TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện công nghệThông tin & Truyền thông

BÁO CÁO FINAL PROJECT

IT3280

Thực hành kiến trúc máy tính

Giảng viên : Lê Bá Vui

Thành viên nhóm :

Lê văn Bảo 20205057

Đặng Quang Đạt 20205064

Hà Nội,22/7/2022

1. Bài 6 : Hàm cấp phát bộ nhớ malloc()

Thực hiện : Đặng Quang Đạt

MSSV : 20205064

1.1 Phân tích bài toán

Chương trình cho bên dưới là hàm malloc(), kèm theo đó là ví dụ minh họa, được viết bằng hợp ngữ 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 đó, hãy hoàn thiện chương trình như sau: (Lưu ý, ngoài viết các hàm đó, cần viết thêm một số ví dụ minh họa để thấy việc sử dụng hàm đó như thế nào)

1) Việc cấp phát bộ nhớ kiểu word/mảng kiểu 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.
2) Viết hàm lấy giá trịcủa biến con trỏ.
3) Viết hàm lấy địa chỉ biến con trỏ.
4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ
6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
7) 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) Sốdòng
   c) Sốcột
8) Tiếp theo câu 7, hãy viết 2 hàm get Array[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i cột j của mảng.

1.2 Ý tưởng thuật toán

Giải thích cách làm:

1. Việc cấp phát bộ nhớ kiểu word/mảng kiểu 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.

Sửa phần tử của các con trỏ thành 4 byte ($a2)

2. Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ

   Duyệt từ địa chỉ trống đầu tiên đến địa chỉ Sys_MyFreeSpace, cho giá trị = 0 và cho giá trị dữ liệu trong ngăn xếp $a0 là địa chỉ các biến con trỏ = 0

3. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.

   Duyệt từ địa chỉ trống đầu tiên đến địa chỉ Sys_MyFreeSpace, mỗi lần lặp thì cộng vào biến s5 4 byte. Kết quả cuối cùng là giá trị s5

4. 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

   Lấy số dòng x số cột x4 rồi cộng vào con trỏ trống đầu tiên sẽ cấp phát đủ bộ nhớ cho mảng\

5. Tiếp theo câu 7, hãy viết 2 hàm get Array[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i cột j của mảng.

   Nhập i, j

   Lấy (i x số cột + j) x4 rồi cộng và biến Sys_MyFreeSpace để lưu giá trị của phần tử hàng i, cột j của mảng

1.3 Source code

```
 .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 kieu Word
ArrayPtr: .word 0

Nhapdulieu: .asciiz "Nhap du lieu : \n"

GtriChar: .asciiz "Gia tri bien con tro Char: "
GtriByte: .asciiz "Gia tri bien con tro Byte: "
GtriWord: .asciiz "Gia tri bien con tro Word: "
DiachiChar: .asciiz "Dia chi con tro Char: "
DiachiByte: .asciiz "Dia chi con tro Byte: "
```

```
DiachiWord: .asciiz "Dia chi con tro Word: "
Nhaphangi: .asciiz "\nA[i][j]\nNhap so hang i: "
Nhapcotj: .asciiz "Nhap so cot j: "
NhapAij: .asciiz "\nNhap A[i][j]\n"
XuatAij: .asciiz "\nXuat A[i][j]\n"
Nhapi:  .asciiz "Nhap i: "
Nhapj:  .asciiz "Nhap j: "
Tieptuc: .asciiz "Tiep tuc?"
Ketqua: .asciiz "A[i][j] = "
Bonho_Capphat: .asciiz "Luong bo nho da cap phat (byte): "
.kdata
Sys_TheTopOfFree: .word  1
Sys_MyFreeSpace:
.text
#Khoi tao vung nho cap phat dong
        li      $s5, 0
        jal     SysInitMem
        la      $a0, CharPtr
        addi    $a1, $zero, 3
        addi    $a2, $zero, 4
        jal     malloc
        jal     NhapDulieu
        la      $a0, BytePtr
        addi    $a1, $zero, 2
        addi    $a2, $zero, 4
        jal     malloc
        jal     NhapDulieu
        la      $a0, WordPtr
        addi    $a1, $zero, 1
        addi    $a2, $zero, 4
        jal     malloc
        jal     NhapDulieu
        j       Giatri

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 laket qua tra ve cua ham
        mul     $t7, $a1,$a2    #Tinh kich thuoc cua mang can cap
phat
        add     $t6, $t8, $t7   #Tinh dia chi dau tien controng
        sw      $t6, 0($t9)     #Luu tro lai dia chi dau tien do vao
bien Sys_TheTopOfFree
        jr  $ra
NhapDulieu:
        li      $s3, 0
        la      $a0, Nhapdulieu
        li      $v0, 4
        syscall
        LapDl:
        li      $v0, 5
        syscall
        sw      $v0, 0($t8)
        addi    $t8, $t8, 4
        addi    $s3, $s3, 1
        beq     $s3, $a1, DungLap
        j       LapDl
        DungLap:
        jr      $ra

#2.Ham lay gia tri bien con tro
Giatri:
        la      $a0, CharPtr
        lw      $t8, 0($a0)
        lw      $t5, 0($t8)
        la      $a0, GtriChar
        li      $v0, 56
        move    $a1, $t5
        syscall

        la      $a0, BytePtr
        lw      $t8, 0($a0)
        lw      $t5, 0($t8)
        la      $a0, GtriByte
        li      $v0, 56
        move    $a1, $t5
        syscall
```

```
        la      $a0, WordPtr
        lw      $t8, 0($a0)
        lw      $t5, 0($t8)
        la      $a0, GtriWord
        li      $v0, 56
        move    $a1, $t5
        syscall
#3. Ham lay dia chi bien con tro
        la      $a0, CharPtr
        lw      $t8, 0($a0)
        la      $a0, DiachiChar
        li      $v0, 56
        move    $a1, $t8
        syscall

        la      $a0, BytePtr
        lw      $t8, 0($a0)
        la      $a0, DiachiByte
        li      $v0, 56
        move    $a1, $t8
        syscall

        la      $a0, WordPtr
        lw      $t8, 0($a0)
        la      $a0, DiachiWord
        li      $v0, 56
        move    $a1, $t8
        syscall
#5. Ham giai phong bo nho da cap phat
        la      $a0, Sys_MyFreeSpace
        la      $a1, Sys_TheTopOfFree
        lw      $a2, 0($a1)

        loop:
        sw      $0, 0($a2)
        beq     $a2, $a0, next
        subi    $a2, $a2, 4
        j       loop
        next:

        la      $a0, CharPtr
        sw      $zero, 0($a0)
```

```
        sw      $zero, 4($a0)
        sw      $zero, 8($a0)
        la      $a0, WordPtr
        lw      $t8, 0($a0)
        la      $a0, DiachiWord
        li      $v0, 56
        move    $a1, $t8
        syscall
```

