```
1           ;代码清单9-1
2           ;文件名：c09_1.asm
3           ;文件说明：用户程序
4           ;创建日期：2011-4-16 22:03
5
6  ;===============================================================================
7  SECTION header vstart=0                     ;定义用户程序头部段
8      program_length  dd program_end          ;程序总长度[0x00]
9
10     ;用户程序入口点
11     code_entry      dw start                ;偏移地址[0x04]
12                     dd section.code.start   ;段地址[0x06]
13
14     realloc_tbl_len dw (header_end-realloc_begin)/4
15                                             ;段重定位表项个数[0x0a]
16
17     realloc_begin:
18     ;段重定位表
19     code_segment    dd section.code.start   ;[0x0c]
20     data_segment    dd section.data.start   ;[0x14]
21     stack_segment   dd section.stack.start  ;[0x1c]
22
23 header_end:
24
25 ;===============================================================================
26 SECTION code align=16 vstart=0           ;定义代码段（16字节对齐）
27 new_int_0x70:
28       push ax
29       push bx
30       push cx
31       push dx
32       push es
33
34   .w0:
35       mov al,0x0a                        ;阻断NMI。当然，通常是不必要的
36       or al,0x80
37       out 0x70,al
38       in al,0x71                         ;读寄存器A
39       test al,0x80                       ;测试第7位UIP
40       jnz .w0                            ;以上代码对于更新周期结束中断来说
41                                          ;是不必要的
42       xor al,al
43       or al,0x80
44       out 0x70,al
45       in al,0x71                         ;读RTC当前时间(秒)
46       push ax
47
48       mov al,2
49       or al,0x80
50       out 0x70,al
51       in al,0x71                         ;读RTC当前时间(分)
52       push ax
53
```

```
54        mov al,4
55        or al,0x80
56        out 0x70,al
57        in al,0x71                        ;读RTC当前时间(时)
58        push ax
59
60        mov al,0x0c                       ;寄存器C的索引。且开放NMI
61        out 0x70,al
62        in al,0x71                        ;读一下RTC的寄存器C，否则只发生一次中断
63                                          ;此处不考虑闹钟和周期性中断的情况
64        mov ax,0xb800
65        mov es,ax
66
67        pop ax
68        call bcd_to_ascii
69        mov bx,12*160 + 36*2              ;从屏幕上的12行36列开始显示
70
71        mov [es:bx],ah
72        mov [es:bx+2],al                  ;显示两位小时数字
73
74        mov al,':'
75        mov [es:bx+4],al                  ;显示分隔符':'
76        not byte [es:bx+5]                ;反转显示属性
77
78        pop ax
79        call bcd_to_ascii
80        mov [es:bx+6],ah
81        mov [es:bx+8],al                  ;显示两位分钟数字
82
83        mov al,':'
84        mov [es:bx+10],al                 ;显示分隔符':'
85        not byte [es:bx+11]               ;反转显示属性
86
87        pop ax
88        call bcd_to_ascii
89        mov [es:bx+12],ah
90        mov [es:bx+14],al                 ;显示两位小时数字
91
92        mov al,0x20                       ;中断结束命令EOI
93        out 0xa0,al                       ;向从片发送
94        out 0x20,al                       ;向主片发送
95
96        pop es
97        pop dx
98        pop cx
99        pop bx
100       pop ax
101
102       iret
103
104 ;-----------------------------------------------------------------------------
105 bcd_to_ascii:                           ;BCD码转ASCII
106                                         ;输入：AL=bcd码
```

```
107                                     ;输出：AX=ascii
108         mov ah,al                   ;分拆成两个数字
109         and al,0x0f                 ;仅保留低4位
110         add al,0x30                 ;转换成ASCII
111
112         shr ah,4                    ;逻辑右移4位
113         and ah,0x0f
114         add ah,0x30
115
116         ret
117
118 ;------------------------------------------------------------------------------
119 start:
120         mov ax,[stack_segment]
121         mov ss,ax
122         mov sp,ss_pointer
123         mov ax,[data_segment]
124         mov ds,ax
125
126         mov bx,init_msg             ;显示初始信息
127         call put_string
128
129         mov bx,inst_msg             ;显示安装信息
130         call put_string
131
132         mov al,0x70
133         mov bl,4
134         mul bl                      ;计算0x70号中断在IVT中的偏移
135         mov bx,ax
136
137         cli                         ;防止改动期间发生新的0x70号中断
138
139         push es
140         mov ax,0x0000
141         mov es,ax
142         mov word [es:bx],new_int_0x70   ;偏移地址。
143
144         mov word [es:bx+2],cs       ;段地址
145         pop es
146
147         mov al,0x0b                 ;RTC寄存器B
148         or al,0x80                  ;阻断NMI
