

삼성 청년 SW 아카데미

임베디드 C언어

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

Day2-1. 진수 변환

챕터의 포인트

- 진수 변환의 중요성
- 2진수와 16진수
- n진수를 10진수로
- 10진수를 2진수로
- 진수 변환을 C언어로

진수 변환의 중요성

목표

- 임베디드 S/W 개발자로서 진수 변환의 중요성을 이해한다.

진수 변환은 임베디드 장치로부터 나온 Data를 분석하는 과정이다.

- 장치는 2진수 → 16진수로 표현한다.
- 개발자는 16진수 → (2진수) → 10진수로 빠르게 변환해서 이해해야 한다.

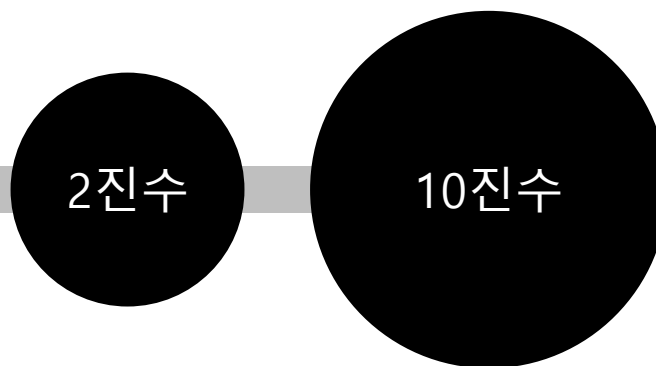
임베디드 장치



타협점



임베디드 개발자



컴퓨터는 0과 1로 구성된 2진수가 Default

- 컴퓨터는 기본적으로 2 진수를 사용한다.
- 사람은 10진수를 사용한다.

그럼 16 진수를 왜 쓸까?

- 사람이 알아보기 힘든 2진수를 10진수로 변환하는데 **계산이 오래 걸린다.**
- 2진수를 보다 **알아보기 쉽게 쓰기 위해** 16진수를 사용한다
- 2진수는 0b00001111 길다
- 16진수는 0x0F 더 짧다

2진수와 16진수

목표

- 2진수, 16진수 표기법을 학습하고 변환한다.

표기 방법

[숫자0] + [b] + [2진수]

- 0b100010110
- 0b1111000111

```
int main()  
{  
    int a = 0b111100;  
  
    return 0;  
}
```

표기 방법

[숫자0] + [x] + [16진수]

- 0x4F10011F

```
int main()
{
    int a = 0x4EFF10;

    return 0;
}
```

10 진수에 10을 곱하면 한 자리수가 더 생긴다.

- $100 * 10 = 1000$

2 진수에 2 를 곱하면 한 자리수가 더 생긴다.

- $0b1101 * 2 = 0b11010$

16 진수에 16을 곱하면 한 자리수가 더 생긴다.

- $0xAB * 16 = 0xAB0$

뒤에서부터

네 자리씩 끊어서 16진수로 즉시 변환 가능

- $0b1111000011110000 = 0xF0F0$
- $0xC78E = 0b1100011110001110$

2진수	10진수	16진수
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	a
1011	11	b
1100	12	c
1101	13	d
1110	14	e
1111	15	f

1. 0xE 를 2진수로 써보기
2. 13을 16진수로 써보기
3. 1101을 16진수로 써보기

1. 0xE 를 2진수로 써보기

- 0b1110

2. 13을 16진수로 써보기

- 0xD

3. 1101을 16진수로 써보기

- 0xD

n진수를 10진수로

목표

- n 진수를 10진수로 변환하는 방법에 대해 학습한다.

1. 적는다.

1011011

2. 1 부터 시작해서, 2의 배수를 모두 적는다.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
2	2	2	2	1	1	1

1011011

=

64	32	16	8	4	2	1
1	0	1	1	0	1	1

1011011

3. 0 인 부분은 지운다.

64 16 8 2 1
1011011

4. 모두 더한다.

- $64 + 16 + 8 + 2 + 1 = 91$

다른 진수를 10진수로 변환하는 방법도 동일하다.

10진수를 2진수로

목표

- 10진수를 2진수로 변환하는 방법에 대해 학습한다.

유명하다

2로 나눠서 나머지 적고

반대로 쓰으~

하지만 이 방식은 느리다!

$$\begin{array}{r} 2 \overline{) 17} \\ 2 \overline{) 8} \dots 1 \\ 2 \overline{) 4} \dots 0 \\ 2 \overline{) 2} \dots 0 \\ \overline{) 1} \dots 0 \end{array}$$

17을 2진수로 바꾸자

1. 우선 17보다 작은 2의 n제곱 수를 순서대로 적는다.

16 8 4 2 1

--	--	--	--	--

2. 17은 16보다는 크니까, 16을 하나 사용한다.

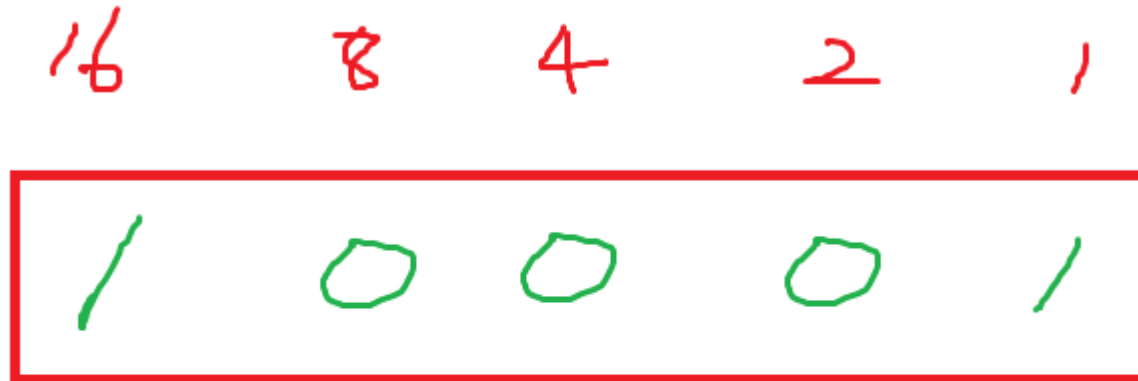
- 남은 숫자는 1 이다.

16 8 4 2 1



$$17 - 16 = 1$$

3. 1 에다 체크하고 나머지는 모두 0으로 적으면 변환 끝!



$$17 - 16 = 1$$

20 를 2 진수로 변환해보기

32 16 8 4 2 1

--	--	--	--	--	--	--

20 를 2 진수로 변환해보기

32 16 8 4 2 1

0 1 0 1 0 0

진수변환을 C언어로

목표

- C언어에서 진수 변환에 도움이 되는 API를 학습한다.
- 계산기를 사용하여 디버깅을 쉽게 한다.

long strtol(char const* str, char** endptr, int base)

- 문자열에서 숫자를 추출하는 함수
- <stdlib.h> 필요
- str : 문자열
- endptr : 문자열 끝 포인터
- base : 변환 될 문자열의 진수 (2, 8, 16, 10 선택)
- return : 성공 → 변환된 정수, 실패 → 0

```
char inputH[10]= "0xFF";  
char inputB[10]= "1010";  
char inputO[10]= "52";  
char inputD[10]= "123";  
  
int result16 = strtol(inputH,NULL,16);  
int result2 = strtol(inputB,NULL,2);  
int result8 = strtol(inputO,NULL,8);  
int result10 = strtol(inputD,NULL,10);
```

<https://gist.github.com/hoconoco/58eb09544b567fff3fa705c34e6e8eba>

int sprintf(char* str, const char* format, ...)

