TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

BÁO CÁO

KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: **ThS. Nguyễn Đình Thuận**

Mã lớp: **107350**

Sinh viên thực hiện:

1. **Nguyễn Viết Ngọc Quang** **20163315**
2. **Phùng Thế Hùng** **20161984**

Hà Nội, tháng 5 năm 2019

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Điểm	**Lời phê của giảng viên**

BÁO CÁO

KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: **ThS. Nguyễn Đình Thuận**

Mã lớp: **107350**

Sinh viên thực hiện:

1. **Nguyễn Viết Ngọc Quang** **20163315**
2. **Phùng Thế Hùng** **20161984**

Hà Nội, tháng 5 năm 2019

MỤC LỤC

	Trang
Bài 4: Postscript CNC Marsbot	**2**
1. **Sinh viên thực hiện**	**2**
2. **Đề bài**	**2**
3. **Phân tích đề bài**	**2**
4. **Xây dựng thuật toán**	**3**
5. **Mã nguồn**	**3**
6. **Kết quả mô phỏng**	**8**
Bài 8: Mô phỏng ổ đĩa RAID 5	**10**
1. **Sinh viên thực hiện**	**10**
2. **Đề bài**	**10**
3. **Phân tích đề bài**	**10**
4. **Xây dựng thuật toán**	**10**
5. **Mã nguồn**	**10**
6. **Kết quả mô phỏng**	**21**

Bài 4: Postscript CNC Marsbot

1. Sinh viên thực hiện:

Nguyễn Viết Ngọc Quang – 20163315

2. Đề bài:

Máy gia công cơ khí chính xác **CNC Marsbot** được dùng để cắt tấm kim loại theo các đường nét được qui định trước. **CNC Marsbot** có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track).
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

- **<Góc chuyển động>**, **<Thời gian>**, **<Cắt/Không cắt>**
- Trong đó **<Góc chuyển động>** là góc của hàm HEADING của Marsbot.
- **<Thời gian>** là thời gian duy trì quá trình vận hành hiện tại.
- **<Cắt/Không cắt>** thiết lập lưu vết/không lưu vết.

Hãy lập trình để **CNC Marsbot** có thể:

- Thực hiện cắt kim loại như đã mô tả.
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn.
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.
- Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt).

3. Phân tích đề bài:

- Viết script theo cấu trúc **<Góc chuyển động>**, **<Thời gian>**, **<Cắt/Không cắt>** để gia công cắt các chữ mong muốn.
- Chương trình phải nhận được button trong bàn phím Key Matrix để biết thực hiện gia công theo script nào.
- Chương trình phải đọc được góc di chuyển rotate, thời gian di chuyển time, cắt hay không cắt track.
- Điều khiển Mars Bot để thực hiện gia công.
- Lặp lại chu kì đọc cấu trúc như vậy cho đến hết script để được chữ hoàn chỉnh.

4. Xây dựng thuật toán:

- Viết các script theo cấu trúc **<Góc chuyển động>**, **<Thời gian>**, **<Cắt/Không cắt>** để cắt các nét mong muốn. Mỗi nét (mỗi cấu trúc) ngăn cách nhau bởi dấu chấm phảy "**;**".
- Đọc vào nút button được chọn để thực hiện script tương ứng được gia công. Nếu phím đọc vào không phải phím 0, 4, 8, tiến hành đọc lại.
- Tiến hành đọc rotate – góc di chuyển.
- Tiến hành đọc time – thời gian di chuyển. Sử dụng sleep.
- Tiến hành đọc track – có lưu vết cắt hay không.
- Lặp lại đọc script để lấy cấu trúc tiếp theo cho đến khi gặp giá trị null.
- Khi gặp giá trị null của script, dừng chương trình.

