

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include "vecinos.h"
5  #include <math.h>
6  #ifndef VACIO
7  #define VACIO 0
8  #endif
9  #define NOVACIO 1
10 int AbrirArchivo(char* path, FILE **fp) {
11     *fp = fopen(path, "r");
12     if(*fp == NULL)
13         return 1;
14     return 0;
15 }
16
17 int CrearArchivoOutput(char *archivo, FILE **fop) {
18     if((*fop = fopen(archivo, "w+b")) == NULL) {
19         /*Tiro error de apertura de archivo*/
20         return 1;
21     }
22     return 0;
23 }
24
25 void CerrarArchivo(FILE *fp) {
26     fclose(fp);
27 }
28
29 unsigned int getPosicion(unsigned int i, unsigned int j, unsigned int N){
30     return i*N+j;
31 }
32
33 void MostrarUso(){
34     printf("Uso:\n");
35     printf("conway -h\nconway -V\nconway i M N inputfile [-o outputprefix]\n");
36     printf("Opciones:\n-h, --help\t Imprime este mensaje\n");
37     printf(" -V, --version\t Da la versiÃ³n del programa\n");
38     printf(" -o\tPrefijo de los archivos de salida\n");
39 }
40
41 void ImprimirError(char* msj){
42     fprintf(stderr, "%s", msj);
43 }
44
45 void InvocacionIncorrecta(){
46     ImprimirError("InvocaciÃ³n del programa incorrecta\n");
47 }
48
49 void GrabarArchivoSalida(unsigned char *matriz, unsigned int M, unsigned int N,
50 char *prefijoSalida, unsigned int j) {
51     unsigned int a = 0, b = 0, c = 0, d = 0;
52     FILE *archivoSalida;
53     int nDigits;
54     char buf[5];
55     if(j > 0)
56         nDigits = floor(log10(j)) + 1;
57     else
58         nDigits = 1;
59
60     char *nombreArchivoSalida = malloc(strlen(prefijoSalida) + (sizeof(char)*5) + si
61 zeof(char)*nDigits);
62     strcpy(nombreArchivoSalida, prefijoSalida);
63     strcat(nombreArchivoSalida, "_");
64     sprintf(buf, "%u", j);
65     strcat(nombreArchivoSalida, buf);
66     strcat(nombreArchivoSalida, ".pbm");
67     if(CrearArchivoOutput(nombreArchivoSalida, &archivoSalida) == 0){
68         printf("Grabando estado %s\n", nombreArchivoSalida);
69         //Grabar Encabezado de archivo
70         fprintf(archivoSalida, "P1\n");
71         fprintf(archivoSalida, "%u %u\n", M*16, N*16);

```

