Báo cáo thực hành môn Kiến trúc máy tính

Họ tên: Nguyễn Thanh Hưng

MSSV: 20225633

1. Assignment 1

Code:

.text

start:

    addi    $s1,        $zero,      -7

    addi    $s2,        $zero,      -12

    li      $t0,        0                   # No Overflow is default status

    addu    $s3,        $s1,        $s2     # s3 = s1 + s2

    xor     $t1,        $s1,        $s2     # Test if $s1 and $s2 have the same sign

    bltz    $t1,        EXIT                # If not, exit

    slt     $t2,        $s3,        $s1

    bltz    $s1,        NEGATIVE            # Test if $s1 and $s2 is negative?

    beq     $t2,        $zero,      EXIT    # s1 and $s2 are positive

    # if $s3 > $s1 then the result is not overflow

    j       OVERFLOW

NEGATIVE:

    bne     $t2,        $zero,      EXIT    # s1 and $s2 are negative

    # if $s3 < $s1 then the result is not overflow

OVERFLOW:

    li      $t0,        1                   # the result is overflow

EXIT:

Giải thích code:

* 2 lệnh addi thực hiện việc gán giá trị cho 2 thanh ghi $1, $2



* Lệnh li sẽ gắn giá trị Overflow ở 0 tức là không có tràn số
* Lệnh addu thực hiện việc cộng 2 giá trị số nguyên không dấu ngay cả khi kết quả vượt quá phạm vi biểu diễn số nguyên không dấu.
* Lệnh xor thực hiện việc kiểm tra xem 2 giá trị $s1 và $s2 có cùng dấu hay không bằng việc so sánh từng bit tương ứng của 2 giá trị và lưu kết quả vào $t1



* Lệnh bltz thực hiện việc kiểm tra xem $t1 có phải số âm hay không bằng cách so sánh với số 0 bởi vì điều đó chứng tỏ 2 giá trị $s1 và $s2 khác dấu nhau. Nếu khác thì sẽ nhảy đến nhãn EXIT

**Kiểm tra Số Nguyên Âm (Nhãn start):**

* slt $t2, $s3, $s1 (kiểm tra nhỏ hơn): So sánh kết quả phép cộng ($s3) với số hạng thứ nhất ($s1). Nếu $s3 nhỏ hơn $s1 (dấu trừ), điều này cho biết tổng là âm và không thể tràn vì trong hệ bù 2, số âm không bị tràn. Kết quả được lưu trữ trong $t2.



* bltz $s1, NEGATIVE (nhảy nếu nhỏ hơn 0): Nếu $s1 là âm (nhỏ hơn 0), điều đó nghĩa là cả hai số hạng đều âm. Chương trình nhảy đến nhãn NEGATIVE để xử lý tiếp.

**Kiểm tra Số Nguyên Dương (Nhãn start):**

* beq $t2, $zero, EXIT (nhảy nếu bằng): Nếu $t2 (kết quả của phép so sánh ở bước 4) bằng 0, điều này cho biết tổng ($s3) không nhỏ hơn $s1 (không tràn đối với số dương). Chương trình nhảy đến nhãn EXIT.
* Nếu không đáp ứng bất kỳ điều kiện nào ở trên (tức là các số hạng có cùng dấu dương và tổng không nhỏ hơn số hạng thứ nhất), thì điều đó có nghĩa là đã xảy ra tràn. Chương trình nhảy đến nhãn OVERFLOW.
* li $t0, 1 (nạp giá trị trực tiếp): Thiết lập $t0 thành 1, báo hiệu có tràn.

(Ở đây giá trị của chương trình không xảy ra việc tràn số nên $t0 = 0)



1. Assignment 2

Code:

.data

result: .word   0

.text

main:

    # Extract MSB of $s0

    andi    $t0,    $s0,    0x80000000

    srl     $t0,    $t0,    24

    # Clear LSB of $s0

    andi    $t1,    $s0,    0xFFFFFFFE

    move    $s0,    $t1

    # Set LSB of $s0 (bits 7 to 0 are set to 1)

    ori     $t2,    $s0,    0x00000001

    move    $s0,    $t2

    # Clear $s0 ($s0=0, must use logical instructions)

    xor     $s0,    $s0,    $s0

    # Store the result in memory

    sw      $s0,    result

    # Exit the program

    li      $v0,    10

    syscall

1. Assignment 3
   1. abs $s0, $s1

$s0 <= | $s1|

.data

    result: .asciiz "The absolute value of $s1 is less than or equal to $s0."

    not\_result: .asciiz "The absolute value of $s1 is greater than $s0."

