TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

----------o0o----------

Báo Cáo Cuối kì

Thực Hành Kiến Trúc Máy Tính

Bài 10 và bài 6

Giảng viên : Nguyễn Thành Trung

Sinh viên thực hiện: Nguyễn Khắc Thái Bình – 20204944
Nhóm: **18** – Lớp: Việt Nhật 02 K65

Hà Nội, 7/2022

Mục lục

A. Bài 10

I. Đề bài

Sử dụng 2 ngoại vi là bàn phím và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, *, /. Do trên bàn phím không có các phím trên nên sẽ dùng các phím

-Bấm phím a để nhập phép tính +
- Bấm phím b để nhập phép tính –
- Bấm phím c để nhập phép tính *
- Bấm phím d để nhập phép tính /
- Bấm phím f để nhập phép =

II. Phân tích

1. Giả mã

*Nhập input và toán hạng

```
int a,b;
scanf(“%d%d”, &a, &b);
switch(chose){
   case ‘a':
      printf(“kết quả phép cộng: %d", a+b);
      break;
   case ‘b':
      printf(“kết quả phép trừ: %d", a-b);
      break;
   case ‘c':
      printf(“kết quả phép nhân: %d", a*b);
      break;
   case ‘d':
      printf(“kết quả phép chia: %d", a/b);
      break;}
```

*Hàm chuyển từ bàn phím hexa sang giá trị nguyên

```
    int r= a & 0xf;
    int c= (a & 0xf0) >> 4;
    int i = -1, j =-1;
    while ( r != 0) {
         i += 1;
          r >>1 ;
}
While( c != 0 ){
       j += 1;
       c >>= 1;}
return j+i*4;
```

2. Cách làm

-Từ đoạn giả mã, ta phải

+nhập 4 số từ digital lab sim gán lên đèn led

+tạo vòng lặp để truyền và ngắt các tham số đầu vào, hiện thị 2 số cuối trên led và lưu tham số vào thanh ghi

+tạo các case để đưa ra toán tử, gán các toán tử vào các phim a,b,c,d,f

+hiển thị kết quả, truyền 2 số cuối của kết quả lên đèn led

-Thứ tự bấm:

+nhập 4 số từ digital lab sim, đèn led hiển thị 2 số cuối

+khi nhập xong 4 chữ số của số thứ nhất, tự động nhập 4 số của số thứ 2
+nhập xong 4 số của số thứ 2, nhập toán tử + - * / bằng phím a,b,c,d
+nhấn phím f (dấu'=')
+hiển thị kết quả trên console, 2 chữ số cuối của kết quả trên led

3. Ý nghĩa các hàm

- **Start:** kích hoạt ngắt từ bàn phím hexa, lưu biến vào các thanh ghi
- **Loop :** tạo vòng lặp để nhập và ngắt số thứ nhất có 4 chữ số, rồi reset đèn
- **Loop1:** tạo vòng lặp để nhập và ngắt số thứ 2 có 4 chữ số, lưu số thứ 2 vào thanh ghi rồi reset đèn
- **Loop2 :** nhập toán tử từ bàn phím, gán toán tử + - * / lần lượt bằng các phím a,b,c,d, tạo các case để lựa chọn
- **Loop3:** gán dấu '=' vào phím f, hiển thị kết quả bằng cách nhảy lên loop2 chạy toán tử
- **Napgiatricholed :** hiển thị và truyền biến cho đèn led trái và phải
- **Hienthi:** lưu code của đèn vào 1 mảng, khi dùng thì gọi phần tử mảng lên
- **Keyscan:** chuyển từ bàn phím hexa sang số nguyên
- **Restore:** khôi phục các tham số truyền vào, lưu thành các bit
- **getInt1:** cho phép bấm phím ở hàng 1, các phim 0,1,2,3
- **show1:** truyền và hiển thị đèn trái và phải
- **getInt2:** cho phép bấm phim ở hàng 2, các phim 4,5,6,7
- **show2:** truyền và hiển thị đèn trái và phải
- **getInt3:** cho phép bấm phím ở hàng 3, các phim 8,9,a,b
- **show3:** truyền và hiển thị đèn trái và phải
- **getInt4:** cho phép bấm phím ở hàng 4, các phim c,d,f
- **show4:** truyền và hiển thị đèn trái và phải
- **Display:** lưu giá trị hiển thị vào 1 mảng, hiển thị lên đèn

