

```

1  org 10000h
2  jmp Label_Start
3
4  %include "fat12.inc"
5
6  BaseOfKernelFile equ 0x00
7  OffsetOfKernelFile equ 0x100000
8
9  BaseTmpOfKernelAddr equ 0x00
10 OffsetTmpOfKernelFile equ 0x7E00
11
12 MemoryStructBufferAddr equ 0x7E00
13
14 [SECTION gdt]
15
16 LABEL_GDT: dd 0,0
17 LABEL_DESC_CODE32: dd 0x0000FFFF,0x00CF9A00
18 LABEL_DESC_DATA32: dd 0x0000FFFF,0x00CF9200
19
20 GdtLen equ $ - LABEL_GDT
21 GdtPtr dw GdtLen - 1
22 dd LABEL_GDT
23
24 SelectorCode32 equ LABEL_DESC_CODE32 - LABEL_GDT
25 SelectorData32 equ LABEL_DESC_DATA32 - LABEL_GDT
26
27 [SECTION gdt64]
28
29 LABEL_GDT64: dq 0x0000000000000000
30 LABEL_DESC_CODE64: dq 0x0020980000000000
31 LABEL_DESC_DATA64: dq 0x0000920000000000
32
33 GdtLen64 equ $ - LABEL_GDT64
34 GdtPtr64 dw GdtLen64 - 1
35 dd LABEL_GDT64
36
37 SelectorCode64 equ LABEL_DESC_CODE64 - LABEL_GDT64
38 SelectorData64 equ LABEL_DESC_DATA64 - LABEL_GDT64
39
40 [SECTION .s16]
41 [BITS 16]
42
43 Label_Start:
44
45 mov ax, cs
46 mov ds, ax
47 mov es, ax
48 mov ax, 0x00
49 mov ss, ax
50 mov sp, 0x7c00
51
52 ;===== display on screen : Start Loader.....
53
54 mov ax, 1301h
55 mov bx, 000fh
56 mov dx, 0200h ;row 2
57 mov cx, 12
58 push ax
59 mov ax, ds
60 mov es, ax
61 pop ax
62 mov bp, StartLoaderMessage
63 int 10h
64
65 ;===== open address A20
66 push ax
67 in al, 92h
68 or al, 00000010b
69 out 92h, al
70 pop ax
71

```

```

72     cli
73
74     db  0x66
75     lgdt    [GdtPtr]
76
77     mov eax,    cr0
78     or  eax,    1
79     mov cr0,    eax
80
81     mov ax, SelectorData32
82     mov fs, ax
83     mov eax,    cr0
84     and al, 11111110b
85     mov cr0,    eax
86
87     sti
88
89 ;=====    reset floppy
90
91     xor ah, ah
92     xor dl, dl
93     int 13h
94
95 ;=====    search kernel.bin
96     mov word    [SectorNo], SectorNumOfRootDirStart
97
98 Lable_Search_In_Root_Dir_Begin:
99
100     cmp word    [RootDirSizeForLoop],    0
101     jz  Label_No_LoaderBin
102     dec word    [RootDirSizeForLoop]
103     mov ax, 00h
104     mov es, ax
105     mov bx, 8000h
106     mov ax, [SectorNo]
107     mov cl, 1
108     call Func_ReadOneSector
109     mov si, KernelFileName
110     mov di, 8000h
111     cld
112     mov dx, 10h
113
114 Label_Search_For_LoaderBin:
115
116     cmp dx, 0
117     jz  Label_Goto_Next_Sector_In_Root_Dir
118     dec dx
119     mov cx, 11
120
121 Label_Cmp_FileName:
122
123     cmp cx, 0
124     jz  Label_FileName_Found
125     dec cx
126     lodsb
127     cmp al, byte    [es:di]
128     jz  Label_Go_On
129     jmp Label_Different
130
131 Label_Go_On:
132
133     inc di
134     jmp Label_Cmp_FileName
135
136 Label_Different:
137
138     and di, 0FFE0h
139     add di, 20h
140     mov si, KernelFileName
141     jmp Label_Search_For_LoaderBin
142

