

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HCM



ASSIGNMENT 1
KIẾN TRÚC MÁY TÍNH

Giáo viên hướng dẫn: Thầy Trần Thanh Bình

Tên SV	MSSV	Lớp
Võ Công Thành	1814038	L09
Nguyễn Hữu Thắng	1814096	L09

Tp HCM, 13 – 12 – 2020

MỤC LỤC

- I. Mô tả đề tài**
- II. Yêu cầu đặt ra**
- III. Thực hiện bài tập lớn**
 - 1. Cơ sở lý thuyết:**
 - 2. Thực hiện:**
 - a) Code:**
 - b) Testcase:**

I. Mô tả đề tài:

- Sắp xếp chuỗi.
- Cho một chuỗi số nguyên 20 phần tử. Sử dụng hợp ngữ assembly MIPS, viết thủ tục sắp xếp chuỗi đó theo tứ tự tăng dần theo giải thuật quick sort. Yêu cầu xuất ra từng bước trong quá trình demo

II. Yêu cầu đặt ra:

Mỗi nhóm sinh viên làm một đề.

- Sử dụng tập lệnh MIPS để thực hiện các thủ tục bên dưới
- Thống kê số lệnh, loại lệnh của chương trình của nhóm
- Tính và trình bày cách tính thời gian chạy của chương trình trên máy tính MIPS có tần số 2GHz
- Code:
 - Code style phải rõ ràng, có comment, phân hoạch công việc theo từng hàm
 - Truyền nhận và trả kết quả gọi hàm theo quy ước sử dụng thanh ghi (\$a0~\$a3 cho argument, \$v0~\$v1 cho kết quả trả về)
 - Xuất kết quả để kiểm tra (sử dụng các hàm hệ thống)

III. Thực hiện bài tập lớn:

1. Cơ sở lý thuyết:

Thuật toán Quick Sort là một thuật toán sắp xếp, còn được gọi là sắp xếp kiểu phân chia (Part Sort). Là một thuật toán hiệu quả dựa trên việc phân chia mảng dữ liệu thành các nhóm phần tử nhỏ hơn.

2. Thực hiện:

a) Code

```

#print array!
L:
    lw      $a0,0($s3)
    li      $v0,1
    syscall
    la $a0,space
    li $v0,4
    syscall
    addi    $s3,$s3,4
    beq     $s3,$s4,exit
    j      L

exit:
    li      $v0,10
    syscall
    #####

put:
    beq     $t7,$t0,done
    li      $v0,5
    syscall
    sw      $v0,array($t7)
    addi    $t7,$t7,4
    j      put

done:
    jr      $ra
    #####

swap2:
    #swap low and pivot
    sw      $a0,0($s1)
    sw      $a1,0($s0)
    jr      $ra

swap1:
    #swap low and high
    sw      $a1,0($s2)
    sw      $a2,0($s1)
    jr      $ra

partition:
    #t4 -> position of pivot = left
    addi    $sp,$sp,-4
    sw      $ra,0($sp)                # return address
    mul     $s5,$t1,4                 #s5 = 4*low
    add     $s1,$s3,$s5               #s1 = arr[low]
    mul     $t6,$t3,4                 #t6 = 4*high
    add     $s0,$s3,$t6               #s0 = arr[high]
    addi    $s2,$s0,-4                #s2 = arr[right]

loop:
    lw      $a0,0($s0)                # a0 = pivot
    lw      $a1,0($s1)                # a1 = arr[left]
    lw      $a2,0($s2)                # a2 = arr[right]

loop1:
    bgt     $t1,$t3,loop2             #left > right?
    bge     $a1,$a0,loop2             #arr[left] >= pivot?
    addi    $t1,$t1,1                 #left ++
    addi    $s1,$s1,4                 #arr[low++]
    lw      $a1,0($s1)
    j      loop1

loop2:
    bgt     $t1,$t3,break_if          #left > right?
    ble     $a2,$a0,break_if          #arr[right] <= pivot?
    addi    $t3,$t3,-1                #right--
    addi    $s2,$s2,-4                #arr[right--]
    lw      $a2,0($s2)
    j      loop2

break_if:
    bge     $t1,$t3,return
    jal     swap1
    addi    $s1,$s1,4
    addi    $s2,$s2,-4
    addi    $t1,$t1,1
    addi    $t3,$t3,-1
    j      loop

#t1 = left = low = 0,1,2..
#t3 = right      = 8,7,6..

