미니프로젝트: ALU 디자인

2017 Fall Logic Design LAB

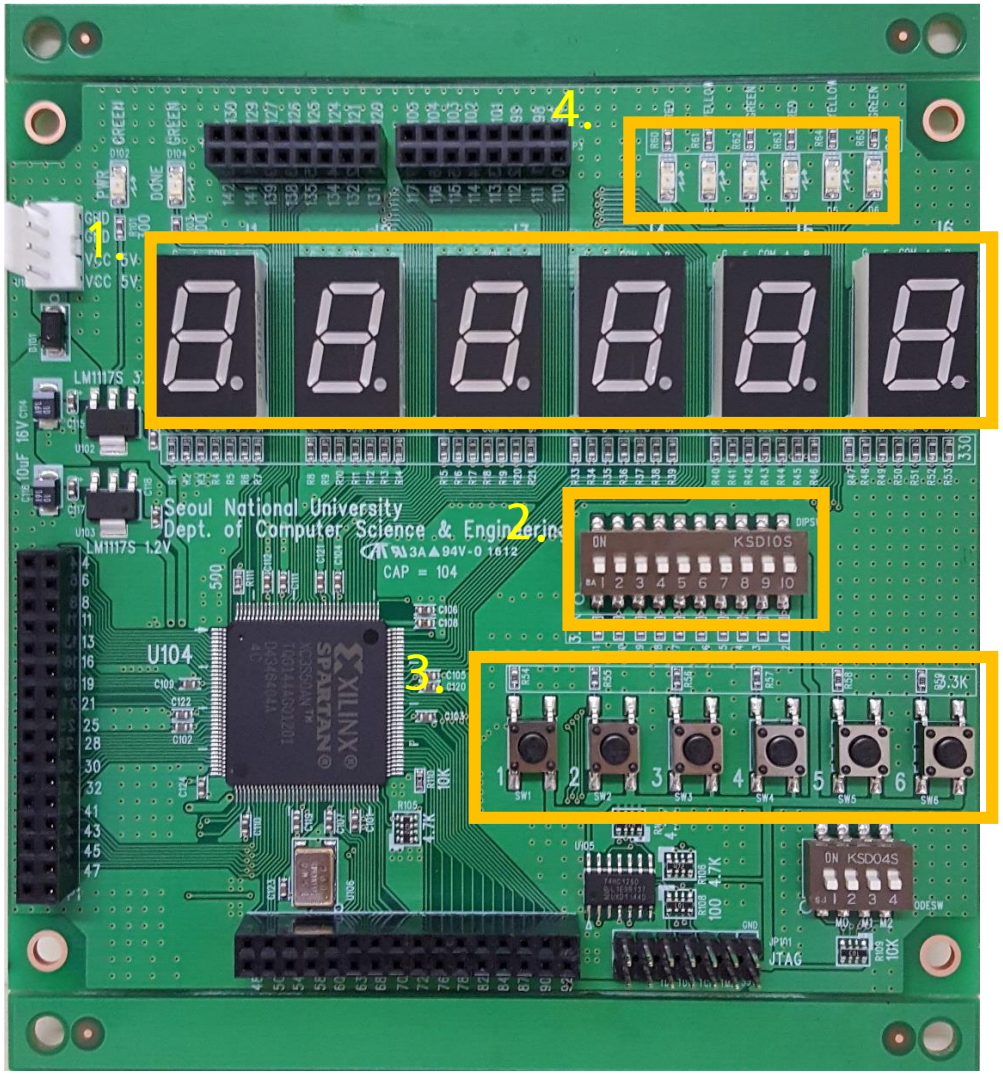
Department of Computer Science and Engineering

Seoul National University

목차

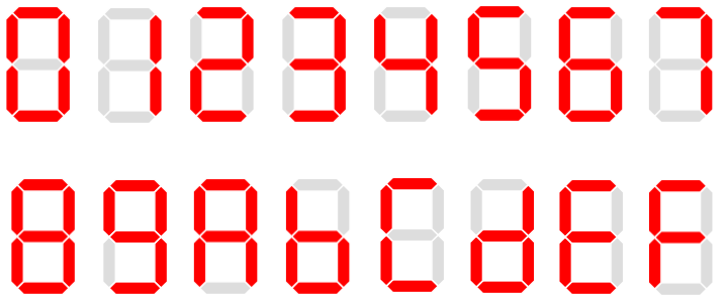
SNU Logic Design Board 설명

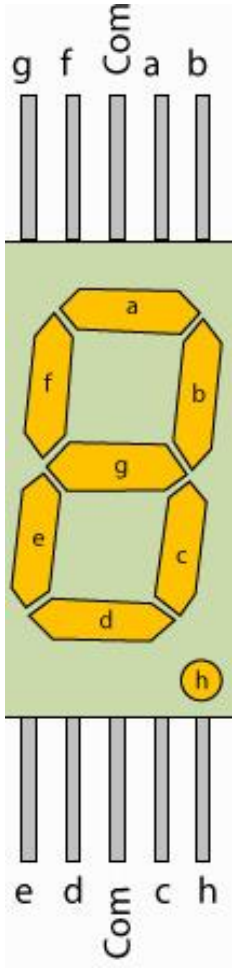
- SNU Logic Design Board 구성
  - 1. 7-Segment display 6개 (J1, J2, J3, J4, J5, J6)
  - 2. 10-pin DIP 스위치
  - 3. Tactile 스위치 6개 (SW1~SW6)
  - 4. LED 6개 (D1 ~ D6)



7-Segment Display

- 디지털 회로에서 숫자를 표시하기 위하여 사용되는 소자
  - 총 8개의 LED가 포함
    - 숫자 표시용 7개(a, b, c, d, e, f, g)
    - 점 표시용 1개(h)
- 7-Segment Display에서의 16진수 표현
  - 0 ~ F의 7-Segment Display 표시

