香港中文大學
The Chinese University of Hong Kong

CSCI2510 Computer Organization

Tutorial 04: Stack and Queue

Yuhong LIANG

yhliang@cse.cuhk.edu.hk

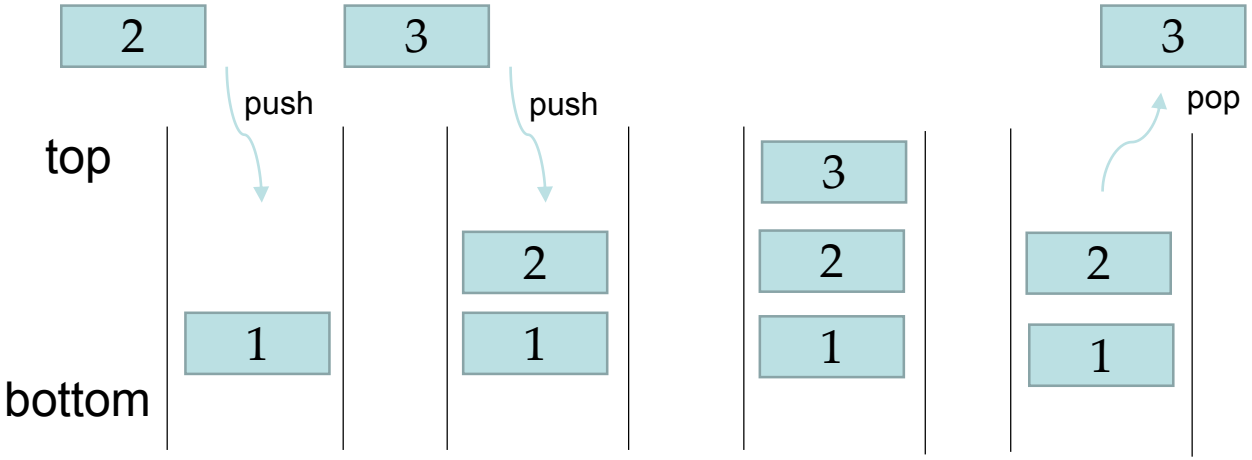
Outline

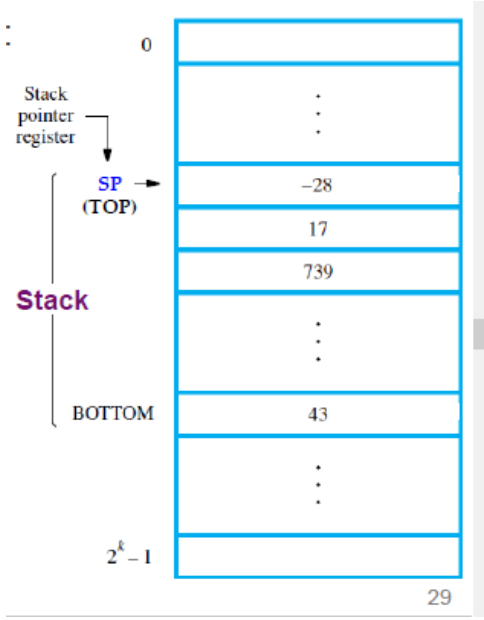
- Basic knowledge of stack and queue
- Implementation of stack and queue
- Hint for assignment3

Basic knowledge of stack and queue

Stack

- push : place data to the top position
- pop: remove data from the top position
- a Last-In-First-Out (LIFO) data structure



