Mục Lục

Bài 5: Biểu thức trung tố hậu tố

Sinh viên thực hiện: Nguyễn Địch Long – 20162525

<u>Đề bài</u>: Viết chương trình tính giá trị biểu thức bất kỳ bằng phương pháp duyệt biểu thức hậu tố.

Các yêu cầu cụ thể:

1. Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8 * 6

2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 * +

3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là số nguyên, trong phạm vi từ 0 □ 99.

Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương

Các yêu cầu cụ thể:

1. Nhập vào biểu thức trung tố, ví dụ: 9 + 2 + 8 *6
2. In ra biểu thức ở dạng hậu tố, ví dụ: 9 2 + 8 6 * +
3. Tính ra giá trị của biểu thức vừa nhập

Các hằng số là số nguyên, trong phạm vi từ 0->99

Toán tử bao gồm phép cộng, trừ, nhân, chia lấy thương.

<u>Thuật toán</u>:

a) Đổi biểu thức trung tố sang hậu tố:

Để đổi biểu thức trung tố sang hậu tố, ta sẽ dùng ngăn xếp và xâu

B1: Đưa một biểu thức trung tố vào xâu kí tự và đặt tên là infix

B2: Tạo ra một xâu mới để lưu biểu thức hậu tố, đặt tên là postfix

B3: Sắp xếp lại xâu:

- Nếu kí tự là toán hạng thì lưu vào postfix.
- Nếu kí tự là toán tử, nếu ngăn xếp trống thì đẩy kí tự vào ngăn xếp.

- Nếu ngăn xếp không trống thì đưa ra khỏi ngăn xếp tất cả các phép toán cho đến khi gặp phép toán có thứ tự ưu tiên thấp hơn hoặc phép toán có tính kết hợp phải có cùng thứ tự ưu tiên, sau đó nạp phép toán đang xét vào ngăn xếp.

B4: Thực hiện bước 3 cho đến khi kết thúc biểu thức và tất cả các toán tử toán hạng được xếp vào postfix, khi đó ta có biểu thức hậu tố.

b) Tính giá trị biểu thức hậu tố:

B1: Duyệt biểu thức hậu tố từ trái sang phải.

B2: Nếu gặp toán hạng thì nạp (push) giá trị của nó vào ngăn xếp.

B3: Nếu gặp phép toán thì thực hiện phép toán này với hai toan hạng được lấy ra (pop) từ ngăn xếp.

B4: Nạp (push) giá trị tính được vào ngăn xếp (như vậy, ba kí hiệu được thay thế bằng một toán hạng).

B5: Tiếp tục duyệt biểu thức cho đến khi trong ngăn xếp còn một giá trị duy nhất. Đó chính là giá trị biểu thức hậu tố.

