TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

*****

Computer Architecture (20212)
Final Project

GVHD: ThS. Lê Bá Vui

Mã lớp: 130938

Thực hành KTMT: IT3280

Nhóm 5: **Trần Huy Hoàng** **- 20194575**

Trịnh Quốc Công **- 20194495**

A. Phân công bài tập

Trần Huy Hoàng - Bài 10

Trịnh Quốc Công - Bài 9

B. Thực hiện bài tập

I. Bài 10

1. Đề bài : Máy tính bỏ túi

Sử dụng 2 ngoại vi là bàn phím keypad và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, *, /,% với các toán hạng là số nguyên. Do trên bàn phím không có các phím trên nên sẽ dùng các phím:

- Bấm phím a để nhập phép tính +
- Bấm phím b để nhập phép tính –
- Bấm phím c để nhập phép tính *
- Bấm phím d để nhập phép tính /
- Bấm phím e để nhập phép tính %
- Bấm phím f để nhập phép =

Yêu cầu cụ thể như sau:

- Khi nhấn các phím số, hiển thị lên LED, do chỉ có 2 LED nên chỉ hiển thị 2 số cuối cùng. Ví dụ khi nhấn phím 1 → hiển thị 01. Khi nhấn thêm phím 2 → hiển thị 12. Khi nhấn thêm phím 3 → hiển thị 23.
- Sau khi nhập số, sẽ nhập phép tính + -* /%
- Sau khi nhấn phím f (dấu =) , tính toán và hiển thị kết quả lên LED.
- Có thể thực hiện các phép tính liên tiếp(tham khảo ứng dụng Calculator trên hệ điều hành Windows)

Chú ý: Do bài toán sẽ có rất nhiều trường hợp xảy ra, yêu cầu cơ bản là thực hiện được phép tính và hiển thị lên LED. Các yêu cầu về bắt lỗi, các trường hợp tràn số, ... là tùy chọn.

2. Công cụ sử dụng

Mars4_5

3. Các lệnh thực hiện

Common MIPS instructions.

Notes: *op*, *funct*, *rd*, *rs*, *rt*, *imm*, *address*, *shamt* refer to fields in the instruction format. The program counter PC is assumed to point to the next instruction (usually 4 + the address of the current instruction). M is the byte-addressed main memory.

Assembly instruction	Instr. format	*op* *op*/*funct*	Meaning	Comments
`add $rd, $rs, $rt`	R	0/32	`$rd = $rs + $rt`	Add contents of two registers
`sub $rd, $rs, $rt`	R	0/34	`$rd = $rs - $rt`	Subtract contents of two registers
`addi $rt, $rs, imm`	I	8	`$rt = $rs + imm`	Add signed constant
`addu $rd, $rs, $rt`	R	0/33	`$rd = $rs + $rt`	Unsigned, no overflow
`subu $rd, $rs, $rt`	R	0/35	`$rd = $rs - $rt`	Unsigned, no overflow
`addiu $rt, $rs, imm`	I	9	`$rt = $rs + imm`	Unsigned, no overflow
`mfc0 $rt, $rd`	R	16	`$rt = $rd`	*rd* = coprocessor register (e.g. epc, cause, status)
`mult $rs, $rt`	R	0/24	`Hi, Lo = $rs * $rt`	64 bit signed product in Hi and Lo
`multu $rs, $rt`	R	0/25	`Hi, Lo = $rs * $rt`	64 bit unsigned product in Hi and Lo
`div $rs, $rt`	R	0/26	`Lo = $rs / $rt, Hi = $rs mod $rt`	
`divu $rs, $rt`	R	0/27	`Lo = $rs / $rt, Hi = $rs mod $rt` (unsigned)	
`mfhi $rd`	R	0/16	`$rd = Hi`	Get value of Hi
`mflo $rd`	R	0/18	`$rd = Lo`	Get value of Lo
`and $rd, $rs, $rt`	R	0/36	`$rd = $rs & $rt`	Logical AND
`or $rd, $rs, $rt`	R	0/37	`$rd = $rs | $rt`	Logical OR
`andi $rt, $rs, imm`	I	12	`$rt = $rs & imm`	Logical AND, unsigned constant
`ori $rt, $rs, imm`	I	13	`$rt = $rs | imm`	Logical OR, unsigned constant
`sll $rd, $rs, shamt`	R	0/0	`$rd = $rs << shamt`	Shift left logical (shift in zeros)
`srl $rd, $rs, shamt`	R	0/2	`$rd = $rs >> shamt`	Shift right logical (shift in zeros)
`lw $rt, imm($rs)`	I	35	`$rt = M[$rs + imm]`	Load word from memory
`sw $rt, imm($rs)`	I	43	`M[$rs + imm] = $rt`	Store word in memory
`lbu $rt, imm($rs)`	I	37	`$rt = M[$rs + imm]`	Load a single byte, set bits 8-31 of $*rt* to zero
`sb $rt, imm($rs)`	I	41	`M[$rs + imm] = $rt`	Store byte (bits 0-7 of $*rt*) in memory
`lui $rt, imm`	I	15	`$rt = imm * 2^16`	Load constant in bits 16-31 of register $*rt*
`beq $rs, $rt, imm`	I	4	`if($rs==$rt) PC = PC + imm` (PC always points to next instruction)	
`bne $rs, $rt, imm`	I	5	`if($rs!=$rt) PC = PC + imm` (PC always points to next instruction)	
`slt $rd, $rs, $rt`	R	0/42	`if($rs<$rt) $rd = 1; else $rd = 0`	
`slti $rt, $rs, imm`	I	10	`if($rs<imm) $rt = 1; else $rt = 0`	
`sltu $rd, $rs, $rt`	R	0/43	`if($rs<$rt) $rd = 1; else $rd = 0` (unsigned numbers)	
`sltiu $rt, $rs, imm`	I	11	`if($rs<imm) $rt = 1; else $rt = 0` (unsigned numbers)	
`j destination`	J	2	`PC = address*4`	Jump to *destination*, *address* = *destination*/4
`jal destination`	J	3	`$ra = PC; PC = address*4` (Jump and link, *address* = *destination*/4)	
`jr $rs`	R	0/8	`PC = $rs`	Jump to address stored in register $*rs*

