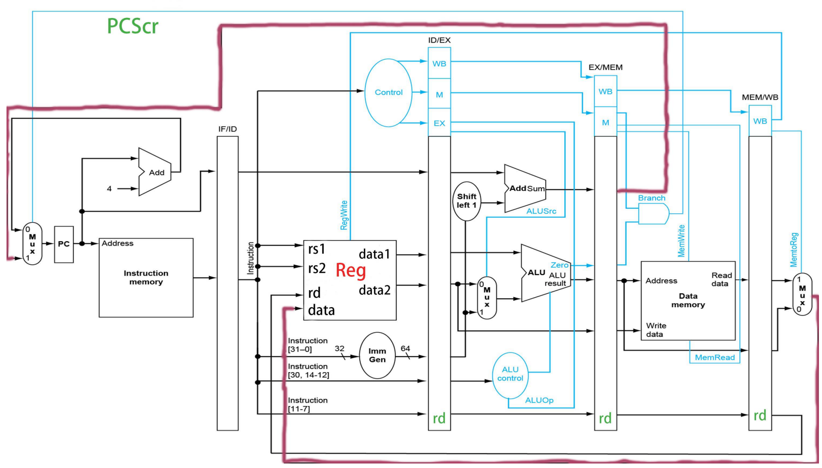

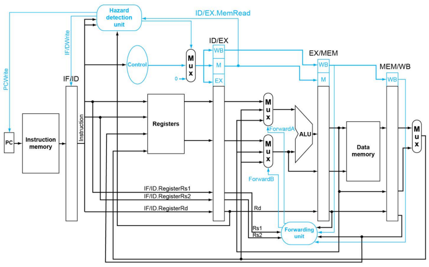

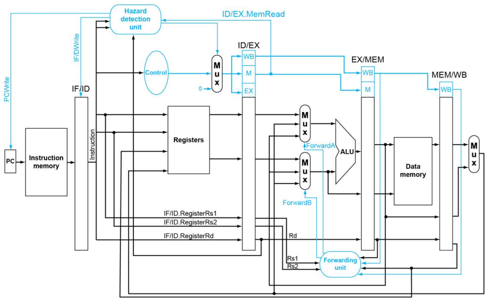

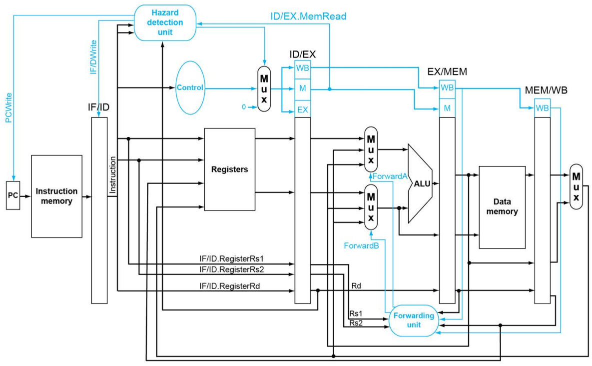

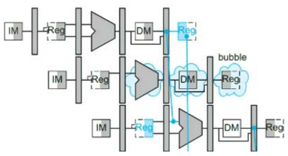

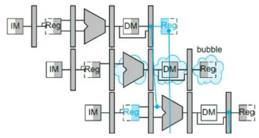

补码一位乘法 $[x]_{补} = x_s.x_1x_2\dots x_n$ $[y]_{补} = y_s.y_1y_2\dots y_n$

① 所有位都参与运算，结果也是补码

② 被乘数和部分积都取双符号位，初值为0，乘数取单符号位，

并且末尾增设初值 $y_{n+1} = 0$

③ 根据 (y_n, y_{n+1})来确定操作

④ 移位按照补码移位的规则

⑤ n+1次操作，n次右移　2 * (-3) = - 6

yn（高位）	yn+1（低位）	部分积操作
0	0	右移1位
1	1	
0	1	加[x]补，右移1位
1	0	加[-x]补，右移1位

iteration	step	Multiplicand	product
0	Initial Values	0010	0000 1101 0
1	1.c:10→Prod=Prod-Mcand	0010	1110 1101 0
	2: shift right Product	0010	1111 0110 1
2	1.b:01→Prod=Prod+Mcand	0010	0001 0110 1
	2: shift right Product	0010	0000 1011 0
3	1.c:10→Prod=Prod-Mcand	0010	1110 1011 0
	2: shift right Product	0010	1111 0101 1
4	1.d: 11 → no operation	0010	1111 0101 1
	2: shift right Product	0010	**1111 1010** 1

