Đại học Quốc gia Thành phố Hồ Chí Minh

Trường Đại học Bách Khoa

Khoa Khoa học và Kĩ thuật Máy tính

BÁO CÁO BÀI TẬP LỚN MÔN KĨ THUẬT MÁY TÍNH

Giáo viên hướng dẫn: Thầy Trần Thanh Bình

Thầy Võ Tấn Phương

Danh sách thành viên

Họ và tên	**MSSV**
Đỗ Đình Cường	1910892
Nguyễn Phi Hùng	1910224
Trần Trung Tuấn	1910666

I Tóm tắt và giải thích chương trình

Để thực hiện phép nhân và chia chương trình gồm các bước cơ bản:

- Nhận vào hai số A,B để xử lý
- Thực hiện phép nhân và xuất kết quả
- Thực hiện phép chia và xuất kết quả

1. Nhận và xử lý số

Tương ứng với mode nhập vào chuỗi input sẽ được xử lý

Mode nhập vào là 0: chuyển từ chuỗi thập phân sang số để tính toán

Mode nhập vào là 1: chuyển từ chuỗi thập lục phân sang số để tính toán

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Dựa vào bảng mã ASCII ta thấy để chuyển đổi kí tự về giá trị thập phân tương ứng ta sẽ thực hiện phép trừ

Với các kí tự từ 0 đến 9 ta sẽ trừ cho 48

Với các kí tự từ A, B, C, D, E, F ta sẽ trừ cho 55

Với hệ thập lục phân:

Ta sẽ đọc từng kí tự của chuỗi và chuyển đổi. Sau đó tính giá trị như chuyển đổi thông thường.

```
hexaStringToDecimalLoop:
    lb $t7, 0 ($t2)
    addi $t4, $0, 57
    slt $t5, $t4, $t7
    beq $t5, $zero, inputSub48        # nếu $t7 bé hơn hoặc bằng char ‘9’ trừ 48
    addi $t7, $t7, -55                # trừ 55 với các kí tự (ABCDEF)
    j inputHexaNormalized
inputSub48:
    addi $t7, $t7, -48                # trừ 48 với các kí tự (0-9)
    j inputHexaNormalized
inputHexaNormalized:
    slt $t5, $t7, $0
    bne $t5, $zero, Convertfinished       # ngưng việc chuyển đổi khi hết chuỗi
    sll $a0, $a0, 4                   # nhân với 16
    add $a0, $a0, $t7                 # cộng vào tổng
    addi $t2, $t2, 1
```

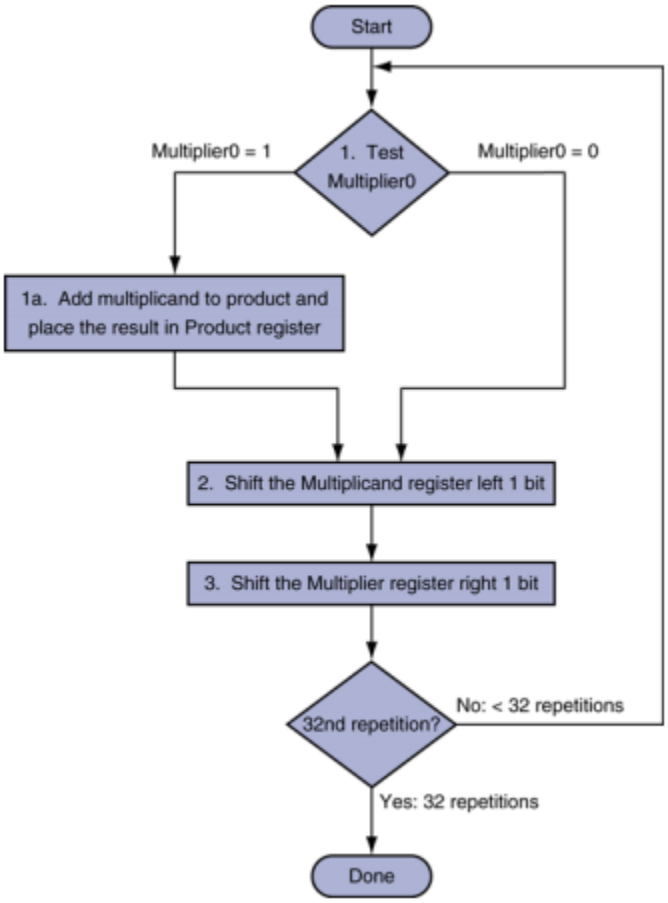
```
        j hexaStringToDecimalLoop
```

