```
f: loadimm16 r4,-10
   loaddimm16 r3,1
l1:sub r2,r3,r2
   jc l2
   add r0,r3,r0
   load r0,r5
   sub r1,r3,r1
   load r1,r6
   xor r5,r6,r5
   xor r6,r2,r6
   sub r5,r6,r7
   mov r6,r7
   movcl r5,r6
   movcl r7,r5
   store r0,r5
   store r1,r6
   add r4,r5,r4
   add r4,r6,r4
   jmp l1
l2:mov r4,r0
   ret
```

La fonction `f` précédente écrite en assembleur est appelée dans le programme C suivant:

```
#include<stdio.h>
int f(int *t, int *u, int n);
void afft(int*t,int n)
{ while(n--) printf("%4d",*t++);
  printf("\n");
}
void p(int num)
{ int t[8], i;
  for(i=8;i;num/=10) t[--i]=num%10;
  afft(t,8);
  t[3]=f(t+1,t+6,3);
  t[4]=f(t+3,t+4,4);
  afft(t,8);
}
int main()
{ p(37001234);
  return 0;
}
```

Ce programme écrit:

```
   3   7   0   0   1   2   3   4
   3   4   0  -8  -2   2   6   7
```

Qu'écrit-il si vous remplacez 37001234 par votre numéro d'étudiant?

Avec votre numéro d'étudiant, qu'écrit-il si vous remplacez les deux `movcl` par des `movcg`?

Même question en remplaçant les deux `movcl` par des `movcb`?

Même question en les remplaçant par des `movca`?

Corrigé

```
#include<stdio.h>
//        r0      r1     r2    r3   r4   r5   r6   r7
int f(int *t, int *u, int n)//  1    s    x    y    z
{ int s=-10;                //f: loadimm16 r4,-10; loaddimm16 r3,1
  while(n--)                //l1:sub r2,r3,r2;  jc l2
  { int x=*++t,             //   add r0,r3,r0;  load r0,r5
        y=*--u, z;          //   sub r1,r3,r1;  load r1,r6
    x^=y;                   //   xor r5,r6,r5
    y^=n;                   //   xor r6,r2,r6
    if(x<y) z=y, y=x, x=z;  //   sub r5,r6,r7; mov r6,r7;
                            //   movcl r5,r6; movcl r7,r5
    *t=x;                   //   store r0,r5
    *u=y;                   //   store r1,r6
    s+=x+y;                 //   add r4,r5,r4; add r4,r6,r4
  }                         //   jmp l1
  return s;                 //l2:mov r4,r0;
}                           //   ret
void afft(int*t,int n)
{ while(n--) printf(" %d",*t++);
  printf("\n");
}
void p(int num)
{ int t[8], i;
  for(i=8;i;num/=10) t[--i]=num%10;
  afft(t,8);
  t[3]=f(t+1,t+6,3);
  t[4]=f(t+3,t+4,4);
  afft(t,8);
}
int main()
{ p(37001234);
  return 0;
}
```

Quand on remplace `movcl` par `movcg` ou `movcb` ou `movca` on remplace respectivement `x<y` par `x>y` ou `(unsigned)x<(unsigned)y` ou `(unsigned)x>(unsigned)y`. Cela donne le programme `ex2ar20.c` qui produit `ex2ar20.txt`.

On peut noter que `mov r4,r0; ret` se traduit par `return s;` et non pas `*t=s; return *t;` qui apparaît dans beaucoup de devoir. En effet le résultat explicite de la fonction doit être rendu dans le registre `r0` qui contenait auparavant le premier argument `t` de la fonction. Mais `t` est un pointeur sur un entier `int*` alors que `s` est un entier `int`. On pourait effectivement dire que `r0` est une variable de type `union{int i; int*p;}` ou écrire quelque chose comme `t=(int*)s; return (int)t;` ou bien `t=s; return t;` qui marche aussi en C sans erreur mais avec des warnings. Mais on ne doit surtout pas écrire `*t=s; return *t;` qui ne fait pas de warning et rend bien la bonne valeur mais l'écrit indûment quelque part dans le tableau. Il ne faut pas confondre `t` et `*t`. L'affectation `t=s` se traduit par `mov r4,r0` alors que `*t=s` se traduit par `store r0,r4`. Il ne faut pas "corriger" bêtement `t=s;` en `*t=s;` sous prétexte que cela enlève un warning du compilateur.

Barème

Chaque nombre juste dans une deuxième ligne rapporte 1 point. Mais chaque point au delà de 10 est divisé par deux.