ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH

Bài tập lớn

Kiến trúc tập lệnh MIPS

GVHD: Trần Thanh Bình
Lớp: L10
SV thực hiện: Tô Hòa – 1910198
Nguyễn Thành Danh – 1910076
Nguyễn Huỳnh Đức – 1910137

Tp. Hồ Chí Minh, Tháng 12/2020

Mục lục

1 Động cơ nghiên cứu **2**

2 Mục tiêu **2**

3 Nhiệm vụ **2**
3.1 Nhiệm vụ 1 2
3.2 Nhiệm vụ 2 8
3.3 Nhiệm vụ 3 12

4 Kết luận **13**

Tài liệu **14**

1 Động cơ nghiên cứu

Kiến trúc tập lệnh (Instruction Set Architecture, viết tắt ISA) là dao diện chính giữa phần cứng và phần mềm, là cái nhìn trừu tượng của phần cứng trên quan điểm phần mềm. Hiểu được kiến trúc tập lệnh giúp cho lập trình viên có thể hiểu được máy tính hoạt động như thế nào. Từ đó, ngoài khả năng truy cập vào phần cứng hệ thống, lập trình viên còn có thể đưa ra những phương án lập trình mang lại hiệu quả cao lẫn về thời gian xử lý và không gian lưu trữ.

2 Mục tiêu

Thông qua bài tập lớn lần này, sinh viên có thể hiểu rõ được kiến trúc tập lệnh MIPS (Microprocessor without Interlocked Pipelined Stages). Đồng thời, sinh viên còn có thể sử dụng hợp ngữ MIPS để hiện thực những giải thuật đơn giản thường gặp phải, giúp cho việc sử dụng các ngôn ngữ lập trình cấp cao sẽ trở nên tối ưu và hiệu quả hơn.

3 Nhiệm vụ

3.1 Sử dụng hợp ngữ assembly MIPS, viết thủ tục sắp xếp chuỗi theo tứ tự tăng dần bằng giải thuật Merge sort

a) Lý thuyết:

- Ý tưởng: thuật toán Merge sort chia mảng cần sắp xếp ra thành thành 2 nửa. Thực hiện sắp xếp trên 2 nửa này bằng cách gọi đệ quy giải thuật Merge sort. Sau cùng gộp các nửa đó thành mảng đã sắp xếp.
- Giải thuật gồm 2 phần phần chính:
  - Hàm mergeSort: thực hiện việc chia mảng thành 2 mảng con và sắp xếp trên mỗi mảng con.
  - Hàm merge: thực hiện việc ghép 2 mảng đã sắp xếp theo thứ tự thành một mảng lớn theo thứ tự.

b) Mã giả:

- Hàm merge:

```
Algorithm merge(pointer arr, integer middle, integer size)

Pre: Two sorted subarrays
Post: A sorted array merged from 2 subarrays

if (middle > size) return
create a temporay array: temp[middle]
copy the first sub array to temp
i = 0, j = middle, k = 0
while i < middle or j < size do
    if (temp[i] <= arr[j]) then
        arr[k] = temp[i]
        i++, k++
    else
        arr[k] = arr[j]
        j++, k++
    end
end
while i < middle do
    arr[k] = temp[i]
    i++, k++
end
while (j < size) do
```

```
    arr[k] = temp[j]
    j++, k++
end
End merge
```

- Hàm mergeSort:

```
Algorithm mergeSort(pointer arr, integer size)
Sorts the array using recursive merge sort

Pre: An unsorted array with its size
Post: Sorted array

if size <= 1 then return
mergeSort(arr, size / 2)
mergeSort(arr + size / 2, size - size / 2)
merge(arr, size/2, size)
End mergeSort
```

c) Hiên thực: Hiện thực giải thuật bằng hợp ngữ MIPS.

d) Kết quả:

- Testcase 1:
  - Input:

```
.data
arr: .word      99, 467, 3, 5, 3, 5, 3, -98, -780, 200000,
                -4, -203, 9, 5, 2090, -41, 0, 3, -8799, -55
size: .word 20
```

  - Output:

```
After Merge: 99 467
After Merge: 3 5
After Merge: 3 3 5
After Merge: 3 3 5 99 467
After Merge: 3 5
After Merge: -780 200000
After Merge: -780 -98 200000
After Merge: -780 -98 3 5 200000
After Merge: -780 -98 3 3 3 5 5 99 467 200000
After Merge: -203 -4
After Merge: 5 2090
After Merge: 5 9 2090
After Merge: -203 -4 5 9 2090
After Merge: -41 0
After Merge: -8799 -55
After Merge: -8799 -55 3
After Merge: -8799 -55 -41 0 3
After Merge: -8799 -203 -55 -41 -4 0 3 5 9 2090
After Merge: -8799 -780 -203 -98 -55 -41 -4 0 3 3 3 3 5 5 5 9 99 467 2090 200000
```

- Testcase 2:
  - Input:

```
.data
arr: .word      296, -388, -414, 861, 199, -711, 391, -513, -581, -37,
                -913, -244, 654, -531, 187, -631, -279, -51, -361, 860
size: .word 20
```

  - Output:

```
After Merge: -388 296
After Merge: 199 861
After Merge: -414 199 861
After Merge: -414 -388 199 296 861
After Merge: -711 391
After Merge: -581 -37
After Merge: -581 -513 -37
After Merge: -711 -581 -513 -37 391
After Merge: -711 -581 -513 -414 -388 -37 199 296 391 861
After Merge: -913 -244
After Merge: -531 187
After Merge: -531 187 654
After Merge: -913 -531 -244 187 654
After Merge: -631 -279
After Merge: -361 860
After Merge: -361 -51 860
After Merge: -631 -361 -279 -51 860
After Merge: -913 -631 -531 -361 -279 -244 -51 187 654 860
After Merge: -913 -711 -631 -581 -531 -513 -414 -388 -361 -279 -244 -51 -37 187 199 296 391 654 860 861
```

- Testcase 3:
  - Input:

```
.data
arr: .word      -760, 942, -244, -425, -777, 1, 386, -527, -859, -589,
                -122, 485, 261, -770, -38, 573, 873, 422, 429, -254
size: .word 20
```

  - Output:

```
After Merge: -760 942
After Merge: -777 -425
After Merge: -777 -425 -244
After Merge: -777 -760 -425 -244 942
After Merge: 1 386
After Merge: -859 -589
After Merge: -859 -589 -527
After Merge: -859 -589 -527 1 386
After Merge: -859 -777 -760 -589 -527 -425 -244 1 386 942
After Merge: -122 485
After Merge: -770 -38
After Merge: -770 -38 261
After Merge: -770 -122 -38 261 485
After Merge: 573 873
After Merge: -254 429
After Merge: -254 422 429
After Merge: -254 422 429 573 873
After Merge: -770 -254 -122 -38 261 422 429 485 573 873
After Merge: -859 -777 -770 -760 -589 -527 -425 -254 -244 -122 -38 1 261 386 422 429 485 573 873 942
```

- Testcase 4:
  - Input:

```
.data
arr: .word      8, -319, -468, 358, 316, -721, 916, -309, 741, 408,
                -112, 836, -447, 992, 241, -207, 878, 206, 664, 251
size: .word 20
```

  - Output:

```
After Merge: -319 8
After Merge: 316 358
After Merge: -468 316 358
After Merge: -468 -319 8 316 358
After Merge: -721 916
After Merge: 408 741
After Merge: -309 408 741
After Merge: -721 -309 408 741 916
After Merge: -721 -468 -319 -309 8 316 358 408 741 916
After Merge: -112 836
After Merge: 241 992
After Merge: -447 241 992
After Merge: -447 -112 241 836 992
After Merge: -207 878
After Merge: 251 664
After Merge: 206 251 664
After Merge: -207 206 251 664 878
After Merge: -447 -207 -112 206 241 251 664 836 878 992
After Merge: -721 -468 -447 -319 -309 -207 -112 8 206 241 251 316 358 408 664 741 836 878 916 992
```