Với hệ thập phân ta cũng thực hiện tương tự nhưng không cần xét đến các kí tự A, B, C, D, E, F. Sau khi tính toán ta xét thêm kí tự đầu để xác định giá trị là dương âm hay bằng 0.

```
Pos_or_not:
        la $t2, A
        lb $t7, 0 ($t2)
        beq $t7, '+', pos
        sub $a0, $0, $a0
pos:
```

2. Phép nhân

Trong phép nhân, ta gọi 2 thừa số lần lượt theo thứ tự là số bị nhân và số nhân. Giải thuật nhân hai số theo dạng "giấy bút" là chọn từng kí tự của số nhân với số bị nhân và dịch các giá trị tạm thời này sang trái ở mỗi phép tính. Ta cũng áp dụng tương tự vào nhân 2 số nhị phân với nhau.



Ta duyệt qua 32 bit của số nhân. Mỗi bước ta chọn ra kí tự cuối của số bị nhân và nhân với số bị nhân, sau đó dịch phải số nhân và dịch trái số bị nhân:

```
sll $t3, $t0, 31
srl $t3, $t3, 31        # Lấy phần tử cuối cùng $t3
beq $t3, 1, Nhan        # Kiểm tra 1 hoặc 0
sra $t0, $t0, 1         # Dịch phải $t0
```

```
sll $t1, $t1, 1          # Dịch trái $t1
j loopmul
```

Cập nhật kết quả phép nhân

Nhan:

```
addu $a0, $a0, $t1    # Cập nhật kết quả
sra $t0, $t0, 1          # Dịch phải $t0
sll $t1, $t1, 1          # Dịch trái $t1
j loopmul
```

3. Phép chia

Ta lưu giá trị số bị chia vào thanh ghi $s0 và giá trị số chia vào thanh ghi $s1.

Đầu tiên kiểm tra xem giá trị số chia khác 0 hay không, nếu giá trị số chia bằng 0 ta in ra dòng "không thể chia cho 0" và kết thúc chương trình.

Đối với trường hợp số chia khác 0 ta thực hiện chia hai số theo bộ chia tuần tự tối ưu.

Ta chia làm 4 trường hợp là : số bị chia >= 0 và số chia > 0, số bị chia < 0 và số chia > 0, số bị chia >= 0 và số chia < 0, số bị chia < 0 và số chia < 0 như sau:

```
slt $t5, $s0, $0
beq $t5, $zero, p
slt $t5, $0, $s1
bne $t5, $zero, np
```

Đoạn code cho trường hợp số bị chia < 0 và số chia < 0

```
j end # nhảy đến kết thúc phép chia
```

```
np:
Đoạn code cho trường hợp  số bị chia < 0 và số chia > 0
j end # nhảy đến kết thúc phép chia
p:
slt $t5, $0, $s1
bne $t5, $zero, pp
Đoạn code cho trường hợp  số bị chia >= 0 và số chia < 0
j end # nhảy đến kết thúc phép chia
pp:
Đoạn code cho trường hợp  số bị chia >= 0 và số chia > 0
j end # nhảy đến kết thúc phép chia
```