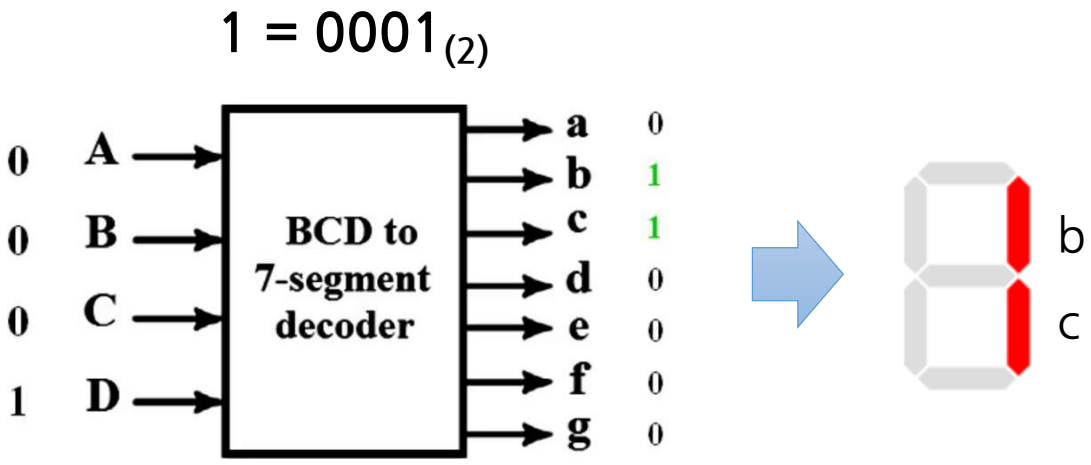


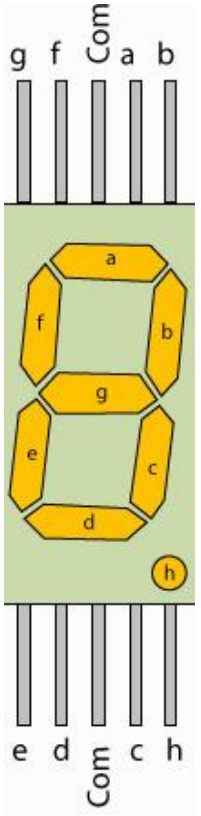


7-Segment 디코더

- Binary-coded Hexadecimal(BCH)를 7-Segment Display 숫자 형태로 변환 (decoding)

- 예) 숫자 1의 변환

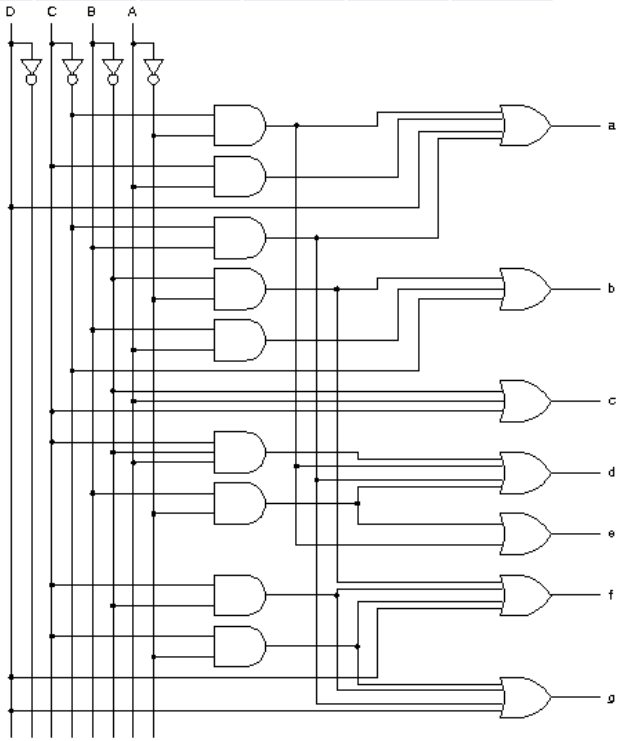




7-Segment 디코더

- 진리표 및 논리식, 회로도

Number	Binary input				7-segment output						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1					
3	0	0	1	1	1	1					
4	0	1	0	0	0	1					
5	0	1	0	1	1	0					
...						...					



a = D'+(C'*A)+(C*A)+B'

b = C+(B'*A')+(B*A)

c = C'+B+A'

d = C'+(C'*B*A')+(C*B')+(C*A)+(B'*A)

e = (C*A)+(B'*A)

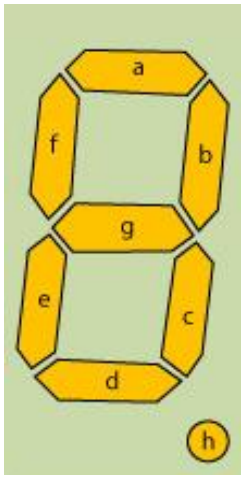
f = D'+(C'*B)+(B'*A)+(B*A)

g = D'+(C'*B)+(C'*A)+(C*B')

