Họ và tên : Lê Ngọc Anh Quân

Mã số sinh viên : 20176852

Báo cáo cuối kì môn thực hành Kiến trúc máy tính

Đề bài :

Project10.Write a program that gets an integer i from the user and creates the table shown below on the screen (example inputs provided). Subroutines are required for power, square, and hexadecimal (in 32 bit arithmetic, attend to overflowed results). Hint: Hexadecimal can be done with shifts and masks because the size is 32 bits.

Bài làm :

Source code :

# Mid Term Project 10

.data

message: .asciiz "Enter an integer"

table: .asciiz "i Power(2, i) Square(i) Hexadecimal(i)\n"

space1: .asciiz " "

space2: .asciiz " "

newLine: .asciiz "\n"

errMessage1: .asciiz "the input must be integer\n"

errMessage2: .asciiz "the input must not be null\n"

errMessage3: .asciiz "the input must greater than -31 , less than 30"

afterErr: .asciiz "please re-input an integer\n"

hex: .space 10

.text

li $v0, 4 # print table header

la $a0, table

syscall

main:

li $v0, 51

la $a0, message

syscall

beq $a1, -1,showErrorMessage1 # $a1 status value (if input is not integer =>show err)

beq $a1, -3,showErrorMessage2 # $a1 status value (if input is null => show err)

beq $a1, -2,exit # exit if user click cancel

blt $a0, -31, showErrorMessage3 # check if i < -31 then input again

bgt $a0, 30, showErrorMessage3 # check if i > 30 then input again

add $s0, $zero, $a0 # $s0 = i

jal power # call function power(2, i)

add $s1, $zero, $s7 #$s1 = $s7= power(2,i)

jal square # call function square(i)

add $s2, $zero, $s7 #s2=$s7=square(i)

jal decimalToHexa # call function decimalToHexa(i)

print:

li $v0, 1 # print i

add $a0, $zero, $s0

syscall

li $v0, 4

la $a0, space1

syscall

#above function to print i Power(2,i) square(i) Hexadecimal(i)

checkPowerResult:

blt $s0, $zero, printFloat

printInt :

li $v0, 1 # print integer power(2, i)

add $a0, $zero, $s1

syscall

li $v0, 4

la $a0, space2

syscall

j printSquare

printFloat: #use printFloat to print power(2,i)when i<0

li $v0, 2 # print float power(2, i)

syscall #because if i<0 then 2^i will be 1/2^(-i)

li $v0, 4

la $a0, space2

syscall

printSquare:

li $v0, 1 # print square(i)

add $a0, $zero, $s2

syscall

li $v0, 4

la $a0, space2

syscall

printHex:

li $v0, 4 # print decimalToHexa(i)

la $a0, hex

syscall

li $v0, 4

la $a0, newLine

syscall

jal clearHex

j main

exit:

li $v0, 10

syscall

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

# @function power

# @param[in] $s0 User entered integer i

# @return $s7 i >= 0 power(2, i)

# @return $f12 i < 0 power(2, i)

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

power:

add $s7,$zero,1

li $a0, 0 # $a0 = 0 =x

add $a1, $zero, $s0 # $a1 = $s0

bge $s0, $zero, powerLoop # $s0 >= 0

sub $a1, $zero, $s0 # $a1 = -$s0

powerLoop:

beq $a0, $a1, powerDone # if x ==i then break loop

sll $s7, $s7, 1 # $s7 = $s7 \* 2

addi $a0, $a0, 1 # $a0 += 1

j powerLoop

powerDone:

bge $s0, $zero, done # if $s0 >= 0 then done else caculate 1/2^(-x)

li $t0, 1

mtc1 $t0, $f0 # $f0 = 1.0

cvt.s.w $f0, $f0

mtc1 $s7, $f2 # $f2 = (float)$s7

cvt.s.w $f2, $f2

div.s $f12, $f0, $f2 # $f12 = $f0/$f2

done:

jr $ra

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

# function square

# User entered integer i

# return $s7 square(i)

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

square:

mul $s7, $s0, $s0 # $v0 = i \* i

jr $ra

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

# function decimalToHexa

# User entered integer i

# return none

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

decimalToHexa:

la $a0, hex # load hex to $a0

add $a1, $zero, $s0 # $a1 = i

li $t1, 48 # add 0x to hex string

sb $t1, 0($a0) #

addi $a0, $a0, 1 #

li $t1, 120 #

sb $t1, 0($a0) #

addi $a0, $a0, 1 #

beqz $s0, hexZero # $s0 = 0 => hex = "0x0"

li $t0, 8 # counter loop through 32 bits

li $t2, 0 # $t2 = 0 this is condition to write the first 0 , next 0 won't write

hexLoop:

beqz $t0, hexDone # counter == 0 => return function

andi $t1, $a1, 0xf0000000 # to get $t1=i mod 16 , we can use ($a1 and 0xf)

srl $t1, $t1, 28 # move 4 bits to most right

beq $t1, $t2, continue # $t1 == $t2 (= 0) => ignore 0

ble $t1, 9, numberChangeString # use ASCII code to write , if $t1<9 then plus 48 else plus 55

addi $t1, $t1, 55 # get [A-F] if $t1=10 , then plus 55 to get A

j writeHex

numberChangeString:

addi $t1, $t1, 48 # [1-9]

writeHex:

addi $t2, $t2, -1 # remove flag ignore 0

sb $t1, 0($a0) # write ASCII Code to hex string

addi $a0, $a0, 1

continue:

sll $a1, $a1, 4 # shift left to get next 4 bits

addi $t0, $t0, -1 # counter -= 1

j hexLoop

hexZero:

