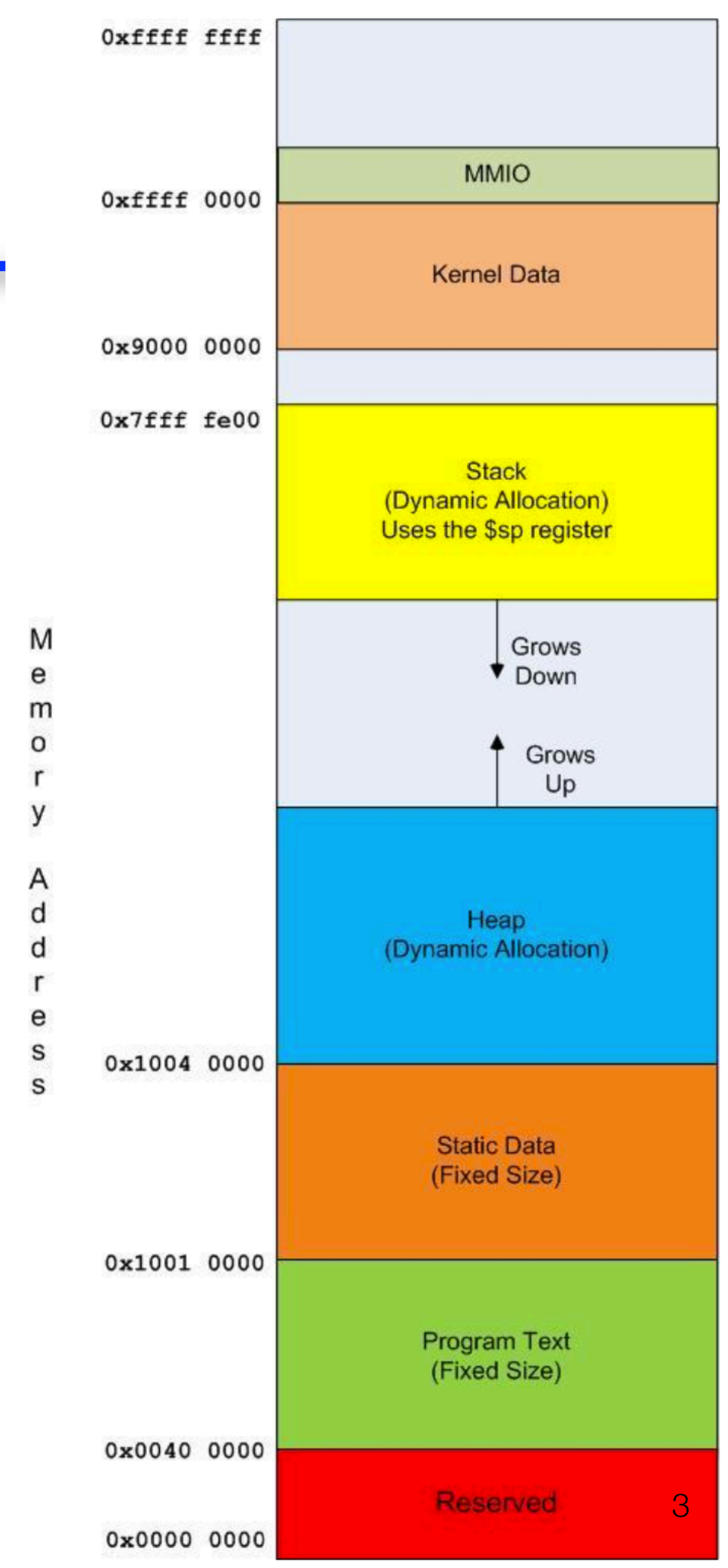
Thực hành ICT4

TS. Nguyễn Thị Thanh Nga
Bộ môn KTMT
Viện CNTT&TT

Tuần 8

Bộ nhớ

- Mô hình địa chỉ 32 bit phẳng
- Có thể đánh địa chỉ cho 4GB dữ liệu
- Bắt đầu từ địa chỉ 0x00000000 đến 0xffffffff
- Không phải tất cả bộ nhớ đều khả dụng



Cấu trúc ngăn xếp dữ liệu

- Là một cấu trúc dữ liệu LIFO (Last-In-First_Out)
- Được thực hiện như là một mảng có 2 thao tác: push và pop
- Thao tác push đặt một phần tử lên đỉnh ngăn xếp, và do đó đặt phần tử đó vào vị trí khả dụng tiếp theo trong mảng, tăng thêm 1 vào chỉ số mảng để trỏ đến vị trí khả dụng tiếp theo.
- Thao tác pop xoá phần tử trên cùng của ngăn xếp, và trừ 1 khỏi kích thước của ngăn xếp.