5. Mã nguồn:

```
        # Mars Bot
                .eqv HEADING 0xffff8010
                .eqv LEAVETRACK 0xffff8020
                .eqv WHEREX 0xffff8030
                .eqv WHEREY 0xffff8040
                .eqv MOVING 0xffff8050
        # Key Matrix
                .eqv IN_ADDRESS_HEXA_KEYBOARD 0xffff0012
                .eqv OUT_ADDRESS_HEXA_KEYBOARD 0xffff0014

        .data
        # (rotate, time, 0=untrack | 1=track);
        # Numpad 0 => postscript-DCE
        script1: .asciiz
"90,3000,0;180,3000,0;180,11270,1;70,2000,1;50,2000,1;30,2000,1;10,2000,1;
350,2000,1;330,2000,1;310,2000,1;290,2000,1;90,11000,0;250,2000,1;230,200
0,1;210,2000,1;190,2000,1;170,2000,1;150,2000,1;130,2000,1;110,2000,1;90,8
000,0;270,5000,1;0,11200,1;90,5000,1;270,5000,0;180,5600,0;90,5000,1;90,20
00,0;"
        # Numpad 4 => postscript-QUANG
        script2: .asciiz
"90,4000,0;180,2000,0;250,1000,1;230,1000,1;210,1000,1;190,1000,1;170,100
0,1;150,1000,1;130,1000,1;110,1000,1;90,500,1;70,1000,1;50,1000,1;30,1000,1
;10,1000,1;350,1000,1;330,1000,1;310,1000,1;290,1000,1;270,500,1;180,4000,
0;90,1000,0;135,2100,1;90,1500,0;0,5300,0;180,4000,1;150,1000,1;120,1000,1
;90,500,1;60,1000,1;30,1000,1;0,4000,1;90,3000,0;200,5400,1;20,2700,0;90,19
00,1;340,2700,0;160,5400,1;90,1500,0;0,5200,1;150,5900,1;0,5200,1;90,4000,0
;110,1000,0;290,1000,1;270,500,1;250,1000,1;230,1000,1;210,1000,1;190,1000
,1;170,1000,1;150,1000,1;130,1000,1;110,1000,1;90,500,1;70,1000,1;0,2000,1;
270,1000,0;90,2000,1;90,1000,0;"
        # Numpad 8 => postscript-HUNG
        script3: .asciiz
"90,3000,0;180,3000,0;180,5300,1;0,2700,0;90,3000,1;0,2600,0;180,5300,1;90,
1500,0;0,5300,0;180,4000,1;150,1000,1;120,1000,1;90,500,1;60,1000,1;30,100
0,1;0,4000,1;90,1500,0;180,5200,1;0,5200,0;150,5900,1;0,5200,1;90,4000,0;11
0,1000,0;290,1000,1;270,500,1;250,1000,1;230,1000,1;210,1000,1;190,1000,1;
170,1000,1;150,1000,1;130,1000,1;110,1000,1;90,500,1;70,1000,1;0,2000,1;27
0,1000,0;90,2000,1;90,1000,0;"

        .text
```