.text

    main:

        move $s0, $s1   # Move the value of $s1 to $s0

        bltz $s0, negate   # Branch if $s0 < 0

        j is\_less\_equal

    negate:

        sub $s0, $zero, $s0   # Negate the value of $s0 #abs $s0

    is\_less\_equal:

        ble $s1, $s0, print\_result   # Branch if $s1 <= $s0

        li $v0, 4

        la $a0, not\_result   # Print "The absolute value of $s1 is greater than $s0."

        syscall

        j end

    print\_result:

        li $v0, 4

        la $a0, result   # Print "The absolute value of $s1 is less than or equal to $s0."

        syscall

    end:

        li $v0, 10   # Exit the program

        syscall

* 1. move $s0,$s1

$s0 <= $s1

.data

    result: .asciiz "The absolute value of $s1 is less than or equal to $s0."

    not\_result: .asciiz "The absolute value of $s1 is greater than $s0."

.text

    main:

        abs $s0, $s1   # Calculate the absolute value of $s1

        ble $s1, $s0, is\_less\_equal   # Branch if $s1 <= $s0

        li $v0, 4

        la $a0, not\_result   # Print "The absolute value of $s1 is greater than $s0."

        syscall

        j end

    is\_less\_equal:

        add $s0, $zero, $s1   # Set $s0 to the value of $s1 #move $s0, $s1

        li $v0, 4

        la $a0, result   # Print "The absolute value of $s1 is less than or equal to $s0."

        syscall

    end:

        li $v0, 10   # Exit the program

        syscall

* 1. not $s0, $s1

$s0 <= bit invert ($s1)

.data

    result: .asciiz "The absolute value of $s1 is less than or equal to $s0."

    not\_result: .asciiz "The absolute value of $s1 is greater than $s0."

    invert\_result: .asciiz "$s0 is less than or equal to bit invert ($s1)."

.text

    main:

        abs $s0, $s1   # Calculate the absolute value of $s1

        ble $s1, $s0, is\_less\_equal   # Branch if $s1 <= $s0

        sub $s1, $zero, $s1   # Negate $s1

        ble $s0, $s1, is\_invert\_less\_equal   # Branch if $s0 <= $s1

        li $v0, 4

        la $a0, not\_result   # Print "The absolute value of $s1 is greater than $s0."

        syscall

        j end

    is\_less\_equal:

        move $s0, $s1   # Move $s1 to $s0

        li $v0, 4

        la $a0, result   # Print "The absolute value of $s1 is less than or equal to $s0."

        syscall

        j end

    is\_invert\_less\_equal:

        li $v0, 4

        la $a0, invert\_result   # Print "$s0 is less than or equal to bit invert ($s1)."

        syscall

    end:

        li $v0, 10   # Exit the program

        syscall

* 1. ble $s1,$s2,label

if ($s1 <= $s2)

j label

.data

    result: .asciiz "The absolute value of $s1 is less than or equal to $s0."

    not\_result: .asciiz "The absolute value of $s1 is greater than $s0."

    invert\_result: .asciiz "$s0 is less than or equal to bit invert ($s1)."

.text

    main:

        abs $s0, $s1   # Calculate the absolute value of $s1

        slt $t0, $s1, $s0   # Set $t0 to 1 if $s1 < $s0, otherwise set it to 0

        beq $t0, $zero, is\_greater   # Branch if $t0 is equal to 0

        not $s1, $s1   # Bit invert $s1

        slt $t1, $s0, $s1   # Set $t1 to 1 if $s0 < $s1, otherwise set it to 0

        beq $t1, $zero, is\_invert\_greater   # Branch if $t1 is equal to 0

        li $v0, 4

        la $a0, not\_result   # Print "The absolute value of $s1 is greater than $s0."

        syscall

        j end

    is\_greater:

        move $s0, $s1   # Move $s1 to $s0

        li $v0, 4

        la $a0, result   # Print "The absolute value of $s1 is less than or equal to $s0."

        syscall

        j end

    is\_invert\_greater:

        li $v0, 4

        la $a0, invert\_result   # Print "$s0 is less than or equal to bit invert ($s1)."

        syscall

    end:

        li $v0, 10   # Exit the program

        syscall

1. Assignment 4

Code:

.text

start:

    addi    $s1,        $zero,      -7

    addi    $s2,        $zero,      -12

    li      $t0,        0                   # No Overflow is default status

    addu    $s3,        $s1,        $s2     # s3 = s1 + s2

    xor     $t1,        $s1,        $s2     # Test if $s1 and $s2 have the same sign

    bltz    $t1,        EXIT                # If not, exit

    slt     $t2,        $s3,        $s1

    bltz    $s1,        NEGATIVE            # Test if $s1 and $s2 is negative?

    beq     $t2,        $zero,      EXIT    # s1 and $s2 are positive

    # if $s3 > $s1 then the result is not overflow

    j       OVERFLOW

NEGATIVE:

    bne     $t2,        $zero,      EXIT    # s1 and $s2 are negative

    # if $s3 < $s1 then the result is not overflow

OVERFLOW:

    li      $t0,        1                   # the result is overflow

EXIT:

1. Assignment 5

Code:

.text

start:

li $t0, 3

li $t1, 16

li $s2, 1

loop:

beq $t1, $s2 , EXIT

sll $t0, $t0, 1

srl $t1, $t1, 1

j loop

EXIT:

sw $t0, result

.data

result: .word 0

.end