li $t1, 48 # add 0x to hex string

sb $t1, 0($a0)

hexDone:

jr $ra

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

# function clearHex

# return none

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

clearHex:

la $a0, hex # load hex to $a0

li $a1, 0

clearLoop:

beq $a1, 10, doneClear

sb $zero, 0($a0)

addi $a0, $a0, 1

addi $a1, $a1, 1

j clearLoop

doneClear:

jr $ra

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

# @function show Error message

# @return none

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

showErrorMessage1:

li $v0,59

la $a0,errMessage1

la $a1,afterErr

syscall

j main

showErrorMessage2:

li $v0,59

la $a0,errMessage2

la $a1,afterErr

syscall

j main

showErrorMessage3:

li $v0,59

la $a0,errMessage3

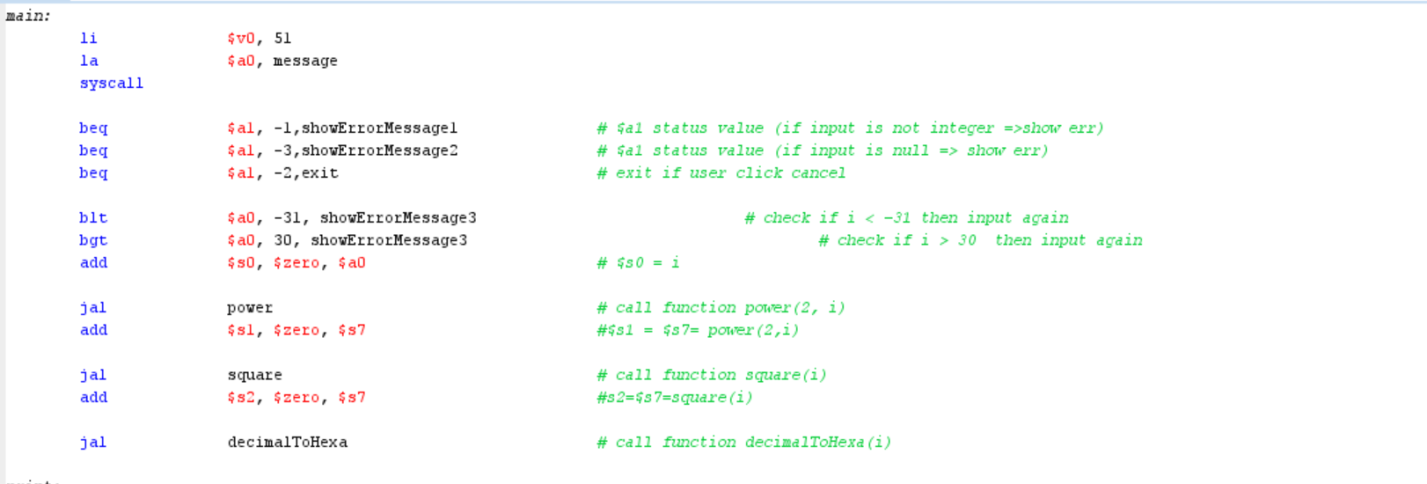
la $a1,afterErr

syscall

j main

Giải thích code :

Hàm main :

Hàm này để tạo 1 cửa sổ dialog để ta nhập số tự nhiên .

Các trường hợp có thể xảy ra khi nhập đó là :

-Nhập kí tự đặc biết không phải chữ số ( a-z , !@#!@#!....)

-Bấm enter nhưng kí tự nhập vào là rỗng

-Bấm cancel

Xử lí các trường hợp có thể xảy ra trên :

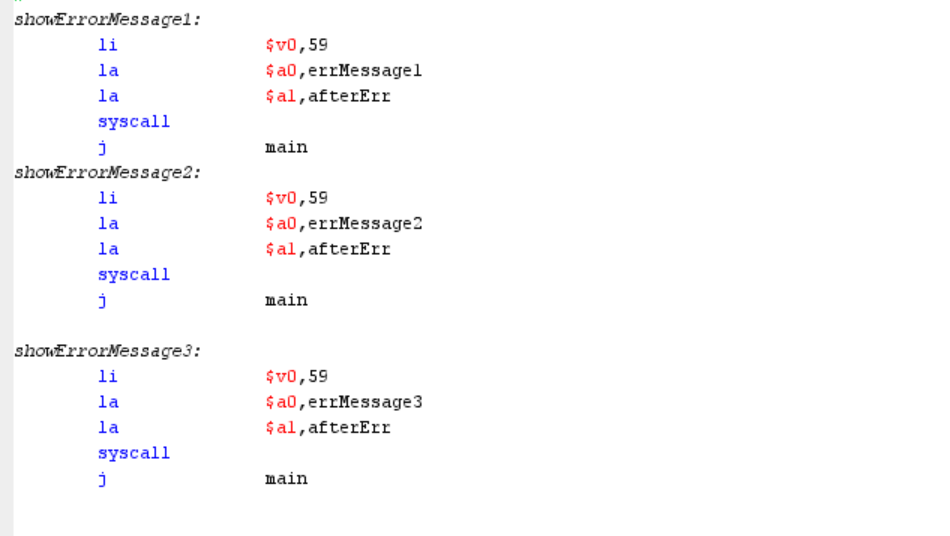
-Tạo 2 cửa sổ ShowDialog với errMessage1 là : "the input must be integer\n" , errMessage2 là : "the input must not be null\n"

