Laboratório 1

Assembly Risc V

Gabriel de Castro Dias, 211055432
João Marcos Melo Monteiro, 130143031
João Victor Pereira Vieira , 211036114
Luiz Henrique Silva de Andrade, 211010430 Sofia
Dy La Fuente Monteiro, 211055530

Grupo 10

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0099 – Organização e Arquitetura de Computadores

Questão 1.2 Exercício a)

Considerações → Main = 5 ciclos de clock (3 jal + 2 jalr)
Sort = 9 ciclos de clock
Swap = 6 ciclos de clock
11 instruções no "for2"
1 instrução no "for1"

Tempo de execução to(n) para vetores já ordenados:
Nesse caso, o procedimento SORT percorrerá o loop "for1" uma única vez, pois os elementos já estão ordenados. O loop "for2" não será executado.

Tempo de execução ti(n) para vetores ordenados inversamente:
Nesse caso, o procedimento SORT percorrerá o loop "for1" n vezes, pois cada iteração é necessária para colocar o maior elemento no final do vetor. O loop "for2" será executado em todas as iterações, pois a condição de entrada será verdadeira.

O número de iterações do loop "for1" é dado por n, e o número de iterações do loop "for2" é dado por (n-1) + (n-2) + ... + 2 + 1, que é a soma dos primeiros n-1 números naturais. Essa soma pode ser expressa como n*(n-1/)2. Também consideramos que para todas as iterações, entraremos nas instruções "SWAP".

Tempo de execução para t(n) para vetores quaisquer (estimativa):
Aqui temos que simular que as iterações irão passar pelo "for1" e "for2", já que não sabemos ao certo o vetor a ser ordenado.

Equações:

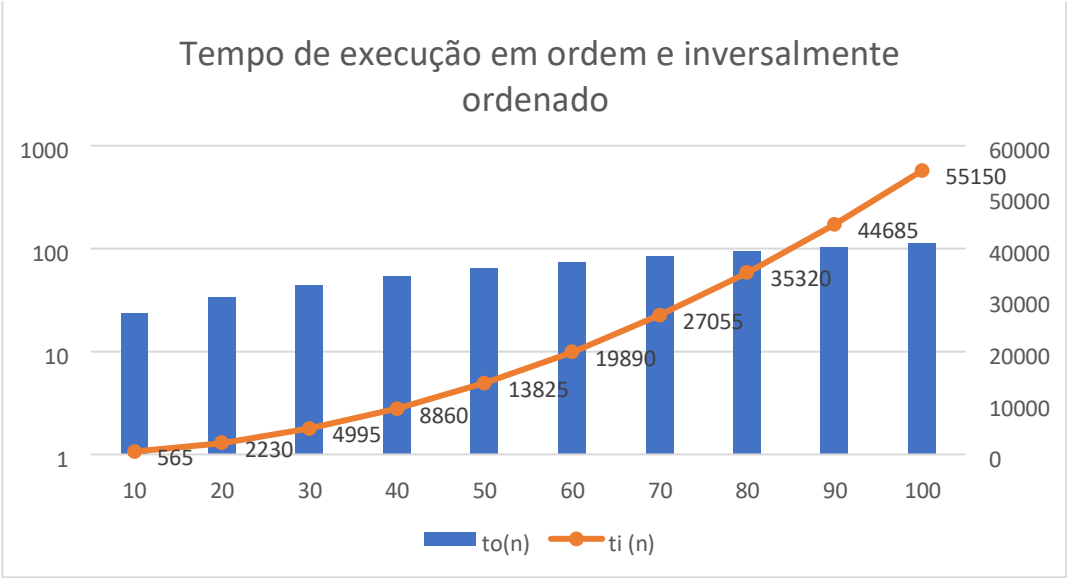
F(to(n)) = Main + Sort + n (número de iterações do loop "for1")
= 5 + 9 + n
≈ n

F(ti(n)) = Main + Sort + n(loop “for1”) + 11* n*(n-1)/2 ciclos de clock(loop “for2”) + Swap =
= 5 + 9 + n + 11 *n*(n-1)/2 + n*6
≈ n + 11 *n*(n-1)/2 + n*6

F(t(n)) = Main + Sort + for1(n)+ for2(n)
= Main + Sort + (n-1) * for1(n-1) + for2(n-1) + (n-1) * to(n)
= 5 + 9 + (n-1) * n(n-1) + 11 * n*(n-1)/2 + n*6 + (n-1) * n
≈ (n-1) * n(n-1) + 11 * n*(n-1)/2 + n*6 + (n-1) * n

OBS: Como “Main” e “Sort” são executados no começo do programa e possuem poucas instruções, podemos ignora-las afim de reduzir a expressão.

Exercício b)



Para a função to(n), o tempo de execução não muda muito, já que a função cresce linearmente (em um vetor “n”, ele vai verificar a ordem “n” vezes, já que todos vão estar em seus devidos lugares).

Já para ti(n), cada elemento fora de ordem precisará passar por todos os outros elementos, isso faz com que a função cresça exponencialmente, já que quanto mais elementos no vetor, mais instruções são executadas para ordena-los corretamente.

OBS: Isso considerando que pra cada instrução, temos 1 ciclo de clock (CPI = 1).

Questão 2.2

Site usado: Compiler Explorer (https://godbolt.org) . Compilaçao feita em -O0

Serão listadas a seguir todas as modificações necesserárias para implementar o codigo:

- Adicionar .data e .text
- Implementar o SHOW para printar
- Mudar o printf por SHOW
- Mudar o putchar por SHOW
- Colocar main no começo do código
- Remover jr ra e adicionar a syscall de fechar o programa na main
- No sort.s o vetor foi salvo em a0, e o .eqv foi salvo em a1 ja na conversão de C para RISC-V gcc 13.1.0 não é assim, o vetor foi dividido em duas partes, bits mais significativos (%hi) e bits menos significativos (%lo) e depois somados em um addi e salvo no a0, depois esse a0 foi gravado na memória no endereço -36(s0). Então para pegar este valor fizemos um lw t0, -36(s0) para o t0 ter os inteiros assim como no sort.s. Tambem fizemos um .eqv N 30 (quantidade de numeros que serão printados por linha) e usamos li t1,N para carregar o valor de N no t1, colocamos t2 como 0 e chamamos a função SHOW que substitui as instruções printf e putchar que vem no código compilado pelo Compiler explorer
- Para resolver o problema do loop infinito, foi preciso mudar a label que o show chama de .L2 para .L3, e remover as linhas de comparaçao e a chamada do SHOW na label .L2

