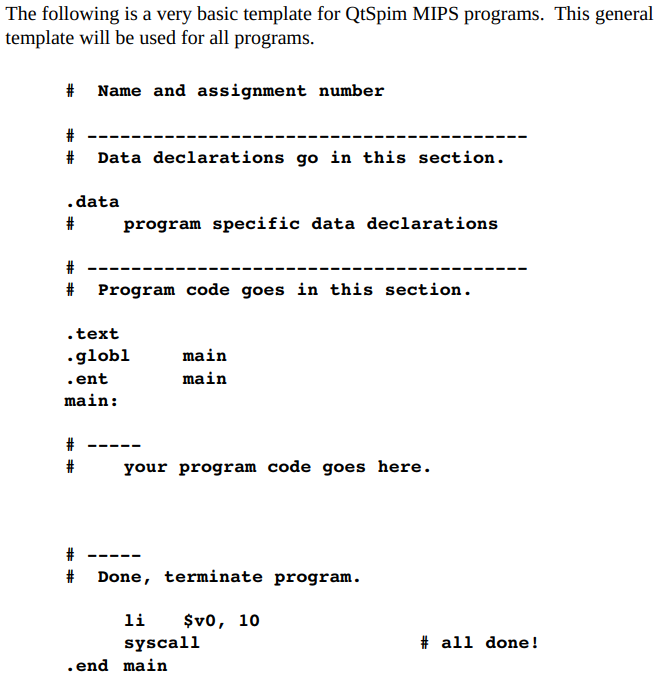
**MIPS SUMMARY**

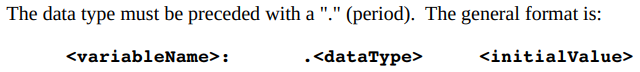
**(ASSEMBLY)**

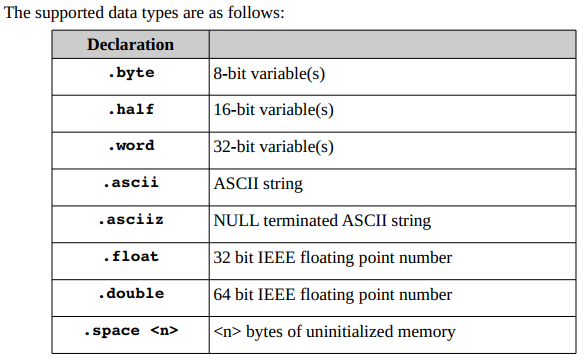
**Link tải phần mềm QTSpim:** [**Tại đây**](https://drive.google.com/file/d/12C6XUxFXr8cKqyvWHkZ4vuTR3MLWRAqx/view?usp=sharing)

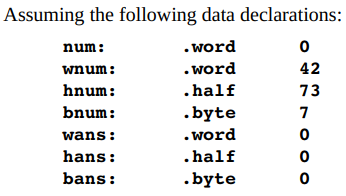
**1. Program Template**

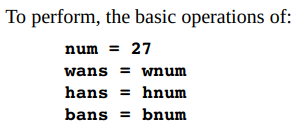
****

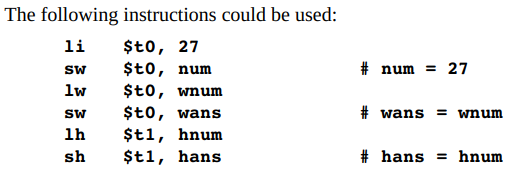
**2. Data Declarations**

****

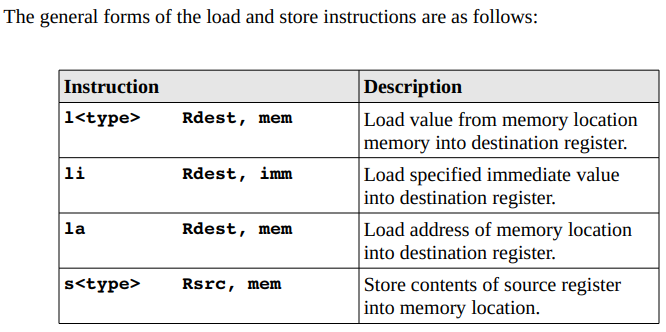
****

******

******



**3. Load / Store Instructions**

****

1. ***load: loading of data from memory (e.g., variables or arrays) into registers***

* ***syntax:***

*l <type> register\_destination*, *RAM\_source* *# Load integer number*

*l****.*** *<type> register\_destination, RAM\_source* *# Load float number*

1. ***load immediate:***

* ***syntax:***

*li register\_destination, value # Load immediate of a integer number*

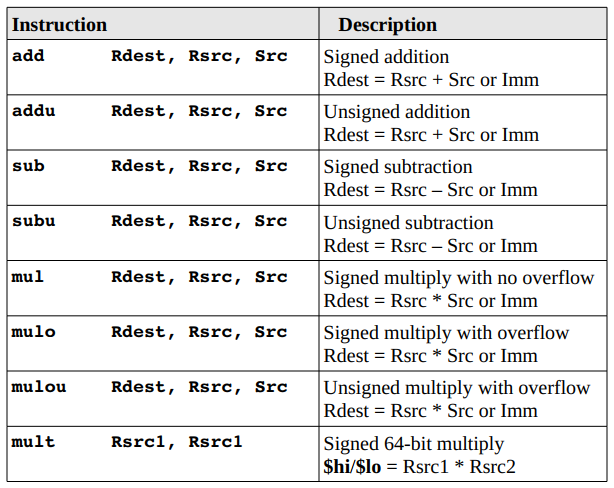
*li.<type> register\_destination, value* *# Load immediate of a float number*