-Khi bấm Cancel sẽ nhảy đến function Exit

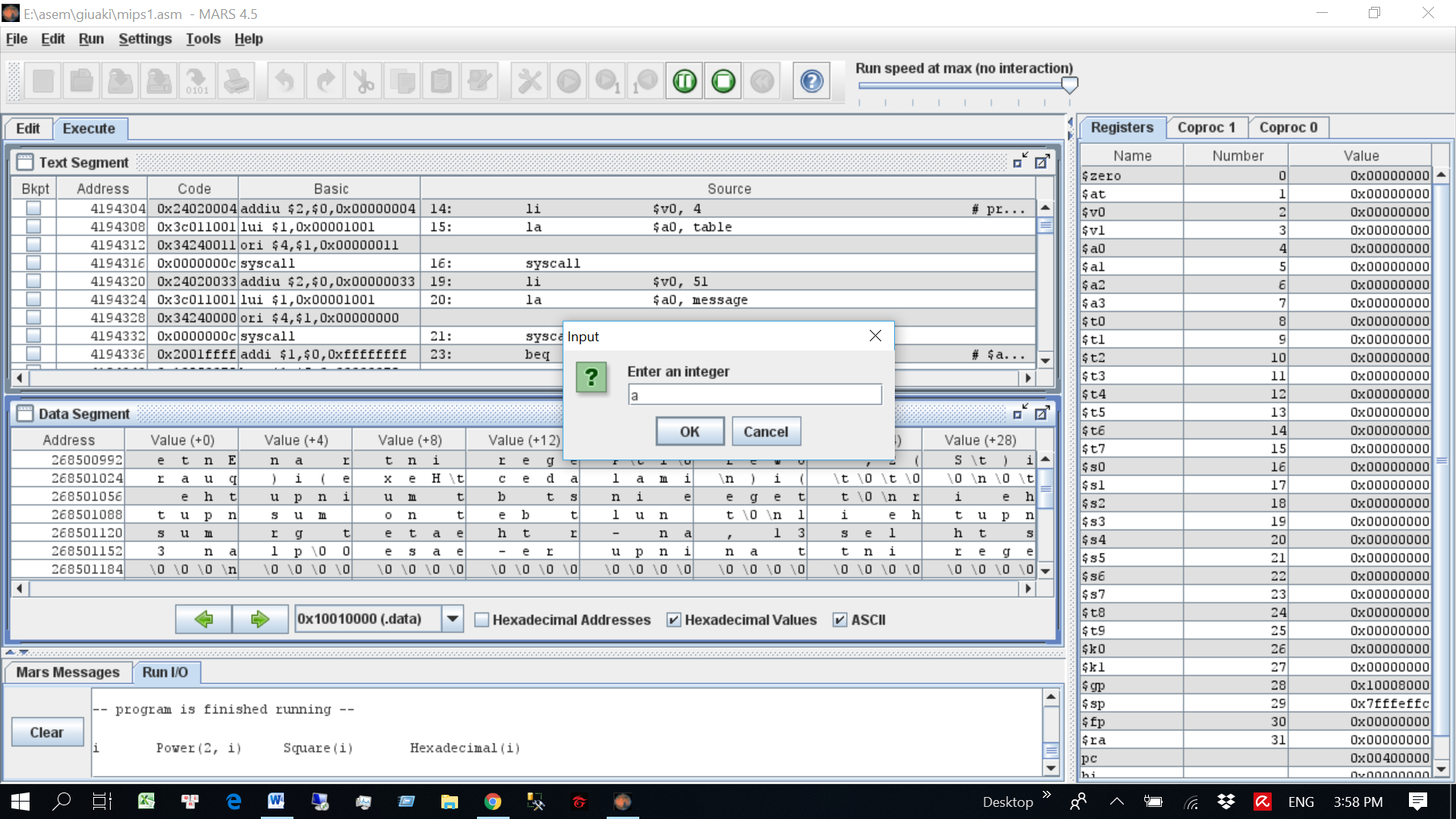
Ở bài này , số nhập vào phải nằm trong khoảng từ -31 đến 30 , vì nếu lớn hơn sẽ bị tràn thanh ghi khi tính toán 2^i.

