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

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**


# BÀI TẬP

# MIPS ASSEMBLY PROGRAMMING

# (ĐIỂM CỘNG)

**HỌC KỲ II**

**Sinh viên: Hà Chí Hào**

**MSSV: 19120219**

**Thành Phố Hồ Chí Minh – Năm 2020**

# Mục lục



Bài báo cáo này em sẽ sử dụng **pseudo code** để diễn đạt ý tưởng.

## 1. Bài 1

Mới đầu em cho người dùng nhập vào 1 số bằng system call code 5

```
$v0 <- 5
syscall
```

Sau đó thì em chuyển kết quả đọc được vào $s0 để dễ tính toán, đồng thời gán $s1 là 1 để chứa kết quả cuối cùng

```
$s0 <- $v0
$s1 <- 1
```

Cách làm của em bài này là tính giai thừa bằng công thức n*(n-1)*(n-2)… tức là em sẽ có 1 biến có giá trị ban đầu là n, sau đó em sẽ giảm dần nó về, trong lúc đó em sẽ lấy biến đó nhân vào 1 biến kết quả. Để làm vậy em có vòng lặp sau:

```
Loop:
if $s0 != 1 jump to Mul
else jump to Done
Mul:
$s1 = $s1 * $s0
$s0 = $s0 – 1
jump to Loop
```

Khi mà $s0 chưa phải là 1 thì em lại cho nó tiếp tục công việc nhân vào biến kết quả và trừ $s0 đi 1, cho tới khi $s0 là 1 thì dừng, là nhảy tới Done. Ở label Done em xuất ra kết quả.

```
Done:
$v0 <- 1
```

```
$a0 <- $s1
syscall
```

2. **Bài 2**

Em sẽ cho người dùng nhập vào kí tự trước, và sau đó để nó vào $s0 để so sánh:

```
Nhap:
$v0 <- 12
syscall
$s0 <- $v0
```

Giờ em sẽ bắt đầu việc so sánh với các mã ASCII để xét xem kí tự đó có phải là chữ cái hay không, ý tưởng của em như sau:

- Nếu kí tự < 65 (‘A’) thì nó không phải là chữ cái
- Nếu không thì: nếu kí tự > 122 (‘z’) thì nó không phải là chữ cái
- Nếu không thì: nếu kí tự > 96 (‘a’-1) thì nó là chữ cái
- Nếu không thì: nếu kí tự < 91 (‘Z’+1) thì nó là chữ cái
- Nếu không thì: kí tự không phải là chữ cái (giữa 91 và 96)

Nếu nó không phải kí tự thì xuất ra màn hình câu tương ứng và về lại chỗ Nhap, nếu nó là kí tự thì xuất ra màn hình câu tương ứng và kết thúc chương trình

```
if ($s0 < 65) jump to notLetter
if ($s0 > 122) jump to notLetter
if ($s0 > 96) jump to notLetter
if ($s0 < 91) jump to notLetter
notLetter:
$v0 <- 4
$a0 <- "\nKi tu khong phai la 1 chu cai, moi ban nhap lai\n"
syscall
jump to Nhap

isLetter:
$v0 <- 4
```

```
$a0 <- "\nKi tu la 1 chu cai"

syscall
```

3. **Bài 3**

Mới vô em cũng cho người dùng nhập vào string cần đảo ngược, với system call code 8, tham số $a0 sẽ chứa biến mà sau khi người dùng nhập, string được nhập sẽ ở biến đó, em sẽ dùng biến buffer để chứa. Còn tham số $a1 sẽ là độ dài tối đa của string người dùng nhập.

```
$v0 <- 8

$a0 <- buffer

$a1 <- 100

syscall
```

Sau đó em để biến buffer vào $s0 để tiện xử lí, đồng thời em cho $s1 bằng 0 và sử dụng nó để tìm độ dài của string vừa nhập trong lúc xử lí.

Ý tưởng lật ngược string của em như sau:

- Lấy địa chỉ hiện tại ở $s0, để vào $t0
- Nếu $t0 = 0, điều đó đồng nghĩa đã chạy hết string
- Nếu chưa hết string, thì push $t0 vào stack, điều đó đồng nghĩa push char hiện tại đang xét của string vào stack
- Tăng địa chỉ đang xét ở $s0
- Cập nhật độ dài ($s1)
- Lặp lại

Lúc này $s1 sẽ bằng đúng độ dài của string được nhập vào. Sau đó để xuất ra màn hình em làm như sau:

- Pop stack, để giá trị được pop vào $t0
- Xuất $t0
- Giảm $s1
- Nếu $s1 = 0, thì lúc đó đã hết độ dài string, lúc này sẽ dừng chương trình
- Nếu không, thì lặp lại

Mã giả:

```
PushString:

$t0 <- 0($s0)

if ($t0 = 0) jump to EndPush
```

```
push <- $t0
$s0++
$s1++
jump to PushString

EndPush:
PopString:
pop -> $t0
$v0 <- 11
$a0 <- $t0
Syscall
$s1—
if ($s1=0) jump to EndPop
else jump to PopString
EndPop:
```