UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Informe de laboratorio final

Desarrollo de procesador virtual

Integrantes
Laura Montenegro
20172020005
Nicolás Baena
201
Juan Felipe Rodríguez
20181020158

Facultad de ingeniería

Bogotá D.C., abril de 2021

Abstract

This laboratory seeks to provide the solution to the virtualization of a processor, which through the use of programming languages and the abstraction of the architecture of the central processing units, in order to create a simulator capable of providing a digital alternative to the representation of the operation of the most basic computers, this based on the implementation of architectures such as SAP.

Keywords: Circuit, Simulation, Integrated, Logic Gates, Development, Computer Architecture.

Resumen

El presente laboratorio se busca dar la solución a la virtualización de un procesador, el cual por medio del empleo de los lenguajes de programación y la abstracción de la arquitectura de las unidades de procesamiento central, con el fin de dar creación a simulador capaz de proveer una alternativa digital a la representación del funcionamiento de los computadores más básicos, esto con base en la implementación de arquitecturas como SAP.

Palabras clave: Circuito, Simulación, Integrados, Compuertas Lógicas, Desarrollo, Arquitectura de computadores.

Introducción

El presente laboratorio se realizará la construcción, por medio de software, de un procesador con arquitectura SAP (Simple As Possible), con el objetivo de reconocer e identificar los componentes y funcionamiento de los procesadores. En el desarrollo de la presente práctica, se implementará el uso del lenguaje de programación Python, junto con PyQt5 para la creación de la interfaz gráfica.

El actual documento se seccionará en cuatro etapas o secciones fundamentales, la primera en la que anunciará el marco teórico de la práctica, revisando los conceptos previos que debemos conocer para la creación del procesador virtual. También dentro de esta etapa se relaciona lo construido en el laboratorio inmediatamente anterior en cuanto a la definición de los códigos de operación que inciden en los procesadores.

En la segunda sección llevaremos a cabo, una explicación del desarrollo del procesador, enunciando las herramientas utilizadas y cómo se utilizaron para llegar a la abstracción del procesador.

En la tercera etapa, se mostrarán los resultados obtenidos de la práctica de laboratorio, enunciando cuáles fueron los logros obtenidos, una discusión de los principales resultados y por último una enumeración de las problemáticas que se dieron en el proceso de desarrollo del programa.

Por último, se llevarán a cabo las conclusiones obtenidas en el laboratorio, enunciando los principales resultados y su interpretación en el marco de la asignatura de arquitectura de computadoras y laboratorio.

Objetivo principal

Desarrollar un aplicativo basado en software, para la simulación de un procesador capaz de ejecutar programas basados en los OpCodes establecidos en el laboratorio anterior.

Objetivos específicos

- Recopilación de información sobre las diferentes arquitecturas que se pueden emular.
- Realizar la abstracción de una CPU con el fin de realizar la implementación en Python.
- Desarrollar el aplicativo en Python, en conjunto con el diseño de una interfaz gráfica.

Marco Teórico

Introducción al marco teórico

El siguiente documento elabora un software de simulación acerca del funcionamiento de una máquina de cómputo, usando las especificaciones diseñadas en los últimos laboratorios planteados anteriormente.

Procesador

El procesador es el cerebro del sistema, justamente procesa todo lo que ocurre en la PC y ejecuta todas las acciones que existen. Cuanto más rápido sea el procesador que tiene una computadora, más rápidamente se ejecutarán las órdenes que se le den a la máquina. Este componente es parte del hardware de muchos dispositivos, no solo de tu computadora.

El procesador es una pastilla de silicio que va colocada en el socket sobre la placa madre dentro del gabinete de la computadora de escritorio, la diferencia en una portátil es que está directamente soldado. El procesador está cubierto de algo que llamamos encapsulado, y de lo cual existen 3 tipos: PGA, LGA y BGA.

El procesador es uno de los componentes de la computadora que más ha evolucionado, dado que se les exige a los ingenieros que cada vez ofrecen mejores procesadores para que las computadoras funcionen más rápido y de forma más eficaz. Su evolución no ha sido únicamente interna, sino que también su forma externa fue modificada. Los fabricantes de procesadores de PC más populares son Intel y AMD.

Este componente es el más importante podríamos decir, y generalmente el más caro, pero sin el resto de los componentes no podría servir ni actuar.

Arquitectura SAP

El SAP-1 es un computador con una arquitectura basada en un bus. Contiene los siguientes elementos:

- Bus multiplexado de datos/direcciones de ancho 8 bits.
- Registros activos por flanco, de 4 u 8 bits. Los registros que pueden escribir en el bus contienen un buffer triestado interno. Los que no escriben en él no tienen dicho buffer. Los registros son:
    - Contador de programa (PC, *program counter*).
    - Registro de entrada y de dirección de memoria (*input and MAR*, *input and memory address register*).
    - Registro de instrucción (IR, *instruction register*).
    - Acumulador (A, *accumulator*).
    - Registro B (*B register*).
    - Registro de salida (OUT, *output register*).
- Memoria de acceso aleatorio (RAM, *random-access memory*) de lectura-escritura de tamaño 16×8 (16 posiciones de 8 bits cada una). Está unificada para albergar a la vez datos y programas.
- Unidad aritmético-lógica (ALU, *arithmetic-logic unit*) con dos operaciones disponibles (sumar y restar) para datos en complemento a 2 de 8 bits (desde -128 hasta +127, ambos inclusive).
- Display de ledes para visualizar datos almacenados en el registro OUT. En este simulador, mostramos directamente el valor numérico que corresponde a los ledes en base decimal.
- Unidad de control (CU, *control unit*), de tipo controlador-secuenciador.

Características principales del SAP-1:

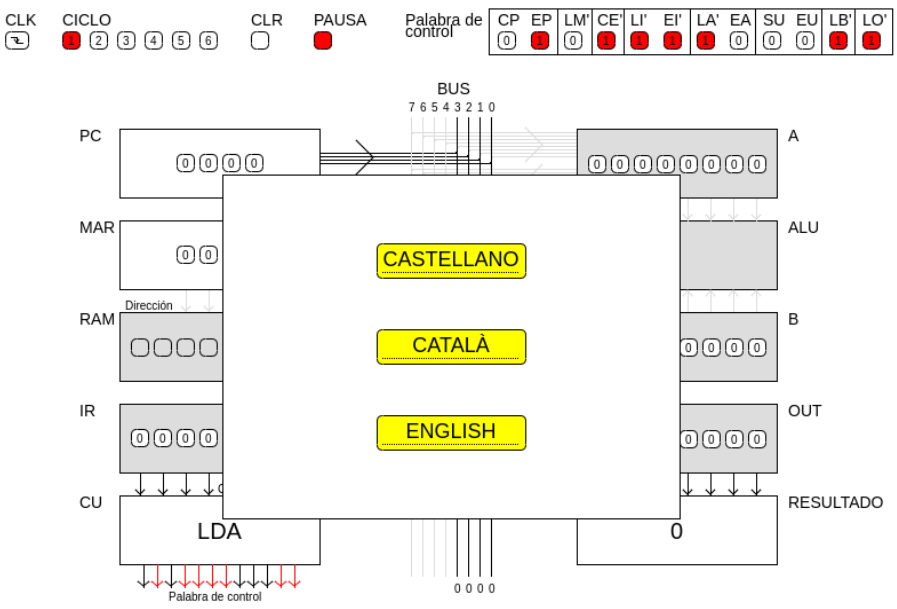

Imagen 1, Programa simulador realizado en processing.

Laboratorio anterior resumido

Realizamos el manejo de entradas al procesador, segmentando las mismas por medio del formato de instrucciones derivado de lo visto en clase, utilizando nuestro propio formato, teniendo en cuenta el tratamiento de opcodes que deseamos manejar. Principalmente, derivamos tres tipos de opcodes que recibirá y entenderá el procesador.

Diseño de formatos de instrucción

Para el diseño de los formatos de instrucción se hicieron, como se dijo anteriormente, tres formatos principales: de registro, de instrucción y de salto. A continuación enunciamos los diferentes formatos de instrucción usados por el procesador.

Opcode	Registro destino (RD)		Registro Fuente A (SA)	

Tabla 1. Registro

Opcode	Registro destino (RD)		Registro Fuente A (SA)	Operando

Tabla 2. Inmediato

Opcode	Dirección izquierda (AL)		Registro Fuente A (SA)	Dirección derecha (AR)

Tabla 2. Salto y bifurcación

Como se puede observar en la primera tabla, nuestro formato, como se especifica en las reglas del laboratorio, está determinado por 8 bits, lo cual nos da un conjunto posible de instrucciones de 256, de las cuales utilizamos o definimos 10 instrucciones básicas, las cuales nos permitirán controlar el funcionamiento de nuestro procesador. Sin embargo, antes de realizar la definición de las mismas procederemos a explicar cómo se realiza el guardado de los datos en nuestro procesador, para esto empleamos la arquitectura ***Big Endian*.**

Dentro de la maquetación de nuestras instrucciones tuvimos en cuenta esto, ya que al almacenar en los registros del procesador se guardaran de la forma:

Almacenamiento de datos con Big Endian

Puesto que se solicitó ser usado el ***Big Endian***, este formato nos permite leer en el orden natural como el lenguaje escrito de izquierda a derecha en el valor hexadecimal Ej:

0x0A0B0C0D = {0A, 0B, 0C, 0D}

El contador de programa necesita 24 bits para poder direccionar todas las posiciones de memoria. El registro de instrucción necesita 32 bits para contener las instrucciones de 32 bits, especialmente 8 de código de operación y 24 del resto. Así dividimos los bits más significativos de los menos significativos-

8 Código de operación	24 bits contienen el resto de información

Tabla 3. División OPCose

Con esta configuración nuestro diseño puede albergar hasta 256 opcodes de los cuales usamos 20 para el manejo de nuestro procesador, los cuales se evidencian en nuestro set de instrucciones. Igual pasa con el operando, que se podría acceder a las 32 posiciones de la memoria.

Cabe recalcar que en este formato el byte más significativo se almacena en primer lugar. Los demás bytes le siguen en orden de significado descendente.

Ejemplos de almacenamiento en nuestro procesador:

32 bits de ancho número 0x12345678 en modo Big-Endian. El método de almacenamiento en la memoria de la CPU

Así es como se verá redistribuido el direccionamiento inmediato.

Inmediato:

-Enteros 8, 16 y 32 bits

-Reales: 32 bits (Simple precisión)

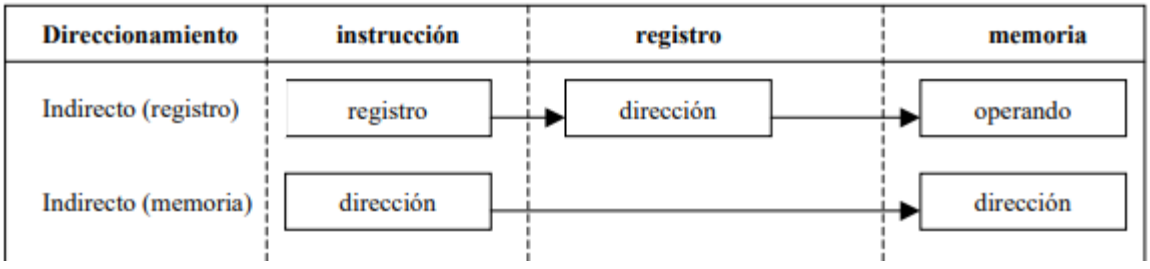

Imagen 2, direccionamiento.

0	CLR	14
1	NOP	15
2	HLT	

Tabla 4. Salto y bifurcación

Limpia memoria del procesador

No realiza acciones

Contenido en Ram:

0000	0001	1110
0001	0010	1101
0010	0011	1100

Tabla 5. Contenido de RAM

Definición de las instrucciones

Especificación de las instrucciones del procesador Unidad de control				
N°	**Instrucción**	**OPCODE**	**Mnemónico**	**Descripción**
0	No operación	0x0	NOP	No realiza acciones Sin operación (No operation)

1	Mueve	0x1	MOV	Mueve el registro
2	Incrementa B	0x2	INCB	Incrementa al registro B el valor 1
3	Incrementa A	0x3	INCA	Incrementa al registro A el valor 1
4	Suma	0x4	SUM	Suma al registro B el valor que se encuentra en la posición de memoria asignada
5	suma + incremento	0x5	SIM	Suma e incrementa la registro asignado el valor que se encuentra en la posición de memoria
6	Substracción	0x6	SUB	Resta al registro asignado el valor que se encuentra en la posición de memoria
7	NO	0x7	NOT	Operación lógica NOT del valor ingresado por el usuario
8	PUSH	0x8	PUSH	Agrega un nuevo registro(byte) en memoria
9	POP	0x9	POP	Elimina el último registro(byte) de la memoria y lo devuelve
10	JUMP	0xa	JMP	Salta a la posición de memoria asignada (Directo)
11	JUMP ZERO	0xb	JZ	Salta a la posición de memoria asignada si se cumple la condición de que el valor es cero
12	JUMP	0xc	JC	Salta a la posición de memoria

	CARRIE			asignada si se cumple la condición de carrie
13	JUMP OVER FLOW	0xd	JOF	Salta a la posición de memoria asignada si se cumple la condición de que el registro sobrepase el máximo permitido
14	JUMP N	0xe	JN	Salta a la posición de memoria asignada si se cumple la condición de que el valor es negativa
15	BREAK	0xf	BRK	Sale del ciclo
16	HALT	0x10	HLT	Apaga el procesador
17	AND	0x11	AND	Operación Lógica AND
18	OR	0x12	OR	Operación Lógica OR
19	XOR	0x13	XOR	Operación Lógica XOR
20	CLEAR	0x14	CLR	Limpia la memoria del procesador

Tabla 6. OP Codes

Marco conceptual

Registro:

Los registros comparten el reloj (CLK), carga (LOAD y las señales (CLR)
Las entradas de registro están vinculadas al bus de datos (DB).
Los registros almacenan cualquier valor que esté en la base de datos en el siguiente flanco ascendente de la CLK, la línea de **salida** baja activa con el OUT (registro de salida) que enviará los datos del registro al bus de datos.

Registro de instrucciones:

Los decodificadores de instrucciones es generar la palabra de control que consta de las líneas de control de los otros módulos, controlando así el comportamiento del procesador y todos sus submódulos.
La palabra de control se crea con base en la instrucción actual, la microinstrucción en esta instrucción y el estado de las banderas de los procesadores.

Reloj:

El módulo de reloj es responsable de toda la coordinación relacionada con el tiempo entre los módulos separados, Como SAP-1 se activa por flanco positivo, esta selección hace que todas las operaciones sincronizadas ocurran a la mitad de cada estado T.

Unidad Aritmético Lógica:

La ALU permite que el Procesador realice operaciones aritméticas simples. De hecho, solo puede proceder la suma o resta de los valores en los dos registros de datos RegistroA y RegristroB.

Unidad de procesamiento central CPU:

El componente principal del ordenador; además de procesar la información, proporciona los puertos necesarios para conectar los distintos periféricos. Se encarga de leer y ejecutar los programas almacenados en la memoria RAM.

Contador del programa:
El contador de programa contiene la dirección RAM de la próxima instrucción a ejecutar. El contador se inicializa en 0, por lo que la primera instrucción que se ejecuta después de un reinicio siempre está en el byte 0 de la RAM.

Memoria de acceso aleatorio:
El módulo RAM permite almacenar hasta **16 bytes** de datos arbitrarios y acceder a ellos direccionando el número de byte en cualquier orden (aleatorio) en lugar de uno tras otro (secuencial). El SAP-1 implementa una arquitectura de von Neumann, lo que significa que solo hay una memoria única para instrucciones y datos, por lo tanto, tanto el programa como todos los datos necesarios deben almacenarse en los 16 bytes de RAM.

Registro de salida:
El registro carga el contenido del bus de datos y lo muestra como un entero decimal en las pantallas de 7 segmentos. El valor que se muestra se lee del bus de datos y se guarda en un *flip-flop(registro-biestables),* LOAD la señal está activa.

Desarrollo

Para el presente desarrollo y virtualización del una arquitectura SAP y más allá de eso, la creación de una CPU netamente digital por medio de la cual se puedan ejecutar una serie de instrucciones lógicas, se realizó el desarrollo pertinente con el uso de herramientas libres. Esto con el fin de observar como se comporta una unidad de procesamiento central digitalizada por medio de software.

Cabe recalcar que todo el desarrollo del aplicativo se encuentra almacenado dentro de la plataforma Github, la cual nos permite llevar cierto control de versiones y nos facilita el trabajo colaborativo, para proceder con el desarrollo del mismo. El repositorio se puede encontrar en la dirección (*https://github.com/Juferoga/arquitectura/tree/main/proyectoFinal*).

Herramientas utilizadas

GIT

Se utilizó GIT con el fin de proporcionar al equipo de desarrollo una plataforma, para la programación de forma conjunta y más que eso, para implementar un control de versiones que nos permitan conocer el estado actual del proyecto, creación de comentarios sobre el código (dentro de la plataforma Github), que nos permitieron llevar de una forma más organizada el desarrollo del proyecto.

Python

Python, como lenguaje de programación y su curva de aprendizaje eficaz, permite tanto realizar el desarrollo del aplicativo de una manera eficiente como aprender sobre la marcha durante la construcción del código. También, al ser multiparadigma nos permite seleccionar el paradigma valga, la redundancia, con el que se desea abordar el problema, en este caso la construcción de una CPU.

Qt

Qt, al igual que Python, tiene muchas capacidades que nos aportaron para la construcción del software, en este caso para la creación de una interfaz gráfica de con la capacidad de ser multiplataforma, tenemos que recordar que al ser un

framework de desarrollo nos aporta una amplia gama de ayudas para la creación de aplicaciones gráficas.

PyQt5

La librería PyQt5 nos permite manipular y hacer uso de las aplicaciones creadas por medio de Qt, nos aporta a su vez en el sentido de la comunicación o el paso de Python a Qt5 de una forma sencilla y eficiente.

Interfaz Gráfica diseñada

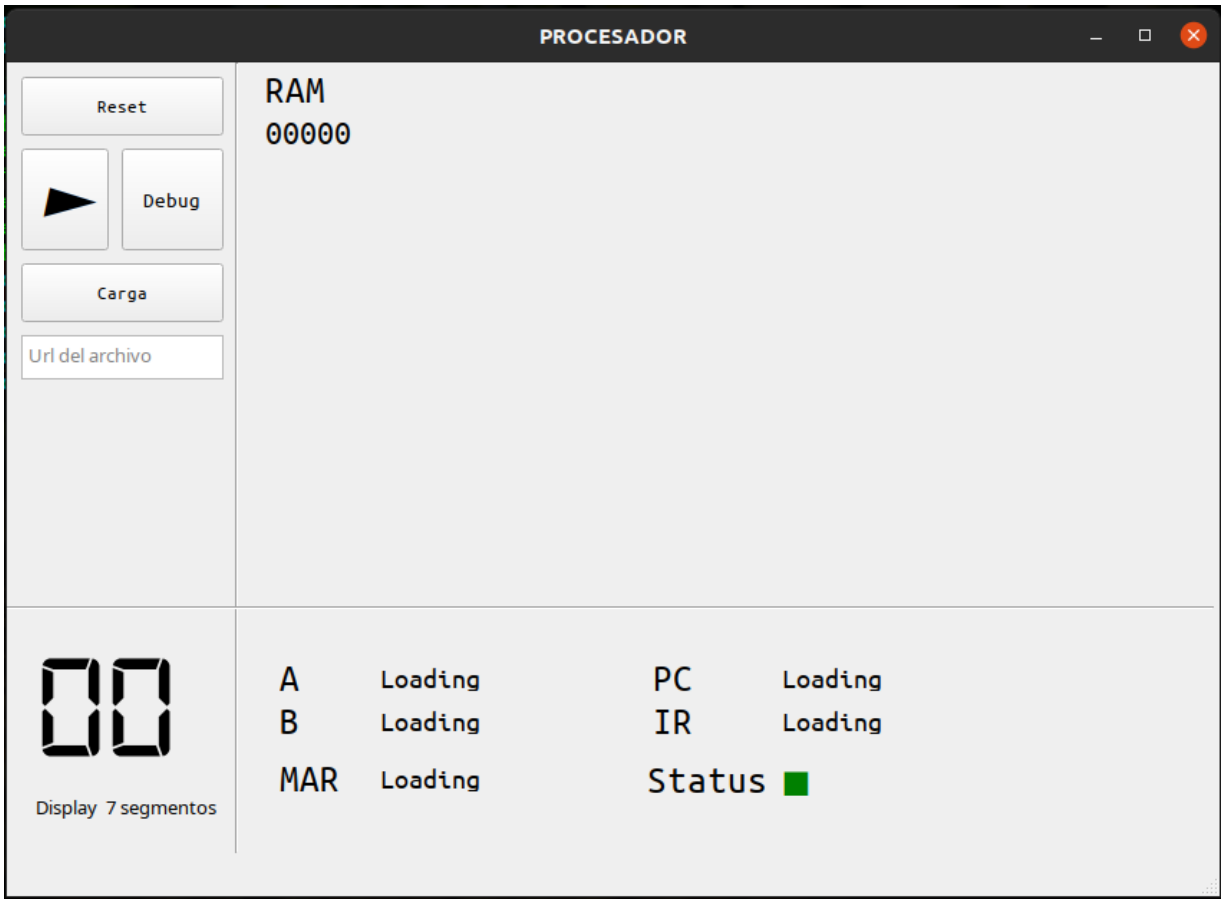

Imagen 3, interfaz principal.

Para la interfaz gráfica, como se mencionó anteriormente, se utiliza Qt por permitirnos tanto trabajar con ella tanto en sistemas operativos como Windows, basados en Linux, entre otros. La anterior interfaz es la interfaz general que nos permite hacer uso del programa, a continuación se realiza una breve descripción de las funcionalidades y disposición del mismo.

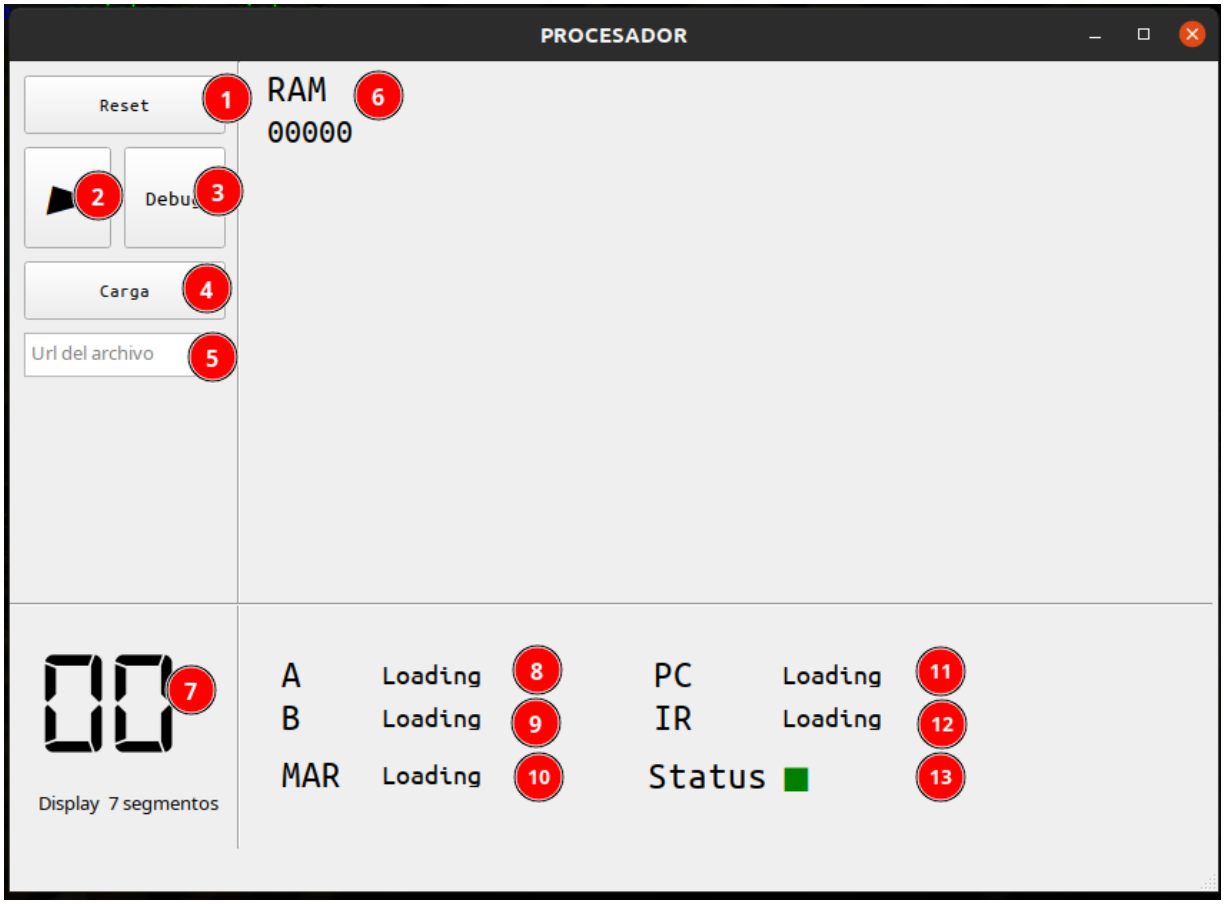

Imagen 4, Descripción interfaz

1. **Botón de reinicio:** Lleva a todas las variables a cero o su estado original.
2. **Botón de inicio:** Inicia el programa que se tenga dentro del aplicativo, de una forma continua haciendo uso del reloj del sistema.
3. **Botón de debug:** Pone en marcha el programa que se tenga dentro del aplicativo, pero con la diferencia de que este lo realiza paso por paso.
4. **Botón de carga:** Efectúa la carga del programa al aplicativo.
5. **Campo dirección:** Recibe la dirección del archivo que se desee ejecutar.
6. **Ram:** Muestra la información almacenada en la memoria RAM.
7. **Display de 7 segmentos:** Muestra la información resultante del procesamiento.
8. **Etiqueta A:** Muestra la información almacenada en el registro A.
9. **Etiqueta B:** Muestra la información almacenada en el registro B.

10. **MAR:** Muestra la información almacenada en el registro MAR.
11. **Etiqueta PC:** Muestra la información almacenada en el registro del contador del programa.
12. **Etiqueta IR:** Muestra la información almacenada en el registro IR.
13. **Etiqueta Status:** Muestra la información almacenada en el estado del programa.

Conclusiones

- El uso de la arquitectura SAP fue indispensable para la realización del proyecto, ya que nos provee una forma simple y sencilla de realizar y pensar en el modelamiento de un procesador, el uso también de herramientas como, Python que poseen un tipado dinámico son bastante útiles para cambiar el tipo de dato de un decimal a binario.

- El tratamiento de los OPCODES dentro de la realización y desarrollo del programa fue fundamental para la creación de la estructura misma, ubicando los lineamientos para la creación y manejo de registros dentro del mismo procesador

Referencias

1. *4-bit d-type register SDLS067A (junio de1999) https://www.ti.com/lit/ds/symlink/sn54173.pdf*
2. *Blanchette, Jasmin; Summerfield, Mark (junio de 2006). «A Brief History of Qt». C++ GUI Programming with Qt 4 (1st edición). Prentice-Hall. pp. xv-xvii.*
3. *Willman, J. (2021). Overview of pyqt5. In Modern PyQt (pp. 1-42). Apress, Berkeley, CA.*
4. *Peiming, G., Shiwei, L., Liyin, S., Xiyu, H., Zhiyuan, Z., Mingzhe, C., & Zhenzhen, L. (2020, August). A PyQt5-based GUI For Operational Verification Of Wave Forcasting System. In 2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS) (pp. 204-211). IEEE.*
5. *MANO Morris y KIME Charles. Fundamentos de Diseño Lógico y Computadoras. Editorial, Prentice Hall. México, 1998.*
6. *RICO Rafael y MARCOS Salvador. Simulación de arquitecturas de computadores mediante*
7. *lenguaje VHDL. Editorial Universidad de Alcalá de Henares. Servicio de Publicaciones. España, 1998.*
8. *http://aserrano.es/portafolio/informatica-con-processing/sap1/*