***Họ và tên****: Nguyễn Trọng Tuệ*

***Mã số sinh viên****: 20194710*

***Mã lớp****: 130935*

**BÁO CÁO PROJECT CUỐI KỲ**

1. ***BÀI CHÍNH (BÀI 6)***
2. ***Nội dung bài toán***

* Chương trình mô phỏng cấp phát hàm malloc() của ngôn ngữ lập trình C bằng hợp ngữ Assembly MIPS để cấp phát bộ nhớ cho một biến con trỏ nào đó. Hãy đọc chương trình và hiểu rõ nguyên tắc cấp phát bộ nhớ động.
* Trên cơ sở chương trình mẫu, hãy hoàn thiện chương trình với những yêu cầu sau (kèm ví dụ minh họa)
* Việc cấp phát bộ nhớ kiểu word/mảng word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
* Viết hàm lấy giá trị Word /Byte của biến con trỏ (tương tự như \*CharPtr, \*BytePtr, \*WordPtr)
* Viết hàm lấy địa chỉ biến con trỏ (tương tự như &CharPtr, &BytePtr, \*WordPtr)
* Viết hàm thực hiện copy 2 con trỏ xâu kí tự (Xem ví dụ về CharPtr)
* Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động
* Hãy viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:  
  a. Địa chỉ đầu của mảng  
  b. Số dòng  
  c. Số cột
* Tiếp theo câu trên, hãy viết 2 hàm GetArray[i][j] và SetArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng I cột j của mảng.

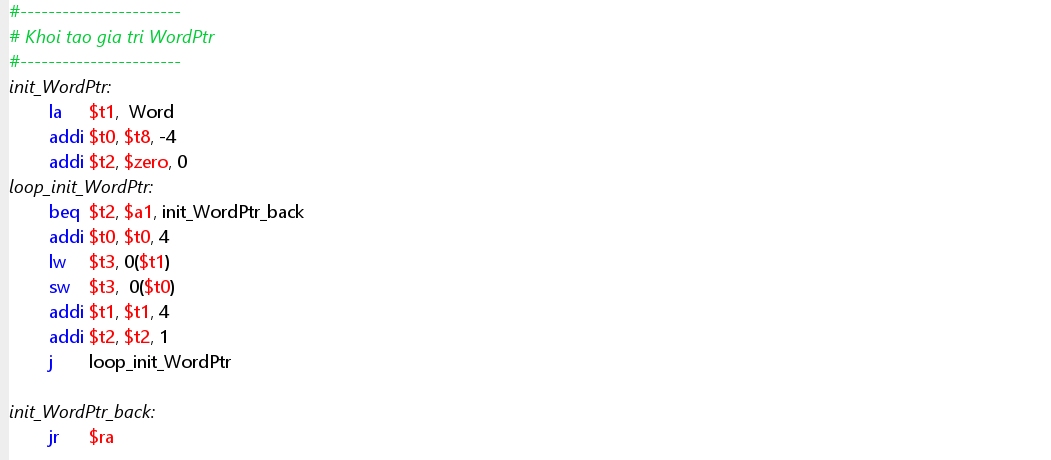
1. ***Phương pháp thực hiện bài toán***

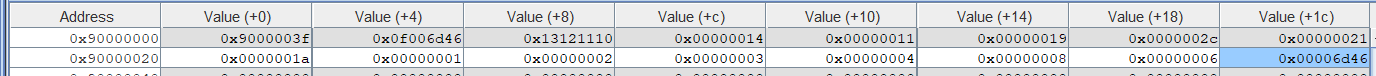
Dựa trên chương trình malloc mẫu, em đã xây dựng chương trình hoàn chỉnh bằng MIPS Assembly thực hiện các chức năng mà yêu cầu đề bài đã đặt ra. Ý tưởng, cách thức xây dựng và thực hiện từng chức năng của chương trình cấp phát bộ nhớ động mô phỏng hàm malloc() của ngôn ngữ lập trình C như sau:

* 1. Cấp phát bộ nhớ động cho các biến trong chương trình
* Thực hiện cấp phát bộ nhớ động cho các biến của chương trình, các biến ở đây do là biến con trỏ nên sẽ có giá trị là 4 bytes (chứa địa chỉ nó trỏ tới trong vùng nhớ .kdata – có thể coi vùng nhớ này tương đồng với vùng nhớ heap thực hiện cấp phát bộ nhớ cho các biến kiểu dữ liệu tham chiếu).