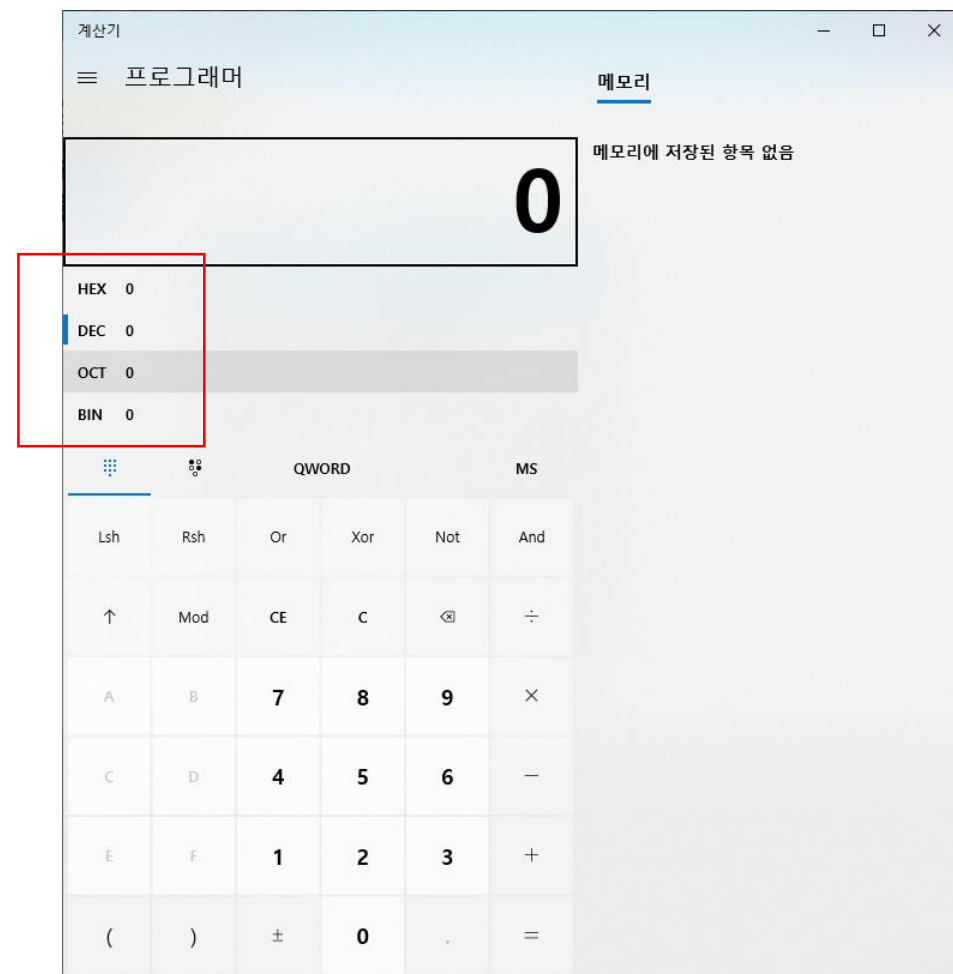
- 문자열에 지정한 형식으로 담는다.
- <stdio.h>
- 문자열 끝에 'W0'이 자동으로 추가된다.
- str : 변환된 문자열을 담을 버퍼
- format : string 문자 서식
- return : 성공 → byte 수, 실패 → -1

<https://gist.github.com/hoconoco/1074c73ca2bd0b777d4b7f11d808b4e4>

```
int main(){  
  
    int a = 100;  
    char buf[10] = {0};  
  
    sprintf(buf, "%X", a);  
    printf("%s", buf);  
  
    return 0;  
}
```

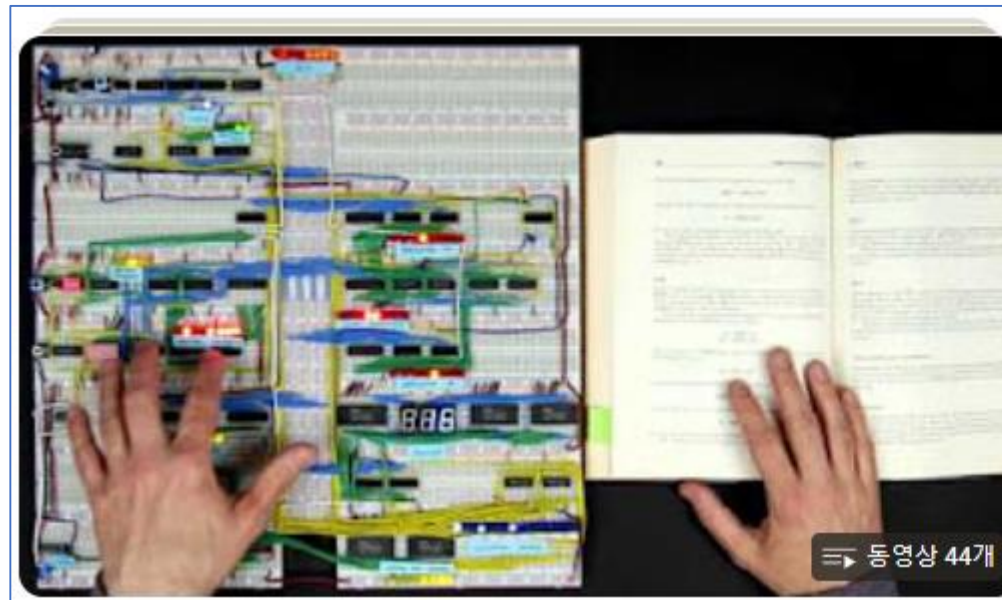

손으로 계산하거나 코드로 짰 값이
맞는 지 확인하기 위해 계산기를 이용한다.

- 윈도우 내장 계산기는 프로그래머용 계산기를 지원한다.
- 원하는 진수를 선택 후 값을 입력하면 변환된다.



8진수를 쓰던 때가 있었다.

- 유X브에 찾아보면 8bit 컴퓨터를 만드는 진짜들이 있다.



8비트 브레드보드 컴퓨터를 만들어 보세요!

벤 이터·재생목록 ✓

8비트 컴퓨터 업데이트 · 6:53

불안정한 555 타이머 · 8비트 컴퓨터 시계 · 1부 · 27:51

모든 재생목록 보기

동영상 44개

<https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTByGqImE405J2565dvjafglHU>

Day2-2. 비트연산의 기본

챕터의 포인트

- 비트 연산의 중요성
- 비트연산자
- 비트연산을 C언어로
- 비트 추출하기
- 비트 set/clear
- 비트 반전

비트 연산의 중요성

목표

- 임베디드 SW 개발자에게 비트연산의 중요성에 대해 학습한다.
- 임베디드 장치에서 비트 연산을 이용해 장치 제어하는 방법에 대해 학습한다.

임베디드 개발자가 비트연산을 공부하는 이유

→ 임베디드 개발자는
비트연산을 사용해서, 장치를 제어하기 때문

MCU

- 하나의 칩 안에
CPU / 메모리 / Disk 까지 모두 들어가 있음

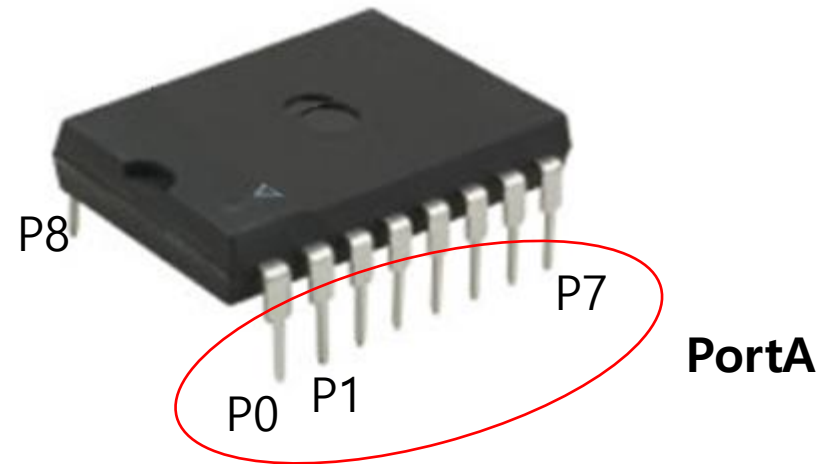
지금부터 설명할 예시는
동작원리를 이해하기 위한
간략한 설명이다.

(정확한 동작은 Firmware시간에 다룬다.)



