

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP LỚN
Thực hành Kiến trúc máy tính

GVHD: ThS. Lê Bá Vui
Mã lớp: 130937

Nhóm 18: Bùi Tôn Diệp – 20194505
Lê Huỳnh Đức – 20205067

A. Phân công bài tập

Bùi Tôn Điệp – bài 7

Lê Huỳnh Đức – bài 9

B. Thực hiện bài tập

I. Bài 7

1. 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.

- Cho biết lệnh hợp ngữ đó cần bao nhiêu chu kì thì mới thực hiện xong.

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, số chu kì thực hiện.

2. Phân tích cách thực hiện

- Yêu cầu người dùng nhập vào một câu lệnh cần kiểm tra, lưu vào một chuỗi `input`.
- Một câu lệnh hợp ngữ gồm 4 phần: mã opcode, toán hạng 1, toán hạng 2, toán hạng 3. Quy ước các dạng của toán hạng: 0 – null, 1 – register, 2 – constant, 3 – label, 4 – cụm `offset(base)`.

- Sau khi tách một thành phần của chuỗi tương ứng với thành phần của câu lệnh hợp ngữ. Đưa từng phần cắt được đi so sánh với kiểu dữ liệu quy ước. Nếu phù hợp thì tiếp tục so sánh các phần phía sau. Nếu không phù hợp thì báo cho người dùng opcode hoặc toán hạng không hợp lệ.
- + Đầu chương trình, khởi tạo một chuỗi opcode chuẩn, opcode sau khi cắt được thì sẽ so sánh với chuỗi này bằng cách so sánh từng ký tự, nếu ký tự khác nhau thì so sánh với opcode tiếp theo trong chuỗi cho đến khi kết thúc.
- + Nếu không có opcode phù hợp -> opcode nhập vào không hợp lệ, -> báo opcode không hợp lệ và exit. Nếu có opcode hợp lệ -> tìm khuôn dạng các toán hạng phù hợp với opcode. Đầu chương trình khi tạo chuỗi opcode chuẩn thì sẽ tạo một chuỗi khuôn dạng tương ứng với từng lệnh bằng cách so vị trí lệnh và khuôn lệnh. Nếu opcode phù hợp thì sẽ lấy được khuôn dạng lệnh và tiến hành kiểm tra lần lượt các toán hạng. Nếu có 1 toán hạng không hợp lệ -> báo ra màn hình run I/O và exit
- + Tương tự opcode chuẩn thì cũng có một chuỗi register chuẩn được tạo ở đầu chương trình. Nếu toán hạng có mã là 1 -> nó là thanh ghi và sẽ so sánh với chuỗi các register chuẩn.
- + Trường hợp toán hạng là constant: một constant hợp lệ có thể có ký tự đầu là dấu trừ '-' hoặc dấu cộng '+', các ký tự phía sau bắt buộc phải từ 0 đến 9.
- + Trường hợp toán hạng là label: label hợp lệ có thể có ký tự đầu là dấu gạch dưới '_' hoặc chữ thường, chữ hoa, nếu có các ký tự tiếp theo, thì các ký tự này bắt buộc phải là chữ cái, chữ số hoặc dấu gạch dưới.
- + Trường hợp toán hạng null: kiểm tra xem chuỗi vừa cắt được có ký tự nào hay không, nếu có -> không hợp lệ, nếu không có ký tự nào -> hợp lệ và thông báo câu lệnh hợp lệ -> Thoát chương trình.

- + Trường hợp toán hạng là một cụm offset(base): kiểm tra xem dấu mở ngoặc '(' và đóng ngoặc ')' có hợp lệ không? Có đủ 2 dấu đóng và mở ngoặc hay không. Nếu không -> không hợp lệ. Nếu có hợp lệ -> cut chuỗi này thành 2 thành phần là constant và register -> kiểm tra từng thành phần. Nếu có bất kì một thành phần nào sai -> Không hợp lệ. Ngược lại -> hợp lệ.
- Sau khi kiểm tra xong 3 toán hạng, chương trình sẽ kiểm tra xem còn kí tự nào khác hay không. Nếu còn -> câu lệnh không hợp lệ. Và báo kết quả của câu lệnh vừa nhập -> Hỏi người dùng có muốn kiểm tra tiếp một câu lệnh khác hay không?
- Nếu người dùng chọn không -> Kết thúc chương trình.