Mã nguồn:

```
1. .data
2. infix: .space 100
3. postfix: .space 100
4. stack: .space 100
5.
6. input_message:   .asciiz "Enter infix expression:"
7. newLine: .asciiz "\n"
8. infix_message: .asciiz "Infix: "
9. postfix_message: .asciiz "Postfix: "
10.  result_message: .asciiz "Result: "
11.
12.  .text
13.  #---------------------------------------------------------------------
   ------
14.  #---------------------------- GET INFIX EXPRESSION ----------------------
   ------
15.  #---------------------------------------------------------------------
   ------
16.          li $v0, 54
17.          la $a0, input_message
18.          la $a1, infix
19.          la $a2, 100
20.          syscall
21.
```

```
22.
23.         la $a0, infix_message
24.         li $v0, 4
25.         syscall
26.
27.         la $a0, infix
28.         li $v0, 4
29.         syscall
30.
31.  #---------------------------------------------------------------------------
     ------
32.  #---------------------- CONVERT INFIX TO POSTFIX EXPRESSION ----------------
     ------
33.  #---------------------------------------------------------------------------
     ------
34.
35.         li $s6, -1                                  # Infix counter
36.         li $s7, -1                                  # Stack counter
37.         li $t7, -1                                  # Postfix counter
38.
39.  while:
40.          la $s1, infix                              # $s1 = &infix
41.          la $t5, postfix                                    # $t5 =
     &postfix
42.          la $t6, stack                              # $t6 = &stack
43.
44.         addi $s6, $s6, 1                            # iCounter ++
45.
46.         # get infix[iCounter]
47.         add $s1, $s1, $s6                           # &infix = &infix +
     iCounter ++
48.         lb $t1, 0($s1)                              # $t1 = value of
     infix[counter]
49.
50.
51.         beq $t1, '+', operator                             # if $t1 == '+'
     then operator
52.         nop
53.         beq $t1, '-', operator                             # if $t1 == '-'
     then operator
54.         nop
55.         beq $t1, '*', operator                      # if $t1 == '*' then
     operator
56.         nop
57.         beq $t1, '/', operator                      # if $t1 == '/' then
     operator
58.         nop
59.         beq $t1, '^', operator                      # if $t1 == '^' then
     operator
60.         nop
61.         beq $t1, 10, not_operator                   # if $t1 == '\n' then
     not_operator
62.         nop
63.         beq $t1, 32, not_operator                   # if $t1 == space then
     not_operator
64.         nop
```

```
65.         beq $t1, $zero, endWhile                    # if $t1 == null then
   endWhile
66.         nop
67.                                                     # else push number to
   postfix
68.
69.         addi $t7, $t7, 1                            # pCounter ++
70.         add $t5, $t5, $t7                           # &Postfix = &Postfix
   + pCounter
71.
72.         sb $t1, 0($t5)                              # store infix[counter]
   to Postfix
73.
74.  # Check the following character in infix
75.
76.         lb $a0, 1($s1)                              # $a0 = &infix + 1
   (check if the following character in infix is number)
77.         jal check_number
78.         nop
79.         beq $v0, 1, not_operator                    # if the following
   character in infix is number, then not_operator
80.         nop                                         # else add_space to
   postfix and go to operator
81.
82.  add_space:
83.         add $t1, $zero, 32
84.         sb $t1, 1($t5)
85.
86.         addi $t7, $t7, 1                            # pCounter ++
87.
88.         j not_operator
89.         nop
90.
91.  operator:
92.         # add operator from infix to stack
93.
94.         beq $s7, -1, pushOperatorToStack            # if stack is empty
   then pushOperatorToStack
95.         nop
96.         add $t6, $t6, $s7                           # else
97.         lb $t2, 0($t6)                              # $t2 = value of
   stack[sCounter]
98.
99.         # check $t1's precedence
100.        beq $t1, '+', set_t1_to1                    # if $t1 == '+' then
   set $t1's precedence = 1
101.        nop
102.        beq $t1, '-', set_t1_to1                    # if $t1 == '-' then
   set $t1's precedence = 1
103.        nop
104.        beq $t1, '^', set_t1_to3                    # if $t1 == '^' then
   set $t1's precedence = 3
105.        nop
106.
107.        li $t3, 2                                   # else set $t3 = $t1's
   precedence = 2
108.
```

```
109.        j check_t2                          # check $t2's
   precedence
110.        nop
111.
112. set_t1_to1:
113.        li $t3, 1                           # set $t3 = $t1's
   precedence = 1
114.        j check_t2                          # check $t2's
   precedence
115.        nop
116. set_t1_to3:
117.        li $t3, 3                           # set $t3 = $t1's
   precedence = 3
118.
119.        # check $t2's precedence
120. check_t2:
121.
122.        beq $t2, '+', set_t2_to1            # if $t2 == '+' then
   set $t2's precedence = 1
123.        nop
124.        beq $t2, '-', set_t2_to1            # if $t2 == '-' then
   set $t2's precedence = 1
125.        nop
126.        beq $t2, '^', set_t2_to3            # if $t2 == '^' then
   set $t2's precedence = 3
127.        nop
128.
129.        li $t4, 2                           # else set $t4 = $t2's
   precedence = 2
130.
131.        j compare_precedence                # compare $t3 ($t1's
   precedence) and $t4 ($t2's precedence)
132.        nop
133.
134. set_t2_to1:
135.        li $t4, 1                           # set $t4 = $t2's
   precedence = 1
136.        j compare_precedence
137.        nop
138.
139. set_t2_to3:
140.        li $t4, 3                           # set $t4 = $t1's
   precedence = 3
141.
142. compare_precedence:
143.        beq $t3, $t4, ltez_precedence       # if $t3 == $t4 then
   pop $t2 and push $t1 to stack
144.        nop
145.        slt $s1, $t3, $t4                   # if $t3 < $t4 then
   pop $t2 and push $t1 to stack
146.        beqz $s1, pushOperatorToStack       # else branch to
   pushOperatorToStack
147.        nop
148.
149. ltez_precedence:
150. # pop t2 from stack and push it to postfix
151. # push $t1 to stack
```

```
152.
153.        sb $zero, 0($t6)
154.        addi $s7, $s7, -1                          # sCounter --
155.        addi $t6, $t6, -1
156.        la $t5, postfix                            # $t5 = &postfix
157.        addi $t7, $t7, 1                           # pCounter ++
158.        add $t5, $t5, $t7
159.
160.        sb $t2, 0($t5)
161.        j operator
162.        nop
163.
164. #-------------------------------------------------------------------------
   ------
165. pushOperatorToStack:
166.        la $t6, stack                              # $t6 = &stack
167.        addi $s7, $s7, 1                           # sCounter ++
168.        add $t6, $t6, $s7
169.        sb $t1, 0($t6)
170.
171. not_operator:
172.        j while
173.        nop
174.
175. #-------------------------------------------------------------------------
   ------
176. endWhile:                                         # add space
177.        addi $s1, $zero, 32
178.        add $t7, $t7, 1
179.        add $t5, $t5, $t7
180.        la $t6, stack
181.        add $t6, $t6, $s7
182.
183. pop_all_stack:
184.        lb $t2, 0($t6)                             # t2 = value of
   stack[counter]
185.        beq $t2, 0, endPostFix
186.        nop
187.        sb $zero, 0($t6)
188.        addi $s7, $s7, -2
189.        add $t6, $t6, $s7
190.
191.        sb $t2, 0($t5)
192.        add $t5, $t5, 1
193.
194.        j pop_all_stack
195.        nop
196.
197. #-------------------------------------------------------------------------
   ------
198. endPostFix:
199.        la $a0, postfix_message                          # print postfix
200.        li $v0, 4
201.        syscall
202.
203.        la $a0, postfix
204.        li $v0, 4
```

```
205.        syscall
206.
207.        la $a0, newLine
208.        li $v0, 4
209.        syscall
210.
211. #-------------------------------------------------------------------------
   ------
212. #------------------------------ CALCULATING STEP -------------------------
   ------
213. #-------------------------------------------------------------------------
   ------
214.
215.        li $s3, 0                                   # $s3 = counter = 0
216.        la $s2, stack                               # $s2 = &stack
217.
218. while_postfix_stack:
219.        la $s1, postfix                             # $s1 = &postfix
220.        add $s1, $s1, $s3
221.        lb $t1, 0($s1)
222.
223.        beqz $t1, end_while_postfix_stack           # if $t1 is null then
   end_while_postfix_stack
224.        nop
225.
226.        add $a0, $zero, $t1
227.        jal check_number
228.        nop
229.
230.        beqz $v0, is_operator
231.        nop
232.
233.        jal push_num_to_stack
234.        nop
235.
236.        j continue
237.        nop
238.
239.
240. is_operator:
241.        jal pop
242.        nop
243.
244.        add $a1, $zero, $v0                         # $a1 = operand "b" =
   $v0
245.
246.        jal pop
247.        nop
248.
249.        add $a0, $zero, $v0                         # $a0 = operand "a" =
   $v0
250.
251.        add $a2, $zero, $t1                         # $a2 = operator "op"
   = $t1
252.
253.        jal calculate
254.        nop
```

```
255. continue:
256.         add $s3, $s3, 1                                    # counter++
257.
258.         j while_postfix_stack
259.         nop
260.
261. #-----------------------------------------------------------------
262. #Function calculate
263. # @brief calculate the operation <a op b>
264. # @param[in] a0 : (int) a
265. # @param[in] a1 : (int) b
266. # @param[in] t1 : operator(op)
267. # @param[out] v0 : <a op b>
268. #-----------------------------------------------------------------
269. calculate:
270.         sw $ra, 0($sp)
271.         li $v0, 0
272.         beq $t1, '*', cal_mul
273.         nop
274.         beq $t1, '/', cal_div
275.         nop
276.         beq $t1, '+', cal_plus
277.         nop
278.         beq $t1, '-', cal_sub
279.         nop
280.         beq $t1, '^', cal_exp
281.         nop
282. cal_mul:
283.         mul $v0, $a0, $a1
284.         j cal_push
285.         nop
286. cal_div:
287.         div $a0, $a1
288.         mflo $v0
289.         j cal_push
290.         nop
291. cal_plus:
292.         add $v0, $a0, $a1
293.         j cal_push
294.         nop
295. cal_sub:
296.         sub $v0, $a0, $a1
297.         j cal_push
298.         nop
299. cal_exp:
300.         li $t0, 1                                          # lCounter = $t9 = 0
301.         move $v0, $a0                                      # set $v0 = $a0 =
   operand "a"
302.         loop:
303.                 mul $v0, $v0, $a0                          # $v0 = $v0 * $a0 =
   $v0 * a
304.                 addi $t0, $t0, 1                           # lCounter ++
305.                 beq $t0, $a1, cal_push                     # if lCounter == b
   then branch to cal_push
306.                 nop
307.                 j loop
308.                 nop
```