#6. Ham tinh luong bo nho da cap phat

```
        la      $a0, Sys_MyFreeSpace
        la      $a1, Sys_TheTopOfFree
        lw      $a2, 0($a1)

        loop1:
        beq     $a2, $a0, next1
        addi    $s5, $s5, 4
        subi    $a2, $a2, 4
        j       loop1
        next1:
        la      $a0, Bonho_Capphat
        move    $a1, $s5
        li      $v0, 56
        syscall
```

#7. Ham malloc2

```
        jal   SysInitMem
        la   $a0, ArrayPtr
        #ham nhap so dong i
        la      $a0, Nhaphangi
        li      $v0, 4
        syscall
        li      $v0, 5
        syscall
        move    $a1, $v0          #so dong
        #Ham nhap so cot j
        la      $a0, Nhapcotj
        li      $v0, 4
        syscall
        li      $v0, 5
        syscall
        move    $a2, $v0          #so cot
        addi    $a3, $zero, 4
```

```
	jal malloc2
	j	cau8
	malloc2:
	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 laket qua tra ve cua ham
	mul	$t7, $a1,$a2	#Tinh kich thuoc cua mang can cap
phat
	mul	$t5, $t7, $a3
	add	$t6, $t8, $t5	#Tinh dia chi dau tien controng
	sw	$t6, 0($t9)	#Luu tro lai dia chi dau tien do vao
bien Sys_TheTopOfFree
	jr	$ra
#8. Ham get, set
cau8:
	#Ham set
	NhapDulieu1:
	la	$t8, Sys_MyFreeSpace
	la	$a0, NhapAij
	li	$v0, 4
	syscall
	#Nhap hang i
	la	$a0, Nhapi
	li	$v0, 4
	syscall
	laptimi:
	li	$v0, 5
	syscall
	slt	$s4, $v0, $a1
	bne	$s4, $0, hetlaptimi
	j	laptimi
	hetlaptimi:
	move	$s1, $v0	#hang i luu vao s1

	#Nhap cot j
	la	$a0, Nhapj
	li	$v0, 4
	syscall
	laptimj:
	li	$v0, 5
	syscall
	slt	$s4, $v0, $a2
```

```
bne     $s4, $0, hetlaptimj
j       laptimj
hetlaptimj:
move    $s2, $v0        #cot j luu vao s2

la      $a0, Ketqua
li      $v0, 4
syscall
li      $v0, 5
syscall
mul     $s3, $s1, $a2
add     $s3, $s3, $s2
mul     $s3, $s3, $a3
add     $t8, $t8, $s3
sw      $v0, 0($t8)

la      $a0, Tieptuc
li      $v0, 50
syscall
beq     $a0, $0, NhapDulieu1
j       XuatDulieu

XuatDulieu:     #ham get

la      $a0, XuatAij
li      $v0, 4
syscall
la      $t8, Sys_MyFreeSpace
#Nhap hang i
la      $a0, Nhapi
li      $v0, 4
syscall
laptimi1:
li      $v0, 5
syscall
slt     $s4, $v0, $a1
bne     $s4, $0, hetlaptimi1
j       laptimi1
hetlaptimi1:
move    $s1, $v0        #hang i luu vao s1

#Nhap cot j
la      $a0, Nhapj
```

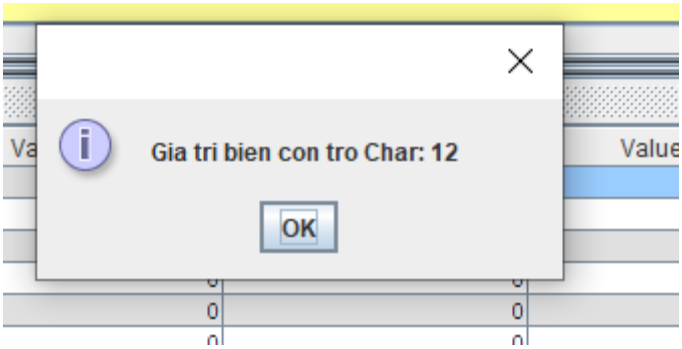
```
        li      $v0, 4
        syscall
        laptimj1:
        li      $v0, 5
        syscall
        slt     $s4, $v0, $a2
        bne     $s4, $0, hetlaptimj1
        j       laptimj1
        hetlaptimj1:
        move    $s2, $v0            #cot j luu vao s2

        la      $a0, Ketqua
        li      $v0, 4
        syscall
        mul     $s3, $s1, $a2
        add     $s3, $s3, $s2
        mul     $s3, $s3, $a3
        add     $t8, $t8, $s3
        lw      $a0, 0($t8)
        li      $v0, 1
        syscall
        la      $a0, Tieptuc
        li      $v0, 50
        syscall
        beq     $a0, $0, XuatDulieu
        j       exit
exit:
```

