

Proyecto Individual o en Parejas

Diseño e Implementación de un ASIP para reverberación de audio

Fecha de asignación: 27 febrero 2024
Grupos: 1-2 personas

Fecha de entrega: 21 marzo 2024
Profesores:
Luis Chavarría Zamora
Jason Leiton Jiménez

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de arquitectura de computadores en el diseño e implementación de una aplicación capaz de realizar efecto rippling en un sonido usando el *reverberación de sonido* empleando un ISA específico.

1. Descripción General: Reverberación

La reverberación es un fenómeno acústico relacionado a la persistencia del sonido en un medio cerrado. En el medio (paredes, muebles, personas, entre otros objetos) se reflecta el sonido emitido por la fuente. Estas reflexiones van decayendo gradualmente mientras son absorbidas por los objetos del medio. El fenómeno de reverberación es representado gráficamente en la Figura 1.

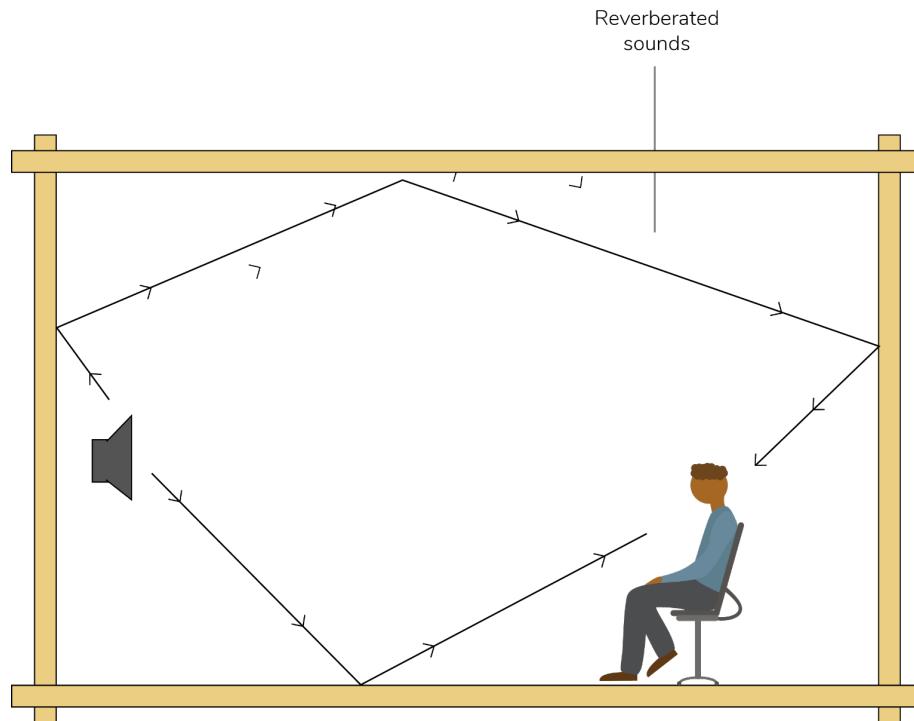


Figura 1: Representación gráfica del fenómeno de reverberación

1.1. Inserción de reverberación a una señal de audio

El efecto de reverberación puede ser modelado con la función de transferencia

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - \alpha}{1 - \alpha z^{-k}}, \quad (1)$$

donde $H(z)$ es la función de transferencia, $Y(z)$ es la salida del sistema y $X(z)$ es la entrada del sistema. α es la atenuación del medio y k es el retardo. Luego de despeje matemático y transformaciones de dominio, se obtiene la ecuación de diferencias, la cual permitiría implementar este fenómeno en un sistema digital

Python →  . $y(n) = (1 - \alpha)x(n) + \alpha y(n - k),$ (2)

donde $y(n)$, es la muestra número n a la salida del sistema, $x(n)$ es la muestra número n a la entrada del sistema y $y(n - k)$ es la muestra n a la salida con un retardo de k muestras.

Como se observa, este sistema tiene un requerimiento de memoria importante pues necesita acceder a salidas anteriores. Si se almacenan en memoria los datos anteriores a la salida se necesitaría una memoria igual al doble del archivo de entrada, o inclusive esta podría ser infinita si el procesamiento fuera en vivo. Una técnica usada para aliviar este requerimiento de memoria es usar un *buffer* circular. Este se ejemplifica en la Tabla 1 con un *buffer* circular para 4 datos.

