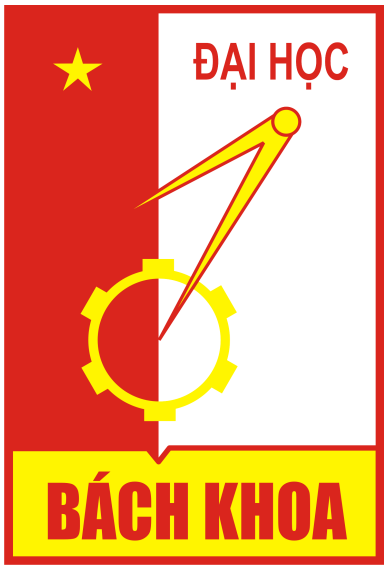
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP CUỐI KÌ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH IT3280

Học kì 20232 - Năm học: 2023 - 2024

Giảng viên hướng dẫn: **ThS. Lê Bá Vui**

Danh sách thành viên: **Nguyễn Đăng Phúc Hưng - 20226084**
Trịnh Quang Minh - 20226092

Mã lớp: **147789**

Hà Nội, 2024

Mục lục

1 Phân công công việc

1.1 Nguyễn Đăng Phúc Hưng

- MSSV: 20226084
- Email: hung.ndp226084@sis.hust.edu.vn
- Bài 6: Hàm cấp phát bộ nhớ malloc()

1.2 Trịnh Quang Minh

- MSSV: 20226092
- Email: minh.tq226092@sis.hust.edu.vn
- Bài 7: Chương trình kiểm tra cú pháp lệnh MIPS

2 HÀM CẤP PHÁT BỘ NHỚ MALLOC()

2.1 Đề bài

6. Hàm cấp phát bộ nhớ malloc()

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 trị của 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. Số dòng
    c. Số cột
8) Tiếp theo câu 7, 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.

2.2 Định hướng cách làm

2.2.1 Ý tưởng

- Chương trình tạo menu lựa chọn cho người dùng, giải quyết các bài toán liên quan đến cấp phát bộ nhớ như cấp phát cho mảng 1 chiều, mảng 2 chiều, bộ nhớ đã cấp phát và copy
- Hiểu rõ nguyên tắc cấp phát bộ nhớ động

2.3 Thuật toán và hàm

2.3.1 Khai báo dữ liệu

```
1     .data
2     CharPtr1: .word 0
3     CharPtr2: .word 0
4     ArrayPtr: .word 0
5     Array2Ptr: .word 0
6     mess1: .asciiz "\n\n1. Mang mot chieu\n"
7     mess2: .asciiz "2. Sao chep mang ky tu\n"
8     mess3: .asciiz "3. Mang hai chieu\n"
9     mess4: .asciiz "4. Giai phong bo nho\n"
10    mess5: .asciiz "5. Hien thi bo nho\n"
11    mess6: .asciiz "6. Ket thuc chuong trinh\n"
12    mess0.1: .asciiz "So phan tu: "
13    mess0.2: .asciiz "So byte moi phan tu (1 hoac 4): "
14    mess0.3: .asciiz "Nhap phan tu: \n"
15    mess1.1: .asciiz "Gia tri cua con tro: "
16    mess1.2: .asciiz "\nDia chi cua con tro: "
17    mess1.3: .asciiz "\nTong bo nho da cap phat: "
```

```
18      mess2.1: .asciiz "So ky tu toi da: "
19      mess2.2: .asciiz "\nNhap chuoi ky tu: "
20      mess2.3: .asciiz "\nChuoi ky tu duoc copy: "
21      mess3.1: .asciiz "\nSo hang: "
22      mess3.2: .asciiz "\nSo cot: "
23      mess3.3: .asciiz "\n1. getArray[i][j]\n"
24      mess3.4: .asciiz "2. setArray[i][j]\n"
25      mess3.5: .asciiz "3. Thoat\n"
26      mess3.6: .asciiz "\nGia tri cua phan tu: "
27      mess3.01: .asciiz "i = "
28      mess3.02: .asciiz "j = "
29      mess4.1: .asciiz "Da giai phong toan bo bo nho cap phat.\n"
30      select: .asciiz "Lua chon: "
31      errmess: .asciiz "\nSo vua nhap khong hop le.\n"
32      .kdata
33      Sys_TheTopOfFree: .word 1
34      Sys_MyFreeSpace:
35      .text
36      #Khoi tao vung nho cap phat dong
37          jal SysInitMem
```

Các dữ liệu bao gồm các string thông báo tới người dùng và khai báo các biến quản lí bộ nhớ: Trong đó:

- Sys_TheTopOfFree:
  - Biến lưu trữ địa chỉ của vùng nhớ đầu tiên trong vùng nhớ tự do.
  - Đóng vai trò con trỏ, chỉ đến vị trí đầu tiên của vùng nhớ không được sử dụng.
  - Khi bộ nhớ được cấp phát, biến này sẽ được cập nhật để chỉ đến vị trí mới của vùng nhớ tự do, sau khi một phần của nó đã được sử dụng.
- Sys_MyFreeSpace:
  - Biến lưu trữ địa chỉ bắt đầu của vùng bộ nhớ không gian tự do.
  - Đóng vai trò con trỏ chỉ đến vị trí đầu tiên của vùng nhớ có sẵn để cấp phát.
  - Khi cấp phát bộ nhớ, con trỏ này sẽ thay đổi để chỉ đến vùng nhớ tự do còn lại sau khi phân bổ.
- SysInitMem: Khởi tạo vùng nhớ cấp phát động

```
1       SysInitMem:
2           la  $t9, Sys_TheTopOfFree    # Lay con tro chua dau tien con
                trong, khoi tao
3           la  $t7, Sys_MyFreeSpace     # Lay dia chi dau tien con trong,
                khoi tao
4           sw  $t7, 0($t9)      # Luu lai
5           jr  $ra
```

  - Gán địa chỉ còn trống đầu tiên vào $t7, lưu địa chỉ đó vào bộ nhớ của $t9

2.3.2 Chương trình menu

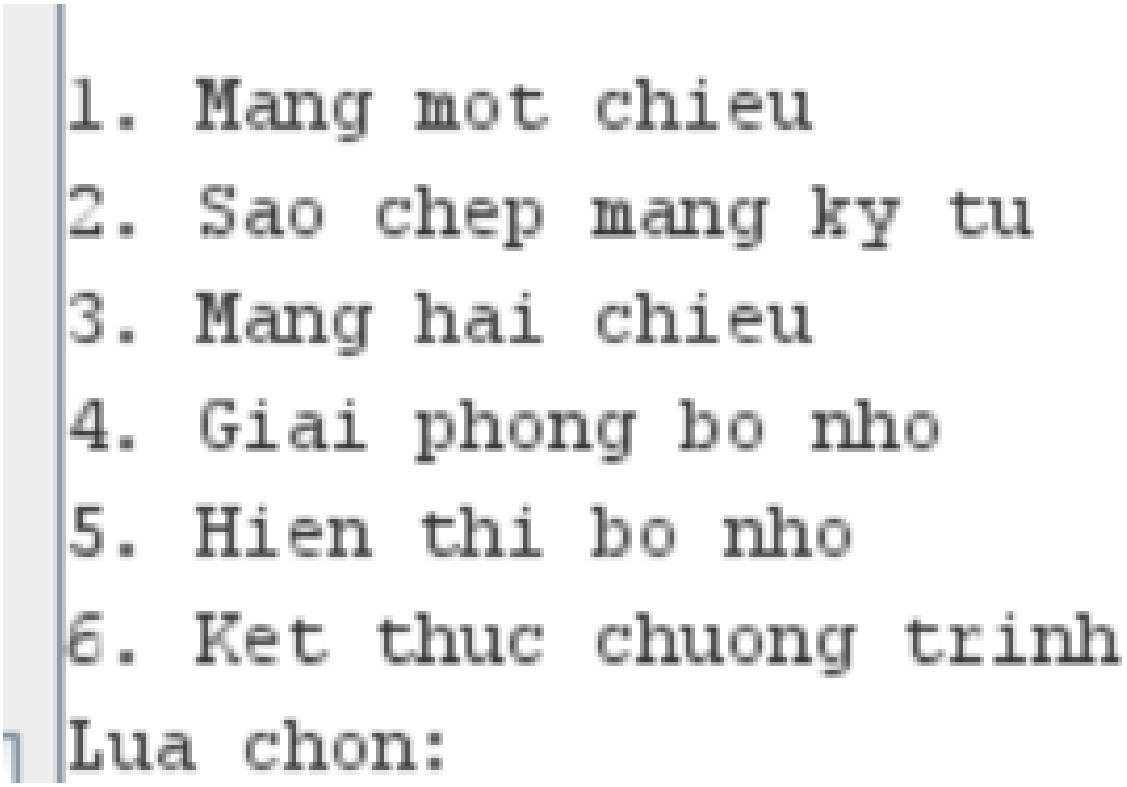
```
1       #Hien thi menu
2       menu:
3           li  $v0, 4
4           la  $a0, mess1
5           syscall
6           la  $a0, mess2
7           syscall
8           la  $a0, mess3
9           syscall
10          la  $a0, mess4
```

```
        syscall
        la  $a0, mess5
        syscall
        la  $a0, mess6
        syscall
        la  $a0, select
        syscall
        li  $v0, 5 #Nhap lua chon
        syscall

    # Process user choice
    beq $v0, 1, case_1
    beq $v0, 2, case_2
    beq $v0, 3, case_3
    beq $v0, 4, case_4
    beq $v0, 5, case_5
    beq $v0, 6, case_6
    j   error   # Handle invalid choice

    error:
    # Handle invalid choice
    li  $v0, 4
    la  $a0, errmess
    syscall
    j   menu
```

- Sử dụng vòng lặp nhập menu và chờ người dùng lựa chọn, lưu vào $v0



- Nếu lựa chọn hợp lệ, chương trình sẽ nhảy đến các chương trình con để xử lí. Nếu không hợp lệ sẽ hiện thông báo lỗi và quay lại menu.