MCU

- 아래 MCU는 핀(PIN)이 총 16개가 있음
 - **PortA** : P0 ~ P7 을 뜻함
 - PortB : P8 ~ P15 를 뜻함

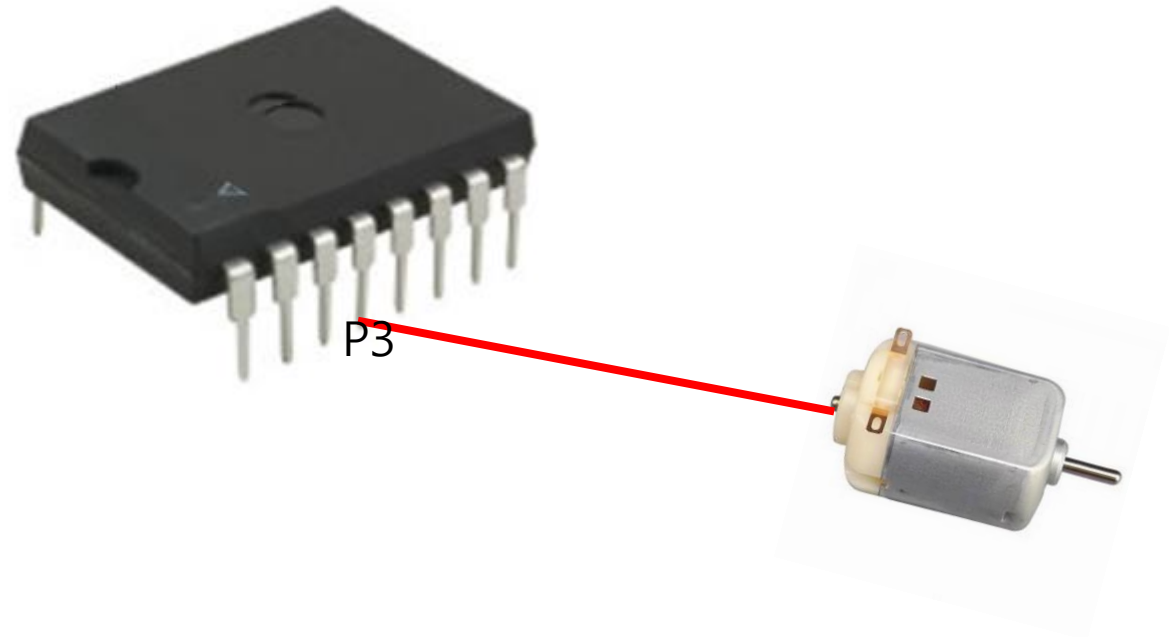


여기에 장치를 연결한다.

Confidential

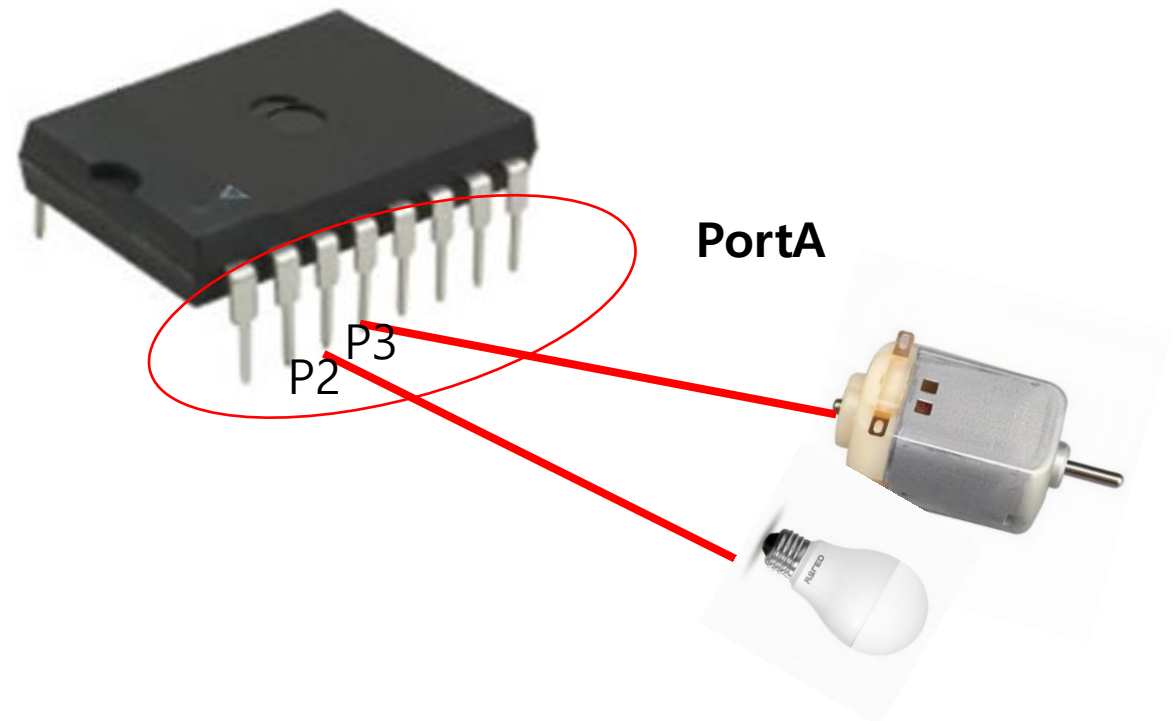
MCU에 여러 장치들을 **전선으로** 연결한다.

- 여러개의 “PIN” 중 하나에다가 장치를 연결한다.



PortA 연결상태

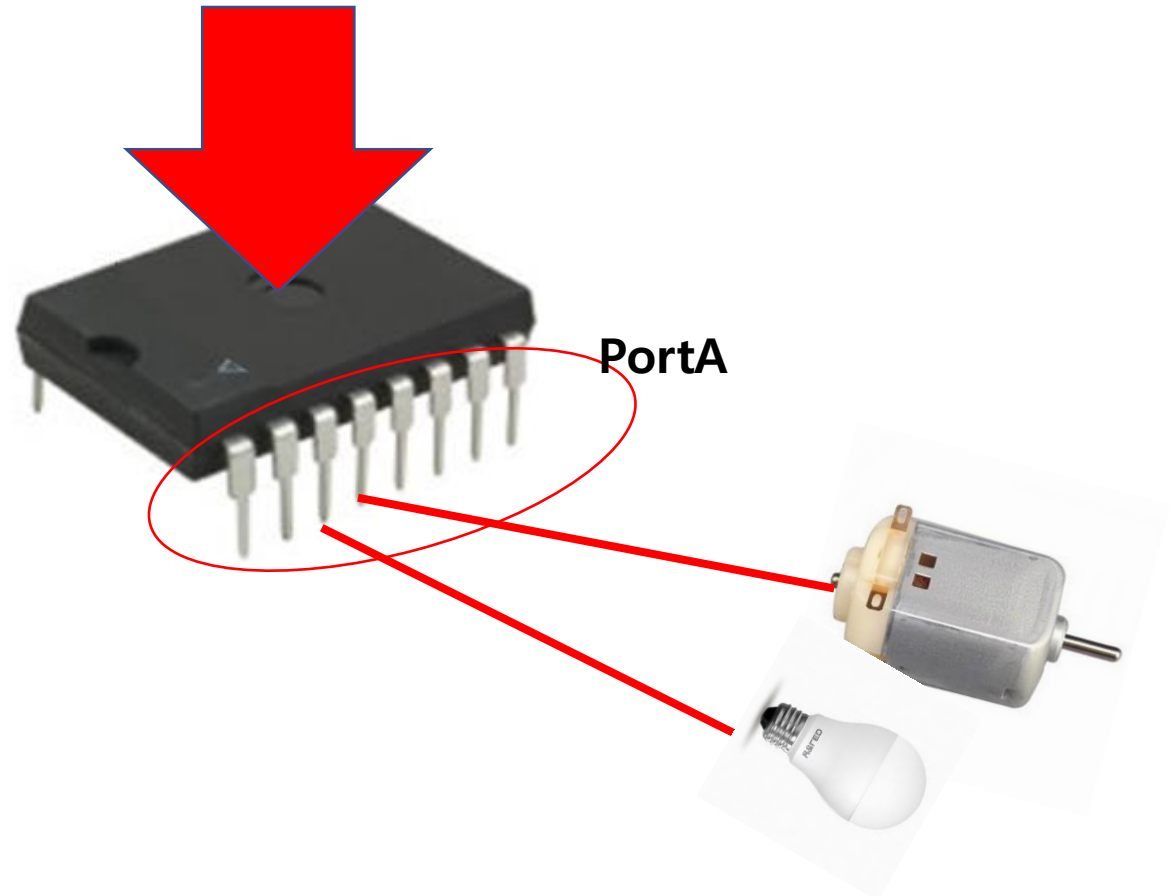
- Pin 3 : 모터 연결
- Pin 2 : 전등 연결



이 칩에서 동작될
작은 운영체제를 만들어야한다.

