ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI NHÂN HAI SỐ NGUYÊN 32 BIT

LỚP: L06

NHÓM: 8

Giảng viên hướng dẫn: Trần Nguyễn Minh Duy

Họ và tên	Mã số sinh viên	Đóng góp
Lê Võ Đăng Khoa	2211606	100%
Võ Xuân Hạ	2210916	100%
Nguyễn Thị Hiền Hạnh	2210920	100%
Thái Trí Thịnh	2213308	100%

Thành phố Hồ Chí Minh, tháng 12, năm 2023

Mục lục

Danh sách hình ảnh

I. Giới thiệu đề tài

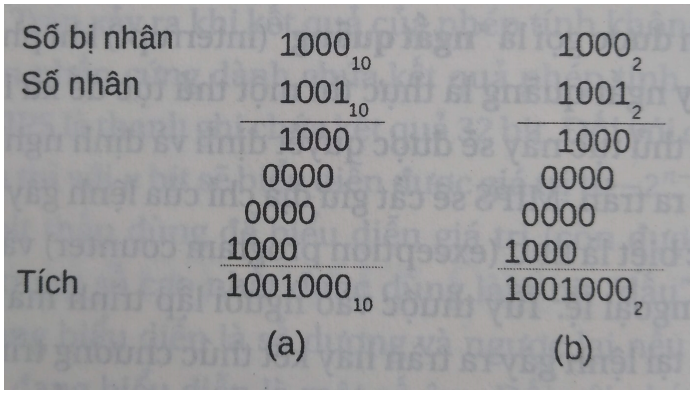
1. Chủ đề thực hiện: Nhân hai số nguyên 32 bit

- Assignment 2 – Đề 1: Viết chương trình hiện thực giải thuật nhân số nguyên trong textbook (hình 3.4). Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa INT2.BIN (2 trị × 4 bytes = 8 bytes). Chương trình được test trên máy tính kiến trúc MIPS có tần số 3.4 GHz.

2. Thực hiện phép nhân trên giấy

- Trong phép nhân, thừa số thứ nhất sẽ được gọi là số bị nhân (multiplicand), thừa số thứ hai là số nhân (multiplier) và kết quả phép nhân là tích (product). Giải thuật nhân hai số ở dạng thập phân sử dụng "giấy và bút" đã được giới thiệu trong trường phổ thông là chọn từng ký số của số nhân từ phải sang trái, thực hiện phép nhân của ký số này với số bị nhân và dịch các giá trị tích tạm thời này sang trái ở mỗi phép tính.

- Nếu giả sử hai giá trị thập phân là giá trị nhị phân (số thập phân nhưng chỉ gồm các ký số 0 và 1) thì việc nhân hai số nhị phân hoàn toàn giống với nhân hai số thập phân.



Hình 1: Giải thuật nhân (a) giá trị hệ thập phân; (b) giá trị hệ nhị phân

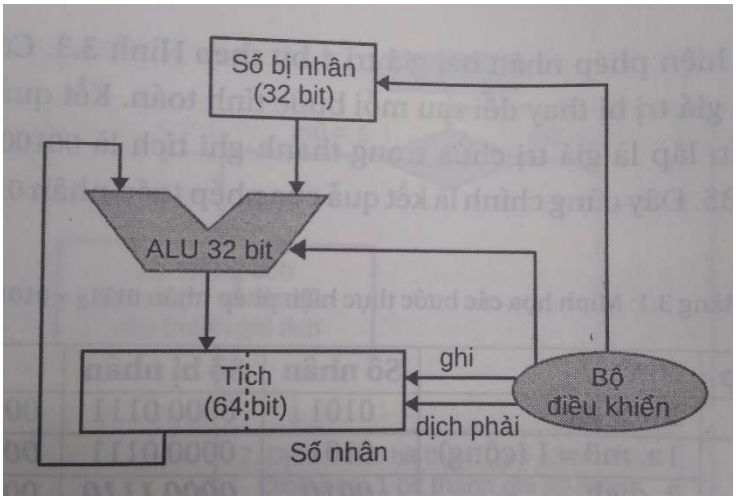
- Nếu nhân hai số nguyên không dấu *m* và *n* bit, tích số của phép nhân sẽ là một số có kích thước *m* + *n* bit.