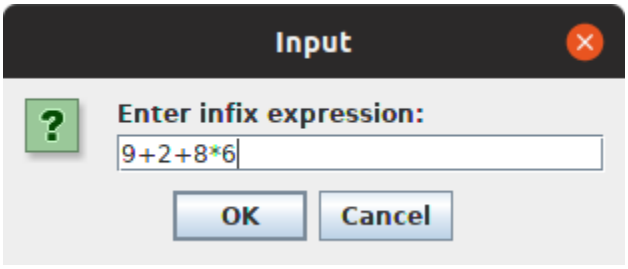
```
309.
310. cal_push:
311.         add $a0, $v0, $zero
312.         jal push
313.         nop
314.         lw $ra, 0($sp)                                  # get return address
   from system stack
315.         jr $ra
316.         nop
317.
318. #---------------------------------------------------------------
319. #Procedure push_num_to_stack
320. # @brief get the number and push to stack at $s2
321. # @param[in] s3 : counter for postfix string
322. # @param[in] s1 : postfix string
323. # @param[in] t1 : current value
324. #---------------------------------------------------------------
325. push_num_to_stack:
326.         sw $ra, 0($sp)                                  # save return address
   to system stack
327.         li $v0, 0
328.
329. while_pnts:
330.         beq $t1, '0', pnts_0
331.         nop
332.         beq $t1, '1', pnts_1
333.         nop
334.         beq $t1, '2', pnts_2
335.         nop
336.         beq $t1, '3', pnts_3
337.         nop
338.         beq $t1, '4', pnts_4
339.         nop
340.         beq $t1, '5', pnts_5
341.         nop
342.         beq $t1, '6', pnts_6
343.         nop
344.         beq $t1, '7', pnts_7
345.         nop
346.         beq $t1, '8', pnts_8
347.         nop
348.         beq $t1, '9', pnts_9
349.         nop
350.
351. pnts_0:
352.         j pnts_end
353.         nop
354. pnts_1:
355.         addi $v0, $v0, 1
356.         j pnts_end
357.         nop
358. pnts_2:
359.         addi $v0, $v0, 2
360.         j pnts_end
361.         nop
362. pnts_3:
363.         addi $v0, $v0, 3
```

```
364.         j pnts_end
365.         nop
366. pnts_4:
367.         addi $v0, $v0, 4
368.         j pnts_end
369.         nop
370. pnts_5:
371.         addi $v0, $v0, 5
372.         j pnts_end
373.         nop
374. pnts_6:
375.         addi $v0, $v0, 6
376.         j pnts_end
377.         nop
378. pnts_7:
379.         addi $v0, $v0, 7
380.         j pnts_end
381.         nop
382. pnts_8:
383.         addi $v0, $v0, 8
384.         j pnts_end
385.         nop
386. pnts_9:
387.         addi $v0, $v0, 9
388.         j pnts_end
389.         nop
390.
391. pnts_end:
392.
393.         add $s3, $s3, 1                                 # counter++
394.         la $s1, postfix
395.         add $s1, $s1, $s3
396.         lb $t1, 0($s1)
397.
398.         beq $t1, $zero, end_while_pnts                  # if $t1 is null then
   end_while_pnts
399.         nop
400.         beq $t1, ' ', end_while_pnts                    # if $t1 is space then
   end_while_pnts
401.         nop
402.
403.         mul $v0, $v0, 10
404.         j while_pnts
405.         nop
406.
407. end_while_pnts:
408.         add $a0, $zero, $v0
409.         jal push
410.         nop
411.         lw $ra, 0($sp)                                  # get return address
   from system stack
412.         jr $ra
413.         nop
414.
415. #---------------------------------------------------------------
416. #Function check_number
417. # @brief check if character is number or not
```

