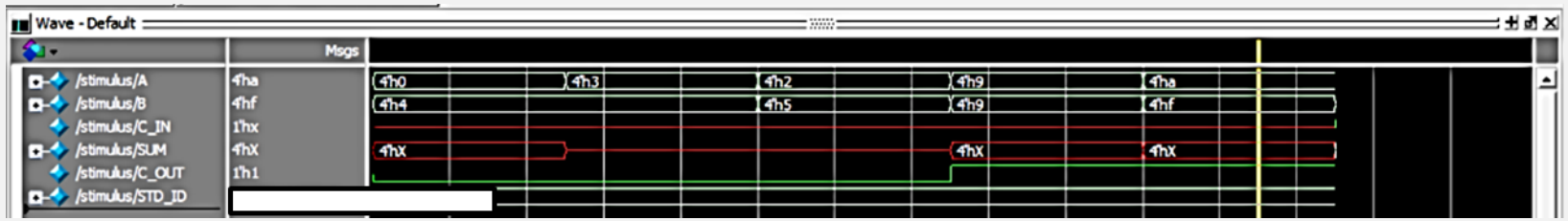
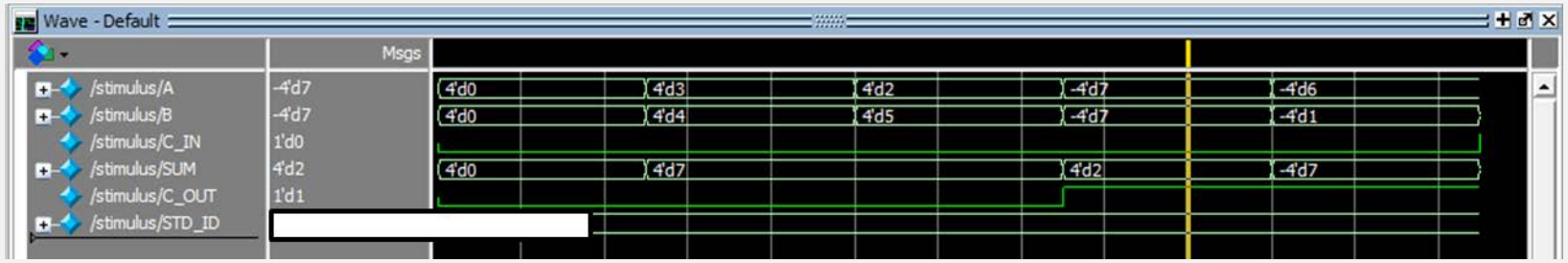
LAB 2:
8-bit Addition/Subtraction Module

Department of Computer Science
Yonsei University

HW1 ModelSim Simulation Recap

- Wave 결과 오류
- Wave radix 오류
- Wave 식별 불가



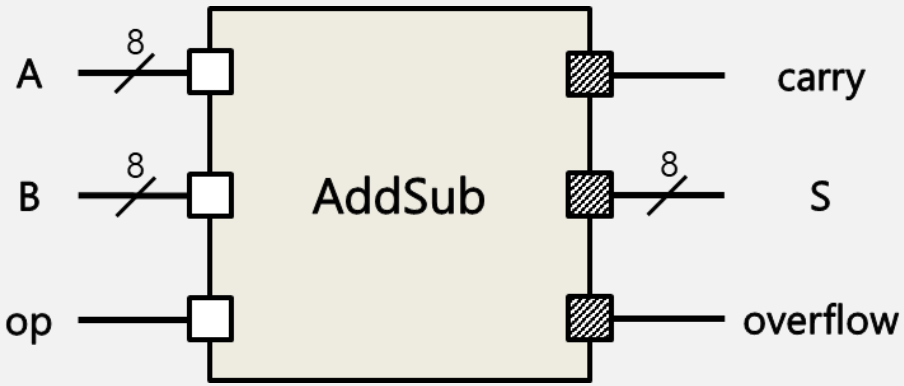


실습수업 일정 및 HW2/HW3 일정

- 일정
  - 5/10 : HW2 설명 + 질의응답
  - 5/17 : HW2 Late Submission 마감
  - 5/24 : HW2 해설 + HW3 사전 설명
  - 5/31 : 4장 (pipeline) 연습문제 풀이
  - 6/14 : 5장 (cache) 연습문제 풀이
  - 6/21 : HW3 제출 마감

