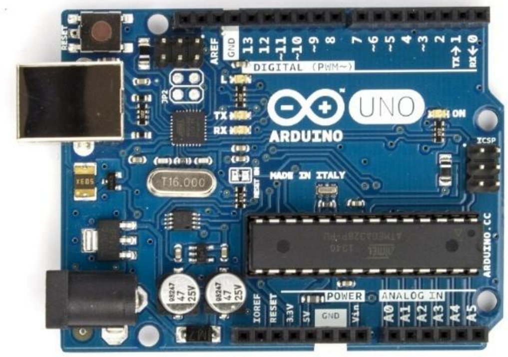
ESTUDIO DE MICROPROCESADORES Y MICROCONTROLADORES

Según Microprocessor 1: Prolegomena (2025), el desarrollo de los microprocesadores y microcontroladores ha sido determinante en la evolución de la computación moderna, al ofrecer soluciones integradas que permiten procesar información de manera eficiente y en tiempo real. Estos componentes son fundamentales para la arquitectura, tiene el propósito de dispositivos inteligentes. Lo microprocesadores están diseñados para ejecutar operaciones complejas, involucrando ciclos de instrucción optimizados y sistemas de gestión de memoria avanzados, como las caches multinivel [1].



Según Philippe Darche (2025), por su parte, los microcontroladores se presentan como soluciones altamente integradas y económicas para tareas específicas. Estos incluyen CPU, RAM, ROM, puertos de entrada/salida, convertidores y temporizadores, todo contenido en un solo chip; su uso es común en automatización industrial, sistemas embebidos, control de sensores y aplicaciones de Internet de las cosas, dado su bajo consumo energético y su capacidad de operar de manera automática [1].

MICROPROCESADOR

ARQUITECTURA BÀSICA

Según Hübner y Becker (2011), la arquitectura de un microprocesador moderno consiste en llevar a cabo múltiples núcleos, cada uno con su propia caché L1 y L2 locales, mientras comparten niveles superiores de caché mediante protocolos de coherencia distribuidos. Esto permite que los datos se muevan eficientemente entre núcleos y memoria, esto ayuda a ir mejorando el rendimiento y la escalabilidad en sistemas multicore. Un ejemplo completo de este tipo de diseño se presenta en la literatura sobre Multiprocessor System-On-Chip, donde se describe la organización de cache y la coherencia en arquitecturas contemporáneas[2].

Según Calimera et al (2021), además, algunos procesadores modernos utilizan arquitecturas con redes internas o Network.on.Chip (NOC) para poder conectar los núcleos y subsistemas de memoria/periféricos. Las NoC permiten múltiples flujos de datos simultáneos de alta velocidad, evitando los cuellos de botella de los buses tradicionales y mejorando la comunicación interna del sistema[3].

CICLO DE INTRUCCIONES Y VELOCIDAD

Según Tang et al (2025), el de instrucción feetch, decode, execute, memory access, write-back; se potencia mediante técnicas de pipelining que permiten solapar las etapas de múltiples instrucciones. Algunas herramientas modernas simulan el comportamiento de pipeline a nivel ciclo, mejorando la precisión de análisis de rendimiento mediante frameworks como PipeDBT, que modelan formalmente los estados de pipeline y permiten simular arquitecturas VLIW con precisión de ciclos [4]

Según Rober J. Colvin (2025) dice que dado que la ejecución fuera de orden (out-of-order) y la especulación pueden dejar residuos en cache que representan canales de fuga de información, es esencial modelar formalmente estos comportamientos. Se han desarrollado modelos semánticos estructurales que permiten comprender estas vulnerabilidades y diseñar procesadores seguros y eficientes simultáneamente [5].

- **Búsqueda (Fetch):** El microprocesador recupera la instrucción desde la memoria principal (RAM), accediendo a la dirección indicada por el contador de programa (PC).
- **Decodificación (Decode):** La unidad de control interpreta la instrucción para identificar la operación que debe ejecutarse y los operandos involucrados.
- **Ejecución (Execute):** Se realiza la operación especificada (suma, carga, salto, etc.).

- **Escritura del resultado (Write-back):** El resultado se guarda en un registro o en memoria.

Formula general para el tiempo de ejecución de un programa:

$$Tiempo\ de\ ejecucion = \frac{Nº\ de\ intrucciones\ \times Ciclos\ por\ instruccion}{Frecuencia\ del\ reloj}$$

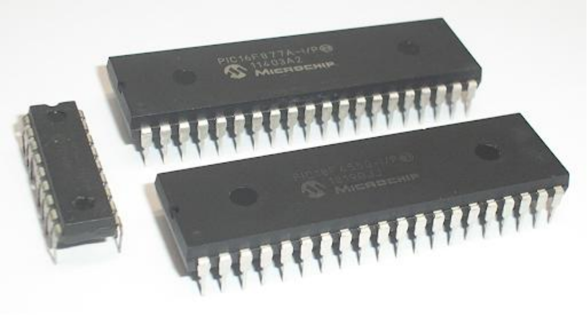
MICROCONTROLADOR

ARQUITECTURA BASICA

Según Proceedings of the ACM/IEEE (2023), la arquitectura de un microcontrolador se basa en la integración del CPU (unidad central de procesamiento), memoria (RAM y ROM); estos son los periféricos de entrada/salida, donde llegan a ser los convertidores analógico-digitales (ADC), y módulos conocidos como la temporización que está dentro de un solo chip. Esta configuración lo convierte en una solución altamente compacta para sistemas embebidos [6].