- Testcase 5:
  - Input:

```
.data
arr: .word      821, -942, -737, -421, 741, -5, -516, -304, -641, -365,
                -173, -964, 341, 260, -935, -399, 654, 567, -239, -810
size: .word 20
```

  - Output:

```
After Merge: -942 821
After Merge: -421 741
After Merge: -737 -421 741
After Merge: -942 -737 -421 741 821
After Merge: -516 -5
After Merge: -641 -365
After Merge: -641 -365 -304
After Merge: -641 -516 -365 -304 -5
After Merge: -942 -737 -641 -516 -421 -365 -304 -5 741 821
After Merge: -964 -173
After Merge: -935 260
After Merge: -935 260 341
After Merge: -964 -935 -173 260 341
After Merge: -399 654
After Merge: -810 -239
After Merge: -810 -239 567
After Merge: -810 -399 -239 567 654
After Merge: -964 -935 -810 -399 -239 -173 260 341 567 654
After Merge: -964 -942 -935 -810 -737 -641 -516 -421 -399 -365 -304 -239 -173 -5 260 341 567 654 741 821
```

- Testcase 6:
  - Input:

```
.data
arr: .word      -876, 360, -761, -742, 817, -758, 560, -179, 440, -84,
                82, -517, 164, -34, -402, 863, 364, 728, -355, 34
size: .word 20
```

  - Output:

```
After Merge: -876 360
After Merge: -742 817
After Merge: -761 -742 817
After Merge: -876 -761 -742 360 817
After Merge: -758 560
After Merge: -84 440
After Merge: -179 -84 440
After Merge: -758 -179 -84 440 560
After Merge: -876 -761 -758 -742 -179 -84 360 440 560 817
After Merge: -517 82
After Merge: -402 -34
After Merge: -402 -34 164
After Merge: -517 -402 -34 82 164
After Merge: 364 863
After Merge: -355 34
After Merge: -355 34 728
After Merge: -355 34 364 728 863
After Merge: -517 -402 -355 -34 34 82 164 364 728 863
After Merge: -876 -761 -758 -742 -517 -402 -355 -179 -84 -34 34 82 164 360 364 440 560 728 817 863
```

- Testcase 7:
  - Input:

```
.data
arr: .word      -184, -738, 123, 872, 51, 509, -961, -169, 721, -754,
                -506, 821, -219, -388, -255, 522, 914, -516, -691, 528
size: .word 20
```

  - Output:

```
After Merge: -738 -184
After Merge: 51 872
After Merge: 51 123 872
After Merge: -738 -184 51 123 872
After Merge: -961 509
After Merge: -754 721
After Merge: -754 -169 721
After Merge: -961 -754 -169 509 721
After Merge: -961 -754 -738 -184 -169 51 123 509 721 872
After Merge: -506 821
After Merge: -388 -255
After Merge: -388 -255 -219
After Merge: -506 -388 -255 -219 821
After Merge: 522 914
After Merge: -691 528
After Merge: -691 -516 528
After Merge: -691 -516 522 528 914
After Merge: -691 -516 -506 -388 -255 -219 522 528 821 914
After Merge: -961 -754 -738 -691 -516 -506 -388 -255 -219 -184 -169 51 123 509 522 528 721 821 872 914
```

- Testcase 8:
  - Input:

```
.data
arr: .word      486, -771, 262, 651, 619, 288, 500, 332, 493, 76,
                -59, -918, -501, 504, 386, -113, -421, 334, -609, 189
size: .word 20
```

  - Output:

```
After Merge: -771 486
After Merge: 619 651
After Merge: 262 619 651
After Merge: -771 262 486 619 651
After Merge: 288 500
After Merge: 76 493
After Merge: 76 332 493
After Merge: 76 288 332 493 500
After Merge: -771 76 262 288 332 486 493 500 619 651
After Merge: -918 -59
After Merge: 386 504
After Merge: -501 386 504
After Merge: -918 -501 -59 386 504
After Merge: -421 -113
After Merge: -609 189
After Merge: -609 189 334
After Merge: -609 -421 -113 189 334
After Merge: -918 -609 -501 -421 -113 -59 189 334 386 504
After Merge: -918 -771 -609 -501 -421 -113 -59 76 189 262 288 332 334 386 486 493 500 504 619 651
```

- Testcase 9:
  - Input:

```
.data
arr: .word      -623, 424, 749, -432, 450, -290, 231, -382, 174, -410,
                331, -893, 100, -333, 766, -174, -373, 149, -343, -104
size: .word 20
```

  - Output:

```
After Merge: -623 424
After Merge: -432 450
After Merge: -432 450 749
After Merge: -623 -432 424 450 749
After Merge: -290 231
After Merge: -410 174
After Merge: -410 -382 174
After Merge: -410 -382 -290 174 231
After Merge: -623 -432 -410 -382 -290 174 231 424 450 749
After Merge: -893 331
After Merge: -333 766
After Merge: -333 100 766
After Merge: -893 -333 100 331 766
After Merge: -373 -174
After Merge: -343 -104
After Merge: -343 -104 149
After Merge: -373 -343 -174 -104 149
After Merge: -893 -373 -343 -333 -174 -104 100 149 331 766
After Merge: -893 -623 -432 -410 -382 -373 -343 -333 -290 -174 -104 100 149 174 231 331 424 450 749 766
```

- Testcase 10:
  - Input:

```
.data
arr: .word      14, -789, 854, 90, 944, 990, 567, -145, 670, 365,
                257, -304, -195, 169, 862, -350, 256, 587, 199, -313
size: .word 20
```

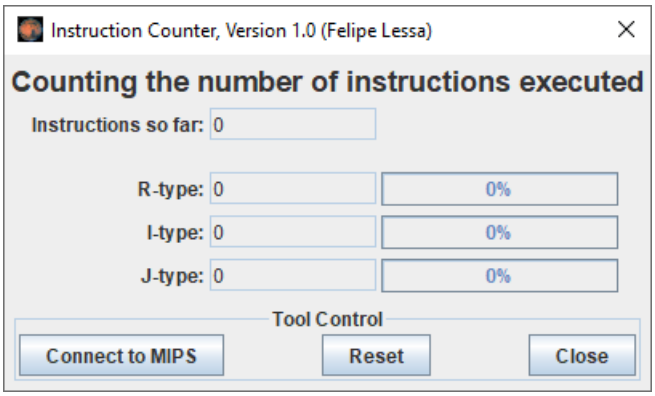
  - Output:

```
After Merge: -789 14
After Merge: 90 944
After Merge: 90 854 944
After Merge: -789 14 90 854 944
After Merge: 567 990
After Merge: 365 670
After Merge: -145 365 670
After Merge: -145 365 567 670 990
After Merge: -789 -145 14 90 365 567 670 854 944 990
After Merge: -304 257
After Merge: 169 862
After Merge: -195 169 862
After Merge: -304 -195 169 257 862
After Merge: -350 256
After Merge: -313 199
After Merge: -313 199 587
After Merge: -350 -313 199 256 587
After Merge: -350 -313 -304 -195 169 199 256 257 587 862
After Merge: -789 -350 -313 -304 -195 -145 14 90 169 199 256 257 365 567 587 670 854 862 944 990
```

3.2 Thống kê số lệnh, loại lệnh đã sử dụng

a) Công cụ:

- Sử dụng công cụ Instruction Counter được cung cấp bởi MARS (Tools/Instruction Counter) xác định tổng số lệnh đã thực thi của chương trình.

