OS_Project 1

2014-19975 송원재

1. 초기화 및 인터럽트 처리 루틴

► 시스템 초기화

- main 함수 : elfags = 0, esp = 0 이고 vector[0] = os_reset_handler 로 jump 한다.
- os_reset_handler : 처음 4096 만큼 stack area 를 확보하고 os_initialization 수행한다.
- os_initialization

os_init_hal() : init_timer_interrupt() - 1 초 간격으로 interrupt 가 발생(index =0)하는 타이머를 생성 / eos_enable_irq_line : 0 번 interrupt unmask

os_init_icb_table() : Interrupt 의 number, handler 정보가 들어있는 icb_table 을 초기화 한다.

os_init_scheduler() : ready_group 과 ready_table 을 초기화, 즉 스케쥴러 초기화

os_init_task() : multi-level ready queue 초기화

os_init_timer : eos_init_counter 로 counter 를 생성하고 timer_interrupt 의 handler 를 설정해 준다.

idle 한 상태에서 수행할 idle_task 생성

user level 의 초기화 작업 수행

multitasking = 1, interrupt enable, schedule 추가한 후에 무한루프 수행

► 인터럽트 요청

- void _gen_irq(int8u_t irq) : irq_pending 에서 해당 irq 비트를 1 로 바꾸고 call_ _____deliver_irq()
- void deliver_irq() : unmask 된 irq 중 pending 된 것이 있을 때, 그리고 eflags = 1, 즉 interrupt 가 활성화 되어 있을 때 eflags = 0 으로 바꾸고 eflags_saved = 1 로 바꾼 다음 vector[3]으로 점프한다. 이후에 다시 돌아오게 된다!!
- vector[3] = os_irq_handler : 현재 레지스터값, eflags_saved 를 push 한 후에 os_common_interrupt_handler call 한다.
- void os_common_interrupt_handler(int32u_t flag) : 마지막에 push 된 변수가 eflags_saved 이므로 flag 에는 eflags_saved 의 값이 들어가게 된다. 처리해야 할 irq 가 있는지 확인하고 (eos_get_irq) 해당 irq 접수(eos_ack_irq)한 다음 eflags 원상복귀 (eos_restore_interrupt) 후 handler 를 실행한다.

- 다시 os_irq_handler 로 돌아와 _IRET 를 수행하는데 _IRET 에서는 또 다시 deliver_ irq()를 재귀적으로 수행하게 된다. 따라서 처리해야 할 모든 irq 를 처리할 때까지 이 작업을 반복하게 된다.

2. 인터럽트 관리 모듈 분석

► interrupt.c
- void eos_ack_irq(int32u irq)
/* clear the corresponding bit in _irq_pending register */
irq = interrupt 정보(index)
_irq_pending 의 irq 비트 0 으로 바꿈 (interrupt 가 접수되었음을 의미)

- int32s_t eos_get_irq
/* get the highest bit position in the _irq_pending register */
pending = 1 and mask = 0 인 가장 높은 비트의 irq_index return

- void eos_disable_irq_line(int32u_t irq)
/* turn on the corresponding bit */
masking – _irq_mask 의 해당 비트를 1 로 바꿈

- void eos_enable_irq_line(int32u_t irq)
/* turn off the corresponding bit */
unmasking - _irq_mask 의 해당 비트(irq)를 0 으로 바꿈

»_irq_pending : 현재 들어온 interrupt 정보
»_irq_mask : 현재 활성화된 interrupt = 0, 비활성화된 interrupt = 1

►interrupt_asm.S

eos_diable_interrupt
/* disable irq and return previous status */
eax register 에 eflags 값 저장 후 _CLI(eflags = 0)

eos_enable_interrupt
/* enable irq by force */
_STI : eflags =1 후 call _deliver_irq()

eos_restore_interrupt
/* restore irq status */
esp 전 값 eflag 저장 (flag 를 parameter 로 받기 때문에 stack 가장 최근값(=0x4(%esp))이 flag)

3. C-서브루틴 코드 분석

주어진 예제 코드를 어셈블리어 파일로 받아 필요한 부분만 추출한 것은 아래와 같다.

```
add:
1       pushl %ebp              ebp push
2       movl  %esp, %ebp        ebp = esp
3       subl  $16, %esp         esp -= 16 (int x 할당)
4       movl  8(%ebp), %edx     edx = (ebp + 8) (변수 b 값)
5       movl  12(%ebp), %eax    eax = (ebp + 12) (변수 a 값)
6       addl  %edx, %eax        eax = (eax) + (edx) (a+b 값 저장)
7       movl  %eax, -4(%ebp)    (ebp-4) = (eax) (x = a+b)
8       movl  -4(%ebp), %eax    eax = (ebp-4) ( eax = x ) - return 값 = (eax)
9       leave                   ebp 까지 모두 pop
10      ret                     eip 저장주소로 return
mul:
11      pushl %ebp              ebp push
12      movl  %esp, %ebp        ebp = esp
13      movl  8(%ebp), %eax     eax = (ebp + 8) - (parameter a)
14      imull 12(%ebp), %eax    eax = eax*(ebp+12) (a*b)
15      popl  %ebp              ebp pop
16      ret                     eip 저장 주소로 return
main:
17      pushl %ebp              ebp push
18      movl  %esp, %ebp        ebp = esp
19      subl  $16, %esp         esp -= 16 (a, b, c, ret 할당)
20      movl  $10, -16(%ebp)    (ebp-16) = 10 ( a = 10 )
21      movl  $10, -12(%ebp)    (ebp-12) = 10 ( b = 10 )
22      movl  $5, -8(%ebp)      (ebp-8) = 5 ( c = 5 )
23      pushl -8(%ebp)          (ebp-8) push - function parameter (c)
24      pushl -12(%ebp)         (ebp-12) push - function parameter (b)
25      call  add               다음 instruction 주소 push 하고 add 로 이동
26      addl  $8, %esp          esp += 8
27      pushl %eax              eax push - return 값
28      pushl -16(%ebp)         (ebp-16) push - 변수 a 값
29      call  mul               다음 instruction 주소 push 하고 mul 로 이동
30      addl  $8, %esp          esp += 8
31      movl  %eax, %edx        edx = eax - return 값
32      movl  -4(%ebp), %eax    eax = (ebp-4) - eax 에 ret(주소값) 저장
33      movl  %edx, (%eax)              (eax) = edx - eax 에 저장된 주소에 edx 값 저장
34      movl  $0, %eax          eax = 0
35      leave                   ebp 까지 모두 pop
36      ret                     eip 저장주소로 return
```

17 pushl		18 movl		19 subl		20 21 22 movl, movl, movl		23 24 pushl, pushl		25 1 2 call add, pushl, movl	
	old eip		old eip		old iep		old iep		old iep		old iep
esp	old ebp	esp, ebp	old ebp	ebp	old ebp	ebp	old ebp	ebp	old ebp		old ebp
					*ret		*ret		*ret		*ret
					c		5		5		5
					b		10		10		10
				esp	a	esp	10		10		10
									5		5
								esp	10		10
											old eip
										esp, ebp	old ebp

3 4 5 subl, movl, movl		6 7 8 addl, movl, movl		9 10 leave, ret		26 27 28 addl, pushl, pushl		29 11 12 call, pushl, movl	
	old iep		old iep		old iep		old iep		old iep
	old ebp		old ebp	ebp	old ebp	ebp	old ebp		old ebp
	*ret		*ret		*ret		*ret		*ret
	5		5		5		5		5
	10		10		10		10		10
	10		10		10		10		10
	5		5		5		5		5
	10		10		10		10		10
	old eip		old eip				15		15
ebp	old ebp	ebp	old ebp	esp		esp	10		10
			15						old eip
								esp, ebp	old ebp
esp		esp							

eax	5	eax	15	eax	15	eax	15	eax	15
edx	10	edx	10	edx	10	edx	10	edx	10

13 14 movl, imull		15 16 popl, ret		30 31 32 addl, movl, movl		33 34 movl, movl		35 36 leave, ret	
	old iep		old iep		old iep		old iep		
	old ebp	ebp	old ebp	ebp	old ebp	ebp	old ebp	esp	
	*ret		*ret		*ret		*ret		
	5		5		5		5		
	10		10		10		10		
	10		10		10		10		
	5		5		5		5		
	10		10		10		10		
	15		15		15		15		
	10		10	esp	10	esp	10		
	old eip								
esp, ebp	old ebp	esp							

eax	15→10→10*15 = 150	eax	150	eax	ret 주소값	eax	0
edx	10	edx	10	edx	150	edx	150
						ret 주소	150