# Introduction to Computer System Organization

SZU Review

Chapter10

# 栈 Stack

## 1. 原始模型

硬币盒

comment:

全体数据一起移动

缺点: 效率低下,无实现意义

两种操作:

(1). push 压入 (入栈)

(2). pop 弹出 (出栈)

## 2. 改良实现 (Last in first out)

comment:

先进后厨,后进先出

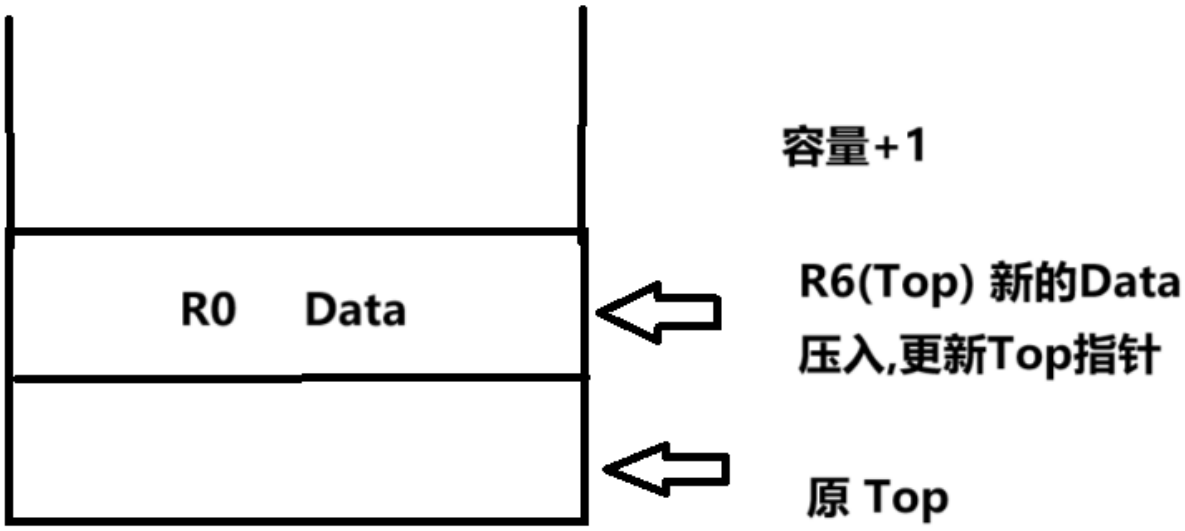
**Top指针:通过移动Top指针,在内存上模拟栈的实现**

栈的数据是一次性有效的

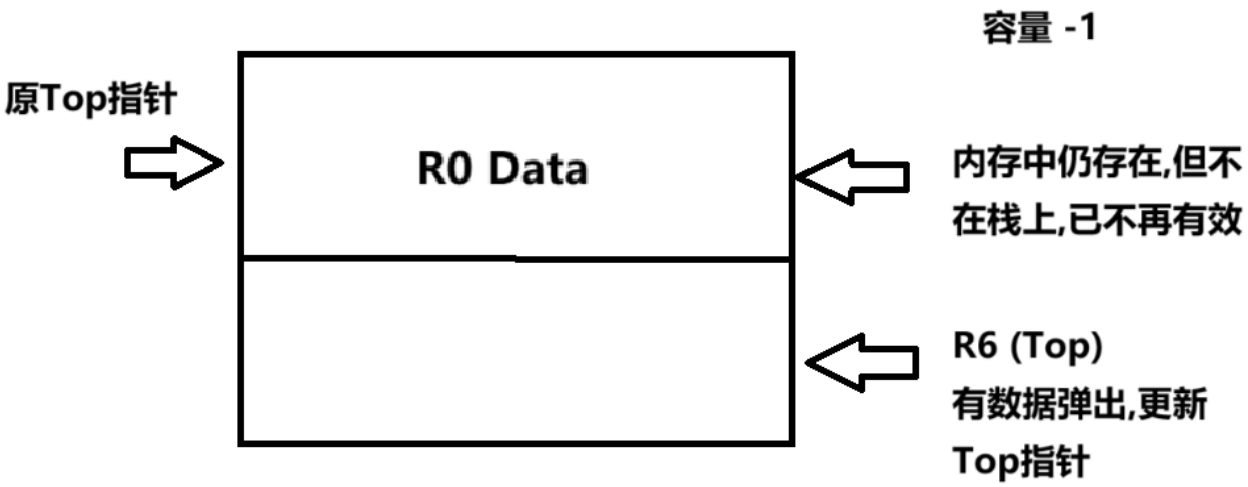
在内存上有效的数据(一直有效)在栈上无效(读过了,Top指针不再指向这里)

## 3. Push 与 Pop 操作的实现指令

```
Push
    ADD R6,R6.#-1
    STR R0,R6,#0
```




```
POP
    LDR R0,R6,#0
    ADD R6,R6,#1
```




## 4. 判断栈空与栈满

**Pop前检查stack是否为empty**

**Push前检查stack是否为full**

**用 R6 寄存器来专门存储 Top 指针.**

## (1). 栈空判断

R6 = endaddress + 1 ~ 栈为空 (栈是向上生长的)

R6 = startaddress - 1 ~栈为空 (栈是向下生长的)

支持下溢出检测的 pop 操作.

入口 R6 ,返回 R0 , R5

**R5 的状态:**

**state 0: success ; state 1: overflow**

;在执行弹出操作前检查是否下溢出(栈空)

## (2).栈满判断

R6 = startaddress ~ 栈为满 (栈是向上生长的)

R6 = endaddress ~ 栈为满 (栈是向下生长的)

# 在LC-3上实现栈:

```
;初始化栈
.ORIG       x3000
LD  R6, STACK_BASE      ;初始化栈指针R6指向栈底
;示例使用
LD  R0, VALUE1          ;准备要压入的值VALUE1
JSR PUSH                ;调用PUSH
LD  R0,VALUE2           ;准备要压入的值VALUE2
JSR PUSH                ;调用PUSH
JSR POP                 ;将压入的值VALUE2弹出,调用POP
JSR POP                 ;将压入的值VALUE1弹出, 调用POP
;Feature: Last in first out .先进后出,后进先出 .
HALT
;栈的存储区域
STACK_BASE .FILL x4000  ;栈底地址(本例中栈向低地址增长)
STACK_MAX  .FILL x3FFF  ;栈最大地址
VALUE1     .FILL x1234  ;测试值1
VALUE2     .FILL x5678  ;测试值2

;PUSH   子程序
;输入: R0---要压入的值
;使用: R6---栈指针 (SP:Stack-Pointer)
PUSH    ADD R6,R6,#-1   ;栈指针减1(栈向低地址增长)
        STR R0,R6,#0    ;存储值到栈顶
        RET

;POP    子程序
;输出: R0---弹出的值
;使用  R6---栈指针 (SP:Stack-Pointer)
POP     LDR  R0,R6,#0   ;从栈顶加载值
        ADD  R6,R6,#1   ;栈指针加1
        RET
      .END
```

## 栈空检测:可以在pop之前检查栈是否为空,防止下溢:

```
POP    LD   R1,STACK_BASE
       NOT  R1,R1
       ADD  R1,R1,#1
       ADD  R1,R6,R1
       BRz  STACK_EMPTY
       LDR  R0,R6,#0
       ADD  R6,R6,#1
       AND  R5,R5,#0

STACK_EMPTY AND R5,R5,#0
            ADD R5,R5,#1
            RET
```

## 栈满检测:可以在PUSH之前检查栈是否已满,防止上溢:

```
PUSH    LD  R1,STACK_MAX
        NOT R1,R1
        ADD R1,R1,#1
        ADD R1,R6,R1
        BRz STACK_FULL
        ADD R6,R6.#-1
        STR R0,R6,#0
        AND R5,R5,#0

STACK_FULL  AND R5,R5,#0
            ADD R5,R5,#1
            RET
```