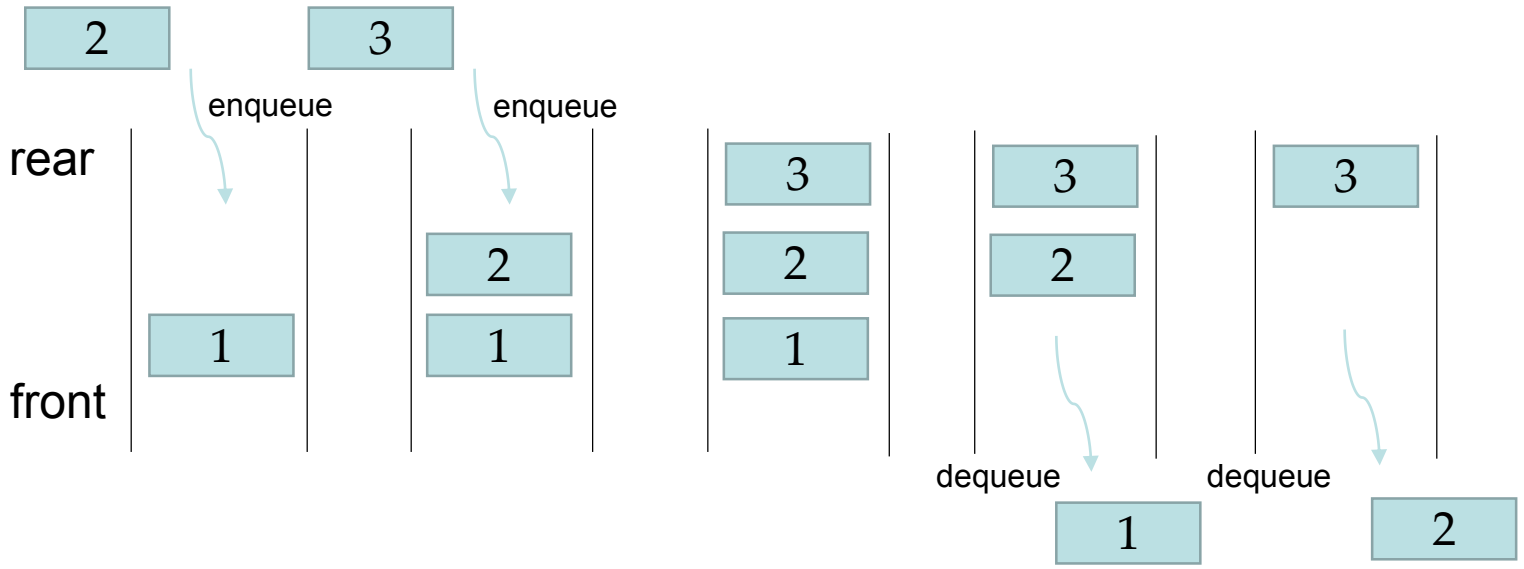


Stack in the memory

Basic knowledge of stack and queue

Queue

- enqueue : place data to the rear position
- dequeue: remove data from the front position
- a First-In-First-Out (FIFO) data structure



Basic knowledge of stack and queue

Syntax of assembly language

mov eax, label ; label represent the address of memory. Load the content in address m to register eax

mov label, eax ; label represent the address of memory. Load the content in register eax to the memory

mov [number] , eax; number represent the address of memory, load the content in eax to the memory

mov eax, [number] , number represent the address of memory, load the content in memory to register eax