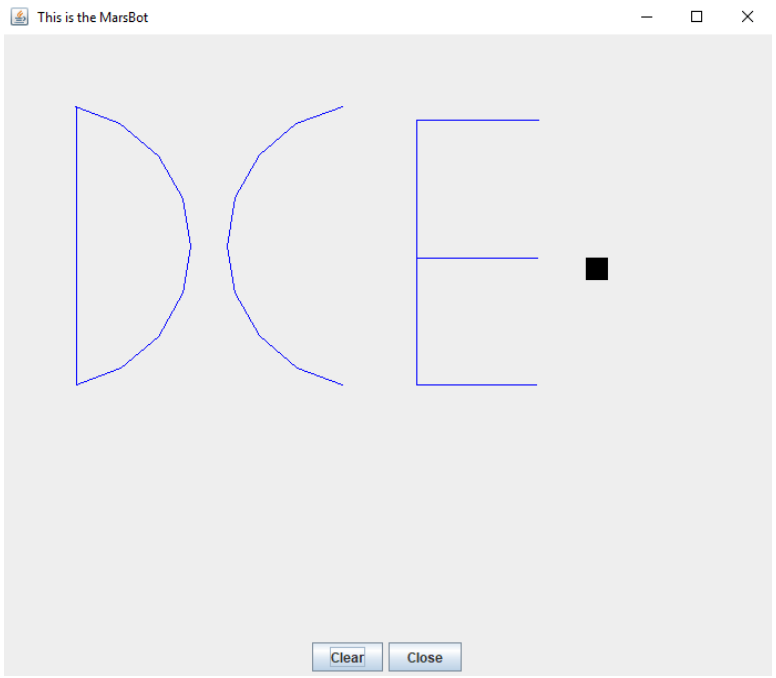
```
41          # < Xu ly tren Key Matrix >
42                  li $t3, IN_ADDRESS_HEXA_KEYBOARD
43                  li $t4, OUT_ADDRESS_HEXA_KEYBOARD
44          polling:
45                  li $t5, 0x01                            # Hang 1 cua Key matrix
46                  sb $t5, 0($t3)
47                  lb $a0, 0($t4)
48                  bne $a0, 0x11, NOT_NUMPAD_0         # Button 0 gia tri 0x11
49                  la $a1, script1
50                  j START
51          NOT_NUMPAD_0:
52                  li $t5, 0x02                            # Hang 2 cua Key matrix
53                  sb $t5, 0($t3)
54                  lb $a0, 0($t4)
55                  bne $a0, 0x12, NOT_NUMPAD_4         # Button 8 gia tri 0x12
56                  la $a1, script2
57                  j START
58          NOT_NUMPAD_4:
59                  li $t5, 0x04                            # Hang 3 cua Key matrix
60                  sb $t5, 0($t3)
61                  lb $a0, 0($t4)
62                  bne $a0, 0x14, COME_BACK            # Button 8 gia tri 0x14
63                  la $a1, script3
64                  j START
65          COME_BACK:
66                  j polling       # Khi 0, 4, 8 khong duoc chon -> Quay lai doc so
67          tiep
68
69          # < Xu li Mars Bot >
70          START:
71                  jal GO
72          READ_SCRIPT:
73                  addi $t0, $zero, 0                  # Luu gia tri rotate
74                  addi $t1, $zero, 0                  # Luu gia tri time
75
76          READ_ROTATE:
77                  add $t7, $a1, $t6                   # Dich bit
78                  lb $t5, 0($t7)                          # Doc cac ki tu cua
79  script
80                  beq $t5, 0, END                     # Ket thuc script khi gap null
```

```
81                  beq $t5, 44, READ_TIME          # Gap ki tu ','
82                  mul $t0, $t0, 10
83                  addi $t5, $t5, -48              # So 0 co thu tu 48 trong bang ASCII
84                  add $t0, $t0, $t5               # Cong cac chu so lai voi nhau
85                  addi $t6, $t6, 1                # Tang so bit can dich chuyen len 1
86                  j READ_ROTATE                   # Quay lai doc tiep den khi gap dau ','
87
88          READ_TIME:
89                  add $a0, $t0, $zero
90                  jal ROTATE
91                  addi $t6, $t6, 1
92                  add $t7, $a1, $t6               # $a1 luu dia chi cua script
93                  lb $t5, 0($t7)
94                  beq $t5, 44, READ_TRACK
95                  mul $t1, $t1, 10
96                  addi $t5, $t5, -48
97                  add $t1, $t1, $t5
98                  j READ_TIME                     # Quay lai doc tiep den khi gap dau ','
99
100         READ_TRACK:
101                 addi $v0, $zero, 32 # Cho Mars Bot tiep tuc chay bang cach
102 sleep
103                 add $a0, $zero, $t1 # Thoi gian sleep $t1
104                 addi $t6, $t6, 1
105                 add $t7, $a1, $t6
106                 lb $t5, 0($t7)
107                 addi $t5, $t5, -48
108                 beq $t5, $zero, CHECK_UNTRACK       # 0=untrack | 1=track
109                 jal UNTRACK
110                 jal TRACK
111                 j INCREAMENT
112
113         CHECK_UNTRACK:
114                 jal UNTRACK
115
116         INCREAMENT:
117                 syscall
118                 addi $t6, $t6, 2                    # Bo qua dau ';'
119                 j READ_SCRIPT
120
```

