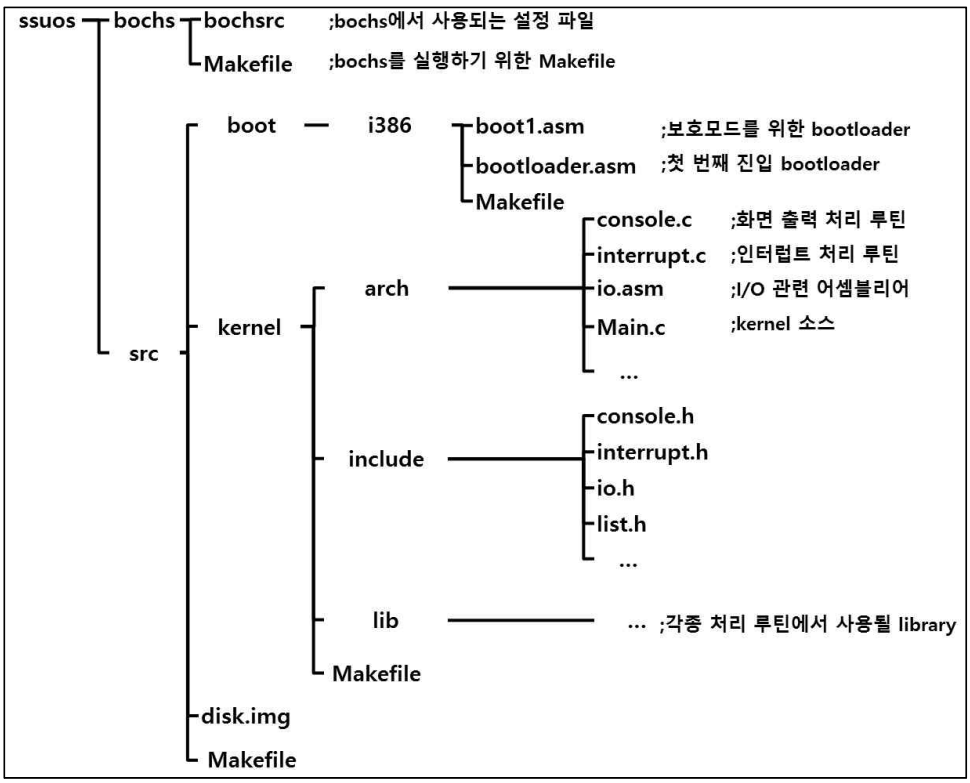
과제 #2 : Bootstrap

○ 과제 목표

- 부트로더 이해 및 분석
- 리얼모드에서 보호모드로 전환 과정 이해
- 부트로더에서 Video, RTC관련 바이오스 인터럽트를 이용한 현재 날짜 출력

○ 기본 배경 지식

- SSU OS 소스 목록



- 리얼모드
  - ✓ 80286 이후의 x86 호환 CPU의 운영 방식
  - ✓ 리얼 주소 모드 혹은 호환 모드라고도 하며, 80186 계열 CPU와 호환을 위해 만듬
  - ✓ 최대 1 MB의 메모리가 번지에 기록될 수 있음
  - ✓ 80286 계열의 CPU는 전원이 켜질 때 리얼모드로 동작함
  - ✓ 80186 계열의 CPU는 하나의 운영방식만 존재했으며, 리얼모드와 동일
- 보호모드
  - ✓ 보호 가상 주소 모드라고도 하며, x86 호환 CPU의 운영 방식
  - ✓ 시스템 소프트웨어가 다중 작업, 가상 메모리, 페이징, 그리고 응용 소프트웨어를 넘는 운영 체제 제어 능력을 높이기 위해 고안된 운영 체제의 다른 기능들을 이용할 수 있게 도와줌
  - ✓ 일반적으로 CPU는 BIOS 이후 리얼모드에서 보호모드로 전환
  - ✓ GDT 등의 부가적 정보가 필요

- GDT (Global Descriptor Table)
  - ✓ 인텔 x86 계열 CPU가 사용하는, 특정 레벨 메모리 영역의 사용 범위와 특성을 정의하기 위한 테이블 자료구조
  - ✓ 보호모드에 들어가기 위해서는 NULL, 코드 세그먼트, 데이터 세그먼트를 서술한 GDT가 필요
- IDT (Interrupt Descriptor Table)
  - ✓ 인텔 x86 계열 CPU가 사용하는, 인터럽트 벡터 테이블을 정의하기 위한 자료구조
  - ✓ CPU가 인터럽트와 예외에 대해 정확한 답변하기 위해 사용
  - ✓ 인터럽트가 발생하였을 때 처리해주는 함수의 루틴을 포함

○ 과제 수행방법

- 리얼모드에서 바이오스 인터럽트를 통한 현재 날짜 출력
  - ✓ boch 에뮬레이터 상에 상단 중앙에 글자색을 바꿔 현재날짜를 출력
  - ✓ 2가지의 바이오스 인터럽트 사용

- (1) boot1.asm 소스 분석
  - ✓ 운영체제 공지사항 게시판에서 소스 파일과 함께 코드분석 한글 파일 다운로드
  - ✓ 소스 코드 줄 단위 빈칸 채우기