3. Mã nguồn

```
.data
#cau lenh mips gom opcode va 3 toan hang.
register: .asciiz "$zero-$at-$v0-$v1-$a0-$a1-$a2-$a3-$t0-$t1-$t2-$t3-$t4-$t5-$t6-$t7-$t8-$t9-$s1-$s2-$s3-$s4-$s5-$s6-$s7-$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-$23-$24-$25-$26-$27-$28-$29-$30-$31-"
#ma opcode hop le:
opcode: .asciiz "lw-lb-sw-sb-addi-add-addiu-addu-and-andi-beq-bne-div-divu-j-jal-lui-mfhi-mflo-mul-nop-nor-ori-sll-slt-slti-sub-subu-syscall-xor-xori-"
#quy uoc toan hang: 1 - thanh ghi, 2 - so, 3 - Label, 4 - offset(base): number(register), 0 - null
#toan hang tuong ung voi cac opcode tren:
operand: .asciiz "140-140-140-140-112-111-112-111-111-112-113-113-110-110-300-300-120-100-100-111-000-111-111-112-112-111-112-111-111-000-111-112-"

msg1: .asciiz "Nhap lenh can kiem tra: "
msg2: .asciiz "\nopcode: "
msg21: .asciiz ": hop le!\n"
msg22: .asciiz ": khong hop le!\n"
msg3: .asciiz "\nToan hang: "
msg4: .asciiz "\nCau lenh"
msg5: .asciiz "\kiem tra them 1 lenh nua? 1(yes)|0(no): "
input: .space 200 # chuoi dau vao
tmp: .space 20 #bien tmp luu thanh phan cat duoc
```

```

tmp2: .space 20 # luu khuon dang code
tmp3:  .space 20 # thanh phan cat duoc o offset(base)
.text
main:
Input: # lay dau vao
li      $v0, 4
la      $a0, msg1
syscall

li      $v0, 8
la      $a0, input
li      $a1, 200
syscall

#tach chu va so sanh
la      $s0, input #dia chi input
add     $s1, $zero, $zero # i -> dem tung ky tu trong tmp
readOpcode:
add     $a0, $s0, $zero # truyen tham so vao cutComponent
add     $a1, $s1, $zero #
la      $a2, tmp
jal     cutComponent
add     $s1, $v0, $zero #
add     $s7, $v1, $zero #so ky tu co trong opcode
checkOpcode:
la      $a0, tmp
add     $a1, $s7, $zero
la      $a2, opcode
jal     compareOpcode
add     $s2, $v0, $zero #check opcode
add     $s3, $v1, $zero #count matching voi khuon dang toan hang
li      $v0, 4
la      $a0, msg2
syscall

li      $v0, 4
la      $a0, tmp
syscall

bne     $s2, $zero, validOpcode # neu opcode hop le -> valid
invalidOpcode: #opcode ko hop le
li      $v0, 4
la      $a0, msg22

```

```

syscall
j      exit
validOpcode:
li     $v0, 4
la     $a0, msg21
syscall

#----- lay khuon dang tuong ung voi opcode
la     $a0, operand
add    $a1, $s3, $zero #truyen vao count
jal    getOperand #tra ve tmp2 - khuon dang

li     $v0, 4
la     $a0, tmp2
syscall

la     $s4, tmp2      #khuon dang
add    $s5, $zero, $zero #toan hang 1 2 3 - dem
add    $t9, $zero, 48 #0
addi   $t8, $zero, 49 #1
addi   $t7, $zero, 50 #2
addi   $t6, $zero, 51 #3
addi   $t5, $zero, 52 #4
Cmp: # kiem tra dang cua tung toan hang va check
slti   $t0, $s5, 3
beq    $t0, $zero, end
#----- lay toan hang 1
add    $a0, $s0, $zero
add    $a1, $s1, $zero
la     $a2, tmp
jal    cutComponent
add    $s1, $v0, $zero
add    $s7, $v1, $zero #so ky tu co trong tmp
#- so sanh toan hang 1
add    $t0, $s5, $s4
lb     $s6, 0($t0) #dang cua toan hang i
beq    $s6, $t8, reg
beq    $s6, $t7, number
beq    $s6, $t6, label
beq    $s6, $t5, offsetbase

```

```

beq    $s6, $t9, null
reg:
la      $a0, tmp
add     $a1, $s7, $zero
la      $a2, register
#tra ve 0 -> error, 1 -> ok
jal     compareOpcode
j       checkValid
number:
la      $a0, tmp
add     $a1, $s7, $zero
jal     checkNumber
j       checkValid
label:
la      $a0, tmp
add     $a1, $s7, $zero
jal     checkLabel
j       checkValid
offsetbase:
la      $a0, tmp
add     $a1, $s7, $zero
jal     checkOffsetBase
j       checkValid
null:
j       print
checkValid:
add     $s2, $v0, $zero
li      $v0, 4
la      $a0, msg3
syscall
li      $v0, 4
la      $a0, tmp
syscall
beq     $s2, $zero, error
j       ok
updateCheck:    #buoc lap
addi    $s5, $s5, 1
j       Cmp

```

```

error:
li      $v0, 4
la      $a0, msg22
syscall
j      exit
ok:
li      $v0, 4
la      $a0, msg21
syscall
j      updateCheck
end:
add     $a0, $s0, $zero
add     $a1, $s1, $zero
jal     cutComponent
add     $s1, $v0, $zero #i hien tai
add     $s7, $v1, $zero #so ky tu co trong tmp
print:
li      $v0, 4
la      $a0, msg4
syscall
bne     $s7, $zero, error
li      $v0, 4
la      $a0, msg21
syscall
exit:
repaetMain:
li      $v0, 4
la      $a0, msg5
syscall
li      $v0, 8
la      $a0, input
li      $a1, 100
syscall
checkRepeat:
addi    $t2, $zero, 48
addi    $t3, $zero, 49
add     $t0, $a0, $zero #ki tu dau tien
lb      $t0, 0($t0)
beq     $t0, $t2, out# =0

