TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện công nghệ Thông tin & Truyền thông

IT3280

Thực hành Kiến trúc máy tính

BÁO CÁO FINAL - PROJECT

Giảng viên hướng dẫn : ThS. Lê Bá Vui

Nhóm sinh viên : Nhóm 16

1. Hoàng Anh Tuấn - 20194705

2. Tạ Quang Linh - 20194605

Mục lục

1. Curiosity Marsbot

1. 1. Đề bài

1. Curiosity Marsbot

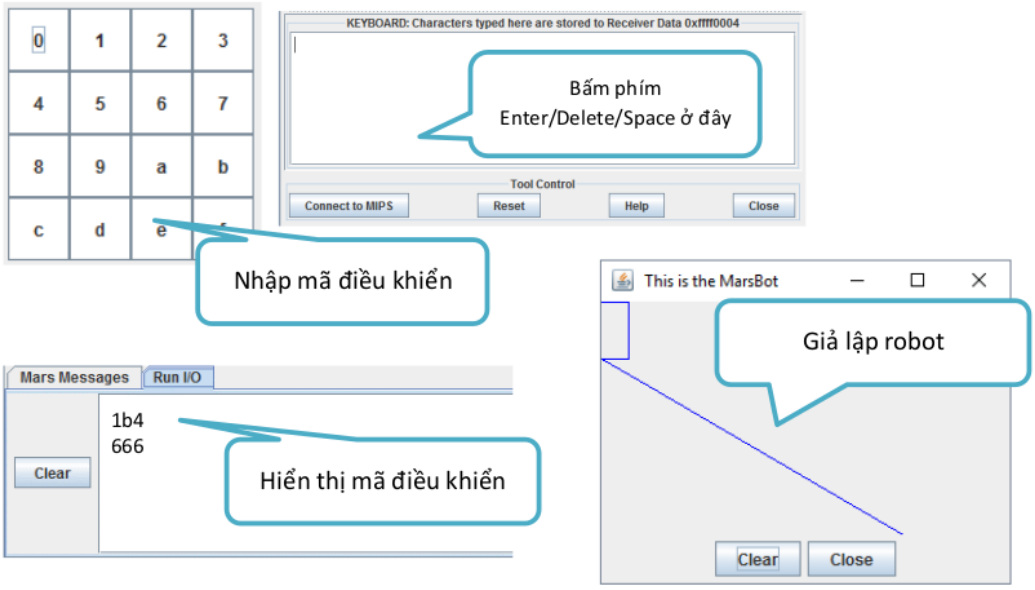
Xe tự hành Curiosity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất. Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot như sau:

Mã điều khiển	Ý nghĩa
1b4	Marsbot bắt đầu chuyển động
c68	Marsbot đứng im
444	Rẽ trái 90 độ so với phương chuyển động gần nhất
666	Rẽ phải 90 độ so vơi phương chuyển động gần nhất
dad	Bắt đầu để lại vết trên đường
cbc	Chấm dứt để lại vết trên đường
999	Tự động đi theo lộ trình ngược lại. Không vẽ vết, không nhận mã khác cho tới khi kết thúc lộ trình ngược. Mô tả: Marsbot được lập trình để nhớ lại toàn bộ lịch sử các mã điều khiển và khoảng thời gian giữa các lần đổi mã. Vì vậy, nó có thể đảo ngược lại lộ trình để quay về điểm xuất phát.

Sau khi nhận mã điều khiển, Curiosity Marsbot sẽ không xử lý ngay, mà phải đợi lệnh kích hoạt mã từ bàn phím Keyboard & Display MMIO Simulator. Có 3 lệnh như vậy:

Kích hoạt mã	Ý nghĩa
Phím Enter	Kết thúc nhập mã và yêu cầu Marsbot thực thi.
Phím Delete	Xóa toàn bộ mã điều khiển đang nhập.
Phím Space	Lặp lại lệnh đã thực hiện trước đó.

Hãy lập trình để Marsbot có thể hoạt động như đã mô tả.
Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe.



1. **2. Thuật toán:**

- **Bước 1**: Mỗi khi người dùng nhập 1 ký tự từ Digital Lab Sim sẽ tạo ra interrupt để lưu kí tự được nhập vào bộ nhớ, tạo nên đoạn code điều khiển.
  Ký tự được nhập vào sẽ được lưu trong chuỗi inputControlCode, tăng độ dài của chuỗi lên 1 và thêm ký tự kết thúc '\0' và cuối chuỗi.

- **Bước 2**: Kiểm tra liên tục xem các lệnh kích hoạt mã có được nhập ở Keyboard & Display MMIO Simulator hay không.
  - Khi ký tự Enter được nhập, sẽ kiểm tra xem đoạn code điều khiển có hợp lệ không (gồm 3 ký tự), nếu không sẽ thông báo code lỗi và sang bước 4. Nếu có thì chuyển sang bước 3.
  - Nếu ký tự Delete được nhập, tiến hành xóa toàn bộ mã điều khiển đang nhập
  - Nếu ký tự Space được nhập, lặp lại lệnh đã thực hiện trước đó.

- **Bước 3**: Lần lượt kiểm tra xem code điều khiển được nhập vào có trùng với các đoạn code điều khiển đã quy định sẵn. Nếu không thì thông báo đoạn code bị lỗi. Ngược lại thực hiện thao tác theo quy định sẵn:
  - Để marsbot có thể nhớ được được toàn bộ mã điều khiển. Chúng ta sử dụng mảng path để lưu trữ 1 structure bao gồm tọa độ của điểm (x, y) và hướng của marsbot tại thời điểm hành động thay đổi.
  - Khi marsbot đi theo lộ trình ngược lại chỉ cần duyệt mảng path từ phía cuối và đảo ngược hướng được lưu tại mỗi điểm.
  - Khi cho marsbot lặp lại lệnh thực hiện trước đó thì cũng cần phải lưu vào mảng path. Để có thể thực hiện lại lệnh trước đó chỉ cần xét hướng trong structure cuối cùng trong mảng path đồng thời sử dụng biến isRight để kiểm tra xem trước đó marsbot đang đi thẳng (2), rẽ phải (1) hay rẽ trái (0). Nếu rẽ phải thì cộng thêm 90 vào hướng được lấy ra, rẽ trái trừ 90 vào hướng đó còn đi thẳng thì không thay đổi hướng.

- **Bước 4**: In ra trên màn hình console code điều khiển đã nhập và xóa lưu trữ trong bộ nhớ.

1. **3. Giải thích mã nguồn**

- Khởi tạo các mã code và gán vào các nhãn.

- **inputControlCode**: Chuỗi lưu trữ mã điều khiển.
  lengthControlCode: Độ dài mảng chuỗi inputControlCode
  nowHeading: Hướng (góc) hiện tại.
  path: mảng lưu trữ vị trí mà marsbot thay đổi hành động. Vị trí là 1 structure (12 bytes) bao gồm tọa độ của marsbot (x, y) và hướng tại điểm hành động thay đổi (z).
  lengthPath: Độ dài mảng path
  isGoing(boolean): Kiểm tra marsbot có đang di chuyển hay không.
  isTracking(boolean): Kiểm tra marsbot có đang để lại vết hay không.
  isRight(int): Kiểm tra marsbot đang đi thẳng, rẽ trái hay rẽ phải..
- **Phần init**(khởi tạo): Khởi tạo gốc ban đầu của marsbot và quay marsbot sang phải để nó không di chuyển ra khỏi hộp dialog, đồng thời lưu trữ gốc đó vào mảng path và tăng chiều dài cuả mảng path thêm 12 để dùng cho lần lưu trữ tiếp theo.
- **check**: Kiểm tra mã điều khiển nhập vào có hợp lệ hay không. Hợp lệ ở đây là chiều dài đã bằng 3 hay chưa hoặc có nằm trong các mã điều khiển đã được quy định hay không. Nếu vi phạm 1 tong 2 điều kiện trên thì thông báo lỗi.
- **repeat**: Thực hiện lệnh trước đó của marsbot. Nếu marsbot đang đứng yên thì vẫn tiếp tục đứng yên, nếu đang di chuyển thẳng trước đó thì vẫn tiếp tục di chuyển thẳng. Lấy structure cuối của mảng path, sau đó trích xuất góc trong structure, kiểm tra xem trước đó marsbot có rẽ trái hay phải không bằng biến isRight. Nếu rẽ phải thì thêm 90 vào góc được lấy ra đó ngược lại thì giảm 90 nếu trước đó rẽ trái.
- **printControlCode**: In mã điều khiển ra console.
- **storePath**: Lưu trữ tọa độ hiện tại thông qua WHEREX, WHEREY và nowHeading vào mảng path. Tăng độ dài mảng path thêm 12 để sử dụng cho lần lưu trữ sau.
- **goBack**: Đi theo lộ trình ngược lại. Đầu tiên là tắt keypad ở DIgitalLabSim và tắt track. Đọc mảng path từ cuối lên, lấy góc trong từng structure sau đó đảo ngược để có thể đi đúng hướng ngược lại. Cho marsbot di chuyển đồng thời kiểm tra tọa độ hiện tại với tọa độ được lưu trữ trong structure. Nếu trùng cả x và y thì quay lại begin để đọc tiếp structure tiếp theo.
- **finish**: Reset lại tọa độ ban đầu, bật interrupt của DigitalLabSim.
- **track**: Nhảy đến nhãn TRACK và in mã lệnh điều khiển.
- **untrack**: Nhảy đến nhãn UNTRACK và in mã lệnh điều khiển.

- **go**: Khi nhập mã điều khiển “1b4” thì cập nhập giá trị của isRight = 2 tức là đang đi thẳng, để dùng cho hàm repeat. Sau đó nhảy đến nhãn GO và in mã lệnh điều khiển.
- **stop**: Nhảy đến nhãn STOP và in mã lệnh điều khiển.
- **goRight**:Cho marsbot dừng, untrack. Cộng hướng hiện tại thêm 90 để marsbot rẽ phải và isRight = 1(repeat). Nếu đang track trước đó thì tiếp tục track.
- **goLeft**: Trừ hướng hiện tại đi 90 để marsbot rẽ trái và isRight = 0(repeat)
- **remove**: Xóa mã lệnh điều khiển trong chuỗi inputControlCode
- **isEqual**: Kiểm tra xem inputControlCode có trùng với các code điều khiển hay không, nếu trùng thì trả về 1 không thì trả về 0 (sử dụng cho hàm check)
- **error**: In mã lỗi và thông báo lỗi sau đó xóa mã lỗi trong inputControlCode để chuẩn bị cho lần nhập tiếp theo.
- **GO**, **STOP**, **TRACK**, **UNTRACK**, **ROTATE** -> thực hiện các hoạt động của marsbot.
- **get_code**, **get_code_in_char**: đọc keypad được nhập từ DigitalLabSIm
- **store_code**: Lưu kí tự vừa được nhập vào chuỗi inputControlCode và tăng chiều dài lengthControlCode lên 1, thêm kí tự kết thúc chuỗi ‘\0’ vào cuối chuỗi.