add eax, 1; the content of eax increase,e.g. eax = 3 then eax = 3 + 1 = 4

sub eax, 1; the content of eax decrease,e.g. eax = 3 then eax = 3 - 1 = 2

cmp eax, ebx; compare the content of eax, ebx

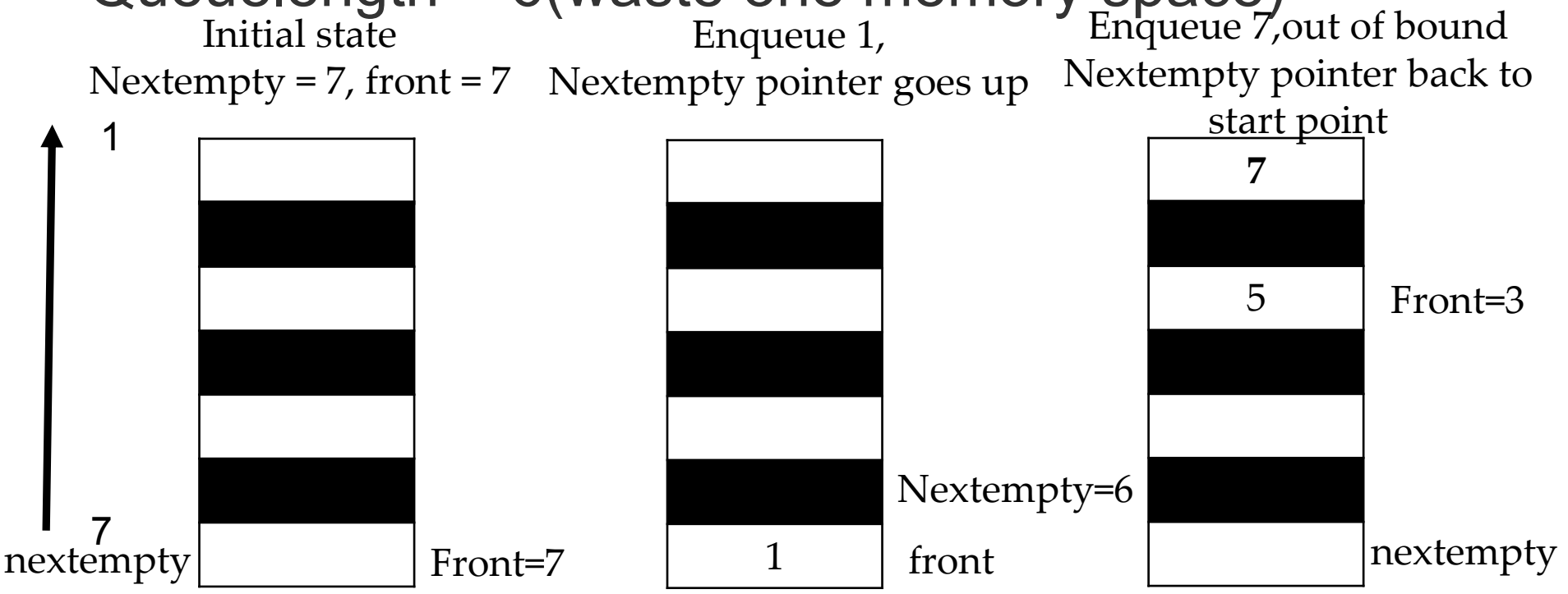
je CODE1 ;if eax is equal ebx, then jump to the CODE1

jmp CODE2 ;jump to CODE2

Implementation of stack and queue

Queue-enqueue

- Front pointer : point to the next data need to dequeue
- Nextempty pointer: point to the empty space
- Queuelength = 6(waste one memory space)



