

**HACETTEPE UNIVERSITY**

**ELECTRICAL AND ELECTRONICS ENGINEERING**

**ELE338 MICROPROCESSOR ARCHITECTURE AND PROGRAMMING LAB.**

**PRELIMINARY WORK 4**

**PROCEDURE AND INTERRUPT USAGE**

**2020-2021 SPRING**

**Student**

**Name: Egemen Can**

**Surname: Ayduğan**

**ID: 21728036**

**Date: 18.04.2021**

**1. Question**

**EMU 8086 CODE**

|  |
| --- |
| **.Model Small**  **.Stack 64**  **.Data**  **String1 DB "Press S/s for square, T/t for triangle: ","$"**  **String2 DB 0Ah,0Dh,"Press enter the height of shape: ","$"**  **Select DB 1 dup(?)**  **Height DB 1 dup(?)**  **NotValid DB "It is not valid input","$"**  **Cross1 DB 0ah,0dh,"X","$"**  **Cross2 DB "X","$"**  **SpaceX DB " ","$"**  **NextLine DB 0Ah,0Dh,"$"**  **.Code**  **Proc Main**  **MOV AX,Data**  **MOV DS,AX**  **Start:**  **Lea BX,Select ; To keep the SELECTION in memory**  **Lea BP,Height ; To keep the HEIGHT in memory**  **Mov AH,9h**  **Mov DX,OFFSET String1 ; To printf String1**  **Int 21h**  **Mov AH,01h**  **Int 21h ; To get the selection**  **Mov [BX],AL ; To keep selection**  **Mov AH,9h**  **Mov DX,OFFSET String2 ; To printf String2**  **Int 21h**  **Mov AH,01h**  **Int 21h ; To get the height**  **Mov [BP],AL ; To keep height**  **SUB [BP],48d ; Numbers in ASCII start at 48**  **; I subtract 48 from the height to find exact number**  **Mov AH,9h**  **Mov DX,OFFSET NextLine ; To pass next line**  **Int 21h**  **Cmp [BX],53h ; IF Select Data is S/s**  **Je Square ; Jump to Square**  **Cmp [BX],73h**  **Je Square**  **Cmp [BX],54h ; IF Select Data is T/t**  **Je Triangle ; Jump to Triangle**  **Cmp [BX],74h**  **Je Triangle**  **Jne Invalid ; If the select is not S/s or T/t, the code jumps to Invalid.**  **Square:**  **CALL DrawSquare ; To draw square**  **Jmp Finish**  **Triangle:**  **CALL DrawTriangle ; To draw triangle**  **Jmp Finish**  **Invalid:**  **Mov AH,9h**  **Mov DX,OFFSET NotValid ; When the select is not S/s or T/t**  **Int 21h ; To print screen "It is not valid input".**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Jmp Start ; It jumps to Start to select a valid input.**  **Endp Main**    **Proc DrawSquare**  **MOV CL,[BP] ; To keep height**  **MOV CH,CL**  **SUB CH,2d ; To keep the height of the middle area**    **TOP: ; TOP OF THE SQUARE**  **Mov AH,9h**  **Mov DX,OFFSET Cross2**  **Int 21h ; To print top of the square**  **Dec CL**  **Cmp CL,00d**  **JNE TOP**    **ALLSIDE: ; MIDDLE PART OF SQUARE**  **MOV CL,[BP]**  **DEC CH**  **FirstX:**  **Mov AH,9h**  **Mov DX,OFFSET Cross1 ; To print first X**  **Int 21h**  **Dec CL**  **Space:**  **Mov AH,9h**  **Mov DX,OFFSET SpaceX ; To print spaces**  **Int 21h**  **Dec CL**  **Cmp CL,01h**  **JNE Space**  **FinalX:**  **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To print final X**  **Int 21h**  **Cmp CH,00d**  **JNE ALLSIDE**    **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**    **MOV CL,[BP]**  **BOTTOM: ; BOTTOM OF THE SQUARE**  **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To print bottom of the square**  **Int 21h**  **Dec CL**  **Cmp CL,00d**  **JNE BOTTOM**  **Ret**  **Endp DrawSquare**  **Proc DrawTriangle**  **TriTop:**  **Mov CX,[BP] ; To keep height**  **Sub CX,01h ; To print space**  **Mov CH,00h ; I gave 0 to CH to fix the number.**  **Spa1:**  **Mov AH,9h**  **Mov DX,OFFSET SpaceX ; To print space up to the HEİGHT-1**  **Int 21h**  **Loop Spa1**    **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To print X**  **Int 21h**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**    **Mov DI,01h**  **Mov BX,01h**  **Mov CX,[BP] ; I edited registers for other parts**  **Sub CX,02h**  **MOV CH,00h**      **TriSide:**  **Mov SI,CX**    **Spa2:**  **Mov AH,9h ; First Spaces**  **Mov DX,OFFSET SpaceX ; To print space up to the HEİGHT-2**  **Int 21h**  **Dec SI**  **Cmp SI,00h**  **Jne Spa2**    **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To printf X**  **Int 21h**    **OtherSpa2:**  **Mov AH,9h ; Second Spaces**  **Mov DX,OFFSET SpaceX**  **Int 21h**  **Dec DI**  **Cmp DI,00h**  **Jne OtherSpa2**    **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To print other X on the same line**  **Int 21h**  **Add BX,02h ; To adjust other spaces ın the lower line**  **Mov DI,BX**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Loop TriSide**  **Mov BX,[BP]**  **Sub BX,01h**  **Mov BH,00h ; I edited registers for other parts**  **mov CX,[BP]**  **Mov CH,00h**  **Add CX,BX**    **TriBottom:**  **Mov AH,9h**  **Mov DX,OFFSET Cross2 ; To print bottom of the triangle**  **Int 21h**  **Loop TriBottom**  **Ret**  **Endp DrawTriangle**  **Finish:**  **Ends** |