1. **4. Code**

```
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

.eqv KEY_CODE 0xFFFF0004     # ASCII code from keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000     # =1 if has a new keycode ?
                                # Auto clear after lw
#------------------------------------------------------------------------------
# Key value
    .eqv KEY_0 0x11
    .eqv KEY_1 0x21
    .eqv KEY_2 0x41
    .eqv KEY_3 0x81
    .eqv KEY_4 0x12
    .eqv KEY_5 0x22
    .eqv KEY_6 0x42
    .eqv KEY_7 0x82
```

```
    .eqv KEY_8 0x14
    .eqv KEY_9 0x24
    .eqv KEY_a 0x44
    .eqv KEY_b 0x84
    .eqv KEY_c 0x18
    .eqv KEY_d 0x28
    .eqv KEY_e 0x48
    .eqv KEY_f 0x88
#------------------------------------------------------------------------
# Marsbot
.eqv HEADING 0xffff8010     # Integer: An angle between 0 and 359
                                # 0 : North (up)
                                # 90: East (right)
                            # 180: South (down)
                            # 270: West (left)
.eqv MOVING 0xffff8050          # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020     # Boolean (0 or non-0):
                                # whether or not to leave a track
.eqv WHEREX 0xffff8030          # Integer: Current x-location of MarsBot
.eqv WHEREY 0xffff8040          # Integer: Current y-location of MarsBot

#=============================================================
====================
#=============================================================
====================
.data

#Control code
    MOVE_CODE:     .asciiz "1b4"
    STOP_CODE:     .asciiz "c68"
    GO_LEFT_CODE:     .asciiz "444"
    GO_RIGHT_CODE:     .asciiz "666"
    TRACK_CODE:     .asciiz "dad"
    UNTRACK_CODE:     .asciiz "cbc"
    GO_BACK_CODE:     .asciiz "999"
```

```
   WRONG_CODE:     .asciiz "Wrong code!"
#-------------------------------------------------------------------------
   inputControlCode:     .space 50
   lengthControlCode:     .word 0
   nowHeading:       .word 0
#---------------------------------------------------------
# duong di cua marsbot duoc luu tru vao mang path
# moi 1 diem duoc luu tru duoi dang 1 structure
# 1 structure co dang {x, y, z}
# trong do:     x, y la toa do diem
#         z la huong cua canh do
# mac dinh:    structure dau tien se la {0,0,90}
#---------------------------------------------------------
   path:         .space 600
   lengthPath:     .word 12          # bytes

   isGoing:    .word 0
   isTracking:    .word 0
   isRight:     .word 0

#=========================================================================
======================
#=========================================================================
======================
.text
main:
   li $k0, KEY_CODE
         li $k1, KEY_READY
#---------------------------------------------------------
# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
#---------------------------------------------------------
   li $t1, IN_ADDRESS_HEXA_KEYBOARD
   li $t3, 0x80 # bit 7 = 1 to enable
   sb $t3, 0($t1)
#---------------------------------------------------------
```

```
# Init
#-----------------------------------------------------------
   sw $zero, isGoing
   sw $zero, isTracking

   # Luu toa do goc xuat phat vao mang path
   la $s2, lengthPath
   lw $s3, 0($s2)     # $s3 = lengthPath (dv: byte)

   la $s4, path
   add $s4, $s4, $s3    # vi tri bat dau luu

   sw $zero, 0($s4)    # luu x
   sw $zero, 4($s4)    # luu y

   li $s1, 90
   sw $s1, 8($s4)     # luu huong ban dau heading = 90
   sw $s1, nowHeading

   jal ROTATE
   nop
   addi $s3, $s3, 12    # lengthPath = lengthPath + 12
                        # 12 = 3 (word) x 4 (bytes)
   sw $s3, 0($s2)

# XU LY LENH KICH HOAT TU KEYBOARD & DISPLAY MMIO
SIMULATOR
loop:           nop
WaitForKey:    lw $t5, 0($k1)              # $t5 = [$k1] = KEY_READY
         beq $t5, $zero, WaitForKey    #neu $t5 == 0 thi lap lai
         nop
         beq $t5, $zero, WaitForKey
ReadKey:    lw $t6, 0($k0)                 # $t6 = [$k0] = KEY_CODE
         # DELETE
         beq $t6, 127 , continue     # neu $t6 == delete key thi xoa input
```

```
                                # 127 la delete key trong ma ascii
        nop
        beq $t6, 32, repeat     # neu $t6 == space key thi lap lai
                                # ma dieu khien truoc do
        nop
        # Neu lenh nhap vao khong nam trong 3 lenh enter, delete va space
        # thi se khong xu ly va tiep tuc cho user nhap lenh dung
        bne $t6, '\n' , loop    # neu $t6 != '\n' thi quay lai loop
        nop
        bne $t6, '\n' , loop

# neu $t6 == Enter key thi xu ly ma dieu khien duoc nhap vao

check:
        # Kiem tra do dai ma dieu khien
        la $s2, lengthControlCode
        lw $s2, 0($s2)
        #----------------
        bne $s2, 3, error

        # So sanh ma dieu khien nhap vao voi ma tieu chuan (1b4, c68,...)
        # neu ma dieu khien hop le thi cho marsbot thuc hien cac hang dong
        # tuong ung
        la $s3, MOVE_CODE
        jal isEqual
        beq $t0, 1, go

        la $s3, STOP_CODE
        jal isEqual
        beq $t0, 1, stop

        la $s3, GO_LEFT_CODE
        jal isEqual
        beq $t0, 1, goLeft
```

```
        la $s3, GO_RIGHT_CODE
        jal isEqual
        beq $t0, 1, goRight

        la $s3, TRACK_CODE
        jal isEqual
        beq $t0, 1, track

        la $s3, UNTRACK_CODE
        jal isEqual
        beq $t0, 1, untrack

        la $s3, GO_BACK_CODE
        jal isEqual
        beq $t0, 1, goBack

        # Neu khong khop voi ma dieu khien nao -> hien thi loi
        beq $t0, 0, error
        nop
repeat:
    # backup
    addi $sp,$sp,4
    sw $s5, 0($sp)
    addi $sp,$sp,4
    sw $s6, 0($sp)
    addi $sp,$sp,4
    sw $s7, 0($sp)
    addi $sp,$sp,4
    sw $t6, 0($sp)
    addi $sp,$sp,4
    sw $t7, 0($sp)
    addi $sp,$sp,4
    sw $t8, 0($sp)
    addi $sp,$sp,4
    sw $t9, 0($sp)
```

```
    addi $sp,$sp,4
    sw $s0, 0($sp)

    # processsing
    lw $t9, isGoing
    beqz    $t9, noGo
    nop
    lw $s0, isTracking
    lw $t6, isRight
    beq $t6, 2, repeat_end
    nop
    jal STOP
    nop
    jal UNTRACK
    nop

    la $s7, path
    la $s5, lengthPath
    lw $s5, 0($s5)     # $s5 = lengthPath
    add $s7, $s7, $s5    # $s7 = &path[lengthPath]

    addi $s5, $s5, -12     # lui lai 1 structure

    addi $s7, $s7, -12    # vi tri luu tru thong tin ve canh cuoi cung
    lw $s6, 8($s7)     # huong cua canh cuoi cung

    la $t8, nowHeading    # gan huong truoc do cho marsbot
    lw $s6, 0($t8)

    beqz $t6, turnLeft    # Neu isRight = 0 thi marsbot
    nop                   # dang re trai
turnRight:
        add $s6, $s6, 90
    j next_repeat
    nop
```

```
turnLeft:
    add $s6, $s6, -90
next_repeat:
    sw $s6, 0($t8)
    jal storePath
    jal ROTATE
    nop

    beqz    $s0, noTrack
    nop
    jal    TRACK
noTrack:    nop

    beqz    $t9, noGo
    nop
    jal    GO
noGo:    nop
repeat_end:
    # restore
    lw $s0, 0($sp)
    addi $sp,$sp,-4
    lw $t9, 0($sp)
    addi $sp,$sp,-4
    lw $t8, 0($sp)
    addi $sp,$sp,-4
    lw $t7, 0($sp)
    addi $sp,$sp,-4
    lw $t6, 0($sp)
    addi $sp,$sp,-4
    lw $s7, 0($sp)
    addi $sp,$sp,-4
    lw $s6, 0($sp)
    addi $sp,$sp,-4
    lw $s5, 0($sp)
    addi $sp,$sp,-4
```

```
    j printControlCode
# Print control code to console
printControlCode:
    li $v0, 4
    la $a0, inputControlCode
    syscall
    nop

# Xoa lenh vua duoc nhap vao va quay lai cho user nhap lenh moi
continue:
    jal remove
    nop
    j loop
    nop
    j loop
#-------------------------------------------------------------
# luu canh hien tai vao mang path
# param[in]     nowHeading
#         lengthPath
#-------------------------------------------------------------
storePath:
    # backup
    addi $sp,$sp,4
    sw $t1, 0($sp)
    addi $sp,$sp,4
    sw $t2, 0($sp)
    addi $sp,$sp,4
    sw $t3, 0($sp)
    addi $sp,$sp,4
    sw $t4, 0($sp)
    addi $sp,$sp,4
    sw $s1, 0($sp)
    addi $sp,$sp,4
    sw $s2, 0($sp)
```

```
addi $sp,$sp,4
sw $s3, 0($sp)
addi $sp,$sp,4
sw $s4, 0($sp)

# processing
li $t1, WHEREX
lw $s1, 0($t1)      # s1 = x
abs $s1, $s1

li $t2, WHEREY
lw $s2, 0($t2)      # s2 = y
abs $s2, $s2

la $s4, nowHeading
lw $s4, 0($s4)      # s4 = now heading

la $t3, lengthPath
lw $s3, 0($t3)      # $s3 = lengthPath (dv: byte)

la $t4, path
add $t4, $t4, $s3    # vi tri bat dau luu

sw $s1, 0($t4)      # luu x
sw $s2, 4($t4)      # luu y
sw $s4, 8($t4)      # luu huong hien tai heading

addi $s3, $s3, 12    # lengthPath = lengthPath + 12
                     # 12 = 3 (word) x 4 (bytes)
sw $s3, 0($t3)

# restore
lw $s4, 0($sp)
addi $sp,$sp,-4
lw $s3, 0($sp)
```

```
    addi $sp,$sp,-4
    lw $s2, 0($sp)
    addi $sp,$sp,-4
    lw $s1, 0($sp)
    addi $sp,$sp,-4
    lw $t4, 0($sp)
    addi $sp,$sp,-4
    lw $t3, 0($sp)
    addi $sp,$sp,-4
    lw $t2, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4

    jr $ra
    nop
    jr $ra
#-------------------------------------------------------
# Dieu khien cho marsbot di nguoc lai
#-------------------------------------------------------
goBack:
    # Disable interrupts when going backward
    li    $t7, IN_ADDRESS_HEXA_KEYBOARD
          sb    $zero, 0($t7)

    # backup
    addi $sp,$sp,4
    sw $s5, 0($sp)
    addi $sp,$sp,4
    sw $s6, 0($sp)
    addi $sp,$sp,4
    sw $s7, 0($sp)
    addi $sp,$sp,4
    sw $t8, 0($sp)
    addi $sp,$sp,4
```

```
    sw $t9, 0($sp)
    addi $sp,$sp,4
    sw $t1, 0($sp)
    addi $sp,$sp,4
    sw $t3, 0($sp)

    jal UNTRACK
    jal GO
    la $s7, path
    la $s5, lengthPath
    lw $s5, 0($s5)     # $s5 = lengthPath
    add $s7, $s7, $s5    # $s7 = &path[lengthPath]
begin:
    addi $s5, $s5, -12     # lui lai 1 structure

    addi $s7, $s7, -12    # vi tri luu tru thong tin ve canh cuoi cung
    lw $s6, 8($s7)     # huong cua canh cuoi cung
    addi $s6, $s6, 180    # dao nguoc lai huong cua canh cuoi cung



    la $t8, nowHeading    # marsbot quay nguoc lai
    sw $s6, 0($t8)
    jal ROTATE

loop_go_back:
    lw $t9, 0($s7)     # toa do x cua diem cuoi cua canh
    li $t8, WHEREX          # toa do x hien tai
    lw $t8, 0($t8)
    abs $t8, $t8

    bne $t8, $t9, loop_go_back
    nop
    bne $t8, $t9, loop_go_back

    lw $t9, 4($s7)     # toa do y cua diem cuoi cua canh
```

```
    li $t8, WHEREY          # toa do y hien tai
    lw $t8, 0($t8)
    abs $t8, $t8

    bne $t8, $t9, loop_go_back
    nop
    bne $t8, $t9, loop_go_back

    beq $s5, 0, finish
    nop
    beq $s5, 0, finish

    j begin
    nop
    j begin
finish:
    jal STOP

    la $t8, nowHeading
    li $t9, 90
    sw $t9, 0($t8)      # cap nhat lai huong
    la $t8, lengthPath
    addi $s5, $zero, 12
    sw $s5, 0($t8)      # gan lai lengthPath = 12


    li $t1, IN_ADDRESS_HEXA_KEYBOARD
    li $t3, 0x80 # bit 7 = 1 to enable
    sb $t3, 0($t1)
    # restore
    lw $t3, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4
    lw $t9, 0($sp)
```

```
    addi $sp,$sp,-4
    lw $t8, 0($sp)
    addi $sp,$sp,-4
    lw $s7, 0($sp)
    addi $sp,$sp,-4
    lw $s6, 0($sp)
    addi $sp,$sp,-4
    lw $s5, 0($sp)
    addi $sp,$sp,-4

    jal ROTATE
    j printControlCode
#---------------------------------------------------------
# Dieu khien marsbot de lai vet va in ra control code
#---------------------------------------------------------
track:     jal TRACK
    j printControlCode
#---------------------------------------------------------
# Dieu khien marsbot khong de lai vet va in ra control code
#---------------------------------------------------------
untrack: jal UNTRACK
     j printControlCode
#---------------------------------------------------------
# Dieu khien cho marsbot bat dau di chuyen va in ra control code
#---------------------------------------------------------
go:
    # backup
    addi $sp,$sp,4
    sw $t6, 0($sp)
    addi $sp,$sp,4
    sw $t7, 0($sp)

    la $t6, isRight
    li $t7, 2     # Neu ma nhap vao la 1b4 thi
                          # luc bam space marsbot se khong doi
```

```
                        # huong
    sw $t7, 0($t6)

    # restore
    lw $t6, 0($sp)
    addi $sp,$sp,-4
    lw $t7, 0($sp)
    addi $sp,$sp,-4
    jal GO
    j printControlCode
#-------------------------------------------------------------
# Dieu khien cho marsbot dung lai va in ra control code
#-------------------------------------------------------------
stop:     jal STOP
    j printControlCode
#-------------------------------------------------------------
# Dieu khien cho marsbot di sang phai va in ra control code
#-------------------------------------------------------------
# goRight procedure, control marsbot to go left and print control code
# param[in] nowHeading
# param[out] nowHeading
#-------------------------------------------------------------
goRight:
    # backup
    addi $sp,$sp,4
    sw $s5, 0($sp)
    addi $sp,$sp,4
    sw $s6, 0($sp)
    addi $sp,$sp,4
    sw $t6, 0($sp)
    #addi $sp,$sp,4
    #sw $t7, 0($sp)
    #addi $sp,$sp,4
    #sw $s0, 0($sp)
    addi $sp,$sp,4
```

```
sw $t8, 0($sp)

# processsing
lw $t7, isGoing
lw $s0, isTracking
la $t8, isRight
jal STOP
nop
jal UNTRACK
nop

la $s5, nowHeading
lw $s6, 0($s5)     # $s6 - huong hien tai
addi $s6, $s6, 90
sw $s6, 0($s5)     # cap nhat lai nowHeading = nowHeading + 90

li $t6, 1
sw $t6, 0($t8)     # isRight = 1

# restore
lw $t8, 0($sp)
addi $sp,$sp,-4
#lw $s0, 0($sp)
#addi $sp,$sp,-4
#lw $t7, 0($sp)
#ddi $sp,$sp,-4
lw $t6, 0($sp)
addi $sp,$sp,-4
lw $s6, 0($sp)
addi $sp,$sp,-4
lw $s5, 0($sp)
addi $sp,$sp,-4

jal storePath
jal ROTATE
```

```
    beqz    $s0, noTrack1
    nop
    jal    TRACK
noTrack1:    nop

    beqz    $t7, noGo1
    nop
    jal    GO
noGo1:    nop

    j printControlCode
#-------------------------------------------------------------
#dieu khien cho marsbot di sang trai va in ra control code
#-------------------------------------------------------------
goLeft:
    # backup
    addi $sp,$sp,4
    sw $s5, 0($sp)
    addi $sp,$sp,4
    sw $s6, 0($sp)
    # add
    # sw $t7, 0($sp)
    # addi $sp,$sp,4
    # sw $s0, 0($sp)
    addi $sp,$sp,4
    sw $t8, 0($sp)

    # processing
    lw $t7, isGoing
    lw $s0, isTracking
    la $t8, isRight
    jal    STOP
    nop
    jal    UNTRACK
```

```
    nop

    la $s5, nowHeading
    lw $s6, 0($s5)     # $s6 huong hien tai
    addi $s6, $s6, -90
    sw $s6, 0($s5)              # cap nhat nowHeading = nowHeading - 90

    sw $zero, 0($t8)            # isRight = 0

    # restore
    lw $t8, 0($sp)
    addi $sp,$sp,-4
    #lw $s0, 0($sp)
    #addi $sp,$sp,-4
    #lw $t7, 0($sp)
    #addi $sp,$sp,-4
    lw $s6, 0($sp)
    addi $sp,$sp,-4
    lw $s5, 0($sp)
    addi $sp,$sp,-4

    jal storePath
    jal ROTATE

    beqz    $s0, noTrack2
    nop
    jal    TRACK
noTrack2:    nop

    beqz    $t7, noGo2
    nop
    jal    GO
noGo2:    nop
    j printControlCode
#-----------------------------------------------------------
```

```
#Xoa inputControlCode
#-------------------------------------------------------------
remove:
   # backup
   addi $sp,$sp,4
   sw $t1, 0($sp)
   addi $sp,$sp,4
   sw $t2, 0($sp)
   addi $sp,$sp,4
   sw $s1, 0($sp)
   addi $sp,$sp,4
   sw $t3, 0($sp)
   addi $sp,$sp,4
   sw $s2, 0($sp)

   # processing
   # Tao 1 vong lap duyet qua tung ky tu cua mang inputControlCode
   # va thay the no bang '\0', sau do cap nhat lai lengthControlCode
   la $s2, lengthControlCode
   lw $t3, 0($s2)                  # $t3 = lengthControlCode
   addi $t1, $zero, -1             # $t1 = -1 = i
   la $s1, inputControlCode        # $s1 = &inputControlCode
   addi $s1, $s1, -1               # $s1 = &inputControlCode - 1
   remove_input_code:
         addi $t1, $t1, 1      # i++

         add $s1, $s1, 1             # $s1 = &inputControlCode + 1
         sb $zero, 0($s1)     # inputControlCode[i] = '\0'

         bne $t1, $t3, remove_input_code    # Neu $t1 <= 3 quay lai
for_loop_to_remove
         nop
         bne $t1, $t3, remove_input_code

   sw $zero, 0($s2)                        # lengthControlCode = 0
```

```
    # restore
    lw $s2, 0($sp)
    addi $sp,$sp,-4
    lw $t3, 0($sp)
    addi $sp,$sp,-4
    lw $s1, 0($sp)
    addi $sp,$sp,-4
    lw $t2, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4

    jr $ra
    nop
    jr $ra
#------------------------------------------------------------
# Kiem tra inputControlCode co trung voi cac code dieu khien
# neu dung thi $t0 = 1 con khong dung thi $t0 = 0
#------------------------------------------------------------
isEqual:
    #backup
    addi $sp,$sp,4
    sw $t1, 0($sp)
    addi $sp,$sp,4
    sw $s1, 0($sp)
    addi $sp,$sp,4
    sw $t2, 0($sp)
    addi $sp,$sp,4
    sw $t3, 0($sp)

    #processing
    addi $t1, $zero, -1             # $t1 = -1 = i
    la $s1, inputControlCode        # $s1 = &inputControlCode
    check_equal:
```

```
        addi $t1, $t1, 1       # i++

        add $t2, $s1, $t1     # $t2 = inputControlCode + i
        lb $t2, 0($t2)                # $t2 = inputControlCode[i]

        add $t3, $s3, $t1     # $t3 = s + i
        lb $t3, 0($t3)                # $t3 = s[i]

        bne $t2, $t3, notEqual      # Neu $t2 != $t3 -> khong trung
        nop

        bne $t1, 2, check_equal           # Neu $t1 <=2 quay lai check_equal
        nop
        bne $t1, 2, check_equal
equal:
   # restore
   lw $t3, 0($sp)
   addi $sp,$sp,-4
   lw $t2, 0($sp)
   addi $sp,$sp,-4
   lw $s1, 0($sp)
   addi $sp,$sp,-4
   lw $t1, 0($sp)
   addi $sp,$sp,-4

   add $t0, $zero, 1    # $t0 = 1 -> return true
   jr $ra
   nop
   jr $ra
notEqual:
   #restore
   lw $t3, 0($sp)
   addi $sp,$sp,-4
   lw $t2, 0($sp)
   addi $sp,$sp,-4
```

```
    lw $s1, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4

    add $t0, $zero, $zero    # $t0 = 0 -> return false
    jr $ra
    nop
    jr $ra
#------------------------------------------------------------
# Thong bao loi
#------------------------------------------------------------
error:
    li $v0, 4
    la $a0, inputControlCode
    syscall
    nop

    li $v0, 55
    la $a0, WRONG_CODE
    syscall
    nop

    nop
    j continue
    nop
    j continue
#------------------------------------------------------------
# Bat dau di chuyen
#------------------------------------------------------------
GO:
    # backup
    addi $sp,$sp,4
    sw $at,0($sp)
    addi $sp,$sp,4
```

```
    sw $k0,0($sp)
    addi $sp,$sp,4
    sw $t7,0($sp)

    # processing
    li $at, MOVING          # change MOVING port
          addi $k0, $zero,1     # to logic 1,
    sb $k0, 0($at)      # to start running

    li    $t7, 1                # isGoing = 1
    sw    $t7, isGoing

    # restore
    lw $t7, 0($sp)
    addi $sp,$sp,-4
    lw $k0, 0($sp)
    addi $sp,$sp,-4
    lw $at, 0($sp)
    addi $sp,$sp,-4

    jr $ra
    nop
    jr $ra
#-----------------------------------------------------------
# Dung marsbot
#-----------------------------------------------------------
STOP:     # backup
    addi $sp,$sp,4
    sw $at,0($sp)

    # processing
    li $at, MOVING          #change MOVING port to 0
    sb $zero, 0($at)     #to stop

    sw $zero, isGoing    # isGoing = 0
```

```
    # restore
    lw $at, 0($sp)
    addi $sp,$sp,-4

    jr $ra
    nop
    jr $ra
#-------------------------------------------------------------
# Bat dau de lai vet
#-------------------------------------------------------------
TRACK:
    # backup
    addi $sp,$sp,4
    sw $at,0($sp)
    addi $sp,$sp,4
    sw $k0,0($sp)
    addi $sp,$sp,4
    sw $s0,0($sp)

    # processing
    li $at, LEAVETRACK     # change LEAVETRACK port
    addi $k0, $zero,1     # to logic 1,
        sb $k0, 0($at)        # to start tracking

        addi    $s0, $zero, 1
        sw    $s0, isTracking    # set isTracking = 1

        # restore
    lw $s0, 0($sp)
    addi $sp,$sp,-4
    lw $k0, 0($sp)
    addi $sp,$sp,-4
    lw $at, 0($sp)
    addi $sp,$sp,-4
```

```
        jr $ra
    nop
    jr $ra
#-----------------------------------------------------------
# Dung de lai vet
#-----------------------------------------------------------
UNTRACK:
    # backup
    addi $sp,$sp,4
    sw $at,0($sp)

    # processing
    li $at, LEAVETRACK     #change LEAVETRACK port to 0
        sb $zero, 0($at)     #to stop drawing tail

        sw    $zero, isTracking    # set isTracking = 0
        # restore
    lw $at, 0($sp)
    addi $sp,$sp,-4

        jr $ra
    nop
    jr $ra
#-----------------------------------------------------------
# chuyen huong cho marsbot
#-----------------------------------------------------------
ROTATE:
    # backup
    addi $sp,$sp,4
    sw $t1,0($sp)
    addi $sp,$sp,4
    sw $t2,0($sp)
    addi $sp,$sp,4
    sw $t3,0($sp)
```

```
    # processing
    li $t1, HEADING     # change HEADING port
    la $t2, nowHeading
    lw $t3, 0($t2)
        sw $t3, 0($t1)        # to rotate marsbot

        # restore
        lw $t3, 0($sp)
    addi $sp,$sp,-4
    lw $t2, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4

        jr $ra
    nop
    jr $ra

#=============================================================
======================
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~
.ktext 0x80000180
#-----------------------------------------------------
# SAVE the current REG FILE to stack
#-----------------------------------------------------
backup:
    addi $sp,$sp,4
    sw $ra,0($sp)
    addi $sp,$sp,4
    sw $t1,0($sp)
    addi $sp,$sp,4
    sw $t2,0($sp)
    addi $sp,$sp,4
```

```
    sw $t3,0($sp)
    addi $sp,$sp,4
    sw $a0,0($sp)
    addi $sp,$sp,4
    sw $at,0($sp)
    addi $sp,$sp,4
    sw $s0,0($sp)
    addi $sp,$sp,4
    sw $s1,0($sp)
    addi $sp,$sp,4
    sw $s2,0($sp)
    addi $sp,$sp,4
    sw $t4,0($sp)
    addi $sp,$sp,4
    sw $s3,0($sp)


#-----------------------------------------------------
# Processing - Nhap ma dieu khien
#-----------------------------------------------------

get_code:
    li $t1, IN_ADDRESS_HEXA_KEYBOARD
    li $t2, OUT_ADDRESS_HEXA_KEYBOARD
scan_row1:
    li $t3, 0x81
    sb $t3, 0($t1)
    lbu $a0, 0($t2)     # Lay ki tu sau khi an
    bnez $a0, get_code_in_char
scan_row2:
    li $t3, 0x82
    sb $t3, 0($t1)
    lbu $a0, 0($t2)
    bnez $a0, get_code_in_char
scan_row3:
```

