

1. (5 puntos) Al realizar la práctica sobre el algoritmo de Tomasulo, un alumno completa las funciones f_lanzamiento_alum.c y f_transferencia_alum.c con el siguiente código (por simplificar, se han omitido algunas líneas):

```
1 void fase_ISS_alum() {
2     ...
3     /** Busca un hueco en la cola */
4
5     if (RB_long < TAM.REORDER) {
6         b = RB_fin;
7     } else {
8         return; /* No hay huecos en el ROB */
9     }
10
11     RB[b].excepcion = I_EXC;
12     RB[b].prediccion = I_PRED;
13     RB[b].pred_data = I_PRED_DATA;
14
15     /** Lanza la instruccion */
16
17     switch (I_OP) {
18         case OP_FP_LD:
19             /** Busca un hueco en el buffer */
20             for (s = INICIO_BUFFER_CARGA; s <= FIN_BUFFER_CARGA; s++)
21                 if (!LB[s].ocupado) break;
22
23             if (s > FIN_BUFFER_CARGA) return;
24             /* No hay sitio en la estacion de reserva */
25
26             /** Reserva el buffer */
27             LB[s].ocupado = SI;
28             LB[s].OP = I_OP;
29             LB[s].rob = b;
30
31             /** Operando 1 (en Rint) */
32             if (Rint[I_S1].rob == MARCANULA) {
33                 LB[s].V1 = Rint[I_S1].valor;
34                 LB[s].Q1 = MARCANULA;
35             } else if (RB[Rint[I_S1].rob].completado) {
36                 LB[s].V1 = RB[Rint[I_S1].rob].valor;
37                 LB[s].Q1 = MARCANULA;
38             } else {
39                 LB[s].Q1 = Rint[I_S1].rob;
40             } /* endif */
41
42             /** Operando 2 */
43             LB[s].Q2 = MARCANULA;
44
45             /** Desplazamiento */
46             LB[s].desplazamiento = I_INM;
47
48             /** Reserva la entrada del ROB */
49             RB[b].ocupado = SI;
50             RB[b].OP = I_OP;
51             RB[b].dest = I_D;
```

```

52         RB[b].completado = NO;
53
54         /* ** Reserva del registro destino */
55         Rfp[I_D].rob = b; // OP_FP_LD
56
57         break;
58
59     case OP_FP_S_D:
60         /* ** Busca un hueco en el bufffer */
61
62         /* INSERTAR CODIGO */
63         for (s = INICIO_BUFFER_ALMACEN; s <= FIN_BUFFER_ALMACEN; s++)
64             if (!SB[s].ocupado) break;
65
66         if (s > FIN_BUFFER_ALMACEN) return;
67         /* No hay sitio en la estacion de reserva */
68
69         /* ** Reserva el buffer */
70
71         /* INSERTAR CODIGO */
72         SB[s].ocupado = SI;
73         SB[s].OP = I_OP;
74
75         /* ** Operando 1 (en Rint) */
76
77         /* INSERTAR CODIGO */
78         if (Rint[I_S1].rob == MARCA_NULA) {
79             SB[s].V1 = Rint[I_S1].valor;
80             SB[s].Q1 = MARCA_NULA;
81         } else if (RB[Rint[I_S1].rob].completado) {
82             SB[s].V1 = RB[Rint[I_S1].rob].valor;
83             SB[s].Q1 = MARCA_NULA;
84         } else {
85             SB[s].Q1 = Rint[I_S1].rob;
86         } /* endif */
87
88         /* ** Operando 2 (en Rfp) */
89
90         /* INSERTAR CODIGO */
91         if (Rfp[I_S2].rob == MARCA_NULA) {
92             SB[s].V2 = Rfp[I_S2].valor;
93             SB[s].Q2 = MARCA_NULA;
94         } else if (RB[Rfp[I_S2].rob].completado) {
95             SB[s].V2 = RB[Rfp[I_S2].rob].valor;
96             SB[s].Q2 = MARCA_NULA;
97         } else {
98             SB[s].Q2 = Rfp[I_S2].rob;
99         } /* endif */
100
101         /* ** Desplazamiento */
102
103         /* INSERTAR CODIGO */
104         SB[s].desplazamiento = I_INM;
105
106         /* ** Reserva la entrada del ROB */
107

```

```

108      /* INSERTAR CODIGO */
109      RB[b].ocupado = SI;
110      RB[b].OP = I_OP;
111      RB[b].dest = s; /* TE */
112
113      break;
114      case OP_FP_ADD_D:
115      case OP_FP_SUB_D:
116          /* ** Busca un hueco en la estacion de reserva */
117
118          /* INSERTAR CODIGO */
119          for (s = INICIO_RS_SUMREST; s <= FIN_RS_SUMREST; s++)
120              if (!RS[s].ocupado) break;
121
122          if (s > FIN_RS_SUMREST) return;
123          /* No hay sitio en la estacion de reserva */
124          /* ** Reserva la estacion de reserva */
125
126          /* INSERTAR CODIGO */
127          RS[s].ocupado = SI;
128          RS[s].OP = I_OP;
129          RS[s].rob = b;
130
131          /* ** Operando 1 (en Rfp) */
132
133          /* INSERTAR CODIGO */
134          if (Rfp[I_S1].rob == MARCA_NULA) {
135              RS[s].V1 = Rfp[I_S1].valor;
136              RS[s].Q1 = MARCA_NULA;
137          } else if (RB[Rfp[I_S1].rob].completado) {
138              RS[s].V1 = RB[Rfp[I_S1].rob].valor;
139              RS[s].Q1 = MARCA_NULA;
140          } else {
141              RS[s].Q1 = Rfp[I_S1].rob;
142          } /* endif */
143
144          /* ** Operando 2 (en Rfp) */
145
146          /* INSERTAR CODIGO */
147          if (Rfp[I_S2].rob == MARCA_NULA) {
148              RS[s].V2 = Rfp[I_S2].valor;
149              RS[s].Q2 = MARCA_NULA;
150          } else if (RB[Rfp[I_S2].rob].completado) {
151              RS[s].V2 = RB[Rfp[I_S2].rob].valor;
152              RS[s].Q2 = MARCA_NULA;
153          } else {
154              RS[s].Q2 = Rfp[I_S2].rob;
155          } /* endif */
156
157          /* ** Reserva la entrada del ROB */
158
159          /* INSERTAR CODIGO */
160          RB[b].ocupado = SI;
161          RB[b].OP = I_OP;
162          RB[b].dest = I_D;
163          RB[b].completado = NO;

```

```

164
165      /** Reserva del registro destino */
166
167      /* INSERTAR CODIGO */
168      Rfp[I_D].rob = b;
169
170      break;
171      case OP_FP_MUL_D:
172      case OP_FP_DIV_D:
173      /** Busca un hueco en la estacion de reserva */
174
175      /* INSERTAR CODIGO */
176      for (s = INICIO_RS_MULTDIV; s <= FIN_RS_MULTDIV; s++)
177          if (!RS[s].ocupado) break;
178
179      if (s > FIN_RS_MULTDIV) return;
180      /* No hay sitio en la estacion de reserva */
181
182      /** Reserva el operador virtual */
183
184      /* INSERTAR CODIGO */
185      RS[s].ocupado = SI;
186      RS[s].OP = I_OP;
187
188
189      /** Operando 1 */
190
191      /* INSERTAR CODIGO */
192      if (Rfp[I_S1].rob == MARCA_NULA) {
193          RS[s].V1 = Rfp[I_S1].valor;
194          RS[s].Q1 = MARCA_NULA;
195      } else if (RB[Rfp[I_S1].rob].completado) {
196          RS[s].V1 = RB[Rfp[I_S1].rob].valor;
197          RS[s].Q1 = MARCA_NULA;
198      } else {
199          RS[s].Q1 = Rfp[I_S1].rob;
200      } /* endif */
201
202      /** Operando 2 */
203
204      /* INSERTAR CODIGO */
205      if (Rfp[I_S2].rob == MARCA_NULA) {
206          RS[s].V2 = Rfp[I_S2].valor;
207          RS[s].Q2 = MARCA_NULA;
208      } else if (RB[Rfp[I_S2].rob].completado) {
209          RS[s].V2 = RB[Rfp[I_S2].rob].valor;
210          RS[s].Q2 = MARCA_NULA;
211      } else {
212          RS[s].Q2 = Rfp[I_S2].rob;
213      } /* endif */
214
215      /** Reserva la entrada del ROB */
216
217      /* INSERTAR CODIGO */
218      RB[b].ocupado = SI;
219      RB[b].OP = I_OP;

