COMPILADORES E INTÉRPRETES
Generación y optimización de código

Práctica 2:

Lee todas las optimizaciones indicadas en el enunciado. Considera la que tienes asignada. Para ella mide los tiempos de ejecución de las versiones con y sin la optimización. Haz varias medidas de cada caso. Haz un estudio para distintos niveles de optimización: -O0. –O1, -O2 y –O3. Haz un estudio para determinar la influencia del tamaño del problema N. Para obtener tiempos de ejecución significativos, los lazos deben repetirse ITER veces (el valor de ITER debe ser suficiente para que el tiempo de ejecución sea de una decena de segundos) incluyendo la línea:
#define ITER 100

Envía un breve informe al profesor para cada optimización que hayas analizado, explicando en qué se basa el beneficio de la optimización e interpretando los resultados.

En los códigos, el tipo de letra "*for*" se corresponde con la versión inicial del código, y el tipo "***for***" al de la mejora. Seguramente sea conveniente escribir ambas versiones en el mismo programa. Inicializa los valores de los datos utilizados. Intenta evitar en las medidas el efecto de las cargas iniciales de los datos en las caches.

1. Considera la optimización de búsqueda de **subexpresiones locales comunes**. Por ejemplo con el siguiente código:

```
int i, j;
float a, b, c, d, k, x, y, z, t;
static float A[N];
for(j=0; j<ITER; j++)
  for(i=0; i<N; i++){
     a = x * y * z * A[i];
     b = x * z * t;
     c = t * A[i];
     d = k * A[i];
     A[i] = z * x * A[i];
     }

for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      register float tmp1, tmp2, tmp3;
      tmp1 = x * z;
      tmp2 = A[i];
      tmp3 = tmp1 * tmp2;
      a = tmp3 * y;
      b = tmp1 * t;
      c = t * tmp2;
      d = k * tmp2;
      A[i] = tmp3;
      }
```

2. Considera la optimización de **desenrolle de lazos internos**. Por ejemplo con el siguiente código:

```
int i, k;
float a, b;
static float x[N], y[N];
for(k=0; k<ITER; k++){
   for(i=0; i<N; i++)
      y[i] = a * x[i] + b;
   }

for(k=0; k<ITER; k++)
   for(i=0; i<N; i+=4){
      y[i]   = a * x[i]   + b;
      y[i+1] = a * x[i+1] + b;
      y[i+2] = a * x[i+2] + b;
      y[i+3] = a * x[i+3] + b;
      }
```

3. Considera la optimización de **resolver condicionales**. Por ejemplo con el siguiente código:

```
int i, k;
static float a, x[N], y[N];
for(k=0; k<ITER; k++)
   for(i=0; i<N; i++)
     if(a*k==0.0) x[i] = 0;
     else y[i] = x[i]*y[i];

for(k=0; k<ITER; k++)    {
   if(a*k==0) {
      for(i=0; i<N; i++)
        x[i] = 0;
      }
   else {
      for(i=0; i<N; i++)
         y[i] = x[i]*y[i];
      }
   }
```

4. Considera la optimización de **paso de bucles dentro de una función**. Por ejemplo con el siguiente código:

```
int i, j;
float x[N], y[N], z[N];

void suma(float x, float y, float *z)
{
*z = x + y;
}

void suma2(float *x, float *y, float *z)
{
register int i, j;
for(j=0; j<ITER; j++)
  for(i=0; i<N; i++)
     z[i] = x[i] + y[i];
}

for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      suma(x[i], y[i], &z[i]);
      }

suma2(x, y, z);
```

5. Considera la optimización de **fusión de lazos**. Por ejemplo con el siguiente código:

```
int i, j;
static float  x[N], y[N], z[N], k[N], t[N], u[N];
for(j=0; j<ITER; j++){
   for(i=0; i<N; i++)
    z[i] = x[i] + y[i];
   for(i=0; i<N; i++)
    k[i] = x[i] - y[i];
   for(i=0; i<N; i++)
    t[i] = x[i] * y[i];
   for(i=0; i<N; i++)
    u[i] = z[i] + k[i] + t[i];
   }

for(j=0; j<ITER; j++)
   for(i=0; i<N; i++) {
      register float tmpx, tmpy;
      tmpx = x[i];
      tmpy = y[i];
      z[i] = tmpx + tmpy;
      k[i] = tmpx - tmpy;
      t[i] = tmpx * tmpy;
      u[i] = z[i] + k[i] + t[i];
      }
```

