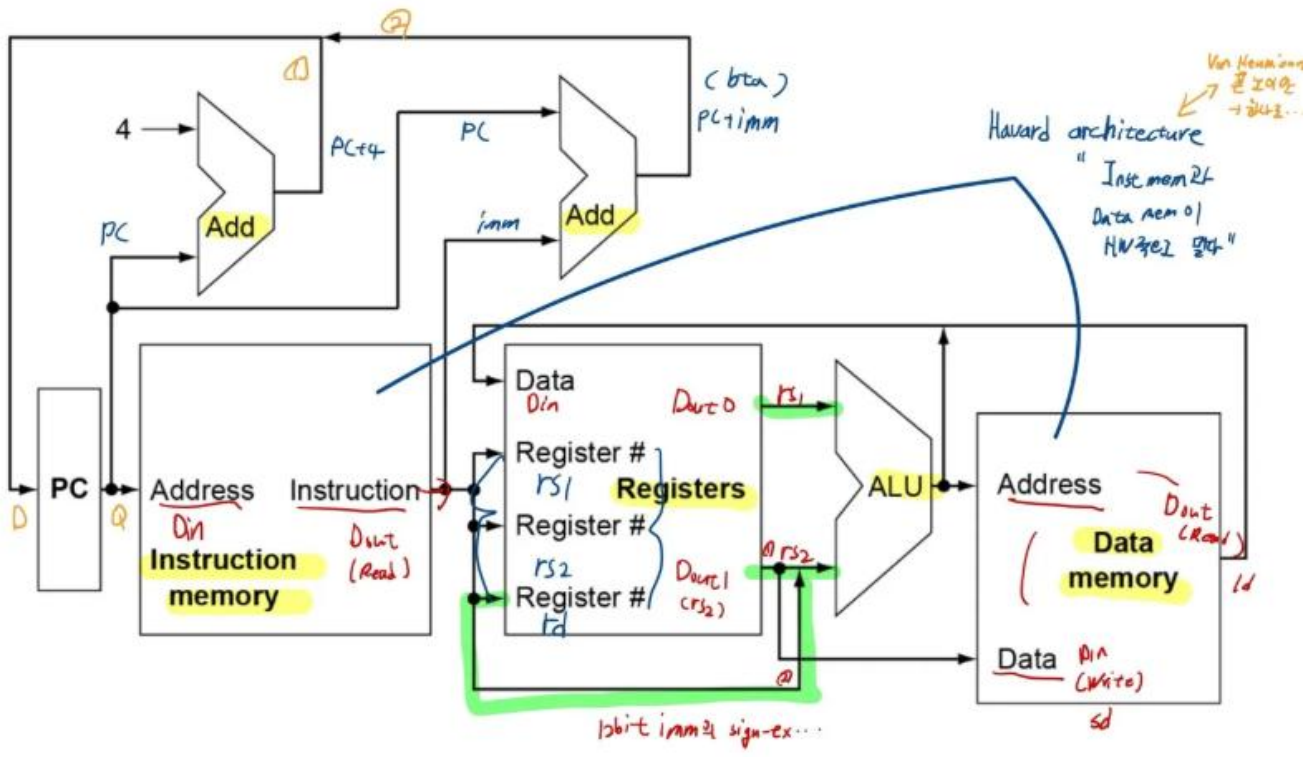
RISC-V CPU 설계

2026.1.

조민우

RISC-V CPU - Program Counter



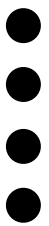

```
// Program Counter with branch/jump support
    always @ (posedge clk, posedge rst)
    begin
        if (rst)
            pc <= 0;
        else
            pc <= next_pc;// Default: pc <= pc + 4;
    end

    // pc 계산
    always @(*) begin
        if (is_jal) begin
            // JAL
            // RISC-V는 특이하게 현재 PC 기준으로 연산한다(not 절대주소)
            next_pc = pc + imm_j;
        end
        else if (is_jalr) begin
            // JALR: PC = (rs1 + imm_i) & ~1
            // rv32i는 LSB가 0
            next_pc = (rs1_data + imm_i) & 32'hFFFFFFFE;
        end
        else if (branch_taken) begin
            // Branch: PC = PC + imm_b
            next_pc = pc + imm_b;
        end
        else begin
            // Default: PC + 4
            next_pc = pc + 4;
        end
    end
```

RISC-V CPU – imm/Branch condition

```
// immediate 값, 32bit sign-extended
always @(*) begin
    // I-type immediate (ADDI, JALR, LOAD)
    imm_i = {{20{inst[31]}}, inst[31:20]};

    // S-type immediate (STORE)
    imm_s = {{20{inst[31]}}, inst[31:25], inst[11:7]};

    // B-type immediate (BRANCH)
    imm_b = {{19{inst[31]}}, inst[31], inst[7], inst[30:25], inst[11:8], 1'b0};

    // U-type immediate (LUI, AUIPC)
    imm_u = {inst[31:12], 12'b0};

    // J-type immediate (JAL)
    imm_j = {{11{inst[31]}}, inst[31], inst[19:12], inst[20], inst[30:21], 1'b0};
end

// Branch condition
always @(*) begin
    branch_taken = 0;
    if (opcode == `OP_B) begin
        case(funct3)
            3'b000: branch_taken = Zflag;          // BEQ (rs1 == rs2)
            3'b001: branch_taken = ~Zflag;         // BNE (rs1 != rs2)
            3'b100: branch_taken = Nflag ^ Vflag;  // BLT (rs1 < rs2, signed) 오버플로우(V) 고려
            3'b101: branch_taken = ~(Nflag ^ Vflag); // BGE (rs1 >= rs2, signed)
            3'b110: branch_taken = ~Cflag;         // BLTU (rs1 < rs2, unsigned)
            3'b111: branch_taken = Cflag;          // BGEU (rs1 >= rs2, unsigned)
            default: branch_taken = 0;
        endcase
    end
end
```