A seguir pode ser conferido o codigo implementado:

```
1  .eqv N 30  # quantos numeros vao ser printados por linha
2
3  .data
4  v:
5          .word   9
6          .word   2
7          .word   5
8          .word   1
9          .word   8
10         .word   2
11         .word   4
12         .word   3
13         .word   6
14         .word   7
15         .word   10
16         .word   2
17         .word   32
18         .word   54
19         .word   2
20         .word   12
21         .word   6
22         .word   3
23         .word   1
24         .word   78
25         .word   54
26         .word   23
27         .word   1
28         .word   54
29         .word   2
30         .word   65
31         .word   3
32         .word   6
33         .word   55
34         .word   31
```

```
.LC0:
        .string "%d\t"  # ?

.text

main:   addi    sp,sp,-16       # libera 4 espaços na memoria
        sw      ra,12(sp)       # grava ra em sp + 12
        sw      s0,8(sp)        # grava s0 em sp + 8
        addi    s0,sp,16        # s0 = sp + 16
        li      a1,30           # a1 = 31
        lui     a5,%hi(v)       # adiciona os high bits em a5
        addi    a0,a5,%lo(v)    # adiciona em a0 os bits superiores e inferiores
        call    show            # chama show
        li      a1,30           # a1 = 30
        lui     a5,%hi(v)       # adiciona os high bits em a5
        addi    a0,a5,%lo(v)    # adiciona em a0 os bits superiores e inferiores
        call    sort            # chama sort
        li      a1,30           # a1 = 30
        lui     a5,%hi(v)       # adiciona os high bits em a5
        addi    a0,a5,%lo(v)    # adiciona em a0 os bits superiores e inferiores
        call    show            # chama show
        li      a5,0            # a5 = 0
        mv      a0,a5           # a0 = a5
        lw      ra,12(sp)       # carrega sp + 12 e salva em ra
        lw      s0,8(sp)        # carrega sp + 8 e salva em s0
        addi    sp,sp,16        # sp = sp + 16
        #jr      ra             # jump to address on ra

        li a7, 10 # finaliza o programa
        ecall

show:   addi    sp,sp,-48       # sp = sp - 48
        sw      ra,44(sp)       # grava ra em sp + 44
        sw      s0,40(sp)       # grava s0 em sp + 40
        addi    s0,sp,48        # s0 = sp + 48
        sw      a0,-36(s0)      # grava a0 em s0 - 36
        sw      a1,-40(s0)      # grava a1 em s0 - 40
        sw      zero,-20(s0)    # grava zero em s0 - 20
        j       .L3             # jump to .L2

.L3:    lw      a5,-20(s0)      # carrega s0 - 20 e salva em a5
        slli    a5,a5,2         # a5 = a5 * 4
        lw      a4,-36(s0)      # carrega s0 - 36 e salva em a4
        add     a5,a4,a5        # a5 = a4 + a5
        lw      a5,0(a5)        # carrega a5 + 0 e salva em a5
        mv      a1,a5           # a1 = a5
        lui     a5,%hi(.LC0)    # adiciona os high bits em a5
        addi    a0,a5,%lo(.LC0) # adiciona em a0 os bits superiores e inferiores
        call    SHOW            # chama SHOW para printar
        lw      a5,-20(s0)      # carrega s0 - 20 e salva em a5
        addi    a5,a5,1         # a5 = a5 + 1
        sw      a5,-20(s0)      # grava a5 em s0 - 20

.L2:    lw      a4,-20(s0)      # carrega s0 - 20 e salva em a4
        lw      a5,-40(s0)      # carrega s0 - 40 e salva em a5
        #blt     a4,a5,.L3      # if a4 < a5  go to .L3
        li      a0,10           # a0 = 10
        #call    SHOW           # chama SHOW para printar
        nop                     # nada
        lw      ra,44(sp)       # carrega sp + 44 e salva em ra
        lw      s0,40(sp)       # carrega sp + 40 e salva em s0
        addi    sp,sp,48        # sp sp + 48
        jr      ra              # jump to address on ra
```

```
 98
 99 swap:   addi    sp,sp,-48       # sp = sp - 48
100         sw      s0,44(sp)       # grava s0 em sp + 44
101         addi    s0,sp,48        # s0 = sp + 48
102         sw      a0,-36(s0)      # grava a0 em s0 - 36
103         sw      a1,-40(s0)      # grava a1 em s0 - 40
104         lw      a5,-40(s0)      # carrega s0 - 40 e salva em a5
105         slli    a5,a5,2         # a5 = a5 * 4
106         lw      a4,-36(s0)      # carrega s0 - 36 e salva em a4
107         add     a5,a4,a5        # a5 = a4 + a5
108         lw      a5,0(a5)        # carrega a5 + 0 e salva em a5
109         sw      a5,-20(s0)      # grava a5 em s0 - 20
110         lw      a5,-40(s0)      # carrega s0 - 40 e salva em a5
111         addi    a5,a5,1         # a5 = a5 + 1
112         slli    a5,a5,2         # a5 = a5 * 4
113         lw      a4,-36(s0)      # carrega s0 - 36 e salva em a4
114         add     a4,a4,a5        # a4 = a4 + a5
115         lw      a5,-40(s0)      # carrega s0 = 40 e salva em a5
116         slli    a5,a5,2         # a5 = a5 * 4
117         lw      a3,-36(s0)      # carrega s0 - 36 e salva em a3
118         add     a5,a3,a5        # a5 = a3 + a5
119         lw      a4,0(a4)        # carrega a4 + 0 e salva em a4
120         sw      a4,0(a5)        # grava a4 em a5 + 0
121         lw      a5,-40(s0)      # carrega s0 - 40 e salva em a5
122         addi    a5,a5,1         # a5 = a5 + 1
123         slli    a5,a5,2         # a5 = a5 * 4
124         lw      a4,-36(s0)      # carrega s0 - 36 e salva em a4
125         add     a5,a4,a5        # a5 = a4 + a5
126         lw      a4,-20(s0)      # carrega s0 - 20 e salva em a4
127         sw      a4,0(a5)        # grava a4 em a5 + 0
128         nop                     # nada
129         lw      s0,44(sp)       # carrega sp + 44 e salva em s0
130         addi    sp,sp,48        # sp = sp + 48
131         jr      ra              # jump to address on ra
```

```
133 sort:   addi    sp,sp,-48       # sp = sp - 48
134         sw      ra,44(sp)       # grava ra em sp + 44
135         sw      s0,40(sp)       # grava s0 em sp + 40
136         addi    s0,sp,48        # s0 = sp + 48
137         sw      a0,-36(s0)      # grava a0 em s0 - 36
138         sw      a1,-40(s0)      # grava a1 em s0 - 40
139         sw      zero,-20(s0)    # grava zero em s0 - 20
140         j       .L6             # jump to .L6
141
142 .L10:   lw      a5,-20(s0)      # carrega s0 - 20 e salva em a5
143         addi    a5,a5,-1        # a5 = a5 - 1
144         sw      a5,-24(s0)      # grava a5 em s0 - 24
145         j       .L7             # jump to .L7
146
147 .L9:    lw      a1,-24(s0)      # carrega s0 - 24 e salva em a1
148         lw      a0,-36(s0)      # carrega s0 - 36 e salva em a0
149         call    swap            # chama swap
150         lw      a5,-24(s0)      # carrega s0 - 24 e salva em a5
151         addi    a5,a5,-1        # a5 = a5 - 1
152         sw      a5,-24(s0)      # grava a5 em s0 - 24
153
```

```
154 .L7:     lw      a5,-24(s0)      # carrega s0 - 24 e saçva e, a5
155          blt     a5,zero,.L8     # if a5 < zero  go to .L8
156          lw      a5,-24(s0)      # carrega s0 - 24 e salva em a5
157          slli    a5,a5,2         # a5 = a5 * 4
158          lw      a4,-36(s0)      # carrega s0 - 36 e salva em a4
159          add     a5,a4,a5        # a5 = a4 + a5
160          lw      a4,0(a5)        # carrega s5 + 0 e salva em a4
161          lw      a5,-24(s0)      # carrega s0 - 24 e salva em a5
162          addi    a5,a5,1         # a5 = a5 + 1
163          slli    a5,a5,2         # a5 = a5 * 4
164          lw      a3,-36(s0)      # carrega s0 - 36 e salva em a3
165          add     a5,a3,a5        # a5 = a3 + a5
166          lw      a5,0(a5)        # carrega a5 + 0 e salva em a5
167          bgt     a4,a5,.L9       # if a4 > a5  go to .L9
168
169 .L8:     lw      a5,-20(s0)      # carrega s0 - 20 e salva em a5
170          addi    a5,a5,1         # a5 = a5 + 1
171          sw      a5,-20(s0)      # grava a5 em s0 - 20
172
173 .L6:     lw      a4,-20(s0)      # carrega s0 - 20 e salva em a4
174          lw      a5,-40(s0)      # carrega s0 - 40 e salva em a5
175          blt     a4,a5,.L10      # if a4 < a5  go to .L10
176          nop                     # nada
177          nop                     # nada
178          lw      ra,44(sp)       # carrega sp + 44 e salva em ra
179          lw      s0,40(sp)       # carrega sp + 40 e salva em s0
180          addi    sp,sp,48        # sp = sp + 48
181          jr      ra              # jump to address on ra
182
```

```
183
184 SHOW:  lw t0, -36(s0)  # t0 = a0
185         li t1,N         # t1 = 30
186         mv t2,zero      # t2 = 0
187
188 loop1:  beq t2,t1,fim1  # if  t2 == t1  go to fim1
189         li a7,1         # sycall de print int
190         lw a0,0(t0)     # carrega t0 + 0 e salva em a0
191         ecall           # chama a syscall para printar a0
192         li a7,11        # syscall de PrintChar
193         li a0,9         # a0 = 9 / a0 = character to print (only lowest byte is considered)
194         ecall           # chama a syscall
195         addi t0,t0,4    # t0 = t0 + 4
196         addi t2,t2,1    # t2 = t2 + 1
197         j loop1         # jump to loop1
198
199 fim1:   li a7,11        # syscall de PrintChar  \n basicamente
200         li a0,10        # a0 = 10 / a0 = character to print (only lowest byte is considered)
201         ecall           # chama a syscall
202         ret             # retorna a quem chamou  (SHOW)
```