```

70 //Recorrer Matriz y grabar archivo
71 for(a = 0; a < M; a++){
72     for (d = 0; d < 16; d++){
73         for(b = 0; b < N; b++) {
74             //Grabo posicion
75             for (c = 0; c < 16; c++)
76                 fprintf(archivoSalida, "%u", matriz[getPosicion(a,b,N)]);
77         }
78         fprintf(archivoSalida, "\n");
79     }
80 }
81 }
82 CerrarArchivo(archivoSalida);
83 free(nombreArchivoSalida);
84 }
85
86 void abortarCargaMatriz(char* mensaje, unsigned char* matriz, FILE* archivo) {
87     ImprimirError(mensaje);
88     free(matriz);
89     CerrarArchivo(archivo);
90     fclose(stdout);
91     fclose(stderr);
92     exit(-1);
93 }
94
95 unsigned char obtenerNumeroFila(unsigned char* matriz, FILE *archivo, unsigned int M) {
96     unsigned char fila = 0;
97     fscanf(archivo, "%hhu", &fila);
98     if (fgetc(archivo) != ' ') {
99         abortarCargaMatriz("\n\nError en el formato del archivo de entrada\n\n", matriz, archivo);
100     }
101     int proximoCaracter = fgetc(archivo);
102     if (!((proximoCaracter >= '0') & (proximoCaracter <= '9'))) {
103         abortarCargaMatriz("\n\nError en el formato del archivo de entrada\n\n", matriz, archivo);
104     }
105     fseek(archivo, -1, SEEK_CUR);
106     if (fila > M-1) {
107         abortarCargaMatriz("\n\nUn numero de fila en el archivo de entrada es mayor que el maximo permitido\n\n", matriz, archivo);
108     }
109     return fila;
110 }
111
112 unsigned char obtenerNumeroColumna(unsigned char* matriz, FILE *archivo, unsigned int N) {
113     unsigned char columna = 0;
114     fscanf(archivo, "%hhu", &columna);
115     if (fgetc(archivo) != '\n') {
116         abortarCargaMatriz("\n\nError en el formato del archivo de entrada\n\n", matriz, archivo);
117     }
118     int proximoCaracter = fgetc(archivo);
119     if (!((!((proximoCaracter >= '0') & (proximoCaracter <= '9'))) & (proximoCaracter != EOF))) {
120         abortarCargaMatriz("\n\nError en el formato del archivo de entrada\n\n", matriz, archivo);
121     }
122     if (proximoCaracter != EOF) {
123         fseek(archivo, -1, SEEK_CUR);
124     }
125     if (columna > N-1) {
126         abortarCargaMatriz("\n\nUn numero de columna en el archivo de entrada es mayor que el maximo permitido\n\n", matriz, archivo);
127     }
128     return columna;
129 }
130
131 void cargarMatriz(unsigned char* matriz, FILE *archivo, unsigned int M, unsigned

```

```

132     int N) {
133         rewind(archivo);
134         while (!feof(archivo)) {
135             unsigned char fila = obtenerNumeroFila(matriz, archivo, M);
136             unsigned char columna = obtenerNumeroColumna(matriz, archivo, N);
137             matriz[fila*N + columna] = NOVACIO;
138         }
139     }
140 int main(int argc, char *argv[]) {
141     unsigned int i = 0, M = 0, N = 0, a = 0, n = 0, v, pos, j, b;
142     char* prefijoSalida;
143     FILE *fp;
144     unsigned char* matriz;
145     //Lectura y carga de argumentos
146     if (argc < 5){
147         if(argc == 2){
148             if (strcmp(argv[1], "-V") == 0 || strcmp(argv[1], "--version") == 0) {
149                 printf("Version 1.0\n");
150                 return 0;
151             } else if (strcmp(argv[1], "-h") == 0 || strcmp(argv[1], "--help") == 0) {
152                 MostrarUso();
153                 return 0;
154             } else {
155                 InvocacionIncorrecta();
156                 MostrarUso();
157                 return 1;
158             }
159         } else {
160             InvocacionIncorrecta();
161             MostrarUso();
162             return 1;
163         }
164     } else if (argc == 5 || argc == 7) {
165         i = (unsigned int)atoi(argv[1]); //cantidad de ciclos
166         if (i == 0){
167             ImprimirError("La cantidad de estados (i) debe ser un n mero mayor a 0.\r\n");
168             return 1;
169         }
170         M = (unsigned int)atoi(argv[2]);
171         if (M == 0){
172             ImprimirError("La cantidad de filas (M) debe ser un n mero mayor a 0.\r\n");
173             return 1;
174         }
175         N = (unsigned int)atoi(argv[3]);
176         if (N == 0){
177             ImprimirError("El tama o de las columnas (N) debe ser un n mero mayor a 0.\r\n");
178             return 1;
179         }
180     } else {
181         InvocacionIncorrecta();
182         MostrarUso();
183         return 1;
184     }
185     //Argumentos opcionales
186     prefijoSalida = argv[4];
187     if(argc == 7){
188         if(strcmp(argv[5], "-o") == 0){
189             prefijoSalida = argv[6];
190             if(strcmp(prefijoSalida, "") == 0){
191                 InvocacionIncorrecta();
192                 ImprimirError("El prefijo de los archivos de salida no puede ser vacio.");
193                 MostrarUso();
194                 return 1;
195             }
196         }
197     } else {
198         InvocacionIncorrecta();
199         MostrarUso();
200         return 1;
201     }

```