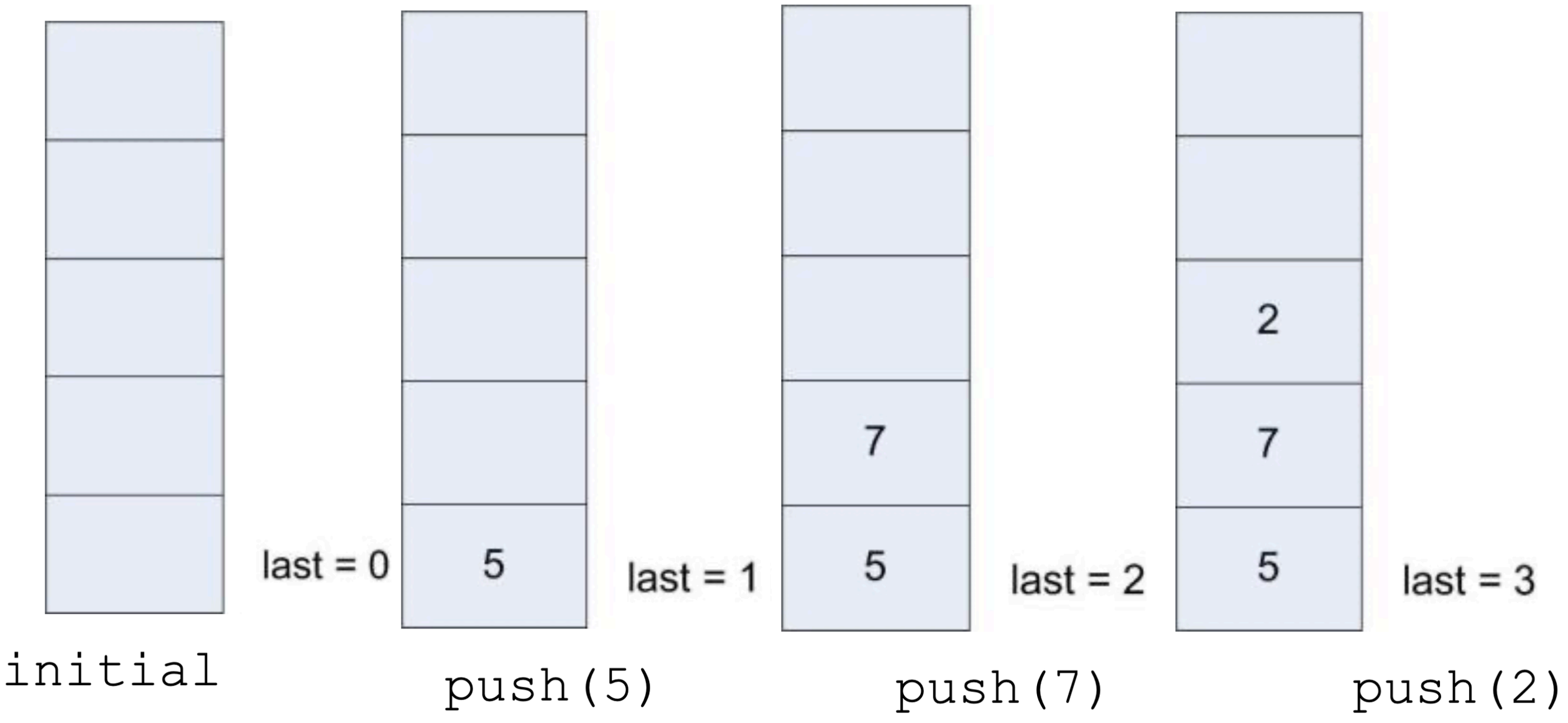
Cấu trúc ngăn xếp dữ liệu

- Push
- Pop

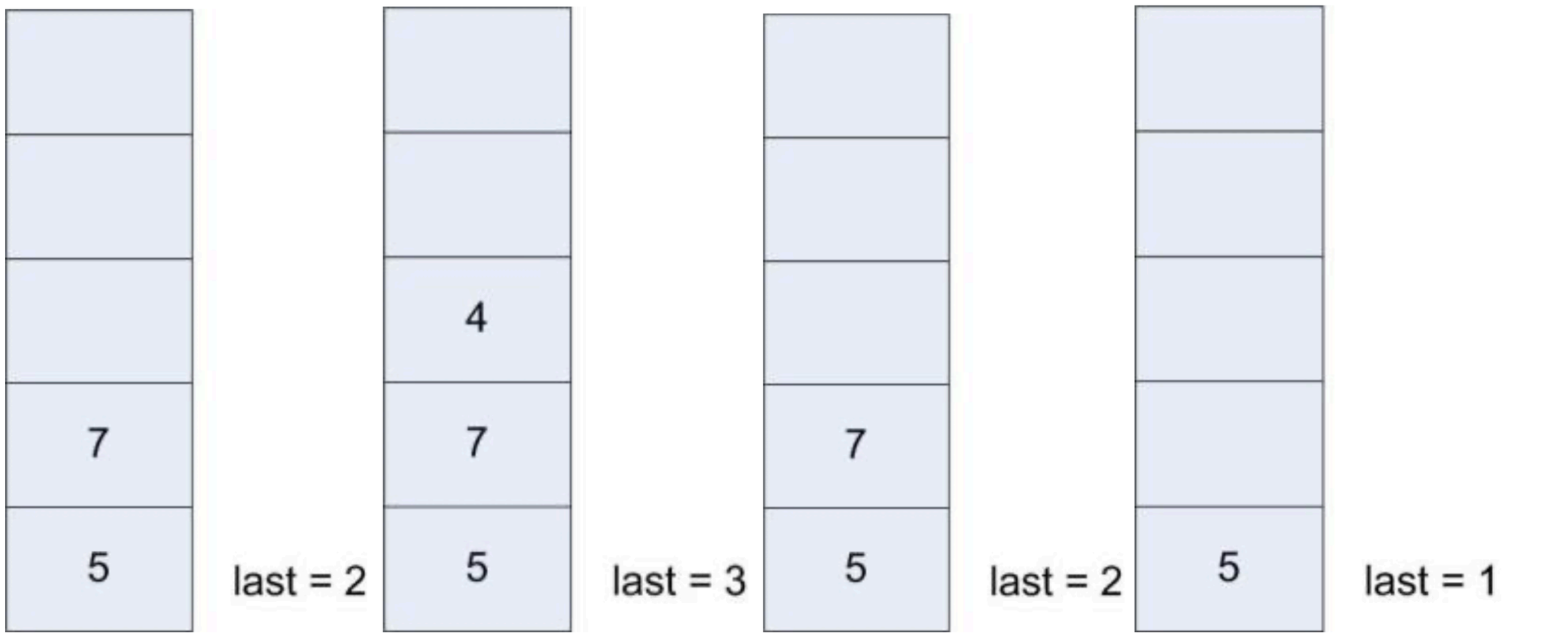
```
class Stack
{
        int SIZE=100;
        int elements[SIZE];
        int last = 0;
        push(int newElement)
        {
            elements[last] = newElement;
            last = last + 1
        }
        int pop()
        {
            last = last - 1;
            return element[last];
        }
}
```

```
push(5)
push(7)
push(2)
print(pop())
push(4)
print(pop())
print(pop())
```