Questão 2.3

Todos não possuem .text e nem .data

sortMOD -O0: ja da pra montar direto, vem com .word

sortMOD -O1: não da pra montar direto, vem sem .word e com label .LANCHOR0 faltando
- compila após colocarmos a label com as .words de comparação

sortMOD -O2: mesma coisa do sortMOD -O1

sortMOD -O3: mesma coisa da sortMOD -O1 porém possui labels usadas várias vezes
- (loop1) e (fim1)
- no total são 3 labels de loop1 e fim1, mudando elas para
loop1 / fim1
loop2 / fim2
loop3 / fim3

atençao, as ordens de show, swap, etc estão todas trocadas

ele compila

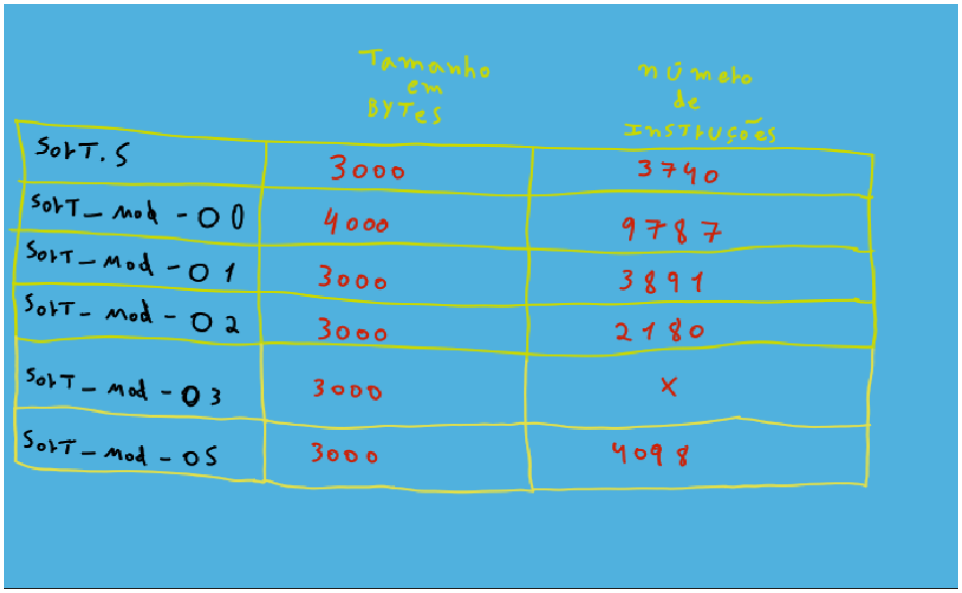
sortMOD -OS: mesma coisa do sortMOD -O1

!!! Tamanho !!!

sortMOD -O0: 4000 bytes
sortMOD -O1: 3000 bytes
sortMOD -O2: 3000 bytes
sortMOD -O3: 3000 bytes
sort.s: 3000 bytes
sortMOD -OS: 3000 bytes

!!! Quant instruçoes !!!

sort.s = 3740
sortMOD -O0: 9787
sortMOD -O1: 3891
sortMOD -O2: 2180
sortMOD -O3: x
sortMOD -OS: 4098



	Tamanho em Bytes	número de instruções
SORT.S	3000	3740
SORT_mod-00	4000	9787
SORT_mod-01	3000	3891
SORT_mod-02	3000	2180
SORT_mod-03	3000	X
SORT_mod-05	3000	4098

Questão 3

3.1) Nessa questão devemos criar um procedimento SORTEIO que dado um inteiro N ele sorteia coordenadas aleatórias.

Para implementar esse algoritmo foi necessário armazenar no vetor L as coordenadas da loja (que são fixas nos pontos X = 155 e Y = 115) e foi necessário criar um vetor C: que alocasse um spaco de 160 bits (20 casas com cada coordenada X e Y ocupando 4 bits cada uma).

Para sortear as coordenadas aleatórias, foi necessario chamar syscall número 42, que gera um número aleatório dentro de um intervalo. Para X o intervalo correspondeu de 0 à 311 e para Y o intervalo correspondeu de 0 à 231 (Com o máximo correspondendo a 1 a mais que o tamanho da tela para a conta ficar correta).