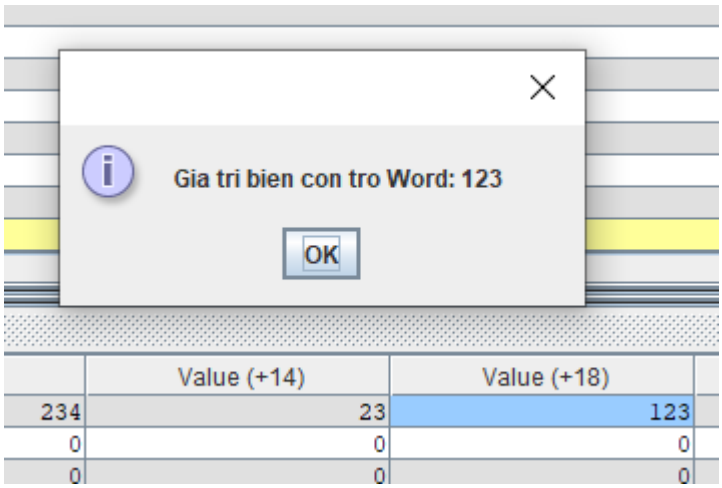
1.4 Kết quả chạy chương trình

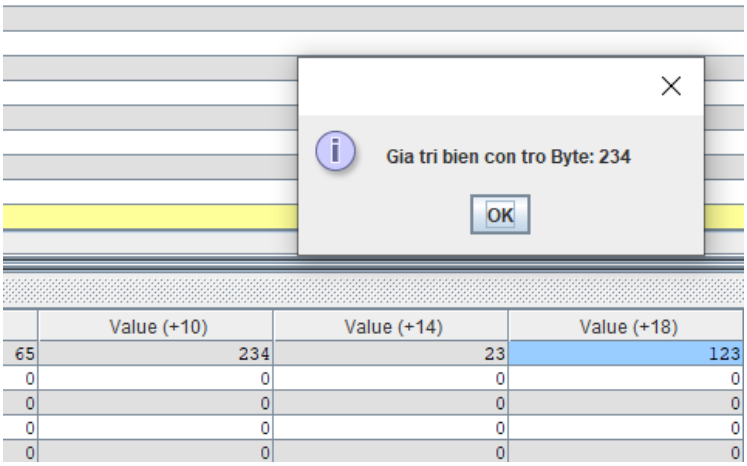
1. Việc cấp phát bộ nhớ kiểu word/mảng kiểu word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.

   Sửa phần tử của các con trỏ thành 4 byte ($a2)

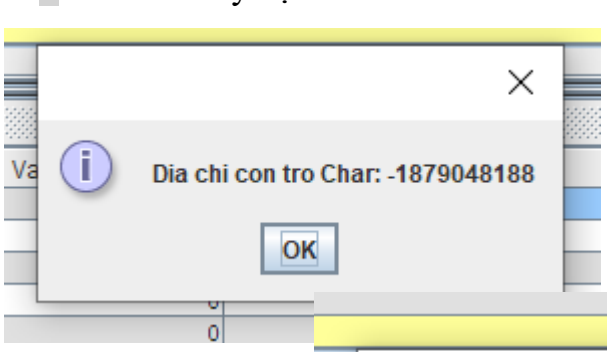
2.Viết hàm lấy giá trịcủa biến con trỏ.

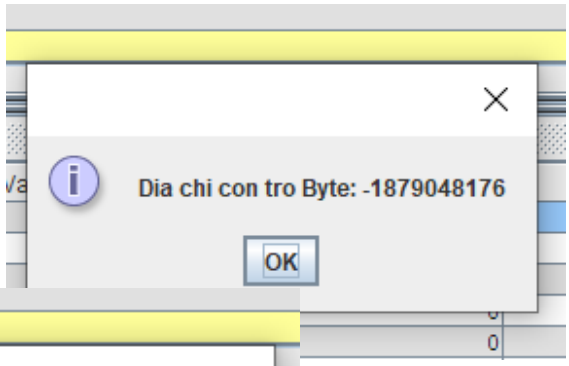


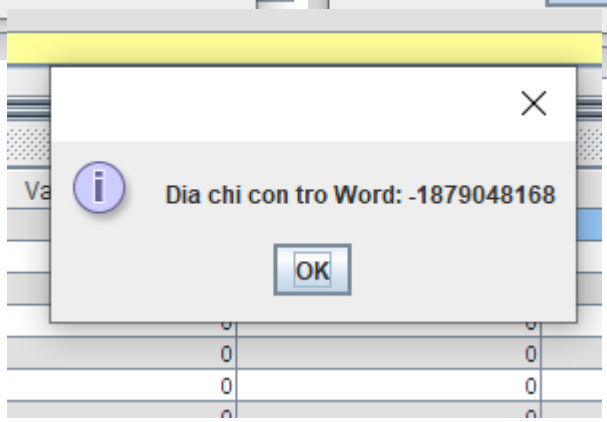




3Viết hàm lấy địa chỉ biến con trỏ.

