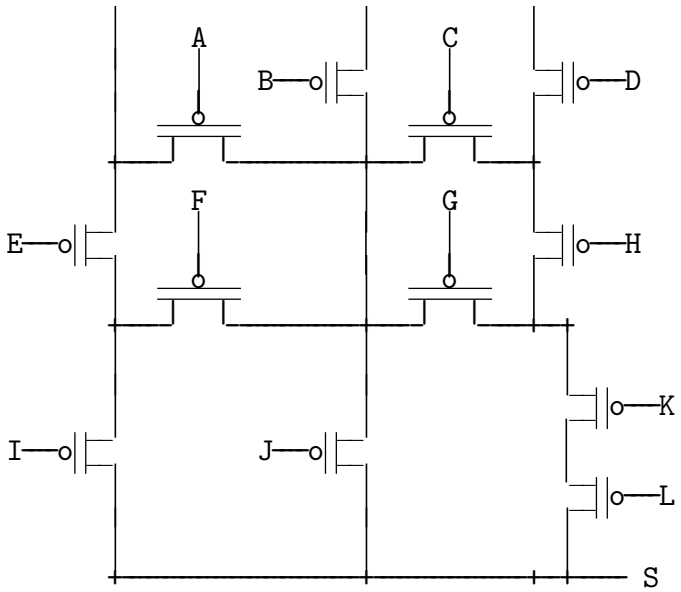
Université de Nanterre mardi 26 juin 2018
architecture des ordinateurs contrôle

Que valent CF, OF, ZF et SF après les opérations 7+11, 6-4, 3-12, 2-9, 14-9, 0+8, 12-7, 13-4, 8+11, 11-13, 11+13, 2+5, 4-6, 6+6, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r4
    add  r3,r4,r4
    add  r4,r4,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

Donnez un équivalent simple en C de la procédure `p`.

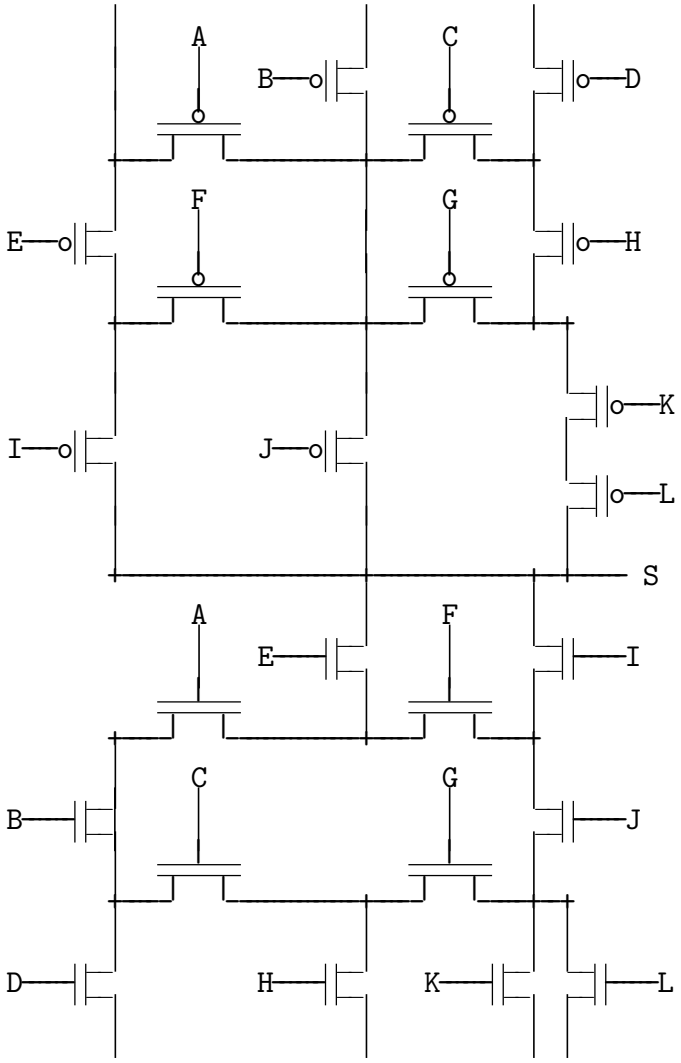
Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{31}.

Corrigé

non signé	signé	CF	OF	ZF	SF
7+ 11= 2	7+ -5= 2	1	0	0	0
6- 4= 2	6- 4= 2	0	0	0	0
3- 12= 7	3- -4= 7	1	0	0	0
2- 9= 9	2- -7=-7	1	1	0	1
14- 9= 5	-2- -7= 5	0	0	0	0
0+ 8= 8	0+ -8=-8	0	0	0	1
12- 7= 5	-4- 7= 5	0	1	0	0
13- 4= 9	-3- 4=-7	0	0	0	1
8+ 11= 3	-8+ -5= 3	1	1	0	0
11- 13=14	-5- -3=-2	1	0	0	1
11+ 13= 8	-5+ -3=-8	1	0	0	1
2+ 5= 7	2+ 5= 7	0	0	0	0
4- 6=14	4- 6=-2	1	0	0	1
6+ 6=12	6+ 6=-4	0	1	0	1



