COMPILADORES E INTÉRPRETES
Generación y optimización de código

Práctica 1:

1. Las medidas de tiempo de ejecución se harán utilizando la función *gettimeofday()*.
   a. Lee el manual de dicha función.
   b. Si queremos medir el tiempo en una parte del código haremos lo siguiente:

```
#include <sys/time.h>
...
struct timeval inicio, final;
double tiempo;
...
gettimeofday(&inicio,NULL);
                {...} /*Código a medir*/
gettimeofday(&final,NULL);
tiempo = (final.tv_sec-inicio.tv_sec+(final.tv_usec-inicio.tv_usec)/1.e6);
```

   c. Comprueba la influencia del tiempo empleado por la función *gettimeofday*.
   d. Mide el tiempo de ejecución con dos ejemplos simples: uno con muchas operaciones aritméticas, y otro con muchas operaciones de entrada/salida. Haz varias medidas.

2. Usaremos el compilador gcc.
   a. Lee el manual de dicho compilador.
   b. El ensamblador que genera es el de la arquitectura IA32 de Intel. Su manual está en: http://www.intel.com/Assets/PDF/manual/253665.pdf. Hojéalo. Tienes información interesante en: https://www.cs.umd.edu/users/meesh/webpages/cmsc311/links/handouts/ia32.pdf y en http://web.stanford.edu/class/cs107/IA32_Cheat_Sheet.pdf
   c. Para los siguientes apartados considera el siguiente código que realiza el producto de dos matrices cuadradas. Comprueba que funciona correctamente.

```
#include <stdio.h>
#define Nmax 600

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

main() {
float A[Nmax][Nmax], B[Nmax][Nmax], C[Nmax][Nmax], t, r;
int i,j,k;

for(i=0;i<Nmax;i++)                /* Valores de las matrices */
        for(j=0;j<Nmax;j++) {
                A[i][j]=(i+j)/(j+1.1);
                B[i][j]=(i-j)/(j+2.1); }
for(i=0;i<Nmax;i++)                /* Producto matricial */
        for(j=0;j<Nmax;j++) {
                t=0;
                for (k=0;k<Nmax;k++) {
                        producto(A[i][k],B[k][j],&r);
                        t+=r; }
                C[i][j]=t; }
}
```

d. La opción –E realiza solamente el preprocesado. Comprueba cómo se sustituyen las constantes.
e. La opción –S genera el código en ensamblador. Compruébalo. Analiza la sintáxis de este ensamblador, cómo se implementan los lazos, cómo se hacen las llamadas a funciones y como son las operaciones en punto flotante.
f. La opción –c genera el código objeto. Compruébalo. Un fichero objeto se puede enlazar con gcc invocando el fichero .o directamente.
g. El enlazado estático se puede hacer con la opción –static de gcc. Comprueba el tamaño del ejecutable. Los ficheros compilados de esta manera son autónomos al quedar incorporadas a su código las librerías.
h. Compíla con las opciones -O0, -O1, -O2, -O3, -Os. Compara el tamaño de los códigos objeto de cada compilación. Compara los tiempos de ejecución. Compara los códigos en ensamblador. Envía un breve informe en pdf sobre este apartado solamente al profesor.

3. Considera el código adjunto. Compílalo primero con la opción -O1 y posteriormente con **–O1 –funroll-loops**. Compara los códigos en ensamblador obtenidos. Analiza el comportamiento (en términos de tiempo de ejecución) de ambas versiones para diferentes valores de N.
Envía un breve informe al profesor en formato pdf.

```
#include <stdio.h>
#define N 10000

double res[N];

main() {
int i;
double x;

for(i=0;i<N;i++)
        res[i]=0.0005*i;
for(i=0;i<N;i++) {
        x=res[i];
        if (x<10.0e6) x=x*x+0.0005;
        else x=x-1000;
        res[i]+=x;
}
printf("resultado= %e\n", res[N-1]);
}
```