4. Cách sử dụng chương trình

- Mở công cụ digital lab sim và chạy chương trình
- Nhấp vào các số trong bàn phím hexa và số tương ứng sẽ xuất hiện trên màn hình led bảy đoạn (chỉ hiển thị 2 số cuối)
- Chọn một phép tính (a – phép cộng, b- phép trừ, c – phép nhân, d – phép chia)
- Nhập số thứ 2, nếu người dùng không nhấp vào phím số, thì số đó sẽ được mặc định là 0
- Nhấp vào f (dấu =) để tính toán biểu thức, 2 chữ số cuối cùng của kết quả sẽ được hiển thị trên màn hình led bảy đoạn. Toàn bộ phép toán sẽ được ghi vào bảng điều khiển

5. Giải pháp

Chương trình chính chỉ đơn giản là một vòng lặp vô hạn. Một giá trị, 0x80, được lưu trữ vào IN_ADDRESS_HEXA_KEYBOARD để cho phép ngắt trên mỗi lần nhấn phím. Khi nhấn một phím, chương trình sẽ chuyển đến phần .ktext, đây là quá trình của máy tính. Trong .ktext:

- Quét bàn phím và lấy mã của phím đã nhấn
- Chuyển đổi mã phím thành mã số và mã của led bảy đoạn
- Kiểm tra xem đó là một số hay một toán tử
    - Nếu là số:
        - Đẩy số vào bộ nhớ
        - Hiển thị nó ở màn hình bảy phân đoạn
        - Thoát khỏi trình xử lý
    - Nếu nó là một toán tử:
        - Nếu không phải là “=”, thì đưa nó vào bộ nhớ, đổi $s0 thành 1 (cho biết bây giờ số tiếp theo đi vào sẽ là toán hạng thứ hai), sau đó kết thúc ngoại lệ
        - Nếu là “=”, sẽ chuyển sang phần tiếp theo
- Sau khi nhấn =, chương trình sẽ hiển thị câu trả lời trên màn hình led bảy đoạn

6. Ý nghĩa thanh ghi

- $t0: LED trái
- $t5: LED phải
- $s0: kiểm tra loại biến nhập vào 0 : so, 1 : toan tu, 2 : terminate key
- $s1: số đang hiển thị ở LED phải
- $s2: số đang hiển thị ở LED trái
- $s3: kiểm tra loại toán tử, 1: cộng, 2 : trừ, 3 : nhân, 4 : chia
- $s4: số thứ nhất
- $s5: số thứ hai
- $s6: kết quả phép tính
- $t9: giá trị tạm thời để tính số nhập vào
- $t1: biến điều khiển hàng keyboard và enable keyboard interrupt
- $t2: biến chứa vị trí key nhập vào
- $t3: lưu giá trị dùng để enable keyboard interrupt và enable kiểm tra nhập từng hàng keyboard
- $t7: giá trị của số hiện trên LED
- $t4: byte hiển thị lên LED , zero → nine