```

```

return:
    jal    swap2
    lw     $ra, 0($sp)
    addi   $sp, $sp, 4
    jr     $ra                                #return left = t1

quickSort:
    #condition to stop
    addi   $sp, $sp, -12
    sw     $t1, 0($sp)                        # low
    sw     $t3, 4($sp)                        # high
    sw     $ra, 8($sp)                        # return address
    move   $t2, $t3                          #t2 = a2 = high
    bge    $t1, $t2, end                     #low >= high ?
    jal    partition
    move   $t5, $t1                          #t5 = pivot
    lw     $t1, 0($sp)                       #t1 = low
    addi   $t3, $t5, -1                      #t3 = pivot - 1
    jal    quickSort
    addi   $t1, $t5, 1                      #t1 = pi + 1
    lw     $t3, 4($sp)                       #t3 = high
    jal    quickSort

end:
    lw     $t1, 0($sp)                      #restore a1
    lw     $t3, 4($sp)                      #restore a2
    lw     $ra, 8($sp)                      #restore return address
    addi   $sp, $sp, 12                     #restore the stack
    jr     $ra                              #return to caller

```

b) Testcase

Testcase 1:

Trong trường hợp này số lệnh là 2726, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là CPU time = Instruction count * CPI / Clock rate = 1363000 (ps)

Mars Messages

Run I/O

Nhập chuỗi số nguyên 20 phần tử lần lượt là:
 100
 25
 4
 5
 2
 32
 100
 25
 34
 6
 49
 7
 9
 80
 1000
 316
 25
 98
 72
 46
 2 4 5 6 7 9 25 25 32 34 46 49 72 80 98 100 100 316 1000
 -- program is finished running --

Clear

Instruction Counter, Version 1.0 (Felipe Lessa)

×

Counting the number of instructions executed

Instructions so far: 2726

R-type: 742

27%

I-type: 1751

64%

J-type: 233

8%

Tool Control

Disconnect from MIPS

Reset

Close

Testcase 2:

Trong trường hợp này số lệnh là 2572, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1286000 \text{ (ps)}$

Nhập chuỗi số nguyên 20 phần tử liên tiếp là:
48
-34
356
34
-45
-56
123
0
8
5
4
-456
456
234
8
5
6
12345
-45
-7
-456 -56 -45 -45 -34 -7 0 4 5 5 6 8 8 34 48 123 234 356 456 12345
-- program is finished running --

Clear

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 2572

R-type: 700

27%

I-type: 1652

64%

J-type: 220

8%

Tool Control

Disconnect from MIPS

Reset

Close

Testcase 3:

Trong trường hợp này số lệnh là 2463, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1231500 \text{ (ps)}$

Nhập chuỗi số nguyên 20 phần tử liên tiếp là:
2
3465
7
5
34
57
-45
345
45
2
3
-56
2
45
547
567
-456
234
67
-56345
-56345 -456 -56 -45 2 2 3 5 7 34 45 45 57 67 234 345 547 567 3465
-- program is finished running --

Clear

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 2463

R-type: 664

26%

I-type: 1581

64%

J-type: 218

8%

Tool Control

Disconnect from MIPS

Reset

Close

Testcase 4:

Trong trường hợp này số lệnh là 2784, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là CPU time = Instruction count * CPI / Clock rate = 1392000 (ps)

Nhap chuoai so nguyen 20 phan tu lan luot la:
1
2
9
8
7
6
5
4
4
3
-1
2
3
4
4
5
6
7
8
9
-1 1 2 2 3 3 4 4 4 4 5 5 6 6 7 7 8 8 9 9
-- program is finished running --