III. Mã nguồn

```
.data

   welc:            .asciiz "¥n May tinh bo tui ¥n"

   p_int:           .asciiz "¥n Nhap vao so thu nhat co do dai 4 chu so VD: nhap 0010 de nhap so 10 .¥n "

   p_int1:          .asciiz "¥n Nhap vao so thu hai co do dai   4 chu so, VD: nhan 0001 de nhap so 1 "

p_toantu:       .asciiz "¥n Nhap vao toan tu ¥n Nhan a de nhap phep cong¥n Nhan b de nhap phep tru ¥n Nhan c de nhap phep nhan¥n Nhan d de nhap phep chia ¥n "

   ketqua:          .asciiz "¥n ket qua la :   "

   xuongdong:       .asciiz "¥n"

   err1:            .asciiz "¥n Xay ra loi. Xin hay nhap lai toan tu¥n"

   phep_nhan:       .asciiz "¥n Ban da nhap phep nhan. ¥n"

   phep_chia:       .asciiz "¥n Ban da nhap phep chia.   ¥n"

   phep_cong :      .asciiz "¥n Ban da nhap phep cong.¥n"

   phep_tru:        .asciiz "¥n Ban da nhap phep tru."
```

```
  sb_daubang:     .asciiz "¥n nhap f de hien thi ket qua"
  erram:          .asciiz "¥nKet qua la so am, khong hien thi duoc¥n."
  rep:            .asciiz "¥nNhap 0 de thoat khoi chuong trinh ¥n Nhap so khac 0 de tiep tuc chuong trinh ¥n "
  goodbye:        .asciiz "¥n Cam on ban da su dung chuong trinh, Tam biet hen gap lai¥n"

.eqv ZERO                   63    # Gia tri byte hien thi so 0 tren den LED
.eqv ONE                    6     # Gia tri byte hien thi so 1 tren den LED
.eqv TWO                    91    # Gia tri byte hien thi so 2 tren den LED
.eqv THREE                  79    # Gia tri byte hien thi so 3 tren den LED
.eqv FOUR                   102   # Gia tri byte hien thi so 4 tren den LED
.eqv FIVE                   109   # Gia tri byte hien thi so 6 tren den LED
.eqv SIX                    125   # Gia tri byte hien thi so 6 tren den LED
.eqv SEVEN                  7     # Gia tri byte hien thi so 7 tren den LED
.eqv EIGHT                  127   # Gia tri byte hien thi so 8 tren den LED
.eqv NINE                   111   # Gia tri byte hien thi so 9 tren den LED

.eqv IN_ADDRESS_HEXA_KEYBOARD     0xFFFF0012   # chua byte dieu khien dong cua ban phim

.eqv OUT_ADDRESS_HEXA_KEYBOARD   0xFFFF0014   # chua byte tra ve vi tri cua phim duoc bam

.eqv LEFT_LED               0xFFFF0010   # chua byte dieu khien den led ben phai
.eqv RIGHT_LED              0xFFFF0011   # chua byte dieu khien den led ben trai

.text
main:
start:
    la $a0,welc
    li $v0, 4
    syscall
```

```
    la $a0,p_int
    li $v0, 4
    syscall
    #   Kich hoat interupt ----------
  li $t1,IN_ADDRESS_HEXA_KEYBOARD
  li $t3,0x80        # Kich hoat interupt tu ban phim hexa
  sb $t3,0($t1)

    # Khai bao bien
     li $t6,0         # $t6: Bien gia tri so cua den LED trai
     li $t7,0         # $t7: Bien gia tri so cua den LED phai
     li $s1,0         # $s1: Gia tri lay ra
     li $s2,0         # $s2: toan tu lay ra
     li $s3,0         # $s3:   dau =
     li $s5,0         # so thu nhat
     li $s6,0         #so thu 2
     li $s4,0         # bien dem s4

# Vong lap cho tin hieu gian doan so thu nhat
Loop: beq $s4,4,nhapso2
     nop
     beq $s4,4,nhapso2
     nop
     beq $s4,4,nhapso2
     nop
     beq $s4,4,nhapso2
     b Loop
     beq $s4,4,nhapso2      #Wait for gian doan
     nop
```

```
        beq $s4,4,nhapso2
        b Loop

nhapso2:
    add   $s5,$s1,$0 # luu gia tri so thu nhat vao s5
    add   $s1,$0,$0   # reset s1
    addi $s4,$0,0     # reset bien dem s4
    move $a0,$s5
    li $v0,1
    syscall

    li $t6,0         # reset den
    li $t7,0
   la $a0,p_int1
    li $v0,4
    syscall
Loop1: # VOng lap cho interupt so thu 2
      beq $s4,4,nhaptoantu
      nop
      beq $s4,4,nhaptoantu
      nop
      beq $s4,4,nhaptoantu
      nop
      beq $s4,4,nhaptoantu
      b Loop1         #Wait for gian doan
      nop
      beq $s4,4,nhaptoantu
      b Loop1
 nhaptoantu:
```

```
        add $s6,$s1,$0 # LUU SO THU 2 VAO S6
        add $s1,$0,$0   #RESET S1
        addi $s4,$0,0   #RESET BIEN DEM S4
        la $a0,xuongdong
        li $v0, 4
        syscall
    move $a0,$s6   # IN SO THU 2
    li $v0,1
    syscall
    li $t6,0   # reset den led
    li $t7,0
    la $a0,p_toantu
    li $v0,4
    syscall

Loop2:
        beq $s4,1,nhaptoantu2
        nop
        beq $s4,1,nhaptoantu2
        nop
        beq $s4,1,nhaptoantu2
        nop
        beq $s4,1,nhaptoantu2
        b Loop2      #Wait for gian doan
        beq $s4,1,nhaptoantu2
        nop
        beq $s4,1,nhaptoantu2
        b Loop2
```

```
nhaptoantu2:
        add $a3,$0,$0
        add $s1,$0,$0 # reset s1
        addi $s4,$0,0 # reset bien dem s4

    cong1: bne $s2,1,tru1
            la $a0,sb_cong
            li $v0, 4
            syscall
            j baonhapdaubang
    tru1:   bne $s2,2,nhan1
            la $a0,phep_tru
              li $v0, 4
              syscall
              j baonhapdaubang
    nhan1: bne $s2,3,chia1
              la $a0,phep_nhan
              li $v0,4
              syscall
              j baonhapdaubang
    chia1: bne $s2,4,baonhapdaubang
              la $a0,phep_chia
              li $v0, 4
              syscall
              j baonhapdaubang
 baonhapdaubang:
            la $a0,sb_daubang
                  li $v0, 4
                    syscall
```

```
Loop3: # Cho nhap vao dau = de hien thi ket qua
  beq $s3,6, show
   nop
   beq $s3,6,show
   nop
   beq $s3,6,show
   nop
   beq $s3,6,show
   b Loop2      #Wait for gian doan
   beq $s3,6,show
   nop
   beq $s3,6,show
   b Loop3
   show:
   case_cong:   bne $s2,1,case_tru   # neu la phep cong
                addu $s7,$s5,$s6      # thuc hien phep cong
                la $a0,ketqua
                li $v0, 4
                syscall
                move $a0,$s7          # in ket qua ra console
                li $v0,1
                syscall
                j showketqua

   case_tru:    bne $s2,2,case_nhan
                la $a0,ketqua
                li $v0, 4
                syscall
```

```
                sub $s7,$s5,$s6
                move $a0,$s7
                li $v0,1
                syscall
                j showketqua
    case_nhan:  bne $s2,3,case_chia
                la $a0,ketqua
                li $v0,4
                syscall
                mul $s7,$s5,$s6
                move $a0,$s7
                li $v0,1
                syscall
                j showketqua
    case_chia:  bne $s2,4,pheptinhdf
                la $a0,ketqua
                li $v0,4
                syscall
                div $s7,$s5,$s6
                move $a0,$s7
                li $v0,1
                syscall
            j showketqua
 pheptinhdf: # bao loi
  la $a0, err1
  li $v0, 4
  syscall

showketqua: li $t9,0
```

```
            li $t8,0
            slt $k1,$s7,$0 # kiem tra ket   qua la so am hay khong
            bne $k1,$0,loisoam
            div $t8,$s7,10
            mfhi $t9
            beq $t8,0,napgiatricholed
            div $t8,$t8,10
            mfhi $t8
napgiatricholed: li $t2,LEFT_LED     # hien thi den LED trai
                 add $s0,$zero,$t9 # truyen bien left
                 jal hienthi
                 nop
                 li $t2,RIGHT_LED   # hien thi den LED phai
                 add $s0,$zero,$t8 # truyen bien right
                 jal hienthi
                 nop
                 j endmain
endmain:
        li $v0, 51
        la $a0,rep
        syscall
        beq $a0, $0,END
        nop
        j start
        nop
END:
       la $a0, goodbye
       li $v0, 4
       syscall
```

```
        la $v0, 10
        syscall
      loisoam: # bao loi ket qua am
                la $a0, erram
                li $v0, 4
                syscall
                li $v0, 51
                la $a0,rep
                syscall
                beq $a0, $0,END
                nop
                j start
                nop
                la $v0, 10
                syscall

   hienthi:
showleds_0: bne $s0,0,showleds_1   # case $s0 = 0
        li $t4,ZERO
        j napgiatri
showleds_1: bne $s0,1,showleds_2   # case $s0 = 1
        li $t4,ONE
        j napgiatri
showleds_2: bne $s0,2,showleds_3   # case $s0 = 2
        li $t4,TWO
        j napgiatri
showleds_3: bne $s0,3,showleds_4   # case $s0 = 3
        li $t4,THREE
```

```
        j napgiatri
showleds_4: bne $s0,4,showleds_5   # case $s0 = 4
        li $t4,FOUR
        j napgiatri
showleds_5: bne $s0,5,showleds_6   # case $s0 = 5
        li $t4,FIVE
        j napgiatri
showleds_6: bne $s0,6,showleds_7   # case $s0 = 6
        li $t4,SIX
        j napgiatri
showleds_7: bne $s0,7,showleds_8   # case $s0 = 7
        li $t4,SEVEN
        j napgiatri
showleds_8: bne $s0,8,showleds_9   # case $s0 = 8
        li $t4,EIGHT
        j napgiatri
showleds_9: bne $s0,9,showleds_df # case $s0 = 9
        li $t4,NINE
        j napgiatri
showleds_df:   jr $ra
napgiatri:      sb $t4,0($t2)
                jr $ra

# Xu ly khi xay ra interupt
.ktext 0x80000180          # Hien thi so vua bam len den led 7 doan
  # SAVE the current REG FILE to stack
keyscan: addi $sp,$sp,4      # Save $ra because we may change it later
            sw $ra,0($sp)
            addi $sp,$sp,4      # Save $ra because we may change it later
```

```
            sw   $at,0($sp)
            addi $sp,$sp,4      # Save $ra because we may change it later
            sw $v0,0($sp)
            addi $sp,$sp,4      # Save $a0, because we may change it later
            sw $a0,0($sp)
            addi $sp,$sp,4      # Save $t1, because we may change it later
            sw $t1,0($sp)
            addi $sp,$sp,4      # Save $t3, because we may change it later
            sw $t3,0($sp)
            addi $sp,$sp,4
            sw $s4,0($sp)
   # Xu ly
         addi $s4,$s4,1
         jal getInt1
         nop
         jal getInt2
         nop
         jal getInt3
         nop
        jal getInt4
         nop
next_pc:      mfc0 $at,$14      # $at <= Copro0.$14 = Coproc0.epc
         addi $at,$at,4           # $at = $at + 4
         mtc0 $at,$14              #Coproc0.$14 = Coproc0.epc <= $at
   # RESTORE the REG FILE from STACK
restore:
         lw $t3,0($sp)
         addi $sp,$sp,-4
         lw $t1,0($sp)
```

```
        addi $sp,$sp,-4
        lw $a0,0($sp)
        addi $sp,$sp,-4
        lw $v0,0($sp)
        addi $sp,$sp,-4
        lw $ra,0($sp)
        addi $sp,$sp,-4
        lw $t4,0($sp)
        addi $sp,$sp,-4
back_main:   eret
  # Thu tuc quet cac phim o hang 1 va xu ly
  # Tham so truyen vao:
  # Tra ve:
getInt1: addi $sp,$sp,4
        sw $ra,0($sp)
        li $t1,IN_ADDRESS_HEXA_KEYBOARD
        li $t3,0x81        # Kich hoat gian doan, cho phep bam phim o hang 1
        sb $t3,0($t1)
        li $t1,OUT_ADDRESS_HEXA_KEYBOARD
        lb $t3,0($t1)     # Nhan byte the hien vi tri cua phim duoc bam trong hang 1
case_0:    li $t5,0x11
        bne $t3,$t5,case_1   # case 0x11
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,0   # left = 0
        mul $s1,$s1,10
        add $s1,$s1,$t6     # factor=factor*10+left
        j show1
case_1:    li $t5,0x21
        bne $t3,$t5,case_2   # case 0x21
```

```
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,1    # left = 1
        mul $s1,$s1,10
        add $s1,$s1,$t6 # factor=factor*10+left
        j show1
case_2:     li $t5,0x41
        bne $t3,$t5,case_3   # case 0x41
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,2    # left = 2
        mul $s1,$s1,10
        add $s1,$s1,$t6 # factor=factor*10+left
        j show1
case_3:     li $t5,0xffffff81
        bne $t3,$t5,case_default1    # case 0xffffff81
        addi $t7,$t6,0                  # left=right
        addi $t6,$zero,3                # left = 3
        mul $s1,$s1,10
        add $s1,$s1,$t6                 # factor=factor*10+left
        j show1
show1:      li $t2,LEFT_LED     # hien thi den LED trai
        add $s0,$zero,$t6       # truyen bien left
        jal displayLED
        nop
        li $t2,RIGHT_LED    # hien thi den LED phai
        add $s0,$zero,$t7 # truyen bien right
        jal displayLED
        nop
case_default1:     j getInt1rt
getInt1rt: lw $ra,0($sp)
```

```
        addi $sp,$sp,-4
        jr $ra
   # Thu tuc quet cac phim o hang 2 va xu ly
   # Tham so truyen vao:
   # Tra ve:
getInt2: addi $sp,$sp,4
        sw $ra,0($sp)
        li $t1,IN_ADDRESS_HEXA_KEYBOARD
        li $t3,0x82        # Kich hoat gian doan, cho phep bam phim o hang 1
        sb $t3,0($t1)
        li $t1,OUT_ADDRESS_HEXA_KEYBOARD
        lb $t3,0($t1)     # Nhan byte the hien vi tri cua phim duoc bam trong hang 1
case_4:    li $t5,0x12
        bne $t3,$t5,case_5   # case 0x11
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,4   # left = 4
        mul $s1,$s1,10
        add $s1,$s1,$t6     # factor=factor*10+left
        j show2
case_5:    li $t5,0x22
        bne $t3,$t5,case_6   # case 0x21
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,5   # left = 5
        mul $s1,$s1,10
        add $s1,$s1,$t6     # factor=factor*10+left
        j show2
case_6:    li $t5,0x42
        bne $t3,$t5,case_7   # case 0x41
        addi $t7,$t6,0         # left=right
```

```
        addi $t6,$zero,6      # left = 6
        mul $s1,$s1,10
        add $s1,$s1,$t6       # factor=factor*10+left
        j show2
case_7:     li $t5,0xffffff82
        bne $t3,$t5,case_default2   # case 0xffffff81
        addi $t7,$t6,0                # left=right
        addi $t6,$zero,7              # left = 7
        mul $s1,$s1,10
        add $s1,$s1,$t6               # factor=factor*10+left
        j show2
show2:      li $t2,LEFT_LED     # hien thi den LED trai
        add $s0,$zero,$t6       # truyen bien left
        jal displayLED
        nop
        li $t2,RIGHT_LED   # hien thi den LED phai
        add $s0,$zero,$t7 # truyen bien right
        jal displayLED
        nop
case_default2:     j getInt2rt
getInt2rt: lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
 ###################### GETCODE 3 ##########
getInt3: addi $sp,$sp,4
        sw $ra,0($sp)
        li $t1,IN_ADDRESS_HEXA_KEYBOARD
        li $t3,0x84       # Kich hoat gian doan, cho phep bam phim o hang 3
        sb $t3,0($t1)
```

```
        li $t1,OUT_ADDRESS_HEXA_KEYBOARD
        lb $t3,0($t1)     # Nhan byte the hien vi tri cua phim duoc bam trong hang 3
case_8:     li $t5,0x00000014
        bne $t3,$t5,case_9   # case 0x14
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,8   # left = 8
        mul $s1,$s1,10
        add $s1,$s1,$t6     # factor=factor*10+left
        j show3
case_9:     li $t5,0x00000024
        bne $t3,$t5,case_a   # case 0x24
        addi $t7,$t6,0      # left=right
        addi $t6,$zero,9   # left = 9
        mul $s1,$s1,10
        add $s1,$s1,$t6     # factor=factor*10+left
        j show3
case_a:     li $t5,0x44
            bne $t3,$t5,case_b # case 0x44
            addi $s2,$0,1
            j case_default3
case_b:     li $t5,0xffffff84
      bne $t3,$t5,case_default3
      addi $s2,$0,2
   j   case_default3

show3:      li $t2,LEFT_LED     # hien thi den LED trai
        add $s0,$zero,$t6 # truyen bien left
        jal displayLED
        nop
```