```
202     }
203 }
204 //Carga inicial
205 n = AbrirArchivo(argv[4], &fp);
206 if(n == 1){
207     //No se pudo abrir el archivo
208     fprintf(stderr, "El archivo %s no pudo ser abierto.\r\n", argv[4]);
209     return 1;
210 }
211
212 // Creo matriz
213 matriz = malloc(M*N*sizeof(unsigned char));
214
215 for(a = 0; a < M*N; a++){
216     matriz[a] = VACIO;
217 }
218
219 //Cargar matriz con archivo de entrada
220 printf("%s/n", "Leyendo estado inicial...");
221 cargarMatriz(matriz, fp, M, N);
222
223 GrabarArchivoSalida(matriz, M, N, prefijoSalida, 0);
224 //Recorro las iteraciones
225 for(j = 0; j < i; j++){
226     unsigned char* copia = malloc(M*N*sizeof(unsigned char));
227     for (a = 0; a < M; a++){
228         for(b = 0; b < N; b++){
229             //Recorro los casilleros
230             v = vecinos(matriz, a, b, M, N);
231             pos = getPosicion(a, b, N);
232             copia[pos] = matriz[pos];
233             if(matriz[pos] == VACIO){
234                 //Estaba muerta
235                 if (v == 3){
236                     copia[pos] = NOVACIO;
237                 }
238             } else {
239                 //Estaba viva
240                 if (v < 2 || v > 3){
241                     //Muere
242                     copia[pos] = VACIO;
243                 }
244             }
245         }
246     }
247     free(matriz);
248     matriz = copia;
249     GrabarArchivoSalida(matriz, M, N, prefijoSalida, j+1);
250 }
251 printf("Listo\r\n");
252 free(matriz);
253 CerrarArchivo(fp);
254 fclose(stdout);
255 fclose(stderr);
256 return 0;
257 }
258
259
```

```

1  #ifndef VACIO
2  #define VACIO 0
3  #endif
4  unsigned int vecinos(unsigned char *a,unsigned int i, unsigned int j, unsigned i
nt M, unsigned int N){
5      unsigned int v = 0;
6      //Modelo Toroidal
7      if(i != 0){
8          //Chequeo arriba
9          if (a[(i-1)*N+j] != VACIO){
10             v++;
11         }
12         //arriba a la izquierda
13         if (j != 0){
14             if (a[(i-1)*N+(j-1)] != VACIO){
15                 v++;
16             }
17         } else {
18             if (a[(i-1)*N+(N-1)] != VACIO){
19                 v++;
20             }
21         }
22         if (j != (N-1)){
23             //arriba a la derecha
24             if (a[(i-1)*N+(j+1)] != VACIO){
25                 v++;
26             }
27         } else {
28             if (a[(i-1)*N] != VACIO){
29                 v++;
30             }
31         }
32     } else {
33         //Chequeo "arriba"
34         if (a[(M-1)*N+j] != VACIO){
35             v++;
36         }
37         //arriba a la izquierda
38         if (j != 0){
39             if (a[(M-1)*N+(j-1)] != VACIO){
40                 v++;
41             }
42         } else {
43             if (a[(M-1)*N+(N-1)] != VACIO){
44                 v++;
45             }
46         }
47         //arriba a la derecha
48         if (j != (N-1)){
49             if (a[(M-1)*N+(j+1)] != VACIO){
50                 v++;
51             }
52         } else {
53             if (a[(M-1)*N] != VACIO){
54                 v++;
55             }
56         }
57     }
58     if (i != (M-1)){
59         //Chequeo abajo
60         if (a[(i+1)*N+j] != VACIO){
61             v++;
62         }
63         //abajo a la izquierda
64         if (j != 0){
65             if (a[(i+1)*N+(j-1)] != VACIO){
66                 v++;
67             }
68         } else {
69             if (a[(i+1)*N+(N-1)] != VACIO){
70                 v++;

```