Sayan Mitra (2023), muestran cómo los diseños modernos pueden llegar a emplear técnicas de mapeo de memoria unificada para mejorar la eficiencia de acceso, así, reduciendo la necesidad de interfaces externas[6].

Según Liu et al (2023), otra característica clave es la inclusión de modos de bajo consumo, se encuentra sleep, Deep sleep e hibernation, especialmente útiles en sistemas alimentados por baterías. Según Liu et al. Los microcontroladores diseñados para aplicaciones de loT suelen incluir controladores DMA y co-procesadores criptográficos integrados, permitiendo realizar tareas complejas con consumo conocido como energético mínimo [7].



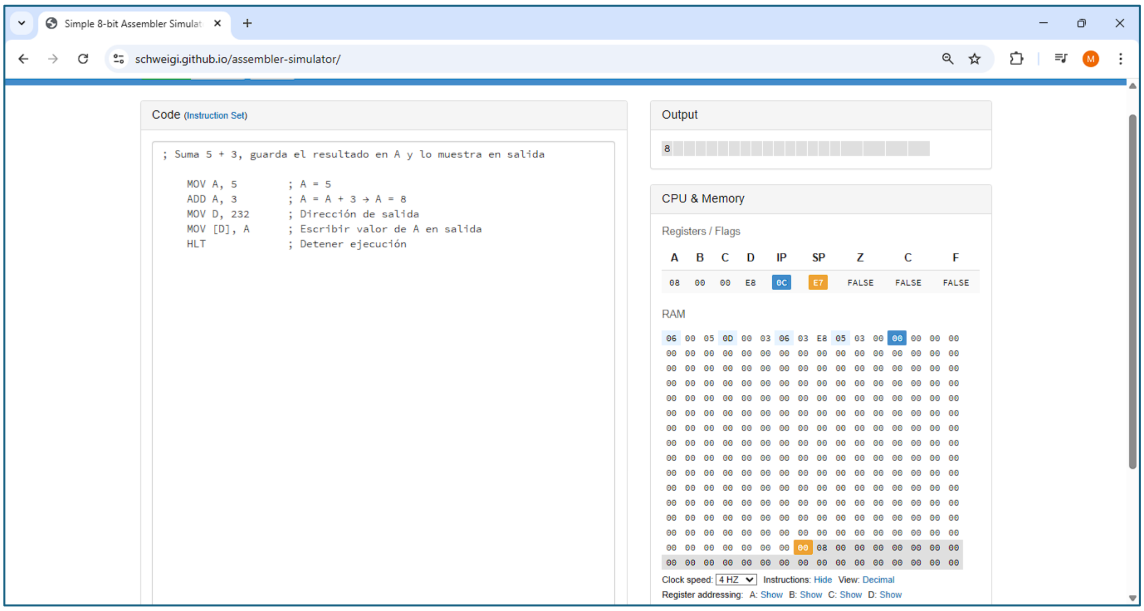
CARACTERISTICAS Y APLICACIONES

Según Ottaviano et al (2024), las Aplicaciones embebidas criticas necesitan que los microcontroladores operan en modos de ultra-bajo consumo (sleep, Deep sleep) y reaccionen rápidamente ante eventos, manteniendo eficiencia energética. Por ejemplo, el sistema EFM32 de Silicon Labs incorpora modos de suspensión profunda con periféricos autónomos capaces de operar sin despertar la CPU, lo que permite prolongar la vida útil de la batería durante años[8]

Según Barchi et al (2024), En el ámbito de TinyMl y sensores inteligentes, se ha demostrado el uso de redes neuronales eficientes implementadas en microcontroladores, optimizando la actividad de spikes y reduciendo significativamente el número de ciclos de computación y consumo de energía en tareas de monitoreo estructura [9].

EJEMPLOS PRACTICOS

1. Suma de dos números



```
; Suma 5 + 3, guarda el resultado en A y lo muestra en salida

    MOV A, 5        ; A = 5
    ADD A, 3        ; A = A + 3 → A = 8
    MOV D, 232      ; Dirección de salida
    MOV [D], A      ; Escribir valor de A en salida
    HLT             ; Detener ejecución
```