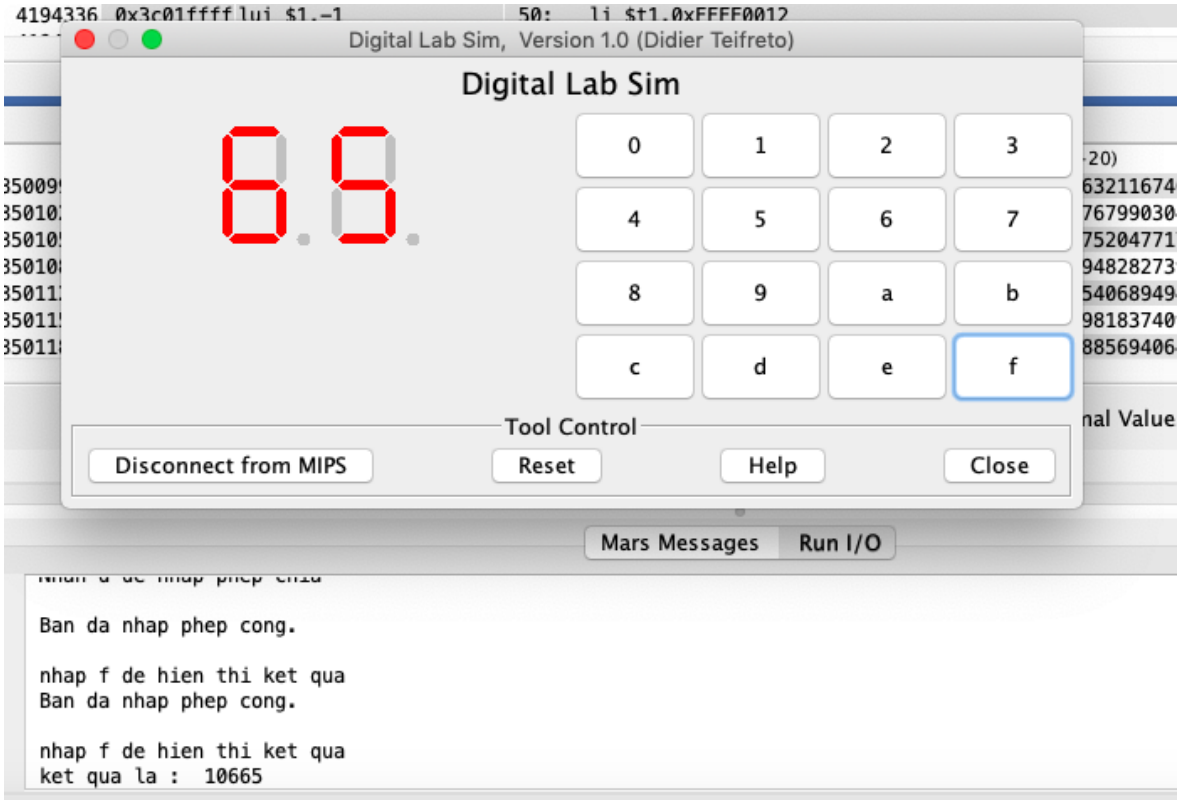
```
        li $t2,RIGHT_LED   # hien thi den LED phai
        add $s0,$zero,$t7 # truyen bien right
        jal displayLED
        nop
case_default3:    j getInt3rt
getInt3rt: lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
#-----------------------------------------getcode 4
 getInt4: addi $sp,$sp,4
        sw $ra,0($sp)
        li $t1,IN_ADDRESS_HEXA_KEYBOARD
        li $t3,0x88        # Kich hoat gian doan, cho phep bam phim o hang 4
        sb $t3,0($t1)
        li $t1,OUT_ADDRESS_HEXA_KEYBOARD
        lb $t3,0($t1)    # Nhan byte the hien vi tri cua phim duoc bam trong hang 4

case_c:    li $t5,0x18
            bne $t3,$t5,case_d # case 0x44
            addi $s2,$0,3
            j case_default3

case_d:    li $t5,0x28
            bne $t3,$t5,case_f
            addi $s2,$0,4
            j   case_default4

case_f:    li $t5,0xffffff88
            bne $t3,$t5,case_default4
```

```
            addi $s3,$0,6
            j case_default4



case_default4:    j getInt4rt
getInt4rt:        lw $ra,0($sp)
                  addi $sp,$sp,-4
                  jr $ra
  # Thu tuc hien thi den LED
  # Tham so truyen vao: $t2 (dia chi cua LEFT_LED hoac RIGHT_LED), $s0 : bien kieu int
  # Den LED $t2 se hien thi so $s0
displayLED: addi $sp,$sp,4
      sw $ra,0($sp)    # save $ra
display:
hienthi_0: bne $s0,0,hienthi_1   # case $s0 = 0
      li $t4,ZERO
      j assign
hienthi_1: bne $s0,1,hienthi_2   # case $s0 = 1
      li $t4,ONE
      j assign
hienthi_2: bne $s0,2,hienthi_3   # case $s0 = 2
      li $t4,TWO
      j assign
hienthi_3: bne $s0,3,hienthi_4   # case $s0 = 3
      li $t4,THREE
      j assign
hienthi_4: bne $s0,4,hienthi_5   # case $s0 = 4
      li $t4,FOUR
      j assign
```

