HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

------***------

ASSEMBLY LANGUAGE AND COMPUTER ARCHITECTURE LAB

FINAL PROJECT REPORT

IT3280E

Instructor: Lê Bá Vui

Group 07

Name	Student ID	Problem
Trịnh Tiến Dũng	20215187	4
Hoàng Tuấn Kỳ	20210505	9

Hanoi, 2024

Table of contents

4. Postscript CNC Marsbot

1. Problem description

To control Marsbot to cut into the desired shape, Marsbot will be loaded a script which is a string consisting of 3 consecutive elements:

- <ANGLE>, <CUT/UNCUT>, <DURATION>
- <ANGLE> is the angle of HEADING command of Marsbot
- <CUT/UNCUT> leave or does not leave a track.
- <DURATION> time for current operation

Create a program that Marsbot can do:

- Cut the metal panel as described above.
- The content of scripts is hardcoded in the source code.
- The source code includes 3 scripts and users can press 0, 4 or 8 in the key matrix to select the script to execute.
- One script should contain DCE. Two remaining scripts are proposed by students (at least 10 lines).

2. Source code

- Declare addresses for Marsbot, KeyMatrix, and 3 postscripts

```
# Mars Bot
.eqv HEADING 0xffff8010 # Integer: An angle between 0 and 359
.eqv MOVING 0xffff8050 # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0): whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location of MarsBot
.eqv WHEREY 0xffff8040 # Integer: Current y-location of MarsBot
#Key Matrix
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

.data
# postscript when 0 is pressed: DCE
postscript0: .word 90,0,3000,180,0,3000,180,1,5800,80,1,500,70,1,500,60,1,500,50,1,500,40,1,500,30,1,500,20,1,500,10,1,
end0: .word
# postscript when 4 is pressed
postscript4: .word 90,0,6000,180,0,3000,270,1,500,260,1,500,250,1,500,240,1,500,230,1,500,220,1,500,210,1,500,200,1,500
end4: .word
# postscript when 8 is pressed
postscript8: .word 90,0,6000,180,0,3000,270,1,1500,240,1,1500,210,1,1500,180,1,1500,150,1,1500,120,1,1500,90,1,1500,60,
end8: .word

```

- Main procedure: enable interrupt of keyboard matrix and create no-end loop to wait for interrupt

```
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
#-----------------------------------------------------------
# Enable interrupts you expect
#-----------------------------------------------------------
# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
li $t1, IN_ADDRESS_HEXA_KEYBOARD
li $t3, 0x80 # bit 7 = 1 to enable
sb $t3, 0($t1)
#-----------------------------------------------------------
# No-end loop
#-----------------------------------------------------------
Loop:
nop
nop
addi $v0, $zero, 32
li $a0, 200
syscall
nop
nop
b Loop # Wait for interrupt
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- Jump to the address 0x80000180. Save the values of registers $at, $v0, $a0, $t1, and $t3 into the stack.

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
#-----------------------------------------------------------
# SAVE the current REG FILE to stack
#-----------------------------------------------------------
IntSR: addi $sp,$sp,4 # Save $at because we may change it later
sw $at,0($sp)
addi $sp,$sp,4 # Save $sp because we may change it later
sw $v0,0($sp)
addi $sp,$sp,4 # Save $a0 because we may change it later
sw $a0,0($sp)
addi $sp,$sp,4 # Save $t1 because we may change it later
sw $t1,0($sp)
addi $sp,$sp,4 # Save $t3 because we may change it later
sw $t3,0($sp)
#-----------------------------------------------------------
```

