ECE 220 Computer Systems & Programming

Stack

Yih-Chun Hu
adapted from material by Profs. Yuting Chen, Sanjay Patel, Volodymyr Kindratenko

ECE ILLINOIS

ILLINOIS

Stack – an abstract data type

A LIFO (last-in first-out) storage structure

- **The first thing you put in is the _____ thing you take out**
- **The last thing you put in is the _____ thing you take out**

This means of access is what defines a stack, not the specific implementation.

Two main operations:

PUSH: add an item to the stack

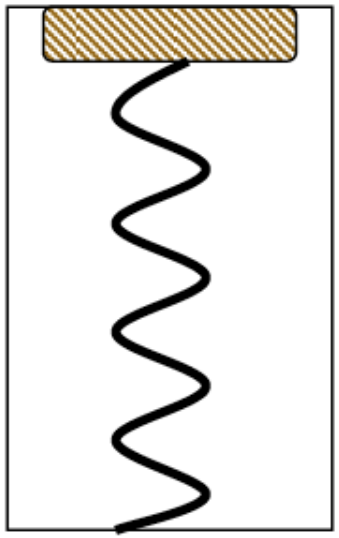
POP: remove an item from the stack

IsFull: check whether the stack is full (_____)

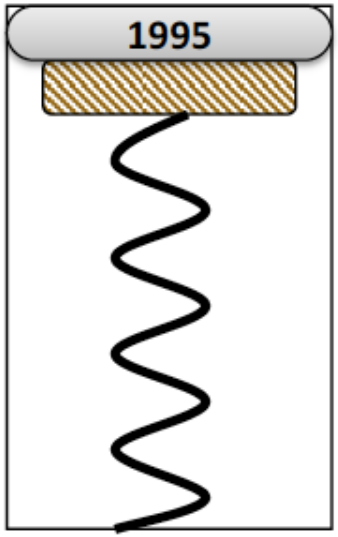
IsEmpty: check whether the stack is empty (_____)