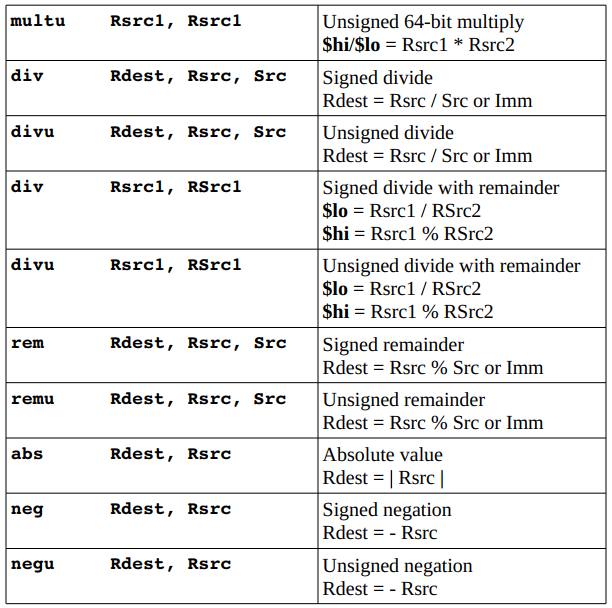
1. ***store: storing of data in register back to memory***

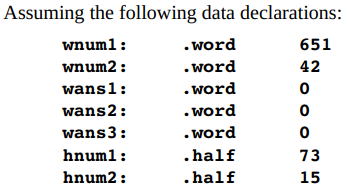
* ***syntax:*** *s <type> register\_source, RAM\_destination* *# Store integer number*

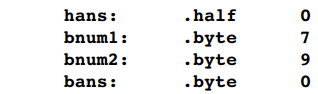
*s* ***.*** *<type> register\_source, RAM\_destination # Store float number*

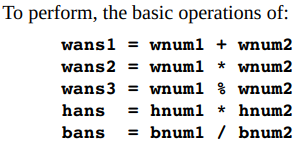
**4. Integer Arithmetic Operations**

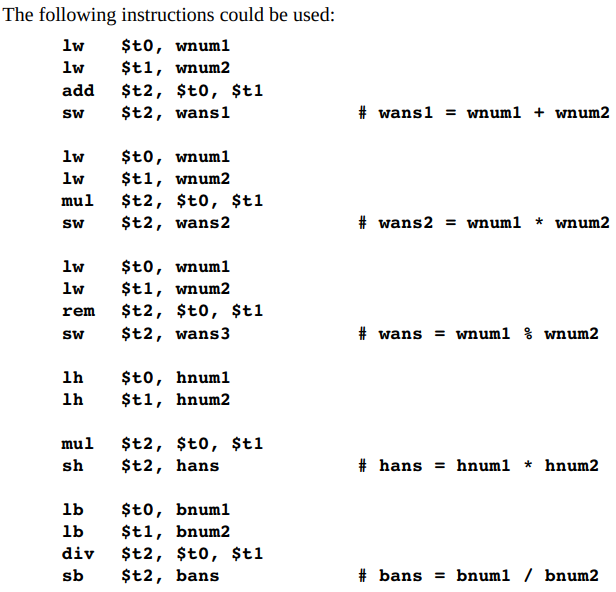
****

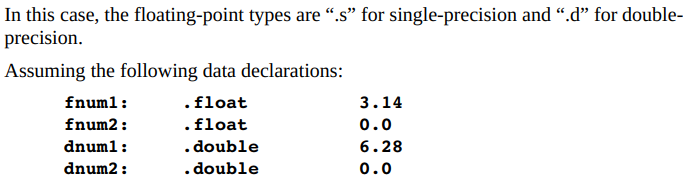
****

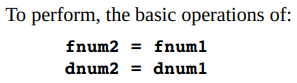
****

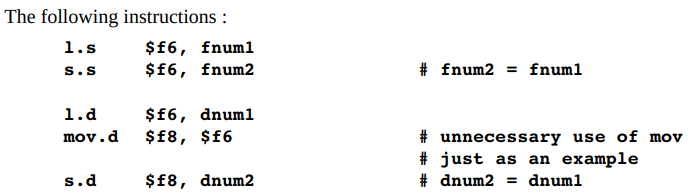
****

****

****

****

****

****

**5. Convert số nguyên thành số thực (Xem chi tiết mục 5.7.3)**

Giả sử để chia 2 số nguyên x, y và kết quả sau khi chia phải hiển thị là kiểu số thực thì chúng ta phải convert 2 số nguyên x, y thành số thực trước khi chia.