- get_key procedure: check row 1, 2, 3 and re-enable bit 7. If a key is pressed then jump to key_pressed procedure

```
64  #---------------------------------------------------------
65  # Processing
66  #---------------------------------------------------------
67  get_key:
68  li $t1, IN_ADDRESS_HEXA_KEYBOARD
69  li $t3, 0x81 # check row 1 and re-enable bit 7
70  sb $t3, 0($t1) # must reassign expected row
71  li $t1, OUT_ADDRESS_HEXA_KEYBOARD
72  lb $a0, 0($t1)
73  bne $a0, 0x0, key_pressed
74
75  li $t1, IN_ADDRESS_HEXA_KEYBOARD
76  li $t3, 0x82 # check row 2 and re-enable bit 7
77  sb $t3, 0($t1) # must reassign expected row
78  li $t1, OUT_ADDRESS_HEXA_KEYBOARD
79  lb $a0, 0($t1)
80  bne $a0, 0x0, key_pressed
81
82  li $t1, IN_ADDRESS_HEXA_KEYBOARD
83  li $t3, 0x84 # check row 3 and re-enable bit 7
84  sb $t3, 0($t1) # must reassign expected row
85  li $t1, OUT_ADDRESS_HEXA_KEYBOARD
86  lb $a0, 0($t1)
87  bne $a0, 0x0, key_pressed
```

- key_pressed procedure: check if key 0, 4, 8 is pressed. If true then assign $a2 the start address and $a1 the end addres of the postscript correpond with the pressed key, and jump to MarsBot_Draw procedure. If flase jump to end_script procedure.

```
88
89  key_pressed:
90  beq $a0, 0x11, key_0 # 0 is pressed
91  beq $a0, 0x12, key_4 # 4 is pressed
92  beq $a0, 0x14, key_8 # 8 is pressed
93  j end_script
94  key_0:
95  la $a2, postscript0 # start address of postscript
96  la $a1, end0 # end address of postscript
97  j MarsBot_Draw
98  key_4:
99  la $a2, postscript4 # start address of postscript
100 la $a1, end4 # end address of postscript
101 j MarsBot_Draw
102 key_8:
103 la $a2, postscript8 # start address of postscript
104 la $a1, end8 # end address of postscript
105 j MarsBot_Draw
106
```

- MarsBot_Draw: read the postscript and draw. First read the angle from the first element then assign to $a0 and jump to procedure ROTATE of Marsbot. Then go to next element to determine to tracking or not. After that, jump to GO procedure to start moving and assign next element to $a0 to keep running by sleeping. Finally, jump to UNTRACK to stop tracking and go to next

element to continue reading the postscript until $a2 equal $a1 (the end address of postscript). When $a2 equal $a1 jump to end_script procedure to stop marsbot.

```
106
107 MarsBot_Draw: # draw mars bot
108 read_script: # read postscript
109 beq $a2, $a1, end_script
110 read_angle:
111 lw $a0, 0($a2) # load angle to $a0
112 jal ROTATE
113 addi $a2, $a2, 4 # go to next parameter of postscript
114 read_cut_uncut: # cut if 1, uncut if 0
115 lw $s0, 0($a2)
116 beq $s0, $0, read_duration
117 jal TRACK # track if parameter is 1
118 read_duration:
119 jal GO
120 addi $a2, $a2, 4 # go to next parameter of postscript
121 lw $a0, 0($a2) # load duration to $a0
122 addi $v0,$zero,32 # Keep running by sleeping
123 syscall
124 jal UNTRACK
125 addi $a2, $a2, 4 # go to next parameter of postscript
126 j read_script # jump back to loop
127
128 end_script:
129 jal STOP
130
```

- Evaluate the return address of the main routine by adding 4 to the value of the epc register (Exception Program Counter) and storing it back to the epc register. Restore the saved register values from the stack and the eret instruction help to return from the exception and resume the main program execution.