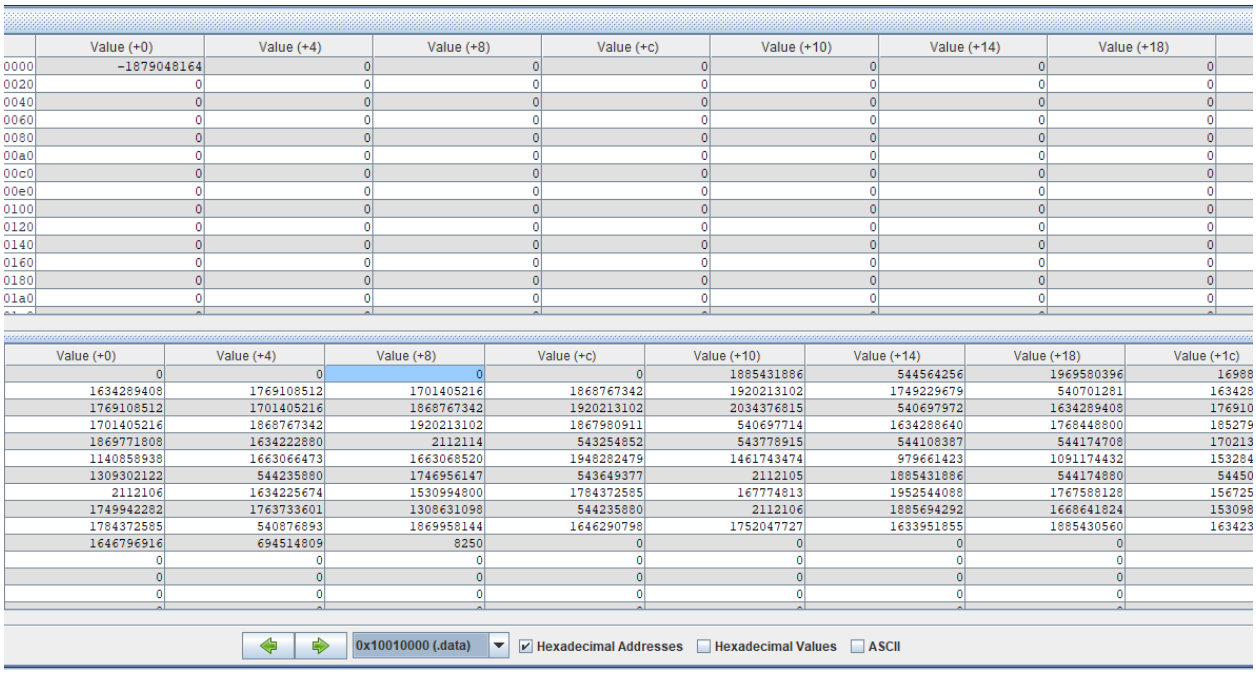






5.Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ

Duyệt từ địa chỉ trống đầu tiên đến địa chỉ Sys_MyFreeSpace, cho giá trị = 0 và cho giá trị dữ liệu trong ngăn xếp $a0 là địa chỉ các biến con trỏ = 0



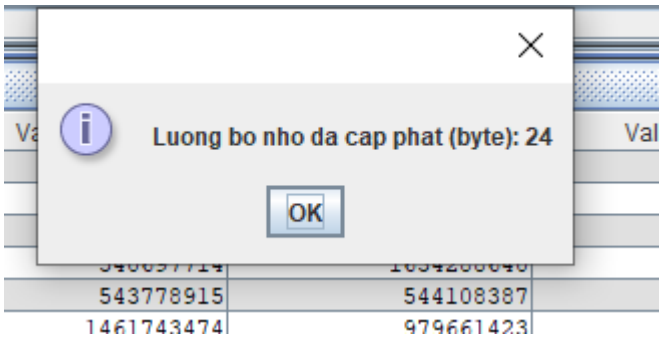
	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0000	-1879048164	0	0	0	0	0	0
0020	0	0	0	0	0	0	0
0040	0	0	0	0	0	0	0
0060	0	0	0	0	0	0	0
0080	0	0	0	0	0	0	0
00a0	0	0	0	0	0	0	0
00c0	0	0	0	0	0	0	0
00e0	0	0	0	0	0	0	0
0100	0	0	0	0	0	0	0
0120	0	0	0	0	0	0	0
0140	0	0	0	0	0	0	0
0160	0	0	0	0	0	0	0
0180	0	0	0	0	0	0	0
01a0	0	0	0	0	0	0	0

Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0	0	0	0	1885431886	544564256	1969580396	16988
1634289408	1769108512	1701405216	1868767342	1920213102	1749229679	540701281	163428
1769108512	1701405216	1868767342	1920213102	2034376815	540697972	1634289408	176910
1701405216	1868767342	1920213102	1867980911	540697714	1634288640	1768448800	185279
1869771808	1634222880	2112114	543254852	543778915	544108387	544174708	170213
1140858938	1663066473	1663068520	1948282479	1461743474	979661423	1091174432	153284
1309302122	544235880	1746956147	543649377	2112105	1885431886	544174880	54450
2112106	1634225674	1530994800	1784372585	167774813	1952544088	1767588128	156725
1749942282	1763733601	1308631098	544235880	2112106	1885694292	1668641824	153098
1784372585	540876893	1869958144	1646290798	1752047727	1633951855	1885430560	163423
1646796916	694514809	8250	0	0	0	0	
0	0	0	0	0	0	0	
0	0	0	0	0	0	0	
0	0	0	0	0	0	0	

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

6.Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.

Duyệt từ địa chỉ trống đầu tiên đến địa chỉ Sys_MyFreeSpace, mỗi lần lặp thì cộng vào biến s5 4 byte. Kết quả cuối cùng là giá trị s5



