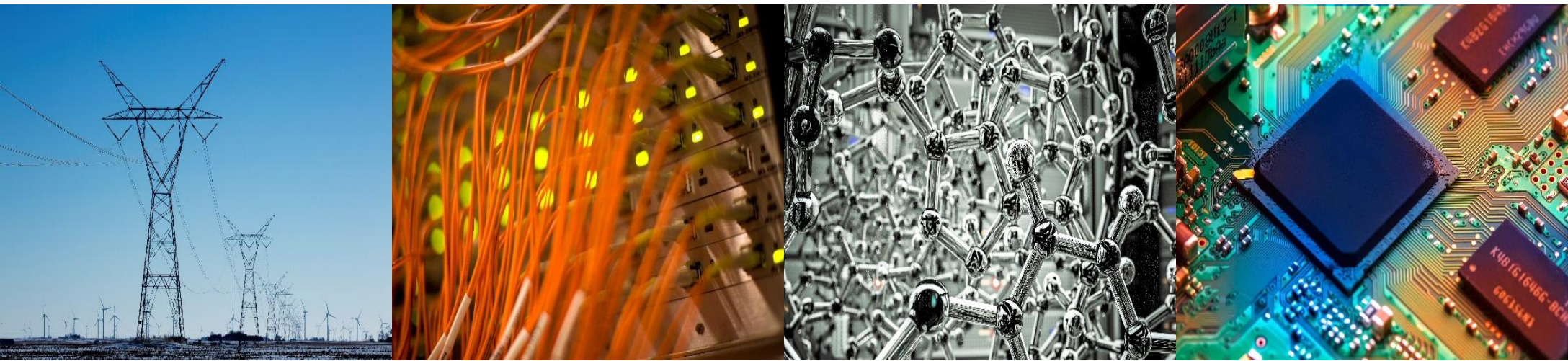# ECE 220 Computer Systems & Programming

Lecture 4 – Programming with Stack

September 5, 2024

- **Quiz 1 is available for reservation**
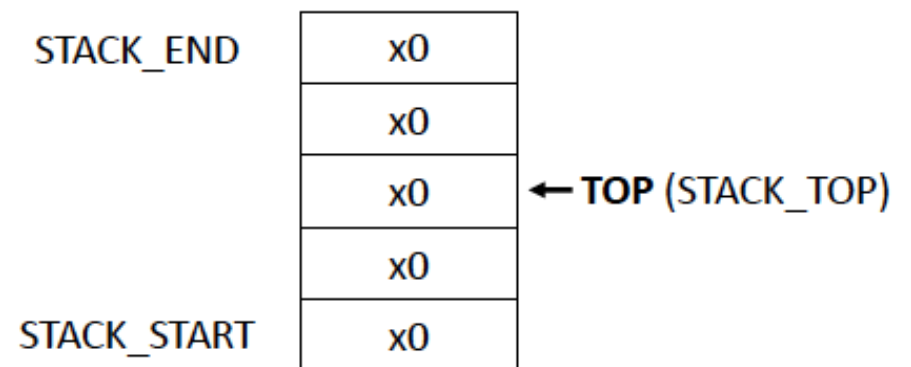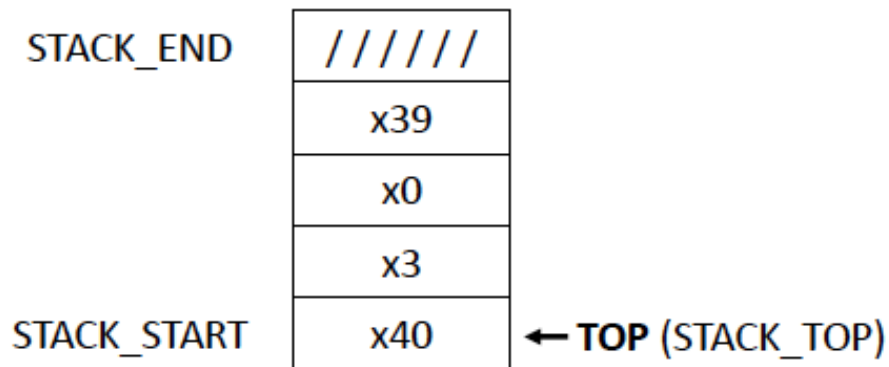- **Mock Quiz is next Monday through Wednesday**

Lecture 3 Review: Stack

❑ Order of Access

❑ Two Main Operations

❑ Overflow vs. Underflow

❑ Hardware vs. Software Implementation

❑ Top of Stack Pointer (stack pointer)

➢ In the following two figures, which stack is empty? (Note: STACK_TOP points to the next available spot.)

