Plancha 3
Números en punto flotante usando lenguaje ensamblador

Arquitectura del Computador
Licenciatura en Ciencias de la Computación

14 de febrero de 2022

1. Introducción

Esta plancha de ejercicios trabaja la representación de números en punto flotante, en los lenguajes C y ensamblador de x86_64.

2. Procedimiento

Resuelva cada ejercicio en computadora. Cree un subdirectorio dedicado para cada ejercicio, que contenga todos los archivos del mismo. Para todo ejercicio que pida escribir código, genere un programa completo, con su función `main` correspondiente; evite dejar fragmentos sueltos de programas.

Asegúrese de que todos los programas que escriba compilen correctamente con `gcc`. Se recomienda además pasar a este las opciones `-Wall` y `-Wextra` para habilitar advertencias sobre construcciones cuestionables en el código.

Una vez terminada la plancha, suba una entrega al repositorio Subversion de la materia, creando una etiqueta en el subdirectorio correspondiente a su grupo:
`https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-222/Alumnos/2020/`

3. Ejercicios en C

4. Ejercicios en ensamblador

1) Escribir una función en Assembler x86-64 para realizar la conversión `h=f*1.5+c`. Su signatura equivalente en C es la siguiente:

```
double convert(double f, int c);
```

Compilar de la siguiente manera:

```
gcc main.c convert.s -o main
```

utilizando la siguiente función `main`:

```
#include<stdio.h>
double convert(double, int);
```

```
int main(){
double f = 3.5;
int c = 32;
printf("El resultado de la conversión es: %f\n", convert(f,c));
return 0;
}
```

2) Implemente en ensamblador de x86_64 la función:

```
int solve(float a, float b, float c, float d, float e, float f, float *x, float *y);
```

que resuelva el sistema de ecuaciones:

$$\begin{aligned} ax + by &= c \\ dx + ey &= f \end{aligned} \tag{1}$$

y escriba el resultado en los punteros `x` e `y`. La función debe devolver 0 si encontró una única solución y -1 en caso contrario.

3) Implemente en ensamblador la siguiente función:

```
void sum(float *a, float *b, int len);
```

que suma dos arreglos de flotantes de longitud `len` dejando el resultado en `a`. Utilice instruciones escalares (i.e.: `addss`).

4) Reimplemente la función anterior utilizando instrucciones SIMD (formato packed). Tenga en cuenta que la longitud del arreglo no necesariamente es múltiplo de 4 (en cuyo caso puede terminar los cálculos utilizando instrucciones escalares). Llámela `sum_simd`. Utilizando la función `clock_gettime` compare el tiempo computacional de cada implementación para arreglos de distinto tamaño (de 1000 a 100.000.000 elementos).