**VD:**

lw $t1, 3 # y=2

lw $t2, 2 # x=3

mtc1 $t1, $f1 # switch register $t1 to $f1

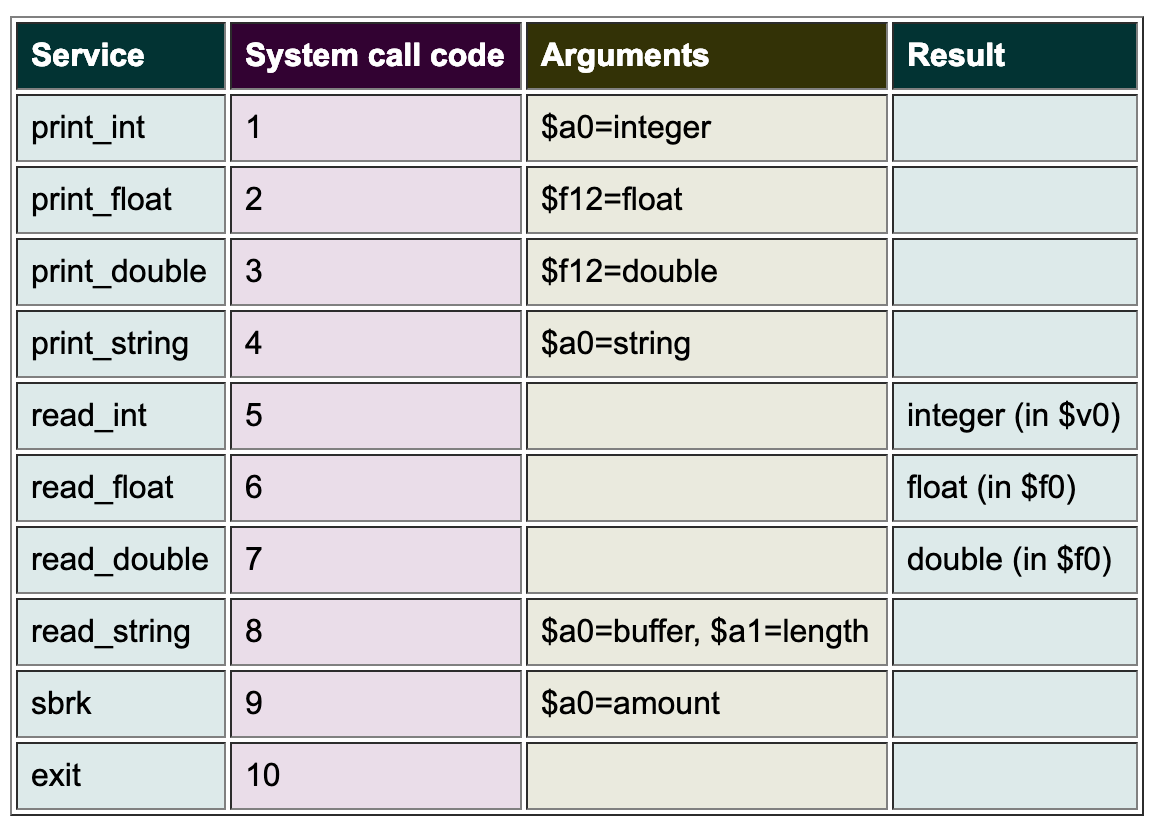
mtc1 $t2, $f2 # switch register $t2 to $f2