7. Source Code

```
.data
zero:  .byte 0x3f
one:   .byte 0x6
two:   .byte 0x5b
three: .byte 0x4f
four:  .byte 0x66
five:  .byte 0x6d
six:   .byte 0x7d
seven: .byte 0x7
eight: .byte 0x7f
nine:  .byte 0x6f

mess1: .asciiz "khong the chia cho so 0 \n"

.eqv IN_ADDRESS_HEXA_KEYBOARD  0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD         0xFFFF0014
.eqv SEVENSEG_LEFT 0xFFFF0011                  # Dia chi cua den led 7 doan trai.
.eqv SEVENSEG_RIGHT 0xFFFF0010                 # Dia chi cua den led 7 doan phai
.text
main:
start:
```

```
	li $t0,SEVENSEG_LEFT		# $t0: Bien gia tri so cua den LED trai
	li $t5,SEVENSEG_RIGHT		# $t5: Bien gia tri so cua den LED phai
	li $s0,0			# bien kiem tra loai bien nhap vao, 0: so, 1 :toan tu, 2:
terminate key
	li $s1,0			# so dang hien thi o led phai
	li $s2,0		# so dang hien thi o led trai
	li $s3,0			# bien kiem tra loai toan tu, 1:cong, 2:tru, 3:nhan, 4:chia
	li $s4,0			# so thu nhat
	li $s5,0		# so thu hai
	li $s6,0			# ket qua 2 so, cong ,tru, nhan, chia
	li $t9,0		# gia tri tam thoi

		li $t1, IN_ADDRESS_HEXA_KEYBOARD  #bien dieu khien hang keyboard va enable
keyboard interrupt
		li $t2, OUT_ADDRESS_HEXA_KEYBOARD #bien chua vi tri key nhap vao the hang va
cot
		li $t3, 0x80			# bit dung enable keyboard interrupt va enable kiem tra tung
hang keyboard
		sb $t3, 0($t1)
		li $t7,0			#gia tri cua so hien tren led
		li $t4,0		#byte hien thi len led ,zero->nine
storefirstvalue:
		li $t7,0		#gia tri cua bit can hien thi ban dau :0
		addi $sp,$sp,4			#day vao stack
	sb $t7,0($sp)
		lb $t4,zero			#bit dau tien can hien thi :0
		addi $sp,$sp,4			#day vao stack
	sb $t4,0($sp)
loop1:	#loop de doi nhap phim tu digital lab sim
		beq $s0,2,endloop1		#neu phim terminate(phim e) duoc bam ,thoat loop
		nop
		nop
		nop
		nop
		b loop1
		nop
		nop
		nop
		b loop1
		nop
		nop
		b loop1
endloop1:
end_main:
		li $v0,10
		syscall
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Xu ly khi xay ra interupt
# Hien thi so vua bam len den led 7 doan
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
```

```
process:
        jal checkrow1                   #check hang 1 xem co phim nao duoc nhap ko
        bnez $t3,convertrow1            #t3 != 0 --> co phim duoc nhap, convert phim do thanh bit
hien ra led
        nop
        jal checkrow2
        bnez $t3,convertrow2
        nop
        jal checkrow3
        bnez $t3,convertrow3
        nop
        jal checkrow4
        bnez $t3,convertrow4
checkrow1:
        addi $sp,$sp,4
      sw $ra,0($sp)             # luu ra lai vi ve sau co the doi
      li $t3,0x81         # Kich hoat interrupt, cho phep bam phim o hang 1
      sb $t3,0($t1)
      jal getvalue              # get vi tri ( hang va cot ) cua phim duoc nhap neu co
      lw $ra,0($sp)
      addi $sp,$sp,-4
      jr $ra
checkrow2:
        addi $sp,$sp,4
      sw $ra,0($sp)
        li $t3,0x82     # Kich hoat interrupt, cho phep bam phim o hang 2
      sb $t3,0($t1)
      jal getvalue
      lw $ra,0($sp)
      addi $sp,$sp,-4
      jr $ra
checkrow3:
        addi $sp,$sp,4
      sw $ra,0($sp)
        li $t3,0x84     # Kich hoat interrupt, cho phep bam phim o hang 3
      sb $t3,0($t1)
      jal getvalue
      lw $ra,0($sp)
      addi $sp,$sp,-4
      jr $ra
checkrow4:
        addi $sp,$sp,4
      sw $ra,0($sp)
        li $t3,0x88     # Kich hoat interrupt, cho phep bam phim o hang 4
      sb $t3,0($t1)
      jal getvalue
      lw $ra,0($sp)
      addi $sp,$sp,-4
      jr $ra
getvalue:
        addi $sp,$sp,4
```

```
        sw $ra,0($sp)
        li $t2,OUT_ADDRESS_HEXA_KEYBOARD  #dia chi chua vi tri phim duoc nhap
        lb $t3,0($t2)                     #load vi tri phim duoc nhap
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
convertrow1:                    # convert tu vi tri sang bit de chuyen den led
            beq $t3,0x11,case_zero                      # 0x11 -->phim o hang 1 cot 1--> 0
            beq $t3,0x21,case_one
            beq $t3,0x41,case_two
            beq $t3,0xffffff81,case_three
case_zero:
            lb $t4,zero             #t4=zero (tuc = 0x3f, tong cac bit thanh ghi de tao thanh so 0 tren led)
            li $t7,0          #t7= 0
            j updatetmp
case_one:
            lb $t4,one
            li $t7,1
            j updatetmp
case_two:
            lb $t4,two
            li $t7,2
            j updatetmp
