

1. Slr instruction in system is slr value,  $a, b$ .

It will set value to 1 if  $a < b$ , otherwise 0.

For comparing if  $a$  is less than  $b$ , computer more prefer

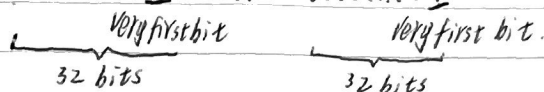
using subtraction:  $a - b$ . If  $a - b < 0$  that means  $a < b$ .

So in the "32-bit ALU", only the very first bit will be set to compare  $a$  and  $b$ , for the rest will be set to false.

So in the very first bit, if  $a < b$  ( $a - b < 0$ ) then less returns true (1)

otherwise less returns false (0). Therefore, the result in the

"32-bit ALU" will be either "0000...00" or "0000...01".



For the control signal, because ALU wants to calculate the subtraction. So  $binvert$  is set to 1. Therefore "01".

For the output of the result, it will be set to less which is "11". Therefore,  $ainvert, binvert$ : "01", operation "11".

The signal values for slr instruction are "0111".

$$2. \ 0101 * 1110 = 0100 \ 0110$$

MD

0101

AC

0000

0000

0000

0101

0101

0010

1010

1110

0011

0101

1000

0100

MR

1110

1110

0111

0111

1011

1011

1101

1101

0110

3. (a)  $1010011101[0] = 413_{10}$   
 $110100111$   
 $256+128+32+(-4)+2+(-1) = 413_{10}$

(b)  $[1010 * 0101] = 1110 0010$

MD	AC	MQ	Q-
1010	0000	0101	0
0101	+ 0110		
+ 1	0110	0101	
0110	0011	0010	1
2's complement	+ 1010		
	1101	0010	
	1110	1001	0
	+ 0110		
	0100	1001	
	0010	0100	1
	1010		
	1100	0100	1
	1110	0010	0
	Product: 1110 0010		



$$11/4 = \overset{Q}{2} \overset{R}{3}$$

$$4. \quad 1011 / 0100 = \overset{R}{0011} \overset{Q}{0010}$$

MD	AC	MR
0100	0000	1011
1	0001	011 □
25 1100	+ 1100	
	1101 < 0	011 □
	1101	011 0
	+ 0100	
	0001	011 0
	0010	110 □
	1100	
	1110 < 0	110 □
	1110	110 0
	+ 0100	
	0010	110 0
	0101	100 □
	+ 1100	
	0001 > 0	100 □
	0001	100 1
	0011	001 □
	+ 1100	
	1111 < 0	001 □
	1111 < 0	001 0
	+ 0100	
	0011	0010
	<u>R</u>	<u>Q</u>

11 4 2 3  
Q R  
5 1011/0100 = 0010 0011

MR	AC	MR
0100	0000	1011
/	0001 > 0	0110
1100 25	+1100	
	1101	0110
	1101	0110
	1010 < 0	1100
	+0100	1100
	1110	1100
	1101 < 0	1000
	+0100	
	0001 > 0	1001
	0011 > 0	0010
	+1100	
	1111 < 0	0010
	+0100	
	0011	0010
	R	R

6. The reason why non-restoring division algorithm achieves higher efficiency is because the non-restoring division only executes if  $(AC < 0)$  then  $AC \leftarrow AC + MD$  on it's last step while restoring division executes that on every steps of it's calculation.