Họ và tên	Nguyễn Đình Dương
MSSV	20225966

BÁO CÁO COMPUTER ARCHITECHTURE TUẦN 4

Assignment 1

1. Case 1:

s1= 2147483647 → số dương lớn

s2 = 1 \rightarrow số dương nhỏ

```
.text
       # Khởi tạo s1 và s2
       li s1, 2147483647
       li s2, 1
       # Thuật toán xác định điều kiện tràn số
       li t0, 0 # Mặc định không có tràn số
       add s3, s1, s2 \# s3 = s1 + s2
       xor t1, s1, s2 # Kiểm tra xem s1 và s2 có cùng dấu hay không
       blt t1, zero, EXIT # Nếu không, thoát
       blt s1, zero, NEGATIVE # Kiểm tra xem s1 và s2 có âm không?
       bge s3, s1, EXIT # s1 và s2 dương
       # nếu s3 >= s1 thì kết quả không bị tràn số
       j OVERFLOW
NEGATIVE:
       bge s1, s3, EXIT # s1 và s2 âm
       # nếu s1 >= s3 thì kết quả không bị tràn số
OVERFLOW:
```

li t0, 1 # Kết quả bị tràn số

-2	-	
t0	5	0x00000000
tl	6	0x00000000
t2	7	0x00000000
s 0	8	0x00000000
sl	9	0x7fffffff
a0	10	0x00000000
al	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001

s1 = 2147483647

s2 = 1

t0 = 0

+) s3 = s1 + s2 = 2147483647 + 1 = 0x7fffffff

⇒ overflows và được lưu vào s3: 0x80000000 là số âm trong kiểu số nguyên có dấu 32-bit, cho thấy một hiện tượng tràn số đã xảy ra.

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000001
tl	6	0x7ffffffe
t2	7	0x00000000
s0	8	0x00000000
sl	9	0x7fffffff
a0	10	0x00000000
al	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x80000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
sll	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400034

Kết quả:

 \Rightarrow

- s1 chứa giá trị 2147483647 (0x7FFFFFFF), là số nguyên dương lớn nhất có dấu trên 32-bit.
- s2 chứa giá trị 1.
- s3, kết quả của phép cộng s1 + s2, có giá trị 0x80000000, là số âm trong hệ thống số nguyên có dấu 32-bit.
- t0 được đặt thành 1, cho biết rằng đã xảy ra hiện tượng tràn số.