cvt.s.w $f1, $f1 # convert integer $f1 to float $f1

cvt.s.w $f2, $f2 # convert integer $f2 to float $f2

div.s $f3, $f2, $f1 # Divide float $f3 = x/y

**6. System Calls and I/O**



* **Read a integer number**

*li $v0, 5 # load code 5 vào $v0 để hệ thống nhận biết nhập số nguyên*

*syscall # Gọi hệ thống để nhập số nguyên và lưu số vừa nhập vào thanh ghi $v0*

* **Read a float**

*li $v0, 6 # load code 6 vào $v0 để hệ thống nhận biết nhập số float*

*syscall # Gọi hệ thống để nhập số float và lưu số vừa nhập vào thanh ghi $f0*

* **Read a double**

*li $v0, 7 # load code 7 vào $v0 để hệ thống nhận biết nhập số double*

*syscall # Gọi hệ thống để nhập số double và lưu số vừa nhập vào thanh ghi $f0*

* **Print a integer**

*li $v0, 1 # load code 1 vào $v0 để hệ thống nhận biết xuất số nguyên*

*lw $a0, number # Load số cần xuất vào thanh ghi $a0*

*syscall # Gọi hệ thống để xuất giá trị chưa trong thanh ghi $a0*

* **Print a float**

*li $v0, 2 # load code 2 vào $v0 để hệ thống nhận biết xuất số float*

*mov.s $f12, $f0 # Di chuyển giá trị trong $f0 vào $f12*

*syscall # Gọi hệ thống để xuất giá trị float chứa trong thanh ghi $f12*

* **Print a double**

*li $v0, 3 # load code 3 vào $v0 để hệ thống nhận biết xuất số double*

*mov.s $f12, $f0 # Di chuyển giá trị trong $f0 vào $f12*

*syscall # Gọi hệ thống để xuất giá trị double chứa trong thanh ghi $f12*

* **Print a string**

*li $v0, 4 # load code 4 vào $v0 để hệ thống nhận biết xuất chuỗi*

*la $a0, msg # Load địa chỉ của chuỗi cần xuất vào thanh ghi $a0*

*syscall # Gọi hệ thống để xuất giá trị chuỗi chứa trong thanh ghi $a0*

* **Exit**

*li $v0, 10 # load code 10 vào $v0 để hệ thống nhận biết thoát chương trình*

*syscall # Gọi hệ thống để thoát*

**Ví dụ 1: Viết chương trình Xuất một chuỗi “Xin chào các bạn” ra màn hình**

Mở Editor soạn thảo code, lưu lại với đuôi .s (VD mở Notepad++, soạn code và Save: Bai1.s)

*.data*

*input: .asciiz "Xin chao cac ban"*

*.text*

*.globl main*

*main:*

*# print input*

*li $v0, 4 # Load code =4 vào thanh ghi $v0 để hệ thống biết print chuỗi*

*la $a0, input # Load địa chỉ của chuỗi input cần xuất vào $a0*

*syscall # Gọi hệ thống để xuất nội dung trong $a0 ra*

*# exit*

*li $v0,10 # Load code =10 vào thanh ghi $v0 để hệ thống biết là exit*

*syscall # Gọi hệ thống để exit*

**Ví dụ 2: Viết chương trình nhập vào 2 số nguyên. Tính tổng và xuất kết quả**

Mở Editor soạn thảo code, lưu lại với đuôi .s (VD mở Notepad++, soạn code và Save: Bai1.s)

.data

msg: .asciiz "Nhap 2 so nguyen de tinh tong, cach nhau boi Enter: \n"

.text

.globl main

main:

# Print string msg

li $v0, 4

la $a0, msg

syscall

# Nhap so nguyen thu nhat

li $v0, 5 # Load code =5 vào thanh ghi $v0 để hệ thống biết là nhập số nguyên

syscall # Gọi hệ thống để cho phép nhập số nguyên, và lưu số vừa nhập vào $v0

move $t1, $v0 # Di chuyển giá trị của thanh ghi $v0 vào $t1 ($t1 = số thứ 1)

#Nhao so nguyen thu hai

li $v0, 5 # Load code =5 vào thanh ghi $v0 để hệ thống biết là nhập số nguyên

syscall # Gọi hệ thống để cho phép nhập số nguyên, và lưu số vừa nhập vào $v0

move $t2, $v0 # Di chuyển giá trị của thanh ghi $v0 vào $t2 ($t2=số thứ 2)

# Tính tổng

add $t3, $t1, $t2 # $t3 = $t1+$t2

#print a interger

li $v0, 1 # Load code=1 để hệ thống nhận biết print a integer

move $a0, $t3 # Di chuyển giá trị trong $t3 vào $a0 để xuất ra

syscall # Gọi hệ thống để xuất giá trị trong $a0 ra ngoài

# Thoát

li $v0, 10

syscall

**Ví dụ 3: Viết chương trình nhập vào 02 số thực. Tính tổng 02 số thực vừa nhập**

.data

msg: .asciiz "Nhap 2 so thuc de tinh tong, cach nhau boi Enter: \n"

.text

.globl main

main:

# Print string msg

li $v0, 4

la $a0, msg

syscall

# Nhap so float thu 1

li $v0, 6 # Load code =6 vào thanh ghi $v0 để hệ thống biết là nhập số float

syscall # Gọi hệ thống để cho phép nhập số float, và lưu số vừa nhập vào $f0

mov.s $f1, $f0 # Di chuyển giá trị của thanh ghi $f0 vào $f1 ($f1 = số thứ 1)

#Nhao so float thu 2

li $v0, 6 # Load code =6 vào thanh ghi $v0 để hệ thống biết là nhập số float

syscall # Gọi hệ thống để cho phép nhập số float, và lưu số vừa nhập vào $f0

mov.s $f2, $f0 # Di chuyển giá trị của thanh ghi $f0 vào $f2 ($f2=số thứ 2)