RISC-V CPU – ctrl signals/ALU mux

```
always @(*) begin
    // Default values
    alusrc = 0;
    regwrite = 0;
    lui = 0;
    memwrite = 0;
    alucontrol = 5'b00000;  // ADD
    is_jal = 0;
    is_jalr = 0;

    case(opcode)
        `OP_I_ARITH: begin  // ADDI, SLTI, XORI, ORI, ANDI, SLLI, SRLI, SRAI
            alusrc = 1;      // Use immediate
            regwrite = 1;
            case(funct3)
                3'b000: alucontrol = 5'b00000;  // ADDI
                3'b010: alucontrol = 5'b00011;  // SLTI
                3'b011: alucontrol = 5'b00100;  // SLTIU
                3'b100: alucontrol = 5'b00101;  // XORI
                3'b110: alucontrol = 5'b01000;  // ORI
                3'b111: alucontrol = 5'b01001;  // ANDI
```



```
    // ALU source selection
    always @(*) begin
        // First operand
        if (opcode == `OP_AUIPC)
            alusrc1 = pc;
        else if (lui)
            alusrc1 = 32'd0;
        else
            alusrc1 = rs1_data;

        // Second operand
        if (alusrc) begin
            if (opcode == `OP_LUI || opcode == `OP_AUIPC)
                alusrc2 = imm_u;
            else if (opcode == `OP_S)
                alusrc2 = imm_s;
            else
                alusrc2 = imm_i;
        end else begin
            alusrc2 = rs2_data;
        end
    end
```

7 Segment Map

```
#ifndef SEVENSEG_H_
#define SEVENSEG_H_

#define  GPIO_BASE      0xFFFF2000
#define  Button_Status  GPIO_BASE + 0
#define  SW_Status      GPIO_BASE + 1   //if pointer of int, :
#define  LEDG           GPIO_BASE + 2
#define  SevenSeg0      GPIO_BASE + 3
#define  SevenSeg1      GPIO_BASE + 4
#define  SevenSeg2      GPIO_BASE + 5
#define  SevenSeg3      GPIO_BASE + 6
#define  SevenSeg4      GPIO_BASE + 7
#define  SevenSeg5      GPIO_BASE + 8
#define  SEG_0      0x7E    /* Display "0" on 7 Segment */
#define  SEG_1      0x30    /* Display "1" on 7 Segment */
#define  SEG_2      0x6D    /* Display "2" on 7 Segment */
#define  SEG_3      0x79    /* Display "3" on 7 Segment */
#define  SEG_4      0x33    /* Display "4" on 7 Segment */
#define  SEG_5      0x5B    /* Display "5" on 7 Segment */
#define  SEG_6      0x5F    /* Display "6" on 7 Segment */
#define  SEG_7      0x72    /* Display "7" on 7 Segment */
#define  SEG_8      0x7F    /* Display "8" on 7 Segment */
#define  SEG_9      0x7B    /* Display "9" on 7 Segment */
#define  SEG_A      0x77    /* Display "A" on 7 Segment */
#define  SEG_B      0x1F    /* Display "B" on 7 Segment */
#define  SEG_C      0x4E    /* Display "C" on 7 Segment */
#define  SEG_D      0x3D    /* Display "D" on 7 Segment */
#define  SEG_E      0x4F    /* Display "E" on 7 Segment */
#define  SEG_F      0x47    /* Display "F" on 7 Segment */
#define  SEG_       0x01    /* Display "-" on 7 Segment */

#endif /* SEVENSEG_H_ */
```

Milestone(1)

C code

```
#include "SevenSeg.h"
int SevenSeg() {
        unsigned int * seg0_addr =
        (unsigned int *) SevenSeg0;

        *seg0_addr = SEG_5; while(1);
        return 0;
}
```

Assembly code

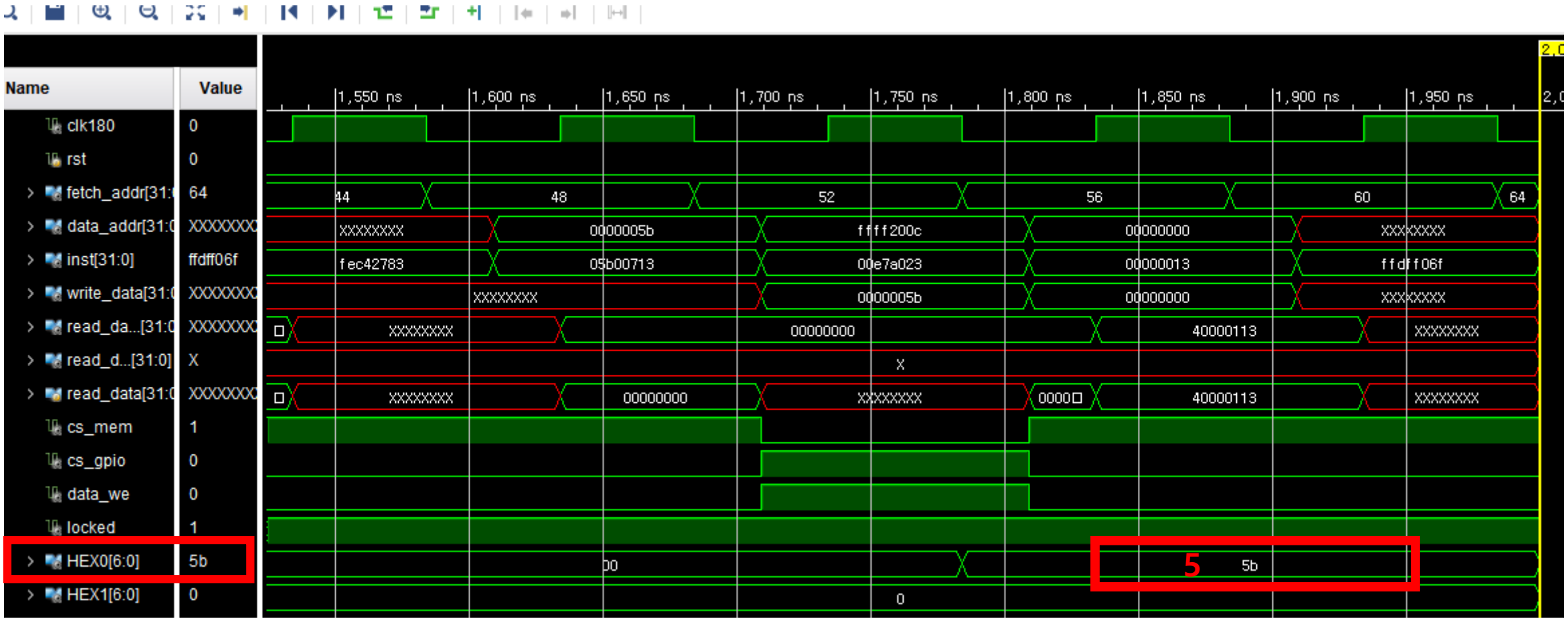
```
SevenSeg:
        addi    sp,sp,-32
        sw      ra,28(sp)
        sw      s0,24(sp)
        addi    s0,sp,32
        li      a5,-57344
        addi    a5,a5,12
        sw      a5,-20(s0)
        lw      a5,-20(s0)
        li      a4,91
        sw      a4,0(a5)
.L2:
        nop
        j       .L2
```

