

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



BÀI TẬP LỚN MÔN HỌC
THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: Hoàng Văn Hiệp
Sinh viên thực hiện : Nguyễn Công Bình
MSSV : 20225695

Hà Nội, tháng 6 năm 2024

Chương 1: Phân tích

- Cần tạo các hàm riêng cộng, trừ, nhân, chia, bình phương để thực hiện các phép toán.
- Cần nghĩ ra cơ chế để lưu chỉ 2 số cuối dựa vào input đầu vào của người nhập và lưu giá trị xuất hiện trên đèn led để có thể hiển thị.
- Cần xử lý được các lỗi có thể xảy ra trong quá trình thực hiện (tràn số, chia cho 0 hoặc trừ ra số âm).
- Cần xử lý được khi mà người dùng muốn nhập các phép tính liên tiếp.
- Cần in ra từng bước thực hiện phép toán để có thể theo dõi kết quả.
- Cần tính được cả phần dư của các phép chia dư.
- Cần nghĩ ra cơ chế để xử lý khi người dùng nhập toán tử và toán hạng.

Chương 2: Thuật toán

2.1. Thuật toán lưu trữ ký tự nhập vào

- Kiểm tra hàng phím: Các hàm “checkrow1”, “checkrow2”, “checkrow3”, “checkrow4” sẽ kích hoạt ngắt để kiểm tra các hàng phím xem có phím nào được nhấn hay không.
- Hàm “getvalue” đọc giá trị của phím được nhấn từ địa chỉ OUT_ADDRESS_HEXKEYBOARD và lưu vào thanh ghi \$t3.
- Các hàm “convertrow1”, “convertrow2”, “convertrow3”, “convertrow4” sẽ chuyển đổi giá trị lưu trong \$t3 thành các giá trị tương ứng để hiển thị lên LED hoặc xử lý tiếp.

```
75 checkrow1:
76     addi $sp,$sp,4
77     sw $ra,0($sp)
78     li $t3,0x81          # Kích hoạt interrupt, cho phép bấm phím ở hàng 1
79     sb $t3,0($t1)
80     jal getValue         # get vi tri ( hang va cot ) của phím được nhập nếu có
81     lw $ra,0($sp)
82     addi $sp,$sp,-4
83     jr $ra
84 checkrow2:
85     addi $sp,$sp,4
86     sw $ra,0($sp)
87     li $t3,0x82          # Kích hoạt interrupt, cho phép bấm phím ở hàng 2
88     sb $t3,0($t1)
89     jal getValue
90     lw $ra,0($sp)
91     addi $sp,$sp,-4
92     jr $ra
93 checkrow3:
94     addi $sp,$sp,4
95     sw $ra,0($sp)
96     li $t3,0x84          # Kích hoạt interrupt, cho phép bấm phím ở hàng 3
97     sb $t3,0($t1)
98     jal getValue
99     lw $ra,0($sp)
100    addi $sp,$sp,-4
101    jr $ra
102 checkrow4:
103    addi $sp,$sp,4
104    sw $ra,0($sp)
105    li $t3,0x88          # Kích hoạt interrupt, cho phép bấm phím ở hàng 4
106    sb $t3,0($t1)
107    jal getValue
108    lw $ra,0($sp)
109    addi $sp,$sp,-4
110    jr $ra
```

```

110     jr $ra
111 getvalue:
112     addi $sp,$sp,4
113     sw $ra,0($sp)
114     li $t2,OUT_ADDRESS_HEX_KEYBOARD #địa chỉ chứa vị trí phím được nhập
115     lb $t3,0($t2)                   #load vị trí phím được nhập
116     lw $ra,0($sp)
117     addi $sp,$sp,-4
118     jr $ra
119 convertrow1:                        #hàm convert từ vị trí sang bit để chuyển đèn led
120     beq $t3,0x11,case_zero          #0x11 -->phím ở hàng 1 cột 1--> 0
121     beq $t3,0x21,case_one
122     beq $t3,0x41,case_two
123     beq $t3,0xfffff81,case_three
124 convertrow2:
125     beq $t3,0x12,case_four
126     beq $t3,0x22,case_five
127     beq $t3,0x42,case_six
128     beq $t3,0xfffff82,case_seven
129 convertrow3:
130     beq $t3,0x14,case_eight
131     beq $t3,0x24,case_nine
132     beq $t3,0x44,case_a
133     beq $t3,0xfffff84,case_
134 convertrow4:
135     beq $t3,0x18,case_c
136     beq $t3,0x28,case_d
137     beq $t3,0x48,case_e
138     beq $t3,0xfffff88,case_f

```

2.2.Hàm tính toán gồm cộng, trừ, nhân, chia.

- Sau khi xác định được các số mà người dùng đã chọn qua các vòng lặp ban đầu bằng cách sử dụng thanh ghi \$s4 và \$s5 để lưu lại giá trị mà người dùng đã nhập, và toán tử được lưu lại trong \$s3. Sau khi người dùng nhấn phím “F” (dấu bằng) , chương trình sẽ tính giá trị của hai giá trị trên và lưu vào thanh ghi \$s6.

Hàm cộng

- Thực hiện phép cộng dựa vào 2 thanh ghi lưu giá trị của 2 toán hạng và lưu kết quả vào trong thanh ghi \$s6, sau đó in ra biểu thức mà người dùng muốn tính, ví dụ: $10 + 2 = 12$.
- Sau khi ra kết quả sẽ chia cho 10 để chia giá trị kết quả thành phần nguyên và phần dư, sau đó chuyển đổi và hiển thị từng phần lên hai đèn LED 7 đoạn.

```

506 cong1:
507     add $s6,$s5,$s4
508     j incong1
509     nop                                # s6=s5+s4
510
511 incong1:
512     li $v0, 1
513     move $a0, $s4
514     syscall
515
516     li $v0, 11
517     li $a0, '+'
518     syscall
519
520     li $v0, 1
521     move $a0, $s5
522     syscall
523
524     li $v0, 11
525     li $a0, '='
526     syscall
527
528     li $v0, 1
529     move $a0, $s6
530     syscall
531
532     li $v0, 11
533     li $a0, '\n'
534     syscall
535     li $s7, 100
536     div $s6, $s7
537     mfhi $s6                        # chỉ lấy 2 chữ số cuối của kết quả để in ra led
538     j splitnumber1                # chuyển đến hàm chia kết quả thành 2 chữ số để hiển thị lên từng led
539     nop