2.3.3 Sửa lỗi không chia hết cho 4 của ví dụ

```
    #1. Sua loi bo nho cua vi du
case_1:
    bne $v0, 1, case_2
```

```
    li  $v0, 4
    la  $a0, mess0.1
    syscall
    li  $v0, 5 #Nhap so phan tu cua day 1 chieu
    syscall
    bltz    $v0, error #Kiem tra so da nhap vao, neu v0 < 0 thi bao loi va
        yeu cau nhap lai
    move    $a1, $v0 #Luu so phan tu vao a1
    li  $v0, 4
    la  $a0, mess0.2
    syscall
    li  $v0, 5 #Nhap kich thuc moi phan tu cua day
    syscall
is1:    beq $v0, 1, ready
is4:    beq $v0, 4, ready
    j   error   #Kiem tra kich thuoc nhap vao, neu kh ng phai 1 hay 4 thi
        bao loi va yeu cau nhap lai
ready:  move    $a2, $v0 #Luu kich thuoc moi phan tu vao a2
    la  $a0, ArrayPtr #Luu dia chi bat dau cua chuoi mot chieu
    jal malloc #Chay ham malloc cho chuoi 1 chieu
    move    $t0, $v0 #Dat dia chi bat dau tra tu malloc vao t0
    li  $v0, 4
    la  $a0, mess0.3
    syscall
    move    $a0, $t0 #Dat a0 = t0
    li  $t0, 0 #Dat t0 = 0
```

- Kiểm tra số phần tử và kích thước phần tử của dãy
- Nếu không thoả mãn số phần tử lớn hơn 0 hoặc kích thước phần tử bằng 1 hoặc 4 thì báo lỗi, còn không thì di chuyển đến hàm malloc

```
malloc:
    la  $t9, Sys_TheTopOfFree
    lw  $t8, 0($t9)     # Lay dia chi dau tien con trong
    bne $a2, 4, initialize  # Neu mang khoi tao co kieu Word, kiem tra
        dia chi dau co dam bao quy tac khong
    andi    $t0, $t8, 0x03      # Lay so du khi chia dia chi trong cho
        4
    beq $t0, 0, initialize  # Neu khong co du, bo qua
    addi    $t8, $t8, 4     # Neu co, tien toi dia chi chia het cho 4
        tiep theo
    subu    $t8, $t8, $t0
initialize:
    sw  $t8, 0($a0) # Cat dia chi do vao bien con tro
    move    $v0, $t8    # Dong thoi la ket 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 con trong
    sw  $t6, 0($t9) # Luu tro lai dia chi dau tien do vao bien
        Sys_TheTopOfFree
    jr  $ra
```

- Hàm malloc, trước hết gán địa chỉ bắt đầu của bộ nhớ còn trống vào biến con trỏ $t8. Nếu mảng khởi tạo là kiểu word (kích thước phần tử bằng 4 byte) thì kiểm tra xem con trỏ có thỏa mãn quy tắc. Đặt $t0 là số dư của con trỏ cho 4. Nếu không dư thì bỏ qua còn nếu có thì tiến tới cho đến khi địa chỉ chia hết cho 4.
- Lưu địa chỉ đã qua kiểm tra vào biến con trỏ. Tính kích thước của mảng cần cập nhật rồi khởi tạo lại địa chỉ bắt đầu của bộ nhớ còn trống rồi quay về.

```
input_loop:
    beq $t0, $a1, input_end #Bat dau vong lap nhap du lieu, ket thuc
        khi t0 = a1
```

```
3           li   $v0, 5
4           syscall
5           bne $a2, 1, byte_4
6       byte_1:
7           sb   $v0, 0($a0)
8           addi     $a0, $a0, 1 #Neu kich thuoc phan tu bang 1 thi con tro
                tien 1 don vi
9           addi     $t0, $t0, 1
10          j    input_loop
11      byte_4:
12          sw   $v0, 0($a0)
13          addi     $a0, $a0, 4 #Neu kich thuoc phan tu bang 4 thi con tro
                tien 4 don vi
14          addi     $t0, $t0, 1
15          j    input_loop
```

- Nếu biến đếm $t0 bằng với số lượng phần tử cần nhập ($a1), nhảy tới nhãn input_end để kết thúc vòng lặp.
- Kiểm tra xem kích thước của từng phần tử trong mảng là 1 byte hay 4 byte. Nếu kích thước không phải 1 byte ($a2 không bằng 1), thì nhảy tới nhãn byte_4 để xử lý.
  - byte_1: Nếu kích thước của từng phần tử là 1 byte, lưu giá trị nhập vào vị trí hiện tại của con trỏ $a0 trong bộ nhớ, sau đó tăng con trỏ $a0 lên 1 byte để trỏ tới phần tử tiếp theo. Tiếp theo, tăng biến đếm $t0 lên 1 và quay lại vòng lặp input_loop.
  - byte_4: Nếu kích thước của từng phần tử là 4 byte, lưu giá trị nhập vào vị trí hiện tại của con trỏ $a0 trong bộ nhớ, sau đó tăng con trỏ $a0 lên 4 byte để trỏ tới phần tử tiếp theo. Tiếp theo, tăng biến đếm $t0 lên 1 và quay lại vòng lặp input_loop.

2.3.4 Lấy giá trị của biến con trỏ

```
1       li   $v0, 4
2       la   $a0, mess1.1
3       syscall
4       la   $a0, ArrayPtr
5       jal getValue
6       move     $a0, $v0
7       li   $v0, 34
8       syscall
9
10      getValue:
11      lw   $v0, 0($a0) # Lay gia tri cua bien con tro trong o nho co dia chi
            luu trong $a0
12      jr   $ra
```

- In ra giá trị con trỏ ArrayPtr
- Load địa chỉ con trỏ ArrayPtr vào thanh ghi $a0
- Hàm getValue lấy giá trị con trỏ đặt vào $v0
- lw $v0, 0($a0): Load dữ liệu từ vị trí mà con trỏ $a0 đang trỏ tới và đặt giá trị này vào thanh ghi $v0

2.3.5 Lấy địa chỉ biến con trỏ

```
    li  $v0, 4
    la  $a0, mess1.2
    syscall
    la  $a0, ArrayPtr
    jal getAddress
    move    $a0, $v0
    li  $v0, 34
    syscall

    getAddress:
        add $v0, $0, $a0    # Lay dia chi tu $a0
        jr  $ra
```

- In ra địa chỉ con trỏ ArrayPtr
- Load địa chỉ con trỏ ArrayPtr vào thanh ghi $a0
- Hàm getAddress lấy địa chỉ con trỏ đặt vào $v0
- add $v0, $0, $a0: Sao chép giá trị của thanh ghi $a0 vào thanh ghi $v0, nơi lưu trữ địa chỉ của con trỏ

2.3.6 Thực hiện copy 2 con trỏ xâu kí tự

```
    case_2:
    bne $v0, 2, case_3
    li  $v0, 4
    la  $a0, mess2.1
    syscall
    li  $v0, 5 #Nhap vao so ky tu toi da cua chu?i
    syscall
    move    $a1, $v0 #Luu so ky tu toi da vao a1
    li  $a2, 1 #Dat a2 = 1
    la  $a0, CharPtr1 #Dat dia chi cua chuoi 1 vao a0 va goi malloc
    jal malloc
    move    $s0, $v0 #Dat s0 lam bien con tro cua chuoi 1
    la  $a0, CharPtr2#Dat dia chi cua chuoi 2 vao a0 va goi malloc
    jal malloc
    move    $s1, $v0 #Dat s0 lam bien con tro cua chuoi 2
    li  $v0, 4
    la  $a0, mess2.2
    syscall
    move    $a0, $s0 #Nhap vao chuoi thu nhat
    li  $v0, 8
    syscall
    move    $a1, $s1 #Dat a1 la bien con tro cua chu?i 2.
    jal strcpy
    li  $v0, 4  #In ra hai chuoi
    la  $a0, mess2.3
    syscall
    move    $a0, $s1
    syscall
    j   menu
```

- Nhập số kí tự tối đa của chuỗi, lưu trong $a1
- Đặt $a2 = 1, cho biết rằng mỗi kí tự trong chuỗi được xem là có kích thước 1 byte
- Đặt $a0 là biến con trỏ chuỗi 1
- Gọi hàm malloc để cấp phát bộ nhớ động cho chuỗi 1, và kết quả sẽ được trả về trong thanh ghi $v0

- Tương tự, cấp phát bộ nhớ cho chuỗi 2. $a1 là biến con trỏ của chuỗi 2
- Hàm strcpy sao chép chuỗi 1 vào chuỗi 2

```
strcpy:
    add $t0, $0, $a0    # Khoi tao $t0 o dau xau ky tu nguon
    add $t1, $0, $a1    # Khoi tao $t1 o dau xau ky tu dich
    addi    $t2, $0, 1  # Khoi tao $t2 la ky tu khac '\0' de chay vong
        lap
cpyLoop:
    beq $t2, 0, cpyLoopEnd  # Neu ky tu duoc copy trong vong lap truoc
        la '\0', dung vong lap
    lb  $t2, 0($t0)     # Doc ky tu o xau ky tu nguon
    sb  $t2, 0($t1)     # Luu ky tu vua doc vao xau ky tu dich
    addi    $t0, $t0, 1     # Chuyen $t0 tro sang vi tri cua phan tu
        tiep theo trong xau ky tu nguon
    addi    $t1, $t1, 1     # Chuyen $t1 tro sang vi tri cua phan tu
        tiep theo trong xau ky tu dich
    j   cpyLoop
cpyLoopEnd:
    jr  $ra
```

- Thanh ghi $t0 lưu địa chỉ của chuỗi nguồn($a0), $t1 lưu địa chỉ của chuỗi đích($a1)
- Hàm cpyLoop:
  - beq $t2, 0, cpyLoopEnd: Kiểm tra nếu ký tự hiện tại là ký tự kết thúc chuỗi ('\0'), thì dừng vòng lặp.
  - lb $t2, 0($t0): Đọc một byte từ vị trí hiện tại của chuỗi nguồn và lưu vào $t2.
  - sb $t2, 0($t1): Lưu byte vừa đọc được vào vị trí hiện tại của chuỗi đích.
  - addi $t0, $t0, 1: Di chuyển con trỏ của chuỗi nguồn sang phần tử kế tiếp.
  - addi $t1, $t1, 1: Di chuyển con trỏ của chuỗi đích sang phần tử kế tiếp.
  - j cpyLoop: Quay lại đầu vòng lặp để tiếp tục sao chép.

2.3.7 Hàm malloc2

```
case_3:
    bne $v0, 3, case_4
    li  $v0, 4
    la  $a0, mess3.1
    syscall
    li  $v0, 5 #Nhap vao so hang
    syscall
    move    $a1, $v0
    li  $v0, 4
    la  $a0, mess3.2
    syscall
    li  $v0, 5 #Nhap vao so cot
    syscall
    move    $a2, $v0
    la  $a0, Array2Ptr #Luu vao a0 dia chi cua mang 2 chieu
    jal malloc2
    move    $t0, $v0 #Gan t0 bien con tro
    li  $v0, 4
    la  $a0, mess0.3
    syscall
    move    $a0, $t0 #Gan a0 thanh bien con tro
    add $t0, $0, $0#Khoi t o t0 = 0
    move    $t1, $a1 #t1 la so hang
    mul $a1, $a1, $a2 #a1 la so phan tu
input_loop2:
```

```
26          beq $t0, $a1, input_end2 #su dung chuoi de nhap vao day, ket thuc
                khi t0 == so phan tu
27          li  $v0, 5
28          syscall
29          sw  $v0, 0($a0)
30          addi    $a0, $a0, 4
31          addi    $t0, $t0, 1
32          j   input_loop2
33      input_end2:
34          move    $a1, $t1 #tra lai so hang ve a1
```

- Lưu số hàng ở $a1, số cột ở $a2, chuẩn bị cho việc khởi tạo mảng 2 chiều bằng hàm malloc2
- Dùng vòng lặp để nhập phần tử có số lượgn bằng số hàng nhân số cột, lưu ở $a1

```
1       malloc2:
2           addi    $sp, $sp, -12   # Luu cac gia tri can thiet de thuc hien 1
                chuong trinh con malloc trong chuong trinh con nay
3           sw  $ra, 8($sp)
4           sw  $a1, 4($sp)
5           sw  $a2, 0($sp)
6           mul $a1, $a1, $a2   # $a1 = so phan tu = so hang * so cot
7           addi    $a2, $0, 4  # $a2 = so byte cua 1 phan tu kieu word = 4
8           jal malloc      # Chuyen mang 2 chieu thanh mang 1 chieu, khoi tao
9           lw  $ra, 8($sp) # Tra lai gia tri cho cac thanh ghi
10          lw  $a1, 4($sp)
11          lw  $a2, 0($sp)
12          addi    $sp, $sp, 12
13          jr  $ra
```

- Hàm malloc2 xử lí logic cua mảng 2 chiều với việc coi như nó là mảng 1 chiều. Cách làm:
  - Tạm thời lưu 1 số thanh ghi cần thiết trên stack
  - Tính số lượng phần tử và byte cần thiết cho mảng. Coi kích thước mỗi phần tử là 4(word)
  - Gọi hàm malloc để cấp phát bộ nhớ
  - Khôi phục thanh ghi đã lưu và trả lại địa chỉ bộ nhớ được cấp phát