case_three:
            lb $t4,three
            li $t7,3
            j updatetmp
convertrow2:
            beq $t3,0x12,case_four
            beq $t3,0x22,case_five
            beq $t3,0x42,case_six
            beq $t3,0xffffff82,case_seven
case_four:
            lb $t4,four
            li $t7,4
            j updatetmp
case_five:
            lb $t4,five
            li $t7,5
            j updatetmp
case_six:
            lb $t4,six
            li $t7,6
            j updatetmp
case_seven:
            lb $t4,seven
            li $t7,7
            j updatetmp
convertrow3:
            beq $t3,0x14,case_eight
            beq $t3,0x24,case_nine
```

```
        beq $t3 0x44,case_a
        beq $t3 0xffffff84,case_b
case_eight:
        lb $t4,eight
        li $t7,8
        j updatetmp
case_nine:
        lb $t4,nine
        li $t7,9
        j updatetmp
case_a:#truong hop phim cong
        addi $a3,$zero,1
        addi $s0,$s0,1          #bien check phim nhap vao chuyen thanh 1(chung to nhap vao 1 toan
tu)
        addi $s3,$zero,1         #bien check loai toan tu chuyen thanh 1(tuc phep cong)

        j setfirstnumber        #chuyen den ham chuyen 2 byte dang hien tren 2 led thanh so de tinh
toan
case_b: #truong hop phim tru
        addi $a3,$zero,2
        addi $s0,$s0,1
        addi $s3,$zero,2
        j setfirstnumber
convertrow4:
        beq $t3,0x18,case_c
        beq $t3,0x28,case_d
        beq $t3,0x48,case_e
        beq $t3 0xffffff88,case_f
case_c: #truong hop phim nhan
        addi $a3,$zero,3
        addi $s0,$s0,1
        addi $s3,$zero,3
        j setfirstnumber
case_d: #truong hop phim chia
        addi $a3,$zero,4
        addi $s0,$s0,1
        addi $s3,$zero,4
        j setfirstnumber

case_e:  #truong hop terminate key
        addi $s0,$s0,2
        j finish
setfirstnumber:                 # ham tinh so dau tien hien thi tren led trong 2 so
        addi $s4, $t9, 0
        li $t9, 0
        j done
case_f:  #truong hop bam =
        addi $s5, $t9, 0

setsecondnumber:  #ham tinh so thu 2 dang hien thi tren led trong 2 so
        #mul $s5,$s2,10         # s5=s2*10+s1
```

```
        #add $s5,$s5,$s1
        beq $s3,1,cong        # s3=1--> cong
        beq $s3,2,tru
        beq $s3,3,nhan
        beq $s3,4,chia
cong:
        add $s6,$s5,$s4
        li $s3,0
        li $t9, 0
        j incong
        nop                # s6=s5+s4

incong:
        li $v0, 1
        move $a0, $s4
        syscall
        li $s4, 0


        li $v0, 11
        li $a0, '+'
        syscall

        li $v0, 1
        move $a0, $s5
        syscall
        li $s5, 0                    #reset $s5

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s6
        syscall
        nop
        #li $s4, $s6

        li $v0, 11
        li $a0, '\n'
        syscall
        li $s7,100
        div $s6,$s7
        mfhi $s6            # chi lay 2 chu so cuoi cua ket qua de in ra led
        j show_result_in_led       # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
        nop

tru:
        sub $s6,$s4,$s5    # s6=s4-s5
        li $s3,0
```

```
        li $t9, 0
        j intru
        nop
intru:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, '-'
        syscall

        li $v0, 1
        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, '\n'
        syscall
        j show_result_in_led       # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
        nop
nhan:
        mul $s6,$s4,$s5     # s6=s4*s5
        li $s3,0
        li $t9, 0
        j innhan
        nop
innhan:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, '*'
        syscall

        li $v0, 1
        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, '='
```

```
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, '\n'
        syscall
        li $s7,100
        div $s6,$s7
        mfhi $s6            # chi lay 2 chu so sau c?ng cua ket qua in ra
        j show_result_in_led       # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
        nop
chia:
        beq $s5,0,chia0
        li $s3,0
        div $s4,$s5         # s6=s4/s5
        mflo $s6
        mfhi $s7
        li $t9, 0
        j inchia
        nop
inchia:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, '/'
        syscall

        li $v0, 1
        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        li $v0, 11
        li $a0, 'r'
```