Cấu trúc ngăn xếp dữ liệu



Cấu trúc ngăn xếp dữ liệu



- Output: 2, 4, 7

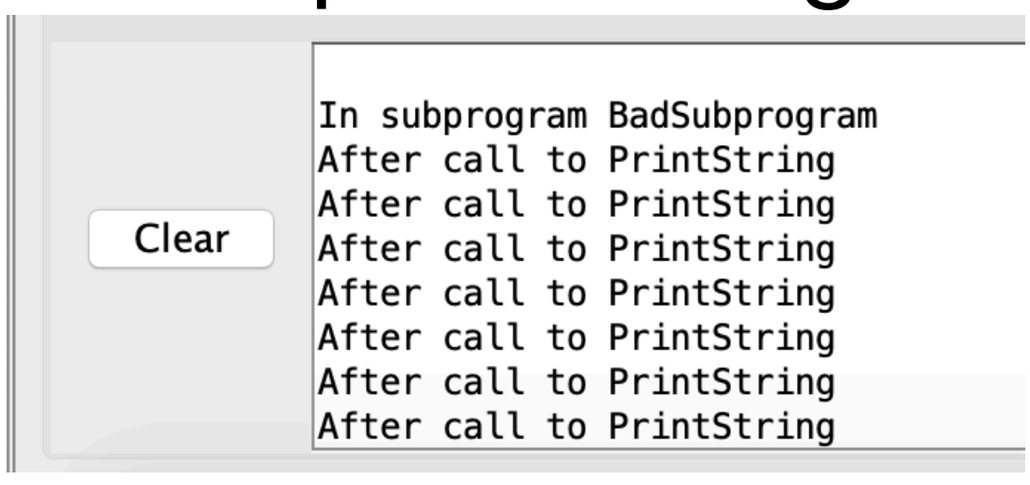
Ngăn xếp chương trình

```
1   .text
2   .globl main
3   main:
4               jal BadSubprogram
5               la $a0, string3
6               jal PrintString
7   jal Exit
8
9   BadSubprogram:
10              la $a0, string1
11              jal PrintString
12              li $v0, 4
13              la $a0, string2
14              syscall
15              jr $ra
16  .data
17  string1: .asciiz "\nIn subprogram BadSubprogram\n"
18  string2: .asciiz "After call to PrintString\n"
19  string3: .asciiz "After call to BadSubprogram\n"
20  .include "utils.asm"
```

- Kết quả muốn in:

```
In subprogram BadSubprogram
After call to PrintString
After call to example
```

- Kết quả chương trình



Ngăn xếp chương trình

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x0c100005	jal 0x00400014	4: jal BadSubprogram
☐	0x00400004	0x3c011001	lui $1,0x00001001	5: la $a0, string3
☐	0x00400008	0x34240039	ori $4,$1,0x00000039	
☐	0x0040000c	0x0c10001e	jal 0x00400078	6: jal PrintString
☐	0x00400010	0x0c100021	jal 0x00400084	7: jal Exit
☐	0x00400014	0x3c011001	lui $1,0x00001001	9: la $a0, string1
☐	0x00400018	0x34240000	ori $4,$1,0x00000000	
☑	0x0040001c	0x0c10001e	jal 0x00400078	10: jal PrintString
☑	0x00400020	0x24020004	addiu $2,$0,0x00000004	11: li $v0, 4

Labels

Label	Address ▲
(global)	
main	0x00400000
mips8-3.asm	
BadSubprogram	0x00400014
PrintNewLine	0x00400034
PrintInt	0x00400048
PromptInt	0x00400060
PrintString	0x00400078

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x206e490a	0x70627573	0x72676f72	0x42206d61	0x75536461	0x6f727062	0x6d617267	0x6641000a
0x10010020	0x20726574	0x6c6c6163	0x206f7420	0x6e697250	0x72745374	0x0a676e69	0x74664100	0x63207265
0x10010040	0x206c6c61	0x42206f74	0x75536461	0x6f727062	0x6d617267	0x000a000a	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages Run I/O

```
Assemble: assembling /Users/ngantt/Dropbox/Baigiang/IT3280/mips8-3.asm

Assemble: operation completed successfully.

Go: running mips8-3.asm

Go: execution paused at breakpoint: mips8-3.asm
```

Clear

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400004
pc		0x0040001c
hi		0x00000000
lo		0x00000000

Ngăn xếp chương trình

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x0040000c	0x0c10001e	jal 0x00400078	6: jal PrintString
☐	0x00400010	0x0c100021	jal 0x00400084	7: jal Exit
☐	0x00400014	0x3c011001	lui $1,0x00001001	9: la $a0, string1
☐	0x00400018	0x34240000	ori $4,$1,0x00000000	
☑	0x0040001c	0x0c10001e	jal 0x00400078	10: jal PrintString
☑	0x00400020	0x24020004	addiu $2,$0,0x00000004	11: li $v0, 4
☐	0x00400024	0x3c011001	lui $1,0x00001001	12: la $a0, string2
☐	0x00400028	0x3424001e	ori $4.$1.0x0000001e	