2.3.8 Hàm getArray[i][j]

```
1       getArray:
2       mul $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
3       add $t0, $t0, $s1
4       sll $t0, $t0, 2 # Do phan tu co kieu word nen phai * 4 de ra khoang
            cach dia chi tuong doi so voi dia chi dau
5       add $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
6       lw  $v0, 0($t0) # Lay gia tri phan tu
7       jr  $ra
```

- Phần tử thứ A[i][j] của mảng 2 chiều là phần tử thứ A[i*số cột + j] của mảng 1 chiều
- Mỗi phần tử cách nhau 4 byte → cần nhân 4 đêt tính khoảng cách từ vị trí đầu

2.3.9 Hàm setArray[i][j]

```
setArray:
    mul $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
    add $t0, $t0, $s1
    sll $t0, $t0, 2 # Do phan tu co kieu word nen phai * 4 de ra
        khoang cach dia chi tuong doi so voi dia chi dau
    add $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
    sw  $v0, 0($t0) # Dat gia tri phan tu
    jr  $ra
```

- Tương tự getArray[i][j]
- Thay vì in ra, ta lưu giá trị mới nhập bằng lệnh sw $v0, 0($t0)

2.3.10 Tính toàn bộ lượng bộ nhớ đã cấp phát

```
bne $v0, 5, case_6
    li  $v0, 4
    la  $a0, mess1.3
    syscall
    jal memoryCalculate
    move    $a0, $v0
    li  $v0, 1
    syscall
    j   menu

memoryCalculate:
la  $t0, Sys_MyFreeSpace    # Lay dia chi dau tien duoc cap phat
la  $t1, Sys_TheTopOfFree   # Lay dia chi luu dia chi dau tien con
    trong
lw  $t2, 0($t1)     # Lay dia chi dau tien con trong
sub $v0, $t2, $t0       # Tru hai dia chi cho nhau
jr  $ra
```

- Ta lấy địa chỉ trống ở top(Sys_TheTopOfFree) trừ đi địa chỉ trống đầu tiên(Sys_MyFreeSpace)

2.3.11 Giải phóng bộ nhớ

```
bne $v0, 4, case_5
jal free
li  $v0, 4
la  $a0, mess4.1
syscall
li  $v0, 4
la  $a0, mess1.3
syscall
jal memoryCalculate
move    $a0, $v0
li  $v0, 1
syscall
j   menu

free:
addi    $sp, $sp, -4    # Khoi tao 1 vi tri trong stack
sw  $ra, 0($sp) # Luu $ra vao stack
jal SysInitMem  # Tai lap lai vi tri cua con tro luu dia chi dau tien
    con trong
lw  $ra, 0($sp) # Tra gia tri cho $ra
addi    $sp, $sp, 4 # Xoa stack
```

- Lưu $ra vào stack

- Gọi hàm SysInitMem, tài khởi tạo hệ thống quản lí bộ nhớ
- Khôi phục $ra từ stack
- Thay đổi sp về vị trí ban đầu

2.4 Kết quả

2.4.1 Mảng 1 chiều

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 1
So phan tu: 4
So byte moi phan tu (1 hoac 4): 4
Nhap phan tu:
1
2
3
4
Gia tri cua con tro: 0x90000004
Dia chi cua con tro: 0x10010008
```

2.4.2 Copy xâu

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 2
So ky tu toi da: 4

Nhap chuoi ky tu: lbv
Chuoi ky tu duoc copy: lbv
```

2.4.3 Mảng 2 chiều

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 3

So hang: 3

So cot: 3
Nhap phan tu:
1
2
3
4
5
6
7
8
9
```

```
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 1
i = 1
j = 2

Gia tri cua phan tu: 6
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 2
i = 1
j = 2
Nhap phan tu:
12
```

```
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 1
i = 1
j = 2

Gia tri cua phan tu: 12
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 3
```

2.4.4 Hiển thị bộ nhớ

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 5

Tong bo nho da cap phat: 60
```

2.4.5 Giải phóng bộ nhớ

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 4
Da giai phong toan bo bo nho cap phat.

Tong bo nho da cap phat: 0
```

→ Chương trình chạy đúng

2.5 SourceCode

```
1       .data
2       CharPtr1:    .word   0
3       CharPtr2:    .word   0
4       ArrayPtr:    .word   0
5       Array2Ptr:   .word   0
6       mess1:   .asciiz "\n\n1. Mang mot chieu\n"
7       mess2:   .asciiz "2. Sao chep mang ky tu\n"
8       mess3:   .asciiz "3. Mang hai chieu\n"
9       mess4:   .asciiz "4. Giai phong bo nho\n"
10      mess5:   .asciiz "5. Hien thi bo nho\n"
11      mess6:   .asciiz "6. Ket thuc chuong trinh\n"
12      mess0.1:     .asciiz "So phan tu: "
13      mess0.2:     .asciiz "So byte moi phan tu (1 hoac 4): "
14      mess0.3:     .asciiz "Nhap phan tu: \n"
15      mess1.1:     .asciiz "Gia tri cua con tro: "
16      mess1.2:     .asciiz "\nDia chi cua con tro: "
17      mess1.3:     .asciiz "\nTong bo nho da cap phat: "
18      mess2.1:     .asciiz "So ky tu toi da: "
19      mess2.2:     .asciiz "\nNhap chuoi ky tu: "
20      mess2.3:     .asciiz "\nChuoi ky tu duoc copy: "
21      mess3.1:     .asciiz "\nSo hang: "
22      mess3.2:     .asciiz "\nSo cot: "
23      mess3.3:     .asciiz "\n1. getArray[i][j]\n"
24      mess3.4:     .asciiz "2. setArray[i][j]\n"
25      mess3.5:     .asciiz "3. Thoat\n"
26      mess3.6:     .asciiz "\nGia tri cua phan tu: "
27      mess3.01:    .asciiz "i = "
28      mess3.02:    .asciiz "j = "
29      mess4.1:     .asciiz "Da giai phong toan bo bo nho cap phat.\n"
30      select:      .asciiz "Lua chon: "
31      errmess:     .asciiz "\nSo vua nhap khong hop le.\n"
32
33      .kdata
34      # Bien chua dia chi dau tien cua vung nho con trong
35      Sys_TheTopOfFree:    .word   1
36      # Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
37      Sys_MyFreeSpace:
38
39      .text
40      #Khoi tao vung nho cap phat dong
```

```
41          jal SysInitMem
42      #Hien thi menu
43      menu:
44          li  $v0, 4
45          la  $a0, mess1
46          syscall
47          la  $a0, mess2
48          syscall
49          la  $a0, mess3
50          syscall
51          la  $a0, mess4
52          syscall
53          la  $a0, mess5
54          syscall
55          la  $a0, mess6
56          syscall
57          la  $a0, select
58          syscall
59          li  $v0, 5 #Nhap lua chon
60          syscall
61
62         # Process user choice
63      beq $v0, 1, case_1
64      beq $v0, 2, case_2
65      beq $v0, 3, case_3
66      beq $v0, 4, case_4
67      beq $v0, 5, case_5
68      beq $v0, 6, case_6
69      j   error   # Handle invalid choice
70
71      #1. Sua loi bo nho cua vi du
72      case_1:
73          bne $v0, 1, case_2
74          li  $v0, 4
75          la  $a0, mess0.1
76          syscall
77          li  $v0, 5 #Nhap so phan tu cua day 1 chieu
78          syscall
79          bltz    $v0, error #Kiem tra so da nhap vao, neu v0 < 0 thi bao
                loi va yeu cau nhap lai
80          move    $a1, $v0 #Luu so phan tu vao a1
81          li  $v0, 4
82          la  $a0, mess0.2
83          syscall
84          li  $v0, 5 #Nhap kich thuc moi phan tu cua day
85          syscall
86      is1:    beq $v0, 1, ready
87      is4:    beq $v0, 4, ready
88          j   error   #Kiem tra kich thuoc nhap vao, neu kh ng phai 1 hay 4
                thi bao loi va yeu cau nhap lai
89      ready:  move    $a2, $v0 #Luu kich thuoc moi phan tu vao a2
90          la  $a0, ArrayPtr #Luu dia chi bat dau cua chuoi mot chieu
91          jal malloc #Chay ham malloc cho chuoi 1 chieu
92          move    $t0, $v0 #Dat dia chi bat dau tra tu malloc vao t0
93          li  $v0, 4
94          la  $a0, mess0.3
95          syscall
96          move    $a0, $t0 #Dat a0 = t0
97          li  $t0, 0 #Dat t0 = 0
98      input_loop:
99          beq $t0, $a1, input_end #Bat dau vong lap nhap du lieu, ket thuc
                khi t0 = a1
100         li  $v0, 5
101         syscall
102         bne $a2, 1, byte_4
103     byte_1:
```

```
104         sb  $v0, 0($a0)
105         addi    $a0, $a0, 1 #Neu kich thuoc phan tu bang 1 thi con tro
                tien 1 don vi
106         addi    $t0, $t0, 1
107         j   input_loop
108     byte_4:
109         sw  $v0, 0($a0)
110         addi    $a0, $a0, 4 #Neu kich thuoc phan tu bang 4 thi con tro
                tien 4 don vi
111         addi    $t0, $t0, 1
112         j   input_loop
113     input_end:
114     #2. Gia tri cua con tro
115         li  $v0, 4
116         la  $a0, mess1.1
117         syscall
118         la  $a0, ArrayPtr
119         jal getValue
120         move    $a0, $v0
121         li  $v0, 34
122         syscall
123     #3. Dia chi cua con tro
124         li  $v0, 4
125         la  $a0, mess1.2
126         syscall
127         la  $a0, ArrayPtr
128         jal getAddress
129         move    $a0, $v0
130         li  $v0, 34
131         syscall
132
133         j   menu
134     #4. Vi?t h m th?c hi?n copy 2 con tr? x u k  t?.
135     case_2:
136         bne $v0, 2, case_3
137         li  $v0, 4
138         la  $a0, mess2.1
139         syscall
140         li  $v0, 5 #Nhap vao so ky tu toi da cua chu?i
141         syscall
142         move    $a1, $v0 #Luu so ky tu toi da vao a1
143         li  $a2, 1 #Dat a2 = 1
144         la  $a0, CharPtr1 #Dat dia chi cua chuoi 1 vao a0 va goi malloc
145         jal malloc
146         move    $s0, $v0 #Dat s0 lam bien con tro cua chuoi 1
147         la  $a0, CharPtr2#Dat dia chi cua chuoi 2 vao a0 va goi malloc
148         jal malloc
149         move    $s1, $v0 #Dat s0 lam bien con tro cua chuoi 2
150         li  $v0, 4
151         la  $a0, mess2.2
152         syscall
153         move    $a0, $s0 #Nhap vao chuoi thu nhat
154         li  $v0, 8
155         syscall
156         move    $a1, $s1 #Dat a1 la bien con tro cua chu?i 2.
157         jal strcpy
158         li  $v0, 4  #In ra hai chuoi
159         la  $a0, mess2.3
160         syscall
161         move    $a0, $s1
162         syscall
163         j   menu
164     #7. Viet ham malloc 2:
165     case_3:
166         bne $v0, 3, case_4
167         li  $v0, 4
```