2. Case 2: 2 negative number

s1 = -2147483647

```
.text
        # Khởi tao s1 và s2
        li s1, -2147483647
        li s2, -1
        # Thuật toán xác định điều kiện tràn số
        li t0, 0 # Mặc định không có tràn số
        add s3, s1, s2 \# s3 = s1 + s2
        xor t1, s1, s2 # Kiểm tra xem s1 và s2 có cùng dấu hay không
        blt t1, zero, EXIT # Nếu không, thoát
        blt s1, zero, NEGATIVE # Kiểm tra xem s1 và s2 có âm không?
        bge s3, s1, EXIT # s1 và s2 dương
        # nếu s3 >= s1 thì kết quả không bị tràn số
        j OVERFLOW
NEGATIVE:
        li, t0, 2
        bge s1, s3, EXIT # s1 và s2 âm
        # nếu s1 \ge s3 thì kết quả không bị tràn số
OVERFLOW:
        li t0, 1 # Kết quả bị tràn số
EXIT:
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
зр	2	0x7fffeffc
qp	3	0x10008000
tp	4	0x00000000
:0	5	0x00000002
t1	6	0x7ffffffe
12	7	0x00000000
30	8	0x00000000
31	9	0x80000001
a O	10	0x00000000
al	11	0x00000000
a2	12	0x00000000
13	13	0x00000000
a 4	14	0x00000000
a5	15	0x00000000
a 6	16	0x00000000
a7	17	0x00000000
32	18	0xffffffff
33	19	0x80000000
34	20	0x00000000
35	21	0x00000000
36	22	0x00000000
37	23	0x00000000
38	24	0x00000000
39	25	0x00000000
s10	26	0x00000000
311	27	0x00000000
:3	28	0x00000000
t 4	29	0x00000000
:5	30	0x00000000
:6	31	0x00000000
pc		0x00400038

Kết quả:

- Kết quả của phép cộng: s1 + s2 = -2147483648.
- Không có tràn số (giá trị t0=0).
- t2: 0x000002: nghĩa là 2 số âm

3. Case 3: Tổng bằng 0 S1 = 69

S2 = -69

Name	Nur	mber	Value	☐ Kết quả	là c	hương trình	thoát khi	găp câu lê
ero		0	0x000000			8	•	5.1
a		1	0x000000	00 1.11		EVIT II NIŚ.	11.4	
p		2	0x7fffef	$f_{c} \mid DIIII, Z$	ero,	EXIT # Nếu	knong, the	oat
)		3	0x100080	00				
)		4	0x000000	00				
		5	0x000000	00				
		6	0xffffff	fe				
		7	0x000000	00				
		8	0x000000					
		9	0x000000					
		10	0x000000					
		11	0x000000					
		12	0x000000					
		13	0x000000					
		14	0x000000					
		15	0x000000					
		16	0x000000					
		17	0x000000					
		18	0xffffff					
		19	0x000000					
		20	0x000000	_				
;		21	0x000000					
		22	0x000000 0x000000					
		23	0x000000					
3								
				_				
		25	0x000000	00				
9		25 26	0x000000 0x000000	00				
10 11		25 26 27	0x000000 0x000000 0x000000	00 00 00				
10		25 26 27 28	0x000000 0x000000 0x000000 0x000000	00 00 00	số là	số 0		
0		25 26 27 28 29	0x000000 0x000000 0x000000 0x000000 0x000000	4. Một	số là	số 0		
0		25 26 27 28	0x000000 0x000000 0x000000 0x000000	4. Một	số là	số 0	Control and Status	
		25 26 27 28 29	0x000000 0x000000 0x000000 0x000000 0x000000	4. Một	o B	Registers Floating Point Name	Control and Status	Value
1		25 26 27 28 29	0x000000 0x000000 0x000000 0x000000 0x000000	4. Một	<u>- 2</u>	Registers Floating Point Name	Number 0	0x00
1	1.0	25 26 27 28 29 30	0x000000 0x000000 0x000000 0x000000 0x000000 0x000000	4. Một	- Z	Registers Floating Point Name zero ra	Number 0 1 2	0x00 0x00 0x7f
1	s1= 0	25 26 27 28 29 30	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 4. Một	o* 23	Registers Floating Point Name zero ra sp	Number 0 1 2 2 3	0x00 0x00 0x7f 0x10
1		25 26 27 28 29 30	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00		Registers Floating Point Name zero ra sp. gp. gp. tp.	Number 0 1 2	0x00 0x00 0x7f 0x10 0x00 0x00