O código implementado pode ser conferido abaixo:

```
.include "MACROSv21.s"
.data

N: .word 19                     # Numero de casas dos clientes + 1 loja
L: .word 155, 115               # Posição da loja (C0) fixa no meio da tela
C: .space 160                   # Espaço de bits correspondentes a 2 coordenadas x 20 casas x bytes

.text

MAIN:
        la s1, C                # s1 = vetor C
        li, t5, 20              # Numero fixo para fazer a conta
        li, t0, 0               # Contador
        jal SORTEIO             # Chama função de sortear

        |

###############################################
# Função de sortear a coordenada das casas #
###############################################

SORTEIO:

        beq t0, t5, FIM         # Caso o contador seja do tamanho do vetor, encerra
        li t2, 311              # Valor maximo permitido para X
        li t3, 231              # Valor maximo permitido para Y

### X ###                                   # Argumentos para a chamada de numeros aleatorios:
        li, a0, 0               # Valor minimo a0 = 0
        mv a1, t2               # Valor maximo a1 = 311

        li a7, 42               # Chama rotina de numero aleatorio
        ecall

        mv t1, a0               # Move o valor de a0 (numero aleatorio) e armazena em t1
        sw t1, 0(s1)            # Armazena o valor do vetor

        addi s1, s1, 4          # Incrementa o vetor

### Y ###                                   # Argumentos para a chamada de numeros aleatorios:
        li, a0, 0               # Valor minimo: a0 = 0
        mv a1, t3               # Valor maximo: a1 = 231

        li a7, 42               # Chama rotina de numero aleatorio
        ecall

        mv t1, a0               # Move o valor de a0 (numero aleatorio) e armazena em t1
        sw t1, 0(s1)            # Armazena o valor do vetor

        addi s1, s1, 4          # Incrementa o vetor
        addi t0, t0, 1          # Incrementa o contador

        j SORTEIO               # Retorna o loop

FIM:
        ret                     # Retorna main
```

Questão 3.2

Nessa questão devemos criar um procedimento DESENHA que desenha os pontos aleatorios criados anteriormente e printar sua posição no bitmap display. A loja deve ser printada ao centro de vermelho e as casas devem ser printadas de verde com seu codigo ASCII.

Para fazer isso, primeiro foi necessário carregar na main as váriaveis que serão usadas:

```
MAIN:
        la s1, C                # s1 = vetor C
        li, t5, 20              # Numero fixo para fazer a conta
        li, t0, 0               # Contador
        jal SORTEIO             # Chama função de sortear

        la t0, N
        lw a0, 0(t0)
        la a1, L
        jal DESENHA             # Chama função de desenhar (Pinta tela de branco e printa a coordenada da loja)
```

Em seguida, é necessário preencher a tela do bitmap de branco, carregando o endereço de cada pixel e escrevendo na memoria o pixel da cor branca (codigo em hexadecimal: 0xFFFFFFF)

```
87  ################################
88  # Preencher a tela de branco #
89  ################################
90
91          li t1,0xFF000000        # endereco inicial da Memoria VGA - Frame 0
92          li t2,0xFF012C00        # endereco final
93          li t3,0xffffffff        # cor branca
94
95  LOOP:
96          beq t1,t2,DESENHA_LOJA  # Se for o último endereço então sai do loop
97          sw t3,0(t1)             # escreve a word na memória VGA
98          addi t1,t1,4            # soma 4 ao endereço
99          j LOOP
100
```

Depois de preenchida a tela de branco, precisamos printar a loja. Para isso iremos carregar o codigo ASCII da letra L em a0, a posição X da loja em a1, a posição Y da loja em a2, em a3 iremos carregar o código hexadecimal para “fundo vermelho” e “letra preta” (0x0700) e em a4 iremos carregar o frame 0. Feito isso, iremos chamar a rotina syscall 111 que printa na tela codigo ASCII mais cor.

O codigo implementado pode ser conferido abaixo:

```
126 ############################
127 # Printa a loja no centro #
128 ############################
129
130 DESENHA_LOJA:
131
132         mv s1,a1                # Move para s1 o valor fixo da loja (L)
133
134         addi a0,s2,76           # Adiciona a a0 o codigo ascii da letra L (loja)
135         lw a1,0(s1)             # Carrega a posicao X loja
136         lw a2,4(s1)             # Carrega a posicao Y da loja
137         li a3,0x0700            # Fundo vermelho e letra preta
138         li a4,0                 # Frame
139
140         li a7,111               # Chama procedimento de mostrar na tela
141         ecall
142
143         ret                     # Retorna main
144
```

Para printar as casas, foi necessario carregar na main novamente os valores das variaveis:

```
17  MAIN:
18          la s1, C                # s1 = vetor C
19          li, t5, 20              # Numero fixo para fazer a conta
20          li, t0, 0               # Contador
21          jal SORTEIO             # Chama funcao de sortear
22
23          la t0, N
24          lw a0, 0(t0)
25          la a1, L
26          jal DESENHA             # Chama funcao de desenhar (Pinta tela de branco e printa a coordenada da loja)
27
28          la t0, N
29          lw a0, 0(t0)
30          la a1, C
31          jal DESENHA_CASAS       # Chama funcao de desenhar as casas
```

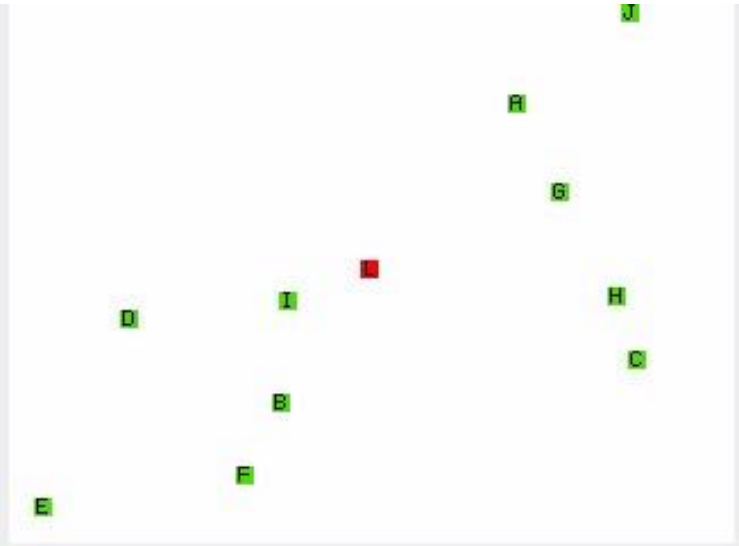
Para desenhar as casinhas no bitmap display seguimos uma lógica parecida com a que usamos para printar a loja. Carregamos em a0 o valor ASCII das letras do alfabeto (começando por A), em a1 carregamos a coordenada X da casa e em a2 a coordenada Y. Ja em a3, usamos o codigo em hexadecimal para printar “fundo verde” e “letra preta” (codigo 0x3200) e em a4 carregamos o frame 0.

