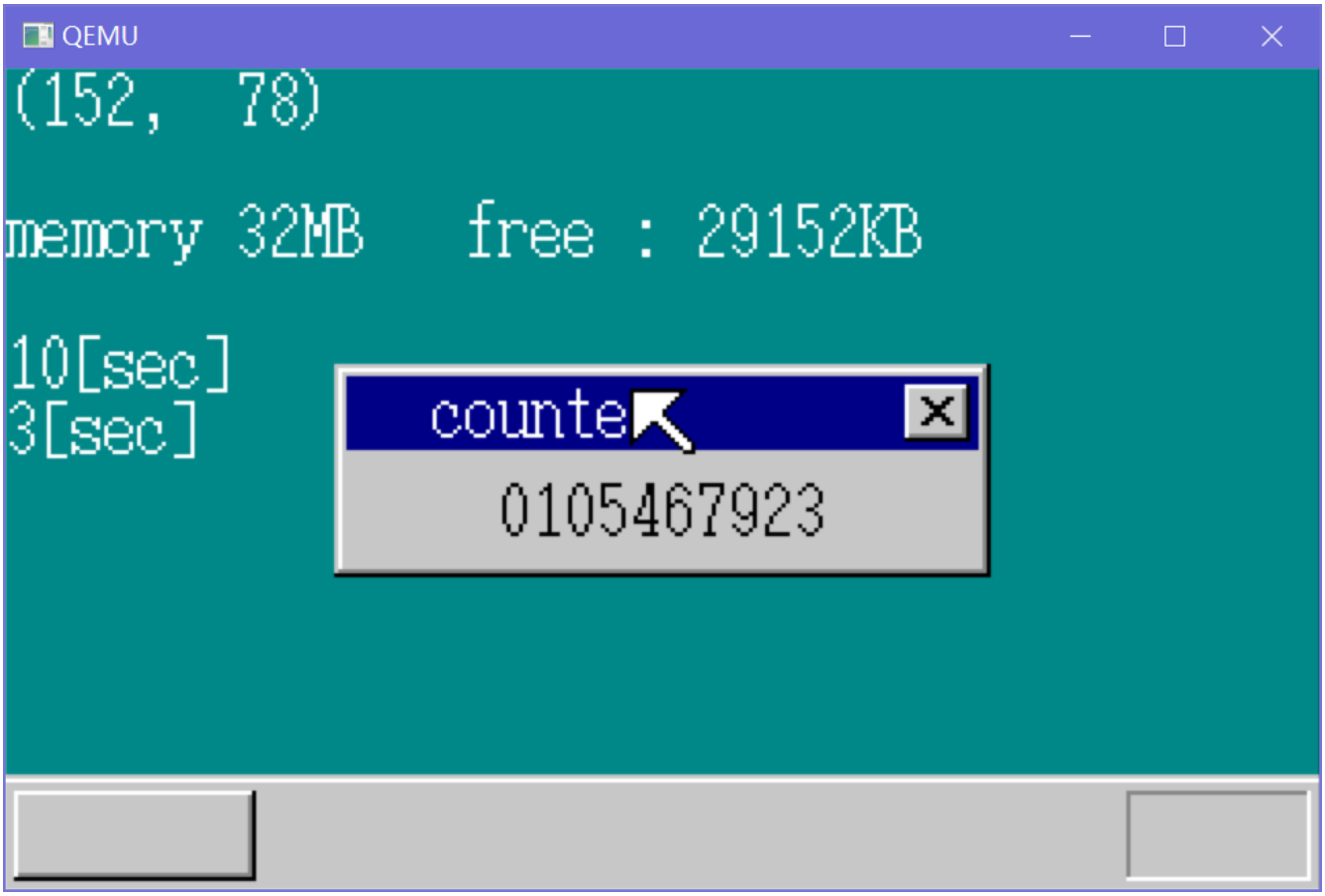