- Phép nhân hai số có dấu có thể thực hiện được bằng cách nhân hai số nguyên không dấu rồi xét dấu kết quả. Nếu hai số nhân cùng dấu thì kết quả là một số nguyên dương; ngược lại kết quả là một số nguyên âm.

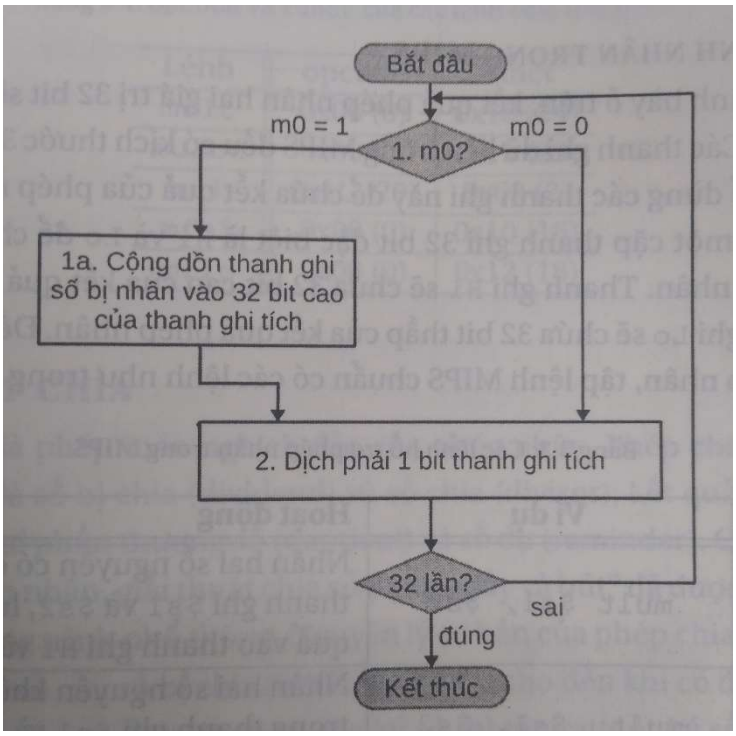
3. Bộ nhân tuần tự tối ưu

- Ta thấy rằng nếu thực hiện kiến trúc bộ nhân tuần tự giống như thuật toán trên thì vẫn còn một số ưu điểm chưa tối ưu như cần phải sử dụng thanh ghi 64 bit để lưu trữ giá trị số

bị nhân, cần sử dụng bộ cộng ALU 64 bit. Kiến trúc bộ nhân tối ưu trong hình 3.4 sẽ khắc phục những vấn đề này bằng cách sử dụng thanh ghi 32 bit để chứa giá trị số bị nhân, sử dụng bộ cộng 64 bit và không cần một thanh ghi riêng biệt để lưu giá trị số nhân.



Hình 2: Kiến trúc bộ nhân tuần tự 32 bit tối ưu (hình 3.4 trong textbook)



Hình 3: Hoạt động của bộ nhân tuần tự trong hình 2

- Đề tài này nhóm em sẽ viết chương trình thực hiện giải thuật nhân hai số nguyên 32 bit như được mô tả trong bộ nhân tuần tự tối ưu ở trên.

II. Giải pháp hiện thực

1. Thuật toán nhân 2 số hệ nhị phân

Gọi phép toán $A_2 \times B_2$: số A là số bị nhân, số B là số nhân.

1.1. Các bước chuẩn bị:

Bước 1: Xét dấu hai số trên xem kết quả sẽ ra số dương hay âm. Nếu ra số âm thì kết quả sẽ phải bù 1 để có được số âm.

Bước 2: Đổi dấu các thanh ghi A và B (nếu có) thành số dương.

Bước 3: Lưu thanh ghi B vào thanh ghi tích và extend từ 32 bit thành 64 bit. Gọi bit thấp nhất của thanh ghi tích là m_0.

1.2. Bắt đầu: Do thực hiện phép nhân trên số 32 bit, ta sẽ xét giá trị m_0 32 lần

Bước 4: Nếu giá trị của m_0 là 1, cộng dồn thanh ghi A vào 32 bit cao của thanh ghi tích.

Bước 5: Nếu giá trị của m_0 là 0, nhảy đến bước 6.

Bước 6: Dịch phải thanh ghi tích 1 bit.

Bước 7: Lặp lại bước 4 cho đến khi đã thực hiện bước 4 32 lần.