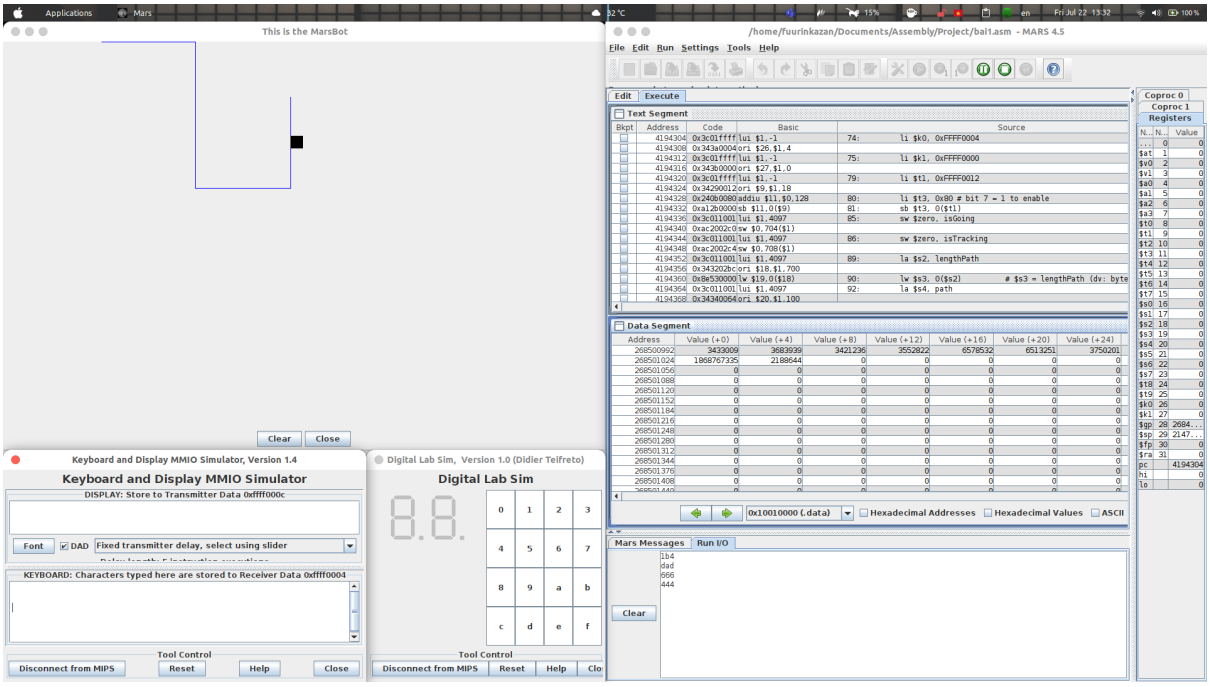
```
    li $t3, 0x84
    sb $t3, 0($t1)
    lbu $a0, 0($t2)
    bnez $a0, get_code_in_char
scan_row4:
    li $t3, 0x88
    sb $t3, 0($t1)
    lbu $a0, 0($t2)
    bnez $a0, get_code_in_char

# Kiem tra ki tu nhap vao khop voi key nao
get_code_in_char:
    beq $a0, KEY_0, case_0
    beq $a0, KEY_1, case_1
    beq $a0, KEY_2, case_2
    beq $a0, KEY_3, case_3
    beq $a0, KEY_4, case_4
    beq $a0, KEY_5, case_5
    beq $a0, KEY_6, case_6
    beq $a0, KEY_7, case_7
    beq $a0, KEY_8, case_8
    beq $a0, KEY_9, case_9
    beq $a0, KEY_a, case_a
    beq $a0, KEY_b, case_b
    beq $a0, KEY_c, case_c
    beq $a0, KEY_d, case_d
    beq $a0, KEY_e, case_e
    beq $a0, KEY_f, case_f

    #$s0 store code in char type
case_0:    li $s0, '0'
    j store_code
case_1:    li $s0, '1'
    j store_code
case_2:    li $s0, '2'
```