```
168         la  $a0, mess3.1
169         syscall
170         li  $v0, 5 #Nhap vao so hang
171         syscall
172         move    $a1, $v0
173         li  $v0, 4
174         la  $a0, mess3.2
175         syscall
176         li  $v0, 5 #Nhap vao so cot
177         syscall
178         move    $a2, $v0
179         la  $a0, Array2Ptr #Luu vao a0 dia chi cua mang 2 chieu
180         jal malloc2
181         move    $t0, $v0 #Gan t0 bien con tro
182         li  $v0, 4
183         la  $a0, mess0.3
184         syscall
185         move    $a0, $t0 #Gan a0 thanh bien con tro
186         add $t0, $0, $0#Khoi t o t0 = 0
187         move    $t1, $a1 #t1 la so hang
188         mul $a1, $a1, $a2 #a1 la so phan tu
189     input_loop2:
190         beq $t0, $a1, input_end2 #su dung chuoi de nhap vao day, ket thuc
                khi t0 == so phan tu
191         li  $v0, 5
192         syscall
193         sw  $v0, 0($a0)
194         addi    $a0, $a0, 4
195         addi    $t0, $t0, 1
196         j   input_loop2
197     input_end2:
198         move    $a1, $t1 #tra lai so hang ve a1
199     #8. Ham getArray va setArray
200     sub_menu:
201         li  $v0, 4
202         la  $a0, mess3.3
203         syscall
204         la  $a0, mess3.4
205         syscall
206         la  $a0, mess3.5
207         syscall
208         la  $a0, select
209         syscall
210         li  $v0, 5
211         syscall
212     sub_case_1:
213         bne $v0, 1, sub_case_2
214         li  $v0, 4
215         la  $a0, mess3.01
216         syscall
217         li  $v0, 5 #Nhap so hang
218         syscall
219         move    $s0, $v0 #Luu vao s0
220         li  $v0, 4
221         la  $a0, mess3.02
222         syscall
223         li  $v0, 5 #Nhap so cot
224         syscall
225         move    $s1, $v0 #Luu vao s1
226         la  $t0, Array2Ptr
227         lw  $a0, 0($t0)
228         jal getArray
229         move    $s2, $v0
230         li  $v0, 4
231         la  $a0, mess3.6
232         syscall
```

```
233         li  $v0, 1
234         move    $a0, $s2
235         syscall
236         j   sub_menu
237     sub_case_2:
238         bne $v0, 2, sub_case_3
239         li  $v0, 4
240         la  $a0, mess3.01
241         syscall
242         li  $v0, 5
243         syscall
244         move    $s0, $v0
245         li  $v0, 4
246         la  $a0, mess3.02
247         syscall
248         li  $v0, 5
249         syscall
250         move    $s1, $v0
251         move    $s2, $v0
252         li  $v0, 4
253         la  $a0, mess0.3
254         syscall
255         li  $v0, 5
256         syscall
257         la  $t0, Array2Ptr
258         lw  $a0, 0($t0)
259         jal setArray
260         j   sub_menu
261     sub_case_3:
262         bne $v0, 3, error
263         j   menu
264     #5. Gi?i ph ng b? nh?
265     case_4:
266         bne $v0, 4, case_5
267         jal free
268         li  $v0, 4
269         la  $a0, mess4.1
270         syscall
271         li  $v0, 4
272         la  $a0, mess1.3
273         syscall
274         jal memoryCalculate
275         move    $a0, $v0
276         li  $v0, 1
277         syscall
278         j   menu
279     #6. Vi?t h m t nh to n b? l??ng b? nh? ?  c?p ph t.
280     case_5:
281
282         bne $v0, 5, case_6
283         li  $v0, 4
284         la  $a0, mess1.3
285         syscall
286         jal memoryCalculate
287         move    $a0, $v0
288         li  $v0, 1
289         syscall
290         j   menu
291     case_6:
292
293         bne $v0, 6, error
294         li $v0, 10
295         syscall
296     error:
297         li  $v0, 4
298         la  $a0, errmess
```

```
299         syscall
300         j   menu
301     #-----------------------------------------
302
303     SysInitMem:
304         la  $t9, Sys_TheTopOfFree   # Lay con tro chua dau tien con trong,
                khoi tao
305         la  $t7, Sys_MyFreeSpace    # Lay dia chi dau tien con trong, khoi
                tao
306         sw  $t7, 0($t9)     # Luu lai
307         jr  $ra
308     #-----------------------------------------
309
310     malloc:
311         la  $t9, Sys_TheTopOfFree
312         lw  $t8, 0($t9)     # Lay dia chi dau tien con trong
313         bne $a2, 4, initialize  # Neu mang khoi tao co kieu Word, kiem tra
                dia chi dau co dam bao quy tac khong
314         andi    $t0, $t8, 0x03      # Lay so du khi chia dia chi trong cho
                4
315         beq $t0, 0, initialize  # Neu khong co du, bo qua
316         addi    $t8, $t8, 4     # Neu co, tien toi dia chi chia het cho 4
                tiep theo
317         subu    $t8, $t8, $t0
318     initialize:
319         sw  $t8, 0($a0) # Cat dia chi do vao bien con tro
320         move    $v0, $t8    # Dong thoi la ket qua tra ve cua ham
321         mul $t7, $a1,$a2    # Tinh kich thuoc cua mang can cap phat
322         add $t6, $t8, $t7   # Tinh dia chi dau tien con trong
323         sw  $t6, 0($t9) # Luu tro lai dia chi dau tien do vao bien
                Sys_TheTopOfFree
324         jr  $ra
325     #-----------------------------------------
326
327     getValue:
328         lw  $v0, 0($a0) # Lay gia tri cua bien con tro trong o nho co dia
                chi luu trong $a0
329         jr  $ra
330     #-----------------------------------------
331
332     getAddress:
333         add $v0, $0, $a0    # Lay dia chi tu $a0
334         jr  $ra
335     #-----------------------------------------
336
337     strcpy:
338         add $t0, $0, $a0    # Khoi tao $t0 o dau xau ky tu nguon
339         add $t1, $0, $a1    # Khoi tao $t1 o dau xau ky tu dich
340         addi    $t2, $0, 1  # Khoi tao $t2 la ky tu khac '\0' de chay vong
                lap
341     cpyLoop:
342         beq $t2, 0, cpyLoopEnd  # Neu ky tu duoc copy trong vong lap truoc
                la '\0', dung vong lap
343         lb  $t2, 0($t0)     # Doc ky tu o xau ky tu nguon
344         sb  $t2, 0($t1)     # Luu ky tu vua doc vao xau ky tu dich
345         addi    $t0, $t0, 1     # Chuyen $t0 tro sang vi tri cua phan tu
                tiep theo trong xau ky tu nguon
346         addi    $t1, $t1, 1     # Chuyen $t1 tro sang vi tri cua phan tu
                tiep theo trong xau ky tu dich
347         j   cpyLoop
348     cpyLoopEnd:
349         jr  $ra
350     #-----------------------------------------
351
352     free:
353         addi    $sp, $sp, -4    # Khoi tao 1 vi tri trong stack
```

```
354         sw  $ra, 0($sp) # Luu $ra vao stack
355         jal SysInitMem  # Tai lap lai vi tri cua con tro luu dia chi dau
                tien con trong
356         lw  $ra, 0($sp) # Tra gia tri cho $ra
357         addi    $sp, $sp, 4 # Xoa stack
358     #-----------------------------------------
359
360     memoryCalculate:
361         la  $t0, Sys_MyFreeSpace    # Lay dia chi dau tien duoc cap phat
362         la  $t1, Sys_TheTopOfFree   # Lay dia chi luu dia chi dau tien con
                trong
363         lw  $t2, 0($t1)     # Lay dia chi dau tien con trong
364         sub $v0, $t2, $t0       # Tru hai dia chi cho nhau
365         jr  $ra
366     #-----------------------------------------
367
368     malloc2:
369         addi    $sp, $sp, -12   # Luu cac gia tri can thiet de thuc hien 1
                chuong trinh con malloc trong chuong trinh con nay
370         sw  $ra, 8($sp)
371         sw  $a1, 4($sp)
372         sw  $a2, 0($sp)
373         mul $a1, $a1, $a2   # $a1 = so phan tu = so hang * so cot
374         addi    $a2, $0, 4  # $a2 = so byte cua 1 phan tu kieu word = 4
375         jal malloc      # Chuyen mang 2 chieu thanh mang 1 chieu, khoi tao
376         lw  $ra, 8($sp) # Tra lai gia tri cho cac thanh ghi
377         lw  $a1, 4($sp)
378         lw  $a2, 0($sp)
379         addi    $sp, $sp, 12
380         jr  $ra
381     #-----------------------------------------
382
383     getArray:
384         mul $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
385         add $t0, $t0, $s1
386         sll $t0, $t0, 2 # Do phan tu co kieu word nen phai * 4 de ra
                khoang cach dia chi tuong doi so voi dia chi dau
387         add $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
388         lw  $v0, 0($t0) # Lay gia tri phan tu
389         jr  $ra
390     #-----------------------------------------
391
392     setArray:
393         mul $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
394         add $t0, $t0, $s1
395         sll $t0, $t0, 2 # Do phan tu co kieu word nen phai * 4 de ra
                khoang cach dia chi tuong doi so voi dia chi dau
396         add $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
397         sw  $v0, 0($t0) # Dat gia tri phan tu
398         jr  $ra
```

3 CHƯƠNG TRÌNH KIỂM TRA CÚ PHÁP LỆNH MIPS

3.1 Đề bài

7. Chương trình kiểm tra cú pháp lệnh MIPS

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.*

Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, toán hạng 2, toán hạng 3.

3.2 Định hướng cách làm

3.2.1 Ý tưởng

- Xây dựng 1 Library có chứa cấu trúc của các lệnh hợp ngữ (Cấu trúc của Library gồm: opcode - toán hạng).
- Các ký hiệu trong Library:
  - 1 - thanh ghi
  - 2 - hằng số nguyên
  - 3 - định danh
  - 4 - dành cho các lệnh lw, sw,... có cấu trúc ví dụ như imm($ rs)
  - 0 - không có
- Tạo các khu vực chứa "character", "number", "register" để thực hiện kiểm tra.
- Duyệt câu lệnh nhập vào từ trái sang phải. Nếu opcode đúng, duyệt tiếp tới các toán hạng. Nếu opcode sai thì thông báo sai (tương tự với các toán hạng cũng duyệt dần như vậy).

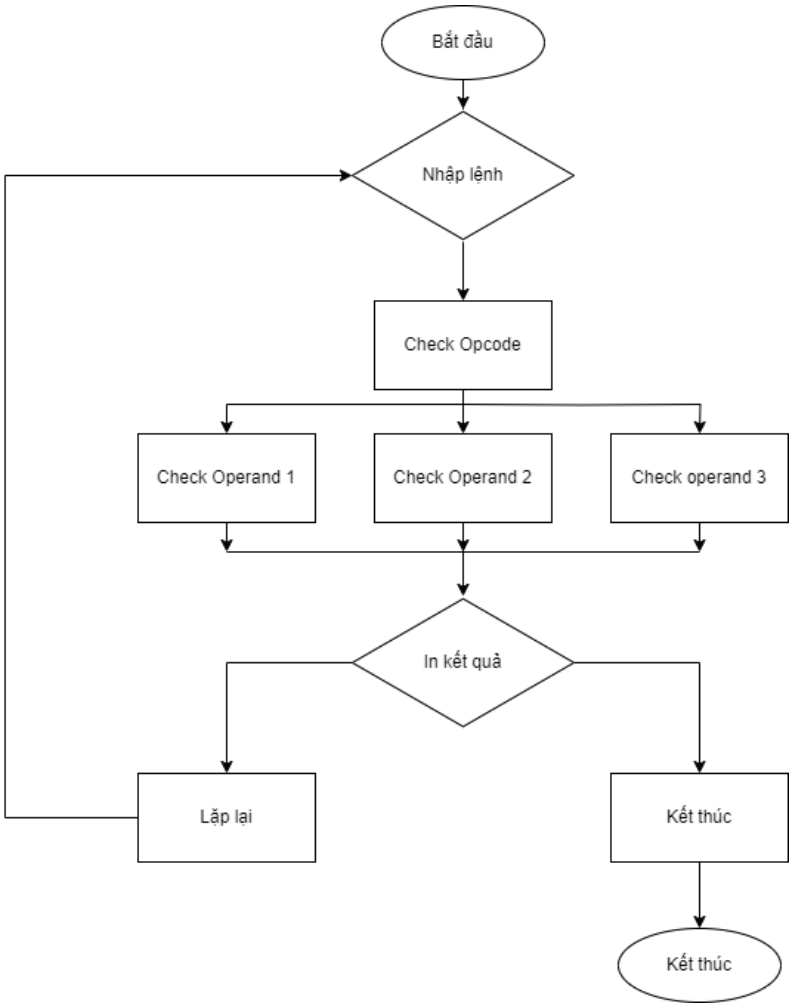
3.2.2 Thuật toán

Bước 1: Menu thực hiện nhập lệnh hoặc thoát - Nhập lệnh MIPS từ bàn phím. **Bước 2:** Kiểm tra câu lệnh:

1. Kiểm tra opcode (add, and, or,...)
2. Sau khi kiểm tra, opcode kiểm tra dần tới các toán hạng. Nếu toán hạng đó là '0' trong Library thì thực hiện kiểm tra đằng sau đó có thừa ký tự gì không.
3. Kiểm tra giữa 2 toán hạng cần phải có dấu ','.
4. Khi kiểm tra các toán hạng cần xem trong Library toán hạng đó là thanh ghi, số nguyên, định danh hay imm($rs). Rồi thực hiện đi đến so sánh từng giá trị này.

Bước 3: Kiểm duyệt thành công in thông báo lệnh đúng cấu trúc và ngược lại, sau đó quay trở lại Menu.

3.2.3 Lưu đồ thuật toán



Hình 1: Lưu đồ thuật toán

3.3 Thuật toán và hàm

3.3.1 Khai báo dữ liệu

