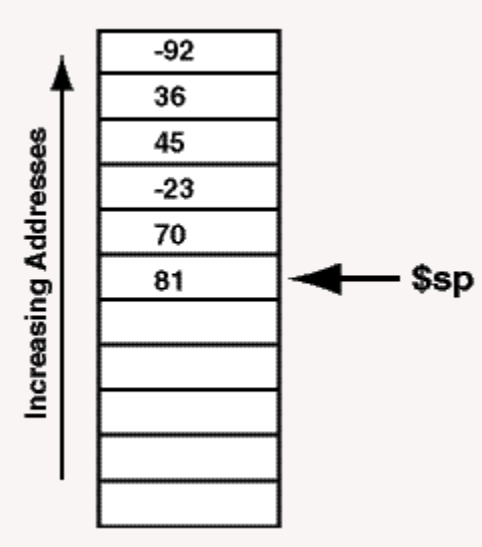
# NGĂN XẾP (STACK)

## 1. Stack

- Stack = LIFO (Last in - First out): Ngăn xếp là vùng nhớ được truy cập theo cơ chế như một băng đạn, nghĩa là phần tử "vào trước ra sau" hoặc "vào sau ra trước".

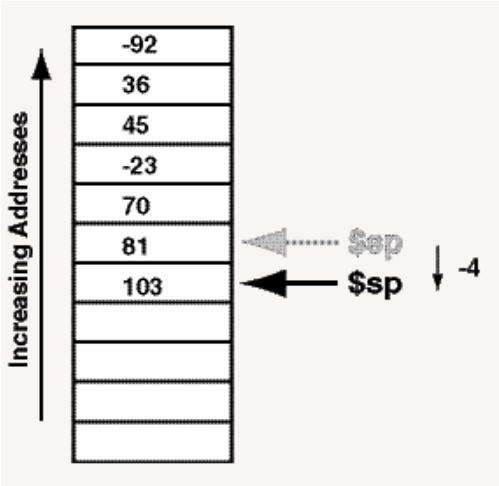



- Mỗi phần tử trong bộ nhớ có kích thước một word (4 bytes)
- Thanh ghi $sp (stack pointer) là thanh ghi luôn luôn trỏ đến đỉnh của stack. Đỉnh stack luôn có địa chỉ thấp hơn.
- Trong Stack ta có 2 thao tác cơ bản
  - Thêm dữ liệu vào stack: PUSH
  - Lấy 1 phần tử ra khỏi Stack: POP

VD1: Giả sử đỉnh stack đang trỏ vào số 81 trong Stack (như hình trên). Nếu ta muốn thêm 1 phần tử 103 mới vào Stack phải giảm địa chỉ con trỏ $sp xuống 4 bytes, sau đó PUSH giá trị mới vào Stack.

```
li $t0, 103          # Load giá trị muốn thêm mới vào thanh ghi tạm $t0
subu  $sp,$sp,4      # Giảm địa chỉ $sp xuống 4 bytes, để trỏ vào đỉnh của stack mới
sw $t0, ($sp)
```

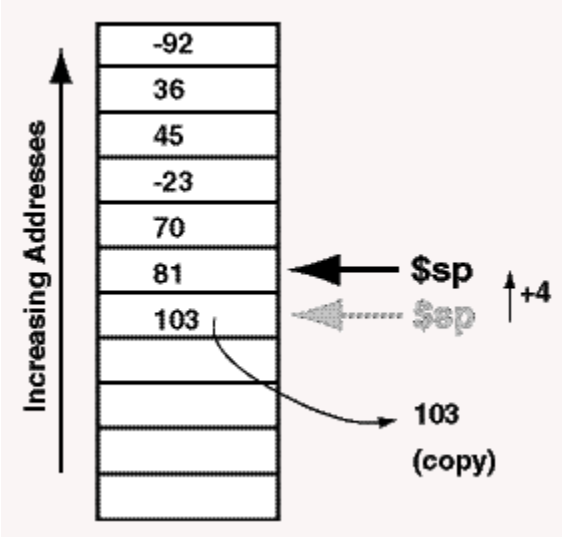
(Xem hình minh hoạ)




Vd2: Muốn lấy giá trị trong đỉnh của Stack ra

lw $t1,($sp)  # Lấy giá trị tại đỉnh của Stack, gán vào $t1

addu $sp,$sp,4  # Tăng lên 4 bytes để con trỏ $sp trỏ vào đỉnh của stack mới




VD3: Viết chương trình đọc các phần tử trong mảng đã cho trước và lưu vào stack; Lấy các phần tử trong Stack lưu trở lại mảng; Xuất các phần tử trong mảng mới ra màn hình.
(VD: [2,3,4,5,6] ⇒ [6,5,4,3,2])

| |
|---|
| 2 |
| 3 |
| 4 |

| |
|---|
| 5 |
| 6 |

```
.data

        array:      .word 2, 3, 4, 5, 6
        N:          .word 5
        newspace:   .asciiz ", "

.text
.globl main
main:
        # Khoi tao ban dau
        la      $t0, array          # Load array address
        li      $t1, 0              # i=0
        lw      $t2, N              # size of array

        #Step1:Loop to read items from array to stack
        pushLoop:
                lw $t4, ($t0)               # get array[i]
                subu $sp, $sp, 4            # $sp: stack pointer register
                sw $t4, ($sp)               # push to stack

                add $t1, $t1, 1             # i=i+1
                add $t0, $t0, 4             # update array address
                blt $t1, $t2, pushLoop

        # Khoi tao ban dau
        la      $t0, array          # Load array address
        li      $t1, 0              # i=0
        lw      $t2, N              # size of array

        #Step2:Loop to pop items from stack to array
        popLoop:
                lw $t4, ($sp)       # Get value from stack to t4
                addu $sp, $sp, 4    # Tang stack
```

```
        sw $t4, ($t0)       # Store to array
        add $t1, $t1, 1             # i=i+1
        add $t0, $t0,4              # Update array address
        blt $t1, $t2, popLoop

    #Step3: print reverse array
    la      $t0, array          # Load array address
    li      $t1, 0              # i=0
    lw      $t2, N              # size of array
    printLoop:
        li      $v0, 1
        lw      $a0, ($t0)
        syscall

        #print ", "
        li      $v0,4
        la      $a0, newspace
        syscall

        addi $t1, $t1, 1            # i=i+1
        addi $t0, $t0, 4            # tang dia chi array
        bne $t1, $t2, printLoop

    #exit:
    li $v0, 10
    syscall
.end main
```

**Bài tập tự thực hành:**

1/ Viết chương trình nhập vào 1 mảng gồm N số nguyên bất kỳ (N nhập từ bàn phím), xuất mảng theo thứ tự ngược lại.

2/ Viết chương trình nhập vào 2 số nguyên a, b. Xuất ra màn hình kết quả của biểu thức: **ab - 12b + 7a** (Dùng stack)

*Gợi ý:* - Bước 1: tính a*b: push KQ vào stack

- Bước 2: tính -12*b: push KQ vào stack
- Bước 3: tính 7*a: push KQ vào stack

- Sau đó POP từng giá trị trong Stack và cộng dồn vào ta sẽ được kết Quả của biểu thức

3/ Viết chương trình nhập vào một chuỗi ký tự, xuất ra chuỗi ngược lại, VD: ABCDEF, xuất ra: FEDCBA

4/ Viết chương trình nhập vào một số nguyên N (VD: N=123). Xuất ra màn hình số ngược lại (VD: 321).