Báo Cáo Giữa Kì Kiến Trúc Máy tính

BTGK 11:

Mô tả bài toán:

Cho đầu vào là tên của 2 học sinh có ký tự trống, chuyển đổi tên từ LastName-FirstName thành FirstName-LastName

Phương pháp giải bài toán:

Tìm ký tự trống tách giữa LastName và FirstName, tiếp theo sao chép FirstName vào biến kết quả, sau đó sao chép nốt LastName vào biến kết quả (phía sau FirstName), cuối cùng in ra biến kết quả

Code C minh họa:

```c
#include <stdio.h>
#include <string.h>

int ConvertName(char str[])
{
    int savei, n = strlen(str), k;
    char result[50] = "";

    for (int i = n - 1; i >= 0; i--)
    {
        if (str[i] == ' ')
        {
            savei = i;
            break;
        }
    }
    k = n - savei;

    for (int i = savei + 1; i < n; i++)
    {
        result[i - savei - 1] = str[i];
    }
    result[k - 1] = ' ';

    for (int i = 0; i <= savei - 1; i++)
    {
        result[k + i] = str[i];
    }
    printf("%s\n", result);
}
```

```c
int main()
{
    char str1[50], str2[50];

    printf("Input name of first students: ");
    gets(str1);
    printf("Input name of second students: ");
    gets(str2);
    printf("Result: \n");

    ConvertName(str1);
    ConvertName(str2);

    return 0;
}
```

Code MIPS assembly:

BTGK_11.asm

```
.data
        title1: .asciiz  "Input name of first students: "
        title2: .asciiz  "Input name of second students: "
        title3: .asciiz  "Ket qua: \n"
        title4: .asciiz  "\n"

        space: .ascii " "
        End: .ascii  "\n"
        Result: .space 100
        name1: .space 100
        name2: .space 100

#-----------------------effects of variables------------
#       s1 = address(X[0])
#       s3 = address(X[n])
#       s4 = space
#       s5 = /0
#       s6 = address(X[wall])
#       s7 = address(Result)
#-------------------------------------------------------

.text
        # input first name
        li $v0, 4
        la $a0, title1
        syscall

        li $v0, 8
        la $a0, name1
        addi $a1, $zero, 100
        syscall

        li $v0, 4
        la $a0, title4
        syscall
```

```
        # input second name
        li $v0, 4
        la $a0, title2
        syscall

        li $v0, 8
        la $a0, name2
        addi $a1, $zero, 100
        syscall

        li $v0, 4
        la $a0, title4
        syscall

        # output
        li $v0, 4
        la $a0, title3
        syscall

jal ProName1
jal ProName2
j EndProgram

ProName1:
        la $t1, name1
        j Initial

ProName2:
        la $t1, name2
        j Initial

EndProgram:
        li $v0, 10
        syscall


Initial:
        add $s1, $t1, $zero     # s1 = address(X)


        la $t0, End             # t0 = address(End)
        lb $s5, 0($t0)          # s5 = End

        la $s7, Result          # t0 = address(Result)

        la $t0, space           # t4 = address(space)
        lb $s4, 0($t0)          # s4 = space

# Process
Start:
        add $s0, $zero, $zero   # s0 = i = 0

Loop:
        add $t0, $s0, $s1       # t0 = address(X[i])
        lb $t1, 0($t0)          # t1 = X[i]
        beq $t1, $s5, Change    # if X[i] == "/n", exit Loop
        addi $s0, $s0, 1        #s0 = s0 + 1 <-> i = i + 1
        j Loop
```

```
# Tim kiem space
Change:
        add $s3, $t0, $zero     # s3 = address(X[n])
        addi $s0, $zero, 1      # s0 = i = 1

LoopChange:
        sub $t2, $s3, $s0       # t2 = address(X[n-i])
        lb $t1, 0($t2)          # t1 = X[n-i]
        beq $t1, $s4, Exe1      # if X[n-i] == ' ', exit LoopChange
        beq $t2, $s1, Nospace   # if there is no space, goto Nospace
        addi $s0, $s0, 1        # s0 = s0 + 1 <-> i = i + 1
        j LoopChange
```

```
# Sao chep Ten
Exe1:
        add $s6, $t2, $zero     # s6 = address(X[wall])

LoopExe1:
        addi $t2, $t2, 1        # t2 = address(X[wall + i])
        lb $t3, 0($t2)          # t3 = X[wall + i]

        beq $t3, $s5, Exe2      # if X[wall + i] == "/n", exit Loop
        sb $t3, 0($s7)          # Result[i-1] = X[wall + i]
        addi $s7, $s7, 1        # s7 = address(Result[i])

        j LoopExe1              #next character
        nop
```

```
# Sao chep ho
Exe2:
        sb $s4, 0($s7)
        add $t1, $s1, $zero     # t1 = address(X[i])

LoopExe2:
        beq $t1, $s6, EndChange # if address(X[i]) == address(X[wall]), exit LoopExe2
        lb $t2, 0($t1)          # t2 = X[i]

        addi $s7, $s7, 1        # s7 = Result[k + i]
        sb $t2, 0($s7)          # Result[k + i]= X[i]

        addi $t1, $t1, 1        # address(X[i + 1])

        nop
        j LoopExe2              # next character

EndChange:


        addi $s7, $s7, 1        # s7 = Result[End]
        sb $s5, 0($s7)          #  Ket thuc xau

        li $v0, 4
        la $a0, Result
        syscall

        jr $ra
```

```
# In ra ten
Nospace:
        li $v0, 4
        add $a0, $s1, $0
        syscall

        jr $ra