* Địa chỉ vùng nhớ trong .kdata tương ứng sẽ được gán giá trị phù hợp khi thực hiện khởi tạo giá trị, như trong đoạn code dưới đây mô tả cách một mảng số nguyên kiểu Word thực hiện gán từng giá trị vào vùng nhớ đã được khởi tạo từ trước đó. (Giá trị từ 0x90000010 đến 0x90000020 với mảng kiểu Word chứa 5 phần tử, mỗi phần tử 4 bytes)

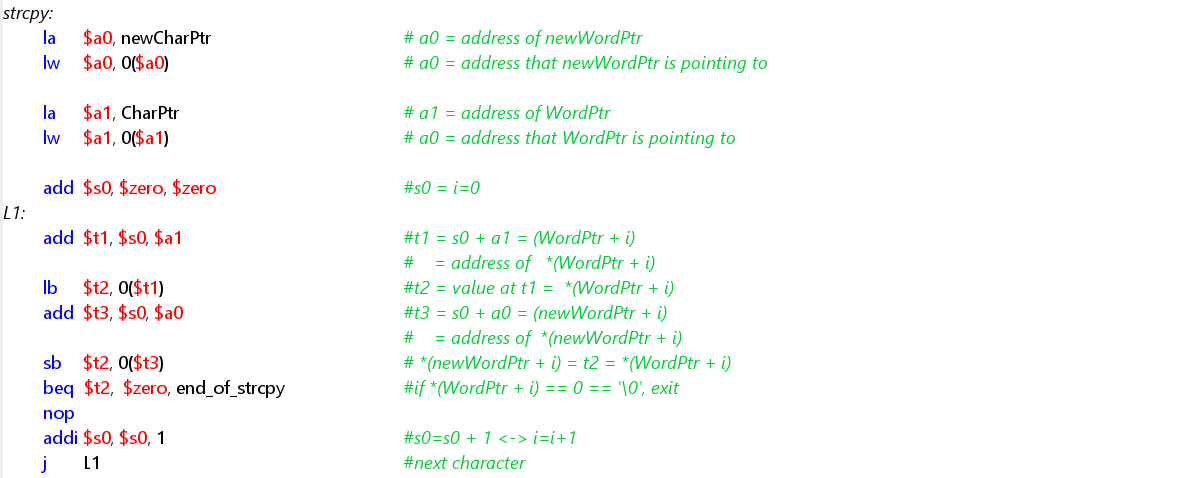




* Với các biến con trỏ kiểu word/ mảng word, ta sẽ phải đảm bảo yêu cầu rằng các biến trên sẽ trỏ đến địa chỉ đầu tiên luôn chia hết cho 4, thỏa mãn tính đúng đắn của dữ liệu. Do đó ta sẽ kiểm tra giá trị tiếp theo trong vùng nhớ .kdata tại địa chỉ 0x90000000 chứa địa chỉ còn trống tiếp theo trong vùng nhớ, sau đó nếu nó không chia hết cho 4 thì cộng phần bù vào từ thanh ghi hi sau khi chia dư cho 4 để thực hiện.
  1. Hiển thị giá trị của địa chỉ con trỏ, địa chỉ con trỏ trỏ đến, giá trị lưu trữ trong địa chỉ mà con trỏ trỏ đến
* Thực hiện lời gọi hệ thống với thanh ghi trả về $v0, load các biến, địa chỉ biến trỏ đến hay giá trị tại địa chỉ mà biến trỏ đến tại thanh ghi $a0 rồi thực hiện syscall



* 1. Viết hàm thực hiện copy 2 con trỏ xâu ký tự
* Tương tự như trong ngôn ngữ lập trình C, ta sẽ lấy giá trị của địa chỉ xâu mà biến đang trỏ đến (ký tự đầu của xâu) sau đó, cộng 1 sau mỗi lần thực hiện vòng lặp ở cả xâu ký tự đã có và xâu ký tự thực hiện copy để copy xâu sang xâu mới.