```



```

beq    $t0, $t3, main
j      repaetMain
out:
li $v0, 10 #exit
syscall
#.....
# tach toan hang,, opcode tu chuoi dau vao
# a0 address input, a1 i-> dem tmp. a2 address tmp
# v0 i = i+ strlen(tmp), v1 strlen(tmp)
#.....
cutComponent:
addi    $sp, $sp, -20
sw      $ra, 16($sp)
sw      $s0, 12($sp) # space
sw      $s2, 8($sp)    #j
sw      $s3, 4($sp) #input[i]
sw      $s4, 0($sp) #dau phay = 44

addi    $s0, $zero, 32 #space
addi    $t2, $zero, 10 #\n
addi    $s4, $zero, 44 #dau phay = 44
addi    $t3, $zero, 9 #\t
checkSpace: #bo qua , \t space
add     $t0, $a0, $a1 #dia chi input[i]
lb      $s3, 0($t0) #input[i]
beq     $s3, $s0 cutSpace #
beq     $s3, $t3 cutSpace
beq     $s3, $s4 cutSpace #
j       cut
cutSpace:
addi    $a1, $a1, 1
j       checkSpace
cut:
add     $s2, $zero, $zero #j = 0
loopCut:
beq     $s3, $s0 endCut
beq     $s3, $s4, endCut
beq     $s3, $zero, endCut
beq     $s3, $t2, endCut

```

```

beq    $s3, $t3 endCut
add    $t0, $a2, $s2 #dia chi tmp[j]
sb     $s3, 0($t0) #luu tmp[j]
addi   $a1, $a1, 1
add    $t0, $a0, $a1 #dia chi input[i]
lb     $s3, 0($t0) #input[i]

```

```

addi   $s2, $s2, 1
j      loopCut

```

endCut:

```

add    $t0, $a2, $s2 #dia chi tmp[j]
sb     $zero, 0($t0) #luu tmp[j] = '\0'
add    $v0, $a1, $zero
add    $v1, $s2, $zero
lw     $ra, 16($sp)
lw     $s0, 12($sp)
lw     $s2, 8($sp)
lw     $s3, 4($sp)
lw     $s4, 0($sp)
addi   $sp, $sp, 20
jr     $ra

```

#.....

so sanh toan hang, opcode voi toan hang, opcode chuan

a0 address tmp, a1 strlen(tmp), a2 adress cua chuoi opcode, register chu?n

v0 0|1, v1 count vi tri cua opcode

#.....

compareOpcode:

```

addi   $sp, $sp, -24
sw     $ra, 20($sp)
sw     $s1, 16($sp) #i -> opcode
sw     $s2, 12($sp) #j -> tmp
sw     $s3, 8($sp) #tmp[j]
sw     $s4, 4($sp)    #luu opcode[i]
sw     $s5, 0($sp) # - 45

```

```

beq    $a1, $zero, endCmp

```

```

add    $s1, $zero, $zero

```

```

add    $s2, $zero, $zero
addi   $s5, $zero, 45
addi   $v0, $zero, 1
addi   $v1, $zero, 1
loopCmp:
add    $t0, $a2, $s1 #dia chi opcode[i]
lb     $s4, 0($t0) #luu opcode[i]
beq    $s4, $s5, checkCmp
beq    $s4, $zero, endCmp
add    $t0, $a0, $s2 #dia chi tmp[j]
lb     $s3, 0($t0) #luu tmp[j]
bne    $s3, $s4, falseCmp

addi   $s1, $s1, 1
addi   $s2, $s2, 1
j      loopCmp
checkCmp:
bne    $a1, $s2, falseCmp
trueCmp:
addi   $v0, $zero, 1
j      endF

falseCmp:
addi   $v0, $zero, 0
addi   $s2, $zero, 0
loopXspace:
beq    $s4, $s5, Xspace
addi   $s1, $s1, 1
add    $t0, $a2, $s1 #dia chi opcode[i]
lb     $s4, 0($t0) #luu opcode[i]
j      loopXspace
Xspace:
add    $v1, $v1, 1
addi   $s1, $s1, 1
j      loopCmp
endCmp:
addi   $v0, $zero, 0
endF:
addi   $v1, $v1, -1