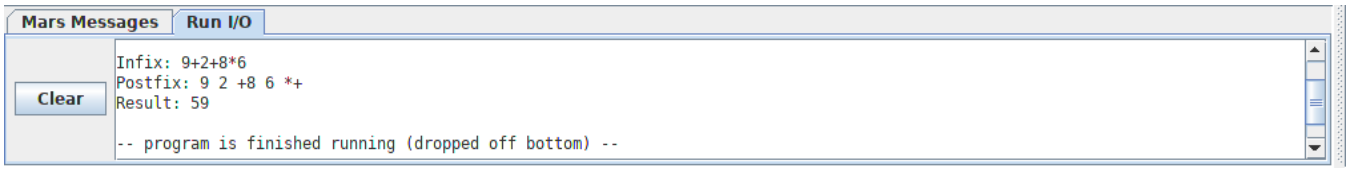
```
418. # @param[int] a0 : character to check
419. # @param[out] v0 : 1 = true; 0 = false
420. #------------------------------------------------------------------
421.
422. check_number:
423.         li $t8, '0'
424.         li $t9, '9'
425.
426.         beq $t8, $a0, check_number_true
427.         nop
428.         beq $t9, $a0, check_number_true
429.         nop
430.
431.         slt $v0, $t8, $a0
432.         beqz $v0, check_number_false
433.         nop
434.
435.         slt $v0, $a0, $t9
436.         beqz $v0, check_number_false
437.         nop
438.
439. check_number_true:
440.         li $v0, 1
441.         jr $ra
442.         nop
443. check_number_false:
444.         li $v0, 0
445.         jr $ra
446.         nop
447.
448.
449. #------------------------------------------------------------------
450. #Procedure pop
451. # @brief pop from stack at $s2
452. # @param[out] v0 : value to popped
453. #------------------------------------------------------------------
454. pop:
455.         lw $v0, -4($s2)
456.         sw $zero, -4($s2)
457.         add $s2, $s2, -4
458.         jr $ra
459.         nop
460.
461. #------------------------------------------------------------------
462. #Procedure push
463. # @brief push to stack at $s2
464. # @param[in] a0 : value to push
465. #------------------------------------------------------------------
466. push:
467.         sw $a0, 0($s2)
468.         add $s2, $s2, 4
469.         jr $ra
470.         nop
471.
472. end_while_postfix_stack:
473.
474. # add null at end of stack
```

```
475.
476. # print result
477.         la $a0, result_message
478.         li $v0, 4
479.         syscall
480.
481.         jal pop
482.         nop
483.         move $a0, $v0
484.         li $v0, 1
485.         syscall
486.
487.         la $a0, newLine
488.         li $v0, 4
489.         syscall
490. exit:
```

Kết quả hiển thị :