1.3. Kết quả:

Bước 8: Nếu kết quả là số âm (được kiểm tra từ bước 1) thì bù 1 thanh ghi tích để đưa kết quả trở thành số âm.

Ví dụ: Thực hiện phép nhân $1011_2 \times 0111_2$

- Chuyển số bị nhân 1011_2 (số âm) thành 0101_2 (số dương). Giá trị kết quả sẽ là số âm.
- Lưu 0111_2 vào thanh ghi tích và extend ra thành 8 bit. Thanh ghi tích bây giờ có giá trị là $0000\ 0111_2$ (bước khởi tạo).
- Bắt đầu thuật toán:

Lần lặp	Bước	Số bị nhân	Tích
0	Khởi tạo	0101	0000 011<u>1</u>
1	5. $m_0 = 1$ (cộng)	0101	0101 0111
	6. Dịch phải	0101	0010 101<u>1</u>
2	5. $m_0 = 1$ (cộng)	0101	0111 1011
	6. Dịch phải	0101	0011 110<u>1</u>
3	5. $m_0 = 1$ (cộng)	0101	1000 1101
	6. Dịch phải	0101	0100 011<u>0</u>
4	4. $m_0 = 0$	0101	0100 0110

	6. Dịch phải	0101	0010 0011

Hình 4: Ví dụ thuật toán nhân hai số 4 bit

- Vì kết quả là số âm, ta bù 1 vào thanh ghi tích, kết quả có được là 1101 1101_2

+ Kiểm tra kết quả: $1011_2 \times 0111_2 = -5 \times 7 = -35$

+ Kết quả thu được: 1101 $1101_2 = -35$

2. Các hàm được dùng trong chương trình

2.1. Hàm hỗ trợ đọc file

- load_fil: Load số nhân và số bị nhân vào thanh ghi $s0, $s1.

Các hàm hỗ trợ load:

+ load_file: Load hai số nguyên dưới dạng chuỗi từ file vào thanh ghi $t0.

+ get_first_length: Xác định độ dài của số nguyên đầu tiên trong chuỗi dữ liệu.

+ end_first_length: Điểm kết thúc xác định độ dài của số nguyên đầu tiên.

+ get_second_length: Xác định độ dài của số nguyên thứ hai trong chuỗi dữ liệu.

+ end_second_length: Điểm kết thúc xác định độ dài của số nguyên thứ hai.

+ get_pow_10_first: Tính giá trị lũy thừa của 10 tương ứng với độ dài của số nguyên đầu tiên.

+ end_pow_10_first: Điểm kết thúc tính giá trị lũy thừa của 10 cho số nguyên đầu tiên.

+ get_pow_10_second: Tính giá trị lũy thừa của 10 tương ứng với độ dài của số nguyên thứ hai.

+ end_pow_10_second: Điểm kết thúc tính giá trị lũy thừa của 10 cho số nguyên thứ hai.

+ loop1: Vòng lặp để chuyển đổi chuỗi thành số nguyên đầu tiên.

+ end1: Điểm kết thúc vòng lặp chuyển đổi cho số nguyên đầu tiên.

+ loop2: Vòng lặp để chuyển đổi chuỗi thành số nguyên thứ hai.

+ end_load: Điểm kết thúc quá trình xử lý chuỗi để chuyển đổi thành số nguyên.

2.2. Các hàm xử lý điều kiện và tính toán

- not_neg: Xem như một điều kiện if, nếu là số âm thì sẽ được xử lý còn dương thì được bỏ qua.

- end: Kết thúc chương trình.

- loop: Chạy 32 lần, mỗi lần chạy sẽ kiểm tra bit nhỏ nhất (LSB), dịch bit, cộng số bị nhân (nếu LSB = 1).

- least_significant_bit: Trả về thanh ghi $a0 chứa LSB của tích (0 hoặc 1).

- add_hi: Cộng số bị nhân vào 32 bit cao của tích.

- shift_right: Dịch phải tích 1 bit.

- neg_or_pos: Trả về thanh ghi $t6 giá trị 1 nếu tích là số dương, giá trị 0 nếu là số âm.

III. Cách cài đặt và chạy chương trình

1. Chỉnh đường dẫn của chương trình đến file đầu vào INT2.BIN

- Copy đường dẫn của file INT2.BIN. Ví dụ đường dẫn đến file INT2.BIN là C:\Users\Admin\OneDrive\Desktop.

