BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 10

Họ và tên: Nguyễn Quý Đức MSSV: 20235682

Mã HP: IT3280 Mã lớp: 156788

1. Assignment 1

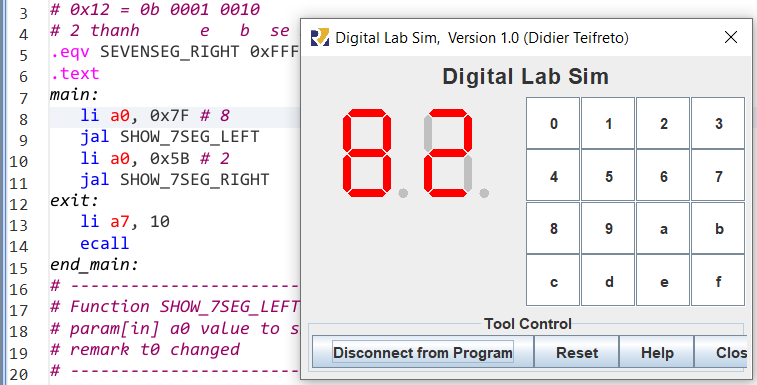
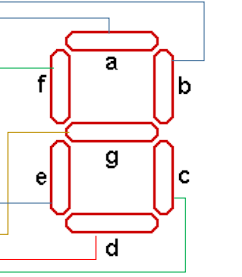
Khi ta store byte vào địa chỉ của LED 7seg-trái/phải thì đèn sẽ sáng theo quy luật:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| nth bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| segment | . | g | f | e | d | c | b | a |
| 1 | bật | | | | | | | |
| 0 | tắt | | | | | | | |

Ta cần đèn phát ra số 82 (trong MSSV 20235682):

- Hiển thị số 2 🡺 abged sáng 🡺 giá trị truyền vào = 0101 1011 = 0x5B

- Hiển thị số 8 🡺 mọi thanh sáng (trừ dấu ‘.’) 🡺 giá trị truyền vào = 0111 1111 = 0x7F



.eqv SEVENSEG\_LEFT 0xFFFF0011 # Dia chi cua den led 7 doan trai

# vd gia tri cua byte tai 0xFFFF0011 la 0x12

# 0x12 = 0b 0001 0010

# 2 thanh e b se nhan tin hieu

.eqv SEVENSEG\_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai

.text

main:

li a0, 0x7F # 8

jal SHOW\_7SEG\_LEFT

li a0, 0x5B # 2

jal SHOW\_7SEG\_RIGHT

exit:

li a7, 10

ecall

end\_main:

# ---------------------------------------------------------------

# Function SHOW\_7SEG\_LEFT : turn on/off the 7seg

# param[in] a0 value to shown

# remark t0 changed

# ---------------------------------------------------------------

SHOW\_7SEG\_LEFT:

li t0, SEVENSEG\_LEFT # assign port's address

sb a0, 0(t0) # assign new value

jr ra

# ---------------------------------------------------------------

# Function SHOW\_7SEG\_RIGHT : turn on/off the 7seg

# param[in] a0 value to shown

# remark t0 changed

# ---------------------------------------------------------------

SHOW\_7SEG\_RIGHT:

li t0, SEVENSEG\_RIGHT # assign port's address

sb a0, 0(t0) # assign new value

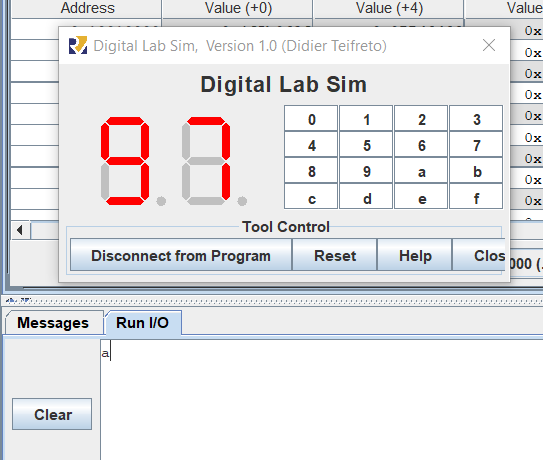
jr ra

1. Assignment 2

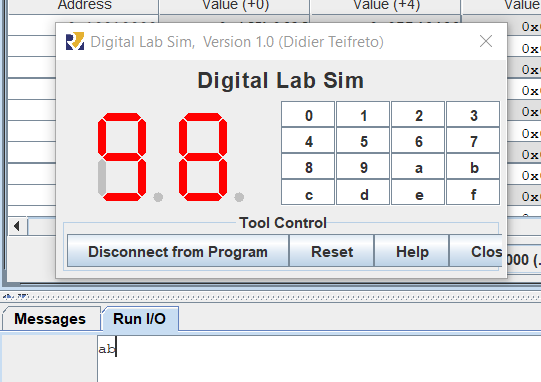
Nội dung chính chương trình:

* Đọc 1 ký tự từ bàn phím (ecall)
* Lấy 2 chữ số cuối của mã ASCII (mod 100)
* Tách thành hàng chục và hàng đơn vị ( lấy mod 10, chia 10, lấy mod 10 tiếp)
* Tra bảng mã 7 đoạn (SEG\_TABLE)
* Hiển thị lên LED trái (hàng chục) và LED phải (hàng đơn vị)
* Lặp lại mãi (j main)

Ví dụ người dùng nhập ‘a’ (= 97)



Sau đó người dùng nhập ‘b’ (= 98):



.eqv SEVENSEG\_LEFT 0xFFFF0011

.eqv SEVENSEG\_RIGHT 0xFFFF0010

.data

SEG\_TABLE:

.byte 0x3F # 0

.byte 0x06 # 1

.byte 0x5B # 2

.byte 0x4F # 3

.byte 0x66 # 4

.byte 0x6D # 5

.byte 0x7D # 6

.byte 0x07 # 7

.byte 0x7F # 8

.byte 0x6F # 9

.text

main:

li a7, 12 # read char

ecall

li t3, 100

remu t0, a0, t3 # get 2 last digits

li t4, 10

divu t1, t0, t4 # tens

remu t2, t0, t4 # ones

la t5, SEG\_TABLE

add t6, t5, t1

lbu a1, 0(t6)

add t6, t5, t2

lbu a2, 0(t6)

li t0, SEVENSEG\_LEFT

sb a1, 0(t0)

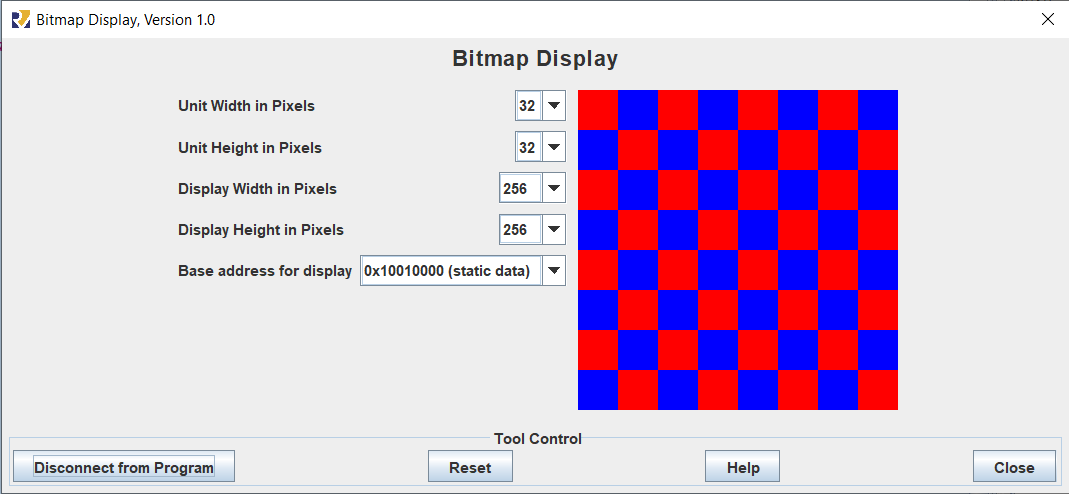
li t0, SEVENSEG\_RIGHT

sb a2, 0(t0)

j main

1. Assignment 3

* Vòng lặp lồng i (hàng) và j (cột) từ 0 đến 7
* Tính địa chỉ ô: addr = MONITOR\_SCREEN + ((i \* 8 + j) \* 4)
* Tính (i + j) % 2 để xác định màu xen kẽ
* Nếu (i + j) chẵn → RED, lẻ → BLUE
* Ghi màu vào địa chỉ tương ứng trong bộ nhớ màn hình
* Vẽ đủ 64 ô rồi thoát bằng syscall ecall
* Mỗi ô là 1 word (4 byte) đại diện cho 1 pixel



.eqv MONITOR\_SCREEN 0x10010000

.eqv RED 0x00FF0000

.eqv BLUE 0x000000FF

.text

main:

li t0, 0 # i = 0 (row)

row\_loop:

li t1, 0 # j = 0 (col)

col\_loop:

# addr = base + (i \* 8 + j) \* 4

slli t3, t0, 3 # t3 = 8\*i

add t3, t3, t1 # t3 = i\*8 + j

slli t3, t3, 2 # t3 \*= 4

li t4, MONITOR\_SCREEN

add t3, t3, t4 # t3 = final addr

add t5, t0, t1 # t5 = i + j

andi t5, t5, 1 # (i+j)%2

beqz t5, use\_red

li t6, BLUE

j store

use\_red:

li t6, RED

store:

sw t6, 0(t3) # store color

addi t1, t1, 1 # j++

li t6, 8

blt t1, t6, col\_loop

addi t0, t0, 1 # i++

blt t0, t6, row\_loop

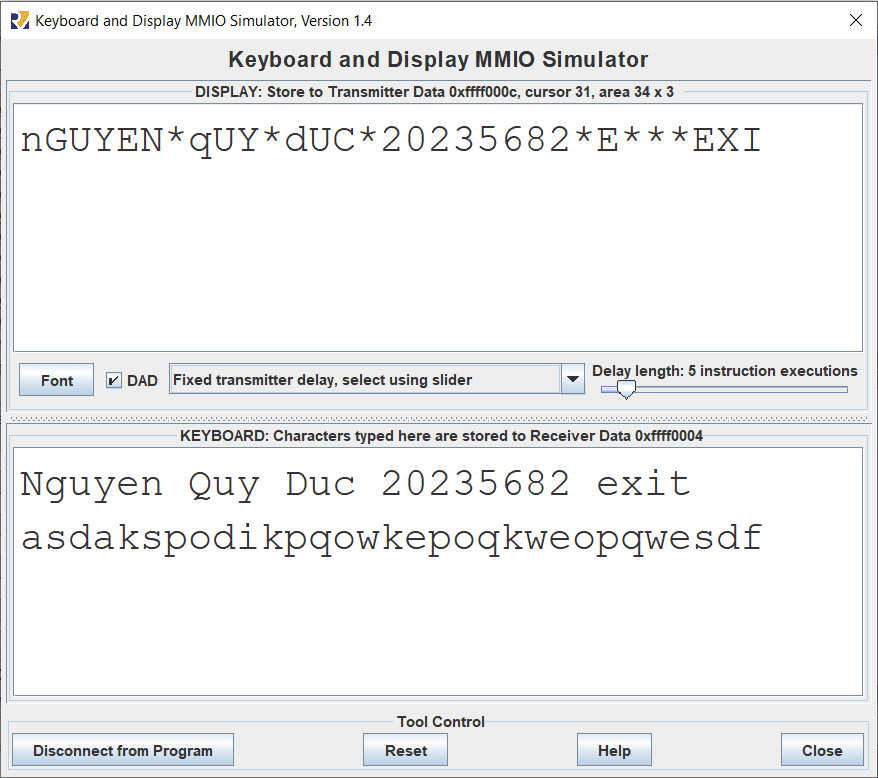
exit:

li a7, 10

ecall

1. Assignment 4

* Đọc phím từ bàn phím khi KEY\_READY == 1
* Cập nhật buffer 4 ký tự gần nhất (s2–s5)
* Nếu buffer là "exit" 🡺 thoát chương trình
* Nếu a ≤ char ≤ z 🡺 char -= 32 (chuyển thành chữ hoa)
* Nếu A ≤ char ≤ Z 🡺 char += 32 (chuyển thành chữ thường)
* Nếu 0 ≤ char ≤ 9 🡺 giữ nguyên
* Ký tự khác 🡺 thay bằng '\*'
* Hiển thị ký tự ra màn hình khi DISPLAY\_READY == 1



.eqv KEY\_CODE 0xFFFF0004

.eqv KEY\_READY 0xFFFF0000

.eqv DISPLAY\_CODE 0xFFFF000C

.eqv DISPLAY\_READY 0xFFFF0008

.text

main:

li a0, KEY\_CODE

li a1, KEY\_READY

li s0, DISPLAY\_CODE

li s1, DISPLAY\_READY

li s6, 'a'

li s7, 'z'

li s8, 'A'

li s9, 'Z'

li s10, '0'

li s11, '9'

li t5, '\*'

li s2, 0

li s3, 0

li s4, 0

li s5, 0

loop:

waitforkey:

lw t1, 0(a1)

beq t1, zero, waitforkey

readkey:

lw t0, 0(a0)

mv s5, s4

mv s4, s3

mv s3, s2

mv s2, t0

li t1, 'e'

bne s5, t1, processchar

li t1, 'x'

bne s4, t1, processchar

li t1, 'i'

bne s3, t1, processchar

li t1, 't'

bne s2, t1, processchar

j exitprogram

processchar:

blt t0, s6, checkupper

blt s7, t0, checkupper

addi t0, t0, -32

j displaychar

checkupper:

blt t0, s8, checkdigit

blt s9, t0, checkdigit

addi t0, t0, 32

j displaychar

checkdigit:

blt t0, s10, setstar

blt s11, t0, setstar

j displaychar

setstar:

mv t0, t5

displaychar:

waitfordis:

lw t2, 0(s1)

beq t2, zero, waitfordis

showkey:

sw t0, 0(s0)

j loop

exitprogram:

li a7, 10

ecall