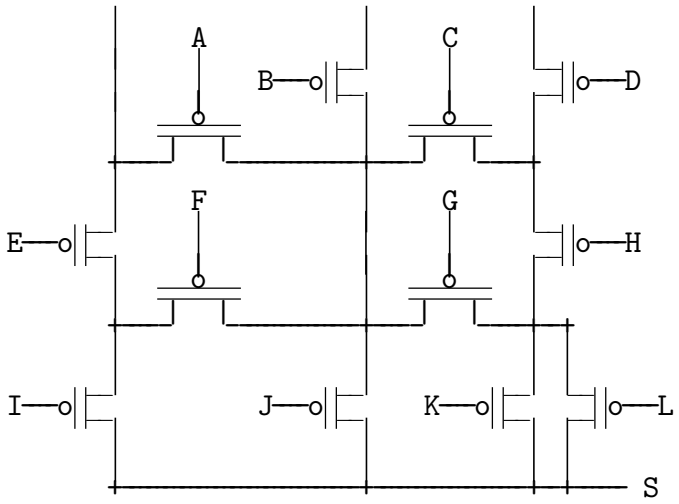
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r4  //     y=x+y;          // a+a^4
    add  r3,r4,r4  //     y=x+y;
    add  r4,r4,r3  //     x=y+y;          // 4a+2a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,6,-2,40,24,3};`
`p(1,&a);`est équivalent à `a=4*a+2*a`^{4}`;`

```
 f: mul r0,r0,r1  // x^2
    mul r0,r1,r1  // x^3
    mul r1,r1,r1  // x^6
    mul r0,r1,r1  // x^7
    mul r1,r1,r1  // x^14
    mul r0,r1,r1  // x^15
    mul r1,r1,r1  // x^30
    mul r0,r1,r0  // x^31
    ret
```

Que valent CF, OF, ZF et SF après les opérations 11-9, 0+15, 13+14, 4-14, 5+14, 14-0, 0+5, 12-5, 8-10, 9+11, 3-6, 6-11, 6+7, 6-3, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r4
    add  r3,r4,r4
    add  r4,r3,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

Donnez un équivalent simple en C de la procédure `p`.

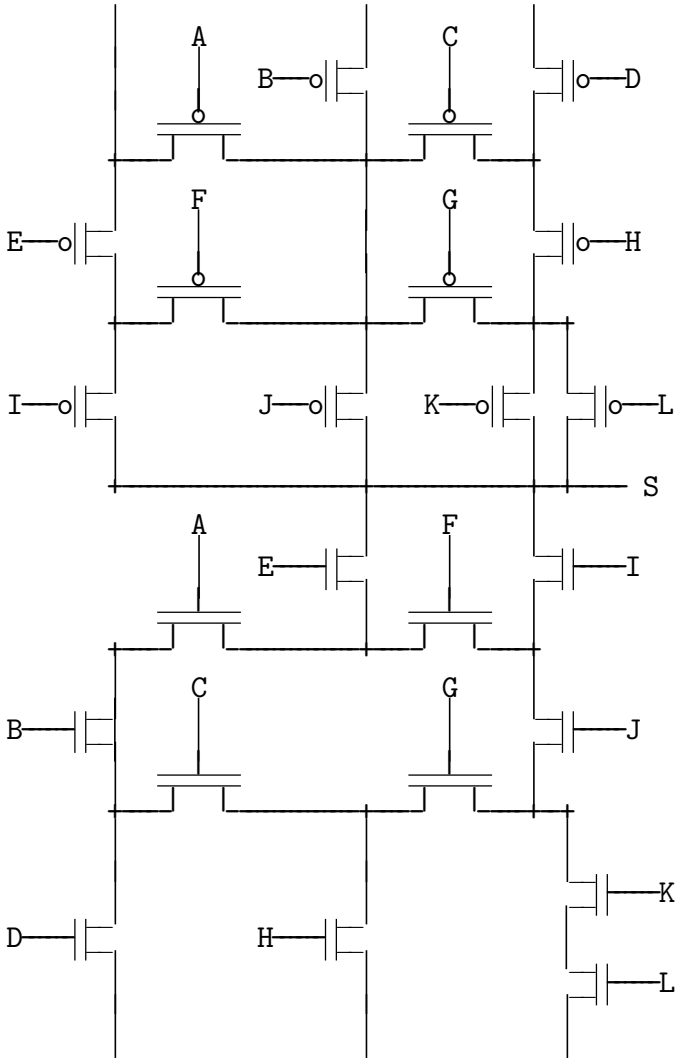
Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{29}.

Corrigé

non signé	signé	CF	OF	ZF	SF
11- 9= 2	-5- -7= 2	0	0	0	0
0+ 15=15	0+ -1=-1	0	0	0	1
13+ 14=11	-3+ -2=-5	1	0	0	1
4- 14= 6	4- -2= 6	1	0	0	0
5+ 14= 3	5+ -2= 3	1	0	0	0
14- 0=14	-2- 0=-2	0	0	0	1
0+ 5= 5	0+ 5= 5	0	0	0	0
12- 5= 7	-4- 5= 7	0	1	0	0
8- 10=14	-8- -6=-2	1	0	0	1
9+ 11= 4	-7+ -5= 4	1	1	0	0
3- 6=13	3- 6=-3	1	0	0	1
6- 11=11	6- -5=-5	1	1	0	1
6+ 7=13	6+ 7=-3	0	1	0	1
6- 3= 3	6- 3= 3	0	0	0	0



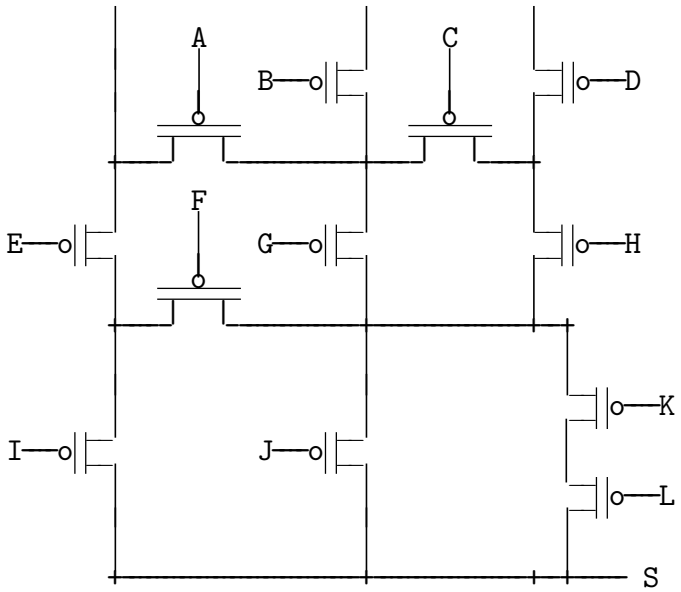
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r4  //     y=x+y;          // a+a^4
    add  r3,r4,r4  //     y=x+y;
    add  r4,r3,r3  //     x=y+x;          // 3a+1a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,4,-2,22,10,3};`
`p(1,&a);`est équivalent à `a=3*a+1*a`^{4}`;`

```
 f: mul r0,r0,r1  // x^2
    mul r0,r1,r1  // x^3
    mul r1,r1,r1  // x^6
    mul r0,r1,r1  // x^7
    mul r1,r1,r1  // x^14
    mul r1,r1,r1  // x^28
    mul r0,r1,r0  // x^29
    ret
```

Que valent CF, OF, ZF et SF après les opérations 8-9, 8+9, 0-9, 9-1, 3+4, 2-3, 11-10, 1+8, 12-6, 5-13, 3+14, 7-7, 11+13, 3+5, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r4
    add  r3,r4,r3
    add  r4,r4,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

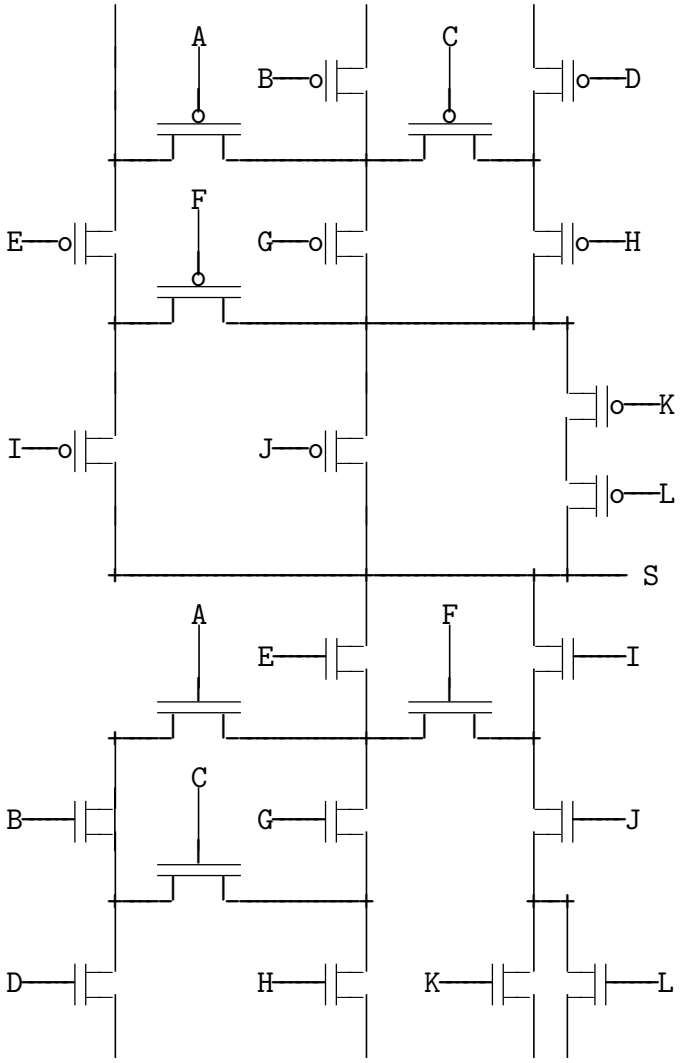
Donnez un équivalent simple en C de la procédure `p`.

Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{27}.

				Corrigé	
non signé	signé	CF	OF	ZF	SF
8- 9=15	-8- -7=-1	1	0	0	1
8+ 9= 1	-8+ -7= 1	1	1	0	0
0- 9= 7	0- -7= 7	1	0	0	0
9- 1= 8	-7- 1=-8	0	0	0	1
3+ 4= 7	3+ 4= 7	0	0	0	0
2- 3=15	2- 3=-1	1	0	0	1
11- 10= 1	-5- -6= 1	0	0	0	0
1+ 8= 9	1+ -8=-7	0	0	0	1
12- 6= 6	-4- 6= 6	0	1	0	0
5- 13= 8	5- -3=-8	1	1	0	1
3+ 14= 1	3+ -2= 1	1	0	0	0
7- 7= 0	7- 7= 0	0	0	1	0
11+ 13= 8	-5+ -3=-8	1	0	0	1
3+ 5= 8	3+ 5=-8	0	1	0	1



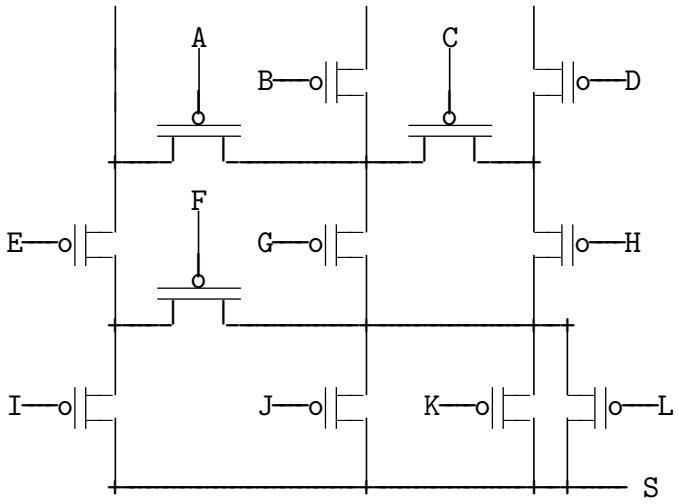
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r4  //     y=x+y;          // a+a^4
    add  r3,r4,r3  //     x=x+y;
    add  r4,r4,r3  //     x=y+y;          // 2a+2a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,4,0,36,28,3};`
`p(1,&a);`est équivalent à `a=2*a+2*a`^{4}`;`

```
 f: mul r0,r0,r1  // x^2
    mul r0,r1,r1  // x^3
    mul r1,r1,r1  // x^6
    mul r1,r1,r1  // x^12
    mul r0,r1,r1  // x^13
    mul r1,r1,r1  // x^26
    mul r0,r1,r0  // x^27
    ret
```

