# BÁO CÁO MINI-PROJECTS THỰC HÀNH KTMT

# Project 5+12

Họ tên: Lê Cao Phong Họ tên: Đinh Thị Hồng Phúc

MSSV: 20215113 MSSV: 20215118

## **Project 5:**

### Code

```
input_mes: .asciiz "Input: "
2
          bin mes: .asciiz "Binary: "
3
         string_bin: .space 32
4
         hex mes: .asciiz "\nHeximal: "
5
          string_hex: .space 8
 6
7 .text
          li $v0, 4
8
          la $a0, input_mes
9
          syscall
10
         li $v0, 5
11
         syscall
13
         la $t0, string_bin
         add $t1, $0, 0x1
         add $s1, $s0, 49 #Ascii 1
add $s2, $s0, 48 #Ascii 0
17 convert_binary:
18 and $t2, $t1, $v0
                              #convert to binary
         beqz $t2, its_zero
19
         sb $s1, 0($t0) #store 1 to address $t0
     j condition
22 its_zero: sb $s2, 0($t0)
                             #store 2 to address $t0
23 condition:
24 add $t0, $t0, 1
25
         sll $t1, $t1, 1
                              #dich trai $t1 1 bit
26
        beq $t1, 0, print_binmes
         j convert binary
```

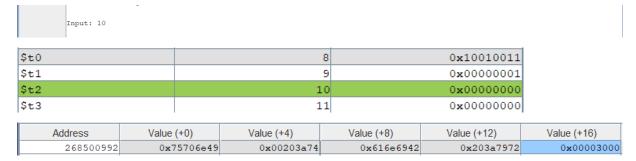
```
28 print_binmes:
    li $v0, 4
29
30
         la $a0, bin_mes
                              #print Binary
31
          syscall
         la $s0, string_bin
33
34
         li $v0, 11
         add $t0, $t0, -1
         add $t5, $0, $t0
                             #$t5 luu dia chi cua bit cuoi cung duoc luu o $t0
36 print_binary:
         lb $a0, 0($t0)
                             #print after converted to Binary
37
38
         syscall
          beq $t0, $s0, end_binary
40
         add $t0, $t0, -1
         j print binary
41
42 end_binary:
     li $v0, 4
43
44
45
         la $a0, hex_mes
                             #print Heximal
         syscall
        la $s0, string_bin
li $v0, 11
```

```
48 convert_hex:
           1b $t1, 0($t5) #lay ra 4 bit duoc luu cuoi cung tai $t0
49
           1b $t2, -1($t5)
50
           1b $t3, -2($t5)
51
52
           1b $t4, -3($t5)
53
           add $t1, $t1, -48
                                 #chuyen chung ve 0 va 1
           add $t2, $t2, -48
54
           add $t3, $t3, -48
55
56
           add $t4, $t4, -48
           mul $t1, $t1, 8
57
58
           mul $t2, $t2, 4
           mul $t3, $t3, 2
59
           mul $t4, $t4, 1
61
           add $t1, $t1, $t2
62
           add $t3, $t3, $t4
           add $t1, $t1, $t3
                                 #$t1= gia tri o he 10 cua 4 bit duoc lav ra
63
           blt $t1, 10, normal
65
           add $a0, $t1, 55
                                 #$t1>=10 thi chuven thanh A.B....F ASCII
66
           j print_hex
67 normal: add $a0, $t1, 48
                                  #chuyen ve ASCII
68 print hex:
69
           syscall
           add $t5, $t5, -4
70
71
           bge $t5, $s0, convert hex
                                         #$t5<$s0 ket thuc vong lap
           li $v0, 10
72 end:
                                 #exit
           syscall
73
```

#### Giải thích

- Chương trình thực hiện yêu cầu nhập vào một số nguyên dạng thập phân sau đó in ra số đó dưới dạng nhị phân và hệ 16. Thuật toán sử dụng để chuyển về hệ nhị phân đó là and từng bit của số đó với 1.
- Đầu tiên gán các giá trị \$t1=0x1, \$s1=49 (mã ASCII của kí tự '1') \$s2=48 (mã ASCII của kí tự '0').
- Tiến vào vòng lặp **convert\_binary**: thực hiện and \$v0 (Input) với \$t1 lưu giá trị vào \$t2 chính là lấy từng bit của \$v0 ra. Nếu \$t2!=1 thì lưu giá trị 1 vào địa chỉ \$t0 (\$t0 ban đầu lưu địa chỉ của string\_bin), nếu \$t2=0 thì lưu 0 vào đia chỉ \$t0.