```
71     }
72 }
73 //abajo a la derecha
74 if (j != (N-1)){
75     if (a[(i+1)*N+(j+1)] != VACIO){
76         v++;
77     }
78 } else {
79     if (a[(i+1)*N] != VACIO){
80         v++;
81     }
82 }
83 } else {
84     //Chequeo abajo
85     if (a[j] != VACIO){
86         v++;
87     }
88     //abajo a la izquierda
89     if (j != 0){
90         if (a[j-1] != VACIO){
91             v++;
92         }
93     } else {
94         if (a[N-1] != VACIO){
95             v++;
96         }
97     }
98     //abajo a la derecha
99     if (j != (N-1)){
100         if (a[j+1] != VACIO){
101             v++;
102         }
103     } else {
104         if (a[0] != VACIO){
105             v++;
106         }
107     }
108 }
109 //Chequeo izquierda
110 if (j != 0){
111     if (a[i*N+(j-1)] != VACIO){
112         v++;
113     }
114 } else {
115     if (a[i*N+(N-1)] != VACIO){
116         v++;
117     }
118 }
119 //Chequeo derecha
120 if (j != (N-1)){
121     if (a[i*N+(j+1)] != VACIO){
122         v++;
123     }
124 } else {
125     if (a[i*N] != VACIO){
126         v++;
127     }
128 }
129 return v;
130 }
```

```

1  #include <mips/regdef.h>
2  .text
3  .align 2
4  .global vecinos
5  .ent  vecinos
6  #define V 0
7  #define GP 8
8  #define FP 12
9  #define A 16
10
11 #define I 20
12 #define J 24
13 #define M 28
14 #define N 32
15 #ifndef VACIO
16 #define VACIO 0
17 #endif
18 vecinos:
19     #ABI
20     subu    sp,sp,16
21     sw      $fp,FP(sp)
22     sw      gp,GP(sp)
23     move    $fp,sp
24
25     sw      a0,A(sp)
26     sw      a1,I(sp)
27     sw      a2,J(sp)
28     sw      a3,M(sp)
29
30     sw      zero,V(sp)
31     #Empieza la función
32     lw      t0,I(sp)
33     beq     t0,zero,esprimerafila
34     # i != 0
35     lw      t0,I(sp)
36     addu    t1,t0,-1
37     lw      t0,N(sp)
38     mult    t1,t0
39     mflo    t1
40     lw      t0,J(sp)
41     addu    t1,t1,t0
42     lw      t0,A(sp)
43     addu    t0,t1,t0
44     lbu     t0,0(t0) #t0 = a[(i-1)*N+j]
45     beq     t0,zero,verarrizq
46     lw      t0,V(sp)
47     addu    t0,t0,1
48     sw      t0,V(sp)
49 verarrizq:
50     lw      t0,J(sp)
51     beq     t0,zero,verarrizqprimcol
52     lw      t0,I(sp)
53     addu    t1,t0,-1
54     lw      t0,N(sp)
55     mult    t1,t0
56     mflo    t1
57     lw      t0,J(sp)
58     addu    t1,t1,t0
59     lw      t0,A(sp)
60     addu    t0,t1,t0
61     addu    t0,t0,-1
62     lbu     t0,0(t0) #t0 = a[(i-1)*N+j-1]
63     beq     t0,zero,verarrder
64     lw      t0,V(sp)
65     addu    t0,t0,1
66     sw      t0,V(sp)
67     b       verarrder
68 verarrizqprimcol:
69     lw      t0,I(sp)
70     addu    t1,t0,-1
71     lw      t0,N(sp)

```