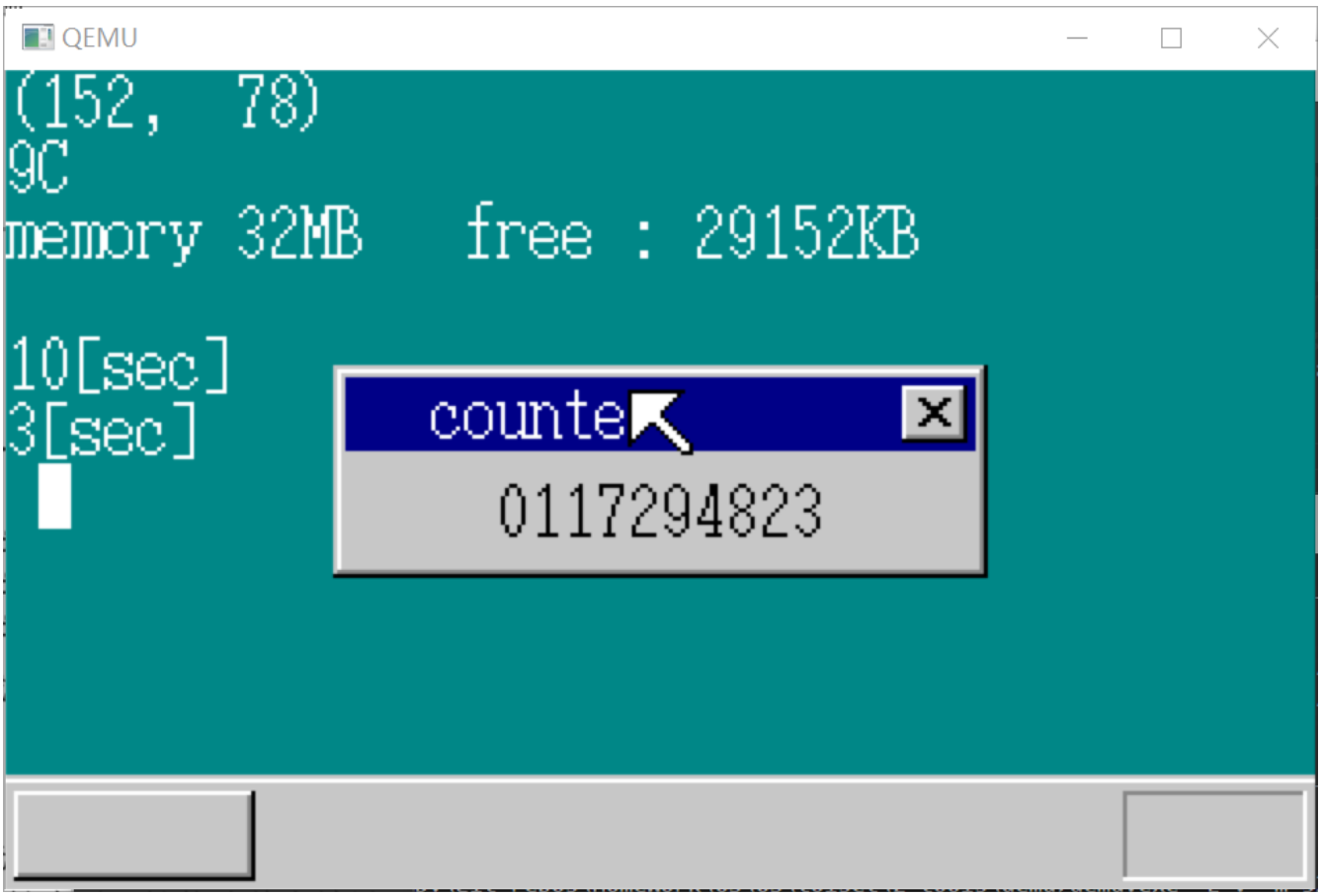
Day 14