```

```

lw      $ra, 20($sp)
lw      $s1, 16($sp)
lw      $s2, 12($sp)
lw      $s3, 8($sp)
lw      $s4, 4($sp)
lw      $s5, 0($sp)
addi    $sp, $sp, 24
jr      $ra
#.....
# lay khuon dang toan hang ung voi opcode
# a0 address chuoi operand - vi tri tuong ung voi opcode, a1 count
# tra ve chuoi opcode tuong ung o tmp2 =
#.....-.....
getOperand:
addi    $sp, $sp, -20
sw      $s0, 16($sp) #i
sw      $s1, 12($sp) #op[i]
sw      $s2, 8($sp) # 45
sw      $s3, 4($sp) # address tmp2
sw      $s4, 0($sp)    # j

addi    $t0, $zero, 4 #moi khuon dang chiem 4 byte
mul     $s0, $a1, $t0 # i hien tai
addi    $s2, $zero, 45 # -
la      $s3, tmp2
add     $s4, $zero, $zero #j
loopGet:
add     $t0, $a0, $s0 #dia chi op
lb      $s1, 0($t0)
beq     $s1, $s2, endGet #gap - -> out
add     $t0, $s3, $s4 #dia chi tmp[i]
sb      $s1, 0($t0)

addi    $s0, $s0, 1
addi    $s4, $s4, 1
j       loopGet
endGet:
add     $t0, $s3, $s4 #dia chi tmp[i]
sb      $zero, 0($t0)

```

```

lw      $s0, 16($sp) #i
lw      $s1, 12($sp) #op[i]
lw      $s2, 8($sp) #
lw      $s3, 4($sp) #
lw      $s4, 0($sp)   #
addi    $sp, $sp, 20
jr      $ra
# .....
# kiem tra chuoi tmp co la so hay ko -> 0 -> sai, 1-> dung
# a0 address tmp, a1 strlen(tmp)
# v0 0|1
# .....
checkNumber:
add     $sp, $sp, -24
sw      $ra, 20($sp)
sw      $s4, 16($sp) #+
sw      $s3, 12($sp) #-
sw      $s0, 8($sp)
sw      $s1, 4($sp)
sw      $s2, 0($sp) #1
add     $v0, $zero, 0
add     $s0, $zero, $zero #dem i

beq     $a1, $zero, endCheckN
checkFirstN:
addi    $s3, $zero, 45 # -
addi    $s4, $zero, 43 # +
addi    $s2, $zero, 1
add     $t0, $a0, $s0 #toanhang[i]
lb      $s1, 0($t0)
#check - + -> 123
checkMinus:
bne     $s1, $s3, checkPlus
beq     $a1, $s2, endCheckN
j       update
checkPlus:
bne     $s1, $s4, _123
beq     $a1, $s2, endCheckN
j       update

```

```
#lb      $t2, 0($s1)
```

```
checkI:
```

```
slt      $t0, $s0, $a1
```

```
beq      $t0, $zero, trueN
```

```
add      $t0, $a0, $s0 #toanhang[i]
```

```
lb       $s1, 0($t0)
```

```
_123: #48 -> 57
```

```
slti     $t0, $s1, 48
```

```
bne      $t0, $zero, endCheckN
```

```
slti     $t0, $s1, 58
```

```
beq      $t0, $zero, endCheckN
```

```
update:
```

```
addi     $s0, $s0, 1
```

```
j        checkI
```

```
trueN:
```

```
addi     $v0, $v0, 1
```

```
endCheckN:
```

```
lw       $ra, 20($sp)
```

```
lw       $s4, 16($sp) #+
```

```
lw       $s3, 12($sp) #-
```

```
lw       $s0, 8($sp)
```

```
lw       $s1, 4($sp)
```

```
lw       $s2, 0($sp)
```

```
add      $sp, $sp, 24
```

```
jr       $ra
```

```
#.....
```

```
# kiểm tra chuỗi tmp có là Label hay không, ký tự đầu tiên: _ | A -> _ | A | 1
```

```
# a0 -> address tmp, a1 strlen(tmp)
```

```
# v0 0|1
```

```
#.....
```

```
checkLabel:
```

```
add      $sp, $sp, -12
```

```
sw       $ra, 8($sp)
```

```
sw       $s1, 4($sp)
```

```
sw       $s0, 0($sp)
```

```
add      $v0, $zero, 0
```

```
add      $s0, $zero, $zero #đếm i
```

```
beq    $a1, $zero, endCheckL
```

```
checkFirstChar:
```

```
add    $t0, $a0, $s0 #toanhang[i]
```

```
lb     $s1, 0($t0)
```

```
j      ABC
```

```
checkIL:
```

```
slt    $t0, $s0, $a1
```

```
beq    $t0, $zero, trueL
```

```
add    $t0, $a0, $s0 #toanhang[i]
```

```
lb     $s1, 0($t0)
```

```
_123L: #48 -> 57
```

```
slti   $t0, $s1, 48
```

```
bne    $t0, $zero, endCheckL
```

```
slti   $t0, $s1, 58
```

```
beq    $t0, $zero, ABC
```

```
addi   $s0, $s0, 1
```

```
j      checkIL
```

```
ABC: #65 -> 90
```

```
slti   $t0, $s1, 65
```

```
bne    $t0, $zero, endCheckL
```

```
slti   $t0, $s1, 91
```

```
beq    $t0, $zero, _
```

```
addi   $s0, $s0, 1
```

```
j      checkIL
```

```
_:
```

```
add    $t0, $zero, 95
```

```
bne    $s1, $t0, abc
```

```
addi   $s0, $s0, 1
```

```
j      checkIL
```

```
abc: #97 -> 122
```

```
slti   $t0, $s1, 97
```

```
bne    $t0, $zero, endCheckL
```

```
slti   $t0, $s1, 123
```

```
beq    $t0, $zero, endCheckL
```

```
addi   $s0, $s0, 1
```

```
j      checkIL
```

```
trueL:
```

```

addi    $v0, $v0, 1
endCheckL:
sw      $ra, 8($sp)
lw      $s1, 4($sp)
lw      $s0, 0($sp)
add     $sp, $sp, 12
jr      $ra
#.....
# kiểm tra chuỗi tmp có đúng cấu trúc offset(base) hay không
# a0 -> address tmp, a1 strlen(tmp)
# v0 0|1
#.....
#a0, tmp a1 strlen
checkOffsetBase:
#0($s1) -> 0_$s1_
add     $sp, $sp, -28
sw      $ra, 24($sp)
sw      $s5, 20($sp) #so ki cut dk
sw      $s4, 16($sp) # )
sw      $s3, 12($sp) # (
sw      $s2, 8($sp) #check
sw      $s1, 4($sp) # tmp[i]
sw      $s0, 0($sp) # dem i
checkO:
slti    $t0, $a1, 5 #it nhat 5 kis tu, vd: 0($s1)
bne     $t0, $zero, falseCheck
addi    $s3, $zero, 40
addi    $s4, $zero, 41
add     $s0, $zero, $zero #i
add     $s2, $zero, $zero #check
addi    $t2, $zero, 1
loopCheck:
add     $t0, $a0, $s0 #dia chi tmp[i]
lb      $s1, 0($t0)
beq     $s1, $zero, endLoopO
beq     $s1, $s3, open_
beq     $s1, $s4, close_
j       updateO
open_:

```



```

bne    $s2, $zero, falseCheck
addi   $s2, $s2, 1
addi   $t1, $zero, 32
sb     $t1, 0($t0)
j      updateO
close_:
bne    $s2, $t2, falseCheck
addi   $s2, $s2, 1
sb     $zero, 0($t0)

addi   $s0, $s0, 1
bne    $s0, $a1, falseCheck

updateO:
addi   $s0, $s0, 1
j      loopCheck
endLoopO:
addi   $t2, $t2, 1 # =2
bne    $s2, $t2, falseCheck
#.....
trueCheck:
add    $s0, $zero, $zero #i
#cut
sw     $a0, -8($sp)
sw     $a1, -4($sp)

la     $a0, tmp
add    $a1, $s0, $zero
la     $a2, tmp3
jal    cutComponent
add    $s0, $v0, $zero
add    $s5, $v1, $zero #so ky tu co trong cutword

lw     $a0, -8($sp)
lw     $a1, -4($sp)
#check number
sw     $a0, -8($sp)
sw     $a1, -4($sp)
la     $a0, tmp3

```

```

add    $a1, $s5, $zero
jal    checkNumber
add    $s2, $v0, $zero
lw     $a0, -8($sp)
lw     $a1, -4($sp)
beq    $s2, $zero, falseCheck
#cut
sw     $a0, -8($sp)
sw     $a1, -4($sp)

la     $a0, tmp
add    $a1, $s0, $zero
la     $a2, tmp3
jal    cutComponent
add    $s0, $v0, $zero
add    $s5, $v1, $zero #so ky tu co trong cutword

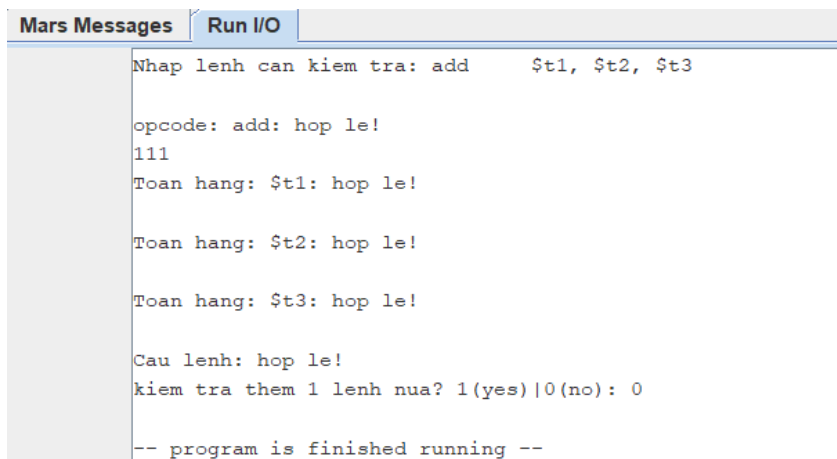
lw     $a0, -8($sp)
lw     $a1, -4($sp)
#checkReg
sw     $a0, -8($sp)
sw     $a1, -4($sp)
sw     $a2, -16($sp)
la     $a0, tmp3
add    $a1, $s5, $zero
la     $a2, register
#tra ve 0 -> error, 1 -> ok
jal    compareOpcode
add    $s2, $v0, $zero
lw     $a0, -8($sp)
lw     $a1, -4($sp)
lw     $a2, -12($sp)
beq    $s2, $zero, falseCheck
#->ket luan
addi   $v0, $zero, 1
j      endO
falseCheck:
add    $v0, $zero, $zero
j      endO