Clear

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 2784

R-type: 750

26%

I-type: 1790

64%

J-type: 244

8%

Tool Control

Disconnect from MIPS

Reset

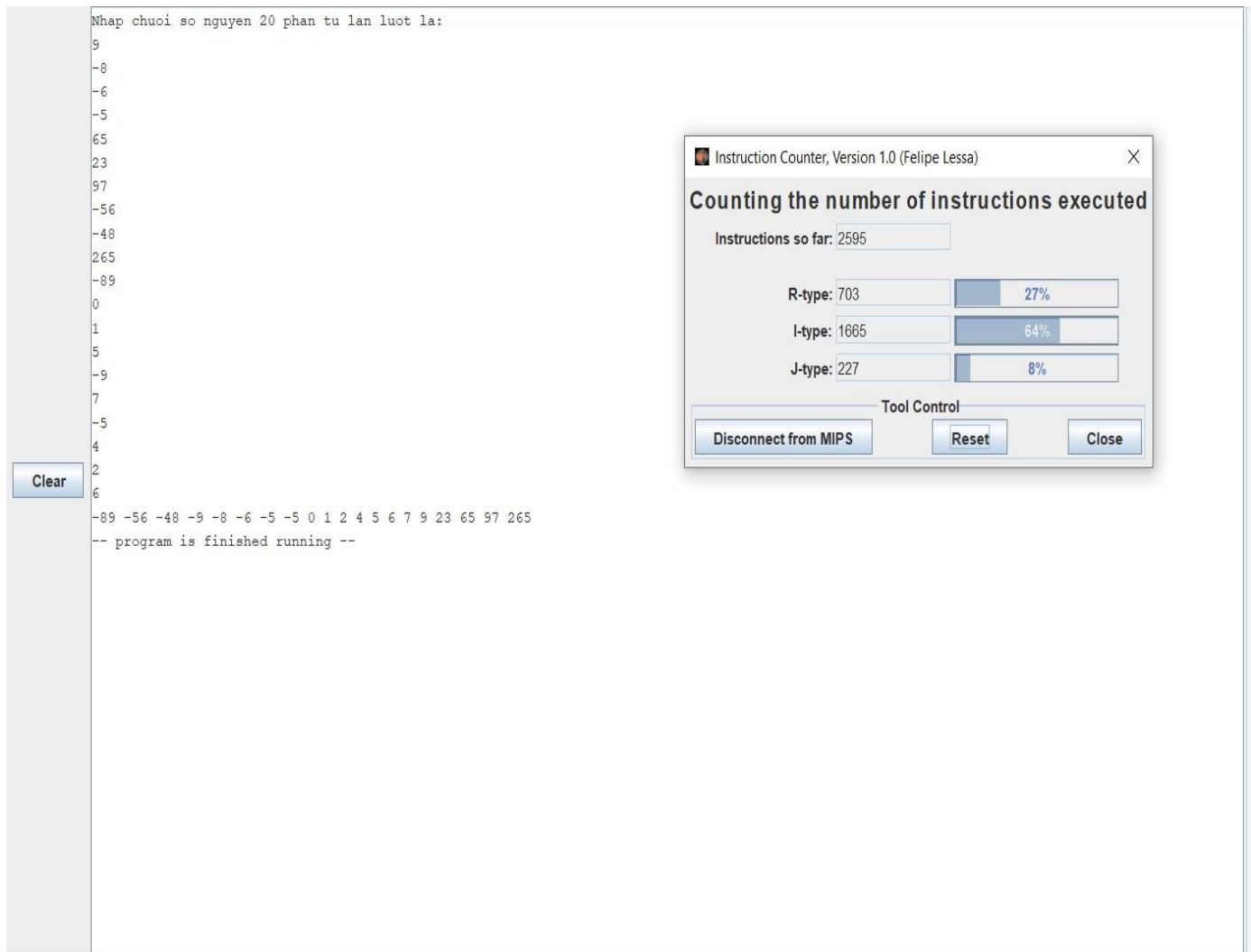
Close

Testcase 5:

Trong trường hợp này số lệnh là 2493, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là CPU time = Instruction count * CPI / Clock rate = 1246500 (ps)