今天我们做的第一件事情是测试性能

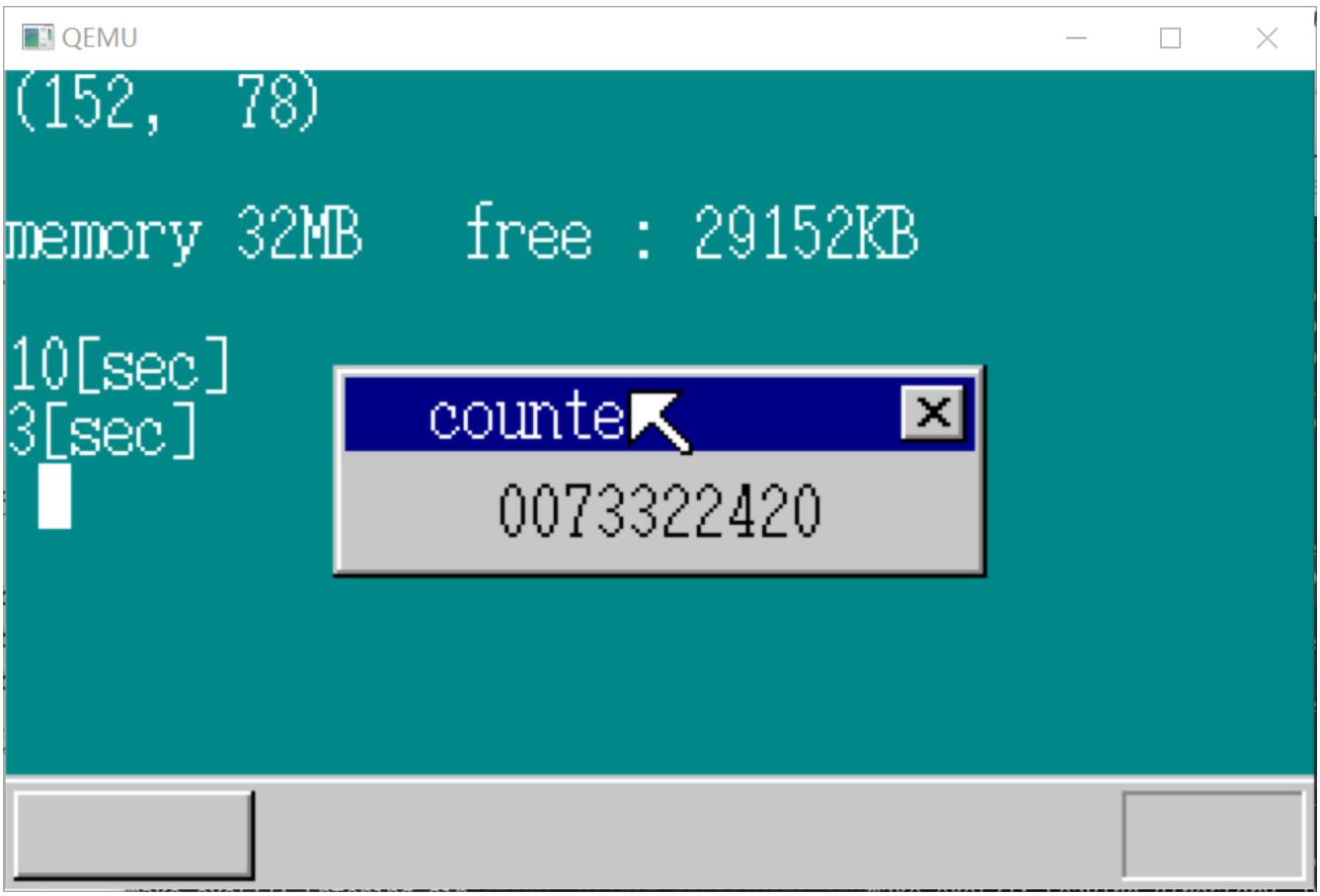
A 追加490个定时器，有移位



B 追加490个定时器，链表，没有哨兵



C 追加490个定时器，链表，没有哨兵



emmm，这跟作者的实机测试结果来比，哪怕三个数据的大小关系都不一样，看来系统上正在运行的其他应用对运行结果的影响蛮大的。

没法跑真机，我们还是暂时忘了他吧。

然后还有一个玄学jmp问题，前面加set490以后各个指令的地址也都会相应地错开几个字节，结果造成JMP指令的地址也略有变化，因此执行时间也稍稍延迟。

然后我们来提高我们操作系统的分辨率吧，在2k屏、3k屏上看320x200的画面真是有点别扭。不同的显卡，高分辨率利用方法也不太一样，我们先只考虑如何在QEMU中使用高分辨率

```
MOV BX,0x4101 ; VBE的640x480x8bit彩色
MOV AX,0x4f02
INT 0x10
MOV BYTE [VMODE],8 ; 记下画面模式（参考C语言）
MOV WORD [SCRNX],640
MOV WORD [SCRNY],480
MOV DWORD [VRAM],0xe000000
```

由于曾经的显卡种类众多，各家标准也不太相同，设定起来不一样，混乱而复杂，为了解决这样的问题，显卡公司们成立了VESA协会，制定了基础通用的设定方法。我们按照VESA指定的格式就可以开启高分辨率模式了。

但是有的公司并没有加入VESA，这种设定方法也不一定能够使用，我们最好还是先检查一下VBE是否存在以及他的版本。

```