**RESULTS**

|  |
| --- |
|  |
|  |
|  |

**COMMENT**

|  |
| --- |
| In the first question, I wrote the algorithm of the shapes that I will create on paper. I drew the shapes and made an outline based on the shapes.  In my code, I first put a lot of strings and free memory in the .data part. In the .code section, first of all, I have specified the addresses to go to the input values that I will get with the LEA command. The reason I do this is to keep the initial values I get until the end of my code. After getting my selection and height values on the screen, I compared my selection input first. If S / s or T / t moved to the next step. If not, I went back to getting input.  If the value I chose is S or s, I started drawing a square with the procedure. In the square plotting part, I first put 'X' up to height of square on top of the square. For the other sides of the square, I put an 'X' first. Then I put space as much as the height-2. And I put another 'X'. I repeat this process up to the height-2. For the bottom of the square, I put 'X' as much as the height.  If the value I chose is T or t, I started drawing a triangle with the procedure. I put space up to height-1 to make the top of the triangle. And I put an 'X'. To do the other sides, I first left spaces, then put 'X', then space again and put 'X' again. I repeat this process up to the height-2. For the bottom of the triangle, I put 'X' as much as the 2\*height-1 . |

**2. Question**

**EMU8086 CODE**

|  |
| --- |
| **.Model Small**  **.Stack 64**  **.Data**  **String1 DB "Press S/s for square, T/t for triangle: ","$"**  **String2 DB 0Ah,0Dh,"Press enter the height of shape: ","$"**  **Select DB 1 dup(?)**  **Height DB 1 dup(?)**  **NotValid DB "It is not valid input","$"**  **NextLine DB 0Ah,0Dh,"$"**  **.Code**  **PROC Main**  **Mov AX,Data**  **Mov DS,AX**  **Start:**  **Lea BX,Select ; To keep the SELECTION in memory**  **Lea BP,Height ; To keep the HEIGHT in memory**  **Mov AH,9h**  **Mov DX,OFFSET String1 ; To printf String1**  **Int 21h**  **Mov AH,01h**  **Int 21h ; To get the selection**  **Mov [BX],AL ; To keep selection**  **Mov AH,9h**  **Mov DX,OFFSET String2 ; To printf String2**  **Int 21h**  **Mov AH,01h ; To get the height**  **Int 21h**  **Mov [BP],AL ; To keep height**  **Sub [BP],48d ; Numbers in ASCII start at 48**  **Mov AL,10d ; I subtract 48 from the height to find exact number**  **Mul [BP] ; I multiplied the number by ten because to enlarge the image on the screen**  **Mov [BP],AL ; I put the number back into memory.**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Cmp [BX],53h ; IF Select Data is S/s**  **Je Square ; Jump to Square**  **Cmp [BX],73h**  **Je Square**  **Cmp [BX],54h ; IF Select Data is T/t**  **Je Triangle ; Jump to Triangle**  **Cmp [BX],74h**  **Je Triangle**  **Jne Invalid ; If the select is not S/s or T/t, the code jumps to Invalid.**  **Square:**  **CALL DrawSquare ; To draw square**  **Jmp Finish**  **Triangle:**  **CALL DrawTriangle ; To draw triangle**  **Jmp Finish**  **Invalid:**  **Mov AH,9h**  **Mov DX,OFFSET NotValid ; When the select is not S/s or T/t**  **Int 21h ; To print screen "It is not valid input".**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Jmp Start ; It jumps to Start to select a valid input.