1	s1= 0 s2 = 69	25 26 27 28 29 30 30 31 4: 7: 8: 9: 10:	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 000 000 4. Một 1+32 1+32 12 a xem 31 và 32 có củu u không, thoát		Registers Floating Point Name zero pp	Number 0 1 1 2 3 3 4	0x00 0x01 0x1f 0x10 0x00 0x00
0 1		25 26 27 28 29 30 31 41 71 81 91 101 111	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Registers Floating Point Name zero ra sp p P P D t t t t t t t t t t t t t t t t t	Number 0 1 2 3 3 4 4 5 6 7 7 8 8	0x00 0x7f 0x1d 0x00 0x00 0x00 0x00 0x00
1		25 26 27 28 29 30 31 41 71 81 91 101 111 122 141	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00		Registers Floating Point Name zero re app gp ttp ccc cci ccl ccl ccl ccl ccl ccl ccl ccl	Number 0 1 2 2 3 4 5 5 6 6 9 9 9	0x00 0x7f 0x10 0x00 0x00 0x00 0x00 0x00 0x00
0 1		25 26 27 28 29 30 31 4: 7: 8: 9: 10: 11: 12: 14: 17:	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00		Registers Floating Point Name zero re spp gp gp tbp tc1 tc2 so so sl sl sl ao al	Number 0 0 1 1 2 2 3 3 4 5 6 6 7 8 9 100 111	0x00 0x000 0x7f 0x10 0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000
0 1		25 26 27 28 29 30 31 41 71 81 91 101 111 122 141 171 181	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	<u> </u>	Registers Floating Point Name zero ra sp pp pp pc co co cl cl cl c2 a0 a1 a0 a1 a2	Number 0 1 1 2 3 3 4 4 5 5 6 6 9 10 11 1 1 1 1 1 2 1 2	0x00 0x000 0x7f 0x10 0x2ff 0x10 0x00 0x000
0 1		25 26 27 28 29 30 31 4: 7: 8: 9: 10: 11: 12: 14: 17:	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	r 2	Registers Floating Point Name zero ra spp pp pp to to til to a o a a a a a a a a a a a a a a a a	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 8 8 9 10 11 12 12 13 14 4 14 14 14 14 14 14 14 14 14 14 14 1	0x00 0x000 0x17 0x10 0x17 0x10 0x00 0x00
1		25 26 27 28 29 30 31 41 71 81 91 101 111 122 141 171 181	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	r d	Registers Floating Point Name zero ra spp gp gp tu	Number 0 1 1 2 3 3 4 5 5 6 7 7 8 9 10 11 12 13 14 4 15 15 15 15 15 15 15 15 15 15 15 15 15	(\$\infty\$)
1		25 26 27 28 29 30 31 41 71 81 91 101 111 122 141 171 181	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	a	Registers Floating Point Name zero ra spp pp pp to to til to a o a a a a a a a a a a a a a a a a	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 8 8 9 10 11 12 12 13 14 4 14 14 14 14 14 14 14 14 14 14 14 1	(\$20,000) (\$0.000) (\$
1		25 26 27 28 29 30 31 41 71 81 91 101 111 122 141 171 181	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 4. Một 00 00 1+82 1+82 1xa xem sl và 82 có của u không, thoàu si	p* Ø	Registers Floating Point Name zero re sep pp pp pp to to to to to to al al a2 a3 a4 a5 a6 a7 a7 a2 a2 a2 a4 a5 a6	Number 0 1 1 2 3 3 4 4 5 5 6 7 7 8 9 9 10 11 12 13 14 15 16 17 18 18 18 18 18 18 18 18 18 18 18 18 18	(\$20,000) (\$0.000) (\$
1		25 26 27 28 29 30 31 41 71 81 82 101 111 122 143 177 181 221	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00		Registers Floating Point Name zero re spp PP P	Number 0 1 1 2 3 3 4 5 5 6 7 7 8 9 100 11 12 13 14 15 16 6 17 7	(\$\times 0\$\times 0\$\times 0\$\times 0\$\times 0\$\times 7f \times 10 \times 0\$\times 0