- (2) src/boot/i386/boot1.asm 추가, 수정

```
org     0x9000

[BITS 16]

                cli             ; Clear Interrupt Flag

                mov     ax, 0xb800
        mov      es, ax
        mov      ax, 0x00
        mov      bx, 0
        mov      cx, 80*25*2
CLS:
        mov      [es:bx], ax
        add      bx, 1
        loop     CLS

Initialize_PIC:
                ;ICW1 - 두 개의 PIC를 초기화
                mov             al, 0x11
                out             0x20, al
                out             0xa0, al
```

```
                ;ICW2 - 발생된 인터럽트 번호에 얼마를 더할지 결정
                mov             al, 0x20
                out             0x21, al
                mov             al, 0x28
                out             0xa1, al

                ;ICW3 - 마스터/슬레이브 연결 핀 정보 전달
                mov             al, 0x04
                out             0x21, al
                mov             al, 0x02
                out             0xa1, al

                ;ICW4 - 기타 옵션
                mov             al, 0x01
                out             0x21, al
                out             0xa1, al

                mov             al, 0xFF
                ;out            0x21, al
                out             0xa1, al

Initialize_Serial_port:
                xor             ax, ax
                xor             dx, dx
                mov             al, 0xe3
                int             0x14

READY_TO_PRINT:
                xor             si, si
                xor             bh, bh
PRINT_TO_SERIAL:
                mov             al, [msgRMode+si]
                mov             ah, 0x01
                int             0x14
                add             si, 1
                cmp             al, 0
                jne             PRINT_TO_SERIAL
PRINT_NEW_LINE:
                mov             al, 0x0a
                mov             ah, 0x01
                int             0x14
                mov             al, 0x0d
                mov             ah, 0x01
```

```
                int             0x14
```

✓ 현재 날짜 출력 코드가 들어갈 위치
✓ 과제 수행 조건
(1) 모든 기능은 반드시 바이오스인터럽트를 사용해서 구현
(2) 글자색을 변경할 것(명세에서 사용한 값 : 0x0e)
(3) 글자가 출력되는 위치를 변경할 것(명세와 같이 화면의 상단 중앙으로)

```
Activate_A20Gate:
                mov             ax,         0x2401
                int             0x15

;Detecting_Memory:
;               mov             ax, 0xe801
;               int             0x15

PROTECTED:
        xor             ax, ax
        mov             ds, ax

                call    SETUP_GDT

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

                jmp             $+2
                nop
                nop
                jmp             CODEDESCRIPTOR:ENTRY32

SETUP_GDT:
                lgdt    [GDT_DESC]
                ret

[BITS 32]

ENTRY32:
                mov             ax, 0x10
                mov             ds, ax
                mov             es, ax
                mov             fs, ax
```

```
                mov             gs, ax

                mov             ss, ax
                mov             esp, 0xFFFE
                mov             ebp, 0xFFFE

                mov             edi, 80*2
                lea             esi, [msgPMode]
                call    PRINT

                ;IDT TABLE
            cld
                mov             ax,     IDTDESCRIPTOR
                mov             es, ax
                xor             eax, eax
                xor             ecx, ecx
                mov             ax, 256
                mov             edi, 0

IDT_LOOP:
                lea             esi, [IDT_IGNORE]
                mov             cx, 8
                rep             movsb
                dec             ax
                jnz             IDT_LOOP

                lidt    [IDTR]

                sti
                jmp     CODEDESCRIPTOR:0x10000

PRINT:
                push    eax
                push    ebx
                push    edx
                push    es
                mov             ax, VIDEODESCRIPTOR
                mov             es, ax
PRINT_LOOP:
                or              al, al
                jz              PRINT_END
                mov             al, byte[esi]
                mov             byte [es:edi], al
```

```
                inc             edi
                mov             byte [es:edi], 0x07

OUT_TO_SERIAL:
                mov             bl, al
                mov             dx, 0x3fd
CHECK_LINE_STATUS:
                in              al, dx
                and             al, 0x20
                cmp             al, 0
                jz              CHECK_LINE_STATUS
                mov             dx, 0x3f8
                mov             al, bl
                out             dx, al

                inc             esi
                inc             edi
                jmp             PRINT_LOOP
PRINT_END:
LINE_FEED:
                mov             dx, 0x3fd
                in              al, dx
                and             al, 0x20
                cmp             al, 0
                jz              LINE_FEED
                mov             dx, 0x3f8
                mov             al, 0x0a
                out             dx, al
CARRIAGE_RETURN:
                mov             dx, 0x3fd
                in              al, dx
                and             al, 0x20
                cmp             al, 0
                jz              CARRIAGE_RETURN
                mov             dx, 0x3f8
                mov             al, 0x0d
                out             dx, al

                pop             es
                pop             edx
                pop             ebx
                pop             eax
                ret
```

```
GDT_DESC:
        dw GDT_END - GDT - 1
        dd GDT
GDT:
                ...
```

○ 과제 수행 전 실행 결과

<bochs GUI>

○ 과제 수행 후 실행 결과

<bochs GUI>

- 과제제출
  ✓ 2017년 9월 12일 (화) 23시 59분까지 제출

- 배점 기준
  ✓ 보고서 20%
    - 개요 2%
    - 상세 설계 명세(기능 명세 포함) 8%
    - 구현 방법 설명(코드 주석 포함) 8%
    - 실행 결과 2%

  ✓ 소스분석 20%
    - 제공된 소스 코드 분석 25%

  ✓ 소스코드 60%
    - 컴파일 여부 10%(설계 요구에 따르지 않고 설계된 경우 0점 부여)
    - 실행 여부 50%(RTC 사용 날짜 출력 20% + 글자 색 변경 15% + 출력위치 변경 15%)

- 최소 구현사항
  ✓ 소스코드 분석 + 현재날짜 출력 구현