```

endO:

```
lw    $ra, 24($sp)
lw    $s5, 20($sp) #so ki cut dk
lw    $s4, 16($sp) #
lw    $s3, 12($sp) #
lw    $s2, 8($sp)
lw    $s1, 4($sp)
lw    $s0, 0($sp) #
add    $sp, $sp, 28
jr     $ra
```

4. Hình ảnh kết quả mô phỏng



```
Mars Messages Run I/O
Nhap lenh can kiem tra: add    $t1, $t2, $t3

opcode: add: hop le!
111
Toan hang: $t1: hop le!

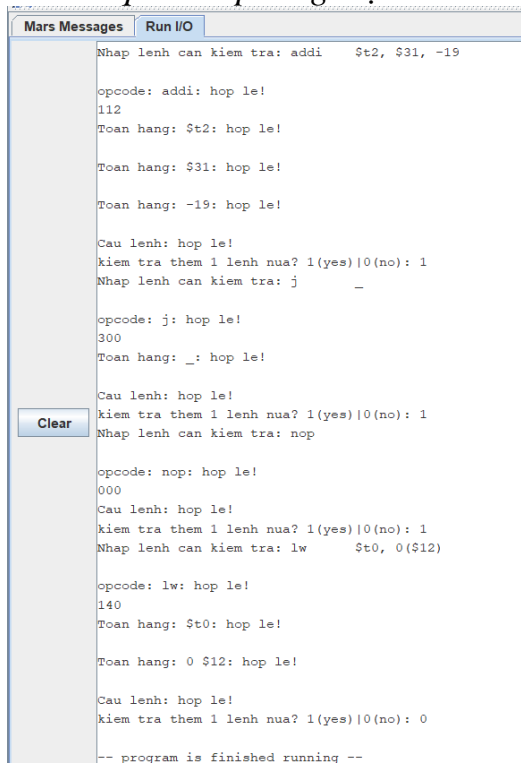
Toan hang: $t2: hop le!

Toan hang: $t3: hop le!

Cau lenh: hop le!
kiem tra them 1 lenh nua? 1(yes)|0(no): 0

-- program is finished running --
```

- *Kết quả mô phỏng một số trường hợp:*



```
Mars Messages Run I/O
Nhap lenh can kiem tra: addi    $t2, $31, -19

opcode: addi: hop le!
112
Toan hang: $t2: hop le!

Toan hang: $31: hop le!

Toan hang: -19: hop le!

Cau lenh: hop le!
kiem tra them 1 lenh nua? 1(yes)|0(no): 1
Nhap lenh can kiem tra: j      _

opcode: j: hop le!
300
Toan hang: _: hop le!

Cau lenh: hop le!
kiem tra them 1 lenh nua? 1(yes)|0(no): 1
Nhap lenh can kiem tra: nop

opcode: nop: hop le!
000
Cau lenh: hop le!
kiem tra them 1 lenh nua? 1(yes)|0(no): 1
Nhap lenh can kiem tra: lw      $t0, 0($12)

opcode: lw: hop le!
140
Toan hang: $t0: hop le!

Toan hang: 0 $12: hop le!

Cau lenh: hop le!
kiem tra them 1 lenh nua? 1(yes)|0(no): 0

-- program is finished running --
```

II. Bài 9

1. Đề bài

Vẽ hình bằng kí tự ascii Cho hình ảnh đã được chuyển thành các kí tự ascii như hình vẽ. Đây là hình của chữ DCE có viền * và màu là các con số

```
*****
*3333333333333*
*2222222222222*
*33333*****
*22222*****22222*
*33333*
*22222*      *22222*
*33333*****
*22222*      *22222*      *****
*3333333333333*
*22222*      *22222*      **11111*****111*
*33333*****
*22222*      *22222*      **1111**      **
*33333*
*22222*      *22222*      *1111*
*33333*****
*22222*****22222*      *1111*
*3333333333333*
*22222222222222*      *1111*
*****
*11111*
*1111*
*1111****      *****
**111111***111*
*****
dce.hust.edu.vn
```

- Hãy hiển thị hình ảnh trên lên giao diện console (hoặc giao diện Display trong công cụ giả lập Keyboard and Display MMIO Simulator)
- Hãy sửa ảnh để các chữ cái DCE chỉ còn lại viền, không còn màu số ở giữa, và hiển thị
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các hoạt tiết đính kèm cũng được phép di chuyển theo.
- Hãy nhập từ bàn phím kí tự màu cho chữ D, C, E, rồi hiển thị ảnh trên với màu mới.