- Thêm dấu "\" vào cạnh dấu "\" của đường dẫn. Đường dẫn bây giờ sẽ trở thành C:\\Users\\Admin\\OneDrive\\Desktop.

- Dán vào dòng thứ 6 trong chương trình, ngay cạnh *filename:* .asciiz như hình dưới.

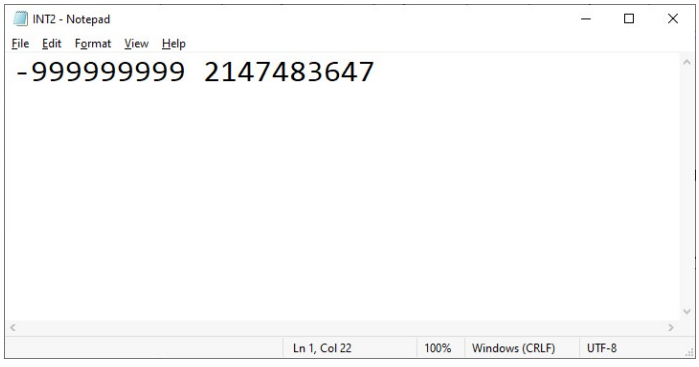
```
1  #Chuong trinh Assignment 2 Bai 1: Nhan hai so nguyen 32bit
2  #-----------------------------------
3  #Data segment
4  .data
5  #Cac dinh nghia bien
6            filename:           .asciiz "C:\\Users\\Admin\\OneDrive\\Desktop\\INT2.bin"
7            buffer:    .space 32000
8            zero:               .word 0
9            one:                .word 1
```

Hình 5: Cài đặt đường dẫn từ chương trình đến file INT2.BIN

2. Chỉnh giá trị của file dữ liệu đầu vào INT2.BIN

- Nhập hai số nhân vào theo hệ 10, hai số cách nhau một dấu cách (như hình dưới).

- Sau đó bấm lưu file rồi ta mới có thể chạy chương trình.



Hình 6: Nhập giá trị của hai số nhân vào file INT2.BIN

IV. Kết quả chương trình

1. Thời gian thực thi của chương trình

- Chương trình được chạy trên máy tính kiến trúc MIPS có tần số 3.4 GHz, với CPI = 1 cho 1 lệnh MIPS chuẩn. Công thức tính thời gian thực thi:

$$CPU_{time} = \frac{IC.CPI}{CR}$$

Trong đó:

+ CPU_{time}: Thời gian thực thi

+ IC (Instruction count): Tổng số lệnh thực thi

+ CPI (Cycle per Instruction): Số chu kỳ thực thi một lệnh

+ CR (Clock rate): Tần số máy tính

2. Kết quả thống kê

- Kết quả của chương trình, bao gồm thống kê số lệnh, loại lệnh sử dụng trong chương trình được tổng hợp lại ở bảng dưới. Thời gian chạy của chương trình và kết quả kiểm thử cũng được thống kê ở trong bảng.

- Các trường hợp tốt nhất (best case) và trường hợp xấu nhất (worst case) được đánh dấu bằng màu xanh lá và màu đỏ trong bảng. Các giá trị trung bình (average) được thống kê ở dòng dưới cùng của bảng và được đánh dấu màu vàng.

- Các test case đều chạy thành công. Ta có thể thấy rằng đối với các số nhân càng lớn và phức tạp thì số lệnh cần thực thi càng nhiều và thời gian thực thi càng lớn.

