# Microprocessor

# Calculator

10조
60091998 방윤환
60092115 정해석
60112067 정상호

December 2012

# 목차

- **숫자입력 및 결과값 출력 자릿수를 정수 12자리, 소수 8자리로 늘린다.**
  - Example 계산기의 결과값 출력 범위가 300*300 을 넘지 못한다.
  - 명지 공학도로써 소수점이 없는 계산기는 용납이 안됨.

- **실제 계산기와 같은 디자인을 구현한다.**
  - 딱딱하게 글씨만 나오던 계산기에서 크리스마스를 대비하여 디자인한 깔끔한 계산기

- **EMU8086 Example 계산기에 추가 기능을 구현한다.**
  - 결과값에 추가 연산 기능
  - 계산기 초기화 기능
  - 종료 기능

# 2. 기존 계산기와의 차이점

## ❑ 차이점

- 숫자 범위의 제한이 없다.
  ( 제작한 계산기는 시연하기 위해 정수 12자리, 소수 8자리로 정함
  실제로 숫자범위를 더 늘릴 수 있음 )

- 연산한 결과값을 기억할 필요 없이 결과값에 이어서 계산 할 수 있다.
  ( 연산된 결과값에 재계산 기능 )

- 연산이 끝난 후 프로그램을 종료 할 필요 없이 초기화 하여 다시 계산 가능.

- 한번의 Register 계산이 아닌 한 자릿수 당, 한 Byte 아스키코드 계산.

- 수의 범위가 2의 제곱형태가 아니라 -9 ~ +9.
  ( 제작한 계산기의 범위 : -999999999999.99999999 ~ +999999999999.99999999 )

정상호

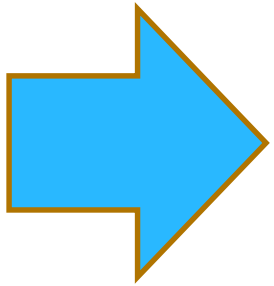
계산기 디자인, 모든 출력부분, 뺄셈

정해석

계산기 추가기능, 연산자입력, 덧셈

방윤환

사용자로부터 숫자입력, 곱셈, 나눗셈



깔끔한 디자인
정수 12, 소수 8 자리
사 칙 연 산
초기화 기능
연속 계산가능
종료 기능

## ❑ 계산기 추가 기능

- **종료 기능**
- **계산기 초기화**
  -> 계산에서 사용한 모든 변수 및 Register를 처음으로 초기화
- **결과값 이어서 계산**
  -> Result 값을 전부 Number1 으로 옮겨준 후 Number2는 초기화,
  연산자 입력 받는 부분으로 JMP