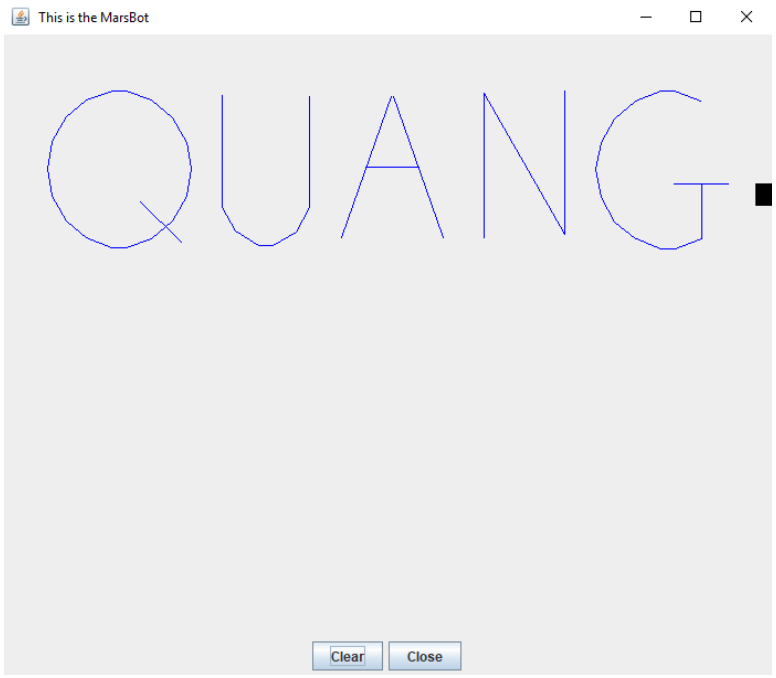
```
GO:
        li $at, MOVING
        addi $k0, $zero, 1
        sb $k0, 0($at)
        jr $ra

STOP:
        li $at, MOVING
        sb $zero, 0($at)
        jr $ra

TRACK:
        li $at, LEAVETRACK
        addi $k0, $zero, 1
        sb $k0, 0($at)
        jr $ra

UNTRACK:
        li $at, LEAVETRACK
        sb $zero, 0($at)
        jr $ra

ROTATE:
        li $at, HEADING
        sw $a0, 0($at)
        jr $ra

END:
        jal STOP
        li $v0, 10
        syscall
        j polling
```