7-Segment 디코더 예

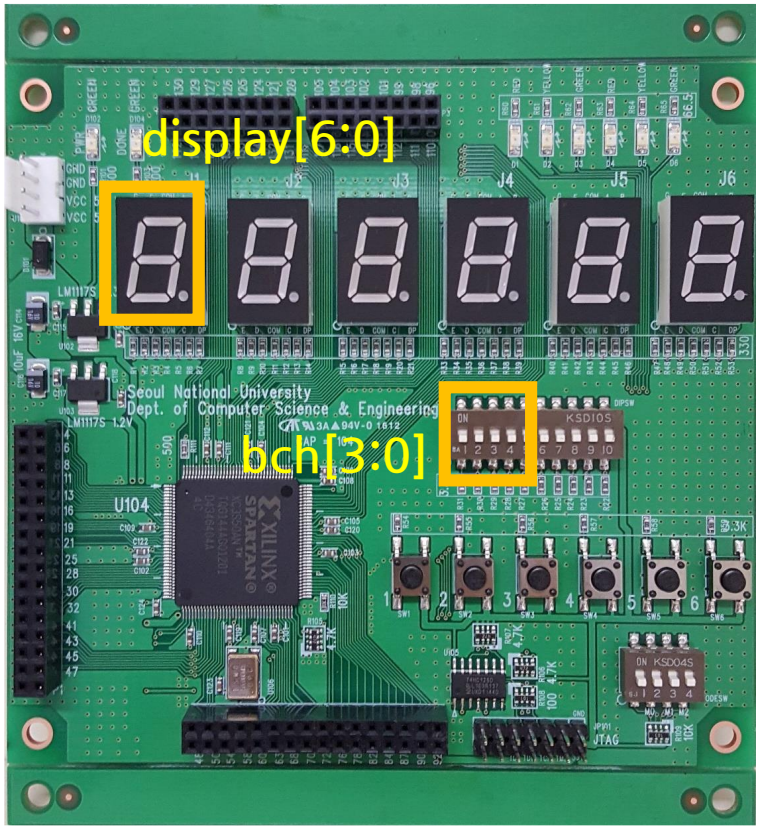
- 0 ~ 9까지 출력 가능한 7-Segment decoder 코드

```
module BCHto7SegmentDecoder (
    input [3:0] bch,
    output reg [6:0] display);
always @ (*)
begin
    case (bch)
        4'd0: display = 7'b1111110;
        4'd1: display = 7'b0110000;
        4'd2: display = 7'b1101101;
        4'd3: display = 7'b1111001;
        4'd4: display = 7'b0110011;
        4'd5: display = 7'b1011011;
        4'd6: display = 7'b1011111;
        4'd7: display = 7'b1110000;
        4'd8: display = 7'b1111111;
        4'd9: display = 7'b1111011;
        default: display = 7'b0000000;
    endcase
end
endmodule
```



실습 1 - 7-Segment Display 실험

- 목표
  - SNU보드의 좌측 첫 번째 7-Segment Display에 숫자 출력
- 입출력
  - 입력: DIP 스위치의 1,2,3,4 번에 4'b0000 부터 4'b1001을 입력
  - 출력: 7-Segment Display(J1)에 해당 숫자를 출력



실습 1 - 7-Segment Display 실험

- FPGA 핀 배정 (SNU Logic Design Board User's Manual 참고)
  - UCF 파일 생성
  - 출력 설정

```
# 7-Segment Display (J1)
NET "display[6]" LOC = P3;
NET "display[5]" LOC = P4;
NET "display[4]" LOC = P5;
NET "display[3]" LOC = P6;
NET "display[2]" LOC = P7;
NET "display[1]" LOC = P8;
NET "display[0]" LOC = P10;
```

  - 입력 설정

```
# DIP 스위치 1~4
NET "bch[3]" LOC = P30;
NET "bch[2]" LOC = P31;
NET "bch[1]" LOC = P32;
NET "bch[0]" LOC = P33;
```

실습 1 - 7-Segment Display 실험

- 변경된 UCF파일과 앞서 제시된 **`BCHto7SegmentDecoder`** 모듈을 사용하여 BCH를 7-Segment Display에 출력

- 확인
  - 7-Segment Display(J1)과 DIP 스위치(SW1)를 이용한 7-Segment 표현을 완성하고 조교에게 동작 검증을 받을 것.

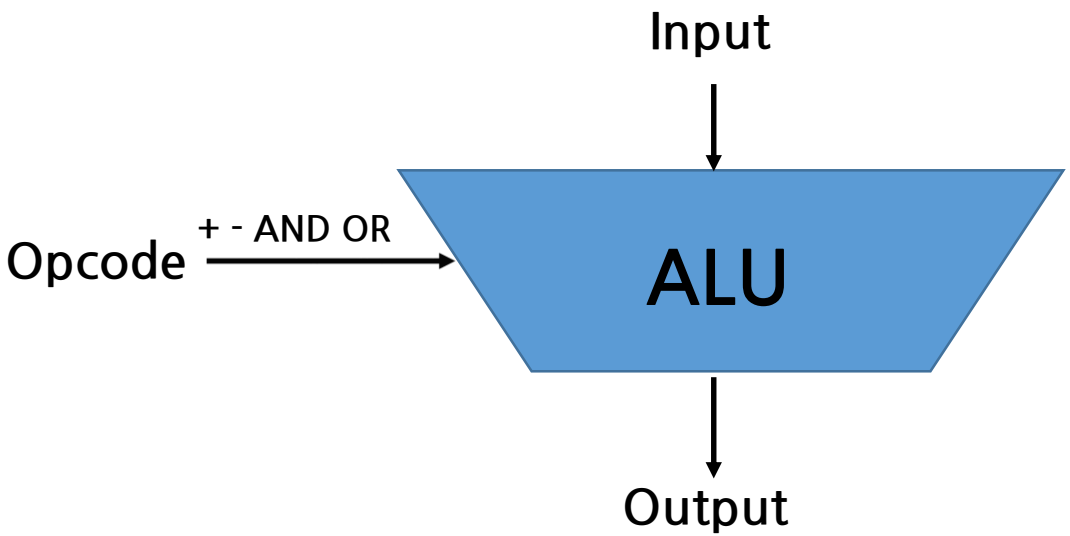
미니프로젝트: ALU 디자인

ALU

- ALU (Arithmetic Logic Unit)
  - 입력에 따라 원하는 operation 수행 후 결과 산출
    - Input에 operation 수행 후 output 산출
    - Opcode가 operation 결정
  - 일반적인 기능
    - 산술(Arithmetic) 연산