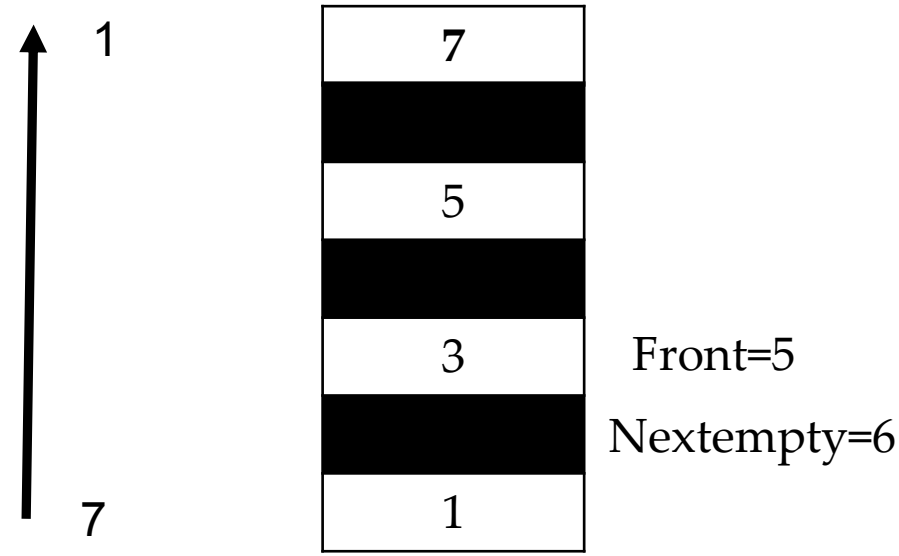
Implementation of stack and queue

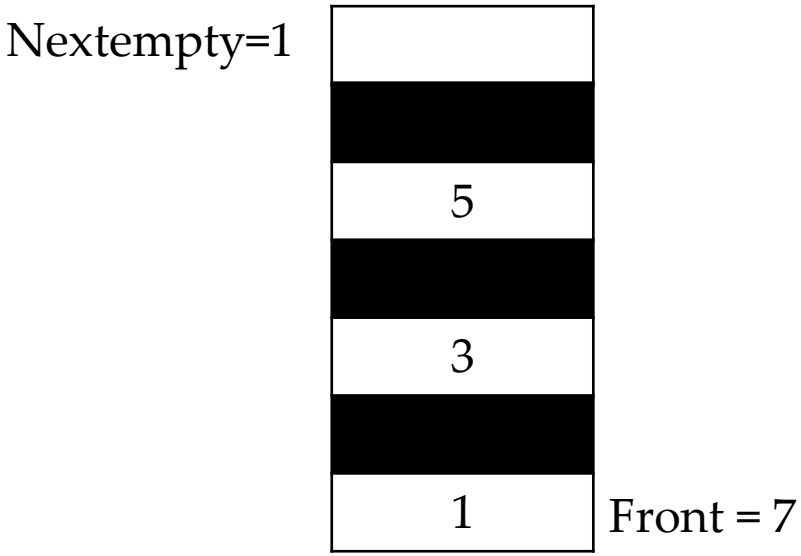
Queue-enqueue

- Front pointer : point to the next data need to dequeue
- Nextempty pointer: point to the empty space

Enqueue, and we get ERROR

If **nextempty = front + 1 or nextempty = front – queuelength** then the queue is full





Implementation of stack and queue

Enqueue

```
enqueue:
    mov EAX, nextempty ; load the nextempty pointer to EAX
    mov EBX, front ; load the front pointer to EAX
    mov ECX, queuelength
    mov EDX, inputnumber
    add EBX, 1 ; see if queue is full(nextempty = front + 1)
    cmp EAX, EBX ; see if queue is full(nextempty = front + 1)
    je enqueueError ; if queue is full, jump to enqueueError
    sub EBX, 1 ; see if queue is full(nextempty = front - queuelength )
    sub EBX, queuelength ; see if queue is full(nextempty = front - queuelength)
    cmp EAX, EBX ; see if queue is full(nextempty = front - queuelength)
    je enqueueError ; if queue is full, jump to enqueueError
    mov [EBP - 4 * 11 + 4 * EAX - 4], EDX ; enqueue the inputnumber to queue in memory
    sub EAX, 1 ; sub the nextempty pointer
    mov nextempty, EAX ; store the pointer to memory
    cmp EAX, 0 ; see if the pointer is out of bound
    je nextemptybacktobegin ; if the pointer is out of bound, move back to the start point
    jmp input
```