Coin Holder Example

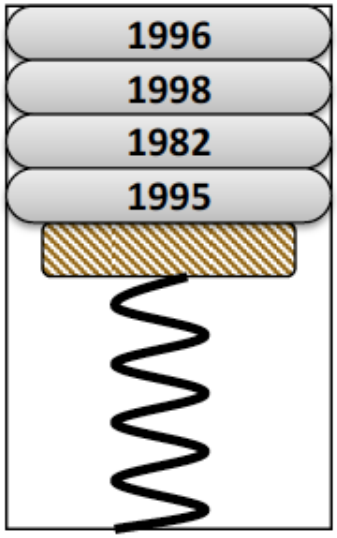
- First coin in is the last coin out



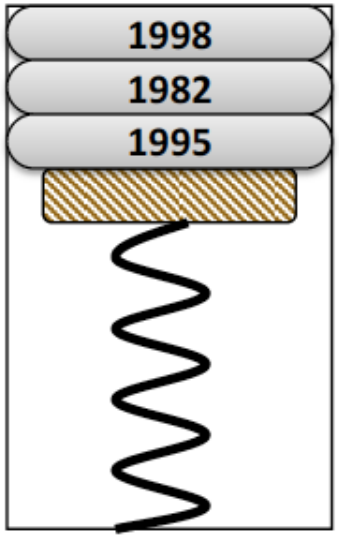
Initial State



After
One Push



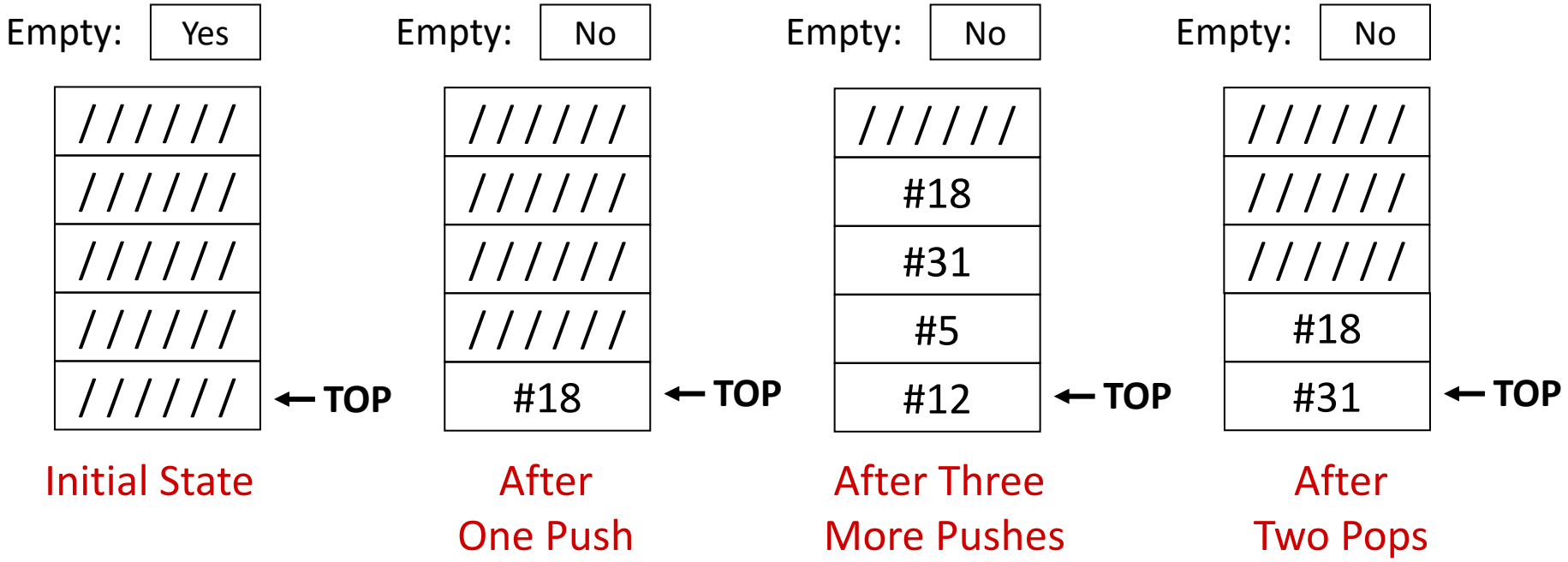
After Three
More Pushes



After
One Pop

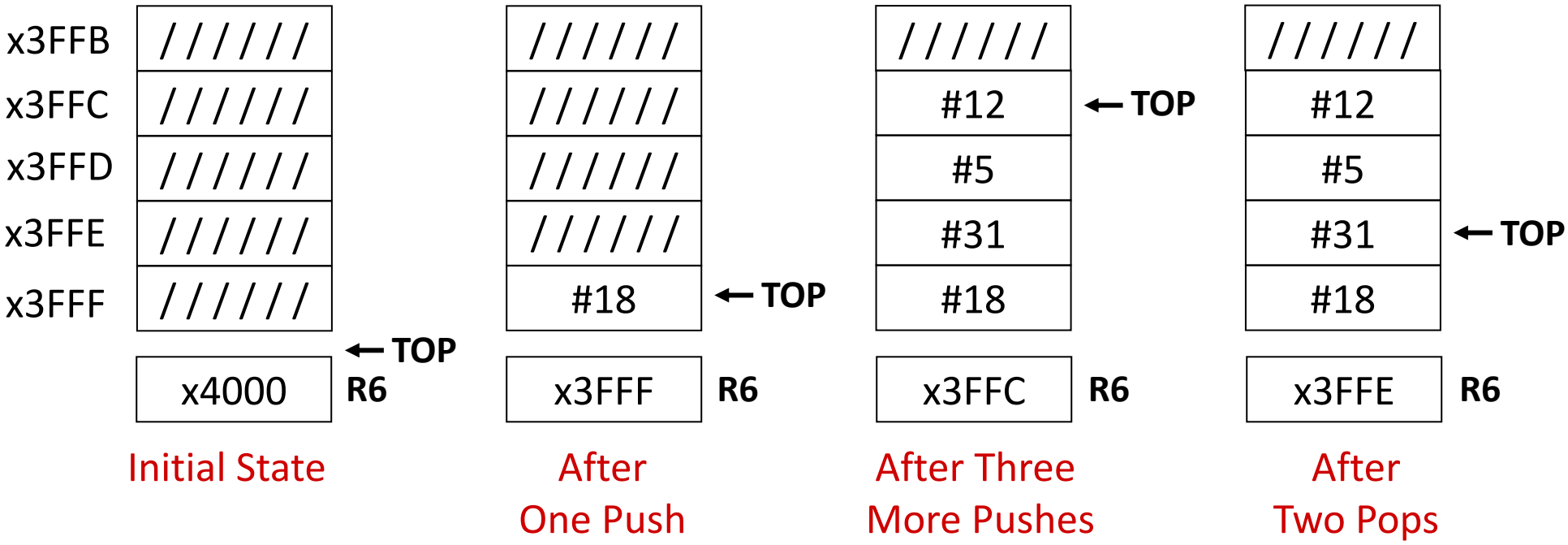
A Hardware Implementation

- **Data items move in memory, top of stack is fixed**



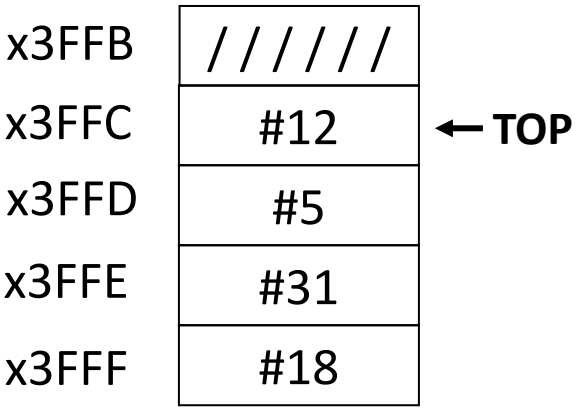
A Software Implementation

- **Data items don't move in memory, just our idea about where the top of the stack is.**



- **By convention, R6 holds the Top of Stack (TOS) pointer**

Basic Push and Pop Code



Using Software Implementation of Stack

- **Push**

```
ADD R6, R6, #-1   ; decrement stack ptr
STR  R0, R6, #0   ; store data (to Top of Stack)
```

- **Pop**

```
LDR  R0, R6, #0   ; load data from stack ptr
ADD R6, R6, #1    ; increment stack ptr
```