- 연습문제 풀이 관련
  - 연습문제는 교재 연습문제 중 풀어볼만한 문제를 미리 알려주고, 실습시간에 풀어줄 예정
  - 기말고사는 연습문제를 많이 참고하여 출제하실 예정이라고 함

HW2: 8-bit Addition/Subtraction Module Interface



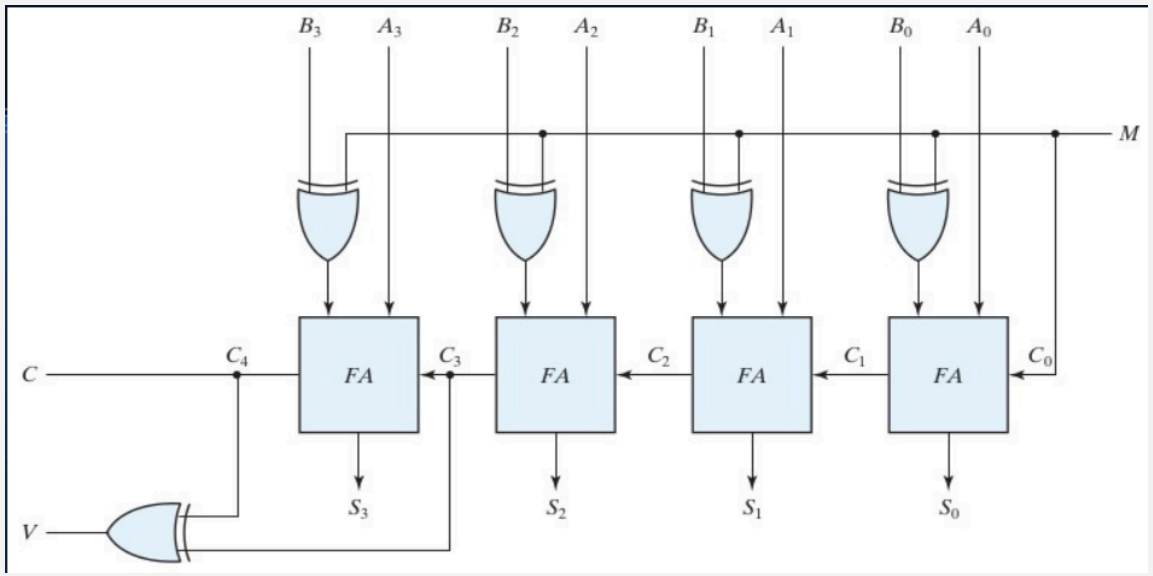
- module AddSub(A, B, op, S, overflow, carry);
  - ✓ A, B : 8-bit input, signed number (in 2's complement, MSB first)
  - ✓ op : 1-bit input, 0 (Addition) / 1 (Subtraction)
    - ◆ op=0 (addition) : $A + B$ 값을 구하는 연산을 수행
    - ◆ op=1 (subtraction) : $A - B$ 값을 구하는 연산을 수행
  - ✓ carry : 1-bit output
  - ✓ S : 8-bit output, signed number (in 2's complement, MSB first)
  - ✓ overflow : 1-bit output
    - ◆ overflow=0 : No overflow. $S = A + B$ or $S = A - B$
    - ◆ overflow=1 : Overflow. $S \neq A + B$ and $S \neq A - B$
  - ✓ 모든 입출력 포트는 MSB first
    - ◆ (ex) input [7:0] A;

Adder can do subtraction

- with 2's complement & little modification on the adder
  - We will see all subtraction as addition :

$$A - B = A + (2's\ complement\ of\ B)$$

  - If 8-bit adder, $\boldsymbol{overflow = C7 \oplus C6}$



https://electronics.stackexchange.com/questions/267966/designing-a-4-bit-adder-subtractor-circuit