If nextempty = front + 1 then the queue is empty,get error

If nextempty = front – queuelength then the queue is empty,get error

Enque the number

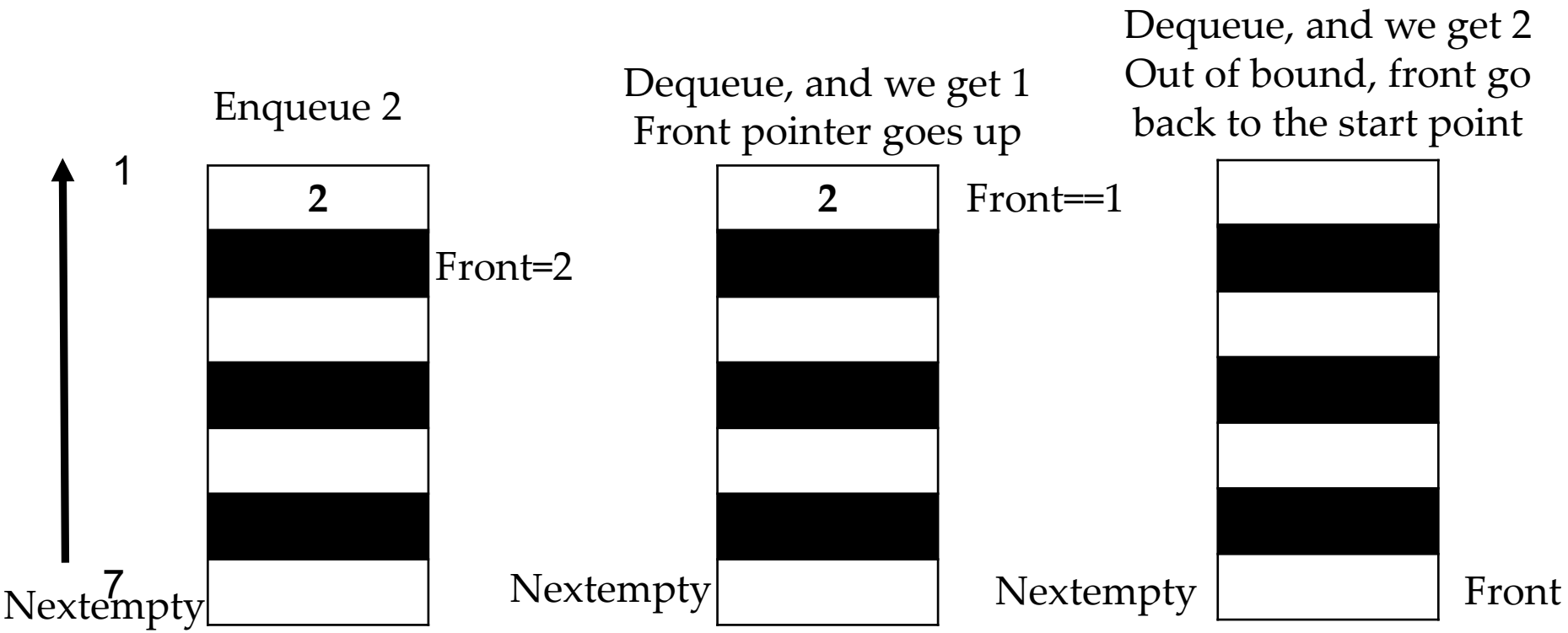
Nextempty pointer goes up

out of bound, go back to start

Implementation of stack and queue

Queue-dequeue

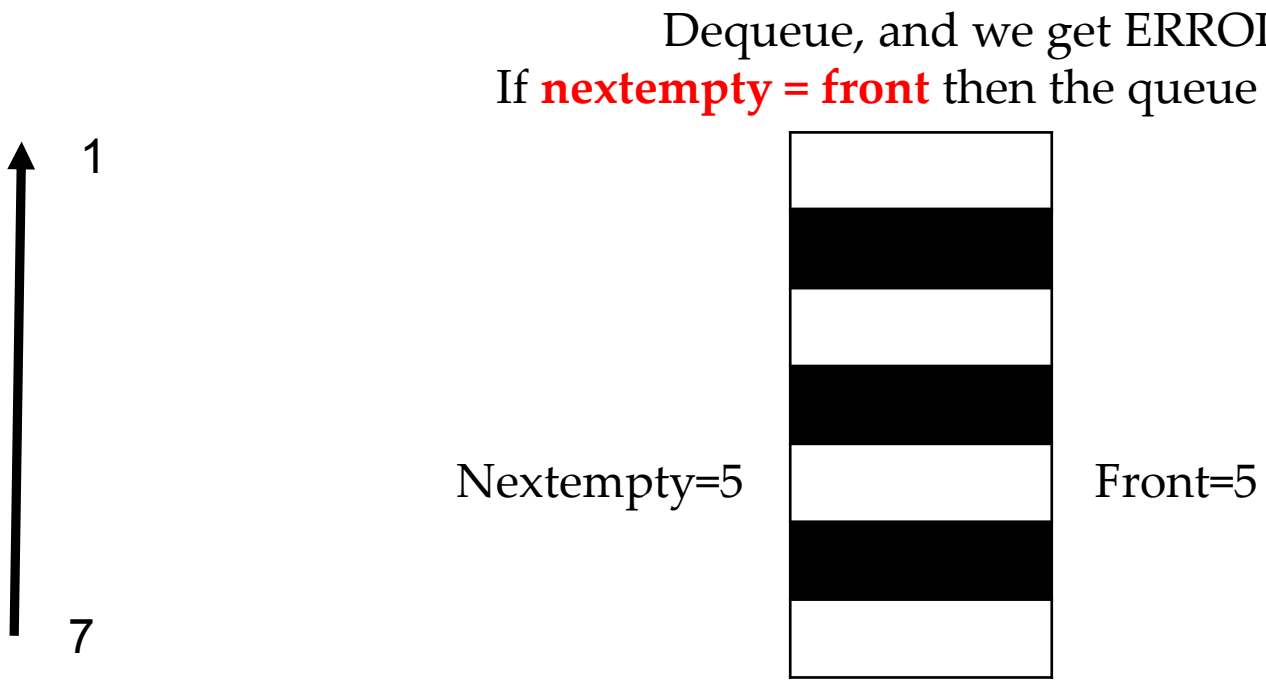
- Front pointer : point to the next data need to dequeue
- Nextempty pointer: point to the empty space