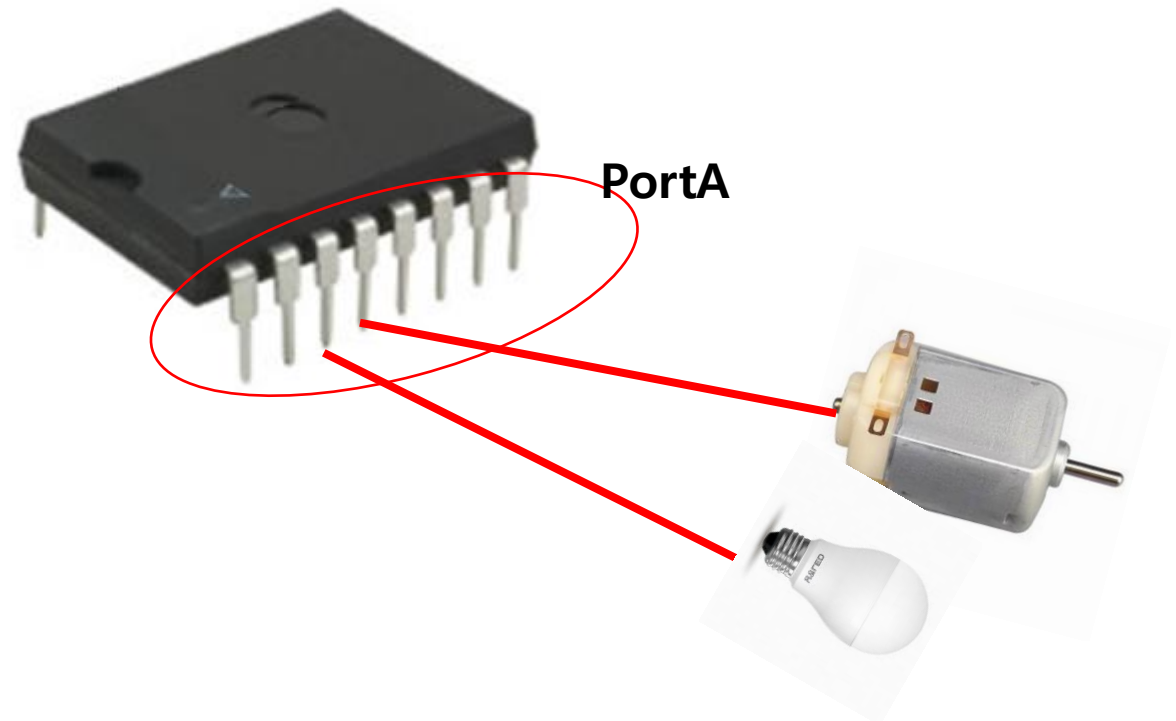
- Firmware : 작은 운영체제

PortA 에
값을 넣어주는
Firmware를 개발하면,
장치를 제어할수있음



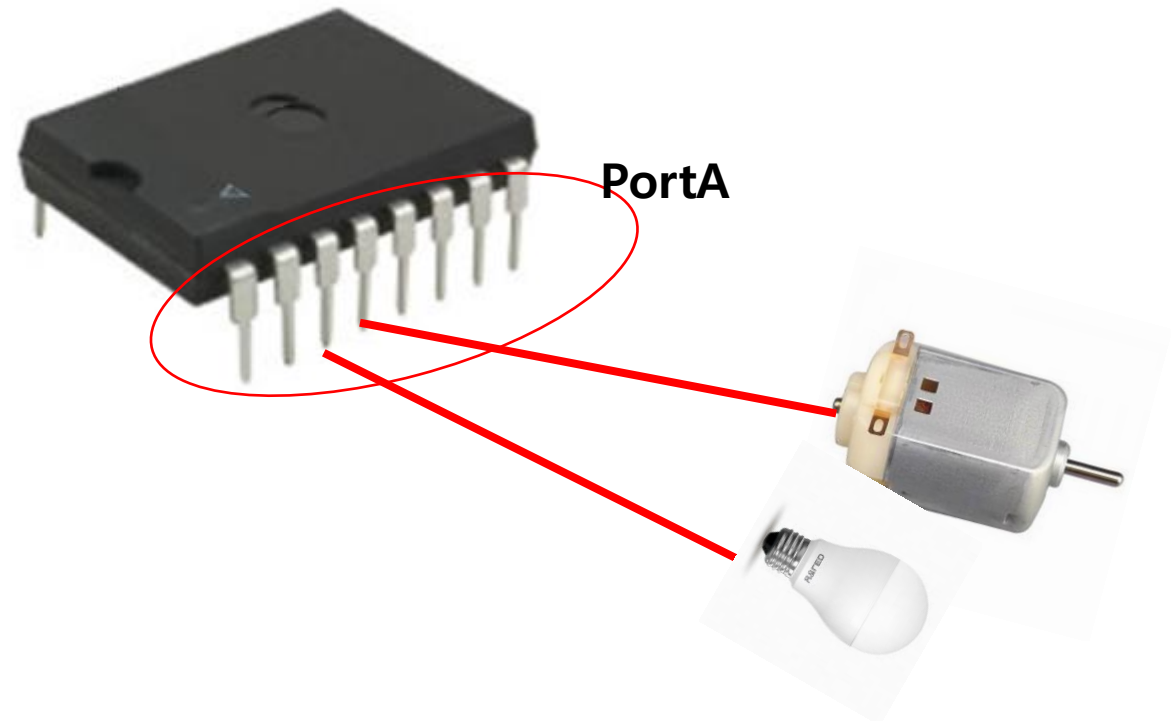
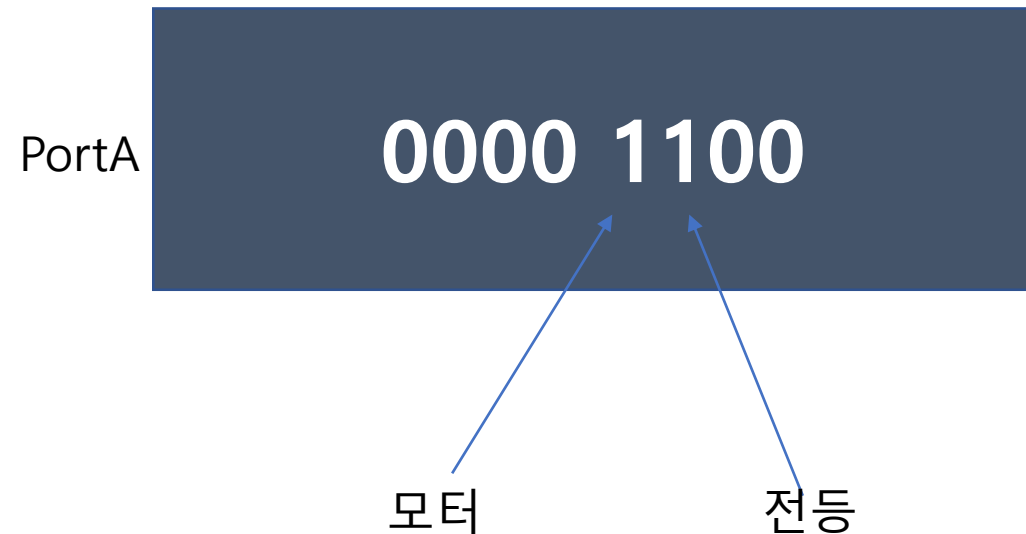
PortA에 적절한 값을 넣으면, 장치가 동작한다.

- PortA = 0000 1000 값을 넣으면 모터가 켜진다. (전등은 꺼진다.)
- PortA = 0000 0100 값을 넣으면 전등이 켜진다. (모터는 꺼진다.)



