

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

```
1  .data
2      input_mes: .ascii "Input: "
3      bin_mes: .ascii "Binary: "
4      string_bin: .space 32
5      hex_mes: .ascii "\nHeximal: "
6      string_hex: .space 8
7  .text
8      li $v0, 4
9      la $a0, input_mes
10     syscall
11     li $v0, 5
12     syscall
13     la $t0, string_bin
14     add $t1, $0, 0x1
15     add $s1, $s0, 49      #Ascii 1
16     add $s2, $s0, 48      #Ascii 0
17 convert_binary:
18     and $t2, $t1, $v0     #convert to binary
19     beqz $t2, its_zero
20     sb $s1, 0($t0)        #store 1 to address $t0
21     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
27     j convert_binary
28 print_binmes:
29     li $v0, 4
30     la $a0, bin_mes       #print Binary
31     syscall
32     la $s0, string_bin
33     li $v0, 11
34     add $t0, $t0, -1
35     add $t5, $0, $t0      #$t5 lưu địa chỉ của bit cuối cùng được lưu ở $t0
36 print_binary:
37     lb $a0, 0($t0)        #print after converted to Binary
38     syscall
39     beq $t0, $s0, end_binary
40     add $t0, $t0, -1
41     j print_binary
42 end_binary:
43     li $v0, 4
44     la $a0, hex_mes       #print Heximal
45     syscall
46     la $s0, string_bin
47     li $v0, 11
```

```

48 convert_hex:
49     lb $t1, 0($t5) #lay ra 4 bit duoc luu cuoi cung tai $t0
50     lb $t2, -1($t5)
51     lb $t3, -2($t5)
52     lb $t4, -3($t5)
53     add $t1, $t1, -48 #chuyen chung ve 0 va 1
54     add $t2, $t2, -48
55     add $t3, $t3, -48
56     add $t4, $t4, -48
57     mul $t1, $t1, 8
58     mul $t2, $t2, 4
59     mul $t3, $t3, 2
60     mul $t4, $t4, 1
61     add $t1, $t1, $t2
62     add $t3, $t3, $t4
63     add $t1, $t1, $t3 # $t1= gia tri o he 10 cua 4 bit duoc lay ra
64     blt $t1, 10, normal
65     add $a0, $t1, 55 # $t1 >= 10 thi chuyen thanh A,B,...F ASCII
66     j print_hex

67 normal: add $a0, $t1, 48 #chuyen ve ASCII
68 print_hex:
69     syscall
70     add $t5, $t5, -4
71     bge $t5, $s0, convert_hex # $t5 < $s0 ket thuc vong lap
72 end: li $v0, 10 #exit
73     syscall

```

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 địa chỉ \$t0.

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

Input: 10					
\$t0	8	0x10010011			
\$t1	9	0x00000001			
\$t2	10	0x00000000			
\$t3	11	0x00000000			

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)
268500992	0x75706e49	0x00203a74	0x616e6942	0x203a7972	0x00003000

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

- [illegible]

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

Với đầu vào Input=512 ta được

```
Input: 512
Binary: 0000000000000000000000001000000000
Heximal: 00000200
-- program is finished running --
```

Project 12:

Code

```
1  .data
2      string: .space 100
3  Input: .asciiz "Input: "
4  Output: .asciiz "Output: "
5  .text
6  main:
7      li $v0, 4
8      la $a0, Input
9      syscall
10     li $v0, 8
11     la $a0, string
12     li $a1, 100
13     syscall # input string
14     jal a_to_i
15     li $v0, 4
16     la $a0, Output
17     syscall
18     li $v0, 1
19     la $a0, ($s0)
20     syscall # print number
21     li $v0, 10
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
29         addi $t0, $a0, -1
30
31     find_end: # find the end of string
32         addi $t0, $t0, 1
33         lb $t1, 0($t0)
34         bne $t1, 10, find_end
35         li $v0, 1 # 10^i
36         li $s0, 0 # number
37     loop:
38         add $t0, $t0, -1
39         lb $s1, 0($t0)
40         addi $s1, $s1, -48 # ASCII to interger
41         mul $s1, $s1, $v0 # $s1 . 10^i
42
43         add $s0, $s0, $s1 # $a0 = $a0 + $s1
44         mul $v0, $v0, 10 # 10^(i+1)
45         bne $t0, $t2, loop
46         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)

\$ra	31	0x00400028
0x00400028	0x24020004	addiu \$2,\$0,0x00000...15: 1i \$v0, 4

- 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=\$s1x10ⁱ, \$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.

Kết quả

- Đầu vào:

Input: 3689

Clear

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

\$s0	16	9
\$s1	17	9

- 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 --

Clear