Test case	Mô tả test case		Kết quả			Số lệnh				Thời gian thực thi(ns)
	Số thứ nhất	Số thứ hai	Kết quả dự tính	Kết quả chương trình		R	I	J	IC	
1	10	5	50	50	Pass	659	1219	110	1988	584.7
2	-23	2	-46	-46	Pass	660	1241	112	2013	592.1
3	14	-8	-112	-112	Pass	659	1239	112	2010	591.2
4	-327	-11	3597	3597	Pass	681	1293	123	2097	616.8
5	-1	-1	1	1	Pass	642	1215	109	1966	578.2
6	-1	2147483647	-2147483647	-2147483647	Pass	975	1649	174	2798	822.9
7	2147483647	2147483647	4611686014132420609	4611686014132420609	Pass	991	1724	207	2922	859.4
8	-2147483648	-2147483648	4611686018427387904	4611686018427387904	Pass	749	1537	181	2467	725.6
9	0	2147483647	0	0	Pass	932	1554	171	2657	781.5
10	2147483647	0	0	0	Pass	684	1338	140	2162	635.9
11	1	-2147483648	-2147483648	-2147483648	Pass	705	1379	144	2228	655.3
12	2147483648	-1	-2147483648	-2147483648	Pass	695	1363	143	2201	647.4
13	0	0	0	0	Pass	630	1175	104	1909	561.5
14	153454685	-3	-460364055	-460364055	Pass	721	1396	141	2258	664.1
15	1111111111	111111111	123456789987654321	123456789987654321	Pass	896	1645	190	2731	803.2
16	123456789	987654321	121932631112635269	121932631112635269	Pass	888	1632	185	2705	795.6
17	2608	-1711	-4462288	-4462288	Pass	750	1388	139	2277	669.7
18	-57865	-22154	1281941210	1281941210	Pass	754	1433	147	2334	686.5
19	198274	345354	68474718996	68474718996	Pass	765	1440	151	2356	692.9
20	-99999	99999	-9999800001	-9999800001	Pass	782	1446	149	2377	699.1
21	999999	-999999	-999998000001	-999998000001	Pass	811	1498	159	2468	725.9
22	5345225	45677488	244156450794800	244156450794800	Pass	845	1546	171	2562	753.5
23	999999	-2147483648	-2147481500516352	-2147481500516352	Pass	735	1469	164	2368	696.5
24	-999999999	2147483647	-2147483644852516353	-2147483644852516353	Pass	1003	1753	206	2962	871.2
25	999999999	999999999	999999998000000001	999999998000000001	Pass	909	1638	189	2736	804.7
26	45456	-54747543	-2488604314608	-2488604314608	Pass	826	1525	164	2515	739.7
27	13579	-24680	-335129720	-335129720	Pass	727	1421	144	2312	680
28	-899999	888888	-799998311112	-799998311112	Pass	782	1476	155	2413	709.7
29	-1234321	-1234321	1523548331041	1523548331041	Pass	800	1522	166	2488	731.8
30	123456789	11111	13717407282579	13717407282579	Pass	791	1486	161	2438	717.1
					Average	781.6	1455	153.7	2391	703.12

Hình 7: Kết quả chạy của tất cả các test case

3. Hình ảnh của một test case (test case 24)

```
So bi nhan (32 bit): 11000100011001010011011000000001 (-999999999)
So nhan (32 bit): 01111111111111111111111111111111 (2147483647)
Ket qua la so am, ta thuc hien nhan khong dau sau do them dau tru vao ket qua
Step 0: 00000000000000000000000000000001111111111111111111111111111111
Step 1:
LSB:    1
Add:    00111011100110101100100111111111101111111111111111111111111111111
Shift:  00011101110011010110010011111111110111111111111111111111111111111
Step 2:
LSB:    1
Add:    01011001011010000010111011111110101111111111111111111111111111111
Shift:  00101100101101000001011101111111010111111111111111111111111111111
Step 3:
LSB:    1
Add:    01101000010011011100001011111100101111111111111111111111111111111
Shift:  00110100001001110111000010111111001011111111111111111111111111111
Step 4:
LSB:    1
Add:    01101111110000100011101010111110001011111111111111111111111111111
Shift:  00110111111000010001110101011111000101111111111111111111111111111
Step 5:
LSB:    1
Add:    01110011011110111110011101011110000101111111111111111111111111111
Shift:  00111001101111011111001110101111000010111111111111111111111111111
Step 6:
LSB:    1
Add:    01110101010110001011110110101110000010111111111111111111111111111
Shift:  00111010101011000101111011010111000001011111111111111111111111111
Step 7:
LSB:    1
Add:    01110110010001110010100011010110000001011111111111111111111111111
Shift:  00111011001000111001010001101011000000101111111111111111111111111
Step 8:
LSB:    1
Add:    01110110101111100101111001101010000000101111111111111111111111111
Shift:  00111011010111110010111100110101000000010111111111111111111111111
Step 9:
LSB:    1
Add:    01110110111110011111100100110100000000010111111111111111111111111
Shift:  00111011011111001111110010011010000000001011111111111111111111111
Step 10:
LSB:    1
Add:    01110111000101111100011010011001000000001011111111111111111111111
Shift:  00111011100010111110001101001100100000000101111111111111111111111
Step 11:
LSB:    1
Add:    01110111001001101010110101001011100000000101111111111111111111111
Shift:  00111011100100110101011010100101110000000001011111111111111111111
Step 12:
LSB:    1
Add:    01110111001011000100000101001001100000000010111111111111111111111
Shift:  00111011100101110001000001010010011000000000101111111111111111111
```