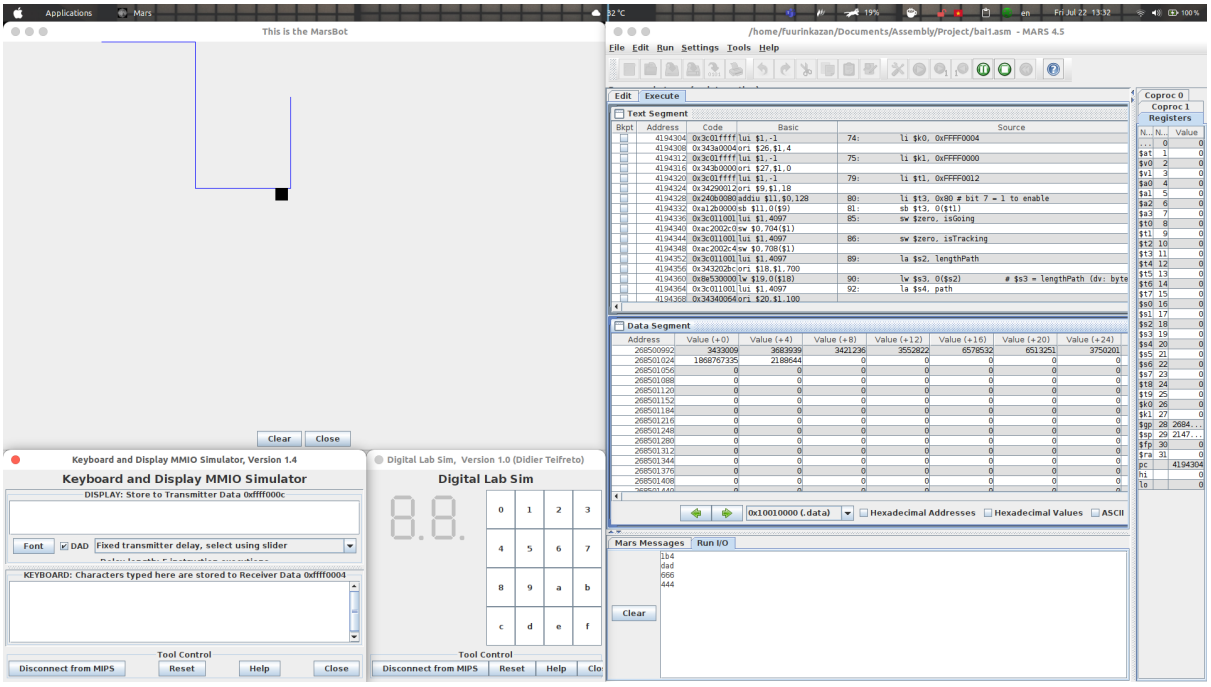
```
    j store_code
case_3:    li $s0, '3'
    j store_code
case_4:    li $s0, '4'
    j store_code
case_5:    li $s0, '5'
    j store_code
case_6:    li $s0, '6'
    j store_code
case_7:    li $s0, '7'
    j store_code
case_8:    li $s0, '8'
    j store_code
case_9:    li $s0, '9'
    j store_code
case_a:    li $s0, 'a'
    j store_code
case_b:    li $s0, 'b'
    j store_code
case_c:    li $s0, 'c'
    j store_code
case_d:    li $s0, 'd'
    j store_code
case_e:    li $s0,    'e'
    j store_code
case_f:    li $s0, 'f'
    j store_code
store_code:
    la $s1, inputControlCode    # $s1 = &inputControlCode
    la $s2, lengthControlCode    # $s2 = &lengthControlCode
    lw $s3, 0($s2)              # $s3 = strlen(inputControlCode)
    addi $t4, $t4, -1           # $t4 = i = -1

    for_loop_to_store_code:
        addi     $t4, $t4, 1            # i++
```

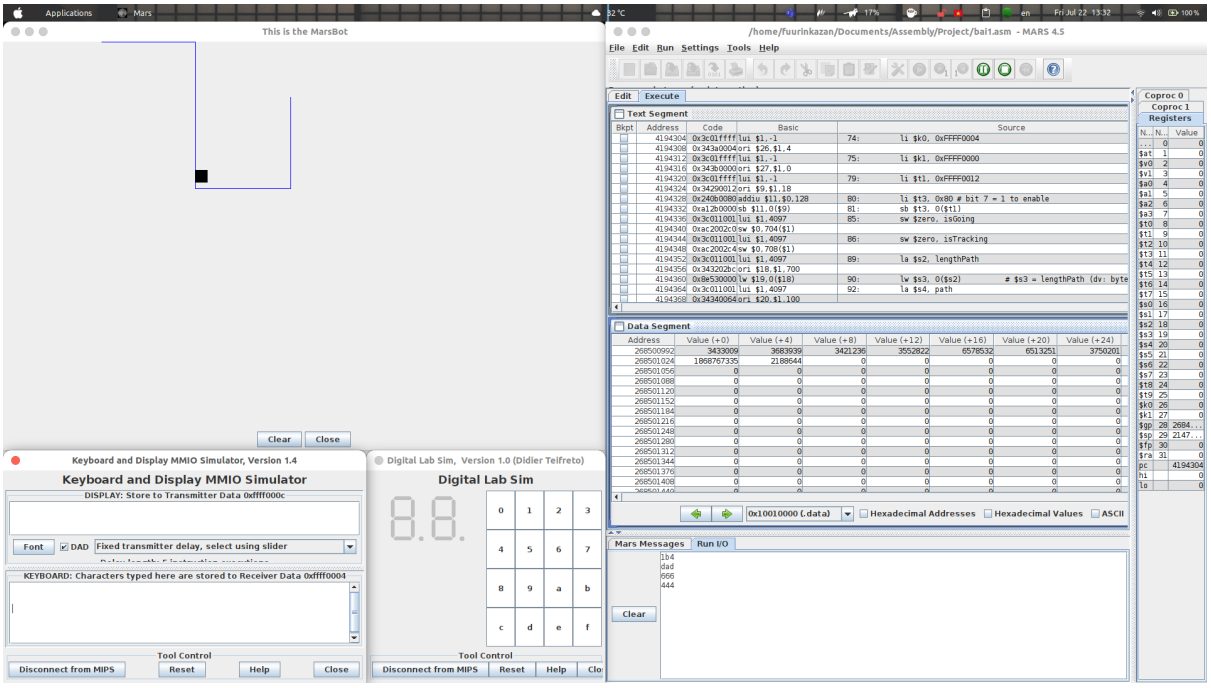
```
    # Neu i != lengthControlCode thi quay lai vong lap
    # va tang i len 1. Cho den khi i = lengthControlCode
    # Luc nay i chinh la vi tri can dien ky tu (code) vua nhap vao
        bne     $t4, $s3, for_loop_to_store_code

        # Gan ky tu moi cho s[i]
        add     $s1, $s1, $t4         # $s1 = inputControlCode[i]
        sb      $s0, 0($s1)           # inputControlCode[i] = $s0

        # Them ky tu ket thuc chuoi vao cuoi chuoi inputControlCode
        addi     $s0, $zero, '\n'
        addi     $s1, $s1, 1
        sb      $s0, 0($s1)

        # Cap nhat do dai cua chuoi inputControlCode
        addi $s3, $s3, 1
        sw $s3, 0($s2)

#---------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#---------------------------------------------------------
next_pc:
    mfc0 $at, $14      # $at <= Coproc0.$14 = Coproc0.epc
    addi $at, $at, 4     # $at = $at + 4 (next instruction)
    mtc0 $at, $14      # Coproc0.$14 = Coproc0.epc <= $at
#---------------------------------------------------------
# RESTORE the REG FILE from STACK
#---------------------------------------------------------
restore:
    lw $s3, 0($sp)
    addi $sp,$sp,-4
    lw $t4, 0($sp)
    addi $sp,$sp,-4
    lw $s2, 0($sp)
```

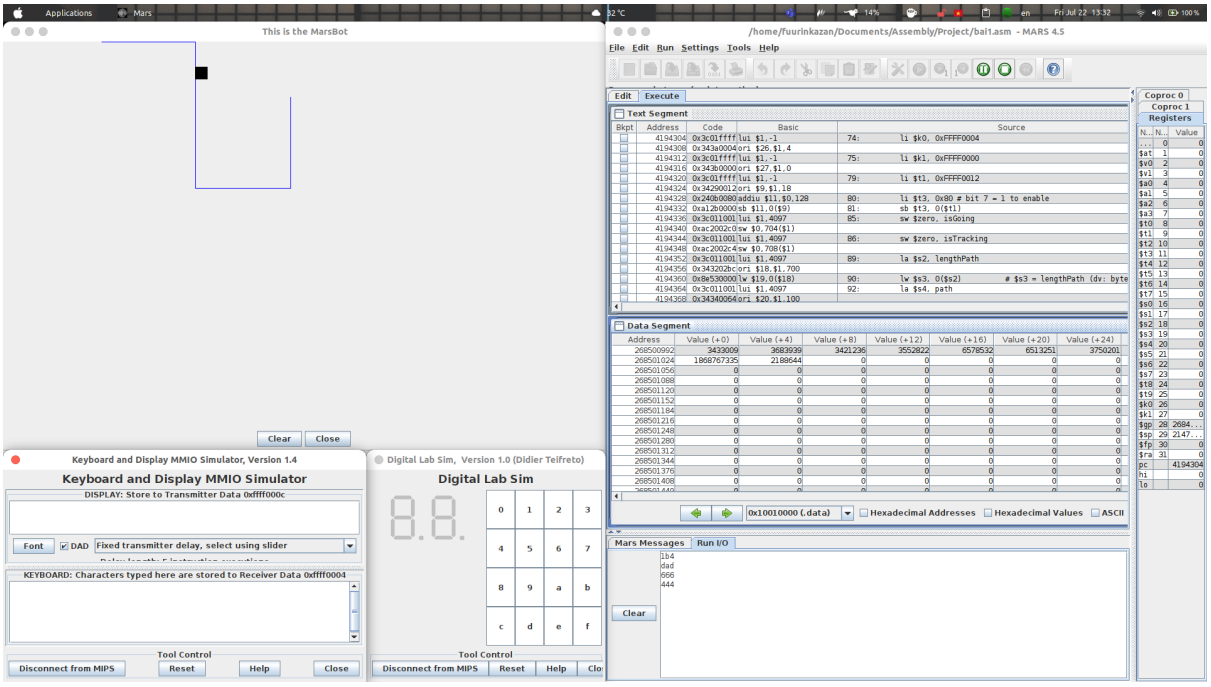
```
    addi $sp,$sp,-4
    lw $s1, 0($sp)
    addi $sp,$sp,-4
    lw $s0, 0($sp)
    addi $sp,$sp,-4
    lw $at, 0($sp)
    addi $sp,$sp,-4
    lw $a0, 0($sp)
    addi $sp,$sp,-4
    lw $t3, 0($sp)
    addi $sp,$sp,-4
    lw $t2, 0($sp)
    addi $sp,$sp,-4
    lw $t1, 0($sp)
    addi $sp,$sp,-4
    lw $ra, 0($sp)
    addi $sp,$sp,-4
return: eret # Return from exception
```