1		25 26 27 28 29 30 31 41 71 81 112 112 114 117 122 148 177 181 221	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 4. Một 00 00 1+92 1+92 7 xem sl và s2 có củn u không, thoát \$ Klým tra xem sl và s2 dỡng à s2 dỡng à s2 đm rán s7		Registers Floating Point Name zero re spp pp pp pp to ti ti ti 22 90 91 a0 a1 a2 a3 a4 a5 a6 a7 a7 a2 a3 a4 a5 a5 a5 a6 a5	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 8 9 1 10 11 12 13 14 15 15 16 17 18 18 19 2 20 2 2 1 2 1	(\$\text{x00}\$) (\$\text{x01}\$) (\$\text{x02}\$) (\$\text{x02}\$) (\$\text{x00}\$)
0 1		25 26 27 28 29 30 31 41 71 81 112 122 143 177 181 221	0x000000 0x000000 0x000000 0x000000 0x000000	00 00 00 00 00 00 00 00 00 00 00 00 00	- A	Registers Floating Point Name zero re sep PP	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 9 9 10 11 11 12 2 13 13 14 15 16 17 18 19 2 2 2 2 2 2 2	(\$20,000) (\$20,0
0 1		25 26 27 28 29 30 31 31 41 72 91 101 112 121 121 122 141 177 188 221	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 4. Một 000 4. Một 000 4. Một 000 4. Một 000 0000 0000 0000 0000 0000 0000 00	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Registers Floating Point Name zero re sero re	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 1 14 15 16 17 18 19 20 2 2 2 2 3 3 2 2 4 4 2 3 3	(\$20,000) (\$20,0
1		25 26 27 28 29 30 31 31 41 72 91 101 112 121 121 122 141 177 188 221	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 000 4. Một 000 4. Một 000 4. Một 000 0000 0000 0000 00000 00000 00000 0000	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Registers Floating Point Name zero re pp pp pp pp to to ti ta ao ai ai ao	Number 0 1 1 2 3 3 4 4 5 5 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 12 2 2 2 2 3 3 4 4 2 2 5 5	(\$\text{xxx}(\$\text{x}
1		25 26 27 28 29 30 31 41 72 81 91 101 112 142 177 182 222 Value 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 000 000 000 0000 0000	p	Registers Floating Point Name zero re pp pp pp pp to to tl tc ze so	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 8 9 10 10 11 1 12 13 14 15 16 17 12 12 12 12 12 12 12 12 12 12 12 12 12	(\$200) (\$
0 1		25 26 27 28 29 30 31 41 71 81 91 101 112 144 177 185 222	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 000 000 000 0000 0000	▼	Registers Floating Point Name zero ze z	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 9 9 10 11 1 12 1 13 1 14 1 15 1 18 1 19 2 2 2 2 3 3 2 2 4 2 5 5 2 2 8	(\$20,000) (\$20,0
		25 26 27 28 29 30 31 31 41 72 81 91 101 112 122 144 177 181 122 140 100 100 100 100 100 100 100 100 100	0x000000 0x000000 0x000000 0x000000 0x000000	000 000 000 000 000 000 000 000 0000 0000	a a a	Registers Floating Point Name zero re pp pp pp pp to to tl tc ze so	Number 0 1 1 2 3 3 4 4 5 5 6 6 7 7 8 8 9 10 10 11 1 12 13 14 15 16 17 12 12 12 12 12 12 12 12 12 12 12 12 12	(\$20,000) (\$20,0

Assignment 2

Viết chương trình thực hiện các công việc sau:

- Trích xuất MSB (byte có trọng số cao) của thanh ghi s0
- Xóa LSB (byte có trọng số thấp) của thanh ghi s0

không bị tràn số và thoát khi gặp lệnh: bge s3, s1, EXIT # s1 và s2 dương

nếu s3 >= s1 thì kết quả không bị tràn số

- Thiết lập LSB của thanh ghi s0 (đặt các bit từ 7 đến 0 thành 1)
- Xóa thanh ghi s0 bằng các lệnh logic (đặt s0 = 0)