Signed number representation - 2's Complement

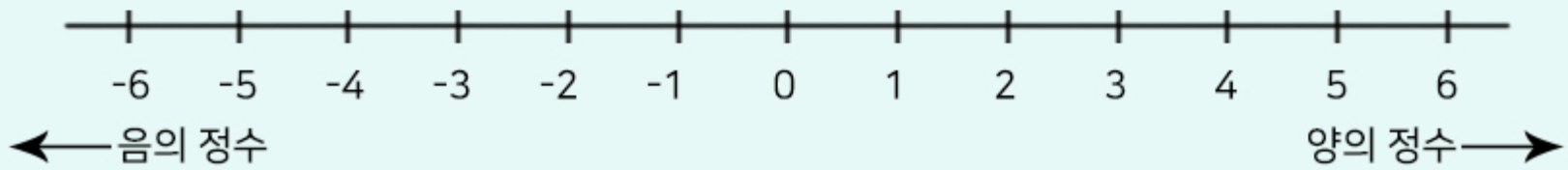
보수

① 양수에 대한 음의 값

② 컴퓨터가 사칙연산을 수행할 때 뺄셈을 덧셈방식으로 변환하여 산출

③ 예) '5-3=2' 를 '5+(-3)=2'로 계산

→ 이때 '-3'이 '3'의 보수, 즉 '음의 정수'가 됨



음의 정수 표현방법

① 부호화 절대치(또는 부호화 절댓값) : 가장 왼쪽 비트가 부호를 의미하며, 그것을 '부호비트'라고 함

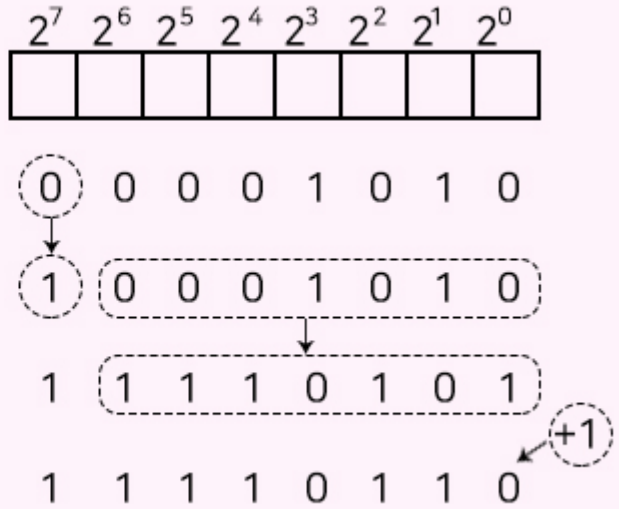
② 1의 보수 : 양수의 반전

③ 2의 보수 : 1의 보수에 1을 더한 값

http://blog.naver.com/javrin4/220981476412

Signed number representation - 2's Complement

'음의 정수' 구하기



-14를 부호화된 2의 보수로 표현하면?