Đoạn code hiện thực giải thuật bộ chia tuần tự tối ưu cho số nhị phân 16 bit (giá trị thương và số dư lưu lần lượt vào 16 bit cao và 16 bit thấp của một thanh ghi): (1)

```
addi $t1, $0, 16  # tạo biến i = 16 để có 16 vòng lặp
sll $s0, $s0, 16          # dịch trái số bị chia 16 bit
srl $s0, $s0, 16          # dịch phải số bị chia 16 bit
sll $s1, $s1, 16          # dịch trái số chia 16 bit nhằm mục đích thực hiện phép trừ 16 bit cao của thanh ghi $s0
loop:   # vòng lặp
beq $t1, $0, xss  # kết thúc vòng lặp nếu $t1 == 0
subi $t1, $t1, 1 # $t1 = $t1 - 1
sll $s0, $s0, 1 # Dịch trái thanh ghi số bị chia 1 bit
subu $t2, $s0, $s1 #  Gán $t2 = $t0 - $t1
```

slt $t5, $t2, $0

bne $t5, $zero, loop # Kiểm tra thanh ghi số dư với giá trị 0, nếu < 0 thì tiếp tục thực hiện vòng lặp nếu >= thì ta gán $s0 cho giá trị thanh ghi $t2 và chỉnh bit thấp nhất thanh ghi $s0 thành 1

add $s0, $t2, $0

addi $s0, $s0, 1

j loop # thực hiện lại vòng lặp

xss:

sra $s3, $s0, 16 # gán $s3 bằng giá trị 16 bit cao của $s0

gán $s2 bằng giá trị 16 bit thấp của $s0

sll $s2, $s0, 16

sra $s2, $s2, 16

Đoạn code điểu chỉnh thanh ghi $s0 sao cho 16 bit cao chứa giá trị thương, 16 bit thấp chứa giá trị số dư : (2)

addi $s0, $0, 0

add $s0, $s2, $s0

sll $s0, $s0, 16

add $s0, $s0, $s3

- Đối với trường hợp số bị chia >= 0 và số chia > 0 ta thực hiện 2 đoạn code trên.

- Đối với trường hợp số bị chia >= 0 và số chia < 0 ta thực hiện đổi dấu thanh ghi $s1 chứa giá trị số chia rồi thực hiện code theo bộ chia tuần tự tối ưu rồi đổi dấu giá trị thương :

subu $s1, $0, $s1

đoạn code (1)

subu $s2, $0, $s2 # đổi dấu thanh ghi $s2 sau khi thực hiện đoạn code (1)

đoạn code (2)

- Đối với trường hợp số bị chia < 0 và số chia > 0 ta thực hiện:

  subu $s0, $0, $s0 # đổi dấu giá trị $s0

  addu $s6, $s1, $0 # gán giá trị thanh ghi $s6 bằng giá trị số chia

  đoạn code (1)

  subu $s2, $0, $s2 # đổi dấu thanh ghi $s2 sau khi thực hiện đoạn code (1)

  beq $s3, $0, mam # nếu số dư bằng 0 nhảy xuống lệnh mam nếu không ta thực hiện

  subi $s2, $s2,1 # trừ giá trị $s2 một đơn vị

  subu $s3, $s6, $s3 # thay đổi số dư $s3 bằng giá trị của số bị chia trừ đi giá trị $s3

  mam:

  đoạn code (2)

- Đối với trường hợp số bị chia < 0 và số chia < 0 ta thực hiện tương tự như trường hợp trên nhưng trước khi thực hiện ta đổi dấu cả thanh ghi $s0 và $s1.

Như vậy ta có thanh ghi $s2,$s3 chứa lần lượt giá trị thương và số dư, thanh ghi $s0 có 16 bit cao chứa giá trị thương và 16 bit thấp chứa giá trị số dư.