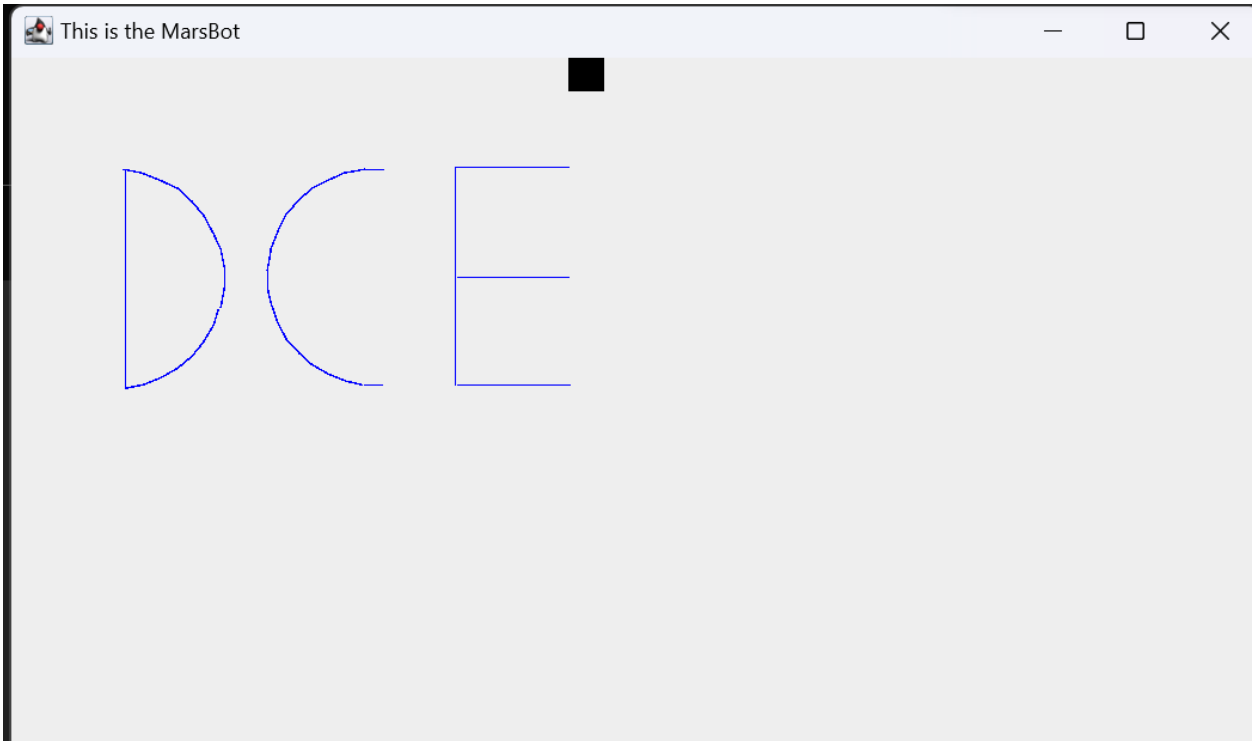
```
130
131 #-----------------------------------------------------------
132 # Evaluate the return address of main routine
133 # epc <= epc + 4
134 #-----------------------------------------------------------
135 next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
136 addi $at, $at, 4 # $at = $at + 4 (next instruction)
137 mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
138 #-----------------------------------------------------------
139 # RESTORE the REG FILE from STACK
140 #-----------------------------------------------------------
141 restore:lw $t3, 0($sp) # Restore the registers from stack
142 addi $sp,$sp,-4
143 lw $t1, 0($sp) # Restore the registers from stack
144 addi $sp,$sp,-4
145 lw $a0, 0($sp) # Restore the registers from stack
146 addi $sp,$sp,-4
147 lw $v0, 0($sp) # Restore the registers from stack
148 addi $sp,$sp,-4
149 lw $at, 0($sp) # Restore the registers from stack
150 addi $sp,$sp,-4
151 return: eret # Return from exception
152
```

- Procedures of Marsbot:

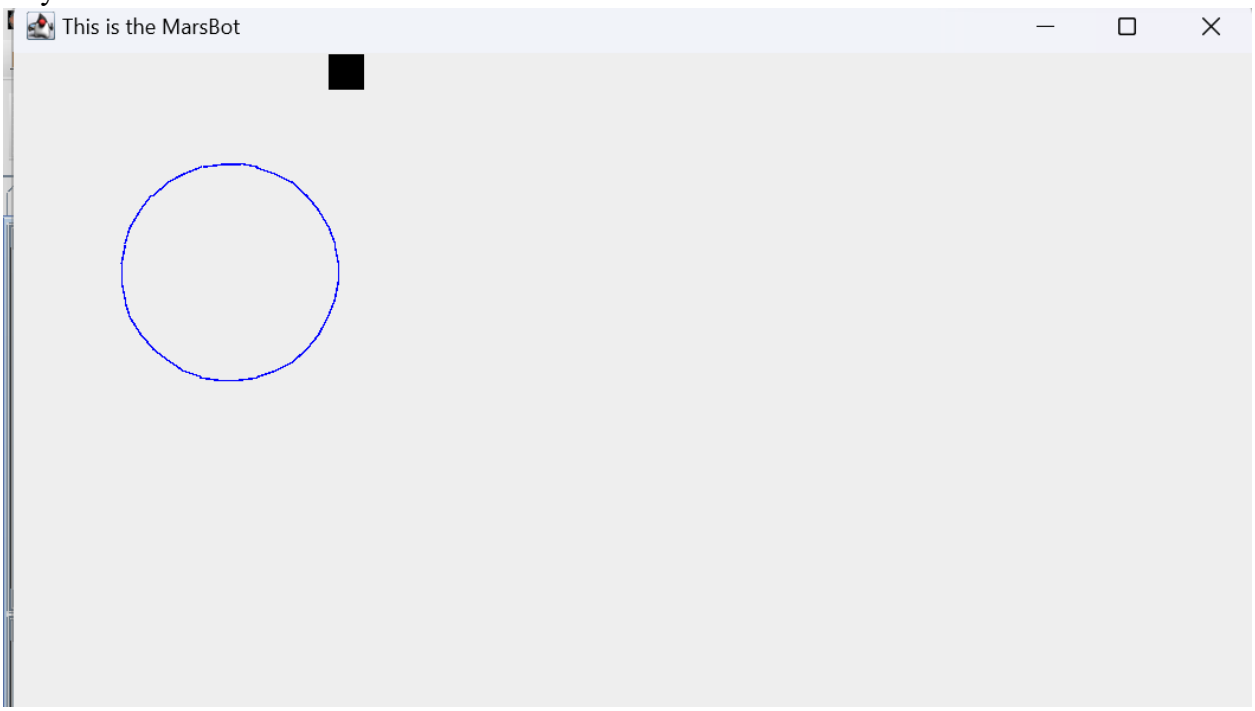
```
155
156 GO: li $at, MOVING # change MOVING port
157 addi $k0, $zero,1 # to logic 1,
158 sb $k0, 0($at) # to start running
159 jr $ra
160
161 STOP: li $at, MOVING # change MOVING port to 0
162 sb $zero, 0($at) # to stop
163 jr $ra
164
165 TRACK: li $at, LEAVETRACK # change LEAVETRACK port
166 addi $k0, $zero,1 # to logic 1,
167 sb $k0, 0($at) # to start tracking
168 jr $ra
169
170 UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0
171 sb $zero, 0($at) # to stop drawing tail
172 jr $ra
173
174 ROTATE: li $at, HEADING # change HEADING port
175 sw $a0, 0($at) # to rotate robot
176 jr $ra
177
```