Labels

Label	Address ▲
(global)	
main	0x00400000
mips8-3.asm	
BadSubprogram	0x00400014
PrintNewLine	0x00400034
PrintInt	0x00400048
PromptInt	0x00400060

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x206e490a	0x70627573	0x72676f72	0x42206d61	0x75536461	0x6f727062	0x6d617267	0x6641000a
0x10010020	0x20726574	0x6c6c6163	0x206f7420	0x6e697250	0x72745374	0x0a676e69	0x74664100	0x63207265
0x10010040	0x206c6c61	0x42206f74	0x75536461	0x6f727062	0x6d617267	0x000a000a	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400020
pc		0x00400020
hi		0x00000000
lo		0x00000000

Mars Messages Run I/O

```
Assemble: assembling /Users/ngantt/Dropbox/Baigi
Assemble: operation completed successfully.
Go: running mips8-3.asm
Go: execution paused at breakpoint: mips8-3.asm
Go: running mips8-3.asm
Go: execution paused at breakpoint: mips8-3.as
```

Clear

Thanh ghi $ra bị ghi đè → mất liên kết để quay trở về chương trình chính

Thanh ghi $ra cần phải được lưu lại khi vào chương trình con và được khôi phục lại trước khi kết thúc chương trình con

Ngăn xếp chương trình

```
1   .text
2   .globl main
3   main:
4           jal GoodSubprogram
5           la $a0, string3
6           jal PrintString
7           jal Exit
8
9   GoodSubprogram:
10          addi $sp, $sp, -4       # save space on the stack (push) for the $ra
11          sw $ra, 0($sp)          # save $ra
12          la $a0, string1
13          jal PrintString
14
15          li $v0, 4
16          la $a0, string2
17          syscall
18
19          lw $ra, 0($sp)          # restore $ra
20          addi $sp, $sp, 4        # return the space on the stack (pop)
21          jr $ra
22  .data
23  string1: .asciiz "\nIn subprogram GoodExample\n"
24  string2: .asciiz "After call to PrintString\n"
25  string3: .asciiz "After call to GoodExample\n"
26  .include "utils.asm"
```

Ngăn xếp chương trình

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400004	0x3c011001	lui $1,0x00001001	5: la $a0, string3
	0x00400008	0x34240037	ori $4,$1,0x00000037	
	0x0040000c	0x0c100022	jal 0x00400088	6: jal PrintString
	0x00400010	0x0c100025	jal 0x00400094	7: jal Exit
	0x00400014	0x23bdfffc	addi $29,$29,0xfffffffc	10: addi $sp, $sp, -4 # save spac...
	0x00400018	0xafbf0000	sw $31,0x00000000($29)	11: sw $ra, 0($sp) # save $ra
☑	0x0040001c	0x3c011001	lui $1,0x00001001	12: la $a0, string1

Labels

Label	Address ▲
(global)	
main	0x00400000
mips8-4.asm	
GoodSubprogram	0x00400014
PrintNewLine	0x00400044
PrintInt	0x00400058
PromptInt	0x00400070

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffefe0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00400004	0x00000000
0x7ffff000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

current $sp ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeff8
$fp	30	0x00000000
$ra	31	0x00400004
pc		0x0040001c
hi		0x00000000
lo		0x00000000

Mars Messages Run I/O

Clear

```
Assemble: assembling /Users/ngantt/Dropbox/Baigiang/IT3280/mips8-4.asm

Assemble: operation completed successfully.

Go: running mips8-4.asm

Go: execution paused at breakpoint: mips8-4.asm
```

Ví dụ 1 – In mảng số nguyên

- In mảng số nguyên:

```
Subprogram PrintIntArray(array, size)
{
    print("[")
    for (int i = 0; i < size; i++)
    {
        print("," + array[i])
    }
    print("]")
}
```

