ASSEMBLY LAB PROJECT

PROJECT NAME: HANGMAN GAME

SUBMITTED TO:	MR. M. TAWFIQUL ISLAM
	LECTURER
	DR. SAIFUDDIN MD. TAREEQ
	PROFFESSOR
SUBMITTED BY:	SABRINA ZAMAN ISHITA
ROLL:	48
	AMIFA RAJ
ROLL:	28

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF DHAKA

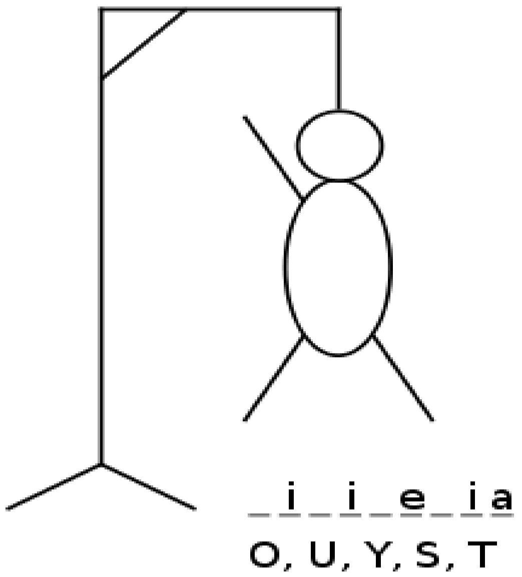
INDEX

1. Game Overview:

Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word, phrase or sentence and the other tries to guess it by suggesting letters or numbers.

The word to guess is represented by a row of dashes, giving the number of letters, numbers and category. If the guessing player suggests a letter or number which occurs in the word, the other player writes it in all its correct positions. If the suggested letter or number does not occur in the word, the other player draws one element of the hanged man stick figure as a tally mark. The game is over when:

1. The guessing player completes the word, or guesses the whole word correctly
2. The other player completes the diagram:



The exact nature of the diagram differs; some players draw the gallows before play and draw parts of the man's body (traditionally the head, then the torso, then the arms and legs one by one). Some players begin with no diagram at all, and drawing the individual elements of the gallows as part of the game, effectively giving the guessing players more chances.

Let's see how we can make this game by using assembly language.

2. Purpose of the Game:

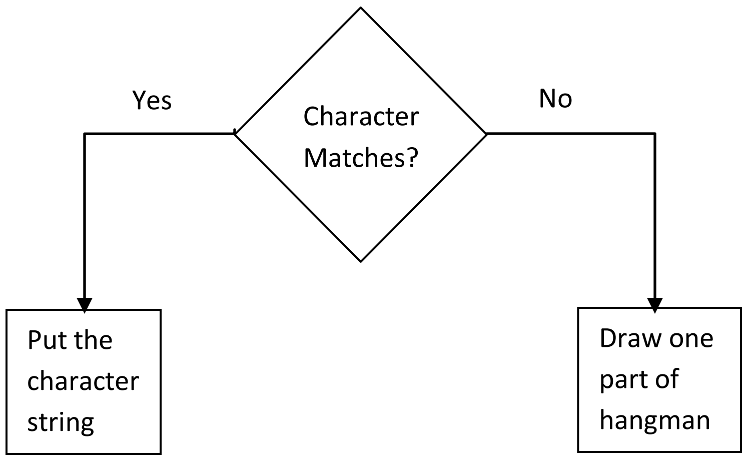
The purpose of making this game is to improve our efficiency in assembly language. Again by making this game we get to know graphical work in this language. And we use various interrupt function. Thus we get a vivid idea of the effective and proper use of assembly language.

3. Winning Condition:

As we have mentioned earlier that there are two conditions to determine that the game is over –

1. The guessing player completes the word, or guesses the whole word correctly
2. The other player completes the diagram

That means if the guessing letter matches to the word then the player will put the character in that position of a string where it is in the main string . If all character matches the second player will win. And if the guessing letter does not match then the player will draw one part of hangman's body. When the hangman's full body is drawn then the second player will lose. It can be shown in the following way:



Thus this game is played and won.

4. Main Logic:

To make this game in assembly we use two logics. As we saw in the winning condition there are two possible conditions in this game. So we have to check these conditions first. As a result we use two condition checks in the code.

At first we count the number of characters in the string that the second player has to guess that means we store the length of the string. We keep another count to see how many guessing characters are matched. After guessing one character if it matches any of the characters in the string we increase the count of the successful guessing character and if it does not then the count remain same. After guessing several characters, when it becomes equal to the length of the string the second player wins.

Now here comes the losing condition. We know that there are three parts in hangman's body (head, body, two hands, two legs, and the rope). If a guessing character does not match to the string we draw one part of the hangman and keep a count that how many parts have been drawn. Gradually when it becomes seven the game ends.

So when the drawing of a hangman's full body is completed before the number of guessing characters become equal to the length of the string, the second player loses the game and vice versa. This is the main logic we use in our game.

5. Game Designing & Planning:

Now we come to the implementation part. There are several parts in the code to implement the whole game. We have to both work in the logic part and graphical part. To demonstrate the whole work we have done in our code we will show it in different parts below.

5.1 Input String:

As the first task is to give a string to the second player to guess we have to take a input of that string. So take a input from the first player declare a string named 'hang' in data segment. For a note we take all types of input from keyboard. To give data to that string we had to use INT 16H function 0 as INT 16H provides keyboard services and AH=0 reads keystroke function. So keep taking input before 'enter' is pressed which we check by comparing AH with 1CH which indicate the enter key.

```
NAME_IN:
        CMP AH, 1CH
        JE FINISH_N
        MOV HANG[SI], AL
        INC SI
       MOV AH, 0
       INT 16H
JMP NAME_IN
```

Fig 1: Taking *string input in NAME_IN label*

5.2 Character Input:

As we have mentioned before that we take all kind of input from keyboard. We take character input in the same way we take string input. But here we will check two conditions before taking each character input. The conditions are if the character matches and gradually the matching characters of input equals to the length or the full body of hangman is formed we will not take more character input.

```
NEXT_:CMP
        CX, LEN
        JE END_
        CMP DX, 7
        JE END_
        MOV SI, 0
        MOV BL, 0
        MOV AH, 0
        INT 16H
```

Fig.2: *Taking character input*

5.3 Print String:

Now we will work on display method. We have to show the characters that have been matched in the screen. We showed it exactly in the same position it was in the given string. To display the characters first we select display mode. We declared a string already named 'Inp' in which will show the characters. Whenever a character matches with the string we push it in the string and display that string. To select display mode we use INT 10H. But before that we determine the position where we want to display the string by declaring the row and column. And we did it by keeping AH=1 which indicates cursor function.

```
INPUT:
        PUSH_REG
        MOV AH,2 ; MOVE CURSOR FUNCTION
        MOV DH,15 ; ROW
        MOV DL,8 ; COL
        MOV BH,0 ; PAGE
        INT 10H ; DISPLAY INTERRUPT
        MOV DX, OFFSET INP
        MOV AH,9
        INT 21H
```

Fig 3: *Display the string*

5.4. Draw Hangman:

Now we will discuss about the most interesting part of the game – drawing of the hangman. We will draw it in several parts as it has seven parts. So we have to draw each part individually. With the mismatch of each character the parts of hangman are shown serially.

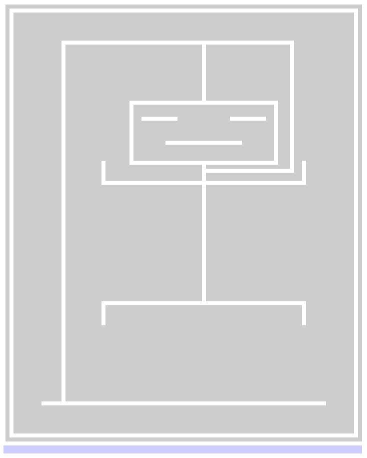


Fig 4:.Snapshot of Hangman

The drawing procedure of each part of the hangman is almost same. We just change the value of row and column. By changing them we were able to create different structure.

```
DRAW_BODY PROC
    MOV CX, 560
    MOV DX, 85
    LEVEL
    DRAW_PIXEL CX, DX
    INC DX
    CMP DX, 120
    JLE LEVEL
    RET
DRAW_BODY ENDP
```

Fig 5: *Draw Body procedure*

We draw the other parts of the body in the same way only by changing the value of the row and column.

5.5 Main menu:

To make the game more represent able we added a main menu where we give some option which can be used by the players. We add “NEW GAME”,”CREDITS”, and “EXIT” options.

By selecting “NEW GAME” the players are able to play the game. By selecting “CREDITS” we can see the name of the players. And by choosing the “EXIT” option players can exit the game.

We use the display mode to display this main menu. We keep this name in strings. Then we use the same process that we use before to show a string.

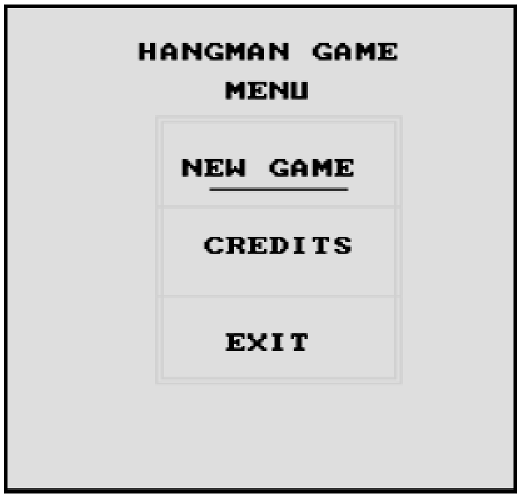


Fig 6:Snapshot of Menu

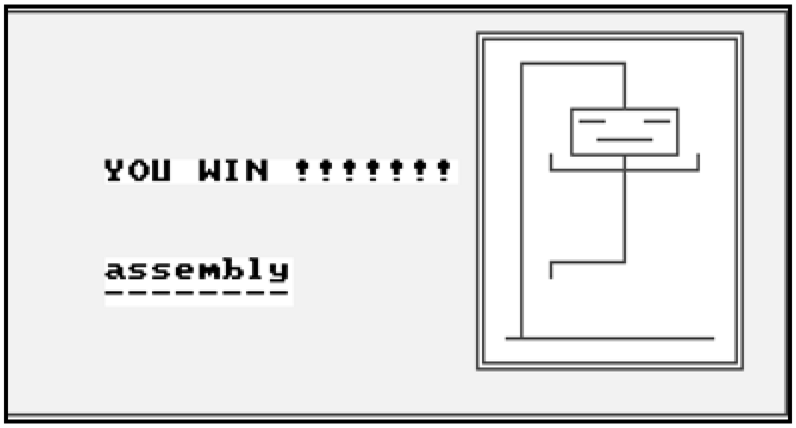
Again we use up and down to select these options. We draw a line to indicate in which stage the player is. We compare AH with 72 and 80 as 72 indicate up arrow and 82 the down. If enter is pressed we access to that option. We will remain in the menu page until enter is pressed.

```
SELECT:
              MOV AH, 0
              INT 16H
             CMP AH, 1CH
             JE PROCESS
             CMP AH, 72
             JE UP
             CMP AH, 80
            JE DOWN
JMP SELECT
```

FIG 7: *The Select level*

6. Output Result:

As we mentioned before the result of the game is simple win or lose. If the player is able to guess the right string then the player can see: "YOU WIN" in the screen and if he loses "YOU LOSS" will be displayed.



```
YOU WIN !!!!!!!

assembly
--------
```

Fig 8: *Snapshot of result*

7. Limitations:

The hangman drawn in the game does not seem like real human being, if image could have been used, it would look more lively. Also the graphics quality is not that good here. There is no score board of the game, only result is to win or lose. Only the keyboard can be used, mouse listener is not used in the game. The game can be run only inside the dosbox or similar virtual environment.

8. Future Plan:

Our future plan with this game is to overcome all the limitations. We want to give more categories like country's name, flower or fruit. Also we will increase more complexity like the chance of guessing will be limited. We also want to make the game more presentable by focusing on graphical work.

9. Discussion:

During making this game we face different kind of problems. Like after using a register for one reason the value can be changed but we also need the previous value that was kept in the register. But we could not get it back. To solve this problem we got to know the register should be push in a stack before further use. When we pop the register we get the previous value.

Again we get to know that we have to use a level if the process in a loop becomes too lengthy.

10. References:

1. - Ytha Yu & Charles Marut . “Assembly Language Programming and Organization of IBM PC” International Edition 1992
2. http://en.wikipedia.org/wiki/Hangman_%28game%29

THANK YOU