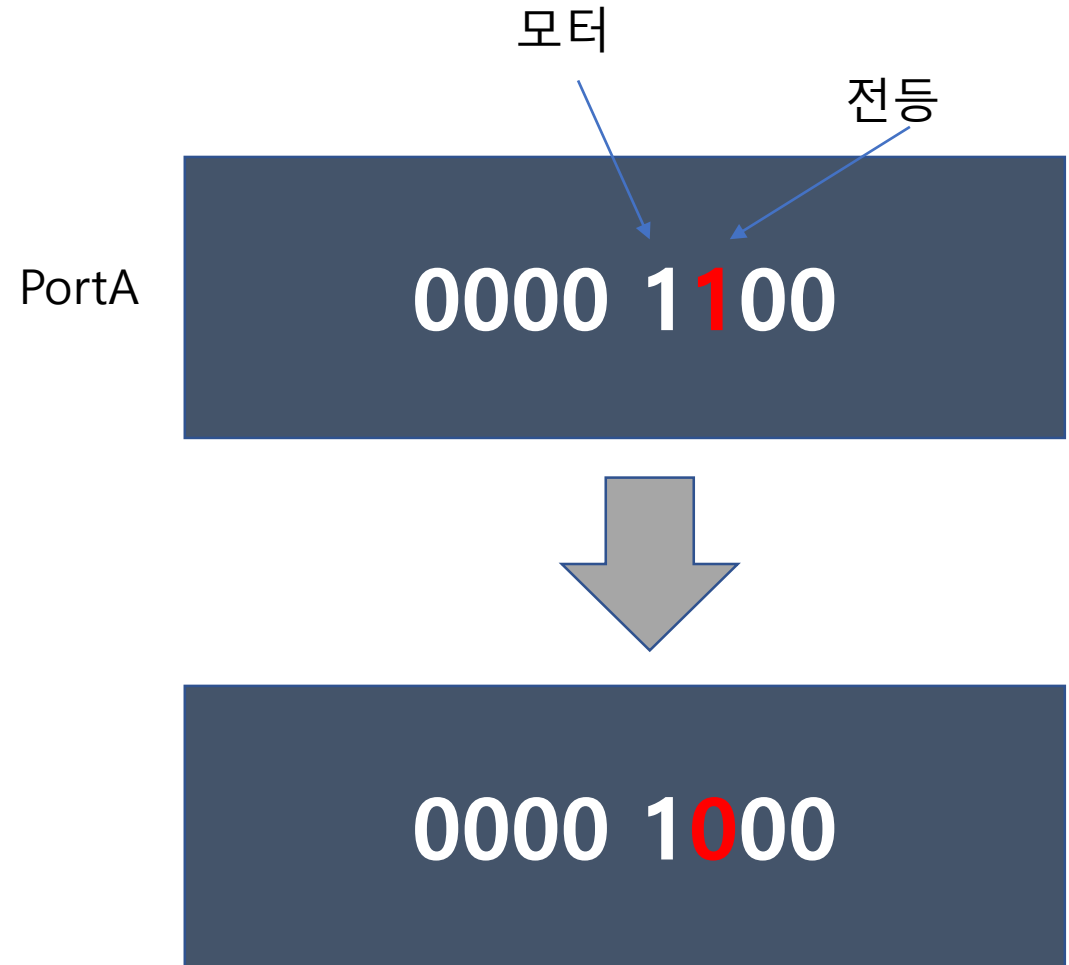
모터 ON, 전등 ON 하려면?

- PortA = 0b00001100
- 위와 같이 코딩하면 된다.



만약 모터는 건드리지 않고,
전등만을 끄고 싶다면?

- $\text{PortA} = \text{PortA} \& \sim(0x1 \ll 2)$
- 위 코드를 수행하면 0000 1000 이 되어,
모터의 값이
어떤 값이던 상관없이
전등만 끌 수 있음



장치를 제어하는 Firmware 를 만들 때
비트연산을 사용해서 장치를 제어하기 때문.

지금 배우는 내용은
리눅스 커널 시간에 진행하는 디바이스 드라이버 개발에도 사용된다.

비트연산자

목표

- 비트 연산에 들어가기 전에 비트와 비트연산자에 대해 학습한다.

bit = 0과 1 을 나타내는 최소 단위

- 1101 = 4bit
- 100111 = 6bit

기본적으로 32bit를 기준으로 이야기하지만,
32bit는 너무 길어서 8bit or 16bit 내에서 연습을 한다.

수를 2진수 8bit로 표현하기

- $0xA = 0b00001010$
- $0xC = 0b00001100$

수를 2진수 10bit로 표현하기

- $0x1E = 0b0000011110$
- $0x33 = 0b0000110011$

1 Byte = 8 bit

- 0번 비트 ~ 7번 비트로 구성
- **MSB** : 7번 Bit
- **LSB** : 0번 Bit

7번 bit
(**MSB**)

1 0 0 1 1 1 1 0

1번 bit 0번 bit
(LSB)

& 연산 (and)

- 값을 추출할 때 사용한다.
- 값에 “1” 을 적은 곳만 추출한다.

$0 \& 0 = 0$
 $0 \& 1 = 0$
 $1 \& 0 = 0$
 $1 \& 1 = 1$

| 연산 (or)

- 2진수 덧셈시에 사용된다.
- 둘 중 한 곳에 “1”이 있으면, 결과는 “1” 이다.

$0 | 0 = 0$
 $0 | 1 = 1$
 $1 | 0 = 1$
 $1 | 1 = 1$

^ 연산 (XOR)

- 청개구리 연산
- 같으면 0, 다르면 1
- 암호화에 사용된다.

$$\begin{array}{l} 0 \wedge 0 = 0 \\ 0 \wedge 1 = 1 \\ 1 \wedge 0 = 1 \\ 1 \wedge 1 = 0 \end{array}$$

Shift 연산자

- 비트 단위로 이동한다.
- Shift 는 2승의 곱 / 나눗셈으로 동작한다.
 - $1 \ll 3 : 1 * (2^3)$ 과 동일함
 - $3 \ll 2 : 3 * (2^2)$ 과 동일함
 - $15 \gg 2 : 15 / (2^2)$ 과 동일함

\ll Left Shift

- 값을 왼쪽으로 밀어서 **숫자를 추가**해준다.

\gg Right Shift

- 값을 오른쪽으로 밀어서 **숫자를 파.괴.한다.**

```
10101 << 2 = 1010100
1110 >> 3 = 1
```


~ 연산 (not)

- 비트가 반대가 된다.
- “기준 비트 수”에 따라 결과 값이 달라질 수 있다.

8 bit에서 ~0xF0 값은?

- ~0xF0 : 1111 0000 → 0000 1111

32 bit에서 ~0xF0 값은?

- 0000 0000 0000 0000 0000 0000 1111 0000 → 1111 1111 1111 1111 1111 1111 0000 1111

비트연산을 C언어로

목표

- 비트 연산을 코드로 작성한다.
- 연산자 우선순위에 주의한다.

비트 연산에서 음수는 배제한다.

- Trace 로 결과를 확인한다.
- 코드로 결과를 보기 전에 계산할 수 있어야 한다.

<https://gist.github.com/hoconoco/9c4fea1cb5eb258b7485ddc2f4259916>

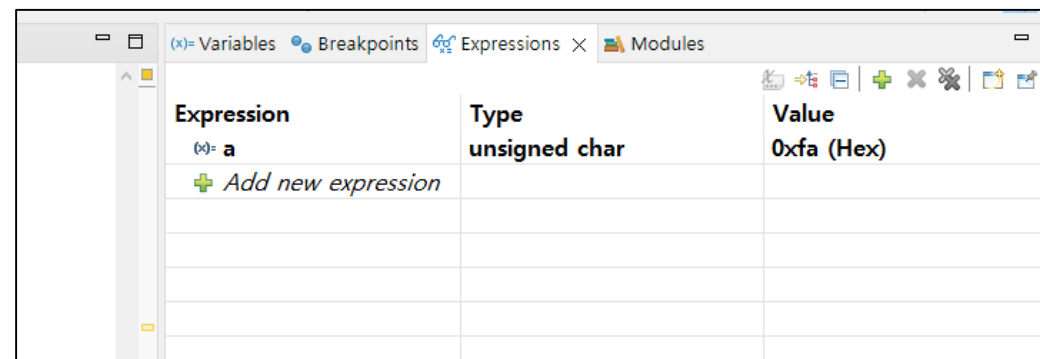
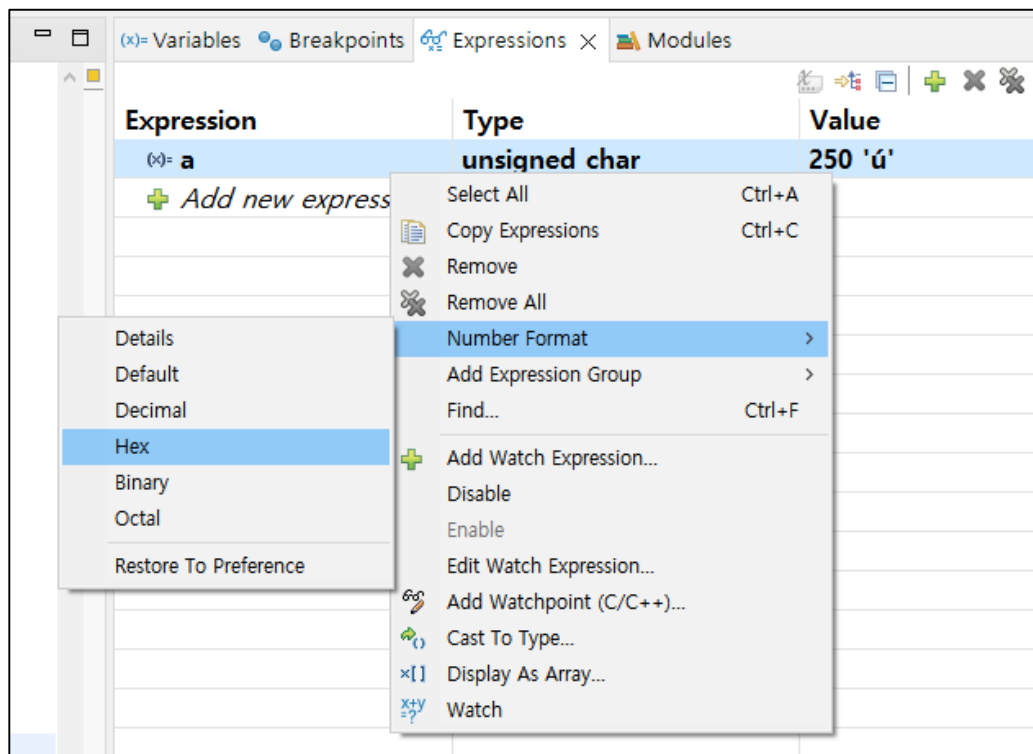
```
int main()
{
    unsigned char a = 0xFA;

    a <<= 1;
    a >>= 1;
    a &= 0x10;
    a |= 0x2E;

    return 0;
}
```