STACK_END	//////
	x39
	x0
	x3
STACK_START	x40	← **TOP** (STACK_TOP)

STACK_END	x0
	x0
	x0	← **TOP** (STACK_TOP)
	x0
STACK_START	x0

Run-Time Stack

- Information of an invoked function (subroutine) is stored in a memory template called the **activation record** or **stack frame**.

- Functions' activation records are pushed onto the Run-Time Stack in the order they are invoked.

- ❖ **Supervisor Stack** is different from Run-Time Stack (more details at the end of the semester).

Address	Region
x0000 – x00FF	Trap Vector Table
x0100 – x01FF	Interrupt Vector Table
x0200 – x2FFF	Operating System and Supervisor Stack
x3000 – xFDFF	Available for User Programs
xFE00 – xFFFF	Device Register Addresses

3
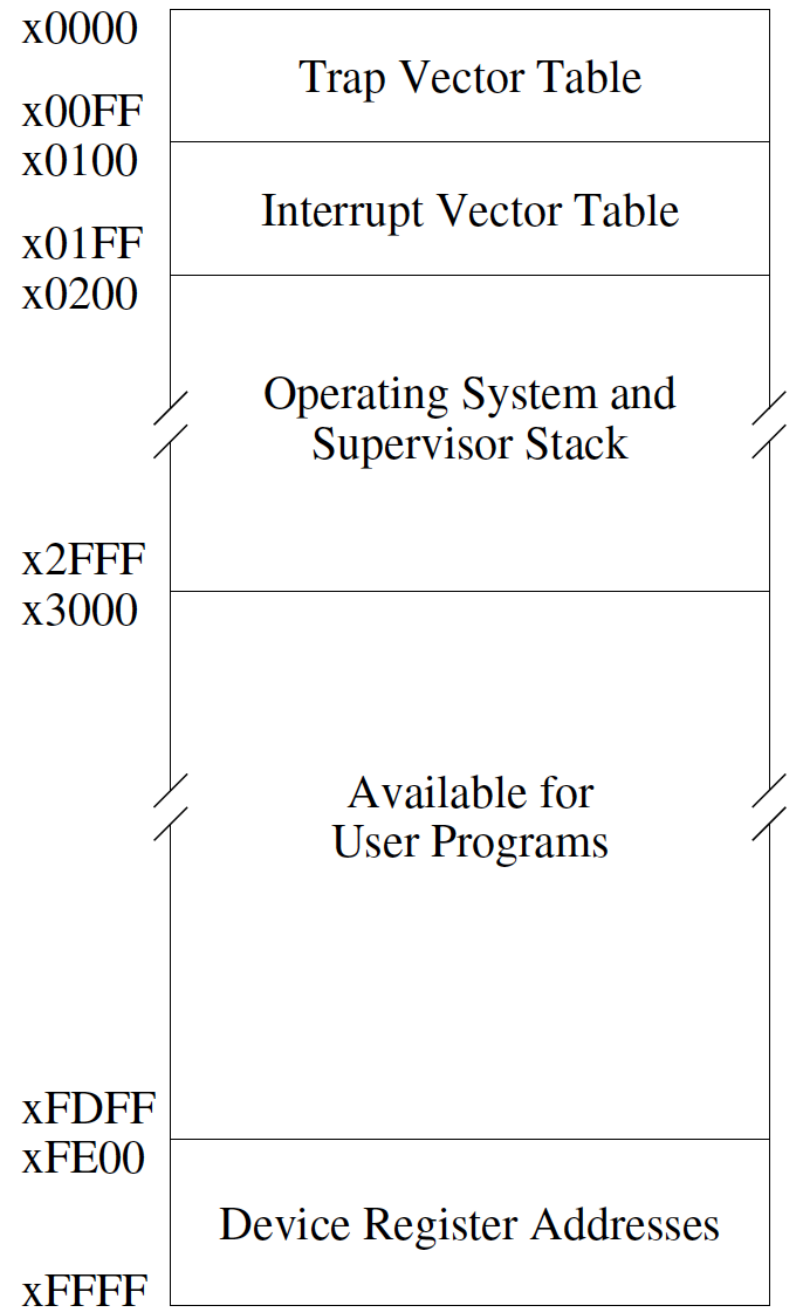
Palindrome Check Using a Stack