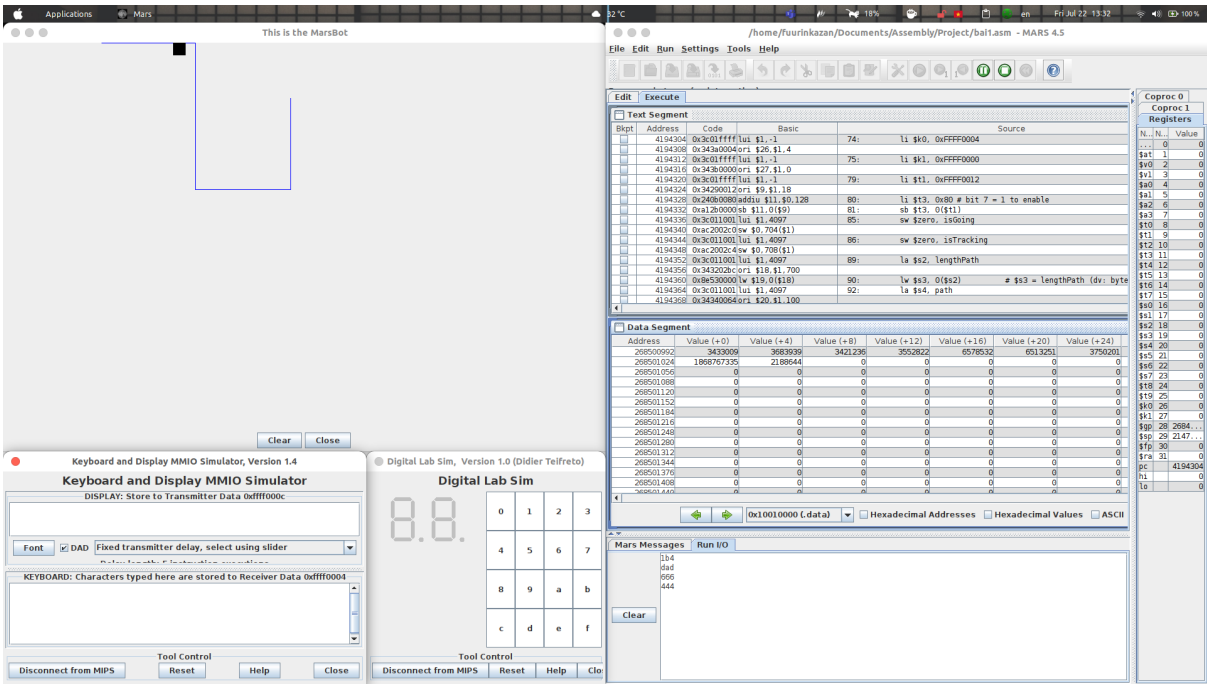
1. **5. Demo**:

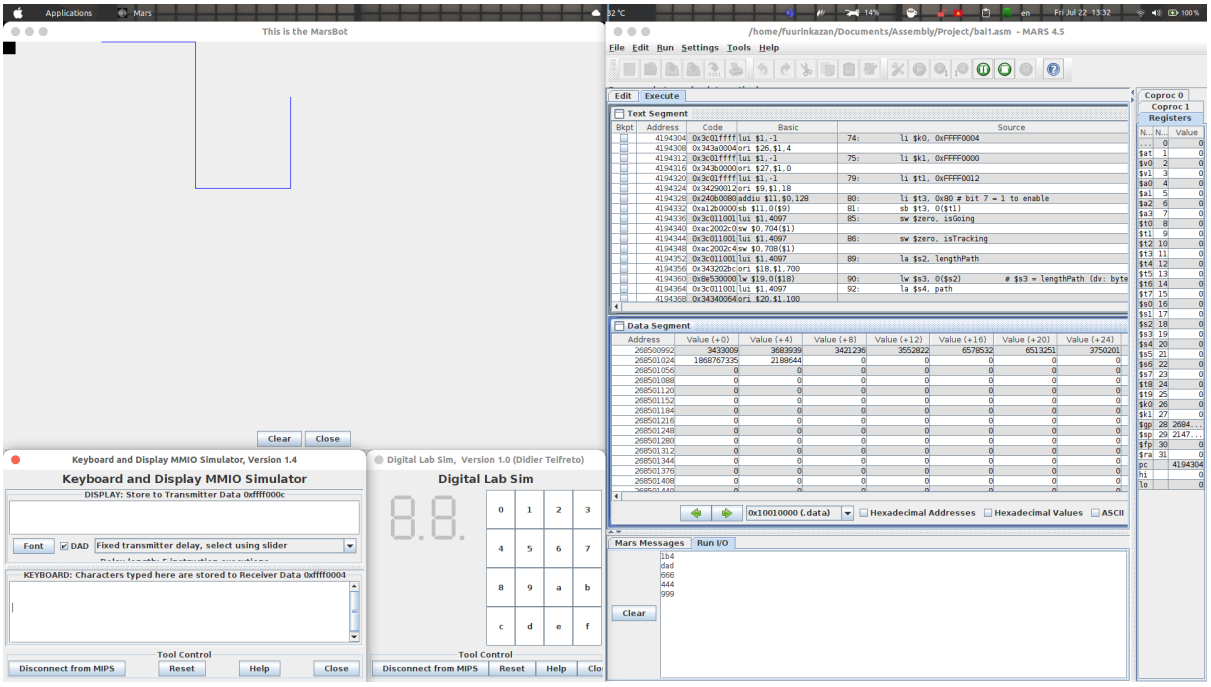

Applications
Mars
This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0 1 2 3
4 5 6 7
8 9 a b
c d e f
Tool Control
Disconnect from MIPS
Reset
Help
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Bkpt Address Code Basic Source
4194304 0x3c01ffff lui $1,-1 74: li $k0, 0xFFFF0004
4194308 0x343a0004 ori $26,$1,4
4194312 0x3c01ffff lui $1,-1 75: li $k1, 0xFFFF0000
4194316 0x343b0000 ori $27,$1,0
4194320 0x3c01ffff lui $1,-1 79: li $t1, 0xFFFF0012
4194324 0x34290012 ori $9,$1,18
4194328 0x240b0080 addiu $11,$0,128 80: li $t3, 0x80 # bit 7 = 1 to enable
4194332 0xa12b0000 sb $11,0($9) 81: sb $t3, 0($t1)
4194336 0x3c011001 lui $1,4097 85: sw $zero, isGoing
4194340 0xac2002c0 sw $0,704($1)
4194344 0x3c011001 lui $1,4097 86: sw $zero, isTracking
4194348 0xac2002c4 sw $0,708($1)
4194352 0x3c011001 lui $1,4097 89: la $s2, lengthPath
4194356 0x343202bc ori $18,$1,700
4194360 0x8e530000 lw $19,0($18) 90: lw $s3, 0($s2) # $s3 = lengthPath (dv: byte
4194364 0x3c011001 lui $1,4097 92: la $s4, path
4194368 0x34340064 ori $20,$1,100
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24)
268500992 3433009 3683939 3421236 3552822 6578532 6513251 3750201
268501024 1868767335 2188644 0 0 0 0 0
268501056 0 0 0 0 0 0 0
268501088 0 0 0 0 0 0 0
268501120 0 0 0 0 0 0 0
268501152 0 0 0 0 0 0 0
268501184 0 0 0 0 0 0 0
268501216 0 0 0 0 0 0 0
268501248 0 0 0 0 0 0 0
268501280 0 0 0 0 0 0 0
268501312 0 0 0 0 0 0 0
268501344 0 0 0 0 0 0 0
268501376 0 0 0 0 0 0 0
268501408 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
1b4
dad
666
444
Clear
Coproc 0
Coproc 1
Registers
N... N... Value
$at 1 0
$v0 2 0
$v1 3 0
$a0 4 0
$a1 5 0
$a2 6 0
$a3 7 0
$t0 8 0
$t1 9 0
$t2 10 0
$t3 11 0
$t4 12 0
$t5 13 0
$t6 14 0
$t7 15 0
$s0 16 0
$s1 17 0
$s2 18 0
$s3 19 0
$s4 20 0
$s5 21 0
$s6 22 0
$s7 23 0
$t8 24 0
$t9 25 0
$k0 26 0
$k1 27 0
$gp 28 2684...
$sp 29 2147...
$fp 30 0
$ra 31 0
pc 4194304
hi 0
lo 0

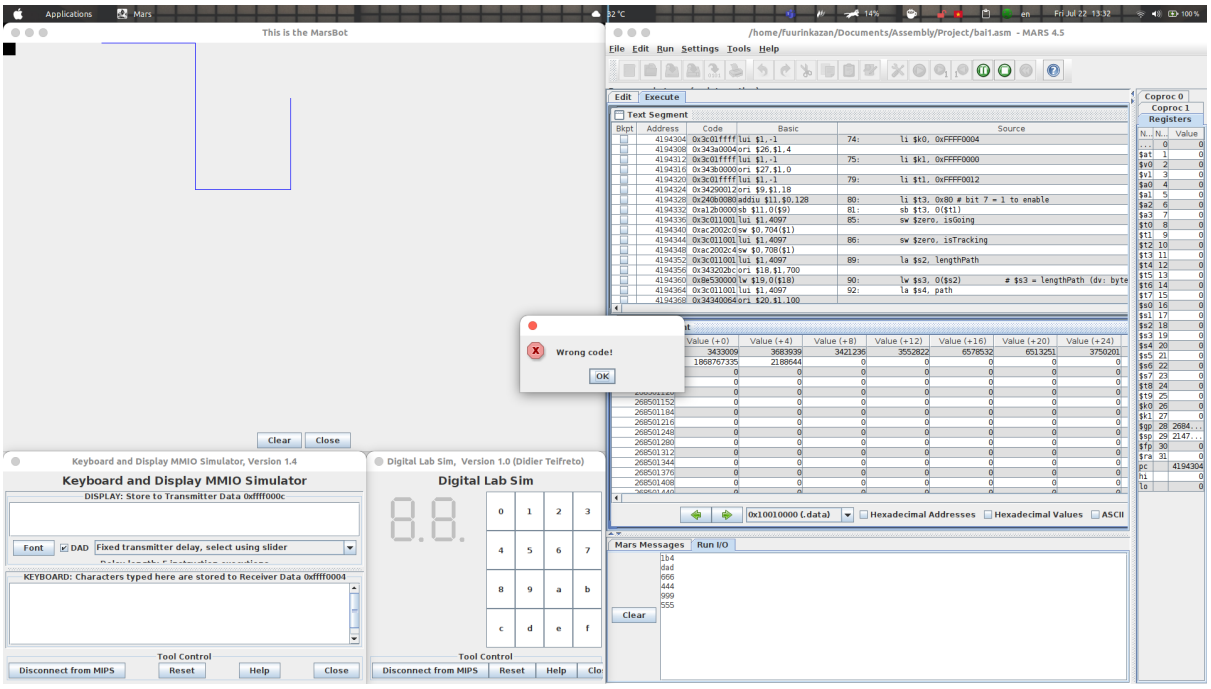

Applications
Mars
This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit
Execute
Text Segment
Bkpt Address Code Basic Source
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24)
Coproc 0
Coproc 1
Registers
Mars Messages
Run I/O
1b4
dad
666
444
Clear

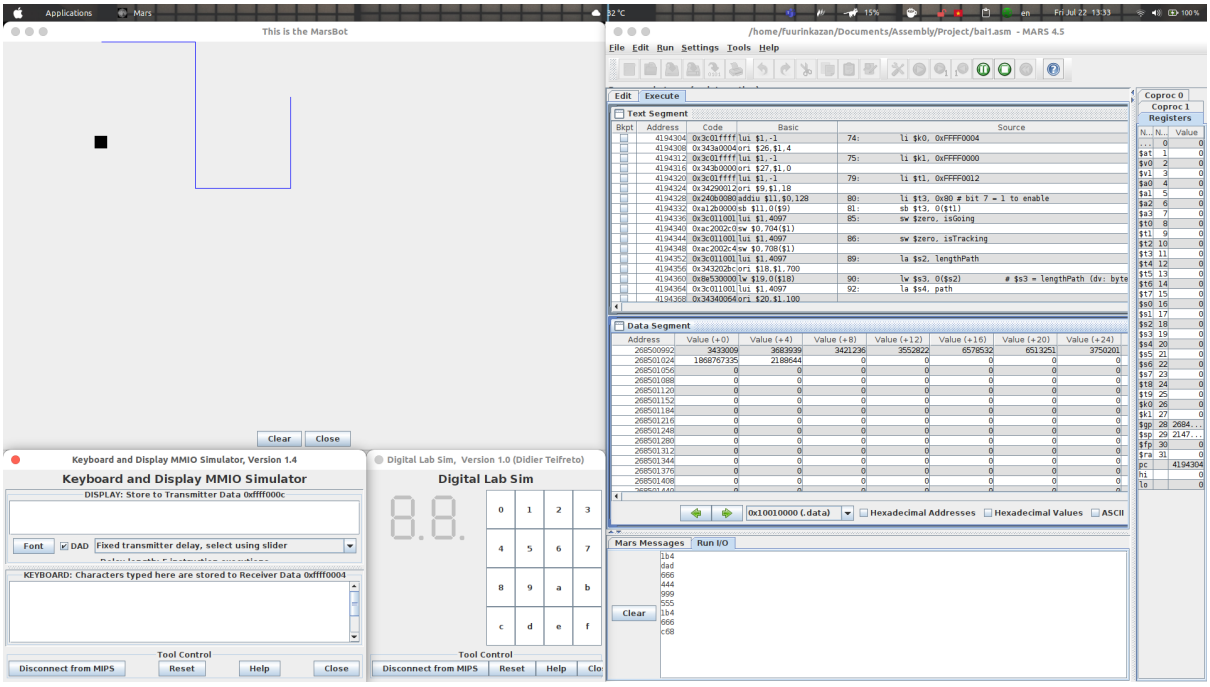

Applications
Mars
This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit
Execute
Text Segment
Bkpt Address Code Basic Source
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24)
Coproc 0
Coproc 1
Registers
Mars Messages
Run I/O
1b4
dad
666
444
Clear