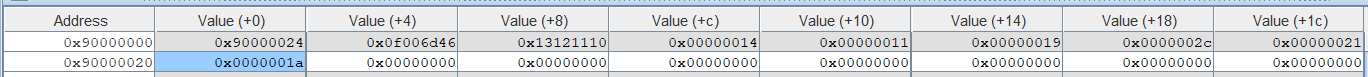
* 1. Liên quan đến cấp phát mảng động 2 chiều
* Ý tưởng: với mảng 2 chiều cỡ (m x n) thì ta sẽ thực hiện biến đổi mảng này về mảng một chiều để có thể lưu trữ giá trị của biến. Ví dụ a[1][1] trong mảng 2 chiều cỡ (2 x 3) sẽ được cấp phát là phần tử có địa chỉ lưu trữ giá trị là (địa chỉ đầu tiên của mảng + (1\*3 + 1)) = (địa chỉ đầu tiên của mảng + 4), tương ứng là phần tử thứ 5 của mảng. Việc thực hiện cấp phát mảng động 2 chiều được nhãn malloc2 trong chương trình thực thi
* Khi duyệt mảng để lấy giá trị hay thay đổi giá trị thì ta sẽ duyệt xem chỉ số nhập vào có vượt quá giới hạn của mảng là m\*n không, nếu vượt quá thì báo lỗi, ngược lại chương trình thực thi chức năng bình thường.

1. ***Kết quả***

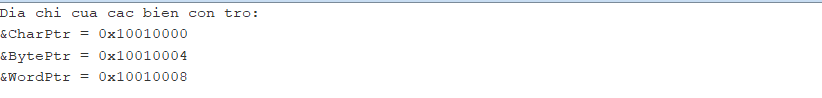
* Câu 1:
* Trước khi cấp phát mảng word, địa chỉ là 0x9000000d không chia hết cho 4

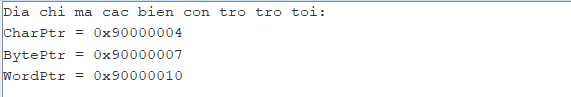


* Sau khi cấp phát, địa chỉ là 0x90000010 chia hết cho 4



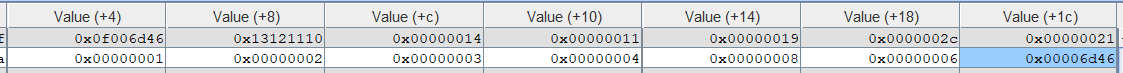
* Câu 2, 3:







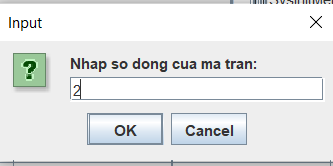
* Câu 4: “Fm” = 0x466d

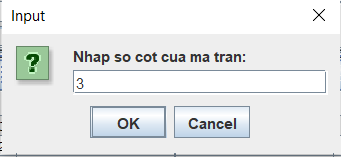


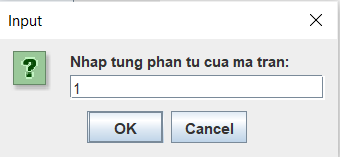
* Câu 5:

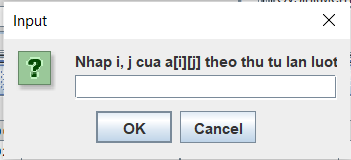


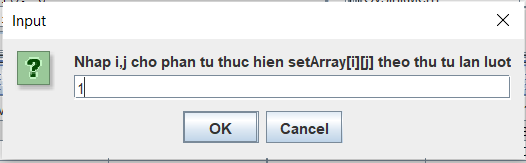
* Câu 6, 7: mảng 2 chiều cỡ 2 x 3 với các phần tử ({1, 2, 3}, {4, 5, 6})













1. ***Kết luận***

* Chương trình được xây dựng giúp em hiểu hơn về cách thức cấp phát bộ nhớ động mô phỏng từ ngôn ngữ lập trình C.
* Nếu có khả năng cải thiện chương trình, em sẽ xây dựng theo hướng thực hiện menu thay vì tuần tự như trong chương trình này.