Với đầu vào Input=10=0x1010 trong lần lặp đầu tiên



Tại địa chỉ \$t0 đã lưu 0x3000 tương đương với 48 mã ASCII 0.

- Trong **condition** tăng \$t0 lên 1 để lưu bit tiếp theo bit vừa được lưu của \$v0. Dịch trái \$t1 đi 1 bit và thực hiện lại **convert\_binary** để lấy ra bit

tiếp theo của \$v0. Nếu \$t1==0 thì chứng tỏ đã dịch trái hết 32 bits và đã lấy ra được tất cả các bit của \$v0 nên chuyển tới **print\_binmes.**Khi \$t1=0 bắt đầu in các bit lần lượt ra

\$t1	9	0x00000000
\$t2	10	0x00000000

Trong **print\_binmes**, **print\_binary** thực hiện in ra "Binary: " và lần lượt các bit từ chuỗi nhị phân đã chuyển đổi từ bit cuối lên bit đầu tiên. Dòng 39 là lệnh điều kiện để kiểm tra xem đã duyệt hết chuỗi nhị phân chưa. Nếu đã duyệt hết, nhảy tới **end\_binary** để kết thúc quá trình in. Nếu chưa duyệt hết thì giảm giá trị \$t0 đi 1 rồi nhảy tới **print\_binary** để tiếp tục vòng lặp in các bit của chuỗi nhị phân.

- Trong **end\_binary** thực hiện in chuỗi "\nHeximal: " lên màn hình, chuẩn bị in số hệ 16 từ chuỗi nhị phân đã chuyển đổi.
- Trong **convert\_hex** thực hiện chuyển đổi từ 4 bit cuối cùng của chuỗi nhị phân sang hệ 16 và in giá trị này ra màn hình. Bằng cách đọc giá trị các byte từ cuối lên đầu từ địa chỉ con trỏ \$t5 lưu vào lần lượt các thanh ghi \$t1, \$t2, \$t3, \$t4.

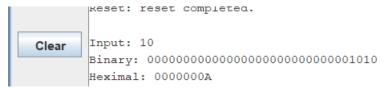
Với lần lấy ra 4 bit đầu tiên với Input=10

\$t1	9	0x00000030
\$t2	10	0x00000030
\$t3	11	0x00000030
\$t4	12	0x00000030

Thấy các thanh ghi đều có chung là giá trị của số 0 ở mã ASCII

- Sau đó chuyển đổi giá trị ASCII của kí tự thành giá trị số bằng cách trừ đi 48 (mã ASCII của kí tự '0'). Từ dòng 57-60, thực hiện nhân các giá trị trong thanh ghi \$t1, \$t2, \$t3, \$t4 với các hệ số tương ứng để tính giá trị số hệ 16 từ 4 bit cuối cùng của chuỗi nhị phân. Sau đó cộng tổng giá trị của các bit đã tính được và lưu trong \$t1 (giá trị số hệ thập phân)

- Kiểm tra giá trị của \$t1 nhỏ hơn 10 thì nhảy tới **normal**, nếu không thì chuyển thành kí tự hệ 16 tương ứng (A, B, ..., F) bằng cách cộng với 55 và tiếp tục tại **print\_hex**.
- Trong normal chuyển đổi giá trị số (từ 0 đến 9) sang kí tự ASCII. Vì \$t1 có giá trị từ 0 đến 9, mã ASCII sẽ từ 48 đến 57 nên chỉ cần cộng 48 để được mã ASCII tương ứng.
- Trong **print\_hex** thực hiện in từng kí tự hệ 16 tương ứng với 4 bit cuối của chuỗi nhị phân ra màn hình. Con trỏ \$t5 được di chuyển trở lại 4 byte để lấy 4 bit tiếp theo từ chuỗi nhị phân và tiếp tục quá trình chuyển đổi, in. Nếu \$t5 (con trỏ đến vị trí cuối cùng) không nhỏ hơn \$s0 (địa chỉ đầu chuỗi) nghĩa là chưa chuyển đổi tất cả các bit, chương trình sẽ nhảy đến **convert\_hex** để tiếp tục chuyển đổi và in các bit tiếp theo thành hệ 16.



Kết quả khi in xong toàn bộ

- Trong **end** thực hiện kết thúc và thoát khỏi chương trình.