Chamamos o syscall 111 novamente e incrementamos o vetor C, que contém todas as coordenadas (x, y) das casa para percorrer todo o vetor e printar todas as casas.

O codigo da implementação pode ser conferido abaixo:

```
120 ##########################
121 # Printa todas as casas #
122 ##########################
123
124 DESENHA_CASAS:
125
126         mv t2, a0               # Move para t2 o valor de N
127         la s1, C                # Move para s1 o vetor das casas (C)
128         li t0, 0                # Contador
129         li t1, 65               # Adiciona em t1 o codigo ascii da letra A
130
131 LOOP_CASAS:
132
133         add a0, s2, t1          # Adiciona a a0 o codigo ascii das letras (começando por A)
134         lw a1,0(s1)             # Carrega a posicao X da casa
135         lw a2,4(s1)             # Carrega a posicao Y da casa
136         li a3,0x3200            # Fundo verde e letra preta
137         li a4,0                 # Frame
138
139         li a7,111               # Chama procedimento de mostrar na tela
140         ecall
141
142         addi s1,s1,8            # Pula o endereço das 1° cordenadas (x, y) e vai para as proximas
143         beq t0, t2, FINAL       # Se o contador for igual ao N, encerra loop
144         addi t0, t0, 1          # Se não, incrementa contador
145         addi t1, t1, 1          # Incrementa t1 para a proxima letra no codigo ascii
146
147         j LOOP_CASAS            # Retorna loop
148
149 FINAL:
150         ret                     # Retorna main
```

No bitmap é possível conferir a seguinte imagem:



Questão 3.3

Para essa questão era necessário criar um procedimento chamado ROTAS que desenha no bitmap as linhas que ligam a loja a todas as casas e todas as casas a todas as outras casas. Além disso era necessario criar uma matriz euclidiana, seguindo a equação demonstrada nas instruções do laboratório.

Para essa implementação foi necessário dividir o procedimento em 2: ROTAS, para printar as rotas da loja a todas as casas e ROTAS2 para printar de todas as casas para todas as outras casas.

Primeiro, carregamos na main as variáveis:

```
17  MAIN:
18
19          la t0, N
20          lw a0, 0(t0)
21          la s1, L
22          la s2, C
23          jal ROTAS               # Chama funcao de desenhar as rotas da loja
24
25          li t5, 8
26          li t4, 18               # Valor fixo em 18
27          li t2, 18               # Valor variave para comparacao 2
28          li t1, 0                # Contador
29          li t3, 0                # Segundo contador
30          la s1, C
31          la s2, C
32          jal ROTAS_2             # Chama funcao de senhar rota das casas
33
```

Para printar da loja a todas as casas, foi necessário criar contadores e carregar em t0 o valor fixo de 9 para comparar e sair do loop. Carregamos em a0 e a1 as coordenadas X e Y da Loja e em a2 e a3 as coordenadas X e Y da primeira casa. Em a4 carregamos o codigo hexadecimal de preto (codigo 0x00) e em a5 o frame 0.

Chamamos a syscall 147, que desenha uma linha ligando duas coordenadas. E por fim, iteramos o vetor C que contém a coordenada das casas para fazer um loop e printar a linha da loja para todas as casas.

O codigo da implementação pode ser conferido abaixo:

```
179 ################################
180 # Printa a rota da loja para  #
181 #    todas as outras casas    #
182 ################################
183
184 ROTAS:
185         li t1, 0              # Contador
186         li t0, 9             # Valor fixo para comparacao
187
188 LOOP_1:
189
190         lw a0, 0(s1)           # Carrega a posicao X da loja
191         lw a1, 4(s1)           # Carrega a posicao Y da loja
192         lw a2, 0(s2)           # Carrega a posicao X da casa
193         lw a3, 4(s2)           # Carrega a posicao Y da casa
194         li a4, 0x00            # Linha preta
195         li a5, 0               # Frame
196
197         li a7, 147             # Chama procedimento de desenhar linha na tela
198         ecall
199
200         beq t1, t0, SAIDA      # Se o contador for igual ao valor fixo, pula para LOOP 2
201
202         addi t1, t1, 1         # Se nao, incrementa contador
203         addi s2, s2, 8         # Pula o endereco das 1Â° cordenadas (x, y) e vai para as proximas
204         j LOOP_1
205
206 SAIDA:
207         ret
```

Para printar das casas para todas as casas, foi implementado um algoritmo parecido com o usado para printar a loja. Porém, dessa vez usaremos 2 loops: Um para percorrer todo o vetor C carregado em s2 de uma só vez e outro loop para iterar o vetor C carregado em s1, todas as vezes que s2 foi percorrido totalmente.

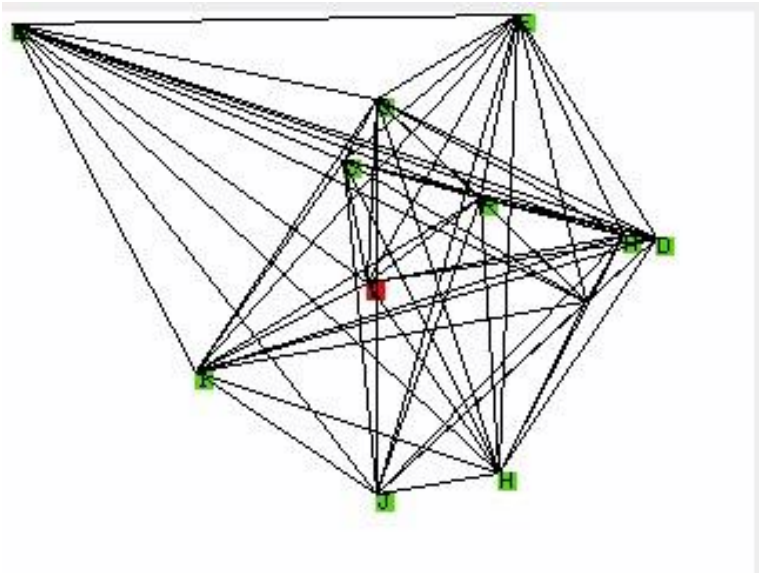
A cada vez que s2 for iterado, chamaremos a mesma rotina syscall de desenhar linha usada no algoritmo acima.