; 确认VBE是否存在
MOV AX,0x9000
MOV ES,AX
MOV DI,0
MOV AX,0x4f00
INT 0x10
CMP AX,0x004f
JNE scrn320
```

```
; 检查VBE的版本
MOV AX,[ES:DI+4]
CMP AX,0x0200
JB scrn320 ; if (AX < 0x0200) goto scrn320
```

```
; 取得画面模式信息
MOV CX,VBEMODE
MOV AX,0x4f01
INT 0x10
CMP AX,0x004f
JNE scrn320
```

类型	位置	备注
WORD	[ES : DI+0x00]	模式属性.....bit7 不是 1 就不好办(能加上 0x4000)
WORD	[ES : DI+0x12]	X 的分辨率
WORD	[ES : DI+0x14]	Y 的分辨率
BYTE	[ES : DI+0x19]	颜色数.....必须为 8
BYTE	[ES : DI+0x1b]	颜色的指定方法.....必须为 4 (4 是调色板模式)
DWORD	[ES : DI+0x28]	VRAM 的地址

获取完画面模式信息之后，我们要检查一下他是否支持我们要选用的高分辨率的模式

```
; 画面模式信息的确认
CMP BYTE [ES:DI+0x19],8
JNE scrn320
CMP BYTE [ES:DI+0x1b],4
JNE scrn320
MOV AX,[ES:DI+0x00]
AND AX,0x0080
JZ scrn320 ; 模式属性的bit7是0，所以放
```