## ❑ 연산자 입력 부분

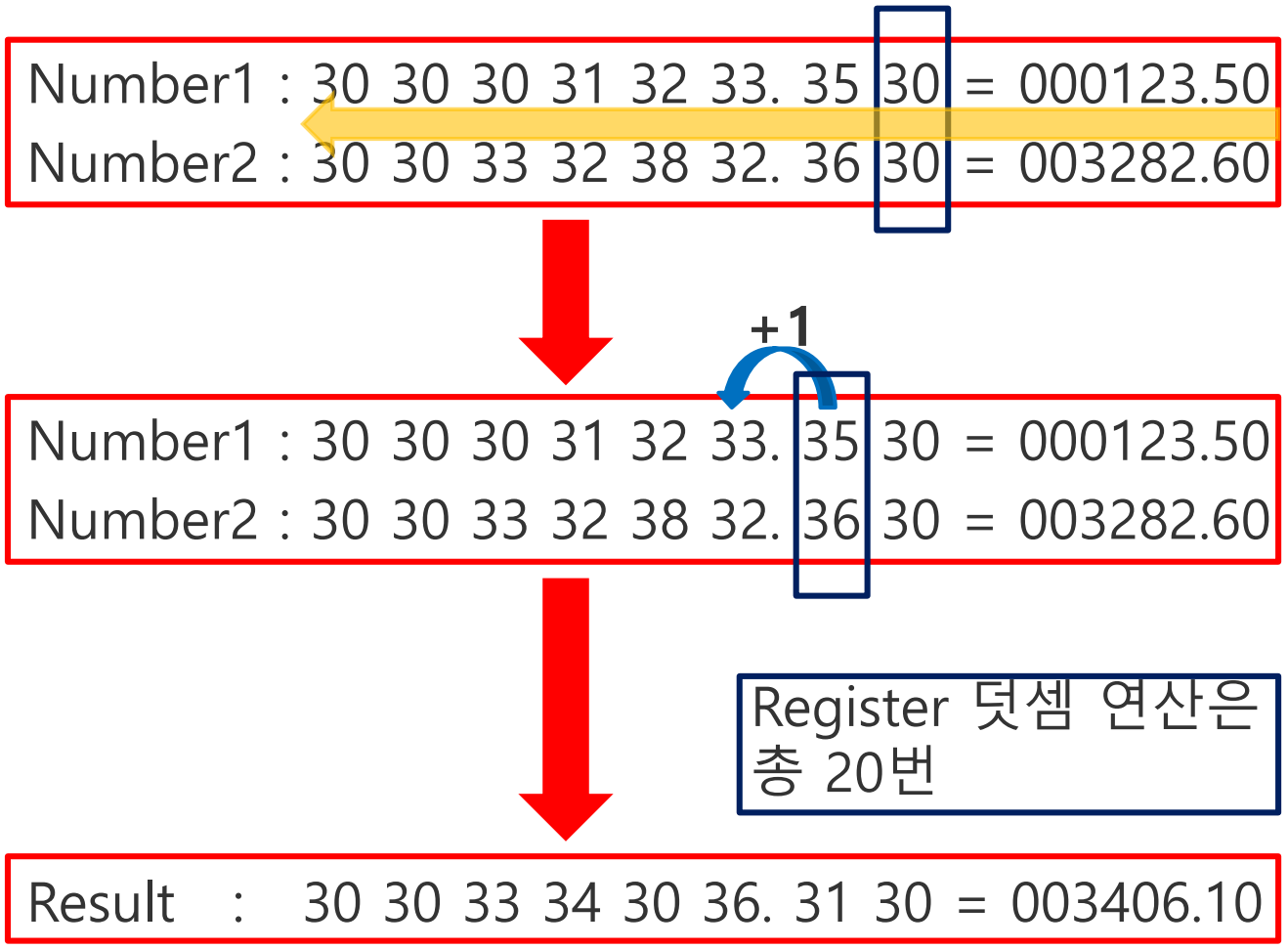
- **INT 21H를 이용하여 사용자로 부터 +, -, *, / 를 입력 받는다.**
  -> **+** : 1, **-** : 2, ***** : 3, **/** : 4

## ❑ 덧셈

- 4가지 경우로 분할

1. X + Y = X + Y → 일반 덧셈 계산
2. X + ( -Y ) = X – Y → 뺄셈으로 넘겨준다
3. -X + Y = Y – X → Number1과 Number2를 바꾸어 준다 → 뺄셈으로 넘겨준다
4. - X + ( -Y ) = - ( X + Y ) → 출력할 때 '-' 부호를 붙여주고 일반 덧셈 계산

## ❑ 덧셈




Number1과 Number2 의 끝자리부터 한자리씩 덧셈 시작 (AAA 명령어)