7.Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm

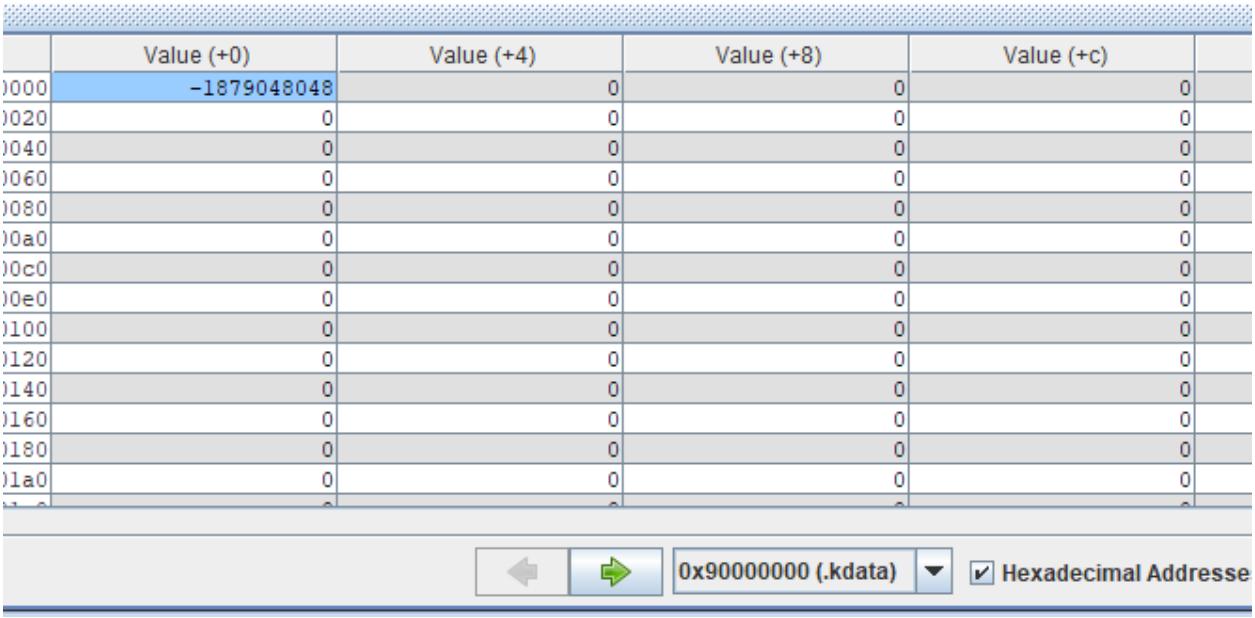
a) Địa chỉ đầu của mảng

d) Số dòng

e) Số cột

Lấy số dòng x số cột x4 rồi cộng vào con trỏ trống đầu tiên sẽ cấp phát đủ bộ nhớ cho mảng

```
A[i][j]
Nhap so hang i: 5
Nhap so cot j: 7
```