변환할 변수에 마우스 우클릭

- Number Format → 선택



char와 unsigned char의 차이

- char

- MSB를 부호로 사용한다.
- 나머지 7 bit를 수를 저장하는 용도로 사용한다.
- [암기] 저장할 수 있는 수 : -128 ~ 127

- unsigned char

- 부호는 없다.
- 8 bit 모두 수를 저장하는 용도로 사용한다.
- [암기] 0 ~ 255 까지 저장 가능

비트연산을 할 때 괄호를 많이 넣자.

- 우선순위 꼬여 버그가 발생하기 쉽다.
- ex. 마이너스 우선순위 > 쉬프트 우선순위
 - $1 \ll 3 - 1 = 4$
- ex. `==` 우선순위 > OR 우선순위
 - `if (a | 3 == 5) { ... }`

```
int a = 7;
```

```
if (a | 3 == 5) {  
    printf("#");  
}
```

예상치 못한 결과 발생

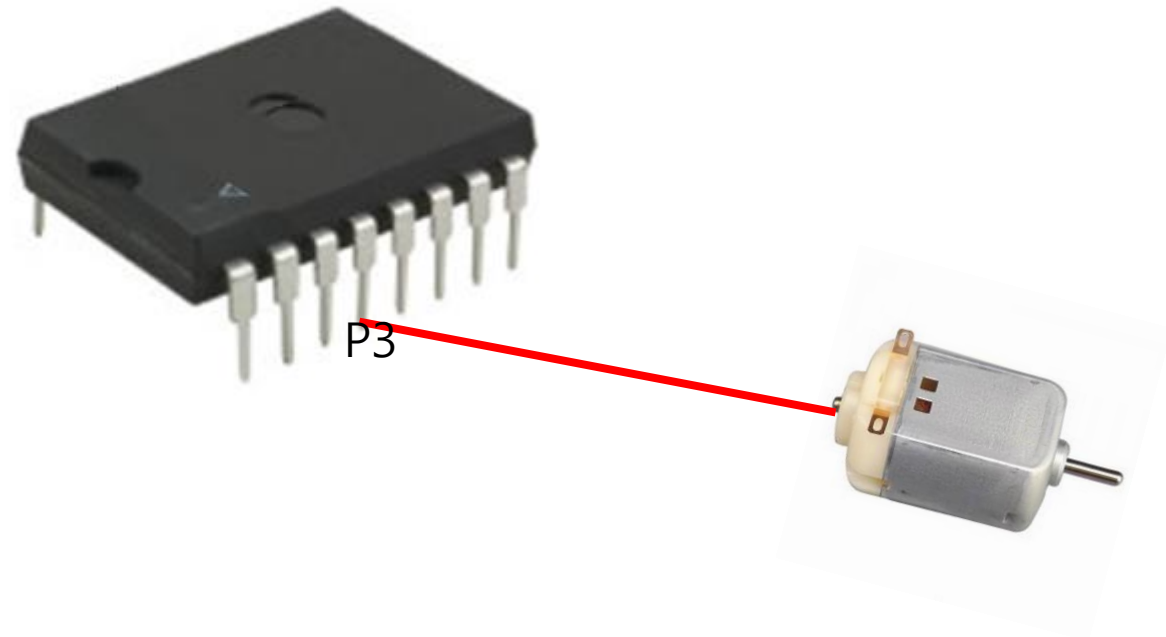
비트 추출하기

목표

- 특정 비트를 추출하는 방법에 대해 학습한다.
- 공식처럼 외우기 보다 왜 그렇게 되는 지 원리를 이해한다.

MCU의 특정 핀에 연결된 장치의 상태를 읽어 올 수 있다.

- 버튼이 눌렸는 지
- 모터가 동작 중인지



2번 bit에서 1개 bit 추출하여 변수 b에 저장하기

- $b = (a \gg 2) \& 0x01;$

이제, 한 단계씩 분석을 해보자.

비트연산 계산방법은
암기하지 않고, 스스로 생각하여
짚수있도록 해야한다.

101**1**000 이런 수가 존재한다.

우리의 목적은 2번 bit를 뽑아내서, 출력하는 것

101**1**00

여기서 1을 뽑아내야 한다.

오른쪽으로 두번 밀자. ($\gg 2$)

→ 101**1**

1011

여기서, 뽑아낼 수를 제외하고 모두 지운다.

0x1 을 & 연산하면 된다.

1011

0001

0001 ← 추출 완료!

결론, 변수 a의 2번 비트 추출해서 b에 저장하기

$a = 101\mathbf{1}00$

$b = (a \gg 2) \& 0x01;$

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b11110101;`

1. 5번 bit에서 1개 bit 추출하여 변수 b에 저장하기

2. 6번 bit에서 2개 bit 추출하여 변수 c에 저장하기

3. 0번 bit에서 3개 bit 추출하여 변수 d에 저장하기

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b11110101;`

1. 5번 bit에서 1개 bit 추출하여 변수 b에 저장하기

- `b = (a >> 5) & 1;`

2. 6번 bit에서 2개 bit 추출하여 변수 c에 저장하기

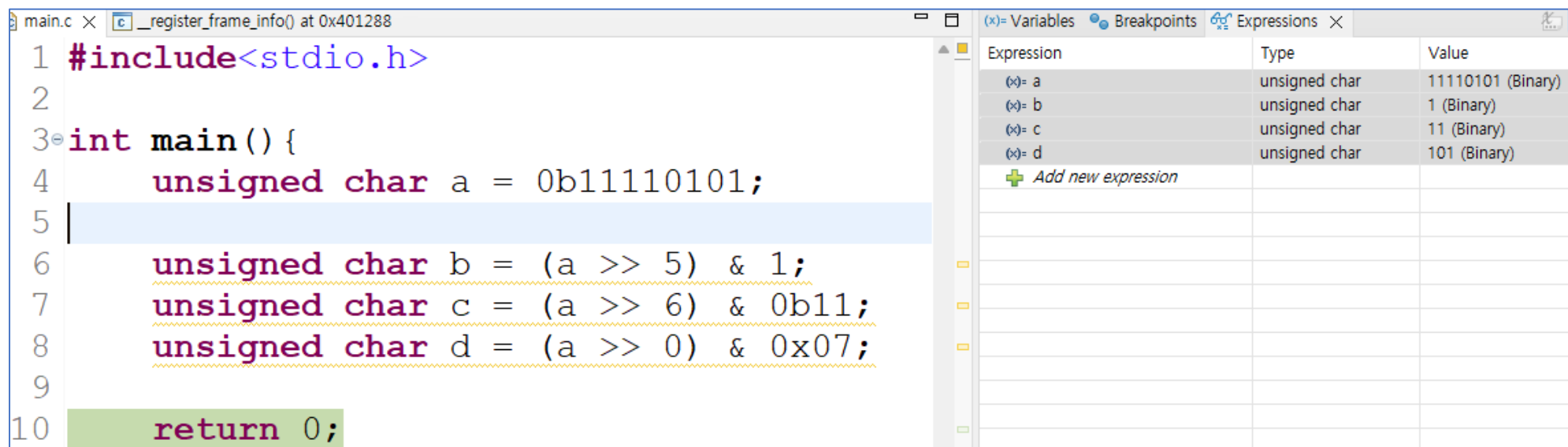
- `c = (a >> 6) & 0b11;`

3. 0번 bit에서 3개 bit 추출하여 변수 d에 저장하기

- `d = (a >> 0) & 0x07;`

<https://gist.github.com/hoconoco/5f1fd30f4200272dc040955132001808>

조사식으로 확인한다.