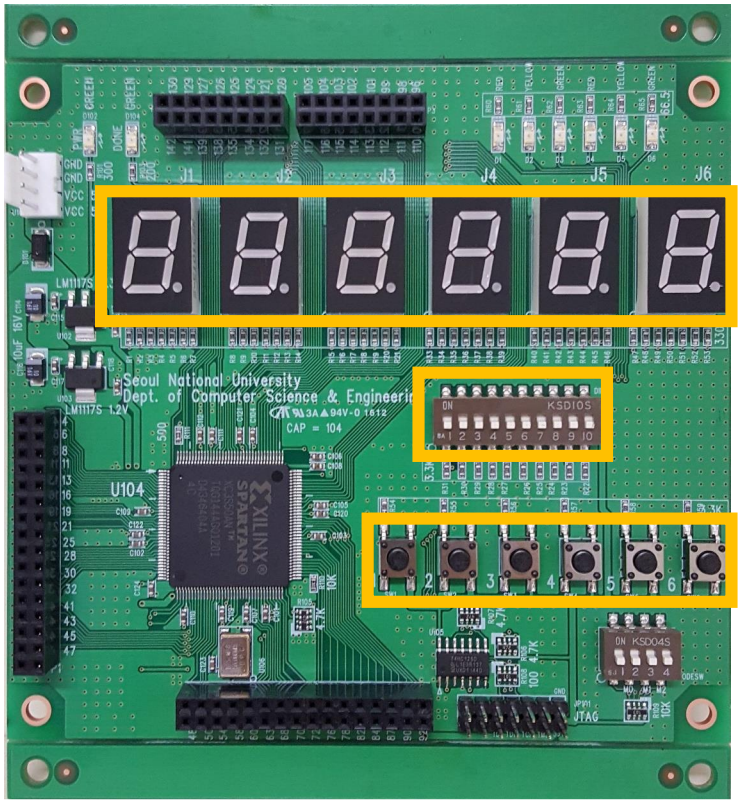
      예) 덧셈, 뺄셈
    - 논리(Logic) 연산

      예) AND, OR



ALU: 입출력 개요

- SNU Logic Design Board활용
  - 7-Segment Display(J1,J2,J3,J4,J5,J6)
    - 연산의 대상이 될 피연산자와 연산 결과값을 지정된 형식에 따라 출력
  - DIP 스위치
    - Input 입력
  - 푸시 스위치(SW1~SW6)
    - 각 스위치의 입력에 따라 정해진 기능을 수행하도록 회로를 동작시키거나 7-Segment Display에 표시하는 모드를 변경 (Opcode)

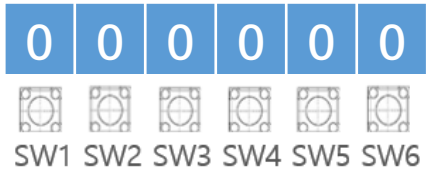


ALU: 연산 개요

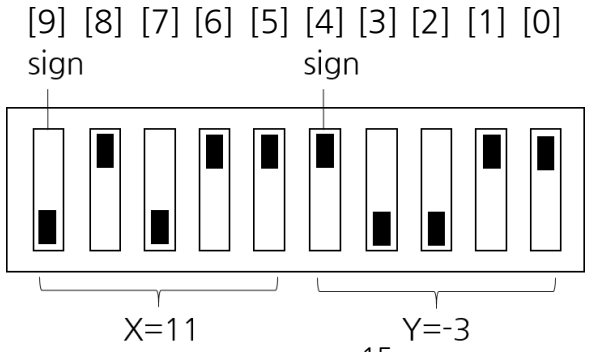
- ALU는 푸시스위치의 입력에 따라 다음의 7가지 방식으로 동작한다.
  - Operation 0: 두 개의 피연산자(Signed 4-bit integer)를 hexadecimal으로 표현
    - 아무런 푸시 스위치도 누르지 않은 경우
  - Operation 1: 한 개의 피연산자(6-bit vector)를 binary로 표현 (SW1 누름)
  - Operation 2: Signed 4-bit adder (SW2 누름)
  - Operation 3: Signed 4-bit multiplier (SW3 누름)
  - Operation 4: Number of 1's & 0's counter (SW4 누름)
  - Operation 5: 6-bit shifter (SW5 누름)
  - Operation 6: 6-bit median filter (SW6 누름)
- 두 개 이상의 푸시 스위치를 누른 경우
  - SW1 > SW2 > SW3 > SW6 > SW5 > SW4의 우선순위를 가짐

ALU 동작 설명 (Operation 0)

- Operation 0: Signed 4-bit Integer의 Hexadecimal 표현
- Operation 수행 조건
  - 아무런 푸시스위치도 누르지 않은 상태:

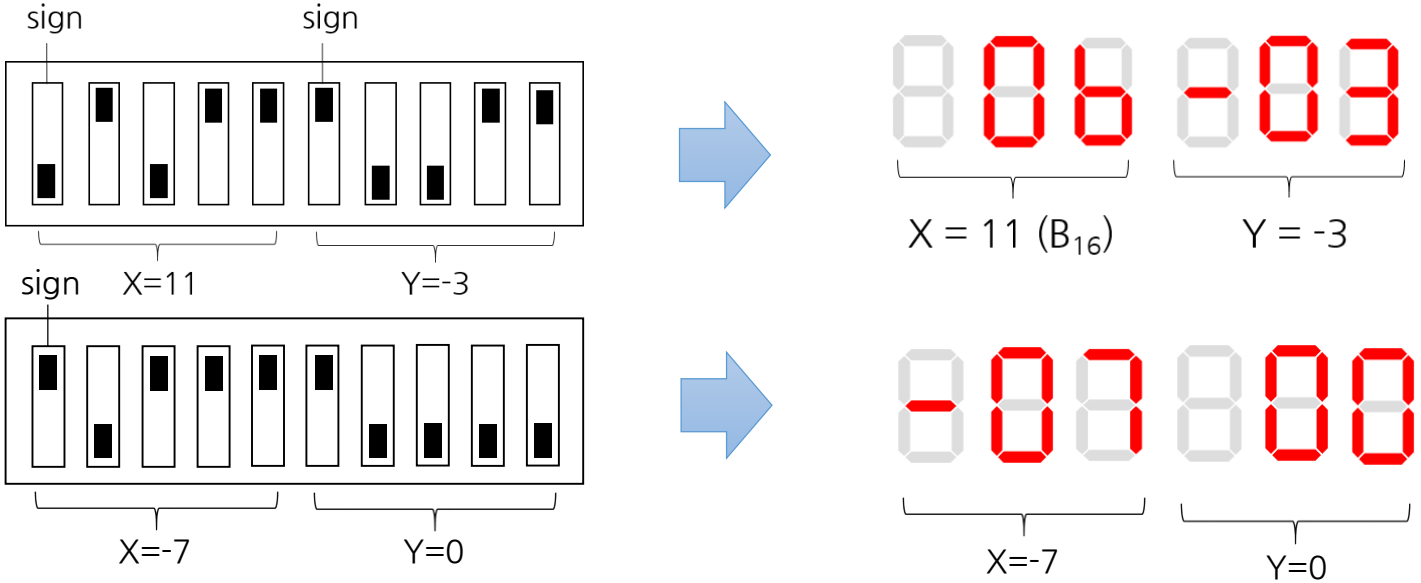


- DIP 스위치 입력 (DIP[9:0], 맨 앞이 index 9)
  - DIP[9:5]는 signed 4-bit integer X 값을 의미
  - DIP[4:0]는 signed 4-bit integer Y 값을 의미
  - 각 숫자의 첫 번째 값(DIP[9], DIP[4])는 sign bit(1일 경우 음수)
  - DIP[8:5]와 DIP[3:0]은 각각 X, Y의 절대값의 binary 표현값 (0x00 ~0x0F)



ALU 동작 설명 (Operation 0)

- 7-Segment 출력
  - Sign은 1자리, 절대값의 2진수는 2개의 7-Segment Display로 표현 (모두 표시)
    - 좌측의 7-Segment Display 3개는 X값을 Hexadecimal로 표현
    - 우측의 7-Segment Display 3개는 Y값을 Hexadecimal로 표현
  - 음수일 경우에는 (-) 부호, 양수일 경우에는 아무 것도 출력하지 않음
    - 0의 경우 무조건 양수로 표현할 것(-00 불허)

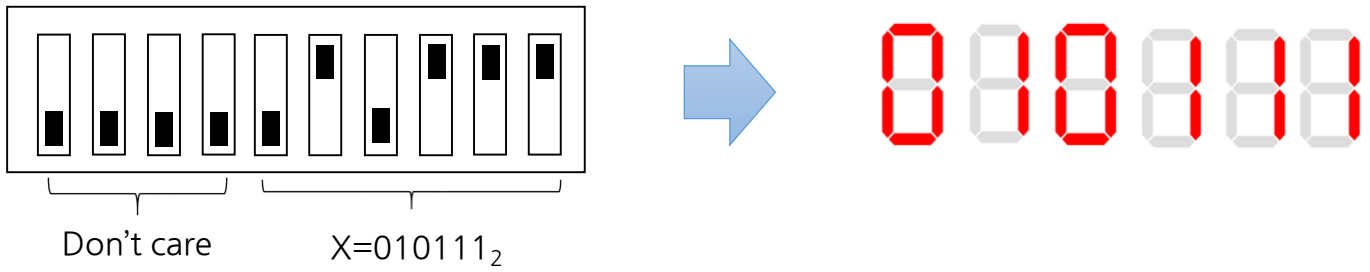


ALU 동작 설명 (Operation 1)

- Operation 1: 6-bit Binary 표시
- Operation 수행 조건
  - SW1을 누른 경우:

1	X	X	X	X	X

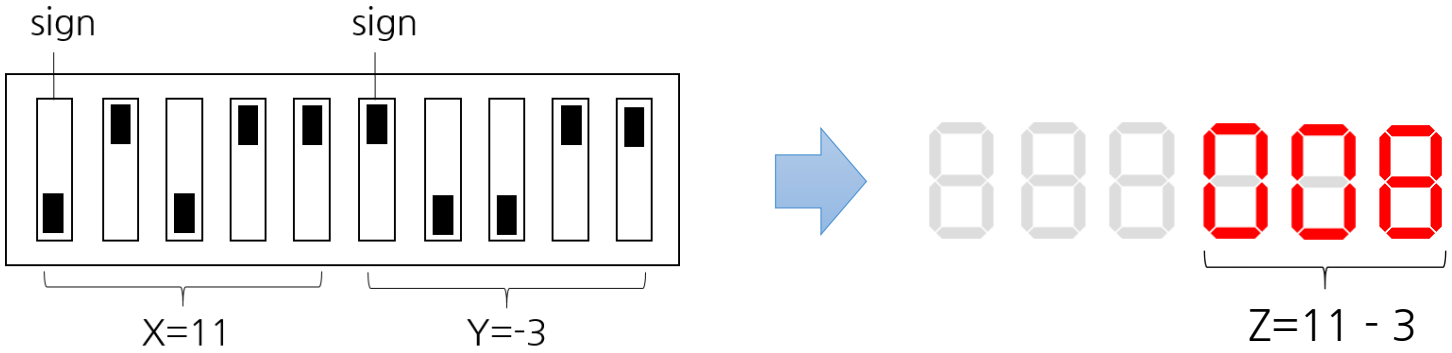
- DIP 스위치의 LSB 6-bit 입력 값을 binary로 7-segment display에 출력
  - 6-bit의 입력에 대해 각 자릿수를 binary로 0 혹은 1로 표현
  - 6개의 7-segment display를 모두 사용



ALU 동작 설명 (Operation 2)

- Operation 2: Signed 4-bit Adder
- Operation 수행 조건
  - SW2을 누른 경우:

- Operation 0에서 DIP 스위치로 입력된 X,Y의 합을 7-segment display에 출력
  - 결과값을 hexadecimal로 표시
  - 우측 4개의 7-segment display에 결과 값 표시
    - 부호: 1자리, 절대값: 3자리 (모두 표시)