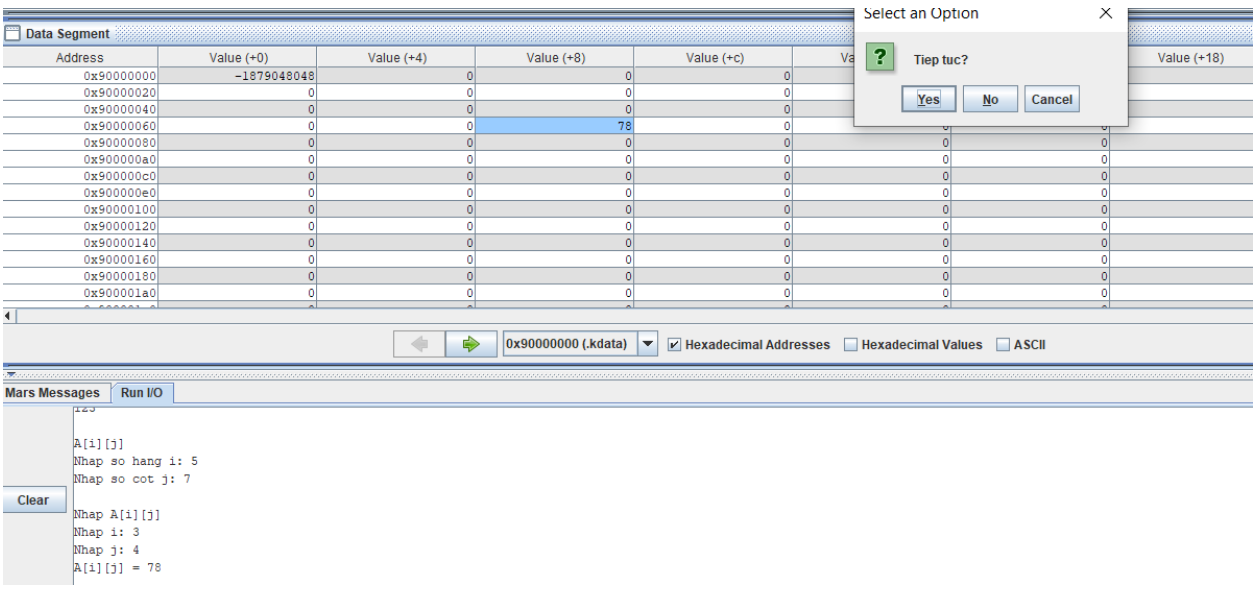


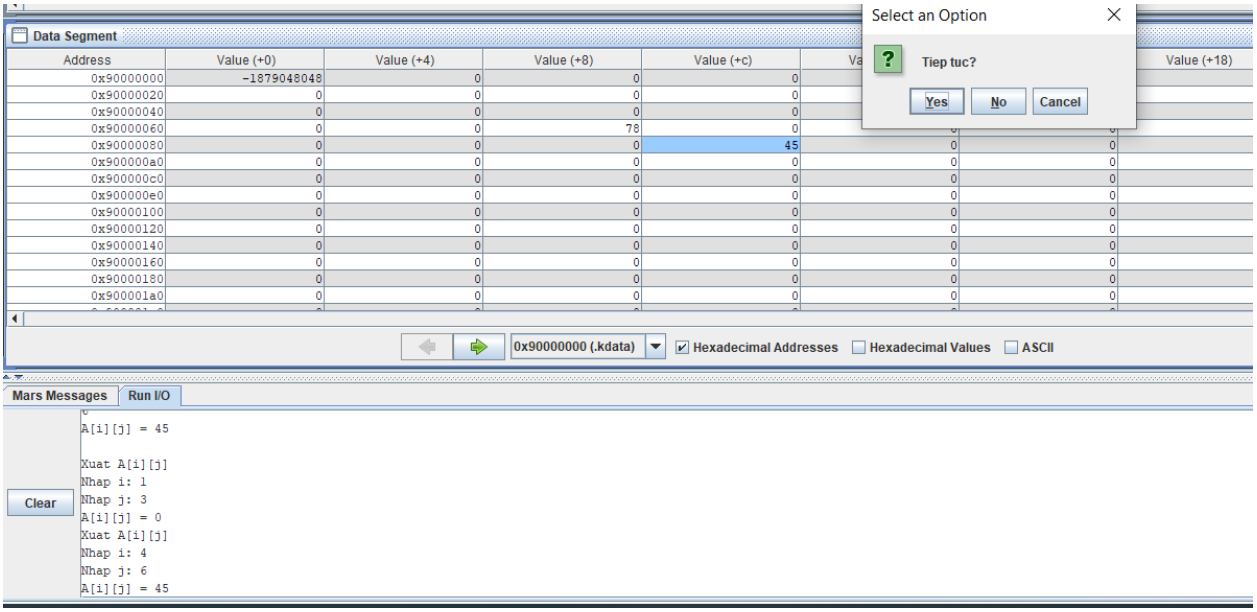
	Value (+0)	Value (+4)	Value (+8)	Value (+c)
)000	-1879048048	0	0	0
)020	0	0	0	0
)040	0	0	0	0
)060	0	0	0	0
)080	0	0	0	0
)0a0	0	0	0	0
)0c0	0	0	0	0
)0e0	0	0	0	0
)100	0	0	0	0
)120	0	0	0	0
)140	0	0	0	0
)160	0	0	0	0
)180	0	0	0	0
)1a0	0	0	0	0

8.Tiếp theo câu 7, hãy viết 2 hàm get Array[i][j] và setArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng i cột j của mảng.

Nhập i, j

Lấy (i x số cột + j) x4 rồi cộng và biến Sys_MyFreeSpace để lưu giá trị của phần tử hàng i, cột j của mảng





2. Bài 7 : Chương trình kiểm tra cú pháp lệnh MIPS

Thực hiện : Lê Văn Bảo

MSSV: 20205057

2.1Phân tích bài toán

Đề bài:

Trình biên dịch của bộ xử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không,

rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ beq s1,31,t4

- Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thị thông báo “opcode: beq, hợp lệ”

- Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.

Yêu cầu bài toán :

Hiện ra thông báo lệnh nhập vào có đúng cú pháp hay không.

2.2 Ý tưởng thuật toán

Cú pháp là: ‘opcode toán_hạng_1,toán_hạng_2,toán_hạng 3 (có thể có hoặc không)’

Từ cú pháp ta có ý tưởng sẽ duyệt các ký từ trong lệnh nhập vào đến khi gặp dấu ‘space’ sẽ là phần opcode, sẽ dùng nó so sánh với các lệnh có sẵn trong cấu trúc các lệnh . Ta sẽ đánh dấu vị trí opcode và từ đó sẽ duyệt các toán hạng gạp các dấu “,” để nhận biết các toán hạng và so sánh với các toán hạng có sẵn.