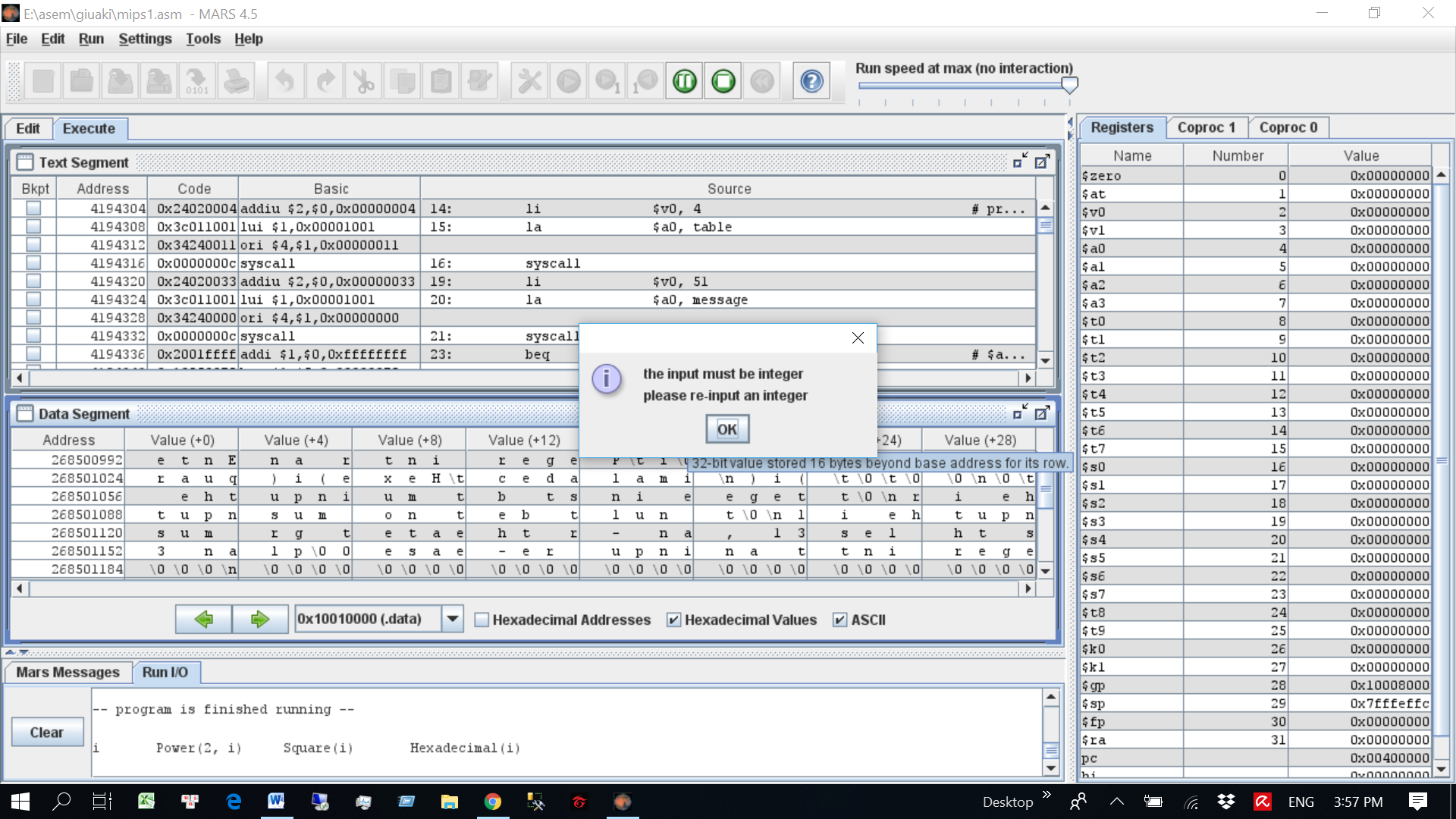
Xử lí : Tạo cửa số showMessage3 với errMessage3 là : “the input must greater than -31 , less than 30”

Hàm showErrorMessage 1+2+3:



* Kết quả :





3 hàm chính thực hiện yêu cầu bài toán là power , square,hexadecimal

Hàm power thực hiện chức năng tính 2^i

Hàm square thực hiện chức năng i^2

Hàm decimalToHexa thực hiện chuyển từ dạng thập phân sang dạng hexa

Chi tiết hàm power :

power:

add $s7,$zero,1

li $a0, 0 # $a0 = 0 =x

add $a1, $zero, $s0 # $a1 = $s0

bge $s0, $zero, powerLoop # $s0 >= 0

sub $a1, $zero, $s0 # $a1 = -$s0

powerLoop:

beq $a0, $a1, powerDone # if x ==i then break loop

sll $s7, $s7, 1 # $s7 = $s7 \* 2

addi $a0, $a0, 1 # $a0 += 1

j powerLoop

powerDone:

bge $s0, $zero, done # if $s0 >= 0 then done else caculate 1/2^(-x)

li $t0, 1

mtc1 $t0, $f0 # $f0 = 1.0

cvt.s.w $f0, $f0

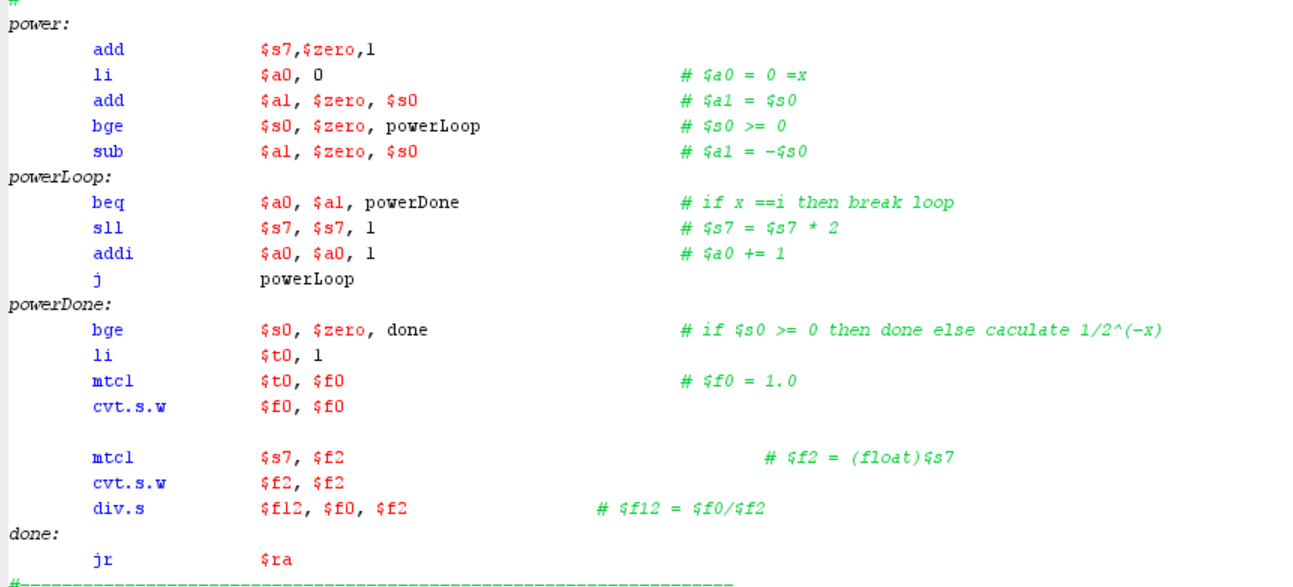
mtc1 $s7, $f2 # $f2 = (float)$s7

cvt.s.w $f2, $f2

div.s $f12, $f0, $f2 # $f12 = $f0/$f2

done:

jr $ra



Các trường hợp xảy ra với hàm power :

Nếu số nhập vào là số dương , ta tính toán hàm như bình thường là 2^i.