II Testcase, thống kê số lượng lệnh, loại lệnh và thời gian thực thi

Để thống kê số lượng lệnh, loại lệnh ta sẽ dùng các tool hỗ trợ trong Mars 4.5

Với giả thiết máy tính MIPS tần số 2GHz, CPI = 1

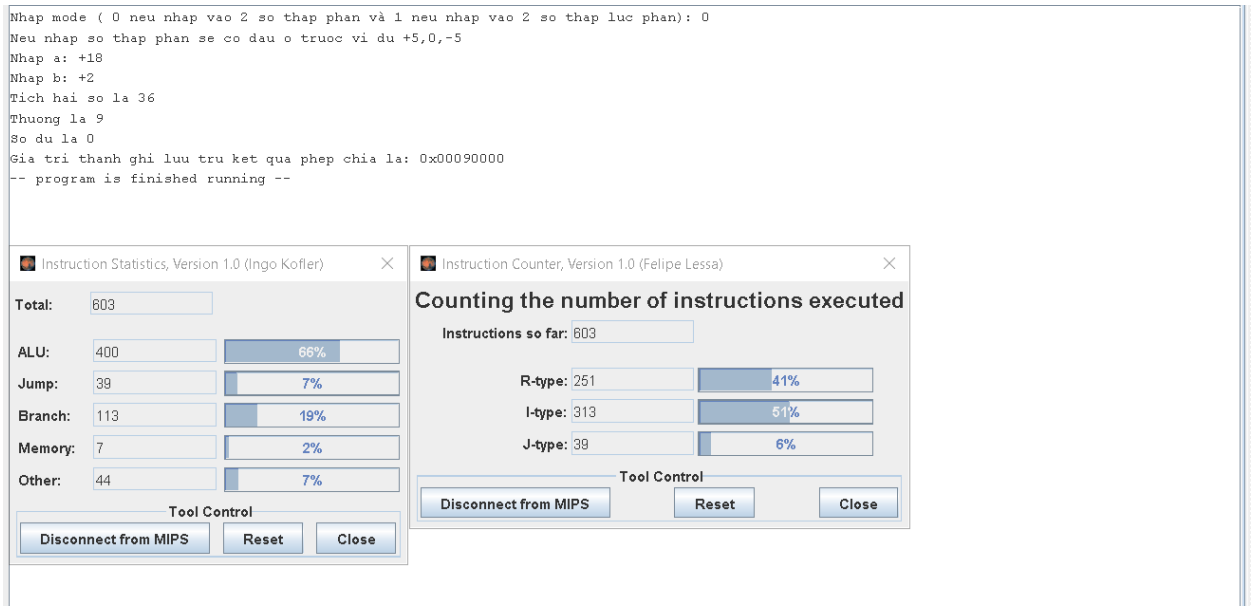
Để tính toán thời gian thực thi của chương trình

Time = Instruction Count * CPI * Clock Cycle Time

= Instruction Count * CPI / Clock Rate

= Instruction Count * 1 / 2 (nanosecond)

