BÁO CÁO GIỮA KỲ

Lớp : 139365 – Học phần: Thực hành Kiến trúc máy tính

Nguyễn Thị Minh Châu – 20214997

Đỗ Văn Bình – 20210103

Project 3:

Create a program to convert from number to text, in English or Vietnamese (choice 1 of 2). The number in range from 0 to 999 999 999

For example:

Input: 1432

Output: one thousand four hundred and thirty two

- **Mã nguồn:**

```
.data
prompt: .asciiz  "Please input an Interger(0-999 999 999): "
zero: .asciiz"zero"
one: .asciiz "One "
two: .asciiz "Two "
three: .asciiz "Three "
four: .asciiz "Four "
five: .asciiz "Five "
six: .asciiz "Six "
seven: .asciiz "Seven "
eight: .asciiz "Eight "
nine: .asciiz "Nine "
ten: .asciiz "Ten "

eleven: .asciiz "Eleven "
twelve: .asciiz "Twelve "
thirteen: .asciiz "Thirteen "
fourteen: .asciiz "Fourteen "
fifteen: .asciiz "Fifteen "
sixteen: .asciiz "Sixteen "
seventeen: .asciiz "Seventeen "
eighteen: .asciiz "Eighteen "
nineteen: .asciiz "Nineteen "
twenty: .asciiz "Twenty-"
thirty: .asciiz "Thirty-"
forty: .asciiz "Forty-"
fifty: .asciiz "Fifty-"
sixty: .asciiz "Sixty-"
```

```
28  sixty: .asciiz "Sixty-"
29  seventy: .asciiz "Seventy-"
30  eighty: .asciiz "Eighty-"
31  ninety: .asciiz "Ninety-"
32
33  hundred: .asciiz "Hundred "
34  thousand: .asciiz "Thounsand "
35  million: .asciiz "Million "
36  error1: .asciiz "The number is smaller than 0, please input again. "
37  error2: .asciiz "The number is greater than 999999999, please input again. "
38  input_buffer: .space 100
39  .text
40  #-----------------------------------------------------------------------------------------------
41  #Procedure main1
42  #@brief Get an Interger from User Input, the interger is checked to be from 0 to 999999999
43  #@param[out] a1 The Interger that User inputted
44  #-----------------------------------------------------------------------------------------------
45  main1:
46          li $v0, 4               # Prompt User to input a Number
47          la $a0, prompt
48          syscall
49
50          li $v0, 5               # Get a Number from the keyboard
51          syscall
52          add $a1,$v0,$0          # Store the Number from User in a1
53  loop:
54          bltz $v0, negative      # If the inputted Number is smaller than 0, go to negative
55          bgt $v0, 999999999, outofrange
```

```
55          bgt $v0, 999999999, outofrange
56          j kb                    # If the inputted Number is greater or equal to 0, go to kb
57  negative:
58          li $v0, 4
59          la $a0, error1          #Print out the message
60          syscall
61          j main1                 # Roll back to main1 to input the number again
62  outofrange:
63          li $v0, 4
64          la $a0, error2
65          syscall
66          j main1
67
68  #-----------------------------------------------------------------------------------------------
69  #Procedure kb
70  #@bfief Assigning the value 10, 100, 1000, 1000000 to s0, s1, s3, s4
71  #-----------------------------------------------------------------------------------------------
72  kb:
73          addi $s0,$s0, 10
74          addi $s1,$s1, 100
75          addi $s3,$s3, 1000
76          addi $s4,$s4, 1000000
77
78          beq $a1,0,zero1 # Handle the special case number 0
79          nop
80          bne $a1,0,main # If the number is not 0 then go to main
81          nop
82  #-----------------------------------------------------------------------------------------------
```

```
#Procedure zero1
#@brief Print out "zero" if the inputted interger is 0
#-------------------------------------------------------------------------------------------------
zero1:
        li $v0, 4
        la $a0,zero
        syscall
        j done


main:
#-------------------------------------------------------------------------------------------------
#@brief We seperate our interger into 3 three-digit parts: million, thousand, and hundred
#-------------------------------------------------------------------------------------------------
        div $a1, $s4                    #divide a1 by 1000000
        mflo $a3                                #store quotient to a3
        beq $a3,0,thousands             #if a3=0 branch to thousands
        addi $t8,$t8,1                  #if a3 is not 0 then t8 = 1
        j print3

print_mils:
        addi $t8,$t8,-1                 #If print million then t8 = 0
        la $a0, million
        syscall

thousands:
        div $a1, $s4                      #divide a1 by 1000000
        mfhi $a3                          #store remainder in a3
        mfhi $a3                          #store remainder in a3
        div $a3,$s3                       #divide a3 by 1000
        mflo $a3                          #store quotient in a3
        beq $a3,0,units                  #if a3 = 0 brach to units
        addi $t9,$t9,1                   #if a3 is not 0 then t9 = 1
        j print3
print_thous:
        addi $t9,$t9,-1                  #If print thousand then t9 = 0
        la $a0, thousand
        syscall

units:
        div $a1, $s3                            #divide a1 by 1000
        mfhi $a3                                #store remainder in a3
        j print3



#-------------------------------------------------------------------------------------------------
#Procedure print3
#@brief After we got 3 three-digit parts, now we turn them into text
#@param[in] a3 The three-digit part we got above
#-------------------------------------------------------------------------------------------------
print3:
        div $a3 ,$s1                    #divide a3 by 100
        mflo $v1                        #store quotient in v1
        jal print_digit                 #branch to print_digit
        nop
```

```
137         nop
138         beq $v1,0,last_2units           #if v1 = 0 branch to last_2units
139         la $a0,hundred                  #if v1 is not 0 then print "hundred"
140         syscall
141
142 last_2units:
143         div $a3 ,$s1                    #divide a3 by 100
144         mfhi $v1                            #store remainder in v1
145         beq $v1, 0, exit                #if v1 = 0 branch to exit
146         bgt $v1,19,print_ty             #if v1>19 branch to print_ty
147         bgt $v1,9, print_teen           #if v1>9 branch to print_teen
148 last_unit:
149         div $v1, $s0                    #divide v1 by 10
150         mfhi $v1                         #store remainder in v1
151         jal print_digit                 #branch to print_digit
152         nop
153         j exit                          #branch to exit
154 #-----------------------------------------------------------------------------------------------
155 #@brief We turn 1, 2, 3, 4, 5, 6, 7, 8, 9 into text
156 #-----------------------------------------------------------------------------------------------
157 print_digit:
158         li $v0,4
159 p1:     bne $v1,1,p2
160         la $a0,one
161         syscall
162         j return
163 p2:     bne $v1,2,p3
164         la $a0,two
```

```
164         la $a0,two
165         syscall
166         j return
167 p3:     bne $v1,3,p4
168         la $a0,three
169         syscall
170         j return
171 p4:     bne $v1,4,p5
172         la $a0,four
173         syscall
174         j return
175 p5:     bne $v1,5,p6
176         la $a0,five
177         syscall
178         j return
179 p6:     bne $v1,6,p7
180         la $a0,six
181         syscall
182         j return
183 p7:     bne $v1,7,p8
184         la $a0,seven
185         syscall
186         j return
187 p8:     bne $v1,8,p9
188         la $a0,eight
189         syscall
190         j return
191 p9:     bne $v1,9,return
```

```
191 p9:     bne $v1,9,return
192         la $a0,nine
193         syscall
194         j return
195
196 #-----------------------------------------------------------------------------------------------
197 #@brief We turn 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 to text
198 #-----------------------------------------------------------------------------------------------
199 print_teen:
200         li $v0,4
201 p10:    bne $v1,10,p11
202         la $a0,ten
203         syscall
204         j exit
205 p11:    bne $v1,11,p12
206         la $a0,eleven
207         syscall
208         j exit
209 p12:    bne $v1,12,p13
210         la $a0,twelve
211         syscall
212         j exit
213 p13:    bne $v1,13,p14
214         la $a0,thirteen
215         syscall
216         j exit
217 p14:    bne $v1,14,p15
218         la $a0,fourteen
```

```
218         la $a0,fourteen
219         syscall
220         j exit
221 p15:    bne $v1,15,p16
222         la $a0,fifteen
223         syscall
224         j exit
225 p16:    bne $v1,16,p17
226         la $a0,sixteen
227         syscall
228         j exit
229 p17:    bne $v1,17,p18
230         la $a0,seventeen
231         syscall
232         j exit
233 p18:    bne $v1,18,p19
234         la $a0,eighteen
235         syscall
236         j exit
237 p19:    bne $v1,19,p20
238         la $a0,nineteen
239         syscall
240         j exit
241
242 #-----------------------------------------------------------------------------------------------
243 #@brief We turn 20, 30, 40, 50, 60, 70, 80, 90 into text
244 #-----------------------------------------------------------------------------------------------
245 print_ty:
```

```
print_ty:
        li $v0,4
p20:    bgt $v1,29,p30
        la $a0,twenty
        syscall
        j last_unit
p30:    bgt $v1,39,p40
        la $a0,thirty
        syscall
        j last_unit
p40:    bgt $v1,49,p50
        la $a0,forty
        syscall
        j last_unit
p50:    bgt $v1,59,p60
        la $a0,fifty
        syscall
        j last_unit
p60:    bgt $v1,69,p70
        la $a0,sixty
        syscall
        j last_unit
p70:    bgt $v1,79,p80
        la $a0,seventy
        syscall
        j last_unit
p80:    bgt $v1,89,p90
        la $a0,eighty
        syscall
        j last_unit
p90:
        la $a0,ninety
        syscall
        j last_unit

return:
        jr $ra
#-------------------------------------------------------------------------------------------------
#@brief We check if we need to print "million" and "thousand" from t8 and t9
#-------------------------------------------------------------------------------------------------
exit:
        beq $t8,1,print_mils            # If t8 = 1, branch to print_mils
        beq $t9,1,print_thous           # If t9 = 1, branch to print_thous
done:
```

- **Kết quả:**

Clear

```
Please input an Interger(0-999 999 999): 1234567
One Million Two Hundred Thirty-Four Thounsand Five Hundred Sixty-Seven
-- program is finished running (dropped off bottom) --
```

- **Vấn đề**: Tạo một chương trình chuyển đổi từ số thành chữ, sử dụng Tiếng Việt hoặc Tiếng Anh. Số trong khoảng từ 0 đến 999 999 999
- **Lời giải**: Ta chuyển số đầu vào thành 3 phần, mỗi phần gồm 3 chữ số: phần triệu, phần nghìn, phần trăm. Từ mỗi phần có 3 chữ số đấy ta lại chia nhỏ thành phần trăm, phần chục, phần đơn vị.
  Từ đây, với các phần đã được chia nhỏ từ số ban đầu, ta duyệt lần lượt để chuyển chúng thành chữ.

- **Thuật toán**: Ta thực hiện lần lượt:
  + Chia số ban đầu cho 1 000 000, thương số (nếu > 0) là phần triệu.
  + Chia phần dư của phép toán trên cho 1000, thương số (nếu > 0) là phần nghìn.
  + Phần dư của phép toán trên là phần trăm.
  + Lần lượt chia phần trăm cho 100 và 10, lấy phần dư để được phần chục và phần đơn vị.

- **Mô tả các thanh ghi, các hàm đã sử dụng:**
  + $v0: tham số cho syscall, đồng thời nhận số nguyên đầu vào từ bàn phím.
  + $a0: lưu địa chỉ các string trong data, từ đó gọi ra khi in kết quả.
  + $a1: lưu số nguyên đầu vào.
  + $a3: lưu các phần 3 chữ số ta tách được.
  + $v1: lưu phần chục, phần đơn vị ta tách được.
  + $s0: s0 = 10
  + $s1: s1 = 100
  + $s3: s3 = 1000
  + $s4: s4 = 1000000
  + $ra: lưu địa chỉ trả về
  + $t8: t8 là tham số kiểm tra có cần in ra “million” hay không.
  + $t9: t9 là tham số kiểm tra có cần in ra “thousand” hay không

Project 13:

Ticket numbers usually consist of an even number of digits. A ticket number is considered lucky if the sum of the first half of the digits is equal to the sum of the second half. Given a ticket number n, determine if it's lucky or not. Example
For n = 1230, the output should be isLucky(n) = true; For n = 239017, the output should be isLucky(n) = false.

- Mã nguồn:

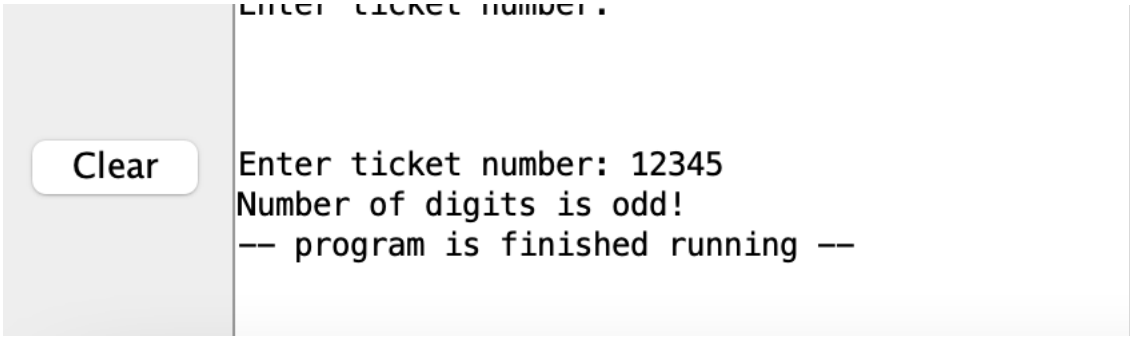
```
.data
message_input: .asciiz "Enter ticket number: "

invalid_big:      .asciiz       "Number is too big!"
invalid_negative: .asciiz       "Number is negative!"
invalid_odd:      .asciiz       "Number of digits is odd!"

lucky: .asciiz  "Lucky number"
not_lucky: .asciiz "Not lucky number"

.text
main:   li $v0, 4               # in thong bao nhan dau vao
        la $a0, message_input
        syscall

        li $v0, 5               # Doc so n nhap tu ban phim
        syscall
        add $s0, $v0, $0        # Luu gia tri n vao s0

        jal check_input         # check_input -> kiem tra dau vao n
        nop

        jal is_lucky            # is_lucky -> kiem tra xem n lucky hay khong va in ket qua
        nop

end_main:       li $v0, 10      # ket thuc chuong trinh
                syscall
#---------------------------------------------------------------------------
# function error($a1)
# In thong bao loi
# $a1: luu dia chi cua string chua noi dung loi
#---------------------------------------------------------------------------
error:  li $v0, 4
        add $a0, $a1, $0
        syscall
        j end_main

#---------------------------------------------------------------------------
# function result($a1)
# In thong bao ket qua, la ham con trong is_lucky
# $a1: luu dia chi cua string chua noi dung ket qua
#---------------------------------------------------------------------------
result: li $v0, 4
        add $a0, $a1, $0
        syscall

        j end_is_lucky
#---------------------------------------------------------------------------
# function check_input($s0)
# Kiem tra so nhap vao n co thoa man dieu kien hay khong
# Cac loi kiem tra: so qua lon, so am, so chu so la so le
# Dong thoi push cac chu so vao stack voi con tro $sp
# $s0: chua gia tri n
# Tra ve: $k0 -> so chu so cua n
#         $k1 -> mot nua so chu so cua n n (1/2 của $k0)
#---------------------------------------------------------------------------
check_input:    add  $s1, $s0, $0       # $s1 = $s0 = n
                li   $t2, 1000000000    # 1.000.000.000 -> gioi han cua n

# check_big -> kiem tra n ($s1) < 1.000.000.000 hay khong
check_big:      slt  $t1, $s1, $t2      # Neu n($s1) < 1.000.000.000 ? t1 = 1 nguoc lai t1 = 0

```

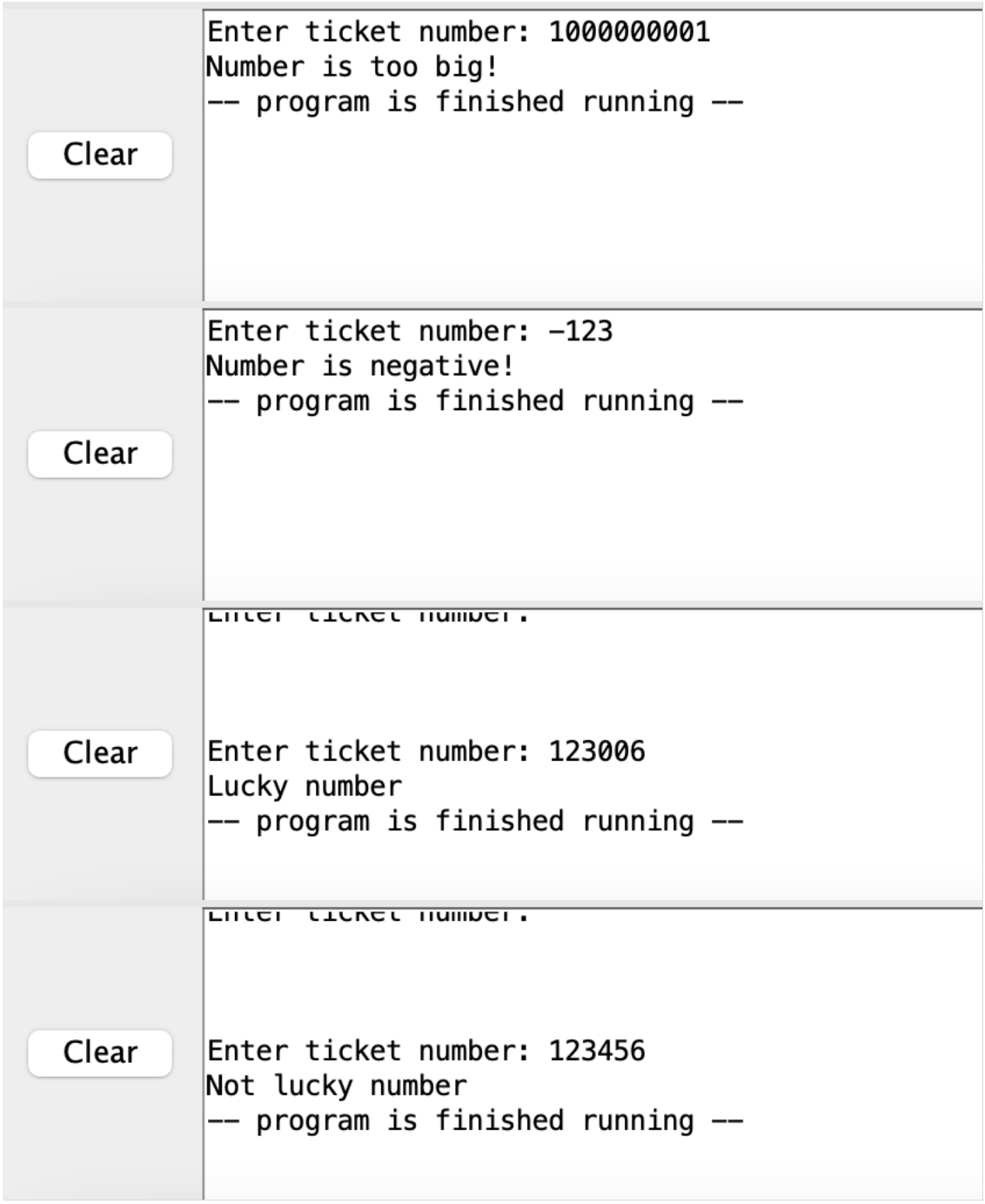
```
check_big:      slt  $t1, $s1, $t2      # Neu n($s1) < 1.000.000.000 ? t1 = 1 nguoc lai t1 = 0

                la   $a1, invalid_big   # gan dia chi string chua loi vao $a1 de su dung ham error
                beqz $t1, error         # in loi neu t1 = 0 ( hay n > 1.000.000.000 )

# check_negative -> kiem tra n la so am hay khong
check_negative: slt  $t1, $0, $s1               # kiem tra 0 < n($s1) ? t1 = 1 : t1 = 0

                la   $a1, invalid_negative      # gan dia chi string chua loi vao $a1 de su dung ham error
                beqz $t1, error                 # in loi neu t1 = 0 ( hay n < 0 )

# check_odd -> kiem tra so chu so cua n chan hay le
check_odd:      addi $t1, $0, 10        # $t1 = 10 -> dung lam so chia de tach cac chu so

# loop -> chia n dan cho 10 va gan lai n bang thuong va so du push dan vao stack
# Vong lap dung khi n = 0 -> het chu so
# $k0 dem so chu so
loop:           beq $s1, 0, countinue_check     # if ($s1) == 0 -> dung loop
                nop

                divu $s1, $t1           # chia n cho 10 lay thuong va so du o lo, hi
                mfhi $t2                # so du o hi duoc luu vao $t2 = n % 10
                mflo $t3                # thuong o lo duoc luu vao $t3 = n / 10
                add  $s1, $t3, $0       # n($s1) = n / 10 = $t3

# push: push chu so hay so du vua tim duoc vao stack
push:           addi $sp, $sp, -4       # danh stack cho mot phan tu
                sw   $t2, 0($sp)        # luu so du hay cac chu cai cua n vao stack
                add  $k0, $k0, 1        # tang bien dem $k0 ++
                j loop

# countinue_check -> sau khi dung vong lap, tiep tuc kiem tra so chu so le hay không (kiem tra $k0 chia het cho 2)
```

```
# countinue_check -> sau khi dung vong lap, tiep tuc kiem tra so chu so le hay không (kiem tra $k0 chia het cho 2)
countinue_check:addi $t4, $0, 2         # gan t4 = 2 -> lam so chia

                div  $k0, $t4           # chia $k0 cho 2 -> thuong va du luu o lo, hi
                mfhi $t4                # lay so du tu hi luu vao $t4
                mflo $k1                # thuong la 1/2 so chu so cua n luu vao $k1

                la   $a1, invalid_odd   # gan dia chi string chua loi vao $a1 de su dung ham error
                bne  $t4, 0, error      # neu khong chia het hay so du ($t4) khac 0 -> bao loi bang ham error
                nop

                jr $ra                  # thoat ham check_input tro ve ham main

#-------------------------------------------------------------------------------
# function is_lucky($sp, $k1)
# Kiem tra tong nua dau va nua sau cua cac chu so cua n (lucky) va in ket qua qua ham result
# Duyet nua stack dau và tinh tong, tuong tu voi nua sau
# So sanh 2 tong va dua ra ket luan
# $sp -> stack chu cac chu so cua n
# $k1 -> 1/2 so chu so cua n hay 1/2 so phan tu cua stack
#-------------------------------------------------------------------------------
is_lucky: addi $t1, $0, 0       # i = 0 -> bien chay vong lap nua dau
          addi $t2, $0, 0       # j = 0 -> bien chay vong lap nua sau
          add $s2, $0, $0       # $s2 luu sum cua nua dau
          add $s3, $0, $0       # $s3 luu sum cua nua sau

# loop1: tinh tong nua dau cua stack
loop1:  beq     $t1, $k1, loop2         # neu i($t1) == $k0 -> dung vong lap -> di den tinh tong nua sau
        nop
# pop1: lay phan tu trong stack ra de cong dan vao $s2 de tinh tong
pop1:   lw      $t3, 0($sp)     # pop chu ra khoi stack vao $t3
```

```
118 # loop1: tinh tong nua dau cua stack
119 loop1:  beq     $t1, $k1, loop2         # neu i($t1) == $k0 -> dung vong lap -> di den tinh tong nua sau
120         nop
121 # pop1: lay phan tu trong stack ra de cong dan vao $s2 de tinh tong
122 pop1:   lw      $t3, 0($sp)     # pop chu ra khoi stack vao $t3
123         addi    $sp, $sp, 4     # xoa 1 muc ra khoi stack
124
125         add     $s2, $s2, $t3   # tinh tong ($s2) += $t3 (chu so vua pop ra)
126         addi    $t1, $t1, 1     # tang bien dem de lap i++
127         j       loop1
128
129 # loop2: tinh tong nua sau cua stack
130 loop2:  beq     $t2, $k1, check_lucky   # neu i($t1) == $k0 -> dung vong lap -> di den so sanh 2 tong o check_lucky
131         nop
132 # pop1: lay phan tu trong stack ra de cong dan vao $s3 de tinh tong
133 pop2:   lw      $t3, 0($sp)     # pop và luu vao $t3
134         addi    $sp, $sp, 4     # xoa 1 muc ra khoi stack
135
136         add     $s3, $s3, $t3   # tinh tong ($s3) += $t3
137         addi    $t2, $t2, 1     # tang bien dem de lap j++
138         j       loop2
139 # check_lucky -> kiem tra tong 2 nua co bang nhau hay khong
140 check_lucky:    la $a1, not_lucky       # luu dia chi thong bao vao $a1 đe su dung ham result
141                 bne $s2, $s3, result    # neu 2 nua khong bang nhau in thong bao not_lucky
142                 nop                     # nguoc lai -> lucky
143                 la $a1, lucky           # luu dia chi thong bao vao $a1 de su dung ham result
144                 j  result
145 end_is_lucky:   jr $ra                  # ket thuc ham is_lucky tro ve ham main
```

- Kết quả:





- **Hướng làm bài:** Nhập số ở vé số n và kiểm tra các điều kiện (số quá lớn, số âm và số các chữ số là số lẻ), tính số các chữ số và chia làm hai nửa, tính tổng các chữ số của từng nửa và so sánh.

- **Giải thích:**
  1. Khởi tạo các chuỗi

2. Hàm main: Thực hiện in chuỗi “Enter ticket number: “ để yêu cầu nhập số vé từ bàn phím. Lưu số vé n vào $s0. Sau đó lần lượt gọi các hàm check_input và is_lucky
3. Các hàm error và result: dùng để in thông báo lỗi và in kết quả
4. Hàm check_input: Kiểm tra giá trị đầu vào thõa mãn. Lưu giá trị của $s0 (n) vào $s1 và gán $t2 = 1.000.000.000 là giới hạn của n
    - Check_big: So sánh s1 và t2, nếu s1 > t2 thì nhảy đến error để thông báo lỗi số quá lớn
    - Check_negative: So sánh s1 và 0, nếu s1 < 0 thì nhảy đến error để tông báo lỗi số âm
    - Check_odd: Sử dụng vòng lặp để chia n cho 10, tiếp tục chia phần nguyên thu được cho 10 cho đến khi phần nguyên = 0 thì nhảy đến countinue_check, lần lượt đẩy số dư thu được vào stack (các số dư chính là các chữ số tạo nên n) và dùng $k0 để lưu số chữ số của n.
        - Countinue_check: chia $k0 cho 2, phần nguyên lưu vào $k1 (đây chính là một nửa số chữ số của n), lưu phần dư vào t4 và so sánh nếu t4 khác 0 thì nhảy đến error in thông báo lỗi.
5. Hàm is_Lucky: tạo $t1 = 0 = i (biến chạy vòng lặp nửa đầu), $t2 = 0 = j (biến chạy vòng lặp nửa sau) và $s2, $s3 lần lượt để lưu tổng nửa đầu, nửa sau
    - Lần lượt pop số ra từ stack và lưu vào t3, xóa 1 mục ra khỏi stack, s2 = s2 + t3, tăng biến chạy t1 (i) và lặp đến khi nào t1 = k1 (i = ½ số chữ số của n) thì dừng. Ta được tổng các chữ số nửa đầu của n
    - Làm tương tự ta tính được tổng các chữ số nửa cuối của n lưu tại s3
    - Check_lucky: nếu s2 khác s3 thì nhảy sang result thông báo kết quả “ Not lucky number “. Ngược lại nhảy sang result và thông báo Lucky number”. Kết thúc và trở về hàm main.

- **Mô tả các thanh ghi đã sử dụng:**

+ $v0: tham số cho syscall, đồng thời nhận số vé n từ bàn phím.
+ $a0, $a1: lưu địa chỉ các string trong data, từ đó gọi ra khi in kết quả.
+ $s0: lưu số vé n đầu vào.
+ $s1: lưu n, lưu phần nguyên khi lấy n chia 10
+ $t2: t2 = 1.000.000.000
+ $t1: lưu các giá trị tạm thời như các giá trị so sánh, 10 (sử dụng làm số chia để tách các chữ số)
+ $k0: lưu số chữ số của n
+ $k1: ½ số chữ số của n
+ $ra: lưu địa chỉ trả về
+ $t1: biến chạy i
+ $t2: biến chạy j
+ $s2: tổng nửa đầu
+ $s3: tổng nửa cuối