1. ***Mã nguồn***

#Final Ex, Task 6

#Nguyen Trong Tue - 20194710

.data

CharPtr: .word 0 # Bien con tro, tro toi kieu asciiz

BytePtr: .word 0 # Bien con tro, tro toi kieu Byte

WordPtr: .word 0 # Bien con tro, tro toi mang kieu Word

Word2DPtr: .word 0 # Bien con tro, tro toi mang 2 chieu kieu Word

newCharPtr: .word 0 # Bien con tro, toi kieu asciiz

Char: .asciiz "Fm"

Byte: .byte 15, 16, 17, 18, 19, 20

Word: .word 17, 25, 44, 33, 26

m: .word 0

n: .word 0

gottenValue: .word

totalAllocatedMemory: .word

textCharPtr: .asciiz "&CharPtr = "

textBytePtr: .asciiz "&BytePtr = "

textWordPtr: .asciiz "&WordPtr = "

textCharDePtr: .asciiz "\*CharPtr = "

textByteDePtr: .asciiz "\*BytePtr = "

textWordDePtr: .asciiz "\*WordPtr = "

textCharValueOfPtr: .asciiz "CharPtr = "

textByteValueOfPtr: .asciiz "BytePtr = "

textWordValueOfPtr: .asciiz "WordPtr = "

message1: .asciiz "Dia chi cua cac bien con tro:\n"

message2: .asciiz "Dia chi ma cac bien con tro tro toi:\n"

message3: .asciiz "Khu con tro:\n"

message4: .asciiz "Tong bo luong bo nho da cap phat cho cac bien dong = "

message5: .asciiz " byte(s)"

message6: .asciiz "Nhap so dong cua ma tran: "

message7: .asciiz "Nhap so cot cua ma tran: "

message8: .asciiz "Nhap tung phan tu cua ma tran: "

message9: .asciiz "Nhap i, j cua a[i][j] theo thu tu lan luot"

message10: .asciiz "getArray[i][j] = "

message11: .asciiz "Nhap i,j cho phan tu thuc hien setArray[i][j] theo thu tu lan luot"

message12: .asciiz "Nhap gia tri moi cua a[i][j]"

message\_err: .asciiz "Chi so phan tu khong hop le!"

message\_end: .asciiz "--------------------END--------------------"

newLine: .asciiz "\n"

.kdata

Sys\_TheTopOfFree: .word 1

Sys\_MyFreeSpace:

.text

#Tong bo luong bo nho da cap phat cho cac bien dong

addi $s0, $zero, 0

#Khoi tao vung nho cap phat dong

jal SysInitMem

#-----------------------

# Cap phat cho bien con tro, gom 3 phan tu,moi phan tu 1 byte

#-----------------------

la $a0, CharPtr

addi $a1, $zero, 3

addi $a2, $zero, 1

jal malloc

jal init\_CharPtr

#-----------------------

# Cap phat cho bien con tro, gom 6 phan tu, moi phan tu 1 byte

#-----------------------

la $a0, BytePtr

addi $a1, $zero, 6

addi $a2, $zero, 1

jal malloc

jal init\_BytePtr

nop

#-----------------------

# Cap phat cho bien con tro, gom 5 phan tu, moi phan tu 4 byte

#-----------------------

la $a0, WordPtr

addi $a1, $zero, 5

addi $a2, $zero, 4

jal malloc\_WordPtr

jal init\_WordPtr

#lock: j lock

#nop

j getPointerAddress

#-----------------------

# Khoi tao gia tri CharPtr

#-----------------------

init\_CharPtr:

la $t1, Char

addi $t0, $t8, -1 #$t0 = $t8

addi $t2, $zero, 0 #$t2: dem so phan tu

loop\_init\_CharPtr:

beq $t2, $a1, init\_CharPtr\_back

addi $t0, $t0, 1

lb $t3, 0($t1)

sb $t3, 0($t0)

addi $t1, $t1, 1

addi $t2, $t2, 1

j loop\_init\_CharPtr

init\_CharPtr\_back:

jr $ra

#-----------------------

# Khoi tao gia tri BytePtr

#-----------------------

init\_BytePtr:

la $t1, Byte

addi $t0, $t8, -1

addi $t2, $zero, 0

loop\_init\_BytePtr:

beq $t2, $a1, init\_BytePtr\_back

addi $t0, $t0, 1

lb $t3, 0($t1)

sb $t3, 0($t0)

addi $t1, $t1, 1

addi $t2, $t2, 1

j loop\_init\_BytePtr

init\_BytePtr\_back:

jr $ra

#-----------------------

# Khoi tao gia tri WordPtr

#-----------------------

init\_WordPtr:

la $t1, Word

addi $t0, $t8, -4

addi $t2, $zero, 0

loop\_init\_WordPtr:

beq $t2, $a1, init\_WordPtr\_back

addi $t0, $t0, 4

lw $t3, 0($t1)

sw $t3, 0($t0)

addi $t1, $t1, 4

addi $t2, $t2, 1

j loop\_init\_WordPtr

init\_WordPtr\_back:

jr $ra

#---------------------------------------------------------------------------------------------

SysInitMem:

la $t9, Sys\_TheTopOfFree #Lay con tro chua dau tien con trong, khoi tao

la $t7, Sys\_MyFreeSpace #Lay dia chi dau tien con trong, khoi tao

sw $t7, 0($t9) #Luu lai

jr $ra

#---------------------------------------------------------------------------------------------

malloc:

la $t9, Sys\_TheTopOfFree #

lw $t8, 0($t9) #Lay dia chi dau tien con trong

sw $t8, 0($a0) #Cat dia chi do vao bien con tro

addi $v0, $t8, 0 # Dong thoi la ket qua tra ve cua ham

mul $t7, $a1,$a2 #Tinh kich thuoc cua mang can cap phat

mflo $t2 #Cap nhat tong bo luong bo nho da cap phat cho cac bien dong

add $s0, $s0, $t2

add $t6, $t8, $t7 #Tinh dia chi dau tien con trong

sw $t6, 0($t9) #Luu tro lai dia chi dau tien do vao bien Sys\_TheTopOfFree

jr $ra

#---------------------------------------------------------------------------------------------

malloc2:

jal malloc\_WordPtr

init2DArr:

#la $t1, Word2DPtr

#lw $t1, 0($t1)

addi $t0, $t8, -4

addi $t2, $zero, 0

loop\_init\_Word2DPtr:

beq $t2, $a3, init\_Word2DPtr\_back

addi $t0, $t0, 4

li $v0, 51

la $a0, message8

syscall

sw $a0, 0($t0)

addi $t2, $t2, 1

j loop\_init\_Word2DPtr

init\_Word2DPtr\_back:

j getArray

#-----------------------------------------------------------------------------------------

#CAU 1: Dia chi kieu word/ mang word phai chia het cho 4 (su dung malloc\_WordPtr)

malloc\_WordPtr:

la $t9, Sys\_TheTopOfFree #

lw $t8, 0($t9) #Lay dia chi dau tien con trong

addi $t5, $zero, 0x4

div $t8, $t5

mfhi $t4

beq $t4, $zero, afterCheckingDivisionBy4 #Chia 4 lay du de kiem tra

beq $t4, 0xfffffffd, missing3 #=-3 => thieu 3

beq $t4, 0xfffffffe, missing2 #=-2 => thieu 2

beq $t4, 0xffffffff, missing1 #=-1 => thieu 1

missing3:

addi $t3, $zero, 3

add $t8, $t8, $t3

j afterCheckingDivisionBy4

missing2:

addi $t3, $zero, 2

add $t8, $t8, $t3

j afterCheckingDivisionBy4

missing1:

addi $t3, $zero, 1

add $t8, $t8, $t3

j afterCheckingDivisionBy4

afterCheckingDivisionBy4:

sw $t8, 0($a0) #Cat dia chi do vao bien con tro

addi $v0, $t8, 0 # Dong thoi la ket qua tra ve cua ham

mul $t7, $a1,$a2 #Tinh kich thuoc cua mang can cap phat

mflo $t2 #Cap nhat tong bo luong bo nho da cap phat cho cac bien dong

add $s0, $s0, $t2

add $t6, $t8, $t7 #Tinh dia chi dau tien con trong

sw $t6, 0($t9) #Luu tro lai dia chi dau tien do vao bien Sys\_TheTopOfFree

jr $ra

#-----------------------------------------------------------------------------------------