6. Kết quả mô phỏng:

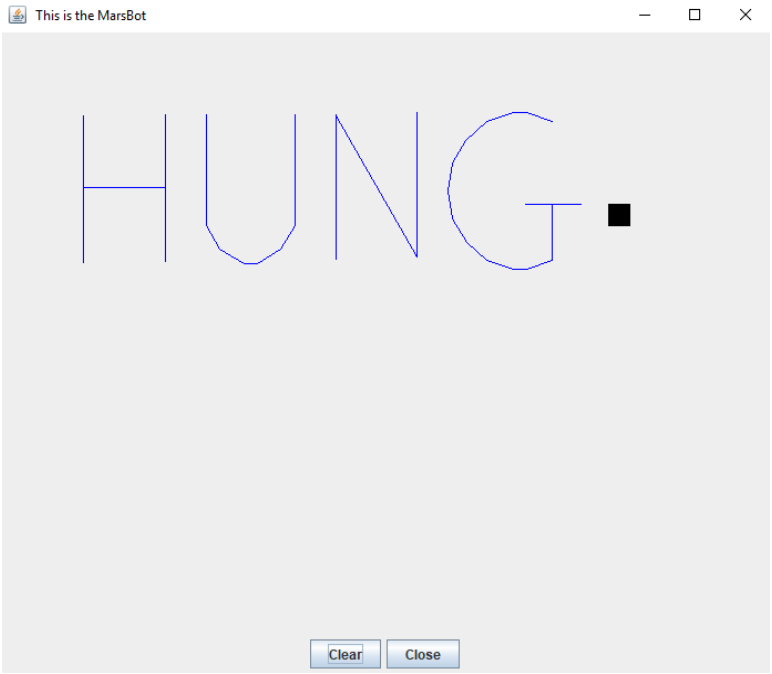
- **script1: Numpad 0 => DCE**



- **script2: Numpad 4 => QUANG**



- **script3: Numpad 8 => HUNG**



Bài 8: Mô phỏng ổ đĩa RAID 5

1. Sinh viên thực hiện:

Phùng Thế Hùng – 20161984

2. Đề bài:

Hệ thống ổ đĩa **RAID5** cần tối thiểu 3 ổ đĩa cứng, trong đó phần dữ liệu parity sẽ được chứa lần lượt lên 3 ổ đĩa như trong hình bên. Hãy viết chương trình mô phỏng hoạt động của **RAID5** với 3 ổ đĩa, với giả định rằng, mỗi block dữ liệu có 4 kí tự. Giao diện như trong minh họa dưới. *Giới hạn chuỗi kí tự nhập vào có độ dài là bội của 8.*

3. Phân tích đề bài:

- Nhập vào chuỗi ký tự có độ dài là bội của 8 rồi tiến hành lưu dữ liệu vào 3 disk theo hệ thống RAID5.
- Block 4 bytes đầu sẽ lưu vào disk 1, block 4 bytes tiếp theo sẽ lưu vào disk 2, dữ liệu lưu vào disk 3 là 4 byte parity được tính từ 2 block đầu tiên theo toán tử XOR. 8 bytes tiếp theo lưu lần lượt vào disk 1 và disk 3, disk 2 lại được tính theo toán tử XOR từ 2 block trước. Cuối cùng là 8 bytes tiếp sẽ lưu lần lượt vào disk 2 và disk 3, disk1 lại được tính theo toán tử XOR từ 2 block trước.
  Cứ lần lượt tính theo chu kỳ như vậy.

4. Xây dựng thuật toán:

- Tiến hành nhập một xâu, kiểm tra độ dài xem hợp lệ chưa.
- Thực hiện tách từng bytes để xử lí đưa vào disk.
- Xây dựng chương trình con để lấy mã hexa khi thực hiện toán tử XOR.
- Khai báo 1 mảng để lưu phần tử parity của từng disk.
- Chia làm 3 part rõ ràng như đã phân tích trên đề bài.

5. Mã nguồn:

```
1           .data
2               mes: .asciiz "Nhap vao chuoi ky tu : "
3               d1:     .space 4
4               d2:     .space 4
5               d3:     .space 4
6               str:    .space 1000
7               array: .space 32
8               error_length: .asciiz "Do dai chuoi khong hop le! Nhap lai!\n"
9               comma: .asciiz ","
10              enter: .asciiz "\n"
11              hex: .byte '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
12              m1: .asciiz "      Disk 1              Disk 2              Disk 3\n"
13              m2: .asciiz "----------------      ----------------      ----------------\n"
14              m3: .asciiz "|     "
15              m4: .asciiz "     |      "
16              m5: .asciiz "[[ "
17              m6: .asciiz "]]      "
18              try: .asciiz "Try again?"
19
20          .text
21              la $s1, d1
22              la $s2, d2
23              la $s3, d3
24              la $a2, array           # Dia chi mang chua parity
25
26          input:
27              li $v0, 4
28              la $a0, mes
29              syscall
30              li $v0, 8               #Nhap vao xau
31              la $a0, str
32              li $a1, 1000
33              syscall
34              move $s0, $a0
35
36      #Kiem tra do dai xau chia het cho 8
37          length:
38              addi $t0, $zero, 0    # t0 = length
39              addi $t1, $zero, 0    # t1 = index
40
```

```
41          checkText:
42                  add $t2, $s0, $t1                   # Dia chi cua str[i]
43                  lb $t3, 0($t2)                      # str[i]
44                  nop
45                  beq $t3, 10, testLength             # Gap ki tu xuong dong "\n"
46                  nop
47                  add $t0, $t0, 1                     # length++
48                  add $t1, $t1, 1                     # index++
49                  j checkText
50                  nop
51
52          testLength:
53                  move $t4, $t0
54                  and $t2, $t0, 0x0000000f            # Xoa tat ca cac byte cua $t0 ve
55      0, chi du lai byte cuoi
56                  bne $t2, 0, check8          # Kiem tra byte cuoi bang 0 hoac bang 8
57                  j head
58
59          check8:
60                  beq $t2, 8, head
61                  j errorLength
62
63          errorLength:
64                  li $v0, 4
65                  la $a0, error_length
66                  syscall
67                  j input
68
69      #Chuong trinh con lay ma hexa (parity)
70          parity:
71                  li $t9, 7
72
73          loopParity:
74                  blt $t9, $0, endParity
75                  sll $s7, $t9, 2                     # s7 = t5*4
76                  srlv $a0, $t8, $s7
77                  andi $a0, $a0, 0x0000000f           # Lay byte cuoi cua a0
78                  la $t7, hex
79                  add $t7, $t7, $a0
80                  bgt $t9, 1, nextChar
```

```
81                  lb $a0, 0($t7)
82                  li $v0, 11
83                  syscall
84
85              nextChar:
86                  add $t9, $t9, -1
87                  j loopParity
88
89              endParity:
90                  jr $ra
91
92          #Chuong trinh chinh
93              head:
94                  li $v0, 4
95                  la $a0, m1
96                  syscall
97                  li $v0, 4
98                  la $a0, m2
99                  syscall
100         #Khoi thu nhat
101             part1:
102                 addi $t1, $zero, 0
103                 addi $t5, $zero, 0
104                 addi $t8, $zero, 0
105                 la $s1, d1
106                 la $s2, d2
107                 la $a2, array
108             p11:
109                 lb $t2, ($s0)
110                 addi $t0, $t0, -1
111                 sb $t2, ($s1)
112             p12:
113                 addi $s4, $s0, 4
114                 lb $t3, ($s4)
115                 addi $t0, $t0, -1
116                 sb $t3, ($s2)
117             p13:
118                 xor $a3, $t2, $t3
119                 sw $a3, ($a2)
120                 addi $a2, $a2, 4
```

```
121             addi $t1, $t1, 1
122             addi $s0, $s0, 1
123             addi $s1, $s1, 1
124             addi $s2, $s2, 1
125             bgt $t1, 3, reset1
126             j p11
127         reset1:
128             la $s1, d1
129             la $s2, d2
130
131         print11:
132             li $v0, 4
133             la $a0, m3
134             syscall
135         print12:
136             lb $a0, ($s1)
137             li $v0, 11
138             syscall
139             addi $t5, $t5, 1
140             addi $s1, $s1, 1
141             bgt $t5, 3, mid11
142             j print12
143         mid11:
144             li $v0, 4
145             la $a0, m4
146             syscall
147             li $v0, 4
148             la $a0, m3
149             syscall
150         print13:
151             lb $a0, ($s2)
152             li $v0, 11
153             syscall
154             addi $t8, $t8, 1
155             addi $s2, $s2, 1
156             bgt $t8, 3, mid12
157             j print13
158         mid12:
159             li $v0, 4
160             la $a0, m4
```

```
161                 syscall
162                 li $v0, 4
163                 la $a0, m5
164                 syscall
165                 la $a2, array
166                 addi $t5, $zero, 0
167             print14:
168                 lb $t8, ($a2)
169                 jal parity
170                 li $v0, 4
171                 la $a0, comma
172                 syscall
173                 addi $t5, $t5, 1
174                 addi $a2, $a2, 4
175                 bgt $t5, 2, end1      # in ra 3 parity dau co dau ",", parity cuoi cung
176         khong co
177                 j print14
178             end1:
179                 lb $t8, ($a2)
180                 jal parity
181                 li $v0, 4
182                 la $a0, m6
183                 syscall
184                 li $v0, 4
185                 la $a0, enter
186                 syscall
187                 beq $t0, 0, exit1
188         #----------------------
189             part2:
190                 la $s1, d1
