ĐẠI HỌC QUỐC GIA

ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH

BÀI TẬP LỚN MÔN KIẾN TRÚC MÁY TÍNH

ĐỀ TÀI: 2

LỚP L07 --- NHÓM 1 --- HK 221

NGÀY NỘP: 24/12/2022

Giảng viên hướng dẫn: Nguyễn Xuân Minh

Sinh viên thực hiện	Mã số sinh viên	Điểm số
Võ Tấn Hưng	2113623	
Nguyễn Phạm Thiên Phúc	2114445	

Thành phố Hồ Chí Minh – 2022

Đề 3: Nhân 2 số nguyên 32 bit

Viết chương trình hiện thực giải thuật nhân số nguyên trong textbook (hình 3.4 hoặc 3.5), áp dụng cho số có dấu. Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa INT2.BIN (2 trị x 4 bytes= 8 bytes)

BÁO CÁO

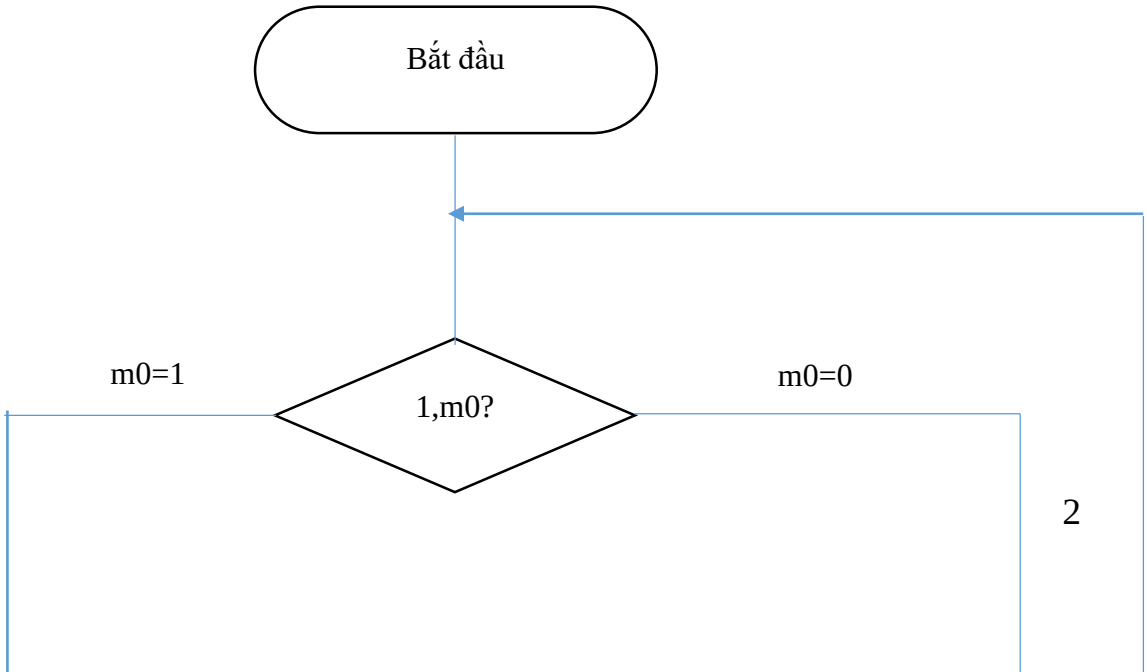
I) Giải pháp hiện thực

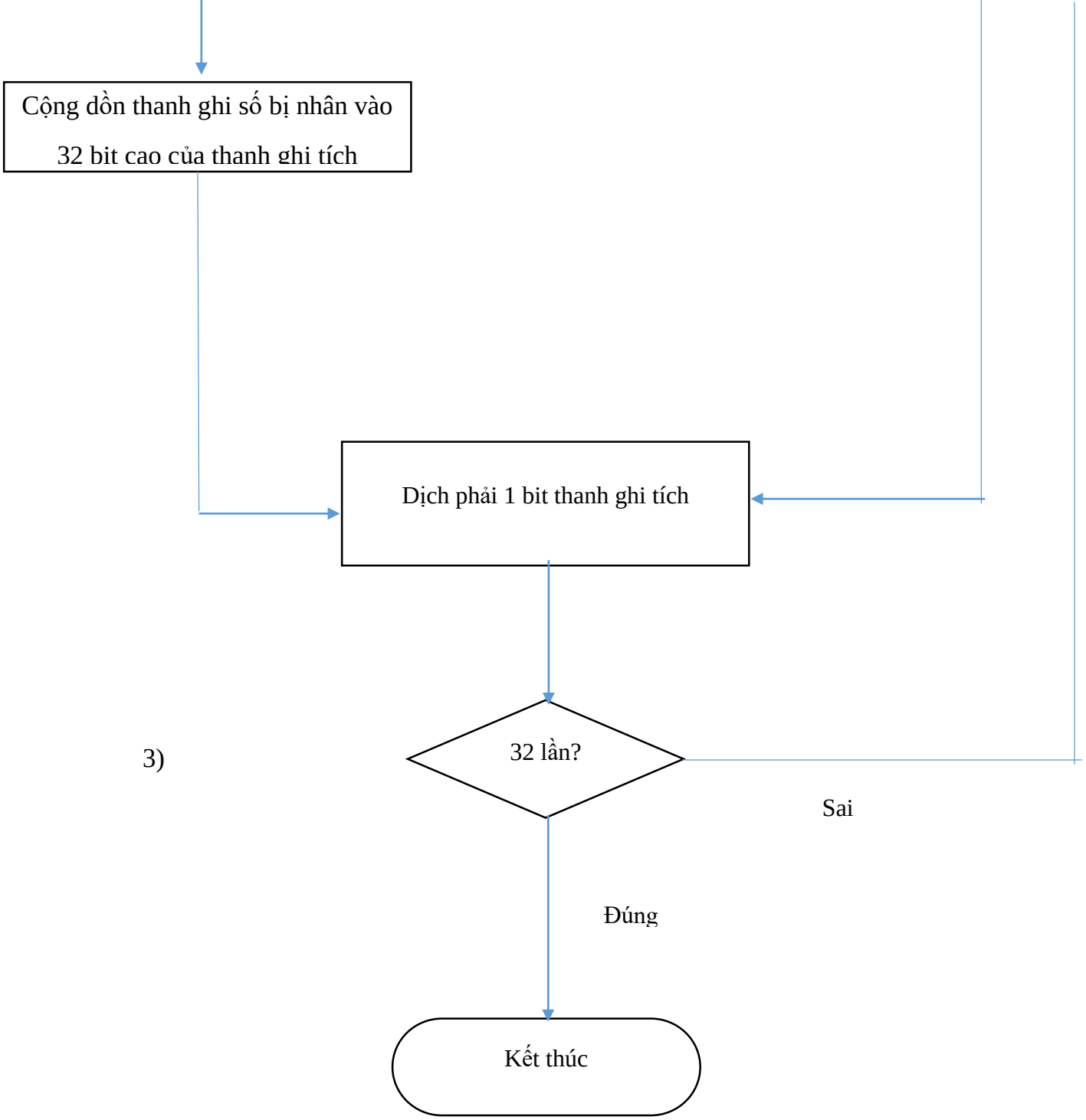
Ta nhận thấy đề yêu câu nhân 2 số nguyên 32 bits tuy nhiên mỗi thanh ghi chỉ chứa đúng 32 bits. Do đó việc áp dụng việc dùng phép cộng để thực thi là không thể vì trường immediate chỉ có tối đa 16 bits. Theo gợi ý của đề bài ta áp dụng những phương pháp sau để giải quyết bài tập này

1) Áp dụng giải thuật trong sách để nhân 2 số nguyên 32 bits
2) Dùng giải thuật để in ra được một số 64 bits bằng việc chuyển đổi thành string

II) Giải thuật

1) Giải thuật để nhân 2 số nguyên trong sách được miêu tả bằng sơ đồ khối sau:





3)

Đầu tiên ta sẽ lưu giá trị của số nhân vào số bị nhân vào thanh ghi 32 bits, và tạo thêm 1 thanh ghi 32 bit HI (giữ giá trị 32bit cao của phép nhân) , 2 thanh ghi 32 bits nhân nhau sẽ tạo ra một số 64bits. Lần lượt ta lấy bit cuối của số bị nhân nhân cho số nhân , nếu bit này là 1 ta sẽ lấy giá trị của số nhân cộng vào phần 32 bit HI của kết quả, rồi dịch phải, nếu là số 0 thì ta chỉ cần dịch phải. Đáp án ta thu được là sự kết hợp của 32 bit HI vào thanh ghi chứa số bị nhân(lúc này giá trị không còn là số bị nhân do đã bị dịch phải)

2) Dùng giải thuật để in ra được một số binary và một số decimal từ kết quả 64 bits:

- Để có thể in được số 64 bits thì không quá khó, nhưng trước khi in ra ta cần kiểm tra xem kết quả là số âm hay dương vì khi code ta đều chuyển số bị nhân và số nhân thành dương (nếu âm) dẫn đến kết quả phép nhân sẽ luôn dương . Dấu của kết quả đã được lưu trong nhãn "***dau***" của chương trình nếu có giá trị là 0 thì không sao, nếu có giá trị là 1 thì ta sẽ làm phép bù 2.

```
Am:
  Bu_1:
  nor $t3,$t3,$zero
  nor $t4,$t4,$zero # Bu 1

  Bu_2:
  addiu $t4,$t4,1 # add 32 bit lo
  sltiu $t5,$t4,1 #carry flag
  addu $t3,$t3,$t5 #add 32 bit hi
```

Figure 1Thuật toán bù 2 cho số 64 bits trong mips

- Cuối cùng ta sẽ dùng lệnh *syscall* 35 để in ra 32 bits cao và 32 bits thấp.
- Để in ra được một số decimal thì phức tạp hơn so với việc in ra dạng binary. Vì trong mips chỉ có thể lưu và in ra một số nguyên dương có giá trị tối đa là 2^{32}-1 nên hướng đi tốt nhất ở đây để đổi 64 bits ra decimal là chuyển nó thành string. Việc đổi từ binary sang decimal thì ta làm như định nghĩa đó là kết quả sẽ cộng dồn 2^{index} với *index* là vị trí của bit bắt đầu từ phải sang trái của kết quả binary. Tuy nhiên như đã nói ở trên vì sự hạn chế về giá trị của số được lưu trong thanh ghi nên mọi phép tính toán cập nhật kết quả trong quá trình này ta đều thực hiện trên kiểu dữ liệu string.

```cpp
string solve(string a, string b) {
    int c = 0;
    int la = a.size() - 1;
    int lb = b.size() - 1;
    string ans = "";
    while ((la >= 0) || (lb >= 0) || (c > 0)) {
        int v = c;
        if (la >= 0) v += a[la --] - '0';
        if (lb >= 0) v += b[lb --] - '0';
        ans = ((char)(48 + v % 10)) + ans;
        c = v/10;
    }
    return ans;
}
```

Figure 2Thuật toán cộng 2 nguyên lớn dạng string viết bằng C++

```cpp
string convert_to_decimal(int binary){

    string ans="0";
    string base = "1";

    while(binary != 0)
    {
        int LSB = binary%10;

        if(LSB == 1) ans = add(ans, base); // ans += base;

        base = add(base, base); // base *=2;
        binary/=10;
    }
    return ans;
}
```

Figure 3Thuật toán đổi từ binary sang decimal bằng C++

- Và cuối cùng kết quả dạng decimal sẽ được lưu trong nhãn "***ans***". Và tương tự như phần in ra binary trước khi in ra kết quả này ta cũng sẽ kiểm tra dấu của kết quả. Nếu âm thì in ra dấu " - " trước rồi mới in ra kết quả, còn dương thì in thẳng ra kết quả.

III) Thống kê số lệnh, loại lệnh (instruction type) sử dụng trong chương trình

Sử dụng Tools -> Instruction count ta đếm được

IC = 111218

R-type count = 19963

I-type count = 75715

J-type count = 15540

Lệnh	Số lần khai báo	Loại lệnh
Srl	8	R
Slt	2	R
Sll	2	R
Or	1	R
li	26	I
La	25	I
Addi	27	I
Subi	22	I
Sw	5	I
Bltz	6	I
Sw	5	I
Beq	1	I
Beqz	20	I
Lw	15	I
Lb	13	I
Sb	8	I
Andi	4	I
Addiu	2	I
J	20	J
Jal	3	J
Jr	3	J

IV) Tính thời gian chạy của chương trình(CR = 1GHz)

- Theo kiến trúc pipeline (5 giai đoạn) ta có:

Số chu kì thực thi = số lệnh + 4 = 111218 + 4 = 111222

- CPI được tính theo công thức:

$$\text{CPI} = \frac{\text{số chu kì thực thi}}{\text{số lệnh}} = \frac{111222}{111218} \approx 1$$

- Thời gian chạy của chương trình là

$$\text{CPU time} = \frac{IC * CPI}{Clock\ rate} = \frac{111218 * 1}{1 * 10^9} \approx 0.11\ (ms)$$

V) Kết quả kiểm thử

1. 45 x 56

```
Du lieu 1 = 45
Du lieu 2 = 56
Ket qua sau khi nhan he 2 la: 00000000000000000000100111011000
Ket qua sau khi nhan he 10 la: 2520

-- program is finished running --
```

2. 1234567897 x 223123124

```
Du lieu 1 = 1234567897
Du lieu 2 = 223123124
Ket qua sau khi nhan he 2 la: 0000001111010010101000011111111110000000100100010010011010010100
Ket qua sau khi nhan he 10 la: 275460645968750228

-- program is finished running --
```

3. 78945225 x 55445562

```
Du lieu 1 = 78945225
Du lieu 2 = 55445562
Ket qua sau khi nhan he 2 la: 0000000000001111100011010000000110001000000001111001001110001010
Ket qua sau khi nhan he 10 la: 4377162367341450

-- program is finished running --
```