**  **ENDP Main**  **Proc DrawTriangle ; TO DRAW TRIANGLE**  **Mov AX,0012H ; To set 640\*480 pixels**  **Int 10H**  **Int 10H**  **Mov AX,[BP]**  **Mov AH,00h**  **Mov BL,02h**  **Div BL**  **Mov CX,320 ; Starting points**  **Mov DX,240**  **Sub DX,AX**  **Mov BX,[BP] ; To keep height**  **RightT: ; THE RİGHT EDGE OF THE TRIANGLE**  **Mov AH,0CH ; Places a single pixel**  **Mov AL,0FH ; Pixel color is white**  **Int 10H**    **Inc DX**  **Inc CX**  **Dec BL ; CX and DX increase until it reaches its height.**  **Cmp BL,00h**  **Jne RightT**    **Mov BX,[BP] ; To keep height**  **Mov AL,2 ; I multiplied the height by two for the bottom.**  **Mul BL**  **Mov BL,AL**  **BottomT: ; THE BOTTOM OF THE TRIANGLE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec CX**  **Dec BL**  **Cmp BL,00h ; CX decrease until it reaches its bottom height.**  **Jne BottomT**    **Mov BX,[BP] ; To keep height**  **LeftT: ; THE LEFT EDGE OF THE TRIANGLE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc CX**  **Dec DX**  **Dec BL ; DX decrease, CX increase until it reaches its height.**  **Cmp BL,00h**  **Jne LeftT**    **Ret**  **Endp DrawTriangle**  **PROC DrawSquare ; TO DRAW SQUARE**  **Mov AX,0012H ; To set 640\*480 pixels**  **Int 10H**  **Int 10H**  **Mov AX,[BP]**  **Mov AH,00h**  **Mov BL,02h**  **Div BL**  **Mov CX,320**  **Add CX,AX ; Starting points**  **Mov DX,240**  **Sub DX,AX**    **Mov BX,[BP] ; To keep height**  **RightS: ; THE RIGHT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**      **Inc DX**  **Dec BL**  **Cmp BL,00h ; DX increase until it reaches its height.**  **Jne RightS**    **Mov BX,[BP] ; To keep height**  **BottomS: ; THE BOTTOM OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec CX**  **Dec BL**  **Cmp BL,00h ; CX decrease until it reaches its height.**  **Jne BottomS**    **Mov BX,[BP] ; To keep height**  **LeftS: ; THE LEFT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec DX**  **Dec BL**  **Cmp BL,00h ; DX decrease until it reaches its height.**  **Jne LeftS**    **Mov BX,[BP] ; To keep height**  **TopS: ; THE RIGHT SIDE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc CX**  **Dec BL**  **Cmp BL,00h ; CX increase until it reaches its height.**  **Jne TopS**  **Ret**  **ENDP DrawSquare**  **Finish:**  **Ends** |

**RESULTS**

|  |
| --- |
| **Center of Square CX=320, DX = 240 ; Height = 50 pixels** |
| **Center of Triangle CX=320, DX = 240 ; Height = 60pixels** |
| **Center of Triangle CX=320, DX = 240; Height = 30pixels** |

**COMMENT**

|  |
| --- |
| In my code, I first put a lot of strings and free memory in the .data part. In the .code section, first of all, I have specified the addresses to go to the input values that I will get with the LEA command. The reason I do this is to keep the initial values I get until the end of my code. After getting my selection and height values on the screen, I compared my selection input first. If S / s or T / t moved to the next step. If not, I went back to getting input.  First of all I set the screen to 640x480 pixels. Then I adjusted my starting points so that the center of the shape I drew was in the middle.  The triangle drawing process consisted of 3 parts, and the square drawing process consisted of 4 parts.  My CX Register behaves like the 'x' axis, my DX Register as the 'y' axis. I decreased or increased my CX or DX recorder with each transaction. These operations continued until the height each time. It only continued until the height\*2 in the bottom of the triangle. |

