プログラムの開発手順

１．プログラム設計（仕様の決定）

２．コーディング（ソースファイルの作成）

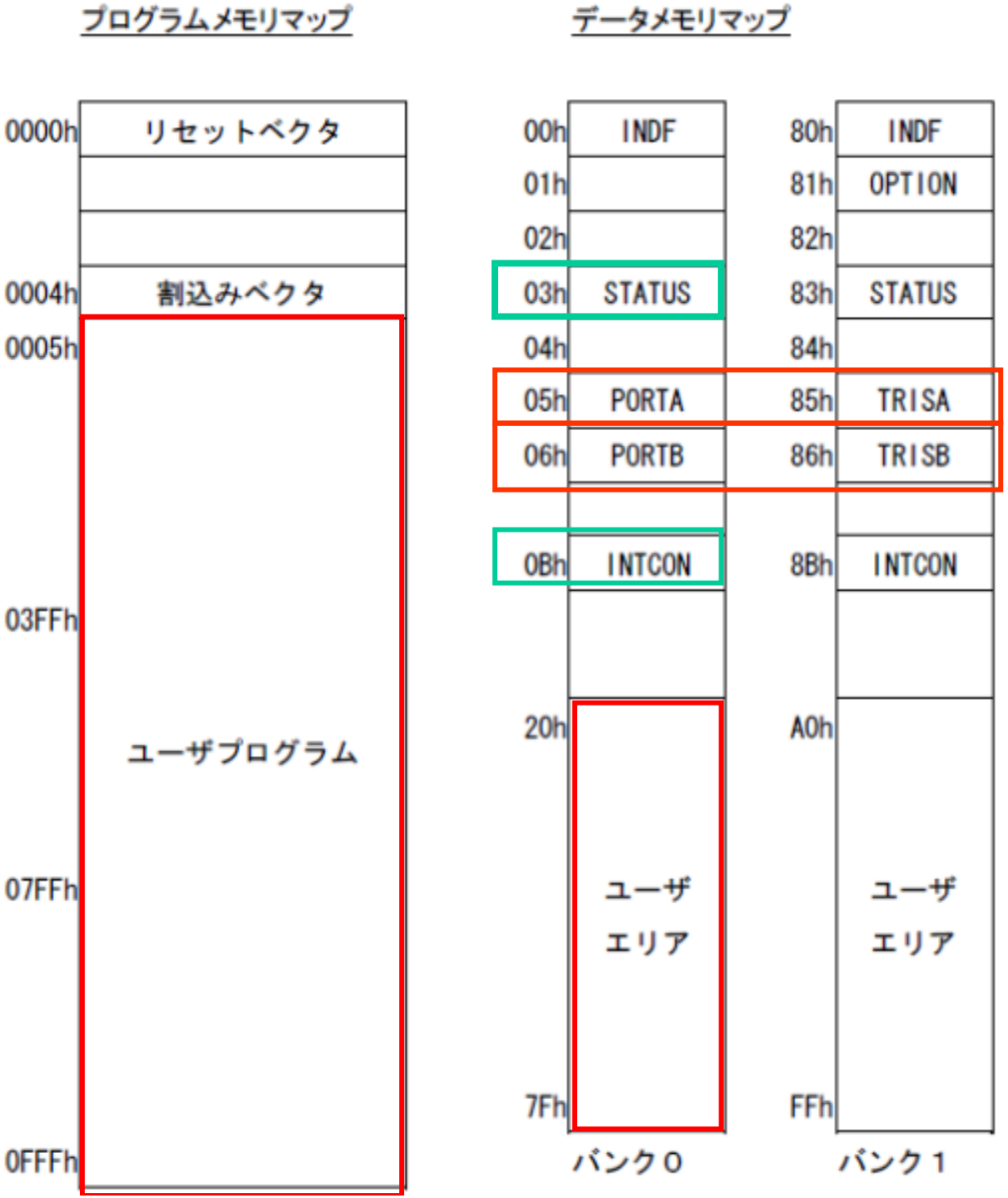
３．アセンブル（オブジェクトファイル
→ヘキサファイルの作成）

４．シミュレーション（誤りの検出）

ＭＰＬＡＢを活用

５．ＰＩＣライタを使用してマイコンにプログラムを書き込む

６．テスト


プログラムメモリマップ
0000h
リセットベクタ
0004h
割込みベクタ
0005h
ユーザプログラム
03FFh
07FFh
0FFFh
データメモリマップ
00h
INDF
01h
02h
03h
STATUS
04h
05h
PORTA
06h
PORTB
0Bh
INTCON
20h
ユーザ
エリア
7Fh
バンク0
80h
INDF
81h
OPTION
82h
83h
STATUS
84h
85h
TRISA
86h
TRISB
8Bh
INTCON
A0h
ユーザ
エリア
FFh
バンク1

アセンブラ命令語

■ バイト処理命令; バイト（8ビット）の値を処理

1)加算　　（UAはユーザが設けたデータ格納レジスタ）

```
ADDWF  UA, 0        ; W = W+UA
ADDWF  UA, 1        ; UA = W+UA
```

（0とすると加算結果をWレジスタに保存）

2) 論理積

```
ANDWF  UA, 0        ; W = W & UA
```

3) 値をゼロクリア

```
CLRF   UA           ; UA=0
CLRW                ; W=0
```

4) 値 (0, 1) の反転

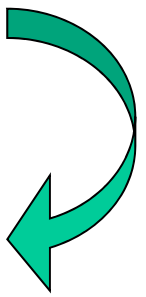
```
        COMF  UA , 1          ; UA = UA ^ 0xFF
```

5) 値を１減らす

```
        DECF  UA ,  1         ; UA = UA-1
```

6) 値を１減らし、0になったら次の命令をスキップ

```
        DECFSZ   UA, 1        ; UA = UA –1
                                  IF (UA==0) SKIP
        GOTO     KURIKAESHI ;
        RETURN      ; サブルーチンから戻る
```



7) 値を１増やす

```
INC   UA, 1          ; UA = UA + 1
```

8) 値を１増やし、０になったら次命令をスキップ

```
INCFSZ  UA, 1        ; UA = UA +1
                     ; IF (UA==0) SKIP;
```

9) 論理和

```
IORWF  UA, 1         ; UA = W | UA
```

10) データの移動（コピー）

```
MOVF  UA, 0          ; W = UA (UAの値をWへ)
MOVWF UA             ; UA = W (Wの値をUAへ)
```

11)何もしない

NOP

12)１ビット左シフト

RLF　UA,1　; UA=‘01000000’,C=1(キャリー)

とすると、

UA=‘10000001’,C=0 になる

13)1ビット右シフト

RRF UA,1　;UA=‘01000000’,C=1(キャリー)

とすると、

UA=‘10100000’,C=0 になる

14) 減算

```
SUBWF   UA, 1        ; UA = UA – W
```

15) 上位4ビットと下位4ビットの値を入れ替え

```
SWAPF   UA, 1        ; UA= ‘01010000’ とすると、
                       UA= ‘00000101’
```

16) 排他的論理和

```
XORWF  UA, 1         ; UA = UA ^ W
```

■ ビット処理命令

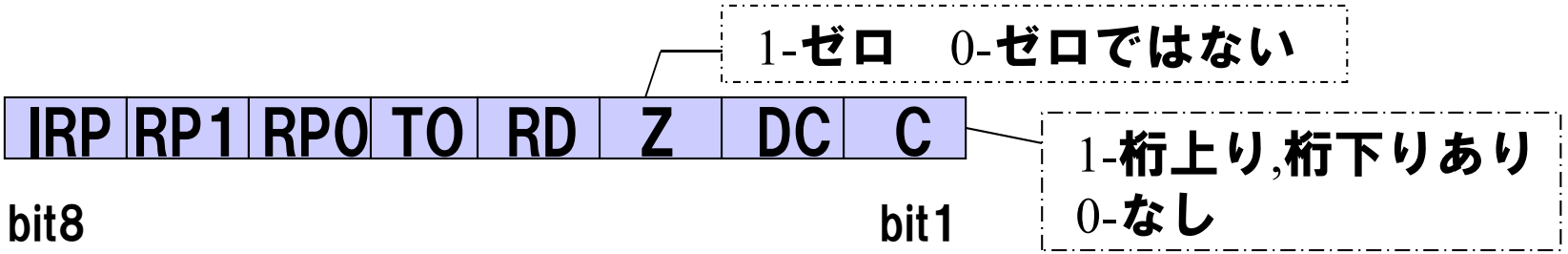
1) あるビットをゼロにする

```
BCF   INTCON, 7      ; INTCONレジスタの7ビットをゼロ
```

2) あるビットを１にセットする

```
BSF   STATUS, 5      ; STATUSレジスタの5ビットを１にする
```

＊ INTCONやSTATUSレジスタは「特殊レジスタ」。
予め機能が定められている。



Statusレジスタの機能

3) ビット検査命令

BTFSC UA, 3 ; UAの3ビットを調べ、ゼロ（クリア）
だったら、次の命令をスキップ

BTFSS UA, 3 ; UAの3ビットを調べ、１（セット）
だったら、次の命令をスキップ

■ リテラル命令; 定数を伴う演算

1) 加算

```
ADDLW   34H       ; W = W + 34H
```

2) 論理積

```
ANDLW   45H       ; W = W & 45H
```

3) 論理和

```
IORLW   56H       ; W = W | 56H
```

4) 定数の読出し（移動）

```
MOVLW   78H       ; W = 78H
```

＊Hが付いている場合は値が16進数表記

5) 減算

```
SUBLW   89H     ; W = 89H – W
```

6) 排他的論理和

```
XORLW   9AH     ; W = W ^ 9AH
```

■ ＣＰＵ動作モード設定・解除

1) ウォッチドックタイマクリア

```
CLRWDT
```

2) スリープモード設定

```
SLEEP
```

■ ジャンプ命令

1）サブルーチンの呼び出し

CALL　SUB1　　；サブルーチンSUB1を呼び出し

＊サブルーチンSUB1内のRETURN命令が実行されると、
CALLの次に記述した命令が実行される。

2）指定ラベルへジャンプ

GOTO　SAKURA ；ラベルSAKURAやジャンプ

3）サブルーチンから戻る

RETURN

RETLW　k　（Wにｋを格納して戻る）

RETFIE　　（割込み利用の場合）

演習課題の進め方

１．製作内容の決定

２．部品の配置を決める（回路図を描く）

３．処理の流れをフローチャートに表す

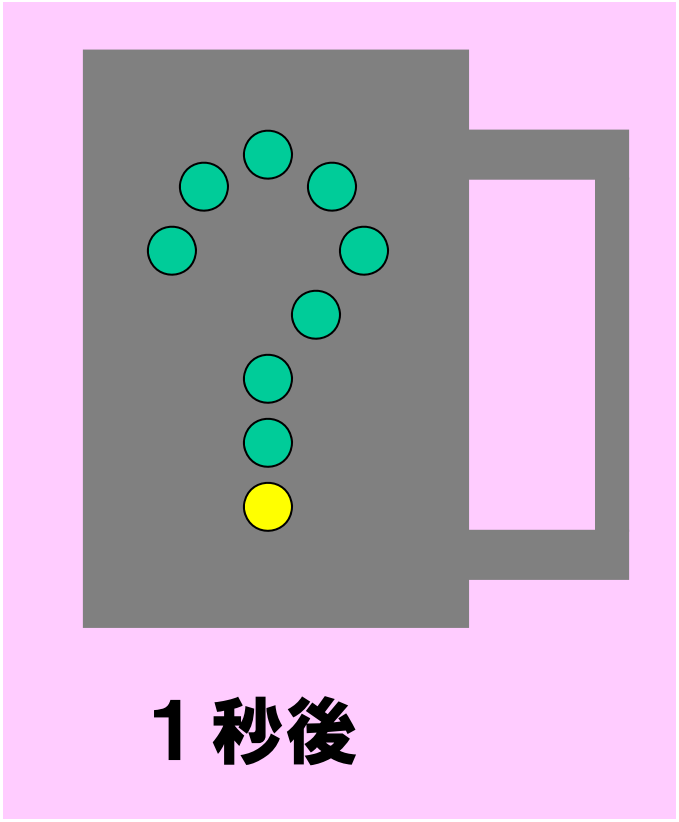
４．プログラム作成＆アセンブル

（ＨＥＸファイルの生成）

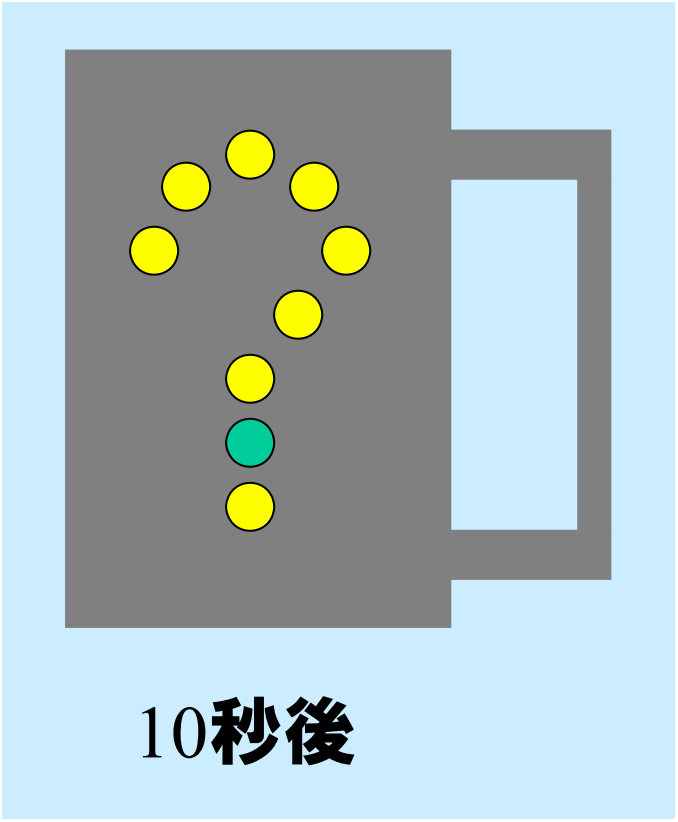
５．PICマイコンへの書込み

６．回路組立＆動作検証

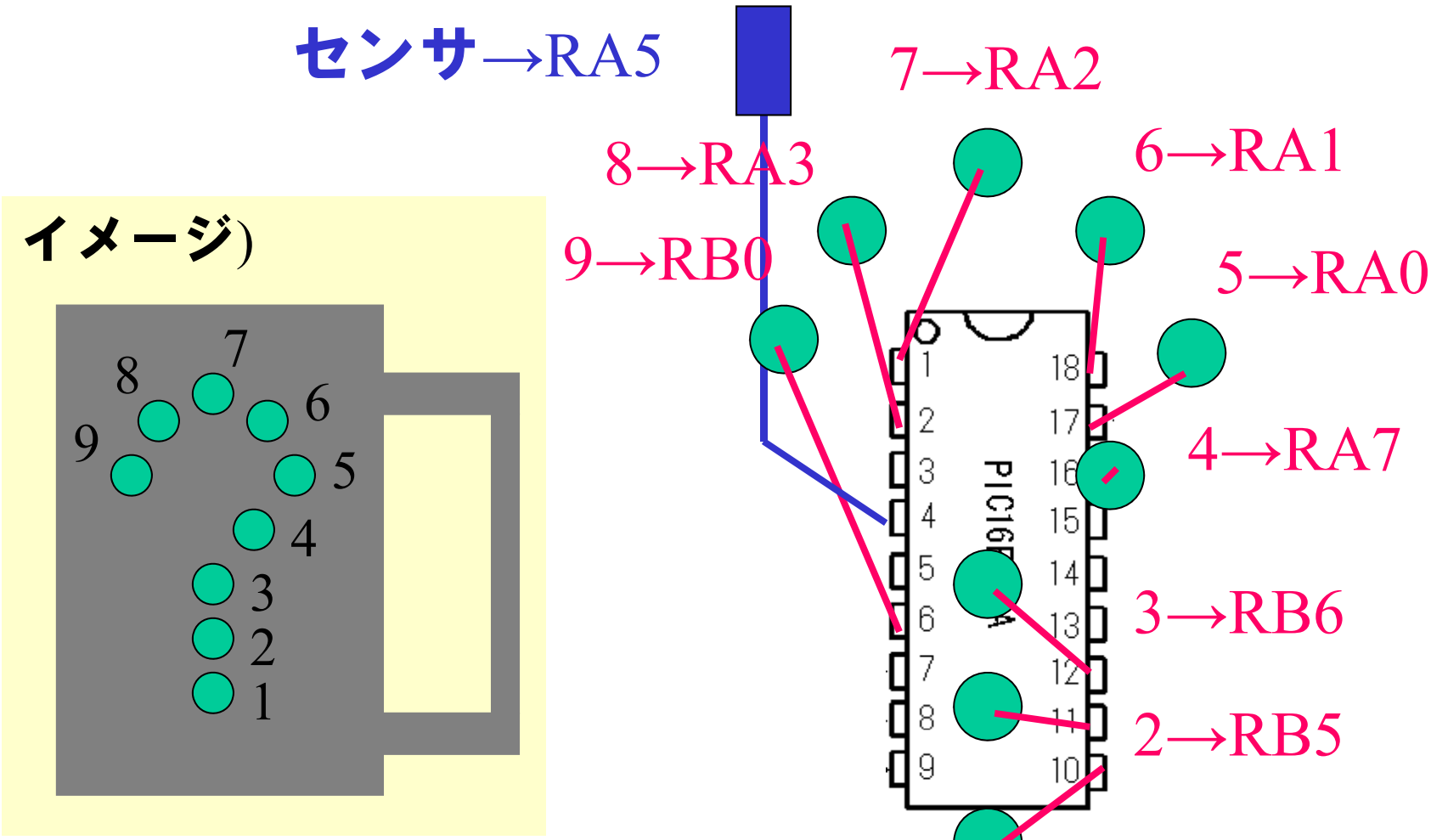
例）グラスを傾けると下から順にLEDが
点灯し，10秒後に“？”マークを描く



１秒後



10秒後



「使用可能な近い端子は何処？」

LEDとセンサの配置

ポート単位でLEDとセンサの配置をまとめる

ポートA:

RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
4		**センサ**		**8**	**7**	**6**	**5**

ポートB:

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	3	**2**	**1**				**9**

LED1**のみを点灯させる時のデータは、**

ポートA: 1111 1111

ポートB: 1110 1111

※点灯させる箇所を「0**」にする**