Báo cáo thực hành kiến trúc máy tính

Chu Việt Anh 20214983 - Đỗ Minh Đức 20215036

Ngày 7 tháng 7 năm 2023

I Tóm tắt báo cáo

Bài tập 2

Bài tập 3

Phần code đã có trong file.mps

II Bài tập 2

II.1 Đề bài

Viết một chương trình sử dụng MIPS để vẽ một quả bóng di chuyển trên màn hình mô phỏng Bitmap của Mars). Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại. Yêu cầu:

1. Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
2. Quả bóng là một đường tròn.

Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), Sang trái (A), Sang phải (D) trong bộ giả lập Keyboard and Display MMIO Simulator). Tốc độ bóng di chuyển là không đổi. Vị trí bóng ban đầu ở giữa màn hình.

II.2 Ý tưởng

Để làm một đối tượng di chuyển ta sẽ xóa đi đối tượng ở vị trí cũ và vẽ đối tượng vào vị trí mới Để xóa đối tượng ta cần vẽ đối tượng đó trùng với màu của màu nền

```
Bước 1: Vẽ hình tròn trên bitmap bằng thuật toán Bresenham's
Midpoint Circle Drawing
Bước 2: Thiết lập các giá trị để khi nhập vào bàn phím thì sẽ
xóa hình tròn ở vị trí hiện tại và vẽ hình tròn ở vị trí mới
```

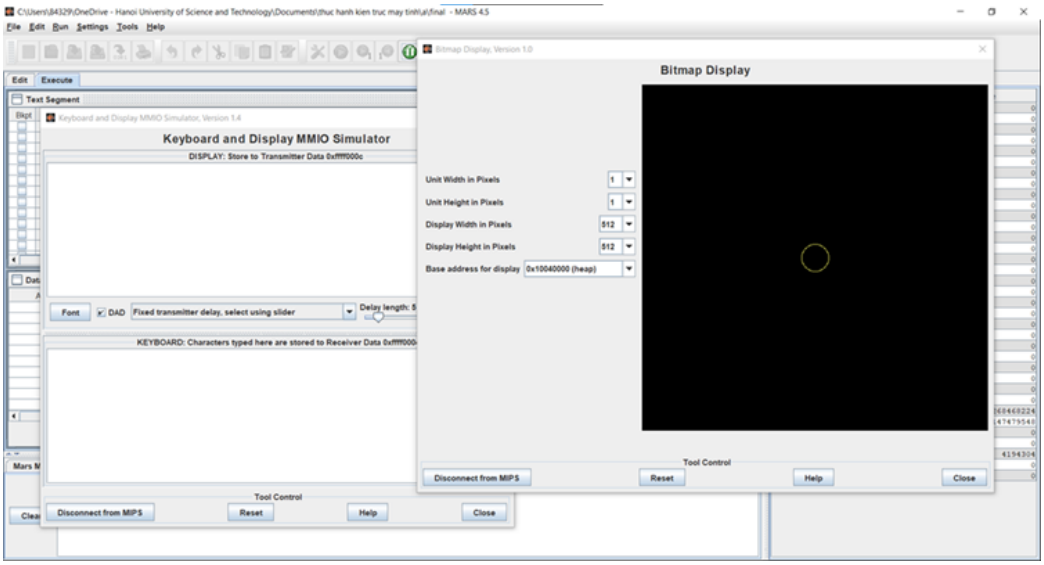
II.3 Ý nghĩa các thanh ghi được sử dụng

```
$s7: lưu chiều rộng của màn hình
$a0: tọa độ x0 của đường tròn
$a1: tọa độ y0 của đường tròn
$a2: r - bán kính của đường tròn
 $s0: màu của hình tròn
Trong @moving:
    -   $t0: giá trị mã ASCII của các phím nhập vào
    -   $t3: giá trị mã ASCII của các phím nhập vào
Trong @DrawCircle:
    -  $t0: lưu giá trị x0
    -  $t1: lưu giá trị y0
    -  $t2: lưu giá trị r
```

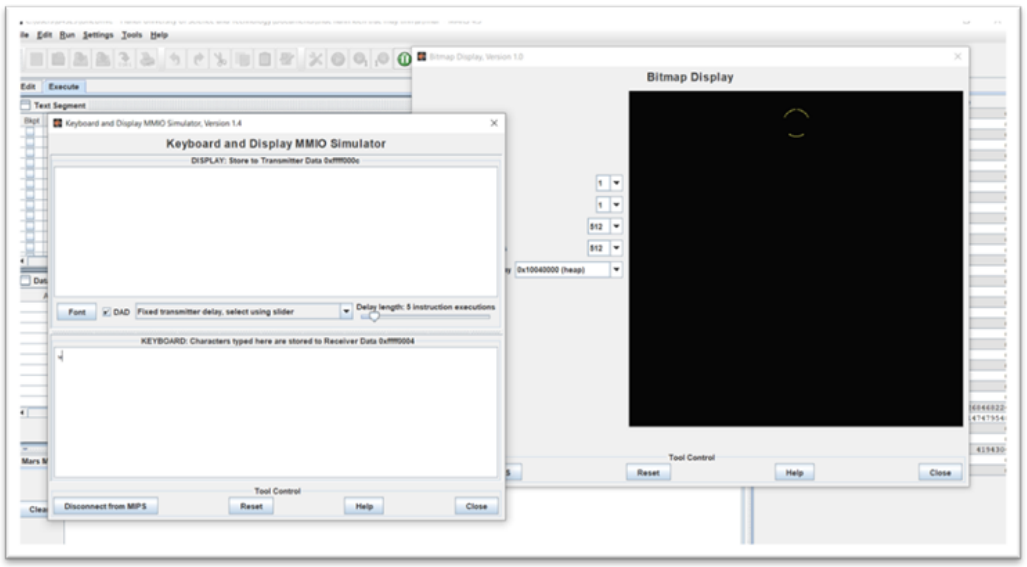
- `$t3: lưu giá trị x = 0`
- `$t4: lưu giá trị y = r`
- `$t5: = 1`
- `$t6: lưu giá trị P = 1 - r`
- `$t7: lưu giá trị 2*x`
- `$t8: lưu giá trị 2*y`
- `$t9: = 2*x - 2*y`

II.4 Kết quả chạy chương trình

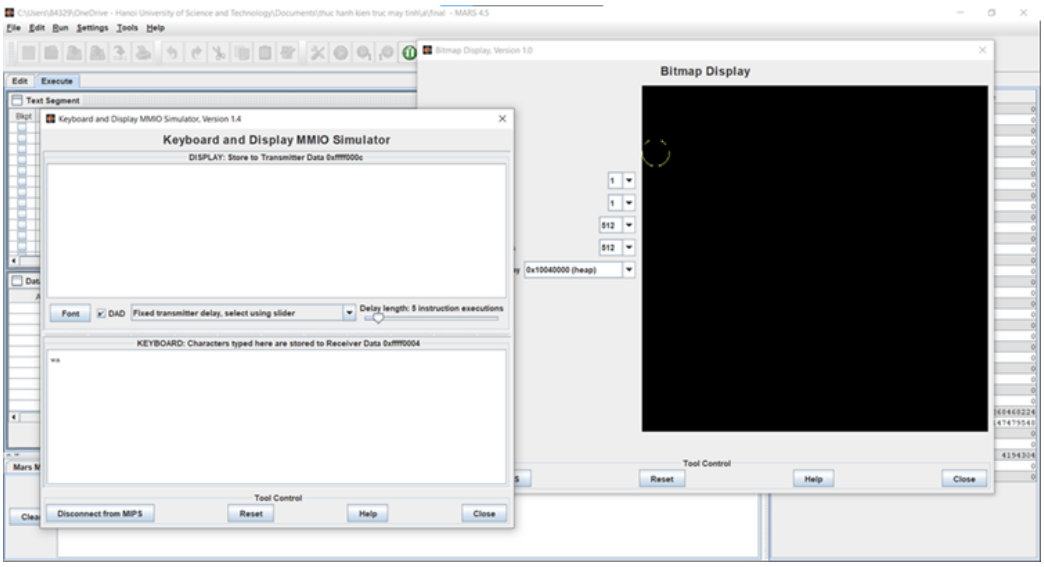
1. Khởi động chương trình



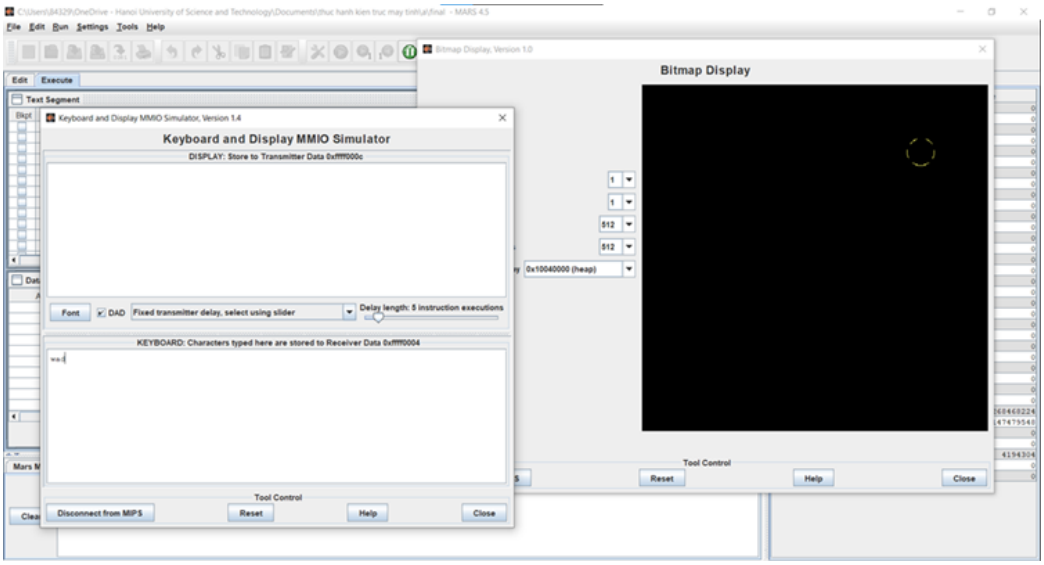
2. Nhấn phím W (Di Chuyển Lên)



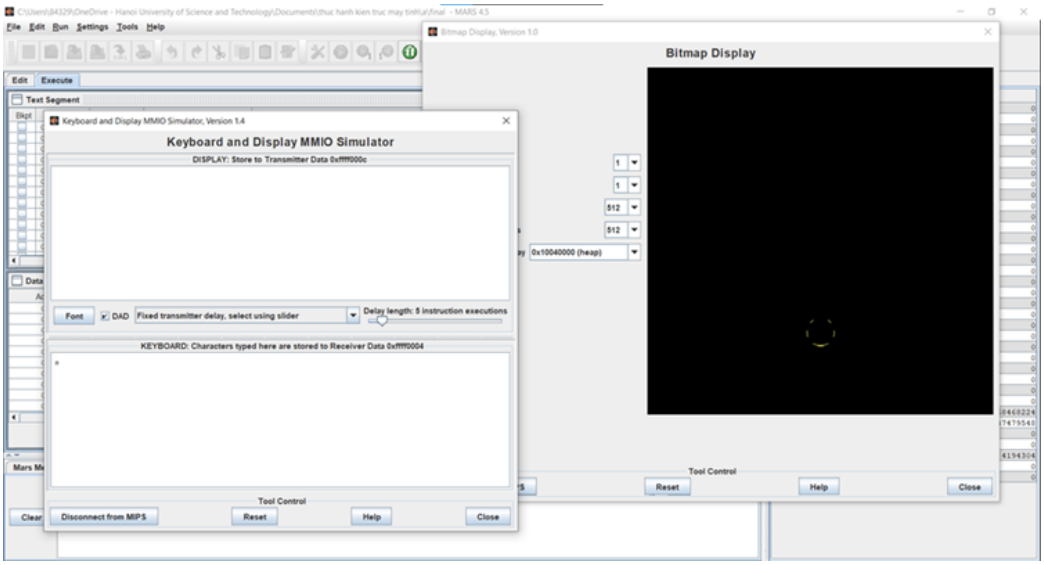
3. Nhấn phím A (Sang Trái)



4. Nhấn phím D (Sang Phải)



5. Nhấn phím S (Di Chuyển Xuống)



II.5 Code

```
.eqv KEY_CODE 0xFFFF0004
.eqv KEY_READY 0xFFFF0000
.eqv DISPLAY_CODE 0xFFFF000C
.eqv DISPLAY_READY 0xFFFF0008

.text
```

```
li $k0, KEY_CODE
li $k1, KEY_READY
li $s2, DISPLAY_CODE
li $s1, DISPLAY_READY

addi $s7, $0, 512
addi $a0, $0, 256
addi $a1, $0, 256
addi $a2, $0, 20
addi $s0, $0, 0x00FFFF66
jal DrawCircle
nop

moving:
beq $t0, 97, left
beq $t0, 100, right
beq $t0, 115, down
beq $t0, 119, up

j Input

left:
addi $s0, $0, 0x00000000
jal DrawCircle
addi $a0, $a0, -1
add $a1, $a1, $0
addi $s0, $0, 0x00FFFF66
jal DrawCircle
jal Pause
bltu $a0, 20, reboundRight
j Input

right:
addi $s0, $0, 0x00000000
jal DrawCircle
addi $a0, $a0, 1
add $a1, $a1, $0
addi $s0, $0, 0x00FFFF66
jal DrawCircle
jal Pause
bgtu $a0, 492, reboundLeft
```

```
j Input

up:
addi $s0, $0, 0x00000000
jal DrawCircle
addi $a1, $a1, -1
add $a0, $a0, $0
addi $s0, $0, 0x00FFFF66
jal DrawCircle
jal Pause
bltu $a1, 20, reboundDown
j Input

down:
addi $s0, $0, 0x00000000
jal DrawCircle
addi $a1, $a1, 1
add $a0, $a0, $0
addi $s0, $0, 0x00FFFF66
jal DrawCircle
jal Pause
bgtu $a1, 492, reboundUp
j Input

reboundLeft:
li $t3, 97
sw $t3, 0($k0)
j Input

reboundRight:
li $t3, 100
sw $t3, 0($k0)
j Input

reboundDown:
li $t3, 115
sw $t3, 0($k0)
j Input

reboundUp:
li $t3, 119
```

```
sw $t3, 0($k0)
j Input

endMoving:
...

Input:
ReadKey: lw $t0, 0($k0)
j moving

Pause:
addiu $sp, $sp, -4
sw $a0, ($sp)
li $a0, 0
li $v0, 32
syscall
nop
lw $a0, ($sp)
DrawCircle:

addiu $sp, $sp, -32
sw $ra, 28($sp)
sw $a0, 24($sp)
sw $a1, 20($sp)
sw $a2, 16($sp)
sw $s4, 12($sp)
sw $s3, 8($sp)
sw $s2, 4($sp)
sw $s0, ($sp)

sub $s2, $0, $a2
add $s3, $0, $a2
add $s4, $0, $0

DrawCircleLoop:
bgt $s4, $s3, exitDrawCircle
nop

jal plot8points
nop
```

```
add $s2, $s2, $s4
addi $s4, $s4, 1
add $s2, $s2, $s4

blt $s2, $0, DrawCircleLoop
nop

sub $s3, $s3, 1
sub $s2, $s2, $s3
sub $s2, $s2, $s3

j DrawCircleLoop
nop

exitDrawCircle:

lw $s0, ($sp)
lw $s2, 4($sp)
lw $s3, 8($sp)
lw $s4, 12($sp)
lw $a2, 16($sp)
lw $a1, 20($sp)
lw $a0, 24($sp)
lw $ra, 28($sp)

addiu $sp, $sp, 32

jr $ra
nop

plot8points:
addiu $sp, $sp, -4
sw $ra, ($sp)

jal plot4points
nop
beq $s4, $s3, skipSecondplot
nop

add $t2, $0, $s4
add $s4, $0, $s3
```

```
add $s3, $0, $t2

jal plot4points
nop

add $t2, $0, $s4
add $s4, $0, $s3
add $s3, $0, $t2

skipSecondplot:
lw $ra, ($sp)
addiu $sp, $sp, 4

jr $ra
nop

plot4points:
addiu $sp, $sp, -4
sw $ra, ($sp)

add $t0, $0, $a0
add $t1, $0, $a1

add $a0, $t0, $s3
add $a2, $t1, $s4

jal SetPixel
nop

sub $a0, $t0, $s3
beq $s3, $0, skipXnotequal0
nop

jal SetPixel
nop

skipXnotequal0:
sub $a2, $t1, $s4
jal SetPixel
nop
```

```
add $a0, $t0, $s3
beq $s4, $0, skipYnotequal0
nop

jal SetPixel
nop

skipYnotequal0:

    add $a0, $0, $t0
    add $a2, $0, $t1

    lw $ra, ($sp)
    addiu $sp, $sp, 4

    jr $ra
    nop

SetPixel:

    addiu $sp, $sp, -20
    sw $ra, 16($sp)
    sw $s1, 12($sp)
    sw $s0, 8($sp)
    sw $a0, 4($sp)
    sw $a2, ($sp)

    lui $s1, 0x1004
    sll $a0, $a0, 2
    add $s1, $s1, $a0
    mul $a2, $a2, $s7
    mul $a2, $a2, 4

    add $s1, $s1, $a2

    sw $s0, ($s1)

    lw $a2, ($sp)
    lw $a0, 4($sp)
    lw $s0, 8($sp)
    lw $s1, 12($sp)
```

```
lw $ra, 16($sp)
addiu $sp, $sp, 20

jr $ra
nop
```

III Bài tập 3

III.1 Đề bài

Chương trình sau sẽ đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn LED 7 đoạn. Nguyên tắc:

1. Cho một đoạn văn mẫu, cố định sẵn trong mã nguồn. Ví dụ "bo mon ky thuat may tinh"
2. Sử dụng bộ định thời TIMER (trong bộ giả lập Digi Lab Sim) để tạo ra khoảng thời gian để đo.

Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.

1. Người dùngnhập các kí tự từbàn phím. Ví dụ nhập "bo mOn ky 5huat may tinh".
2. Chương trình cần phải đếm sốkí tự đúng (trong ví dụtrên thì người dùng gõ sai chữ Ovà 5) mà người dùng đã gõ và hiển thị lên các đèn led. Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.

III.2 Thuật toán

Kiểm tra tốc độ gõ (số ký tự gõ được trong 2s)

```
Kiểm tra trạng thái của phím từ bàn phím (KEY_READY).
Nếu có phím được nhấn, tăng biến đếm số ký tự nhập được trong 2s.
Kiểm tra xem đã đủ 2s chưa.
Nếu đã đủ 2s, hiển thị số ký tự nhập được trên đèn LED 7 đoạn.
Tiếp tục vòng lặp.
```

Kiểm tra độ chính xác (so với chuỗi ban đầu là "xin chao cac ban")

```
 Kiểm tra từng ký tự một với string đã cho ban đầu
Nếu 2 ký tự giống nhau thì tăng biến đếm số ký tự chính xác lên 1
Khi gặp ký tự enter thì kết thúc và in số ký tự chính xác lên đèn
LED 7 đoạn
```

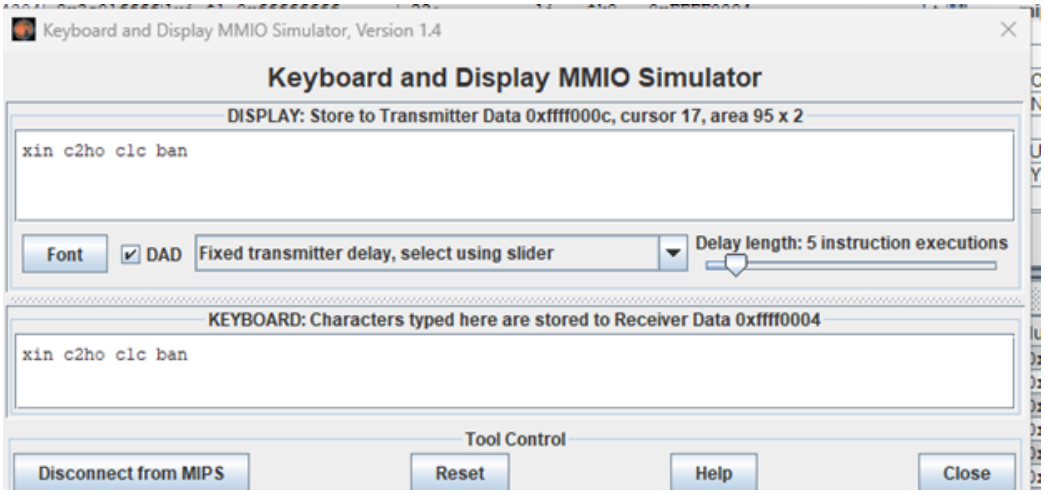
III.3 Trình bày

Thanh ghi $s4 để lưu tổng số kí tự nhập vào
Thanh ghi $s3 để lưu số vòng lặp
Thanh ghi $t4 có giá trị 10 để phục vụ cho việc chia thành hàng chục và hàng đơn vị để hiển thị ở đèn trái và đèn phải
Thanh ghi $t6 lưu số ký tự nhập vào trong 2 giây để có thể hiển thị ra màn hình tốc độ nhập chữ
Đọc và lưu kết quả nhập vào thanh ghi $t1, nếu không có dữ liệu được nhập và thì tiếp tục tăng giá trị thanh ghi $t3 lên 1 .
Khi giá trị $t3 =400 khì kết thúc lặp và chuyển sang SETCOUNT (gán giá trị $t3=0 và in giá trị $t3 ra màn hình) . Nếu chưa đủ 400 thì nhảy sang nhãn sleep.
Nếu $t1 có chứa kí tự thì sử lý ngắt , đọc kí tự vào thanh ghi $t0, chờ qua 2 bước WAIT_FOR_DIS và SHOW_KEY và strore ký tự . Nếu giá trị ký tự ($t0) là 10, tức là ký tự xuống dòng ('\n'), chương trình sẽ nhảy đến nhãn "END" để kết thúc quá trình nhập và xử lý.
Phần kiểm tra số ký tự đúng thì như trên thuật toán em đã trình bày.
Bước cuối để alert có return lại chương trình hay không nếu chọn yes sẽ tiếp tục quá trình nhập, nếu no thì j exit .

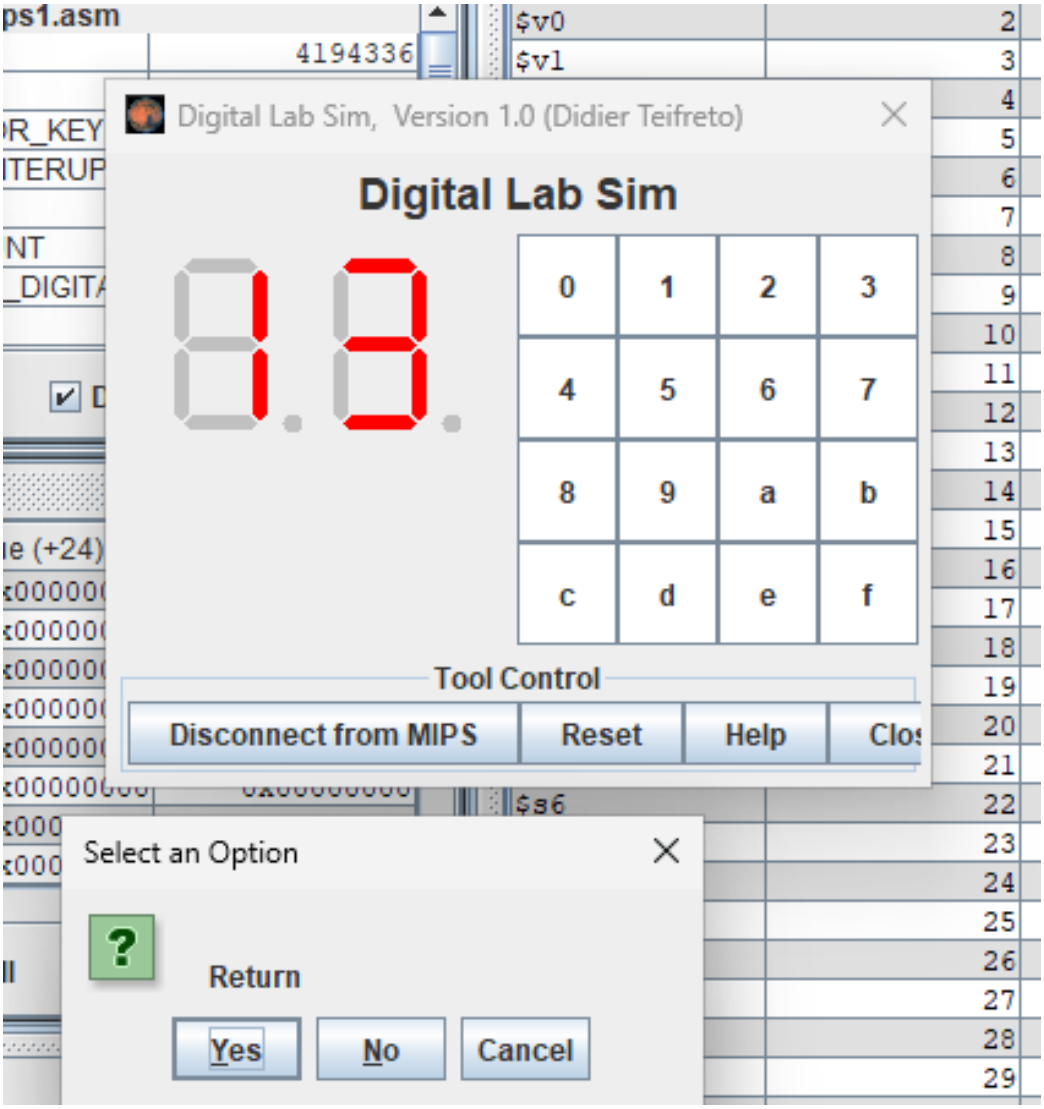
III.4 Kết quả

Chương trình chạy đúng và dưới đây là ảnh kết quả

1. Chuỗi nhập vào từ bàn phím



2. Kết quả hiển thị trên đèn led 7 thanh




```
Number of characters entered in 2s:  5
Number of characters entered in 2s:  2
Number of characters entered in 2s:  4
Number of characters entered in 2s:  5
xin c2co c1c ban
The correct number of characters is: 13
-- program is finished running (dropped off bottom) --
```

III.5 Code

```
.eqv SEVENSEG_LEFT 0xFFFF0011
```

```
.eqv SEVENSEG_RIGHT 0xFFFF0010
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv KEY_CODE 0xFFFF0004
.eqv KEY_READY 0xFFFF0000
.eqv DISPLAY_CODE 0xFFFF000C
.eqv DISPLAY_READY 0xFFFF0008
.eqv MASK_CAUSE_KEYBOARD 0x0000034

.data
bytehex: .byte 63,6,91,79,102,109,125,7,127,111
storestring: .space 1000
stringsource: .asciiz "xin chao cac ban"
Message: .asciiz "\n Number of characters entered in 2s: "
numkeyright: .asciiz "\n The correct number of characters is: "
notification: .asciiz "\n Return "

.text
li $k0, KEY_CODE
li $k1, KEY_READY
li $s0, DISPLAY_CODE
li $s1, DISPLAY_READY

MAIN:
li $s4, 0
li $s3, 0
li $t4, 10
li $t5, 400
li $t6, 0
li $t9, 0

LOOP:
WAIT_FOR_KEY:
lw $t1, 0($k1)
beq $t1, $zero, Polling

MAKE_INTERUPT:
addi $t6, $t6, 1
teqi $t1, 1

Polling:
```

```
addi $s3, $s3, 1
div $s3, $t5
mfhi $t7
bne $t7, 0, SLEEP

SETCOUNT:
li $s3, 0
li $v0, 4
la $a0, Message
syscall

li $v0, 1
add $a0, $t6, $zero
syscall

DISPLAY_DIGITAL:
div $t6, $t4
mflo $t7
la $s2, bytehex
add $s2, $s2, $t7
lb $a0, 0($s2)
jal SHOW_7SEG_LEFT

mfhi $t7
la $s2, bytehex
add $s2, $s2, $t7
lb $a0, 0($s2)
jal SHOW_7SEG_RIGHT

li $t6, 0
beq $t9, 1, ASK_LOOP

SLEEP:
addi $v0, $zero, 32
li $a0, 5
syscall
nop
b LOOP

END_MAIN:
li $v0, 10
```

```
syscall

SHOW_7SEG_LEFT:
li $t0, SEVENSEG_LEFT
sb $a0, 0($t0)
jr $ra

SHOW_7SEG_RIGHT:
li $t0, SEVENSEG_RIGHT
sb $a0, 0($t0)
jr $ra

.ktext 0x80000180
mfc0 $t1, $13
li $t2, MASK_CAUSE_KEYBOARD
and $at, $t1, $t2
beq $at, $t2, COUNTER_KETYBOARD
j END_PROCESS

COUNTER_KETYBOARD:
READ_KEY:
lb $t0, 0($k0)
WAIT_FOR_DIS:
lw $t2, 0($s1)
beq $t2, $zero, WAIT_FOR_DIS
SHOW_KEY:
sb $t0, 0($s0)
la $t7, storestring
add $t7, $t7, $s4
sb $t0, 0($t7)
addi $s4, $s4, 1
beq $t0, 10, END
END_PROCESS:
NEXT_PC:
mfc0 $at, $14
addi $at, $at, 4
mtc0 $at, $14
RETURN:
eret
END:
li $v0, 11
```

```
li $a0, '\n'
syscall
li $t1, 0
li $t3, 0
li $t8, 24
slt $t7, $s4, $t8
bne $t7, 1, CHECK_STRING
add $t8, $0, $s4
addi $t8, $t8, -1
CHECK_STRING:
la $t2, storestring
add $t2, $t2, $t1
li $v0, 11
lb $t5, 0($t2)
move $a0, $t5
syscall
la $t4, stringsource
add $t4, $t4, $t1
lb $t6, 0($t4)
bne $t6, $t5, CONTINUE
addi $t3, $t3, 1
CONTINUE:
addi $t1, $t1, 1
beq $t1, $t8, PRINT
j CHECK_STRING
PRINT:
li $v0, 4
la $a0, numkeyright
syscall
li $v0, 1
add $a0, $0, $t3
syscall
li $t9, 1
li $t6, 0
li $t4, 10
add $t6, $0, $t3
b DISPLAY_DIGITAL
ASK_LOOP:
li $v0, 50
la $a0, notification
syscall
```

```
beq $a0, 0, MAIN
b EXIT
EXIT:
```