1. Setting up a bootstrap

10 November 2014 19:15

```
section .text
; BEGIN - Multiboot Signature
MultibootSignature dd 464367618
MultibootFlags dd 3
MultibootChecksum dd -464367621
MultibootGraphicsRuntime_VbeModeInfoAddr dd 2147483647
MultibootGraphicsRuntime_VbeControlInfoAddr dd 2147483647
MultibootGraphicsRuntime_VbeMode dd 2147483647
MultiBootInfo_Memory_High dd 0
MultiBootInfo_Memory_Low dd 0
MultiBootInfo_Structure dd 0
; END - Multiboot Signature
; BEGIN - Stack Memory Allocation
global Before_Kernel_Stack
Before_Kernel_Stack: TIMES 65535 db 0
Kernel_Stack:
; END - Stack Memory Allocation
; BEGIN - Kernel Start
global Kernel_Start
Kernel_Start:
xchg bx, bx
        cli

        ; MultiBoot compliant loader provides info in registers:
        ; EBX=multiboot_info
        ; EAX=0x2BADB002     - check if it's really Multiboot loader
        ;                    - if true, continue and copy mb info

        ; BEGIN - Multiboot Info
        mov dword ecx, 0x2BADB002
        cmp ecx, eax
        jne Kernel_Start_HandleNoMultiboot
mov dword [MultiBootInfo_Structure], EBX
        add dword EBX, 0x4
        mov dword EAX, [EBX]
        mov dword [MultiBootInfo_Memory_Low], EAX
```

```
        add dword EBX, 0x4
        mov dword EAX, [EBX]
        mov dword [MultiBootInfo_Memory_High], EAX
        ; END - Multiboot Info
; Enable Protected Mode
        mov eax, cr0
        or eax, 0x1
        mov cr0, eax

; END - Kernel Start
; BEGIN - Init stack
mov dword ESP, Kernel_Stack ; Set the stack pointer to point at our pre-allocated block of memory
; END - Init stack
; BEGIN - Handle No Multiboot
jmp Kernel_Start_HandleNoMultiboot_End ; Skip over this code - we don't want to run it by accident!
Kernel_Start_HandleNoMultiboot:
; Not entirely sure if we'd ever actually get as far as due to code structure but anyway...
; Displays a warning message to the user saying "No multiboot" indicating the multiboot signature
; (which should have been in eax) was not detected so we don't think we have a valid boot setup
; so we are aborting the boot to avoid damage

        ; Output following text to first bit of vid mem
        ; N    o    M u l t i b o o t
        ; 78 111 32 109 117 108 116 105 98 111 111 116
        mov byte [0xB8000], 78
        mov byte [0xB8002], 111
        mov byte [0xB8004], 32
        mov byte [0xB8006], 109
        mov byte [0xB8008], 117
        mov byte [0xB800A], 108
        mov byte [0xB800C], 116
        mov byte [0xB800E], 105
        mov byte [0xB8010], 98
        mov byte [0xB8012], 111
```

```
        mov byte [0xB8014], 111
        mov byte [0xB8016], 116
; Set the colour of the outputted text to:
        ; Red background (0x4-),
        ; White foreground (0x-F)
        mov dword eax, 0x4F
        mov byte [0xB8001], al
        mov byte [0xB8003], al
        mov byte [0xB8005], al
        mov byte [0xB8007], al
        mov byte [0xB8009], al
        mov byte [0xB800B], al
        mov byte [0xB800D], al
        mov byte [0xB800F], al
        mov byte [0xB8011], al
        mov byte [0xB8013], al
        mov byte [0xB8015], al
        mov byte [0xB8017], al
cli ; Prevent any more interrupt requests re-awakening us
        hlt ; Halt the OS / execution / etc.
        jmp Kernel_Start_HandleNoMultiboot ; Just in case...
Kernel_Start_HandleNoMultiboot_End:
; END - Handle No Multiboot
; BEGIN - Main Entrypoint
call __MAIN_ENTRYPOINT__ ; Call our main entry point
                         ; - not strictly necessary but good for setting up
esp etc.

__MAIN_ENTRYPOINT__:
push dword ebp
        mov dword ebp, esp
; This bit of video output is optional / for testing purposes.
; Output following text to first bit of vid mem
        ; M   u   l   t   i   b   o   o   t
        ; 109 117 108 116 105 98 111 111 116
        mov byte [0xB8000], 109
        mov byte [0xB8002], 117
        mov byte [0xB8004], 108
        mov byte [0xB8006], 116
```

```
        mov byte [0xB8008], 105
        mov byte [0xB800A], 98
        mov byte [0xB800C], 111
        mov byte [0xB800E], 111
        mov byte [0xB8010], 116
; Set the colour of the outputted text to:
        ; Green background (0x2-),
        ; White foreground (0x-F)
        mov dword eax, 0x2F
        mov byte [0xB8001], al
        mov byte [0xB8003], al
        mov byte [0xB8005], al
        mov byte [0xB8007], al
        mov byte [0xB8009], al
        mov byte [0xB800B], al
        mov byte [0xB800D], al
        mov byte [0xB800F], al
        mov byte [0xB8011], al

        ; Call your main method here.

        ; In a proper OS, you shouldn't ever get to this point. But just in
case you do...
        jmp Reset ; Stop / reset the CPU forever
; END - Main Entrypoint
; BEGIN - Reset
Reset:
        cli ; Clear all interrupts so we aren't re-awoken
        hlt     ; Halt the OS / CPU / etc.
        jmp Reset ; Just in case...
; END - Reset
```