Implementation of stack and queue

Queue-dequeue

- Front pointer : point to the next data need to dequeue
- Nextempty pointer: point to the empty space



Implementation of stack and queue

Dequeue

```
dequeue:
    mov EAX, nextempty
    mov EBX, front
    mov ECX, queuelength
    cmp EAX, EBX ; see if queue is empty(nextempty = front + 1)
    je dequeueError ; if queue is empty, jump to enqueueError
    mov EDX, [EBP - 4 * 11 + 4 * EBX - 4] ; get the front data of the queue, and load it to ECX
    invoke crt_printf, addr outputFormat, EDX ; print the front data of the queue, and load it to ECX
    mov EBX, front
    sub EBX, 1 ; sub the front pointer
    mov front, EBX ; store the pointer to memory
    cmp EBX, 0 ; see if the pointer is out of bound
    je frontbacktobegin ; if the pointer is out of bound, move back to the start point
    jmp input
```

If nextempty = front then the queue is empty

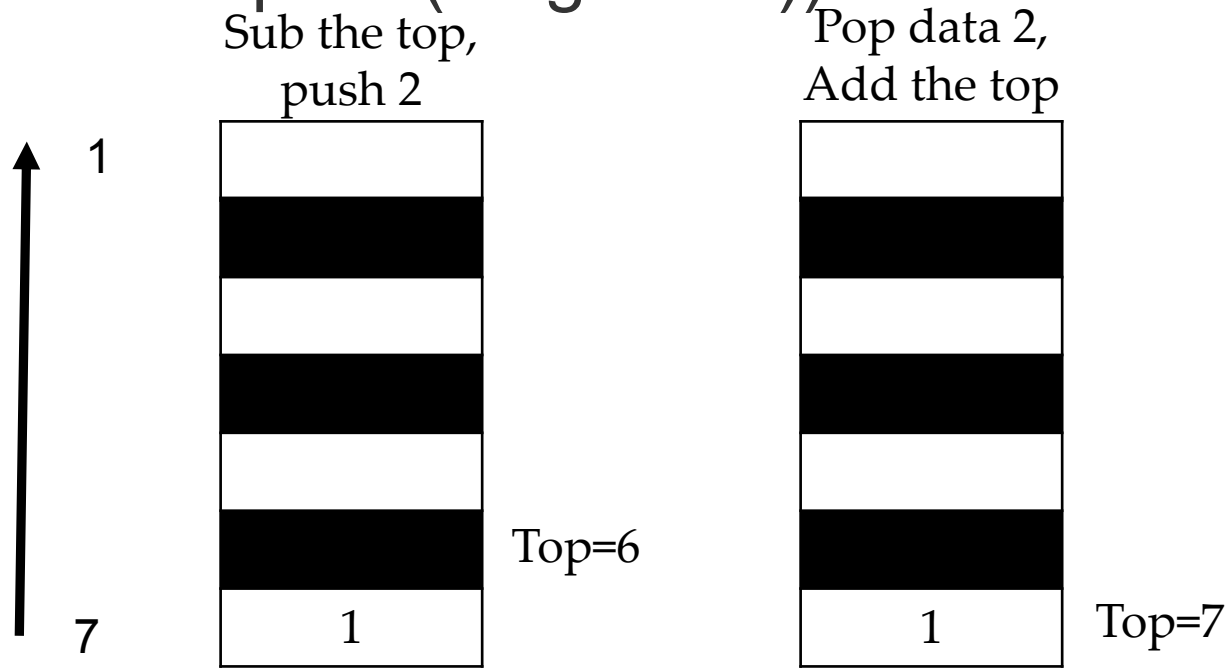
Front pointer goes up

out of bound, go back to start

Implementation of stack and queue

Stack-push & pop

- Top pointer : point to the top data of the stack
- stack length = 7(7 memory space)
- (initial state: top = 8(length + 1))