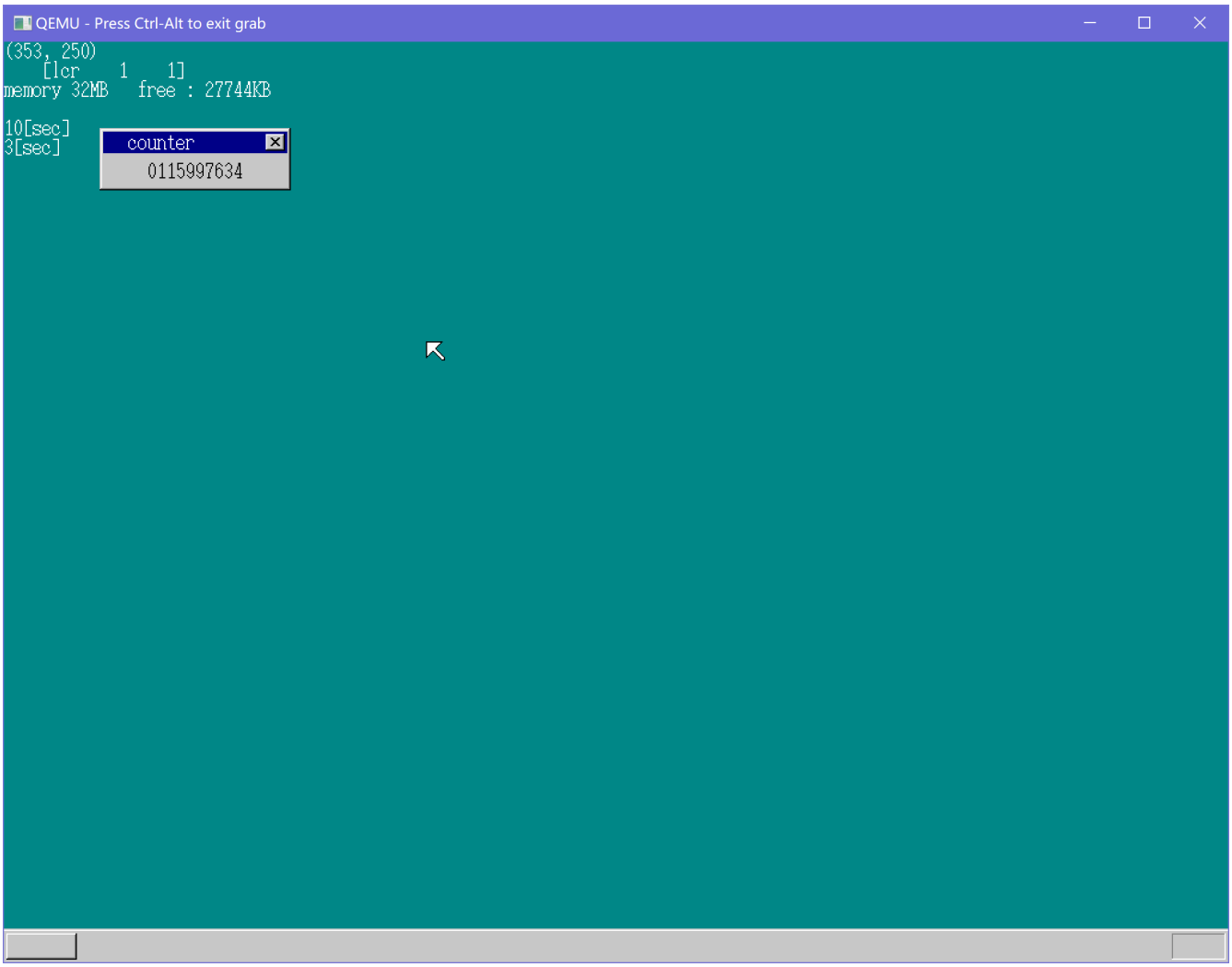
做完这些还不够，记得我们之前进入32位模式的之前我们做了什么嘛，我们把画面的相关信息存入了bootinfo中，我们这次也要这么做

```
; 画面模式的切换
MOV BX,VBEMODE+0x4000
MOV AX,0x4f02
INT 0x10
MOV BYTE [VMODE],8 ; 记下画面模式（参考C语言）
MOV AX,[ES:DI+0x12]
MOV [SCRNX],AX
MOV AX,[ES:DI+0x14]
MOV [SCRNY],AX
MOV EAX,[ES:DI+0x28]
MOV [VRAM],EAX
JMP keystatus
```

如果任意一个地方有问题，我们就要将就使用320x200画面了

```
scrn320:
MOV AL,0x13 ; VGA图、320x200x8bit彩色
MOV AH,0x00
INT 0x10
MOV BYTE [VMODE],8 ; 记下画面模式（参考C语言）
MOV WORD [SCRNX],320
MOV WORD [SCRNY],200
MOV DWORD [VRAM],0x000a0000
```