Ví dụ 1 – In mảng số nguyên

- Sv nhập chương trình sau, biên dịch, chạy, quan sát và giải thích sự thay đổi của $sp:

```
.text
.globl main
main:
    la $a0, array_base
    lw $a1, array_size
    jal PrintIntArray
    jal Exit

.data
    array_size: .word 5
    array_base:
                .word 12
                .word 7
                .word 3
                .word 5
                .word 11

.text
# Subprogram: PrintIntArray
# Purpose: print an array of ints
# inputs: $a0 - the base address of the array
# $a1 - the size of the array
#
PrintIntArray:
    addi $sp, $sp, -16  # Stack record
    sw $ra, 0($sp)
    sw $s0, 4($sp)
    sw $s1, 8($sp)
    sw $s2, 12($sp)

    move $s0, $a0               # save the base of the array to $s0


    # initialization for counter loop
    # $s1 is the ending index of the loop
    # $s2 is the loop counter
    move $s1, $a1
    move $s2, $zero

    la $a0 open_bracket         # print open bracket
    jal PrintString

loop:
    # check ending condition
    sge $t0, $s2, $s1
    bnez $t0, end_loop

        sll $t0, $s2, 2         # Multiply the loop counter by
                                # by 4 to get offset (each element
                                # is 4 big)
        add $t0, $t0, $s0       # address of next array element
        lw $a1, 0($t0)          # Next array element
        la $a0, comma
        jal PrintInt            # print the integer from array

        addi $s2, $s2, 1        #increment $s0
        b loop
end_loop:
    li $v0, 4
    la $a0, close_bracket       # print close bracket
    syscall

    lw $ra, 0($sp)
    lw $s0, 4($sp)
    lw $s1, 8($sp)
    lw $s2, 12($sp)             # restore stack and return
    addi $sp, $sp, 16
    jr $ra

.data
    open_bracket: .asciiz "["
    close_bracket: .asciiz "]"
    comma: .asciiz ","
.include "utils.asm"
```

Ví dụ 2 – Phép nhân đệ quy

- Định nghĩa:

$$M(m,n) = m \text{ (when } n = 1)$$

$$\text{else} = m + M(m,n-1)$$

Ví dụ 2 – Phép nhân đệ quy

- Mã giả

```
subprogram global main()
{
    register int multiplicand
    register int multiplier
    register int answer
    m = prompt("Enter the multiplicand")
    n = prompt("Enter the multiplier")
    answer = Multiply(m, n)
    print("The answer is: " + answer)
}
subprogram int multiply(int m, int n)
{
    if (n == 1)
        return m;
    return m + multiply(m,n-1)
}
```

Ví dụ 2 – Phép nhân đệ quy

- Sv nhập chương trình sau, biên dịch, chạy, quan sát và giải thích sự thay đổi của các thanh ghi:

```
1   .text
2   .globl main
3   main:
4       # register conventions
5       # $s0 - m
6       # $s1 - n
7       # $s2 - answer
8       la $a0, prompt1     # Get the multiplicand
9       jal PromptInt
10      move $s0, $v0
11
12      la $a0, prompt2     # Get the multiplier
13      jal PromptInt
14      move $s1, $v0
15
16      move $a0, $s0
17      move $a1, $s1
18
19      jal Multiply        # Do multiplication
20      move $s2, $v0
21
22      la $a0, result      #Print the answer
23      move $a1, $s2
24      jal PrintInt
25
26      jal Exit
27
```

```
28  Multiply:
29      addi $sp, $sp -8    # push the stack
30      sw $a0, 4($sp)      # save $a0
31      sw $ra, 0($sp)      # save the $ra
32
33      seq $t0, $a1, $zero # if (n == 0) return
34      addi, $v0, $zero, 0 # set return value
35      bnez $t0, Return
36
37      addi $a1, $a1, -1   # set n = n-1
38      jal Multiply        # recurse
39      lw $a0, 4($sp)      # retrieve m
40      add $v0, $a0, $v0   # return m +
41                          #  multiply(m,n-1)
42      Return:
43      lw $ra, 0($sp)      #pop the stack
44      addi $sp, $sp, 8
45      jr $ra
46
47  .data
48  prompt1: .asciiz "Enter the multiplicand: "
49  prompt2: .asciiz "Enter the multiplier: "
50  result: .ascii "The answer is: "
51  .include "utils.asm"
```

Khung ngăn xếp

- Mỗi chương trình con được gọi sẽ tạo ra một khung ngăn xếp duy nhất cho chương trình con đó.