```

Giải thích code:

- Vùng dữ liệu, chứa các khai báo biến:

BTGK_11.asm

```
.data
        title1: .asciiz  "Input name of first students: "
        title2: .asciiz  "Input name of second students: "
        title3: .asciiz  "Ket qua: \n"
        title4: .asciiz  "\n"

        space: .ascii " "
        End: .ascii  "\n"
        Result: .space 100
        name1: .space 100
        name2: .space 100

#-----------------------effects of variables-------------
#       s1 = address(X[0])
#       s3 = address(X[n])
#       s4 = space
#       s5 = /0
#       s6 = address(X[wall])
#       s7 = address(Result)
#-------------------------------------------------------
```

- Thanh ghi s1: chứa địa chỉ của ký tự đầu tiên trong chuỗi
- Thanh ghi s3: chứa địa chỉ của ký tự cuối cùng trong chuỗi
- Thanh ghi s4: chứa ký tự space
- Thanh ghi s5: chứa ký tự kết thúc chuỗi
- Thanh ghi s6: chứa địa chỉ ký tự space gần cuối chuỗi nhất
- Thanh ghi s7: địa chỉ của biến result

- Vùng lệnh, chứa các lệnh hợp ngữ:
  + *Nhóm lệnh 1:*

```
.text
        # input first name
        li $v0, 4
        la $a0, title1
        syscall

        li $v0, 8
        la $a0, name1
        addi $a1, $zero, 100
        syscall

        li $v0, 4
        la $a0, title4
        syscall
```

```
37          # input second name
38          li $v0, 4
39          la $a0, title2
40          syscall
41
42          li $v0, 8
43          la $a0, name2
44          addi $al, $zero, 100
45          syscall
46
47          li $v0, 4
48          la $a0, title4
49          syscall
50
51          # output
52          li $v0, 4
53          la $a0, title3
54          syscall
```

- Với *$v0 = 4* thì thực hiện chức năng in chuỗi, *$a0* = địa chỉ chuỗi để in
- Với *$v0 = 8* thì thực hiện chức năng đọc chuỗi, *$a0* = địa chỉ của bộ đệm đầu vào, $a1 = số ký tự tối đa để đọc
- Dòng 23 - 35: Đọc tên của học sinh thứ nhất
- Dòng 37 - 49: Đọc tên của học sinh thứ hai
- Dòng 51 - 54: In ra title “kết quả”

+ *Nhóm lệnh 2:*

```
55
56  jal ProName1
57  jal ProName2
58  j EndProgram
59
60  ProName1:
61          la $t1, name1
62          j Initial
63
64  ProName2:
65          la $t1, name2
66          j Initial
67
68  EndProgram:
69          li $v0, 10
70          syscall
71
72
73  Initial:
74          add $s1, $t1, $zero     # s1 = address(X)
75
76
77          la $t0, End             # t0 = address(End)
78          lb $s5, 0($t0)          # s5 = End
79
80          la $s7, Result          # t0 = address(Result)
81
82          la $t0, space           # t4 = address(space)
83          lb $s4, 0($t0)          # s4 = space
```