# Kết quả

Với đầu vào Input=512 ta được

Input: 512

Binary: 000000000000000000000001000000000

Heximal: 00000200

-- program is finished running --

## **Project 12:**

### Code

```
1 .data
           string: .space 100
 3 Input: .asciiz "Input: "
 4 Output: .asciiz "Output: "
   .text
 6 main:
           li $v0, 4
           la $aO, Input
8
9
           syscall
           li $v0, 8
10
           la $aO, string
11
       li $a1, 100
12
           syscall # input string
13
           jal a_to_i
14
           li $v0, 4
15
           la $aO, Output
16
17
           syscall
           li $v0, 1
18
           la $a0, ($s0)
19
           syscall # print number
20
           li $v0, 10
21
22
           syscall # terminate
23
24 # Funtion a to i: starting address of the string will be in $a0
25 # and return a 32-bit interger in $s0
26
27 a_to_i:
28
           la $t2, ($a0) # string address
           addi $t0, $a0, -1
29
```

```
30 find_end: # find the end of string
           addi $t0, $t0, 1
31
32
           lb $t1, 0($t0)
           bne $t1, 10, find_end
33
34
           li $v0, 1 # 10^i
35
           li $s0, 0 # number
36 loop:
37
           add $t0, $t0, -1
           lb $s1, 0($t0)
38
           addi $s1, $s1, -48 # ASCII to interger
39
           mul $s1, $s1, $v0 # $s1 . 10^i
40
```

```
41 add $s0, $s0, $s1 # $a0 = $a0 + $s1

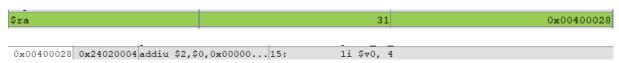
42 mul $v0, $v0, 10 # 10^(i+1)

43 bne $t0, $t2, loop

44 jr $ra
```

### Giải thích

- Project 12 trình bày thuật toán in ra màn hình một số nguyên từ đầu vào là một chuỗi số sử dụng jal, con trỏ \$ra để quay trở về dòng lệnh ngay sau vòng lặp.
- Đầu tiên khởi tạo các thông báo cho người sử dụng "Input" và "Output"
- Trong hàm **main** sử dụng syscall để in ra các thông báo và cho người dùng nhập vào một chuỗi số.
- Sử dụng jal để rẽ nhánh tới a\_to\_i con trỏ \$ra ghi nhớ địa chỉ của dòng lệnh ngay dưới lệnh jal (dòng lệnh 15)



- Trong **a\_to\_i** thực hiện: lưu địa chỉ của \$a0 (phần tử đầu tiên) vào thanh ghi \$t2 và \$t0=\$a0-1.
- Tiến vào vòng lặp **find\_end**: \$t0=\$t0+1 nên lúc này thanh ghi \$t0 đang lưu địa chỉ phần tử đầu tiên của chuỗi. Từ địa chỉ đó lấy giá trị lưu vào \$t1. Kiểm tra điều kiện để kết thúc vòng lặp. Nếu \$t1==10 tức giá trị NULL (phần tử cuối cùng của xâu) thì dừng vòng lặp chuyển tới **loop.**Nếu \$t1!=10 thì tiếp tục vòng lặp với phần tử thứ 2, 3,... của xâu. Trước khi vào vòng lặp loop thì khởi tạo \$v0=1=i và \$s0=0.
- Trong **loop**: đầu tiên \$t0=\$t0-1, lúc này \$t0 đang lưu giữ địa chỉ của phần tử ngay trước phần tử NULL tức số cuối cùng trong chuỗi số nhập từ Input. Từ địa chỉ này lưu giá trị vào \$s1. \$s1=\$s1-48 (ban đầu các chữ số đang trong hệ ASCII nên giá trị sẽ không giống giá trị của chữ số cần -48 để về với giá trị của chữ số). Sau đó thực hiện \$s1=\$s1x10<sup>i</sup>, \$s0=\$s0+1. Kiểm tra \$t0==\$t2 hay chưa (kiểm tra xem vòng lặp đã lặp tới phần tử đầu tiên của chuỗi số chưa) nếu đúng kết thúc vòng lặp sử dụng \$ra để quay về địa chỉ của dòng lệnh đã lưu.
- Dòng lệnh từ 15 tới 22 in ra Output và giúp thoát chương trình.

- Đầu vào



Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)
268500992	9 8 6 3	\0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0
268501024	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0
268501056	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0

- Vòng lặp loop sau khi thực hiện lần lặp đầu tiên:



- Vậy khi vòng lặp loop sau khi thực hiện lần lặp cuối cùng

\$s0	16	3689
\$s1	17	3000

- Đầu ra:

```
Input: 3689
Output: 3689
-- program is finished running --
```