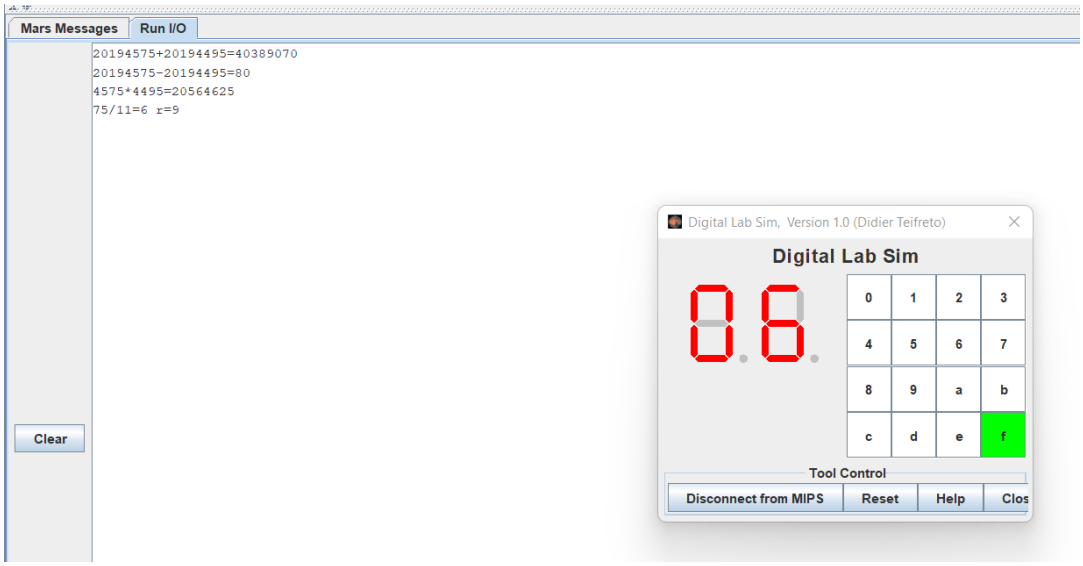
```
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s7
        syscall

        li $v0, 11
        li $a0, '\n'
        syscall
        j show_result_in_led       # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
        nop
chia0:
        li $v0, 55
        la $a0, mess1
        li $a1, 0
        syscall
        j resetled

show_result_in_led:    #ham chia ket qua thanh 2 chu so de hien thi len tung led
        li $t8,10
        div $s6,$t8    #s6/10
        mflo $t7       #t7 = result
        jal convert    #chuyen den ham chuyen t7 thanh bit hien thi len led
      #---------
      sb $t4,0($t0)  # hien thi len led trai
        add $sp,$sp,4
        sb $t7,0($sp)       #day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t4,0($sp)       #day bit nay vao stack
        add $s2,$t7,$zero   #s1 = gia tri bit led phai
        #----------
        mfhi $t7       #t7= remainder
        jal convert    #convert t7 thanh bit hien thi len led
      sb $t4,0($t5)  #hien thi len led phai
        add $sp,$sp,4
        sb $t7,0($sp)       # day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t4,0($sp)       # day bit nay vao stack
        add $s1,$t7,$zero   # s1 = gia tri bit led phai
      j resetled     # ham reset lai led
convert:
        addi $sp,$sp,4
      sw $ra,0($sp)
      beq $t7,0,case_0    # t7=0 -->ham chuyen 0 thanh bit zero hien thi len led
      beq $t7,1,case_1
      beq $t7,2,case_2
```

```
        beq $t7,3,case_3
        beq $t7,4,case_4
        beq $t7,5,case_5
        beq $t7,6,case_6
        beq $t7,7,case_7
        beq $t7,8,case_8
        beq $t7,9,case_9
case_0:#ham chuyen 0 thanh bit zero hien thi len led
            lb $t4,zero    #t4=zero
            j finishconvert #ket thuc
case_1:
            lb $t4,one
            j finishconvert
case_2:
            lb $t4,two
            j finishconvert
case_3:
            lb $t4,three
            j finishconvert
case_4:
            lb $t4,four
            j finishconvert
case_5:
            lb $t4,five
            j finishconvert
case_6:
            lb $t4,six
            j finishconvert
case_7:
            lb $t4,seven
            j finishconvert
case_8:
            lb $t4,eight
            j finishconvert
case_9:
            lb $t4,nine
            j finishconvert

finishconvert:
            lw $ra,0($sp)
            addi $sp,$sp,-4
            jr $ra
updatetmp:
             mul $t9, $t9, 10
             add $t9, $t9, $t7
done:
            beq $s0,1,resetled   # s0=1-->toan tu-->chuyen den ham reset led
            nop
loadtoleftled:   # ham hien thi bit len led trai
            lb $t6,0($sp)       #load bit hien thi led tu stack
            add $sp,$sp,-4
```

```
        lb $t8,0($sp)       #load gia tri cua bit nay
        add $sp,$sp,-4
        add $s2,$t8,$zero   #s2 = gia tri bit led trai
        sb $t6,0($t0)       # hien thi len led trai
loadtorightled:# ham hien thi bit len led phai
        sb $t4,0($t5)       # hien thi bit len led phai
        add $sp,$sp,4
        sb $t7,0($sp)       #day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t4,0($sp)       #day bit nay vao stack
        add $s1,$t7,$zero   #s1 = gia tri bit led phai
        j finish
resetled:
        li $s0,0          #s0=0--> doi nhap so tiep theo trong 2 so
      li $t8,0
        addi $sp,$sp,4
      sb $t8,0($sp)
      lb $t6,zero       # day bit zero vao stack
        addi $sp,$sp,4
      sb $t6,0($sp)
finish:
        j end_exception
        nop
end_exception:
        # return to start of the loop instead of where the interrupt occur, since the loop doesn't do
meaningful thing
        la $a3, loop1
        mtc0 $a3, $14
        eret
```

8. *Hình ảnh kết quả mô phỏng*

- Các phép toán cộng, trừ, nhân, chia

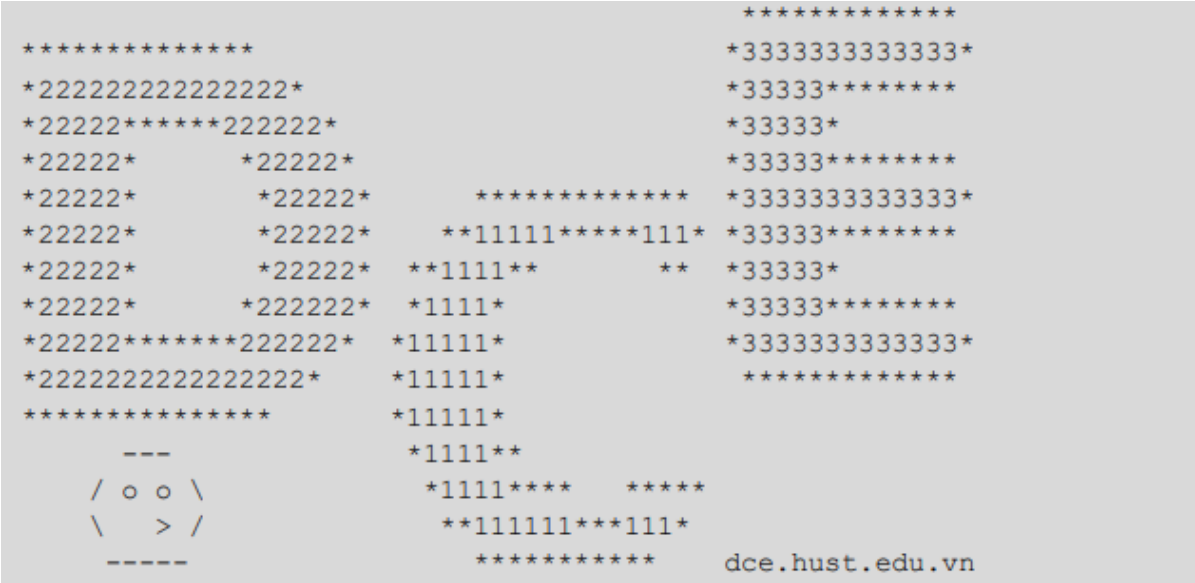


- Trường hợp chia cho 0

II. Bài 9

1. Đề bài: Vẽ hình bằng kí tự ASCII

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, và hiển thị
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các hoạt 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 ảnh trên với màu mới.

2. Công cụ sử dụng

3. Phân tích cách thực hiện

Dữ liệu hình ảnh sẽ được tổ chức thành 16 dòng, mỗi dòng có 62 kí tự

- Hiển thị hình ảnh
  - Chức năng in ra màn hình sẽ in lần lượt từng dòng dữ liệu để tạo thành hình ảnh
- Hiển thị hình ảnh chỉ có viền, không có màu
  - Duyệt từng kí tự theo từng dòng
  - Nếu gặp chữ số (≥ 0 & ≤ 9) thì thay kí tự đó bằng space để xóa màu

    Nếu gặp kí tự khác chữ số thì in ra như bình thường
- Hiển thị DCE
  - Chia hình ảnh thành 4 phần:
    - Chữ D: Từ cột 0 đến cột 21
    - Chữ C: Từ cột 22 đến cột 41
    - Chữ E: Từ cột 42 đến cột 59
    - Còn lại: Từ cột 60 đến 62
  - In từng kí tự theo từng dòng lần lượt các kí tự từ vị trí 42 => 59 (chữ E), sau đó từ cột 22 => 41 (chữ C) , sau đó từ cột 0 => 21 (chữ D) và từ cột 60 => 62
- Đổi màu
  - Lưu các màu hiện tại của D, C, E lần lượt vào các thanh ghi t5, t6, t7
  - Nhập màu muốn thay đổi lần lượt cho D, C, E và lưu vào các thanh ghi s3, s4, s5. Nếu số nhập không phải màu từ 0 => 9 yêu cầu nhập lại

4. Source Code

```
.data
String1:  .asciiz "                                                  *************     \n"
String2:  .asciiz "  ****************                              *3333333333333*   \n"
String3:  .asciiz "  *2222222222222222*                            *33333********    \n"
String4:  .asciiz "  *22222*******222222*                          *33333*           \n"
String5:  .asciiz "  *22222*           *22222*                     *33333********    \n"
String6:  .asciiz "  *22222*            *22222*       *************    *3333333333333* \n"
String7:  .asciiz "  *22222*            *22222*     **11111*****111*   *33333********   \n"
String8:  .asciiz "  *22222*            *22222*   **1111**          **    *33333*           \n"
String9:  .asciiz "  *22222*           *22222*   *1111*                  *33333********   \n"
String10: .asciiz " *22222*******222222*   *11111*                    *3333333333333* \n"
String11: .asciiz " *2222222222222222*      *11111*                     *************    \n"
String12: .asciiz " ****************        *11111*                                      \n"
String13: .asciiz "            ---           *1111**                                    \n"
String14: .asciiz "         /  o o  \\          *1111****     *****                      \n"
String15: .asciiz "         \\     >  /          **111111***111*                        \n"
String16: .asciiz "           -----              ***********          dce.hust.edu.vn \n"

        Message0:       .asciiz "------------MENU-----------\n"
        Phan1:          .asciiz"1. In ra chu\n"
        Phan2:          .asciiz"2. In ra chu rong\n"
        Phan3:          .asciiz"3. Thay doi vi tri\n"
        Phan4:          .asciiz"4. Doi mau cho chu\n"
        Thoat:          .asciiz"5. Thoat\n"
        Nhap:           .asciiz"Nhap gia tri: "
        ChuD:           .asciiz"Nhap mau cho chu D(0->9): "
        ChuC:           .asciiz"Nhap mau cho chu C(0->9): "
        ChuE:           .asciiz"Nhap mau cho chu E(0->9): "
.text
#######################################
        li $t5, 50 #t5 mau chu hien tai cua chu D
        li $t6, 49 #t6 mau chu hien tai cua chu C
        li $t7, 51 #t7 mau chu hien tai cua chu E
#######################################
main:
        la $a0, Message0        # nhap menu
        li $v0, 4
        syscall

        la $a0, Phan1
        li $v0, 4
        syscall
        la $a0, Phan2
        li $v0, 4
        syscall
        la $a0, Phan3
        li $v0, 4
        syscall
```

```
        la $a0, Phan4
        li $v0, 4
        syscall
        la $a0, Thoat
        li $v0, 4
        syscall
        la $a0, Nhap
        li $v0, 4
        syscall

        li $v0, 5
        syscall

        Case1menu:
                addi $v1, $0, 1
                bne $v0, $v1, Case2menu
                j Menu1
        Case2menu:
                addi $v1, $0, 2
                bne $v0, $v1, Case3menu
                j Menu2
        Case3menu:
                addi $v1, $0, 3
                bne $v0, $v1, Case4menu
                j Menu3
        Case4menu:
                addi $v1, $0, 4
                bne $v0, $v1, Case5menu
                j Menu4
        Case5menu:
                addi $v1, $0, 5
                bne $v0, $v1, defaultmenu
                j Exit
        defaultmenu:
                j main
##############in ra ####################
Menu1:
        addi $t0, $0, 0 #bien dem =0
        addi $t1, $0, 16

        la $a0,String1
Loop:           beq $t1, $t0, main
                li $v0, 4
                syscall

                addi $a0, $a0, 62
                addi $t0, $t0, 1
                j Loop

############ bo het so o giua chi giu lai vien################
Menu2:          addi $s0, $0, 0 #bien dem tung hang =0
```

```
        addi $s1, $0, 16
        la $s2,String1  # $s2 la dia chi cua string1

Lap:    beq $s1, $s0, main
        addi $t0, $0, 0 # $t0 la bien dem tung ki tu cua 1 hang =0
        addi $t1, $0, 62 # $t1 max 1 hang la 62 ki tu

In1hang:
        beq $t1, $t0, End
        lb $t2, 0($s2)  # $t2 luu gia tri cua tung phan tu trong string1
#       li $a1 47       #so -1 tuong duong vs gia tri 47
#       li $a2 57       #so 9 tuong duong vs gia tri 57

        bgt     $t2, 47, Lonhon0 #neu lon hon 0 thi nhay den Lonhon0
        j Tmp
        Lonhon0:        bgt     $t2, 57, Tmp #neu lon hon 9 nua thi van ko doi
                        addi $t2, $0, 0x20 # thay doi $t2 thanh dau cach
                        j Tmp
Tmp:    li $v0, 11 # in tung ki tu
        addi $a0, $t2, 0
        syscall

        addi $s2, $s2, 1 #sang chu tiep theo
        addi $t0, $t0, 1# bien dem chu
        j In1hang
End:    addi $s0, $s0, 1 # tang bien dem hang len 1
        j Lap

##################doi vi tri chu #############
Menu3:          addi $s0, $0, 0#bien dem tung hang =0
        addi $s1, $0, 16
        la $s2,String1 #$s2 luu dia chi cua string1
Lap2:   beq $s1, $s0, main
        #tao thanh 3 string nho
        sb $0, 21($s2)
        sb $0, 41($s2)
        sb $0, 59($s2)
        #doi vi tri
        li $v0, 4
        la $a0, 42($s2) #in chu E
        syscall

        li $v0, 4
        la $a0, 22($s2) # in chu C
        syscall

        li $v0, 4
        la $a0, 0($s2) # in chu D
        syscall

        li $v0, 4
```

