****

**JOMO KENYATTA UNIVERSITY**

**OF AGRICULTURE AND TECHNOLOGY**

**UNIT: MICROPROCESSORS I**

**UNIT CODE: EEE2406**

**TITLE: 10 BY 10 RANDOM MATRIX**

**ASSIGNMENT I**

**MEMBERS**

**Davies Momanyi ENE211-0233/2016**

**Allan Mwirigi ENE211-0232/2016**

**Vitalis Ayacko ENE211-0025/2016**

**Thoenes Mbui ENE211-0016/2016**

**Allan Kipkosgei ENE211-0249/2016**

**INSTRUCTIONS**

Develop a 10 by 10 array of two digit random numbers. Use XY coordinate method to key in any location within the array. The numbers can be displayed at locations TOP LEFT, TOP RIGHT, CENTRE, BOTTOM LEFT and BOTTOM RIGHT of the VDU and each time at five seconds interval.

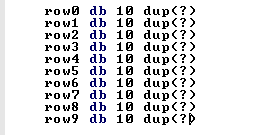
**IMPLEMENTATION**

Our implementation of this task consists of a number of steps:

* Generation of the random numbers and storing them in 10 arrays of size 10 each
* Running a loop to display all the numbers in the 10 by 10 matrix
* Displaying selected numbers at specific points on the screen

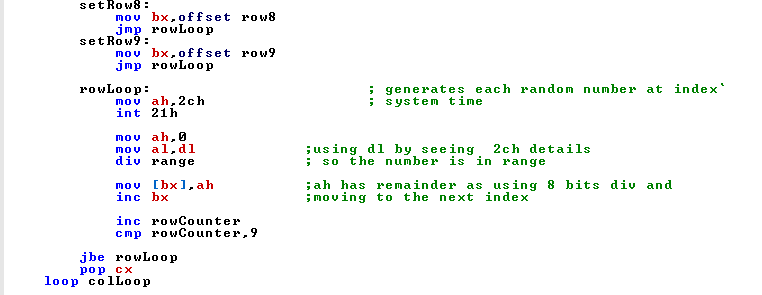
1. **Generation of random numbers**

10 arrays are defined as follows with null values initially. These are used to store the random numbers to be generated.



Two loops are used in generating the random numbers i.e. an outer loop and an inner loop. The inner loop is used to generate a random number and assign it to a location on the selected array. The outer loop is used to shift from one array to the next after each if filled with the required 10 elements. As such, all the 10 arrays are populated with 10 random values.





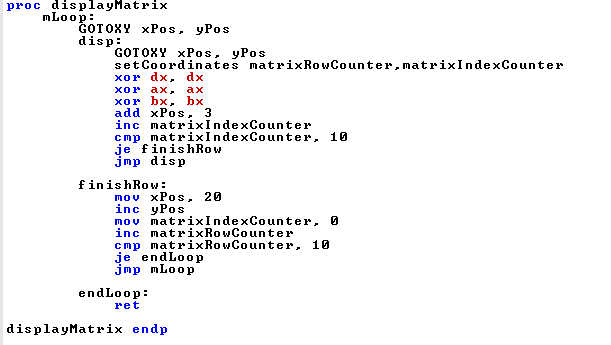
In the above code, the outer loop (**colLoop**) runs ten times as 10 is the initial value of the CX register. The appropriate array is selected based on the current value of the CX register. This is accomplished using a series of CMP and JE statements to match the value of CX to the array to be used.

For example, **JE setRow4** jumps to the **setRow4** label if the value of CX is 4. **setRow4** label assigns the offset address of the row4 array to the BX register then jumps to the inner loop called **rowLoop**. It is in this inner loop that random numbers are generated. The 2CH interrupt is used to give the current system time. When executed it returns into CH = hour CL = minute DH = second DL = 1/100 seconds. The value in DL is used as the seed value for generating random numbers. It is assigned to the AL register and divided by the value of range which is 98. This gives a reminder in AL which is a number in the range of 0 to 98.

This reminder is assigned to BX (register indirect addressing mode) and BX incremented for use in the next iteration of the loop. This process is carried out 10 times using a conditional jump statement depending on the value of **rowCounter** variable. **rowCounter** is incremented from 0 to 9 while the numbers are assigned to the array.

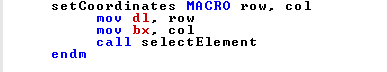
1. **Displaying the 10 by 10 Matrix**

The **displayMatrix** procedure is called to display the contents of the 10 arrays in a matrix format.