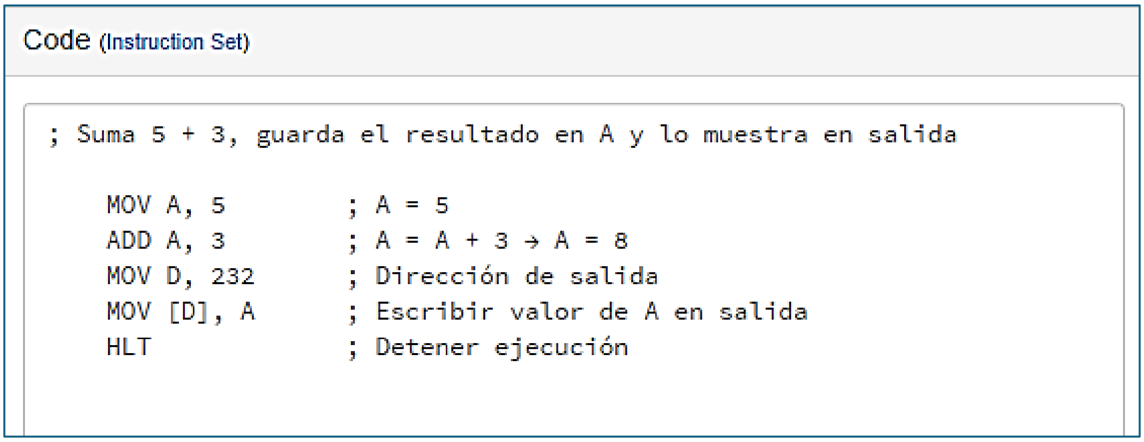
8

A	B	C	D	IP	SP	Z	C	F
08	00	00	E8	0C	E7	FALSE	FALSE	FALSE

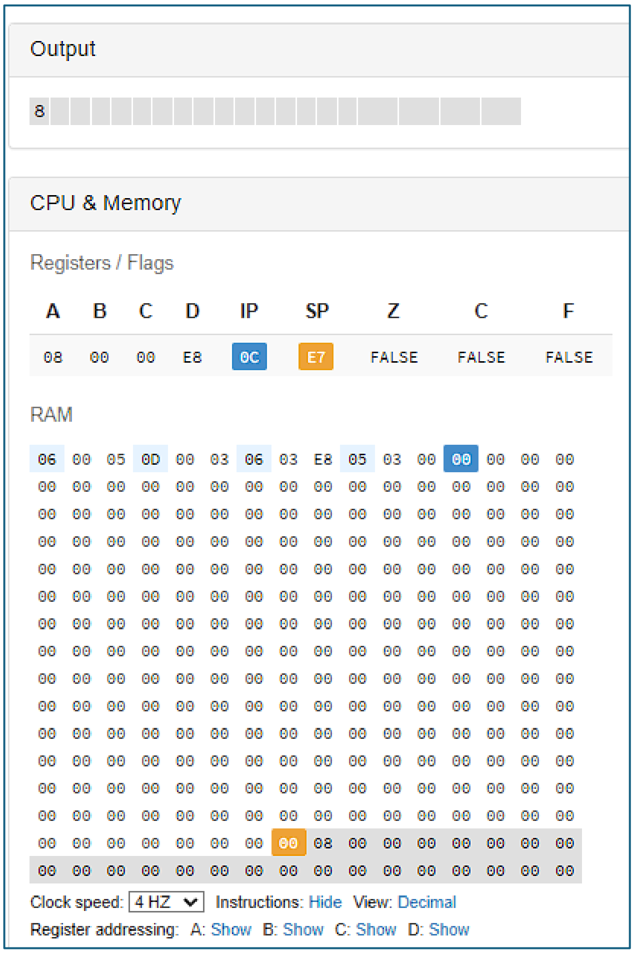
```
06 00 05 0D 00 03 06 03 E8 05 03 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Clock speed: 4 HZ Instructions: Hide View: Decimal

Register addressing: A: Show B: Show C: Show D: Show



```
; Suma 5 + 3, guarda el resultado en A y lo muestra en salida

    MOV A, 5        ; A = 5
    ADD A, 3        ; A = A + 3 → A = 8
    MOV D, 232      ; Dirección de salida
    MOV [D], A      ; Escribir valor de A en salida
    HLT             ; Detener ejecución
```



Output

```
8
```

CPU & Memory

Registers / Flags

A	B	C	D	IP	SP	Z	C	F
08	00	00	E8	0C	E7	FALSE	FALSE	FALSE

RAM

```
06 00 05 0D 00 03 06 03 E8 05 03 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Clock speed: 4 HZ Instructions: Hide View: Decimal

Register addressing: A: Show B: Show C: Show D: Show

ANÁLISIS DEL CICLO:

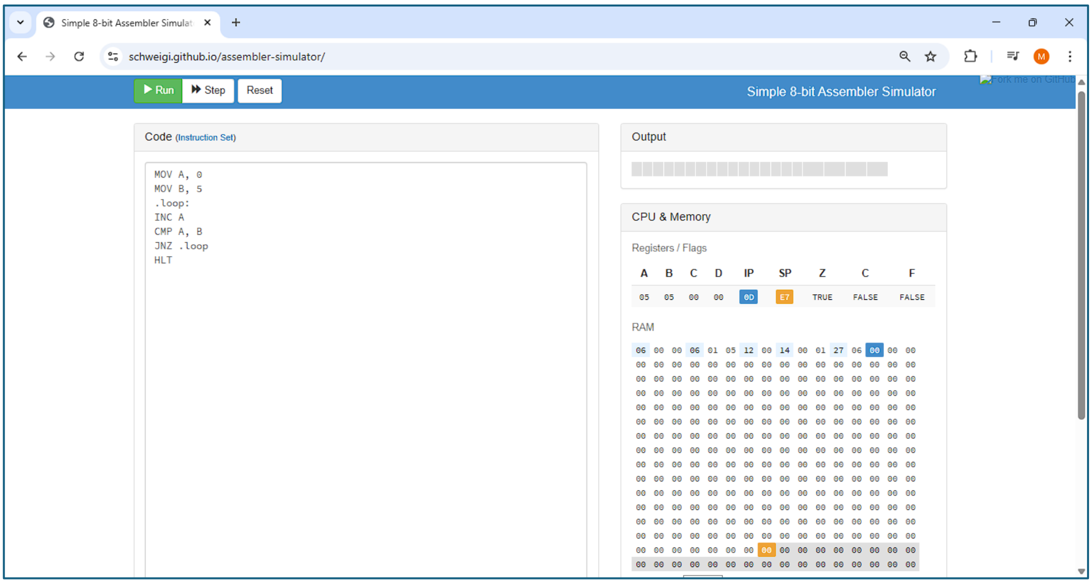
- **MOV** A, 5 : 1 ciclo
- **ADD** A, 3 : 1 ciclo
- **MOV** D, 232 : 1 ciclo
- **MOV** [D], A : 1 ciclo (escritura en salida)
- **HLT** : 1 ciclo
- **Total:** 5 ciclos

