Fundamentos de Sistemas Digitais - Avaliação AA4

Desenvolvido pelos alunos Leonardo Chou da Rosa e Thiago Schuch Zilberknop

Os números das nossas matrículas são:
Leonardo Rosa – 2220062**8**-0

Thiago Zilberknop – 2220047**6**-4

Como a especificação é definida pelo menor último dígito, fizemos a especificação 3.

Pseudocódigo (código escrito em C++):

pseudocodigo.cpp

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int a[6] = {111, 222, 333, 444, 555, 666}; //vetor a
    int b[6] = {543, 431, 332, 54, 0, 1}; //vetor b
    int c[6] = {53, 340, 193, 12, 4, 999}; //vetor c
    vector<int> d; //vetor dinamico d, que recebera os valores de a, b, e c que são maiores que vm
    int mediaa = 0; //media do vetor a
    int mediab = 0; //media do vetor b
    int mediac = 0; //media do vetor c
    int vm = 0; //menor media entre os 3 vetores
    int k = 0; //quantidade de elementos no vetor dinamico d

    for (int i = 0; i < 6; i++) {
        mediaa += a[i];
        mediab += b[i];
        mediac += c[i];
    } // for que adiciona todos os valores dos vetores as suas respectivas medias

    //divide as médias pela quantidade de elementos (6)
    mediaa /= 6;
    mediab /= 6;
    mediac /= 6;

    //acha a menor média e coloca esse valor em vm
    if ((mediaa < mediab) && (mediaa < mediac)) {
        vm = mediaa;
    }
    if ((mediab < mediaa) && (mediab < mediac)) {
        vm = mediab;
    }
    if ((mediac < mediab) && (mediac < mediaa)) {
        vm = mediac;
    }

    //acha os valores maiores que vm dos vetores e insere eles para o vetor d
    for (int i = 0; i < 6; i++) {
        if (a[i] > vm) {
            d.push_back(a[i]);
        }
        if (b[i] > vm) {
            d.push_back(b[i]);
        }
        if (c[i] > vm) {
            d.push_back(c[i]);
        }
    }
    k = d.size(); //acha o tamanho do vetor d