2.3 Source code

```
.data
        mess_input: .asciiz "Nhap lenh: "
        mess_error: .asciiz "Loi cu phap!\n"
        mess_not_found: .asciiz "Khong tim duoc lenh nay!\n"
        mess_correct: .asciiz "\nLenh vua nhap dung voi cu phap\n"
        mess_opcode: .asciiz "Opcode: "
        mess_operand: .asciiz "Toan hang: "
        mess_valid: .asciiz "hop le.\n"
        mess_continue: .asciiz "Ban muon tiep tuc chuong
        trinh?(0.Yes/1.No)"
        command: .space 100
        opcode: .space 10
        token: .space 20
        number: .space 15
```

```
        label: .space 30
        # quy luat cua CommandData: opcode co do dai = 5 byte
        # moi lenh co 3 toan hang va chi co 4 loai la: thanh ghi = 1, hang
        so nguyen =2, dinh danh = 3 hoac khong co = 0.
        CommandData: .asciiz
        "or***1111;xor**1111;lui**1201;jr***1001;jal**3002;addi*1121
        ;add**1111;sub**1111;ori**1121;and**1111;beq**1132;bne**11
        32;j****3002;nop**0001;"
        CharData: .asciiz
        "1234567890qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDF
        GHJKLZXCVBNM_`~[]{}\|;':<>/?,.!@#$%^&*(()+-="
        TokenData: .asciiz "$zero $at  $v0  $v1  $a0  $a1  $a2  $a3
        $t0  $t1  $t2  $t3  $t4  $t5  $t6  $t7  $s0  $s1  $s2  $s3  $s4
        $s5  $s6  $s7  $t8  $t9  $k0  $k1  $gp  $sp  $fp  $ra  $0  $1
        $2  $3  $4  $5  $7  $8  $9  $10  $11  $12  $13  $14  $15
        $16  $17  $18  $19  $20  $21  $22  $21  $22  $23  $24  $25
        $26  $27  $28  $29  $30  $31  "

.text
# nhap lenh tu ban phim
enter_input:
        li $v0, 4
        la $a0, mess_input
        syscall
        li $v0, 8
        la $a0, command
        li $a1, 100
        syscall

main:
        li $t2, 0 #i
#doc opcode tu du lieu dau vao
Read_Opcode:
        la $a1, opcode
        add $t3, $a0, $t2
        add $t4, $a1, $t2
        lb $t1, 0($t3)
        sb $t1, 0($t4)
        beq $t1, ' ', done # gap ki tu ' ' thi luu vao opcode
        beq $t1, 0, done
        addi $t2, $t2, 1
        j Read_Opcode
done:
```

```
        li $t7,-10
        la $a2, CommandData

#xu li opcode
Processing_Opcode:
        li $t1, 0 # i
        li $t2, 0 # j
        addi $t7,$t7,10
        add $t1,$t1,$t7
        #so sanh opcode
        Compare_Opcode:
        add $t3, $a2, $t1 # t3 la con tro cua CommandData
        lb $s0, 0($t3)
        beq $s0, 0, notFound # khong tim thay opcode trong CommandData
        beq $s0, '*', Check_Opcode # gap ki tu  '*' -> kiem tra dau cach
        add $t4, $a1, $t2
        lb $s1, 0($t4)
        bne $s0,$s1,Processing_Opcode # so sanh ki tu
        addi $t1,$t1,1 # i+=1
        addi $t2,$t2,1 # j+=1
        j Compare_Opcode

        Check_Opcode:
        add $t4, $a1, $t2
        lb $s1, 0($t4)
        bne $s1, ' ', Check_error
        End_Opcode:
        add $t9,$t9,$t2 # t9 la vi tri opcode
        li $v0, 4
        la $a0, mess_opcode
        syscall
        li $v0, 4
        la $a0, opcode
        syscall
        li $v0, 4
        la $a0, mess_valid
        syscall
        j Read_operand_1
        # check '\n'
        Check_error:
        bne $s1, 10, notFound
        j End_Opcode
```

```
#xu li toan hang
Read_operand_1:
	# xac dinh kieu toan hang trong CommanData
	# t7 dang chua vi tri khuon dang lenh trong CommanData
	li $t1, 0
	addi $t7, $t7, 5
	add $t1, $a2, $t7 # a2 chua dia chi CommandData
	lb $s0, 0($t1)
	addi $s0,$s0,-48 #char -> int
	li $t8, 1
	beq $s0, $t8, Check_Token_Register
	li $t8, 2
	beq $s0, $t8, Check_Integer
	li $t8, 3
	beq $s0, $t8, Check_Label
	li $t8, 0
	beq $s0, $t8, Check_Null_Token
	j end

#check token
Check_Token_Register:
	la $a0, command
	la $a1, token
	li $t1, 0
	li $t2, -1
	addi $t1, $t9, 0
	Read_Token:
		addi $t1, $t1, 1 # i
		addi $t2, $t2, 1 # j
		add $t3, $a0, $t1
		add $t4, $a1, $t2
		lb $s0, 0($t3)
		add $t9, $zero, $t1 # vi tri toan hang sau opcode trong
command
		beq $s0, ',',Read_Token_Done
		beq $s0, 0,Read_Token_Done
		sb $s0, 0($t4)
		j Read_Token

	Read_Token_Done:
		sb $s0, 0($t4) # luu ',' de compare
		li $t1, -1 # i
		li $t2, -1 # j
```

```
                li $t4, 0
                li $t5, 0
                add $t2, $t2, $k1
                la $a1, token
                la $a2, TokenData
                j Compare_Token

Compare_Token:
        addi $t1,$t1,1
        addi $t2,$t2,1
        add $t4, $a1, $t1
        lb $s0, 0($t4)
        beq $s0, 0, end
        add $t5, $a2, $t2
        lb $s1, 0($t5)
        beq $s1, 0, notFound
        beq $s1, 32, Check_End_Token
        bne $s0,$s1, jump
        j Compare_Token

        Check_End_Token:
                beq $s0, 44, End_Token
                beq $s0, 10, End_Token
                j Token_error
        jump:
                addi $k1,$k1,6
                j Read_Token_Done
        End_Token:
                la $a0, mess_operand
                syscall
                li $v0, 4
                la $a0, token
                syscall
                li $v0, 4
                la $a0, mess_valid
                syscall
                addi $v1, $v1, 1 # so toan hang da xu li
                li $k1, 0 # reset buoc nhay
                beq $v1, 1, Read_Operand_2
                beq $v1, 2, Read_Operand_3
                j end
        Token_error:
                j notFound
```

```
#hang so nguyen
Check_Integer:
	la $a0, command
	la $a1, number
	li $t1, 0
	li $t2, -1
	addi $t1, $t9, 0
	Read_Number:
		addi $t1, $t1, 1 # i
		addi $t2, $t2, 1 # j
		add $t3, $a0, $t1
		add $t4, $a1, $t2
		lb $s0, 0($t3)
		add $t9, $zero, $t1 # vi tri toan hang theo trong command
		beq $s0, 44, Read_Number_Done # gap dau ','
		beq $s0, 0, Read_Number_Done
		sb $s0, 0($t4)
		j Read_Number
	Read_Number_Done:
		sb $s0, 0($t4) # luu ',' de compare
		li $t1, -1 # i
		li $t4, 0
		la $a1, number