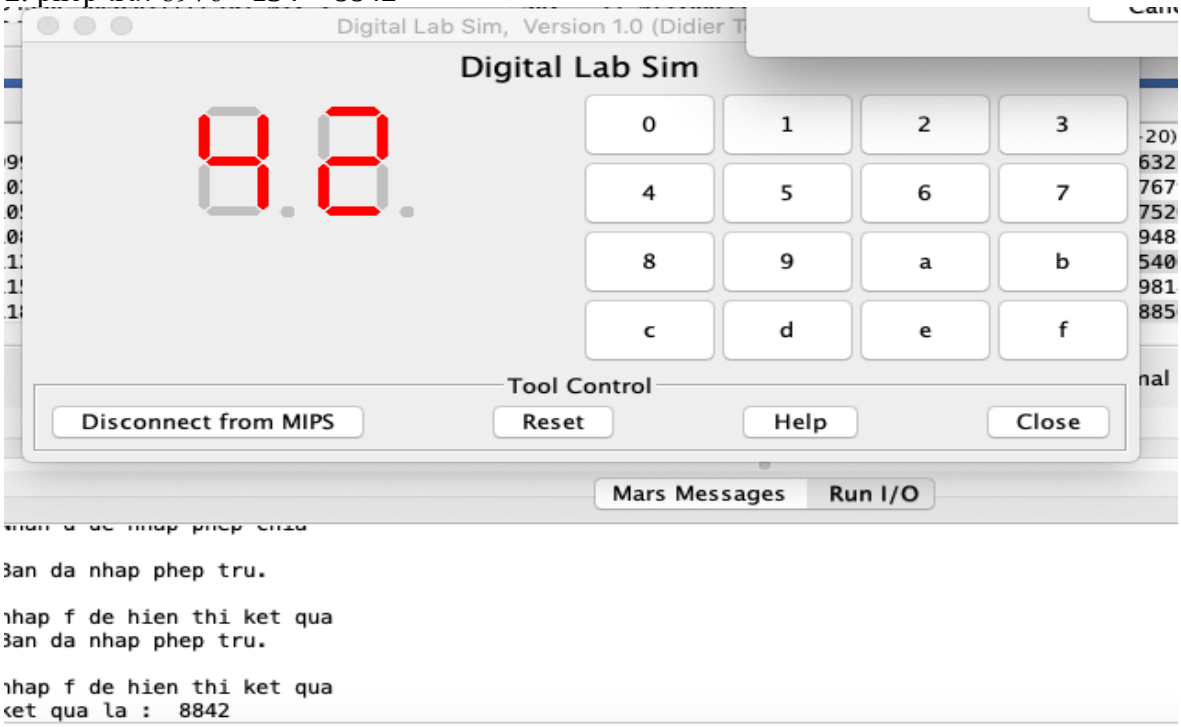
```
hienthi_5: bne $s0,5,hienthi_6   # case $s0 = 5
        li $t4,FIVE
        j assign
hienthi_6: bne $s0,6,hienthi_7   # case $s0 = 6
        li $t4,SIX
        j assign
hienthi_7: bne $s0,7,hienthi_8   # case $s0 = 7
        li $t4,SEVEN
        j assign
hienthi_8: bne $s0,8,hienthi_9   # case $s0 = 8
        li $t4,EIGHT
        j assign
hienthi_9: bne $s0,9,hienthi_df # case $s0 = 9
        li $t4,NINE
        j assign
hienthi_df:   j displayLEDrt
assign:     sb $t4,0($t2)
displayLEDrt: lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
```