- *jal:* jump and link
- Dòng 60 - 62: Gán $s1 = address(name1) và nhảy đến nhãn “Initial”
- Dòng 64 - 66: Gán $s1 = address(name1) và nhảy đến nhãn “Initial”
- Dòng 68 - 70: Với $s0 = 10 thì kết thúc chương trình

+ *Nhóm lệnh 3:*

```
86  # Process
87  Start:
88          add $s0, $zero, $zero   # s0 = i = 0
89
90  Loop:
91          add $t0, $s0, $s1       # t0 = address(X[i])
92          lb $t1, 0($t0)          # t1 = X[i]
93          beq $t1, $s5, Change    # if X[i] == "/n", exit Loop
94          addi $s0, $s0, 1        #s0 = s0 + 1 <-> i = i + 1
95          j Loop
```

- *lb:* load byte (Lệnh này nạp giá trị byte nhớ có địa chỉ *($t0 + 0)* vào byte thấp của thanh ghi *$t1*)
- *Loop:* Tìm địa chỉ của ký tự kết thúc chuỗi tên
- Sau đó nhảy đến nhãn Change

+ *Nhóm lệnh 4:*

```
 98  # Tim kiem space
 99  Change:
100         add $s3, $t0, $zero     # s3 = address(X[n])
101         addi $s0, $zero, 1      # s0 = i = 1
102
103 LoopChange:
104         sub $t2, $s3, $s0       # t2 = address(X[n-i])
105         lb $t1, 0($t2)          # t1 = X[n-i]
106         beq $t1, $s4, Exel      # if X[n-i] == ' ', exit LoopChange
107         beq $t2, $s1, Nospace   # if there is no space, goto Nospace
108         addi $s0, $s0, 1        # s0 = s0 + 1 <-> i = i + 1
109         j LoopChange
```

- *LoopChange:* Tìm địa chỉ ký tự space sau cùng, là ký tự ngăn cách giữa Tên và Họ
- Sau đó nhảy đến nhãn Exel
- Nếu không tìm thấy space thì nhảy đến nhãn Nospace

+ *Nhóm lệnh 5:*

```
111 # Sao chep Ten
112 Exel:
113         add $s6, $t2, $zero     # s6 = address(X[wall])
114
115 LoopExel:
116         addi $t2, $t2, 1        # t2 = address(X[wall + i])
117         lb $t3, 0($t2)          # t3 = X[wall + i]
118
119         beq $t3, $s5, Exe2      # if X[wall + i] == "/n", exit Loop
120         sb $t3, 0($s7)          # Result[i-1] = X[wall + i]
121         addi $s7, $s7, 1        # s7 = address(Result[i])
122
123         j LoopExel              #next character
124         nop
```

- *LoopExel:* Sao chép ký tự từ vị trí space sau cùng đến hết chuỗi vào biến Result, hay chính là sao chép tên vào biến Result.
- Sau đó nhảy đến nhãn Exe2

+ *Nhóm lệnh 6*:

```
127 # Sao chep ho
128 Exe2:
129         sb $s4, 0($s7)
130         add $t1, $s1, $zero     # t1 = address(X[i])
131
132 LoopExe2:
133         beq $t1, $s6, EndChange # if address(X[i]) == address(X[wall]), exit LoopExe2
134         lb $t2, 0($t1)          # t2 = X[i]
135
136         addi $s7, $s7, 1        # s7 = Result[k + i]
137         sb $t2, 0($s7)          # Result[k + i]= X[i]
138
139         addi $t1, $t1, 1        # address(X[i + 1])
140
141         nop
142         j LoopExe2              # next character
143
144 EndChange:
145
146
147         addi $s7, $s7, 1        # s7 = Result[End]
148         sb $s5, 0($s7)          #  Ket thuc xau
149
150         li $v0, 4
151         la $a0, Result
152         syscall
153
154         jr $ra
```