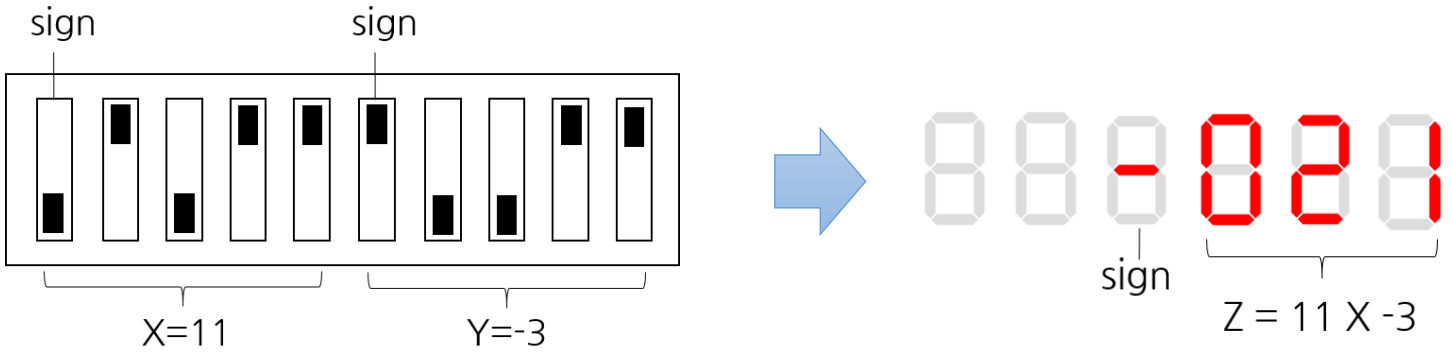


ALU 동작 설명 (Operation 3)

- Operation 3: Signed 4-bit Multiplier
- Operation 수행 조건
  - SW3를 누른 경우:

0	0	1	X	X	X

- Operation 0에서 DIP 스위치로 입력된 X,Y의 곱을 7-segment display에 출력
  - 결과값을 hexadecimal로 표시
  - 우측 4개의 7-segment display에 결과 값 표시
    - 부호: 1자리, 절대값: 3자리 (모두 표시)

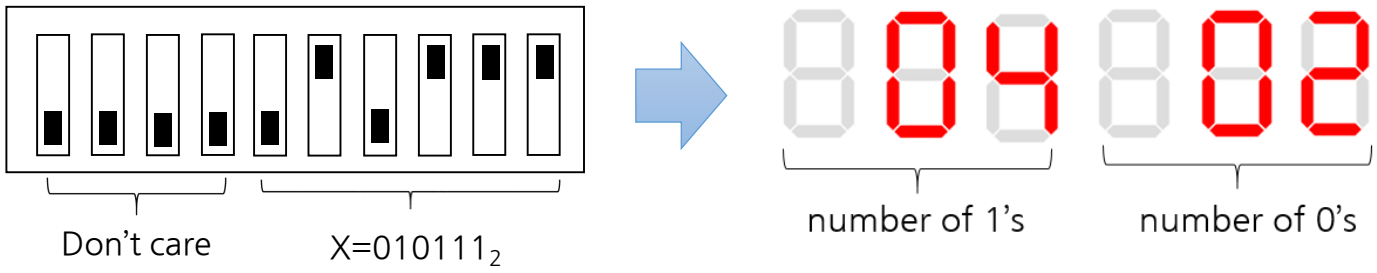


ALU 동작 설명 (Operation 4)

- Operation 4: Number of 1's & 0's
- Operation 수행 조건
  - SW4을 누른 경우:

0	0	0	1	X	X

- Operation 1의 LSB 6-bit vector를 입력으로 받아 1의 개수와 0의 개수를 각각 7-segment display에 출력
  - 좌측 3개의 7-segment display에는 1의 개수를 출력 (2자리 모두 표시)
  - 우측 3개의 7-segment display에는 0의 개수를 출력 (2자리 모두 표시)

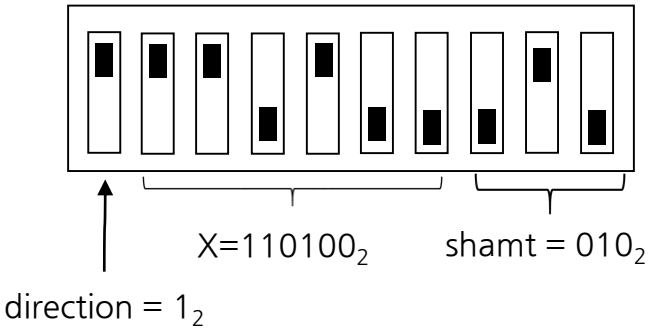


ALU 동작 설명 (Operation 5)

- Operation 5: 6-bit Shifter
- Operation 수행 조건
  - SW5을 누른 경우:

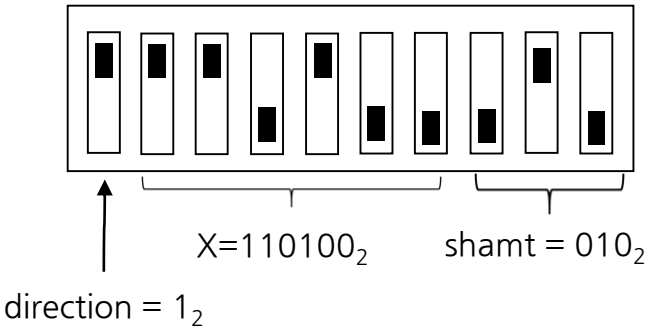
0	0	0	0	1	X

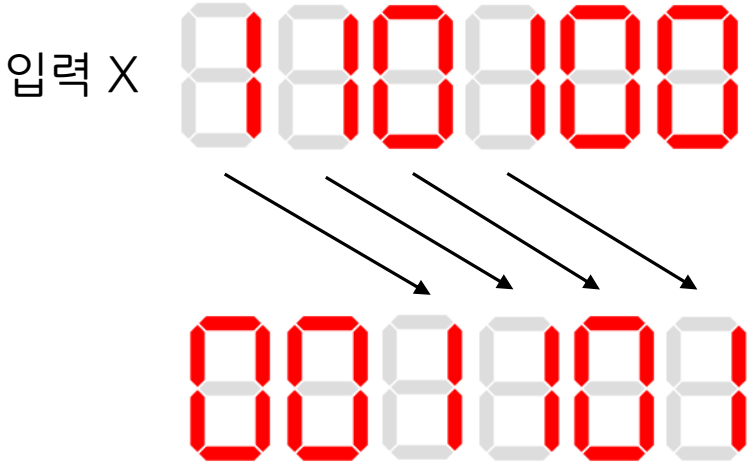
- 입력
  - X: DIP 스위치의 좌측 2번째부터 6-bit (DIP[8:3])
  - Shift amount(shamt): DIP 스위치의 우측 3-bit(DIP[2:0])
  - Direction: DIP 스위치의 제일 왼쪽 1-bit(DIP[9])
    - 왼쪽은 1'b0, 오른쪽은 1'b1



