Looping 100,000 times

```
COUNT1  EQU 0x20      ; Outer loop counter
COUNT2  EQU 0x21      ; Middle loop counter
COUNT3  EQU 0x22      ; Inner loop counter

        ORG 0x00
        GOTO MAIN

MAIN:
                      ; Initialize PORTB = 55H
        MOVLW 0x55
        MOVWF PORTB

                      ; Load outer loop count = 10
        MOVLW d'10'
        MOVWF COUNT1

LOOP1:
                      ; Load middle loop count = 100
        MOVLW d'100'
        MOVWF COUNT2

LOOP2:
                      ; Load inner loop count = 100
        MOVLW d'100'
        MOVWF COUNT3

LOOP3:
        COMF PORTB, F          ; Toggle PORTB
        DECFSZ COUNT3, F       ; Inner loop counter --
        GOTO LOOP3

        DECFSZ COUNT2, F       ; Middle loop counter --
        GOTO LOOP2

        DECFSZ COUNT1, F       ; Outer loop counter --
        GOTO LOOP1

        END
```

Low byte & High byte

```
 L_BYTE  EQU 0x05       ; Low byte of sum
 H_BYTE  EQU 0x06       ; High byte of sum

         ORG 0x00
         CLRF   L_BYTE          ; Clear low byte
         CLRF   H_BYTE          ; Clear high byte

                                ; Load values into memory
         MOVLW  0x4C     ; W = 4Ch
         MOVWF  0x40     ; Store at 0x40
```

```
         MOVLW  0x3F          ; W = 3Fh
        MOVWF  0x41        ; Store at 0x41

        MOVLW  0xD3        ; W = D3h
        MOVWF  0x42        ; Store at 0x42

        MOVLW  0xA1        ; W = A1h
        MOVWF  0x43        ; Store at 0x43

                           ; --- Add first number (0x40) ---
        MOVF   0x40, W
        ADDWF  L_BYTE, F
        BNC    N1
        INCF   H_BYTE, F
N1:
                           ; --- Add second number (0x41) ---
        MOVF   0x41, W
        ADDWF  L_BYTE, F
        BNC    N2
        INCF   H_BYTE, F
N2:
                           ; --- Add third number (0x42) ---
        MOVF   0x42, W
        ADDWF  L_BYTE, F
        BNC    N3
        INCF   H_BYTE, F
N3:
                           ; --- Add fourth number (0x43) ---
        MOVF   0x43, W
        ADDWF  L_BYTE, F
        BNC    OVER
        INCF   H_BYTE, F

OVER:
        END
```

Set to swap even and odd bits of 56H and store in new file register

```
SRC     EQU 0x56
DST     EQU 0x57
TEMP1   EQU 0x20
TEMP2   EQU 0x21

        ORG 0x00

        ; --- Extract even bits (mask = 01010101) ---
        MOVF    SRC, W
        ANDLW   0x55
        MOVWF   TEMP1
        RLF     TEMP1, F    ; shift left → even bits move to odd positions

        ; --- Extract odd bits (mask = 10101010) ---
        MOVF    SRC, W
        ANDLW   0xAA
        MOVWF   TEMP2
        RRF     TEMP2, F    ; shift right → odd bits move to even positions

        ; --- Combine ---
        MOVF    TEMP1, W
        IORWF   TEMP2, W
        MOVWF   DST

       END
```

a) Role of Stack and subroutines in PIC. What happens when the Stack overflows?

- **Stack role:**
    - The stack stores the return address whenever a subroutine (CALL) or interrupt is executed.
    - When the subroutine finishes (RETURN), the stored address is popped back so the program continues from the correct location.
- **Subroutines role:**
    - Subroutines allow code reuse by grouping instructions into callable blocks.
    - Helps reduce program size and improves modularity.
- **Stack overflow:**
    - If the stack becomes full and another CALL/interrupt occurs, it overwrites old return addresses.
    - This causes the program to return to the wrong place → **program crashes or behaves unpredictably.**

b) CALL vs RCALL for PIC18 with 4K program memory

- **CALL:**
    - Absolute call to any address in the entire program memory (useful for large memory space).
- **RCALL:**
    - Relative call (address specified as an offset from current location).
    - More compact instruction (uses fewer program words).
    - Suitable for small memory devices.

Since the chip has **only 4K program memory**, **RCALL** is more useful because it saves memory and is efficient for short-range subroutine calls.

Final short answers:
a) Stack stores return addresses for subroutines/interrupts. Overflow → overwriting → wrong return address → program error.
b) For a 4K ROM PIC18, **RCALL** is preferred (saves memory, efficient for small program space).