****

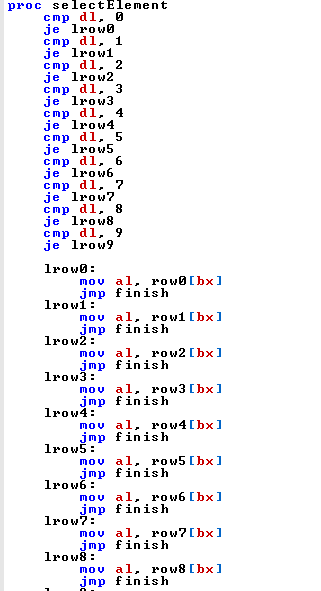
It consists of a loop which displays each element in each array while setting the cursor position for appropriate formatting and spacing of the displayed numbers. **GOTO\_XY** is a macro in the emu8086.inc library included in the code. It shifts the cursor to a specified position on the screen. It is supplied with the values of x-coordinate and y-coordinate to use by the **xPos** and **yPos** variables respectively. **xPos** is incremented in the loop by 3 in order to space adjacent numbers, otherwise they would be overlapping with each other. When the value of **matrixIndexCounter** reaches 10, this signifies that all the elements of the array have been displayed. It jumps to the **finishRow** label which resets the **xPos** to the initial value and increments **yPos** to move the cursor to the next line.

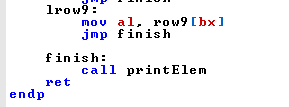
To display each number, the **setCoordinate** macro is first invoked. This is its implementation.



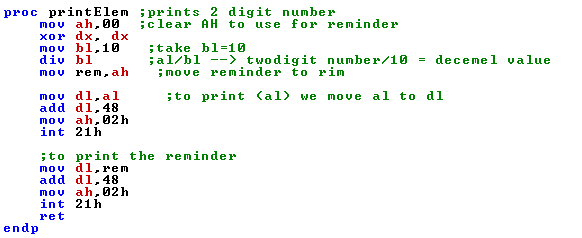
It is used to set the value of the value of the array in use (**row**) and the offset address of the element to be displayed (**col**). These values are assigned to the DL and BX registers respectively, which will be used in the **selectElement** procedure. The **selectElement** procedure is called to set the number matching these coordinates before it is displayed. Its implementation is shown on the next page.

The value of DL is used to set the array to use. For example, if the current value of DL is 3, there is a jump to the label lRow3. In this label, indexed addressing mode is used to assign the value corresponding to BX to register AL, then jumps to label **finishRow.** In this label, the **printElem** procedure is called which is what actually displays the number in AL on the screen.





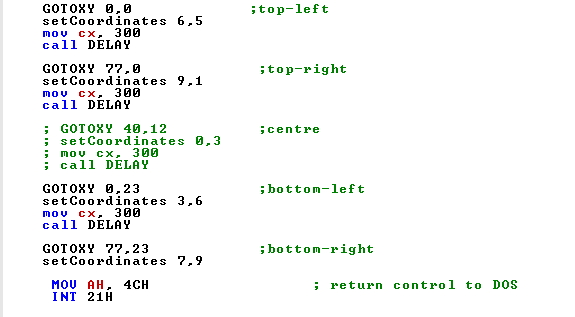
**printElem** is implemented as follows.



Registers AH and DX are cleared before just in case there are values within. Since AL is a two digit number, it is divided by 10 (in BL) such that the quotient is stored in AL and the reminder in AH. This gives us the two digits of the number which can then be printed separately. The value in AH is stored in variable **rem** because AH is required for the interrupts. To print each digit, it is first assigned to DL and 48 added to DL to give the right ASCII value. The 02H interrupt is then invoked to print the number on the screen.

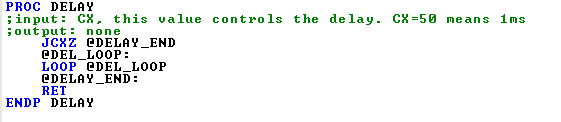
1. **Displaying selected numbers at specific points on the screen**

The following code is used to display selected numbers at various positions on the screen i.e. top-left, top-right, bottom-left, bottom-right



The **GOTO\_XY** macro shifts the cursor to a specified position on the screen. It is supplied with the values of x-coordinate and y-coordinate to use. The values specified above correspond to the top-left, top-right, bottom-left, bottom-right positions on the screen.

The **setCoordinates** macro is then invoked. It has been explained in Part B above. It sets the array and offset address of the number to display, then the **selectElement** procedure is called to assign the number to display to AL. The value in AL is then displayed using the **printElem** procedure. The **DELAY** procedure is then called to provide delay between each number being displayed.



This procedure uses a long running empty loop to create a delay before the execution of the next code.

The 4CH interrupt is then invoked to terminate the program.

**PROGRAM OUTPUT**

The following shows the results of the program on the emu8086 emulator.

