**Computer Architecture Lab**

**Final Project Report**

**Group 13:**

**Nguyen Chan Hung – 20215209**

**Do Tung Lam - 20215218**

Subject 4 – Nguyen Chan Hung:

Source code:

.eqv HEADING 0xffff8010

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.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

.eqv WHEREY 0xffff8040

.data

dce: .word 90,0,2000,180,1,7000,60,1,2631,30,1,1443,0,1,1000,-30,1,1443,-60,1,2631,90,0,8000,-100,1,2682,-170,1,2066,170,1,2066,100,1,2682,90,0,2000,0,1,5000,90,1,3000,180,0,2500,-90,1,3000,180,0,2500,90,1,3000

nch: .word 180,1,5000,0,0,5000,149,1,5700,0,1,5000,90,0,6000,-100,1,2682,-170,1,2066,170,1,2066,100,1,2682,90,0,2000,0,1,5000,180,0,2500,90,1,3000,0,1,2500,180,1,5000

number: .word 180,0,5000,0,1,5000,90,0,2000,90,1,3000,180,1,2500,-90,1,3000,180,1,2500,90,1,3000,90,0,2000,90,1,3000,0,1,5000,-90,1,3000,180,0,2500,90,1,3000

.text

main:

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x80 # bit 7 = 1 to enable

sb $t3, 0($t1)

Loop:

sleep: addi $v0,$zero,32

li $a0,300 # sleep 300 ms

syscall

nop # WARNING: nop is mandatory here.

b Loop # Loop

end\_main:

li $v0,10

syscall

.ktext 0x80000180

li $s5,18

li $s6,14

li $s7,13

jal GO

jal UNTRACK

li $a0,180

jal ROTATE

li $v0,32

li $a0,2000

syscall

li $a0,90

jal ROTATE

li $v0,32

li $a0,2000

syscall

jal STOP

IntSR: addi $sp,$sp,4 # Save $at because we may change it later

sw $at,0($sp)

addi $sp,$sp,4 # Save $sp because we may change it later

sw $v0,0($sp)

addi $sp,$sp,4 # Save $a0 because we may change it later

sw $a0,0($sp)

addi $sp,$sp,4 # Save $t1 because we may change it later

sw $t1,0($sp)

addi $sp,$sp,4 # Save $t3 because we may change it later

sw $t3,0($sp)

row1:li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x81 # check row 1

sb $t3, 0($t1) # must reassign expected row

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

beq $a0,0x00000011,cut\_DCE\_lbl

row2: li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x82 # check row 1

sb $t3, 0($t1) # must reassign expected row

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

beq $a0,0x00000012,cut\_NCH\_lbl

row3: li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t3, 0x84 # check row 1

sb $t3, 0($t1) # must reassign expected row

li $t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

beq $a0,0x00000014,cut\_123\_lbl

j next\_pc

cut\_DCE\_lbl:

add $a0,$s5,$zero

la $a1,dce

jal cut

j next\_pc

cut\_NCH\_lbl:

add $a0,$s6,$zero

la $a1,nch

jal cut

j next\_pc

cut\_123\_lbl:

add $a0,$s7,$zero

la $a1,number

jal cut

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:lw $t3, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $t1, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $a0, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $v0, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $at, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

return: eret # Return from exception

GO: li $at, MOVING # change MOVING port

addi $k0, $zero,1 # to logic 1,

sb $k0, 0($at) # to start running

jr $ra

STOP: li $at, MOVING # change MOVING port to 0

sb $zero, 0($at) # to stop

jr $ra

TRACK: li $at, LEAVETRACK # change LEAVETRACK port

addi $k0, $zero,1 # to logic 1,

sb $k0, 0($at) # to start tracking

jr $ra

UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0

sb $zero, 0($at) # to stop drawing tail

jr $ra

ROTATE: li $at, HEADING # change HEADING port

sw $a0, 0($at) # to rotate robot

jr $ra

cut:

addi $s4,$a0,0 # total no. instructions

li $t0,0 # number of instructions

li $t8,0 # index of dce

la $t1,($a1) # $t1 = addr(dce[0])

jal GO

loop\_cut:

bgt $t0,$s4,end\_loop\_cut

mul $t2,$t8,4

add $t3,$t1,$t2 # $t3 = addr(dce[i])

