

# Relatório do Experimento 1 de OAC

Arthur S. Couto: 16/0002575  
Cristiano Silva Júnior: 13/0070629  
Cristiano Silva Júnior: 13/0070629  
Cristiano Silva Júnior: 13/0070629

5 de Maio de 2017

## 1 Exercício 1

### 1.1 Exercício 1.1

Lendo o programa *sort.s* dado, nota-se que ele vai realizar um ordenamento decrescente devido à única comparação presente no código estar invertida. Após inverter os registradores para realizar a comparação correta, podemos analisar o programa usando a ferramenta *Instruction Counter* do *Mars*, contamos 551 instruções, sendo 204 do tipo R, 294 do tipo I e 53 do tipo J para o vetor dado. Utilizando a ferramenta de estatísticas *Instructions statistics*, foram 31% de instruções de ULA; 13% do tipo *jump*; 13% do tipo *branch*; 27% de memória; e 16% de outros tipos.

### 1.2 Exercício 1.2

Reutilizando o programa *sort.s*, podemos analisar este algoritmo de ordenamento para outras entradas. Considerando as entradas  $v_o(n) = 1, 2, 3, \dots, n$  e  $v_i(n) = n, n-1, n-2, \dots, 1$  para  $n = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ , podemos calcular o número de instruções para cada  $n$  e calcular o tempo de execução  $t$  com os valores sugeridos de frequência de clock e de CPI. A relação  $n \cdot t$  nas figuras 1 e 2.

Nota-se que o ordenamento de um vetor ordenado é simplesmente checar se o vetor já está ordenado, logo se espera que esta operação tenha complexidade  $O(n)$ ; enquanto que a mesma operação em um vetor inversamente ordenado é pior caso da operação de ordenação. Como está sendo usado o algoritmo Bubble Sort, a complexidade deste ordenamento é  $O(n^2)$ . Estas complexidades ficam evidentes nos gráficos gerados.

## 2 Exercício 2

### 2.1 Exercício 2.1

O programa *sortc.s* foi comentado para identificar o que cada uma das diretivas faz baseado nas especificações do GCC.

### 2.2 Exercício 2.2

Para identificar quais diretivas poderiam ser utilizadas pelo MARS, comparou-se quais as diretivas aceitas pelo MARS que eram geradas pelo GCC. Aquelas diretivas que não eram comuns a ambos

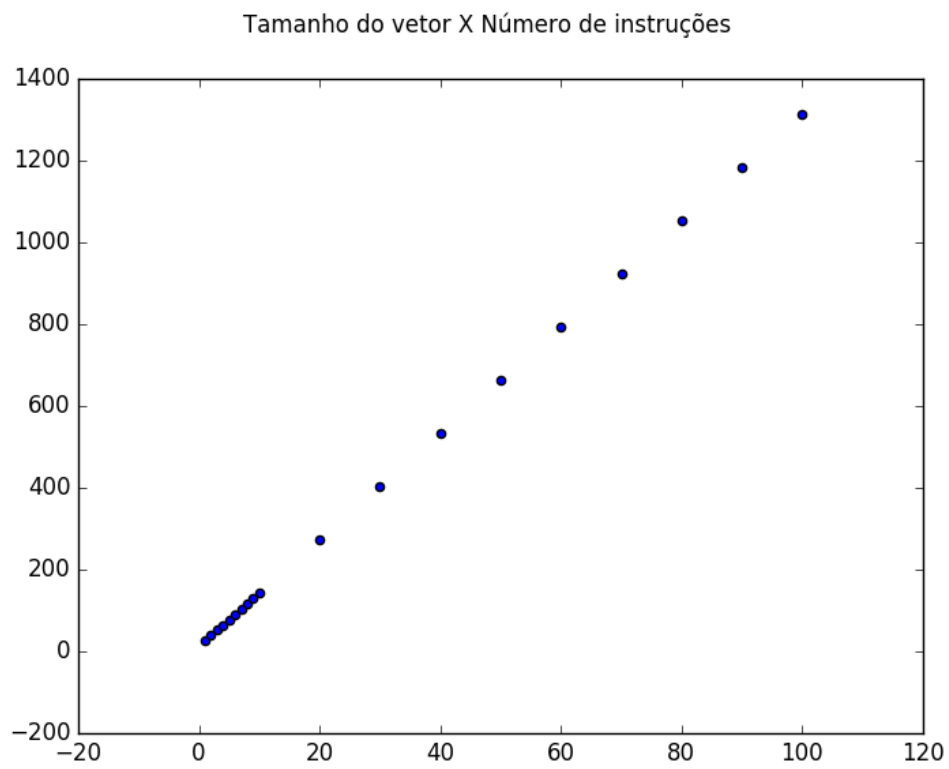


Figure 1: Melhores casos de ordenamento

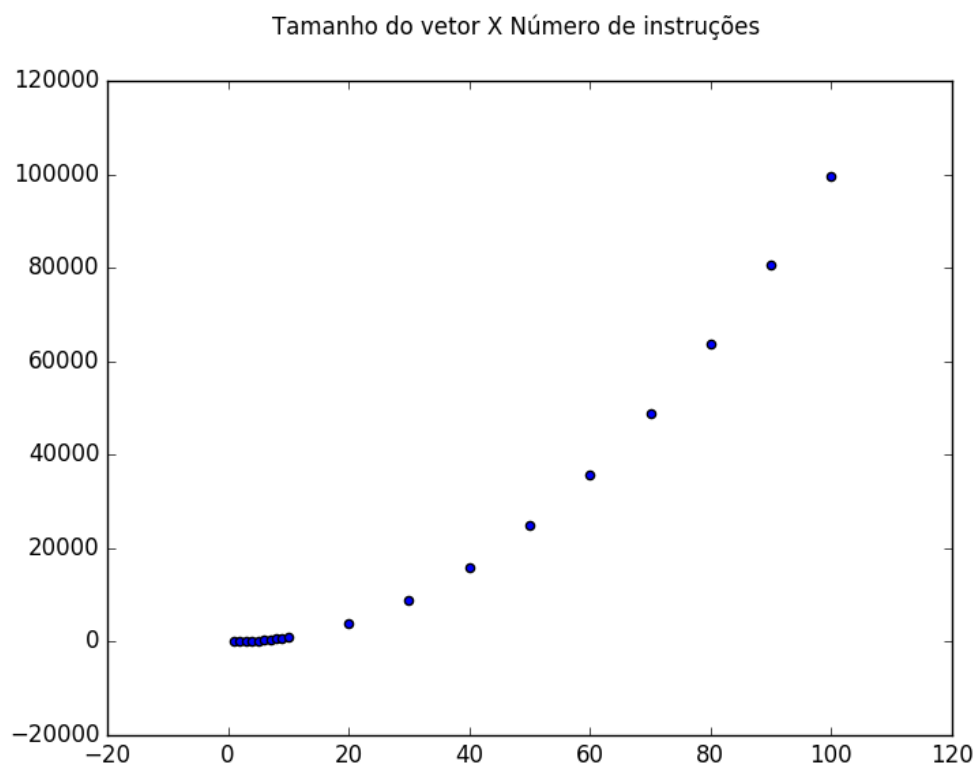


Figure 2: Piores casos de ordenamento

programas foram removidas. A saber, somente as seguintes diretivas são aceitas pelo MARS e pelo GCC:

- `.globl`
- `.align`
- `.word`
- `.ascii`
- `.data`
- `.text`
- `.set`

Todas as outras foram removidas. Aparentemente, o MARS é capaz de ignorar algumas diretivas que não fazem parte da sua especificação, como `.ident` e `.end`. Além disso, o GCC compreende que algumas instruções são realizadas de forma diferente pelo MARS. Foram elas:

- o GCC produzia sempre dois pares de instruções *lui* e *addiu* chamando *labels* em seus argumentos, o que é inválido para o MARS. Neste caso, substituiu-se esse par de instruções pelas pseudo-instrução *la \$rt, label*
- Dever-se-á substituir as chamadas de funções da *stdio.h* do GCC por operações equivalentes em MIPS dependendo da necessidade do programa. Funções como *printf* e *scanf* fazem operações bastante complexas com strings, que são muito complicadas de serem implementadas em Assembly à princípio.

## 2.3 Exercício 2.3

## 3 Exercício 3

Falar sobre como desenhamos os pontos, as linhas e tudo mais.