**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**


# MÔN HỌC: KIẾN TRÚC MÁY TÍNH (THỰC HÀNH) (CO2008)

Bài tập/Thực hành 4

CHƯƠNG 2 KIẾN TRÚC TẬP LỆNH MIPS: GỌI HÀM (LẬP TRÌNH CẤU TRÚC), THỜI GIAN THỰC THI

**LỚP THỰC HÀNH L03 – HỌC KỲ 212**

*Giảng viên hướng dẫn: Vũ Trọng Thiên*


| Sinh viên thực hiện | Mã số sinh viên |
|---|---|
| Phạm Duy Quang | 2011899 |


*Thành phố Hồ Chí Minh, tháng 04 năm 2022*

**Bài tập và Thực hành**

Sinh viên chuyển chương trình C bên dưới qua hợp ngữ MIPS tương ứng.

1. Leaf function (hàm lá)

Chuyển thủ tục "reverse" (đảo thứ tự chuỗi) từ ngôn ngữ C sang hợp ngữ MIPS. Thủ tục reverse được gọi khi thực thi lệnh jal reverse từ vùng .text. cArray, cArray_size được gắn vào các thanh ghi thanh ghi $a_0$, $a_1$. Giá trị trả về (nếu có) chứa vào $v_0$. Xuất chuỗi ra console.

```
char[] cArray = "Computer Architecture 2020"
int cArray_size = 26;
void reverse(char[] cArray, int cArray_size)
{
    int i;
    char temp;
    for (i =0 ; i <cArray_size/2; i ++)
    {
        temp = cArray[i];
        cArray[i] = cArray[cArray_size -1 -i];
        cArray[cArray_size -1 -i] = temp;
    }
}
```

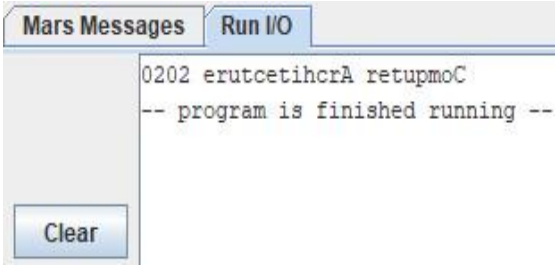
Lưu ý: Dùng "jal reverse" để gọi thủ tục "reverse" và dùng "jr $ra" trở về vị trí thanh ghi $ra đánh dấu.

❖ **Code:**

```
.data

cArray: .asciiz "Computer Architecture 2020"

.text
main:
        li $a1, 26
        la $a0, cArray

        jal reverse

        li $v0, 4
        syscall
exit:
        li $v0, 10
        syscall
```

```
reverse:
	# i -> $t0, cArray_size / 2 -> $t1, temp -> $t2
	div $t1, $a1, 2
	li $t0, 0
	for:
		beq $t0, $t1, end

		la $t2, ($a0)
		la $t3, ($a0)

		add $t2, $t2, $t0

		add $t3, $t3, $a1
		sub $t3, $t3, $t0
		subi $t3, $t3, 1

		lb $t4, ($t2)
		lb $t5, ($t3)

		sb $t5, ($t2)
		sb $t4, ($t3)

		addi $t0, $t0, 1
		j for
	end:
	jr $ra
```

❖ **Run I/O:**



2. Non-leaf function (là hàm/thủ tục gọi một hàm/thủ tục bên trong).

Chuyển thủ tục range từ C sang hợp ngữ MIPS tương đương.

```
int iArray[10];
int iArray_size = 10;
int range(iArray, iArray_size)
{
    int temp1 = max(iArray, iArray_size);
    int temp2 = min(iArray, iArray_size);
    int range = temp1 - temp2;
    return range;
}
```

Chương trình bắt đầu từ vùng .text, sau đó nó gọi hàm range. Trong hàm range lại gọi 2 hàm con là max và min. Giả sử địa chỉ và kích thước iarray được gắn lần lượt vào các thanh ghi $a0, $a1. Xuất giá trị range ra ngoài console.

**Lưu ý:** Khi gọi các hàm/thủ tục thanh ghi $ra sẽ tự đánh dấu lệnh tiếp theo như là vị trí trở về. Do đó trước khi gọi hàm con trong hàm range thì sinh viên cần lưu lại giá trị thanh ghi $ra trong stack. Sau khi thực thi xong, sinh viên cần phục hồi lại giá trị cho thanh ghi $ra từ stack. Dùng "jal range", "jal max", "jal min" để gọi thủ tục range, max, min. Dùng "jr $ra" để trở về vị trí lệnh mà thanh ghi $ra đã đánh dấu.