Que valent CF, OF, ZF et SF après les opérations 2-6, 12-12, 6-5, 3+4, 0+14, 0-11, 13-4, 10-12, 12-7, 6-10, 10+10, 14+15, 6+14, 5+6, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r4
    add  r3,r4,r3
    add  r4,r3,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

Donnez un équivalent simple en C de la procédure `p`.

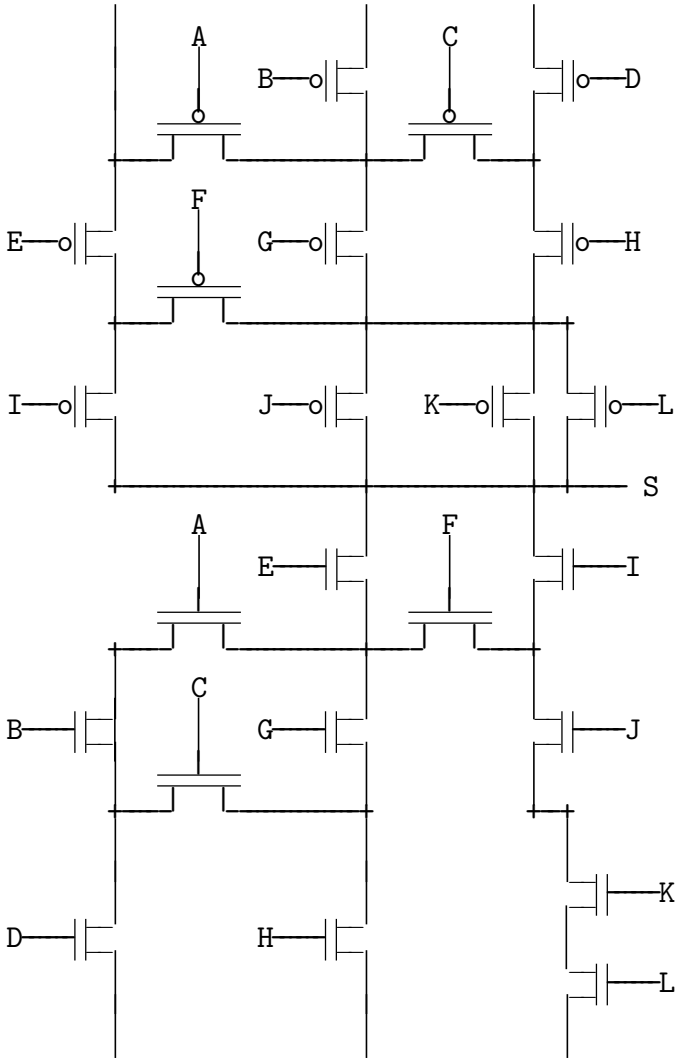
Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{25}.

Corrigé

non signé	signé	CF	OF	ZF	SF
2- 6=12	2- 6=-4	1	0	0	1
12- 12= 0	-4- -4= 0	0	0	1	0
6- 5= 1	6- 5= 1	0	0	0	0
3+ 4= 7	3+ 4= 7	0	0	0	0
0+ 14=14	0+ -2=-2	0	0	0	1
0- 11= 5	0- -5= 5	1	0	0	0
13- 4= 9	-3- 4=-7	0	0	0	1
10- 12=14	-6- -4=-2	1	0	0	1
12- 7= 5	-4- 7= 5	0	1	0	0
6- 10=12	6- -6=-4	1	1	0	1
10+ 10= 4	-6+ -6= 4	1	1	0	0
14+ 15=13	-2+ -1=-3	1	0	0	1
6+ 14= 4	6+ -2= 4	1	0	0	0
5+ 6=11	5+ 6=-5	0	1	0	1



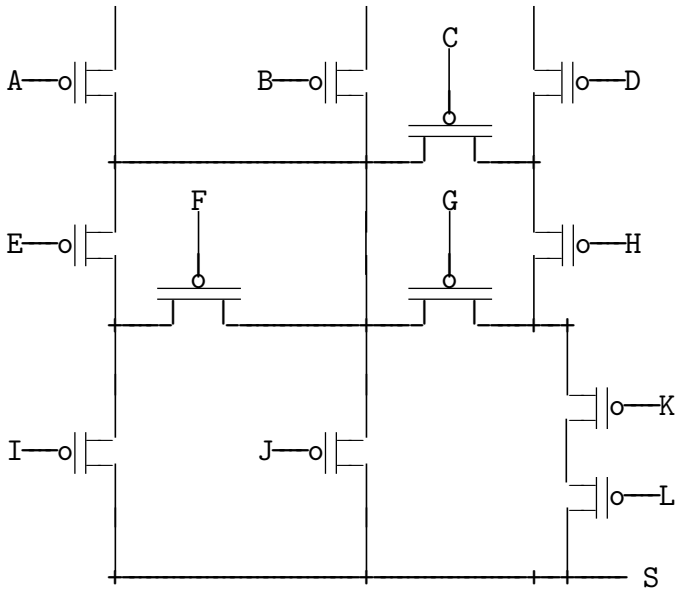
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r4  //     y=x+y;          // a+a^4
    add  r3,r4,r3  //     x=x+y;
    add  r4,r3,r3  //     x=y+x;          // 3a+2a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,5,-1,38,26,3};`
`p(1,&a);`est équivalent à `a=3*a+2*a`^{4}`;`

```
f: mul r0,r0,r1  // x^2
   mul r0,r1,r1  // x^3
   mul r1,r1,r1  // x^6
   mul r1,r1,r1  // x^12
   mul r1,r1,r1  // x^24
   mul r0,r1,r0  // x^25
   ret
```