① 14를 2진수로 바꾸어 8비트로 표현 ----------14를 2진수로 전환하면 00001110

② 양수를 음수로 바꾸기 위해 부호비트 전환--------가장 왼쪽의 부호비트 전환하면 10001110

③ 양수와 음수를 전환하여 1의 보수 산출-------부호비트는 유지하고 나머지값 전환 11110001

④ 1의 보수에 +1을 하여 2의 보수 산출----------1의 보수에 +1을 하면 11110010

http://blog.naver.com/javrin4/220981476412

C99 Standard - Data Type Ranges

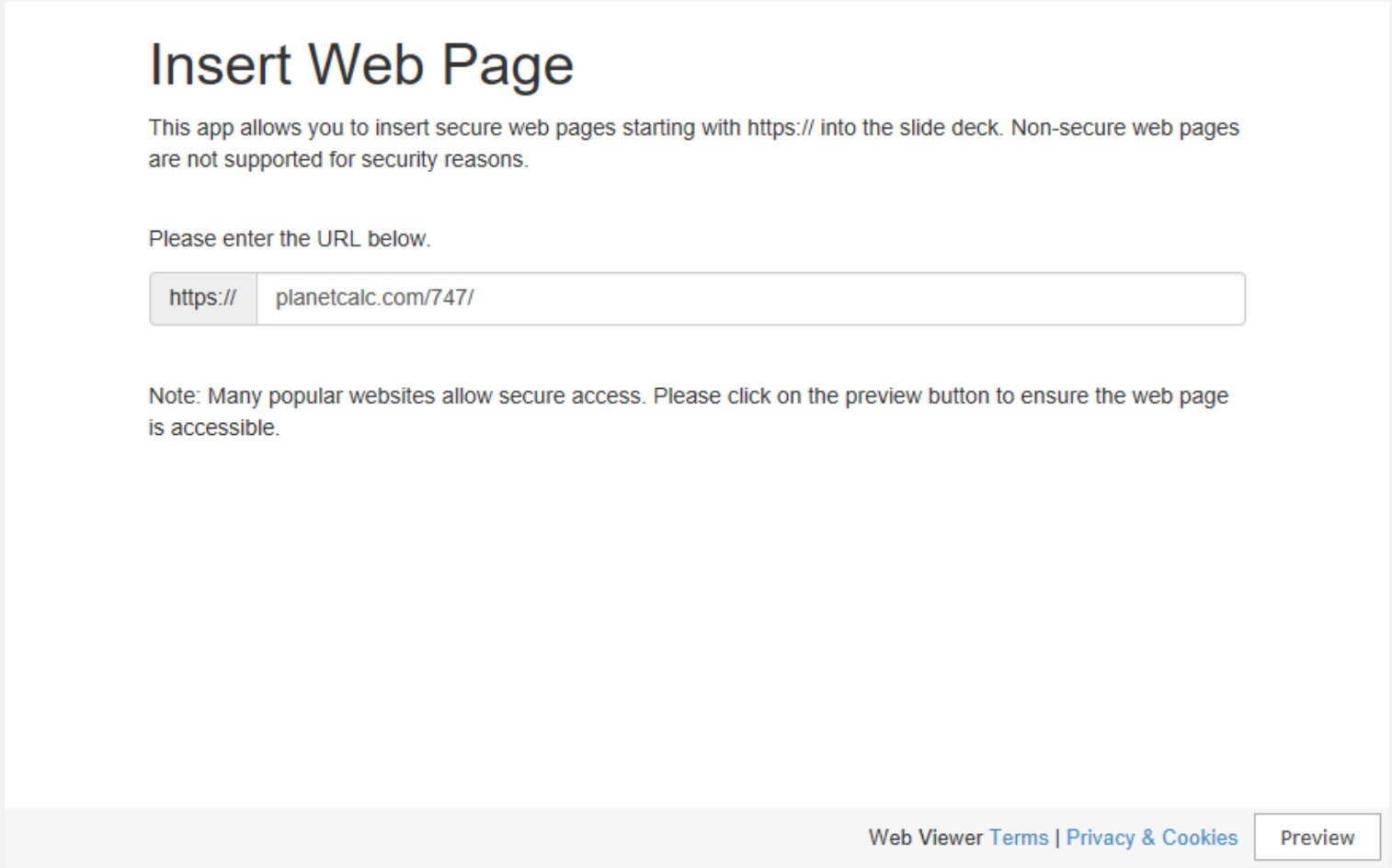
- MSDN definition of <stdint>

```
// These macros must exactly match those in the Windows SDK's intsafe.h.
#define INT8_MIN         (-127i8 - 1)
#define INT16_MIN        (-32767i16 - 1)
#define INT32_MIN        (-2147483647i32 - 1)
#define INT64_MIN        (-9223372036854775807i64 - 1)
#define INT8_MAX         127i8
#define INT16_MAX        32767i16
#define INT32_MAX        2147483647i32
#define INT64_MAX        9223372036854775807i64
#define UINT8_MAX        0xffui8
#define UINT16_MAX       0xffffui16
#define UINT32_MAX       0xffffffffui32
#define UINT64_MAX       0xffffffffffffffffui64
```

8-bit 2's complement		
Binary value	**2's complement interpretation**	**Unsigned interpretation**
00000000	0	0
00000001	1	1
⋮	⋮	⋮
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
⋮	⋮	⋮
11111110	−2	254
11111111	−1	255

Signed number representation - 2's Complement

- https://planetcalc.com/747



8-bit 2's complement		
Binary value	**2's complement interpretation**	**Unsigned interpretation**
00000000	0	0
00000001	1	1
⋮	⋮	⋮
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
⋮	⋮	⋮
11111110	−2	254
11111111	−1	255

Arithmetic Operations with Signed Numbers

- Using the signed number notation with negative numbers in 2's complement form simplifies addition and subtraction of signed numbers.
- Rules for **addition**: Add the two signed numbers. Discard any final carries. The result is in signed form.
- Examples:

00011110 = +30	00001110 = +14	11111111 = –1
00001111 = +15	11101111 = –17	11111000 = –8
00101101 = +45	11111101 = –3	11110111 = –9

Discard carry

Arithmetic Operations with Signed Numbers

- Rules for **subtraction**: 2's complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form.
- Examples:

00011110 (+30)	00001110 (+14)	11111111 (–1)
– 00001111 –(+15)	– 11101111 –(–17)	– 11111000 –(–8)

2's complement subtrahend and add:

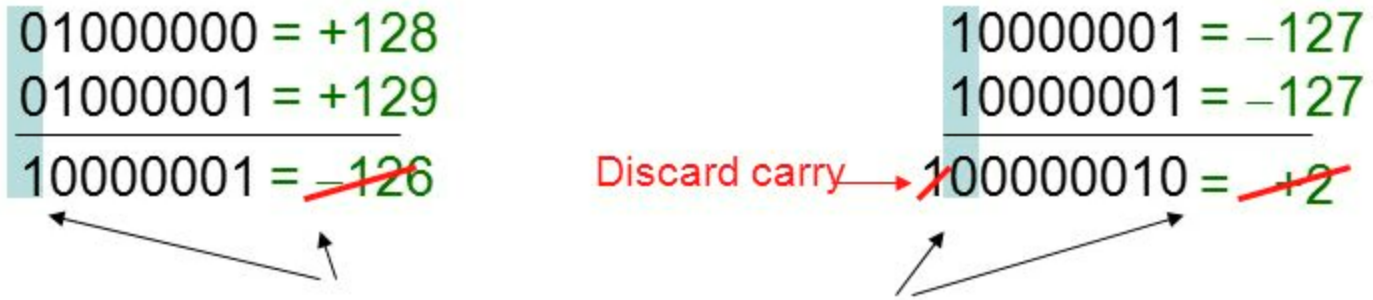
00011110 = +30	00001110 = +14	11111111 = –1
11110001 = –15	00010001 = +17	00001000 = +8
~~1~~ 00001111 = +15	00011111 = +31	~~1~~ 00000111 = +7
Discard carry		Discard carry

Carry and Overflow

- Carry is important when …
  - Adding or subtracting unsigned integers
  - Indicates that the unsigned sum is out of range
  - Either < 0 or > maximum unsigned *n*-bit value
- Overflow is important when …
  - Adding or subtracting signed integers
  - Indicates that the signed sum is out of range
- Overflow occurs when
  - Adding two positive numbers and the sum is negative
  - Adding two negative numbers and the sum is positive
  - Can happen because of the fixed number of sum bits