```

```

220         RB[b].dest = I.D;
221         RB[b].completado = NO;
222
223         /** Reserva del registro destino */
224
225         /* INSERTAR CODIGO */
226         Rfp[I.D].rob = b;
227
228         break;
229         ...
230     } /* endswitch */
231
232     /** La instruccion se ha lanzado correctamente */
233
234     Control_1.Parar = NO;
235     RB_fin = (RB_fin + 1) % TAM.REORDER;
236     RB_long++;
237
238     return;
239
240 } /* end fase_ISS */

```

```

1 void fase_WB_alum() {
2     ...
3     /** Busca RS con resultados disponibles */
4
5     orden = MAX_ORDEN;
6     s = 0;
7
8     for (i = 0; i < TAM.ESTACIONES; i++) {
9         if (RS[i].ocupado && RS[i].estado == FINALIZADA && RS[i].orden < orden) {
10             s = i;
11             orden = RS[i].orden;
12         } /* endif */
13     } /* endif */
14
15     if (orden >= MAX_ORDEN) return; /* No hay RS con resultados */
16
17     /** Volcado de resultados */
18
19     /* INSERTAR CODIGO */
20     BUS.valor = RS[s].resultado;
21     BUS.codigo = RS[s].rob;
22     BUS.condicion = RS[s].condicion;
23     BUS.control = RS[s].control;
24
25     /** Libera la RS */
26
27     /* INSERTAR CODIGO */
28     RS[s].ocupado = NO;
29
30     /** Lectura de resultados */
31
32     // Escritura en el ROB
33
34     /* INSERTAR CODIGO */

```

```
35 RB[BUS.codigo].valor = BUS.valor;
36 RB[BUS.codigo].completado = SI;
37 RB[BUS.codigo].condicion = BUS.condicion;
38 RB[BUS.codigo].control = BUS.control;
```

```
39
40 /* Estaciones de reserva */
```

```
41
42 for (s = INICIO_RS.ENTEROS;
43      s <= FIN_RS.ENTEROS; s++) {
```

```
44
45 /* INSERTAR CODIGO */
```

```
46     if (RS[s].Q1 == BUS.codigo) {
47         RS[s].V1 = BUS.valor;
48         RS[s].Q1 = MARCA_NULA;
49     } /* endif */
```

```
50
51     if (RS[s].Q2 == BUS.codigo) {
52         RS[s].V2 = BUS.valor;
53         RS[s].Q2 = MARCA_NULA;
54     } /* endif */
```

```
55
56 } /* endfor */
```

```
57
58 for (s = INICIO_RS.SUMREST;
59      s <= FIN_RS.SUMREST; s++) {
```

```
60
61 /* INSERTAR CODIGO */
```

```
62     if (RS[s].Q1 == BUS.codigo) {
63         RS[s].V1 = BUS.valor;
64         RS[s].Q1 = MARCA_NULA;
65     } /* endif */
```

```
66     if (RS[s].Q2 == BUS.codigo) {
67         RS[s].V2 = BUS.valor;
68         RS[s].Q2 = MARCA_NULA;
69     } /* endif */
```

```
70 } /* endfor */
```

```
71
72 for (s = INICIO_RS.MULTDIV;
73      s <= FIN_RS.MULTDIV; s++) {
```

```
74
75 /* INSERTAR CODIGO */
```

```
76     if (RS[s].Q1 == BUS.codigo) {
77         RS[s].V1 = BUS.valor;
78         RS[s].Q1 = MARCA_NULA;
79     } /* endif */
```

```
80     if (RS[s].Q2 == BUS.codigo) {
81         RS[s].V2 = BUS.valor;
82         RS[s].Q2 = MARCA_NULA;
83     } /* endif */
```

```
84 } /* endfor */
```

```
85
86 for (s = INICIO_BUFFER.CARGA;
87      s <= FIN_BUFFER.CARGA; s++) {
```

```
88
89 /* INSERTAR CODIGO */
```

```
90     if (LB[s].Q1 == BUS.codigo) {
```