```
1     .data
2     menu_mess:        .asciiz "\n----- MENU ------\n1. Kiem tra cu phap
          lenh\n2. Thoat \nChon: "
3     menu_error_mess: .asciiz "\nNhap sai, vui long nhap lai!\n"
4     input_mess:       .asciiz "\nNhap vao lenh Mips: "
5     opcode_mess:     .asciiz "Opcode: "
6     toanHang_mess:   .asciiz "Toan hang: "
7     hopLe_mess:       .asciiz " - hop le.\n"
8     error_mess:       .asciiz "\nLenh hop ngu khong hop le, sai khuon dang
          lenh !\n"
9     completed_mess: .asciiz "\nLenh hop ngu chinh xac !\n"
10    command:     .space 100   # Luu cau lenh
11    opcode:      .space 30    # Luu ma lenh, vi du: add, and,...
12    ident:       .space 30    # nhan | hoac number
13    token:       .space 30    # cac thanh ghi, vi du: $zero, $at,...
14
15    # Cau truc cua library:
16    # opcode (7) - operation (3)
17    # Trong so luong operation: 1 - thanh ghi; 2 - hang so nguyen; 3 -
          dinh danh (ident); 4 - imm($rs); 0 - khong co
18    library:     .asciiz
```

```
"add****111;sub****111;addi***112;addu***111;addiu**112;subu***111;mfc0
    ***110;mult***110;multu**110;div****110;mfhi***100;mflo***100;and****11
    1;or*****111;andi***112;ori****112;sll****112;srl****112;lw*****140;sw*
    ****140;lbu****140;sb*****140;lui****120;beq****113;bne****113;slt****1
    11;slti***112;sltiu**112;j******300;jal****300;jr*****100;nop****000"
    numberGroup:    .asciiz "0123456789-"
    characterGroup: .asciiz
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_"
    # Moi thanh ghi cach nhau 6 byte
    tokenRegisters: .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    $6    $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   "
```

Các dữ liệu bao gồm các string thông báo tới người dùng và các thư viện. Trong đó :

- Library: Với cấu trúc là opcode(7) - operation (3) với trọng số trong operation :
  - 1 - Thanh ghi
  - 2 - Hằng số nguyên
  - 3 - Định danh (ident)
  - 4 - imm($rs)
  - 0 - Không có
- numberGroup: lưu các số
- characterGroup: lưu toàn bộ chữ cái
- tokenRegisters: lưu toàn bộ thanh ghi (mỗi thanh ghi cách nhau 6 byte)

3.3.2 Chương trình menu

```
# ----- MENU -----
m_menu_start:
    li $v0, 4
    la $a0, menu_mess
    syscall

    # Read number input menu
    li $v0, 5
    syscall

    beq $v0, 2, end_main        # 2: ket thuc
    beq $v0, 1, m_menu_end      # 1: thuc hien kiem tra

    li $v0, 4
    la $a0, menu_error_mess     # Nhap sai
    syscall

    j m_menu_start
m_menu_end:

# ----- READ INPUT -----
m_input:
    jal  input
    nop

# ----- START CHECK -----

m_check:
```

```
    jal check
    nop

    j m_menu_start

end_main:
    li $v0, 10
    syscall

input:
    li $v0, 4
    la $a0, input_mess
    syscall

    li $v0, 8
    la $a0, command
    li $a1, 100
    syscall

    jr $ra
```

- Menu có 2 lựa chọn:
    1. Tiến hành kiểm tra lệnh
    2. Kết thúc
- Khi chọn 1 thì chương trình sẽ nhảy đến hàm input và bắt đầu nhập lệnh
- Sau khi lưu vào mảng **command** chương trình sẽ gọi chương trình con check

3.3.3 Chương trình check

```
check:
    # Luu $ra de tro ve main
    addi $sp, $sp, -4
    sw   $ra, 0($sp)

    addi $s7, $zero, 0      # Thanh ghi $s7 luu index cua command

    # START CHECK OPCODE
    jal check_opcode
    nop

    # START CHECK OPERAND 1
    li  $s3, 7          # Vi tri operand trong Library
    jal check_operand
    nop

    # START CHECK OPERAND 2     # Neu khong co dau ',' ngan cach giua
        operand_1 va operand_2 => FALSE
    li  $s3, 8          # Vi tri operand trong Library
    add $t0, $s5, $s3
    lb  $t0, 0($t0)
    beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky
        tu 0 trong ASCII

    la   $a0, command
    add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
    lb   $t1, 0($t0)
    bne  $t1, 44, not_found     # Dau ','
    add  $s7, $s7, 1

    jal check_operand
    nop
```

```
31
32      # START CHECK OPERAND 3      # Neu khong co dau ',' ngan cach giua
            operand_1 va operand_2 => FALSE
33      li  $s3, 9          # Vi tri operand trong Library
34      add $t0, $s5, $s3
35      lb  $t0, 0($t0)
36      beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky
            tu 0 trong ASCII
37
38      la   $a0, command
39      add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
40      lb   $t1, 0($t0)
41      bne  $t1, 44, not_found     # Dau ','
42      add  $s7, $s7, 1
43
44      jal check_operand
45      nop
46
47      # KIEM TRA KY TU THUA
48      j check_none
49
50      # Tra lai $ra de tro ve main
51      lw   $ra, 0($sp)
52      addi $sp, $sp, 4
53      jr   $ra
```

- Đầu tiên chương trình lưu lại địa chỉ để trả về menu, sau đó lưu index của **command**
- Kiểm tra opcode (add, and, or,...) tên lenh
- Kiểm tra lần lượt các operand (Toan hang)
- Kiểm tra ở cuối xem có ký tự thừa ko
- Giải thích các thanh ghi sử dụng:
  - $s7: Lưu index command
  - $s3: Vị trí của từng toán hạng trong Library
  - $s5: Vị trí của của câu lệnh đang xét trong library (Vị trí sẽ được tìm sau khi check xong opcode)

3.3.4 Tách Opcode và kiểm tra loại opcode

```
1   check_opcode:
2       la  $a0, command                # Dia chi cua command
3       la  $a1, opcode                 # Dia chi cua opcode
4       li  $t0, 0
5
6   remove_space_command:                   # Xoa cac dau cach phia truoc lenh
7       add $t1, $a0, $t0
8       lb  $t2, 0($t1)
9       bne $t2, 32, end_remove_space_command       # Neu khong phai ' ' ->
            Ket thuc
10      addi $t0, $t0, 1
11      j remove_space_command
12  end_remove_space_command:
13
14      li  $t9, 0                  # index for opcode
15      li  $s6, 0                  # so luong cac ki tu cua opcode = 0
16  read_opcode:
17      add $t1, $a0, $t0               # Dich bit cua command
18      add $t2, $a1, $t9               # Dich bit cua opcode
19      lb  $t3, 0($t1)
```

```
20
21      beq $t3, 32, read_opcode_done           # Neu co dau cach ' ' ket thuc
            read opcode
22      beq $t3, 10, read_opcode_done           # Neu dau '\n' ket thuc read
            opcode
23      beq $t3, 0,  read_opcode_done           # Ket thuc chuoi
24
25      sb  $t3, 0($t2)
26      addi $t9, $t9, 1
27      addi $t0, $t0, 1
28      j read_opcode
29  read_opcode_done:
30
31      addi $s6, $t9, 0                # $s6: So luong ki tu cua opcode
32      add $s7, $s7, $t0               # luu index cua command
33      la $a2, library
34      li $t0, -11
35
36  check_opcode_inlib:
37      addi $t0, $t0, 11               # Buoc nhay bang 10 de nhay den tung
            Instruction
38      li $t1, 0                   # i = 0
39      li $t2, 0                   # j = 0
40      add $t1, $t1, $t0               # Cong buoc nhay
41
42      compare_opcode:
43          add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua
                tung Instruction
44          lb  $t4, 0($t3)
45          beq $t4, 0, not_found
46          beq $t4, 42, check_len_opcode       # Neu gap ky tu '*' => Kiem
                tra do dai
47          add $t5, $a1, $t2           # Load opcode
48          lb  $t6, 0($t5)
49          bne $t4, $t6, check_opcode_inlib    # So sanh 2 ki tu, neu khong
                bang nhau thi tinh den Instruction tiep theo.
50          addi $t1, $t1, 1            # i = i + 1
51          addi $t2, $t2, 1            # j = j + 1
52          j compare_opcode
53      check_len_opcode:
54          bne $t2, $s6, check_opcode_inlib
55  end_check_opcode_inlib:
56
57      add $s5, $t0, $a2               # Luu lai vi tri Instruction trong
            Library.
58
59      # ----- In thong tin ra man hinh -----
60      li $v0, 4
61      la $a0, opcode_mess
62      syscall
63
64      la $a3, opcode
65      li $t0, 0
66      print_opcode:
67          beq $t0, $t9, end_print_opcode
68          add $t1, $a3, $t0
69          lb  $t2, 0($t1)
70          li $v0, 11
71          add $a0, $t2, $zero
72          syscall
73          addi $t0, $t0, 1
74          j print_opcode
75      end_print_opcode:
76
77      li $v0, 4
78      la $a0, hopLe_mess
```

```
79      syscall
80
81      jr $ra
```

Các bước thực hiện:

Bước 1: Khởi tạo các biến để chuẩn bị check Opcode

Bước 2: Gọi hàm **remove_space_command**, hàm này dùng để xóa toàn bộ khoảng trắng đứng trước Opcode

Bước 3: Gọi hàm **read_opcode**, hàm này sẽ duy trì 2 con trỏ(1 con trỏ dịch bit của command, 1 con trỏ dịch bit của opcode) toàn bộ opcode sẽ được lưu vào mảng **opcode**. Index của command sẽ tiếp tục lưu và $s7.

Bước 4: Sau khi đọc xong Opcode, chương trình sẽ tiến hành tìm kiếm Opcode ở trong library và tiến hành so sánh

Bước 5: In kết quả ra màn hình là opcode có hợp lệ hay ko

Công dụng của các thanh ghi:

- $a0: địa chỉ command
- $a1: địa chỉ Opcode
- $s7: index của command
- $t9: index của Opcode

3.3.5 Nhãn check_operand

```
1   check_operand:
2       # Luu $ra de tro ve check
3       addi $sp, $sp, -4
4       sw   $ra, 0($sp)
5
6       add $t9, $s5, $s3           # Tro toi operand trong Library
7       lb  $t9, 0($t9)
8       addi $t9, $t9, -48          # Char -> Number
9
10      la  $a0, command
11      add $t0, $a0, $s7
12
13      li $t1, 0                   # i = 0
14      space_remove:               # Xoa cac khoang trang thua
15          add $t2, $t0, $t1
16          lb  $t2, 0($t2)         # Lay ky tu tiep theo
17          bne $t2, 32, end_space_remove   # Ky tu ' '
18          addi $t1, $t1, 1        # i = i + 1
19          j space_remove
20      end_space_remove:
21
22      add $s7, $s7, $t1           # Cap nhat lai index command
23
24      li $s2, 0               # Tat kich hoat check number_register
25      li $t8, 0               # Khong co
26      beq $t8, $t9, check_none
27      li $t8, 1               # Thanh ghi
28      beq $t8, $t9, go_register
29      li $t8, 2               # So hang nguyen
30      beq $t8, $t9, go_number
31      li $t8, 3               # Ident
32      beq $t8, $t9, go_ident
```

```
    li $t8, 4                # Check number & register
    beq $t8, $t9, go_number_register

end_check_operand:
    # Tra lai $ra de tro ve check
    lw   $ra, 0($sp)
    addi $sp, $sp, 4
    jr   $ra
```

Sau khi hoàn hiện bước tách opcode và kiểm ra loại opcode, chương trình sẽ nhảy đến nhãn **check_operand** để tiến hành kiểm tra loại toán hạng nào:

Bước 1: Chương rình sẽ lấy được loại toán hạng qua việc lấy thông tin câu lệnh trong thư viện

Bước 2: Chương trình sẽ thực hiện xóa khoảng trắng phía trước toán hạng

Bước 3: Chương trình sẽ nhảy đến chương trình kiểm tra loại toán hạng tương ứng

Công dụng của các thanh ghi:

- $a0: địa chỉ command
- $s5: vị trí của loại code đấy trong Library
- $s7: index của command
- $t9: giá trị toán hạng trong Library

3.3.6 Hàm kiểm tra thanh ghi

```
check_register:
    la $a0, command
    la $a1, token
    la $a2, tokenRegisters
    add $t0, $a0, $s7               # Tro den vi tri cac instruction

    li $t1, 0                   # i = 0
    li $t9, 0                   # index cua token

read_token_register:
    add $t2, $t0, $t1               # command
    add $t3, $a1, $t1               # token
    lb $t4, 0($t2)

    beq $t4, 41, end_read_token         # Gap ky tu ')'
    beq $t4, 44, end_read_token         # Gap ky tu ' , '
    beq $t4, 10, end_read_token         # Gap ky tu '\n'
    beq $t4, 0, end_read_token          # Ket thuc

    addi $t1, $t1, 1
    beq $t4, 32, read_token_register        # Neu gap dau ' ' thi tiep tuc

    sb $t4, 0($t3)
    addi $t9, $t9, 1
    j read_token_register

end_read_token:
    add $s7, $s7, $t1               # Cap nhat lai gia tri index

    li $t0, -6
compare_token_register:
    addi $t0, $t0, 6                # Buoc nhay bang 6 de nhay den tung
        Register

```

```
34      li $t1, 0                   # i = 0
35      li $t2, 0                   # j = 0
36
37      add $t1, $t1, $t0               # Cong buoc nhay
38
39      compare_reg:
40          add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua
                tung Register
41          lb  $t4, 0($t3)
42          beq $t4, 0, not_found
43          beq $t4, 32, check_len_reg      # Neu gap ky tu ' ' => Kiem tra do
                dai
44
45          add $t5, $a1, $t2           # Load token
46          lb  $t6, 0($t5)
47
48          bne $t4, $t6, compare_token_register    # So sanh 2 ki tu, neu
                khong bang nhau thi tinh den Register tiep theo.
49          addi $t1, $t1, 1            # i = i + 1
50          addi $t2, $t2, 1            # j = j + 1
51          j compare_reg
52
53      check_len_reg:
54          bne $t2, $t9, compare_token_register    # Neu do dai khong bang
                nhau di den register tiep theo
55
56  end_compare_token_register:
57
58      # >>>>>>>>> In thong tin ra man hinh <<<<<<<<<<
59      beq $s2, 1, on_token_number_register
60      li $v0, 4
61      la $a0, toanHang_mess
62      syscall
63
64      la $a3, token
65      li $t0, 0
66      print_token_register:
67          beq $t0, $t9, end_print_token_register
68          add $t1, $a3, $t0
69          lb  $t2, 0($t1)
70          li $v0, 11
71          add $a0, $t2, $zero
72          syscall
73          addi $t0, $t0, 1
74          j print_token_register
75      end_print_token_register:
76
77      li $v0, 4
78      la $a0, hopLe_mess
79      syscall
80      jr $ra
81
82  on_token_number_register:
83
84      la $a3, token
85      li $t0, 0
86      print_on_token_register:
87          beq $t0, $t9, end_print_on_token_register
88          add $t1, $a3, $t0
89          lb  $t2, 0($t1)
90          li $v0, 11
91          add $a0, $t2, $zero
92          syscall
93          addi $t0, $t0, 1
94          j print_on_token_register
95      end_print_on_token_register:
```

```
96
97      li $v0, 11
98      li $a0, 41
99      syscall
100     li $v0, 4
101     la $a0, hopLe_mess
102     syscall
103     jr $ra
```

Bước 1: Chương trình sẽ tiến hành đọc thanh ghi trong nhãn **read_token_register** và lưu bộ vào mảng token

Bước 2: Chương trình so sánh để tìm kiếm giá trị trong mảng token giống với giá trị trong mảng **token_register**

Bước 3: Sau khi so sánh, chương trình sẽ tiến hành in ra màn hình thông báo thanh ghi ấy có hợp lệ hay ko. Trong hàm này sẽ có 2 chương trình in ra màn hình , 1 là đối với các lệnh bình thường và 2 là đối với các lệnh đặc biệt có cả số và thanh ghi kết hợp

Công dụng của các thanh ghi:

- $a0: địa chỉ command
- $a1: địa chỉ token
- $a2: địa chỉ tokenRegisters
- $s7: index của command
- $t9: index của token
- $s2: Lưu kích hoạt check_number_register

3.3.7 Hàm kểm tra số và nhãn

```
1       check_ident:
2       la $a0, command
3       la $a1, ident
4
5       add $t0, $a0, $s7           # Tro den vi tri cac instruction
6
7       li $t1, 0               # i = 0
8       li $t9, 0               # index cua ident
9
10  read_ident:
11      add $t2, $t0, $t1           # command
12      add $t3, $a1, $t1           # ident
13      lb $t4, 0($t2)
14
15      beq $t4, 40, end_read_ident     # Gap ky tu '('
16      beq $t4, 44, end_read_ident     # Gap ky tu ' , '
17      beq $t4, 10, end_read_ident     # Gap ky tu '\n'
18      beq $t4, 0, end_read_ident      # Ket thuc
19
20      addi $t1, $t1, 1
21      beq $t4, 32, read_ident         # Neu gap dau ' ' thi tiep tuc
22
23      sb $t4, 0($t3)
24      addi $t9, $t9, 1
25      j read_ident
26
27  end_read_ident:
```

```
28      add $s7, $s7, $t1           # Cap nhat lai gia tri index
29      beq $t9, 0, not_found           # Khong co label
30
31      #li $v0, 10
32      #syscall
33
34      li $t2, 0               # index cho Ident
35  compare_ident:
36      beq  $t2, $t9, end_compare_ident    # ket thuc chuoi
37      li   $t1, 0             # index cho characterGroup
38
39      add  $t3, $a1, $t2
40      lb   $t3, 0($t3)            # Tung char trong Ident
41
42      loop_Group:             # Kiem tra tung ky tu Ident co trong Group hay
            khong
43          add $t4, $a2, $t1
44          lb $t4, 0($t4)
45          beq $t4, 0, not_found       # Khong co -> Khong tim thay
46          beq $t4, $t3, end_loop_Group
47
48          addi $t1, $t1, 1
49          j loop_Group
50
51      end_loop_Group:
52
53      addi $t2, $t2, 1
54
55      j compare_ident
56
57  end_compare_ident:
58
59      beq $s2, 1, on_number_register
60
61      # ----- In thong tin ra man hinh -----
62      li $v0, 4
63      la $a0, toanHang_mess
64      syscall
65
66      la $a3, ident
67      li $t0, 0
68      print_ident:
69          beq $t0, $t9, end_print_ident
70          add $t1, $a3, $t0
71          lb  $t2, 0($t1)
72          li $v0, 11
73          add $a0, $t2, $zero
74          syscall
75          addi $t0, $t0, 1
76          j print_ident
77      end_print_ident:
78
79      li $v0, 4
80      la $a0, hopLe_mess
81      syscall
82      jr $ra
83
84  on_number_register:
85      li $v0, 4
86      la $a0, toanHang_mess
87      syscall
88
89      la $a3, ident
90      li $t0, 0
91      print_on_ident:
92          beq $t0, $t9, end_print_on_ident
```

```
93          add $t1, $a3, $t0
94          lb  $t2, 0($t1)
95          li $v0, 11
96          add $a0, $t2, $zero
97          syscall
98          addi $t0, $t0, 1
99          j print_on_ident
100     end_print_on_ident:
101
102     li $v0, 11
103     li $a0, 40
104     syscall
105     jr $ra
```

Bước 1: Chương trình sẽ tiến hành đọc thanh ghi trong nhãn **read_ident** và lưu bộ vào mảng ident

Bước 2: Chương trình so sánh để tìm kiếm giá trị trong mảng ident giống với giá trị trong mảng **characterGroup | numberGroup**

Bước 3: Sau khi so sánh, chương trình sẽ tiến hành in ra màn hình thông báo thanh ghi ấy có hợp lệ hay ko. Trong hàm này sẽ có 2 chương trình in ra màn hình , 1 là đối với các lệnh bình thường và 2 là đối với các lệnh đặc biệt có cả số và thanh ghi kết hợp

Công dụng của các thanh ghi:

- $a0: địa chỉ command
- $a1: địa chỉ indent
- $a2: địa chỉ characterGroup | numberGroup
- $s7: index của command
- $t9: index của ident
- $s2: Lưu kích hoạt check_number_register

3.3.8 Hàm kiểm tra lệnh đặc biệt

```
1       check_number_register:
2       # Luu $ra de tro ve
3       addi $sp, $sp, -4
4       sw   $ra, 0($sp)
5
6       li $s2, 1               # Bat kich hoat number_register
7
8       # Check number
9       la $a2, numberGroup
10      jal check_ident
11      nop
12
13      la $a0, command
14      add $t0, $a0, $s7           # Tro den vi tri cac instruction
15      lb $t0, 0($t0)
16      bne $t0, 40, not_found          # Neu ki tu khong phai la dau '('
17      addi $s7, $s7, 1
18
19      # Check register
20      jal check_register
21      nop
22      la $a0, command
```

```
23      add $t0, $a0, $s7           # Tro den vi tri cac instruction
24      lb $t0, 0($t0)
25      bne $t0, 41, not_found          # Neu ki tu khong phai la dau ')'
26      addi $s7, $s7, 1
27
28      # Tra lai $ra de tro ve
29      lw   $ra, 0($sp)
30      addi $sp, $sp, 4
31      jr $ra
```

Bước 1: Chương trình sẽ gọi hàm **check_ident** để kiểm tra sô đừng trước dâu ngoặc

Bước 2: Chương trình sẽ gọi hàm **check_register** để kiểm tra thanh ghi trong dấu ngoặc

Công dụng của các thanh ghi:

- $a0: địa chỉ command
- $s7: index của command
- $s2: Nhận biết có phải lệnh đặc biệt hay ko

3.3.9 Hàm kiểm tra thừa

```
1   check_none:
2       la $a0, command
3       add $t0, $a0, $s7
4
5       lb $t1, 0($t0)
6
7       beq $t1, 10, none_ok     # Ky tu '\n'
8       beq $t1, 0, none_ok # Ket thuc chuoi
9
10      j not_found
11
12  none_ok:
13      li $v0, 4
14      la $a0, completed_mess
15      syscall
16      j m_menu_start
```

Sau khi kiểm tra xong toàn bộ các toán hạng, chương trình sẽ gọi hàm **check_none** để kiểm tra xem liệu có thừa ở cuối không. Sau đó sẽ in kết quả ra màn hình

3.4 Kết quả

```
----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: lw $s1, 0($ra)
Opcode: lw - hop le.
Toan hang: $s1 - hop le.
Toan hang: 0($ra) - hop le.