Quando todo o vetor C em s1 for percorrido, retornamos a main. O codigo pode ser conferido abaixo:

```
209 ##############################
210 # Printa a rota de todas as  #
211 # casas para todas as outras #
212 ##############################
213
214 # Definicao dos registradores:
215 # t0 = Fixo em 1 para fazer a subtracao
216 # t1 = Contador do Rotas 2 (serve para contar se percorremos todo o vetor
217 # t2 = Decrementa o valor usado de referenia para o jump Fim loop
218 # t3 = Contador do fim loops (serve para contar quantas vezes passamos)
219 # t4 = registrador fixo em 19
220 # t5 = registrador para incrementar os vetores
221
222 ROTAS_2:
223
224         li, t0, 1               # Fixo para conta
225         lw a0, 0(s1)            # Carrega a posicao X da casa
226         lw a1, 4(s1)            # Carrega a posicao Y da casa
227         lw a2, 8(s2)            # Carrega a posicao X da casa
228         lw a3, 12(s2)           # Carrega a posicao Y da casa
229         li a4, 0x00             # Linha preta
230         li a5, 0                # Frame
231
232         li a7, 147              # Chama procedimento de desenhar linha na tela
233         ecall
234
235         beq t1, t2, FIM_LOOPS   # Se o contador for igual ao valor fixo, pula para LOOP 2
236
237         addi t1, t1, 1          # Se nao, incrementa contador
238         addi s2, s2, 8          # Pula o endereco das 1Â° cordenadas (x, y) e vai para as proximas
239         j ROTAS_2
```

```
239         j ROTAS_2
240
241
242 FIM_LOOPS:
243
244         la s2, C                # Reseta o registrador s2 com o valor do vetor c
245         li t1, 0                # Reseta contador
246
247         sub t2, t2, t0          # Subitrai o valor fixo
248         add s2, s2, t5          # Adiciona o valor de t5 (8) em s2
249         addi t5, t5, 8          # Incrementa o valor de t5
250
251         beq t3, t4, END         # Se t3 for igual a t2, pula para end
252
253         addi t3, t3, 1          # Se nao, incrementa contador
254         addi s1, s1, 8          # Incrementa o vetor em s1
255
256         j ROTAS_2               # Pula para o loop 2
257
258 END:
259         ret                     # Retorna main
260
```

O codigo visto no bitmap pode ser conferido abaixo:



Para printar a matriz euclidiana das casas e da loja, foi necessário implementar a equação descrita nas instruções do laboratório 1. Primeiro carregamos na main as variáveis a serem utilizadas

```
45
46  MAIN:
47          li t3, 0
48          li t0, 0
49          li t1, 9
50          la s1, C
51          la s2, C
52          li t4, 0
53          la s6, D
54          jal LOJA_MATRIZ         # Chama funcao de criar matriz euclidiana, começando pela loja (L)
55
```

Primeiro é printada os valores (em float) da loja para todas as casas, então para isso carregamos as coordenadas X e Y da loja em a1 e a2, e as coordenadas X e Y da primeira casa em a3 e a4.

Em seguida subtraimos os X da loja com o X da casa e armazenamos sem s3. Fazemos o mesmo para os Y e armazenamos em s4. Em seguida, calculados os quadrados da diferença ja calculados e armazenamos.
Somamos esses dois quadrados e passamos o valor encontrado para float e por fim, calculamos a raiz quadrada desse valor e armazenamos no vetor D (que armazena todas as distancias das matrizes).

O codigo para printar as matrizes euclidianas das casas para as outras casas, é similiar com o demonstrado a cima, com a única diferença que incrementamos em s2 o vetor C que será percorrido totalmente e incrementamos em s1 o vetor C para ser percorrido uma vez a cada percorrida completa do vetor s2.

O codigo da implementação da matriz para N = 10 pode ser conferido abaixo:

```
257  ########################################
258  #  Funcao de criar matriz euclidiana #
259  ########################################
260
261  # Definicão das variaveis
262  # t0 = contador da matriz 2
263  # t1 = valor fixo em 9
264  # t3 = contador do vetor das letras
265  # t4 = contador do fim matriz
266  # S1 = C (vetor lento)
267  # S2 = C (vetor rapido)
268  # s6 = D (vetor das matrizes)
```

(Definição das variáveis)

```
270  ####################
271  # Matriz da loja #
272  ####################
273
274  LOJA_MATRIZ:
275
276          la s1, L                # Adicioba no s1 o vetor da loja
277          la s2, C                # Adiciona no s2 o vetor das casas
278
279          la a0, loja             # Carrega o caractere no registrador a0
280
281          li a7, 4                # Chama rotina de printar string
282          ecall
283
284          li a7,11                # Printa o \n
285          li a0,10
286          ecall
288  LOOP_MATRIZ:
289
290          lw a1, 0(s1)            # Carrega a posicao X da loja
291          lw a2, 4(s1)            # Carrega a posicao Y da loja
292          lw a3, 0(s2)            # Carrega a posicao X da casa
293          lw a4, 4(s2)            # Carrega a posicao Y da casa
294
295          sub s3, a1, a3          # Subtracoes dos X's da casa e loja
296          sub s4, a2, a4          # Subtracoes dos Y's da casa e loja
297
298          mul s3, s3, s3          # Quadrado da diferenca dos X's
299          mul s4, s4, s4          # Quadrado da diferenca dos Y's
300
301          add s5, s3, s4          # Soma dos quadrados
302          fcvt.s.w f1, s5         # Passa s5 para float
303
304          fsqrt.s  fa0,f1         # Realiza a raiz quadrada
305
306          fsw fa0, 0(s6)          # Adiciona no vetor matrizes da loja o float
307          addi s6, s6, 4          # Incrementa o vetor para proxima posicao
308
309          li a7, 2                # Printa o Float
310          ecall
```

```
        ecall

        li a7,11              # Printa espaco
        li a0,9
        ecall

        beq t0, t1, RESTAURA_MATRIZ   # Se o contador for igual ao valor fixo, pula RESTAURA_MATRIZ

        addi t0, t0, 1          # Se nao, incrementa contador
        addi s2, s2, 8          # Pula o endereco das 1° cordenadas (x, y) e vai para as proximas

        j LOOP_MATRIZ

RESTAURA_MATRIZ:

        li t3, 0                # Reseta os valores para
        li t0, 0                # printar agora as
        li t1, 9                # matrizes das casas
        la s1, C
        la s2, C
        li t4, 0
        la s6, D

        j MATRIZ_CASA
```