# Tính tổng

add.s $f3, $f1, $f2 # $f3 = $f1+$f2

#print float

li $v0, 2 # Load code=2 để hệ thống nhận biết print a float

mov.s $f12, $f3 # Di chuyển giá trị trong $f3 vào $f12 để xuất ra

syscall # Gọi hệ thống để xuất giá trị trong $f12 ra ngoài

# Thoát

li $v0, 10

syscall

**Ví dụ 3: Viết chương trình tính diện tích, chu vi**

# Data Declarations

.data

pi: .float 3.14159

fourPtZero: .float 4.0

threePtZero: .float 3.0

radius: .float 17.25

surfaceArea: .float 0.0

volume: .float 0.0

.text

.globl main

main:

l.s $f2, fourPtZero

l.s $f4, pi

mul.s $f4, $f2, $f4 # 4.0 \* pi

l.s $f6, radius # radius

mul.s $f8, $f6, $f6 # radius^2

mul.s $f8, $f4, $f8 # 4.0 \* pi \* radius^2

s.s $f8, surfaceArea # store final answer

l.s $f8, threePtZero

div.s $f2, $f4, $f8 # (4.0 \* pi / 3.0)

mul.s $f10, $f2, $f2

mul.s $f10, $f10, $f6 # radius^3

mul.s $f12, $f6, $f10 # 4.0\*pi/3.0\*radius^3

s.s $f12, volume # store final answer

# #Done, terminate program.

li $v0, 10 # terminate call code

syscall # system call

.end main

**Bài tập thực hành:**

1/ Viết chương trình in ra màn hình câu "Sinh viên TDTU xin chào các bạn"

2/ Viết chương trình nhập vào hai số nguyên từ bàn phím, tính tổng của 2 số nguyên đó.

3/ Viết chương trình nhập vào một số nguyên từ bàn phím, tính bình phương của số nguyên đó.

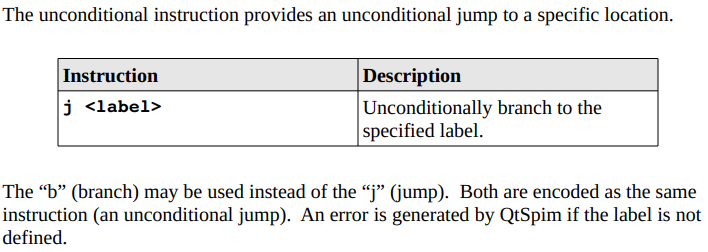
4/ Viết chương trình nhập vào 2 số thực từ bàn phím, tính tổng 2 số thực đó

5/ Viết chương trình nhập vào 2 số nguyên x, y từ bàn phím, tính thương 2 số nguyên đó. (Lưu ý: Thương của 2 số x/y là một số thực)

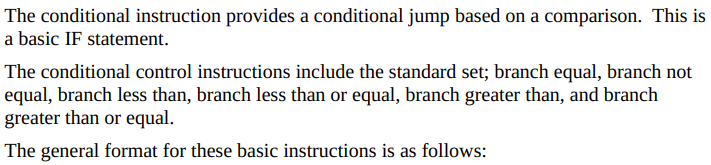
**Control Instructions and Loop**

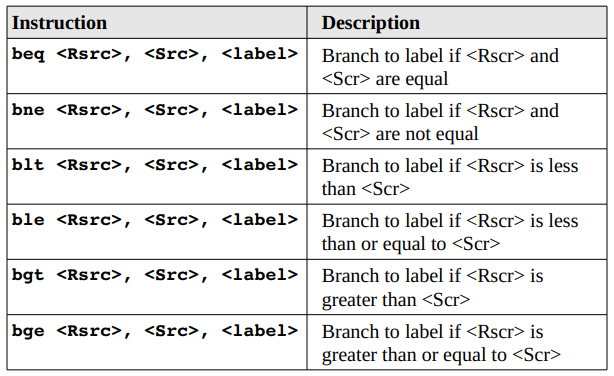
**(IF ….THEN…; For.../While...do)**

**1. Unconditional Control Instructions**

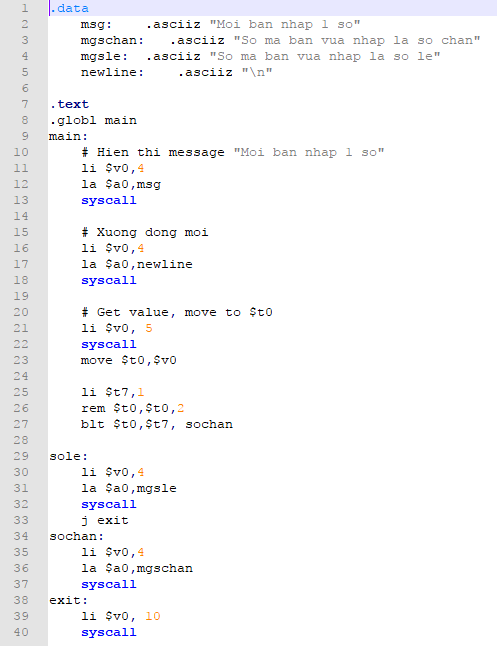
****

**2. Conditional Control Instructions**

****

****

**Ví dụ 1: Viết chương trình nhập vào một số nguyên, Xuất kết quả kiểm tra số vừa nhập là số chẵn hay lẻ**

****

**3. Loop**

|  |  |
| --- | --- |
| **Ngôn ngữ bậc cao** | **Hợp ngữ** |
| sum = 0;  for (i=1; i<=n; i++)  sum = sum + i;  print sum result | li $t0, 0 # sum = 0  li $t1, 1 # i = 1  lw $t2, n # N  Sum\_Loop:  bgt $t1, $t2, ExitLoop # i>n thi thoát lặp  add $t0, $t0, $t1 # sum = sum + i  add $t1, $t1, 1 # i = i+1  j Sum\_Loop  ExitLoop:  Print sum result  …... |
|  |  |