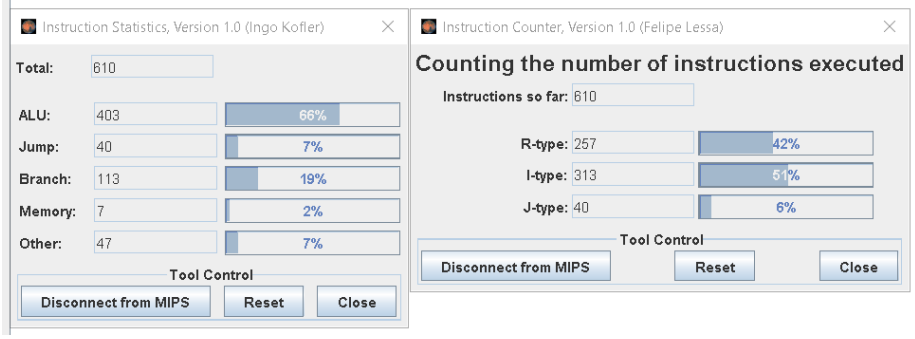
Testcase 1 : 18, 2



Thời gian thực thi là 301.5 ns

Testcase 2 : 47, -9

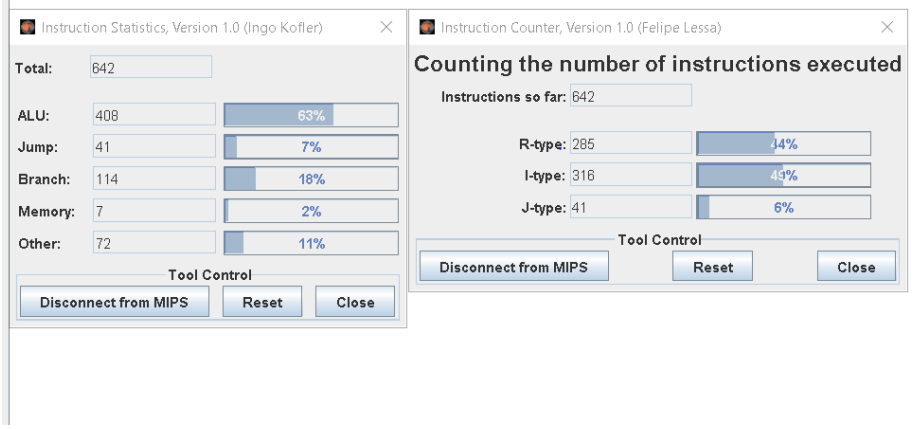
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 0
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: +47
Nhap b: -9
Tich hai so la -423
Thuong la -5
So du la 2
Gia tri thanh ghi luu tru ket qua phep chia la: 0xfffb0002
-- program is finished running --
```



Thời gian thực thi là 305 ns

Testcase 3 : -36, 5

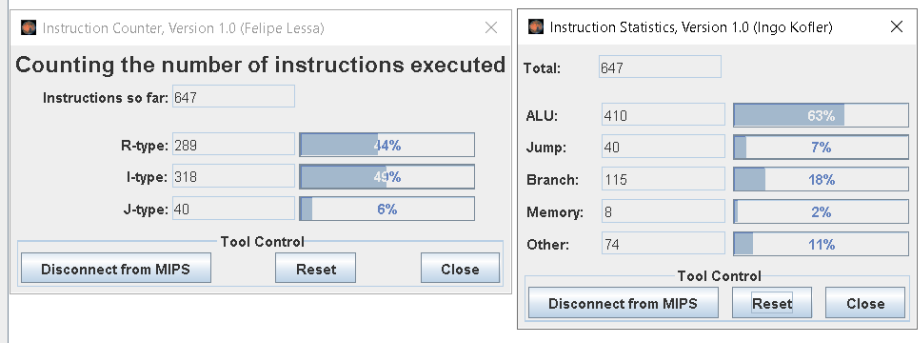
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 0
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: -36
Nhap b: +5
Tich hai so la -180
Thuong la -8
So du la 4
Gia tri thanh ghi luu tru ket qua phep chia la: 0xfff80004
-- program is finished running --
```