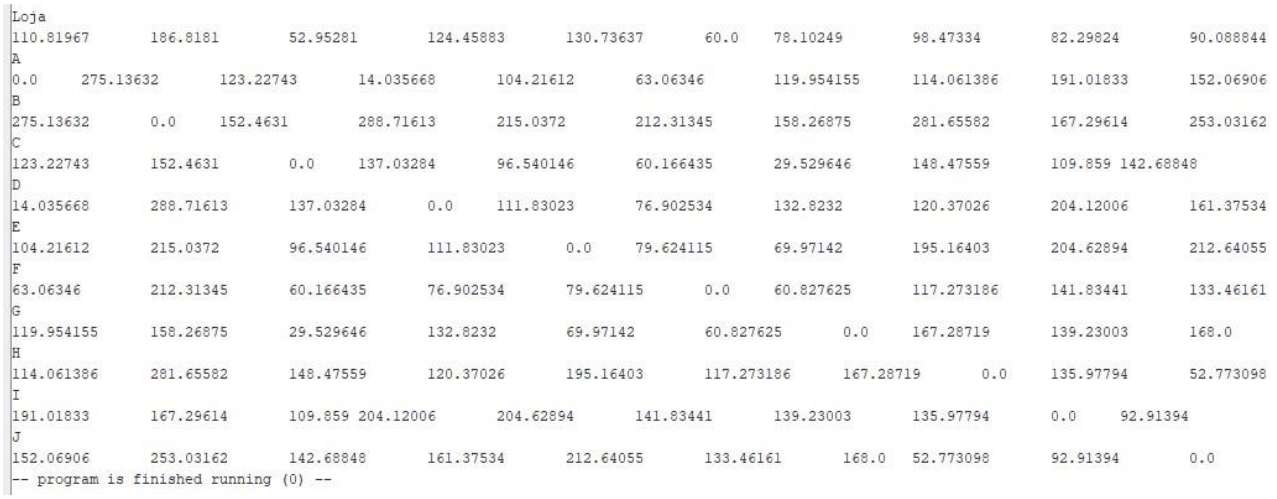
Agora o codigo para printar a matriz das casas:

```
##########################
# Matrizes das casa #
##########################

MATRIZ_CASA:

        li a7,11              # Printa o \n
        li a0,10
        ecall

        la a5, letras
        add a5, a5, t3        # Calcula o endereco do caractere
        lb a5, 0(a5)          # Carrega o caractere no registrador a0

        li a7, 11             # Chama rotina de printar caracter
        mv a0, a5
        ecall

        li a7,11              # Printa o \n
        li a0,10
        ecall
```

```
357 MATRIZ_2:
358
359         lw a1, 0(s1)            # Carrega a posicao X da casa
360         lw a2, 4(s1)            # Carrega a posicao Y da casa
361         lw a3, 0(s2)            # Carrega a posicao X da casa
362         lw a4, 4(s2)            # Carrega a posicao Y da casa
363
364         sub s3, a1, a3          # Subtracoes dos X's da casa
365         sub s4, a2, a4          # Subtracoes dos Y's da casa
366
367         mul s3, s3, s3          # Quadrado da diferenca dos X's
368         mul s4, s4, s4          # Quadrado da diferenca dos Y's
369
370         add s5, s3, s4          # Soma dos quadrados
371         fcvt.s.w f1, s5         # Passa s5 para float
372
373         fsqrt.s  fa0,f1         # Realiza a raiz quadrada
374
375         fsw fa0,0(s6)           # Salva o resultado no vetor de matriz
376         addi, s6, s6, 4         # Incrementa o vetor
377
378         li a7, 2                # Printa o Float
379         ecall
380
381         li a7,11                # Printa espaco
382         li a0,9
383         ecall
384
385         beq t0 t1, FIM_MATRIZ   # Se o contador for igual ao valor fixo, pula FIM_MATRIZ
386
387         addi t0, t0, 1          # Se nao, incrementa contador
388         addi s2, s2, 8          # Pula o endereco das 1° cordenadas (x, y) e vai para as proximas
389
390         j MATRIZ_2              # Retorna loop
```

```
393 FIM_MATRIZ:
394
395         li s3,0
396         li s4,0                 # Reseta variaveis
397         li s5,0
398         li t0,0
399
400         beq t1, t4, END_MATRIZ  # Se t1 for igual ao t4, encerra
401         addi t4, t4, 1          # Se não, incrementa
402
403         la s2, C                # Reseta s2, que eh a onde ta o vetor que anda mais rapido
404         addi  t3,t3,1           # Incrementa o vetor de letras
405         addi s1, s1, 8          # Anda uma casa do vetor que anda mais lento
406
407         j MATRIZ_CASA           # Retorna loop
408
409
410 END_MATRIZ:
411         ret
```

Na imagem a seguir será mostrado como ficou a matriz euclidiana para N = 10 (para valores alatórios)



Nessa imagem cada linha está com seu respectivo nome escrito em cima, e as colunas correspondem a ordem do alfabeto: Primeira coluna corresponde a A, segunda coluna a B e assim por diante.

Questão 3.4

Para essa questão é necessário implementar um procedimento ORDENA, que desenha no bitmap display, em linhas vermelhas, a melhor rota a ser percorrida.

Conseguimos implementar o codigo que ordena, dentro do procedimento de ROTAS e das matrizes euclidianas. Implementamos o codigo de modo que ele nunca considere a si mesmo como a melhor rota, porém não foi possível implementar um algoritmo que verifica se ja passamos por uma casa. Então, caso a casa A possua como menor distancia a casa B e a casa B possua como menor distancia a casa A, o condigo de repete por não possuir a condição de comparação.

Para implementar essa condição, criamos um vetor que armazenaria o codigo de cada letra (A=0, B=1,C=2...) e usariamos ele para comparar com o vetor que ordena.

O codigo da nossa tentativa pode ser visto a segui:

```
45
46          li t0, 1000
47          fcvt.s.w ft0, t0
48          li t3, 0
49          li t0, 0
50          fcvt.s.w ft2, t0
51          li t1, 9
52          la s1, C
53          la s2, C
54          li t4, 0
55          li t5, 8
56          la s6, D
57          li s7, 0
58          li s9, 1
59          jal LOJA_MATRIZ         # Chama funcao de criar matriz euclidiana, começando pela loja (L)
60
61          li a7, 10
62          ecall                   # Funcao de encerrar programa
```