Que valent CF, OF, ZF et SF après les opérations 7-14, 14-7, 5+10, 5-3, 14-6, 14-14, 9-15, 11+15, 5+13, 8+8, 5-7, 0-9, 0+6, 6+7, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r3
    add  r3,r4,r4
    add  r4,r4,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

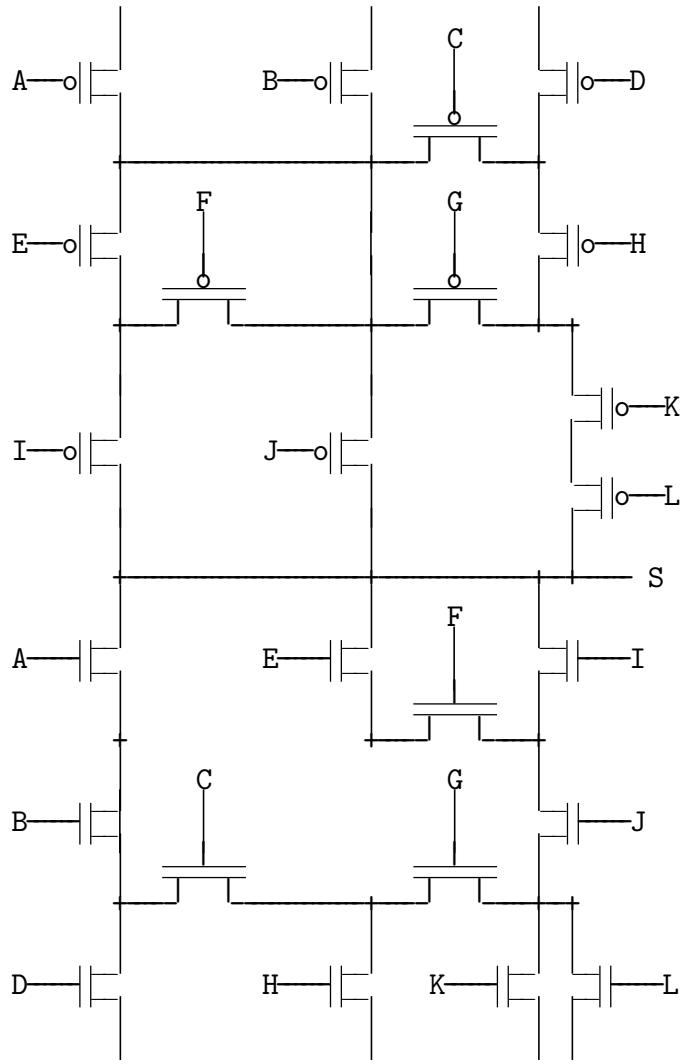
Donnez un équivalent simple en C de la procédure `p`.

Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{23}.

					Corrigé
non signé	signé	CF	OF	ZF	SF
7- 14= 9	7- -2=-7	1	1	0	1
14- 7= 7	-2- 7= 7	0	1	0	0
5+ 10=15	5+ -6=-1	0	0	0	1
5- 3= 2	5- 3= 2	0	0	0	0
14- 6= 8	-2- 6=-8	0	0	0	1
14- 14= 0	-2- -2= 0	0	0	1	0
9- 15=10	-7- -1=-6	1	0	0	1
11+ 15=10	-5+ -1=-6	1	0	0	1
5+ 13= 2	5+ -3= 2	1	0	0	0
8+ 8= 0	-8+ -8= 0	1	1	1	0
5- 7=14	5- 7=-2	1	0	0	1
0- 9= 7	0- -7= 7	1	0	0	0
0+ 6= 6	0+ 6= 6	0	0	0	0
6+ 7=13	6+ 7=-3	0	1	0	1



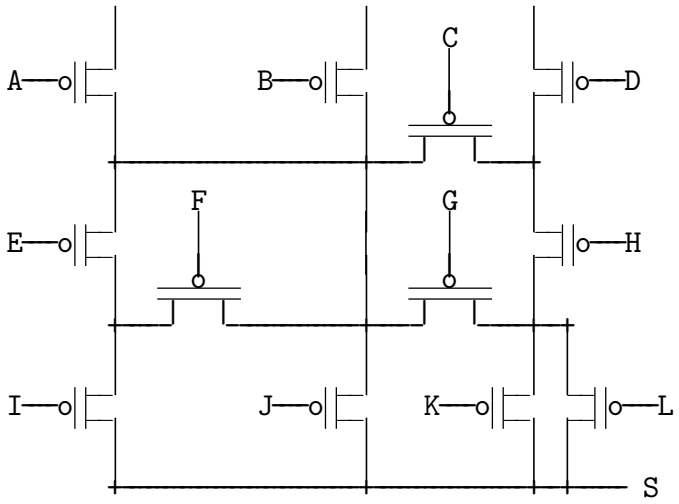
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r3  //     x=x+y;          // a+a^4
    add  r3,r4,r4  //     y=x+y;
    add  r4,r4,r3  //     x=y+y;          // 2a+4a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);` est équivalent à `int t[]={0,6,2,68,60,3};`
`p(1,&a);` est équivalent à `a=2*a+4*a^4;`

```
 f: mul r0,r0,r1  // x^2
    mul r1,r1,r1  // x^4
    mul r0,r1,r1  // x^5
    mul r1,r1,r1  // x^10
    mul r0,r1,r1  // x^11
    mul r1,r1,r1  // x^22
    mul r0,r1,r0  // x^23
    ret
```

