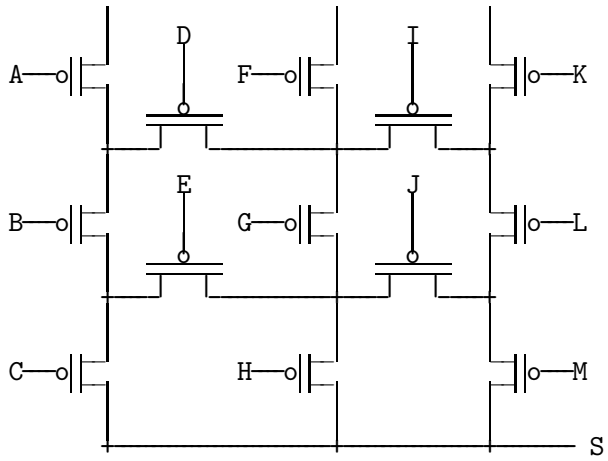
Université de Nanterre lundi 24 avril 2017
architecture des ordinateurs contrôle

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

Complétez le bas du schéma avec autant de transistors.




```
f:  loadimm16 r3,1
    xor  r4,r4,r4
l1: sub  r0,r3,r0
    jc   l2
    load r1,r5
    load r2,r6
    sub  r5,r6,r5
    mul  r5,r5,r5
    add  r4,r5,r4
    add  r1,r3,r1
    add  r2,r3,r2
    jmp  l1
l2: mov  r4,r0
    ret
```

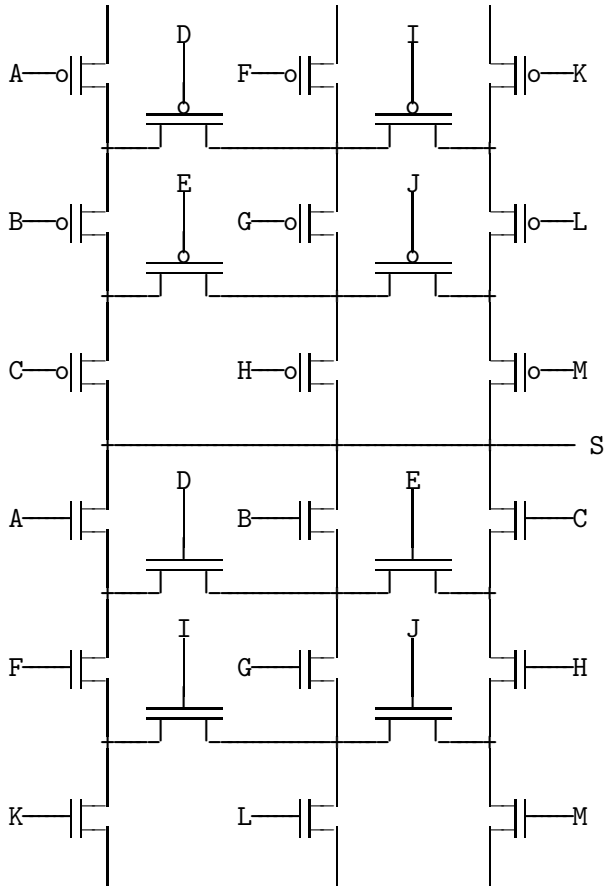
Donnez un équivalent simple en C de la fonction `f`.
Que vaut `x` dans `int t[]={3,6,1,5,3,4,2}, x=f(3,t+1,t+3);`
Donnez une formule donnant la valeur de `f(n,t,u)`.
Redonnez la valeur de `x` et la formule de `f(n,t,u)` si on duplique la ligne `mul r5,r5,r5`.
Même question en remplaçant la ligne `mul r5,r5,r5` par

```
mul r5,r5,r6
mul r6,r5,r5
```

Ecrire en C puis en assembleur la fonction `int g(int n, int *t, int *u);` qui rend $\sum_{i=0}^{n-1}(t[i]+u[i]+1)^8$.

Corrigé

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



```
                    //           r0     r1     r2    r3    r4     r5   r6
f:  loadimm16 r3,1  // int  f(int n, int*t, int*u) // 1     s   x=*t   y=*u
    xor  r4,r4,r4   // { int s=0;
l1: sub  r0,r3,r0
    jc   l2         //   while(n--)
    load r1,r5      //   { int x=*t,
    load r2,r6      //         y=*u;
    sub  r5,r6,r5   //     x-=y; //*t-*u;
    mul  r5,r5,r5   //     x*=x; //(*t-*u)^2
    add  r4,r5,r4   //     s+=x; //s+=(*t-*u)^2
    add  r1,r3,r1   //     t++;               int f(int n, int*t, int*u)
    add  r2,r3,r2   //     u++;               { while(n--)
    jmp  l1         //   }                      { int x=*t++-*u++; s+=x*x; }
l2: mov  r4,r0      //   return s;              return s;
    ret             // }                      }
```

$$x = (6-5)^2 + (1-3)^2 + (5-4)^2 = 1 + 4 + 1 = 6$$

$$f(n,t,u) = \sum_{i=0}^{n-1} (t[i] - u[i])^2$$

$$x = (6-5)^4 + (1-3)^4 + (5-4)^4 = 1 + 16 + 1 = 18$$

$$f(n,t,u) = \sum_{i=0}^{n-1} (t[i] - u[i])^4$$

$$x = (6-5)^3 + (1-3)^3 + (5-4)^3 = 1 - 8 + 1 = -6$$

$$f(n,t,u) = \sum_{i=0}^{n-1} (t[i] - u[i])^3$$

```
int g(int n, int*t, int*u)
{ while(n--)
  { int x=*t++ +*u++ +1;
    x*=x;
    x*=x;
    s+=x*x;
  }
  return s;
}
```

Dans la fonction `f` donnée dans l'énoncé on remplace les deux lignes

```
    sub  r5,r6,r5   //     x-=y; //*t-*u
    mul  r5,r5,r5   //     x*=x; //(*t-*u)^2
```

par

```
    add  r5,r6,r5   //     x+=y; //*t+*u
    add  r5,r3,r5   //     x++;  //*t+*u+1
    mul  r5,r5,r5   //     x*=x; //(*t+*u+1)^2
    mul  r5,r5,r5   //     x*=x; //(*t+*u+1)^4
    mul  r5,r5,r5   //     x*=x; //(*t+*u+1)^8
```

Barème

1) 7pt=14x0.5pt

Chaque opération: 0 ou 0.5pt.

2) 4pt

-0.5pt pour toute erreur : Il manque 1 ou plusieurs !. Chaque lettre (commande) manquante ou en trop ou mal placée.

3) 6pt

`f` en C	1.5 pt-0.5pt par erreur comme argument manquant, test de boucle faux, incrément oublié
6	0.75pt
formule	0.75pt
18	0.75pt
formule	0.75pt
-6	0.75pt
formule	0.75pt

4) 3pt

On ne tient pas compte des commentaires. -0.5pt pour chaque instruction assembleur fausse ou manquante.