Applications
Mars
This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit
Execute
Text Segment
Bkpt Address Code Basic Source
4194304 0x3c01ffff lui $1,-1 74: li $k0, 0xFFFF0004
4194308 0x343a0004 ori $26,$1,4
4194312 0x3c01ffff lui $1,-1 75: li $k1, 0xFFFF0000
4194316 0x343b0000 ori $27,$1,0
4194320 0x3c01ffff lui $1,-1 79: li $t1, 0xFFFF0012
4194324 0x34290012 ori $9,$1,18
4194328 0x240b0080 addiu $11,$0,128 80: li $t3, 0x80 # bit 7 = 1 to enable
4194332 0xa12b0000 sb $11,0($9) 81: sb $t3, 0($t1)
4194336 0x3c011001 lui $1,4097 85: sw $zero, isGoing
4194340 0xac2002c0 sw $0,704($1)
4194344 0x3c011001 lui $1,4097 86: sw $zero, isTracking
4194348 0xac2002c4 sw $0,708($1)
4194352 0x3c011001 lui $1,4097 89: la $s2, lengthPath
4194356 0x343202bc ori $18,$1,700
4194360 0x8e530000 lw $19,0($18) 90: lw $s3, 0($s2) # $s3 = lengthPath (dv: byte
4194364 0x3c011001 lui $1,4097 92: la $s4, path
4194368 0x34340064 ori $20,$1,100
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24)
268500992 3433009 3683939 3421236 3552822 6578532 6513251 3750201
268501024 1868767335 2188644 0 0 0 0 0
268501056 0 0 0 0 0 0 0
268501088 0 0 0 0 0 0 0
268501120 0 0 0 0 0 0 0
268501152 0 0 0 0 0 0 0
268501184 0 0 0 0 0 0 0
268501216 0 0 0 0 0 0 0
268501248 0 0 0 0 0 0 0
268501280 0 0 0 0 0 0 0
268501312 0 0 0 0 0 0 0
268501344 0 0 0 0 0 0 0
268501376 0 0 0 0 0 0 0
268501408 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
1b4
dad
666
444
Clear
Coproc 0
Coproc 1
Registers
Value
$at 0
$v0 0
$v1 0
$a0 0
$a1 0
$a2 0
$a3 0
$t0 0
$t1 0
$t2 0
$t3 0
$t4 0
$t5 0
$t6 0
$t7 0
$s0 0
$s1 0
$s2 0
$s3 0
$s4 0
$s5 0
$s6 0
$s7 0
$t8 0
$t9 0
$k0 0
$k1 0
$gp 268468224
$sp 2147483612
$fp 0
$ra 0
pc 4194304
hi 0
lo 0


Applications
Mars
This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit
Execute
Text Segment
Bkpt Address Code Basic Source
4194304 0x3c01ffff lui $1,-1 74: li $k0, 0xFFFF0004
4194308 0x343a0004 ori $26,$1,4
4194312 0x3c01ffff lui $1,-1 75: li $k1, 0xFFFF0000
4194316 0x343b0000 ori $27,$1,0
4194320 0x3c01ffff lui $1,-1 79: li $t1, 0xFFFF0012
4194324 0x34290012 ori $9,$1,18
4194328 0x240b0080 addiu $11,$0,128 80: li $t3, 0x80 # bit 7 = 1 to enable
4194332 0xa12b0000 sb $11,0($9) 81: sb $t3, 0($t1)
4194336 0x3c011001 lui $1,4097 85: sw $zero, isGoing
4194340 0xac2002c0 sw $0,704($1)
4194344 0x3c011001 lui $1,4097 86: sw $zero, isTracking
4194348 0xac2002c4 sw $0,708($1)
4194352 0x3c011001 lui $1,4097 89: la $s2, lengthPath
4194356 0x343202bc ori $18,$1,700
4194360 0x8e530000 lw $19,0($18) 90: lw $s3, 0($s2) # $s3 = lengthPath (dv: byte
4194364 0x3c011001 lui $1,4097 92: la $s4, path
4194368 0x34340064 ori $20,$1,100
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+12) Value (+16) Value (+20) Value (+24)
268500992 3433009 3683939 3421236 3552822 6578532 6513251 3750201
268501024 1868767335 2188644 0 0 0 0 0
268501056 0 0 0 0 0 0 0
268501088 0 0 0 0 0 0 0
268501120 0 0 0 0 0 0 0
268501152 0 0 0 0 0 0 0
268501184 0 0 0 0 0 0 0
268501216 0 0 0 0 0 0 0
268501248 0 0 0 0 0 0 0
268501280 0 0 0 0 0 0 0
268501312 0 0 0 0 0 0 0
268501344 0 0 0 0 0 0 0
268501376 0 0 0 0 0 0 0
268501408 0 0 0 0 0 0 0
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
1b4
dad
666
444
Clear
Coproc 0
Coproc 1
Registers
Value
$at 0
$v0 0
$v1 0
$a0 0
$a1 0
$a2 0
$a3 0
$t0 0
$t1 0
$t2 0
$t3 0
$t4 0
$t5 0
$t6 0
$t7 0
$s0 0
$s1 0
$s2 0
$s3 0
$s4 0
$s5 0
$s6 0
$s7 0
$t8 0
$t9 0
$k0 0
$k1 0
$gp 268468224
$sp 2147483612
$fp 0
$ra 0
pc 4194304
hi 0
lo 0


This is the MarsBot
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit
Execute
Text Segment
Data Segment
Mars Messages
Run I/O
Clear
Hexadecimal Addresses
Hexadecimal Values
ASCII
Registers
Coproc 0
Coproc 1


This is the MarsBot
Wrong code!
OK
Clear
Close
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
/home/fuurinkazan/Documents/Assembly/Project/bai1.asm - MARS 4.5
Mars Messages
Run I/O
Clear



2.Chương trình kiểm tra cú pháp lệnh MIPS

2. 1. Đề bài

7. Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộ xử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ *beq s1,31,t4*
- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo *"opcode: beq, hợp lệ"*
- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, *toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.*

Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3.

2. 2. Thuật toán

Yêu cầu người dùng nhập vào một câu lệnh cần kiểm tra, lưu vào một chuỗi input .

- Một câu lệnh hợp ngữ gồm 4 phần: mã opcode, toán hạng 1, toán hạng 2, toán hạng 3. Các toán hạng được quy ước như sau:

0 – null, 1 – register, 2 – constant, 3 – label, 4 - cụm offset(base).

- Sau khi tách một thành phần của chuỗi tương ứng với thành phần của câu lệnh hợp ngữ. Đưa từng phần sau khi cắt được so sánh với kiểu dữ liệu quy ước.
- Nếu phù hợp thì tiếp tục so sánh các phần phía sau. Nếu không phù hợp thì báo cho người dùng opcode hoặc toán hạng không hợp lệ.
- Đầu chương trình, khởi tạo một chuỗi opcode chuẩn, opcode sau khi cắt được thì so sánh với chuỗi này bằng cách so sánh từng ký tự, nếu ký tự khác nhau thì so sánh với opcode tiếp theo trong chuỗi cho đến khi kết thúc.
- Nếu không có opcode phù hợp thì báo opcode không hợp lệ và exit. Nếu có opcode hợp lệ -> tìm khuôn dạng các toán hạng phù hợp với opcode. Đầu chương trình khi tạo chuỗi opcode chuẩn thì sẽ tạo một chuỗi khuôn dạng tương ứng với từng lệnh bằng cách so vị trí lệnh và khuôn lệnh. Nếu opcode phù hợp thì sẽ lấy được khuôn dạng lệnh và tiến hành kiểm tra lần lượt các toán hạng. Nếu có 1 toán hạng không hợp lệ -> báo ra màn hình run I/O và exit
- Tương tự opcode chuẩn thì cũng có một chuỗi register chuẩn được tạo ở đầu chương trình. Nếu toán hạng có mã là 1 -> nó là thanh ghi và sẽ so sánh với chuỗi các register chuẩn.
- Trường hợp toán hạng là constant: một constant hợp lệ có thể có ký tự đầu là dấu trừ ‘- ’ hoặc dấu cộng ‘+’, các ký tự phía sau bắt buộc phải từ 0 đến 9.
- Trường hợp toán hạng là label: label hợp lệ có thể có ký tự đầu là dấu gạch dưới ‘_’ hoặc chữ thường, chữ hoa, nếu có các ký tự tiếp theo, thì các ký tự này bắt buộc phải là chữ cái, chữ số hoặc dấu gạch dưới.
- Trường hợp toán hạng null: kiểm tra xem chuỗi vừa cắt được có kí tự nào hay không, nếu có -> không hợp lệ, nếu không có kí tự nào -> hợp lệ và thông báo câu lệnh hợp lệ -> Thoát chương trình.
- Trường hợp toán hạng là một cụm offset(base): kiểm tra xem dấu mở ngoặc ‘(’ và đóng ngoặc ‘)’ có hợp lệ không? Có đủ 2 dấu đóng và mở ngoặc hay không. Nếu không -> không hợp lệ. Nếu có hợp lệ -> cắt chuỗi này thành 2 thành phần là constant và register -> kiểm tra từng thành phần. Nếu có bất kì một thành phần nào sai -> Không hợp lệ. Ngược lại -> hợp lệ.
- Sau khi kiểm tra xong 3 toán hạng, chương trình sẽ kiểm tra xem còn kí tự nào khác hay không. Nếu còn -> câu lệnh không hợp lệ. Và báo kết quả của câu lệnh vừa nhập -> Hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không?
- Nếu người dùng chọn không -> Kết thúc chương trình.

2. 3. Phân tích

Phân tích chương trình con

- **compareOpcode**:

Hàm thực hiện tìm và trả về vị trí của chuỗi (opcode, register) trong dãy opcode, register chuẩn

- Tham số truyền vào : a0 (opcode, register), a1 (độ dài opcode/register), a2 (dãy opcode, register chuẩn)
- Giá trị trả về: v0 (Trả về 1 nếu opcode/register có trong dãy opcode/register chuẩn,ngược lại sẽ trả về 0), v1 (Trả về vị trí của opcode)

- CutComponent

Hàm thực hiện tách opcode và các toán hạng

- Tham số truyền vào: a0 (địa chỉ của mảng nhập), a1 (chỉ số mảng nhập), a2 (địa chỉ của opcode, toán hạng để trả về)
- Giá trị trả về: v0 (Giá trị i hiện tại), v1 (độ dài opcode/toán hạng)

- getOperand

Hàm thực hiện lấy khuôn dạng tương ứng với Opcode

- Tham số truyền vào: a0 (địa chỉ dãy opcode chuẩn), a1 (Vị trí opcode)
- Trả về khuôn dạng lệnh trong biến tmp2

- checkNumber

Kiểm tra chuỗi có phải số hay không

- Tham số truyền vào: a0 (địa chỉ dãy truyền vào), a1 (độ dài của dãy)
- Nếu là số lưu giá trị thanh ghi v0 = 1 và ngược lại lưu thanh ghi v0 = 0

- checkLabel

Kiểm tra chuỗi có phải là nhãn hay không

- Tham số truyền vào: a0 (địa chỉ dãy truyền vào), a1 (độ dài của dãy)
- Nếu là số lưu giá trị thanh ghi v0 = 1 và ngược lại lưu thanh ghi v0 = 0

- checkOffsetBase

Kiểm tra chuỗi tmp có đúng cấu trúc offset base hay không

- Tham số truyền vào: a0 (địa chỉ dãy truyền vào), a1 (độ dài của dãy)
- Nếu là số lưu giá trị thanh ghi v0 = 1 và ngược lại lưu thanh ghi v0 = 0

2. 4. Code

```
.data
        #cau lenh mips gom opcode va 3 toan hang.
        register: .asciiz
"$zero-$at-$v0-$v1-$a0-$a1-$a2-$a3-$t0-$t1-$t2-$t3-$t4-$t5-$t6-$t7-$t8-$t9-$s1-$s2-$s
3-$s4-$s5-$s6-$s7-$k0-$k1-$gp-$sp-$fp-$ra-$0-$1-$2-$3-$4-$5-$6-$7-$8-$9-$10-$11-$
```

```
12-$13-$14-$15-$16-$17-$18-$19-$20-$21-$22-$23-$24-$25-$26-$27-$28-$29-$30-$31
-"
        #ma opcode hop le:
        opcode: .asciiz
"lw-lb-sw-sb-addi-add-addiu-addu-and-andi-beq-bne-div-divu-j-jal-lui-mfhi-mflo-mul-no
p-nor-or-ori-sll-slt-slti-sub-subu-syscall-xor-xori-"
        #quy uoc toan hang: 1 - thanh ghi, 2 - so, 3 - Label, 4 - offset(base):
number(register), 0 - null
        #toan hang tuong ung voi cac opcode tren:
        operand: .asciiz
"140-140-140-140-112-111-112-111-111-112-113-113-110-110-300-300-120-100-100-11
1-000-111-111-112-112-111-112-111-111-000-111-112-"

        msg1:           .asciiz "Nhap lenh can kiem tra: "
        msg2:           .asciiz "\nopcode: "
        msg21:          .asciiz ": hop le!\n"
        msg22:          .asciiz ": khong hop le!\n"
        msg3: .asciiz "\nToan hang: "
        msg4:           .asciiz "\nCau lenh"
        msg5:           .asciiz "\kiem tra them 1 lenh nua? 1(yes)|0(no): "
        input:          .space          200             # chuỗi đầu vào
        tmp:    .space 20               # biến tmp lưu các thành phần cắt được
        tmp2:           .space          20              # lưu khuôn dạng code
        tmp3: .space            20              # thành phần cắt được offset(base)
.text
main:
Input:                                          # lấy đầu vào
        li      $v0, 4
        la      $a0, msg1
        syscall
        li      $v0, 8
        la      $a0, input
        li      $a1, 200
        syscall

#------------------------Tách chữ và so sánh----------------------
```

```
        la      $s0, input              # Lưu địa chỉ input
        add     $s1, $zero, $zero       # i -> đếm kí tự trong tmp
readOpcode:
        add     $a0, $s0, $zero         # a0 -> &input
        add     $a1, $s1, $zero         # a1 = i
        la      $a2, tmp
        jal     cutComponent
        add     $s1, $v0, $zero         # i
        add     $s7, $v1, $zero         # j - số kí tự trong opcode
checkOpcode:
        la      $a0, tmp                # a0 -> tmp
        add     $a1, $s7, $zero             # a1 = j
        la      $a2, opcode             # a2 -> opcode
        jal     compareOpcode
        add     $s2, $v0, $zero         # s2 = check
        add     $s3, $v1, $zero         # s3 = vị trí opcode
        li      $v0, 4
        la      $a0, msg2
        syscall
        li      $v0, 4
        la      $a0, tmp
        syscall
        bne     $s2, $zero, validOpcode  # if (check != 0) validOpcode    //check == 1
                                         # else invalidOpcode
invalidOpcode:                                  # opcode không hợp lệ
        li      $v0, 4
        la      $a0, msg22
        syscall
        j       exit
validOpcode:                                    # opcode hợp lệ
        li      $v0, 4
        la      $a0, msg21
        syscall

#----------------- Lấy khuôn dạng tương ứng với opcode --------------------

        la      $a0, operand
```

```
        add     $a1, $s3, $zero         # a1 = vị trí của opcode
        jal     getOperand              # Trả về tmp2(khuôn dạng code)

        li      $v0, 4
        la      $a0, tmp2
        syscall

        la      $s4, tmp2               # khuon dang
        add     $s5, $zero, $zero       # toan hang 1 2 3  - dem
        add     $t9, $zero, 48              #0
        addi    $t8, $zero, 49              #1
        addi    $t7, $zero, 50              #2
        addi    $t6, $zero, 51              #3
        addi    $t5, $zero, 52              #4
Cmp:
# -------------------Kiểm tra dạng của từng toán hạng và check----------------
        slti    $t0, $s5, 3
        beq     $t0, $zero, end             # if (s5 >= 3) end
#------------------- lấy toán hạng ----------------------
        add     $a0, $s0, $zero
        add     $a1, $s1, $zero
        la      $a2, tmp
        jal     cutComponent
        add     $s1, $v0, $zero
        add     $s7, $v1, $zero         # số kí tự trong tmp
#------------------- so sánh toán hạng ------------------
        add     $t0, $s5, $s4
        lb      $s6, 0($t0)             # dạng của toán hạng i
        beq     $s6, $t8, reg
        beq     $s6, $t7, number
        beq     $s6, $t6, label
        beq     $s6, $t5, offsetbase
        beq     $s6, $t9, null
reg:
        la      $a0, tmp
        add     $a1, $s7, $zero
        la      $a2, register
```

```
#       return  0 -> error
#               1 -> ok
        jal     compareOpcode
        j       checkValid
number:
        la      $a0, tmp
        add     $a1, $s7, $zero
        jal     checkNumber
        j       checkValid
label:
        la      $a0, tmp
        add     $a1, $s7, $zero
        jal     checkLabel
        j       checkValid
offsetbase:
        la      $a0, tmp
        add     $a1, $s7, $zero
        jal     checkOffsetBase
        j       checkValid
null:
        j       print
checkValid:
        add     $s2, $v0, $zero
        li      $v0, 4
        la      $a0, msg3
        syscall
        li      $v0, 4
        la      $a0, tmp
        syscall
        beq     $s2, $zero, error       # if (check == 0) error
        j       ok                      # else ok
updateCheck:
        addi    $s5, $s5, 1
        j       Cmp                     # Quay lại check Cmp

error:
        li      $v0, 4
```

```
        la      $a0, msg22
        syscall
        j       exit
ok:
        li      $v0, 4
        la      $a0, msg21
        syscall
        j       updateCheck
end:
        add     $a0, $s0, $zero
        add     $a1, $s1, $zero
        jal     cutComponent
        add     $s1, $v0, $zero             # i hiện tại
        add     $s7, $v1, $zero             # s7 = strlen(tmp)
print:
        li      $v0, 4
        la      $a0, msg4
        syscall
        bne     $s7, $zero, error
        li      $v0, 4
        la      $a0, msg21
        syscall
exit:
repeatMain:
        li      $v0, 4
        la      $a0, msg5
        syscall
        li      $v0, 8
        la      $a0, input
        li      $a1, 100
        syscall
        checkRepeat:
                addi    $t2, $zero, 48                  # t2 = '0'
                addi    $t3, $zero, 49                  # t3 = '1'
                add     $t0, $a0, $zero             # t0 = &input
                lb      $t0, 0($t0)                 # t0 = input[0]
                beq     $t0, $t2, out               # if (t0 == '0') out
```

```
            beq     $t0, $t3, main                  # if (t0 == '1') main
            j       repeatMain              # else repeatMain
out:
        li $v0, 10                          #exit
        syscall
#-------------------------------------------------------
# tách toán hạng, opcode từ chuỗi đầu vào
# a0 address input, a1 = i (chỉ số mảng input). a2 address tmp
# v0 = i, v1 strlen(tmp)
#-------------------------------------------------------
cutComponent:
        addi    $sp, $sp, -20
        sw      $ra, 16($sp)
        sw      $s0, 12($sp)                    # space
        sw      $s2, 8($sp)             # j
        sw      $s3, 4($sp)             # input[i]
        sw      $s4, 0($sp)             # dấu phảy = 44

        addi    $s0, $zero, 32                  # space
        addi    $t2, $zero, 10                  # \n
        addi    $s4, $zero, 44                  # dấu phảy = 44
        addi    $t3, $zero, 9                   # \t

#------ Bỏ qua dấu space và tab ------#
checkSpace:
        add     $t0, $a0, $a1                   # &input[i]
        lb      $s3, 0($t0)             # value input[i]
        beq     $s3, $s0, cutSpace      # if (input[i] == ' ')cutSpace
        beq     $s3, $t3, cutSpace      # if (input[i] == '\t')cutSpace
        beq     $s3, $s4, cutSpace      # if (input[i] == ',')cutSpace
        j       cut
cutSpace:
        addi    $a1, $a1, 1             # i++
        j       checkSpace              # Quay lại checkSpace
cut:
        add     $s2, $zero, $zero       # j = 0
loopCut:
```

```
        beq     $s3, $zero, endCut      # if (input[i] == '\0') endCut
        beq     $s3, $t2, endCut        # if (input[i] == '\n')endCut
        beq     $s3, $s0, endCut        # if (input[i] == ' ')endCut
        beq     $s3, $t3, endCut        # if (input[i] == '\t')endCut
        beq     $s3, $s4, endCut        # if (input[i] == ',')endCut

        add     $t0, $a2, $s2                   # &tmp[j]
        sb      $s3, 0($t0)             # tmp[j] = input[i]
        addi    $a1, $a1, 1             # i++
        add     $t0, $a0, $a1                   # &input[i]
        lb      $s3, 0($t0)             # value input[i]

        addi    $s2, $s2, 1             # j++
        j       loopCut
endCut:
        add     $t0, $a2, $s2                   # &tmp[j]
        sb      $zero, 0($t0)                   # tmp[j] = '\0'
        add     $v0, $a1, $zero                 # return i
        add     $v1, $s2, $zero                 # return j

        lw      $ra, 16($sp)
        lw      $s0, 12($sp)
        lw      $s2, 8($sp)
        lw      $s3, 4($sp)
        lw      $s4, 0($sp)
        addi    $sp, $sp, 20

        jr      $ra

#-----------------------------------------------------
# so sánh toán hạng, opcode với toán hạng, opcode chuẩn
# a0 = &tmp, a1 = strlen(tmp), a2  = opcode, register chuẩn
# boolean v0 = 0 or 1 (check), v1 vị trí opcode
#-----------------------------------------------------
compareOpcode:
        addi    $sp, $sp, -24
        sw      $ra, 20($sp)
```

```
        sw      $s1, 16($sp)                    # i -> opcode
        sw      $s2, 12($sp)                    # j -> tmp
        sw      $s3, 8($sp)             # value tmp[j]
        sw      $s4, 4($sp)             # value opcode[i]
        sw      $s5, 0($sp)             # '-'

        beq     $a1, $zero, endCmp              # if (a1 == 0) endCmp

        add     $s1, $zero, $zero       # s1 = i (chỉ số của chuỗi opcode)
        add     $s2, $zero, $zero       # s2 = j (chỉ số của chuỗi tmp)
        addi    $s5, $zero, 45                  # s5 = '-'
        addi    $v0, $zero, 1                   # v0 = 1
        addi    $v1, $zero, 0                   # v1 = 0
loopCmp:
        add     $t0, $a2, $s1                   # &opcode[i]
        lb      $s4, 0($t0)             # value opcode[i]
        beq     $s4, $s5, checkCmp              # if (opcode[i] == '-') checkCmp
        beq     $s4, $zero, endCmp              # if (opcode[i] == '0') endCmp
        add     $t0, $a0, $s2                   # &tmp[j]
        lb      $s3, 0($t0)             # value tmp[j]
        bne     $s3, $s4, falseCmp      # if (tmp[j] != opcode[i]) falseCmp

        addi    $s1, $s1, 1             # i++
        addi    $s2, $s2, 1             # j++
        j       loopCmp
checkCmp:
        bne     $a1, $s2, falseCmp      # if (strlen(tmp) != j) falseCmp
trueCmp:
        addi    $v0, $zero, 1                   # check = 1 (tìm thấy opcode)
        j       endFunc

#-----------------------------------------------
falseCmp:
#--- if (strlen(tmp) != j ){
#---    j == 0;
#---    check = 0;
#---    -> bỏ qua opcode
```

```
#--- }
        addi    $v0, $zero, 0                   # check = 0
        addi    $s2, $zero, 0           # j = 0
loopXspace:
        beq     $s4, $s5, Xspace        # if (opcode[i] == '-') Xspace
        addi    $s1, $s1, 1             # i++
        add     $t0, $a2, $s1                   # &opcode[i]
        lb      $s4, 0($t0)             # value opcode[i]
        j       loopXspace
Xspace:
        add     $v1, $v1, 1             # v1 = v1+1
        addi    $s1, $s1, 1             # i++
        j       loopCmp
#-------------------------------------------------
endCmp:
        addi    $v0, $zero, 0
endFunc:
        lw      $ra, 20($sp)
        lw      $s1, 16($sp)
        lw      $s2, 12($sp)
        lw      $s3, 8($sp)
        lw      $s4, 4($sp)
        lw      $s5, 0($sp)
        addi    $sp, $sp, 24
        jr      $ra
#-------------------------------------------------------
# Lấy khuôn dạng tương ứng với Opcode
# a0 = &operand - a1 = vị trí Opcode
# return tmp2 (trả về khuôn dạng code tương ứng)
#-------------------------------------------------------
getOperand:
        addi    $sp, $sp, -20
        sw      $s0, 16($sp)                    # i
        sw      $s1, 12($sp)                    # op[i]
        sw      $s2, 8($sp)             # 45
        sw      $s3, 4($sp)             # &tmp2
        sw      $s4, 0($sp)             # j
```

```
        addi    $t0, $zero, 4                   # mỗi khuôn chiếm 4 byte
        mul     $s0, $a1, $t0                   # i = count*4
        addi    $s2, $zero, 45                  # '-'
        la      $s3, tmp2               # s3 = &tmp
        add     $s4, $zero, $zero       # j
loopGet:
        add     $t0, $a0, $s0                   # $operand[i]
        lb      $s1, 0($t0)             # value of $operand[i]
        beq     $s1, $s2, endGet        # if (operand[i] == '-') endGet
        add     $t0, $s3, $s4                   # &tmp[i]
        sb      $s1, 0($t0)

        addi    $s0, $s0, 1             # i++
        addi    $s4, $s4, 1             # j++
        j       loopGet
endGet:
        add     $t0, $s3, $s4                   # &tmp[i]
        sb      $zero, 0($t0)

        lw      $s0, 16($sp)
        lw      $s1, 12($sp)
        lw      $s2, 8($sp)
        lw      $s3, 4($sp)
        lw      $s4, 0($sp)
        addi    $sp, $sp, 20

        jr $ra
#-----------------------------------------------------
# Kiểm tra chuỗi tmp có là số hay không
# a0 = &tmp, a1 = strlen(tmp)
# Nếu là số return 1, ngược lại return 0
#-----------------------------------------------------
checkNumber:
        add     $sp, $sp, -24
        sw      $ra, 20($sp)
        sw      $s4, 16($sp)                    # '+'
```

```
        sw      $s3, 12($sp)                    # '-'
        sw      $s0, 8($sp)
        sw      $s1, 4($sp)
        sw      $s2, 0($sp)             #
        add     $v0, $zero, 0
        add     $s0, $zero, $zero       #s0 = i

        beq     $a1, $zero, endCheckN           # return v0 = 0
checkFirstN:
        addi    $s3, $zero, 45                  # s3 = '-'
        addi    $s4, $zero, 43                  # s4 = '+'
        addi    $s2, $zero, 1           # s2 = 1
        add     $t0, $a0, $s0                   # &tmp[i]
        lb      $s1, 0($t0)             # value of tmp[i]
        #check - +  -> 123
checkMinus:
        bne     $s1, $s3, checkPlus             # if (tmp[i] != '-') checkPlus
        beq     $a1, $s2, endCheckN             # if (strlen(tmp) == 1) endCheckN
        # if (tmp[i] == '-')
        j       update
checkPlus:
        bne     $s1, $s4, _123                  # if (tmp[i] != '+') _123
        beq     $a1, $s2, endCheckN
        j       update

checkI:
        beq     $s0, $a1, trueN                 # if (i == strlen(tmp)) trueN
        add     $t0, $a0, $s0                   # t0 = $tmp[i]
        lb      $s1, 0($t0)             # s1 = tmp[i]
_123: #48 -> 57
        slti    $t0, $s1, 48            # if (s1 < 48) return 0
        bne     $t0, $zero, endCheckN
        slti    $t0, $s1, 58            # if (s1 >= 58) return 0
        beq     $t0, $zero, endCheckN
update:
        addi    $s0, $s0, 1             # i++
        j       checkI
```

```
trueN:
        addi    $v0, $v0, 1
endCheckN:
        lw      $ra, 20($sp)
        lw      $s4, 16($sp)
        lw      $s3, 12($sp)
        lw      $s0, 8($sp)
        lw      $s1, 4($sp)
        lw      $s2, 0($sp)
        add     $sp, $sp, 24
        jr      $ra
#-----------------------------------------------------
# Kiểm tra chuỗi tmp có là Label hay không, kí tự đầu tiên: _ | A -> _ | A | 1
# a0 =  &tmp, a1 = strlen(tmp)
# v0 0|1
#-----------------------------------------------------
checkLabel:
        add     $sp, $sp, -12
        sw      $ra, 8($sp)
        sw      $s1, 4($sp)
        sw      $s0, 0($sp)
        add     $v0, $zero, 0
        add     $s0, $zero, $zero           # s0 = i

        beq     $a1, $zero, endCheckL           # if (strlen(tmp) == 0) endCheckL

checkFirstChar:
        add     $t0, $a0, $s0                   # t0 -> &tmp[i]
        lb      $s1, 0($t0)                 # s1 = tmp[i]
        j       ABC

checkIL:
        slt     $t0, $s0, $a1               # if (i == strlen(tmp)) trueL
        beq     $t0, $zero, trueL
        add     $t0, $a0, $s0                   # else { t0 = &tmp[i]; s1 = tmp[i]
        lb      $s1, 0($t0)                 # }
_123L: #48 -> 57
```

```
        slti    $t0, $s1, 48
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 58
        beq     $t0, $zero, ABC
        addi    $s0, $s0, 1
        j       checkIL
ABC: #65 -> 90
        slti    $t0, $s1, 65            # if (tmp[i] < 65) endCheckL
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 91            # if (tmp[i] >= 91) _
        beq     $t0, $zero, _
        addi    $s0, $s0, 1             # i++
        j       checkIL
_:
        add     $t0, $zero, 95
        bne     $s1, $t0, abc           # if (tmp[i] != '_') abc
        addi    $s0, $s0, 1             # i++
        j       checkIL
abc: #97  -> 122
        slti    $t0, $s1, 97            # if (tmp[i] < 97) endCheckL
        bne     $t0, $zero, endCheckL
        slti    $t0, $s1, 123                   # if (tmp[i] >= 123) endCheckL
        beq     $t0, $zero, endCheckL
        addi    $s0, $s0, 1             # i++
        j       checkIL
trueL:
        addi    $v0, $v0, 1             #  check = 1
endCheckL:                              # return
        sw      $ra, 8($sp)
        lw      $s1, 4($sp)
        lw      $s0, 0($sp)
        add     $sp, $sp, 12
        jr      $ra
#-----------------------------------------------------------
# Kiểm tra chuỗi tmp có đúng cấu trúc offset base hay không
# a0 =  &tmp, a1 = strlen(tmp)
# v0 0|1
```

```
#-----------------------------------------------------
checkOffsetBase:
#0($s1) -> 0 $s1
        add     $sp, $sp, -28
        sw      $ra, 24($sp)
        sw      $s5, 20($sp)                    # độ dài xâu tmp
        sw      $s4, 16($sp)                    # ')'
        sw      $s3, 12($sp)                    # '('
        sw      $s2, 8($sp)             # check
        sw      $s1, 4($sp)             # tmp[i]
        sw      $s0, 0($sp)             # s0 = i
checkO:
        slti    $t0, $a1, 5             # có ít nhất 5 kí tự, vd: 0($s1)
        bne     $t0, $zero, falseCheck          # if (a1 < 5)falseCheck
        addi    $s3, $zero, 40                  # s3 = '('
        addi    $s4, $zero, 41                  # s4 = ')'
        add     $s0, $zero, $zero       # i = 0
        add     $s2, $zero, $zero       # boolean: check
        addi    $t2, $zero, 1           # t2 = 1
loopCheck:
        add     $t0, $a0, $s0                   # t0 = &tmp[i]
        lb      $s1, 0($t0)             # s1 = tmp[i]
        beq     $s1, $zero, endLoopO            # if (tmp[i] == 0)endLoop0
        beq     $s1, $s3, open_                 # if (tmp[i] == '(' ) open_
        beq     $s1, $s4, close_        # if (tmp[i] == ')' ) close_
        j       updateO
open_:
        bne     $s2, $zero, falseCheck          # if (check == 1) falseCheck
        addi    $s2, $s2, 1             # else  check = 1;
        addi    $t1, $zero, 32                  #       t1 = ' '
        sb      $t1, 0($t0)             #       tmp[i] = ' '
        j       updateO
close_:
        bne     $s2, $t2, falseCheck            # if (check == 0) falseCheck
        addi    $s2, $s2, 1             # else  check += 1;
        sb      $zero, 0($t0)                   #       tmp[i] == 0;
```