Que valent CF, OF, ZF et SF après les opérations 9-2, 12+13, 5+6, 8+8, 9-13, 5-8, 0+0, 1-3, 6-1, 11-10, 3+10, 4-13, 13-1, 5+15, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r3
    add  r3,r4,r4
    add  r4,r3,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

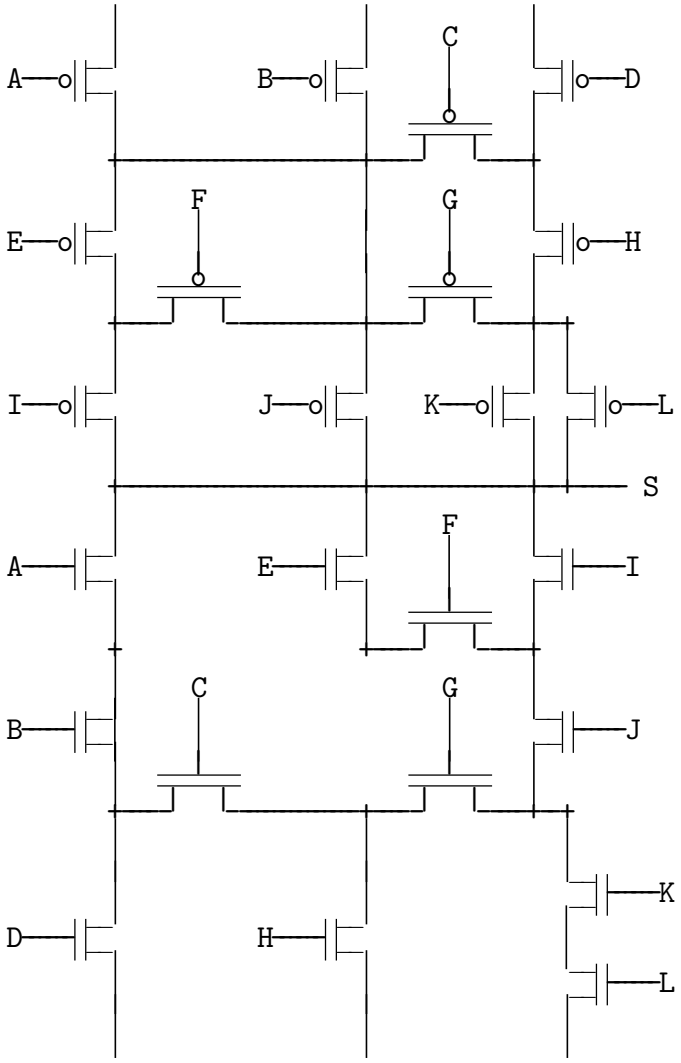
Donnez un équivalent simple en C de la procédure `p`.

Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{21}.

					Corrigé
non signé	signé	CF	OF	ZF	SF
9- 2= 7	-7- 2= 7	0	1	0	0
12+ 13= 9	-4+ -3=-7	1	0	0	1
5+ 6=11	5+ 6=-5	0	1	0	1
8+ 8= 0	-8+ -8= 0	1	1	1	0
9- 13=12	-7- -3=-4	1	0	0	1
5- 8=13	5- -8=-3	1	1	0	1
0+ 0= 0	0+ 0= 0	0	0	1	0
1- 3=14	1- 3=-2	1	0	0	1
6- 1= 5	6- 1= 5	0	0	0	0
11- 10= 1	-5- -6= 1	0	0	0	0
3+ 10=13	3+ -6=-3	0	0	0	1
4- 13= 7	4- -3= 7	1	0	0	0
13- 1=12	-3- 1=-4	0	0	0	1
5+ 15= 4	5+ -1= 4	1	0	0	0



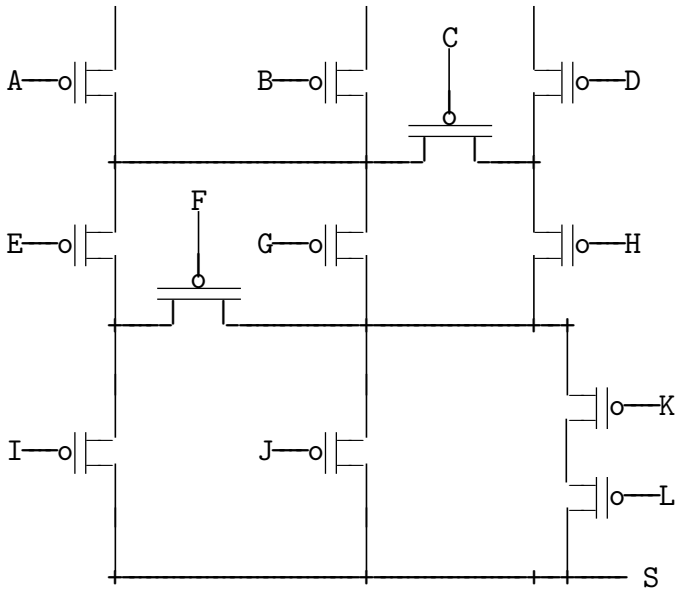
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r3  //     x=x+y;          // a+a^4
    add  r3,r4,r4  //     y=x+y;
    add  r4,r3,r3  //     x=y+x;          // 2a+3a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,5,1,52,44,3};`
`p(1,&a);`est équivalent à `a=2*a+3*a`^{4}`;`

```
f: mul r0,r0,r1  // x^2
   mul r1,r1,r1  // x^4
   mul r0,r1,r1  // x^5
   mul r1,r1,r1  // x^10
   mul r1,r1,r1  // x^20
   mul r0,r1,r0  // x^21
   ret
```