A word, phrase, number or other sequence of characters which **reads the same forward or backward**.
- Madam
- Kayak
- Was it a car or a cat I saw
- 123456654321

➢ How can we perform a palindrome check using a stack?

Balanced Parentheses Check Using a Stack

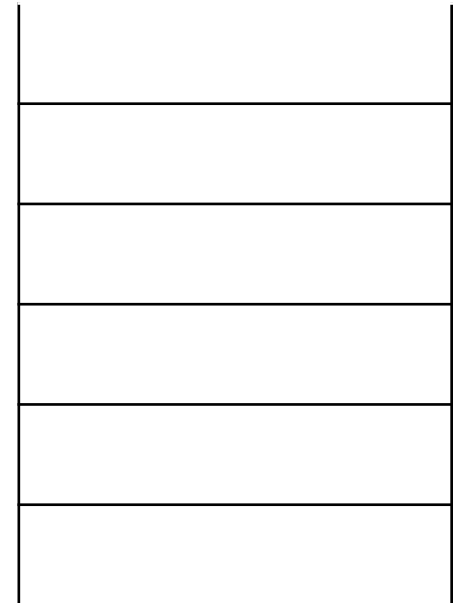Examples of <u>balanced</u> parentheses:

(( ) ( ) ( ) ( ))   (((( ))))   (( ) ((( )) ( )))

Examples of <u>unbalanced</u> parentheses:

((((((( ))   ( )))   (( ) ( ) (( )

Open parenthesis ' **(** ' – _____ to the Stack

Close parenthesis ' **)** ' – _____ from the stack

Assuming the expression would fit into the stack, unbalanced expression can be found under two situations:

1. At the end of the expression –

2. While entering expression –

Postfix Expression (input is single digit operand)

Infix

(3+4)-5

2^(8-4)

