

```

1  Entrance:      j Start
2                  nop
3  EXCINTHandler: mfc0 $k0, $l2          # $k0 <- CP0.$cause
4                  andi $k1, $k0, 0xc    # $k1 = EXcCode (cause[3:2])
5                  addi $s1, $zero, 0x4   # 0x0100, syscall
6                  addi $s2, $zero, 0x8   # 0x1000, UnInstr
7                  addi $s3, $zero, 0xC   # 0x1100, OV
8                  beq $k1, $s1, Handle_SYSCALL
9                  beq $k1, $s2, Handle_UnInstr
10                 beq $k1, $s3, Handle_OV
11  Handle_INT:    sll $v1, $v1, 0x1
12                 ori $v1, $v1, 0x1      # 循环右移, 每次在最低位补 1
13                 addi $fp, $fp, 0x4
14                 andi $fp, $fp, 0x003F  # 更新 $fp, 因为预置数字和预置图像都是 16
15                 addi $v0, $v0, 0x1     # 个数据一组, 所以用 6 位 mask (4 + 2, 地址最低两位恒为 2'b00)
16                 bne $v0, $at, Disp     # increase $v0 for SYSCALL
17                 step is useless in this program, since $v0 [0~32])
18                 addi $v0, $zero, 0x5
19  Disp:          addi $s1, $zero, 0x8    # 5'b01000, SW[4:3]=2'b01 && SW[0]=1
20                 addi $s2, $zero, 0x10  # 5'b10000, SW[4:3]=2'b10 && SW[0]=1
21                 addi $s3, $zero, 0x18  # 5'b11000, SW[4:3]=2'b11 && SW[0]=0
22                 lw $s5, 0x0($a2)
23                 andi $s5, $s5, 0x18    # 0x18 = 5'b11000, mask to get SW[4:3]
24                 beq $s5, $zero, SW_00  # SW[4:3]=2'b00 (&& SW[0]=0), dot/line of
25                 SSeg7 shift in loop.
26                 beq $s5, $s1, SW_01    # SW[4:3]=2'b01 (&& SW[0]=0), 0x00000000 ->
27                 0x11111111 -> ... -> 0xFFFFFFFF
28                 beq $s5, $s2, SW_10    # SW[4:3]=2'b10 (&& SW[0]=0), show cycle
29                 accumulation of $v0
30                 beq $s5, $s3, SW_11    # SW[4:3]=2'b11 (&& SW[0]=0), show pictures
31  SW_00:         bne $v1, $at, L3       # if ($v1 = 0xFFFFFFFF)
32                 sll $v1, $v1, 0x1     # $v1 <<= 0x1 // $v1 = 0xFFFFFFFFE
33  L3:            sw $v1, 0x0($a1)       # else
34                 j Disp_done           # // show $t0 on SSeg7
35  SW_01:         lw $k0, 0x20($fp)      # 显示预置数字
36                 sw $k0, 0x0($a1)
37                 j Disp_done
38  SW_10:         sw $v0, 0x0($a1)       # 显示 $v0 (累加)
39                 j Disp_done
40  SW_11:         lw $k0, 0x60($fp)      # show PictureSet1
41                 sw $k0, 0x0($a1)
42  Disp_done:     lw $s1, 0x0($a2)       # $s1 = {counte$0_out, counte$1_out,
43                 counte$2_out, led_out[0x12:0x0], SW}
44                 sll $s1, $s1, 0x2
45                 sw $s1, 0x0($a2)       # Align SW[0x15:0x0] with LED && choose
46                 counter0
47                 addi $s2, $zero, 0x7fff # reset counter0 init_value
48                 sw $s2, 0x0($a3)
49                 nop                    # 128 nop, to ensure that counter0 has reset.
50                 nop
51                 nop
52                 nop
53                 nop
54                 nop
55                 nop
56                 nop
57                 nop
58                 nop
59                 nop
60                 nop
61                 nop
62                 nop
63                 nop
64                 nop
65                 nop
66                 nop