		j Compare_Number
Compare_Number:
	addi $t1, $t1, 1
	add $t4, $a1, $t1
	lb $s0, 0($t4)
	beq $s0, 0, end
	beq $s0, 45, Compare_Number # bo dau '-'
	beq $s0, 10, End_Compare_Number
	beq $s0, 44, End_Compare_Number
	li $t2, 48
	li $t3, 57
	slt $t5, $s0, $t2
	bne $t5, $zero, Number_Error
	slt $t5, $t3, $s0
	bne $t5, $zero, Number_Error
	j Compare_Number

	End_Compare_Number:
		la $a0, mess_operand
```

```
                syscall
                li $v0, 4
                la $a0, number
                syscall
                li $v0, 4
                la $a0, mess_valid
                syscall
                addi $v1, $v1, 1 # so toan hang da xu li
                li $k1, 0 # reset buoc nhay
                beq $v1, 1, Read_Operand_2
                beq $v1, 2, Read_Operand_3
                j end
        Number_Error:
                j notFound

#check ten ham
Check_Label:
        la $a0, command
        la $a1, label
        li $t1, 0
        li $t2, -1
        addi $t1, $t9, 0
        Read_Label:
                addi $t1, $t1, 1 # i
                addi $t2, $t2, 1 # j
                add $t3, $a0, $t1
                add $t4, $a1, $t2
                lb $s0, 0($t3)
                add $t9, $zero, $t1 # vij tri tiep theo trong command
                beq $s0, 44, Read_Label_Done # gap dau ','
                beq $s0, 0, Read_Label_Done
                sb $s0, 0($t4)
                j Read_Label
        Read_Label_Done:
                sb $s0, 0($t4) # luu ',' de compare
                loopj:
                li $t1, -1 # i
                li $t2, -1 # j
                li $t4, 0
                li $t5, 0
                add $t1, $t1, $k1
                la $a1, label
                la $a2, CharData
```

```
                j Compare_Label
Compare_Label:
        addi $t1,$t1,1
        add $t4, $a1, $t1
        lb $s0, 0($t4)
        beq $s0, 0, end
        beq $s0, 10, End_Compare_Label
        beq $s0, 44, End_Compare_Label
        loop:
        addi $t2,$t2,1
        add $t5, $a2, $t2
        lb $s1, 0($t5)
        beq $s1, 0, Error_Label
        beq $s0, $s1, jumpIdent # so sanh ki tu tiep theo trong label
        j loop # tiep tuc so sanh ki tu tiep theo theo trong CharData

        jumpIdent:
                addi $k1,$k1,1
                j loopj

        End_Compare_Label:
                la $a0, mess_operand
                syscall
                li $v0, 4
                la $a0, label
                syscall
                li $v0, 4
                la $a0, mess_valid
                syscall
                addi $v1, $v1, 1 #so toan hang da xu li
                li $k1, 0 # reset buoc nhay
                beq $v1, 1, Read_Operand_2
                beq $v1, 2, Read_Operand_3
                j end
        Error_Label:
                j notFound

#kiem tra khong co toan hang
Check_Null_Token:
        la $a0, command
        li $t1, 0
        li $t2, 0
        addi $t1, $t9, 0
```

```
        add $t2, $a0, $t1
        lb $s0, 0($t2)
        addi $v1, $v1, 1 #so toan hang da xu li
        li $k1, 0 # reset b??c nh?y
        beq $v1, 1, Read_Operand_2
        beq $v1, 2, Read_Operand_3
#<--check Token Register 2-->
Read_Operand_2:
        # xac dinh kieu toan hang trong CommanData
        # t7 dang chua vi tri khuon dang lenh trong CommanData
        li $t1, 0
        la $a2, CommandData
        addi $t7, $t7, 1 # chuyen den vi tri toan hang 2 trong CommandData
        add $t1, $a2, $t7 # a2 chua dia chi CommandData
        lb $s0, 0($t1)
        addi $s0,$s0,-48 # chuyen tu char -> int
        li $t8, 1 # thanh ghi = 1
        beq $s0, $t8, Check_Token_Register
        li $t8, 2 # hang so nguyen = 2
        beq $s0, $t8, Check_Integer
        li $t8, 3 # dinh danh = 3
        beq $s0, $t8, Check_Label
        li $t8, 0 # khong co toan hang = 0
        beq $s0, $t8, Check_Null_Token
        j end

Read_Operand_3:
        # xac dinh kieu toan hang trong CommanData
        # t7 dang chua vi tri khuon dang lenh trong CommanData
        li $t1, 0
        la $a2, CommandData
        addi $t7, $t7, 1
        add $t1, $a2, $t7
        lb $s0, 0($t1)
        addi $s0,$s0,-48 #char -> int
        li $t8, 1
        beq $s0, $t8, Check_Token_Register
        li $t8, 2
        beq $s0, $t8, Check_Integer
        li $t8, 3
        beq $s0, $t8, Check_Label
        li $t8, 0
        beq $s0, $t8, Check_Null_Token
```

```
        j end
continue: # lap lai chuong trinh.
        li $v0, 4
        la $a0, mess_continue
        syscall
        li $v0, 5
        syscall
        add $t0, $v0, $zero
        beq $t0, $zero, resetAll
        j TheEnd
resetAll:
        li $v0, 0
        li $v1, 0
        li $a0, 0
        li $a1, 0
        li $a2, 0
        li $a3, 0
        li $t0, 0
        li $t1, 0
        li $t2, 0
        li $t3, 0
        li $t4, 0
        li $t5, 0
        li $t6, 0
        li $t7, 0
        li $t8, 0
        li $t9, 0
        li $s0, 0
        li $s1, 0
        li $s2, 0
        li $s3, 0
        li $s4, 0
        li $s5, 0
        li $s6, 0
        li $s7, 0
        li $k0, 0
        li $k1, 0
        j enter_input
notFound:
        li $v0, 4
        la $a0, mess_not_found
        syscall
        j TheEnd
```

```
error:
        li $v0, 4
        la $a0, mess_error
        syscall
        j TheEnd
end:
        li $v0, 4
        la $a0, mess_correct
        syscall
        j continue
TheEnd:
        li $v0,10
        syscall
```

3.4 Kết quả chạy chương trình

Lệnh hợp lệ

```
Nhap lenh: addi $s1,$s1,2910
Opcode: addi hop le.
Toan hang: $s1,hop le.
Toan hang: $s1,hop le.
Toan hang: 2910
hop le.

Lenh vua nhap dung voi cu phap
```

Lệnh không hợp lệ

```
Nhap lenh: addi $s1,100,4s1
Opcode: addi hop le.
Toan hang: $s1,hop le.
Khong tim duoc lenh nay!
```