IV. Chạy mô phỏng

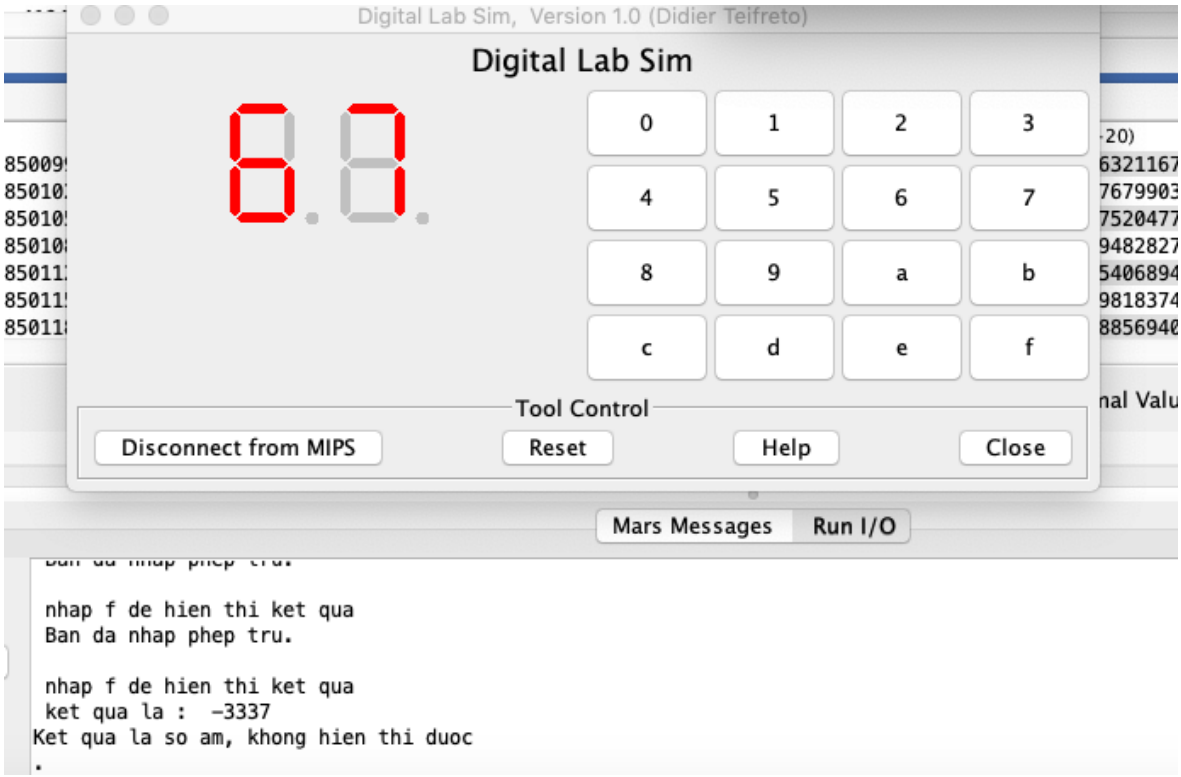
1. Phép cộng: 4455+6210 = 10665, hiển thị số 65 trên máy tính



