# CENG 329 Project: Push or Perish

## Team Members

- Hasan Emre Usta (202111301)  
- Gökay Çetinakdoğan (202111050)  
- Kayra Dalçık (202111023)

## Objective

The objective of this project is to design and implement a competitive game using an MSP430 microcontroller, a 7-segment display, two buttons, and two LEDs. The game, "Push or Perish," requires players to press buttons strategically to win based on predefined rules.

## Project Specifications

### Countdown Mechanism

- The 7-segment display counts down from 3 to 0, decreasing by 1 every second.  
- When the countdown reaches 0, the display shows a "-" (dash) to indicate the game has ended.

### Win Conditions

1. If a player presses their button before the countdown reaches 0, the other player automatically wins, and their corresponding LED lights up.  
2. If both players wait until the countdown reaches 0, the LED of the first player to press their button will light up.

### Game Restart

- After one player wins, the game pauses for 3 seconds before automatically restarting for a new round.

## Components Used

- 1 x MSP430 Microcontroller  
- 1 x 7-Segment Display  
- 2 x Buttons (for Player 1 and Player 2)  
- 2 x LEDs (one for each player)

## Implementation Details

### Hardware Setup

1. **LEDs**:  
 - Red LED (Player 1): Configured as output (P1.0).  
 - Green LED (Player 2): Configured as output (P2.1).  
2. **Buttons**:  
 - Player 1 Button: Configured as input (P1.3) with a pull-up resistor.  
 - Player 2 Button: Configured as input (P2.3) with a pull-up resistor.  
3. **7-Segment Display**:  
 - Pins configured for digital I/O.  
 - Segments controlled via P1OUT and P2OUT registers.

### Software Flow

1. **Countdown Display**:  
 - The 7-segment display cycles through numbers 3 to 0 using subroutines. Each subroutine activates the appropriate segments for the digit and then calls the delay function.  
 - At 0, a dash "-" is displayed to indicate the end of the countdown.  
2. **Button Detection**:  
 - The program constantly monitors the button states.  
 - Logic determines whether the press occurred before or after the countdown reaches 0.  
3. **LED Control**:  
 - If a button is pressed prematurely, the opponent's LED lights up.  
 - If a button is pressed after the countdown, the corresponding player's LED lights up.  
4. **Game Reset**:  
 - A 3-second delay is triggered before restarting the game.

## Assembly Code Highlights

The program uses MSP430 assembly language. Below are key excerpts and explanations:

### Countdown Sequence

show\_1:

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_0

show\_2:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT7, &P1OUT ; Turn on e

bic.b #BIT2, &P2OUT ; Turn on g

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_1

show\_3:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT2, &P2OUT ; Turn on g

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_2

show\_0:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT7, &P1OUT ; Turn on e

bic.b #BIT0, &P2OUT ; Turn on f

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_

show\_:

bic.b #BIT2, &P2OUT ; Turn on g

jmp MAIN\_LOOP

### Delay Function

delay:  
 mov.w #500, r5  
out\_loop:  
 mov.w #100, r6  
in\_loop:  
 call #rule\_2 ; Check button states during delay  
 dec.w r6  
 jnz in\_loop  
 dec.w r5  
 jnz out\_loop  
 ret  
**Explanation**:  
- The `delay` function implements a nested loop for timing.  
- During the delay, the `rule\_2` subroutine is called repeatedly to check if any button is pressed. This ensures real-time response without waiting for the delay to complete.

## Results and Observations

- The system successfully implements the "Push or Perish" game with the specified rules.  
- The countdown and button press detection work as intended.  
- LEDs and the 7-segment display are synchronized with game logic.

## Challenges

- Fine-tuning the delay loops for consistent timing.  
- Debugging hardware connections for the 7-segment display.