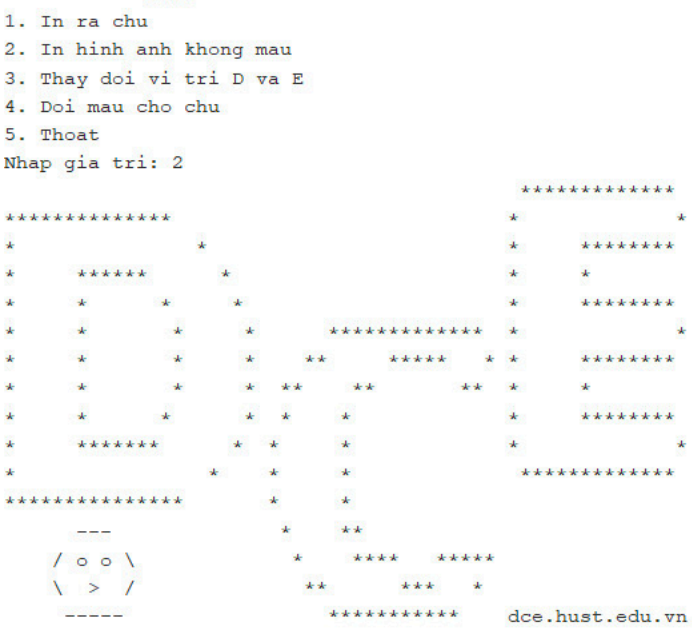
```
        la $a0, 60($s2)
        syscall
        # ghep lai thanh string ban dau
        addi $t1, $0, 0x20
        sb $t1, 21($s2)
        sb $t1, 41($s2)
        sb $t1, 59($s2)

        addi $s0, $s0, 1
        addi $s2, $s2, 62
        j Lap2

############ doi mau cho chu ################
Menu4:
NhapmauD:   li      $v0, 4
            la      $a0, ChuD
            syscall

            li      $v0, 5          # lay mau cua ki tu D
            syscall

            blt     $v0,0, NhapmauD
            bgt     $v0,9, NhapmauD

            addi    $s3, $v0, 48   #$s3 luu mau cua chu D
NhapmauC:   li      $v0, 4
            la      $a0, ChuC
            syscall

            li      $v0, 5          # lay mau cua ki tu C
            syscall

            blt     $v0, 0, NhapmauC
            bgt     $v0, 9, NhapmauC

            addi    $s4,  $v0, 48  #$s4 luu mau cua chu C
NhapmauE:   li      $v0, 4
            la      $a0, ChuE
            syscall

            li      $v0, 5          # lay mau cua ki tu E
            syscall

            blt     $v0, 0, NhapmauE
            bgt     $v0, 9, NhapmauE

            addi    $s5, $v0, 48   #$s5 luu mau cua chu E

        addi $s0, $0, 0 #bien dem tung hang =0
        addi $s1, $0, 16
        la $s2,String1  # $s2 la dia chi cua string1
```

```
        li $a1, 48 #gia tri cua so 0
        li $a2, 57 #gia tri cua so 9
#       li $t3 21
#       li $t4 43
Lapdoimau:    beq $s1, $s0, updatemau
              addi $t0, $0, 0 # $t0 la bien dem tung ki tu cua 1 hang =0
              addi $t1, $0, 62 # $t1 max 1 hang la 62 ki tu

In1hangdoimau:
        beq $t1, $t0, Enddoimau
        lb $t2, 0($s2)  # $t2 luu gia tri cua tung phan tu trong string1
        CheckD: bgt   $t0, 21, CheckC #kiem tra het chu D chua
              beq     $t2, $t5, fixD
              j Tmpdoimau
        CheckC: bgt   $t0, 41, CheckE #kiem tra het chu E chua
              beq     $t2, $t6, fixC
              j Tmpdoimau
        CheckE: beq   $t2, $t7, fixE
              j Tmpdoimau

fixD:   sb $s3 0($s2)
        j Tmpdoimau
fixC:   sb $s4 0($s2)
        j Tmpdoimau
fixE:   sb $s5 0($s2)
        j Tmpdoimau
Tmpdoimau:   addi $s2, $s2, 1 #sang chu tiep theo
              addi $t0, $t0, 1# bien dem chu
              j In1hangdoimau
Enddoimau:          li $v0, 4
              addi $a0, $s2, -62
              syscall
              addi $s0, $s0, 1 # tang bien dem hang len 1
              j Lapdoimau
updatemau: move $t5, $s3
        move $t6, $s4
        move $t7, $s5
        j main
Exit:
```