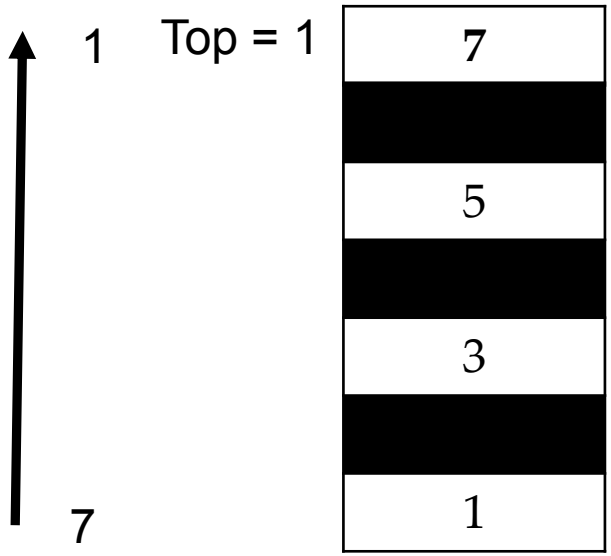


Implementation of stack and queue

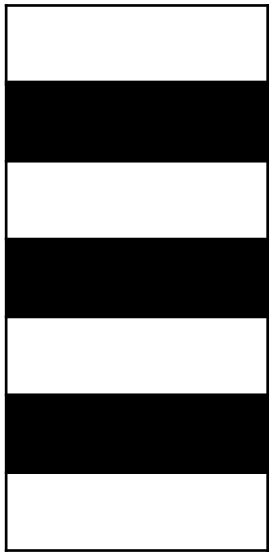
Stack-full & empty

- Top pointer : point to the top data of the stack
- (initial state: top = 8)

Push 9 then get ERROR
If Top = 1
The stack is full



pop, and we get ERROR
If Top = stack length+1
The stack is empty



Top = 8

Hint for assignment2

Homework - Implementation of stack

1. According to the hint, write the 10 code(marked with “fill it”).
2. One hint, one code

```
input:
    invoke crt_printf, addr inputStatement
    invoke crt_scanf, addr numberFormat, addr inputnumber ; input number
    mov ECX, inputnumber ; load the inputnumber in register ECX
    ; compare content of ECX with 0(fill it)
    ; if content of ECX is equal to 0, then jump to popnumber(inputnumber 0 represents action "pop")(fill it)
    ; jmp to pushnumber(fill it)
```

```
pushnumber:
    mov EAX, top ; load the top pointer to EAX
    cmp EAX, 1 ; see if stack is full(when the stack full, top == 1)
    je pusherror ; if stack is full, jump to pusherror
    ; sub the pointer(fill it)
    mov top, EAX ; store the pointer to memory
    mov ECX, inputnumber
    ; push the inputnumber in stack in memory(fill it)
    jmp input
```

Hint for assignment2

Homework - Implementation of stack

1. According to the hint, write the 10 code(marked with “fill it”).
2. One hint, one code

```
popnumber:
    mov EAX, top ; load the top pointer to EAX
    mov EBX, stacklength ; load the stack length to EBX
    add EBX, 1;
    ; see if the stack is empty(when the stack empty, top = stacklength + 1)(fill it)
    ; if the stack is empty jump to poperror(fill it)
    mov ECX, [EBP - 4 * 10 + 4 * EAX - 4] ; get the top data of in the stack in memory, and load it to ECX
    invoke crt_printf, addr outputFormat, ECX ; print out the top data
    mov EAX, top ; load the top pointer to EAX
    ; add the pointer(fill it)
    ; store the pointer to memory(fill it)
    jmp input
```

```
pusherror:
    invoke crt_printf, addr pushErrorStatement ; print error message
    jmp exitprogram

poperror:
    ; print error message(fill it)
    jmp exitprogram
```

Hint for assignment2

Homework - Implementation of stack

1. According to the hint, write the 10 code(marked with “fill it”).
2. One hint, one code

Eg.
1); load the top pointer to EAX(fill it) then you should write one line code
mov EAX top

2); if stack is full, jump to pusherror(fill it) then you should write one line code
je pusherror

Summary

- Stack and Queue

- Implementation of stack and queue