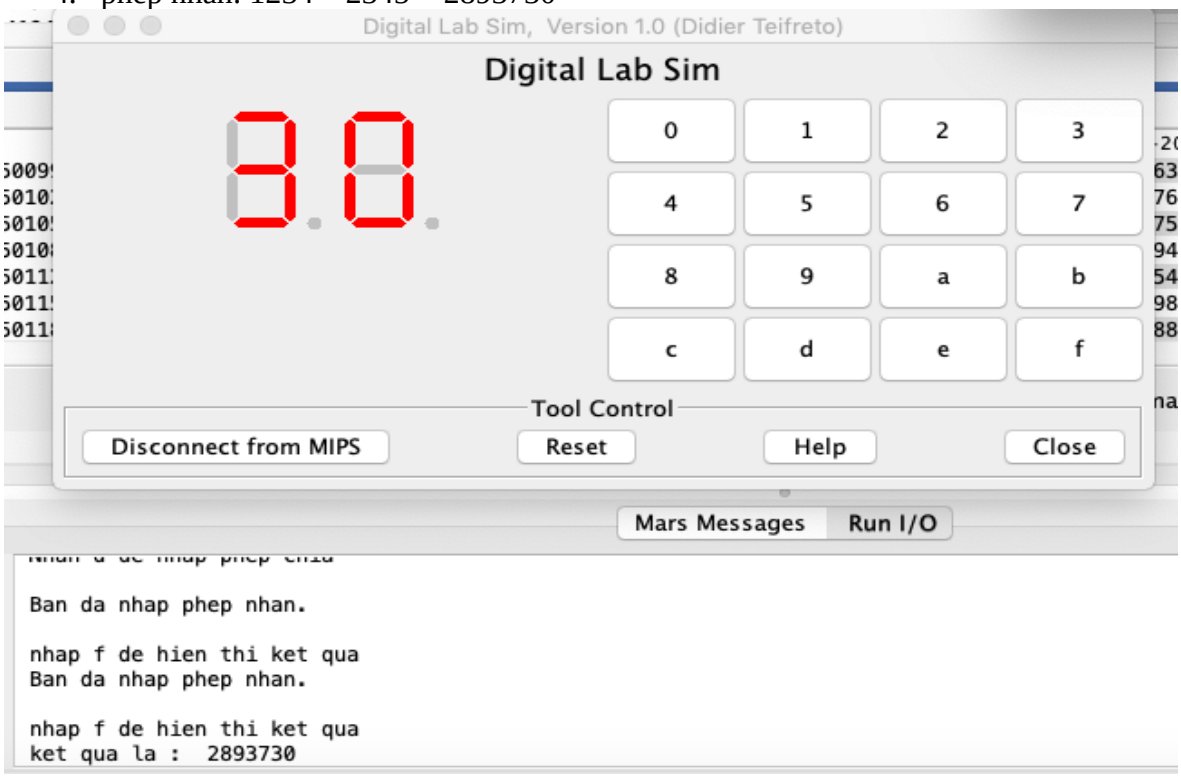
2. phép trừ: 8976 - 134 = 8842



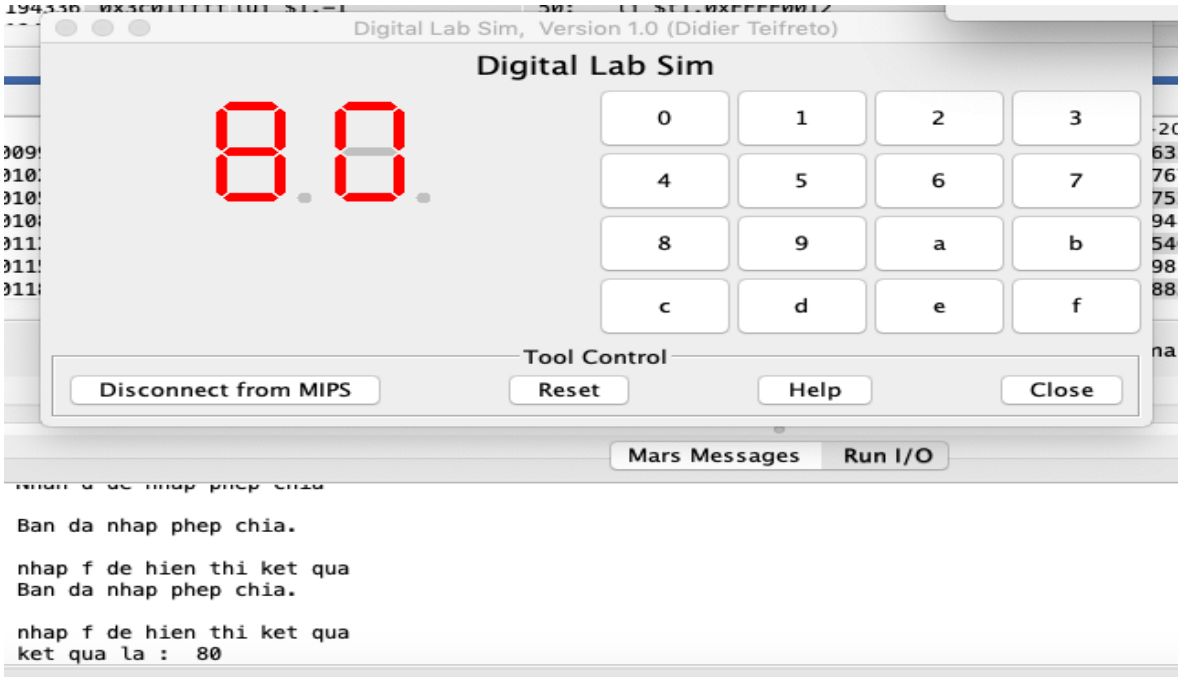
phép trừ số âm: 1230-4567 =-3337



4. phép nhân: 1234 * 2345 = 2893730



5. Phép chia: 9876 / 123 = 80.29



B. Bài 6

I. Đề bài

Viết chương trình sử dụng MIPS để tìm các vị trí xuất hiện của xâu kí tự con trong xâu kí tự lớn. Chương trình sẽ yêu cầu người dùng nhập vào một xâu và một xâu cần tìm kiếm (cả 2 xâu đều là xâu ASCII). Đầu ra của chương trình sẽ là chỉ số nơi mà xâu cần tìm kiếm xuất hiện trong xâu gốc.

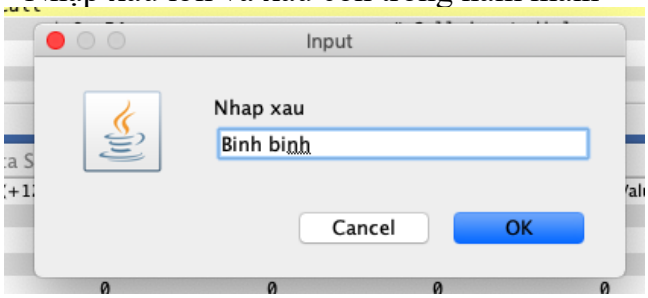
Hàm tìm kiếm cho phép tìm theo các cách

- Không phân biệt chữ hoa, chứ thường
- Phân biệt chữ hoa, thường

II. Phân tích

B1: Nhập liệu

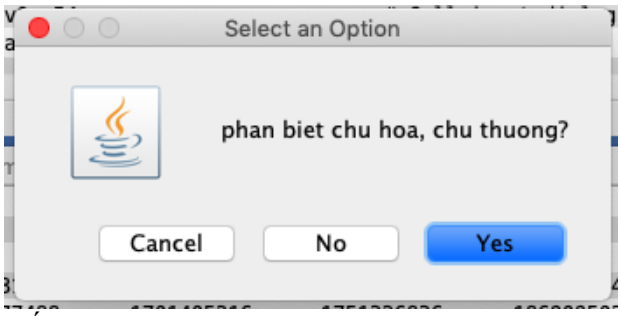
- Nhập xâu lớn và xâu con trong hàm main



B2: Thuật toán thực hiện:

- Chương trình hỏi người dung có muốn phấn biệt chữ hoa chữ thường hay không: YES: có

NO: không



Nếu chọn YES: -> phan_biet
Duyệt xâu lớn từ kí tự đầu tiên:
+Nếu gặp kí tự trùng với kí tự đầu của xâu cần tìm thì bắt đầu so sánh. So sánh đến khi nào đủ xâu con thì lưu giá trị vị trí bắt đầu của xâu cần tìm trong xâu lớn
+Nếu không trùng kí tự đầu của xâu cần tìm thì duyệt đến kí tự tiếp theo
+Cứ như vậy đến khi duyệt hết xau lớn thì in ra kết quả
- Nếu chọn NO: -> khong_phan_biet Đưa toàn bộ xâu lớn và xâu cần tìm thành chư hoa rồi tìm với thuật toán giống như YES

III. Mã nguồn

```
.data
cap_phat_bo_nho1:       .space 600              # cap 600 byte bo nho, chua duoc khoi tao
cap_phat_bo_nho:        .space 1000             # cap 1000 byte bo nho, chua duoc khoi tao

phan_biet:              .word 1                 # gia tri khoi tao la 1
khong_phan_biet:        .word 0                 # gia tri khoi tao la 0
do_dai_chuoi_max:       .word 128
chuoi1:                 .asciiz     "Nhap xau "
chuoi2:                 .asciiz     "nhap xau can tim kiem"
tim_kiem:               .asciiz     "phan biet chu hoa, chu thuong? "

bo_nho_chuoi_1:         .space 129
bo_nho_chuoi_2:         .space 129

ngan_cach:              .ascii ","
bao_loi:                .asciiz     "Khong tim thay chuoi!!!"
ket_qua:                .asciiz     "Chuoi can tim o vi tri: "

.text
#################### main function ####################
main:
        li      $v0, 54                         # Call input dialog
        la      $a0, chuoi1
        la      $a1, bo_nho_chuoi_1             # Read the first string
        lw      $a2, do_dai_chuoi_max               #
        syscall

        li      $v0, 54                       # Call input dialog
        la      $a0, chuoi2
        la      $a1, bo_nho_chuoi_2             # Read the second string
        lw      $a2, do_dai_chuoi_max               #
        syscall

        li      $v0, 50                       # Call confirm dialog
        la      $a0, tim_kiem                   # User will select mode of searching here
        syscall

        beq     $a0, 0, phan_biet_mode      # chọn CÓ, đặt tham số chế độ của chức năng tìm kiếm thành phan_biet
        j       search_insensitive_mode             # Nếu không, đặt tham số chế độ chức năng tìm kiếm thành
khong_phan_biet

phan_biet_mode:
        lw      $a2, phan_biet    # Set the search mode
        j       call_search_function            # Then call search function
```

```
search_insensitive_mode:
	lw	$a2, khong_phan_biet	# Set the search mode
	j	call_search_function		# Then call search function

call_search_function:
	la	$a0, bo_nho_chuoi_1	# Call search function
	la	$a1, bo_nho_chuoi_2	# a0 = str1, a1 = str2, a2 = mode
	la	$a3, cap_phat_bo_nho1
	jal	search			# search(a0, a1, a2, a3)
	beq	$v0, 0, search_not_found	# search(a0, a1, a2, a3)

	la	$a0, cap_phat_bo_nho1		#
	addi	$a1, $v0, 0		# Create a string that join all the result
	la	$a2, cap_phat_bo_nho	#
	lb	$a3, ngan_cach		#
	jal	joinint			#

	la	$a0, ket_qua		#
	la	$a1, cap_phat_bo_nho	# Call message dialog, print the joined result
	li	$v0, 59			#
	syscall
	j	exit

search_not_found:
	la	$a0, bao_loi		#
	li	$a1, 1			# Call message dialog, string not found
	li	$v0, 55			#
	syscall
	j	exit

exit:
	li	$v0, 10
	syscall

#################### search function ####################
#
#	prototype: search($a0, $a1, $a2, a3)
#	$a0: string
#	$a1: string
#	$a2: mode ( 1 if case sensitive, otherwise, case insensitive)
#	$a3: int array, store all the position $a1 appear in $a0
#
#	tim_kiem string $a1 in string $a0
#
search:
	subi	$sp, $sp, 4
	sw	$ra, 4($sp)
	addi	$s0, $a0, 0		# $s0 = $a0
	addi	$s1, $a1, 0		# $s1 = $a1
	addi	$s3, $a3, 0		# $s3 = $a3 = result array
	li	$s4, 0			# $s4 = len($s3)

	addi	$a0, $s0, 0		#
	jal	strlen			# $s2 = strlen($a0)
	addi	$s2, $v0, 0		#

	addi	$a0, $s1, 0		#
	jal	strlen			# $s2 = strlen($a0) - strlen($a1)
	sub	$s2, $s2, $v0		#

	li	$s5, 0			# $s5 = current position in $s0

search_loop:
	bgt	$s5, $s2, search_return	#
	add	$a0, $s0, $s5		#
	addi	$a1, $s1, 0		# call startwith($a0+i, $a1, mode=$s2)
```