- Sao chép ký tự từ vị trí đầu tiên của chuỗi đến vị trí space sau cùng vào biến Result, hay chính là sao chép họ vào Result.
- Sau đó nhảy đến nhãn Endchange
- Thêm ký tự kết thúc xâu và in ra kết quả

+ *Nhóm lệnh 7*:

```
158 # In ra ten
159 Nospace:
160         li $v0, 4
161         add $a0, $s1, $0
162         syscall
163
164         jr $ra
165
```

- Với *$v0 = 4* thì thực hiện chức năng in chuỗi, *$a0* = địa chỉ chuỗi để in
- Trong trường hợp không tìm thấy ký tự space thì chứng tỏ chuỗi chỉ chứa một từ nên sẽ được in ra luôn

BTGK 19:

Mô tả bài toán:

Viết chương trình với đầu vào là một số tên. Tên chỉ bao gồm ký tự tiếng anh, số, dấu gạch dưới và không bắt đầu bằng số. Kiểm tra điều kiện trên với các tên đầu vào.

Phương pháp giải bài toán:

- Kiểm tra ký tự đầu tiên nếu là ký tự tiếng anh, dấu gạch dưới thì tiếp tục kiểm tra các kí tự tiếp theo, nếu không phải thì in ra false
- Kiểm tra từng ký tự còn lại trong chuỗi tên cho đến khi hết chuỗi hoặc gặp ký tự không thỏa mãn điều kiện
- Cách kiểm tra ký tự là ký tự tiếng anh: ‘a’ ≤ c ≤ ‘z’ hoặc ‘A’ ≤ c ≤ ‘Z’
- Cách kiểm tra ký tự là số: ‘0’ ≤ c ≤ ‘9’
- Cách kiểm tra ký tự là dấu gạch dưới: c = ‘_’
- Nếu kiểm tra đến hết chuỗi mà không có ký tự vi phạm thì chuỗi tên đã nhập là đúng

Code C minh họa:

```c
int main()
{
    char str1[50];
    int check = 1;

    while (check == 1)
    {
        printf("name = ");
        fflush(stdin);
        gets(str1);
        printf("name1: %s\n", str1);

        if (str1[0] <= 'z' && str1[0] >= 'a')
            goto CheckRemainingCharacters;
        if (str1[0] <= 'Z' && str1[0] >= 'A')
            goto CheckRemainingCharacters;
        if (str1[0] == '_')
            goto CheckRemainingCharacters;
        printf("variableName(name) = false\n");
        goto choose;
```

```
22      CheckRemainingCharacters:
23          for (int i = 1; i < strlen(str1); i++)
24          {
25              if (str1[i] <= 'z' && str1[i] >= 'a')
26                  continue;
27              if (str1[i] <= 'Z' && str1[i] >= 'A')
28                  continue;
29              if (str1[i] <= '9' && str1[i] >= '0')
30                  continue;
31              if (str1[i] == '_')
32                  continue;
33              printf("variableName(name) = false\n");
34              goto choose;
35          }
36
37          printf("variableName(name) = true\n");
38
39      choose:
40          printf("choose 1 to cotinue, 0 to stop: ");
41          scanf("%d", &check);
42          if (check != 1 && check != 0)
43              goto choose;
44      }
45  }
```