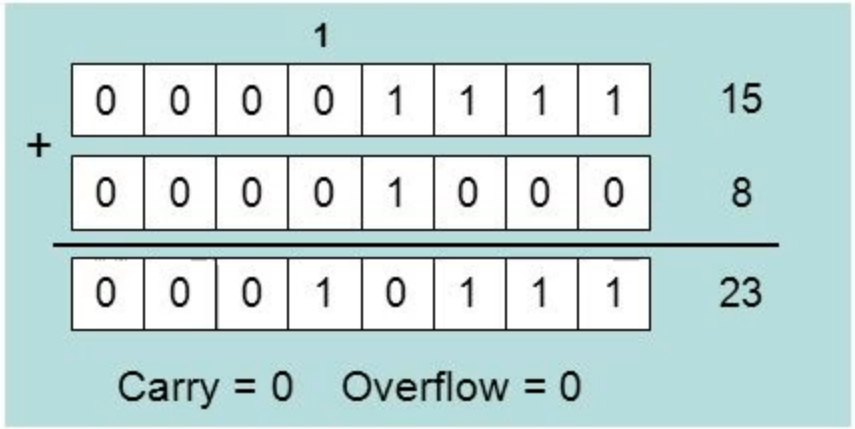
Overflow

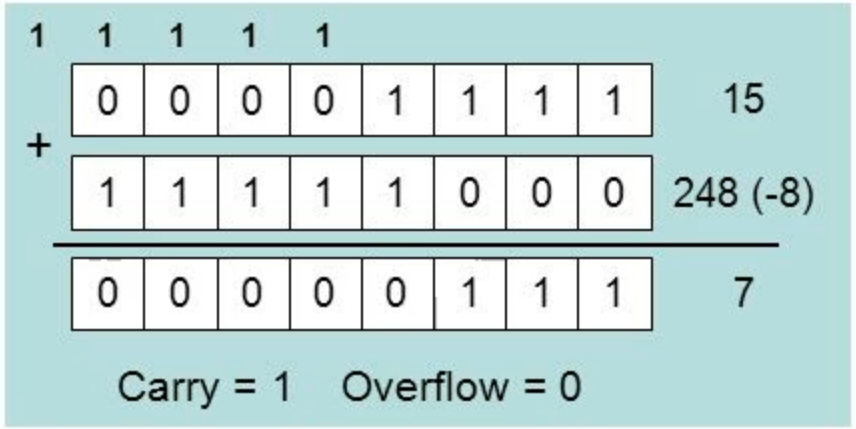
- If the number of bits required for the answer is exceeded, overflow will occur.
- An overflow can occur only when both numbers are positive or both numbers are negative.
- The overflow will be indicated by an incorrect sign bit.

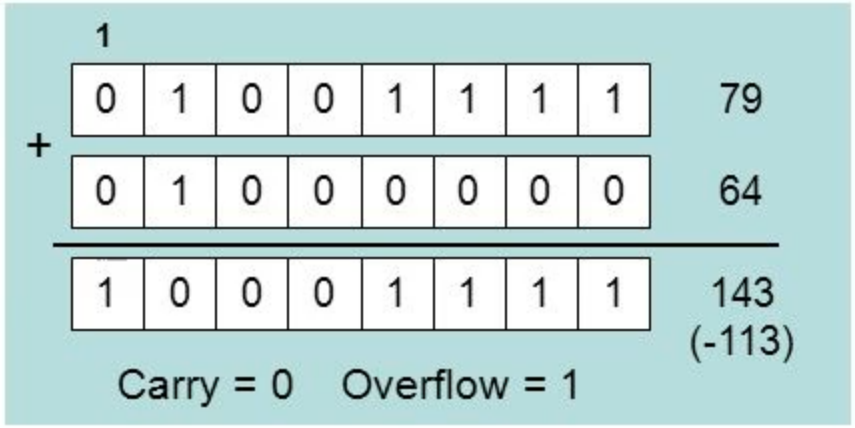


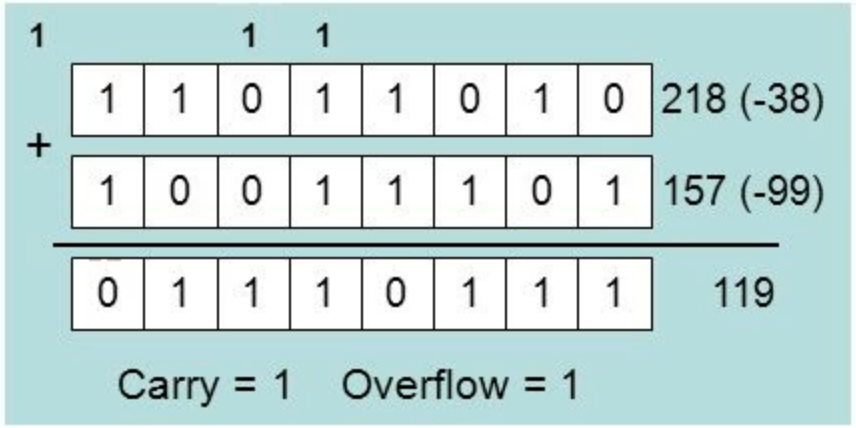
Carry and Overflow Examples

- We can have carry without overflow and vice-versa
- Four cases are possible (Examples are 8-bit numbers)









8-bit Addition/Subtraction Module

- Ref) http://slideplayer.com/slide/8076769

Inputs			Output			Decimal
A	B	op	carry	S	overflow	
00001100	00001100	0	0	00011000	0	12+12 = 24
00001100	00001100	1	1	00000000	0	12-12 = 0
00001101	00011001	1	0	11110100	0	13-25 = (-12)
01100100	00110010	0	0	10010110	1	100+50 = 150
10110000	00111100	1	1	01110100	1	(-80)-60 = (-140)