3. Result

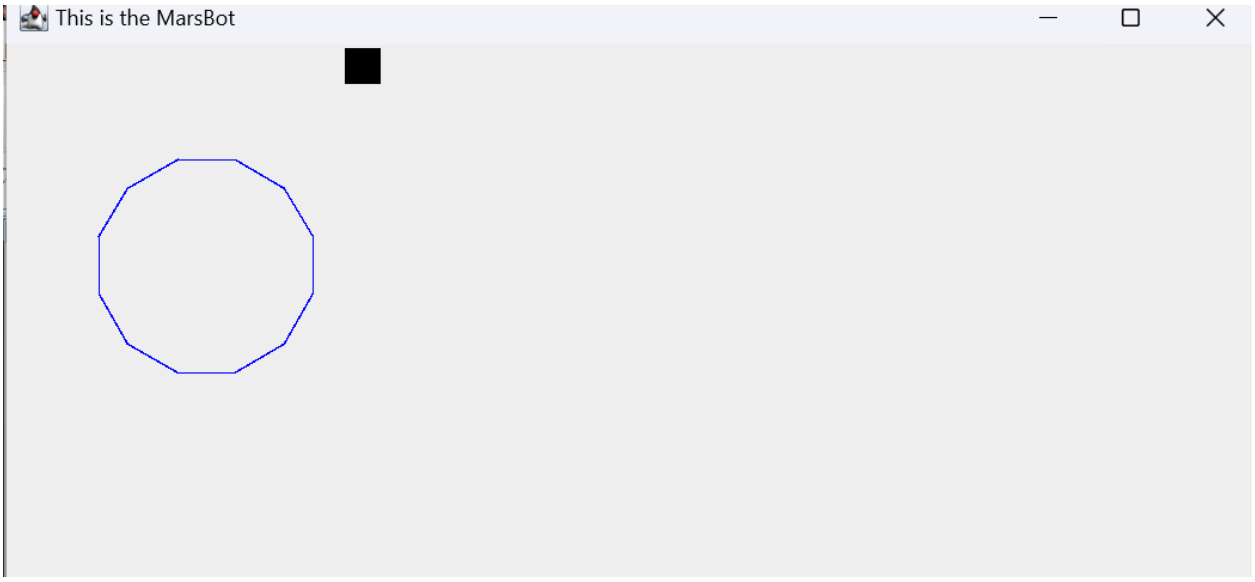
- Key 0:



- Key 4:



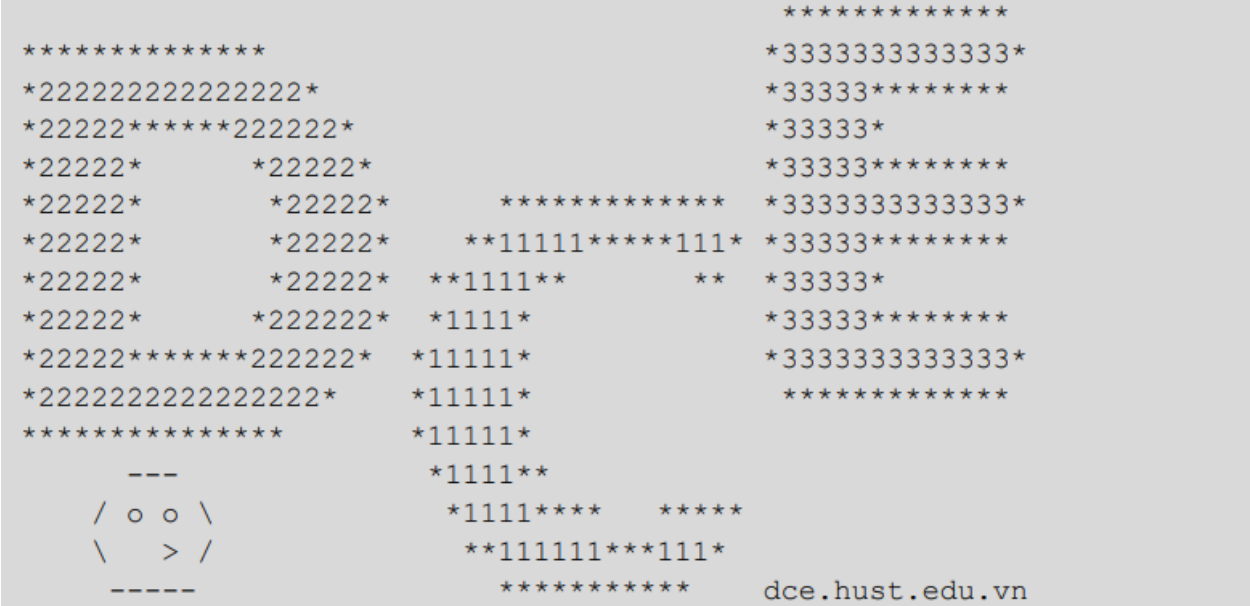
- Key 8:



9. Drawing shape using ASCII characters

1. Problem description

Given a picture translated to ASCII characters as follows, this is the shapes of DCE with border * and colors are digits.