```

Hàm trừ

- Thực hiện phép trừ dựa vào 2 thanh ghi lưu giá trị của 2 toán hạng và lưu kết quả vào trong thanh ghi \$s6, sau đó in ra biểu thức mà người dùng muốn tính, ví dụ: $5 - 2 = 3$.
- Sau khi ra kết quả sẽ chia cho 10 để chia giá trị kết quả thành phần nguyên và phần dư, sau đó chuyển đổi và hiển thị từng phần lên hai đèn LED 7 đoạn.

```

541 trul:
542     sub $s6,$s4,$s5                # s6=s4-s5
543     blt $s6, 0, truaml
544     j intrul
545     nop
546 intrul:
547     li $v0, 1
548     move $a0, $s4
549     syscall
550
551     li $v0, 11
552     li $a0, '-'
553     syscall
554
555     li $v0, 1
556     move $a0, $s5
557     syscall
558
559     li $v0, 11
560     li $a0, '='
561     syscall
562
563     li $v0, 1
564     move $a0, $s6
565     syscall
566
567     li $v0, 11
568     li $a0, '\n'
569     syscall
570     j splitnumber1                # chuyển đến hàm chia kết quả thành 2 chữ số để hiển thị lên từng led
571     nop

```

Hàm nhân

- Thực hiện phép nhân dựa vào 2 thanh ghi lưu giá trị của 2 toán hạng và lưu kết quả vào trong thanh ghi \$s6, sau đó in ra biểu thức mà người dùng muốn tính, ví dụ: $5 - 2 = 3$.
- Sau khi ra kết quả sẽ chia cho 10 để chia giá trị kết quả thành phần nguyên và phần dư, sau đó chuyển đổi và hiển thị từng phần lên hai đèn LED 7 đoạn.

```
572 nhan1:
573     mul $s6,$s4,$s5      # s6=s4*s5
574     j innhan1
575     nop
576 innhan1:
577     li $v0, 1
578     move $a0, $s4
579     syscall
580
581     li $v0, 11
582     li $a0, '*'
583     syscall
584
585     li $v0, 1
586     move $a0, $s5
587     syscall
588
589     li $v0, 11
590     li $a0, '='
591     syscall
592
593     li $v0, 1
594     move $a0, $s6
595     syscall
596
597     li $v0, 11
598     li $a0, '\n'
599     syscall
600     li $s7, 100
601     div $s6,$s7
602     mfhi $s6              # chỉ lấy 2 chu số sau cùng của kết quả in ra
603     j splitnumber1       # chuyển đến hàm chia kết quả thành 2 chu số để hiển thị lên từng led
604     nop
```

Hàm chia

- Thực hiện phép chia dựa vào 2 thanh ghi lưu giá trị của 2 toán hạng và lưu kết quả vào trong thanh ghi \$s6, số dư vào trong thanh ghi \$s7, sau đó in ra biểu thức mà người dùng muốn tính kèm số dư, ví dụ: $3/2 = 1$ dư 1.
- Sau khi ra kết quả sẽ chia cho 10 để chia giá trị kết quả thành phần nguyên và phần dư, sau đó chuyển đổi và hiển thị từng phần lên hai đèn LED 7 đoạn.

```

605 chial:
606     beq $s5,0,chia01
607     div $s4,$s5      # s6=s4/s5
608     mflo $s6
609     mfhi $s7
610     j inchial
611     nop
612 inchial:
613     li $v0, 1
614     move $a0, $s4
615     syscall
616
617     li $v0, 11
618     li $a0, '/'
619     syscall
620
621     li $v0, 1
622     move $a0, $s5
623     syscall
624
625     li $v0, 11
626     li $a0, '='
627     syscall
628
629     li $v0, 1
630     move $a0, $s6
631     syscall
632
633     li $v0, 11
634     li $a0, ' '
635     syscall
636
637     li $v0, 11
638     li $a0, 'x'
639     syscall
640
641     li $v0, 11
642     li $a0, '-'
643     syscall
644
645     li $v0, 1
646     move $a0, $s7
647     syscall
648
649     li $v0, 11
650     li $a0, '\n'
651     syscall
652     j splitnumber1    # chuyển đến hàm chia kết quả thành 2 chữ số để hiển thị lên từng led
653     nop

```

2.3.Hàm chia phép tính thành hai chữ số cuối cùng

- **Chuẩn bị chia số:**

- `li $t8, 10`: Tải giá trị 10 vào thanh ghi \$t8 để sử dụng cho phép chia.
- `div $s6, $t8`: Chia giá trị trong \$s6 cho 10. Kết quả chia được lưu trong thanh ghi đặc biệt LO (quá số) và HI (số dư).

- **Lấy phần nguyên của phép chia:**

- `mflo $t7`: Di chuyển giá trị quá số từ thanh ghi LO vào \$t7.
- `jal convert`: Gọi hàm convert để chuyển đổi giá trị số trong \$t7 thành mã 7 đoạn để hiển thị trên LED.

- **Hiển thị phần nguyên trên LED trái:**

- `sb $t4, 0($t0)`: Lưu mã 7 đoạn (tương ứng với chữ số phần nguyên) vào đèn LED trái.
- `addi $sp, $sp, -4`: Giảm con trỏ stack xuống 4 byte.
- `sb $t7, 0($sp)`: Lưu chữ số phần nguyên vào stack.
- `addi $sp, $sp, -4`: Giảm con trỏ stack xuống 4 byte.
- `sb $t4, 0($sp)`: Lưu mã 7 đoạn vào stack.

- **Lấy phần dư của phép chia:**

- mfhi \$t7: Di chuyển giá trị số dư từ thanh ghi HI vào \$t7.
- jal convert: Gọi hàm convert để chuyển đổi giá trị số trong \$t7 thành mã 7 đoạn để hiển thị trên LED.

- **Hiển thị phần dư trên LED phải:**

- sb \$t4, 0(\$t5): Lưu mã 7 đoạn (tương ứng với chữ số phần dư) vào đèn LED phải.
- addi \$sp, \$sp, -4: Giảm con trỏ stack xuống 4 byte.
- sb \$t7, 0(\$sp): Lưu chữ số phần dư vào stack.
- addi \$sp, \$sp, -4: Giảm con trỏ stack xuống 4 byte.
- sb \$t4, 0(\$sp): Lưu mã 7 đoạn vào stack.

- **Kết thúc và chuyển đến hàm resetled**

- j resetled: Chuyển đến hàm resetled để reset lại LED cho lần nhập tiếp theo.

```

388 splitnumber:    #ham chia ket qua thanh 2 chu so de hien thi len tung led
389     li $t8,10
390     div $s6,$t8    #s6/10
391     mflo $t7        #t7 = result
392     jal convert     #chuyen den ham chuyen t7 thanh bit hien thi len led
393
394     sb $t4,0($t0)   # hien thi len led trai
395     add $sp,$sp,4
396     sb $t7,0($sp)    #day gia tri bit nay vao stack
397     add $sp,$sp,4
398     sb $t4,0($sp)    #day bit nay vao stack
399     add $s2,$t7,$zero #s1 = gia tri bit led phai
400
401
402     mfhi $t7        #t7= remainder
403     jal convert     #convert t7 thanh bit hien thi len led
404     sb $t4,0($t5)   #hien thi len led phai
405     add $sp,$sp,4
406     sb $t7,0($sp)    #day gia tri bit nay vao stack
407     add $sp,$sp,4
408     sb $t4,0($sp)    #day bit nay vao stack
409     add $s1,$t7,$zero #s1 = gia tri bit led phai
410     j resetled      #ham reset lai led

```


Chương 3: Xử lý lỗi

3.1 Lỗi với số âm

- Còi chương trình không thể tính được số âm nên khi thực hiện phép trừ ra số âm chương trình sẽ nhảy đến nhãn 'truam' và reset lại máy tính.

3.2 Lỗi chia cho số 0

- Chương trình sẽ kiểm tra xem toán hạng thứ 2 mà người dùng nhập vào có phải số 0 không trong hàm chia của chương trình. Nếu là số 0 sẽ nhảy đến nhãn xử lý lỗi tương ứng là 'chia0' và reset lại máy tính.

Chương 4: Mã nguồn

```
.data
    zero: .byte 0x3f
    one: .byte 0x6
    two: .byte 0x5b
    three: .byte 0x4f
    four: .byte 0x66
    five: .byte 0x6d
    six: .byte 0x7d
    seven: .byte 0x7
    eight: .byte 0x7f
    nine: .byte 0x6f
    mess1: .asciiz "khong the tinh duoc so am \n"
    mess2: .asciiz "khong the chia cho so 0 \n"

.eqv IN_ADDRESS_HEXА_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXА_KEYBOARD 0xFFFF0014
.eqv SEVENSEG_LEFT 0xFFFF0011      # Dia chi cua den led 7 doan trai.
.eqv SEVENSEG_RIGHT 0xFFFF0010    # Dia chi cua den led 7 doan phai
.text
main:
    li $t0,SEVENSEG_LEFT      # $t0: Bien gia tri so cua den LED trai
    li $t5,SEVENSEG_RIGHT     # $t1: Bien gia tri so cua den LED phai
    li $s0,0                  # bien kiem tra loai bien nhap vao, 0: so, 1 :toan tu, 2:
terminate key
    li $s1,0                  # so dang hien thi o led phai
    li $s2,0                  # so dang hien thi o led trai
    li $s3,0                  # bien kiem tra loai toan tu, 1:cong, 2:tru, 3:nhân, 4:chia
    li $s4,0                  # so thu nhât
    li $s5,0                  # so thu 2
    li $s6,0                  # ket qua 2 so, cong ,tru, nhân, chia
    li $t1, IN_ADDRESS_HEXА_KEYBOARD #bien dieu khien hang keyboard va
enable keyboard interrupt
    li $t2, OUT_ADDRESS_HEXА_KEYBOARD #bien chua vi tri key nhap vao the hang
va cot
    li $t3, 0x80              # bit dung enable keyboard interrupt va enable kiem tra
tung hang keyboard
    sb $t3, 0($t1)
    li $t7,0                  #gia tri cua so hien tren led
    li $t4,0                  #byte hien thi len led ,zero->nine
storefirstvalue:
```

```

        li $t7,0                #gia tri bit can hien thi ban dau :0
        addi $sp,$sp,4          #day vao stack
        sb $t7,0($sp)
        lb $t4,zero             #bit dau tien can hien thi :0
        addi $sp,$sp,4          #day vao stack
        sb $t4,0($sp)
loop1: #loop de doi nhap phim tu digital lab sim
        beq $s0,2,endloop1      #neu phim terminate(phim e) duoc bam ,thoat loop
        nop
        nop
        nop
        nop
        b loop1
        nop
        nop
        nop
        b loop1
        nop
        nop
        b loop1

endloop1:
end_main:
        li $v0,10
        syscall
        .ktext 0x80000180
process:
        jal checkrow1           #check hang 1 xem co phim nao duoc nhap ko
        bnez $t3,convertrow1    #t3 != 0 --> co phim duoc nhap, convert phim do thanh
bit hien ra led
        nop
        jal checkrow2
        bnez $t3,convertrow2
        nop
        jal checkrow3
        bnez $t3,convertrow3
        nop
        jal checkrow4
        bnez $t3,convertrow4
checkrow1:
        addi $sp,$sp,4
        sw $ra,0($sp)
        li $t3,0x81             # Kich hoat interrupt, cho phep bam phim o hang 1
        sb $t3,0($t1)
        jal getvalue             # get vi tri ( hang va cot ) cua phim duoc nhap neu co
        lw $ra,0($sp)
        addi $sp,$sp,-4
        jr $ra
checkrow2:
        addi $sp,$sp,4

```

```

    sw $ra,0($sp)
    li $t3,0x82    # Kich hoat interrupt, cho phep bam phim o hang 2
    sb $t3,0($t1)
    jal getvalue
    lw $ra,0($sp)
    addi $sp,$sp,-4
    jr $ra
checkrow3:
    addi $sp,$sp,4
    sw $ra,0($sp)
    li $t3,0x84    # Kich hoat interrupt, cho phep bam phim o hang 3
    sb $t3,0($t1)
    jal getvalue
    lw $ra,0($sp)
    addi $sp,$sp,-4
    jr $ra
checkrow4:
    addi $sp,$sp,4
    sw $ra,0($sp)
    li $t3,0x88    # Kich hoat interrupt, cho phep bam phim o hang 4
    sb $t3,0($t1)
    jal getvalue
    lw $ra,0($sp)
    addi $sp,$sp,-4
    jr $ra
getvalue:
    addi $sp,$sp,4
    sw $ra,0($sp)
    li $t2,OUT_ADDRESS_HEX_KEYBOARD #dia chi chua vi tri phim duoc nhap
    lb $t3,0($t2)                #load vi tri phim duoc nhap
    lw $ra,0($sp)
    addi $sp,$sp,-4
    jr $ra
convertrow1:                #ham convert tu vi tri sang bit de chuyen den led
    beq $t3,0x11,case_zero    #0x11 -->phim o hang 1 cot 1--> 0
    beq $t3,0x21,case_one
    beq $t3,0x41,case_two
    beq $t3,0xfffff81,case_three
case_zero:
    lb $t4,zero                #t4=zero (tuc = 0x3f, tong cac bit thanh ghi de tao thanh so 0 tren
led)
    li $t7,0                    #t7= gia tri cua t4
    j done
case_one:
    lb $t4,one
    li $t7,1
    j done
case_two:
    lb $t4,two
    li $t7,2

```

```

        j done
case_three:
    lb $t4,three
    li $t7,3
    j done
convertrow2:
    beq $t3,0x12,case_four
    beq $t3,0x22,case_five
    beq $t3,0x42,case_six
    beq $t3,0xfffff82,case_seven
case_four:
    lb $t4,four
    li $t7,4
    j done
case_five:
    lb $t4,five
    li $t7,5
    j done
case_six:
    lb $t4,six
    li $t7,6
    j done
case_seven:
    lb $t4,seven
    li $t7,7
    j done
convertrow3:
    beq $t3,0x14,case_eight
    beq $t3,0x24,case_nine
    beq $t3,0x44,case_a
    beq $t3,0xfffff84,case_b
case_eight:
    lb $t4,eight
    li $t7,8
    j done
case_nine:
    lb $t4,nine
    li $t7,9
    j done
case_a:# cong
    addi $a3,$zero,1
    addi $s0,$s0,1      #bien check phim nhap vao chuyen thanh 1(chung to nhap vao 1
toan tu
    bne $s3,0,setnextoperator
    addi $s3,$zero,1      #bien check loai toan tu chuyen thanh 1(tuc phep cong)

    j setfirstnumber      #chuyen den ham chuyen 2 byte dang hien tren 2 led thanh so de
tinh toan
case_b: # tru
    addi $a3,$zero,2

```

```

        addi $s0,$s0,1
        bne $s3,0,setnextoperator
        addi $s3,$zero,2
        j setfirstnumber
convertrow4:
        beq $t3,0x18,case_c
        beq $t3,0x28,case_d
        beq $t3,0x48,case_e
        beq $t3,0xfffff88,case_f
case_c: # nhan
        addi $a3,$zero,3
        addi $s0,$s0,1
        bne $s3,0,setnextoperator
        addi $s3,$zero,3
        j setfirstnumber
case_d: #truong hop phim chia
        addi $a3,$zero,4
        addi $s0,$s0,1
        bne $s3,0,setnextoperator
        addi $s3,$zero,4
        j setfirstnumber

case_e: #truong hop terminate key
        addi $s0,$s0,2
        j finish

setfirstnumber:                # so dau tien hien thi tren led trong 2 so
        mul $s4,$s2,10          # s4=s2*10+s1
        add $s4,$s4,$s1
        j done

case_f: #truong hop bam =
setsecondnumber: #ham tinh so thu 2 dang hien thi tren led trong 2 so
        mul $s5,$s2,10          # s5=s2*10+s1
        add $s5,$s5,$s1
        beq $s3,1,cong          # chia trg hop nhan chia cong tru
        beq $s3,2,tru
        beq $s3,3,nhan
        beq $s3,4,chia
cong:
        add $s6,$s5,$s4
        li $s3,0
        j incong
        nop                    # s6=s5+s4

incong:
        li $v0, 1
        move $a0, $s4
        syscall

```

```

    li $v0, 11
    li $a0, '+'
    syscall

    li $v0, 1
    move $a0, $s5
    syscall

    li $v0, 11
    li $a0, '='
    syscall

    li $v0, 1
    move $a0, $s6
    syscall

    li $v0, 11
    li $a0, '\n'
    syscall
    li $s7, 100
    div $s6, $s7
    mfhi $s6      # chi lay 2 chu so cuoi cua ket qua de in ra led
    j splitnumber # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
    nop

tru:
    sub $s6, $s4, $s5 # s6=s4-s5
    li $s3, 0
    blt $s6, 0, truam
    j intru
    nop
intru:
    li $v0, 1
    move $a0, $s4
    syscall

    li $v0, 11
    li $a0, '-'
    syscall

    li $v0, 1
    move $a0, $s5
    syscall

    li $v0, 11
    li $a0, '='
    syscall

    li $v0, 1

```

```

    move $a0, $s6
    syscall

    li $v0, 11
    li $a0, '\n'
    syscall
    j splitnumber    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
    nop
nhan:
    mul $s6,$s4,$s5    # s6=s4*s5
    li $s3,0
    j innhan
    nop
innhan:
    li $v0, 1
    move $a0, $s4
    syscall

    li $v0, 11
    li $a0, '*'
    syscall

    li $v0, 1
    move $a0, $s5
    syscall

    li $v0, 11
    li $a0, '='
    syscall

    li $v0, 1
    move $a0, $s6
    syscall

    li $v0, 11
    li $a0, '\n'
    syscall
    li $s7,100
    div $s6,$s7
    mfhi $s6    # chi lay 2 chu so sau cung cua ket qua in ra
    j splitnumber    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
    nop
chia:
    beq $s5,0,chia0
    li $s3,0
    div $s4,$s5    # s6=s4/s5
    mflo $s6
    mfhi $s7
    j inchia
    nop

```


inchia:

```
li $v0, 1
move $a0, $s4
syscall
```

```
li $v0, 11
li $a0, '/'
syscall
```

```
li $v0, 1
move $a0, $s5
syscall
```

```
li $v0, 11
li $a0, '='
syscall
```

```
li $v0, 1
move $a0, $s6
syscall
```

```
li $v0, 11
li $a0, ' '
syscall
```

```
li $v0, 11
li $a0, 'r'
syscall
```

```
li $v0, 11
li $a0, '='
syscall
```

```
li $v0, 1
move $a0, $s7
syscall
```

```
li $v0, 11
li $a0, '\n'
syscall
```

```
j splitnumber    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung led
nop
```

chia0:

```
li $v0, 55
la $a0, mess2
li $a1, 0
syscall
j resetled
```

truam:

```
li $v0, 55
```

```

    la $a0, mess1
    li $a1, 0
    syscall
    j resetled

splitnumber:  #ham chia ket qua thanh 2 chu so de hien thi len tung led
    li $t8,10
    div $s6,$t8  #s6/10
    mflo $t7      #t7 = result
    jal convert  #chuyen den ham chuyen t7 thanh bit hien thi len led

    sb $t4,0($t0) # hien thi len led trai
    add $sp,$sp,4
    sb $t7,0($sp)  #day gia tri bit nay vao stack
    add $sp,$sp,4
    sb $t4,0($sp)  #day bit nay vao stack
    add $s2,$t7,$zero  #s1 = gia tri bit led phai

    mfhi $t7      #t7= remainder
    jal convert  #convert t7 thanh bit hien thi len led
    sb $t4,0($t5) #hien thi len led phai
    add $sp,$sp,4
    sb $t7,0($sp)  #day gia tri bit nay vao stack
    add $sp,$sp,4
    sb $t4,0($sp)  #day bit nay vao stack
    add $s1,$t7,$zero  #s1 = gia tri bit led phai
    j resetled  #ham reset lai led

convert:
    addi $sp,$sp,4
    sw $ra,0($sp)
    beq $t7,0,case_0  #t7=0ham chuyen 0 thanh bit zero hien thi len led
    beq $t7,1,case_1
    beq $t7,2,case_2
    beq $t7,3,case_3
    beq $t7,4,case_4
    beq $t7,5,case_5
    beq $t7,6,case_6
    beq $t7,7,case_7
    beq $t7,8,case_8
    beq $t7,9,case_9
case_0:#ham chuyen 0 thanh bit zero hien thi len led
    lb $t4,zero  #t4=zero
    j finishconvert #ket thuc
case_1:
    lb $t4,one
    j finishconvert
case_2:
    lb $t4,two
    j finishconvert

```

```

case_3:
    lb $t4,three
    j finishconvert
case_4:
    lb $t4,four
    j finishconvert
case_5:
    lb $t4,five
    j finishconvert
case_6:
    lb $t4,six
    j finishconvert
case_7:
    lb $t4,seven
    j finishconvert
case_8:
    lb $t4,eight
    j finishconvert
case_9:
    lb $t4,nine
    j finishconvert
finishconvert:
    lw $ra,0($sp)
    addi $sp,$sp,-4
    jr $ra
done:
    beq $s0,1,resetled #s0=1-->toan tu-->chuyen den ham reset led
loadtoleftled: # ham hien thi bit len led trai
    lb $t6,0($sp) #load bit hien thi led tu stack
    add $sp,$sp,-4
    lb $t8,0($sp) #load gia tri cua bit nay
    add $sp,$sp,-4
    add $s2,$t8,$zero #s2 = gia tri bit led trai
    sb $t6,0($t0) # hien thi len led trai
loadtorightled: # ham hien thi bit len led phai
    sb $t4,0($t5) # hien thi bit len led phai
    add $sp,$sp,4
    sb $t7,0($sp) #day gia tri bit nay vao stack
    add $sp,$sp,4
    sb $t4,0($sp) #day bit nay vao stack
    add $s1,$t7,$zero #s1 = gia tri bit led phai
    j finish
resetled:
    li $s0,0 #s0=0--> doi nhap so tiep theo trong 2 so
    li $t8,0
    addi $sp,$sp,4
    sb $t8,0($sp)
    lb $t6,zero # day bit zero vao stack
    addi $sp,$sp,4
    sb $t6,0($sp)

```

```

finish:
    j end_exception
    nop
end_exception:
    # return to start of the loop instead of where the interrupt occur, since the loop doesn't
do meaningful thing
    la $a3, loop1
    mtc0 $a3, $14
    eret
setnextoperator:
setsecondnumber1: #ham tinh so thu 2 dang hien thi tren led trong 2 so
    mul $s5,$s2,10      # s5=s2*10+s1
    add $s5,$s5,$s1
    beq $s3,1,cong1      # s3=1--> cong
    beq $s3,2,tru1
    beq $s3,3,nhan1
    beq $s3,4,chia1
cong1:
    add $s6,$s5,$s4
    j incong1
    nop                # s6=s5+s4

incong1:
    li $v0, 1
    move $a0, $s4
    syscall

    li $v0, 11
    li $a0, '+'
    syscall

    li $v0, 1
    move $a0, $s5
    syscall

    li $v0, 11
    li $a0, '='
    syscall

    li $v0, 1
    move $a0, $s6
    syscall

    li $v0, 11
    li $a0, '\n'
    syscall
    li $s7,100
    div $s6,$s7
    mfhi $s6          # chi lay 2 chu so cuoi cua ket qua de in ra led

```

```

        j splitnumber1    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
    nop

tru1:
    sub $s6,$s4,$s5    # s6=s4-s5
    blt $s6,0,truam1
    j intru1
    nop
intru1:
    li $v0, 1
    move $a0, $s4
    syscall

    li $v0, 11
    li $a0, '-'
    syscall

    li $v0, 1
    move $a0, $s5
    syscall

    li $v0, 11
    li $a0, '='
    syscall

    li $v0, 1
    move $a0, $s6
    syscall

    li $v0, 11
    li $a0, '\n'
    syscall
    j splitnumber1    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
    nop
nhan1:
    mul $s6,$s4,$s5    # s6=s4*s5
    j innhan1
    nop
innhan1:
    li $v0, 1
    move $a0, $s4
    syscall

    li $v0, 11
    li $a0, '*'
    syscall

    li $v0, 1

```

```

        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, '\n'
        syscall
        li $s7, 100
        div $s6, $s7
        mfhi $s6      # chỉ lấy 2 chu số sau cùng của kết quả in ra
        j splitnumber1  # chuyển đến hàm chia kết quả thành 2 chu số để hiển thị lên từng
led:
        nop
chia1:
        beq $s5, 0, chia01
        div $s4, $s5      # s6=s4/s5
        mflo $s6
        mfhi $s7
        j inchia1
        nop
inchia1:
        li $v0, 1
        move $a0, $s4
        syscall

        li $v0, 11
        li $a0, '/'
        syscall

        li $v0, 1
        move $a0, $s5
        syscall

        li $v0, 11
        li $a0, '='
        syscall

        li $v0, 1
        move $a0, $s6
        syscall

        li $v0, 11
        li $a0, ''

```

```

syscall

li $v0, 11
li $a0, 'r'
syscall

li $v0, 11
li $a0, '='
syscall

li $v0, 1
move $a0, $s7
syscall

li $v0, 11
li $a0, '\n'
syscall
j splitnumber1    # chuyen den ham chia ket qua thanh 2 chu so de hien thi len tung
led
    nop
chia01:
    li $v0, 55
    la $a0, mess2
    li $a1, 0
    syscall
    j resetled1
truam1:
    li $v0, 55
    la $a0, mess1
    li $a1, 0
    syscall
    j resetled1
splitnumber1: #ham chia ket qua thanh 2 chu so de hien thi len tung led
    li $t8, 10
    div $s6, $t8    #s6/10
    mflo $t7        #t7 = result
    jal convert1    #chuyen den ham chuyen t7 thanh bit hien thi len led
#-----
#sb $t4, 0($t0) # hien thi len led trai
    add $sp, $sp, 4
    sb $t7, 0($sp)    #day gia tri bit nay vao stack
    add $sp, $sp, 4
    sb $t4, 0($sp)    #day bit nay vao stack
    add $s2, $t7, $zero    #s1 = gia tri bit led phai

#-----
    mfhi $t7        #t7= remainder
    jal convert1    #convert t7 thanh bit hien thi len led
# sb $t4, 0($t5) #hien thi len led phai
    add $sp, $sp, 4

```

```

        sb $t7,0($sp)    #day gia tri bit nay vao stack
        add $sp,$sp,4
        sb $t4,0($sp)    #day bit nay vao stack
        add $s1,$t7,$zero #s1 = gia tri bit led phai
        j resetled1      #ham reset lai led
convert1:
        addi $sp,$sp,4
        sw $ra,0($sp)
        beq $t7,0,case_01 #t7=0 -->ham chuyen 0 thanh bit zero hien thi len led
        beq $t7,1,case_11
        beq $t7,2,case_21
        beq $t7,3,case_31
        beq $t7,4,case_41
        beq $t7,5,case_51
        beq $t7,6,case_61
        beq $t7,7,case_71
        beq $t7,8,case_81
        beq $t7,9,case_91
case_01:      #ham chuyen 0 thanh bit zero hien thi len led
        lb $t4,zero    #t4=zero
        j finishconvert1 #ket thuc
case_11:
        lb $t4,one
        j finishconvert1
case_21:
        lb $t4,two
        j finishconvert1
case_31:
        lb $t4,three
        j finishconvert1
case_41:
        lb $t4,four
        j finishconvert1
case_51:
        lb $t4,five
        j finishconvert1
case_61:
        lb $t4,six
        j finishconvert1
case_71:
        lb $t4,seven
        j finishconvert1
case_81:
        lb $t4,eight
        j finishconvert1
case_91:
        lb $t4,nine
        j finishconvert1
finishconvert1:
        lw $ra,0($sp)

```



```

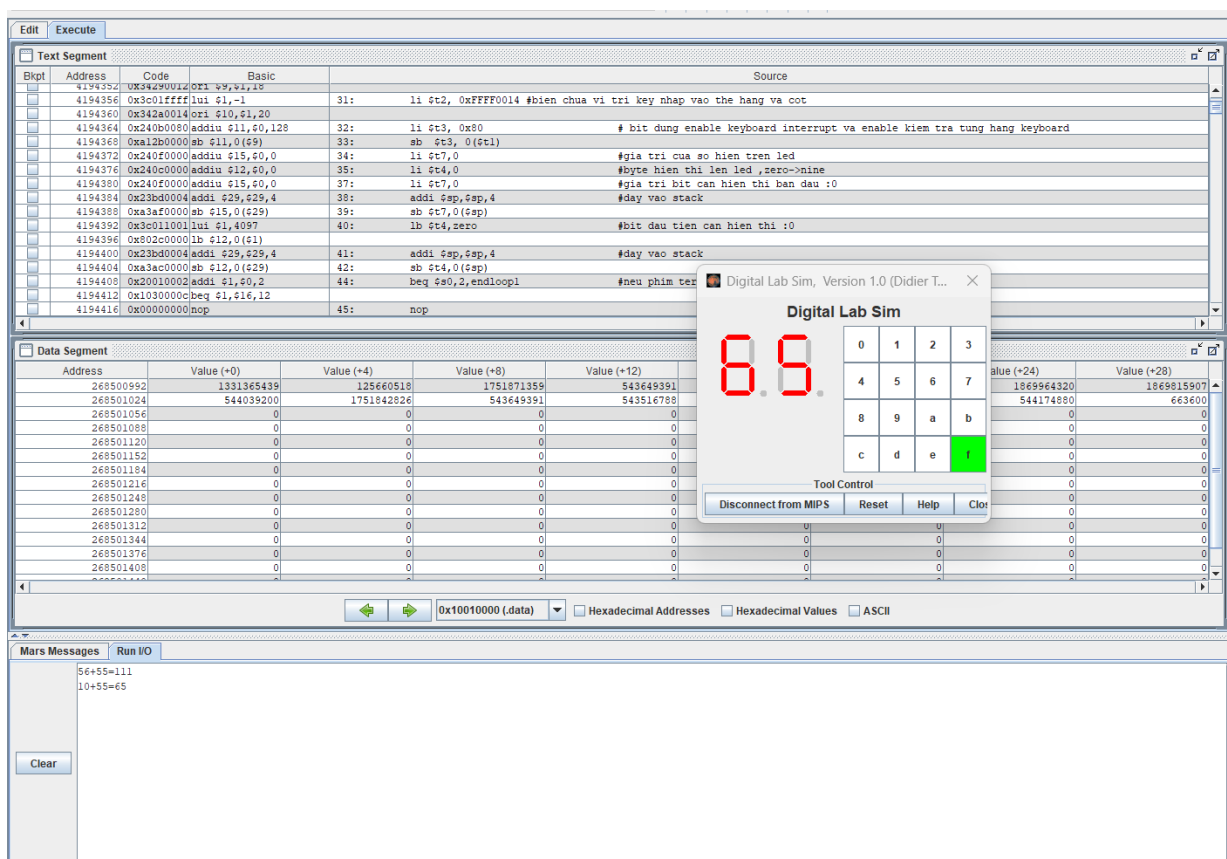
        addi $sp,$sp,-4
        jr $ra
done1:   beq $s0,1,resetled1
resetled1:
        li $s0,0      #s0=0--> doi nhap so tiep theo trong 2 so
        li $t8,0
        addi $sp,$sp,4
        sb $t8,0($sp)
        lb $t6,zero    # day bit zero vao stack
        addi $sp,$sp,4
        sb $t6,0($sp)
        mul $s4,$s2,10    # s4=s2*10+s1
        add $s4,$s4,$s1
        beq $a3,1,setadd
        nop
        beq $a3,2,sub
        nop
        beq $a3,3,setmul
        nop
        beq $a3,4,setdiv
        nop
setadd: addi $s3,$zero,1
        j finish1
        nop
sub:    addi $s3,$zero,2
        j finish1
        nop
setmul: addi $s3,$zero,3
        j finish1
        nop
setdiv: addi $s3,$zero,4
        j finish1
        nop

finish1:
        j end_exception1
        nop
end_exception1:
        # return to start of the loop instead of where the interrupt occur, since the loop doesn't
do meaningful thing
        la $a3, loop1
        mtc0 $a3, $14
        eret

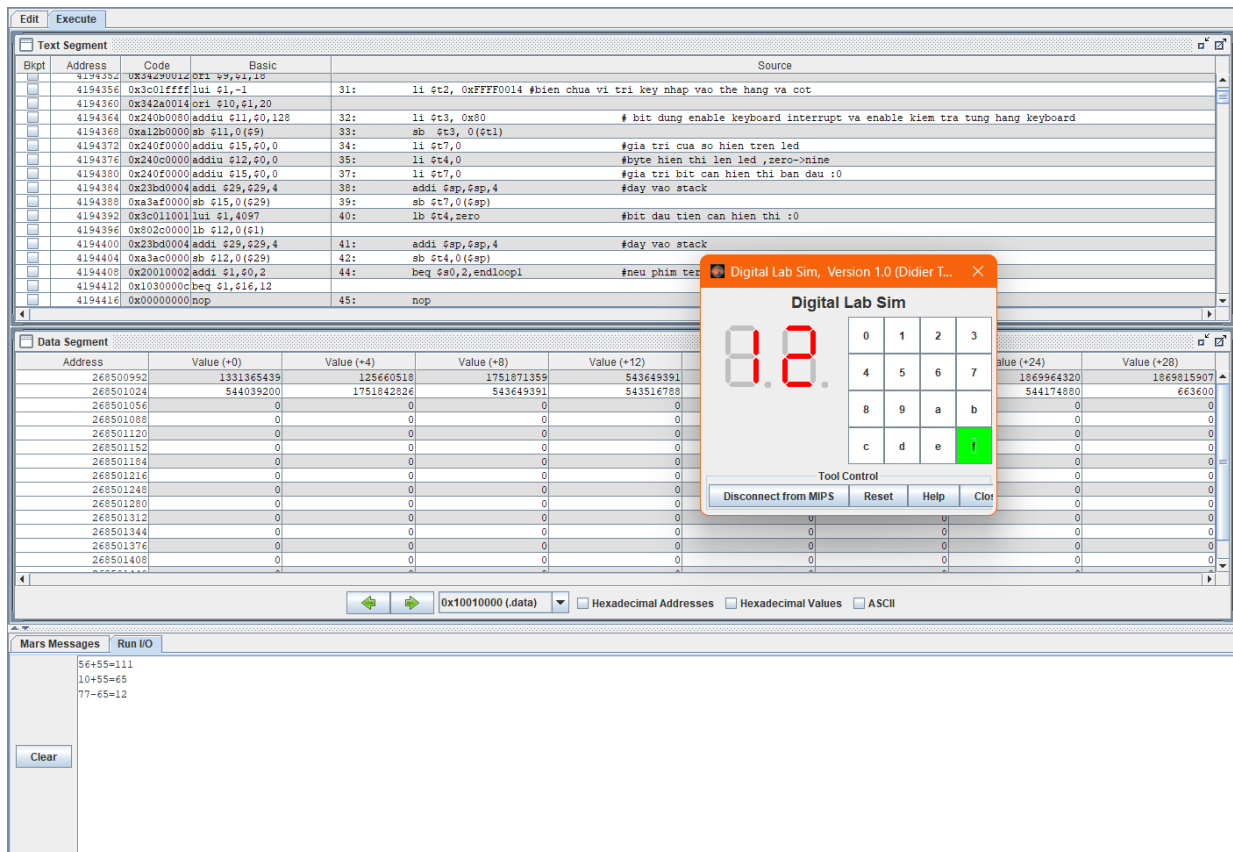
```