    //imprime os valores
    cout << "media a = " << mediaa << endl;
    cout << "media b = " << mediab << endl;
    cout << "media c = " << mediac << endl;
    cout << "vm = " << vm << endl;
    for (int i = 0; i < k; i++) {
        cout << "d = " << d[i] << " na posicao " << i << endl;
    }
    cout << "k = " << k << endl;
}
```

Linhas 7-15: definindo as variáveis que serão usadas no programa (vetores a, b, e c com valores, vetor dinâmico d, as médias e o valor de k)

Linhas 17-21: soma todos os valores de cada vetor para as suas respectivas médias

Linhas 24-26: divide a soma dos vetores pela quantidade de inteiros presentes no vetor

Linhas 29-37: determina qual média é a menor; atribui esse valor para vm

Linhas 40-50: percorre os vetores a, b e c e se algum valor for maior que vm, adiciona esse valor para o vetor dinâmico d

Linha 51: atribui a quantidade de elementos no vetor d para a variável k

Linhas 54-62: imprime os respectivos valores

Output do pseudocódigo:

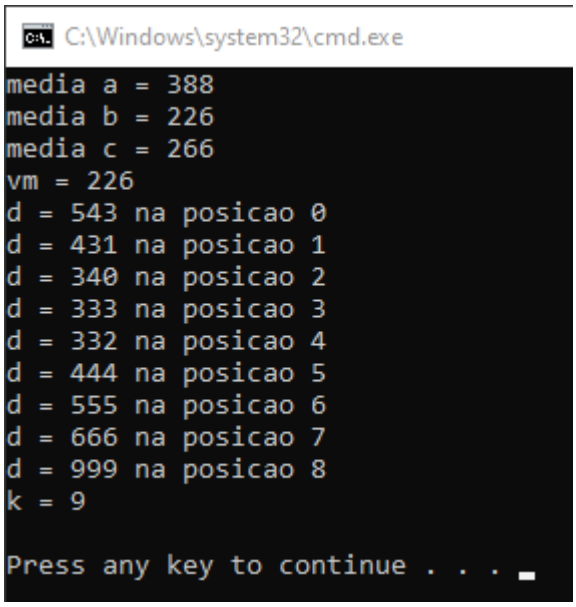


Tabela de registradores (valores de cima são as varáveis do pseudocódigo e os valores de baixo são os do código em assembly):

6 (tam vet)	a[6]	b[6]	c[6]	Vector <int> d	mediaa	mediab	mediac	i
N ($s0)	vetA ($t3)	vetB ($t4)	vetC ($t5)	vetD ($t6)	mediaA ($t6)	mediaB ($t7)	mediaC ($t8)	$s2

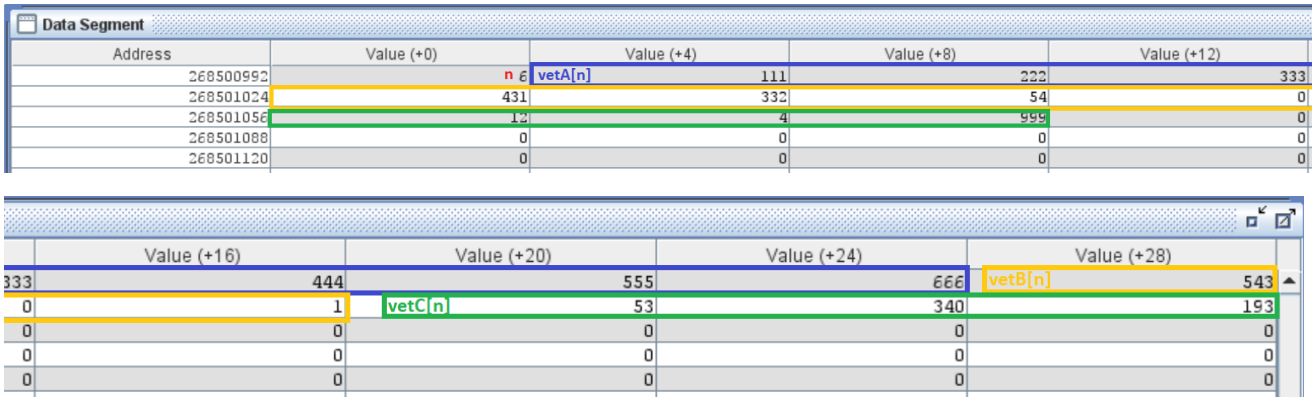
vm	a[i]	b[i]	c[i]	k
vm ($t0)	$s3	$s4	$s5	$t7

Alguns registradores cumpriram múltiplas funções que não são explícitas no código de alto nível:

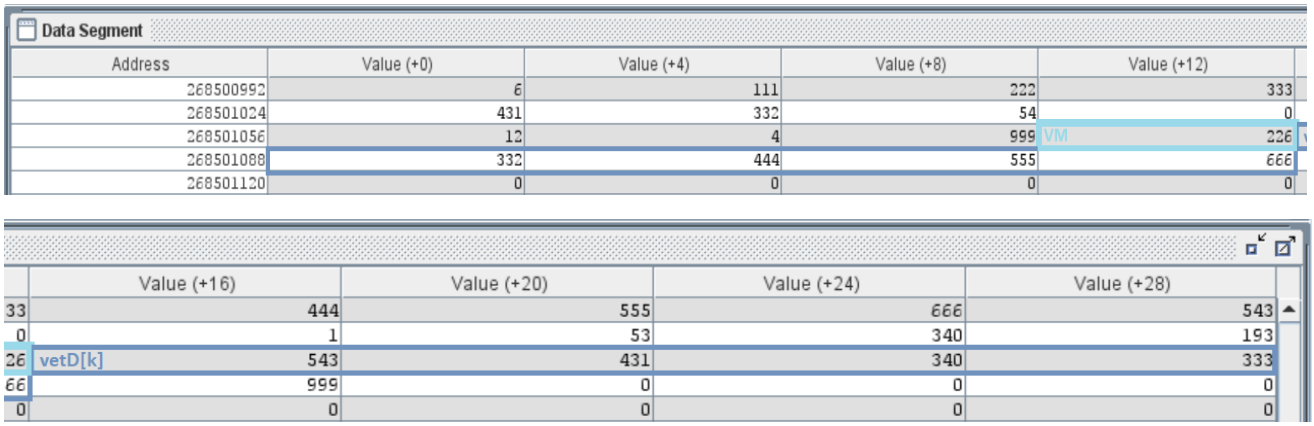
$t0	$t1	$t2
Endereço e valor de VM	----------	----------
Resultado das comparações “media1 < media2”	Valor ‘1’, usado em comparações com $t0	Resultado das comparações “a[i]/b[i]/c[i] < VM”
Usado na divisão	Usado na divisão	Usado na divisão

$t6	$t7	$t8
Endereço e valor de mediaA	Endereço e valor de mediaB	Endereço e valor de mediaC
Endereço de vetD	Endereço e valor de k	Elementos a serem adicionados em vetD
----------	----------	----------

Áreas de dados (momento inicial):



Áreas de dados (após execução):



Como pode observar, todos os valores que estão no vetorD estão presentes ou no vetorA, vetorB e vetorC e também são maiores que o valor de vm. Consequentemente, os valores dos vetores A, B e C que são menores que o valor de vm são excluídos do vetorD.

Código no simulador MARS:

```
.data
n:      .word 6
vetA:   .word 111 222 333 444 555 666
vetB:   .word 543 431 332 54 0 1
vetC:   .word 53 340 193 12 4 999

vm:     .word 0                         #valor mÃ©dio minimo
vetD:   .word                           #vetor de tamanho indefinido com valores de a, b, c que sÃ£o maiores que vm
k:      .word 0                         #numero de elementos de d

mediaA: .word 0
mediaB: .word 0
mediaC: .word 0
```

Banco de dados; definindo as variáveis:

N – tamanho dos vetores vetA, vetB e vetC

vetA, vetB e vetC – vetores que guardam valores

Vm – valor médio mínimo

VetD – vetor com tamanho inicial indeterminado que irá guardar os valores de vetA, vetB e vetC que são maiores do que o valor de vm

K – tamanho do vetD

MediaA- valor da média dos valores dentro do vetA

MediaB- valor da média dos valores dentro do vetB

MediaC- valor da média dos valores dentro do vetC

Variável: Valores:

n	6
vm	0
k	0
mediaA	0
mediaB	0
mediaC	0

vetA	111	222	333	444	555	666
vetB	543	431	332	54	0	1
vetC	53	340	193	12	4	999
vetD	-	-	-	-	-	-

Carrega de endereços:

```
1  .text
2          .globl main
3
4  main:           xor     $s2, $s2, $s2          #zera $s2 (i)
5
6                  la      $t3, vetA              #carrega o endereco de a
7                  la      $t4, vetB              #carrega o endereco de b
8                  la      $t5, vetC              #carrega o endereco de c
9
10                 la      $t6, mediaA            #carrega o endereco de mediaA
11                 la      $t7, mediaB            #carrega o endereco de mediaB
12                 la      $t8, mediaC            #carrega o endereco de mediaC
13
14                 la      $s0, n                 #carrega o endereco n
15                 lw      $s0, 0($s0)            #poe n (6) em $s0
16
```

Linha 4 – zera o valor de i (vai ser usado para realizar um loop depois)

Linhas 6-8 – carrega os endereços de vetA, vetB e vetC

Linhas 10-12 – Carrega os endereços de mediaA, mediaB e mediaC

Linha 14 – Carrega o endereço da variável n

Linha 15 – coloca o valor de n em $s0

Cálculo do valor total dos vetores:

```
17 calculo:        lw      $s3, 0($t3)            #a[i] em $s3
18                 lw      $s4, 0($t4)            #b[i] em $s4
19                 lw      $s5, 0($t5)            #c[i] em $s5
20
21                 #calculo A
22                 lw      $t6, 0($t6)            #$t6 = mediaA
23                 add     $s7, $t6, $s3          #adiciona media A com $s3 em $s7
24                 la      $t6, mediaA            #$t6 = &mediaA
25                 sw      $s7, 0($t6)            #coloca soma total no conteudo de $t6
26
27                 #calculo B
28                 lw      $t7, 0($t7)            #$t7 = mediaB
29                 add     $s7, $t7, $s4
30                 la      $t7, mediaB            #$t7 = &mediaB
31                 sw      $s7, 0($t7)
32
33                 #calculo C
34                 lw      $t8, 0($t8)            #$t8 = mediaC
35                 add     $s7, $t8, $s5
36                 la      $t8, mediaC            #$t8 = &mediaC
37                 sw      $s7, 0($t8)
38
```

```
39                 #loop
40                 addi    $s2, $s2, 1            #i++
41                 addi    $t3, $t3, 4            #a[i++]
42                 addi    $t4, $t4, 4            #b[i++]
43                 addi    $t5, $t5, 4            #c[i++]
44                 bne     $s2, $s0, calculo      #i != n
45
```

Linhas 17-20 – carrega os valores de a[i], b[i] e c[i] e coloca em $s3, $s4 e $s5 respectivamente

Linha 22 – carrega o valor de mediaA em $t6

Linhas 23-24 – adiciona o valor de $t6 (mediaA) junto com o valor de $s3 (a[i]) e coloca essa soma em $s7

Linha 25 – atualiza o valor de $t6 (mediaA) com o novo valor da soma

Linhas 27 – 37 – mesmo processo para a mediaB e mediaC; os três são somados no mesmo loop, por isso se usa registradores diferentes

Linhas 40-43 – incrementa os valores respectivos (i++, a[i++], b[i++], c[i++])

Linha 44 – condicional para continuar o loop (enquanto o valor de i for diferente de n (6, tamanho dos vetores), continua o loop)

Divide os valores totais pelo número de variáveis por n (número de variáveis nos vetores):

```
46              #divide A
47              lw      $t6, 0($t6)
48              add     $s1, $t6, $zero
49              la      $t6, mediaA
50
51              jal     divider
52
53              sw      $v1, 0($t6)
54
55              #divide B
56              lw      $t7, 0($t7)
57              add     $s1, $t7, $zero
58              la      $t7, mediaB
59
60              jal     divider
61
```

```
61
62              sw      $v1, 0($t7)
63
64              #divide C
65              lw      $t8, 0($t8)
66              add     $s1, $t8, $zero
67              la      $t8, mediaC
68
69              jal     divider
70
71              sw      $v1, 0($t8)
72
```

Linha 47 – carrega o valor de mediaA em $t6

Linha 48 – coloca a soma de $t6 com 0 em $s1

Linha 49 – carrega o endereço de mediaA em $t6

Linha 51 – jump para o divider, com retorno na linha 52

Linha 53 – coloca a soma do divider em $t6

Linhas 55-71 – mesmo processo, mas para as médias B e C

Define o valor médio menor:

```
73  defineVM:       xor     $t1, $t1, $t1
74                  addi    $t1, $t1, 1             #$t1 = 1
75
76  testaA:         lw      $t6, 0($t6)             #media A
77                  lw      $t7, 0($t7)             #media B
78                  lw      $t8, 0($t8)             #media C
79                  slt     $t0, $t6, $t7           #media A < media B, $t0 = 1, caso contrario $t0 = 0
80                  bne     $t0, $t1, testaB        #se media A >= media B, pula pra testaB
81                  slt     $t0, $t6, $t7           #media A < media C, $t0 = 1, caso contrario $t0 = 0
82                  bne     $t0, $t1, testaC        #se media A >= media C, pula pra testaC
83                  la      $t0, vm
84                  sw      $t6, 0($t0)             #media A foi a menor media, coloca em vm
85                  j       vetorD
86
87  testaB:         slt     $t0, $t7, $t8           #media B < media C, $t0 = 1, caso contrario $t0 = 0
88                  bne     $t0, $t1, testaC
89                  la      $t0, vm
90                  sw      $t7, 0($t0)             #media B foi a menor media, coloca em vm
91                  j       vetorD
92
93  testaC:         slt     $t0, $t8, $t6           #media C < media A, $t0 = 1, caso contrario $t0 = 0
94                  bne     $t0, $t1, end           #deu erro (nenhuma media foi a menor)
95                  la      $t0, vm
96                  sw      $t8, 0($t0)             #media C foi a menor media, coloca em vm
97
```

Linhas 73-74 – reseta o valor de $t1 e incrementa 1 ($t1 = 1)

Linhas 76-78 – carrega os valores das três médias

Linha 79 – Comparação mediaA < mediaB, cujo resultado é guardado em $t0

Linha 80 -- Comparação entre $t0 e $t1. Caso eles não sejam iguais, isto é, mediaA é maior ou igual a mediaB, ocorre um salto para o label testaB, pois já sabemos que a mediaA não é a menor das médias

Linhas 81-85 – Se mediaA for menor, o código segue em frente e faz a mesma comparação com mediaC. Se a mediaA continuar sendo a menor, o endereço de VM é carregado em $t0, e o valor da média é posto dentro de $t0. Feito isso, ocorre um salto para o label vetorD, pois VM já foi descoberto, então não é necessário passar pelas próximas etapas de comparação.

Linhas 87-96 – Como nas linhas anteriores, testando primeiro se mediaB < mediaC. Se isso for verdade, o valor da mediaB é salvo no endereço de VM e ocorre um salto para o label vetorD. Caso contrário, é testado se mediaC < mediaA é verdade. Se for, o valor de mediaC é guardado no endereço de VM, e desta vez não ocorre um salto para vetorD, visto que o label é a parte seguinte do código. Se mediaC < mediaA não for verdade, houve um erro e ocorre um salto para o fim do programa (linha 94), pois o programa concluiu que nenhuma média é a menor.

```
 98  vetorD:         xor     $s2, $s2, $s2          #zera $s2 (i)
 99                  xor     $t2, $t2, $t2          #zera $t2 (flag booleano)
100                  la      $t6, vetD              #carrega end. d
101                  la      $t7, k                 #carrega end. k
102                  lw      $t7, 0($t7)            #carrega valor de k
103                  xor     $t7, $t7, $t7
104                  lw      $t0, 0($t0)            #carrega valor de vm
105                  la      $t3, vetA
106                  la      $t4, vetB
107                  la      $t5, vetC
```

Linhas 98-99 -- $s2 (i) e $t2 são zerados.

Linha 100 – $t6 é sobrescrito com o endereço de vetD.

Linhas 101-102 -- $t7 é sobrescrito com o endereço, e depois o valor, de k.

Linha 103 – reseta o valor de $t7

Linha 104-- $t0 é sobrescrito com o valor de VM.

Linhas 105-107 – Os endereços dos vetores vetA, vetB e vetC são recarregados em $t3, $t4, $t5, respectivamente.