```

```

143 Label_Goto_Next_Sector_In_Root_Dir:
144
145     add word    [SectorNo], 1
146     jmp  Lable_Search_In_Root_Dir_Begin
147
148 ;=====      display on screen : ERROR:No KERNEL Found
149
150 Label_No_LoaderBin:
151
152     mov ax, 1301h
153     mov bx, 008Ch
154     mov dx, 0300h      ;row 3
155     mov cx, 21
156     push ax
157     mov ax, ds
158     mov es, ax
159     pop ax
160     mov bp, NoLoaderMessage
161     int 10h
162     jmp $
163
164 ;=====      found loader.bin name in root director struct
165
166 Label_FileName_Found:
167     mov ax, RootDirSectors
168     and di, 0FFE0h
169     add di, 01Ah
170     mov cx, word    [es:di]
171     push cx
172     add cx, ax
173     add cx, SectorBalance
174     mov eax,      BaseTmpOfKernelAddr ;BaseOfKernelFile
175     mov es, eax
176     mov bx, OffsetTmpOfKernelFile      ;OffsetOfKernelFile
177     mov ax, cx
178
179 Label_Go_On_Loading_File:
180     push ax
181     push bx
182     mov ah, 0Eh
183     mov al, '.'
184     mov bl, 0Fh
185     int 10h
186     pop bx
187     pop ax
188
189     mov cl, 1
190     call Func_ReadOneSector
191     pop ax
192
193 ;;;;;;;;;;;;;;;;;;;;;;;;;;
194     push cx
195     push eax
196     push fs
197     push edi
198     push ds
199     push esi
200
201     mov cx, 200h
202     mov ax, BaseOfKernelFile
203     mov fs, ax
204     mov edi,      dword    [OffsetOfKernelFileCount]
205
206     mov ax, BaseTmpOfKernelAddr
207     mov ds, ax
208     mov esi,      OffsetTmpOfKernelFile
209
210 Label_Mov_Kernel:      ;-----
211
212     mov al, byte    [ds:esi]
213     mov byte    [fs:edi], al

```

```

214
215     inc esi
216     inc edi
217
218     loop    Label_Mov_Kernel
219
220     mov eax,    0x1000
221     mov ds,    eax
222
223     mov dword   [OffsetOfKernelFileCount],    edi
224
225     pop esi
226     pop ds
227     pop edi
228     pop fs
229     pop eax
230     pop cx
231     ;;;;;;;;;;;;;;;;;;
232
233     call    Func_GetFATEntry
234     cmp ax,    0FFFh
235     jz     Label_File_Loaded
236     push    ax
237     mov dx,    RootDirSectors
238     add ax,    dx
239     add ax,    SectorBalance
240
241     jmp     Label_Go_On_Loading_File
242
243 Label_File_Loaded:
244
245     mov ax,    0B800h
246     mov gs,    ax
247     mov ah,    0Fh                ; 0000: 黑底      1111: 白字
248     mov al,    'G'
249     mov [gs:((80 * 0 + 39) * 2)], ax    ; 屏幕第 0 行, 第 39 列。
250
251 KillMotor:
252
253     push    dx
254     mov dx,    03F2h
255     mov al,    0
256     out dx,    al
257     pop dx
258
259 ;=====    get memory address size type
260
261     mov ax,    1301h
262     mov bx,    000Fh
263     mov dx,    0400h                ;row 4
264     mov cx,    24
265     push    ax
266     mov ax,    ds
267     mov es,    ax
268     pop ax
269     mov bp,    StartGetMemStructMessage
270     int 10h
271
272     mov ebx,    0
273     mov ax,    0x00
274     mov es,    ax
275     mov di,    MemoryStructBufferAddr
276
277 Label_Get_Mem_Struct:
278
279     mov eax,    0x0E820
280     mov ecx,    20
281     mov edx,    0x534D4150
282     int 15h
283     jc     Label_Get_Mem_Fail
284     add di,    20

```

```

285
286     cmp ebx, 0
287     jne Label_Get_Mem_Struct
288     jmp Label_Get_Mem_OK
289
290 Label_Get_Mem_Fail:
291
292     mov ax, 1301h
293     mov bx, 008Ch
294     mov dx, 0500h ;row 5
295     mov cx, 23
296     push ax
297     mov ax, ds
298     mov es, ax
299     pop ax
300     mov bp, GetMemStructErrorMessage
301     int 10h
302     jmp $
303
304 Label_Get_Mem_OK:
305
306     mov ax, 1301h
307     mov bx, 000Fh
308     mov dx, 0600h ;row 6
309     mov cx, 29
310     push ax
311     mov ax, ds
312     mov es, ax
313     pop ax
314     mov bp, GetMemStructOKMessage
315     int 10h
316
317 ;===== get SVGA information
318
319     mov ax, 1301h
320     mov bx, 000Fh
321     mov dx, 0800h ;row 8
322     mov cx, 23
323     push ax
324     mov ax, ds
325     mov es, ax
326     pop ax
327     mov bp, StartGetSVGAVBEInfoMessage
328     int 10h
329
330     mov ax, 0x00
331     mov es, ax
332     mov di, 0x8000
333     mov ax, 4F00h
334
335     int 10h
336
337     cmp ax, 004Fh
338
339     jz .KO
340
341 ;===== Fail
342
343     mov ax, 1301h
344     mov bx, 008Ch
345     mov dx, 0900h ;row 9
346     mov cx, 23
347     push ax
348     mov ax, ds
349     mov es, ax
350     pop ax
351     mov bp, GetSVGAVBEInfoErrorMessage
352     int 10h
353
354     jmp $
355

```

```

356 .KO:
357
358     mov ax, 1301h
359     mov bx, 000Fh
360     mov dx, 0A00h        ;row 10
361     mov cx, 29
362     push ax
363     mov ax, ds
364     mov es, ax
365     pop ax
366     mov bp, GetSVGAVBEInfoOKMessage
367     int 10h
368
369 ;=====    Get SVGA Mode Info
370
371     mov ax, 1301h
372     mov bx, 000Fh
373     mov dx, 0C00h        ;row 12
374     mov cx, 24
375     push ax
376     mov ax, ds
377     mov es, ax
378     pop ax
379     mov bp, StartGetSVGAModeInfoMessage
380     int 10h
381
382
383     mov ax, 0x00
384     mov es, ax
385     mov si, 0x800e
386
387     mov esi,    dword    [es:si]
388     mov edi,    0x8200
389
390 Label_SVGA_Mode_Info_Get:
391
392     mov cx, word    [es:esi]
393
394 ;=====    display SVGA mode information
395
396     push ax
397
398     mov ax, 00h
399     mov al, ch
400     call Label_DisPAL
401
402     mov ax, 00h
403     mov al, cl
404     call Label_DisPAL
405
406     pop ax
407
408 ;=====
409
410     cmp cx, 0FFFFh
411     jz  Label_SVGA_Mode_Info_Finish
412
413     mov ax, 4F01h
414     int 10h
415
416     cmp ax, 004Fh
417
418     jnz Label_SVGA_Mode_Info_FAIL
419
420     add esi,    2
421     add edi,    0x100
422
423     jmp Label_SVGA_Mode_Info_Get
424
425 Label_SVGA_Mode_Info_FAIL:
426

```

```

427     mov ax, 1301h
428     mov bx, 008Ch
429     mov dx, 0D00h           ;row 13
430     mov cx, 24
431     push ax
432     mov ax, ds
433     mov es, ax
434     pop ax
435     mov bp, GetSVGAModeInfoErrMsg
436     int 10h
437
438 Label_SET_SVGA_Mode_VESA_VBE_FAIL:
439
440     jmp $
441
442 Label_SVGA_Mode_Info_Finish:
443
444     mov ax, 1301h
445     mov bx, 000Fh
446     mov dx, 0E00h           ;row 14
447     mov cx, 30
448     push ax
449     mov ax, ds
450     mov es, ax
451     pop ax
452     mov bp, GetSVGAModeInfoOKMessage
453     int 10h
454
455 ;===== set the SVGA mode(VESA VBE)
456
457     mov ax, 4F02h
458     mov bx, 4180h           ;=====mode : 0x180 or 0x143
459     int 10h
460
461     cmp ax, 004Fh
462     jnz Label_SET_SVGA_Mode_VESA_VBE_FAIL
463
464 ;===== init IDT GDT goto protect mode
465
466     cli                     ;=====close interrupt
467
468     db 0x66
469     lgdt [GdtPtr]
470
471 ; db 0x66
472 ; lidt [IDT_POINTER]
473
474     mov eax, cr0
475     or  eax, 1
476     mov cr0, eax
477
478     jmp dword SelectorCode32:GO_TO_TMP_Protect
479
480 [SECTION .s32]
481 [BITS 32]
482
483 GO_TO_TMP_Protect:
484
485 ;===== go to tmp long mode
486
487     mov ax, 0x10
488     mov ds, ax
489     mov es, ax
490     mov fs, ax
491     mov ss, ax
492     mov esp, 7E00h
493
494     call support_long_mode
495     test eax, eax
496
497     jz no_support

```

```

498
499 ;=====    init temporary page table 0x90000
500
501     mov dword    [0x90000],    0x91007
502     mov dword    [0x90800],    0x91007
503
504     mov dword    [0x91000],    0x92007
505
506     mov dword    [0x92000],    0x000083
507
508     mov dword    [0x92008],    0x200083
509
510     mov dword    [0x92010],    0x400083
511
512     mov dword    [0x92018],    0x600083
513
514     mov dword    [0x92020],    0x800083
515
516     mov dword    [0x92028],    0xa00083
517
518 ;=====    load GDTR
519
520     db    0x66
521     lgdt    [GdtPtr64]
522     mov ax, 0x10
523     mov ds, ax
524     mov es, ax
525     mov fs, ax
526     mov gs, ax
527     mov ss, ax
528
529     mov esp,    7E00h
530
531 ;=====    open PAE
532
533     mov eax,    cr4
534     bts eax,    5
535     mov cr4,    eax
536
537 ;=====    load    cr3
538
539     mov eax,    0x90000
540     mov cr3,    eax
541
542 ;=====    enable long-mode
543
544     mov ecx,    0C0000080h    ; IA32_EFER
545     rdmsr
546
547     bts eax,    8
548     wrmsr
549
550 ;=====    open PE and paging
551
552     mov eax,    cr0
553     bts eax,    0
554     bts eax,    31
555     mov cr0,    eax
556
557     jmp SelectorCode64:OffsetOfKernelFile
558
559 ;=====    test support long mode or not
560
561 support_long_mode:
562
563     mov eax,    0x80000000
564     cpuid
565     cmp eax,    0x80000001
566     setnb    al
567     jb    support_long_mode_done
568     mov eax,    0x80000001

```



```

569     cpuid
570     bt  edx,    29
571     setc  al
572 support_long_mode_done:
573
574     movzx  eax,    al
575     ret
576
577 ;=====    no support
578
579 no_support:
580     jmp $
581
582 ;=====    read one sector from floppy
583
584 [SECTION .s16lib]
585 [BITS 16]
586
587 Func_ReadOneSector:
588
589     push    bp
590     mov bp, sp
591     sub esp,    2
592     mov byte    [bp - 2],    cl
593     push    bx
594     mov bl, [BPB_SecPerTrk]
595     div bl
596     inc ah
597     mov cl, ah
598     mov dh, al
599     shr al, 1
600     mov ch, al
601     and dh, 1
602     pop bx
603     mov dl, [BS_DrvNum]
604 Label_Go_On_Reading:
605     mov ah, 2
606     mov al, byte    [bp - 2]
607     int 13h
608     jc  Label_Go_On_Reading
609     add esp,    2
610     pop bp
611     ret
612
613 ;=====    get FAT Entry
614
615 Func_GetFATEntry:
616
617     push    es
618     push    bx
619     push    ax
620     mov ax, 00
621     mov es, ax
622     pop ax
623     mov byte    [Odd],    0
624     mov bx, 3
625     mul bx
626     mov bx, 2
627     div bx
628     cmp dx, 0
629     jz  Label_Even
630     mov byte    [Odd],    1
631
632 Label_Even:
633
634     xor dx, dx
635     mov bx, [BPB_BytesPerSec]
636     div bx
637     push    dx
638     mov bx, 8000h
639     add ax, SectorNumOfFAT1Start

```

```

640     mov cl, 2
641     call Func_ReadOneSector
642
643     pop dx
644     add bx, dx
645     mov ax, [es:bx]
646     cmp byte [Odd], 1
647     jnz Label_Even_2
648     shr ax, 4
649
650 Label_Even_2:
651     and ax, 0FFFh
652     pop bx
653     pop es
654     ret
655
656 ;=====      display num in al
657
658 Label_DisAL:
659
660     push     ecx
661     push     edx
662     push     edi
663
664     mov edi, [DisplayPosition]
665     mov ah, 0Fh
666     mov dl, al
667     shr al, 4
668     mov ecx, 2
669 .begin:
670
671     and al, 0Fh
672     cmp al, 9
673     ja .1
674     add al, '0'
675     jmp .2
676 .1:
677
678     sub al, 0Ah
679     add al, 'A'
680 .2:
681
682     mov [gs:edi], ax
683     add edi, 2
684
685     mov al, dl
686     loop .begin
687
688     mov [DisplayPosition], edi
689
690     pop edi
691     pop edx
692     pop ecx
693
694     ret
695
696 ;=====      tmp IDT
697
698 IDT:
699     times 0x50    dq 0
700 IDT_END:
701
702 IDT_POINTER:
703     dw IDT_END - IDT - 1
704     dd IDT
705
706 ;=====      tmp variable
707
708 RootDirSizeForLoop dw RootDirSectors
709 SectorNo           dw 0
710

```

```
711 Odd db 0
712 OffsetOfKernelFileCount dd OffsetOfKernelFile
713
714 DisplayPosition dd 0
715
716 ;===== display messages
717
718 StartLoaderMessage: db "Start Loader"
719 NoLoaderMessage: db "ERROR:No KERNEL Found"
720 KernelFileName: db "KERNEL BIN",0
721 StartGetMemStructMessage: db "Start Get Memory Struct."
722 GetMemStructErrMsg: db "Get Memory Struct ERROR"
723 GetMemStructOKMessage: db "Get Memory Struct SUCCESSFUL!"
724
725 StartGetSVGAVBEInfoMessage: db "Start Get SVGA VBE Info"
726 GetSVGAVBEInfoErrMsg: db "Get SVGA VBE Info ERROR"
727 GetSVGAVBEInfoOKMessage: db "Get SVGA VBE Info SUCCESSFUL!"
728
729 StartGetSVGAModeInfoMessage: db "Start Get SVGA Mode Info"
730 GetSVGAModeInfoErrMsg: db "Get SVGA Mode Info ERROR"
731 GetSVGAModeInfoOKMessage: db "Get SVGA Mode Info SUCCESSFUL!"
732
```