Tabla 1: *Buffer* circular para 4 entradas de audio

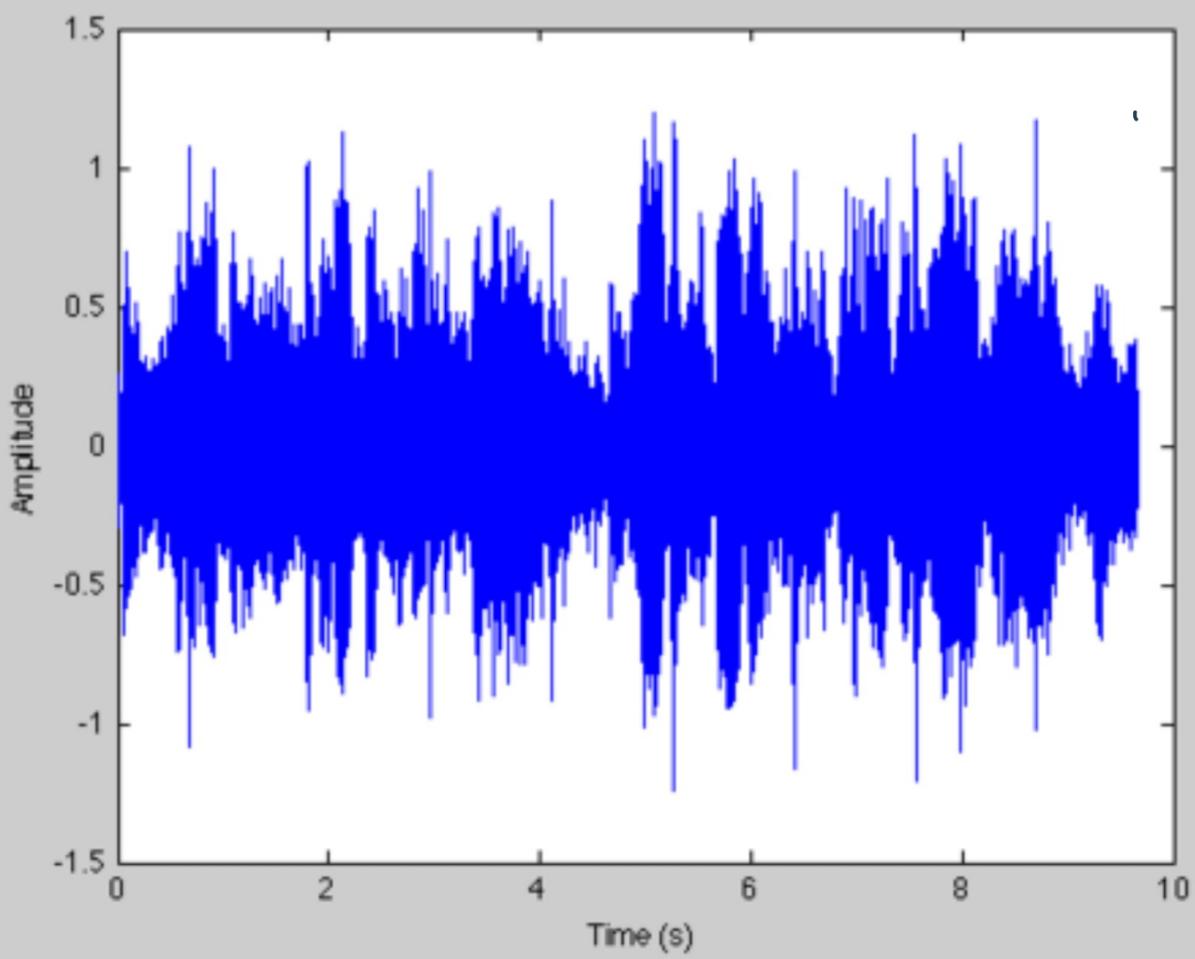
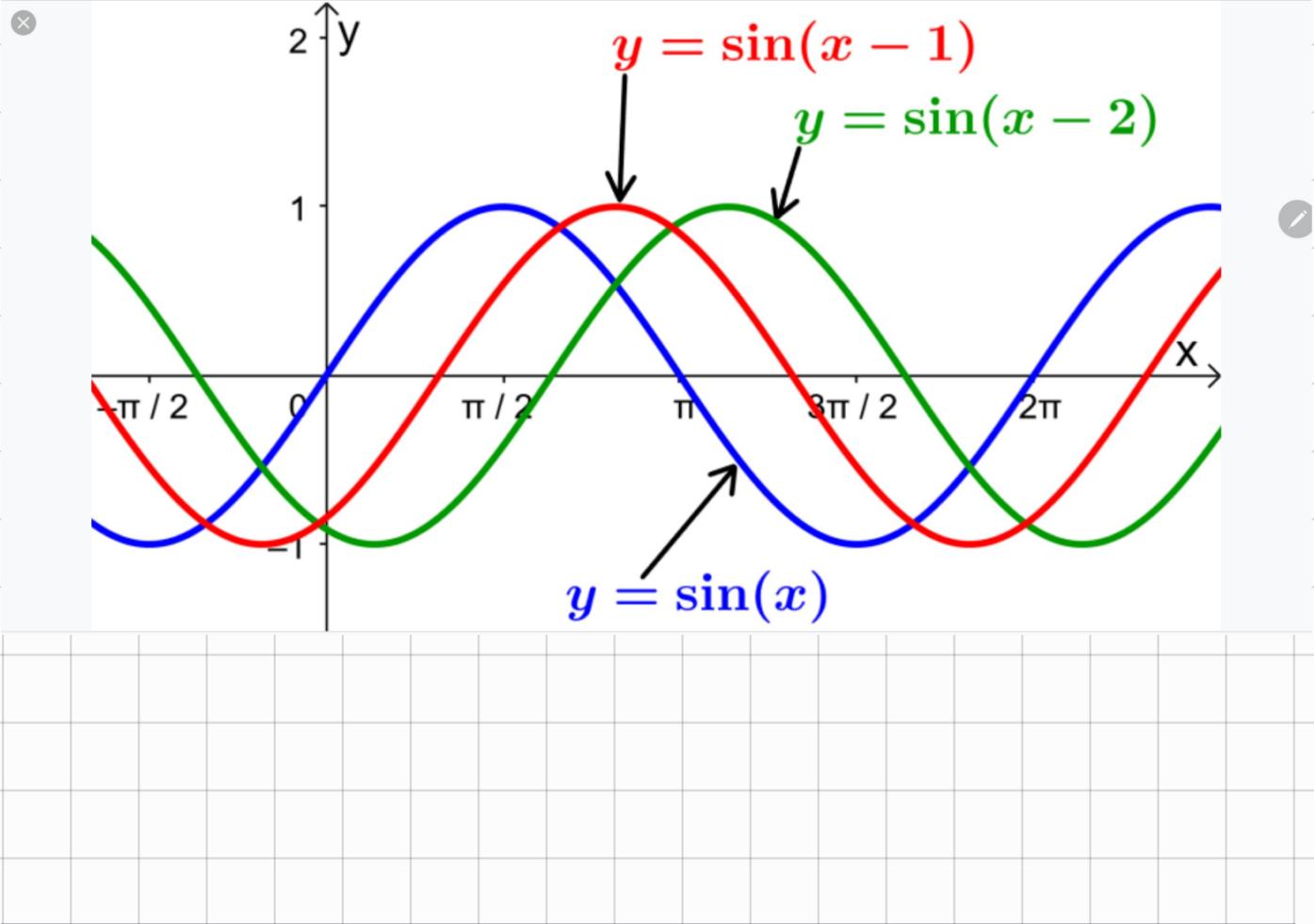
Memoria	Dato	Índice
0x0100		
0x0104	0.304	$y(n-2)$
0x0108	0.350	$y(n-1)$
0x010c	-0.604	$y(n)$
0x0110	0.566	$y(n-3)$
0x0118		

(a) Buffer circular en
un momento n

Memoria	Dato	Índice
0x0100		
0x0104	0.304	$y(n-3)$
0x0108	0.350	$y(n-2)$
0x010c	-0.604	$y(n-1)$
0x0110	-0.348	$y(n)$
0x0118		

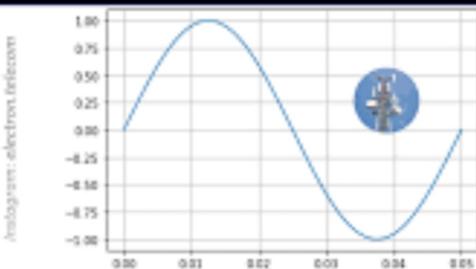
(b) Buffer circular
después de la siguiente
muestra

Este *buffer* es útil para usar eficientemente la memoria porque la ecuación (2) solo necesita a lo sumo la muestra $n - k$ y luego, no se vuelve a utilizar para el procesamiento. **¿Cuál sería el tamaño recomendado para el *buffer* circular?**

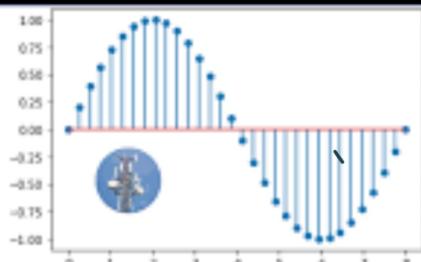




FUNCIÓN CONTINUA:



FUNCIÓN DISCRETA:



T_s : Time sampling
 $\frac{1}{T_s} = F_s \Rightarrow$ Frequency sampling



Ingenieros Electrónicos y de Telecomunicaciones - Seno ...

http://google.com/search?client=firefox-b-m&sca_esv=74866465c804140e&biw=1307&bih=723&tbs=isch&sxsrf=ACQVn0-5xGRzQoyn0VV4bcbgdg-yBa106A%3A

.wav } $\left[\begin{array}{l} x \\ 0 \\ 1 \\ 2 \\ \vdots \\ - \end{array} \right] \quad 0.1 \text{ Hz} \quad \boxed{44100 \text{ Hz}} = F_s$

.mp3 } $y[n] \leftarrow 0 \Rightarrow y[n]=0$

: } \vdots

.mp4 } $2 \text{ MB/s segm ISO}_0 \text{ Linux}$

999 21.0345431

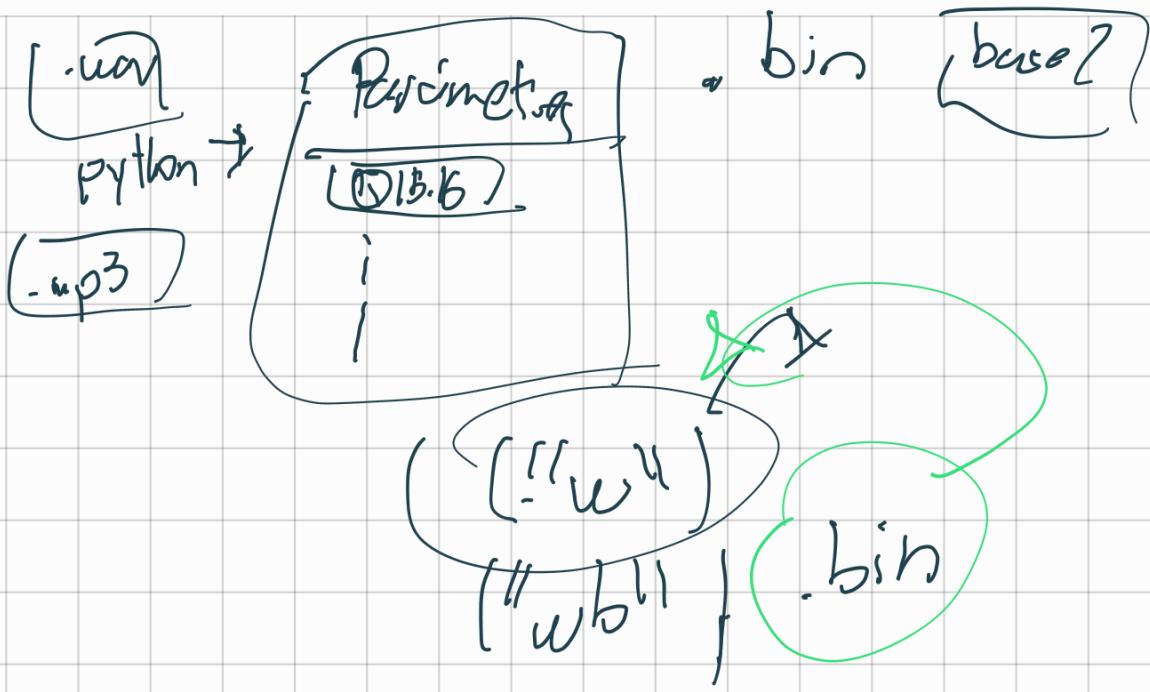
$$n=55 \rightarrow k=50 ; d=0.3$$

$$y[n] = [1-d]x[n] + d y[n-k]$$

$$y[5] = (1) \times [5] + 0.3 \cdot y[-50]$$

$$y[5] = 0.7 \cdot 1 = 0.3 \cdot 0$$

$$\boxed{y[5] = 0.7}$$



$$n=55; k=15; \alpha=0.3$$

$$\gamma[55] = (1-\alpha) \underbrace{x[55]}_{\gamma[55]} + \alpha \gamma[55-15]$$

$$\gamma[55] = 0.7 \cdot 1 + 0.3 \gamma[5]$$

$$\gamma[5] = 0.7 + 0.3 \gamma[2]$$

--

$$\gamma[5] = 0.28$$

$$56 - 50 = 6$$

$$\gamma[50]$$

1.2. Reducción de reverberación en una señal de audio

Para eliminar el eco en una señal se necesita hacer el proceso inverso, este puede ser modelado con la función de transferencia

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - \alpha z^{-k}}{1 - \alpha}. \quad (3)$$

Luego de despeje matemático y transformaciones de dominio, se obtiene la ecuación de diferencias, la cual permitiría implementar este fenómeno en un sistema digital

$$y(n) = \frac{x(n) - \alpha x(n - k)}{1 - \alpha} = \frac{1}{1 - \alpha}[x(n) - \alpha x(n - k)], \quad (4)$$

donde $x(n - k)$ es la muestra k anterior a la actual. Como un aspecto interesante, el factor del denominador es un ajuste de la magnitud para volver el sonido a la magnitud original antes de insertarle el efecto de reverberación.

Algunos valores para α y k se muestran en la Tabla 2, para una frecuencia de muestreo F_s . Se pueden combinar en diferentes combinaciones obteniendo diferentes ejemplos de reverberación.

Tabla 2: Valores recomendados para k y α

Retardo (k)	Atenuación (α)
$F_s \times 50ms$	0.6
$F_s \times 250ms$	0.4
$F_s \times 500ms$	0.2

Se recomienda usar [Octave](#) como software de alto nivel para modelado. El código se muestra a continuación para $k = F_s \times 50ms$ y $\alpha = 0,6$ ¹:

```
% Autor: Luis Chavarría Zamora
% Software: GNU/Octave
% Código de ejemplo para inserción y retiro de reverberación

% Borra lo anterior
clear
clc

% Lectura del archivo
[archivo, fs]=audioread('nombre_archivo');
```

¹Sin embargo, pueden ser diferentes combinaciones

con
10ms

0.5ms
0.25ms
0.125ms

out
¿Cuando nulos
¿scorias q-eras de retraso?

quiere variacion

0.1ms



$$f_s = 44100 \text{ Hz}$$

$$k \Rightarrow 50ms \cdot 44100 \text{ Hz} = 2205$$

$$k \Rightarrow 25ms \cdot 44100 \text{ Hz} = 11105$$

$$k \Rightarrow 12.5ms \cdot 44100 \text{ Hz} = 2295$$

$$\alpha = 0.6$$

$$\alpha = 0.4$$

$$\alpha = 0.2$$

k

J

```
% Parmetros del filtro
alpha=0.6;
k_sinf=0.05;
% Ajuste para el retardo k
k=fs*k_sinf;
% Redondeo hacia abajo, pues k debe ser entero
k=floor(k);

% Se crea una matriz de k filas llena de
% ceros para implementar el retardo hasta k muestras
polos=zeros(1,k);
% Se coloca el y(n)
polos(1)=1;
% Se coloca el retardo a la muestra y(n-k)
polos(k)=alpha;
% Se coloca la atenuación (1 - alpha) a la muestra x(n)
ceros=1-alpha;
% Aplica el filtro
salida_filtro=filter(ceros, polos, archivo); Una mierda
% Reproduce el sonido
player_salida=audioplayer(salida_filtro, fs);
play(player_salida);

% Eliminación de reverberación
input('Presione enter para escuchar reverberación eliminada');
% Se invierten Y y X al ser un proceso inverso
salida_filtro=filter(poles, zeros, salida_filtro);
player_salida=audioplayer(salida_filtro, fs);
play(player_salida);
```

El código mostrado anteriormente, es solo para modelar el resultado y efecto de la reverberación. Se recomienda usarlo como instrumento de comprobación. En el sistema digital solo se deberían implementar las ecuaciones de diferencias (2) y (4). Algunas observaciones importantes sobre los datos:

1. Los valores tanto a la entrada como a la salida de audio en cada momento están entre -1 y 1 (a veces es más reducido el rango). Cualquier valor fuera de este rango puede causar saturación en el sonido.
2. Los valores de α están entre 0 y 1. Por ende, $1 - \alpha$ también está en este rango.

3. El valor de k es entero al ser un entorno digital cuantizado.

Como se observa, al trabajar con valores en el rango entre -1 y 1, se necesita algún tipo de representación para esto, por esa razón se debe utilizar punto flotante o punto fijo.

1.3. Aritmética en punto fijo



Se debe usar esta representación para los valores negativos e inferiores a la unidad. A continuación se muestran ejemplos de suma y multiplicación en punto fijo con formato Q7.8 (conformado por un signo, 7 bits en parte entera y 8 bits en la parte decimal).

1.3.1. Suma

Para suma se muestran las Tablas 3, 4 y 5.

Tabla 3: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
3.25	0	000 0011	0100 0000
4.75	0	000 0100	1100 0000

0180

Tabla 4: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
-3.25	1	111 1100	1100 0000
-1.75	1	111 1110	0100 0000

Tabla 5: Suma en punto fijo Q7.8

Número decimal	Signo	Parte entera	Parte fraccionaria
-1.5	1	111 1110	1000 0000
3.25	0	000 0011	0100 0000
1.75	0	000 0001	1100 0000

¿Cómo tratar los valores negativos?, ¿Cómo detectar los overflow?

1.3.2. Multiplicación

Para la multiplicación en punto fijo para Q7.8 se usa el siguiente formato:

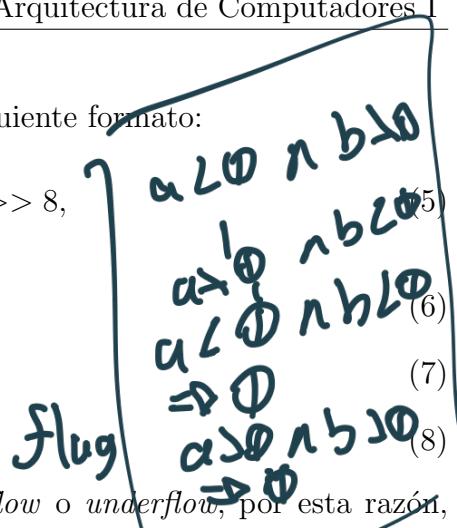
$$result = high << 8 + mid + low >> 8,$$

donde si se multiplica $Qa.b \times Qc.d$

$$high = a \times c,$$

$$mid = a \times d + b \times c,$$

$$low = b \times d.$$



Se nota que por los corrimientos es sencillo caer en *overflow* o *underflow*, por esta razón, se debe evaluar cada caso. La multiplicación sin corrimientos se muestra en la Tabla 6. Con corrimientos en la Tabla 7. En la Tabla 8 se muestra el resumen de la operación.

Tabla 6: Multiplicación en punto fijo Q7.8, de $1,5 \times 3,25$ sin corrimientos

Parte operacion	Entera (MSB)	Decimal (MSB)	Decimal (LSB)
high ($a \times c$)		, 0000 0011	
mid ($a \times d + b \times c$)	000 0001	, 1100 0000	
low ($b \times d$)	010 0000	, 0000 0000	

Tabla 7: Multiplicación en punto fijo Q7.8, de $1,5 \times 3,25$ con corrimientos

Parte operacion	Entera (MSB)	Decimal (MSB)	Decimal (LSB)
high ($a \times c) << 8$	000 0011	, 0000 0000	0000 0000
mid ($a \times d + b \times c$)	000 0001	, 1100 0000	0000 0000
low ($b \times d) >> 8$	000 0000	, 0010 0000	0000 0000
result (4.875)	000 0100	, 1110 0000	0000 0000

Tabla 8: Multiplicación en punto fijo Q7.8, de $1,5 \times 3,25$

Número decimal	Signo	Parte entera	Parte fraccionaria
1.5	0	000 0001	1000 0000
3.25	0	000 0011	0100 0000
4.875	0	000 0100	1110 0000

¿Cómo tratar los valores negativos?, ¿Cómo detectar los overflow?

2. Especificación

Se le solicita desarrollar en **completamente en ensamblador** (asm) un programa que tome un sonido de al menos 10 segundos con y sin eco. Este programa recibe **solamente** un sonido en formato crudo (por ejemplo **txt**). El procesamiento del sonido agregando reverberación y retirando la reverberación, **son dos modos diferentes**. Este proceso **NO** se realizará completamente en ASM, se realizará bajo los siguientes requisitos generales de funcionalidad:

1. El sonido deben ser escuchado en algún software de alto nivel (e.g., Python, Matlab, Octave, etc). El **procesamiento** de las muestras es **completamente en ensamblador (asm)**.
2. El sonido tendrá una frecuencia de muestreo 44,1 kHz.
3. El sonido que usará es en formato 'crudo', es decir como si fuese un txt con los datos en formato decimal de un byte cada uno.
4. El simulador del ISA (solo podrá escoger entre ARM, RISC-V, ningún otro ISA podrá ser usado) y otras herramientas de desarrollo podrán ser elegida abiertamente por cada estudiante. Todo el procesamiento de la imagen debe ser en este simulador seleccionado, recuerde que el lenguaje de alto nivel es solo para audición, **si usa lenguaje de alto nivel en procesamiento obtendrá nota de cero**.
5. El programa automáticamente debe generar los archivos de salida según el modo:
 - Agregar reverberación.
 - Eliminar reverberación.
6. Por facilidad todo este programa debe ser integrado en un solo *framework*. Que genere el sonido en formato 'crudo', procese los datos en ASM y la muestre al usuario.

El proceso de diseño debe incluir la documentación de las propuestas y comparación de viabilidad de las mismas.

3. Metodología de trabajo

El proyecto debe seguir los siguientes aspectos de desarrollo, sino, la parte funcional no será calificada y obtendrá nota de cero. El estudiante deberá seguir los siguientes lineamientos:

1. Los estudiantes pueden usar herramientas de inteligencia artificial u otras fuentes para guía. Debe documentar los procedimientos automatizados y los mismos deben ser validados por

usted contra referencias verificadas y confiables ². En este proyecto usted será un **diseñador(a)**, no un ejecutador(a) o buscador sin criterio técnico crítico. **Esto debe formar parte de los anexos en el documento entregado.**

2. Todo código debe ser documentado de forma interna.
3. Manejo por repositorio:
 - a) Utilice una cuenta de repositorio gratuita.
 - b) El repositorio debe ser enviado al profesor no más de 5 hábiles después de la entrega de este enunciado. Sino, habrá **penalización** de un punto por hora en la nota final del proyecto después de las 11:59 pm del quinto día hábil.
 - c) Cree un repositorio con el siguiente nombre: <user_id>_computer_architecture_1_2024_s1. El user_id estará compuesto por la primera letra del nombre y el apellido de 1 de los integrantes (si fuese en pareja). Por ejemplo, para el estudiante Luis Chavarría, el nombre del repositorio será:
`lchavarria_computer_architecture_1_2024_s1.`
 - d) El repositorio será privado, proporcione acceso a `jatecCE` y `luchazam` (bitbucket) o `luchazam` (GitHub) . Si no es privado desde el inicio recibirá penalización.
 - e) El repositorio de Git contendrá dos ramas principales: `master` y `development`.
 - f) Inicialmente, la rama de `development` se crea a partir del `master`.
 - g) Al trabajar en un proyecto, el estudiante debe crear una nueva rama de trabajo desde `develop` y cuando la función esté lista, la rama debe fusionarse para `develop`. Cualquier corrección o modificación adicional después de `merge` debería requerir que se repita el proceso (es decir, crear la rama desde `develop` y fusionar los cambios más tarde). Una vez que el código de desarrollo esté listo, se fusionará con `master` y se debe crear una `tag`. El proceso se describe en la Figura 2.
Adicionalmente se le recomienda leer este [enlace](#).
 - h) Solo debe contemplar código, recuerde que es un repositorio de versiones de código, no un [Drive](#).
 - i) Después de haber realizado algunos proyectos la rama `master` debe verse así:
 - `master/`
 - `proyecto_1`
 - `proyecto_2`
 - ...

²Algunas veces estas herramientas inducen errores y no deben ser tomadas sin cuestionamiento o verificación (como todo en ingeniería).

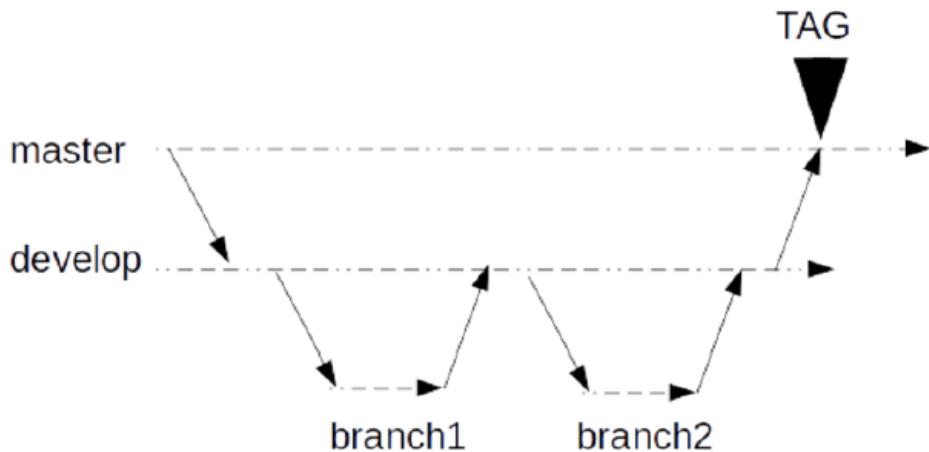


Figura 2: *Git workflow*

■ ...

Donde cada directorio de `proyecto_x` contiene todos los entregables para cada proyecto.

No es permitido realizar todo el trabajo en un solo commit, es decir, que realice el trabajo de forma local y solo suba el último entregable en el repositorio. Si no, será penalizado en todos los rubros de evaluación de la defensa disminuyendo un escalafón en la calificación de cada rubro. Debe mostrar avance incremental (se revisarán estadísticas).

4. Evaluación y entregables

La evaluación del proyecto se da bajos los siguientes rubros:

- Presentación proyecto 100 % funcional (75 %): Una defensa de 20 minutos donde el profesor evaluará el rendimiento del algoritmo. El/la estudiante llevará a la defensa una imagen libre que cumpla los requerimientos. **Debe notificar al profesor del repositorio a más tardar el lunes después de realizar la entrega del enunciado a las 11:59 pm.**
- Documentación de diseño (25 %): Este documento se encuentra directamente ligado con el atributo AP. La documentación del diseño deberá contener las siguientes secciones:
 1. Listado de requerimientos del sistema: Cada estudiante deberá determinar los requerimientos de ingeniería del problema planteado, considerando partes involucradas,

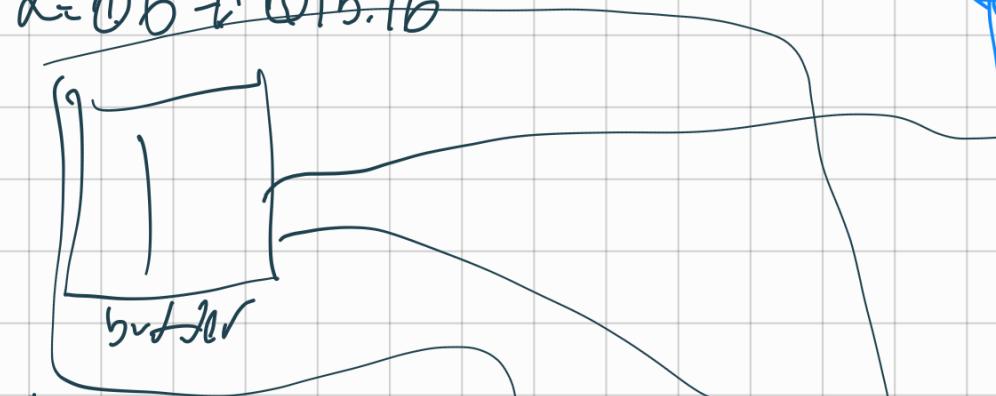
estado del arte, estándares, normas, entre otros. **No colocar características del computador, sino del algoritmo y los relacionados. No es válido mencionar la razón de curva de aprendizaje, debe ser técnica la razón.**

2. Elaboración de opciones de solución al problema: Para el problema planteado deberá documentarse la solución de forma completa y robusta. Debe documentar las opciones de solución. **Las características listadas se deben centrar en el ISA.**
 3. Validación de opciones de solución: Se deberá evaluar las opciones de solución si son óptimas. **La comparación debe ser objetiva basado en las características técnicas y teóricas del ISA.**
- Los documentos serán sometidos a control de plagios para eliminar cualquier intento de plagio con trabajos de semestres anteriores, actual o copias textuales, tendrán nota de cero los datos detectados. Se prohíbe el uso de referencias hacia sitios no confiables.
 - No coloque código fuente en los documentos, quita espacio y aporta poco. Mejor explique el código, páselo a pseudocódigo o use un diagrama.

① Sie & Rückwärts

L Sie & Rückwärts in reverse order

$k=5 \Rightarrow 0x0037$
 $\alpha=05 \Rightarrow 015.16$



$ldr = buffer \Delta$
 $ldr r1, [r0]$
mov r1, l1

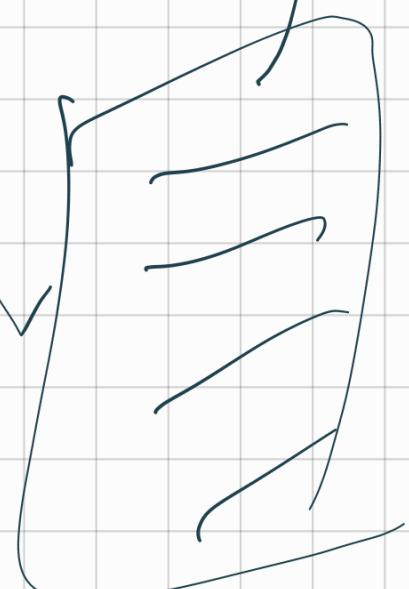
main.s

Arbeits
Q15.16-LE
ENFILE

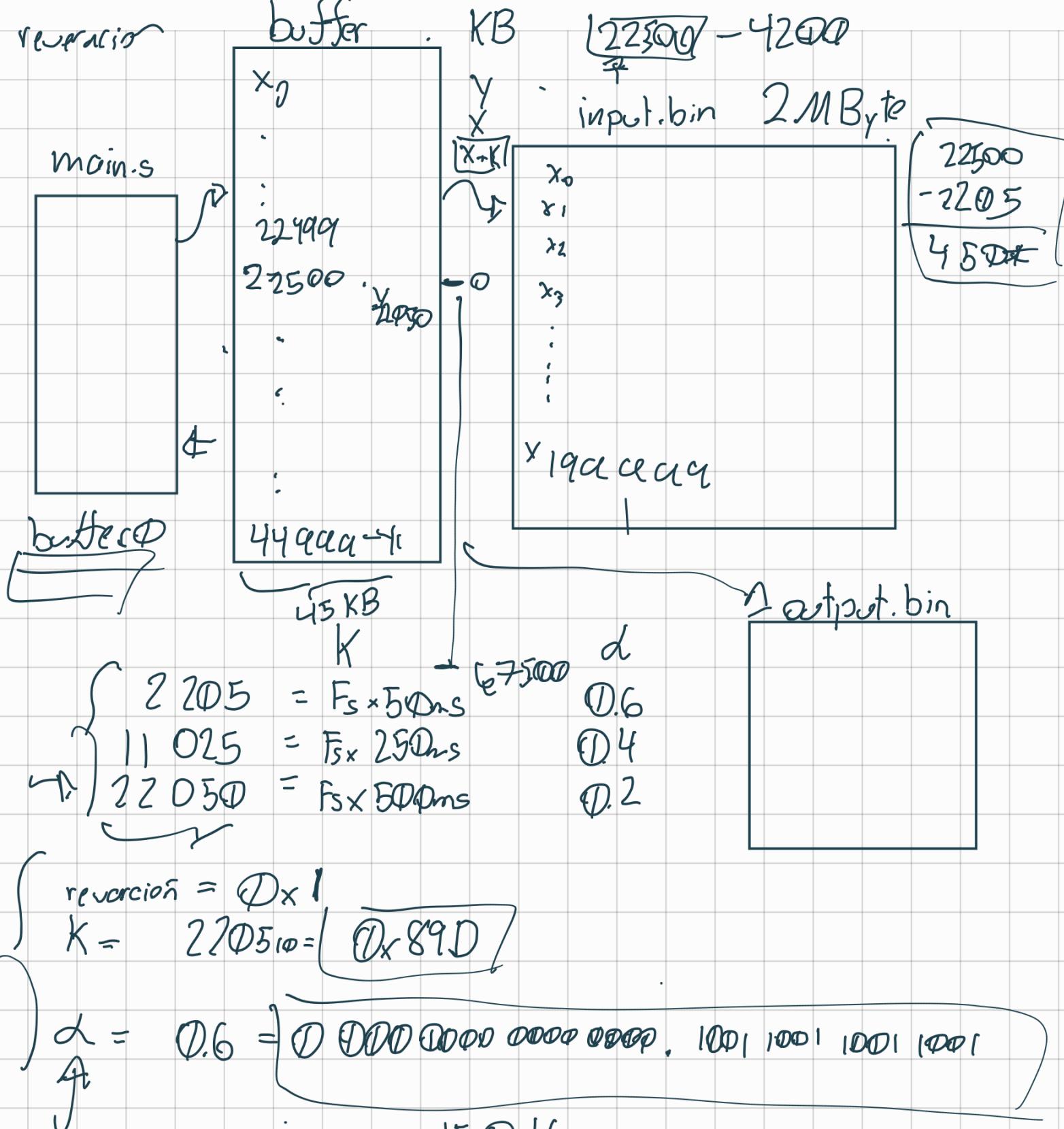
input.bin

2 MB Python

Link
FSO



Output.bin

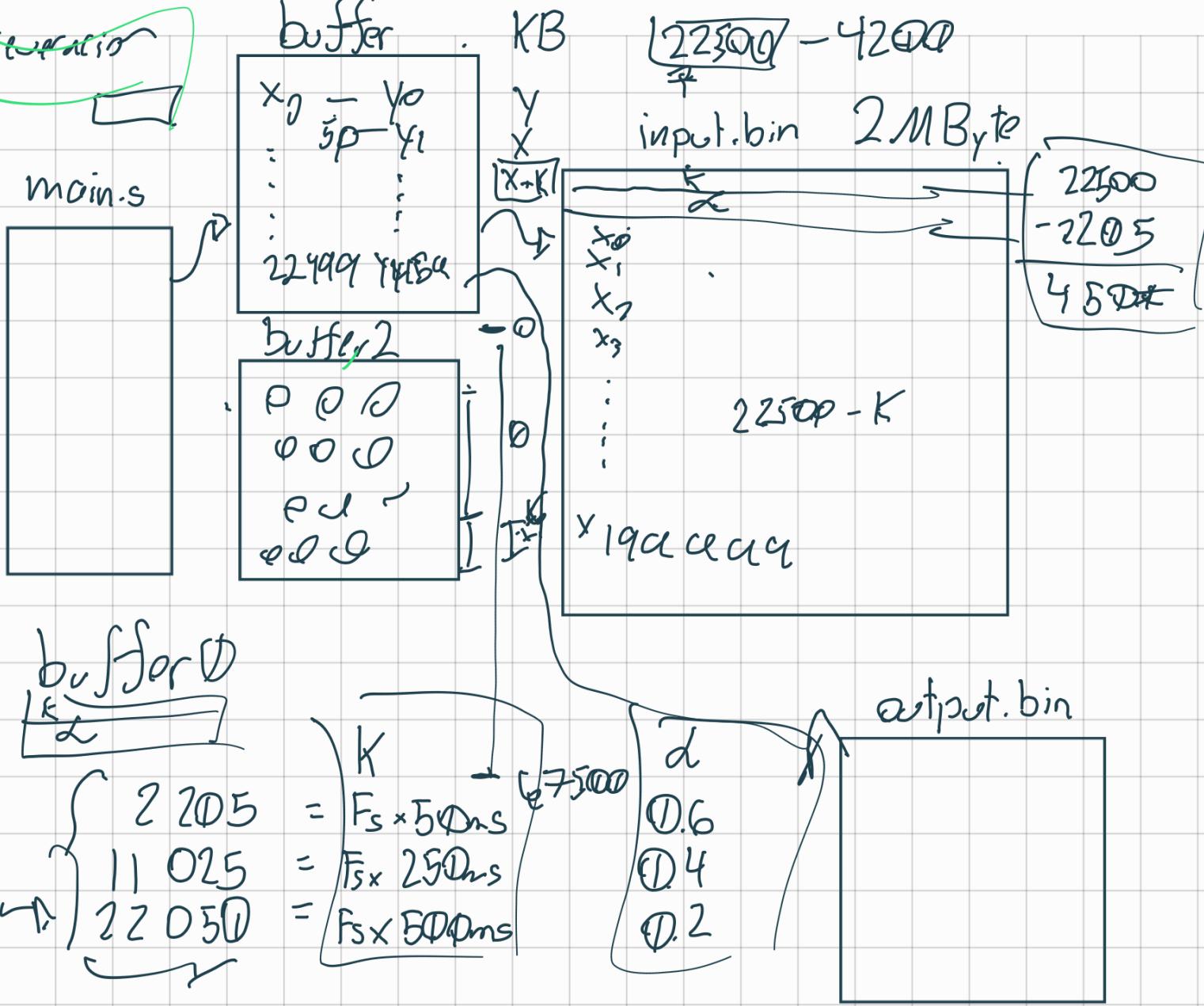


$$k = 2\pi/a$$

$$f = \phi_0 \circ g$$

15@16

DxD004



cancion.wav → input : bin → canción-revercién.wav

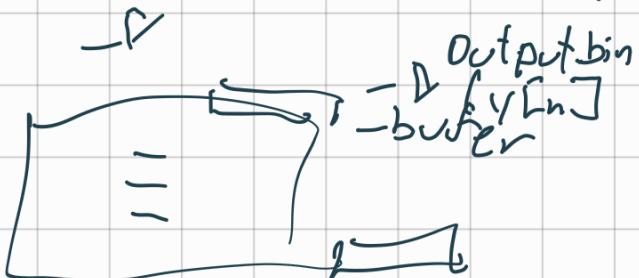


python [F] genera output.bin
de reverción

: output.bin

input.bin(x)

[L] output.wav ✓



buffer = 1024 bytes

load buffer from input
buffer size 12

ldr mode

ldr K

ldr alpha ha

str mode, K, alpha
buffersize 12

str n-K , = circular

while d

load buffer

r6 es bytecounter_reed

call apply reversion

Guarda buffer en output
hasta r6

n-K = donde quedó circular

}

close input

close output

finish program

call apply reveration

for ($n = \text{buffer} [\text{buffer} + r6] ; n + 4$) {

 if

$n - k - k \geq (\text{address_circular})$

 explicacion
 de esto

$y_{\text{old}} = \text{circular}(n - k - k)$

 else

$y_{\text{old}} = \text{circular}[\text{size_circular} + (n - k - k)]$

$$y[n] := d \cdot y_{nk} + (1-d) \times [n]$$

$$\text{circular}_{n-k+4}^n = y[n]$$

reviso si no supera el tamaño del circular

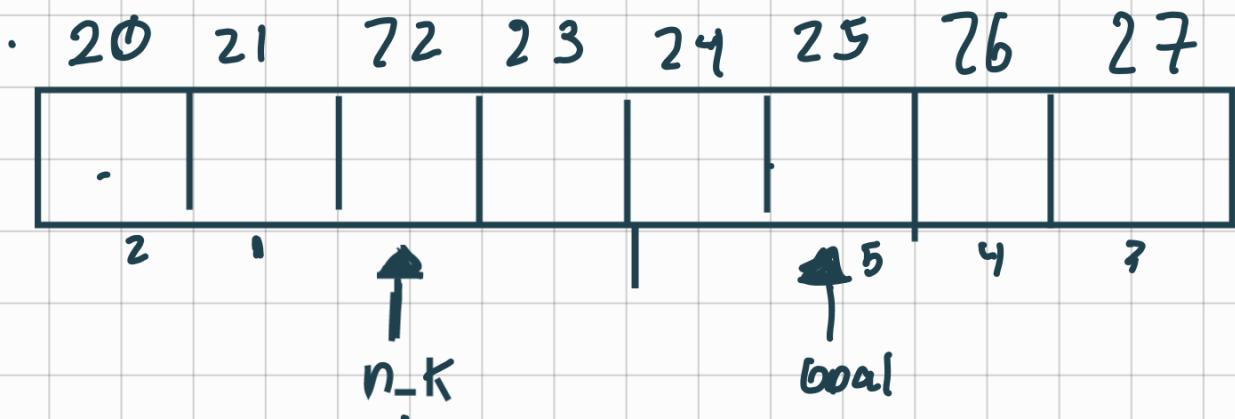
if ($n - k \geq (\text{size_circular} + \text{circular_init})$)

 dr $n - k$, $= \text{circular}$ {

endif

 buffer [n] = y[n]

Circular Buffer



circular inicia en 20 (initial = 20)

$K=5$ retrocedo 5 valores

Size = 8 tamaño del circular

$n-k=22$ posición actual del circular

$$\underbrace{n-k-k}_{\rightarrow} = 22-5 = 17$$

no puedo usarlo yo quiero
25 realmente
entonces

$$\text{initial} - (n-k-k) = 20 - 17 = 3$$

$$\text{Entonces } \underline{\underline{\text{size} + \text{initial} - \text{initial} + (n-k-k)}} = 28 - 3 = 25$$

$$\underline{\underline{\frac{28}{28}}} \text{size} + (n-k-k) = 25$$

$$\text{size} + (n-k-k) = 25$$

$n = (=buffer)$.

for ($n < (=buffer + rb)$

if ($n - k - k \geq (=circular)$)

$y_{old} = circular[n - k - k]$

else

$y_{old} = circular[n - k - k + size]$

$x_n = buffer[n]$

$temp = 1 - \alpha$

$temp2 = temp * x_n$

$temp3 = \alpha * y_{old}$

$y = temp2 + temp3$

$circular[n - k] = y;$

$n - k + 1; i$

} if ($n - k \geq size + (=circular)$)
 $n - k (=circular)$

$buffer[n] = y$

b