VELOCIDAD:

A 1 Hz = 5 segundos

A 4 Hz = 1.25 segundos

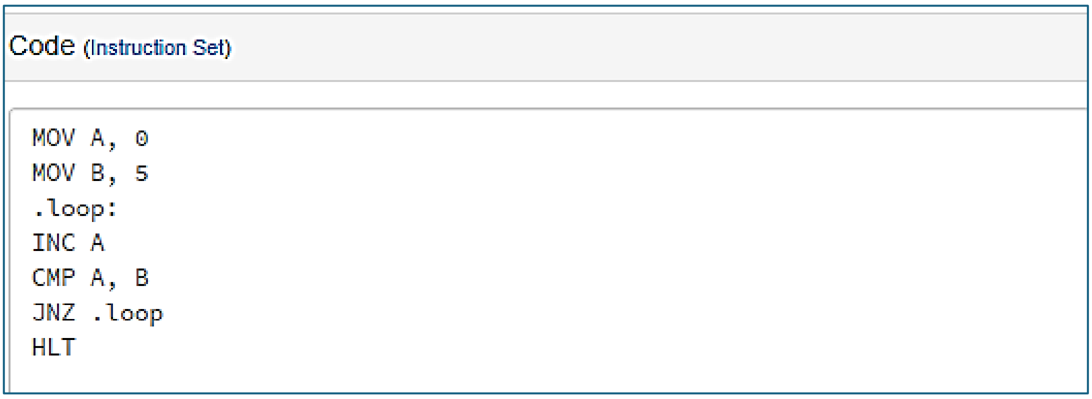
2. Bucle que incrementa A hasta 5



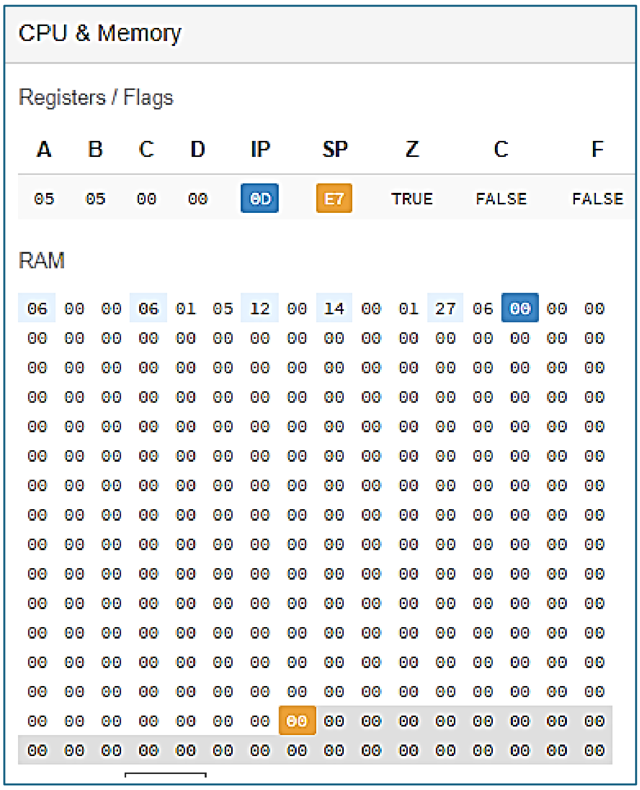
```
MOV A, 0
MOV B, 5
.loop:
INC A
CMP A, B
JNZ .loop
HLT
```