```

67	nop
68	nop
69	nop
70	nop
71	nop
72	nop
73	nop
74	nop
75	nop
76	nop
77	nop
78	nop
79	nop
80	nop
81	nop
82	nop
83	nop
84	nop
85	nop
86	nop
87	nop
88	nop
89	nop
90	nop
91	nop
92	nop
93	nop
94	nop
95	nop
96	nop
97	nop
98	nop
99	nop
100	nop
101	nop
102	nop
103	nop
104	nop
105	nop
106	nop
107	nop
108	nop
109	nop
110	nop
111	nop
112	nop
113	nop
114	nop
115	nop
116	nop
117	nop
118	nop
119	nop
120	nop
121	nop
122	nop
123	nop
124	nop
125	nop
126	nop
127	nop
128	nop
129	nop
130	nop
131	nop
132	nop
133	nop
134	nop
135	nop
136	nop
137	nop
138	nop
139	nop

```

140      nop
141      nop
142      nop
143      nop
144      nop
145      nop
146      nop
147      nop
148      nop
149      nop
150      nop
151      nop
152      nop
153      nop
154      nop
155      nop
156      nop
157      nop
158      nop
159      nop
160      nop
161      nop
162      nop
163      nop
164      nop
165      nop
166      nop
167      nop
168      nop
169      nop
170      nop
171      eret
172  Handle_SYSCALL: addi $s7, $zero, 0x20
173                  sll $s7, $s7, 0x2
174                  add $v0, $zero, $zero      # set $v0 to 0, it will be reused in
SYSCALL loop
175                  add $k1, $zero, $zero      # use k1 as a tmp_cnt
176                  lui $s6, 0x10              # use $s6 as the tmp_cnt's threshold
177  Show_PicSet2:   addi $k1, $k1, 0x1
178                  bne $k1, $s6, Show_PicSet2 #
用$k1进行计数, 直到0x0010_0000时, 才可以改变内存
179                  add $k1, $zero, $zero      # reset $k1 = 0, 计数用完, 重新赋值为0
180                  lw $k0, 0xA0($v0)          # PicSet2 baseAddr 0xA0 ($v0 ==
0, 使用的是RAM的地址)
181                  sw $k0, 0x0($a1)           # a1 ==
0xE000_0000, 将0x0000_00A0(对应要除以4, 也就是coe文件中的0x0000_0028)
处的值(0xFFFFFFFF7)放到Seg7里面
182                  addi $v0, $v0, 0x4
183                  bne $v0, $s7, Show_PicSet2 # $s7 = 0x80, 用其进行计数, 总共0x80 /
4要循环32次
184                  add $v0, $zero, $zero      # reset $v0 to 0
185                  add $s7, $zero, $zero
186  SYSCALL_done:  j Handle_EPCp4
187  Handle_UnInstr: j Handle_EPCp4             # 对于出现异常的指令一律不执行, 跳过之
188  Handle_OV:     nop
189  Handle_EPCp4:  mfc0 $26, $14
190                  addi $26, $26, 0x4          #
返回EPC+4处, 说明OV产生时, 存入EPC的值必须是本条指令的PC地址, 而不是PC
Plus4, 而我在实现的时候是用ID_EX_REG的PCPlus4, 要减8
191                  mtc0 $26, $14
192                  eret
193                  nop
194                  nop
195  Start:         add $a0, $zero, $zero      # $a0 0x0000_0000 RAM
196                  lui $a1, 0xE000          # $a1 0xE000_0000 SSeg7
197                  lui $a2, 0xF000          # $a2 0xF000_0000 Switch/LED (SPIO)
198                  ori $a3, $a2, 0x4        # $a3 0xF000_0004 CounterX
都是在准备地址
199                  lui $at, 0xFFFF
200                  ori $at, $at, 0xFFFF      # $at = 0xFFFFFFFF
201                  addi $t9, $zero, 0x20     # 32 (DEM) $t9 = 0x20
202                  add $v1, $at, $zero
203                  sll $v1, $v1, 0x1         # $v1 = 0xFFFFFFFF

```

```

204      addi $t0, $zero, 0xE      # 关中断 设置Status位为0b1110,
      被设置为0的位是被屏蔽的Excp
205      mtc0 $t0, $13
206      lui  $t0, 0x7FFF
207      ori  $t0, $t0, 0xFFFF    # $t0 = 0x7FFFFFFF
208      addi $t1, $zero, 0x2
209      add  $t0, $t0, $t1        # overflow here
210      break
211      addi $t0, $zero, 0x2AB    # ...10101010_11, {GPIOf0[13:0], LED,
      counter_set}
212      addi $t1, $zero, 0x7fff  # counter0 init val 0x00080000
213      sw $t0, 0x0($a2)          # choose Ctrl_Reg, also set init_val of LED
214      sw $zero, 0x0($a3)        # write Ctrl_Reg, counter0 WorkMode = 2'b00
215      lw $t3, 0x0($a2)          # $t3 = {counter0_out, counter1_out,
      counter2_out, led_out[12:0], SW}
216      sll $t3, $t3, 0x2         # Align SW[15:0] with LED && choose counter0
      (srl makes $t3[1:0] = 2'b00)
217      sw $t3, 0x0($a2)
218      sw $t1, 0x0($a3)          # write counter0 init value (== 0x00080000)
219      addi $t0, $zero, 0xF      # 开中断
220      mtc0 $t0, $13
221      Loop: lw $t0, 0x0($a2)      # $t0 = {counter0_out, counter1_out,
      counter2_out, led_out[12:0], SW}
222      sll $t0, $t0, 0x2         # Align SW[15:0] with LED
223      sw $t0, 0x0($a2)
224      bne $v0, $t9, Loop
225      SYSCALL
226      add $v0, $zero, $zero     # reset cnt $v0
227      j Loop
228      nop
229      nop

```