```
li s0, 0x12345678 # Giá trị mẫu cho s0

srli t0, s0, 24 # Trích xuất MSB (dịch phải 24 bit)

andi t1, s0, 0xFFFFFF00 # Xóa LSB (AND với 0xFFFFF00 để xóa LSB)

ori t2, s0, 0x000000FF # Thiết lập LSB (OR với 0xFF để đặt các bit từ 7 đến 0)

li s0, 0 # Xóa thanh ghi s0 (đặt s0 = 0)
```

- Thanh ghi s0:

- Ban đầu, s0 được khởi tao với giá tri 0x12345678.
- Ở cuối chương trình, lênh li s0, 0 sẽ đặt giá tri của s0 thành 0.

Giá trị cuối cùng của s0: 0x00000000

- Thanh ghi t0 (trích xuất MSB của s0):
 - Lệnh srli t0, s0, 24 dịch phải giá trị trong s0 24 bit, để lại 8 bit cao nhất (MSB).
 - Giá trị ban đầu của s0 là 0x12345678, khi dịch phải 24 bit, chỉ còn lại giá trị 0x12.

Giá trị cuối cùng của t0: 0x00000012

- Thanh ghi t1 (xóa LSB của s0):
 - Lệnh andi t1, s0, 0xFFFFFF00 sẽ giữ nguyên tất cả các bit ngoại trừ 8 bit thấp nhất (LSB) bằng cách AND với 0xFFFFFF00.
 - Giá trị ban đầu của s0 là 0x12345678, khi AND với 0xFFFFFF00, kết quả sẽ là 0x12345600.

Giá trị cuối cùng của t1: 0x12345600

- Thanh ghi t2 (thiết lập LSB của s0):
 - Lệnh ori t2, s0, 0x000000FF sẽ thiết lập 8 bit thấp nhất của s0 thành 1 (hoặc 0xFF).
 - Giá trị ban đầu của s0 là 0x12345678, khi OR với 0x000000FF, kết quả sẽ là 0x123456FF.

Giá trị cuối cùng của t2: 0x123456FF

Registers	Floating Point	Control and Status	
Name		Number	Value
zero		0	0x00000000
ra		1	0x00000000
sp		2	0x7fffeffc
gp		3	0x10008000
tp		4	0x00000000
t0		5	0x00000012
t1		6	0x12345600
t2		7	0x123456ff
s0		8	0x12345678
s1		9	0x00000000
a0		10	0x00000000
a1		11	0x00000000
a2		12	0x00000000
a.3		13	0x00000000
a.4		14	0x00000000
a5		15	0x00000000
a6		16	0x00000000
a.7		17	0x00000000
s2		18	0x00000000
5 3		19	0x00000000
s4		20	0x00000000
s5		21	0x00000000
s6		22	0x00000000
s7		23	0x00000000
5 8		24	0x00000000
a9		25	0x00000000
s10		26	0x00000000
s11		27	0x00000000
t3		28	0x00000000
t4		29	0x00000000
t5		30	0x00000000
t6		31	0x00000000
DC			0x00400014

Assignment 3

1. Giả lệnh neg s0, s1 (s0 = -s1):

Lệnh này có chức năng lấy giá trị âm của thanh ghi s1 và lưu kết quả vào thanh ghi s0. Trong RISC-V, lệnh chính thống để thực hiện phép lấy giá trị âm là sử dụng lệnh sub với toán hạng x0 (thanh ghi không) để trừ giá trị của s1 từ 0.

sub s0, x0, s1 # s0 = 0 - s1

2. Giả lênh my s0, s1 (s0 = s1):

Lệnh này có chức năng sao chép giá trị từ thanh ghi s1 sang thanh ghi s0. Trong RISC-V, có thể thực hiện việc này bằng cách dùng lênh addi với giá tri công thêm là 0.

addi s0, s1, 0 # s0 = s1 + 0 (sao chép giá tri từ s1 sang s0)

3. Giả lệnh not s0 (s0 = bit invert(s0)):

Lệnh này có chức năng đảo tất cả các bit của thanh ghi s0. Trong RISC-V, có thể thực hiện phép này bằng cách sử dụng lệnh xori với giá trị -1 (hay tất cả các bit là 1 trong số nguyên có dấu).

xori s0, s0, -1 # s0 = s0 XOR (-1) (đảo tất cả các bit của s0)

4. Giả lệnh ble s1, s2, label (nhảy tới label nếu s1 \leq s2):

Lệnh này thực hiện phép so sánh hai thanh ghi s1 và s2, và nếu s1 nhỏ hơn hoặc bằng s2, chương trình sẽ nhảy tới nhãn label. Trong RISC-V, không có lệnh ble (branch if less than or equal), nhưng có thể sử dụng lệnh bge (branch if greater than or equal) để kiểm tra s2 >= s1, tương đương với s1 <= s2.

bge s2, s1, label # Nếu s2 \geq = s1 (tương đương s1 \leq = s2), nhảy tới label

Assignment 4

