Medidas de performance II

Clase de Práctico N°5

Procesamiento Digital de Señales

INGENIERÍA
BIOLÓGICA

Contenidos

01

Revisión del Hito 2.1

Verificar implementación del FIR causal online y estimación de la complejidad teórica

02

Tutorial

Valgrind: callgrind y kcachegrind

03

Ejemplos

Conteo de ciclos para arreglo1.c y arreglo2.c

04

Chequeo del FIR

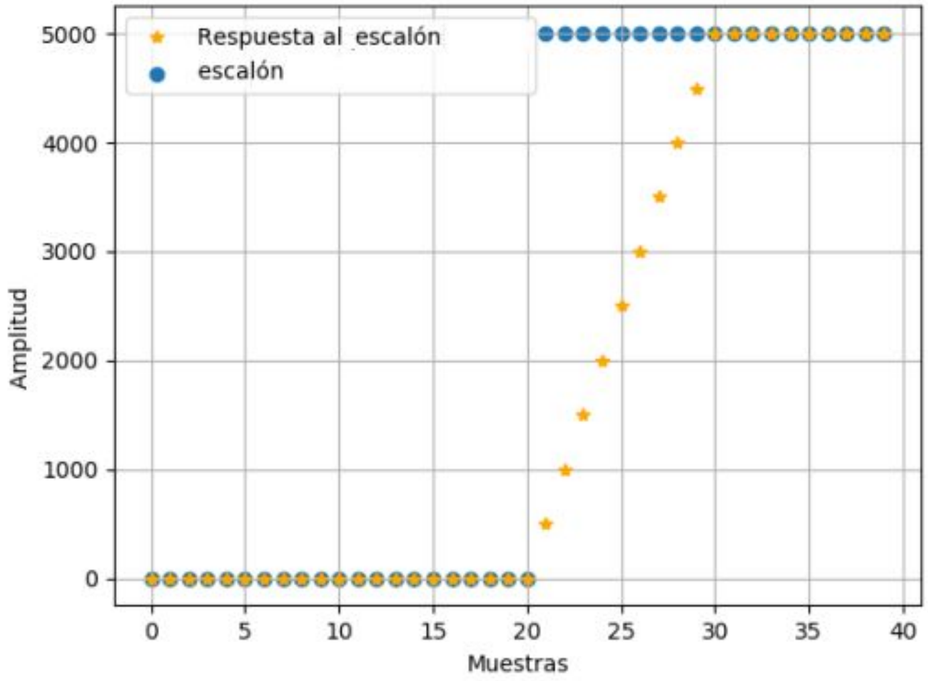
Medir el uso de CPU por parte del FIR implementado

01

Revisión del Hito 2.1

Implementación del FIR causal

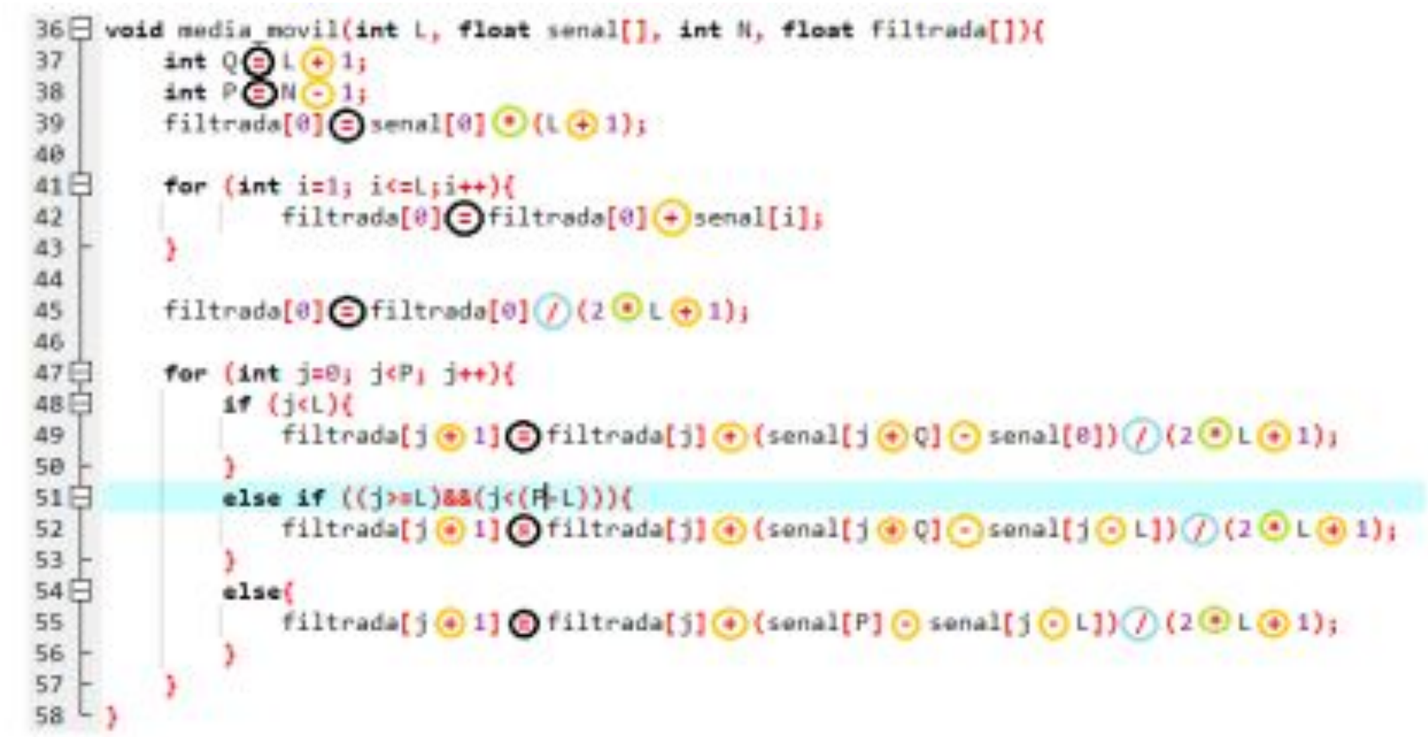
- Ver respuesta al escalón
- Verificar que filtra muestra a muestra



Cálculo de complejidad

- Considerar que una suma, resta, multiplicación, división o asignaciones consumen un ciclo del procesador cada una.
- Las operaciones que se realizan dentro de un bucle se multiplican por la cantidad de veces que se entra al bucle.
- Parte práctica, dos parámetros “N” y “L”:
    - Para determinar la complejidad del filtro en función de N se fija un valor de L.
    - Para determinar la complejidad del filtro en función de L se fija un valor de N.

Cálculo de complejidad teórica



```
36  void media_movil(int L, float senal[], int N, float filtrada[]){
37      int Q = L + 1;
38      int P = N - 1;
39      filtrada[0] = senal[0] * (L + 1);
40
41      for (int i=1; i<=L;i++){
42          filtrada[0] = filtrada[0] + senal[i];
43      }
44
45      filtrada[0] = filtrada[0] / (2 * L + 1);
46
47      for (int j=0; j<P; j++){
48          if (j<L){
49              filtrada[j + 1] = filtrada[j] + (senal[j + Q] - senal[0]) / (2 * L + 1);
50          }
51          else if ((j>=L)&&(j<(P-L))){
52              filtrada[j + 1] = filtrada[j] + (senal[j + Q] - senal[j - L]) / (2 * L + 1);
53          }
54          else{
55              filtrada[j + 1] = filtrada[j] + (senal[P] - senal[j - L]) / (2 * L + 1);
56          }
57      }
58  }
```

- **Negro asignaciones**
- **Naranja sumas y restas**
- **Verde multiplicaciones**
- **Celeste divisiones**

Cálculo de complejidad teórica

```
36  void media_movil(int L, float senal[], int N, float filtrada[]){
37      int Q = L + 1;
38      int P = N - 1;
39      filtrada[0] = senal[0] * (L + 1);
40
41      for (int i=1; i<=L;i++){
42          filtrada[0] = filtrada[0] + senal[i];
43      }
44
45      filtrada[0] = filtrada[0] / (2 * L + 1);
46
47      for (int j=0; j<P; j++){
48          if (j<L){
49              filtrada[j + 1] = filtrada[j] + (senal[j + Q] - senal[0]) / (2 * L + 1);
50          }
51          else if ((j>=L)&&(j<(P-L))){
52              filtrada[j + 1] = filtrada[j] + (senal[j + Q] - senal[j - L]) / (2 * L + 1);
53          }
54          else{
55              filtrada[j + 1] = filtrada[j] + (senal[P] - senal[j - L]) / (2 * L + 1);
56          }
57      }
58  }
```

Número de veces	Sumas/Restas	Productos	Divisiones	Asignaciones	Subtotal
1	L + 4	2	1	L +3	2L +10
L	5	1	1	1	8
N-2L-1	6	1	1	1	9
L	5	1	1	1	8

Cuadro 1: Cálculo de operaciones para el filtro no causal.

$$Clk_{nc}[N] = 1 + 9N$$

02

Tutorial

Valgrind:
callgrind y
kcachegrind

Valgrind

- Callgrind herramienta de Valgrind para medir el uso de CPU.
- Ejecución
  - `valgrind --tool=callgrind ./programa`
- Kchegrind herramienta de visualización para archivos callgrind.out.N°
  - Instalación
    - `sudo apt-get install valgrind kcachegrind graphviz`
  - Ejecución:
    - Abrir los archivos desde la interfaz gráfica de kcachegrind
    - Mostrar ciclos relativos o ciclos totales
    - Agrupar las funciones según su origen

03

Ejemplos

callgrind con arreglo.c y arreglo2.c

Arreglo

- arreglo.c: inserción de valores en un arreglo
  - `gcc -g arreglo.c -o arreglo -lm`
  - `valgrind --tool=callgrind ./arreglo`
  - `Abrir callgrind.out con kcachegrind`
  - `Ver ciclos de cada función`

Arreglo2

- `arreglo2.c: ordenamiento de los elementos de un vector (Insertion Sort, Selection Sort y Quick Sort)`
  - `gcc -g arreglo2.c -o arreglo2`
  - `valgrind --tool=callgrind ./arreglo2`
  - `Abrir callgrind.out con kcachegrind`
  - `Ver ciclos de cada función`

04

Chequeo del FIR

Chequeo de uso de CPU sobre el FIR implementado

Gracias

¿Preguntas?

Renato Sosa Machado

renato.sosast@gmail.com

Lucía Lemes

llemes@cup.edu.uy