**Bài tập thực hành:**

1. Viết chương trình nhập vào một số nguyên N, tính tổng các số từ 1 đến N
2. Viết chương trình nhập vào một số nguyên N, tính tổng bình phương các số từ 1 đến N
3. Viết chương trình nhập vào một số nguyên N, tính tổng các số chẵn từ 1 đến N
4. Viết đoạn code nhập vào một số nguyên, nếu đó là số chia hết cho 3 thì thông báo ra màn hình.
5. Viết đoạn code nhập vào một số nguyên, xuất ra thông báo đó là số dương, số âm hay số 0.
6. Viết đoạn code cho nhập vào 2 số nguyên a và b, xác định a > b hay b > a hay 2 số bằng nhau.
7. Viết đoạn cho nhập một số nguyên dương n, và nếu người dùng nhập số âm hoặc số 0 thì yêu cầu nhập lại cho đến khi nào nhận được giá trị nguyên dương.
8. Nhập vào hai số nguyên dương a và b, tính tổng các số nguyên dương có giá trị nằm trong đoạn [a, b].

**ARRAY**

**1. Khai báo mảng:**

str1: .asciiz “xin chao cac ban”

songuyen: .byte 13, 14, -3 # Cấp 3 bytes và lưu các giá trị

list: .word 1, 7, 9 # Cấp 3 word, (mỗi word = 4 bytes)

N: .word 5

arrr1: .word 0

**2. Truy xuất mảng:**

# Dùng 1 vòng lặp gồm 3 tham số: biến đếm i, kích thước mảng N, list[i]

**VD:**

mangA: .word 11, 13, 15, 17, 19

length: .word 5

…..

main:

la $s0, mangA # S0=địa chỉ của mảng A

li $s1, 0 # i = 0

lw $s2, length # Size N

printLoop: # Lặp để xuất các phần tử

li $v0, 1 # code=1 print integer

lw $a0, ($s0) # Get mangA[i]

syscall

add $s1, $s1, 1 # i=i+1

add $s0, $s0, 4 # nhảy tới địa chỉ tiếp theo (4 bytes)

blt $s1, $s2, printLoop # i<N thì lặp lại

# i=N thì end

li $v0, 10

syscall

.end main

**3. Nhập mảng:**

sw $v0, ($s0)

**VD:**

# Nhap mang

readLoop:

#Get a integer number

li $v0, 5 # read integer, store to $v0

syscall

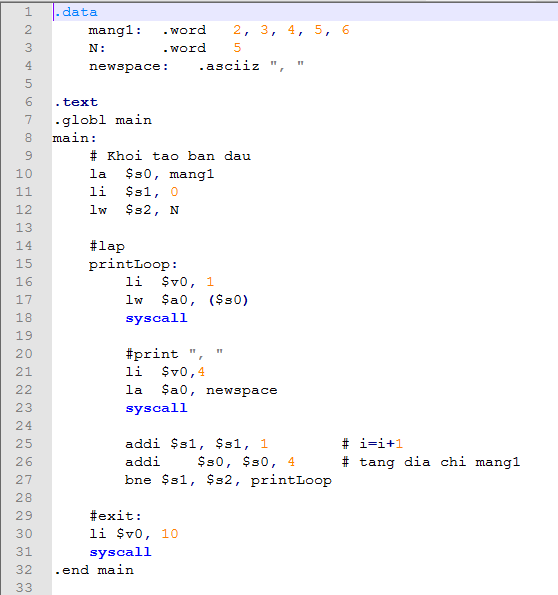
sw $v0, ($s0) # store value to dia chi cua mang1