Lenh hop ngu chinh xac !

----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: addi $s1,$t1,32
Opcode: addi - hop le.
Toan hang: $s1 - hop le.
Toan hang: $t1 - hop le.
Toan hang: 32 - hop le.

Lenh hop ngu chinh xac !

----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: j main
Opcode: j - hop le.
Toan hang: main - hop le.

Lenh hop ngu chinh xac !
```

(a) Những trường hợp đúng định dạng

```
----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: addi $s1,$s1,$s2
Opcode: addi - hop le.
Toan hang: $s1 - hop le.
Toan hang: $s1 - hop le.

Lenh hop ngu khong hop le, sai khuon dang lenh !

----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: j 1234
Opcode: j - hop le.

Lenh hop ngu khong hop le, sai khuon dang lenh !

----- MENU ------
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: beq $s2, $s3,               label       0
Opcode: beq - hop le.
Toan hang: $s2 - hop le.
Toan hang: $s3 - hop le.

Lenh hop ngu khong hop le, sai khuon dang lenh !
```

(b) Những trường hợp sai

3.5 Source Code

```
.data
    menu_mess:        .asciiz "\n----- MENU ------\n1. Kiem tra cu phap
        lenh\n2. Thoat \nChon: "
    menu_error_mess: .asciiz "\nNhap sai, vui long nhap lai!\n"
    input_mess:       .asciiz "\nNhap vao lenh Mips: "
    opcode_mess:     .asciiz "Opcode: "
    toanHang_mess:   .asciiz "Toan hang: "
    hopLe_mess:       .asciiz " - hop le.\n"
    error_mess:       .asciiz "\nLenh hop ngu khong hop le, sai khuon dang
        lenh !\n"
    completed_mess: .asciiz "\nLenh hop ngu chinh xac !\n"
    command:     .space 100   # Luu cau lenh
    opcode:      .space 30    # Luu ma lenh, vi du: add, and,...
    ident:       .space 30    # nhan | hoac number
    token:       .space 30    # cac thanh ghi, vi du: $zero, $at,...

    # Cau truc cua library:
    # opcode (7) - operation (3)
    # Trong so luong operation: 1 - thanh ghi; 2 - hang so nguyen; 3 -
        dinh danh (ident); 4 - imm($rs); 0 - khong co
    library:     .asciiz
 "add****111;sub****111;addi***112;addu***111;addiu**112;subu***111;mfc0
     ***110;mult***110;multu**110;div****110;mfhi***100;mflo***100;and****11
     1;or*****111;andi***112;ori****112;sll****112;srl****112;lw*****140;sw*
     ****140;lbu****140;sb*****140;lui****120;beq****113;bne****113;slt****1
     11;slti***112;sltiu**112;j******300;jal****300;jr*****100;nop****000"
    numberGroup:     .asciiz "0123456789-"
    characterGroup: .asciiz
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_"
```

```
22      # Moi thanh ghi cach nhau 6 byte
23      tokenRegisters: .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    $6    $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   "
24
25  .text
26  main:
27  # ----- MENU -----
28  m_menu_start:
29      li $v0, 4
30      la $a0, menu_mess
31      syscall
32
33      # Read number input menu
34      li $v0, 5
35      syscall
36
37      beq $v0, 2, end_main        # 2: ket thuc
38      beq $v0, 1, m_menu_end      # 1: thuc hien kiem tra
39
40      li $v0, 4
41      la $a0, menu_error_mess     # Nhap sai
42      syscall
43
44      j m_menu_start
45  m_menu_end:
46
47  # ----- READ INPUT -----
48  m_input:
49      jal  input
50      nop
51
52  # ----- START CHECK -----
53
54  m_check:
55      jal check
56      nop
57
58      j m_menu_start
59
60  end_main:
61      li $v0, 10
62      syscall
63
64  #-------------------------------------------------------------
65  # 1. @input: Nhap vao lenh Mips tu ban phim
66  #-------------------------------------------------------------
67  input:
68      li $v0, 4
69      la $a0, input_mess
70      syscall
71
72      li $v0, 8
73      la $a0, command
74      li $a1, 100
75      syscall
76
77      jr $ra
78
79  #-------------------------------------------------------------
80  # 2. @check: Kiem tra cau lenh
81  # - Buoc 1: Kiem tra opcode (add, and, or,...) ten lenh
```

```
82  # - Buoc 2: Kiem tra Operand lan luot cac operand (Toan hang)
83  # - Giua 2 toan hang can kiem tra xem co dau ',' hay khong.
84  # $s7: Luu index cua command
85  # $s3: Vi tri cua tung toan hang trong Library
86  #-------------------------------------------------------------
87  check:
88      # Luu $ra de tro ve main
89      addi $sp, $sp, -4
90      sw   $ra, 0($sp)
91
92      addi $s7, $zero, 0      # Thanh ghi $s7 luu index cua command
93
94      # START CHECK OPCODE
95      jal check_opcode
96      nop
97
98      # START CHECK OPERAND 1
99      li  $s3, 7          # Vi tri operand trong Library
100     jal check_operand
101     nop
102
103     # START CHECK OPERAND 2     # Neu khong co dau ',' ngan cach giua
            operand_1 va operand_2 => FALSE
104     li  $s3, 8          # Vi tri operand trong Library
105     add $t0, $s5, $s3
106     lb  $t0, 0($t0)
107     beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky
            tu 0 trong ASCII
108
109     la   $a0, command
110     add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
111     lb   $t1, 0($t0)
112     bne  $t1, 44, not_found     # Dau ','
113     add  $s7, $s7, 1
114
115     jal check_operand
116     nop
117
118     # START CHECK OPERAND 3     # Neu khong co dau ',' ngan cach giua
            operand_1 va operand_2 => FALSE
119     li  $s3, 9          # Vi tri operand trong Library
120     add $t0, $s5, $s3
121     lb  $t0, 0($t0)
122     beq $t0, 48, check_none     # Kiem tra neu operand = 0 -> ket thuc; ky
            tu 0 trong ASCII
123
124     la   $a0, command
125     add  $t0, $a0, $s7      # tro toi vi tri tiep tuc cua command
126     lb   $t1, 0($t0)
127     bne  $t1, 44, not_found     # Dau ','
128     add  $s7, $s7, 1
129
130     jal check_operand
131     nop
132
133     # KIEM TRA KY TU THUA
134     j check_none
135
136     # Tra lai $ra de tro ve main
137     lw   $ra, 0($sp)
138     addi $sp, $sp, 4
139     jr   $ra
140
141 #-------------------------------------------------------------
142 # 2.1 @check_opcode: Kiem tra cau lenh
143 # - Buoc 1: Lay cac opcode trong command da nhap
```

```
#           Xoa cac dau cach thua phia truoc
# - Buoc 2: So sanh voi trong bo tu dien xem co opcode do khong
#       - Neu khong co ket thuc va quay lai menu
#       - Meu co, luu lai dia chi opcode trong library va tiep tuc kiem tra
# $a0: command
# $a1: opcode
# $s7: index of command
# $t9: index of opcode
#-------------------------------------------------------------
check_opcode:
    la  $a0, command                # Dia chi cua command
    la  $a1, opcode                 # Dia chi cua opcode
    li  $t0, 0

remove_space_command:                   # Xoa cac dau cach phia truoc lenh
    add $t1, $a0, $t0
    lb  $t2, 0($t1)
    bne $t2, 32, end_remove_space_command       # Neu khong phai ' ' -> Ket thuc
    addi $t0, $t0, 1
    j remove_space_command
end_remove_space_command:

    li  $t9, 0                  # index for opcode
    li  $s6, 0                  # so luong cac ki tu cua opcode = 0
read_opcode:
    add $t1, $a0, $t0               # Dich bit cua command
    add $t2, $a1, $t9               # Dich bit cua opcode
    lb  $t3, 0($t1)

    beq $t3, 32, read_opcode_done           # Neu co dau cach ' ' ket thuc read opcode
    beq $t3, 10, read_opcode_done           # Neu dau '\n' ket thuc read opcode
    beq $t3, 0,  read_opcode_done           # Ket thuc chuoi

    sb  $t3, 0($t2)
    addi $t9, $t9, 1
    addi $t0, $t0, 1
    j read_opcode
read_opcode_done:

    addi $s6, $t9, 0                # $s6: So luong ki tu cua opcode
    add $s7, $s7, $t0               # luu index cua command
    la $a2, library
    li $t0, -11

check_opcode_inlib:
    addi $t0, $t0, 11               # Buoc nhay bang 10 de nhay den tung Instruction
    li $t1, 0                   # i = 0
    li $t2, 0                   # j = 0
    add $t1, $t1, $t0               # Cong buoc nhay

    compare_opcode:
        add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua tung Instruction
        lb  $t4, 0($t3)
        beq $t4, 0, not_found
        beq $t4, 42, check_len_opcode       # Neu gap ky tu '*' => Kiem tra do dai
        add $t5, $a1, $t2           # Load opcode
        lb  $t6, 0($t5)
        bne $t4, $t6, check_opcode_inlib    # So sanh 2 ki tu, neu khong bang nhau thi tinh den Instruction tiep theo.
        addi $t1, $t1, 1            # i = i + 1
```

```
203         addi $t2, $t2, 1            # j = j + 1
204         j compare_opcode
205     check_len_opcode:
206         bne $t2, $s6, check_opcode_inlib
207 end_check_opcode_inlib:
208 
209     add $s5, $t0, $a2               # Luu lai vi tri Instruction trong
        Library.
210 
211     # ----- In thong tin ra man hinh -----
212     li $v0, 4
213     la $a0, opcode_mess
214     syscall
215 
216     la $a3, opcode
217     li $t0, 0
218     print_opcode:
219         beq $t0, $t9, end_print_opcode
220         add $t1, $a3, $t0
221         lb  $t2, 0($t1)
222         li $v0, 11
223         add $a0, $t2, $zero
224         syscall
225         addi $t0, $t0, 1
226         j print_opcode
227     end_print_opcode:
228 
229     li $v0, 4
230     la $a0, hopLe_mess
231     syscall
232 
233     jr $ra
234 
235 
236 #-------------------------------------------------------------
237 # 2.2 @check_operand:
238 # $a0: command.
239 # $s7: Luu index cua command.
240 # $s5: vi tri cua instruction trong library.
241 # $t9: Gia tri cua toan hang trong Library.
242 #-------------------------------------------------------------
243 
244 check_operand:
245     # Luu $ra de tro ve check
246     addi $sp, $sp, -4
247     sw   $ra, 0($sp)
248 
249     add $t9, $s5, $s3           # Tro toi operand trong Library
250     lb  $t9, 0($t9)
251     addi $t9, $t9, -48          # Char -> Number
252 
253     la  $a0, command
254     add $t0, $a0, $s7
255 
256     li $t1, 0                   # i = 0
257     space_remove:               # Xoa cac khoang trang thua
258         add $t2, $t0, $t1
259         lb  $t2, 0($t2)         # Lay ky tu tiep theo
260         bne $t2, 32, end_space_remove   # Ky tu ' '
261         addi $t1, $t1, 1        # i = i + 1
262         j space_remove
263     end_space_remove:
264 
265     add $s7, $s7, $t1           # Cap nhat lai index command
266 
267     li $s2, 0               # Tat kich hoat check number_register
```