191                 la $s3, d3
192                 la $a2, array
193                 addi $s0, $s0, 4
194                 addi $t1, $zero, 0
195             p21:
196                 lb $t2, ($s0)
197                 addi $t0, $t0, -1
198                 sb $t2, ($s1)
199             p23:
200                 addi $s4, $s0, 4
```

```
201             lb $t3, ($s4)
202             addi $t0, $t0, -1
203             sb $t3, ($s3)
204         p22:
205             xor $a3, $t2, $t3
206             sw $a3, ($a2)
207             addi $a2, $a2, 4
208             addi $t1, $t1, 1
209             addi $s0, $s0, 1
210             addi $s1, $s1, 1
211             addi $s3, $s3, 1
212             bgt $t1, 3, reset2
213             j p21
214         reset2:
215             la $s1, d1
216             la $s3, d3
217             addi $t5, $zero, 0
218         print21:
219             li $v0, 4
220             la $a0, m3
221             syscall
222         print22:
223             lb $a0, ($s1)
224             li $v0, 11
225             syscall
226             addi $t5, $t5, 1
227             addi $s1, $s1, 1
228             bgt $t5, 3, mid21
229             j print22
230         mid21:
231             li $v0, 4
232             la $a0, m4
233             syscall
234             la $a2, array
235             addi $t5, $zero, 0
236             li $v0, 4
237             la $a0, m5
238             syscall
239         print23:
240             lb $t8, ($a2)
```

```
241                 jal parity
242                 li $v0, 4
243                 la $a0, comma
244                 syscall
245                 addi $t5, $t5, 1
246                 addi $a2, $a2, 4
247                 bgt $t5, 2, mid22
248                 j print23
249         mid22:
250                 lb $t8, ($a2)
251                 jal parity
252                 li $v0, 4
253                 la $a0, m6
254                 syscall
255                 li $v0, 4
256                 la $a0, m3
257                 syscall
258                 addi $t8, $zero, 0
259         print24:
260                 lb $a0, ($s3)
261                 li $v0, 11
262                 syscall
263                 addi $t8, $t8, 1
264                 addi $s3, $s3, 1
265                 bgt $t8, 3, end2
266                 j print24
267         end2:
268                 li $v0, 4
269                 la $a0, m4
270                 syscall
271                 li $v0, 4
272                 la $a0, enter
273                 syscall
274                 beq $t0, 0, exit1
275       #----------------------
276         part3:
277                 la $a2, array
278                 la $s2, d2
279                 la $s3, d3
280                 addi $s0, $s0, 4
```

```
281             addi $t1, $zero, 0
282         p32:
283             lb $t2, ($s0)
284             addi $t0, $t0, -1
285             sb $t2, ($s2)
286         p33:
287             addi $s4, $s0, 4
288             lb $t3, ($s4)
289             addi $t0, $t0, -1
290             sb $t3, ($s3)
291         p31:
292             xor $a3, $t2, $t3
293             sw $a3, ($a2)
294             addi $a2, $a2, 4
295             addi $t1, $t1, 1
296             addi $s0, $s0, 1
297             addi $s2, $s2, 1
298             addi $s3, $s3, 1
299             bgt $t1, 3, reset3
300             j p32
301         reset3:
302             la $s2, d2
303             la $s3, d3
304             la $a2, array
305             addi $t5, $zero, 0
306         print31:
307             li $v0, 4
308             la $a0, m5
309             syscall
310         print32:
311             lb $t8, ($a2)
312             jal parity
313             li $v0, 4
314             la $a0, comma
315             syscall
316             addi $t5, $t5, 1
317             addi $a2, $a2, 4
318             bgt $t5, 2, mid31
319             j print32
320         mid31:
```

```
321             lb $t8, ($a2)
322             jal parity
323             li $v0, 4
324             la $a0, m6
325             syscall
326             li $v0, 4
327             la $a0, m3
328             syscall
329             addi $t5, $zero, 0
330         print33:
331             lb $a0, ($s2)
332             li $v0, 11
333             syscall
334             addi $t5, $t5, 1
335             addi $s2, $s2, 1
336             bgt $t5, 3, mid32
337             j print33
338         mid32:
339             addi $t5, $zero, 0
340             li $v0, 4
341             la $a0, m4
342             syscall
343             li $v0, 4
344             la $a0, m3
345             syscall
346         print34:
347             lb $a0, ($s3)
348             li $v0, 11
349             syscall
350             addi $t5, $t5, 1
351             addi $s3, $s3, 1
352             bgt $t5, 3, end3
353             j print34
354
355         end3:
356             li $v0, 4
357             la $a0, m4
358             syscall
359             li $v0, 4
360             la $a0, enter
```

```
361                 syscall
362                 beq $t0, 0, exit1
363         #----------het 6 khoi dau-------
364         #----6 khoi tiep theo-----
365             nextPart:
366                 addi $s0, $s0, 4
367                 j part1
368             exit1:
369                 li $v0, 4
370                 la $a0, m2
371                 syscall
372                 j ask
373         #--------try again-----------
374             ask:
375                 li $v0, 50
376                 la $a0, try
377                 syscall
378                 beq $a0, 0, clear
379                 nop
380                 j exit
381                 nop
382         # Dua string ve trang thai ban dau de thuc hien lai qua trinh
383             clear:
384                 la $s0, str
385                 add $s3, $s0, $t4     # s3: Dia chi byte cuoi cung duoc su dung
386         trong string
387                 li $t2, 0
388             goAgain:
389                 sb $t2, ($s0)         # Set byte o dia chi s0 thanh 0
390                 nop
391                 addi $s0, $s0, 1
392                 bge $s0, $s3, input
393                 nop
394                 j goAgain
395                 nop
396         #--------end---------
397
398         exit:   li $v0, 10
399                 syscall
```