Input

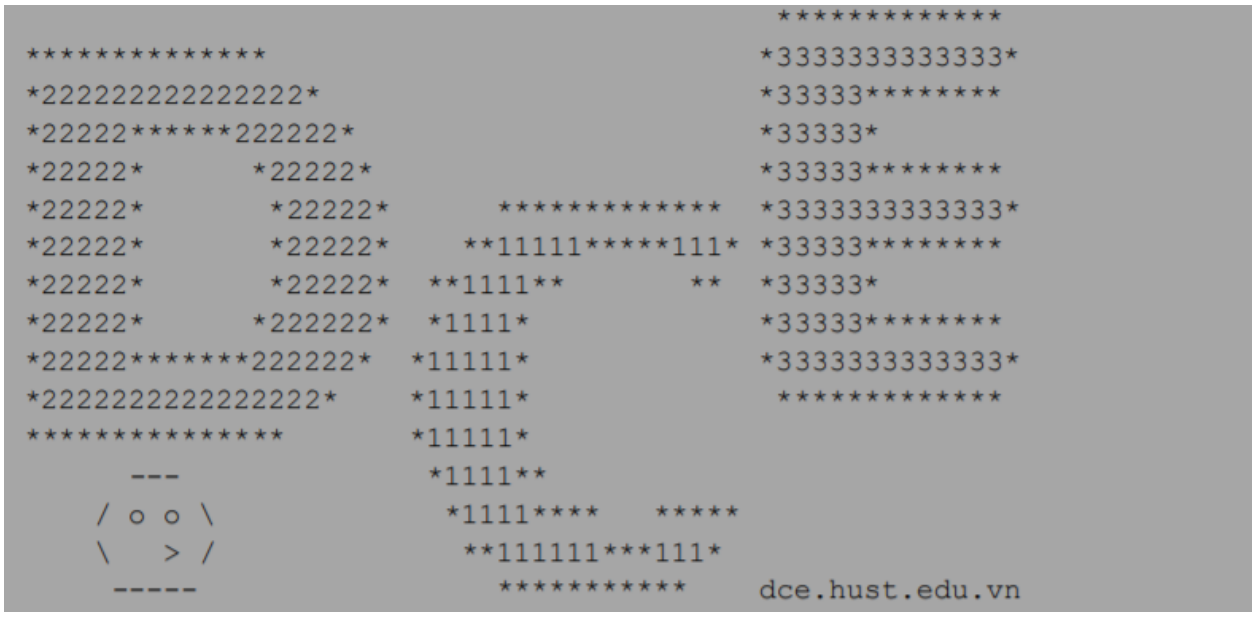


Output

Bài 9: Vẽ hình bằng kí tự ASCII

Sinh viên thực hiện: Trần Văn Thông – 20167386

<u>Đề bài</u>: Cho hình ảnh đã được chuyển thành các kí tự ASCII như hình vẽ. Đây là hình của chữ DCE có viền * và màu là các con số.



- Hãy hiển thị hình ảnh trên lên giao diện console (hoặc giao diện Display trong công cụ giả lập Keyboard and Display MMIO Simulator)
- Hãy sửa ảnh để các chữ cái DCE chỉ còn lại viền, không còn màu số ở giữa, cà hiển thị
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các họa tiết đính kèm cũng được phép di chuyển theo.
- Hãy nhập từ bàn phím kí tự màu cho chữ D, C, E rồi hiển thị hình ảnh trên màu mới.

Chú ý: ngoài vùng nhớ lớn chứa ảnh được chứa sẵn trong code, không được tạo thêm vùng nhớ mới để chứa ảnh hiệu chỉnh.

Trình bày:

- Tạo menu để dễ dàng thao tác hơn

Menu:

---------- Ve hinh bang ki tu ASCII----------

1. Hien thi hinh anh
2. Hien thi hinh anh ko co mau
3. Doi vi tri cua D va E
4. Doi mau
5. Thoat

Select option:

Thuật toán:

- Option 1: Sử dụng vòng lặp để in ra string của từng dòng, đến khi hết 16 dòng chuyển về Menu.
- Option2: Duyệt từng kí tự theo từng dòng, nếu không phải số (>0 hoặc >9) thì in ra kí tự đó. Nếu là chữ số thì thay chữ số đó bằng space (dấu cách) để xóa màu.
- Option3: Chia hình ảnh ra thành 4 phần:
    - Phần 1 (Chữ D): Từ cột 0 đến cột 23 ← Gán char ở cột 23 = NULL
    - Phần 2 (Chữ C): Từ cột 24 đến cột 44 ← Gán char ở cột 44 = NULL
    - Phần 3 (Chữ E) : Từ cột 45 đến cột 61 ← Gán char ở cột 61 = NULL
    - Phần 4: Các cột còn lại (space và \n)

    Sau đó in theo từng hàng lần lượt các kí tự từ cột 45 -> cột 61 (Chữ E),

    cột 24 -> cột 44 (Chữ C), cột 0 -> cột 23 (chữ D). Rồi trả lại các cột 23, 44, 61 về space.

- Option4: Sau khi bam Option 4 sẽ hiện ra màn hình lần lượt:
    4.1. Doi mau cho D (0->9):
    4.2. Doi mau cho C (0->9):
    4.3. Doi mau cho E (0->9):

Để người dùng chọn màu để đổi cho D, C, E.

Lưu các kí tự người dùng nhập vào rồi duyệt từng kí tự theo từng dòng.

Xét theo từng chữ cái, với chữ D (0->23) nếu gặp màu cũ (kí tự số cũ) thì đổi sang màu mới (kí tự số người dùng nhập). Tương tự với C và E.

- Option 5: Kết thúc chương trình

Mã nguồn:

```
1. .data
2.
3.   String0: .space 5000
4.   String1: .asciiz  "
   *************   \n"
5.   String2: .asciiz  " ****************
   *3333333333333*  \n"
6.   String3: .asciiz  " *2222222222222222*
   *33333********   \n"
7.   String4: .asciiz  " *22222*******222222*                          *33333*
   \n"
8.   String5: .asciiz  " *22222*      *222222*
   *33333********   \n"
9.   String6: .asciiz  " *22222*       *222222*      *************
   *3333333333333*  \n"
10.          String7: .asciiz  " *22222*       *222222*    **11111*****111*
   *33333********   \n"
11.          String8: .asciiz  " *22222*       *222222*   *1111**      ***
   *33333*          \n"
12.          String9: .asciiz  " *22222*      *222222*   *1111*
   *33333********   \n"
13.          String10: .asciiz " *22222*******222222*   *11111*
   *3333333333333*  \n"
14.          String11: .asciiz " *2222222222222222*     *11111*
   *************   \n"
15.          String12: .asciiz " ****************       *11111*
   \n"
16.          String13: .asciiz "      ---                *1111*
   \n"
17.          String14: .asciiz "    / o o \\              *1111****   *****
   \n"
18.          String15: .asciiz "    \\   > /               **111111***111*
   \n"
19.          String16: .asciiz "     -----                  ************
   dce.hust.edu.vn  \n"
20.
21.          Stack: .space 1000
22.          Message0:       .asciiz "------------Ve hinh bang ki tu ASCII--------
   ----\n"
23.          Option1:                .asciiz"1. Hien thi hinh anh\n"
24.          Option2:                .asciiz"2. Hien thi hinh anh ko co mau\n"
25.          Option3:                .asciiz"3. Doi vi tri cua D va E\n"
26.          Option4:                .asciiz"4. Doi mau\n"
27.          Exit_Menu:              .asciiz"5. Thoat\n"
28.          Select:         .asciiz"Select option: "
29.          StringD:                .asciiz"4.1. Doi mau cho D (0->9): "
30.          StringC:                .asciiz"4.2. Doi mau cho C (0->9): "
31.          StringE:                .asciiz"4.3. Doi mau cho E (0->9): "
32.  .text
33.  ####################################
34.
35.
36.
37.          li $t5 50 #t5 mau ban dau cua D
```

```
38.         li $t6 49 #t6 mau ban dau cua C
39.         li $t7 51 #t7 mau ban dau cua E
40. ####################################
41. main:
42.
43. ################################################ bat dau in man hinh menu
44.         la $a0, Message0       # In Menu
45.         li $v0, 4              # Goi print
46.         syscall
47.
48.         la $a0, Option1 # Option 1 In string
49.         li $v0, 4
50.         syscall
51.         la $a0, Option2 # Option 2 In string khong co mau
52.         li $v0, 4
53.         syscall
54.         la $a0, Option3 # Option 3 Doi vi tri cua D va E
55.         li $v0, 4
56.         syscall
57.         la $a0, Option4 # Option 4 Doi mau
58.         li $v0, 4
59.         syscall
60.         la $a0, Exit_Menu # Option 5 Thoat menu
61.         li $v0, 4
62.         syscall
63.         la $a0, Select  # Chon option
64.         li $v0, 4
65.         syscall
66. ########################################## ket thuc in menu
67.
68.         li $v0, 5        #Doc so duoc nhap tu ban phim roi luu vao $v0
69.         syscall
70. ########################################## bat dau xu ly cac case
71.         Case1menu:
72.                 addi $v1 $0 1 # if select option 1
73.                 bne $v0 $v1 Case2menu
74.                 j Menu1
75.         Case2menu:
76.                 addi $v1 $0 2 # if select option 2
77.                 bne $v0 $v1 Case3menu
78.                 j Menu2
79.         Case3menu:
80.                 addi $v1 $0 3 # if select option 3
81.                 bne $v0 $v1 Case4menu
82.                 j Menu3
83.         Case4menu:
84.                 addi $v1 $0 4 # if select option 4
85.                 bne $v0 $v1 Case5menu
86.                 j Menu4
87.         Case5menu:
88.                 addi $v1 $0 5 # if select option 5
89.                 bne $v0 $v1 defaultmenu
90.                 j Exit
91.         defaultmenu:
92.                 j main # Neu chon so khong hop le thi nhay ve Menu
93.
94. ############ In ####################
```

```
95.  Menu1:
96.          addi $s0, $0, 0# init count variable i = 0
97.          addi $s1, $0, 16       # 16 la tong so hang
98.          la $s2 , String1       # load string cua hang dau tien vao $s2
99.  Loop_Menu1:
100.         beq $s1, $s0, main # if print hoan thanh tro ve menu
101.         la $a0, ($s2)      # else in ra string cua hang dang luu o $s2
102.         li $v0, 4
103.         syscall
104.         addi $s2, $s2, 64  # nhay sang dia chi cua string cua hang tiep theo
   vi 1 hang co 64 ki tu
105.         addi $s0, $s0, 1   # next sang hang tiep theo
106.         j Loop_Menu1
107.
108. ############ In khong co mau (xoa mau) ################
109. Menu2:  addi $s0, $0, 0# variable i dem cac hang
110.         addi $s1, $0, 16 # so hang
111.         la $s2, String1# Gan $s2 cho xau cua hang 1
112.
113. Loop_Menu2:            #for i=0;i<16
114.         beq $s1, $s0, main
115.         addi $t0, $0, 0# variable j dem cot
116.         addi $t1, $0, 64 # so cot
117.
118. menu2_In1hang:
119.         beq $t1, $t0, menu2_Endline  # neu j==64 thi ket thuc 1 hang
120.         lb $t2, 0($s2) # load 1 ki tu
121.         bgt $t2, '0', menu2_compare_with_9 # if ki tu duoc luu o $t2 > '0'
   thi nhay vao ham menu2_Compare_With_9 de kiem tra tiep voi 9
122.         j menu2_print_char   #neu char < '0' thi in ra char
123. menu2_compare_with_9:
124.         bgt $t2, '9', menu2_print_char # if char > '9' thi in ra char
125.         addi $t2, $0, 0x20                  # neu char la number thi gan =
   space
126.         j menu2_print_char     # jump to menu 2
127. menu2_print_char:
128.         li $v0, 11 # in ki tu
129.         addi $a0,$t2,0 # print $t2
130.         syscall
131.         addi $s2 $s2 1 # ki tu tiep theo
132.         addi $t0, $t0, 1# Tang bien dem cot len 1
133.
134.         #beq $t2, '*', menu2_storeStar
135.         j menu2_In1hang
136.
137. menu2_Endline:
138.         addi $s0 $s0 1 # tang hang len 1
139.         j Loop_Menu2
140.
141. ################ Doi vi tri D va E ############
142. Menu3:  addi $s0, $0, 0# bien dem cac hang
143.         addi $s1, $0, 16 # so hang
144.         la $s2, String1 # Gan $s2 cho string cua hang 1
145. Loop2:  beq $s1, $s0, main  # neu doc het rows thi quay ve main
146.         # Phan thanh 4 vung
147.         sb $0 23($s2)      #gan ki tu o string[23] = null
148.         sb $0 44($s2)     #gan ki tu o string[44] = null
```

```
149.         sb $0 61($s2)      #gan ki tu o string[61] = null
150.         # Doi vi tri
151.         li $v0, 4
152.         la $a0 45($s2)    # in cac ki tu cua E ra dau tien bang cach doc
   string tu String[45] den ki tu null tai String[61]
153.         syscall
154.
155.         li $v0, 4
156.         la $a0 24($s2)     # in cac ki tu cua C ra bang cach doc string tu
   String[24] den ki tu null tai String[44]
157.         syscall
158.
159.         li $v0, 4
160.         la $a0 0($s2)      # in cac ki tu cua D ra bang cach doc string tu
   String[0] den ki tu null tai String[23]
161.         syscall
162.
163.         li $v0, 4         # in cac ki tu space va xuong dong` o cuoi line tu
   String [61] den ki tu null tai String [63]
164.         la $a0 62($s2)
165.         syscall
166.         # Chuoi co so
167.         addi $t1 $0 0x20
168.         sb $t1 23($s2)     # tra lai gia tri space cho cac gia tri bi gan la
   null luc truoc
169.         sb $t1 44($s2)
170.         sb $t1 61($s2)
171.
172.         addi $s0 $s0 1  # tang so hang len 1
173.         addi $s2 $s2 64 # doc dia chi cua String cua hang tiep theo
174.         j Loop2
175.
176. ############ doi mau cho chu ################
177. Menu4:
178. Input_D_Color: li      $v0, 4
179.                la      $a0, StringD   # hoi yeu cau nguoi dung nhap vao mau
   cho D
180.                syscall
181.
182.                li      $v0, 5          # nhan vao gia tri tu nguoi dung read
   integer
183.                syscall
184.
185.                blt     $v0, 0, Input_D_Color # neu gtri <0 thi yeu cau nguoi
   dung nhap lai
186.                bgt     $v0, 9, Input_D_Color # neu gtri >9 thi yeu cau nguoi
   dung nhap lai
187.
188.                addi    $s3, $v0, 48    # $s3 store D's color
189.
190. Input_C_Color: li      $v0, 4
191.                la      $a0, StringC   # hoi yeu cau nguoi dung nhap vao mau
   cho C
192.                syscall
193.
194.                li      $v0, 5          # nhan vao gia tri tu nguoi dung read
   integer
```

```
195.                syscall
196.
197.                blt     $v0, 0, Input_C_Color # neu gtri <0 thi yeu cau nguoi
   dung nhap lai
198.                bgt     $v0, 9, Input_C_Color # neu gtri >9 thi yeu cau nguoi
   dung nhap lai
199.
200.                addi    $s4, $v0, 48    # $s3 store C's color
201.
202. Input_E_Color: li      $v0, 4
203.                la      $a0, StringE    # hoi yeu cau nguoi dung nhap vao mau
   cho E
204.                syscall
205.
206.                li      $v0, 5          # nhan vao gia tri tu nguoi dung read
   integer
207.                syscall
208.
209.                blt     $v0, 0, Input_C_Color # neu gtri <0 thi yeu cau nguoi
   dung nhap lai
210.                bgt     $v0, 9, Input_C_Color # neu gtri >9 thi yeu cau nguoi
   dung nhap lai
211.
212.                addi    $s5, $v0, 48    # $s3 store E's color
213.
214.
215.        addi $s0, $0, 0# count rows
216.        addi $s1, $0, 16 # number rows
217.        la $s2,String1 # Doc vao String cua row dau tien
218. Loop_Change_Color:
219.                beq $s1, $s0, Update_ColorCode # neu doc het row thi luu lai
   gia tri cua new color roi quay ve main
220.                addi $t0, $0, 0 # doc tung ki tu
221.                addi $t1, $0, 64 # so ki tu max cua 1 row
222.
223. Loop_Row_Change_Color:
224.        beq $t1, $t0, Enddoimau_row # neu doc het row thi in row ra va next
   sang row tiep theo
225.        lb $t2, 0($s2)  # load char
226. CheckD: # char cua D tu String[0] den String[23]
227.        bgt     $t0, 23, CheckC # neu ki tu o column >23 thi check C
228.        beq     $t2, $t5, fixD # if current char == D base color thi doi mau
   D
229.        j Next_Char
230. CheckC: # char cua C tu String[24] den String[44].
231.        bgt     $t0, 44, CheckE # neu ki tu o column >44 thi check E
232.        beq     $t2, $t6, fixC # if current char == C base color thi doi mau
   C
233.        j Next_Char
234. CheckE: # char cua E tu String[45] den String[63]
235.        beq     $t2, $t7, fixE # if current char == E base color thi doi mau
   E
236.        j Next_Char
237.
238. fixD:   sb $s3 0($s2) # gan new color vao char hien tai
239.        j Next_Char
240. fixC:   sb $s4 0($s2) # gan new color vao char hien tai
```

```
241.         j Next_Char
242. fixE:   sb $s5 0($s2) # gan new color vao char hien tai
243.         j Next_Char
244. Next_Char:      addi $s2 $s2 1 # to next Char
245.                 addi $t0, $t0, 1 # increase count variable column
246.                 j Loop_Row_Change_Color
247. Enddoimau_row:
248.         li $v0, 4
249.         addi $a0 $s2 -64 # print string coulumn
250.         syscall
251.         addi $s0 $s0 1 # increase rows
252.         j Loop_Change_Color
253. Update_ColorCode:       # neu doc het row thi luu lai gia tri cua new color
   roi quay ve main
254.         move $t5 $s3
255.         move $t6 $s4
256.         move $t7 $s5
257.         j main
258.
259. Exit:
```

Kết quả hiển thị:

Menu

```
------------Ve Hinh bang ki tu ASCII------------
1. Hien thi hinh anh
2. Hien thi hinh anh ko co mau
3. Doi vi tri cua D va E
4. Doi mau
5. Thoat
```

Kết quả của Option 1

```
Select option: 1
                                          *************
*****************                         *3333333333333*
*2222222222222222*                        *33333********
*22222*******222222*                      *33333*
*22222*      *222222*                     *33333********
*22222*       *222222*      *************  *3333333333333*
*22222*       *222222*    **lllll*****lll* *33333********
*22222*       *222222*   *llll**      ***  *33333*
*22222*      *222222*   *llll*             *33333********
*22222*******222222*   *llllll*            *3333333333333*
*2222222222222222*     *llllll*             *************
*****************      *llllll*
     ---               *llll*
   / o o \             *llll****   *****
   \   > /              **lllllll***lll*
    -----                  ************    dce.hust.edu.vn
```

Kết quả của Option 2

```
Select option: 2
                                          *************
*****************                         *           *
*               *                         *    ********
*    *******      *                       *    *
*    *      *     *                       *    ********
*    *       *     *     *************     *           *
*    *       *     *   **     *****   *    *    ********
*    *       *     *   *   **     ***  *    *
*    *      *      *   *   *            *    ********
*    *******      *   *    *            *           *
*               *     *    *             *************
*****************     *    *
     ---              *    *
   / o o \            *    ****   *****
   \   > /            **     ***   *
    -----                 ************    dce.hust.edu.vn
```

Kết quả của Option 3

```
Select option: 3
 *************
*3333333333333*                    ****************
*33333********                     *2222222222222222*
*33333*                            *22222*******222222*
*33333********                     *22222*      *222222*
*3333333333333*      *************  *22222*       *222222*
*33333********     **lllll*****lll* *22222*       *222222*
*33333*           *llll**      ***  *22222*       *222222*
*33333********   *llll*             *22222*      *222222*
*3333333333333* *llllll*            *22222*******222222*
 *************  *llllll*            *2222222222222222*
                *llllll*            ****************
                 *llll*                  ---
                 *llll****   *****      / o o \
                  **lllllll***lll*      \   > /
dce.hust.edu.vn      ************        -----
```

Giao diện Menu sau khi chọn Option4

```
------------Ve Hinh bang ki tu ASCII------------
1. Hien thi hinh anh
2. Hien thi hinh anh ko co mau
3. Doi vi tri cua D va E
4. Doi mau
5. Thoat
Select option: 4
4.1. Doi mau cho D (0->9): 7
4.2. Doi mau cho C (0->9): 1
4.3. Doi mau cho E (0->9): 3
```

Kết quả của Option 4

```
                                           *************
*****************                         *3333333333333*
*7777777777777777*                        *33333********
*77777*******777777*                      *33333*
*77777*      *777777*                     *33333********
*77777*       *777777*      *************  *3333333333333*
*77777*       *777777*    **lllll*****lll* *33333********
*77777*       *777777*   *llll**      ***  *33333*
*77777*      *777777*   *llll*             *33333********
*77777*******777777*   *llllll*            *3333333333333*
*7777777777777777*     *llllll*             *************
*****************      *llllll*
     ---               *llll*
   / o o \             *llll****   *****
   \   > /              **lllllll***lll*
    -----                  ************    dce.hust.edu.vn
```