The screenshot shows a debugger window with a C code file named `main.c` open. The code is as follows:

```
1 #include<stdio.h>
2
3 int main() {
4     unsigned char a = 0b11110101;
5
6     unsigned char b = (a >> 5) & 1;
7     unsigned char c = (a >> 6) & 0b11;
8     unsigned char d = (a >> 0) & 0x07;
9
10    return 0;
```

On the right side, the 'Expressions' pane shows the following data:

Expression	Type	Value
(x)= a	unsigned char	11110101 (Binary)
(x)= b	unsigned char	1 (Binary)
(x)= c	unsigned char	11 (Binary)
(x)= d	unsigned char	101 (Binary)
+ Add new expression		

<https://gist.github.com/hoconoco/9e65a1bf1a52aef3f1b8b5e63ed4e4da>

방금 한 비트 추출에서도 다양하게 표현이 가능하다

- $b = (a \gg 2) \& 0x01$
- $b = (a \gg 2) \& 1$
- $b = (a \gg 2) \& 0b1$
- 등등

관련 자료를 좀 더 찾아보고 싶어서 검색을 하거나
다른 사람이 짠 코드를 보면, 다양한 방식으로 표현이 된다.

- 그 코드를 보고 이해할 수 있어야 한다.

비트 Set / Clear

목표

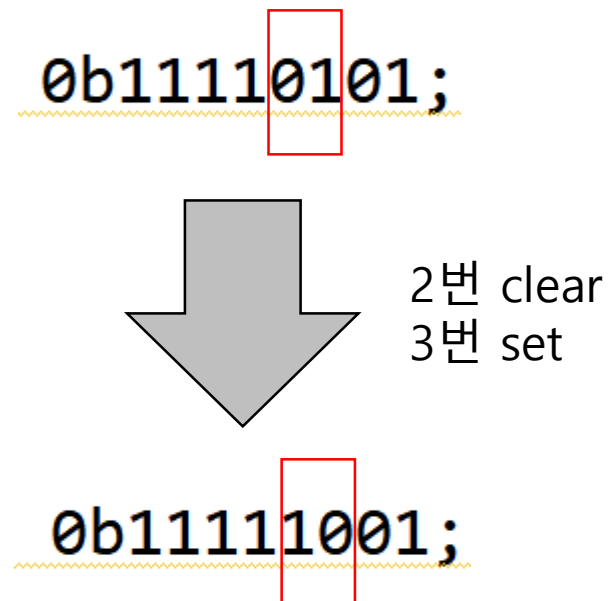
- 특정 비트를 set / clear 하는 방법에 대해 학습한다.
- 공식처럼 외우기 보다 왜 그렇게 되는 지 원리를 이해한다.

비트 clear

- 특정 비트를 0 으로 만드는 것을 clear 한다고 표현한다.

비트 set

- 특정 비트를 1로 만드는 것을 set 한다고 한다.



1번 bit에서 1개 bit set 한 값을 저장하기

• $a1set = (0x1 \ll 1) | a;$

```
unsigned char a = 0b11000001;
```

```
unsigned char a1set = 0;
```

1	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

비트 연산 계산방법은
암기하지 않고, 스스로 생각하여
짚 수 있도록 해야한다.

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b00000001;`

1. 6번 bit에서 1개 bit set하여 변수 b에 저장하기

2. 3번 bit에서 2개 bit set하여 변수 c에 저장하기

3. 0번 bit에서 3개 bit set하여 변수 d에 저장하기

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b00000001;`

1. 6번 bit에서 1개 bit set하여 변수 b에 저장하기

- `b = (1<<6) | a`

2. 3번 bit에서 2개 bit set하여 변수 c에 저장하기

- `c = (0b11 << 3) | a`

3. 0번 bit에서 3개 bit set하여 변수 d에 저장하기

- `d = (0x7<<0) | a`

<https://gist.github.com/hoconoco/373cf2c91f2771883d8b153ceb5a2790>

2번 bit에서 1개 bit, clear 하기

• $a2clr = a \& \sim(1 \ll 2);$

```
unsigned char a = 0b11001110;
```

```
unsigned char a2clr = 0;
```

비트연산 계산방법은
암기하지 않고, 스스로 생각하여
짚수있도록 해야한다.

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b11111111;`

1. 1번 bit에서 1개 bit clear하여 변수 b에 저장하기

2. 3번 bit에서 2개 bit clear하여 변수 c에 저장하기

3. 5번 bit에서 3개 bit clear하여 변수 d에 저장하기

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b11111111;`

1. 1번 bit에서 1개 bit clear하여 변수 b에 저장하기

- `b = ~(1<<1) & a`

2. 3번 bit에서 2개 bit clear하여 변수 c에 저장하기

- `c = ~(0x3<<3) & a`

3. 5번 bit에서 3개 bit clear하여 변수 d에 저장하기

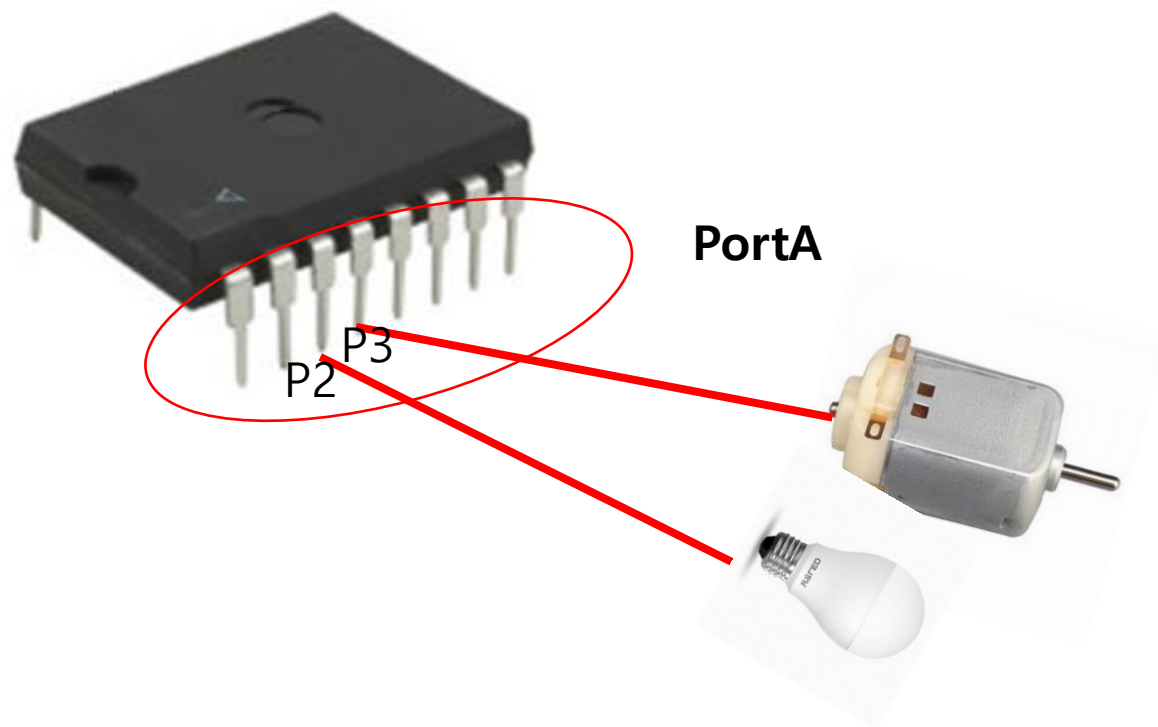
- `d = ~(0b111<<5) & a`

<https://gist.github.com/hoconoco/2d74a0b2a5c1fda03f6fedc5051107aa>

1번 bit에 1을 넣고 싶으면 $a = 0b10$ 을 하면 되지 않을까?

왜? 비트 연산으로 번거롭게 할까?

- 2번 bit 에 있는 전등을 끄자고 $a = 0b10$ 으로 나머지 bit에 전부 0을 주면, 3번 bit에 연결된 모터도 꺼진다.
- 해당 모터의 기능이 정말 중요한 냉각 장치의 모터였는데 꺼졌다면???
- 시말서로 끝나지 않을 수 있다.



비트 반전

XOR를 이용하여 2번 bit 반전

- $b2xor = a \wedge (1 \ll 2);$

```
unsigned char a = 0b10011100;  
unsigned char b2xor = 0;
```

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b10011100;`

1. 1번 bit에서 1개 bit 반전하여 변수 b에 저장
2. 변수 b의 1번 bit를 1개 bit 반전하여 c에 저장
3. 5번 bit에서 3개 bit 반전하여 변수 d에 저장
4. 변수 d의 5번 bit에서 3개 bit 반전하여 변수 e에 저장

아래 퀴즈를 코드로 작성한다.

- `unsigned char a = 0b10011100;`

1. 1번 bit에서 1개 bit 반전하여 변수 b에 저장

- `b = a ^ (1<<1)`

2. 변수 b의 1번 bit를 1개 bit 반전하여 c에 저장

- `c = b ^ (1<<1)`

3. 5번 bit에서 3개 bit 반전하여 변수 d에 저장

- `d = a ^ (0b111<<5)`

4. 변수 d의 5번 bit에서 3개 bit 반전하여 변수 e에 저장

- `e = d ^ (0x7<<5)`

<https://gist.github.com/hoconoco/3aea3a7c35ee50d69ff1e71e75c34efa>

Day2-5. 도전

0x8FAA2D

1. 2진수로 변환하기
2. LSB 값은?
3. 2, 7, 10번 bit 값은?
4. 이 값을 저장하려면, 최소 Byte 공간이 필요할까?
5. 24 bit 기준으로 ~0x8FAA2D 값은 무엇인가?

다음 지시대로 했을 때, 나오는 숫자를 맞춰보자.

- 이 수는 2Byte 로 표현된다.
- MSB는 1로 세팅
- 2번 bit 부터 5번 bit까지 1로 세팅
- 8번 ~ 13번 bit까지 1로 세팅
- 3번 ~ 7번 bit를 반전(~)시키기
- 최상위 3 bit를 모두 0으로 세팅하기

나오는 정답을 16진수로 나타내시오.

a 에서 비트연산을 이용하여, 비트를 추출한다.

- b5 에는 5번 bit 값을 저장한다.
- b3 에는 0번 bit 값을 저장한다.
- b67 에는 6~7번 bit 값을 저장한다.

```
unsigned char a = 0b11110101;
```

```
unsigned char b5 = 0;
```

```
unsigned char b0 = 0;
```

```
unsigned char b67 = 0;
```

a 에서 비트연산을 이용하여, 비트를 추출한다.

- msb 한 비트와 lsb 한 비트를 추출하여, 합친 값을 저장한다.
- 만약 msb가 1 이고, lsb가 1 이라면
저장해야 하는 값은 11 이다.

```
unsigned char a = 0b11110101;  
  
unsigned char msb_lsb;
```

a 값을 다음과 같이 변경한다.

1. MSB Clear
2. LSB Set
3. 2~7번 bit Set
4. 5~6번 bit Clear
5. 3번 Clear

```
unsigned char a = 0b10000000;
```



```
0b10010101;
```

16진수로 된 문자열과 2진수로 된 문자열을 입력 받는다.

- 두 수를 A, B를 입력 받는다. 8bit 가정

1. $\sim(A \& B) \ll 4$ 를 계산한 결과를 출력한다. (10진수)

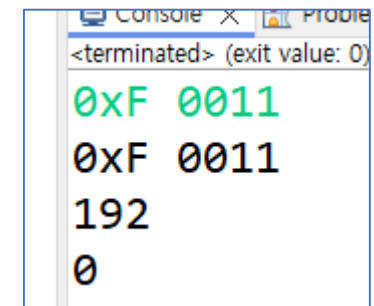
ex) A : 0xF, B : 0011

결과 : $0000\ 1111 \& 0000\ 0011 \rightarrow \sim(0000\ 0011) \rightarrow 1111\ 1100 \ll 4 \rightarrow 1100\ 0000 \rightarrow 0xC0\ (192)$

2. $(A \wedge 0xCD) \& (B \gg 3)$ 를 계산한 결과를 출력한다.

ex) A : 0xF, B : 0011

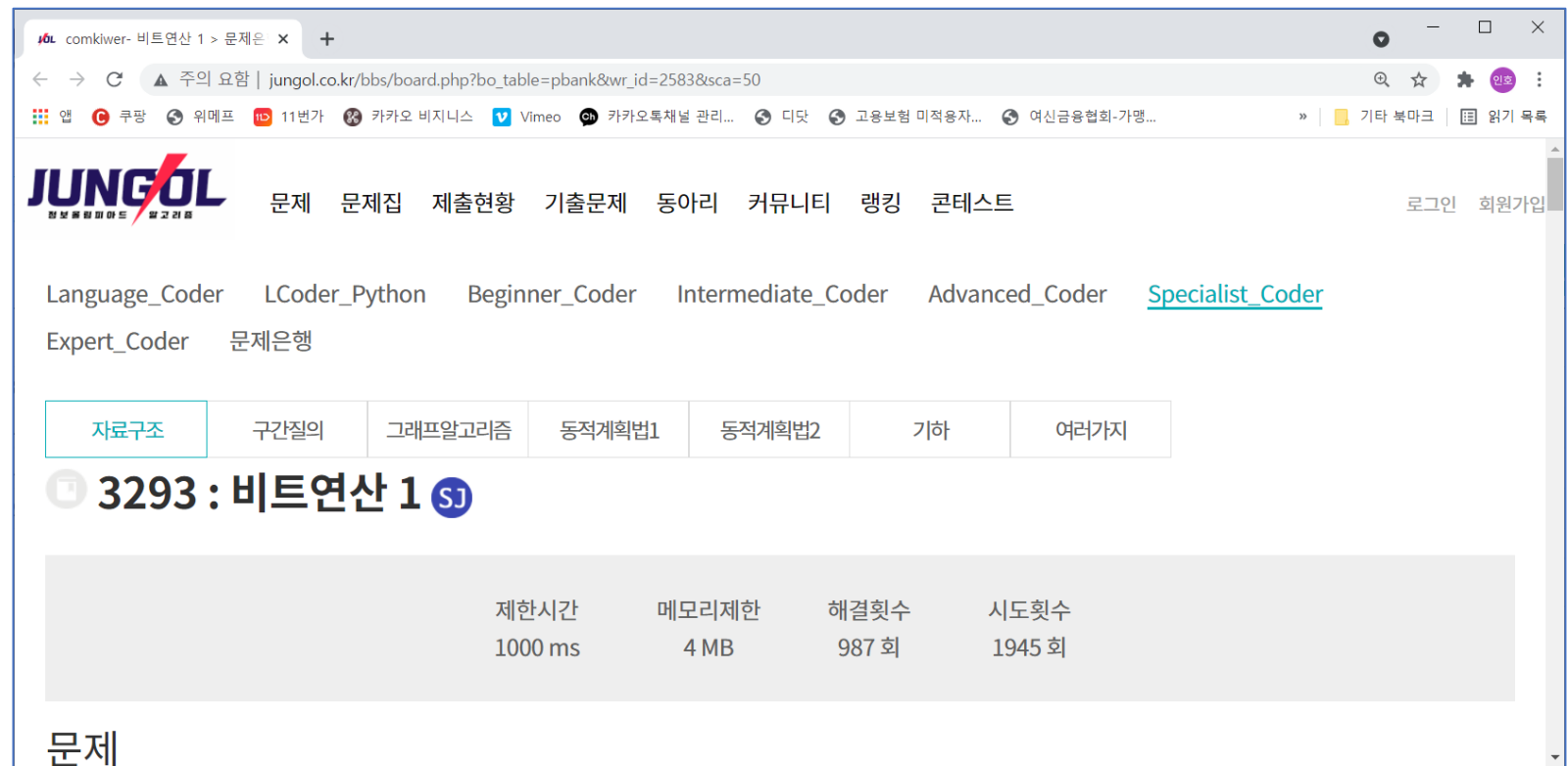
결과 : $0000\ 1111 \wedge 1100\ 1101 \rightarrow 1100\ 0010 \& (0011 \gg 3) \rightarrow 1100\ 0010 \& 0000\ 0000 \rightarrow 0000\ 0000\ (0)$



```
Console x | Problem
<terminated> (exit value: 0)
0xF 0011
0xF 0011
192
0
```


문제를 푼다.

- <https://www.jungol.co.kr/problem/3293>



<https://gist.github.com/hoconoco/b0f306763332390b733568dc323830bb>

내일 방송에서 만나요!

삼성 청년 SW 아카데미

[과제1] p.91 코드와 결과물 캡처해서 제출

[과제2] p.92 코드와 결과물 캡처해서 제출

[과제3] p.93 코드와 결과물 캡처해서 제출

[과제4] p.94 코드와 결과물 캡처해서 제출

[과제5] p.95 코드와 결과물 캡처해서 제출

[과제6] p.96 코드와 결과물 캡처해서 제출

[과제7] p.97 코드와 결과물 캡처해서 제출