lw $s0,($t3) # $s0 = ANGLE

addi $t8,$t8,1

mul $t2,$t8,4

add $t3,$t1,$t2 # $t3 = addr(dce[i+1])

lw $s1,($t3) # #s1 = CUT/UNCUT

addi $t8,$t8,1

mul $t2,$t8,4

add $t3,$t1,$t2 # $t3 = addr(dce[i+2])

lw $s2,($t3) # #s2 = DURATION

add $a0,$s0,$zero

jal ROTATE

beq $s1,1,toggle\_track

j toggle\_untrack

toggle\_track:

jal TRACK

j sleep\_command

toggle\_untrack:

jal UNTRACK

sleep\_command:

li $v0,32

add $a0,$s2,$zero

syscall

continue:

addi $t0,$t0,1

addi $t8,$t8,1

jal UNTRACK

j loop\_cut

end\_loop\_cut:

jal STOP

jr $ra

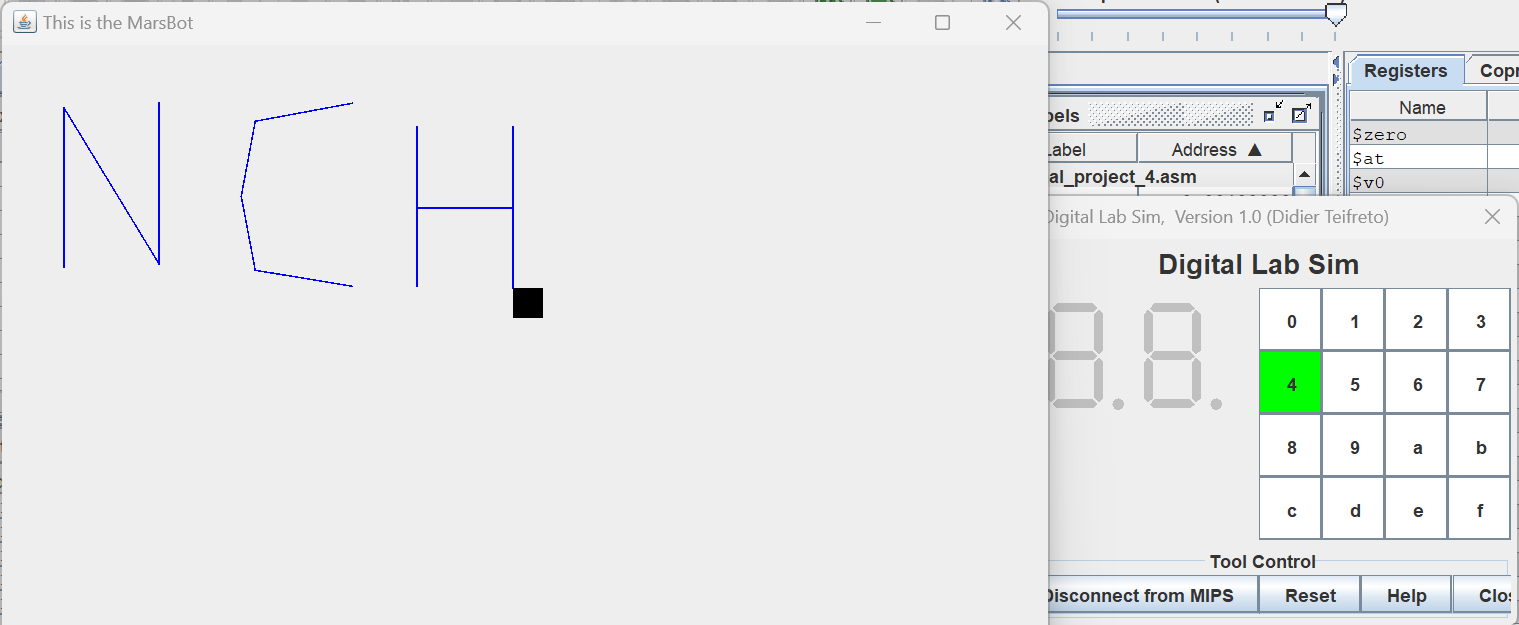
Results:

* Postscript 1: DCE

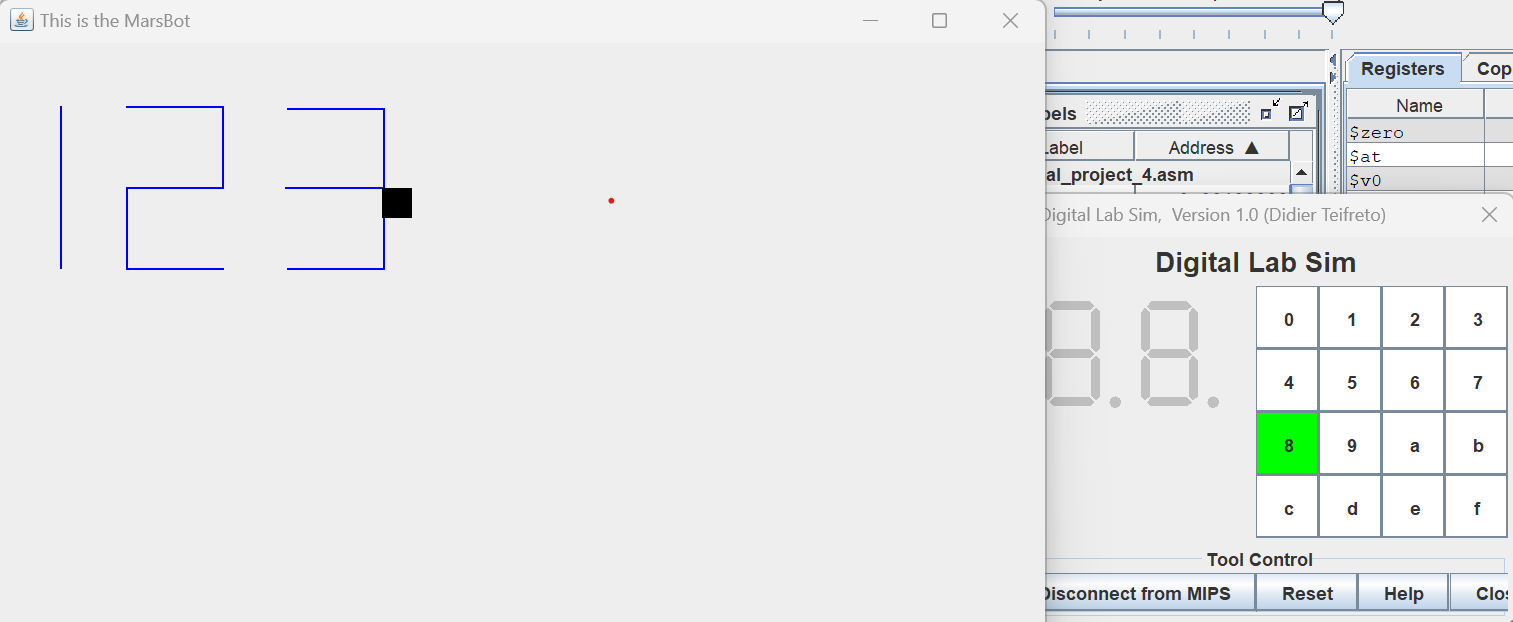
A screenshot of a computer

Description automatically generated

* Postscript 2: NCH



* Postscript 3: 123



Subject 10 – Do Tung Lam:

Source code:

# Check not pressing. If all rows return $a0 == 0, update {code} = 0.

# If {code} == 0, run `back\_to\_polling`.

# Check pressing. If pressed && $a0 != $s0, run `update\_code`.

# Run `handle\_code` to get key's value && switch mode.

# Mode 1: If old {operator} is "=", remove old info. Update {operand}, output new {operand}.

# Mode 2: If old {operator} doesn't change, do nothing.

# Else, update {operator}, update {answer} = {operand}, output old {operand}, remove old {operand}.

# Mode 3: Update {answer} = {answer} {operator} {operand}, update {operator}, update {operand} = {answer}, output new {answer}.

.eqv SEVENSEG\_LEFT 0xFFFF0011

.eqv SEVENSEG\_RIGHT 0xFFFF0010

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv CODE\_0 0x11

.eqv CODE\_1 0x21

.eqv CODE\_2 0x41

.eqv CODE\_3 0x81

.eqv CODE\_4 0x12

.eqv CODE\_5 0x22

.eqv CODE\_6 0x42

.eqv CODE\_7 0x82

.eqv CODE\_8 0x14

.eqv CODE\_9 0x24

.eqv CODE\_ADD 0x44

.eqv CODE\_SUB 0x84

.eqv CODE\_MUL 0x18

.eqv CODE\_DIV 0x28

.eqv CODE\_MOD 0x48

.eqv CODE\_EQL 0x88

.data

NUMS: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F

.text

main:

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

init:

li $s0, 0 # code

li $s1, 0 # key's value (0 -> 15)

li $s2, 0 # mode (1, 2, 3)

li $s3, 0 # operand

li $s4, 0 # operator

li $s5, 0 # answer

polling:

check1:

li $t3, 0x01 # Check 0, 1, 2, 3

sb $t3, 0($t1) # Must reassign expected row

lbu $a0, 0($t2) # Read scan code of key button

beq $a0, 0, check2

bne $a0, $s0, update\_code

beq $a0, $s0, back\_to\_polling

check2:

li $t3, 0x02 # Check 4, 5, 6, 7

sb $t3, 0($t1) # Must reassign expected row

lbu $a0, 0($t2) # Read scan code of key button

beq $a0, 0, check3

bne $a0, $s0, update\_code

beq $a0, $s0, back\_to\_polling

check3:

li $t3, 0x04 # Check 8, 9, a, b

sb $t3, 0($t1) # Must reassign expected row

lbu $a0, 0($t2) # Read scan code of key button

beq $a0, 0, check4

bne $a0, $s0, update\_code

beq $a0, $s0, back\_to\_polling

check4:

li $t3, 0x08 # Check c, d, e, f

sb $t3, 0($t1) # Must reassign expected row

lbu $a0, 0($t2) # Read scan code of key button

beq $a0, 0, update\_code

bne $a0, $s0, update\_code

beq $a0, $s0, back\_to\_polling

update\_code:

add $s0, $zero, $a0

beq $s0, 0, back\_to\_polling

beq $s0, CODE\_0, handle\_code\_0

beq $s0, CODE\_1, handle\_code\_1

beq $s0, CODE\_2, handle\_code\_2

beq $s0, CODE\_3, handle\_code\_3

beq $s0, CODE\_4, handle\_code\_4

beq $s0, CODE\_5, handle\_code\_5

beq $s0, CODE\_6, handle\_code\_6

beq $s0, CODE\_7, handle\_code\_7

beq $s0, CODE\_8, handle\_code\_8

beq $s0, CODE\_9, handle\_code\_9

beq $s0, CODE\_ADD, handle\_code\_add

beq $s0, CODE\_SUB, handle\_code\_sub

beq $s0, CODE\_MUL, handle\_code\_mul

beq $s0, CODE\_DIV, handle\_code\_div

beq $s0, CODE\_MOD, handle\_code\_mod

beq $s0, CODE\_EQL, handle\_code\_eql

handle\_code\_0:

li $s1, 0

li $s2, 1

j after\_update

handle\_code\_1:

li $s1, 1

li $s2, 1

j after\_update

handle\_code\_2:

li $s1, 2

li $s2, 1

j after\_update

handle\_code\_3:

li $s1, 3

li $s2, 1

j after\_update

handle\_code\_4:

li $s1, 4

li $s2, 1

j after\_update

handle\_code\_5:

li $s1, 5

li $s2, 1

j after\_update

handle\_code\_6:

li $s1, 6

li $s2, 1

j after\_update

handle\_code\_7:

li $s1, 7

li $s2, 1

j after\_update

handle\_code\_8:

li $s1, 8

li $s2, 1

j after\_update

handle\_code\_9:

li $s1, 9

li $s2, 1

j after\_update

handle\_code\_add:

li $s1, 10

li $s2, 2

j after\_update

handle\_code\_sub:

li $s1, 11

li $s2, 2

j after\_update

handle\_code\_mul:

li $s1, 12

li $s2, 2

j after\_update

handle\_code\_div:

li $s1, 13

li $s2, 2

j after\_update

handle\_code\_mod:

li $s1, 14

li $s2, 2

j after\_update

handle\_code\_eql:

li $s1, 15

li $s2, 3

j after\_update

after\_update:

beq $s2, 1, mode1

beq $s2, 2, mode2

beq $s2, 3, mode3

# Mode 1: If old {operator} is "=", remove old info. Update {operand}, output new {operand}.

mode1:

beq $s4, 15, mode1\_1

j mode1\_2

mode1\_1:

li $s3, 0 # Reset

li $s4, 0 # Reset

li $s5, 0 # Reset

mode1\_2:

mul $s3, $s3, 10

add $s3, $s3, $s1

add $a0, $zero, $s3 # Output {operand}

li $v0, 1

syscall

jal display # Output {operand} to SEVENSEG

li $a0, '|' # Output delimiter

li $v0, 11

syscall

j sleep

# Mode 2: If old {operator} doesn't change, do nothing.

# Else, update {operator}, update {answer} = {operand}, output old {operand}, remove old {operand}.

mode2:

beq $s4, $s1, sleep

add $s4, $zero, $s1 # Update {operator}

add $s5, $zero, $s3 # Update {answer} = {operand}

add $a0, $zero, $s3 # Output {operand}

li $v0, 1

syscall

jal display # Output {operand} to SEVENSEG

li $a0, '|' # Output delimiter

li $v0, 11

syscall

li $s3, 0 # Remove {operand}

j sleep

# Mode 3: Update {answer} = {answer} {operator} {operand}, update {operator}, update {operand} = {answer}, output new {answer}.

mode3:

beq $s4, 10, calc\_add

beq $s4, 11, calc\_sub

beq $s4, 12, calc\_mul

beq $s4, 13, calc\_div

beq $s4, 14, calc\_mod

beq $s4, 15, calc\_eql

calc\_add:

add $s5, $s5, $s3

j after\_calc

calc\_sub:

sub $s5, $s5, $s3

j after\_calc

calc\_mul:

mul $s5, $s5, $s3

j after\_calc

calc\_div:

div $s5, $s3

mflo $s5

j after\_calc

calc\_mod:

div $s5, $s3

mfhi $s5

j after\_calc

calc\_eql:

j after\_calc

after\_calc:

li $s4, 15 # Update {operator} = "="

add $s3, $zero, $s5 # Update {operand} = {answer}

add $a0, $zero, $s5 # Output {answer}

li $v0, 1

syscall

jal display # Output {answer} to SEVENSEG

li $a0, '|' # Output delimiter

li $v0, 11

syscall

j sleep

sleep:

li $a0, 1000 # Sleep 1000ms

li $v0, 32

syscall

back\_to\_polling:

j polling # Continue polling

# function display:

# param[in] $a0 interger to display

display:

display\_save:

add $sp, $sp, -24 # Expand stack

sw $ra, 20($sp) # Save

sw $s0, 16($sp) # Save

sw $a0, 12($sp) # Save

sw $a1, 08($sp) # Save

sw $t0, 04($sp) # Save

sw $t1, 00($sp) # Save

display\_body:

li $t0, 10

add $t1, $zero, $a0

div $t1, $t0

mfhi $a0

li $a1, SEVENSEG\_RIGHT

jal draw

mflo $t1

div $t1, $t0

mfhi $a0

li $a1, SEVENSEG\_LEFT

jal draw

display\_load:

lw $t1, 00($sp) # Load

lw $t0, 04($sp) # Load

lw $a1, 08($sp) # Load

lw $a0, 12($sp) # Load

lw $s0, 16($sp) # Load

lw $ra, 20($sp) # Load

add $sp, $sp, +24 # Shrink stack

jr $ra

# function draw:

# param[in] $a0 number to display

# param[in] $a1 SEVENSEG to display

draw:

draw\_save:

add $sp, $sp, -12 # Expand stack

sw $ra, 08($sp) # Save

sw $t0, 04($sp) # Save

sw $t1, 00($sp) # Save

draw\_body:

la $t0, NUMS

sll $t1, $a0, 2 # i \* 4

add $t0, $t0, $t1 # Address(NUMS[i])

lw $t0, 0($t0) # NUMS[i]

sb $t0, 0($a1) # Draw number

draw\_load:

lw $t1, 00($sp) # Load

lw $t0, 04($sp) # Load

lw $ra, 08($sp) # Load

add $sp, $sp, +12 # Shrink stack

jr $ra

Results:

* **Press 1, 2, 3**

**Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Song song

Mô tả được tạo tự động**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

* **Press ‘a’ to add. Press 56.**

**A picture containing table

Description automatically generated A picture containing table

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence**

* **Press f to calculate. Here, 123 + 56 = 179.A screenshot of a computer

  Description automatically generated with medium confidence**
* **Keep calculating using previous answer. Press d to divide. Press 2. Press f to calculate. Here, 179 / 2 = 89 (mod 1). So the answer is 89.**

**A screenshot of a computer

Description automatically generated with medium confidence**