原码一位乘法　$[x]_{原} = x_s.x_1x_2\dots x_n$ $[y]_{原} = y_s.y_1y_2\dots y_n$

① 被乘数和乘数均取绝对值参与运算，看作无符号数，符号位为 $x_s \oplus y_s$

②从乘数的**最低位** y_n开始判断，若 $y_n = 1$，则部分积加上被乘数 $|x|$，然后右移一位

若 $y_n = 0$ 则部分积加上0，右移一位　62×12

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	110 010	000 000 001 010
1	lsb=0, no op	110 010	000 000 001 010
	Rshift Product	110 010	000 000 000 101
2	Prod=Prod+Mcand	110 010	110 010 000 101
	Rshift Mplier	110 010	011 001 000 010
3	lsb=0, no op	110 010	011 001 000 010
	Rshift Mplier	110 010	001 100 100 001
4	Prod=Prod+Mcand	110 010	111 110 100 001
	Rshift Mplier	110 010	011 111 010 000
5	lsb=0, no op	110 010	011 111 010 000
	Rshift Mplier	110 010	001 111 101 000
6	lsb=0, no op	110 010	001 111 101 000
	Rshift Mplier	110 010	000 111 110 100

减法用补码加法实现，商的符号由异或实现。 被除数$[x]_{原} = x_s.x_1x_2\dots x_n$ ，除数$[y]_{原} = y_s.y_1y_2\dots y_n$

① 商的符号 $Q_s = x_s \oplus y_s$　商的数值$|Q_s| = \frac{|x_s|}{|y_s|}$

② 先用被除数减去除数 $|x| - |y| = |x| + (-|y|) = |x| + [-|y|]_{补}$

当余数为正时，商加上1，余数和商左移一位，再减去除数；
当余数为负时，商加上0，余数和商左移一位，再加上除数；

③ 当第n+1步余数为负时，则加上 $|y|$　得到第n+1步正确的余数（余数和被除数同号）

74/21 ＝ 3 remainder 11

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	010 001	000 000 111 100
1	R<<	010 001	000 001 111 000
	Rem=Rem–Div	010 001	111 000 111 000
	Rem<0,R+D	010 001	000 001 111 000
2	R<<	010 001	000 011 110 000
	Rem=Rem–Div	010 001	110 010 110 000
	Rem<0,R+D	010 001	000 011 110 000
3	R<<	010 001	000 111 100 000
	Rem=Rem–Div	010 001	110 110 110 000
	Rem<0,R+D	010 001	000 111 100 000
4	R<<	010 001	001 111 000 000
	Rem=Rem–Div	010 001	111 110 000 000
	Rem<0,R+D	010 001	001 111 000 000

减法用补码加法实现，商的符号由异或实现。 被除数$[x]_{原} = x_s.x_1x_2\dots x_n$ ，除数$[y]_{原} = y_s.y_1y_2\dots y_n$

① 商的符号 $Q_s = x_s \oplus y_s$　商的数值$|Q_s| = \frac{|x_s|}{|y_s|}$

② 先用被除数减去除数 $|x| - |y| = |x| + (-|y|) = |x| + [-|y|]_{补}$

当余数为正时，商加上1，余数和商左移一位，再减去除数；
当余数为负时，商加上0，余数和商左移一位，再加上除数；

③ 当第n+1步余数为负时，则加上 $|y|$　得到第n+1步正确的余数（余数和被除数同号）

74/21 ＝ 3 remainder 11

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	010 001	000 000 111 100
1	R<<	010 001	000 001 111 000
	Rem=Rem–Div	010 001	111 000 111 000
	Rem<0,R+D	010 001	000 001 111 000
2	R<<	010 001	000 011 110 000
	Rem=Rem–Div	010 001	110 010 110 000
	Rem<0,R+D	010 001	000 011 110 000
3	R<<	010 001	000 111 100 000
	Rem=Rem–Div	010 001	110 110 110 000
	Rem<0,R+D	010 001	000 111 100 000
4	R<<	010 001	001 111 000 000
	Rem=Rem–Div	010 001	111 110 000 000
	Rem<0,R+D	010 001	001 111 000 000


```
int fact (int n)
{
    if (n < 1) return (1);
        else return (n * fact(n-1));
}
```

```
fact:  addi   sp, sp, -16     // adjust stack for 2 items
       sd     x1, 8(sp)       // save return address
       sd     x10, 0(sp)      // save n
       addi   x5, x10, -1     // x5 = n - 1
       bge    x5, x0, L1      // if n >= 1, go to L1(else)
       addi   x10, x0, 1      // return 1 if n <1
       addi   sp, sp, 16      // Recover sp
       jalr   x0, 0(x1)       // return to caller
L1:    addi   x10, x10, -1
       jal    x1, fact
       add    x6, x10, x0
       ld     x10, 0(sp)
       ld     x1, 8(sp)
       addi   sp, sp, 16
       mul    x10, x10, x6
       jalr   x0, 0(x1)
```