Exercise: Input stream – Z Y X W V U T S R

Create a sequence of pushes and pops such that the output stream is

Y X V U W Z S R T

Implement PUSH and POP Subroutines

x3FF0	
x3FF1	
x3FF2	
x3FF3	
x3FF4	
x3FF5	
x3FF6	
x3FF7	
x3FF8	
x3FF9	
x3FFA	
x3FFB	
x3FFC	
x3FFD	
x3FFE	
x3FFF	
x4000	

← STACK_END

← STACK_START

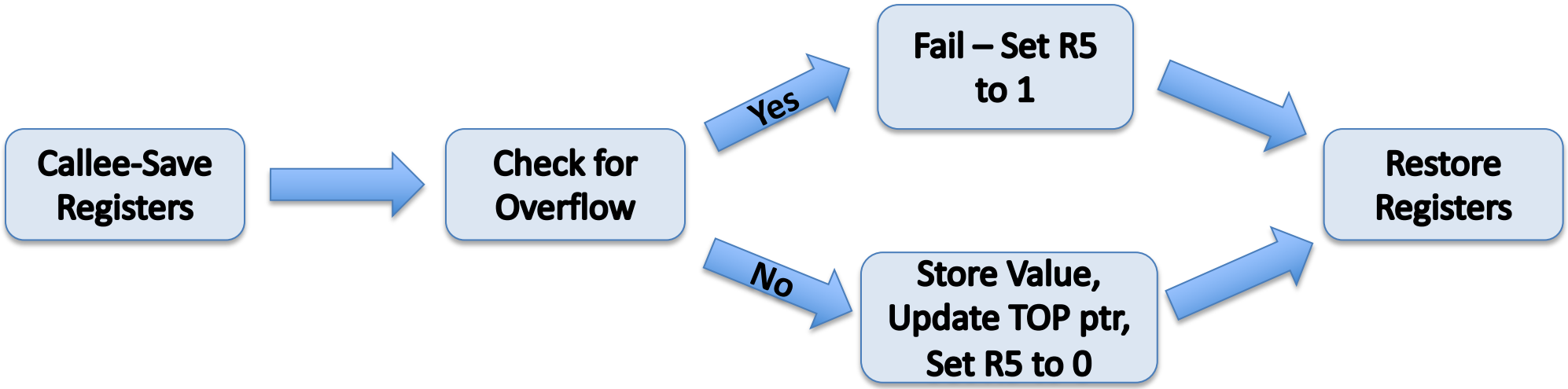
← STACK_TOP (next available spot)

```
.ORIG x3000
…



STACK_START .FILL x4000
STACK_END   .FILL x3FF0
STACK_TOP   .FILL x4000

.END
```




```
; PUSH subroutine
; IN: R0 (value)
; OUT: R5 (0 – success, 1 – fail)
; R3: STACK_END
; R6: STACK_TOP
PUSH
; save registers that will be modified in PUSH subroutine
```

; check for overflow (when stack is full)

; store value in the stack

; indicate the overflow condition on return

; restore modified registers and return

; POP Subroutine

; OUT: R0 (value)

; OUT: R5 (0 – success, 1 – fail)

; R3: STACK_START

; R6: STACK_TOP