Thời gian thực thi là 321 ns

Testcase 4: -81, -40

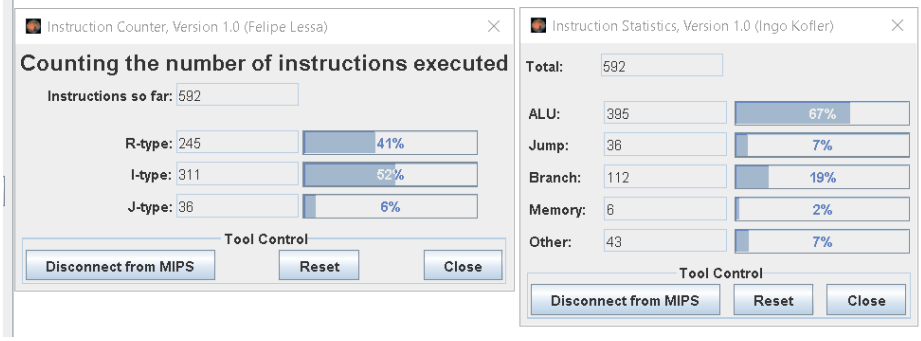
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 0
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: -81
Nhap b: -40
Tich hai so la 3240
Thuong la 3
So du la 39
Gia tri thanh ghi luu tru ket qua phep chia la: 0x00030027
-- program is finished running --
```



Thời gian thực thi là 323.5 ns

Testcase 5: 0, 45

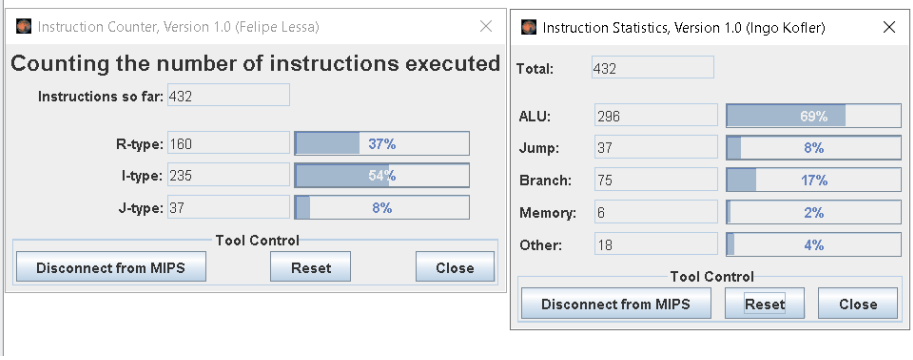
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 0
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: 0
Nhap b: +45
Tich hai so la 0
Thuong la 0
So du la 0
Gia tri thanh ghi luu tru ket qua phep chia la: 0x00000000
-- program is finished running --
```



Thời gian thực thi là 296 ns

Testcase 6: 72, 0

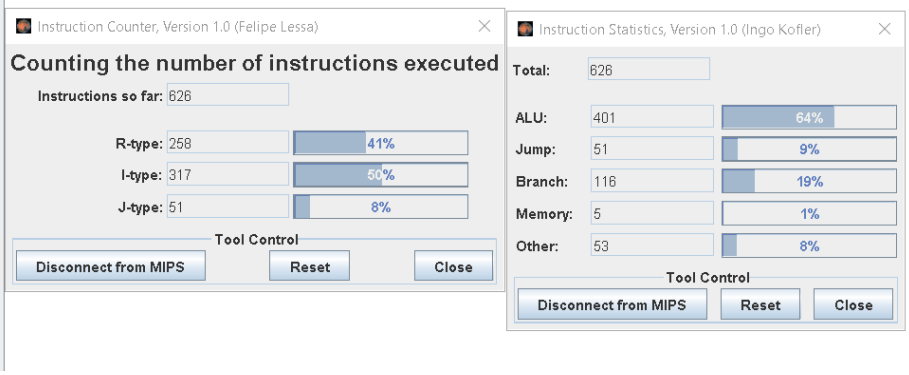
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 0
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: +72
Nhap b: 0
Tich hai so la 0
Khong the chia cho 0
-- program is finished running --
```



Thời gian thực thi là 216 ns

Testcase 7: 0xFA, 0x2

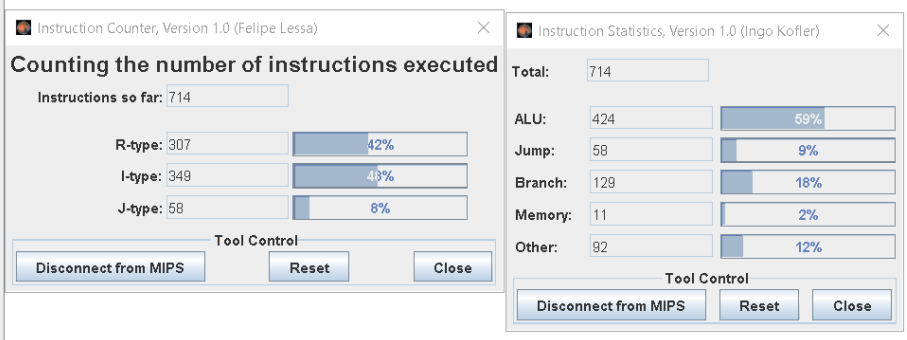
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 1
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: 0xFA
Nhap b: 0x2
Tich hai so la 0x000001f4
Thuong la 0x0000007d
So du la 0x00000000
Gia tri thanh ghi luu tru ket qua phep chia la: 0x007d0000
-- program is finished running --
```