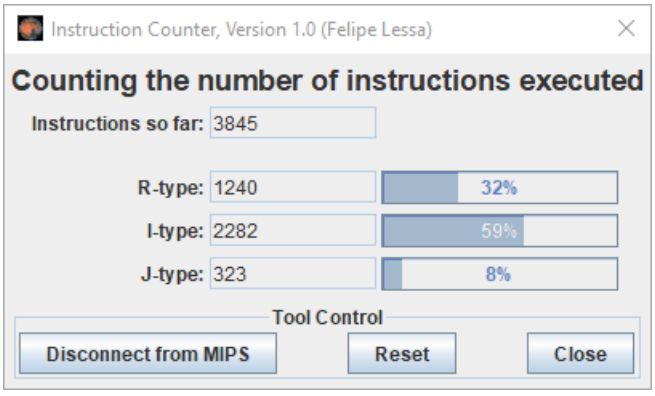


Trong đó:

- Instructions so far là tổng số câu lệnh mà chương trình đã thực thi.
- R-type là tổng số câu lệnh loại R mà chương trình đã thực thi.
- I-type là tổng số câu lệnh loại I mà chương trình đã thực thi.
- J-type là tổng số câu lệnh loại J mà chương trình đã thực thi.

b) Kết quả:

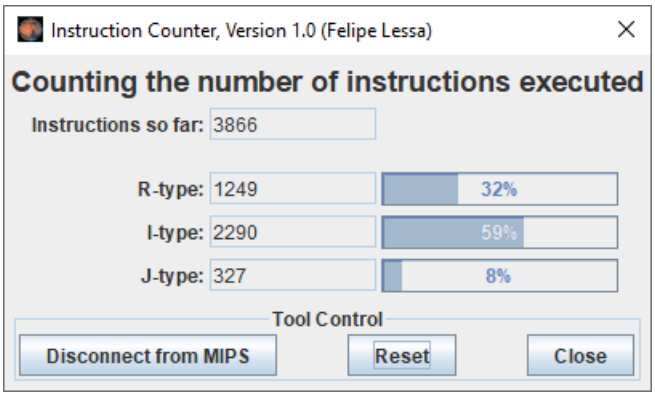
- Testcase 1:



Chương trình đã thực thi tổng cộng 3845 câu lệnh, trong đó:

- R-type là 1240 câu lệnh (chiếm 32%).
- I-type là 2282 câu lệnh (chiếm 59%).
- J-type là 323 câu lệnh (chiếm 8%).

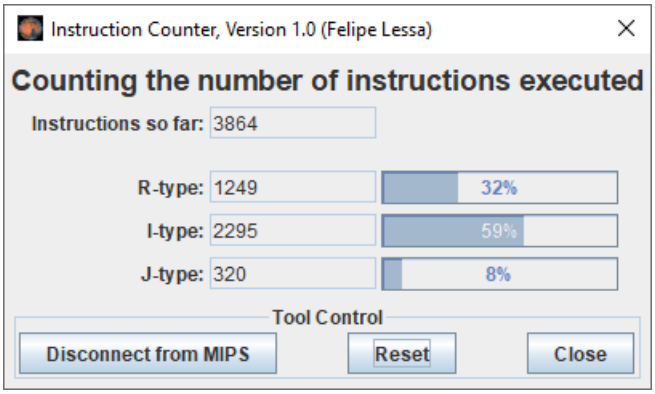
- Testcase 2:



Chương trình đã thực thi tổng cộng 3866 câu lệnh, trong đó:

- R-type là 1249 câu lệnh (chiếm 32%).
- I-type là 2290 câu lệnh (chiếm 59%).
- J-type là 327 câu lệnh (chiếm 8%).