```

72      mult      t1,t0
73      mflo      t1
74      lw        t0,N(sp)
75      addu      t1,t1,t0
76      lw        t0,A(sp)
77      addu      t0,t1,t0
78      addu      t0,t0,-1
79      lbu       t0,0(t0) #t0 = a[(i-1)*N+N-1]
80      beq       t0,zero,verarrder
81      lw        t0,V(sp)
82      addu      t0,t0,1
83      sw        t0,V(sp)
84  verarrder:
85      lw        t0,N(sp)
86      addu      t1,t0,-1
87      lw        t0,J(sp)
88      beq       t0,t1,verarrderultcol
89      lw        t0,I(sp)
90      addu      t1,t0,-1
91      lw        t0,N(sp)
92      mult      t1,t0
93      mflo      t1
94      lw        t0,J(sp)
95      addu      t1,t1,t0
96      lw        t0,A(sp)
97      addu      t0,t1,t0
98      addu      t0,t0,1
99      lbu       t0,0(t0) #t0 = a[(i-1)*N+j+1]
100     beq       t0,zero,verultimafila
101     lw        t0,V(sp)
102     addu      t0,t0,1
103     sw        t0,V(sp)
104     b         verultimafila
105  verarrderultcol:
106     lw        t0,I(sp)
107     addu      t1,t0,-1
108     lw        t0,N(sp)
109     mult      t1,t0
110     mflo      t1
111     lw        t0,A(sp)
112     addu      t0,t1,t0
113     lbu       t0,0(t0) #t0 = a[(i-1)*N]
114     beq       t0,zero,verultimafila
115     lw        t0,V(sp)
116     addu      t0,t0,1
117     sw        t0,V(sp)
118     b         verultimafila
119  esprimerafila:
120     lw        t0,M(sp)
121     addu      t1,t0,-1
122     lw        t0,N(sp)
123     mult      t1,t0
124     mflo      t1
125     lw        t0,J(sp)
126     addu      t1,t1,t0
127     lw        t0,A(sp)
128     addu      t0,t1,t0
129     lbu       t0,0(t0) #t0 = a[(M-1)*N+j]
130     beq       t0,zero,verarribaizqpf
131     lw        t0,V(sp)
132     addu      t0,t0,1
133     sw        t0,V(sp)
134  verarribaizqpf:
135     lw        t0,J(sp)
136     beq       t0,zero,verarribaizqprimcolprimfila
137     lw        t0,M(sp)
138     addu      t1,t0,-1
139     lw        t0,N(sp)
140     mult      t1,t0
141     mflo      t1
142     lw        t0,J(sp)

```



```

143     addu    t1,t1,t0
144     lw      t0,A(sp)
145     addu    t0,t1,t0
146     addu    t0,t0,-1
147     lbu     t0,0(t0) #t0 = a[(M-1)*N+j-1]
148     beq     t0,zero,verarribaderpf
149     lw      t0,V(sp)
150     addu    t0,t0,1
151     sw      t0,V(sp)
152     b       verarribaderpf
153 verarribaizqprimcolprimfila:
154     lw      t0,M(sp)
155     addu    t1,t0,-1
156     lw      t0,N(sp)
157     mult    t1,t0
158     mflo    t1
159     lw      t0,N(sp)
160     addu    t1,t1,t0
161     lw      t0,A(sp)
162     addu    t0,t1,t0
163     addu    t0,t0,-1
164     lbu     t0,0(t0) #t0 = a[(M-1)*N+N-1]
165     beq     t0,zero,verarribaderpf
166     lw      t0,V(sp)
167     addu    t0,t0,1
168     sw      t0,V(sp)
169 verarribaderpf:
170     lw      t0,N(sp)
171     addu    t1,t0,-1
172     lw      t0,J(sp)
173     beq     t0,t1,verarribaderultcolpf
174     lw      t0,M(sp)
175     addu    t1,t0,-1
176     lw      t0,N(sp)
177     mult    t1,t0
178     mflo    t1
179     lw      t0,J(sp)
180     addu    t1,t1,t0
181     lw      t0,A(sp)
182     addu    t0,t1,t0
183     addu    t0,t0,1
184     lbu     t0,0(t0) #t0 = a[(M-1)*N+j+1]
185     beq     t0,zero,verultimafila
186     lw      t0,V(sp)
187     addu    t0,t0,1
188     sw      t0,V(sp)
189     b       verultimafila
190 verarribaderultcolpf:
191     lw      t0,M(sp)
192     addu    t1,t0,-1
193     lw      t0,N(sp)
194     mult    t1,t0
195     mflo    t1
196     lw      t0,A(sp)
197     addu    t0,t1,t0
198     lbu     t0,0(t0) #t0 = a[(M-1)*N]
199     beq     t0,zero,verultimafila
200     lw      t0,V(sp)
201     addu    t0,t0,1
202     sw      t0,V(sp)
203 verultimafila:
204     lw      t0,M(sp)
205     addu    t1,t0,-1
206     lw      t0,I(sp)
207     beq     t0,t1,verabajoultimafila
208     lw      t0,I(sp)
209     addu    t1,t0,1
210     lw      t0,N(sp)
211     mult    t1,t0
212     mflo    t1
213     lw      t0,J(sp)

```