Chương 5: Kết quả chạy mô phỏng

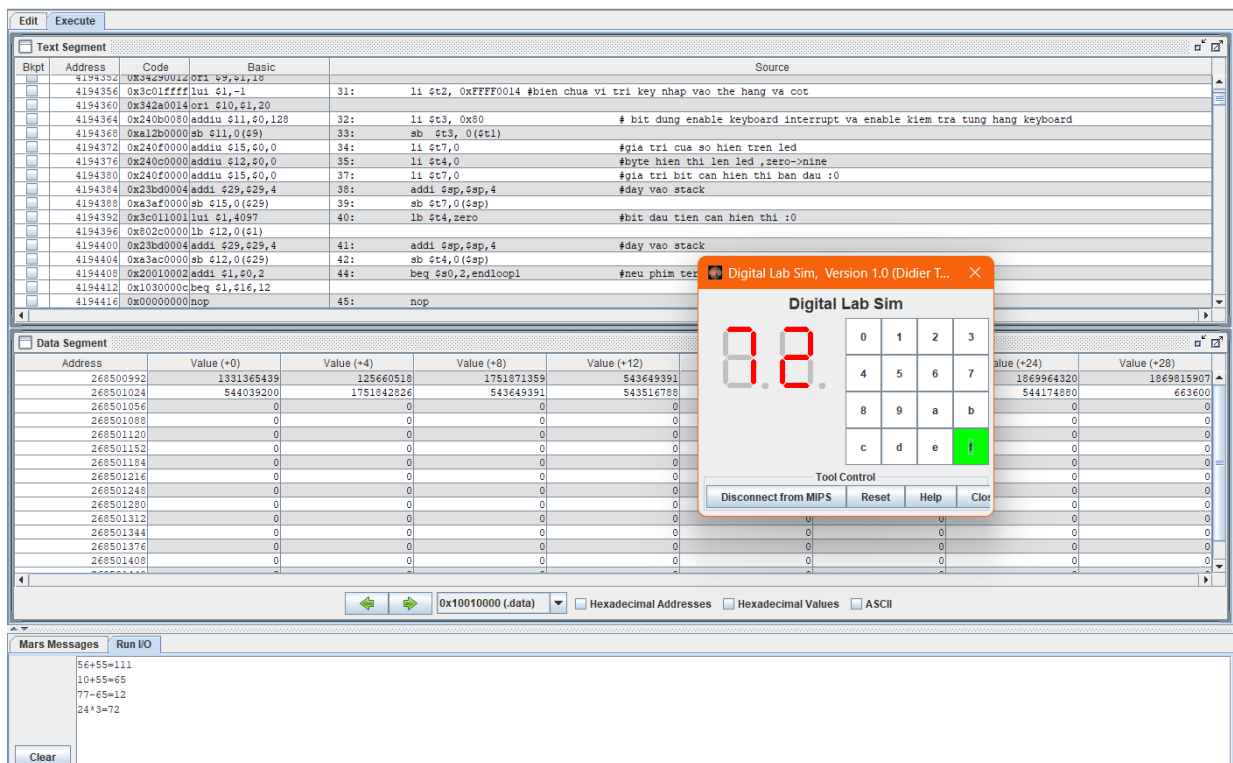
5.1 Ví dụ phép tính cộng $10+55=65$



5.2 Ví dụ phép tính cộng $77-65=12$



5.3 Ví dụ phép tính nhân $24 \times 3 = 72$



5.4 Ví dụ phép tính chia $72/3=24$

The screenshot displays the Digital Lab Sim software interface, which is used for simulating MIPS assembly code. The main window is divided into several sections:

- Text Segment:** Contains the MIPS assembly code for the division example. The code starts with a loop that calculates $72/3=24$ and stores the result in a register. The code is as follows:


```

      4194352: 0x342a0014 ori $t1,$t1,18
      4194356: 0x3c0fffff lui $t1,-1
      4194360: 0x342a0014 ori $t0,$t0,20
      4194364: 0x240b0080 addiu $t1,$t0,128
      4194368: 0xa12b0000 sb $t1,0($t0)
      4194372: 0x240f0000 addiu $t5,$t0,0
      4194376: 0x240c0000 addiu $t2,$t0,0
      4194380: 0x240f0000 addiu $t5,$t0,0
      4194384: 0x23bd0004 addi $t2,$t2,4
      4194388: 0xa3af0000 sb $t5,0($t2)
      4194392: 0x3c011001 lui $t4,4097
      4194396: 0x802c0000 lb $t2,0($t1)
      4194400: 0x23bd0004 addi $t2,$t2,4
      4194404: 0xa3ac0000 sb $t2,0($t2)
      4194408: 0x20010002 addi $t1,$t0,2
      4194412: 0x1030000c beq $t1,$t6,12
      4194416: 0x00000000 nop
      
```
- Data Segment:** Contains a table of memory addresses and their corresponding values. The table is as follows:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)
268500992	1331365439	125660518	1751871359	543649391
268501024	544039200	1751842826	543649391	543516788
268501056	0	0	0	0
268501088	0	0	0	0
268501120	0	0	0	0
268501152	0	0	0	0
268501184	0	0	0	0
268501216	0	0	0	0
268501248	0	0	0	0
268501280	0	0	0	0
268501312	0	0	0	0
268501344	0	0	0	0
268501376	0	0	0	0
268501408	0	0	0	0
- Digital Lab Sim Window:** A floating window showing a digital display with the value 2.4. It includes a numeric keypad and a tool control bar with buttons for Disconnect from MIPS, Reset, Help, and Close.
- Mars Messages:** A section at the bottom showing the results of the simulation. The messages are:


```

      10+55=65
      77-65=12
      24*3=72
      72/72=1 r=0
      1/72=0 r=1
      0/72=0 r=0
      0/72=0 r=0
      0/23=0 r=0
      72/72=1 r=0
      1/1=1 r=0
      1+1=2
      72/3=24 r=0
      
```