```nasm
; CpS 230 Lab 9: Stephen J. Sidwell (ssidw712)
;---------------------------------------------------
; Bootloader that loads/runs a single-sector payload
; program from the boot disk.
;---------------------------------------------------
bits 16

; Our bootloader is 512 raw bytes: we treat it all as code, although
; it has data mixed into it, too.
section .text

; The BIOS will load us into memory at 0000:7C00h; NASM needs
; to know this so it can generate correct absolute data references.
org 0x7C00

; First instruction: jump over initial data and start executing code
start:  jmp main

; Embedded data
boot_msg    db  "CpS 230 Team Project", 13, 10
        db  "by Nathan Collins and Stephen Sidwell", 13, 10, 0
boot_disk   db  0       ; Variable to store the number of the disk we boot from
retry_msg   db  "Error reading payload from disk; retrying...", 13, 10, 0
key_msg     db  `Press any key to start the kernel!\n`, 0
counter     dw  0

main:
    ;Referenced from http://forum.osdev.org/viewtopic.php?f=1&t=7762
    ;Not sure why this fixed my issue but it seems resetting the disk works best
    ;Call interupt to reset the drive
    ; xor ah, ah
    ; int 0x13
    ; TODO: Set DS == CS (so data addressing is normal/easy)
    mov     ax, cs
    mov     ds, ax
    ;Set up es to be the correct offset
    mov     ax, 0x0800
    mov     es, ax
    mov     ax, 512
    imul    word[counter]
    mov     bx, ax ; Zero out bx for offset purposes
    ; TODO: Save the boot disk number (we get it in register DL
    mov     [boot_disk], dl
    ; TODO: Set SS == 0x0800 (which will be the segment we load everything into later)
    mov     ax, 0x0800
    mov     ss, ax
    ; TODO: Set SP == 0x0000 (stack pointer starts at the TOP of segment; first push decrements by 2, to 0xFFFE)
    mov     ax, 0x0000
    mov     sp, ax
    ; TODO: use BIOS raw disk I/O to load sector 2 from disk number <boot_disk> into memory at 0800:0000h (retry on failure)
    mov     ah, 0x02 ;INT 13 number to read sectors
    mov     al, 1; Read one sector
    mov     ch, 0; Track number is always 0
    mov     cl, 2; Read sector 2
    add     cl, [counter]
    inc     word[counter]
    mov     dh, 0; Head number is always 0
    mov     dl, [boot_disk]
    ;Call BIOS interupt
    int     0x13
    ;Interupt sets the carry flag on failure
    ;So jump if the carry flag is set
    cmp     ah, 0
    jz      .interrupt
    mov     dx, retry_msg
    call    puts
    jc      main
    ; Finally, jump to address 0800h:0000h (sets CS == 0x0800 and IP == 0x0000)
.interrupt:
    cmp     word[counter], 63
    jl      main
    ; TODO: Print the boot message/banner
    mov     dx, boot_msg
    call    puts
    mov     dx, key_msg
    call    puts
    xor     ah, ah
    int     0x16
    jmp     0x0800:0x0000

; print NUL-terminated string from DS:DX to screen using BIOS (INT 10h)
; takes NUL-terminated string pointed to by DS:DX
; clobbers nothing
; returns nothing
puts:
    push    ax
    push    cx
    push    si

    mov     ah, 0x0e
    mov     cx, 1       ; no repetition of chars

    mov     si, dx
.loop:
    mov     al, [si]
    inc     si
    cmp     al, 0
    jz      .end
    int     0x10
    jmp     .loop
.end:
    pop     si
    pop     cx
    pop     ax
    ret

; NASM mumbo-jumbo to make sure the boot sector signature starts 510 bytes from our origin
; (logic: subtract the START_ADDRESS_OF_OUR_SECTION [$$] from the CURRENT_ADDRESS [$],
;   yielding the number of bytes of code/data in the section SO FAR; then subtract
;   this size from 510 to give us BYTES_OF_PADDING_NEEDED; finally, emit
;   BYTES_OF_PADDING_NEEDED zeros to pad out the section to 510 bytes)
    times   510 - ($ - $$)  db  0

; MAGIC BOOT SECTOR SIGNATURE (*must* be the last 2 bytes of the 512 byte boot sector)
    dw  0xaa55
```