6. Kết quả mô phỏng:

- Chuỗi nhập vào: **12345**
- Chuỗi nhập vào: **124587sd**
- Chuỗi nhập vào: **ILOVEYOU3000HIHI**
- Chuỗi nhập vào: **DCE.****ABCD1234HUSTHUST**

```
Nhap vao chuoi ky tu : 12345
Do dai chuoi khong hop le! Nhap lai!
Nhap vao chuoi ky tu : 124587sd
      Disk 1                Disk 2                Disk 3
----------------      ----------------      ----------------
|     1245     |      |     87sd     |      [[ 09,05,47,51]]
----------------      ----------------      ----------------
Nhap vao chuoi ky tu : ILOVEYOU3000HIHI
      Disk 1                Disk 2                Disk 3
----------------      ----------------      ----------------
|     ILOV     |      |     EYOU     |      [[ 0c,15,00,03]]
|     3000     |      [[ 7b,79,78,79]]      |     HIHI     |
----------------      ----------------      ----------------
Nhap vao chuoi ky tu : DCE.****ABCD1234HUSTHUST
      Disk 1                Disk 2                Disk 3
----------------      ----------------      ----------------
|     DCE.     |      |     ****     |      [[ 6e,69,6f,04]]
|     ABCD     |      [[ 70,70,70,70]]      |     1234     |
[[ 00,00,00,00]]      |     HUST     |      |     HUST     |
----------------      ----------------      ----------------

-- program is finished running --
```