#CAU 3: Dia chi con tro

getPointerAddress:

li $v0, 4

la $a0, message1

syscall

#&CharPtr

li $v0, 4

la $a0, textCharPtr

syscall

li $v0, 34

la $a0, CharPtr

syscall

li $v0, 4

la $a0, newLine

syscall

#&BytePtr

li $v0, 4

la $a0, textBytePtr

syscall

li $v0, 34

la $a0, BytePtr

syscall

li $v0, 4

la $a0, newLine

syscall

#&WordPtr

li $v0, 4

la $a0, textWordPtr

syscall

li $v0, 34

la $a0, WordPtr

syscall

li $v0, 4

la $a0, newLine

syscall

nop

#----------------------------------------------------------------------------------------

#CAU 3: Dia chi con tro tro de

getAddressPointedbyThePointer:

li $v0, 4

la $a0, newLine

syscall

li $v0, 4

la $a0, message2

syscall

#CharPtr

li $v0, 4

la $a0, textCharValueOfPtr

syscall

li $v0, 34

la $t0, CharPtr

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, newLine

syscall

#BytePtr

li $v0, 4

la $a0, textByteValueOfPtr

syscall

li $v0, 34

la $t0, BytePtr

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, newLine

syscall

#WordPtr

li $v0, 4

la $a0, textWordValueOfPtr

syscall

li $v0, 34

la $t0, WordPtr

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, newLine

syscall

#----------------------------------------------------------------------------------------

#CAU 2: Khu tham chieu

getValuebyDereferenceThePointer:

li $v0, 4

la $a0, newLine

syscall

li $v0, 4

la $a0, message3

syscall

#\*CharPtr

li $v0, 4

la $a0, textCharDePtr

syscall

la $t1, CharPtr

lw $t2, 0($t1)

lb $t1, 0($t2)

la $t3, gottenValue

sw $t1, 0($t3)

li $v0, 4

la $a0, gottenValue

syscall

li $v0, 4

la $a0, newLine

syscall

#\*BytePtr

li $v0, 4

la $a0, textByteDePtr

syscall

la $t1, BytePtr

lw $t2, 0($t1)

lb $t1, 0($t2)

la $t3, gottenValue

sw $t1, 0($t3)

li $v0, 1

la $t0, gottenValue

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, newLine

syscall

#\*WordPtr

li $v0, 4

la $a0, textWordDePtr

syscall

la $t1, WordPtr

lw $t2, 0($t1)

lw $t1, 0($t2)

la $t3, gottenValue

sw $t1, 0($t3)

li $v0, 1

la $t0, gottenValue

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, newLine

syscall

#----------------------------------------------------------------------------------------

#CAU 5: Tong bo luong bo nho da cap phat cho cac bien dong

totalAllocatedCapacity:

li $v0, 4

la $a0, newLine

syscall

li $v0, 4

la $a0, message4

syscall

la $t0, totalAllocatedMemory

sw $s0, 0($t0)

li $v0, 1

la $t0, totalAllocatedMemory

lw $a0, 0($t0)

syscall

li $v0, 4

la $a0, message5

syscall

li $v0, 4

la $a0, newLine

syscall

#lock:

#j lock

#----------------------------------------------------------------------------------------

#CAU 6: Malloc2 cap phat dong mang hai chieu

li $v0, 51

la $a0, message6

syscall

addi $t1, $a0, 0

li $v0, 51

la $a0, message7

syscall

addi $t2, $a0, 0

#-----------------------

# Cap phat cho bien con tro, gom 3 phan tu,moi phan tu 1 byte

#-----------------------

la $a0, Word2DPtr

mul $a3, $t1, $t2

addi $a1, $a3, 0

addi $a2, $zero, 4

la $t0, m

sw $t1, 0($t0)

la $t0, n

sw $t2, 0($t0)

jal malloc2

#-------------------------------------------------------------------------------------------------------------------------