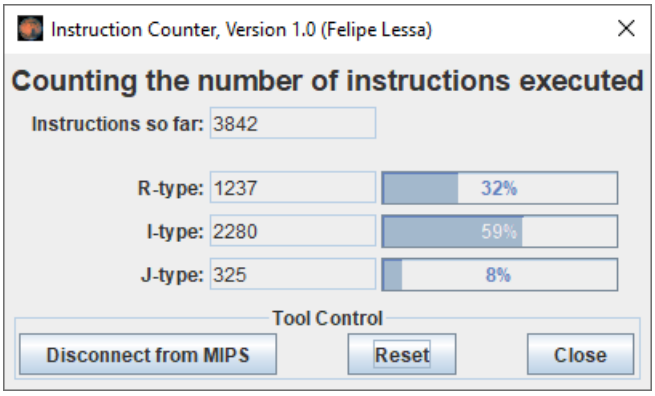
- Testcase 3:



Chương trình đã thực thi tổng cộng 3864 câu lệnh, trong đó:

- R-type là 1249 câu lệnh (chiếm 32%).
- I-type là 2295 câu lệnh (chiếm 59%).
- J-type là 320 câu lệnh (chiếm 8%).

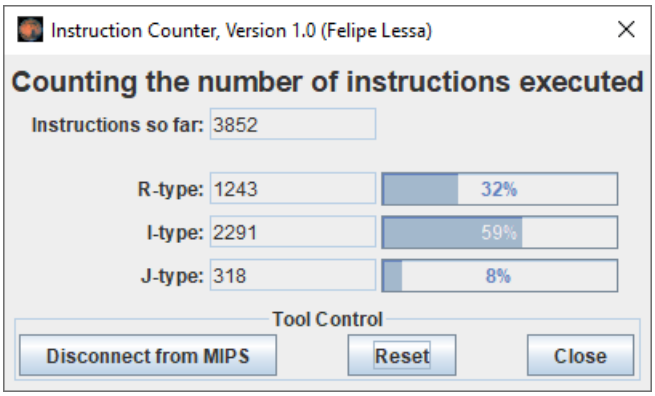
- Testcase 4:



Chương trình đã thực thi tổng cộng 3842 câu lệnh, trong đó:

- R-type là 1237 câu lệnh (chiếm 32%).
- I-type là 2280 câu lệnh (chiếm 59%).
- J-type là 325 câu lệnh (chiếm 8%).

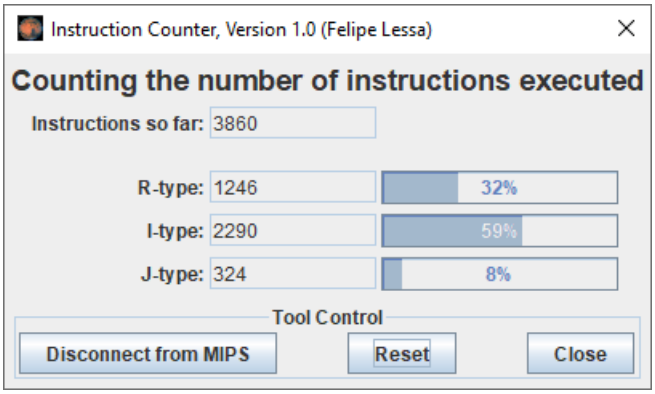
- Testcase 5:



Chương trình đã thực thi tổng cộng 3852 câu lệnh, trong đó:

- R-type là 1243 câu lệnh (chiếm 32%).
- I-type là 2291 câu lệnh (chiếm 59%).
- J-type là 318 câu lệnh (chiếm 8%).

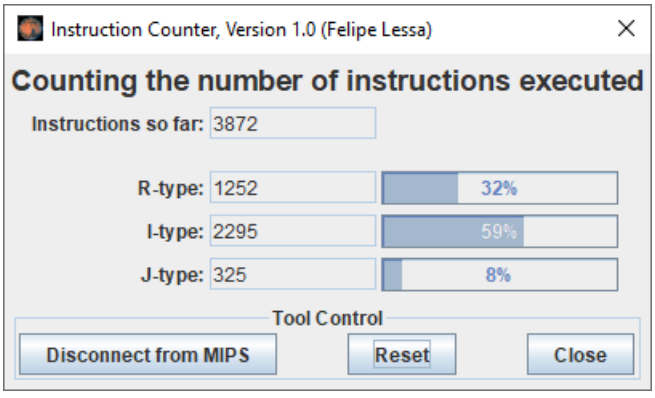
- Testcase 6:



Chương trình đã thực thi tổng cộng 3860 câu lệnh, trong đó:

- R-type là 1246 câu lệnh (chiếm 32%).
- I-type là 2290 câu lệnh (chiếm 59%).
- J-type là 324 câu lệnh (chiếm 8%).

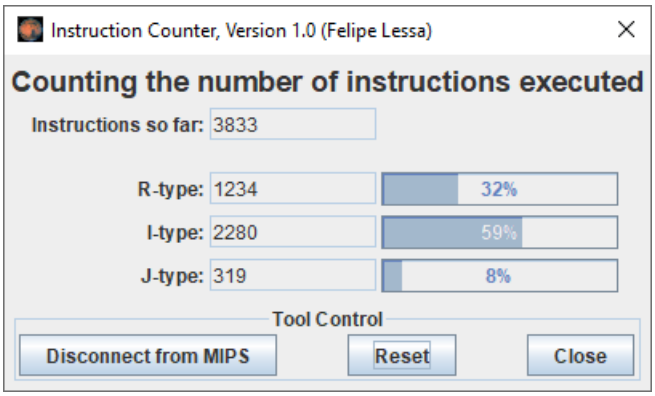
- Testcase 7:



Chương trình đã thực thi tổng cộng 3872 câu lệnh, trong đó:

- R-type là 1252 câu lệnh (chiếm 32%).
- I-type là 2295 câu lệnh (chiếm 59%).
- J-type là 325 câu lệnh (chiếm 8%).

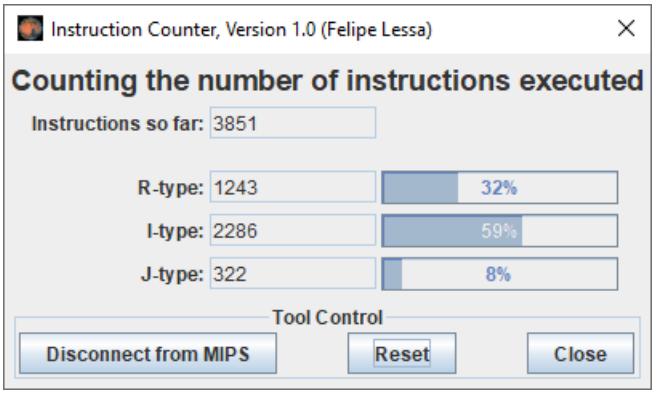
- Testcase 8:



Chương trình đã thực thi tổng cộng 3833 câu lệnh, trong đó:

- R-type là 1234 câu lệnh (chiếm 32%).
- I-type là 2280 câu lệnh (chiếm 59%).
- J-type là 319 câu lệnh (chiếm 8%).

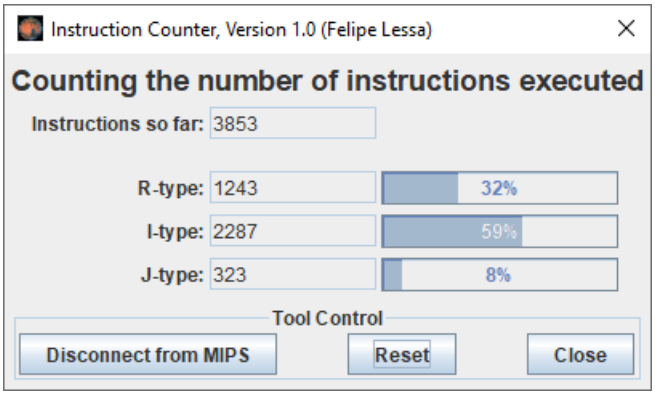
- Testcase 9:



Chương trình đã thực thi tổng cộng 3851 câu lệnh, trong đó:

- R-type là 1243 câu lệnh (chiếm 32%).
- I-type là 2286 câu lệnh (chiếm 59%).
- J-type là 322 câu lệnh (chiếm 8%).