ALU 동작 설명 (Operation 5)

- 동작
  - 입력된 X 값을 shamt (0~7)만큼 왼쪽 또는 오른쪽으로 shift한다.
  - 방향: 왼쪽은 1'b0, 오른쪽: 1'b1
  - Shift 연산으로 인해 비워지는 자리는 0으로 채워 넣는다.
    - 예) shamt가 6 또는 7인 경우, 입력에 상관없이 모든 자릿수는 0

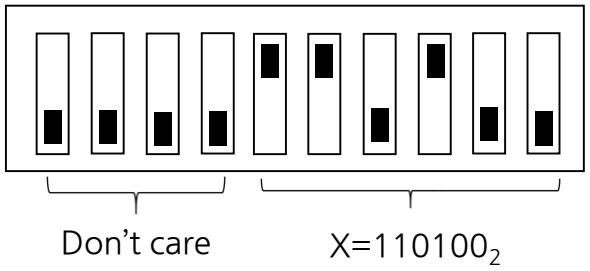


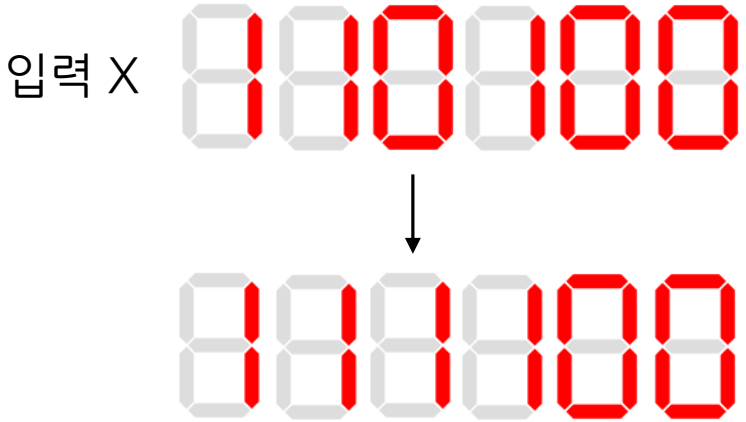


ALU 동작 설명 (Operation 6)

- Operation 6: 6-bit Median Filter
- Operation 수행 조건
  - SW6을 누른 경우:

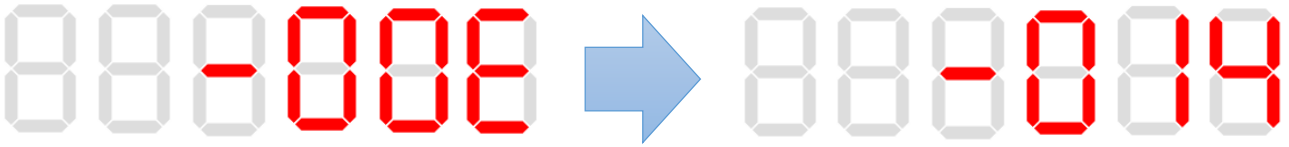
- 입력 X: DIP 스위치의 LSB 6-bit (DIP[5:0])
- 동작: 6-bit Vector에서 '101' 패턴을 '111'로 치환





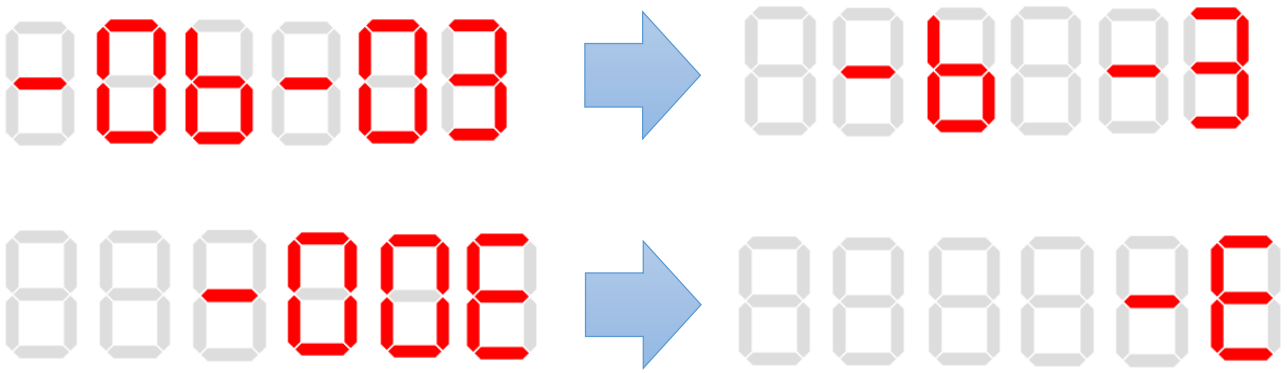
ALU 동작 설명 (추가구현 1)

- 숫자의 10진수 표현 (Operation 0, 2, 3)
  - 앞에서 Hexadecimal로 표현하는 모드들을 모두 Decimal로 표현
  - 10진수로 표현한 경우 16진수로 표현하는 기본 스펙을 만족한 것으로 간주

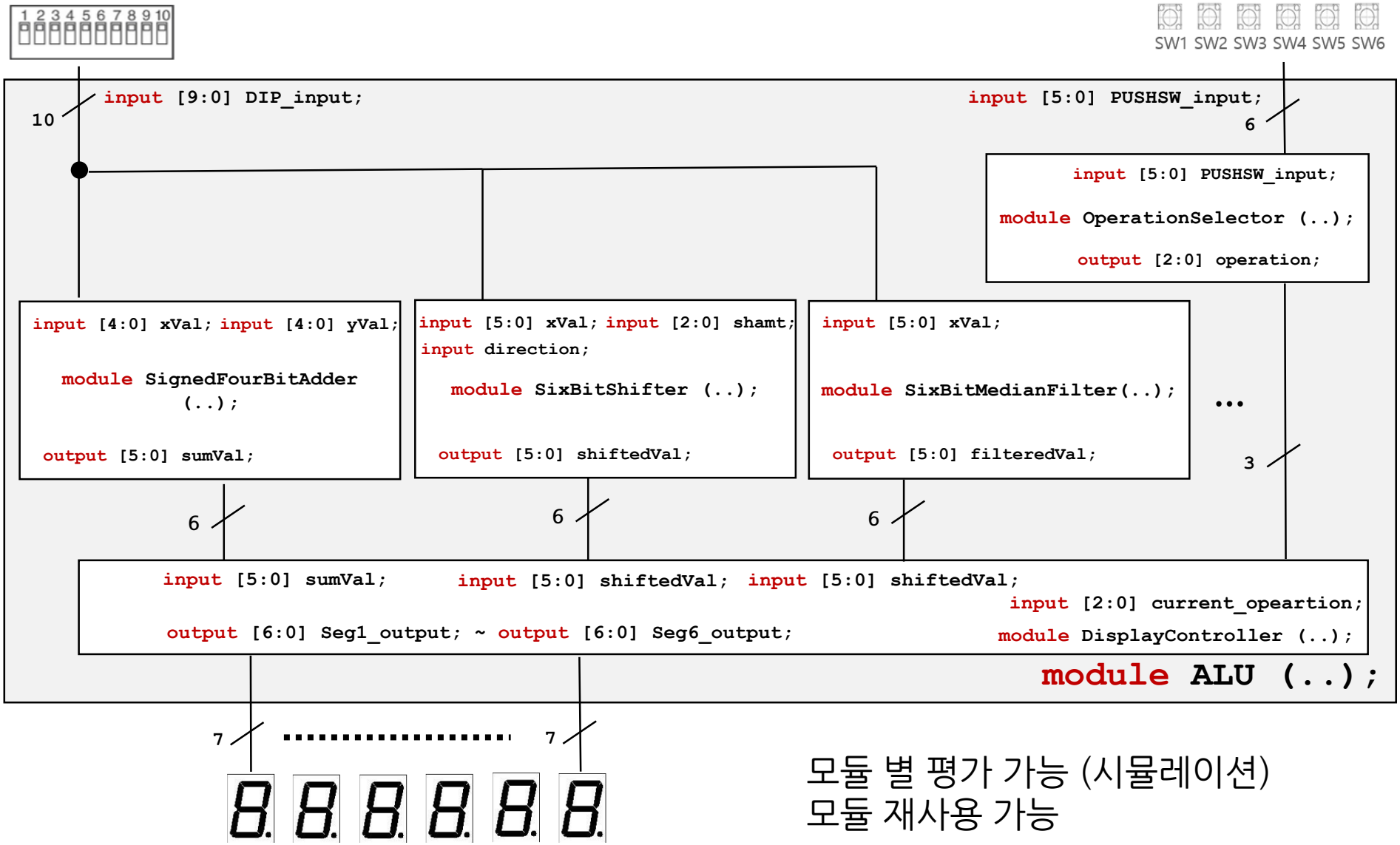


ALU 동작 설명 (추가구현 2)

- Leading zero 미표시 및 부호 표시 이동 (Operation 0, 2, 3, 4)
  - 지정된 자리 수(피 연산자: 2자리, 결과값: 3자리)를 모두 채우지 못하는 경우
    - 해당하는 수가 양수이면 앞의 0을 출력하지 않음
    - 해당하는 수가 음수이면 앞의 0을 출력하지 않으면서 숫자의 가장 높은 자릿수 좌측에 부호를 표시



ALU 디자인 예시



모듈 별 평가 가능 (시뮬레이션)
모듈 재사용 가능

평가: 배점

- 기본 구현

모드	기능	배점
Operation 0	Signed 4-bit Integer의 Hexadecimal 표현	40점
Operation 1	6-bit Binary 표시	15점
Operation 2	Signed 4-bit Adder	20점
Operation 3	Signed 4-bit Multiplier	30점
Operation 4	Number of 1's & 0's	15점
Operation 5	6-bit Shifter	15점
Operation 6	6-bit Median Filter	15점
총점		150점

- 추가 구현

모드	기능	배점
추가구현 1	Signed 4-bit Integer의 Decimal 표현	Bonus Day 1일 and 출석 1회
추가구현 2	Leading Zero 미 표시	Bonus Day 1일 and 출석 1회
총점		0점

보고서 제출 및 평가 방법

- 제출 기한: 11월 15일 실습시간 (18시 30분) 전까지 ETL에 업로드
- 프로젝트 폴더와 보고서를 압축하여 제출
  - 보고서
    - 구현 사항, 모듈 설명 포함 (2~3장 내외)

      예) K-map, 모듈 연결 등 설계 사항
  - 프로젝트 폴더
    - Cleanup Project Files 하여 제출
- 평가 방법
  - 동작 검사: 11월 15일 실습시간에 조교들에게 동작 확인
  - Copy 적발 시 F

Help Desk

- Visiting Q&A Session
  - 연구실에 직접 방문하여 논리설계 조교에게 도움 요청
  - 시간: 11월 8일 수요일 18:00~18:30, 11월 13일 월요일 18:30~20:30
  - 장소: 연구실 방문 (302동 315-2호 임베디드 시스템 연구실)