**3.Question**

**EMU8086 Code**

|  |
| --- |
| **.Model Small**  **.Stack 64**  **.Data**  **SquareEdges DW 4 dup(0)**  **String1 DB "Press S/s for square, T/t for triangle: ","$"**  **String2 DB 0Ah,0Dh,"Press enter the height of shape: ","$"**  **Select DB 1 dup(?)**  **Height DB 1 dup(?)**  **NotValid DB "It is not valid input","$"**  **NextLine DB 0Ah,0Dh,"$"**  **.Code**  **PROC Main**  **Mov AX,Data**  **Mov DS,AX**  **Start:**  **Lea BX,Select ; To keep the SELECTION in memory**  **Lea BP,Height ; To keep the HEIGHT in memory**  **Mov AH,9h**  **Mov DX,OFFSET String1 ; To printf String1**  **Int 21h**  **Mov AH,01h**  **Int 21h ; To get the selection**  **Mov [BX],AL ; To keep selection**  **Mov AH,9h**  **Mov DX,OFFSET String2 ; To printf String2**  **Int 21h**  **Mov AH,01h ; To get the height**  **Int 21h**  **Mov [BP],AL ; To keep height**  **Sub [BP],48d ; Numbers in ASCII start at 48**  **Mov AL,10d ; I subtract 48 from the height to find exact number**  **Mul [BP] ; I multiplied the number by ten because to enlarge the image on the screen**  **Mov [BP],AL ; I put the number back into memory.**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Cmp [BX],53h ; IF Select Data is S/s**  **Je Square ; Jump to Square**  **Cmp [BX],73h**  **Je Square**  **Cmp [BX],54h ; IF Select Data is T/t**  **Je Triangle ; Jump to Triangle**  **Cmp [BX],74h**  **Je Triangle**  **Jne Invalid ; If the select is not S/s or T/t, the code jumps to Invalid.**  **Square:**  **CALL DrawSquare ; To draw square**  **CALL DetectionSquare ; To detect square**  **Jmp Finish**  **Triangle:**  **CALL DrawTriangle ; To draw triangle**  **Jmp Finish**  **Invalid:**  **Mov AH,9h**  **Mov DX,OFFSET NotValid ; When the select is not S/s or T/t**  **Int 21h ; To print screen "It is not valid input".**  **Mov AH,9h**  **Mov DX,OFFSET NextLine**  **Int 21h**  **Jmp Start ; It jumps to Start to select a valid input.**  **ENDP Main**  **Proc DrawTriangle ; TO DRAW TRIANGLE**  **Mov AX,0012H ; To set 640\*480 pixels**  **Int 10H**  **Int 10H**  **Mov AX,[BP]**  **Mov AH,00h**  **Mov BL,02h ; Center of the Triangle(320,240)**  **Div BL**  **Mov CX,320 ; Starting points**  **Mov DX,240**  **Sub DX,AX**  **Mov BX,[BP] ; To keep height**  **RightT: ; THE RİGHT EDGE OF THE TRIANGLE**  **Mov AH,0CH ; Places a single pixel**  **Mov AL,0FH ; Pixel color is white**  **Int 10H**  **Inc DX**  **Inc CX**  **Dec BL ; CX and DX increase until it reaches its height.**  **Cmp BL,00h**  **Jne RightT**    **Mov BX,[BP] ; To keep height**  **Mov AL,2 ; I multiplied the height by two for the bottom.**  **Mul BL**  **Mov BL,AL**  **BottomT: ; THE BOTTOM OF THE TRIANGLE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**  **Dec CX**  **Dec BL**  **Cmp BL,00h ; CX decrease until it reaches its bottom height.**  **Jne BottomT**    **Mov BX,[BP] ; To keep height**  **LeftT: ; THE LEFT EDGE OF THE TRIANGLE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**  **Inc CX**  **Dec DX**  **Dec BL ; DX decrease, CX increase until it reaches its height.**  **Cmp BL,00h**  **Jne LeftT**  **Ret**  **Endp DrawTriangle**  **PROC DrawSquare ; TO DRAW SQUARE**  **Mov AX,0012H ; To set 640\*480 pixels**  **Int 10H**  **Int 10H**  **Mov AX,[BP]**  **Mov AH,00h ; Center of the Square(320,240)**  **Mov BL,02h**  **Div BL**  **Mov CX,320**  **Add CX,AX ; Starting points**  **Mov DX,240**  **Sub DX,AX**    **Mov BX,[BP] ; To keep height**  **RightS: ; THE RIGHT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc DX**  **Dec BL**  **Cmp BL,00h ; DX increase until it reaches its height.**  **Jne RightS**    **Mov BX,[BP] ; To keep height**  **BottomS: ; THE BOTTOM OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec CX**  **Dec BL**  **Cmp BL,00h ; CX decrease until it reaches its height.**  **Jne BottomS**    **Mov BX,[BP] ; To keep height**  **LeftS: ; THE LEFT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec DX**  **Dec BL**  **Cmp BL,00h ; DX decrease until it reaches its height.**  **Jne LeftS**    **Mov BX,[BP] ; To keep height**  **TopS: ; THE RIGHT SIDE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc CX**  **Dec BL**  **Cmp BL,00h ; CX increase until it reaches its height.**  **Jne TopS**  **Ret**  **ENDP DrawSquare**  **Proc DetectionSquare**    **Mov AX,[BP]**  **Mov AH,00h ; AL Register is holding my height**  **Mov BP,OFFSET SquareEdges ; All edges of square**  **Mov [SquareEdges],CX ; 1. Edge**  **Mov [SquareEdges+2],DX**  **Add [SquareEdges+2],AX ; 2. Edge**  **Mov [SquareEdges+4],CX**  **Sub [SquareEdges+4],AX ; 3. Edge**  **Mov [SquareEdges+6],DX ; 4. Edge**    **GetXY:**  **Mov AX,03**  **Int 33H**  **Shr CX,1**  **Cmp BX,1 ; When I press the left mouse button.**  **Je Interval ; Goes to Interval control**  **JMP GetXY**    **; It checks whether the mouse's position is inside or outside of the square by looking at the edges one by one.**  **Interval:**  **Cmp CX,[SquareEdges]**  **Jnc Getxy**  **Cmp CX,[SquareEdges+4]**  **Jc Getxy**  **Cmp DX,[SquareEdges+6]**  **Jc Getxy**  **Cmp DX,[SquareEdges+2]**  **Jnc Getxy**  **Ret**  **Endp DetectionSquare**  **Finish:**  **Ends** |