```
268     li $t8, 0               # Khong co
269     beq $t8, $t9, check_none
270     li $t8, 1               # Thanh ghi
271     beq $t8, $t9, go_register
272     li $t8, 2               # So hang nguyen
273     beq $t8, $t9, go_number
274     li $t8, 3               # Ident
275     beq $t8, $t9, go_ident
276     li $t8, 4               # Check number & register
277     beq $t8, $t9, go_number_register
278
279 end_check_operand:
280     # Tra lai $ra de tro ve check
281     lw   $ra, 0($sp)
282     addi $sp, $sp, 4
283     jr   $ra
284
285 #-------------------------------------------------------------
286 #  jal toi cac ham check de kiem tra
287 #-------------------------------------------------------------
288     go_register:                # Check register
289         jal check_register
290         nop
291     j end_check_operand
292
293     go_number:              # Check number
294         la $a2, numberGroup
295         jal check_ident
296         nop
297     j end_check_operand
298
299     go_ident:               # Check Ident
300         la $a2, characterGroup
301         jal check_ident
302         nop
303     j end_check_operand
304
305     go_number_register:         # Check number-register
306         jal check_number_register
307         nop
308     j end_check_operand
309
310 #-------------------------------------------------------------
311 #  @check_none: Kiem tra xem con ky tu nao o cuoi khong
312 #-------------------------------------------------------------
313 check_none:
314     la $a0, command
315     add $t0, $a0, $s7
316
317     lb $t1, 0($t0)
318
319     beq $t1, 10, none_ok    # Ky tu '\n'
320     beq $t1, 0, none_ok # Ket thuc chuoi
321
322     j not_found
323
324 none_ok:
325     li $v0, 4
326     la $a0, completed_mess
327     syscall
328     j m_menu_start
329
330 #-------------------------------------------------------------
331 # @check_register: Kiem tra xem register co hop le hay khong
332 # $a0: command (vi tri luu command)
333 # $a1: token (vi tri luu thanh ghi)
```

```
334 # $a2: tokenRegisters
335 # $s7: Luu index cua command
336 # $t9: index cua token
337 #-------------------------------------------------------------
338
339 check_register:
340     la $a0, command
341     la $a1, token
342     la $a2, tokenRegisters
343     add $t0, $a0, $s7               # Tro den vi tri cac instruction
344
345     li $t1, 0                   # i = 0
346     li $t9, 0                   # index cua token
347
348 read_token_register:
349     add $t2, $t0, $t1               # command
350     add $t3, $a1, $t1               # token
351     lb $t4, 0($t2)
352
353     beq $t4, 41, end_read_token         # Gap ky tu ')'
354     beq $t4, 44, end_read_token         # Gap ky tu ' , '
355     beq $t4, 10, end_read_token         # Gap ky tu '\n'
356     beq $t4, 0, end_read_token          # Ket thuc
357
358     addi $t1, $t1, 1
359     beq $t4, 32, read_token_register        # Neu gap dau ' ' thi tiep tuc
360
361     sb $t4, 0($t3)
362     addi $t9, $t9, 1
363     j read_token_register
364
365 end_read_token:
366     add $s7, $s7, $t1               # Cap nhat lai gia tri index
367
368     li $t0, -6
369 compare_token_register:
370     addi $t0, $t0, 6                # Buoc nhay bang 6 de nhay den tung
        Register
371
372     li $t1, 0                   # i = 0
373     li $t2, 0                   # j = 0
374
375     add $t1, $t1, $t0               # Cong buoc nhay
376
377     compare_reg:
378         add $t3, $a2, $t1           # t3 tro thanh vi tri tro den dau cua
            tung Register
379         lb  $t4, 0($t3)
380         beq $t4, 0, not_found
381         beq $t4, 32, check_len_reg      # Neu gap ky tu ' ' => Kiem tra do
            dai
382
383         add $t5, $a1, $t2           # Load token
384         lb  $t6, 0($t5)
385
386         bne $t4, $t6, compare_token_register    # So sanh 2 ki tu, neu
            khong bang nhau thi tinh den Register tiep theo.
387         addi $t1, $t1, 1            # i = i + 1
388         addi $t2, $t2, 1            # j = j + 1
389         j compare_reg
390
391     check_len_reg:
392         bne $t2, $t9, compare_token_register    # Neu do dai khong bang
            nhau di den register tiep theo
393
394 end_compare_token_register:
```

```
395
396     # >>>>>>>>> In thong tin ra man hinh <<<<<<<<<<
397     beq $s2, 1, on_token_number_register
398     li $v0, 4
399     la $a0, toanHang_mess
400     syscall
401
402     la $a3, token
403     li $t0, 0
404     print_token_register:
405         beq $t0, $t9, end_print_token_register
406         add $t1, $a3, $t0
407         lb  $t2, 0($t1)
408         li $v0, 11
409         add $a0, $t2, $zero
410         syscall
411         addi $t0, $t0, 1
412         j print_token_register
413     end_print_token_register:
414
415     li $v0, 4
416     la $a0, hopLe_mess
417     syscall
418     jr $ra
419
420 on_token_number_register:
421
422     la $a3, token
423     li $t0, 0
424     print_on_token_register:
425         beq $t0, $t9, end_print_on_token_register
426         add $t1, $a3, $t0
427         lb  $t2, 0($t1)
428         li $v0, 11
429         add $a0, $t2, $zero
430         syscall
431         addi $t0, $t0, 1
432         j print_on_token_register
433     end_print_on_token_register:
434
435     li $v0, 11
436     li $a0, 41
437     syscall
438     li $v0, 4
439     la $a0, hopLe_mess
440     syscall
441     jr $ra
442
443 #-------------------------------------------------------------
444 # @check_ident: Kiem tra ident (label) HOAC number
445 # $a0: command (vi tri luu command)
446 # $a1: ident (vi tri luu ident)
447 # $a2: characterGroup | numberGroup
448 # $s7: luu index cua command
449 # $t9: index cua ident
450 #-------------------------------------------------------------
451 check_ident:
452     la $a0, command
453     la $a1, ident
454
455     add $t0, $a0, $s7           # Tro den vi tri cac instruction
456
457     li $t1, 0               # i = 0
458     li $t9, 0               # index cua ident
459
460 read_ident:
```

```
461     add $t2, $t0, $t1           # command
462     add $t3, $a1, $t1           # ident
463     lb $t4, 0($t2)
464
465     beq $t4, 40, end_read_ident     # Gap ky tu '('
466     beq $t4, 44, end_read_ident     # Gap ky tu ' , '
467     beq $t4, 10, end_read_ident     # Gap ky tu '\n'
468     beq $t4, 0, end_read_ident      # Ket thuc
469
470     addi $t1, $t1, 1
471     beq $t4, 32, read_ident         # Neu gap dau ' ' thi tiep tuc
472
473     sb $t4, 0($t3)
474     addi $t9, $t9, 1
475     j read_ident
476
477 end_read_ident:
478     add $s7, $s7, $t1           # Cap nhat lai gia tri index
479     beq $t9, 0, not_found           # Khong co label
480
481     #li $v0, 10
482     #syscall
483
484     li $t2, 0               # index cho Ident
485 compare_ident:
486     beq  $t2, $t9, end_compare_ident    # ket thuc chuoi
487     li   $t1, 0             # index cho characterGroup
488
489     add  $t3, $a1, $t2
490     lb   $t3, 0($t3)            # Tung char trong Ident
491
492     loop_Group:             # Kiem tra tung ky tu Ident co trong Group hay
            khong
493         add $t4, $a2, $t1
494         lb $t4, 0($t4)
495         beq $t4, 0, not_found       # Khong co -> Khong tim thay
496         beq $t4, $t3, end_loop_Group
497
498         addi $t1, $t1, 1
499         j loop_Group
500
501     end_loop_Group:
502
503     addi $t2, $t2, 1
504
505     j compare_ident
506
507 end_compare_ident:
508
509     beq $s2, 1, on_number_register
510
511     # ----- In thong tin ra man hinh -----
512     li $v0, 4
513     la $a0, toanHang_mess
514     syscall
515
516     la $a3, ident
517     li $t0, 0
518     print_ident:
519         beq $t0, $t9, end_print_ident
520         add $t1, $a3, $t0
521         lb  $t2, 0($t1)
522         li $v0, 11
523         add $a0, $t2, $zero
524         syscall
525         addi $t0, $t0, 1
```

```
526         j print_ident
527     end_print_ident:
528 
529     li $v0, 4
530     la $a0, hopLe_mess
531     syscall
532     jr $ra
533 
534 on_number_register:
535     li $v0, 4
536     la $a0, toanHang_mess
537     syscall
538 
539     la $a3, ident
540     li $t0, 0
541     print_on_ident:
542         beq $t0, $t9, end_print_on_ident
543         add $t1, $a3, $t0
544         lb  $t2, 0($t1)
545         li $v0, 11
546         add $a0, $t2, $zero
547         syscall
548         addi $t0, $t0, 1
549         j print_on_ident
550     end_print_on_ident:
551 
552     li $v0, 11
553     li $a0, 40
554     syscall
555     jr $ra
556 
557 #-------------------------------------------------------------
558 # @check_number_register: Kiem tra number - ident
559 # $a0: command (vi tri luu command)
560 # $s7: luu index cua command
561 # $s2: Luu kich hoat check number register
562 #-------------------------------------------------------------
563 
564 check_number_register:
565     # Luu $ra de tro ve
566     addi $sp, $sp, -4
567     sw   $ra, 0($sp)
568 
569     li $s2, 1               # Bat kich hoat number_register
570 
571     # Check number
572     la $a2, numberGroup
573     jal check_ident
574     nop
575 
576     la $a0, command
577     add $t0, $a0, $s7           # Tro den vi tri cac instruction
578     lb $t0, 0($t0)
579     bne $t0, 40, not_found          # Neu ki tu khong phai la dau '('
580     addi $s7, $s7, 1
581 
582     # Check register
583     jal check_register
584     nop
585     la $a0, command
586     add $t0, $a0, $s7           # Tro den vi tri cac instruction
587     lb $t0, 0($t0)
588     bne $t0, 41, not_found          # Neu ki tu khong phai la dau ')'
589     addi $s7, $s7, 1
590 
591     # Tra lai $ra de tro ve
```

```
592     lw   $ra, 0($sp)
593     addi $sp, $sp, 4
594     jr $ra
595 
596 #---------------------------------------------------------------
597 #  @not_found: Khong tim thay khuon dang lenh
598 #---------------------------------------------------------------
599 not_found:
600     li $v0, 4
601     la $a0, error_mess
602     syscall
603     j m_menu_start
604 
605 #---------------------------------------------------------------
606 #  END
607 #---------------------------------------------------------------
```