2. Phân tích cách thực hiện

Tách hình ảnh thành từng string theo từng dòng. (16 dòng => 16 string)

- Hiển thị hình ảnh
 - Sử dụng vòng lặp để in ra từng string tương ứng từng dòng
- Hiển thị hình ảnh chỉ có viền, không có màu
 - Duyệt từng kí tự theo từng dòng
 - Nếu gặp chữ số (≥ 0 & ≤ 9) thì thay kí tự đó bằng space để xóa màu
 - Nếu gặp kí tự khác chữ số thì in ra như bình thường

- Hiển thị ECD

- Chia hình ảnh thành 4 phần:
 - Chữ D: Từ cột 0 đến cột 22
 - Chữ C: Từ cột 23 đến cột 42
 - Chữ E: Từ cột 43 đến cột 58
 - Còn lại: Từ cột 59 đến 61
- In từng kí tự theo từng dòng lần lượt các kí tự từ vị trí 43 => 58, sau đó từ cột 23 => 42, sau đó từ cột 0 => 22 và từ cột 59 => 61

- Đổi màu

- Lưu các màu hiện tại của D, C, E lần lượt vào các thanh ghi t1, t3, t5
- Nhập màu muốn thay đổi lần lượt cho D, C, E và lưu vào các thanh ghi t2, t4, t6. Nếu số nhập không phải màu từ 0 => 9 yêu cầu nhập lại
- Duyệt từng kí tự theo từng dòng lần lượt theo từng chữ cái
 - Chữ D (0->23): so sánh kí tự hiện tại với thanh ghi t1 lưu màu lúc đầu của D, nếu bằng nhau thì lưu giá trị màu mới (t2) vào kí tự hiện tại, nếu không thì duyệt kí tự tiếp theo. Nếu kí tự hiện tại ở vị trí 24 thì duyệt sang chữ C
 - Tương tự với chữ C (24->43) và E(44->61)
 - Sau khi duyệt hết 1 dòng thì in ra và duyệt dòng tiếp theo cho đến khi hết 16 dòng

3. Mã nguồn

```
.data
String0: .space 5000
String1: .asciiz "***** \n"
String2: .asciiz "***** *333333333333* \n"
String3: .asciiz " *222222222222222* *33333***** \n"
String4: .asciiz " *22222*****22222* *33333* \n"
String5: .asciiz " *22222* *22222* *33333***** \n"
String6: .asciiz " *22222* *22222* ***** *333333333333* \n"
String7: .asciiz " *22222* *22222* **11111*****111* *33333***** \n"
String8: .asciiz " *22222* *22222* **1111** ** *33333* \n"
```

```

String9: .asciiz " *22222*      *22222* *1111*      *33333***** \n"
String10: .asciiz " *22222*****22222* *11111*      *33333333333333* \n"
String11: .asciiz " *2222222222222222* *11111*      ***** \n"
String12: .asciiz " ***** *11111*      \n"
String13: .asciiz " --- *1111**      \n"
String14: .asciiz " / o o \\\n"
String15: .asciiz " \\\n"
String16: .asciiz " ----- ***** dce.hust.edu.vn \n"

```

```

Message1: .asciiz "\n\n.....Print DCE .....
.....\n\n"
Message2: .asciiz "\n\n.....Print without color .....
.....\n\n"
Message3: .asciiz "\n\n----- Change position of D & E -----
.....\n\n"
Message4: .asciiz "\n\n.....Change color .....
.....\n\n"

```

```

StringD: .asciiz "New color of D (0->9): "
StringC: .asciiz "New color of C (0->9): "
StringE: .asciiz "New color of E (0->9): "
.text

```

```

##### print #####
title1: la $a0, Message1
li $v0,4
syscall
main1: li $s0,0      # i=0
la $s2, String1
Loop_print: addi $a0,$s2,0
li $v0,4
syscall
addi $s2, $s2, 62 # next string(line)
addi $s0, $s0,1    # i++
beq $s0, 16, end_print # if i=16 => end print (line =16)
j Loop_print
end_print:

```

```

##### IN KHONG MAU #####
title2: la $a0, Message2
li $v0,4
syscall
main2: li $s0,0      #i=0
la $s2, String1
Loop: li $s1,0#j=0
print_line: add $t1, $s2, $s1 # t1 = address of stringX[j]
lb $t2, 0($t1) # t2 = stringX[j]
blt $t2, 48, printc # t2 < 48('0') jump printc

```

```

bgt $t2, 57, printc # t2 > 57('9') jump printc
addi $t2, $zero, 32
printc: addi $a0, $t2, 0      # print character
li $v0, 11
syscall
addi $s1, $s1, 1 # j= j+1
beq $s1, 62, next_line
j print_line
next_line:      addi $s0, $s0, 1 #i= i+1
addi $s2, $s2, 62      # next string
beq $s0, 16, end_loop
j Loop
end_loop:

##### change D & E #####
title3: la $a0, Message3
li $v0, 4
syscall
# D: 0-> 22
# C: 23 -> 42
# E: 43 -> 58
main3: li $s0, 0 #i=0
la $s2, String1
Loop1:      li $s1, 43 # j=43
Print_E:      add $t0, $s2, $s1 # t0 = address of stringX[j]
lb $t2, 0($t0) # t2 = stringX[j]
addi $a0, $t2, 0
li $v0, 11      # print character
syscall
addi $s1, $s1, 1 # j++
beq $s1, 59, Loop2 # if j = 59 jump loop2
j Print_E
Loop2: li $s1, 23      # j=23
Print_C:      add $t0, $s2, $s1 # t0 = address of stringX[j]
lb $t2, 0($t0) # t2 = stringX[j]
addi $a0, $t2, 0
li $v0, 11      # print character
syscall
addi $s1, $s1, 1
beq $s1, 43, Loop3      # if i =43 jump loop3
j Print_C
Loop3:      li $s1, 0      # j=0
Print_D: add $t0, $s2, $s1 # t0 = address of stringX[j]
lb $t2, 0($t0) # t2 = stringX[j]
addi $a0, $t2, 0
li $v0, 11      # print character
syscall
addi $s1, $s1, 1