149         out 0x70,al
150         mov al,0x12                 ;设置寄存器B，禁止周期性中断，开放更
151         out 0x71,al                 ;新结束后中断，BCD码，24小时制
152
153         mov al,0x0c
154         out 0x70,al
155         in al,0x71                  ;读RTC寄存器C，复位未决的中断状态
156
157         in al,0xa1                  ;读8259从片的IMR寄存器
158         and al,0xfe                 ;清除bit 0(此位连接RTC)
159         out 0xa1,al                 ;写回此寄存器
```

```
160
161       sti                                   ;重新开放中断
162
163       mov bx,done_msg                       ;显示安装完成信息
164       call put_string
165
166       mov bx,tips_msg                       ;显示提示信息
167       call put_string
168
169       mov cx,0xb800
170       mov ds,cx
171       mov byte [12*160 + 33*2],'@'          ;屏幕第12行，35列
172
173  .idle:
174       hlt                                   ;使CPU进入低功耗状态，直到用中断唤醒
175       not byte [12*160 + 33*2+1]            ;反转显示属性
176       jmp .idle
177
178 ;-------------------------------------------------------------------------------
179 put_string:                                 ;显示串(0结尾)。
180                                             ;输入：DS:BX=串地址
181          mov cl,[bx]
182          or cl,cl                           ;cl=0 ？
183          jz .exit                           ;是的，返回主程序
184          call put_char
185          inc bx                             ;下一个字符
186          jmp put_string
187
188    .exit:
189          ret
190
191 ;-------------------------------------------------------------------------------
192 put_char:                                   ;显示一个字符
193                                             ;输入：cl=字符ascii
194          push ax
195          push bx
196          push cx
197          push dx
198          push ds
199          push es
200
201          ;以下取当前光标位置
202          mov dx,0x3d4
203          mov al,0x0e
204          out dx,al
205          mov dx,0x3d5
206          in al,dx                           ;高8位
207          mov ah,al
208
209          mov dx,0x3d4
210          mov al,0x0f
211          out dx,al
212          mov dx,0x3d5
```

```
213         in al,dx                        ;低8位
214         mov bx,ax                       ;BX=代表光标位置的16位数
215
216         cmp cl,0x0d                     ;回车符?
217         jnz .put_0a                     ;不是。看看是不是换行等字符
218         mov ax,bx                       ;
219         mov bl,80
220         div bl
221         mul bl
222         mov bx,ax
223         jmp .set_cursor
224
225 .put_0a:
226         cmp cl,0x0a                     ;换行符?
227         jnz .put_other                  ;不是，那就正常显示字符
228         add bx,80
229         jmp .roll_screen
230
231 .put_other:                             ;正常显示字符
232         mov ax,0xb800
233         mov es,ax
234         shl bx,1
235         mov [es:bx],cl
236
237         ;以下将光标位置推进一个字符
238         shr bx,1
239         add bx,1
240
241 .roll_screen:
242         cmp bx,2000                     ;光标超出屏幕？滚屏
243         jl .set_cursor
244
245         mov ax,0xb800
246         mov ds,ax
247         mov es,ax
248         cld
249         mov si,0xa0
250         mov di,0x00
251         mov cx,1920
252         rep movsw
253         mov bx,3840                     ;清除屏幕最底一行
254         mov cx,80
255 .cls:
256         mov word[es:bx],0x0720
257         add bx,2
258         loop .cls
259
260         mov bx,1920
261
262 .set_cursor:
263         mov dx,0x3d4
264         mov al,0x0e
265         out dx,al
```

```
266         mov dx,0x3d5
267         mov al,bh
268         out dx,al
269         mov dx,0x3d4
270         mov al,0x0f
271         out dx,al
272         mov dx,0x3d5
273         mov al,bl
274         out dx,al
275
276         pop es
277         pop ds
278         pop dx
279         pop cx
280         pop bx
281         pop ax
282
283         ret
284
285 ;===================================================================================
286 SECTION data align=16 vstart=0
287
288     init_msg       db 'Starting...',0x0d,0x0a,0
289
290     inst_msg       db 'Installing a new interrupt 70H...',0
291
292     done_msg       db 'Done.',0x0d,0x0a,0
293
294     tips_msg       db 'Clock is now working.',0
295
296 ;===================================================================================
297 SECTION stack align=16 vstart=0
298
299                  resb 256
300 ss_pointer:
301
302 ;===================================================================================
303 SECTION program_trail
304 program_end:
```