Que valent CF, OF, ZF et SF après les opérations 6-3, 0+9, 11-5, 11-2, 9-8, 3-15, 13-15, 13+14, 1-7, 5-8, 5+12, 0+6, 8+10, 5+7, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r3
    add  r3,r4,r3
    add  r4,r4,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

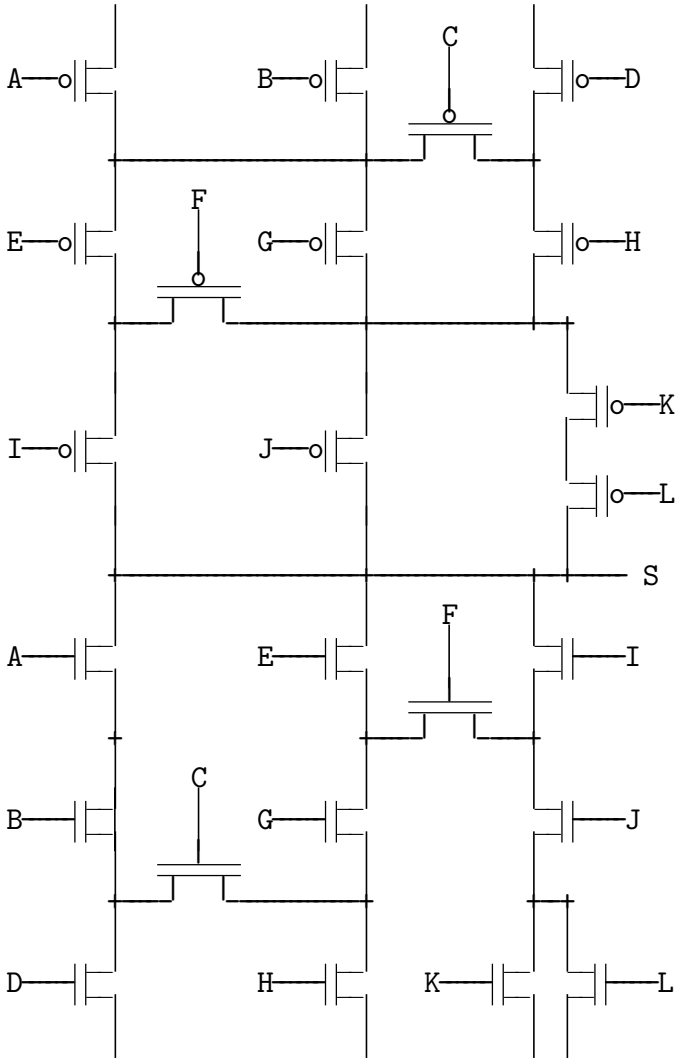
Donnez un équivalent simple en C de la procédure `p`.

Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{19}.

					Corrigé
non signé	signé	CF	OF	ZF	SF
6- 3= 3	6- 3= 3	0	0	0	0
0+ 9= 9	0+ -7=-7	0	0	0	1
11- 5= 6	-5- 5= 6	0	1	0	0
11- 2= 9	-5- 2=-7	0	0	0	1
9- 8= 1	-7- -8= 1	0	0	0	0
3- 15= 4	3- -1= 4	1	0	0	0
13- 15=14	-3- -1=-2	1	0	0	1
13+ 14=11	-3+ -2=-5	1	0	0	1
1- 7=10	1- 7=-6	1	0	0	1
5- 8=13	5- -8=-3	1	1	0	1
5+ 12= 1	5+ -4= 1	1	0	0	0
0+ 6= 6	0+ 6= 6	0	0	0	0
8+ 10= 2	-8+ -6= 2	1	1	0	0
5+ 7=12	5+ 7=-4	0	1	0	1



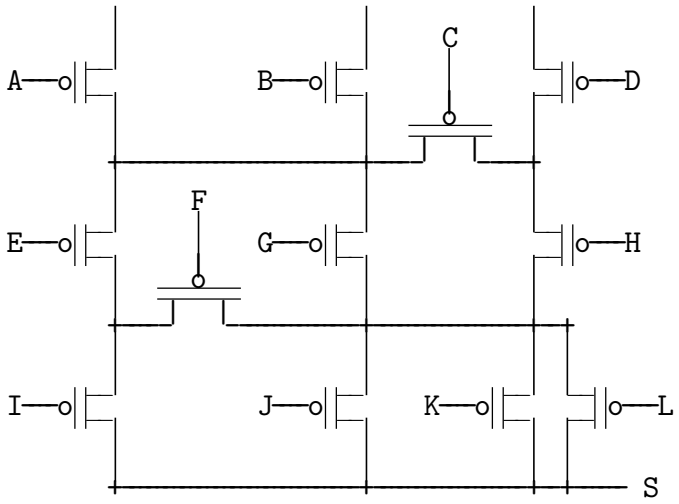
```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r3  //     x=x+y;          // a+a^4
    add  r3,r4,r3  //     x=x+y;
    add  r4,r4,r3  //     x=y+y;          // 0a+2a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,2,2,32,32,3};`
`p(1,&a);`est équivalent à `a=0*a+2*a`^{4}`;`

```
f: mul r0,r0,r1  // x^2
   mul r1,r1,r1  // x^4
   mul r1,r1,r1  // x^8
   mul r0,r1,r1  // x^9
   mul r1,r1,r1  // x^18
   mul r0,r1,r0  // x^19
   ret
```

Que valent CF, OF, ZF et SF après les opérations 0-15, 13-13, 0+13, 7-11, 2-0, 14+15, 8-10, 9-4, 0+0, 12-4, 7+14, 2-7, 8+8, 6+6, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
p:  loadimm16 r2,1
l1: sub  r0,r2,r0
    jc   l2
    load r1,r3
    mul  r3,r3,r4
    mul  r4,r4,r4
    add  r3,r4,r3
    add  r3,r4,r3
    add  r4,r3,r3
    store r1,r3
    add  r1,r2,r1
    jmp  l1
l2: ret
```

Donnez un équivalent simple en C de la procédure `p`.