```

```

beq $s1,23,Loop4    # if i=23 jump loop4
j Print_D
Loop4: li $s1,60      # j=59
Print: add $t0, $s2, $s1 # t0 = address of stringX[j]
lb $t2, 0($t0) # t2 = stringX[j]
addi $a0, $t2, 0
li $v0, 11          # print character
syscall
addi $s1,$s1,1
beq $s1,62,Line # if j = 62 => next line
j Print
Line:  addi $s0,$s0,1
beq $s0,16,end      # if i = 16 end
addi $s2,$s2,62
j Loop1
end:

##### ?ô?i ma?u #####
title4: la $a0, Message4
li $v0,4
syscall
####
li $t1, 50 # color base of D
li $t3, 49 # color base of C
li $t5, 51 # color base of E

#### input color
Input_D: li $v0,4
la $a0, StringD
syscall
li $v0,5
syscall
blt $v0,0, Input_D #if v0<0 input again
bgt $v0,9, Input_D #if v0>9 input again
addi $t2,$v0,48 # '0' ascii: 48 , t2: store new color of D
Input_C: li $v0,4
la $a0, StringC
syscall
li $v0,5
syscall
blt $v0,0, Input_C #if v0<0 input again
bgt $v0,9, Input_C #if v0>9 input again
addi $t4,$v0,48 # t4: store new color of C
Input_E: li $v0,4
la $a0, StringE
syscall
li $v0,5
syscall

```



```

blt $v0,0, Input_E #if v0<0 input again bgt
$v0,9, Input_E #if v0>9 input again addi
$t6,$v0,48 # t6: store new color of E

```

```

####

```

```

main4: li $s0, 0 # i=0la
$s2,String1

```

```

row: li $s1,0 # j=0
check: add $t0,$s2, $s1 # t0 = address of StringX[j] lb
$a0,($t0) # a0 = stringX[j]
checkD:      bgt $s1, 23, checkC # if j>23 check C
beq $a0, $t1, fixD # if stringX[j] = color base => fix j
next_char
fixD:  sb $t2, 0($t0) # stote new color into stringX[j]j
next_char
checkC:      bgt $s1, 43, checkE # if j>43 check E
beq $a0, $t3, fixC # if stringX[j] = color base => fix j
next_char
fixC:  sb $t4, 0($t0) # stote new color into stringX[j]j
next_char
checkE:      #bgt $s1, 23, checkC # if j>23 check C
beq $a0, $t5, fixE # if stringX[j] = color base => fix
j next_char
fixE:
j next_char
next_char:   addi $s1,$s1, 1 # j++
beq $s1,62,end_row
j check
end_row:     addi $a0, $s2,0 # print lineli
$v0,4
syscall
addi $s0,$s0,1
beq $s0,16, end_change
addi $s2,$s2,62
j row
end_change:

```

➔ Kết quả thực hiện:

```

----- Print DCE -----

                                *****
*****
*2222222222222222*          *33333333333333*
*22222*****22222*          *33333*****
*22222*          *22222*      *33333*****
*22222*          *22222*      ***** *33333333333333*
*22222*          *22222*      *11111*****111* *33333*****
*22222*          *22222*      *1111*          * * *33333*
*22222*          *22222*      *1111*          *33333*****
*22222*****22222*          *11111*          *33333333333333*
*2222222222222222*          *11111*          *****
*****
---
/ o o \          *1111***** *****
\   > /          **111111**111*
-----          *****
                                dce.hust.edu.vn

```

- In hình ảnh không màu

[illegible]

- Thay đổi vị trí E & D

[illegible]

- Thay đổi màu cho DCE

```

Change color -----
New color of D (0->9): 0
New color of C (0->9): 8
New color of E (0->9): 7

*****
*****
*00000000000000000000*
*00000*****000000*
*00000*      *00000*
*00000*      *00000*      *****
*00000*      *00000*      **88888*****888*
*00000*      *00000*      **8888**          **
*00000*      *00000*      *8888*
*00000*****000000*      *88888*
*00000000000000000*      *88888*
*****
***
/ o o \
\   > /
-----
*****
dce.hust.edu.vn

```

- Khi nhập sai màu mới

```
----- Change color -----
New color of D (0->9): 11
New color of D (0->9): -1
New color of D (0->9):
```