Nếu số nhập vào là số âm , ta tính toán 2^(-i) sau đó lấy 1 chia cho kết quả vừa tính được

Từ 2 trường hợp trên sẽ dẫn đến 2 cách in khác nhau , nó sẽ được nói rõ hơn ở phần bên dưới .

power:

add $s7,$zero,1

li $a0, 0 # $a0 = 0 =x

add $a1, $zero, $s0 # $a1 = $s0

bge $s0, $zero, powerLoop # $s0 >= 0

sub $a1, $zero, $s0 # $a1 = -$s0

Gán $a1=$s0 , nếu $s0 >= 0 ta thực hiện trực tiếp loop để tính 2^I , nếu $s0<0 ta lấy $a1=-$s0 và thực hiện loop với $a1.

powerLoop:

beq $a0, $a1, powerDone # if x ==i then break loop

sll $s7, $s7, 1 # $s7 = $s7 \* 2

addi $a0, $a0, 1 # $a0 += 1

j powerLoop

Gán $a0=0 ta thực hiện biến loop cho đến khi $a0=1. Kết quả được lưu vào $s7.

powerDone:

bge $s0, $zero, done # if $s0 >= 0 then done else caculate 1/2^(-x)

li $t0, 1

mtc1 $t0, $f0 # $f0 = 1.0

cvt.s.w $f0, $f0

mtc1 $s7, $f2 # $f2 = (float)$s7

cvt.s.w $f2, $f2

div.s $f12, $f0, $f2 # $f12 = $f0/$f2

done:

jr $ra

Nếu $s0 >0 thì ta hoàn thành việc tính power(2,i) và về main . Ngược lại , ta gán $f2=float$s7 và tính $f12 =1/$f2

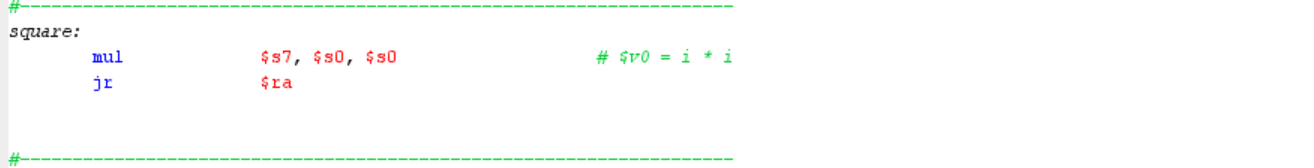
* Hàm power trả về $s7 nếu như $s0>0 và trả về $f12 nếu $s0<0.

Sau đó thực hiện hàm SQUARE để tính i^2.

square:

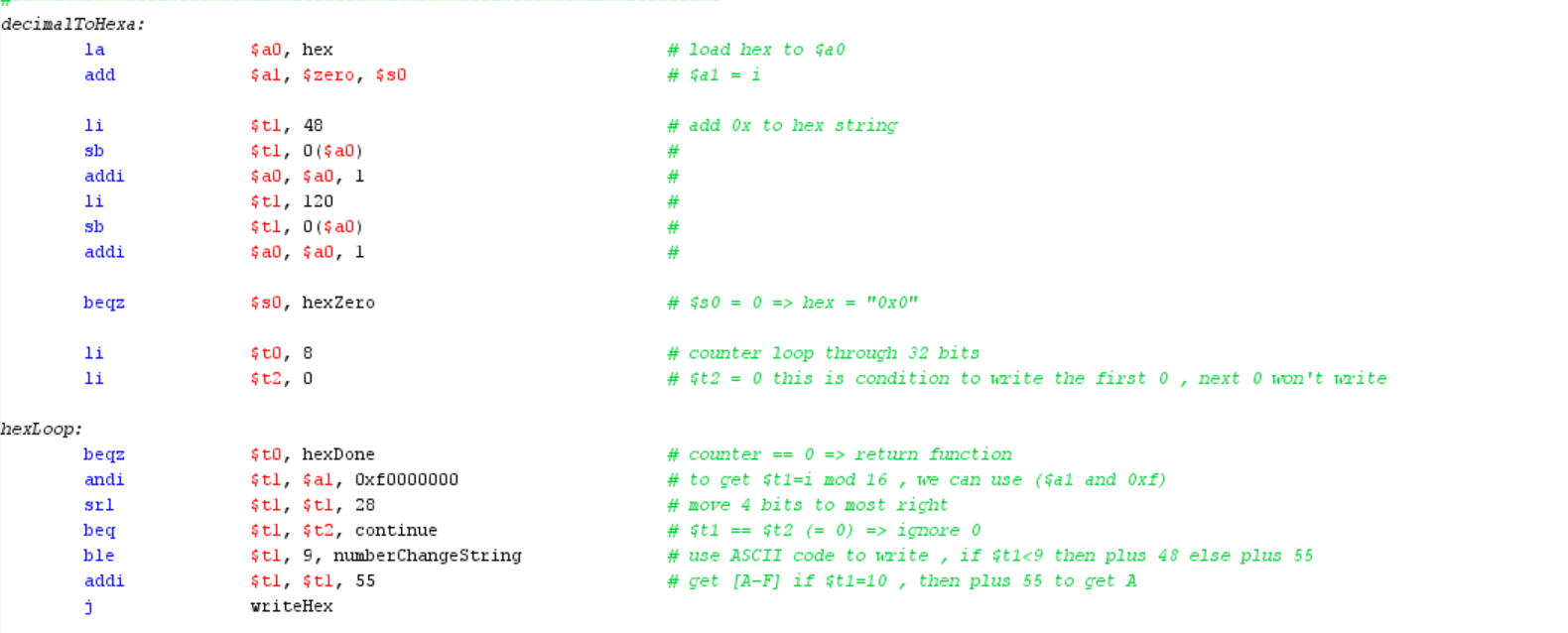
mul $s7, $s0, $s0 # $v0 = i \* i

jr $ra



* Hàm này chỉ sử dụng mul để tính $s7=$s0 \* $s0

Hàm tiếp theo :

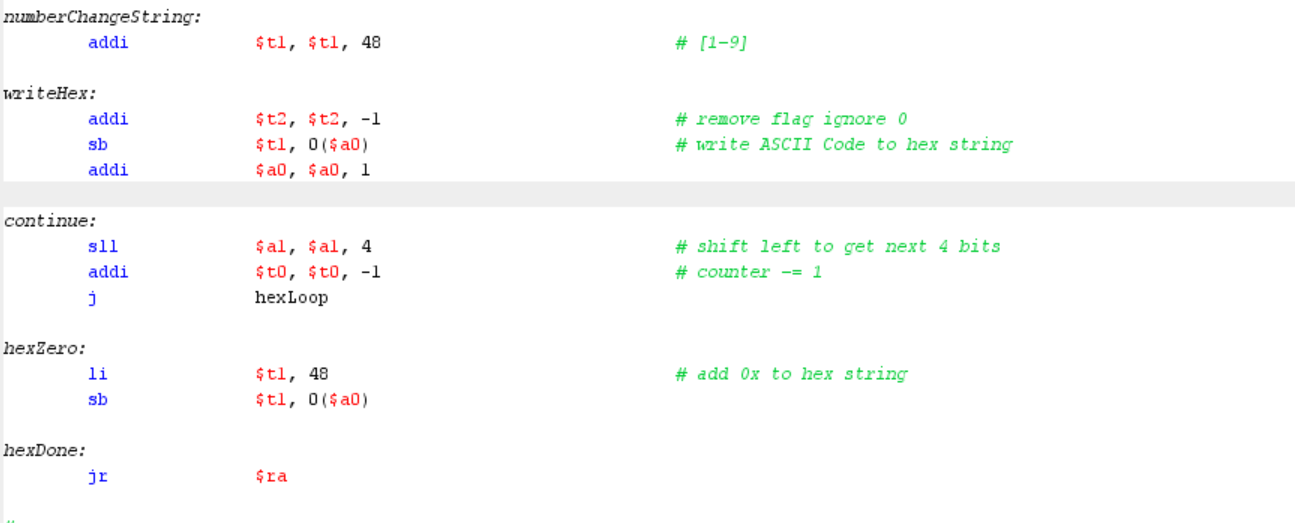


Đầu tiên ta đưng 0x vào trong hex ( hex được khai báo ở đầu là 1 string để in ra kết quả ).

Để lấy chuyển được ta phải tìm được phần dư khi chia cho 16 . Ở đây ta sử dụng thuật toán như sau .

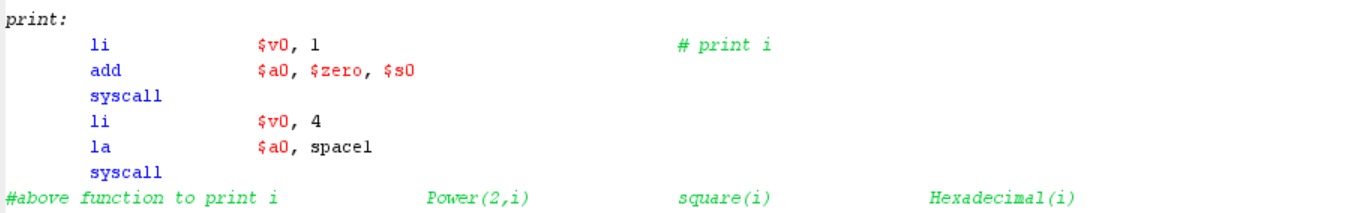
Sử dụng toán tử AND với 0xf0000000 sau đó để nó là 4bits ngoài cùng bên phải của thanh ghi $t1.

Từ đó lấy được giá trị thanh ghi $t1 . nếu giá trị đó <10 thì ta cộng với 48 , nếu lớn hơn 10 thì ta cộng 55 ( do 48 là số 0 trong bảng mã ascii , và 65 là chữ cái A trong bảng mã ) rồi lưu vào hex . Ở đây để giống với ví dụ 3 -> 0x3 nên ta thực hiện phương pháp như trên để loại bỏ các số 0 để lưu vào hex .

Ở đây với ví dụ số nhập vào là 1 thì chỉ ở lần lặp cuối ta mới lấy được số 1 và lưu vào hex.

