

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

\*\*\*\*\*



# **BÁO CÁO**

**Bài tập thực hành giữa kỳ**

**Học phần: Thực hành kiến trúc máy tính**

**Giảng viên hướng dẫn: Lê Bá Vui**

**Sinh viên thực hiện: Nguyễn Bình An - 20225591**

**Hà Nội, tháng 4 năm 2024**

## Assignment 1

### Code:

#assignment 1: 10\_Nhập số nguyên dương N từ bàn phím, in ra tổng các chữ số trong biểu diễn nhị phân của N.

#author: Nguyễn Bình An\_20225591

.data

INPUT: .ascii "Nhập vào số nguyên N "

result: .ascii "Tổng số bit: "

.text

main:

# Prompt for input

li \$v0, 4

la \$a0, INPUT

syscall

# Read integer input

li \$v0, 5

syscall

move \$a1, \$v0 # Move read integer to \$a1

jal decimalToBinary # Call the function to process the number

# Print the total '1' bits counted

li \$v0, 4

la \$a0, result

syscall

li \$v0, 1

move \$a0, \$s0

syscall

# Exit the program

li \$v0, 10

syscall

decimalToBinary:

# Initialize registers

move \$t2, \$a1 # Move input number into \$t2 for processing

```

move $s0, $zero    # Initialize $s0 to count the number of '1' bits

# Loop to process each bit of the input number
li $t0, 32          # Prepare to process 32 bits
loop1:
    andi $t3, $t2, 1    # Isolate the last bit of $t2
    beq $t3, $zero, skip_increment # Skip incrementing $s0 if the bit is '0'
    addi $s0, $s0, 1    # Increment the result variable if bit is '1'
skip_increment:
    srl $t2, $t2, 1     # Shift right $t2 to get the next bit
    addi $t0, $t0, -1    # Decrement the bit index
    bgtz $t0, loop1     # Continue if there are more bits to process

jr $ra              # Return from the function

```

### **Giải thích code:**

**\$a0** : tham số đầu vào

**\$v0** : giá trị trả về của chương trình con

**\$a0** : lưu lại kết quả của bài toán

- Gọi hàm decimalToBinary: Số nguyên đã nhập được truyền sang hàm decimalToBinary để xử lý. Hàm này được gọi bằng lệnh **jal decimalToBinary**.
- Xử lý trong hàm:
  - Khởi tạo: Số nhập vào được chuyển từ \$a1 sang \$t2. Thanh ghi \$s0 được khởi tạo bằng 0 để dùng đếm số bit 1.
  - Vòng lặp xử lý bit: Trong mỗi lần lặp, bit cuối cùng của \$t2 được kiểm tra. Nếu bit này là 1 (**andi \$t3, \$t2, 1** cho ra kết quả 1), giá trị trong \$s0 được tăng lên 1. Sau đó, \$t2 được dịch phải một bit để chuẩn bị bit tiếp theo (**srl \$t2, \$t2, 1**).
  - Tiếp tục vòng lặp: Quá trình này tiếp tục cho đến khi đã xử lý đủ 32 bit.

Kết thúc hàm: Sau khi đã xử lý xong tất cả các bit, hàm trả về và quay lại chương trình chính.

### **Chạy thử nghiệm:**

Với **\$a0** = 11



```

main:
    # Input the number of elements in the array
    li $v0, 4
    la $a0, size
    syscall

    li $v0, 5
    syscall
    move $t0, $v0      # $t0 is the number of elements in the array

    # Input the array elements
    li $v0, 4
    la $a0, elements
    syscall

    la $t1, array      # $t1 is the base address of the array
    li $t2, 0          # $t2 is the loop counter

input_loop:
    blt $t2, $t0, read_element # Check loop condition
    j process
read_element:
    li $v0, 5
    syscall
    sw $v0, 0($t1)      # Store the input value into the array
    addiu $t1, $t1, 4    # Increment array address
    addiu $t2, $t2, 1    # Increment loop counter
    j input_loop

process:
    # Input M and N
    li $v0, 4
    la $a0, M
    syscall
    li $v0, 5
    syscall
    move $t3, $v0      # $t3 is M

    li $v0, 4
    la $a0, N
    syscall
    li $v0, 5
    syscall
    move $t4, $v0      # $t4 is N

```

```

# Count elements in the range (M, N)
li $t5, 0          # $t5 is the counter
la $t1, array      # Reset the address of the array
li $t2, 0          # Reset the loop counter

count_loop:
    bge $t2, $t0, print_result
    lw $t6, 0($t1)
    addiu $t1, $t1, 4
    addiu $t2, $t2, 1

    ble $t6, $t3, count_loop # Check if the element is in the range (M, N)
    bge $t6, $t4, count_loop
    addiu $t5, $t5, 1
    j count_loop

print_result:
    li $v0, 4
    la $a0, result_msg
    syscall

    li $v0, 1
    move $a0, $t5
    syscall

    li $v0, 10
    syscall

```

### **Giải thích code:**

**\$t1** : Lưu địa chỉ các phần tử của mảng

**\$t3,t4** : Lưu các giá trị M,N

**\$v0** : chứa giá trị trả về của hàm count\_loop()

Sử dụng **jal** để nhảy đến lệnh của chương trình con **\$ra** sẽ chứa địa chỉ quay lại sau khi chương trình con kết thúc.

Khi chương trình con đã hoàn thành sử dụng **jr \$ra** để quay lại lệnh tiếp của chương trình chính.

- Nhập dữ liệu:

- Khởi động thanh ghi \$v0, đưa giá trị số lượng các phần tử của mảng vào thanh ghi \$t0

Nhap so phan tu cua mang: 5	\$t0	8	5
-----------------------------	------	---	---

- Tiếp tục đưa giá trị các phần tử của mảng vào bằng thanh ghi \$v0, thanh ghi \$t1 sẽ chứa các địa chỉ của mảng:

\$t1	9	0x1001000c
------	---	------------

Nhap vao cac phan tu: 8

6

7

\$v0	2	0x00000007
------	---	------------

- Nhập vào địa chỉ của M, N vào thanh ghi \$t3, \$t4 qua câu lệnh:

```
li $v0, 4
la $a0, M
syscall
li $v0, 5
syscall
move $t3, $v0
li $v0, 4
la $a0, N
syscall
li $v0, 5
syscall
move $t4, $v0
```

\$t3	11	10
\$t4	12	23

- Đếm các phần tử trong đoạn (M,N):
  - Trong quá trình duyệt mảng:
    - \$t1 lại được sử dụng để trỏ đến từng phần tử hiện tại trong mảng.
    - \$t2 lưu trữ số phần tử đã kiểm tra
    - \$t5 đếm các phần tử thỏa mãn
    - \$t6 tạm thời lưu trữ giá trị của phần tử đang được kiểm tra.

- \$t2 tiếp tục là bộ đếm vòng lặp.

\$t0	8	5
\$t1	9	268501012
\$t2	10	5
\$t3	11	10
\$t4	12	23
\$t5	13	1
\$t6	14	28
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0

### **Chạy thử nghiệm:**

Số lượng phần tử của mảng: 5

Các phần tử trong mảng: [8,6,7,12,28]

Giá trị M,N: (10,23)

Kết quả mong đợi: 1 (tức số 12)

Kết quả trả về:

Nhap so phan tu cua mang: 5		
Nhap vao cac phan tu: 8		
6		
7		
12		
28		
Gia tri M: 10		
Gia tri N: 23		
So phan tu nam trong khoang (M, N): 1		
\$a0	4	1



8 <-- \$t1 <-- X		8		8		8		8	
6		6 <-- \$t1 <-- X		6		6		6	
7		7		7 <-- \$t1 <-- X		7		7	
12		12		12		12 <-- \$t1 <-- 0		12	
28		28		28		28		28 <-- \$t1 <-- 0	
\$t2 = 1 \$t5 = 0	\$t6 = 8 < 10	\$t2 = 2 \$t5 = 0	\$t6 = 6 < 10	\$t2 = 3 \$t5 = 0	\$t6 = 7 < 10	\$t2 = 4 \$t5 = 1	\$t6 = 12 ok	\$t2 = 5 \$t5 = 1	\$t6 = 28 > 23

### Assignment 3

#### Code:

#assignment 3: 6\_Nhập vào xâu ký tự và ký tự C. In ra số lần xuất hiện ký tự C trong xâu (không phân biệt chữ hoa hay chữ thường)

#author: Nguyễn Bình An\_20225591

```
.data
input_msg: .asciiz "Enter a string: "
char_msg: .asciiz "Enter a character to count: "
result_msg: .asciiz "\nNumber of occurrences: "
buffer: .space 256 # Memory space for the input string
```

```
.text
.globl main
```

main:

```
# Prompt and read the string
li $v0, 4
la $a0, input_msg
syscall
```

```
# Read string from keyboard
li $v0, 8
la $a0, buffer
li $a1, 256
syscall
li $v0, 4
la $a0, char_msg
syscall
```

```
# Read character from keyboard
li $v0, 12
syscall
move $a1, $v0 # Store the character to count in $a1
```

```
# Convert character to lowercase if it is uppercase
li $t4, 'A'
li $t5, 'Z'
blt $a1, $t4, skip_convert_char
bgt $a1, $t5, skip_convert_char
addi $a1, $a1, 32 # Convert to lowercase
```

```

skip_convert_char:
    # Initialize occurrence count to 0
    li $t0, 0

    # Count occurrences of the character in the string
    la $t1, buffer # Pointer to the start of the buffer
count_loop:
    lbu $t2, 0($t1) # Load byte (unsigned) from the string
    beqz $t2, count_end # If byte is zero (end of string), exit loop

    # Convert to lowercase if uppercase
    li $t3, 'A'
    li $t4, 'Z'
    blt $t2, $t3, check_char
    bgt $t2, $t4, check_char
    addi $t2, $t2, 32

check_char:
    beq $t2, $a1, increment # If the character matches, increment count

    addi $t1, $t1, 1 # Move pointer to the next character
    j count_loop

increment:
    addi $t0, $t0, 1 # Increment the count
    addi $t1, $t1, 1 # Move pointer to the next character
    j count_loop

count_end:
    li $v0, 4 #Print the result message
    la $a0, result_msg
    syscall

    move $a0, $t0 # Print the number of occurrences
    li $v0, 1
    syscall

    li $v0, 10 # Exit the program
    syscall

```

### **Giải thích code:**

**\$v0**: Dùng để đọc chuỗi, đọc ký tự, in ra màn hình, vv

**\$a0, \$a1**: Dùng để lưu trữ các giá trị cho các hàm hệ thống, như địa chỉ của chuỗi cần in hoặc giá trị của ký tự cần đọc.

\$t0 - \$t5: Các thanh ghi tạm thời được sử dụng để thực hiện các phép toán và lưu trữ giữ liệu tạm thời.

- Đầu tiên kiểm tra xem chữ đã nhập có ký tự hoa hay không (bằng cách soát các thanh ghi \$t4, \$t5 là các giá trị từ 'A' – 'Z'), nếu có ký tự hoa thì nén thành ký tự thường (bằng lệnh `addi $t2, $t2, 32` sẽ cộng thêm 32 vào giá trị ASCII để chuyển thành chữ thường)
- So sánh ký tự:
  - Label 'check\_char' đánh dấu tiếp tục của vòng lặp nơi diễn ra việc so sánh ký tự.

skip_convert_char	0x0040005c
count_loop	0x00400068
check_char	0x0040008c

- Lệnh `beq $t2, $a1, increment` So sánh ký tự đã được chuyển đổi trong \$t2 với ký tự mục tiêu trong \$a1 (đã được chuyển thành chữ thường). Nếu chúng giống nhau, nhảy tới nhãn increment để tăng số lần đếm.
- Còn nếu không giống nhau, thì con trỏ \$t1 sẽ chuyển tới ký tự tiếp theo

### Chạy thử nghiệm:

Input: 'binhan'

c: 'n' -> \$a1 = 0x0000006e

Enter a string: binhan
Enter a character to count: n
Number of occurrences: 2
\$a0
4
2

b	<-- \$t1 <-- X		b			b			b			b			b		
i			i	<-- \$t1 <-- X		i			i			i			i		
n			n			n	<-- \$t1 <-- X		n			n			n		
h			h			h			h	<-- \$t1 <-- X		h			h		
a			a			a			a			a	<-- \$t1 <-- X		a		
n			n			n			n			n			n	<-- \$t1 <-- 0	
\$t0 = 0	\$t2 = 'b' >< 'n'		\$t0 = 0	\$t2 = 'i' >< 'n'		\$t0 = 1	\$t2 = 'n' ok		\$t0 = 1	\$t2 = 'h' >< 'n'		\$t0 = 1	\$t2 = 'a' >< 'n'		\$t0 = 2	\$t2 = 'n' ok	