- Show this picture in the console window.
- Change the picture so that DCE has only a border without color inside.
- Change the order of DCE to ECD.
- Enter the new color number from the keyboard, update the picture with new colors.

2. Source code and explanation

a) Picture section

```
.data
line1: .asciiz  "                                          *************       \n"
line2: .asciiz  " **************                          *3333333333333*      \n"
line3: .asciiz  " *222222222222222*                        *33333********       \n"
line4: .asciiz  " *22222******222222*                      *33333*              \n"
line5: .asciiz  " *22222*      *22222*                     *33333********       \n"
line6: .asciiz  " *22222*       *22222*        *************  *3333333333333*      \n"
line7: .asciiz  " *22222*       *22222*      **11111*****111* *33333********       \n"
line8: .asciiz  " *22222*       *22222*    **1111**       **  *33333*             \n"
line9: .asciiz  " *22222*      *222222*    *1111*             *33333********       \n"
line10: .asciiz " *22222*******222222*    *11111*            *3333333333333*      \n"
line11: .asciiz " *2222222222222222*      *11111*             *************       \n"
line12: .asciiz " ***************         *11111*                                  \n"
line13: .asciiz "      ---                 *1111**                                 \n"
line14: .asciiz "    / o o \\               *1111****   *****                      \n"
line15: .asciiz "    \\   > /                **111111***111*                       \n"
line16: .asciiz "     -----                   ***********      dce.hust.edu.vn    \n"
```

We consider the picture is the combination of 16 lines with each line contains 68 ASCII characters.

b) Menu section:

menu_message: .asciiz "\n\n ----MENU----\n 1. Show picture.\n 2. Show picture with only border.\n 3. Change the order.\n 4. Enter new color number and update.\n 5. Exit.\n Enter your choice: "

error_message: "Input must be a integer from 1 to 5"

```
menu:
 li $v0,4
 la $a0,menu_message
 syscall

 li $v0,5
 syscall

 beq $v0,1,menu_func1
 beq $v0,2,menu_func2
 beq $v0,3,menu_func3
 beq $v0,4,menu_func4
 beq $v0,5,exit

 li $v0,4
 la $a0,error_message
 syscall
 j menu
exit:
 li $v0,10
 syscall
```

-When we run program, a menu will appear and require a number from 1 to 5

-If a inappropriate input is entered, the error message is printed and the menu appears again until a satisfy input is entered

-After the program finishes an option from 1 to 4, the menu appears again

-When user choose option 5, the program terminates

c) Function Print

```
print:
 li $t0,16
 li $t1,0
 la $a0,line1
print_loop:
 beq $t1,$t0,end_print_loop
 li $v0,4
 syscall
 addi $a0,$a0,68
 addi $t1,$t1,1
 j print_loop
end_print_loop:
 jr $ra
```

- Option 1, 3, 4 all have a “print” function to print the picture after performing modification to the picture’s data.

- The “print” function is a loop to print all 16 lines of the picture.

d) Option 1: Show this picture in the console window.

```
menu_func1:
 jal print
 j menu
```

- Call “print” function and jump to menu.

e) Option 2: Change the picture so that DCE has only a border without color inside.

```
menu_func2:
 li $t0,16
 li $t1,68
 li $t2,0
 li $t3,0
 la $s0,line1
loop_row:
 beq $t3,$t0,end_loop_row
 li $t2,0
loop_column:
 beq $t2,$t1,end_loop_column
 lb $a1,0($s0)
 addi $s0,$s0,1
 bgt $a1,57,print_char
 blt $a1,48,print_char
 li $a1,32
```

```
print_char:
 li $v0, 11
 move $a0,$a1
 syscall
 addi $t2,$t2,1
 j loop_column
end_loop_column:
 addi $t3,$t3,1
 j loop_row
end_loop_row:
 j menu
```