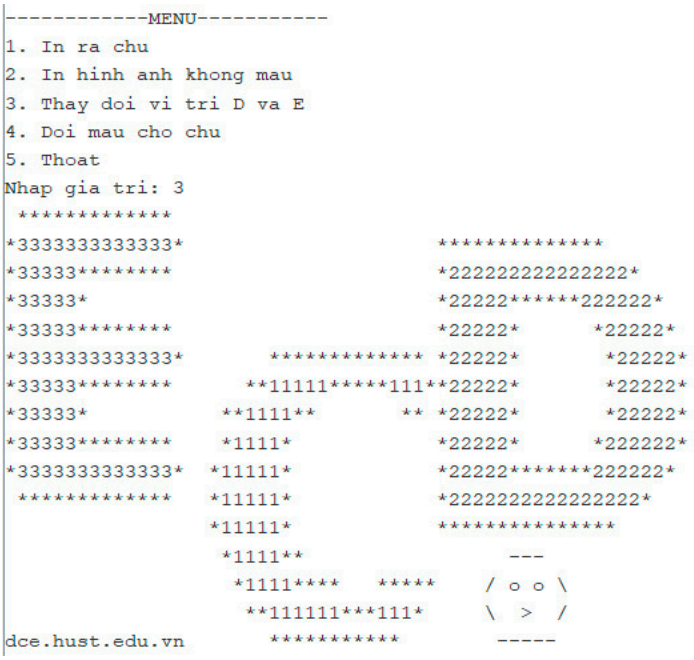
5. Hình ảnh kết quả mô phỏng

- Hiển thị hình ảnh

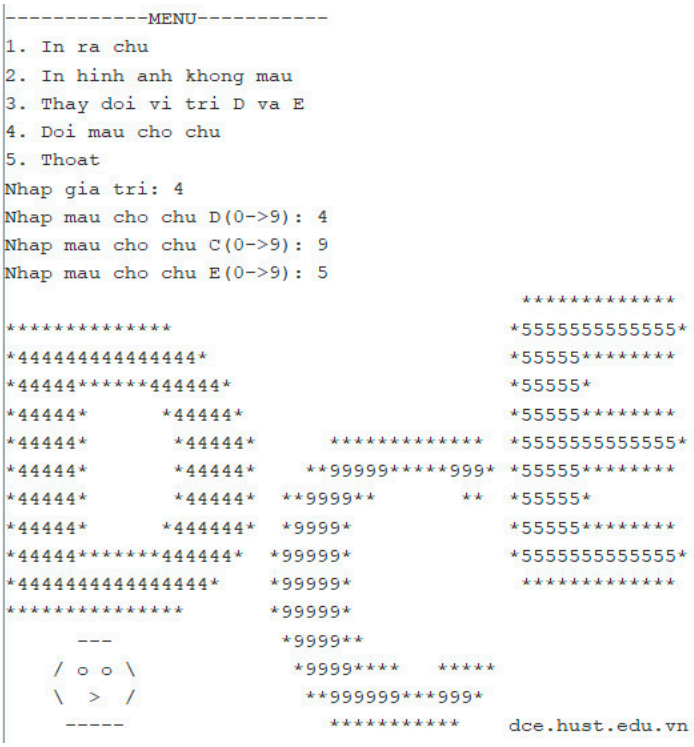
- In hình ảnh không màu



- Thay đổi vị trí E & D



- Thay đổi màu cho D, C, E



- Khi nhập sai số ứng với màu: (sẽ bắt nhập lại cho đến khi nhập đúng)