Code MIPS assembly:

```
.data
        title1: .asciiz "\nname = "
        title2: .asciiz "variableName(name) = "
        title3: .asciiz "choose 1 to cotinue, 0 to stop: "
        X: .space 100
        output1: .asciiz "true \n"
        output2: .asciiz "false \n"
        end: .asciiz "\n"

        a: .ascii "`"
        z: .ascii "{"
        aa: .ascii "@"
        zz: .ascii "["
        under: "_"
        num0: .ascii "/"
        num9: .ascii ":"

.text
#-------------------------------------------------
#       s1 = address(X)
#       s2 = "`" là kí tự trước "a" trong bảng asciiz
#       s3 = "{" là kí tự sau "z" trong bảng asciiz
#       s4 = "@" là kí tự trước "A" trong bảng asciiz
#       s5 = "[" là kí tự sau "Z" trong bảng asciiz
#       s6 = "_" là kí tự gạch dưới
#       t8 = "0" là kí tự số không
#       t9 = "9" là kí tự số chín
#       s7 = "\n" là kí tự kết thúc chuỗi
#
#       t2 = variable check head
#       t3 = variable check tail
#       t4 = t2 + t3
#
#
#-------------------------------------------------

input_name:
        li $v0, 4               # In chuoi title1
        la $a0, title1
        syscall

        li $v0, 8               # Doc chuoi name
        la $a0, X
        addi $a1, $zero, 100
        syscall

        li $v0, 4               # Xuong dong
        la $a0, end
        syscall

        li $v0, 4               # In chuoi title2
        la $a0, title2
        syscall
```

```
Initial:
        la $t0, end
        lb $s7, 0($t0)          # s7 = "\n"

        la $s1, X               # s1 = address(X)

        la $t0, a
        lb $s2, 0($t0)          # s2 = "`"

        la $t0, z
        lb $s3, 0($t0)          # s3 = "{"

        la $t0, aa
        lb $s4, 0($t0)          # s4 = "@"

        la $t0, zz
        lb $s5, 0($t0)          # s5 = "["

        la $t0, under
        lb $s6, 0($t0)          # s6 = "_"

        la $t0, num0
        lb $t8, 0($t0)          # t8 = "0"

        la $t0, num9
        lb $t9, 0($t0)          # t9 = "9"
```

```
# Kiểm tra kí tự đầu tiên
CheckFirstCharacter:
        lb $t1, 0($s1)          # t1 = kí tự đầu tiên
        jal CheckLetter1        # kiểm tra kí tự có thuộc A-Z hay không
        jal CheckLetter2        # kiểm tra kí tự có thuộc a-z hay không
        jal CheckUnderscores    # kiểm tra kí tự có là "_" hay không
        j Done2                 # Nếu chuỗi bắt đầu bằng số hoặc kì tự đặc biệt (trừ "_")
                                # thì sẽ in ra false
```

```
# Kiem tra từng kí tự trong các kí tự còn lại
CheckRemainingCharacters:
        addi $s1, $s1, 1
        lb $t1, 0($s1)          # t1 = kí tự tiếp theo
        beq $t1, $s7, Done1     # Nếu t1 = s7 = "\n" => kết thúc chuỗi
        jal CheckNumber         # Kiểm tra kí tự có phải số hay không
        jal CheckLetter1        # kiểm tra kí tự có thuộc A-Z hay không
        jal CheckLetter2        # kiểm tra kí tự có thuộc a-z hay không
        jal CheckUnderscores    # kiểm tra kí tự có là "_" hay không
        j Done2
```

```
# Kiểm tra xem kí tự = (0->9)
CheckNumber:
        slt $t2, $t8, $t1       # Nếu t8 < t1 (0 < X[i]) thì t2 = 1
        slt $t3, $t1, $t9       # Nếu t1 < t9 (X[i] < 9) thì t3 = 1
        add $t4, $t2, $t3       # t4 = t2 + t3
        j CheckGeneral
```

```
# Kiểm tra xem kí tự = (A->Z)
CheckLetter1:
        slt $t2, $s4, $t1       # Nếu s4 < t1 (A < X[i]) thì t2 = 1
        slt $t3, $t1, $s5       # Nếu t1 < s5 (X[i] < Z) thì t3 = 1
        add $t4, $t2, $t3       # t4 = t2 + t3
        j CheckGeneral