```
.data
message no overflow: .asciz "No overflow detected.\n"
message overflow:
                     .asciz "Overflow detected!\n"
  .text
  .globl start
start:
  # Giả sử chúng ta cần công hai số dương hoặc hai số âm, ở đây sử dụng hai số dương.
     s0, 0x7FFFFFFF # Toán hạng thứ nhất (số lớn nhất dương)
                  # Toán hạng thứ hai (giá trị cần cộng)
  li s1, 1
  # Cộng hai số
  add t0, s0, s1
                    # t0 = s0 + s1
  # Kiểm tra tràn số:
  # 1. Kiểm tra nếu s0 và s1 đều là số dương
  # 2. Nếu tổng t0 là số âm, điều đó có nghĩa là có tràn số dương.
  bltz s0, check negative # Nếu s0 là số âm, chuyển tới kiểm tra số âm
  bltz s1, check negative # Nếu s1 là số âm, chuyển tới kiểm tra số âm
```

```
bltz t0, overflow
                       # Nếu tổng là âm mà các số hạng đều dương, có tràn dương
     no overflow
                       # Nếu không, không có tràn
check negative:
  # Kiểm tra nếu cả s0 và s1 đều là số âm
  # Nếu tổng là số dương, điều đó có nghĩa là có tràn số âm.
  bgez s0, no overflow # Nếu s0 là số dương, không có tràn số
  bgez s1, no overflow # Nếu s1 là số dương, không có tràn số
  bgez t0, overflow
                        # Nếu tổng là dương mà các số hang đều âm, có tràn âm
no overflow:
  # In ra thông báo không có tràn số
  la a0, message no overflow # Đưa địa chỉ chuỗi "No overflow detected." vào a0
  li a7, 4
                      # Sử dụng syscall 4 (print string)
                      # Thực hiện syscall để in chuỗi
  ecall
                      # Nhảy tới kết thúc chương trình
     end
overflow:
  # In ra thông báo có tràn số
                              # Đưa địa chỉ chuỗi "Overflow detected!" vào a0
  la a0, message overflow
                      # Sử dung syscall 4 (print string)
  li a7, 4
  ecall
                      # Thực hiện syscall để in chuỗi
end:
  li a7, 10
                       # Sử dung syscall 10 (exit)
                      # Thoát chương trình
  ecall
```

Giải thích mã lênh:

1. Khởi tạo hai toán hạng:

- o li s0, 0x7FFFFFFF: Khởi tạo thanh ghi s0 với giá trị lớn nhất dương của số nguyên 32-bit (0x7FFFFFF).
- o li s1, 1: Khởi tạo thanh ghi s1 với giá trị 1.

2. Cộng hai toán hạng:

 Lệnh add t0, s0, s1 thực hiện phép cộng hai số trong thanh ghi s0 và s1, kết quả được lưu vào t0

3. Kiểm tra tràn số dương:

Nếu cả hai số trong s0 và s1 đều dương, ta dùng lệnh bltz để kiểm tra dấu của t0. Nếu t0 âm, thì tràn dương xảy ra, và chương trình sẽ nhảy tới nhãn overflow để in ra thông báo tràn.

4. Kiểm tra tràn số âm:

Nếu cả hai số trong s0 và s1 đều âm, ta kiểm tra nếu tổng trong t0 là số dương. Nếu đúng, tràn âm xảy ra, và chương trình sẽ nhảy tới nhãn overflow.

5. In thông báo:

- Nếu không có tràn số, chương trình nhảy tới nhãn no_overflow và in ra thông báo "No overflow detected."
- Nếu có tràn số, chương trình nhảy tới nhãn overflow và in ra thông báo "Overflow detected!"

6. Thoát chương trình:

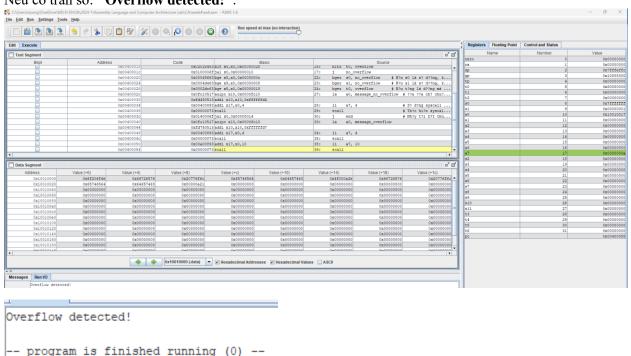
O Chương trình kết thúc bằng lệnh li a7, 10 và thực hiện syscall ecall để thoát chương trình.

Kiểm tra tràn số với các trường hợp:

- **Trường hợp cộng hai số dương lớn**: Khi s0 = 0x7FFFFFFF (giá trị lớn nhất dương) và s1 = 1, kết quả sẽ là tràn dương và chương trình sẽ in thông báo "Overflow detected!"
- Trường hợp cộng hai số âm lớn: Nếu s0 = -2 (ví dụ) và s1 = -2147483647, kết quả sẽ không có tràn vì tổng vẫn nhỏ hơn giá trị nhỏ nhất của số nguyên âm 32-bit.

Kết quả khi chạy chương trình:

• Nếu có tràn số: "Overflow detected!".



RISC-V assembly code để nhân một số với lũy thừa của 2 bằng cách sử dụng dịch trái

li x1, 5 # x1 = 5
li t0, 3 # t0 = 3 (để thực hiện phép nhân với
$$2^3$$
 = 8)

sll x2, x1, t0 # Dịch trái giá trị trong thanh ghi x1 với số bit bằng giá trị trong t0 (3 bit) # Việc này tương đương với phép nhân x1 với 2^3 (5 * 8 = 40)

Sau khi thực hiện lệnh trên, thanh ghi x2 sẽ chứa kết quả của phép nhân x1 * 8 # (kết quả là 40 vì 5 * 8 = 40)

Kết quả:

Registers	Floating Point	Control and Status	
Name		Number	Value
zero		0	0x0000000
ra		1	0x0000000
sp		2	0x7fffeff
gp		3	0x1000800
tp		4	0x0000000
t0		5	0x000000
tl		6	0x0000000
t2		7	0x000000
s0		8	0x000000
sl		9	0x000000
a0		10	0x000000
al		11	0x000000
a2		12	0x000000
a3		13	0x000000
a4		14	0x000000
a5		15	0x000000
a6		16	0x000000
a7		17	0x000000
s2		18	0x000000
s 3		19	0x000000
s4		20	0x000000
s5		21	0x000000
s6		22	0x000000
s7		23	0x000000
s 8		24	0x000000
s 9		25	0x000000
s10		26	0x000000
sll		27	0x000000
t3		28	0x000000
t4		29	0x000000
t5	30		0x000000
t6		31	0x000000
рс			0x004000

s2: 0x00000028: đổi sang decimal là 40, đúng với dự tính từ đầu

Như vậy nếu muốn nhân với lũy thừa của 2, thì ra chỉ cần dịch trái n bit, thì sẽ trả về kết quả là nhân với 2^n