7+(9-6)/3

Postfix

34+5-

512+4*+3-

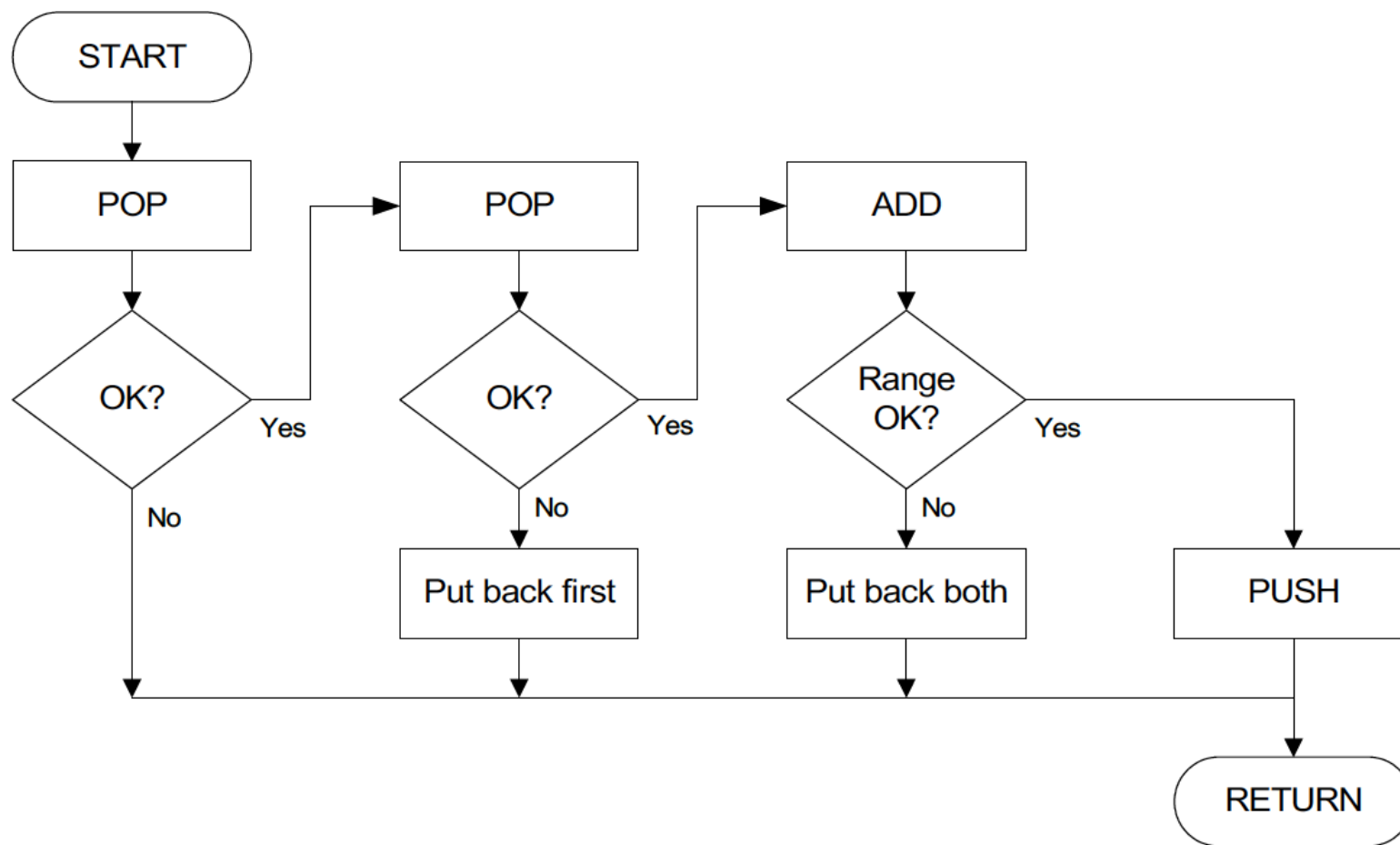Note: '12-' is 1-2 not 2-1

➤ Are these inputs valid postfix expressions? How would your program know?

- 46*-
- 13+57

ECE ILLINOIS

Arithmetic Using a Stack

Implement an ADD subroutine that pops two numbers from a stack and perform the add operation (see flowchart below).

ECE ILLINOIS

Implement ADD Subroutine

- **R6** should be used as stack pointer (points to the **next available spot** on the stack)
- Assume **PUSH**, **POP** and **CHECK_RANGE** subroutines are given & callee-saved

```
; PUSH
; Input: R0 (value to be stored on stack)
; Output: R5 (0 – success, 1 – failure)


; POP
; Output: R0 (value to be loaded from stack)
; Output: R5 (0 – success, 1 – failure)


; CHECK_RANGE: return 0 if value is within -100 to 100 decimal,
; otherwise return 1
; Input: R0 (value to be checked)
; Output: R5 (0 – success, 1 – failure)
```

➢ **What do we need to consider when implementing the ADD subroutine?**

8

ECE ILLINOIS

```
; ADD subroutine – pop two numbers from stack,
; perform '+' operation and then push result back to the stack
; Output: R5 (0 – success, 1 – failure)

; save registers




; Initialize R5


; first pop



; check return value of first pop, go to EXIT if failed (R5 = 1)
```

```
; second pop



; check result of second pop, go to RESTORE_1 if it failed




; add two numbers



; check range of sum, go to RESTORE_2 if it failed




; everything is good, push sum to stack
```

```
RESTORE_1
; put back first number




RESTORE_2
; put back both numbers




EXIT
; restore registers




RET
STACK_START      .FILL x4000
STACK_END        .FILL x3FF0
STACK_TOP        .FILL x4000
```