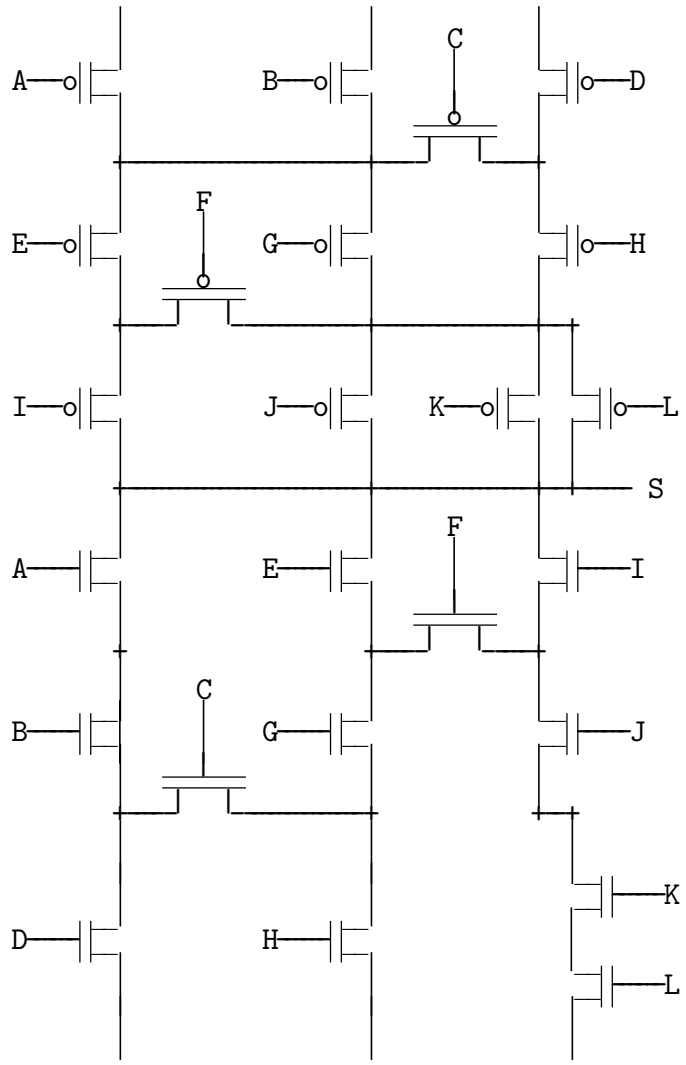
Que contient `t` après `int t[]={0,1,-1,2,-2,3}; p(5,t);`?

Donnez une formule donnant la nouvelle valeur de `x` en fonction de l'ancienne après `p(1,&x);`

Ecrivez en C puis en assembleur la fonction `int f(int x);` qui rend x^{17}.

Corrigé

non signé	signé	CF	OF	ZF	SF
0- 15= 1	0- -1= 1	1	0	0	0
13- 13= 0	-3- -3= 0	0	0	1	0
0+ 13=13	0+ -3=-3	0	0	0	1
7- 11=12	7- -5=-4	1	1	0	1
2- 0= 2	2- 0= 2	0	0	0	0
14+ 15=13	-2+ -1=-3	1	0	0	1
8- 10=14	-8- -6=-2	1	0	0	1
9- 4= 5	-7- 4= 5	0	1	0	0
0+ 0= 0	0+ 0= 0	0	0	1	0
12- 4= 8	-4- 4=-8	0	0	0	1
7+ 14= 5	7+ -2= 5	1	0	0	0
2- 7=11	2- 7=-5	1	0	0	1
8+ 8= 0	-8+ -8= 0	1	1	1	0
6+ 6=12	6+ 6=-4	0	1	0	1



```
                  //           r0    r1    r2   r3   r4
p:  loadimm16 r2,1 // void p(int n,int*t) // 1    x    y
l1: sub  r0,r2,r0
    jc   l2        // { while(n--)
    load r1,r3     //   { int x=*t,       // a
    mul  r3,r3,r4  //         y=x*x;      // a^2
    mul  r4,r4,r4  //     y*=y;           // a^4
    add  r3,r4,r3  //     x=x+y;          // a+a^4
    add  r3,r4,r3  //     x=x+y;
    add  r4,r3,r3  //     x=y+x;          // 1a+3a^4
    store r1,r3    //     *t++=x;
    add  r1,r2,r1
    jmp  l1        //   }
l2: ret            // }
```

`int t[]={0,1,-1,2,-2,3}, x=f(5,t);`est équivalent à `int t[]={0,4,2,50,46,3};`
`p(1,&a);`est équivalent à `a=1*a+3*a^4;`

```
f: mul r0,r0,r1  // x^2
   mul r1,r1,r1  // x^4
   mul r1,r1,r1  // x^8
   mul r1,r1,r1  // x^16
   mul r0,r1,r0  // x^17
   ret
```

Barème

1) 4pt

Chaque opération partiellement fausse ou manquante : -0.3pt.

2) 6pt

-0.5pt pour un transistor dont une patte est mal raccordée.

-1pt pour un transistor dont les deux pattes sont mal raccordées.

3) 6pt

`p` en C	1.5pt	-0.5pt par erreur comme argument manquant, test de boucle faux, incrément oublié
`t`	2.5pt	-0.5pt pour chaque erreur comme instruction fausse ou manquante
$ax + bx^4$	2pt	-0.5pt pour chaque erreur comme a faux ou b faux.

4) 4pt

`f` en C	1 pt
`assembleur`	3 pt

-0.5pt pour chaque erreur.