- Loop through all bytes of all 16 lines.
  + Load value of each byte to $a1
  + If the value is from 48 to 57 (from ‘0’ to ‘9’), it becomes 32 (‘ ‘). Else, the value has no change.
  + Print the ASCII char corresponding to the above value.
- Jump to menu

f) Option 3: Change the order of DCE to ECD.

```
menu_func3:
 li $t9,0
func3:
 li $t0,12
 li $t1,22
 li $t2,1
 li $t3,1
 la $a0,line1
 add $a0,$a0,68
```

```
change_order_row_loop:
 beq $t2,$t0,end_change_order_row_loop
 li $t3,1
change_order_column_loop:
 beq $t3,$t1,end_change_order_column_loop
 add $a1,$a0,$t3
 lb $s1,0($a1) #char in d
 sub $a2,$a1,24
 lb $s2,0($a2) #char in e
 add $a3,$a1,294
 lb $s3,0($a3) #char in c
 sb $s2,0($a1)
 sb $s1,0($a2)
 addi $t3,$t3,1
 j change_order_column_loop
end_change_order_column_loop:
 addi $a0,$a0,68
 addi $t2,$t2,1
 j change_order_row_loop
end_change_order_row_loop:
 bne $t9,0,end_func3
 jal print
 li $t9,1
 j func3
end_func3:
 j menu
```

- We consider 3 letter C, D, E all have the same length and width (length: 22 characters, width: 11 characters
- The program will run func3 two times: first time to change data and print, second time to restore original data.
- Func3 loop through all characters of letter D and find the corresponding position of the characters in C and E:
  + From the location of a character of letter D: add 294 to get the locations of the corresponding character in C and subtract 24 to get the location of the character in E.
  +Swap value of two corresponding locations in D and E.
- If it is the first func3, call “print” function and repeat func3 one more time to restore data. Then, jump to menu.

g) Option 4: Enter the new color number from the keyboard, update the picture with new colors.

```
menu_func4:
input_d_co:
 li $v0,4
 la $a0,input_d_color
 syscall
 li $v0,5
 syscall
 bgt $v0,9,input_d_co
 blt $v0,0,input_d_co
 addi $t4,$v0,48 # d color
input_c_co:
 li $v0,4
```

```
 la $a0,input_c_color
 syscall
 li $v0,5
 syscall
 bgt $v0,9,input_c_co
 blt $v0,0,input_c_co
 addi $t5,$v0,48
input_e_co:
 li $v0,4
 la $a0,input_e_color
 syscall
 li $v0,5
 syscall
 bgt $v0,9,input_e_co
 blt $v0,0,input_e_co
 addi $t6,$v0,48
```

- Input new colors for D, C, E and store in t4, t5, t6. The program will require an integer from 0 to 9. If user inputs an inappropriate number, the program will ask again until user enters a valid input.

```
change_color:
 li $t0,12
 li $t1,22
 li $t2,1
 li $t3,1
 la $a0,line1
 addi $a0,$a0,68
change_color_row_loop:
 beq $t2,$t0,end_change_color_row_loop
 li $t3,1
change_color_column_loop:
 beq $t3,$t1,end_change_color_column_loop
 add $a1,$a0,$t3
 lb $s1,0($a1) #char in d
 move $t8,$t4
 jal modifycolor
 sb $s1,0($a1)

 sub $a2,$a1,24
 lb $s1,0($a2) #char in e
 move $t8,$t6
 jal modifycolor
 sb $s1,0($a2)

 add $a3,$a1,294
 lb $s1,0($a3) #char in c
 move $t8,$t5
 jal modifycolor
 sb $s1,0($a3)

 addi $t3,$t3,1
 j change_color_column_loop
end_change_color_column_loop:
 addi $a0,$a0,68
 addi $t2,$t2,1
 j change_color_row_loop
end_change_color_row_loop:
 jal print
 j menu
```