#CAU 7: GET\_ARRAY

getArray:

li $v0, 51,

la $a0, message9

syscall

addi $t1, $a0, 0

li $v0, 51

la $a0, message9

syscall

addi $t2, $a0, 0

la $a3, Word2DPtr

lw $a3, 0($a3)

la $a1, m

lw $a1, 0($a1)

la $a2, n

lw $a2, 0($a2)

slt $t5, $t1, $a1

slt $t6, $t2, $a2

beq $t5, $zero, outOfRange

beq $t6, $zero, outOfRange

mul $t3, $t1, $a2

add $t3, $t3, $t2

mul $t3, $t3, 4

add $a3, $a3, $t3

li $v0, 4

la $a0, newLine

syscall

li $v0, 4

la $a0, message10

syscall

li $v0, 1

lw $a0, 0($a3)

syscall

li $v0, 4

la $a0, newLine

syscall

j setArray

outOfRange:

li $v0, 55

la $a0, message\_err

li $a1, 0

syscall

j end

#--------------------------------------------------------------------

#CAU 7: SET ARRAY

setArray:

li $v0, 4

la $a0, newLine

syscall

li $v0, 51,

la $a0, message11

syscall

addi $t1, $a0, 0

li $v0, 51

la $a0, message11

syscall

addi $t2, $a0, 0

la $a3, Word2DPtr

lw $a3, 0($a3)

la $a1, m

lw $a1, 0($a1)

la $a2, n

lw $a2, 0($a2)

slt $t5, $t1, $a1

slt $t6, $t2, $a2

beq $t5, $zero, outOfRange

beq $t6, $zero, outOfRange

mul $t3, $t1, $a2

add $t3, $t3, $t2

mul $t3, $t3, 4

add $a3, $a3, $t3

li $v0, 51

la $a0, message12

syscall

sw $a0, 0($a3)

li $v0, 4

la $a0, newLine

syscall

j copyCharPtr

#-----------------------------------------------------------------------------

#CAU 4: Copy CharPtr (strcpy simulation)

copyCharPtr:

la $a0, newCharPtr

addi $a1, $zero, 3

addi $a2, $zero, 1

jal malloc

jal init\_CharPtr

strcpy:

la $a0, newCharPtr # a0 = address of newWordPtr

lw $a0, 0($a0) # a0 = address that newWordPtr is pointing to

la $a1, CharPtr # a1 = address of WordPtr

lw $a1, 0($a1) # a0 = address that WordPtr is pointing to

add $s0, $zero, $zero #s0 = i=0

L1:

add $t1, $s0, $a1 #t1 = s0 + a1 = (WordPtr + i)

# = address of \*(WordPtr + i)

lb $t2, 0($t1) #t2 = value at t1 = \*(WordPtr + i)

add $t3, $s0, $a0 #t3 = s0 + a0 = (newWordPtr + i)

# = address of \*(newWordPtr + i)

sb $t2, 0($t3) # \*(newWordPtr + i) = t2 = \*(WordPtr + i)

beq $t2, $zero, end\_of\_strcpy #if \*(WordPtr + i) == 0 == '\0', exit

nop

addi $s0, $s0, 1 #s0=s0 + 1 <-> i=i+1

j L1 #next character

end\_of\_strcpy:

nop

j end

#------------------------------------------- END-----------------------------------------

end:

1. ***BÀI PHỤ(BÀI 7)***

* Ý tưởng thực hiện bài toán:
* Nhập dòng lệnh hợp ngữ vào chương trình.
* Thực hiện cắt chuỗi ra thành các thành phần nhỏ qua dấu cách ‘ ‘ hay ký tự dấu phẩy ‘,’.
* Kiểm tra tên lệnh có trong danh sách các lệnh không, ví dụ beq thỏa mãn.
* Nếu thỏa mãn thì kiểm tra tiếp các thanh ghi, số lượng thanh ghi hay hằng số (tùy vào dạng lệnh I hay R) có tương ứng với danh sách thanh ghi hay hằng số mà lệnh yêu cầu không.
* Nếu thỏa mãn tất cả các yêu cầu trên thì lệnh hợp ngữ là hợp lệ và ngược lại thì là không hợp lệ.
* Ví dụ: beq $t1, $t2, label thỏa mãn vì beq là một lệnh hợp ngữ, với danh sách tham số thanh ghi là $t1 và $t2 để so sánh và nhãn label để nhảy nếu thỏa mãn điều kiện là hợp lệ, từ đó cấu thành câu lệnh hợp ngữ hợp lệ.