`make run`一下



键盘输入

有如下的表格

按下键时的数值表

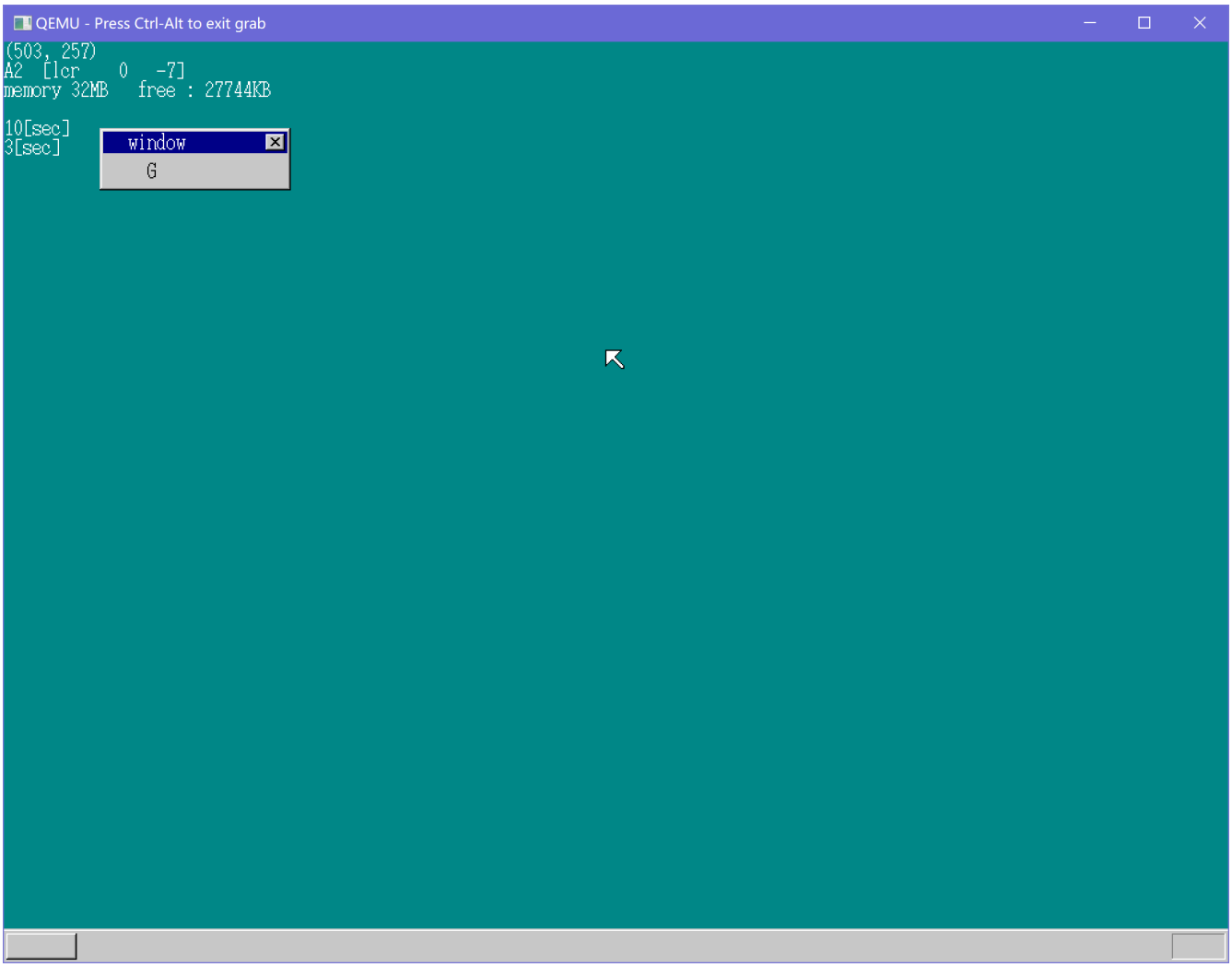
00：没有被分配使用	20：D	40：F6	60：保留
01：ESC	21：F	41：F7	61：保留？
02：主键盘的1	22：G	42：F8	62：保留？
03：主键盘的2	23：H	43：F9	63：保留？
04：主键盘的3	24：J	44：F10	64：保留？
05：主键盘的4	25：K	45：NumLock	65：保留？
06：主键盘的5	26：L	46：ScrollLock	66：保留？
07：主键盘的6	27：;	47：数字键的7	67：保留？
08：主键盘的7	28：:（在英语键盘是'）	48：数字键的8	68：保留？
09：主键盘的8	29：全角・半角（在英语键盘是`）	49：数字键的9	69：保留？
0A：主键盘的9	2A：左Shift	4A：数字键的-	6A：保留？
0B：主键盘的0	2B：]（在英语键盘是backslash（反斜线））	4B：数字键的4	6B：保留？
0C：主键盘的-	2C：Z	4C：数字键的5	6C：保留？
0D：主键盘的^（在英语键盘是=）	2D：X	4D：数字键的6	6D：保留？
0E：退格键	2E：C	4E：数字键的+	6E：保留？
0F：TAB键	2F：V	4F：数字键的1	6F：保留？
10：Q	30：B	50：数字键的2	70：平假名（日文键盘）
11：W	31：N	51：数字键的3	71：保留？
12：E	32：M	52：数字键的0	72：保留？
13：R	33：,	53：数字键的.	73：_
14：T	34：.	54：SysReq	74：保留？
15：Y	35：/	55：保留？	75：保留？
16：U	36：右Shift	56：保留？	76：保留？
17：I	37：数字键的*	57：F11	77：保留？
18：O	38：左Alt	58：F12	78：保留？
19：P	39：Space	59：保留？	79：变换（日文键盘）
1A：@（在英语键盘是[）	3A：CapsLock	5A：保留？	7A：保留？
1B：[（在英语键盘是]）	3B：F1	5B：保留？	7B：无变换（日文键盘）
1C：主键盘的Enter	3C：F2	5C：保留？	7C：保留？
1D：左Ctrl	3D：F3	5D：保留？	7D：\
1E：A	3E：F4	5E：保留？	7E：保留？
1F：S	3F：F5	5F：保留？	7F：保留？