Hàm tiếp theo:

print

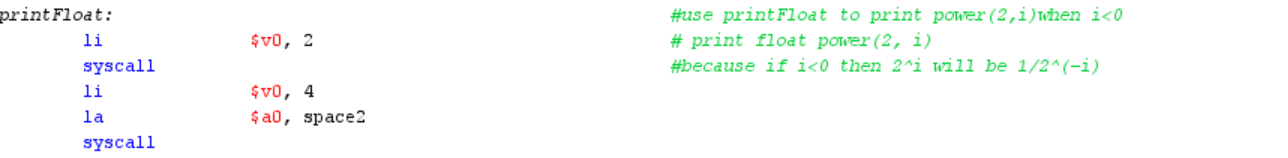
checkPowerResult: để check xem nếu $s0<0 thì chạy hàm printFloat , nếu k thì chạy printInt



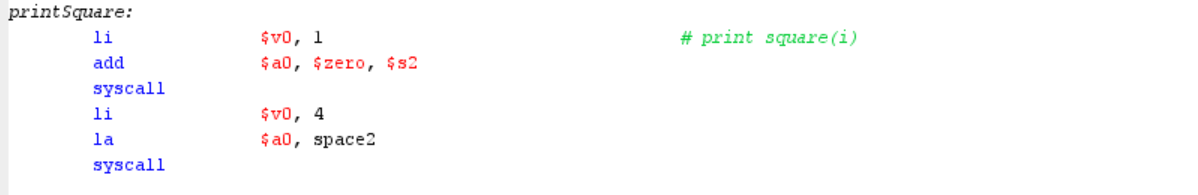
printInt : để in ra power(2,i) với I > 0



printFloat: để in ra power(2,i) với I < 0



printSquare: Để in ra i^2



printHex: In chuyển từ hệ thập phân sang hệ hexa ( ví dụ 3 -> 0x3)



exit:

Sau khi tính toán hết , ta thực hiện in ra .

Trước hết , hàm print để in form :

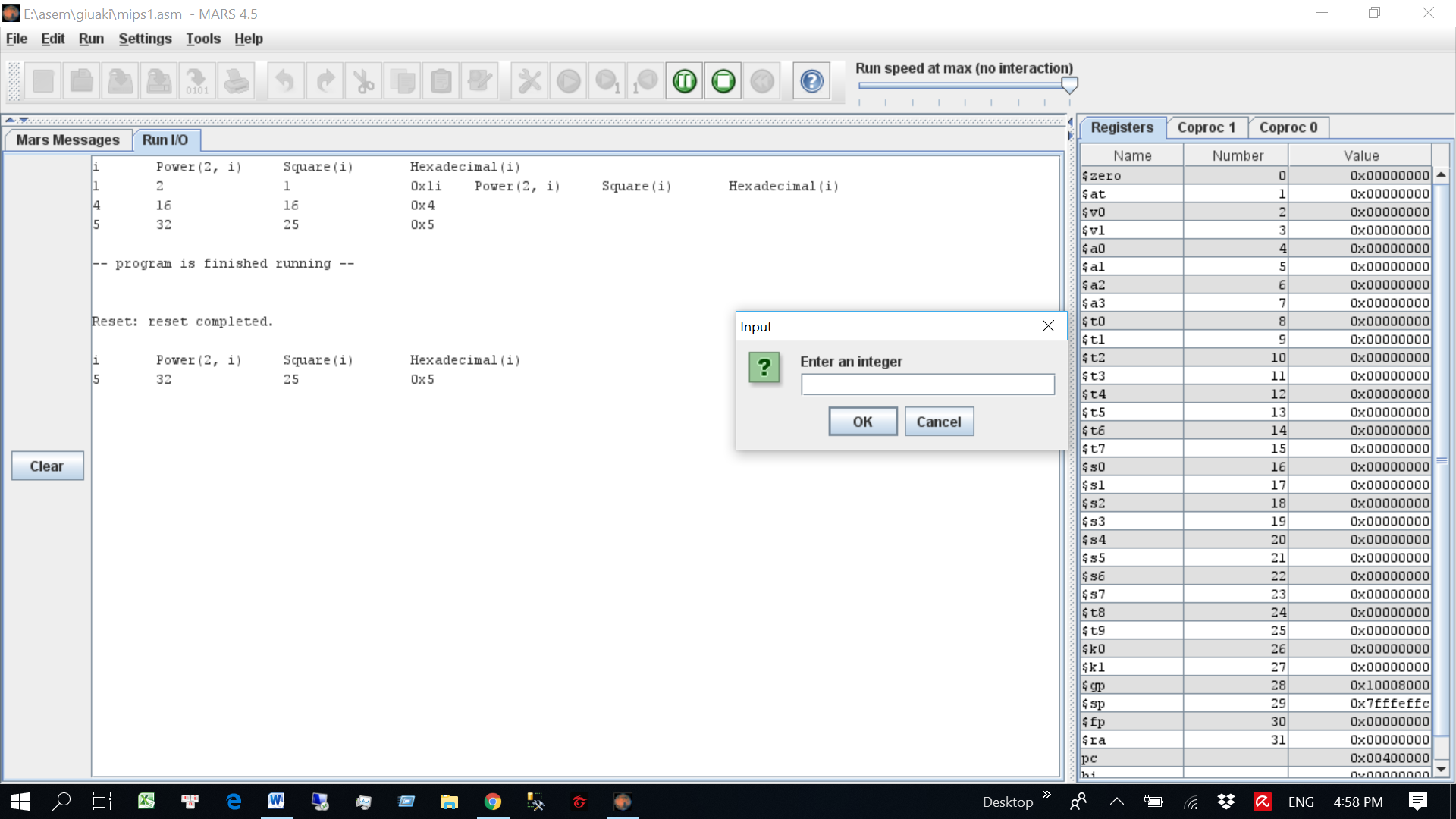
I Power(I,2) square hexadecimal

Hàm: checkPowerResult:

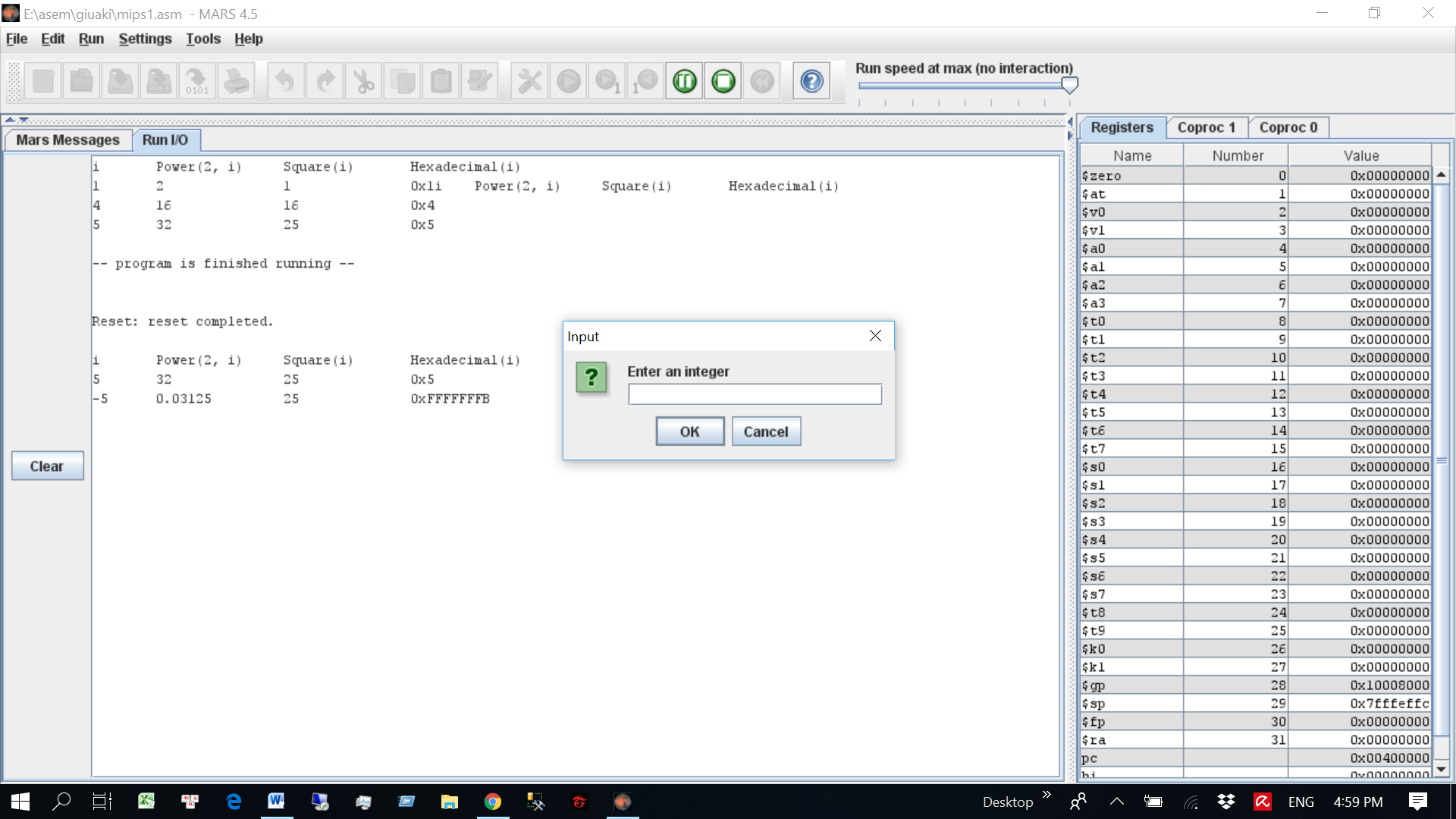
blt $s0, $zero, printFloat

hàm trên để check xem nếu $s0<0 thì ta chạy hàm printFloat , nếu $s0>0 thì ta chạy hàm printInt

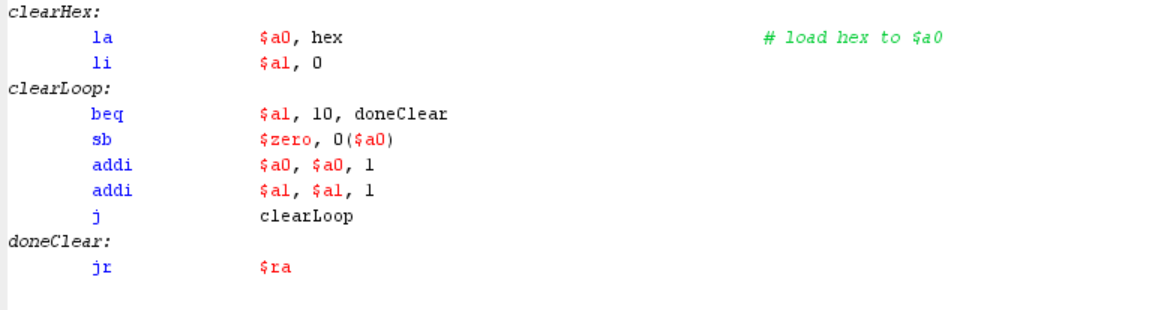
Kết quả nếu nhập i=5:



Kết quả khi nhập -5 :



Sau khi chạy xong printHex ta chạy hàm clearHex để sử dụng cho các lần nhập tiếp theo :



Khi khai báo hex , như sau : hex: .space 10

Ta để 10 khoảng cho hex , nên khi clear ta cho vòng loop đến 10 để clear