**COMMENT**

|  |
| --- |
| **I could not do the Triangle control part of this problem. Only Square control part.**  My explanations about drawing triangle and square are available in the previous question. I will only explain the mouse control part.  I opened a 4 Word-size data in memory for mouse control. Here I think of the sides of the square as axes and record them in the data. If I press the left mouse button. The code jumps into the interval check. Then, I checked whether my mouse is above or below these axes one by one. If it is outside of the square, my code always worked, the moment the mouse entered the square, my code was finished. |

**4.Question**

**EMU8086 Code**

|  |
| --- |
| **.Model Small**  **.Stack 64**  **.Data**  **XandY DW 2 dup(?)**  **.Code**  **PROC Main**  **Mov AX,Data**  **Mov DS,AX**    **MOV AX, 12H ; To set 640\*480 pixels**  **INT 10H**  **INT 10H**    **Start:**  **Mov BX,0000h**  **Call Detect ; Checks if the left mouse button has been pressed.**  **DrawS:**  **Call DrawSquare ; If the left button is pressed, it draws a square there.**  **Jmp Start**    **Endp Main**  **PROC Detect**  **Lea BP,XandY**    **MOV AX, 01**  **INT 33H**    **GetXY:**  **Mov AX,03**  **Int 33H**  **Shr CX,1**  **MOV [XandY],CX ; WRITE X COOR. TO MEMORY "X"**  **MOV [XandY+2],DX ; WRITE Y COOR. TO MEMORY "Y"**  **Cmp BX,1**  **Je DrawS**  **JMP GetXY**  **Endp Detect**  **PROC DrawSquare ; TO DRAW SQUARE**    **Add [XandY],04h**  **Sub [XandY+2],04h ; Starting Points**  **Mov CX,[XandY]**  **Mov DX,[XandY+2]**    **Mov BL,10d ; To keep height**  **RightS: ; THE RIGHT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc DX**  **Dec BL**  **Cmp BL,00h ; DX increase until it reaches its height.**  **Jne RightS**    **Mov BL,10d ; To keep height**  **BottomS: ; THE BOTTOM OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**      **Dec CX**  **Dec BL**  **Cmp BL,00h ; CX decrease until it reaches its height.**  **Jne BottomS**    **Mov BL,10d ; To keep height**  **LeftS: ; THE LEFT EDGE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Dec DX**  **Dec BL**  **Cmp BL,00h ; DX decrease until it reaches its height.**  **Jne LeftS**    **Mov BL,10d ; To keep height**  **TopS: ; THE RIGHT SIDE OF THE SQUARE**  **Mov AH,0CH**  **Mov AL,0FH**  **Int 10H**    **Inc CX**  **Dec BL**  **Cmp BL,00h ; CX increase until it reaches its height.**  **Jne TopS**  **Ret**  **ENDP DrawSquare**  **Ends** |

**RESULTS**

|  |  |
| --- | --- |
|  |  |

**Comment**

|  |
| --- |
| I used two procedures in this code and my code works forever. First, I defined 2 16-bit free memories in the .data section. These memories will take the x and y axes where the mouse is on the screen.  My first procedure is to get its position from the screen and it controls my left click. When I left click on the screen, we go to the other procedure. It draws a square, the center of which is the location I clicked on. |