## YouTube Link

<https://www.youtube.com/shorts/hu23h-f-BZg>

## Source Code

BIS.B #BIT0, &P1DIR ; Set P1.0 as output (Red LED)

BIC.B #BIT0, &P1OUT ; Ensure Red LED is off initially

BIC.B #BIT3, &P1DIR ; Set P1.3 as input (Button 1)

BIS.B #BIT3, &P1REN ; Enable pull-up/down resistor for P1.3

BIS.B #BIT3, &P1OUT ; Configure pull-up resistor on P1.3

BIS.B #BIT1, &P2DIR ; Set P2.1 as output (Green LED)

BIC.B #BIT1, &P2OUT ; Ensure Green LED is off initially

BIC.B #BIT3, &P2DIR ; Set P2.3 as input (Button 2)

BIS.B #BIT3, &P2REN ; Enable pull-up/down resistor for P2.3

BIS.B #BIT3, &P2OUT ; Configure pull-up resistor on P2.3

bic.b #10111110b, &P1SEL ; make P1.1, 2, 3, 4, 5, and 7 Digital I/O

bic.b #10111110b, &P1SEL2 ; make P1.1, 2, 3, 4, 5, and 7 Digital I/O

bic.b #00000101b, &P2SEL ; make P2.0 and 2 Digital I/O

bic.b #00000101b, &P2SEL2 ; make P2.0 and 2 Digital I/O

bis.b #10110110b, &P1DIR ; make P1.1, 2, 4, 5, and 7 output

bis.b #00000101b, &P2DIR ; make P2.0 and 2 output

bic.b #BIT3, &P1DIR ; make P1.3 input

bis.b #BIT3, &P1REN ; enable pull-up resistor for P1.3

bis.b #BIT3, &P1OUT ; enable pull-up resistor for P1.3

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

initial:

BIC.B #BIT0, &P1OUT ; Ensure Red LED is off initially

BIC.B #BIT1, &P2OUT ; Ensure Green LED is off initially

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

call #show\_3

show\_1:

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_0

show\_2:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT7, &P1OUT ; Turn on e

bic.b #BIT2, &P2OUT ; Turn on g

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_1

show\_3:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT2, &P2OUT ; Turn on g

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_2

show\_0:

bic.b #BIT1, &P1OUT ; Turn on a

bic.b #BIT2, &P1OUT ; Turn on b

bic.b #BIT4, &P1OUT ; Turn on c

bic.b #BIT5, &P1OUT ; Turn on d

bic.b #BIT7, &P1OUT ; Turn on e

bic.b #BIT0, &P2OUT ; Turn on f

call #delay

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

jmp show\_

show\_:

bic.b #BIT2, &P2OUT ; Turn on g

jmp MAIN\_LOOP

MAIN\_LOOP:

BIT.B #BIT3, &P2IN ; Check state of Button 2 (P2.3)

JZ BUTTONPRESSED2

BIT.B #BIT3, &P1IN ; Check state of Button 1 (P1.3)

JZ BUTTONPRESSED1

JMP MAIN\_LOOP

BUTTONPRESSED1:

BIS.B #BIT0, &P1OUT ; Turn on Red LED (P1.0)

JMP reset\_game

BUTTONPRESSED2:

BIS.B #BIT1, &P2OUT ; Turn on Green LED (P2.1)

jmp reset\_game

rule\_2:

BIT.B #BIT3, &P2IN ; Check state of Button 2 (P2.3)

JZ bt2\_r2

BIT.B #BIT3, &P1IN ; Check state of Button 1 (P1.3)

JZ bt1\_r2

ret

bt2\_r2:

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

bic.b #BIT2, &P2OUT ; Turn on g

BIS.B #BIT0, &P1OUT ; Turn on Red LED (P1.0)

JMP reset\_game

bt1\_r2:

bis.b #10110110b, &P1OUT ; All segments OFF

bis.b #00000101b, &P2OUT ; All segments OFF

bic.b #BIT2, &P2OUT ; Turn on g

BIS.B #BIT1, &P2OUT ; Turn on Green LED (P2.1)

jmp reset\_game

delay2:

mov.w #1000,r5

out\_loop2:

mov.w #750,r6

in\_loop2:

dec.w r6

jnz in\_loop2

dec.w r5

jnz out\_loop2

ret

delay:

mov.w #500,r5

out\_loop:

mov.w #100,r6

in\_loop:

call #rule\_2

dec.w r6

jnz in\_loop

dec.w r5

jnz out\_loop

ret

reset\_game:

call #delay2

jmp initial

end:

nop