```
108 loopD:          lw      $s3, 0($t3)          #a[i]
109                 lw      $s4, 0($t4)          #b[i]
110                 lw      $s5, 0($t5)          #c[i]
111
112
113 comparaA:       slt     $t2, $t0, $s3        #a[i] > vm?
114                 bne     $t2, $t1, comparaB   #nao, prox comparacao
115                 add     $t8, $s3, $zero      #else $t8 = a[i]
116                 jal     adiciona
117
118 comparaB:       slt     $t2, $t0, $s4        #b[i] > vm?
119                 bne     $t2, $t1, comparaC   #nao, prox comparacao
120                 add     $t8, $s4, $zero      #else $t8 = b[i]
121                 jal     adiciona
122
123 comparaC:       slt     $t2, $t0, $s5        #c[i] > vm?
124                 bne     $t2, $t1, incrementa #nao, incrementa vars
125                 add     $t8, $s5, $zero      #else $t8 = c[i]
126                 jal     adiciona
127
128 incrementa:     addi    $s2, $s2, 1          #i++
129                 addi    $t3, $t3, 4          #a[i++]
130                 addi    $t4, $t4, 4          #b[i++]
131                 addi    $t5, $t5, 4          #c[i++]
132                 bne     $s2, $s0, loopD
```

Linhas 108-110 -- Começo do laço onde é averiguado se o elemento a[i], b[i] e c[i] (todos respectivamente carregados em $s3, $s4 e $s5) deve ser inserido em d[k].

Linhas 113-114 – Comparação a[i] > VM. Se isso não for verdade, ocorre um salto para a próxima comparação (comparaB).

Linhas 115-116 – Se o resultado da comparação for verdadeiro, o valor de a[i] ($s3) é posto em $t8, registrador que guarda temporariamente o valor a ser adicionado em vetD no label adiciona, para o qual ocorre um salto com retorno na próxima instrução (linha 117).

Linhas 118-126 – Os mesmos processos de comparação, saltos, e adição de elementos são feitos para os valores b[i] e c[i]. Na linha 124, ocorre um salto para o label incrementa se nenhum elemento deve ser adicionado a vetD.

Linhas 128-131 – São incrementados os valores de $s2 (i), $t3 (endereço de vetA), $t4 (endereço de vetB) e $t5 (endereço de vetC).

Linha 132 – Ocorre um salto para o início do laço se $s2 (i) for menor que $s0 (n).