按键弹起的数值是按下的数值加上0x80

根据这张表，我们可以将显示的内容从之前的一个字节十六进制数变为可读的。

看起来这张表和我们用的标准美式键盘不太一样，所以我在作者的基础上稍微修改了几个字符，不过修改的还不完全。

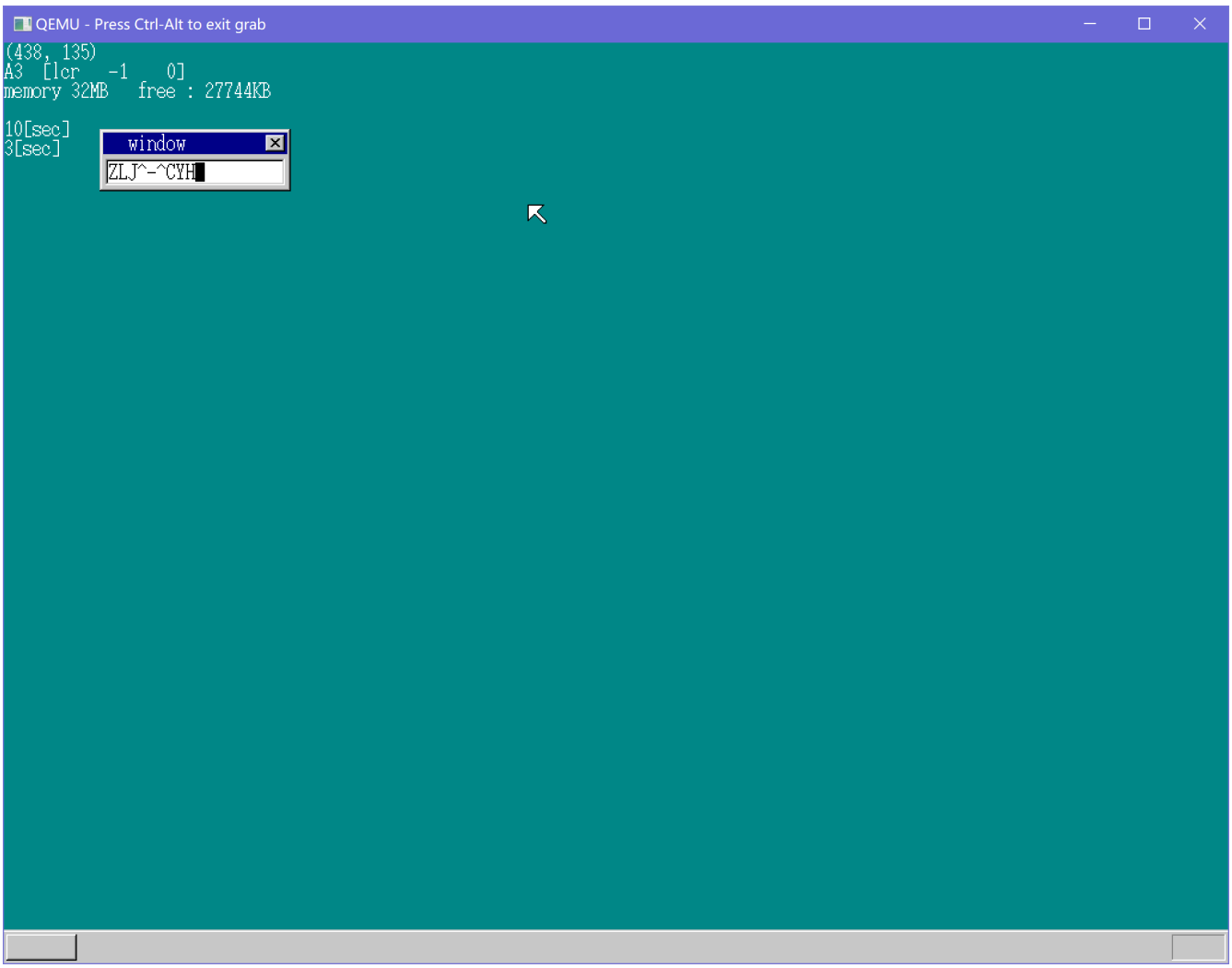
```
static char keytable[0x54] = {
        0,   0,   '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-', '^', 0,   0,
        'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', '@', '[', 0,   0,   'A', 'S',
        'D', 'F', 'G', 'H', 'J', 'K', 'L', ';', ':', 0,   0,   ']', 'Z', 'X', 'C', 'V',
        'B', 'N', 'M', ',', '.', '/', 0,   '*', 0,   ' ', 0,   0,   0,   0,   0,   0,
        0,   0,   0,   0,   0,   0,   0,   '7', '8', '9', '-', '4', '5', '6', '+', '1',
        '2', '3', '0', '.'
    };
```



我们还可以以此来模仿一个简单的输入框，它的主要原理是遇到按键光标和打印位置都向右移动8个像素，按退格的时候则各像左移动8个像素

```
if (256 <= i && i <= 511) {
                sprintf(s, "%02X", i - 256);
                putfonts8_asc_sht(sht_back, 0, 16, COL8_FFFFFF, COL8_008484, s, 2);
                if (i < 0x54 + 256) {
                    if (keytable[i - 256] != 0 && cursor_x < 144) {
                        s[0] = keytable[i - 256];
                        s[1] = 0;
```

```
                        putfonts8_asc_sht(sht_win, cursor_x, 28, COL8_000000, COL8_FFFFFF,
s, 1);
                        cursor_x += 8; // 输入处理
                    }
                }
                if (i == 256 + 0x0e && cursor_x > 8) {
                    putfonts8_asc_sht(sht_win, cursor_x, 28, COL8_000000, COL8_FFFFFF, " ",
1);
                    cursor_x -= 8;
                    // 退格处理
                }
                boxfill8(sht_win->buf, sht_win->bxsize, cursor_c, cursor_x, 28, cursor_x +
7, 43);
                sheet_refresh(sht_win, cursor_x, 28, cursor_x + 8, 44);
            }
```



然后我们制作窗体移动的功能，其实比较简单，我们只要修改sheet左上角的坐标就好了，让他为鼠标点击位置减去一个固定的向量。

```
if ((mdec.btn & 0x01) != 0) {
    sheet_slide(sht_win, mx - 80, my - 8);
}
```

ok!