- Testcase 10:



Chương trình đã thực thi tổng cộng 3853 câu lệnh, trong đó:

- R-type là 1243 câu lệnh (chiếm 32%).
- I-type là 2287 câu lệnh (chiếm 59%).
- J-type là 323 câu lệnh (chiếm 8%).

3.3 Tính và trình bày cách tính thời gian chạy của chương trình

a) Các bước thực hiện

- Xác định tổng số câu lệnh chương trình đã thực thi (đã thực hiện ở mục 3.2).
- Xác đinh thời gian thực thi của chướng trình bằng công thức sau:

$$\text{CPU Execution Time} = \frac{\text{Clock cycles}}{\text{Clock rate}} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock rate}}$$

Trong đó:

- CPU Execution Time là thời gian thực thi của CPU.
- Clock cycles là số chu kì xung nhịp của cần để thực thi chương trình.
- Instruction Count là tổng số câu lệnh chương trình đã thực thi.
- CPI là số chu kì xung nhịp trên mỗi câu lệnh (trên máy tính MIPS, đối với những lệnh giả giá trị này bằng 1).
- Clock rate là tần số của của 1 xung nhịp (chúng ta sẽ sử dụng máy tính MIPS có tần số 2 GHz).

b) Tính toán:

- Testcase 1: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3845 \times 1}{2 \times 10^9} = 1.9225 \times 10^{-6}\,\text{s}$$

- Testcase 2: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3866 \times 1}{2 \times 10^9} = 1.933 \times 10^{-6}\,\text{s}$$

- Testcase 3: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3864 \times 1}{2 \times 10^9} = 1.932 \times 10^{-6}\,\text{s}$$

- Testcase 4: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3842 \times 1}{2 \times 10^9} = 1.921 \times 10^{-6}\,\text{s}$$

- Testcase 5: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3852 \times 1}{2 \times 10^9} = 1.926 \times 10^{-6}\,\text{s}$$

- Testcase 6: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3860 \times 1}{2 \times 10^9} = 1.93 \times 10^{-6}\,\text{s}$$

- Testcase 7: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3872 \times 1}{2 \times 10^9} = 1.936 \times 10^{-6}\,\text{s}$$

- Testcase 8: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3833 \times 1}{2 \times 10^9} = 1.9165 \times 10^{-6}\,\text{s}$$

- Testcase 9: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3851 \times 1}{2 \times 10^9} = 1.9255 \times 10^{-6}\,\text{s}$$

- Testcase 10: Thời gian thực thi của chương trình.

$$\text{CPU Execution Time} = \frac{3853 \times 1}{2 \times 10^9} = 1.9265 \times 10^{-6}\,\text{s}$$

4 Kết luận

Từ bài tập lớn lần này, các thành viên trong nhóm đã có thêm những kiến thức về kiến trúc tập lệnh MIPS cũng như khả năng sử dụng hợp ngữ MIPS để hiện thực những giải thuật đơn giản. Qua đó, việc lập trình ở các ngôn ngữ cấp cao khác được cải thiện và trở nên hiệu quả.

Tài liệu

[1] Phạm Quốc Cường. Kiến trúc máy tính. NXB Đại Học Quốc Gia Hồ Chí Minh.

[2] Merge Sort. Link: https://www.geeksforgeeks.org/merge-sort/. Truy cập cuối: 4/12/2020.