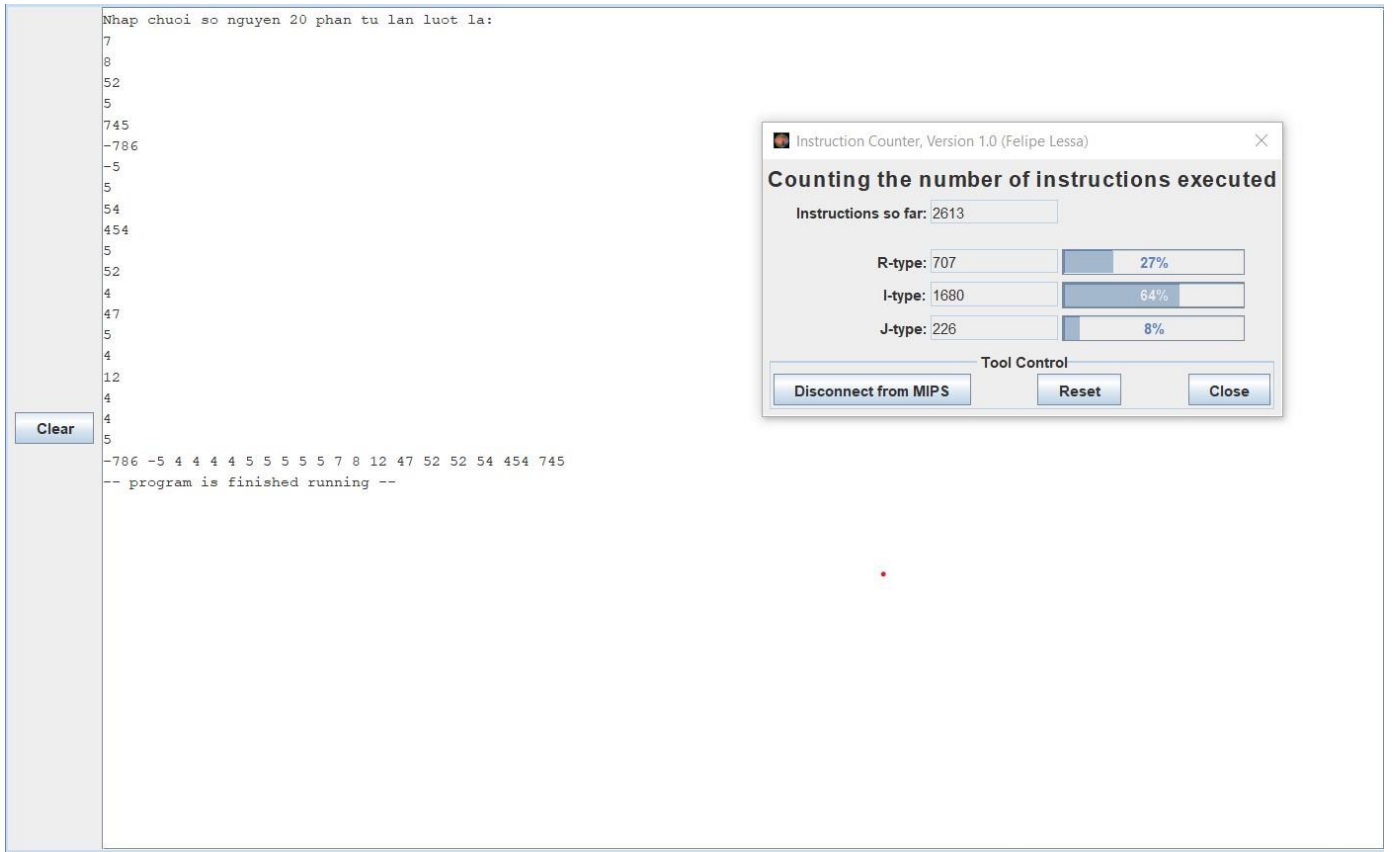
Testcase 7:

Trong trường hợp này số lệnh là 2595, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1297500 \text{ (ps)}$



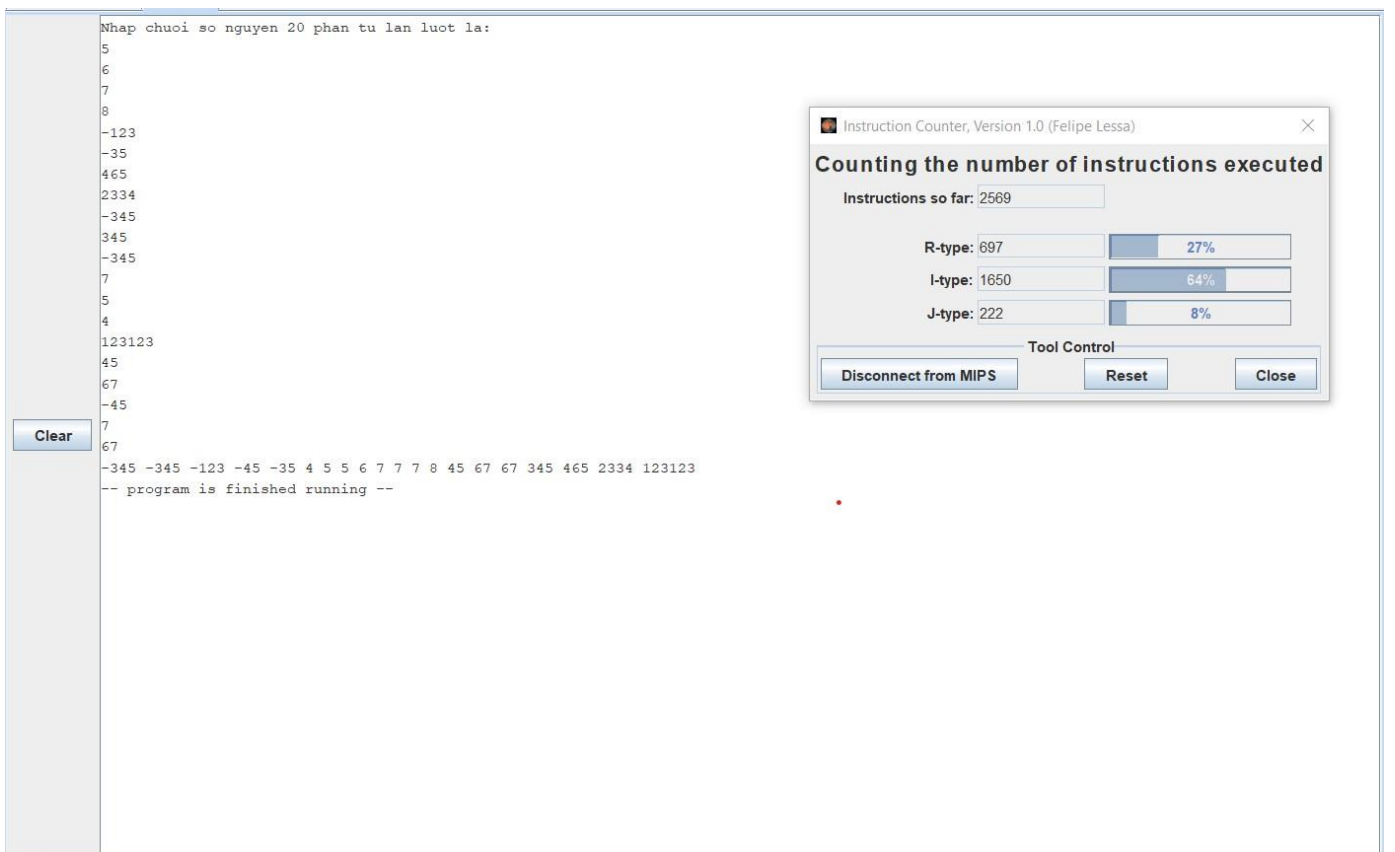
Testcase 8:

Trong trường hợp này số lệnh là 2613, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1306500 \text{ (ps)}$



Testcase 9:

Trong trường hợp này số lệnh là 2569, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1284500 \text{ (ps)}$



Testcase 10:

Trong trường hợp này số lệnh là 2646, theo đề có CPI = 1, clock rate = 2GHz Nên ta có thể tính thời gian là $\text{CPU time} = \text{Instruction count} * \text{CPI} / \text{Clock rate} = 1323000 \text{ (ps)}$

Nhập chuỗi số nguyên 20 phần tử lần lượt là:

4
7
9
10
-2
-4
1
2
3
-3
-5
-6
6
7
12
0
5
6
-4
-34
-6 -5 -4 -3 -2 0 1 2 3 4 5 6 6 7 7 9 10 12 34
-- program is finished running --

Clear

Instruction Counter, Version 1.0 (Felipe Lessa)

Counting the number of instructions executed

Instructions so far: 2646

R-type: 711 26%

I-type: 1695 64%

J-type: 240 9%

Tool Control

Disconnect from MIPS Reset Close