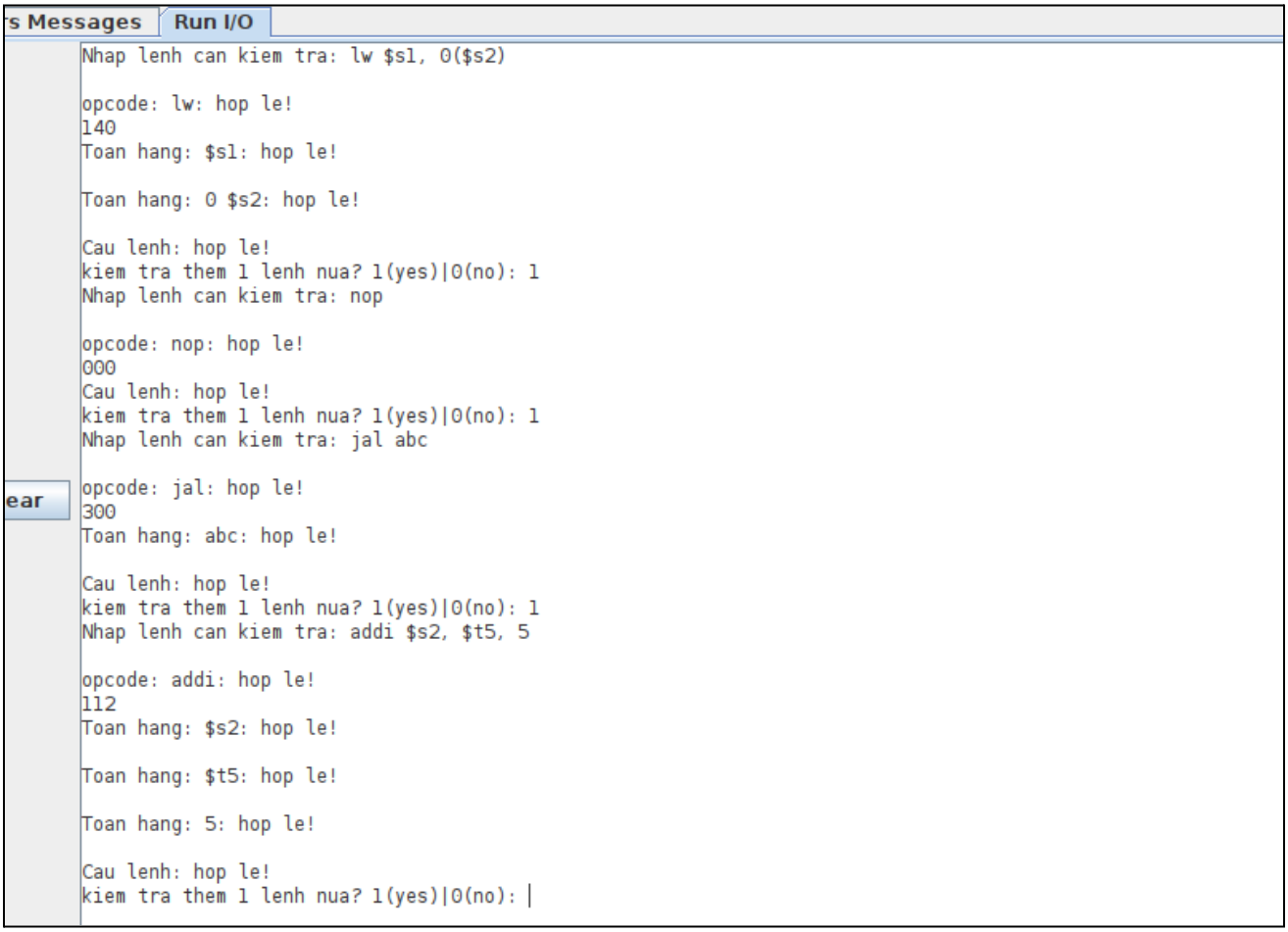
```
        addi    $s0, $s0, 1             #       i++;
        bne     $s0, $a1, falseCheck            #       if (i != strlen(tmp)) falseCheck

updateO:
        addi    $s0, $s0, 1             # i++
        j       loopCheck
endLoopO:
        addi    $t2, $t2, 1             # t2 = 2
        bne     $s2, $t2, falseCheck            # if(check != t2)falseCheck
#----
trueCheck:
        add     $s0, $zero, $zero       # i
# -------------- cut component -----------------
        addi    $sp, $sp, -8
        sw      $a0, 4($sp)
        sw      $a1, 0($sp)

        la      $a0, tmp                # a0 -> tmp
        add     $a1, $s0, $zero                 # a1 = 0
        la      $a2, tmp3               #
        jal     cutComponent
        add     $s0, $v0, $zero
        add     $s5, $v1, $zero         # strlen(tmp3)

        lw      $a0, 4($sp)
        lw      $a1, 0($sp)
        addi    $sp, $sp, 8

# -------------- check number -----------------
        addi    $sp, $sp, -8
        sw      $a0, 4($sp)
        sw      $a1, 0($sp)
        la      $a0, tmp3
        add     $a1, $s5, $zero
        jal     checkNumber
        add     $s2, $v0, $zero
        lw      $a0, 4($sp)
```

```
        lw      $a1, 0($sp)
        addi    $sp, $sp, 8

        beq     $s2, $zero, falseCheck
# -------------- cutComponent ----------------
        addi    $sp, $sp, -8
        sw      $a0, 4($sp)
        sw      $a1, 0($sp)

        la      $a0, tmp
        add     $a1, $s0, $zero
        la      $a2, tmp3
        jal     cutComponent
        add     $s0, $v0, $zero
        add     $s5, $v1, $zero #so ky tu co trong cutword

        lw      $a0, 4($sp)
        lw      $a1, 0($sp)
        addi    $sp, $sp, 8
# -------------- checkReg ------------------------
        addi    $sp, $sp, -12
        sw      $a0, 8($sp)
        sw      $a1, 4($sp)
        sw      $a2, 0($sp)

        la      $a0, tmp3
        add     $a1, $s5, $zero
        la      $a2, register
        #tra ve 0 -> error, 1 -> ok
        jal     compareOpcode
        add     $s2, $v0, $zero

        lw      $a0, 8($sp)
        lw      $a1, 4($sp)
        lw      $a2, 0($sp)
        addi    $sp, $sp, 12
```

```
        beq     $s2, $zero, falseCheck
        #->ket luan
        addi    $v0, $zero, 1
        j       endO
falseCheck:
        add     $v0, $zero, $zero               # v0 = 0;
        j       endO
endO:
        lw      $ra, 24($sp)
        lw      $s5, 20($sp)
        lw      $s4, 16($sp)
        lw      $s3, 12($sp)
        lw      $s2, 8($sp)
        lw      $s1, 4($sp)
        lw      $s0, 0($sp)
        add     $sp, $sp, 28
        jr      $ra
```

2. 5. Demo

- Các trường hợp nhập lệnh hợp lệ:



- Các trường hợp nhập lệnh không hợp lệ: