```
org 10000h
    jmp Label_Start

%include    "fat12.inc"

BaseOfKernelFile    equ 0x00
OffsetOfKernelFile  equ 0x100000

BaseTmpOfKernelAddr equ 0x00
OffsetTmpOfKernelFile   equ 0x7E00

MemoryStructBufferAddr  equ 0x7E00

[SECTION gdt]

LABEL_GDT:      dd  0,0
LABEL_DESC_CODE32:  dd  0x0000FFFF,0x00CF9A00
LABEL_DESC_DATA32:  dd  0x0000FFFF,0x00CF9200

GdtLen  equ $ - LABEL_GDT
GdtPtr  dw  GdtLen - 1
    dd  LABEL_GDT

SelectorCode32  equ LABEL_DESC_CODE32 - LABEL_GDT
SelectorData32  equ LABEL_DESC_DATA32 - LABEL_GDT

[SECTION gdt64]

LABEL_GDT64:        dq  0x0000000000000000
LABEL_DESC_CODE64:  dq  0x0020980000000000
LABEL_DESC_DATA64:  dq  0x0000920000000000

GdtLen64    equ $ - LABEL_GDT64
GdtPtr64    dw  GdtLen64 - 1
        dd  LABEL_GDT64

SelectorCode64  equ LABEL_DESC_CODE64 - LABEL_GDT64
SelectorData64  equ LABEL_DESC_DATA64 - LABEL_GDT64

[SECTION .s16]
[BITS 16]

Label_Start:

    mov ax, cs
    mov ds, ax
    mov es, ax
    mov ax, 0x00
    mov ss, ax
    mov sp, 0x7c00

;=======    display on screen : Start Loader......

    mov ax, 1301h
    mov bx, 000fh
    mov dx, 0200h       ;row 2
    mov cx, 12
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, StartLoaderMessage
    int 10h

;=======    open address A20
    push    ax
    in  al, 92h
    or  al, 00000010b
    out 92h,    al
    pop ax

```

```
	cli

	db	0x66
	lgdt	[GdtPtr]

	mov	eax,	cr0
	or	eax,	1
	mov	cr0,	eax

	mov	ax,	SelectorData32
	mov	fs,	ax
	mov	eax,	cr0
	and	al,	11111110b
	mov	cr0,	eax

	sti

;=======	reset floppy

	xor	ah,	ah
	xor	dl,	dl
	int	13h

;=======	search kernel.bin
	mov	word	[SectorNo],	SectorNumOfRootDirStart

Lable_Search_In_Root_Dir_Begin:

	cmp	word	[RootDirSizeForLoop],	0
	jz	Label_No_LoaderBin
	dec	word	[RootDirSizeForLoop]
	mov	ax,	00h
	mov	es,	ax
	mov	bx,	8000h
	mov	ax,	[SectorNo]
	mov	cl,	1
	call	Func_ReadOneSector
	mov	si,	KernelFileName
	mov	di,	8000h
	cld
	mov	dx,	10h

Label_Search_For_LoaderBin:

	cmp	dx,	0
	jz	Label_Goto_Next_Sector_In_Root_Dir
	dec	dx
	mov	cx,	11

Label_Cmp_FileName:

	cmp	cx,	0
	jz	Label_FileName_Found
	dec	cx
	lodsb
	cmp	al,	byte	[es:di]
	jz	Label_Go_On
	jmp	Label_Different

Label_Go_On:

	inc	di
	jmp	Label_Cmp_FileName

Label_Different:

	and	di,	0FFE0h
	add	di,	20h
	mov	si,	KernelFileName
	jmp	Label_Search_For_LoaderBin

```

```
Label_Goto_Next_Sector_In_Root_Dir:

    add word    [SectorNo], 1
    jmp Lable_Search_In_Root_Dir_Begin

;=======    display on screen : ERROR:No KERNEL Found

Label_No_LoaderBin:

    mov ax, 1301h
    mov bx, 008Ch
    mov dx, 0300h       ;row 3
    mov cx, 21
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, NoLoaderMessage
    int 10h
    jmp $

;=======    found loader.bin name in root director struct

Label_FileName_Found:
    mov ax, RootDirSectors
    and di, 0FFE0h
    add di, 01Ah
    mov cx, word    [es:di]
    push    cx
    add cx, ax
    add cx, SectorBalance
    mov eax,    BaseTmpOfKernelAddr ;BaseOfKernelFile
    mov es, eax
    mov bx, OffsetTmpOfKernelFile   ;OffsetOfKernelFile
    mov ax, cx

Label_Go_On_Loading_File:
    push    ax
    push    bx
    mov ah, 0Eh
    mov al, '.'
    mov bl, 0Fh
    int 10h
    pop bx
    pop ax

    mov cl, 1
    call    Func_ReadOneSector
    pop ax

;;;;;;;;;;;;;;;;;;;;;;;
    push    cx
    push    eax
    push    fs
    push    edi
    push    ds
    push    esi

    mov cx, 200h
    mov ax, BaseOfKernelFile
    mov fs, ax
    mov edi,    dword   [OffsetOfKernelFileCount]

    mov ax, BaseTmpOfKernelAddr
    mov ds, ax
    mov esi,    OffsetTmpOfKernelFile

Label_Mov_Kernel:   ;------------------

    mov al, byte    [ds:esi]
    mov byte    [fs:edi],   al

    inc esi
    inc edi

    loop    Label_Mov_Kernel

    mov eax,    0x1000
    mov ds, eax

    mov dword   [OffsetOfKernelFileCount],  edi

    pop esi
    pop ds
    pop edi
    pop fs
    pop eax
    pop cx
;;;;;;;;;;;;;;;;;;;;;;;

    call    Func_GetFATEntry
    cmp ax, 0FFFh
    jz  Label_File_Loaded
    push    ax
    mov dx, RootDirSectors
    add ax, dx
    add ax, SectorBalance

    jmp Label_Go_On_Loading_File

Label_File_Loaded:

    mov ax, 0B800h
    mov gs, ax
    mov ah, 0Fh             ; 0000: 黑底    1111: 白字
    mov al, 'G'
    mov [gs:((80 * 0 + 39) * 2)], ax    ; 屏幕第 0 行, 第 39 列。

KillMotor:

    push    dx
    mov dx, 03F2h
    mov al, 0
    out dx, al
    pop dx

;=======    get memory address size type

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0400h       ;row 4
    mov cx, 24
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, StartGetMemStructMessage
    int 10h

    mov ebx,    0
    mov ax, 0x00
    mov es, ax
    mov di, MemoryStructBufferAddr

Label_Get_Mem_Struct:

    mov eax,    0x0E820
    mov ecx,    20
    mov edx,    0x534D4150
    int 15h
    jc  Label_Get_Mem_Fail
    add di, 20
    cmp ebx,    0
    jne Label_Get_Mem_Struct
    jmp Label_Get_Mem_OK

Label_Get_Mem_Fail:

    mov ax, 1301h
    mov bx, 008Ch
    mov dx, 0500h       ;row 5
    mov cx, 23
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetMemStructErrMessage
    int 10h
    jmp $

Label_Get_Mem_OK:

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0600h       ;row 6
    mov cx, 29
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetMemStructOKMessage
    int 10h

;=======    get SVGA information

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0800h       ;row 8
    mov cx, 23
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, StartGetSVGAVBEInfoMessage
    int 10h

    mov ax, 0x00
    mov es, ax
    mov di, 0x8000
    mov ax, 4F00h

    int 10h

    cmp ax, 004Fh

    jz  .KO

;=======    Fail

    mov ax, 1301h
    mov bx, 008Ch
    mov dx, 0900h       ;row 9
    mov cx, 23
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetSVGAVBEInfoErrMessage
    int 10h

    jmp $

```

```
.KO:

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0A00h       ;row 10
    mov cx, 29
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetSVGAVBEInfoOKMessage
    int 10h

;=======    Get SVGA Mode Info

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0C00h       ;row 12
    mov cx, 24
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, StartGetSVGAModeInfoMessage
    int 10h


    mov ax, 0x00
    mov es, ax
    mov si, 0x800e

    mov esi,    dword   [es:si]
    mov edi,    0x8200

Label_SVGA_Mode_Info_Get:

    mov cx, word    [es:esi]

;=======    display SVGA mode information

    push    ax

    mov ax, 00h
    mov al, ch
    call    Label_DispAL

    mov ax, 00h
    mov al, cl
    call    Label_DispAL

    pop ax

;=======

    cmp cx, 0FFFFh
    jz  Label_SVGA_Mode_Info_Finish

    mov ax, 4F01h
    int 10h

    cmp ax, 004Fh

    jnz Label_SVGA_Mode_Info_FAIL

    add esi,    2
    add edi,    0x100

    jmp Label_SVGA_Mode_Info_Get

Label_SVGA_Mode_Info_FAIL:

    mov ax, 1301h
    mov bx, 008Ch
    mov dx, 0D00h       ;row 13
    mov cx, 24
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetSVGAModeInfoErrMessage
    int 10h

Label_SET_SVGA_Mode_VESA_VBE_FAIL:

    jmp $

Label_SVGA_Mode_Info_Finish:

    mov ax, 1301h
    mov bx, 000Fh
    mov dx, 0E00h       ;row 14
    mov cx, 30
    push    ax
    mov ax, ds
    mov es, ax
    pop ax
    mov bp, GetSVGAModeInfoOKMessage
    int 10h

;=======    set the SVGA mode(VESA VBE)

    mov ax, 4F02h
    mov bx, 4180h   ;========================mode : 0x180 or 0x143
    int     10h

    cmp ax, 004Fh
    jnz Label_SET_SVGA_Mode_VESA_VBE_FAIL

;=======    init IDT GDT goto protect mode

    cli         ;======close interrupt

    db  0x66
    lgdt    [GdtPtr]

;   db  0x66
;   lidt    [IDT_POINTER]

    mov eax,    cr0
    or  eax,    1
    mov cr0,    eax

    jmp dword SelectorCode32:GO_TO_TMP_Protect

[SECTION .s32]
[BITS 32]

GO_TO_TMP_Protect:

;=======    go to tmp long mode

    mov ax, 0x10
    mov ds, ax
    mov es, ax
    mov fs, ax
    mov ss, ax
    mov esp,    7E00h

    call    support_long_mode
    test    eax,    eax

    jz  no_support
```

```

;=======	init temporary page table 0x90000

	mov	dword	[0x90000],	0x91007
	mov	dword	[0x90800],	0x91007

	mov	dword	[0x91000],	0x92007

	mov	dword	[0x92000],	0x000083

	mov	dword	[0x92008],	0x200083

	mov	dword	[0x92010],	0x400083

	mov	dword	[0x92018],	0x600083

	mov	dword	[0x92020],	0x800083

	mov	dword	[0x92028],	0xa00083

;=======	load GDTR

	db	0x66
	lgdt	[GdtPtr64]
	mov	ax,	0x10
	mov	ds,	ax
	mov	es,	ax
	mov	fs,	ax
	mov	gs,	ax
	mov	ss,	ax

	mov	esp,	7E00h

;=======	open PAE

	mov	eax,	cr4
	bts	eax,	5
	mov	cr4,	eax

;=======	load	cr3

	mov	eax,	0x90000
	mov	cr3,	eax

;=======	enable long-mode

	mov	ecx,	0C0000080h		;IA32_EFER
	rdmsr

	bts	eax,	8
	wrmsr

;=======	open PE and paging

	mov	eax,	cr0
	bts	eax,	0
	bts	eax,	31
	mov	cr0,	eax

	jmp	SelectorCode64:OffsetOfKernelFile

;=======	test support long mode or not

support_long_mode:

	mov	eax,	0x80000000
	cpuid
	cmp	eax,	0x80000001
	setnb	al
	jb	support_long_mode_done
	mov	eax,	0x80000001
```

```
    cpuid
    bt  edx,    29
    setc    al
support_long_mode_done:

    movzx   eax,    al
    ret

;=======    no support

no_support:
    jmp $

;=======    read one sector from floppy

[SECTION .s16lib]
[BITS 16]

Func_ReadOneSector:

    push    bp
    mov bp, sp
    sub esp,    2
    mov byte    [bp - 2],   cl
    push    bx
    mov bl, [BPB_SecPerTrk]
    div bl
    inc ah
    mov cl, ah
    mov dh, al
    shr al, 1
    mov ch, al
    and dh, 1
    pop bx
    mov dl, [BS_DrvNum]
Label_Go_On_Reading:
    mov ah, 2
    mov al, byte    [bp - 2]
    int 13h
    jc  Label_Go_On_Reading
    add esp,    2
    pop bp
    ret

;=======    get FAT Entry

Func_GetFATEntry:

    push    es
    push    bx
    push    ax
    mov ax, 00
    mov es, ax
    pop ax
    mov byte    [Odd],  0
    mov bx, 3
    mul bx
    mov bx, 2
    div bx
    cmp dx, 0
    jz  Label_Even
    mov byte    [Odd],  1

Label_Even:

    xor dx, dx
    mov bx, [BPB_BytesPerSec]
    div bx
    push    dx
    mov bx, 8000h
    add ax, SectorNumOfFAT1Start
    mov cl, 2
    call    Func_ReadOneSector

    pop dx
    add bx, dx
    mov ax, [es:bx]
    cmp byte    [Odd],  1
    jnz Label_Even_2
    shr ax, 4

Label_Even_2:
    and ax, 0FFFh
    pop bx
    pop es
    ret

;=======    display num in al

Label_DispAL:

    push    ecx
    push    edx
    push    edi

    mov edi,    [DisplayPosition]
    mov ah, 0Fh
    mov dl, al
    shr al, 4
    mov ecx,    2
.begin:

    and al, 0Fh
    cmp al, 9
    ja  .1
    add al, '0'
    jmp .2
.1:

    sub al, 0Ah
    add al, 'A'
.2:

    mov [gs:edi],   ax
    add edi,    2

    mov al, dl
    loop    .begin

    mov [DisplayPosition],  edi

    pop edi
    pop edx
    pop ecx

    ret


;=======    tmp IDT

IDT:
    times   0x50    dq  0
IDT_END:

IDT_POINTER:
        dw  IDT_END - IDT - 1
        dd  IDT

;=======    tmp variable

RootDirSizeForLoop  dw  RootDirSectors
SectorNo        dw  0
```

```
Odd          db   0
OffsetOfKernelFileCount dd  OffsetOfKernelFile

DisplayPosition     dd   0

;=======    display messages

StartLoaderMessage: db   "Start Loader"
NoLoaderMessage:    db   "ERROR:No KERNEL Found"
KernelFileName:     db   "KERNEL  BIN",0
StartGetMemStructMessage:   db   "Start Get Memory Struct."
GetMemStructErrMessage: db   "Get Memory Struct ERROR"
GetMemStructOKMessage:  db   "Get Memory Struct SUCCESSFUL!"

StartGetSVGAVBEInfoMessage: db   "Start Get SVGA VBE Info"
GetSVGAVBEInfoErrMessage:   db   "Get SVGA VBE Info ERROR"
GetSVGAVBEInfoOKMessage:    db   "Get SVGA VBE Info SUCCESSFUL!"

StartGetSVGAModeInfoMessage:    db   "Start Get SVGA Mode Info"
GetSVGAModeInfoErrMessage:  db   "Get SVGA Mode Info ERROR"
GetSVGAModeInfoOKMessage:   db   "Get SVGA Mode Info SUCCESSFUL!"
```