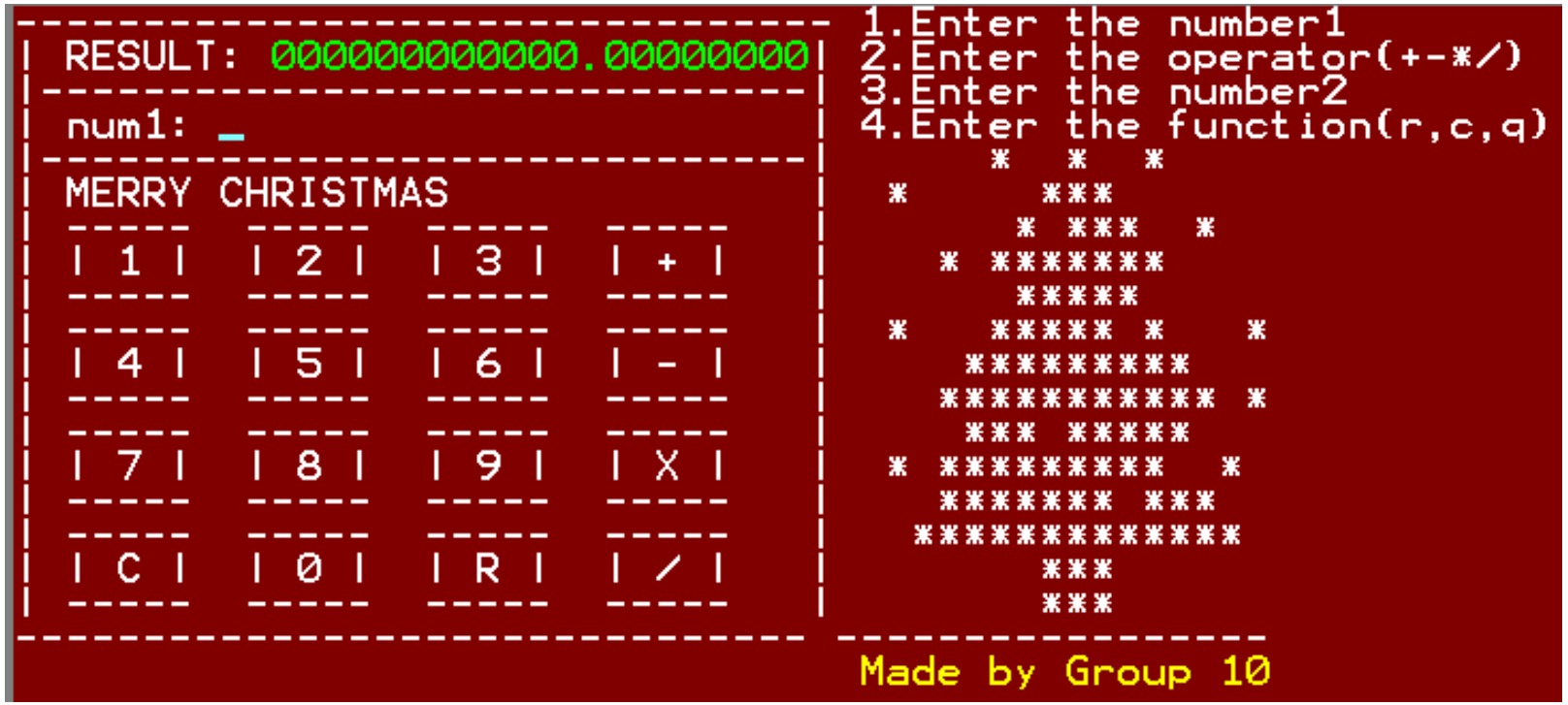
Carry 발생시 다음 자릿수에 +1 을 해준다.
(손 계산과 동일)

Register 덧셈 연산은 총 20번

이와 같은 방법을 반복하여 결과값을 얻는다.

명지대학교 MYONGJI UNIVERSITY

## ❑ 디자인




- ❑ INT 21H 를 이용한 문자 및 문자열 출력
- ❑ INT 10H 를 이용한 바탕색(빨강), 글씨 색(흰색, 노랑색 ,초록색) 변경
- ❑ INT 10H 를 이용한 커서이동

## ❑ 뺄셈

Number1 : 30 30 32 35 31 32. 30 30 = 002512.00
Number2 : 30 30 33 32 32 35. 30 30 = 003225.00

Number1과 Number2 의 크기를 Offset address 부터 비교

Number2 : 30 30 33 32 32 35. 30 30 = 003225.00
Number1 : 30 30 32 35 31 32. 30 30 = 002512.00

Number2가 Number1 보다 큼으로 Number2 에서 Number1 을 빼준다 (AAS 명령어 사용)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Carry : | 0 | | 0 | | 1 | | | | 0 | | | | | | |
| | | | | | 30 | 30 | 33 | | 32 | | | | | | |
| - | | | | | 30 | 30 | 32+1 | | 35 | | | | | | |
| **연산결과 :** | **0** | | **0** | | **0** | | | | **7** | | | | | | |

뺄셈 도중 캐리 발생시 발생한 캐리를 Number1의 다음 자릿수에 더해준 후 뺄셈을 한다

Register 뺄셈 연산은 총 20번

Result : 30 30 30 37 31 33. 30 30 = 000713.00

연산된 결과값을 출력할 때 '-' 만 붙여서 출력

## ❑ 숫자 입력 기능

```
MOV AH, 01
INT   21H
```

→ 사용자로부터 1 Char 입력 ( ASCII )

### ▪ 정수 입력

Scan_number(12 byte) : 31h 30h 30h 30h ..... 30h

SI + 1

Scan_number(12 byte) : 31h 32h 30h 30h ..... 30h

SI + 1

Scan_number(12 byte) : 31h 32h 33h 30h ..... 30h

사용자로부터
1 2 3 을 순서대로
입력 받았을 경우

- 소수 입력

Scan_dot_num(8 byte) : 31h 30h 30h 30h ..... 30h

SI + 1

Scan_dot_num(8 byte) : 31h 32h 30h 30h ..... 30h

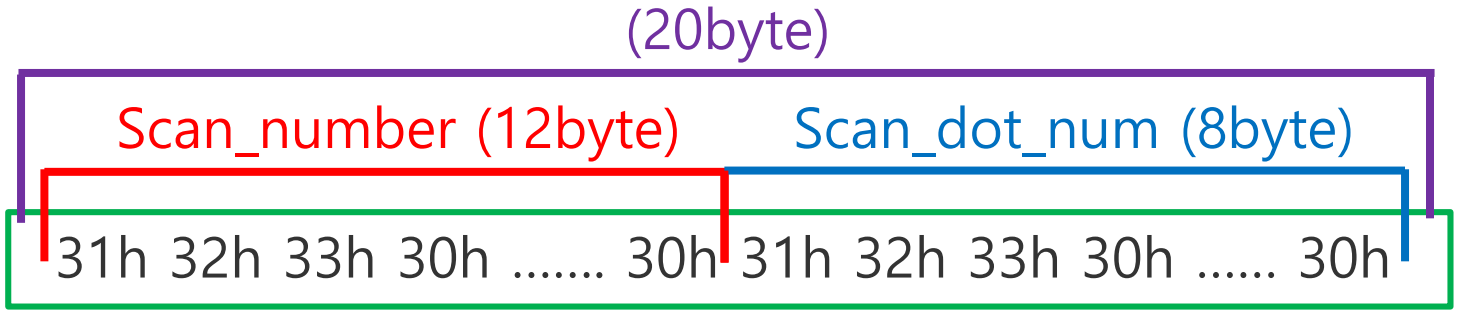
SI + 1

Scan_dot_num(8 byte) : 31h 32h 33h 30h ..... 30h

사용자로부터 '.'을 입력 받은 후

1 2 3 를 순서대로 입력 받았을 경우

- 최종입력



- **최종입력**

Scan_number + Scan_dot_num : 31h 32h 33h 30h ..... 30h . 31h 32h 33h 30h ..... 30h

사용자로부터 Enter를 입력 받았을 경우
정수부분 Shift 한 후 Number1에 저장

**Number1 :** 30h ..... 30h 31h 32h 33h 31h 32h 33h 30h ..... 30h

(20byte)

## ❑ 입력 예외 처리

- 알파벳을 받았을 경우
- '.' 을 두 번 입력 받았을 경우
- Enter를 입력 받았을 경우

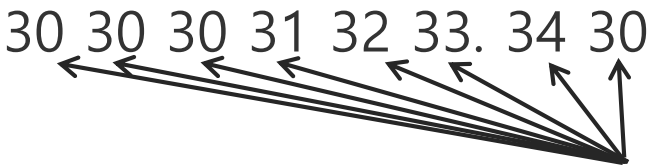
## ❑ 곱셈

- 결과값을 출력하는 소수점 자릿수가 총 8자리 까지 이기 때문에 입력 받은 숫자의 곱셈 연산은 소수점 4자리까지 이다

Number1 : 30 30 30 31 32 33. 34 30 = 000123.40
Number2 : 30 30 30 30 31 32. 33 34 = 000012.34

- 곱셈 연산 과정

- **곱셈 연산 과정**




Number1 * Number2 30 30 30 30 31 32. 33 34

곱셈계산은 손 계산과 동일한 방법으로 AAM 명령어를 이용,

AH = Carry값, AL = 몫 이 담긴다.

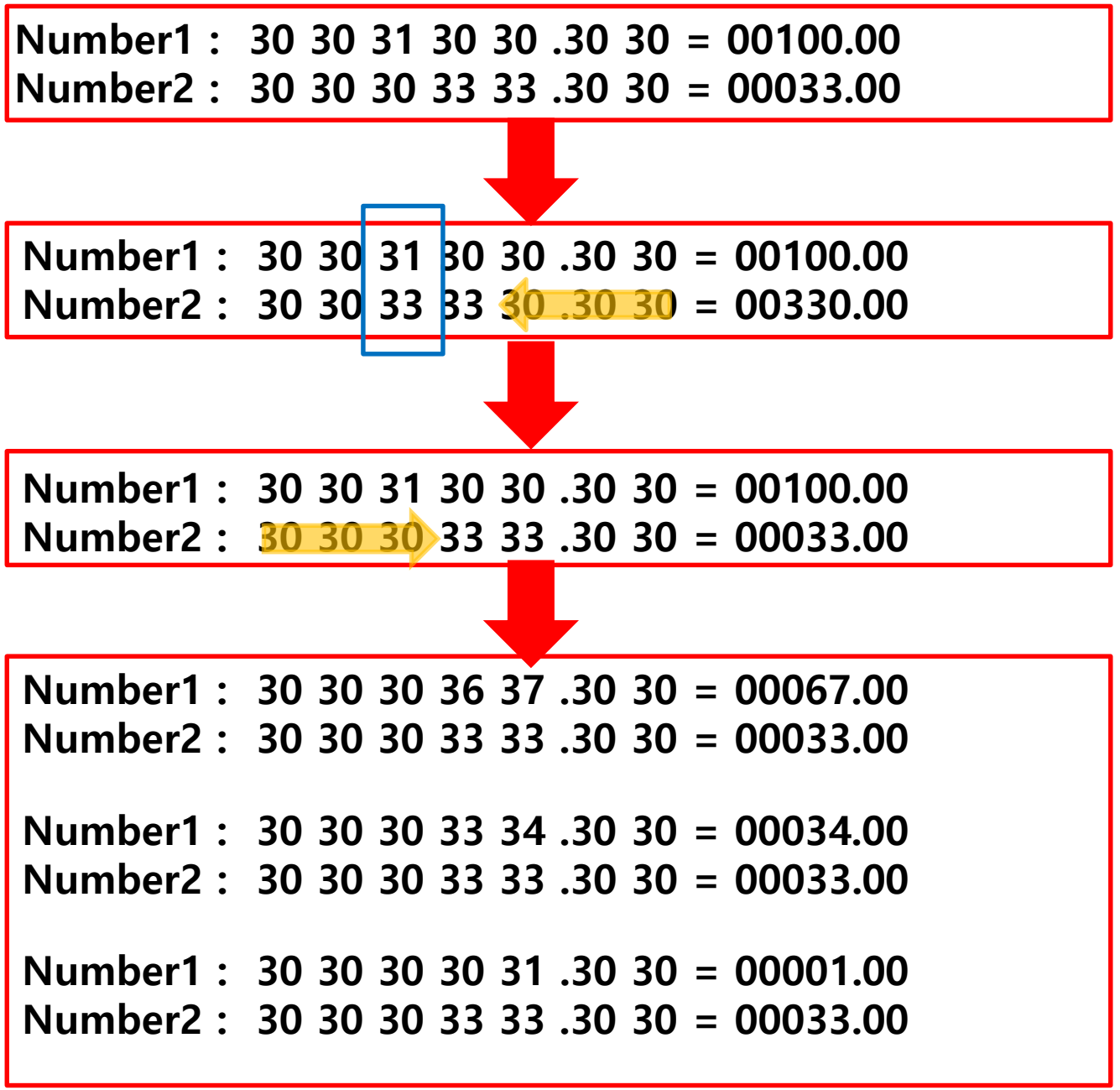
Number2의 자릿수를 올려가면서 연산한 값들을 한자리씩 Shift 해가며 Result에 더한다.

```
+ 00 00 00 04 09 03 06 00
+ 00 00 03 07 00 02 00
+ 00 02 04 06 08 00
+ 01 02 03 04 00
+ 00 00 00 00
+ 00 00 00
+ 00 00
+ 00
```

Result : 01 05 02 02. 07 05 06 34

( Register 곱셈과 덧셈이 총 210번 )

## ❑ 나눗셈



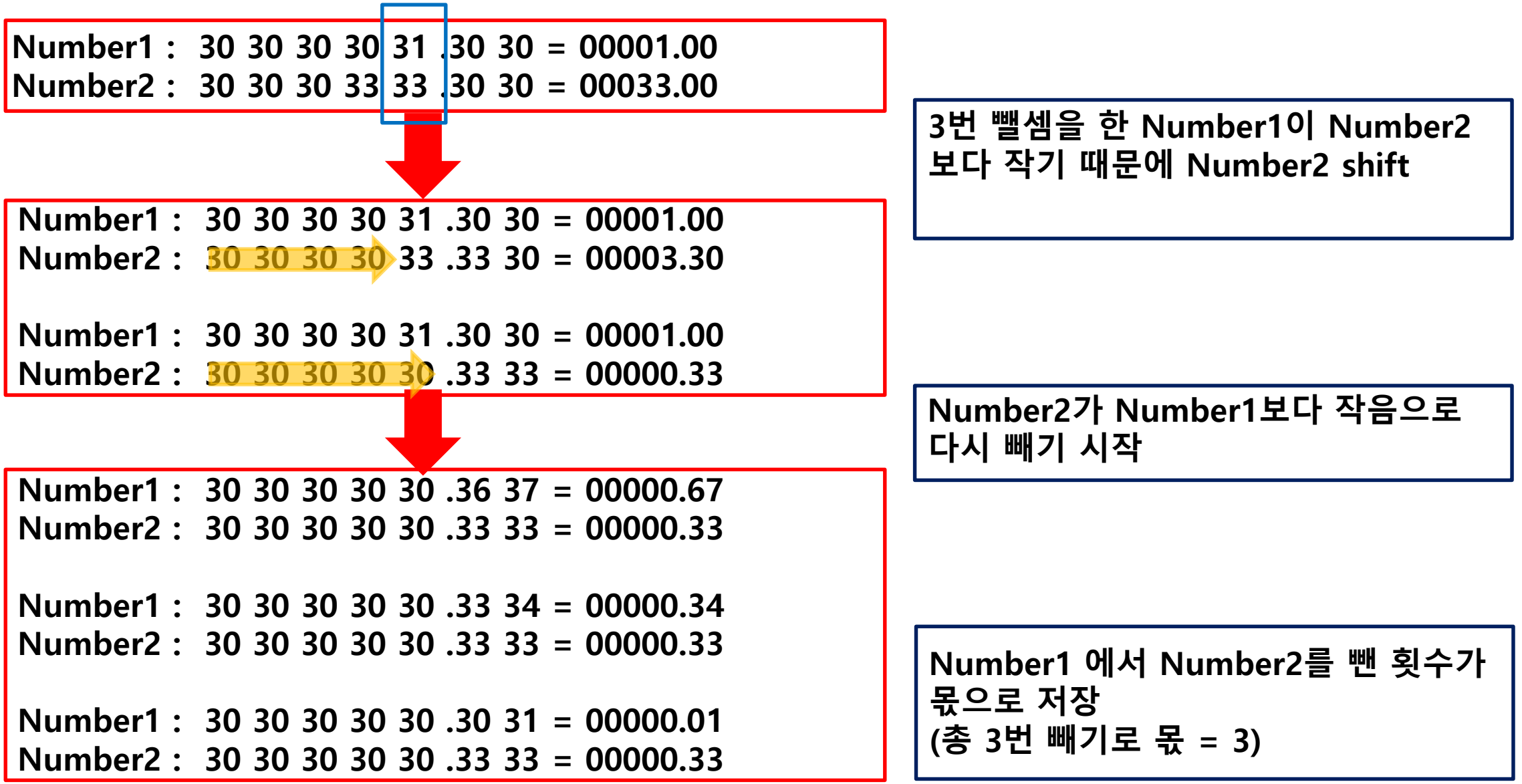

입력 받은 Number1과의 자릿수를 맞춰주기 위하여 Number2를 shift

Number1과 Number2의 크기를 비교한 후 작으면 한자릿수 shift

Number2가 Number1보다 작으면 빼기를 시작, 뺀 값은 Number1에 저장됨

Number1 에서 Number2를 뺀 횟수가 몫으로 저장
(총 3번 빼기로 몫 = 3)

## ❑ 나눗셈




3번 뺄셈을 한 Number1이 Number2 보다 작기 때문에 Number2 shift

Number2가 Number1보다 작음으로 다시 빼기 시작

Number1 에서 Number2를 뺀 횟수가 몫으로 저장
(총 3번 빼기로 몫 = 3)

## ❑ 나눗셈

Number1 : 30 30 30 30 30 .30 31 = 00000.01
Number2 : 30 30 30 30 30 .33 33 = 00000.33

3번 뺄셈을 한 Number1이 Number2 보다 작기 때문에 Number2 shift

Number1 : 30 30 30 30 30 .30 31 = 00000.01
Number2 : 30 30 30 30 30 .30 33 = 00000.03

Number1 : 30 30 30 30 30 .30 31 = 00000.01
Number2 : 30 30 30 30 30 .30 30 = 00000.00

Number2가 Number1보다 작음으로 다시 빼기 시작

Number1 : 30 30 30 30 30 .30 31 = 00000.01
Number2 : 30 30 30 30 30 .30 30 = 00000.00

Shift된 number2의 값이 0임으로 Loop 탈출

result : 30 30 30 30 33 .30 33 = 00003.03

결과값 = 3.03

# 프로그램 시연