addi $s1,$s1,1 # i=i+1

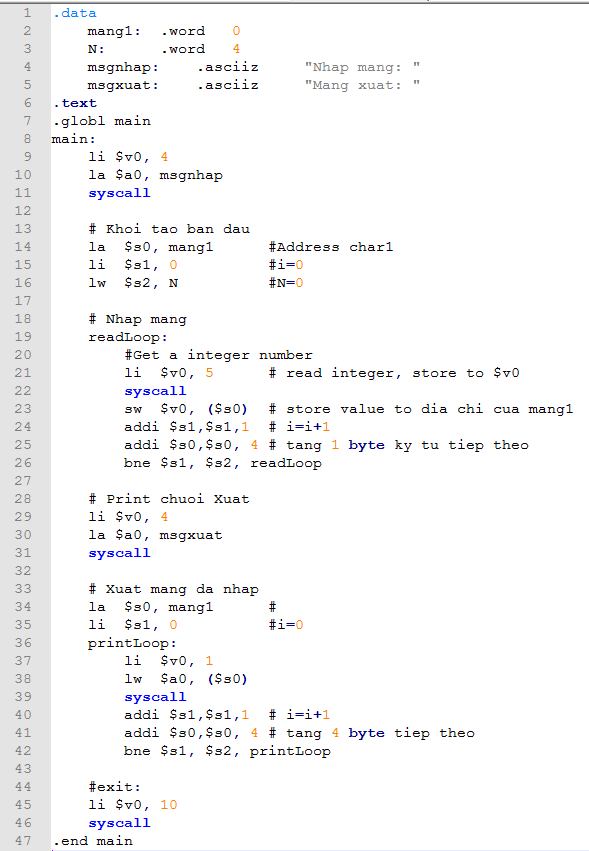
addi $s0,$s0, 4 # tang 4 byte để đến ô nhớ chứa phần tử theo

blt $s1, $s2, readLoop

**Ví dụ 1: Xuất mảng gồm 5 phần tử đã cho ra màn hình**

****

**Ví dụ 2: Viết chương trình nhập vào một mảng gồm 4 số nguyên nhập từ bàn phím, xuất mảng vừa nhập ra màn hình**



**Bài tập tự thực hành:**

1. Viết chương trình nhập vào một mảng gồm N số nguyên (N nhập từ bàn phím), xuất mảng N số nguyên đó ra màn hình.
2. Viết chương trình nhập vào một mảng gồm N số nguyên (N nhập từ bàn phím), tính tổng và trung bình các số trong mảng.
3. Viết chương trình nhập vào một mảng gồm N số nguyên (N nhập từ bàn phím), tìm số min, max của các phần tử trong mảng.
4. Viết chương trình nhập vào một mảng gồm N số nguyên (N nhập từ bàn phím), tính tổng các số chẵn/lẻ trong mảng.
5. Viết chương trình nhập vào một mảng gồm N số nguyên (N nhập từ bàn phím), sắp xếp và xuất ra màn hình danh sách mảng theo thứ tự tăng dần.

**CHƯƠNG TRÌNH CON**

**(Read section 8.0 Page 67)**

**1. Procedure Format**

The general syntax is as follows:

**.globl procedureName**

**.ent procedureName**

**procedureName:**

*# code goes here*

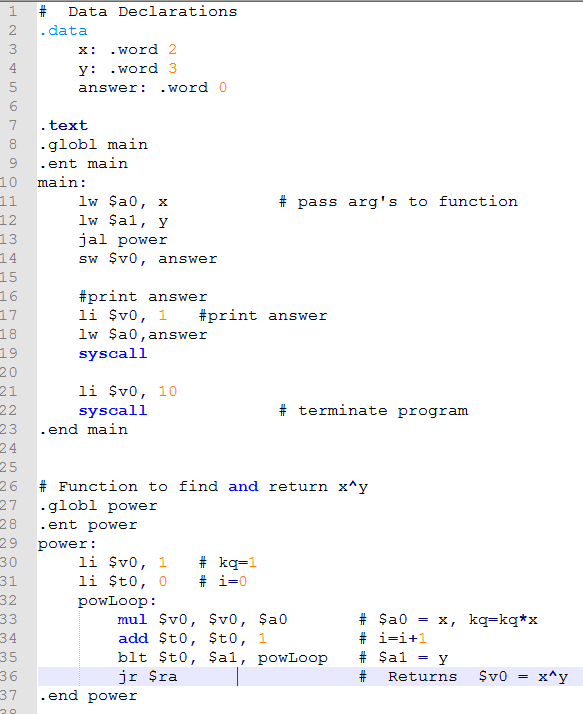
**.end procedureName**

* Dùng **jal** và **jr** (jal: jump and link; jr: jump return)

jal procedureName # Call procedureName

jr $ra # Return from procedure

* **VD: Viết chương trình tính lũy thừa x^y, trả về kết quả answer = power(x, y)**



**Bài tập thực hành:**

1/ Viết lại chương trình tính lũy thừa x^y, với x, y nhập từ bàn phím (Dùng chương trình con Power).

2/ Viết chương trình con tìm số min/max trong 2 số x, y (với x, y nhập từ bàn phím).

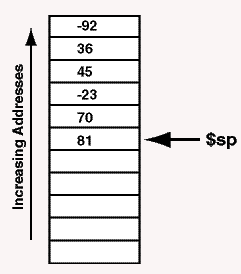
3/ Viết chương trình con tính n!