```

```
118 # Kiểm tra xem kí tự = (a->z)
119 CheckLetter2:
120         slt $t2, $s2, $t1       # Nếu s2 < t1 (a < X[i]) thì t2 = 1
121         slt $t3, $t1, $s3       # Nếu t1 < s3 (X[i] < z) thì t3 = 1
122         add $t4, $t2, $t3       # t4 = t2 + t3
123         j CheckGeneral
```

```
125 # Kiem tra xem ki tu = (_)
126 CheckUnderscores:
127         beq $t1, $s6, CheckRemainingCharacters          # Nếu t1 = "_" thì đến kí tự tiếp theo
128         jr $ra
129
```

```
130 # hàm kiểm tra tính đúng sai của các hàm check bên trên
131 CheckGeneral:
132         beq $t4, 2, CheckRemainingCharacters            # Nếu t4 = 2 thì đến kí tự tiếp theo
133         jr $ra
134
```

```
135 # In ra thông báo true
136 Done1:
137         li $v0, 4
138         la $a0, output1
139         syscall
140         j Continue
141
142 # In ra thông báo false
143 Done2:
144         li $v0, 4
145         la $a0, output2
146         syscall
147         j Continue
```

```
149 # Tiếp tục hay không
150 Continue:
151
152         li $v0, 4               # Xuong dong
153         la $a0, end
154         syscall
155
156         li $v0, 4               # In chuoi title3
157         la $a0, title3
158         syscall
159
160         li $v0, 5               # Đọc số nguyên
161         syscall
162
163         beq $v0, 1, input_name  # nếu v0 = 1 thì tiếp tục
164         beq $v0, 0, Endpro      # nếu v0 = 0 thì dừng lại
165         j Continue              # nếu nhập lựa chọn khác thì yêu cầu nhập lại
166
```

```
167
168 # kết thúc chương trình
169 Endpro:
170         li $v0, 10
171         syscall
172
```

Giải thích code:

```
.data
        title1: .asciiz "\nname = "
        title2: .asciiz "variableName(name) = "
        title3: .asciiz "choose 1 to cotinue, 0 to stop: "
        X: .space 100
        output1: .asciiz "true \n"
        output2: .asciiz "false \n"
        end: .asciiz "\n"

        a: .ascii "`"
        z: .ascii "{"
        aa: .ascii "@"
        zz: .ascii "["
        under: "_"
        num0: .ascii "/"
        num9: .ascii ":"

.text
#-------------------------------------------------
#       s1 = address(X)
#       s2 = "`" là kí tự trước "a" trong bảng asciiz
#       s3 = "{" là kí tự sau "z" trong bảng asciiz
#       s4 = "@" là kí tự trước "A" trong bảng asciiz
#       s5 = "[" là kí tự sau "Z" trong bảng asciiz
#       s6 = "_" là kí tự gạch dưới
#       t8 = "0" là kí tự số không
#       t9 = "9" là kí tự số chín
#       s7 = "\n" là kí tự kết thúc chuỗi
#
#       t2 = variable check head
#       t3 = variable check tail
#       t4 = t2 + t3
#
#
#-------------------------------------------------
```

=> Trong ảnh là phần khai báo biến và tác dụng của các thanh ghi

- Các nhóm lệnh trong vùng lệnh hợp ngữ
  + *Nhóm lệnh 1:*

```
38  input_name:
39          li $v0, 4               # In chuoi title1
40          la $a0, title1
41          syscall
42
43          li $v0, 8               # Doc chuoi name
44          la $a0, X
45          addi $a1, $zero, 100
46          syscall
47
48          li $v0, 4               # Xuong dong
49          la $a0, end
50          syscall
51
52          li $v0, 4               # In chuoi title2
53          la $a0, title2
54          syscall
```

- với *$v0 = 4* thì thực hiện chức năng in chuỗi, *$a0* = địa chỉ chuỗi để in
- Với *$v0 = 8* thì thực hiện chức năng đọc chuỗi, *$a0* = địa chỉ của bộ đệm đầu vào, $a1 = số ký tự tối đa để đọc
- Dòng 39 - 46: Đọc tên chuỗi đầu vào
- Dòng 52 - 54: In ra title “variableName(name) = ”

  + *Nhóm lệnh 2*:

```
57  Initial:
58          la $t0, end
59          lb $s7, 0($t0)          # s7 = "\n"
60
61          la $s1, X               # s1 = address(X)
62
63          la $t0, a
64          lb $s2, 0($t0)          # s2 = "`"
65
66          la $t0, z
67          lb $s3, 0($t0)          # s3 = "{"
68
69          la $t0, aa
70          lb $s4, 0($t0)          # s4 = "@"
71
72          la $t0, zz
73          lb $s5, 0($t0)          # s5 = "["
74
75          la $t0, under
76          lb $s6, 0($t0)          # s6 = "_"
77
78          la $t0, num0
79          lb $t8, 0($t0)          # t8 = "0"
80
81          la $t0, num9
82          lb $t9, 0($t0)          # t9 = "9"
```

