 4 for Visa cards and 5 for Master cards Ex: 4388576018402626)
    iv. Expiry date (in the form MM/YY EX: 10/18)
    v. CVV (3 digit number)

vi. Card Holder Name
vii. Payment Amount

c. Output:
   i. If the transaction is valid, print the message "Your transaction is complete. Your payment has been successfully processed".
   ii. If the transaction is not valid, print the message "Your transaction failed. No amount was debit from your balance" and mention the reason why it failed.

Steps:

1. Read Input stated above, Then:

- Adding new credit card :

1. checking credit card number validity:
   a. Credit card numbers follow certain patterns.
   b. A credit card number must have between 13 and 16 digits. It must start with:
      i. 4 for Visa cards
      ii. 5 for Master cards
   c. The problem can be solved by using **Luhn algorithm**:
      a. Luhn check or the Mod 10 check, which can be described as follows (for illustration, consider the card number 4388576018402626):
   d. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number (like for 12:1+2, 18=1+8).
   e. Now add all single-digit numbers from Step a.
      a. 4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37
   f. Add all digits in the odd places from right to left in the card number.
      a. 6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38
   g. Sum the results from Step b and Step c.
      a. 37 + 38 = 75
   h. If the result from Step d is divisible by 10, the card number is valid; otherwise, it is invalid.
   - If Credit Card Number is not valid display error message in the output above.
2. Searching for name in file Named **Credit_Card_information.txt** if not found add record else display error message in the output above.

- Deleting existing credit card:
  - Searching for name or credit card number in file Named **Credit_Card_information.txt** if found delete it and display success message else display error message.
- Editing existing credit card:
  - Searching for name or credit card number in file Named **Credit_Card_information.txt** if found increment the balance by entered amount and display success message else display error message.
- Paying by credit card:
  a. Checking credit card number as in adding step.
  b. Access file Named **Credit_Card_information.txt** and find the record for the input credit card number:

1. The file has the data in the following order (Credit Card number,ExpiryDate,CVV,Name,Balance) values are separated by commas, Name is separated by space and each credit card record is in a separate line.
2. If credit card number not found display error message in the output above.

c. Validate Input data (Expiry Date, CVV, and Name) against found record for credit card.
   1. If any of the input data is not display error message in the output above.

d. Check balance against Input Payment Amount, if transaction is valid, Then Subtract Payment amount from balance in the credit card record in the file **Credit_Card_information.txt.**
   1. If no enough balance display error message in the output above.

e. Finally, print the valid transaction message.

Bonus:

1. GUI.
2. Log both valid and invalid transactions in separate files.

Team:

- 4-5 members

7. Hangman

Game Story

Irvine was taken as a prisoner in an unknown computer island. We could only communicate to the kidnappers by through a very low programming language, Assembly!

As a student learning assembly course, you dedicated your time and effort to save the poor Irvine from death. The first step is to build a communication with the kidnappers, so you decide to build a hangman game.

The game will allow the player (students) to play and try to win, otherwise, the kidnappers will hang and kill Irvine. This will result in a disaster where all assembly codes will get destroyed, thereafter, all higher level languages will vanish. This is a very important project, it will not only save Irvine, but also all programming languages!

Game Details

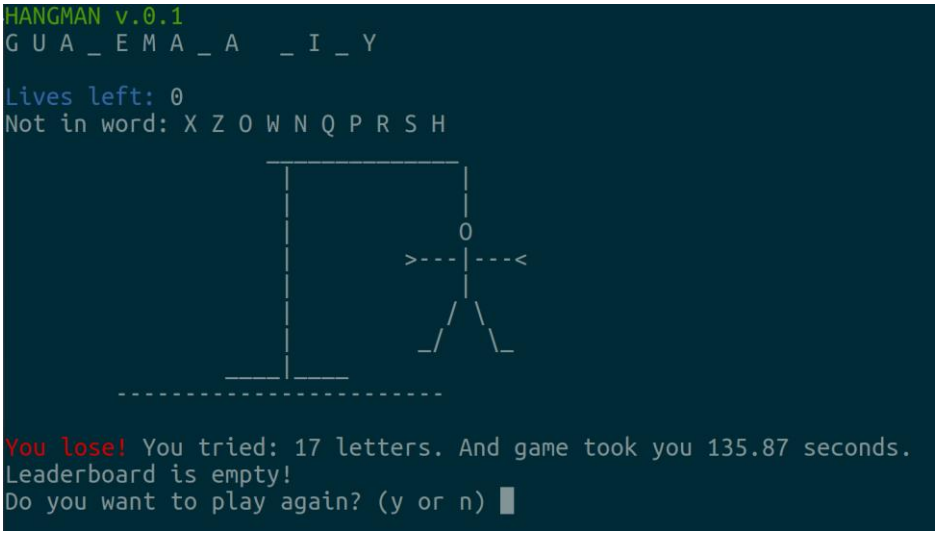
The game is about guessing some sentences correct and gives you 5 trials (where 5 changes based on the level and there are 5 levels). For each trial, you guess a letter, if the letter is correct, the kidnappers show you the location of these letters in the sentence. If the letter is incorrect (i.e. this letter does not exist in the sentence), then the drawing of the hangman is built one step at a time: the rope, the head, the upper body and hands, the lower body and lastly the legs. For a more difficult level, there are only 2 trials (so the drawing shall be the upper and then the lower part of the body)

Game Features

- Initial **Menu** that contains the following buttons with a decoration similar to the image below (make your own decoration style, but use colors)
    - Play
    - Hall of Fame (shows the previous players and their scores)
    - Credits (shows information about the developers)
    - Exit

- Each player when selecting Play a new game is asked to enter his/her name

- The Game shall be similar (not mandatory exact) as in the following image:



- Initially, a line with underscores is shown to represent the words.
  As an example, if the sentence is "I love Assembly", then the following line is initially shown to the user "_ _ _ _ _ _ _ _ _ _ _" where each word is separated by 2 spaces and each letter is separated by 1 space

- For each correctly guessed letter, the word is displayed in its exact location (instead of the underscore)

- For each incorrectly guessed letter, a part of the hangman picture is drawn in the middle of the screen (or to the left/right most of the screen, whatever you like)

- You shall show the number of trials and time taken to play the game. The time taken will be the user score in the Hall of Fame Menu.

- The Hall of Fame screen will read from a file the previous players scores. Only those players who passed the game are saved in this file.

- Develop at least 2 levels and select 1 sentence out of 20 sentences in a random manner. The easy level will allow up to 5 mistakes while the Hard level will allow only 2 mistakes. The hall of fame screen will show only the top 5.

Team:

- 4-5 members