**STACK**

(Section 7.0, Page 61)

**1. Stack:**

* Stack = LIFO (Last in - First out): Ngăn xếp (Stack) là vùng nhớ được truy cập theo cơ chế như 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 (32 bit = 4 bytes).
* Thanh ghi **$sp** (stack pointer) là thanh ghi luôn luôn trỏ đến đỉnh của stack. Đỉnh của Stack luôn có địa chỉ thấp hơn.
* Stack có 2 thao tác cơ bản: **Push:** thêm 1 phần tử vào stack; **Pop:** Lấy 1 phần tử ra khỏi Stack

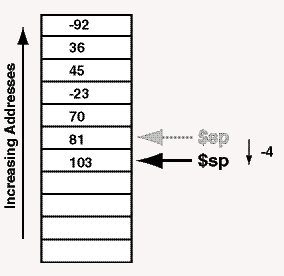
***VD1:* Giả sử đỉnh Stack đang trỏ vào số 81 như hình ở trên. Nếu bạn muốn thêm một phần tử 103 vào Stack thì giảm địa chỉ con trỏ $sp xuống 4 bytes, sau đó push giá trị vào Stack.**

*li $t0, 103 # load giá trị vào $t0*

*subu $sp, $sp, 4 # Giảm xuống 4 bytes để con trỏ* ***$sp*** *trỏ vào đỉnh Stack mới*

*sw $t0, ($sp) # push giá trị vào đỉnh Stack mới*

*(Xem minh họa như hình bên dưới)*

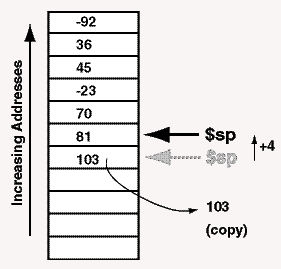


**VD2: Lấy giá trị trong đỉnh của Stack ra**

lw $t1, ($sp) # Lấy giá trị 103 trong đỉnh Stack, gán vào St1

addu $sp, $sp, 4 # tăng lên 4 bytes để con trỏ **$sp** trỏ vào đỉnh Stack mới

*(Xem minh họa như hình bên dưới)*



**VD 3: Viết Chương trình đọc các phần tử trong mảng đã cho và lưu vào stack; Lấy các phần tử trong stack lưu trở lại vào mảng; Xuất các phần tử trong mảng mới ra màn hình. (Chương trình này minh họa việc đảo ngược các phần tử bằng cách dùng Stack)**

*.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).

**BÀI TẬP LỚN MÔN TỔ CHỨC MÁY TÍNH**

**YÊU CẦU:** Sinh viên làm MỘT trong các đề bài theo danh sách được phân công. Hoàn thành bài tập lớn và **nộp vào tuần thứ 10.**

Sinh viên được gán đề bài theo công thức sau: **Số Đề = (Lấy 2 số cuối của mã SV) chia cho 10, lấy số dư +1;**

**Ví dụ: Mã sinh viên: 51900722 thì Số đề = (22 mod 10)+1 = 3**

**Đề:**

**Viết chương trình bằng ngôn ngữ assembly trên QTSpim hoặc một chương trình phù hợp để thực hiện các công việc sau:**

1. Nhập một chuỗi ký tự từ bàn phím, in ra chuỗi đảo ngược của chuỗi trên. Ví du: nhập vào: abcdef, xuất ra màn hình: fedcba.

2. Nhập một số từ bàn phím, xuất ra màn hình kết quả số đó có phải là số chính phương hay không.

3. Nhập một số từ bàn phím, xuất ra màn hình kết quả số đó có phải là số hoàn thiện hay không.

4. Nhập một số từ bàn phím, xuất ra màn hình kết quả số đó có phải là số nguyên tố hay không.

5. Nhập hai số từ bàn phím, xuất ra màn hình ước chung lớn nhất của hai số đó.

6. Viết chương trình nhập vào một mảng gồm N phần tử (N nhập từ bàn phím). Tính tổng các phần tử chia hết cho 3 và xuất ra màn hình tổng trên. (chú ý: Các giá trị trong mảng được nhập trực tiếp từ bàn phím.)

7. Viết chương trình nhập vào một mảng gồm N phần tử (N nhập từ bàn phím). Sắp xếp mảng theo thứ tự tăng dần/giảm dần, xuất ra màn hình 2 mảng sau khi sắp xếp. (chú ý: Các giá trị trong mảng được nhập trực tiếp từ bàn phím.)

8. Viết chương trình nhập vào một số từ bàn phím, in ra màn hình số trên ở hệ cơ số 2 (hệ nhị phân).

9. Viết chương trình nhập vào một số từ bàn phím, in ra màn hình số trên ở hệ cơ số 16 (hệ hecxa).

10. In ra màn hình kết quả kết quả của hàm số f(x) = 1/2 + 2/3 + 3/4 + … + n/n+1, với n là số nhập từ bàn phím.

**Chú ý:**

**- Làm đúng theo đề tài đã được phân công. Số đề được gán theo công thức ở trên. Làm sai đề sẽ không được chấm điểm.**

**- Mọi trường hợp copy bài của bạn hoặc cho bạn copy đều bị 0đ và CẤM THI.**

**- Trong Source code phải viết ghi chú rõ ràng, nếu không có ghi chú hoặc ghi chú không rõ ràng thì bị trừ điểm.**