Estado al final del ciclo

	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	Dir	+0	+1	+2	+3
rob					# 3	# 4																											a				
valor	10.50	2.00	20.00																														8196			10.50	
	R0 (zero)	R1 (at)	R2 (v0)	R3 (v1)	R4 (a0)	R5 (a1)	R6 (a2)	R7 (a3)	R8 (t0)	R9 (t1)	R10 (t2)	R11 (t3)	R12 (t4)	R13 (t5)	R14 (t6)	R15 (t7)	R16 (s0)	R17 (s1)	R18 (s2)	R19 (s3)	R20 (s4)	R21 (s5)	R22 (s6)	R23 (s7)	R24 (t8)	R25 (t9)	R26 (k1)	R27 (k2)	R28 (gp)	R29 (sp)	R30 (fp)	R31 (ra)	b				
rob																																	c				
valor																													8192	65536			8212			20.00	

Cod Ocup.	Instrucción	Completo.	Destino	Valor	Pred.	Exc.	PC	Cod Ocup.	Op	Q1	V1	Q2	V2	rob	Resultado	Estado
0	NO l.d f0,%disp[a](r28)	Sí	F0	10.50			inicio	e1	NO							
1	NO l.d f1,%disp[b](r28)	Sí	F1	2.00			4100	e2	NO							
2	NO l.d f2,%disp[c](r28)	Sí	F2	20.00			4104	e3	NO							
3	Sí add.d f4,f0,f1	Sí	F4	12.50			4108	e4	NO							
4	Sí mul.d f5,f2,f4	NO	F5				4112	e5	NO							
5	Sí s.d f4,%disp[s1](r28)	NO	s1				4116									
6	Sí s.d f5,%disp[s2](r28)	NO	s2				4120									
7	Sí trap 0	Sí				32	final	a1	NO	+	10.50		2.00	# 3		
8	NO							a2	NO							
9	NO							a3	NO							
10	NO							m1	SI	*	20.00		12.50	# 4		
11	NO							m2	NO							
12	NO															
13	NO															
14	NO															
15	NO															
16	NO															
17	NO															
18	NO															
19	NO															

Cod Ocup.	Op	Q1	V1	Desp(etiq)	Direc.	rob	Resultado	Estado	Cod Ocup.	Op	Q1	V1	Desp(etiq)	Direc.	rob	Q2	V2	Confirm.	Estado
l1	NO	l.d	8192	0 (%disp[a])		# 0			s1	SI	s.d	8192	24 (%disp[s1])		s1	# 5	# 3 ???	NO	
l2	NO	l.d	8192	8 (%disp[b])		# 1			s2	SI	s.d	8192	32 (%disp[s2])		s2	# 6	# 4	NO	
l3	NO	l.d	8192	16 (%disp[c])		# 2			s3	NO									

b) (2,5 puntos) Indicar en qué consiste este segundo error y qué líneas en el código habría que corregir o añadir para solucionarlo.