Hình 8: Kết quả chạy chương trình của test case 24 từ lúc khởi tạo đến step 12

```
Step 13:
LSB:    1
Add:    01110111001100011101101001010001011000000000101111111111111111111
Shift:  00111011100110001110110100101000101100000000010111111111111111111
Step 14:
LSB:    1
Add:    01110111001100111011011100100111101100000000010111111111111111111
Shift:  00111011100110011101101110010011110110000000001011111111111111111
Step 15:
LSB:    1
Add:    01110111001101001010010110010010110110000000001011111111111111111
Shift:  00111011100110100101001011001001011011000000000101111111111111111
Step 16:
LSB:    1
Add:    01110111001101010001110011001000011011000000000101111111111111111
Shift:  00111011100110101000111001100100001101100000000010111111111111111
Step 17:
LSB:    1
Add:    01110111001101010101100001100011001101100000000010111111111111111
Shift:  00111011100110101010110000110001100110110000000001011111111111111
Step 18:
LSB:    1
Add:    01110111001101010111011000110000100110110000000001011111111111111
Shift:  00111011100110101011101100011000010011011000000000101111111111111
Step 19:
LSB:    1
Add:    01110111001101011000010100010111010011011000000000101111111111111
Shift:  00111011100110101100001010001011101001101100000000010111111111111
Step 20:
LSB:    1
Add:    01110111001101011000110010001010101001101100000000010111111111111
Shift:  00111011100110101100011001000101010100110110000000001011111111111
Step 21:
LSB:    1
Add:    01110111001101011001000001000100010100110110000000001011111111111
Shift:  00111011100110101100100000100010001010011011000000000101111111111
Step 22:
LSB:    1
Add:    01110111001101011001001000100000100101001101100000000010111111111
Shift:  00111011100110101100100100010000010010100110110000000001011111111
Step 23:
LSB:    1
Add:    01110111001101011001001100001111100101001101100000000010111111111
Shift:  00111011100110101100100110000111110010100110110000000001011111111
Step 24:
LSB:    1
Add:    01110111001101011001001110000110110010100110110000000001011111111
Shift:  00111011100110101100100111000011011001010011011000000000101111111
Step 25:
LSB:    1
Add:    01110111001101011001001111000010011001010011011000000000101111111
Shift:  00111011100110101100100111100001001100101001101100000000010111111
```

Hình 9: Kết quả chạy chương trình của test case 24 từ step 13 đến step 25

```
Step 26:
LSB:    1
Add:    01110111001101011001001111100000001100101001101100000000010111111
Shift:  00111011100110101100100111110000000110010100110110000000001011111
Step 27:
LSB:    1
Add:    01110111001101011001001111101111000110010100110110000000001011111
Shift:  00111011100110101100100111110111100011001010011011000000000101111
Step 28:
LSB:    1
Add:    01110111001101011001001111111011010001100101001101100000000101111
Shift:  00111011100110101100100111111101101000110010100110110000000010111
Step 29:
LSB:    1
Add:    01110111001101011001001111111101001000110010100110110000000010111
Shift:  00111011100110101100100111111110100100011001010011011000000001011
Step 30:
LSB:    1
Add:    01110111001101011001001111111110000100011001010011011000000001011
Shift:  00111011100110101100100111111111000010001100101001101100000000101
Step 31:
LSB:    1
Add:    01110111001101011001001111111110100010001100101001101100000000101
Shift:  00111011100110101100100111111111010001000110010100110110000000010
Step 32:
LSB:    0
Shift:  00011101110011010110010011111111101000100011001010011011000000001
Them dau tru vao ket qua
Ket qua duoi dang nhi phan (64 bit): 1110001000110010100110110000000001011101110011010110010011111111111
-- program is finished running --
```

Hình 10: Kết quả chạy chương trình của test case 24 từ step 26 đến hết

- Vì Mars không xử lý được số lớn hơn 32 bit nên ta chỉ có thể thu về kết quả 64 bit theo dạng nhị phân mà không thể đổi về dạng số thập phân được.