```

214     addu    t1,t1,t0
215     lw      t0,A(sp)
216     addu    t0,t1,t0
217     lbu     t0,0(t0) #t0 = a[(i+1)*N+j]
218     beq     t0,zero,verabajoizqultfila
219     lw      t0,V(sp)
220     addu    t0,t0,1
221     sw      t0,V(sp)
222 verabajoizqultfila:
223     lw      t0,J(sp)
224     beq     t0,zero,verabajoizqprimeracol
225     lw      t0,I(sp)
226     addu    t1,t0,1
227     lw      t0,N(sp)
228     mult    t1,t0
229     mflo    t1
230     lw      t0,J(sp)
231     addu    t1,t1,t0
232     lw      t0,A(sp)
233     addu    t0,t1,t0
234     addu    t0,t0,-1
235     lbu     t0,0(t0) #t0 = a[(i+1)*N+(j-1)]
236     beq     t0,zero,verabajoderechaultfil
237     lw      t0,V(sp)
238     addu    t0,t0,1
239     sw      t0,V(sp)
240     b       verabajoderechaultfil
241 verabajoizqprimeracol:
242     lw      t0,I(sp)
243     addu    t1,t0,1
244     lw      t0,N(sp)
245     mult    t1,t0
246     mflo    t1
247     lw      t0,N(sp)
248     addu    t1,t1,t0
249     lw      t0,A(sp)
250     addu    t0,t1,t0
251     addu    t0,t0,-1
252     lbu     t0,0(t0) #t0 = a[(i+1)*N+(N-1)]
253     beq     t0,zero,verabajoderechaultfil
254     lw      t0,V(sp)
255     addu    t0,t0,1
256     sw      t0,V(sp)
257 verabajoderechaultfil:
258     lw      t0,N(sp)
259     addu    t1,t0,-1
260     lw      t0,J(sp)
261     beq     t0,t1,verproximafilacol0
262     lw      t0,I(sp)
263     addu    t1,t0,1
264     lw      t0,N(sp)
265     mult    t1,t0
266     mflo    t1
267     lw      t0,J(sp)
268     addu    t1,t1,t0
269     lw      t0,A(sp)
270     addu    t0,t1,t0
271     addu    t0,t0,1
272     lbu     t0,0(t0) #t0 = a[(i+1)*N+(j+1)]
273     beq     t0,zero,verizquierda
274     lw      t0,V(sp)
275     addu    t0,t0,1
276     sw      t0,V(sp)
277     b       verizquierda
278 verproximafilacol0:
279     lw      t0,I(sp)
280     addu    t1,t0,1
281     lw      t0,N(sp)
282     mult    t1,t0
283     mflo    t1
284     lw      t0,A(sp)

```

```

285     addu    t0,t1,t0
286     lbu     t0,0(t0) #t0 = a[(i+1)*N]
287     beq     t0,zero,verizquierda
288     lw      t0,V(sp)
289     addu    t0,t0,1
290     sw      t0,V(sp)
291     b       verizquierda
292 verabajoultimafila:
293     lw      t1,A(sp)
294     lw      t0,J(sp)
295     addu    t0,t1,t0
296     lbu     t0,0(t0) #t0 = a[j]
297     beq     t0,zero,vercerocolanterior
298     lw      t0,V(sp)
299     addu    t0,t0,1
300     sw      t0,V(sp)
301 vercerocolanterior:
302     lw      t0,J(sp)
303     beq     t0,zero,verceroabajodetodo
304     lw      t1,A(sp)
305     lw      t0,J(sp)
306     addu    t0,t1,t0
307     addu    t0,t0,-1
308     lbu     t0,0(t0) #t0 = a[j-1]
309     beq     t0,zero,ultfilaabajoder
310     lw      t0,V(sp)
311     addu    t0,t0,1
312     sw      t0,V(sp)
313     b       ultfilaabajoder
314 verceroabajodetodo:
315     lw      t1,A(sp)
316     lw      t0,N(sp)
317     addu    t0,t1,t0
318     addu    t0,t0,-1
319     lbu     t0,0(t0) #t0 = a[N-1]
320     beq     t0,zero,ultfilaabajoder
321     lw      t0,V(sp)
322     addu    t0,t0,1
323     sw      t0,V(sp)
324 ultfilaabajoder:
325     lw      t0,N(sp)
326     addu    t1,t0,-1
327     lw      t0,J(sp)
328     beq     t0,t1,ver00
329     lw      t1,A(sp)
330     lw      t0,J(sp)
331     addu    t0,t1,t0
332     addu    t0,t0,1
333     lbu     t0,0(t0) #t0 = a[j+1]
334     beq     t0,zero,verizquierda
335     lw      t0,V(sp)
336     addu    t0,t0,1
337     sw      t0,V(sp)
338     b       verizquierda
339 ver00:
340     lw      t0,A(sp)
341     lbu     t0,0(t0) #to = a[0]
342     beq     t0,zero,verizquierda
343     lw      t0,V(sp)
344     addu    t0,t0,1
345     sw      t0,V(sp)
346 verizquierda:
347     lw      t0,J(sp)
348     beq     t0,zero,izqprimeracol
349     lw      t1,I(sp)
350     lw      t0,N(sp)
351     mult    t1,t0
352     mflo    t1
353     lw      t0,J(sp)
354     addu    t1,t1,t0
355     lw      t0,A(sp)

```

```

356     addu    t0,t1,t0
357     addu    t0,t0,-1
358     lbu     t0,0(t0) #t0 = a[i*N+(j-1)]
359     beq     t0,zero,verderecha
360     lw      t0,V(sp)
361     addu    t0,t0,1
362     sw      t0,V(sp)
363     b       verderecha
364 izqprimeracol:
365     lw      t1,I(sp)
366     lw      t0,N(sp)
367     mult    t1,t0
368     mflo    t1
369     lw      t0,N(sp)
370     addu    t1,t1,t0
371     lw      t0,A(sp)
372     addu    t0,t1,t0
373     addu    t0,t0,-1
374     lbu     t0,0(t0) #t0 = a[i*N+(N-1)]
375     beq     t0,zero,verderecha
376     lw      t0,V(sp)
377     addu    t0,t0,1
378     sw      t0,V(sp)
379 verderecha:
380     lw      t0,N(sp)
381     addu    t1,t0,-1
382     lw      t0,J(sp)
383     beq     t0,t1,derultimafila
384     lw      t1,I(sp)
385     lw      t0,N(sp)
386     mult    t1,t0
387     mflo    t1
388     lw      t0,J(sp)
389     addu    t1,t1,t0
390     lw      t0,A(sp)
391     addu    t0,t1,t0
392     addu    t0,t0,1
393     lbu     t0,0(t0) #t0 = a[i*N+(j+1)]
394     beq     t0,zero,finvecinos
395     lw      t0,V(sp)
396     addu    t0,t0,1
397     sw      t0,V(sp)
398     b       finvecinos
399 derultimafila:
400     lw      t1,I(sp)
401     lw      t0,N(sp)
402     mult    t1,t0
403     mflo    t1
404     lw      t0,A(sp)
405     addu    t0,t1,t0
406     lbu     t0,0(t0) #t0 = a[i*N]
407     beq     t0,zero,finvecinos
408     lw      t0,V(sp)
409     addu    t0,t0,1
410     sw      t0,V(sp)
411
412 finvecinos:
413     lw      v0,V(sp)
414     move    sp,$fp
415     lw      $fp,FP(sp)
416     lw      gp,GP(sp)
417     addu    sp,sp,16
418     j       ra
419 .end vecinos

```