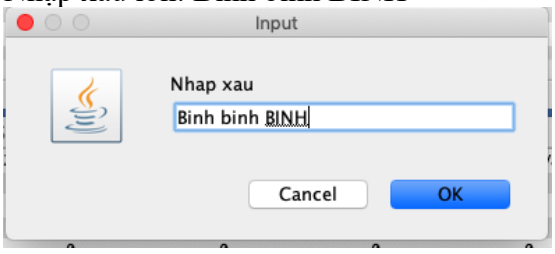
```
        jal     startswith              #
        beq     $v0, 0, search_prepare_new_loop  # Not matched :((

        mul     $t1, $s4, 4             # We found a match here
        add     $t0, $s3, $t1                   # Add current offset to the
        sw      $s5, ($t0)              #       integer array $s3
        addi    $s4, $s4, 1             # and also increase $s4, we will return it later.

search_prepare_new_loop:
        addi    $s5, $s5, 1             # increase $s5
        j       search_loop                     # and start the new loop

search_return:
        addi    $v0, $s4, 0             # return $s4, which is length of integer array $a3, which store matched offsets
        lw      $ra, 4($sp)             # restore return address
        addi    $sp, $sp, 4             #       and the stack
        jr      $ra


#################### compare function ####################
#       prototype: compare($a0, $a1, $a2)
#       $a0: char
#       $a1: char
#       $a2: mode ( 1 if case sensitive, otherwise, case insensitive)
#       return 1 if a0 = a1, otherwise, return 0
#
compare:
        beq     $a2, 1, compare_return          # Case sensitive, jump to compare_return
        beq     $a0, $a1, compare_return        # $a0 = $a1 :))))
        bge     $a0, 0x61, compare_ge_a
        bge     $a0, 0x41, compare_ge_A

compare_ge_a:
        bgt     $a0, 0x7A, compare_return       # if 'a' <= $a0 <= 'z', then change $a0 to uppercase
        subi    $a0, $a0, 32                    # otherwise, $a0 remain lowercase
        j       compare_return                  #

compare_ge_A:
        bgt     $a0, 0x5A, compare_return       # if 'A' <= $a0 <= 'Z', then change $a0 to lowercase
        addi    $a0, $a0, 32                    # otherwise, $a0 remain uppercase
        j       compare_return                  #

compare_return:                                 # Compare a0 and a1 after modifying
        seq     $v0, $a0, $a1                   #
        jr      $ra                             # set $pc to $ra

#################### strlen function ####################
#
#       prototype: strlen($a0)
#       $a0: string
#
#       return: $a0's length
#
strlen:
        li      $v0, 0
strlen_loop:
        add     $t0, $a0, $v0                   #
        lb      $t1, ($t0)                      # Very simple
        beq     $t1, 0, strlen_return                   # No comment :)))
        beq     $t1, 0x0A, strlen_return        #
        addi    $v0, $v0, 1                     #
        j       strlen_loop                     #
strlen_return:
        jr      $ra
#################### startswith function ####################
#
#       prototype: startswith($a0, $a1, $a2)
```

```
#       $a0: string
#       $a1: string
#       $a2: mode ( 1 if case sensitive, otherwise, case insensitive)
#
#       return: 1 if $a0 is started with $a1
#

startswith:
        subi    $sp, $sp, 12                    #
        sw      $ra, 12($sp)                    # Save return address
        sw      $s0, 8($sp)             # We also save $s0, $s1 to stack here
        sw      $s1, 4($sp)             # because $s1, $s2 will store $a1 and $a2, respectively
        addi    $s0, $a0, 0             #
        addi    $s1, $a1, 0             #

startswith_loop:
        lb      $a0, ($s0)              # load the first character of $s0, $s1
        lb      $a1, ($s1)              # to $a0, $a1, respectively
        beq     $a1, 0, startswith_return       #
        beq     $a1, 0x0a, startswith_return    #
        jal     compare                         # Then pass it to compare function
        beq     $v0, 0, startswith_return       #
        addi    $s0, $s0, 1             # increse offset of $s0, $s1
        addi    $s1, $s1, 1             #
        j       startswith_loop                 #

startswith_return:
        lw      $s0, 8($sp)             #
        lw      $s1, 4($sp)             # restore old $s0, $s1
        lw      $ra, 12($sp)                    # and $ra as well
        addi    $sp, $sp, 12                    #
        jr      $ra

#################### atoi function ###################
#
#       prototype: atoi($a0, $a1)
#       $a0: int
#       $a1: space to store the result
#       return: length of $a1
atoi:
        li      $v0, 1
        li      $t0, 1                  # Clone $a0 to $t1 and $t2
        addi    $t1, $a0, 0             # We use $t1 to get length of the number
        addi    $t2, $a0, 0             # and $t2 to save each digit to array $a1
        li      $t5, 10
atoi_loop1:
        div     $t1, $t1, 10            # Get length of the number
        beq     $t1, 0, atoi_loop2_prepare      # by deviding until it's zero
        addi    $v0, $v0, 1             # Store the length in $v0
        j       atoi_loop1              #

atoi_loop2_prepare:
        add     $t3, $a1, $v0                   #
        sb      $0, ($t3)               #
atoi_loop2:
        div     $t2, $t5                # For each deviding of $t2
        mfhi    $t4                     # Get the remainder, transform it to ascii
        addi    $t4, $t4, 0x30                  #
        mflo    $t2                     # Get the quotient,
        subi    $t3, $t3, 1             #
        sb      $t4, ($t3)              #
        beq     $t2, 0, atoi_return             # continue the loop if current number is greater than 0
        j       atoi_loop2              #
atoi_return:
        jr      $ra

#################### joinint function ###################
```

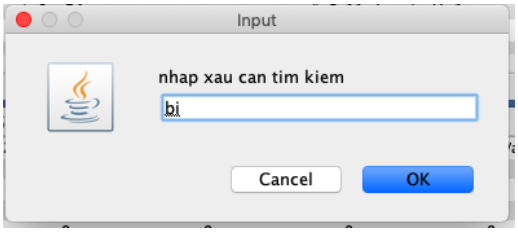
```
#
#       prototype: jointint($a0, $a1, $a2, $a3)
#       $a0: array of integer
#       $a1: length($a0)
#       $a2: space, which will store joining result
#       $a3: char, the separator
#
#       return: length of $a1 after joining
joinint:
        subi    $sp, $sp, 28                    # So many thing to save
        sw      $ra, 28($sp)                    # We save $ra, and the $s registers
        sw      $s0, 24($sp)                    #       because in later, we 'll call a sub function (atoi)
        sw      $s1, 20($sp)                    #
        sw      $s2, 16($sp)                    #
        sw      $s3, 12($sp)                    #
        sw      $s4, 8($sp)             #
        sw      $s5, 4($sp)             #
        addi    $s0, $a0, 0             # clone $a registers to $s registers
        addi    $s1, $a1, 0             # ...
        addi    $s2, $a2, 0             # ...
        addi    $s3, $a3, 0             # ...
        li      $s4, 0                          # $s4 = length of the result string
        li      $s5, 0                          # $s5 = current offset of number in array $a2

joinint_loop:
        bge     $s5, $s1, joinint_return        #
        lw      $a0, ($s0)              #
        addi    $a1, $s2, 0             #
        jal     atoi                    #
        add     $s4, $v0, $s4                   # Transform number to string, then add to $s2
        add     $s2, $s2, $v0                   #
        sb      $s3, ($s2)              # Add a separator character as well
        add     $s2, $s2, 1             # Increase the length
        add     $s4, $s4, 1             #
                                        #
        addi    $s0, $s0, 4             #
        addi    $s5, $s5, 1             # Increase the current offset
        j       joinint_loop                    # Continue the loop

joinint_return:
        sb      $0, -1($s2)             # Add the NULL character to the end of joined string
        lw      $ra, 28($sp)                    # And   finally, restore all saved register
        lw      $s0, 24($sp)                    # $s registers and $ra register
        lw      $s1, 20($sp)                    # ...
        lw      $s2, 16($sp)                    # ...
        lw      $s3, 12($sp)                    # ...
        lw      $s4, 8($sp)             # ...
        lw      $s5, 4($sp)             # ...
        add     $sp, $sp, 28                    # restore the stack
        addi    $v0, $s4, 0             # return $s4
        jr      $ra
```

IV.Chạy mô phỏng

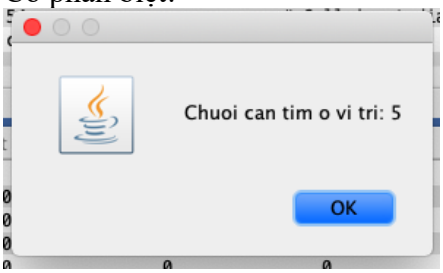
Nhập xâu lớn: Binh binh BINH



Nhập xâu con: bi



Có phân biệt:



Không phân biệt: