

이름: 양해찬 (2016124145)

동작 원리



```
//Lab5_Part1_main.s
#include "./address_map_nios2.s"

.text

.global _start

_start:

    movia sp, SDRAM_END           # SP에 SDRAM_END 주소 복사

    movia r22, HEX3_HEX0_BASE

    stwio r0, 0(r22)

    movia r22, KEY_BASE           # r22에 KEY_BASE 주소 복사

    movi r23, 0b01111            # mask 0b01111(key0~3까지 사용)
```

전자 HW 설계 – 실습 보고서

```
stwio r23, 8(r22)      # interrupt mask 레지스터를 0b1111 설정

movi r23, 2            # ienable을 0b10 설정
wrctl ienable,r23      # ienable을 0b10 설정
movi r23, 1
wrctl status, r23      # status를 0b1 설정

IDLE:
    br IDLE
    .end

//Lab5_Part1_service.s
.section .reset, "ax"
    movia r2, _start
    jmp r2

.section .exceptions, "ax" /*Allocatable and eExecutable*/
.global ISR

ISR:  subi sp, sp, 16    # 스택에 et를 저장
      stw et, 0(sp)
      rdctl et, ctl4     # HW interrupt인지 확인
      beq et, r0, SKIP_EA_DEC # SW interrupt
      subi ea, ea, 4      # HW interrupt의 경우 실행하지 못했던 interrupt 이전 명령어를 실행

SKIP_EA_DEC: stw ea, 4(sp) # exception handling에 사용되는 레지스터 백업
             stw ra, 8(sp)
             stw r22, 12(sp)
             rdctl et, ctl4 #ipending read
             bne et, r0, ON_EXT_INT
             #SW interrup->on_trap HW->on_ext_int

ON_TRAP:    br END_ISR

ON_EXT_INT: andi r22, et, 0b10 #key에 의한것인지?
            beq r22, r0, END_ISR #아니라면 end
            call ON_KEY_PRESS    #맞다면 call ON_KEY_PRESS

END_ISR:    ldw et, 0(sp)        # 데이터 복구
            ldw ea, 4(sp)
            ldw ra, 8(sp)
            ldw r22, 12(sp)
            addi sp, sp, 16
            eret
            .end

//Lab5_Part1_key_isr.s
.include "./address_map_nios2.s"
.global ON_KEY_PRESS
```

전자 HW 설계 – 실습 보고서

```
ON_KEY_PRESS:  subi    sp,    sp, 8          # 데이터 백업
                stw     r22, 0(sp)
                stw     r23, 4(sp)
                movia   r22, KEY_BASE
                ldwio   r23, 0xC(r22)
                stwio   r23, 0xC(r22) /*Turn off IRQ by writing a value to the edgecapture register*/
                movia   r22, HEX3_HEX0_BASE
                movi    r15, 0b00111111      #0 SEGCODE
                movi    r16, 0b00000110      #1 SEGCODE
                movi    r17, 0b01011011      #2 SEGCODE
                movi    r18, 0b01001111      #3 SEGCODE
                slli    r16, r16, 8          #HEX1로 출력하기위해 8BIT SHIFT
                slli    r17, r17, 16         #HEX2로 출력하기위해16BIT SHIFT
                slli    r18, r18, 24         #HEX3로 출력하기위해24BIT SHIFT
                #Edge capture와 비교해 몇번째 key인지 확인
                andi    r19, r23, 0b1000
                bne     r19, r0, KEY3
                andi    r19, r23, 0b0001
                bne     r19, r0, KEY0
                andi    r19, r23, 0b0010
                bne     r19, r0, KEY1
                andi    r19, r23, 0b0100
                bne     r19, r0, KEY2


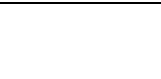
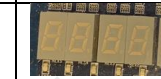
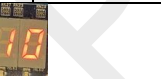
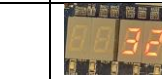

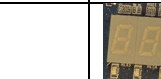

                # 데이터 복구
ON_KEY_PRESS1: ldw     r22, 0(sp)
                ldw     r23, 4(sp)
                addi    sp, sp, 8
                ret
                # KEY0일때
KEY0:          ldwio   r23, 0(r22) /*Read the previous value*/
                and     r23, r23, r15  #이전에 0이 켜진상태이면
                beq     r23, r15, OFF0  #켜져있었다면 0을 끈다
                ldwio   r23, 0(r22)    #꺼져있었다면 다시 load
                add     r23, r23, r15  #0 ON
                stwio   r23, 0(r22) /*Write the new value*/
                br      ON_KEY_PRESS1
                # KEY1일때
KEY1:          ldwio   r23, 0(r22) /*Read the previous value*/
                and     r23, r23, r16  #이전에 1이 켜진상태이면
                beq     r23, r16, OFF1  #켜져있었다면 1을 끈다
                ldwio   r23, 0(r22)    #꺼져있었다면 다시 load
                add     r23, r23, r16  #1 ON
                stwio   r23, 0(r22) /*Write the new value*/
```

전자 HW 설계 – 실습 보고서

```
br ON_KEY_PRESS1
# KEY2일때
KEY2:    ldwio r23, 0(r22) /*Read the previous value*/
and      r23, r23, r17    #이전에 2이 켜진상태이면
beq      r23, r17, OFF2   #켜져있었다면 2을 끝다
ldwio    r23, 0(r22)      #꺼져있었다면 다시 load
add      r23, r23, r17    #2 ON
stwio    r23, 0(r22) /*Write the new value*/
br ON_KEY_PRESS1
# KEY3일때
KEY3:    ldwio r23, 0(r22) /*Read the previous value*/
and      r23, r23, r18    #이전에 3이 켜진상태이면
beq      r23, r18, OFF3   #켜져있었다면 3을 끝다
ldwio    r23, 0(r22)      #꺼져있었다면 다시 load
add      r23, r23, r18    #3 ON
stwio    r23, 0(r22) /*Write the new value*/
br ON_KEY_PRESS1
OFF0:    ldwio r23, 0(r22)    #0 OFF
sub      r23, r23, r15
stwio    r23, 0(r22)
br ON_KEY_PRESS1
OFF1:    ldwio r23, 0(r22)    #1 OFF
sub      r23, r23, r16
stwio    r23, 0(r22)
br ON_KEY_PRESS1
OFF2:    ldwio r23, 0(r22)    #2 OFF
sub      r23, r23, r17
stwio    r23, 0(r22)
br ON_KEY_PRESS1
OFF3:    ldwio r23, 0(r22)    #3 OFF
sub      r23, r23, r18
stwio    r23, 0(r22)
br ON_KEY_PRESS1
.end
```

전자 HW 설계 - 실습 보고서

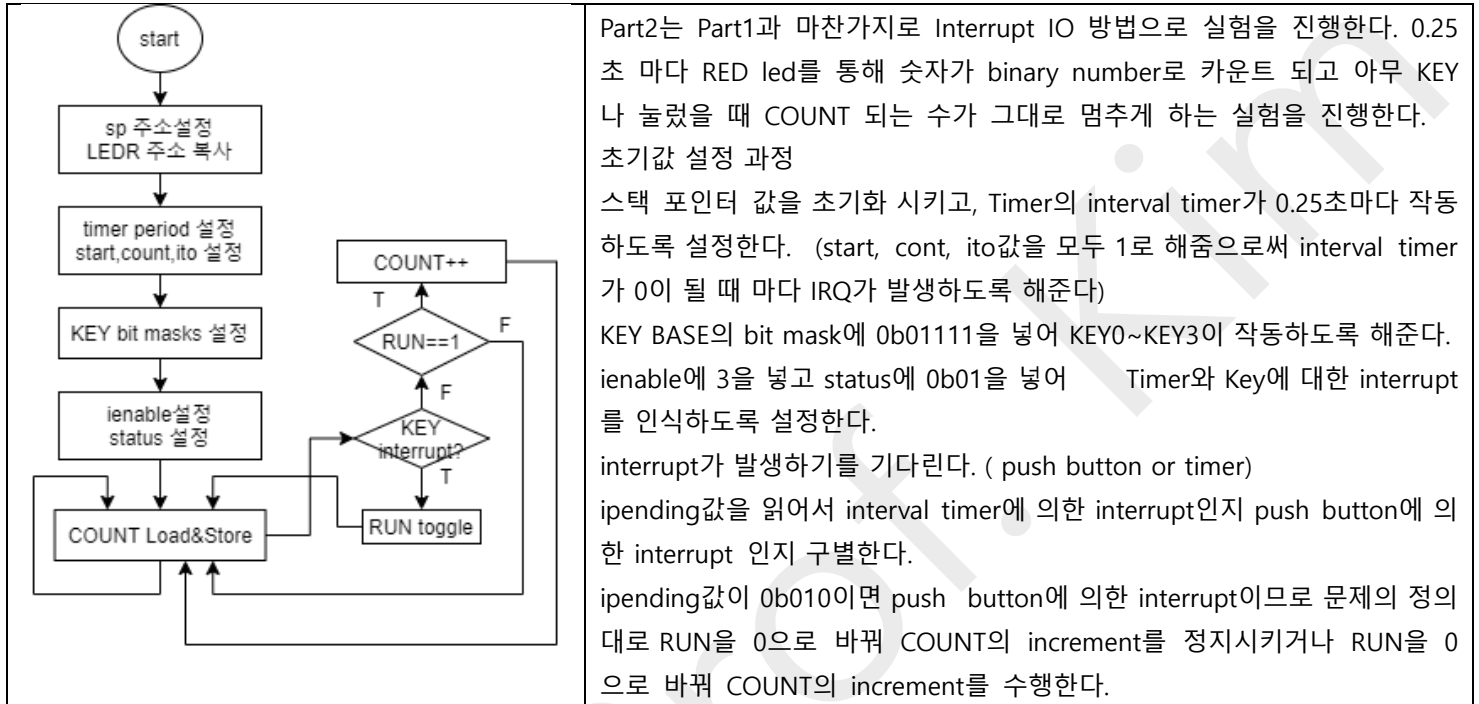
결과 및 토의

Registers	Registers	Registers	Registers	Registers	Registers	Registers	Registers
Reg	Value	Reg	Value	Reg	Value	Reg	Value
pc	0x0000354	pc	0x0000354	pc	0x0000354	pc	0x0000354
zero	0x00000000	zero	0x00000000	zero	0x00000000	zero	0x00000000
r1	0x00000000	r1	0x00000000	r1	0x00000000	r1	0x00000000
r2	0x00000000	r2	0x00000000	r2	0x00000000	r2	0x00000000
r3	0x00000000	r3	0x00000000	r3	0x00000000	r3	0x00000000
r4	0x00000000	r4	0x00000000	r4	0x00000000	r4	0x00000000
r5	0x00000000	r5	0x00000000	r5	0x00000000	r5	0x00000000
r6	0x00000000	r6	0x00000000	r6	0x00000000	r6	0x00000000
r7	0x00000000	r7	0x00000000	r7	0x00000000	r7	0x00000000
r8	0x00000000	r8	0x00000000	r8	0x00000000	r8	0x00000000
r9	0x00000000	r9	0x00000000	r9	0x00000000	r9	0x00000000
r10	0x00000000	r10	0x00000000	r10	0x00000000	r10	0x00000000
r11	0x00000000	r11	0x00000000	r11	0x00000000	r11	0x00000000
r12	0x00000000	r12	0x00000000	r12	0x00000000	r12	0x00000000
r13	0x00000000	r13	0x00000000	r13	0x00000000	r13	0x00000000
r14	0x00000000	r14	0x00000000	r14	0x00000000	r14	0x00000000
r15	0x0000003F	r15	0x0000003F	r15	0x0000003F	r15	0x0000003F
r16	0x00000600	r16	0x00000600	r16	0x00000600	r16	0x00000600
r17	0x00580000	r17	0x00580000	r17	0x00580000	r17	0x00580000
r18	0x4F000000	r18	0x4F000000	r18	0x4F000000	r18	0x4F000000
r19	0x00000001	r19	0x00000002	r19	0x00000004	r19	0x00000008
r20	0x00000000	r20	0x00000000	r20	0x00000000	r20	0x00000000
r21	0x00000000	r21	0x00000000	r21	0x00000000	r21	0x00000000
r22	0xFF200050	r22	0xFF200050	r22	0xFF200050	r22	0xFF200050
r23	0x00000001	r23	0x00000001	r23	0x00000001	r23	0x00000001
et	0x00000000	et	0x00000000	et	0x00000000	et	0x00000000
bt	0xFFFFFFFF	bt	0xFFFFFFFF	bt	0xFFFFFFFF	bt	0xFFFFFFFF
gp	0x00000000	gp	0x00000000	gp	0x00000000	gp	0x00000000
sp	0x03FFFFFF	sp	0x03FFFFFF	sp	0x03FFFFFF	sp	0x03FFFFFF
fp	0x00000000	fp	0x00000000	fp	0x00000000	fp	0x00000000
ea	0x00000354	ea	0x00000354	ea	0x00000354	ea	0x00000354
ba	0xFFFFFFFF	ba	0xFFFFFFFF	ba	0xFFFFFFFF	ba	0xFFFFFFFF
ra	0x00000000	ra	0x00000000	ra	0x00000000	ra	0x00000000
status	0x00000001	status	0x00000001	status	0x00000001	status	0x00000001
estatus	0x00000001	estatus	0x00000001	estatus	0x00000001	estatus	0x00000001
bstatus	0xFFFFFFFF	bstatus	0xFFFFFFFF	bstatus	0xFFFFFFFF	bstatus	0xFFFFFFFF
ienable	0x00000002	ienable	0x00000002	ienable	0x00000002	ienable	0x00000002
ipending	0x00000000	ipending	0x00000000	ipending	0x00000000	ipending	0x00000000
cpuid	0x00000000	cpuid	0x00000000	cpuid	0x00000000	cpuid	0x00000000
KEY0 0 ON	KEY0 0 OFF	KEY1 1 ON	KEY1 1 OFF	KEY2 2 ON	KEY2 2 OFF	KEY3 3 ON	KEY3 3 OFF
							
Key0을 한번 눌렀을 때		그 상태에서 key1을 눌렀을 때		그 상태에서 key 2와 3을 눌렀을 때		그 상태에서 key3과 key1을 눌렀을 때	
R19는 몇 번째 key를 눌렀을 때 인지 알 수 있다. KEY0은 1 KEY1은 2 KEY2는 4 KEY3은 8이다. PC값을 통해 IDLE을 LOOP하고 있는 것을 알 수 있다. STATUS는 1이 저장되어 있는데 이는 PIE가 1임을 나타내고 interrupt에 대한 허용을 의미한다. ienagle은 2가 저장되어 있는데 PUSHBUTTON에 의한 interrupt를 나타낸다.							
https://www.youtube.com/watch?v=29oh_PzaDuw							

전자 HW 설계 – 실습 보고서

✓ Part II

동작 원리.



구현 코드 설명

```
//Lab5_Part2_main.s
#include "./address_map_nios2.s"
.text
.global _start

_start: movia sp, SDRAM_END # stack pointer 주소를 설정
        movia r8, LEDR_BASE # r10에 LEDR_BASE 주소를 저장
        call CONFIG_TIMER
        call CONFIG_KEYS
        /* enable Nios II processor interrupts */
        movi    r23, 3 # ienable contrl register에 0b11 store
        wrctl   ienable, r23
        movi    r23, 1
        wrctl   status, r23
        /* initialize LEDR value */

LOOP:   ldw r9, COUNT(r0) /* global variable */
        stwio r9, (r8) # r8에 COUNT store
```

전자 HW 설계 – 실습 보고서

```
br LOOP

/* Configure the interval timer to create interrupts at 0.25 second intervals */
CONFIG_TIMER:
    movia r22, TIMER_BASE
    movui r23, 0xE100
    stwio r23, 8(r22) # 앞 16bit를 Timer의 Periodl에 store
    movui r23, 0x05F5
    stwio r23, 12(r22) # 뒤 16bit를 Timer의 Periodh에 store
    movi r23, 0b0111 # start=1 cont=1 ito=1
    stwio r23, 4(r22)
    ret

/* Configure the pushbutton KEYS to generate interrupts */
CONFIG_KEYS:
    movia r22, KEY_BASE #KEY_BASE 주소를 저장
    movi r23, 0b01111
    stwio r23, 8(r22)
    ret

/* Global variables */
.global COUNT
COUNT: .word 0x0 # used by timer
.global RUN # used by pushbutton KEYS
RUN: .word 0x1 # initial value to increment COUNT
.end

//Lab5_Part2_service
.section .reset, "ax"
    movia r2, _start
    jmp r2

.section .exceptions, "ax" /*Allocatable and eXecutable.*/
.global ISR
    # sp 주소에 16을 감소시킴
ISR:
    subi sp, sp, 16 /*et, ea, ra, r22 will be stored into memory*/
    stw et, 0(sp) # 0(sp)에 exception temporary 저장
    rdctl et, ctl4 /*et is 0 only if no external (hardware) interrupts occurred; i.e. trap*/
    beq et, r0, SKIP_EA_DEC # ipending이 0이라면 ea-4 과정 skip
    subi ea, ea, 4 /*ea is decreased only for trap*/

SKIP_EA_DEC:
    stw ea, 4(sp) # ipending ctl register 데이터를 et에 저장
    stw ra, 8(sp) # 8(sp)에 return address 저장
    stw r22, 12(sp) # 12(sp)에 r22 저장
    rdctl et, ctl4 # ipending ctl register 데이터를 et에 저장
    bne et, r0, ON_EXT_INT # ipending 데이터가 0이면 END

ON_TRAP:
    br END_ISR

ON_EXT_INT:
    andi r22, et, 0b10 # et에 저장된 ipending data와 10을 and 시킴
    bne r22, r0, CALL_KEY # and 값이 0이아니면 CALL_KEY로 br
```

전자 HW 설계 – 실습 보고서

```
andi r22, et, 0b01 # et에 저장된 ipending data와 01을 and 시킴
bne r22, r0, CALL_TIMER # and 값이 0이아니면 CALL_TIMER로 br
br END_ISR

CALL_TIMER:    call  ON_TIMER
               br   END_ISR

CALL_KEY:      call ON_KEY_PRESS
               br   END_ISR

END_ISR:       # 순서에 맞게 stack에 있는 데이터를 load 시킴
               ldw   et, 0(sp)/ *et, ea, ra, r22 are restored from memory*/
               ldw   ea, 4(sp)
               ldw   ra, 8(sp)
               ldw   r22, 12(sp)
               addi   sp, sp, 16
               eret
               .end

//Lab5_Part2_key_isr
#include "../address_map_nios2.s"
.global ON_KEY_PRESS

ON_KEY_PRESS:  subi sp, sp, 8 # sp 주소를 감소시킨후 사용할 register를 stack에 저장
               stw   r22, 0(sp)
               stw   r23, 4(sp)
               movia r22, KEY_BASE
               ldwio r23, 0xC(r22) # KEY_BASE의 edgecapture load
               stwio r23, 0xC(r22) /*Turn off IRQ by writing a value to the edgecapture register*/
               movia r22, RUN # global. RUN에 저장되어 있는 수 확인
               ldw   r23, (r22) # global. RUN에 저장되어 있는 수 load
               xori   r23, r23, 0b01 # 0b01과 r22를 xor 연산하여 데이터를 toggle
               stw   r23, (r22) #RUN 업데이트
               ldw   r22, 0(sp) #백업데이터 복구
               ldw   r23, 4(sp)
               addi   sp, sp, 8
               ret
               .end

//Lab5_Part2_timer_isr
#include "../address_map_nios2.s"
.global ON_TIMER

ON_TIMER:      subi sp, sp, 8
               stw   r22, 0(sp)
               stw   r23, 4(sp)
               movia r22, TIMER_BASE # r22에 TIMER_BASE 주소 할당
               stwio r0, (r22) #TO를 다시 0으로
               movia r22, RUN # global. RUN에 저장되어 있는 수 확인
               ldw   r23, (r22) #global. RUN에 저장되어 있는 수 load
```


전자 HW 설계 – 실습 보고서

```

beq    r23, r0,    END_ON_TIMER    #RUN이 0이면 end
movia  r22, COUNT    # RUN이 1이면 COUNT
ldw    r23, (r22)    # RUN이 1이면 COUNT
addi   r23, r23, 1 /*Increment*/
stw    r23, (r22) /*Turn off IRQ by writing a value to the edgecapture register*/

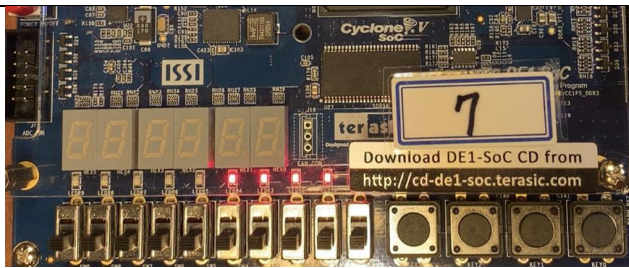
```

```

END_ON_TIMER: ldw r22, 0(sp)
              ldw r23, 4(sp)
              addi sp, sp, 8
              ret
              .end

```

결과 및 토의



COUNT 중

Registers			
Reg	Value		
pc	0x000026C	r16	0x00000000
zero	0x00000000	r17	0x00000000
r1	0x00000000	r18	0x00000000
r2	0x00000000	r19	0x00000000
r3	0x00000000	r20	0x00000000
r4	0x00000000	r21	0x00000000
r5	0x00000000	r22	0xFF200050
r6	0x00000000	r23	0x00000001
r7	0x00000000	et	0x00000000
r8	0xFF200000	bt	0xFFFFFFFF
r9	0x00000029	gp	0x00000000
r10	0x00000000	sp	0x03FFFFFF
r11	0x00000000	fp	0x00000000
r12	0x00000000	ea	0x00000268
r13	0x00000000	ba	0xFFFFFFFF
r14	0x00000000	ra	0x00000258
r15	0x00000000	status	0x00000001
r16	0x00000000	estatus	0x00000001
r17	0x00000000	bstatus	0xFFFFFFFF
		ienable	0x00000003
		ipending	0x00000000
		cpuid	0x00000000

Count 도중(RUN=1)

R9에 현재 COUNT값이 저장되어 있다.
R8은 LED_BASE 주소 값이다.
pc값을 통해 IDLE을 무한 LOOP중 인 것을 알 수 있다.
R23의 1은 status.pie=1을 의미한다.
R20을 통해 stop, start cont, ito가 각각 0 1 1 1인 것을 알 수 있다.
ienable 은 timer와 pushbutton을 모두 사용할 수 있도록 3이 저장되어 있다.

<https://www.youtube.com/watch?v=s6CPVju6Yvc>



KEY를 눌러 멈춤

Registers			
Reg	Value		
pc	0x000026C	r16	0x00000000
zero	0x00000000	r17	0x00000000
r1	0x00000000	r18	0x00000000
r2	0x00000000	r19	0x00000000
r3	0x00000000	r20	0x00000000
r4	0x00000000	r21	0x00000000
r5	0x00000000	r22	0xFF200050
r6	0x00000000	r23	0x00000001
r7	0x00000000	et	0x00000000
r8	0xFF200000	bt	0xFFFFFFFF
r9	0x0000002C	gp	0x00000000
r10	0x00000000	sp	0x03FFFFFF
r11	0x00000000	fp	0x00000000
r12	0x00000000	ea	0x0000026C
r13	0x00000000	ba	0xFFFFFFFF
r14	0x00000000	ra	0x00000258
r15	0x00000000	status	0x00000001
r16	0x00000000	estatus	0x00000001
r17	0x00000000	bstatus	0xFFFFFFFF
		ienable	0x00000003
		ipending	0x00000000
		cpuid	0x00000000

Count STOP(RUN=0)