```
134 end:            j end
135
136
137 adiciona:       sw      $t8, 0($t6)          #guarda o conteudo de $t8 (elemento a ser adicionado em D) em d[k]
138                 addi    $t7, $t7, 1          #k++
139                 addi    $t6, $t6, 4          #d[k++]
140                 jr      $ra                  #retorna
```

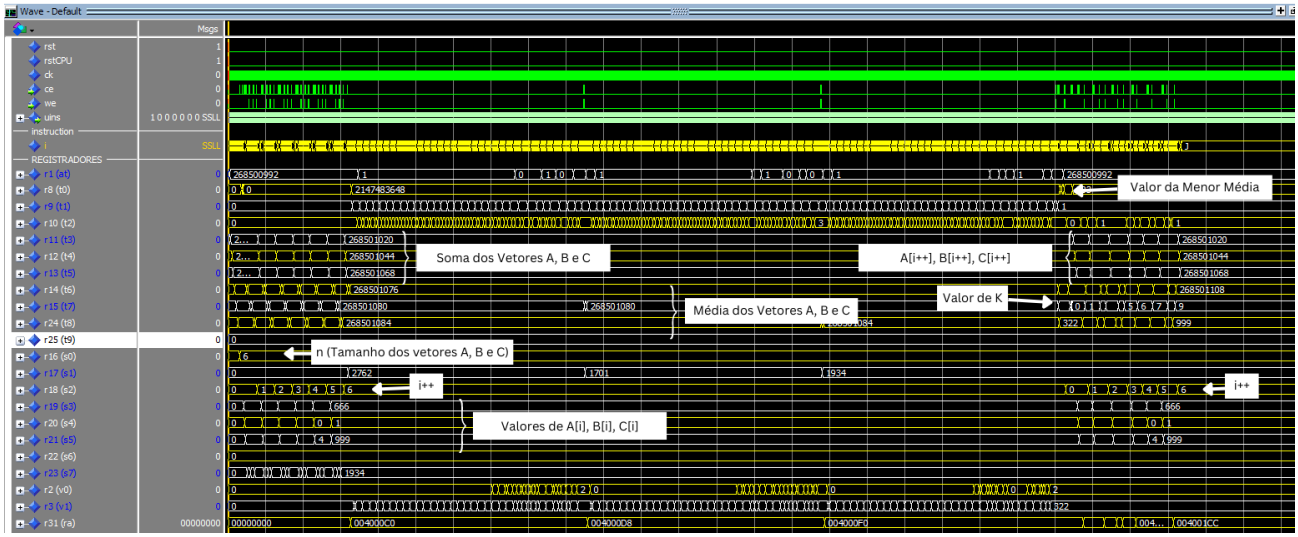
Linha 134 – Com o término do laço, o programa terminou e se mantém preso no label end.

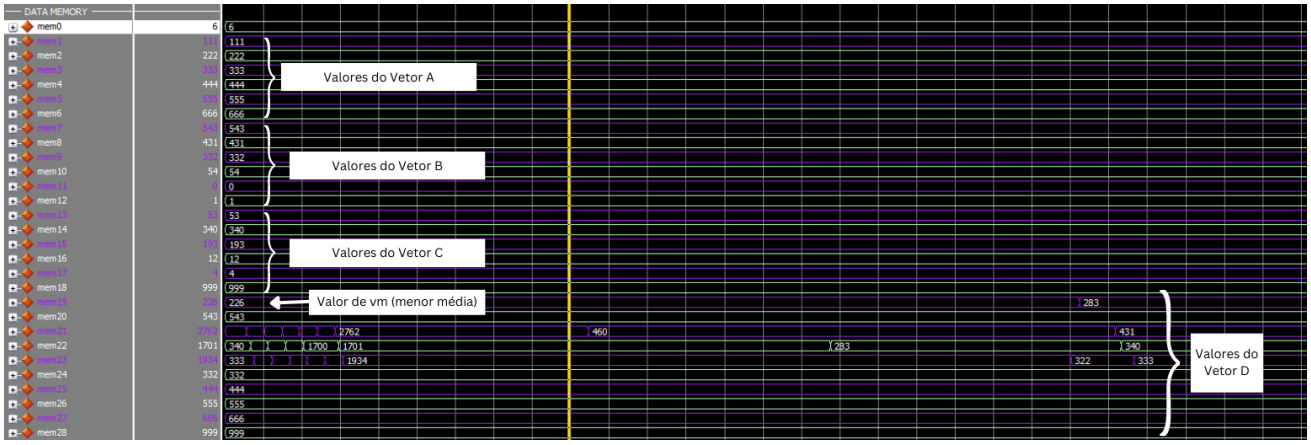
Linha 137 – O conteúdo de $t8, vulgo o elemento a ser adicionado em vetD, é armazenado no endereço atual de vetD ($t6).

Linhas 138-139 -- São incrementados k ($t7), vulgo o número de elementos em vetD, e vetD ($t6).

Linha 140 – Salto que retorna à instrução salva em $ra quando o salto para adiciona foi executado.

Simulação no Modelsim:


Valor da Menor Média
Soma dos Vetores A, B e C
A[i++], B[i++], C[i++]
Valor de K
Média dos Vetores A, B e C
n (Tamanho dos vetores A, B e C)
i++
i++
Valores de A[i], B[i], C[i]


Valores do Vetor A
Valores do Vetor B
Valores do Vetor C
Valor de vm (menor média)
Valores do Vetor D