Thời gian thực thi là 313 ns

Testcase 8: 0xFFFFFFFF, 0x8

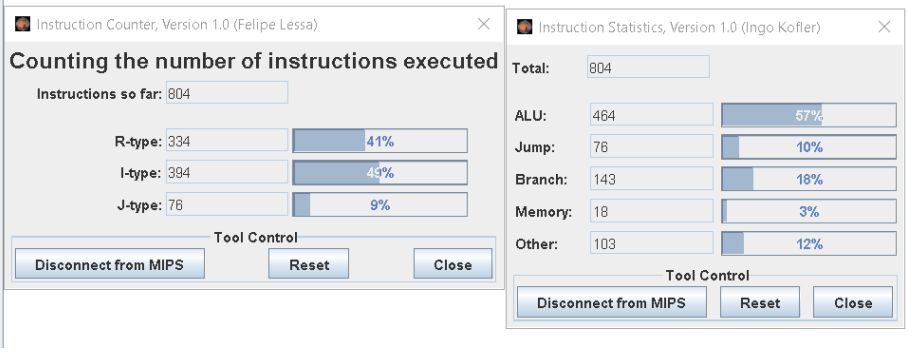
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 1
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: 0xFFFFFFFF
Nhap b: 0x8
Tich hai so la 0xfffffff8
Thuong la 0xffffffff
So du la 0x00000007
Gia tri thanh ghi luu tru ket qua phep chia la: 0xffff0007
-- program is finished running --
```



Thời gian thực thi là 356 ns

Testcase 9: 0xFFFFFFAB,0xFFFFFFFF

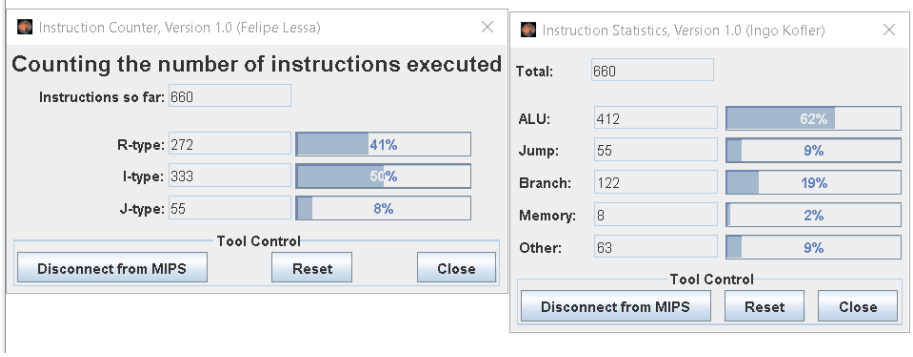
```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 1
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: 0xFFFFFFAB
Nhap b: 0xFFFFFFFF
Tich hai so la 0x00000055
Thuong la 0x00000055
So du la 0x00000000
Gia tri thanh ghi luu tru ket qua phep chia la: 0x00550000
-- program is finished running --
```



Thời gian thực thi là 402 ns

Testcase 10: 0xEF12, 0xABCD

```
Nhap mode ( 0 neu nhap vao 2 so thap phan và 1 neu nhap vao 2 so thap luc phan): 1
Neu nhap so thap phan se co dau o truoc vi du +5,0,-5
Nhap a: 0xABCD
Nhap b: 0xEF
Tich hai so la 0x00a06463
Thuong la 0x000000b8
So du la 0x00000005
Gia tri thanh ghi luu tru ket qua phep chia la: 0x00b80005
-- program is finished running --
```



Thời gian thực thi là 330 ns