Machine Language

```
memory_initialization_radix=16;
memory_initialization_vector=
40000113,
00C0006F,
00000013,
00000013,
FE010113,
00112E23,
00812C23,
02010413,
FFFF27B7,
00C78793,
FEF42623,
FEC42783,
05B00713,
00E7A023,
00000013,
FFDFF06F,
00000000,
```

Milestone(1)-7 Seg 5 출력

Waveform Analysis



Milestone(2)- 7 seg LED 5/A toggle

C code

```
#include "SevenSeg.h"
void SevenSeg()
{
unsigned int * seg0_addr = (unsigned int *) SevenSeg0;
unsigned int * led_addr = (unsigned int *) LEDG;
unsigned int i, data;
data = 0x155;


while (1){
*seg0_addr = SEG_5;
*led_addr = data;
data = data ^ 0x3FF;



//for (i=0; i<0xFFFFF; i++) ;
for (i=0; i<0x10; i++) ;


*seg0_addr = SEG_A;
*led_addr = data;


data = data ^ 0x3FF;
//for (i=0; i<0xFFFFF; i++) ;
for (i=0; i<0x10; i++) ;
}
return;
```

Assembly code

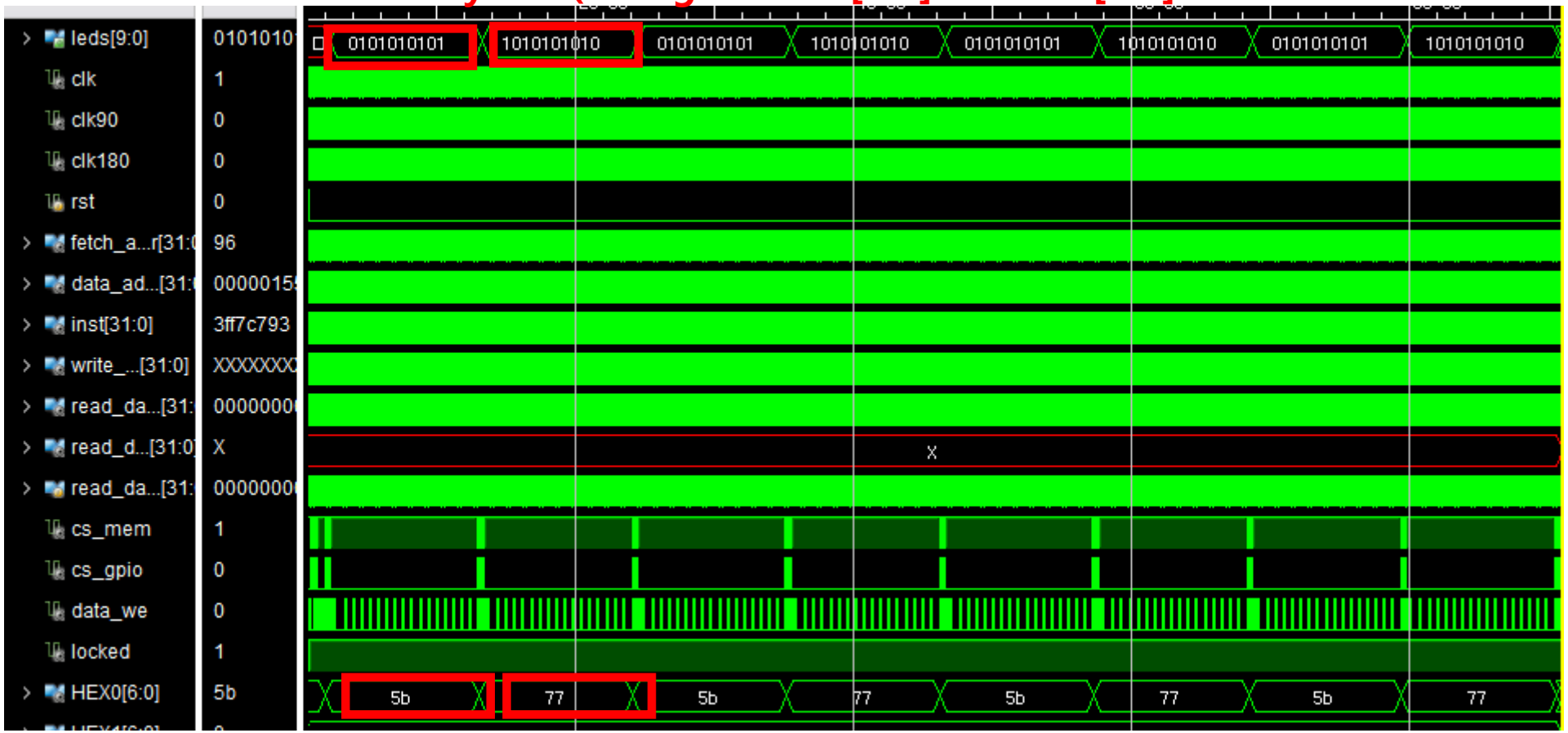
```
SevenSeg:
        addi    sp,sp,-32
        sw      ra,28(sp)
        sw      s0,24(sp)
        addi    s0,sp,32
        li      a5,-57344
        addi    a5,a5,12
        sw      a5,-28(s0)
        li      a5,-57344
        addi    a5,a5,8
        sw      a5,-32(s0)
        li      a5,341
        sw      a5,-24(s0)
.L6:
        lw      a5,-28(s0)
        li      a4,91
        sw      a4,0(a5)
        lw      a5,-32(s0)
        lw      a4,-24(s0)
        sw      a4,0(a5)
        lw      a5,-24(s0)
        xori    a5,a5,1023
        sw      a5,-24(s0)
        sw      zero,-20(s0)
        j       .L2
.L3:
        lw      a5,-20(s0)
        addi    a5,a5,1
        sw      a5,-20(s0)
.L2:
        lw      a4,-20(s0)
        li      a5,15
        bleu    a4,a5,.L3
        lw      a5,-28(s0)
        li      a4,119
        sw      a4,0(a5)
        lw      a5,-32(s0)
        lw      a4,-24(s0)
        sw      a4,0(a5)
        lw      a5,-24(s0)
        xori    a5,a5,1023
        sw      a5,-24(s0)
        sw      zero,-20(s0)
        j       .L4
.L5:
        lw      a5,-20(s0)
        addi    a5,a5,1
        sw      a5,-20(s0)
.L4:
        lw      a4,-20(s0)
        li      a5,15
        bleu    a4,a5,.L5
        j       .L6
```

Machine Language

```
memory_initialization_radix=16;
memory_initialization_vector=
40000113,
00C0006F,
00000013,
00000013,
FE010113,
00112E23,
00812C23,
02010413,
FFFF27B7,
00C78793,
FEF42223,
FFFF27B7,
00878793,
FEF42023,
15500793,
FEF42423,
FE442783,
05B00713,
00E7A023,
FE042783,
FE842703,
00E7A023,
FE842783,
3FF7C793,
FEF42423,
FE042623,
0100006F,
FEC42783,
00178793,
FEF42623,
FEC42703,
00F00793,
FEE7F6E3,
FE442783,
07700713,
00E7A023,
FE042783,
FE842703,
00E7A023,
FE842783,
3FF7C793,
FEF42423,
FE042623,
0100006F,
FEC42783,
00178793,
FEF42623,
FEC42703,
00F00793,
FEE7F6E3,
F79FF06F,
```

Milestone(2)- 7 seg LED 5/A toggle

Array LED(changed LEDS[3:0] to LEDS[9:0]



Milestone(2)- 7 seg LED 1/2 toggle(Branch)

C code

```
#include "SevenSeg.h"
void display (int);
void SevenSeg()
{
unsigned int * led_addr = (unsigned int *) LEDG;
unsigned int i, data;
data = 0x155;
while (1){
display(SEG_1);
*led_addr = data;
data = data ^ 0x3FF;
//for (i=0; i<0xFFFFF; i++) ;
for (i=0; i<0x2; i++) ;
display(SEG_2);
*led_addr = data;
data = data ^ 0x3FF;
//for (i=0; i<0xFFFFF; i++) ;
for (i=0; i<0x2; i++) ;
}
return;
}

void display (int num)
{
unsigned int * seg0_addr = (unsigned int *) SevenSeg0;
*seg0_addr = num;
return;
}
```

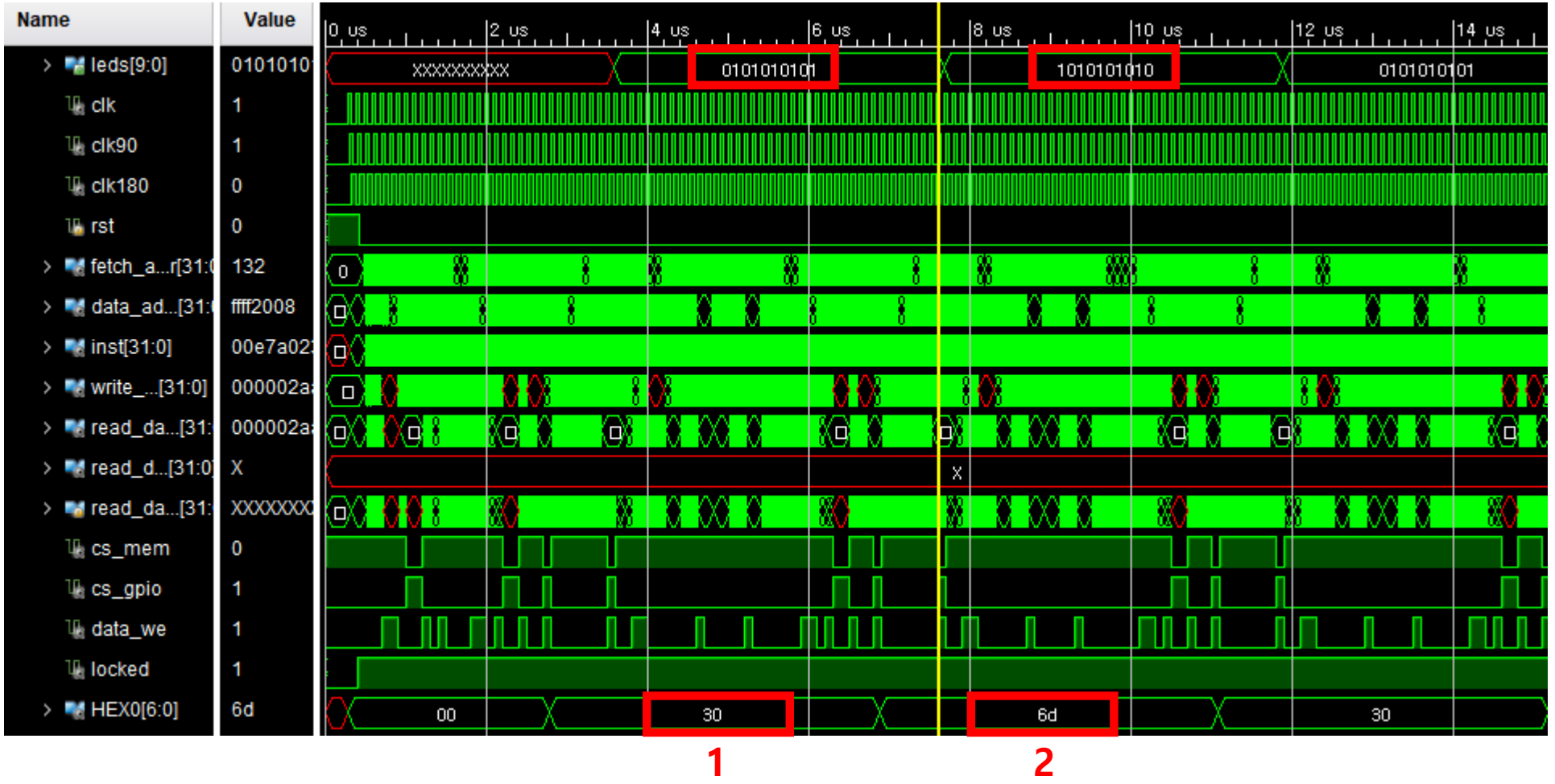
Assembly code

```
SevenSeg:
        addi    sp,sp,-32
        sw      ra,28(sp)
        sw      s0,24(sp)
        addi    s0,sp,32
        li      a5,-57344
        addi    a5,a5,8
        sw      a5,-28(s0)
        li      a5,341
        sw      a5,-24(s0)
.L6:
        li      a0,48
        call    display
        lw      a5,-28(s0)
        lw      a4,-24(s0)
        sw      a4,0(a5)
        lw      a5,-24(s0)
        xori    a5,a5,1023
        sw      a5,-24(s0)
        sw      zero,-20(s0)
        j       .L2
.L3:
        lw      a5,-20(s0)
        addi    a5,a5,1
        sw      a5,-20(s0)
.L2:
        lw      a4,-20(s0)
        li      a5,1
        bleu    a4,a5,.L3
        li      a0,109
        call    display
        lw      a5,-28(s0)
        lw      a4,-24(s0)
        sw      a4,0(a5)
        lw      a5,-24(s0)
        xori    a5,a5,1023
        sw      a5,-24(s0)
        sw      zero,-20(s0)
        j       .L4
.L5:
        lw      a5,-20(s0)
        addi    a5,a5,1
        sw      a5,-20(s0)
.L4:
        lw      a4,-20(s0)
        li      a5,1
        bleu    a4,a5,.L5
        j       .L6
        .size   SevenSeg, .-SevenSeg
        .align  2
        .globl  display
        .type   display, @function
display:
        addi    sp,sp,-48
        sw      ra,44(sp)
        sw      s0,40(sp)
        addi    s0,sp,48
        sw      a0,-36(s0)
        li      a5,-57344
        addi    a5,a5,12
        sw      a5,-20(s0)
        lw      a4,-36(s0)
        lw      a5,-20(s0)
        sw      a4,0(a5)
        nop
        lw      ra,44(sp)
        lw      s0,40(sp)
        addi    sp,sp,48
        jr      ra
```

Machine Language

```
memory_initialization_radix=16;
memory_initialization_vector=
40000113,
00C0006F,
00000013,
00000013,
FE010113,
00112E23,
00812C23,
02010413,
FFFF27B7,
00878793,
FEF42223,
15500793,
FEF42423,
03000513,
080000EF,
FE442783,
FE842703,
00E7A023,
FE842783,
3FF7C793,
FEF42423,
FE042623,
0100006F,
FEC42783,
00178793,
FEF42623,
FEC42703,
00100793,
FEE7F6E3,
06D00513
040000EF,
FE442783,
FE842703,
00E7A023,
FE842783,
3FF7C793,
FEF42423,
FE042623,
0100006F,
FEC42783,
00178793,
FEF42623,
FEC42703,
00100793,
FEE7F6E3,
F81FF06F,
FD010113,
02112623,
02812423,
03010413,
FCA42E23,
FFFF27B7,
00C78793,
FEF42623,
FDC42703,
FEC42783,
00E7A023,
00000013,
02C12083,
02812403,
03010113,
00008067,
```

Milestone(2)- 7 seg LED 1/2 toggle(Branch)



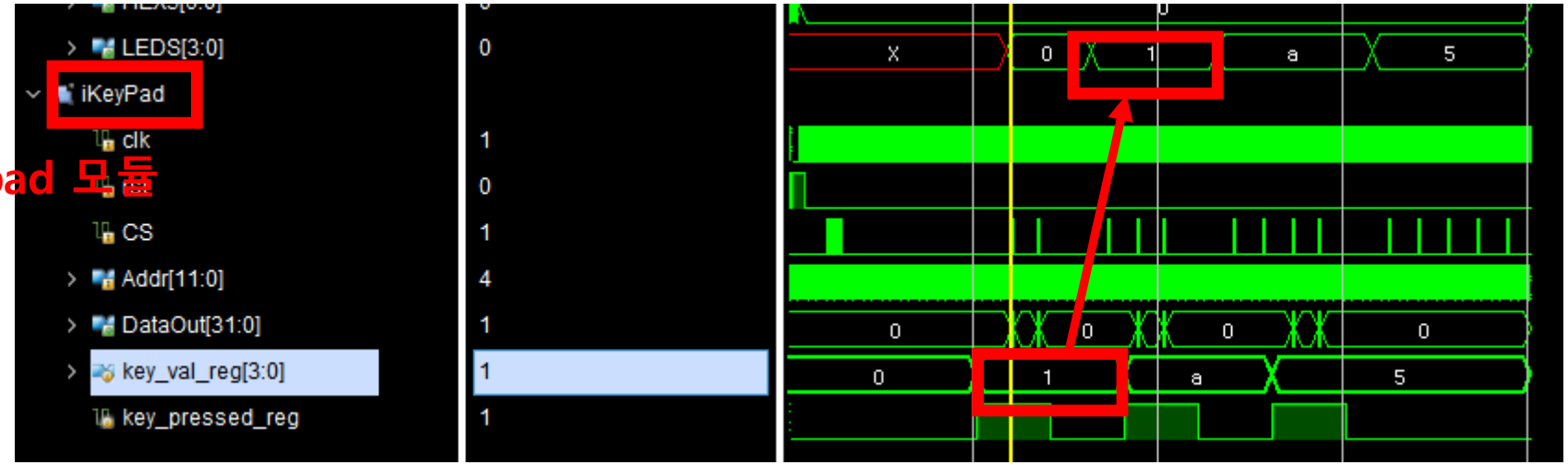
Milestone(3)- Keypad 입력 감지 Polling

```
#include "SevenSeg.h"

void SevenSeg()
{
    volatile unsigned int * key_status_addr = (unsigned int *) Key_Status;
    volatile unsigned int * key_valid_addr = (unsigned int *) Key_Valid;
    volatile unsigned int * seg0_addr = (unsigned int *) SevenSeg0;
    volatile unsigned int * led_addr = (unsigned int *) LEDG;

    unsigned int key_value;
    unsigned int key_pressed;
    unsigned int prev_valid = 0;

    unsigned int seg_table[16];
    seg_table[0] = SEG_0;
    seg_table[1] = SEG_1;
    seg_table[2] = SEG_2;
    seg_table[3] = SEG_3;
    seg_table[4] = SEG_4;
    seg_table[5] = SEG_5;
    seg_table[6] = SEG_6;
    seg_table[7] = SEG_7;
    seg_table[8] = SEG_8;
    seg_table[9] = SEG_9;
    seg_table[10] = SEG_A;
    seg_table[11] = SEG_B;
    seg_table[12] = SEG_C;
    seg_table[13] = SEG_D;
    seg_table[14] = SEG_E;
    seg_table[15] = SEG_F;

    *seg0_addr = SEG_;
    *led_addr = 0;

    while(1) {
        key_pressed = *key_valid_addr;

        if (key_pressed && !prev_valid) {
            key_value = *key_status_addr;
            key_value = key_value & 0xF;
            *seg0_addr = seg_table[key_value];
            *led_addr = key_value;
        }
        prev_valid = key_pressed;
    }
}
```

Milestone(3)-Keypad Input 검증(1, A, 5)

(Testbench로 외부입력 구현)

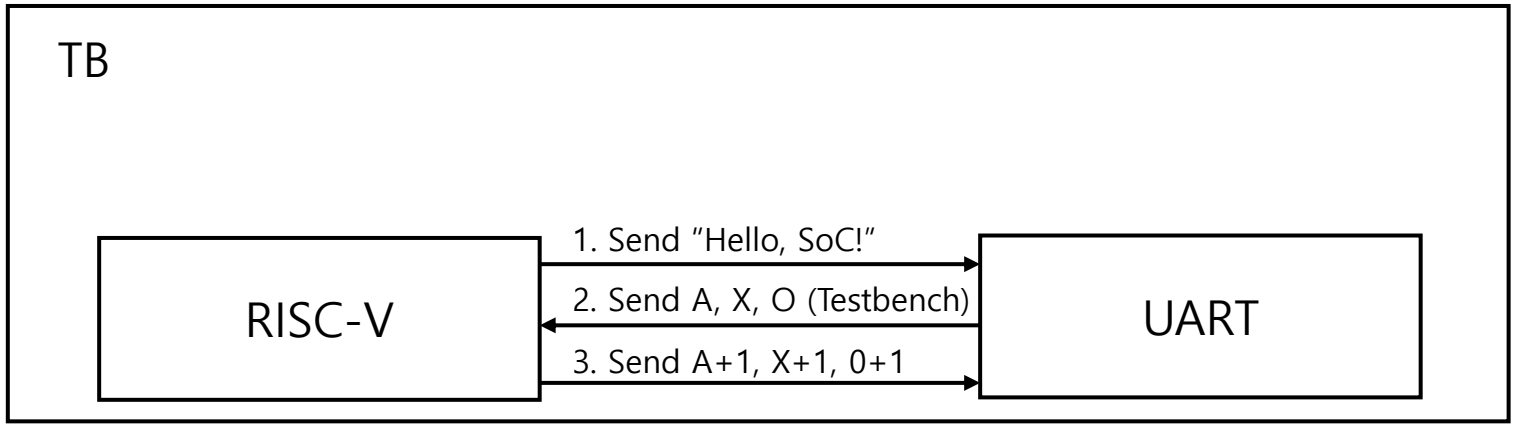
새로 추가한 keypad 모듈



Milestone(4)-UART I/O C code

```
#define UART_DATA (*(volatile unsigned int *)(0xFFFF3000))
#define UART_STAT (*(volatile unsigned int *)(0xFFFF3004))

#define STAT_TX_FULL  (1 << 0) // Bit 0
#define STAT_RX_EMPTY (1 << 1) // Bit 1

char uart_getc() {
    while (UART_STAT & STAT_RX_EMPTY);
    return (char)(UART_DATA & 0xFF);
}

void uart_putc(char c) {
    while (UART_STAT & STAT_TX_FULL);
    UART_DATA = c;
}

void uart_puts(char *s) {
    while(*s) uart_putc(*s++);
}

int main() {
    char rx_val;

    // RISCV -> TX
    uart_puts("Hello, SoC!₩n");

    // TB -. rx -> RISCV -> TX
    while(1) {
        // wait TB
        rx_val = uart_getc();

        // val + 1 (by RISCV)
        uart_putc(rx_val + 1);
    }

    return 0;
}
```

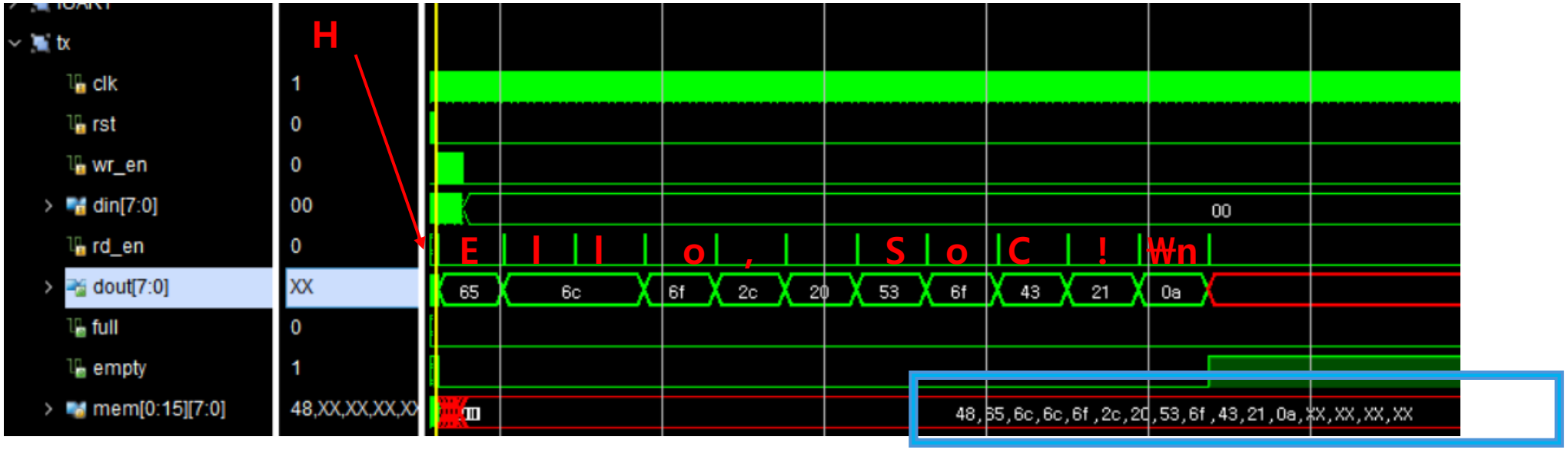
검증 1. TX로 "Hello, SoC!" 전송
검증 2. TB로 생성한 A, X, 0를 UART 모듈을 통해 RX로 전송
검증 3. RX로 받은 A, X, 0값에 +1하여 다시 TX로 전송



Milestone(4)-UART I/O

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
64	0x40	@	80	0x50	P	96	0x60	`	112	0x70	p
65	0x41	A	81	0x51	Q	97	0x61	a	113	0x71	q
66	0x42	B	82	0x52	R	98	0x62	b	114	0x72	r
67	0x43	C	83	0x53	S	99	0x63	c	115	0x73	s
68	0x44	D	84	0x54	T	100	0x64	d	116	0x74	t
69	0x45	E	85	0x55	U	101	0x65	e	117	0x75	u
70	0x46	F	86	0x56	V	102	0x66	f	118	0x76	v
71	0x47	G	87	0x57	W	103	0x67	g	119	0x77	w
72	0x48	H	88	0x58	X	104	0x68	h	120	0x78	x
73	0x49	I	89	0x59	Y	105	0x69	i	121	0x79	y
74	0x4A	J	90	0x5A	Z	106	0x6A	j	122	0x7A	z
75	0x4B	K	91	0x5B	[	107	0x6B	k	123	0x7B	{
76	0x4C	L	92	0x5C	₩	108	0x6C	l	124	0x7C	|
77	0x4D	M	93	0x5D	]	109	0x6D	m	125	0x7D	}
78	0x4E	N	94	0x5E	^	110	0x6E	n	126	0x7E	~
79	0x4F	O	95	0x5F	_	111	0x6F	o	127	0x7F	DEL

검증 1. TX로 "Hello, SoC!₩n" 전송

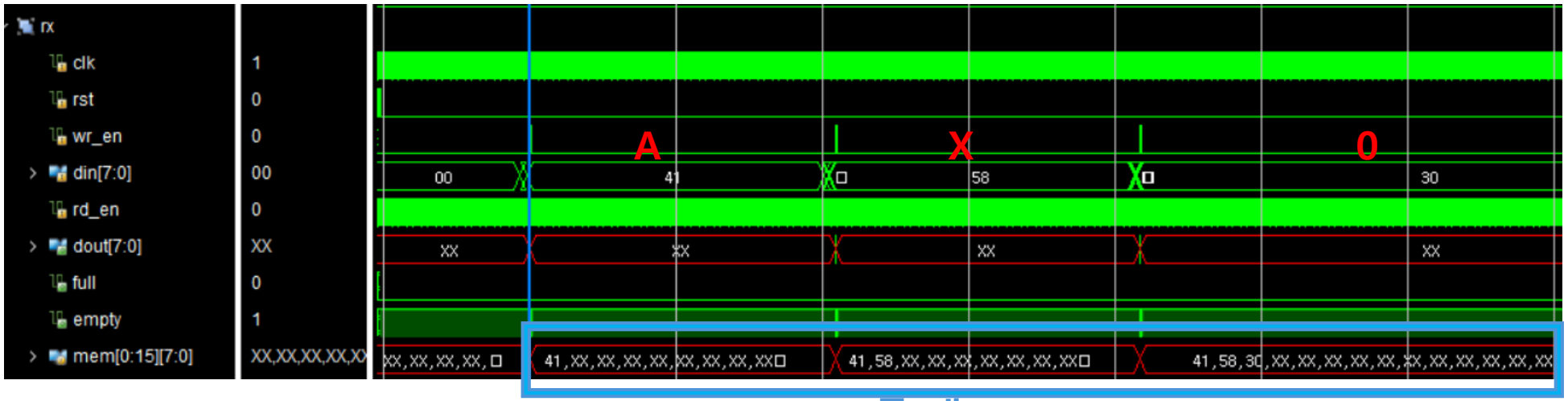


UART 모듈 내 MEM(FIFO)

Milestone(4)-UART I/O

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
64	0x40	@	80	0x50	P	96	0x60	`	112	0x70	p
65	0x41	A	81	0x51	Q	97	0x61	a	113	0x71	q
66	0x42	B	82	0x52	R	98	0x62	b	114	0x72	r
67	0x43	C	83	0x53	S	99	0x63	c	115	0x73	s
68	0x44	D	84	0x54	T	100	0x64	d	116	0x74	t
69	0x45	E	85	0x55	U	101	0x65	e	117	0x75	u
70	0x46	F	86	0x56	V	102	0x66	f	118	0x76	v
71	0x47	G	87	0x57	W	103	0x67	g	119	0x77	w
72	0x48	H	88	0x58	X	104	0x68	h	120	0x78	x
73	0x49	I	89	0x59	Y	105	0x69	i	121	0x79	y
74	0x4A	J	90	0x5A	Z	106	0x6A	j	122	0x7A	z
75	0x4B	K	91	0x5B	[	107	0x6B	k	123	0x7B	{
76	0x4C	L	92	0x5C	₩	108	0x6C	l	124	0x7C	|
77	0x4D	M	93	0x5D	]	109	0x6D	m	125	0x7D	}
78	0x4E	N	94	0x5E	^	110	0x6E	n	126	0x7E	~
79	0x4F	O	95	0x5F	_	111	0x6F	o	127	0x7F	DEL

검증 2. TB로 생성한 A, X, 0를 UART 모듈을 통해 RX로 전송

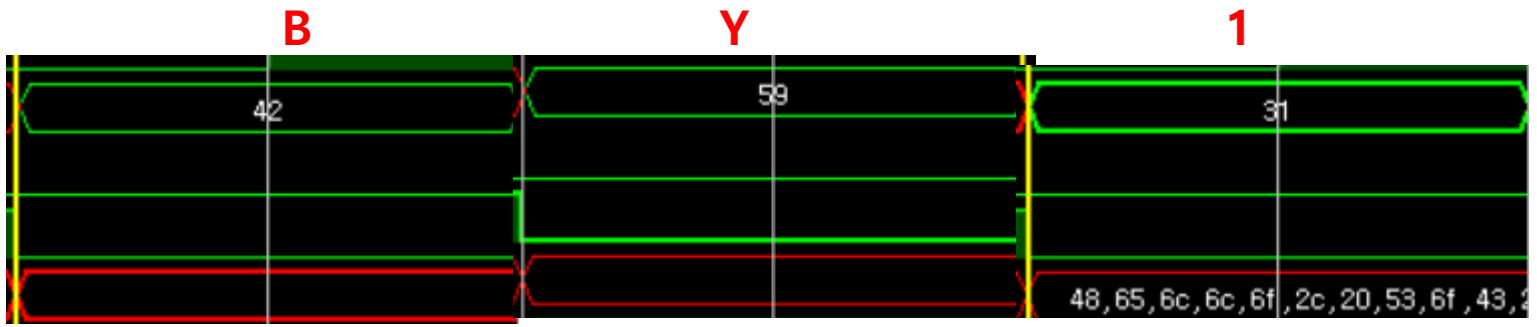


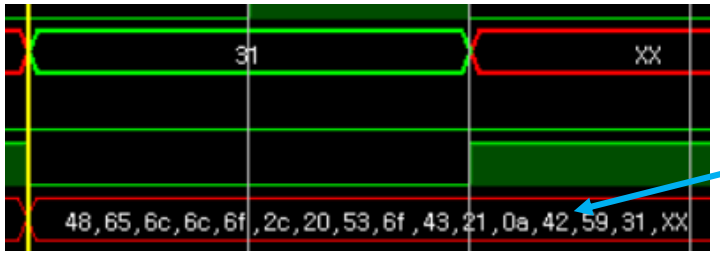
UART 모듈 내 MEM(FIFO)

Milestone(4)-UART I/O

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
64	0x40	@	80	0x50	P	96	0x60	`	112	0x70	p
65	0x41	A	81	0x51	Q	97	0x61	a	113	0x71	q
66	0x42	B	82	0x52	R	98	0x62	b	114	0x72	r
67	0x43	C	83	0x53	S	99	0x63	c	115	0x73	s
68	0x44	D	84	0x54	T	100	0x64	d	116	0x74	t
69	0x45	E	85	0x55	U	101	0x65	e	117	0x75	u
70	0x46	F	86	0x56	V	102	0x66	f	118	0x76	v
71	0x47	G	87	0x57	W	103	0x67	g	119	0x77	w
72	0x48	H	88	0x58	X	104	0x68	h	120	0x78	x
73	0x49	I	89	0x59	Y	105	0x69	i	121	0x79	y
74	0x4A	J	90	0x5A	Z	106	0x6A	j	122	0x7A	z
75	0x4B	K	91	0x5B	[	107	0x6B	k	123	0x7B	{
76	0x4C	L	92	0x5C	₩	108	0x6C	l	124	0x7C	|
77	0x4D	M	93	0x5D	]	109	0x6D	m	125	0x7D	}
78	0x4E	N	94	0x5E	^	110	0x6E	n	126	0x7E	~
79	0x4F	O	95	0x5F	_	111	0x6F	o	127	0x7F	DEL

검증 3. RX로 받은 A, X, 0값에 +1하여 다시 TX로 전송





UART 모듈 내 MEM(FIFO)으로도 추적 가능

Milestone(5)-SPI

spi.h

```
#ifndef SPI_H_
#define SPI_H_

// SPI Base Address
#define SPI_BASE      0xFFFF4000

// SPI Register Addresses
#define SPI_DATA      (SPI_BASE + 0x00)    // Data Register (TX/RX)
#define SPI_STATUS    (SPI_BASE + 0x04)    // Status Register
#define SPI_CTRL      (SPI_BASE + 0x08)    // Control Register

// Status Register Bits
#define SPI_BUSY      0x01
#define SPI_TX_DONE   0x02

// Function prototypes
void delay(unsigned int count);
unsigned char spi_transfer(unsigned char data);
void spi_test(void);

#endif /* SPI_H_ */
```

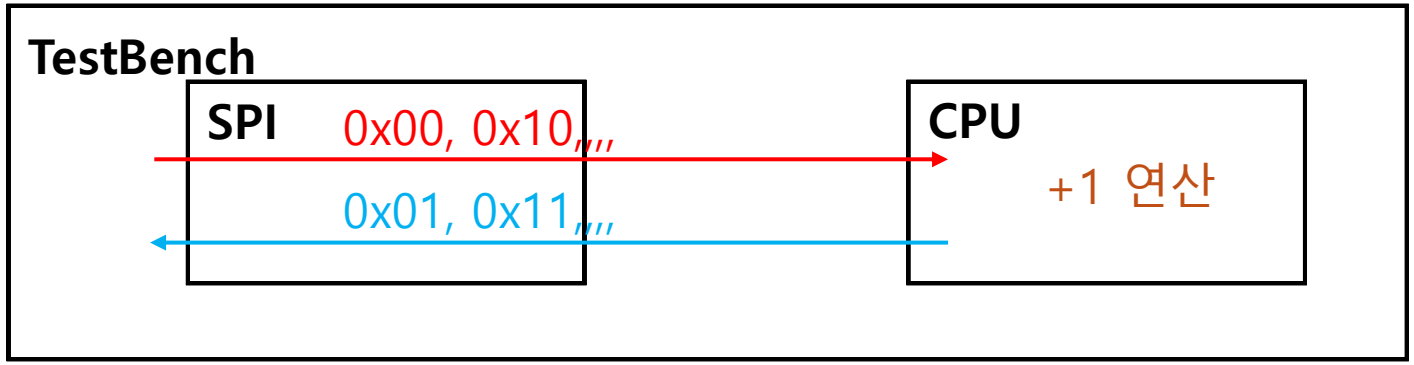
spi.c

```
#include "spi.h"

void delay(unsigned int count) {
    unsigned int i;
    for (i = 0; i < count; i++);
}

unsigned char spi_transfer(unsigned char data) {
    volatile unsigned int *spi_data_reg = (unsigned int *)SPI_DATA;
    volatile unsigned int *spi_ctrl_reg = (unsigned int *)SPI_CTRL;
    volatile unsigned int *spi_status_reg = (unsigned int *)SPI_STATUS;
    unsigned char rx_data;

    *spi_data_reg = data;
    *spi_ctrl_reg = 0x01;
    while ((*spi_status_reg & SPI_BUSY) != 0);
    rx_data = (unsigned char)(*spi_data_reg & 0xFF);

    return rx_data;
}

void spi_test(void) {
    unsigned char rx_data, tx_data;
    unsigned int i;

    tx_data = 0x00;

    while(1) {
        for (i = 0; i < 20; i++) {
            rx_data = spi_transfer(tx_data);
            tx_data = rx_data + 1;
            delay(0x100);
        }

        tx_data = 0x00;
        delay(0x1000);
    }
}
```

polling

CPU 연산

Milestone(5)-SPI



Tcl Console

```
[TB] Transaction #11: Sending 0xa0 to CPU
Transaction #11: Received 0x91 from CPU
PASS: CPU sent (prev + 1) = 0x91

[TB] Transaction #12: Sending 0xb0 to CPU
Transaction #12: Received 0xa1 from CPU
PASS: CPU sent (prev + 1) = 0xa1
```

검증과정