- la: load address (Gán địa chỉ của biến vào thanh ghi)
- lb: load byte (Lệnh này nạp giá trị byte nhớ có địa chỉ ($t0 + 0) vào byte thấp của thanh ghi được chỉ định)

+ *Nhóm lệnh 3:*

```
84  # Kiểm tra kí tự đầu tiên
85  CheckFirstCharacter:
86          lb $t1, 0($s1)          # t1 = kí tự đầu tiên
87          jal CheckLetter1        # kiểm tra kí tự có thuộc A-Z hay không
88          jal CheckLetter2        # kiểm tra kí tự có thuộc a-z hay không
89          jal CheckUnderscores    # kiểm tra kí tự có là "_" hay không
90          j Done2                 # Nếu chuỗi bắt đầu bằng số hoặc kì tự đặc biệt (trừ "_")
91                                  # thì sẽ in ra false
```

- lb: load byte (Lệnh này nạp giá trị byte nhớ có địa chỉ ($s1 + 0) vào byte thấp của thanh ghi $t1)
- jal: jump and link
- Sử dụng lệnh jal để nhảy đến lệnh ở các nhãn được chỉ định
- Dòng 87: Kiểm tra ký tự đang xét có phải chữ hoa hay không
- Dòng 88: Kiểm tra ký tự đang xét có phải chữ thường hay không
- Dòng 89: Kiểm tra ký tự đang xét có phải dấu gạch dưới hay không
- Dòng 90: Nếu đều không phải 3 loại trên thì ngay lập tức in ra false

+ *Nhóm lệnh 4:*

```
93  # Kiem tra từng kí tự trong các kí tự còn lại
94  CheckRemainingCharacters:
95          addi $s1, $s1, 1
96          lb $t1, 0($s1)          # t1 = kí tự tiếp theo
97          beq $t1, $s7, Done1     # Nếu t1 = s7 = "\n" => kết thúc chuỗi
98          jal CheckNumber         # Kiểm tra kí tự có phải số hay không
99          jal CheckLetter1        # kiểm tra kí tự có thuộc A-Z hay không
100         jal CheckLetter2        # kiểm tra kí tự có thuộc a-z hay không
101         jal CheckUnderscores    # kiểm tra kí tự có là "_" hay không
102         j Done2
```

- CheckRemainingCharacters: Kiểm tra từng ký tự cho đến khi kết thúc xâu hoặc tìm được ký tự không thích hợp
- Dòng 97: Nếu ký tự đang xét là “\n” tức kết thúc chuỗi thì in ra true
- Dòng 98: Kiểm tra ký tự đang xét có phải số hay không
- Dòng 99: Kiểm tra ký tự đang xét có phải chữ Hoa hay không
- Dòng 100: Kiểm tra ký tự đang xét có phải chữ thường hay không
- Dòng 101: Kiểm tra ký tự đang xét có phải dấu gạch dưới hay không
- Dòng 102: Nếu đều không phải 3 loại trên thì ngay lập tức in ra false

+ *Nhóm lệnh 5:*

```
104 # Kiểm tra xem kí tự = (0->9)
105 CheckNumber:
106         slt $t2, $t8, $t1       # Nếu t8 < t1 (0 < X[i]) thì t2 = 1
107         slt $t3, $t1, $t9       # Nếu t1 < t9 (X[i] < 9) thì t3 = 1
108         add $t4, $t2, $t3       # t4 = t2 + t3
109         j CheckGeneral
```

- Đây là code để kiểm tra ký tự có phải số hay không
- slt: set less than (Nếu $t8 < $t1 thì $t2 = 1 ngược lại thì $t2 = 0)
- Nếu $t4 = 2 thì tức là ký tự đang xét nằm trong khoảng 0->9
- Sau đó nhảy đến lệnh ở nhãn CheckGeneral để kiểm tra giá trị của $t4

+ *Nhóm lệnh 6:*

```
111 # Kiểm tra xem kí tự = (A->Z)
112 CheckLetter1:
113         slt $t2, $s4, $t1       # Nếu s4 < t1 (A < X[i]) thì t2 = 1
114         slt $t3, $t1, $s5       # Nếu t1 < s5 (X[i] < Z) thì t3 = 1
115         add $t4, $t2, $t3       # t4 = t2 + t3
116         j CheckGeneral
117
```

- Đây là code để kiểm tra ký tự có phải chữ hoa hay không
- slt: set less than (Nếu $s4 < $t1 thì $t2 = 1 ngược lại thì $t2 = 0)
- Nếu $t4 = 2 thì tức là ký tự đang xét nằm trong khoảng A->Z
- Sau đó nhảy đến lệnh ở nhãn CheckGeneral để kiểm tra giá trị của $t4

+ *Nhóm lệnh 7:*

```
118 # Kiểm tra xem kí tự = (a->z)
119 CheckLetter2:
120         slt $t2, $s2, $t1       # Nếu s2 < t1 (a < X[i]) thì t2 = 1
121         slt $t3, $t1, $s3       # Nếu t1 < s3 (X[i] < z) thì t3 = 1
122         add $t4, $t2, $t3       # t4 = t2 + t3
123         j CheckGeneral
```

- Đây là code để kiểm tra ký tự có phải chữ thường hay không
- slt: set less than (Nếu $s4 < $t1 thì $t2 = 1 ngược lại thì $t2 = 0)
- Nếu $t4 = 2 thì tức là ký tự đang xét nằm trong khoảng a->z
- Sau đó nhảy đến lệnh ở nhãn CheckGeneral để kiểm tra giá trị của $t4

+ *Nhóm lệnh 8:*

```
125 # Kiem tra xem ki tu = (_)
126 CheckUnderscores:
127         beq $t1, $s6, CheckRemainingCharacters          # Nếu t1 = "_" thì đến kí tự tiếp theo
128         jr $ra
129
```

- beq: branch if equal (Nếu $t1 = $s6 thì nhảy đến lệnh ở nhãn CheckRemainingCharacters)
- jr: jump register unconditionally (Nhảy quay lại lệnh jal sử dụng gần nhất)

+ *Nhóm lệnh 9:*

```
130 # hàm kiểm tra tính đúng sai của các hàm check bên trên
131 CheckGeneral:
132         beq $t4, 2, CheckRemainingCharacters            # Nếu t4 = 2 thì đến kí tự tiếp theo
133         jr $ra
134
```

- Nếu $t4 = 2 thì nhảy quay lại CheckRemainingCharacters để kiểm tra ký tự tiếp theo
- Nếu $t4 khác 2 thì quay lại lệnh jal sử dụng gần nhất

+ *Nhóm lệnh 10:*

```
135 # In ra thông báo true
136 Done1:
137         li $v0, 4
138         la $a0, output1
139         syscall
140         j Continue
141
142 # In ra thông báo false
143 Done2:
144         li $v0, 4
145         la $a0, output2
146         syscall
147         j Continue
```

- Dòng 137 - 140: In ra thông báo “true” và nhảy đến lệnh ở nhãn Continue
- Dòng 144 - 147: In ra thông báo “false” và nhảy đến lệnh ở nhãn Continue

+ *Nhóm lệnh 11:*

```
149 # Tiếp tục hay không
150 Continue:
151
152         li $v0, 4               # Xuong dong
153         la $a0, end
154         syscall
155
156         li $v0, 4               # In chuoi title3
157         la $a0, title3
158         syscall
159
160         li $v0, 5               # Đọc số nguyên
161         syscall
162
163         beq $v0, 1, input_name  # nếu v0 = 1 thì tiếp tục
164         beq $v0, 0, Endpro      # nếu v0 = 0 thì dừng lại
165         j Continue              # nếu nhập lựa chọn khác thì yêu cầu nhập lại
166
```

- Continue: Nhóm lệnh để hỏi người dùng có tiếp tục nhập và kiểm tra tên mới hay không
- Với $v0 = 5 thực hiện chức năng đọc số nguyên và $v0 = số nguyên nhập vào
- Nếu người dùng nhập 1 thì tiếp tục chương trình, nếu là không thì dừng, nếu là số nguyên khác thì yêu cầu nhập lại

+ *Nhóm lệnh 12:*

```
167
168 # kết thúc chương trình
169 Endpro:
170         li $v0, 10
171         syscall
172
```

- Endpro: Với $v0 = 10 thì kết thúc chương trình