- Function change color loop through all characters of letter D and find the corresponding position of the characters in C and E:

+ From the location of a character of letter D: add 294 to get the locations of the corresponding character in C and subtract 24 to get the location of the character in E.
+Modify each character by function modifycolor below:

```
modifycolor:
 bgt $s1,57,end_modify
 blt $s1,48,end_modify
 move $s1,$t8
end_modify:
 jr $ra
```

+If the character has a value from 48 to 57, change its value to the color corresponding to which letter that char belongs to

- Call “print” function and jump to menu.

3. Result

a) Menu

```
----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice: -1
Input must be a integer from 1 to 5

----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice:
```

b) Option 1: Show this picture in the console window.

```
----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice: 1
                                                 *************
***************                                 *3333333333333*
*222222222222222*                                *33333********
*22222******222222*                              *33333*
*22222*      *22222*                             *33333********
*22222*       *22222*        *************      *3333333333333*
*22222*       *22222*      **llllll*****lll*    *33333********
*22222*       *22222*    **llll**       **      *33333*
*22222*      *222222*    *llll*                 *33333********
*22222*******222222*    *llllll*                *3333333333333*
*2222222222222222*      *llllll*                 *************
***************         *llllll*
     ---                 *llll**
   / o o \                *llll****   *****
   \   > /                 **lllllll***lll*
    -----                    ***********      dce.hust.edu.vn
```

c) Option 2: Change the picture so that DCE has only a border without color inside

```
----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice: 2
                                                 *************
***************                                 *             *
*               *                               *     ********
*     ******      *                             *     *
*     *      *     *                            *     ********
*     *       *     *         *************      *             *
*     *       *     *       **     *****   * *    ********
*     *       *     *    **    **        **  *     *
*     *      *      *    *     *              *     ********
*     *******      *    *     *              *             *
*               *      *     *               *************
***************        *     *
     ---                *    **
   / o o \                *    ****   *****
   \   > /                 **      ***   *
    -----                    ***********      dce.hust.edu.vn
```

d) Option 3: Change the order of DCE to ECD.

```
----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice: 3
                                          **************
 *************                            *222222222222222*
*3333333333333*                           *22222******222222*
*33333********                            *22222*      *22222*
*33333*                                   *22222*       *22222*
*33333********               *************  *22222*       *22222*
*3333333333333*            **llllll*****lll* *22222*       *22222*
*33333********           **llll**       **  *22222*      *222222*
*33333*                   *llll*             *22222*******222222*
*33333********          *llllll*             *2222222222222222*
*3333333333333*         *llllll*             ***************
 *************          *llllll*
     ---                 *llll**
   / o o \                *llll****   *****
   \   > /                 **lllllll***lll*
    -----                    ***********      dce.hust.edu.vn
```

e) Option 4: Enter the new color number from the keyboard, update the picture with new colors.

```
----MENU----
1. Show picture.
2. Show picture with only border.
3. Change the order.
4. Enter new color number and update.
5. Exit.
Enter your choice: 4
Enter color for D (integer from 0-9):-1
Enter color for D (integer from 0-9):10
Enter color for D (integer from 0-9):5
Enter color for C (integer from 0-9):7
Enter color for E (integer from 0-9):6
                                              ************
**************                               *6666666666666*
*555555555555555*                             *66666********
*55555******555555*                           *66666*
*55555*      *55555*                          *66666********
*55555*       *55555*        ************     *6666666666666*
*55555*       *55555*      **77777*****777*   *66666********
*55555*       *55555*    **7777**       **     *66666*
*55555*      *555555*    *7777*                *66666********
*55555******555555*    *77777*                *6666666666666*
*5555555555555555*     *77777*                 ************
***************        *77777*
     ---                *7777**
   / o o \               *7777****   *****
   \   > /               **777777***777*
    -----                   **********      dce.hust.edu.vn
```