Função main

```
261 ########################################
262 #  Funcao de criar matriz euclidiana #
263 ########################################
264
265 # Definição das variaveis
266 # t0 = contador da matriz 2
267 # t1 = valor fixo em 9
268 # t3 = contador do vetor das letras
269 # t2 = salva o contador t0 para saber qual casa pertence a coordenada
270 # t4 = contador do fim matriz
271 # t5 = Valor fixo em 8 para fazer a conta
272 # t6 = Variavel temporaria de comparação
273 # S1 = C (vetor lento)
274 # S2 = C (vetor rapido)
275 # s3, s4, s5 = Variaveis de armazenamento da conta
276 # s6 = D (vetor das matrizes)
277 # s7 = valor fixo em 0 (para comparação no ORDENA 2)
278 # s8 = Contem a coordenada da melhor rota
279 # s9 = valor fixo  em 1
280 # ft0 = contem o valor da menor distancia
281 # ft2 = valor fixo em 0.0 (para não considerar a menor rota ele mesmo)
282
```

```
283 ######################
284 # Matriz da loja #
285 ######################
286
287 LOJA_MATRIZ:
288
289         la s1, L                # Adicioba no s1 o vetor da loja
290         la s2, C                # Adiciona no s2 o vetor das casas
291
292         la a0, loja             # Carrega o caractere no registrador a0
293
294         li a7, 4                # Chama rotina de printar string
295         ecall
296
297         li a7,11                # Printa o \n
298         li a0,10
299         ecall
300
301 LOOP_MATRIZ:
302
303         lw a1, 0(s1)            # Carrega a posicao X da loja
304         lw a2, 4(s1)            # Carrega a posicao Y da loja
305         lw a3, 0(s2)            # Carrega a posicao X da casa
306         lw a4, 4(s2)            # Carrega a posicao Y da casa
307
308         sub s3, a1, a3          # Subtracoes dos X's da casa e loja
309         sub s4, a2, a4          # Subtracoes dos Y's da casa e loja
310
311         mul s3, s3, s3          # Quadrado da diferenca dos X's
312         mul s4, s4, s4          # Quadrado da diferenca dos Y's
313
314         add s5, s3, s4          # Soma dos quadrados
315         fcvt.s.w f1, s5         # Passa s5 para float
316
317         fsqrt.s  fa0,f1         # Realiza a raiz quadrada
318
319         fsw fa0, 0(s6)          # Adiciona no vetor matrizes da loja o float
320         addi s6, s6, 4          # Incrementa o vetor para proxima posicao
321
322         li a7, 2                # Printa o Float
323         ecall
324
325         li a7,11                # Printa espaco
326         li a0,9
327         ecall
328
329         flt.s t6, fa0, ft0      # Caso o valor armazenado em fa0 seja menor que s7, flag = 1
330         beq t6, s9, ORDENA      # Caso o valor armazenado na flag seja 1, pula ORDENA
331 AB:
332         beq t0, t1, RESTAURA_MATRIZ   # Se o contador for igual ao valor fixo, pula RESTAURA_MATRIZ
333
334         addi t0, t0, 1          # Se nao, incrementa contador
335         addi s2, s2, 8          # Pula o endereco das 1° cordenadas (x, y) e vai para as proximas
336         li t6, 0                # Reseta valor de t6
337
338         j LOOP_MATRIZ
339
340 RESTAURA_MATRIZ:
341
342 ###################################
343 # Printa melhor rota de vermelho #
344 ###################################
```

```
345
346         lw a0, 0(s1)            # Carrega a posicao 1X
347         lw a1, 4(s1)            # Carrega a posicao 1Y
348         lw a2, 0(s8)            # Carrega a posicao 2X
349         lw a3, 4(s8)            # Carrega a posicao 2Y
350         li a4, 0x07             # Linha vermelha
351         li a5, 0                # Frame
352
353         li a7, 147              # Chama procedimento de desenhar linha na tela
354         ecall
355 #---------------------------------
356
357         li t3, 0
358         li t0, 1000
359         fcvt.s.w ft0, t0        # Reseta os valores para
360         li t0, 0                # printar agora as
361         li t1, 9                # matrizes das casas
362         la s2, C
363         li t4, 0
364         li s8, 0
365         la s6, D
366         la s1, C                # Incrementa o vetor C em s1
367         mv t3, t2               # Incrementar o vetor das letras
368         mul t2, t2, t5          # Carrega a proxima casa (Contador x 8)
369         add s1, s1, t2          # Acrescenta valor em s1
370         addi t2, t2, 0          # Reseta t2
371
372         j MATRIZ_CASA
373
```

```
374 ######################
375 # Matrizes das casa #
376 ######################
377
378 MATRIZ_CASA:
379
380         li a7,11              # Printa o \n
381         li a0,10
382         ecall
383
384         la a5, letras
385         add a5, a5, t3        # Calcula o endereco do caractere
386         lb a5, 0(a5)          # Carrega o caractere no registrador a0
387
388         li a7, 11             # Chama rotina de printar caracter
389         mv a0, a5
390         ecall
391
392         li a7,11              # Printa o \n
393         li a0,10
394         ecall
395
396 MATRIZ_2:
397
398         lw a1, 0(s1)          # Carrega a posicao X da casa
399         lw a2, 4(s1)          # Carrega a posicao Y da casa
400         lw a3, 0(s2)          # Carrega a posicao X da casa
401         lw a4, 4(s2)          # Carrega a posicao Y da casa
402
403         sub s3, a1, a3        # Subtracoes dos X's da casa
404         sub s4, a2, a4        # Subtracoes dos Y's da casa
405
406         mul s3, s3, s3        # Quadrado da diferenca dos X's
407         mul s4, s4, s4        # Quadrado da diferenca dos Y's
```

```
408
409         add s5, s3, s4          # Soma dos quadrados
410         fcvt.s.w f1, s5         # Passa s5 para float
411
412         fsqrt.s  fa0,f1         # Realiza a raiz quadrada
413
414         fsw fa0,0(s6)           # Salva o resultado no vetor de matriz
415         addi, s6, s6, 4         # Incrementa o vetor
416
417         li a7, 2                # Printa o Float
418         ecall
419
420         li a7,11                # Printa espaco
421         li a0,9
422         ecall
423
424         flt.s t6, fa0, ft0       # Caso o valor armazenado em fa0 seja menor que s7, flag = 1
425         beq t6, s9, ORDENA_CASA  # Caso o valor armazenado na flag seja 1, pula ORDENA
426 CD:
427         beq t0 t1, FIM_MATRIZ  # Se o contador for igual ao valor fixo, pula FIM_MATRIZ
428
429         addi t0, t0, 1          # Se nao, incrementa contador
430         addi s2, s2, 8          # Pula o endereco das 1° cordenadas (x, y) e vai para as proximas
431         li t6, 0                # Reseta t6
432
433         j MATRIZ_2              # Retorna loop
434
435
436 FIM_MATRIZ:
```

```
438 ####################################
439 # Printa melhor rota de vermelho #
440 ####################################
441
442         lw a0, 0(s1)            # Carrega a posicao 1X
443         lw a1, 4(s1)            # Carrega a posicao 1Y
444         lw a2, 0(s8)            # Carrega a posicao 2X
445         lw a3, 4(s8)            # Carrega a posicao 2Y
446         li a4, 0x07             # Linha vermelha
447         li a5, 0                # Frame
448
449         li a7, 147              # Chama procedimento de desenhar linha na tela
450         ecall
451 #---------------------------------
452
453         li t3, 0
454         li t0, 1000
455         fcvt.s.w ft0, t0        # Reseta os valores para
456         li t0, 0                # printar agora as
457         li t1, 9                # matrizes das casas
458         la s2, C
459         li s8, 0
460         la s6, D
461         beq t1, t4, END_MATRIZ  # Se t1 for igual ao t4, encerra
462         addi t4, t4, 1          # Se não, incrementa
463
464         la s1, C                # Incrementa o vetor C em s1
465         mv t3, t2               # Incrementar o vetor das letras
466         mul t2, t2, t5          # Carrega a proxima casa (Contador x 8)
467         add s1, s1, t2          # Acrescenta valor em s1
468         addi t2, t2, 0          # Reseta t2
469
470         j MATRIZ_CASA             # Retorna loop
471
```

```
473 END_MATRIZ:
474         ret
475
476
477 ######################
478 # Caixeiro Viajante #
479 ######################
480
481 ORDENA:
482         fmv.s ft0, fa0          # ft0 possui a melhor rota
483         mv s8, s2               # Coordenada X da melhor rota
484         mv t2, t0
485         j AB
486
487 ORDENA_CASA:
488
489         feq.s t6, fa0, ft2      # Se fa0 for igual a ft2 significa que a casa X esta comparando a rota com ela mesma
490         beq t6, s7, CONTINUA    # Se não tiver, substitui os valores de fa0 e s8
491         j CD
492
493 CONTINUA:
494         fmv.s ft0, fa0          # ft0 possui a melhor rota
495         mv s8, s2               # Coordenada X da melhor rota
496         mv t2, t0
497         j CD
498
```

Questão 3.5