**Để lưu (push) giá trị $ra vào stack, ta có thể dùng các lệnh sau:**

```
addi $sp, $sp, -4    # adjust stack for 1 item
sw   $ra, 0($sp)     # save return address
```

**Để phục hồi (pop) $ra từ stack, ta có thể dùng các lệnh sau:**

```
lw   $ra, 0($sp)     # restore return address
addi $sp, $sp, 4     # pop 1 item from stack
```

❖ **Code:**

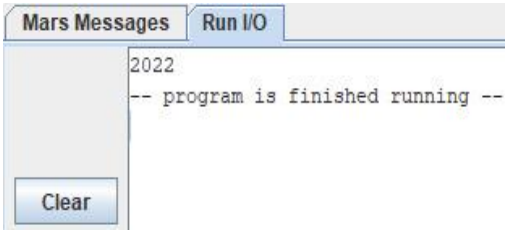
```
.data
iArray: .word 7 75 40 -1176 14 94 89 152 846 -687
.text
        la $a0, iArray
        li $a1, 15

        jal range

        move $a0, $v0
```

```
        li $v0, 1
        syscall

exit:
        li $v0, 10
        syscall

range:
        subi $sp, $sp, 4
        sw $ra, ($sp)

        jal max
        move $s1, $v0

        jal min
        move $s2, $v0

        sub $v0, $s1, $s2

        lw $ra, ($sp)
        addi $sp, $sp, 4
        jr $ra

max:
        subi $sp, $sp, 4
        sw $ra, ($sp)

        li $t0, 1
        subi $t1, $a1, 1
        la $s0, ($a0)

        lw $v0, ($a0)
```

```
        for:
                beq $t0, $t1, end
                if:
                        lw $t2, 4($s0)
                        blt $t2, $v0, endif
                        move $v0, $t2
                endif:
                addi $s0, $s0, 4
                addi $t0, $t0, 1
                j for
        end:
        lw $ra, ($sp)
        addi $sp, $sp, 4
        jr $ra

min:
        subi $sp, $sp, 4
        sw $ra, ($sp)

        li $t0, 1
        subi $t1, $a1, 1
        la $s0, ($a0)

        lw $v0, ($a0)
        for1:
                beq $t0, $t1, end1
                if1:
                        lw $t2, 4($s0)
                        bgt $t2, $v0, endif1
                        move $v0, $t2
                endif1:
                addi $s0, $s0, 4
```

```
        addi $t0, $t0, 1
        j for1
end1:
lw $ra, ($sp)
addi $sp, $sp, 4
jr $ra
```

❖ **Run I/O:**




3. Cho đoạn code hợp ngữ MIPS bên dưới

```
1           addi $a0, $zero, 100    // upper threshold
2           addi $a1, $zero, 0      // count variable
3           add  $a2, $zero, $zero  // sum initialization
4   loop:
5           beq  $a0, $a1, exit
6           add  $a2, $a2, $a1
7           addi $a1, $a1, 1
8           j loop
9   exit:
```

(a) Xác định giá trị của thanh ghi $a2 sau khi thực thi đoạn code trên.

Vòng lặp thực hiện chức năng tính tổng từ 0 đến 99 và lưu kết quả vào thanh ghi $a2: $\sum_0^{99} X = 4950 = 1356_{Hex}$.

Vậy giá trị của thanh ghi $a2 sau đoạn code trên là 0x00001356.

(b) Xác định tổng số chu kỳ thực thi khi thực thi đoạn chương trình trên. Giả sử CPI của các lệnh là 1.

Vòng lặp thực hiện 100 lần, mỗi lần 4 lệnh, 3 lệnh add trước vòng lặp for và một lệnh beq để nhảy đến exit, vậy có tổng cộng: $IC = 3 + 100 \times 4 + 1 = 404$ (lệnh).

$\Rightarrow$ Tổng số chu kỳ thực thi $= CPI \times IC = 1 \times 404 = 404$ (chu kỳ).

(c) Giả sử vùng .text (text segment - vùng để chứa các lệnh thực thi) bắt đầu từ địa chỉ 0x10080000. Xác định mã máy của lệnh "j loop" ở dạng HEX.

Địa chỉ đầu tiên bắt đầu từ 0x10080000.

Dòng beq $a0, $a1, exit có địa chỉ 0x1008000c.

Dòng j loop có địa chỉ 0x10080018 thực hiện nhảy vô điều kiện đến loop chứa địa chỉ 0x1008000c.

$$\text{target address} = \{\text{PC}\ [31:28], \text{address}\ [25:0], 00\}$$

$$0\text{x}1008000\text{c} = \{0000_2, \text{address}\ [25:0], 00_2\}$$

$$\Rightarrow \text{address} = 00\_0000\_0010\_0000\_0000\_0000\_0011$$

Vậy mã máy của lệnh j loop là:

$$0000\_1000\_0000\_0010\_0000\_0000\_0000\_0011 = 0\text{x}08020003$$