A	B	C	D	IP	SP	Z	C	F
05	05	00	00	0D	E7	TRUE	FALSE	FALSE

```
06 00 00 06 01 05 12 00 14 00 01 27 06 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



MOV A, 0
MOV B, 5
.loop:
INC A
CMP A, B
JNZ .loop
HLT



A B C D IP SP Z C F
05 05 00 00 0D E7 TRUE FALSE FALSE
06 00 00 06 01 05 12 00 14 00 01 27 06 00 00 00

CICLO DE INSTRUCCIONES POR VUELTA:

- **INC** A (1 ciclo)
- **CMP** A, B (1 ciclo)
- **JNZ** (1 ciclo)

El bucle se repite 5 veces, entonces:

- 3 ciclos × 5 = 15 ciclos

Más los 2 primeros (MOV A, MOV B) + HLT: 3 ciclos

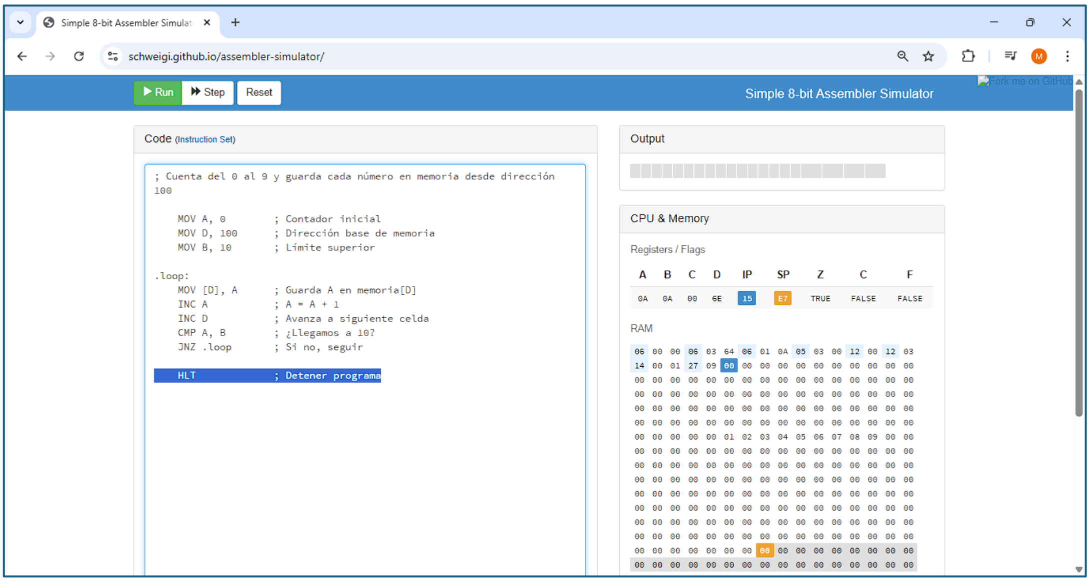
Total: 18 ciclos

VELOCIDAD:

18 ciclos a 1 Hz = 18 segundos

18 ciclos a 4 Hz = 4.5 segundos

3. Contar del 0 al 9 y guardar en memoria



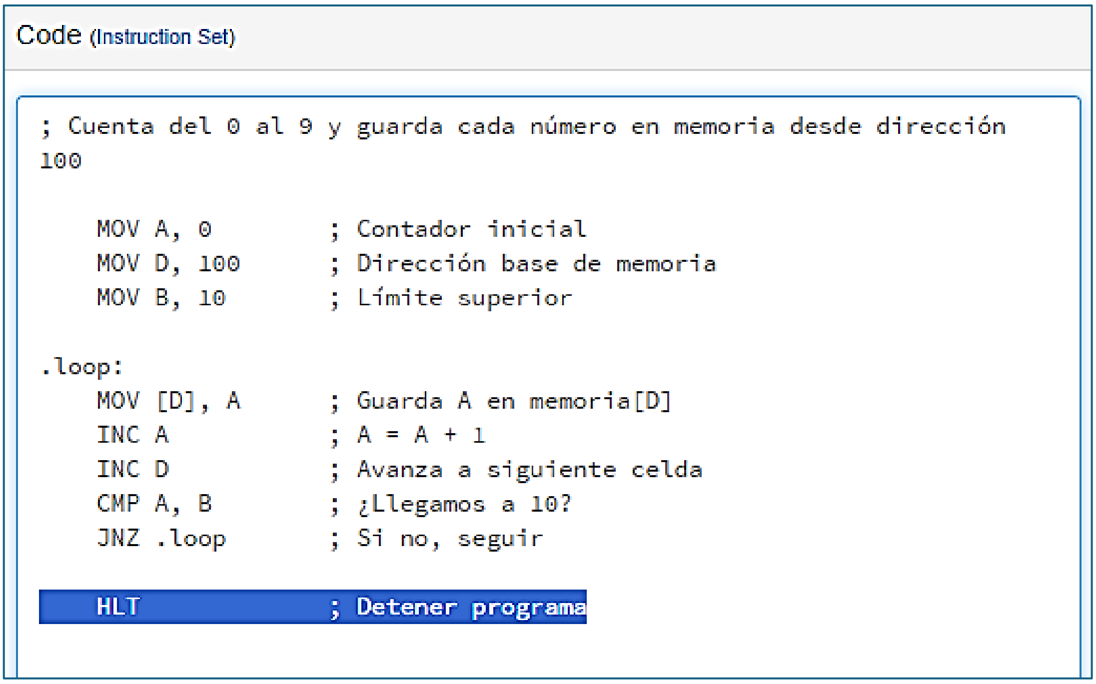
```
; Cuenta del 0 al 9 y guarda cada número en memoria desde dirección
100

    MOV A, 0        ; Contador inicial
    MOV D, 100      ; Dirección base de memoria
    MOV B, 10       ; Límite superior