6. Considera la optimización de **desenrolle de lazos internos y optimización de reducciones**. Por ejemplo con el siguiente código:

```
int i, k;
static float x[N], y[N];
register float a, a1, a2, a3;
for(k=0; k<ITER; k++){
   a=0.0;
   for(i=0; i<N; i++)
      a = a + x[i] * y[i];
   }

for(k=0; k<ITER; k++){
   a = a1 = a2 = a3 = 0.0;
   for(i=0; i<N; i+=4){
      a  = a  + x[i]   * y[i];
      a1 = a1 + x[i+1] * y[i+1];
      a2 = a2 + x[i+2] * y[i+2];
      a3 = a3 + x[i+3] * y[i+3];
      }
   a = a + a1 + a2 + a3;
   }
```

7. Considera la optimización de **reemplazo de multiplicaciones y divisiones enteras por operaciones de desplazamiento**. Por ejemplo con el siguiente código:

```
int i, j, a, b;
for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      a = i * 128;
      b = a / 32;
      }

for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      a = i << 7;
      b = a >> 5;
      }
```

8. Considera la optimización de **peeling de lazos**. Por ejemplo con el siguiente código:

```
int i, k;
static float  x[N], y[N];
for(k=0; k<ITER; k++)
   for(i=0; i<N; i++)
     if(i==0) x[i] = 0;
     else if(i==N-1) x[i] = N-1;
     else x[i] = x[i]+y[i];

for(k=0; k<ITER; k++)    {
   x[0] = 0;
   x[N-1] = N-1;
   for(i=1; i<N-1; i++)
      x[i] = x[i]+y[i];
}
```

9. Considera la optimización de **minimizar la sobrecarga de los condicionales**. Por ejemplo con el siguiente código:

```
int i, j, k;
static float x[2*N][2*N], z[2*N][2*N];
for(k=0; k<ITER; k++)
   for(j=-N; j<N; j++)
      for(i=-N; i<N; i++){
         if(i*i+j*j != 0)
           z[N+j][N+i] = x[N+j][N+i] / (i*i+j*j);
         else
           z[N+j][N+i] = 0.0;
         }

for(k=0; k<ITER; k++)
   for(j=-N; j<N; j++) {
      for(i=-N; i<0; i++)
         z[N+j][N+i] = x[N+j][N+i] / (i*i+j*j);
      z[N+j][N] = 0.0F;
      for(i=1; i<N; i++)
         z[N+j][N+i] = x[N+j][N+i] / (i*i+j*j);
      }
```

10. Considera la optimización de **desenrolle de lazos internos y optimización de reducciones**. Por ejemplo con el siguiente código:

```
int i, k;
static int x[N];
register int a, a0, a1, a2, a3;
for(k=0; k<ITER; k++){
   a=1.0;
   for(i=0; i<N; i++)
      a = a * x[i]; }

for(k=0; k<ITER; k++){
   a = 1.0;
   for(i=0; i<N; i+=4){
      a0 = x[i];
      a1 = x[i+1];
      a2 = x[i+2];
      a3 = x[i+3];
      a = a * (a0*(a1*(a2*a3)));
      } }
```

11. Considera la optimización de **inlining**: sustituir la llamada a una función por el propio cuerpo de la función. Por ejemplo con el siguiente código:

```
int i, j;
static float x[N], y[N], z[N];
void add(float x, float y, float *z)
{
*z = x + y;
}
for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      add(x[i], y[i], &z[i]);
      }

for(j=0; j<ITER; j++)
  for(i=0; i<N; i++)
     z[i] = x[i] * y[i];
```

12. Considera la optimización de **balancear operaciones**. Por ejemplo con el siguiente código:

```
int i, j, k;
static float x[N], y[N];
for(k=0; k<ITER; k++){
   for(i=0; i<N; i++)
      x[i] = sqrt((float)i);
   for(i=0; i<N; i++)
      y[i] = (float)i+2.0;
   }

for(k=0; k<ITER; k++)
   for(i=0; i<N; i++) {
      x[i] = sqrt((float)i);
      y[i] = (float)i+2.0;
      }
```

13. Considera la optimización de **desenrolle de lazos internos y optimización de reducciones**. Por ejemplo con el siguiente código:

```
int i, j, k;
static int x[N];
register int a, a0, a1, a2, a3;
for(k=0; k<ITER; k++){
   a=1.0;
   for(i=0; i<N; i++)
      a *= x[i];
   }

for(k=0; k<ITER; k++){
   a = a1 = a2 = a3 = 1.0;
   for(i=0; i<N; i+=4){
      a  *= x[i];
      a1 *= x[i+1];
      a2 *= x[i+2];
      a3 *= x[i+3];
      }
   a = a * (a1*(a2*a3));
   }
```

14. Considera la optimización de **fusión de lazos**. Por ejemplo con el siguiente código:

```
int i, j;
typedef struct {float x, y, z;} nuevotipo;
static nuevotipo v1[N], v2[N], v3[N];
for(j=0; j<ITER; j++){
   for(i=0; i<N; i++)
    v3[i].x = v1[i].x + v2[i].x;
   for(i=0; i<N; i++)
    v3[i].y = v1[i].y - v2[i].y;
   for(i=0; i<N; i++)
    v3[i].z = v1[i].z * v2[i].z;
   }
```

```
for(j=0; j<ITER; j++){
   for(i=0; i<N; i++) {
    v3[i].x = v1[i].x + v2[i].x;
    v3[i].y = v1[i].y - v2[i].y;
    v3[i].z = v1[i].z * v2[i].z;
    }   }
```

15. Considera la optimización de **reemplazo de divisiones en punto flotante por una sola división y multiplicaciones**. Por ejemplo con el siguiente código:

```
int i, j;
float a, b, c , d, t, u, v, x, y, z, tmp;
for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      x = i+1.0;
      y = j+x;
      a = u / x;
      b = t / x;
      c = z / y;
      d = v /(x*y);
      }
```

```
for(j=0; j<ITER; j++)
   for(i=0; i<N; i++){
      x = i+1.0;
      y = j+x;
      tmp = 1/(x*y);
      a = u * y * tmp;
      b = t * y * tmp;
      c = z * x * tmp;
      d = v * tmp;
      }
```

16. Considera la optimización de **unroll and jam**. Por ejemplo con el siguiente código:

```
register int i, j, k, l;
static float  a[N][N], b[N][N], c[N][N];
for(l=0; l<ITER; l++)
   for(i=0; i<N; i++)
    for(j=0; j<N; j++)
      for(k=0; k<N; k++)
        c[k][i] += a[k][j] * b[j][i];

for(l=0; l<ITER; l++)
   for(i=0; i<N; i+=2)
      for(j=0; j<N; j+=2)
        for(k=0; k<N; k++){
          c[k][i] += a[k][j] * b[j][i] + a[k][j+1] * b[j+1][i] ;
          c[k][i+1] += a[k][j] * b[j][i+1] + a[k][j+1] * b[j+1][i+1] ;
          }
```

17. Considera la optimización de **intercambio de lazos**. Por ejemplo con el siguiente código:

```
int i, j, k;
static float  x[N][N], y[N][N];
for(k=0; k<ITER; k++){
   for(i=0; i<N; i++)
      for(j=0; j<N; j++)
         y[j][i] = x[j][i] + 3.0;
      }

for(k=0; k<ITER; k++){
   for(j=0; j<N; j++)
      for(i=0; i<N; i++)
         y[j][i] = x[j][i] + 3.0;
      }
```