.loop:
    MOV [D], A      ; Guarda A en memoria[D]
    INC A           ; A = A + 1
    INC D           ; Avanza a siguiente celda
    CMP A, B        ; ¿Llegamos a 10?
    JNZ .loop       ; Si no, seguir

    HLT             ; Detener programa
```

Registers / Flags

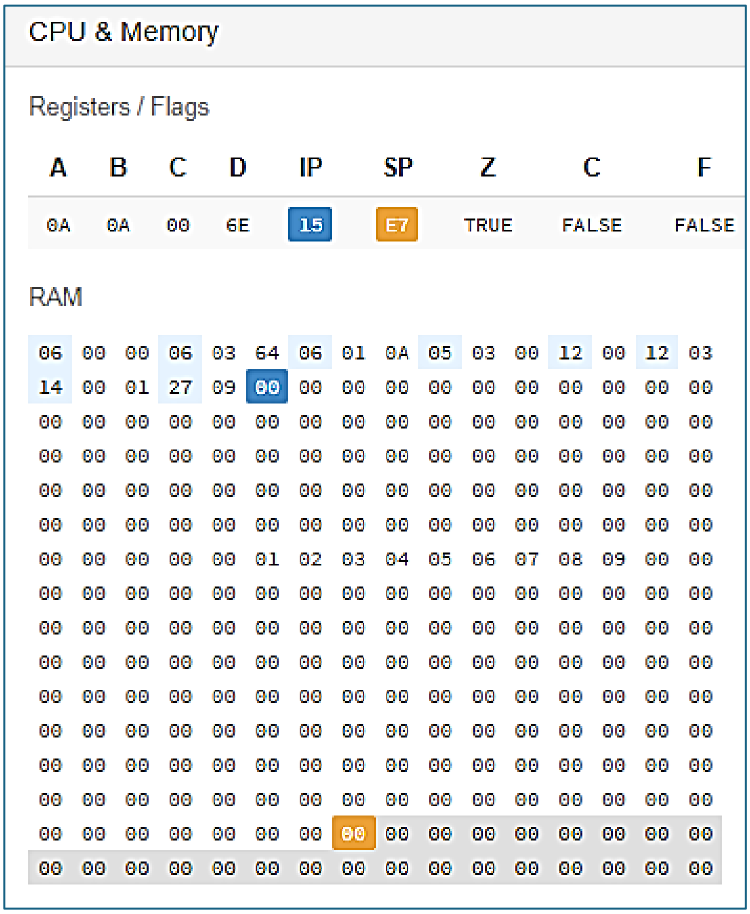
A	B	C	D	IP	SP	Z	C	F
0A	0A	00	6E	15	E7	TRUE	FALSE	FALSE

RAM

```
06 00 00 06 03 64 06 01 0A 05 03 00 12 00 12 03
14 00 01 27 09 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 01 02 03 04 05 06 07 08 09 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
; Cuenta del 0 al 9 y guarda cada número en memoria desde dirección
100

    MOV A, 0        ; Contador inicial
    MOV D, 100      ; Dirección base de memoria
    MOV B, 10       ; Límite superior

.loop:
    MOV [D], A      ; Guarda A en memoria[D]
    INC A           ; A = A + 1
    INC D           ; Avanza a siguiente celda
    CMP A, B        ; ¿Llegamos a 10?
    JNZ .loop       ; Si no, seguir

    HLT             ; Detener programa
```



A	B	C	D	IP	SP	Z	C	F
0A	0A	00	6E	15	E7	TRUE	FALSE	FALSE

```
06 00 00 06 03 64 06 01 0A 05 03 00 12 00 12 03
14 00 01 27 09 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 01 02 03 04 05 06 07 08 09 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

ANÁLISIS DEL CICLO:

Instrucciones iniciales (MOV A, MOV D, MOV B) : 3 ciclos

Bucle:

- **MOV** [D], A : 1 ciclo
- **INC** A : 1 ciclo
- **INC** D : 1 ciclo
- **CMP** A, B : 1 ciclo
- **JNZ** .loop : 1 ciclo

El bucle se repite 10 veces, entonces:

- =5 × 10 = 50 ciclos

HLT : 1 ciclo

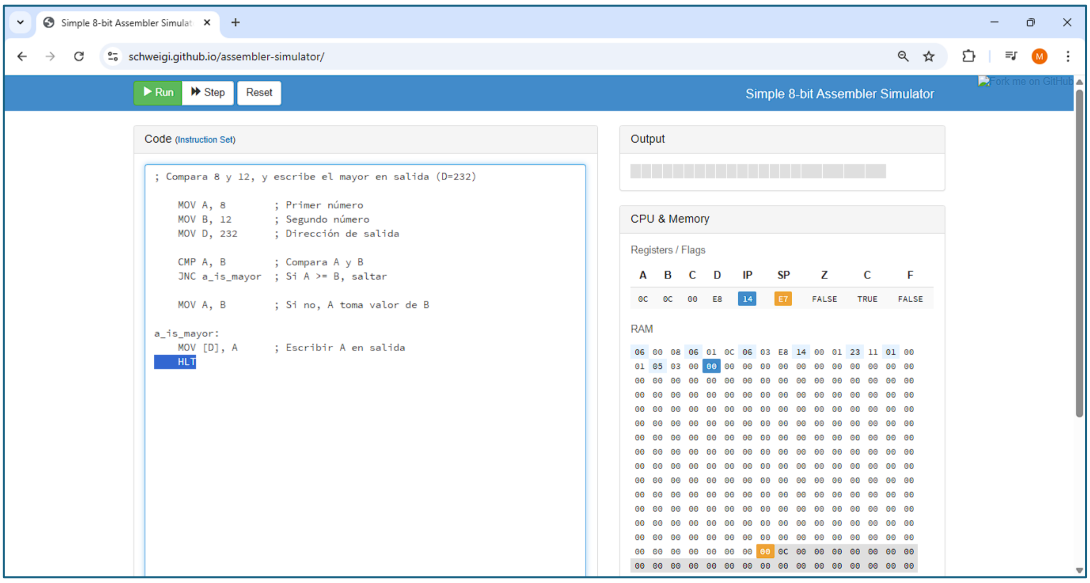
Total: 3 + 50 + 1 = 54 ciclos

VELOCIDAD:

A 1 Hz = 54 segundos

A 4 Hz = 13.5 segundos

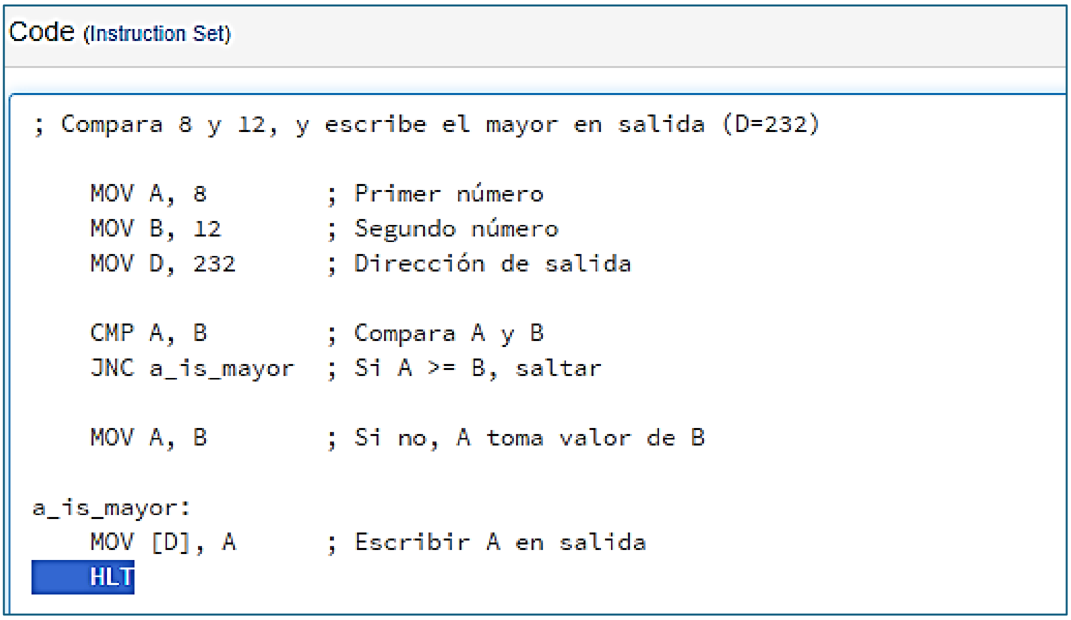
4. **Comparar 8 y 12, mostrar el mayor**



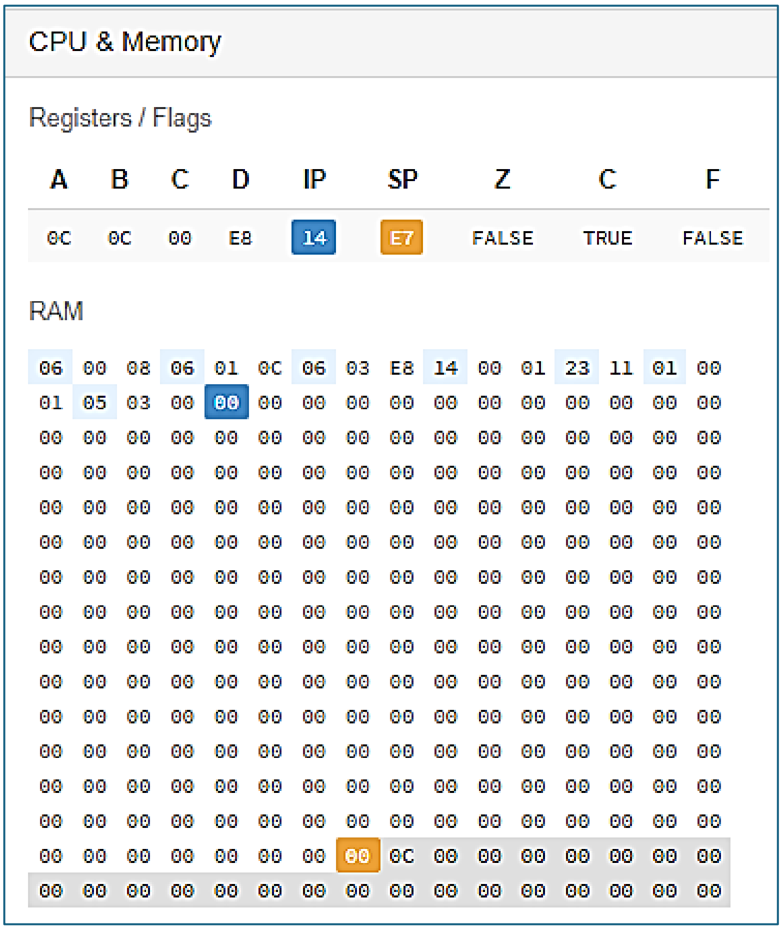
```
; Compara 8 y 12, y escribe el mayor en salida (D=232)

    MOV A, 8        ; Primer número
    MOV B, 12       ; Segundo número
    MOV D, 232      ; Dirección de salida

    CMP A, B        ; Compara A y B
    JNC a_is_mayor  ; Si A >= B, saltar

    MOV A, B        ; Si no, A toma valor de B

a_is_mayor:
    MOV [D], A      ; Escribir A en salida
    HLT
```

A	B	C	D	IP	SP	Z	C	F
0C	0C	00	E8	14	E7	FALSE	TRUE	FALSE

```
06 00 08 06 01 0C 06 03 E8 14 00 01 23 11 01 00
01 05 03 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
; Compara 8 y 12, y escribe el mayor en salida (D=232)

    MOV A, 8        ; Primer número
    MOV B, 12       ; Segundo número
    MOV D, 232      ; Dirección de salida

    CMP A, B        ; Compara A y B
    JNC a_is_mayor  ; Si A >= B, saltar

    MOV A, B        ; Si no, A toma valor de B

a_is_mayor:
    MOV [D], A      ; Escribir A en salida
    HLT
```



A	B	C	D	IP	SP	Z	C	F
0C	0C	00	E8	14	E7	FALSE	TRUE	FALSE

```
06 00 08 06 01 0C 06 03 E8 14 00 01 23 11 01 00
01 05 03 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

ANÁLISIS DEL CICLO:

MOV A, 8 : 1 ciclo

MOV B, 12 : 1 ciclo

MOV D, 232 : 1 ciclo

CMP A, B : 1 ciclo

JNC : 1 ciclo

En caso de que A < B : se ejecuta MOV A, B : 1 ciclo

MOV [D], A : 1 ciclo

HLT : 1 ciclo

Total: 7 u 8 ciclos (según resultado de comparación)

VELOCIDAD:

A 1 Hz = 7–8 segundos

A 4 Hz = 1.75–2 segundos

[1] Philippe Darche, “Microprocessor 1: Prolegomena - Calculation and Storage Functions - Models of Computation and Computer Architecture | Wiley eBooks | IEEE Xplore.” [Online]. Available: https://ieeexplore.ieee.org/book/10523213

[2] M. Hübner and J. Becker, “Multiprocessor System-on-Chip,” *Multiprocessor System-on-Chip: Hardware Design and Tool Integration*, pp. 1–270, 2011, [Online]. Available: https://link.springer.com/10.1007/978-1-4419-6460-1

[3] A. Calimera, P.-E. Gaillardon, K. Korgaonkar, S. Kvatinsky, and R. Reis, Eds., “VLSI-SoC: Design Trends,” vol. 621, 2021, doi: 10.1007/978-3-030-81641-4.

[4] T. Tang, Y. Man, X. Zhou, and D. Wang, “Pipe-DBT: enhancing dynamic binary translation simulators to support pipeline-level simulation,” *Automated Software Engineering*, vol. 32, no. 2, pp. 1–35, Nov. 2025, doi: 10.1007/S10515-025-00506-8/TABLES/6.

[5] R. J. Colvin and R. C. Su, “Structural Operational Semantics for Functional and Security Verification of Pipelined Processors,” pp. 363–388, 2025, doi: 10.1007/978-3-031-98668-0_18.

[6] “Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023) | ACM Conferences.” Accessed: Jul. 26, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3576841

[7] K. Akhunov, E. Yildiz, and K. S. Yildirim, “Enabling Efficient Intermittent Computing on Brand New Microcontrollers via Tracking Programmable Voltage Thresholds,” *ENSsys 2023 - Proceedings of the 2023 11th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems, Part of: SenSys 2023*, vol. 12, pp. 16–22, Nov. 2023, doi: 10.1145/3628353.3628547.

[8] A. Ottaviano *et al.*, “ControlPULP: A RISC-V On-Chip Parallel Power Controller for Many-Core HPC Processors with FPGA-Based Hardware-In-The-Loop Power and Thermal Emulation,” *Int J Parallel Program*, vol. 52, no. 1–2, pp. 93–123, Apr. 2024, doi: 10.1007/S10766-024-00761-4/FIGURES/11.

[9] F. Barchi, E. Parisi, L. Zanatta, A. Bartolini, and A. Acquaviva, “Energy efficient and low-latency spiking neural networks on embedded microcontrollers through spiking activity tuning,” *Neural Comput Appl*, vol. 36, no. 30, pp. 18897–18917, Oct. 2024, doi: 10.1007/S00521-024-10191-5/TABLES/5.