



Simulación de la Máquina plus Microprogramada
Arquitectura de Computadoras

Cátedra: Ing. Dignani Jorge, Lic. Pacheco Cristian

Alumnos: Sierra Yesica, Sepúlveda Ariel

2022

Índice

| | |
|---|-----------|
| 1. Descripción del Proyecto | 1 |
| 2. Máquina plus | 1 |
| 3. Evolución del Circuito | 2 |
| 4. Unidad de Control Microprogramada | 14 |
| 5. Descripción del Circuito | 14 |
| 5.1. Unidad de Control | 14 |
| 5.2. Tabla de Instrucciones | 15 |
| 5.2.1. Descripción del Contenido | 18 |
| 5.3. Memoria de Control | 20 |
| 5.3.1. Señales de Control | 20 |
| 5.3.2. Descripción del Contenido | 22 |
| 5.3.3. Descripción detallada de la memoria de control | 26 |
| 5.4. Memoria de Direcciónamiento Explícito | 45 |
| 6. ALU | 46 |
| 6.1. Códigos de Operación de la ALU | 47 |
| 7. Banco de Registros | 48 |
| 8. IR | 49 |
| 9. Simulación de un programa | 49 |
| 9.1. MOV C | 51 |
| 9.2. ADD A | 59 |
| 9.3. ANA B | 69 |
| 9.4. STA 0020 | 78 |
| 10. Apéndice | 90 |
| 10.1. Logisim | 90 |
| 11. Conclusiones | 91 |
| 12. Referencias | 91 |

Índice de cuadros

| | | |
|-----|---|----|
| 1. | Señales de control de la versión de Borda, Paula y Verdi, Lautaro | 8 |
| 2. | Incorporación de la operación NOT en la Alu | 12 |
| 3. | Tabla de instrucciones 1/3 | 18 |
| 4. | Tabla de instrucciones 2/3 | 19 |
| 5. | Tabla de instrucciones 3/3 | 20 |
| 6. | Señales de control | 21 |
| 7. | Memoria de control detallada 1/19 | 26 |
| 8. | Memoria de control detallada 2/19 | 27 |
| 9. | Memoria de control detallada 3/19 | 28 |
| 10. | Memoria de control detallada 4/19 | 29 |
| 11. | Memoria de control detallada 5/19 | 30 |
| 12. | Memoria de control detallada 6/19 | 31 |
| 13. | Memoria de control detallada 7/19 | 32 |
| 14. | Memoria de control detallada 8/19 | 33 |
| 15. | Memoria de control detallada 9/19 | 34 |
| 16. | Memoria de control detallada 10/19 | 35 |
| 17. | Memoria de control detallada 11/19 | 36 |
| 18. | Memoria de control detallada 12/19 | 37 |
| 19. | Memoria de control detallada 13/19 | 38 |
| 20. | Memoria de control detallada 14/19 | 39 |
| 21. | Memoria de control detallada 15/19 | 40 |
| 22. | Memoria de control detallada 16/19 | 41 |
| 23. | Memoria de control detallada 17/19 | 42 |
| 24. | Memoria de control detallada 18/19 | 43 |
| 25. | Memoria de control detallada 19/19 | 44 |
| 26. | Descripción de los códigos de memoria de direccionamiento explícito | 46 |
| 27. | Códigos de Operación de la ALU | 47 |

Índice de figuras

| | | |
|-----|--|----|
| 1. | Unidad de Control de Alexis de Barrientos | 2 |
| 2. | Circuito completo de la UC de Alexis Barrientos | 3 |
| 3. | Circuito en Logisim de la Máquina plus de Jaureguibehere | 4 |
| 4. | Micro Memoria implementada por Jaureguibehere | 5 |
| 5. | Descripción de los tres bits para indicar la próxima dirección | 5 |
| 6. | Unidad de Control de la Máquina plus de Jaureguibehere | 6 |
| 7. | ALU de la Máquina plus de Jaureguibehere | 6 |
| 8. | Banco de Registros de la Máquina plus de Jaureguibehere | 7 |
| 9. | Micro Memoria mejorada por Alvarado y Fernandez | 10 |
| 10. | Descripción de los tres bits que indican la próxima dirección | 10 |
| 11. | Unidad de Control mejorada de Alvarado y Fernandez | 11 |
| 12. | ALU mejorada por Alvarado y Fernandez | 11 |
| 13. | Vista del sistema de fases de reloj de Cotrena, Federico | 12 |
| 14. | Vista general de la Máquina plus con el sistema de fases de reloj | 13 |
| 15. | Unidad de Control de la Máquina plus | 15 |
| 16. | Vista parcial de la Tabla de Instrucciones | 16 |
| 17. | Vista completa de la Tabla de Instrucciones en Editor Hex de Logisim | 16 |
| 18. | Descripción visual de la Memoria de Control. 1/4. | 22 |
| 19. | Descripción visual de la Memoria de Control. 2/4. | 23 |
| 20. | Descripción visual de la Memoria de Control. 3/4. | 24 |
| 21. | Descripción visual de la Memoria de Control. 4/4. | 25 |
| 22. | Memoria de Direcciónamiento Explícito | 45 |
| 23. | Unidad Aritmética Lógica (ALU) | 46 |
| 24. | Banco de Registros | 48 |
| 25. | IR | 49 |
| 26. | Interfaz de Logisim | 91 |

1. Descripción del Proyecto

El proyecto consiste en simular la Máquina plus, que fue vista en la materia Elementos de Informática (EDI), donde realizamos nuestras primeras prácticas de programación en lenguaje Assembler.

La simulación de la Máquina plus se realizó en Logisim, un simulador de circuitos electrónicos que aprendimos a usar en la materia Arquitectura de Computadoras (AdC), correlativa a EDI.

El proyecto fue pasando por distintos estudiantes de la materia AdC, los cuales fueron agregando y mejorando las partes del circuito, como la memoria principal, la unidad de control, los registros, el uso de relojes, la tabla de instrucciones, etc.

2. Máquina plus

La Máquina plus es una herramienta que permite simular un procesador que posee las características de la arquitectura Von Neumann. El objetivo del procesador es ejecutar instrucciones máquina, donde se parte de un programa máquina en memoria principal y se observa el flujo de ejecución. Como las instrucciones están en memoria se accede a ellas mediante direcciones, una vez que se conoce la próxima dirección, la ejecución de la instrucción contiene las siguientes fases:

1. Búsqueda de la instrucción (fetch).
2. Decodificación.
3. Cálculo de la dirección de los operandos.
4. Búsqueda de los operandos.
5. Ejecución (realiza la operación y almacena el resultado).

Cada instrucción máquina debe especificar:

1. La operación a realizar en el Código de Operación (C.O.)
2. Información para calcular las direcciones de los operandos y donde se guarda el resultado (Modos de Direccionamiento).
3. Información de la dirección de la próxima instrucción a ejecutar.

La Máquina plus originalmente está hecha íntegramente en lenguaje C mediante el compilador Turbo C, por Iñigo Gomez Aguinaga orientado por José M. Angulo Usategui. Cuenta con las siguientes características:

- CPU de 8 bits
- Registros generales de 8 bits

- Bus de datos de 8 bits
- Bus de direcciones de 16 bits
- Registro PC (Program Counter) de 16 bits
- Registro MAR (Memory Address Register) de 16 bits
- Modos de direccionamiento:
 - Inmediato: El operador se encuentra explícitamente en la instrucción.
 - Directo registro: El operando se encuentra en un registro.
 - Directo: El operando se encuentra en la dirección de memoria principal que contiene la instrucción.
 - Indirecto registro: El o los registros contienen la dirección en donde se encuentra el operando.

También nos permite cargar programas en memoria para probarlos y ver el flujo de ejecución.

3. Evolución del Circuito

El primer trabajo didáctico que se realizó sobre la Máquina plus fue el de Barrientos, Alexis quien simuló la unidad de control microprogramada de Wilkes en Logisim Evolution (v2.14.6). Esta cuenta con una memoria de control de 64 posiciones de 25 bits cada una, fue pensada como un módulo con las siguientes entradas 6 bits para direccionamiento, 1 bit para la bandera C (Carry), 1 bit para la bandera Z (Zero), y una salida de 16 bits de señales de control.

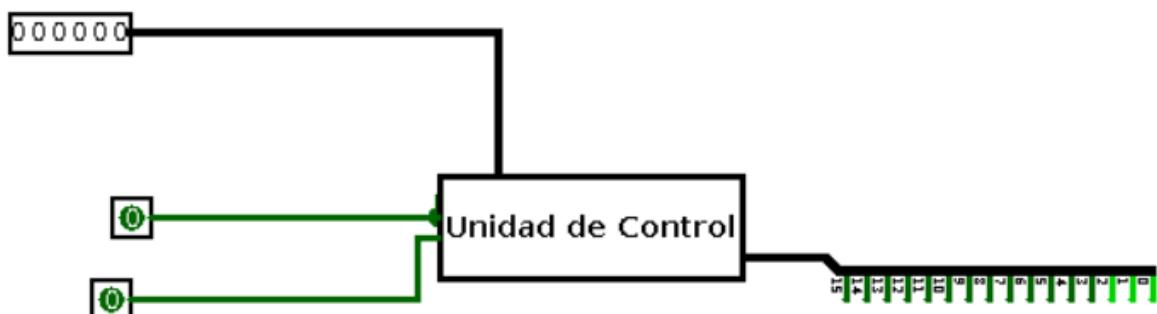


Figura 1: Unidad de Control de Alexis de Barrientos

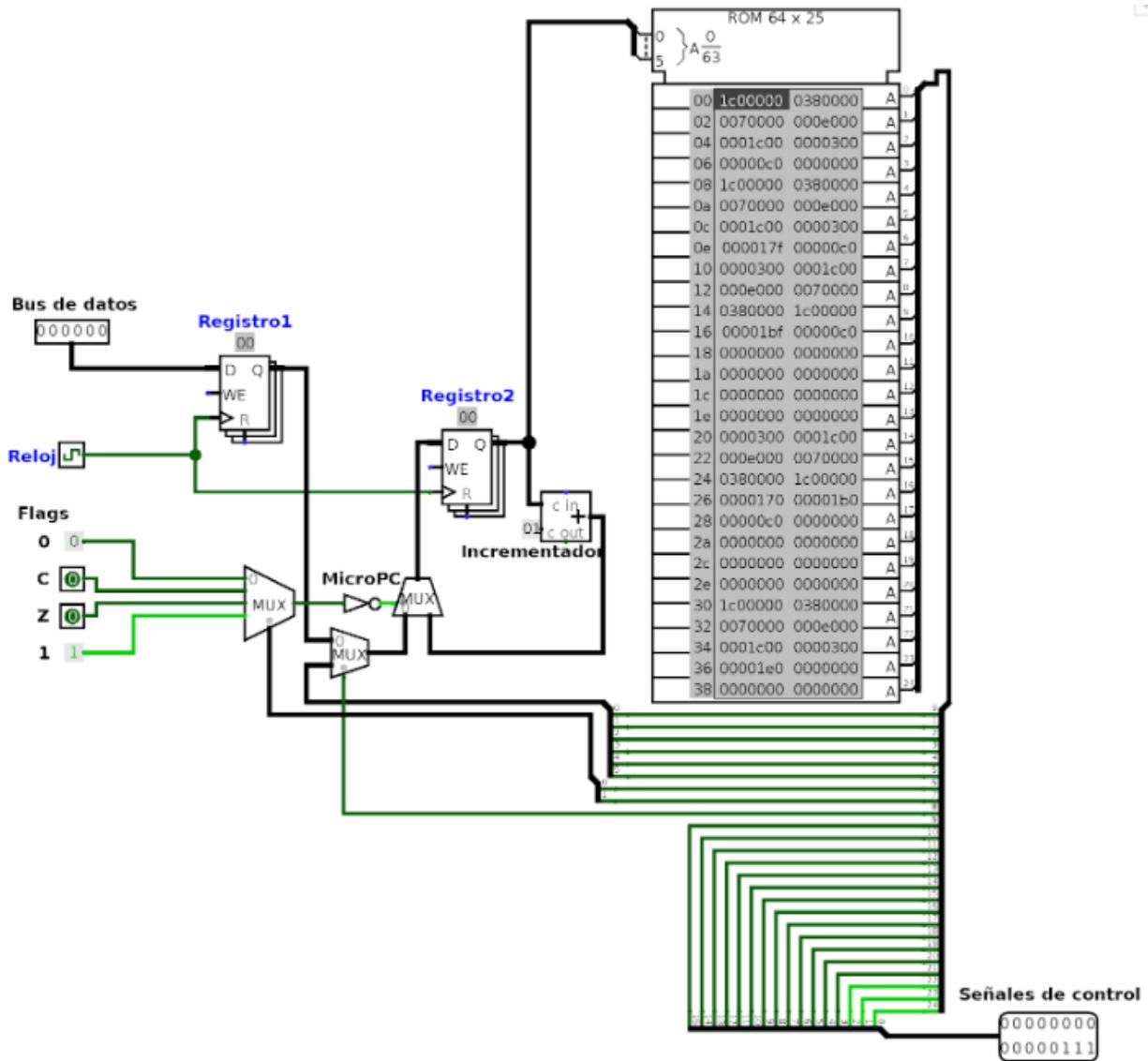


Figura 2: Circuito completo de la UC de Alexis Barrientos

La segunda versión del circuito fue la implementación de la arquitectura y diseño de la Máquina plus en Logisim Evolution, utilizando y mejorando la unidad de control de Barrientos por parte del estudiante Jaureguibehere, Markel. En esta versión tenemos una memoria principal de 64k con ocho bits en cada posición, un banco de registros con los registros A, B, C, D, E de ocho bits, registros PC y MAR de 16 bits, la ALU con entrada de dos registros para los operandos y tres bits para la operación a realizar, la salida de la ALU con el resultado y indicadores para las banderas Carry y Zero. Cuenta con una unidad de control Enriquecida con una memoria para direccionamiento explícito de 256 posiciones de 11 bits cada una, se aumentó el tamaño de la memoria de control a 256 palabras de 32 bits y ocho bits para direccionamiento a ambas memorias. La salida ahora es de 32 señales de control.

Si bien esta versión funciona porque fue probada con un programa, no era una versión funcional porque no tenía cargadas las instrucciones de la Máquina plus.

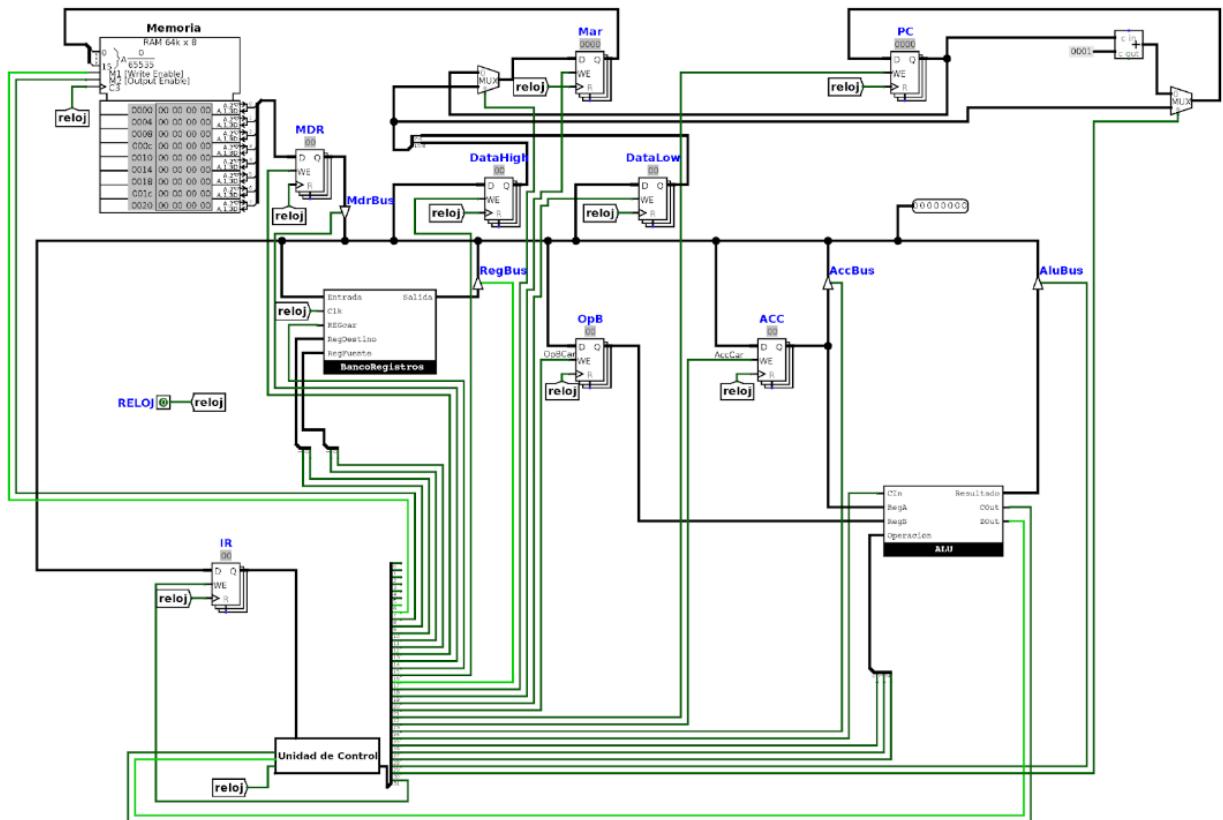


Figura 3: Circuito en Logisim de la Máquina plus de Jaureguibehere

Micro memoria de 256 palabras de 43 bits

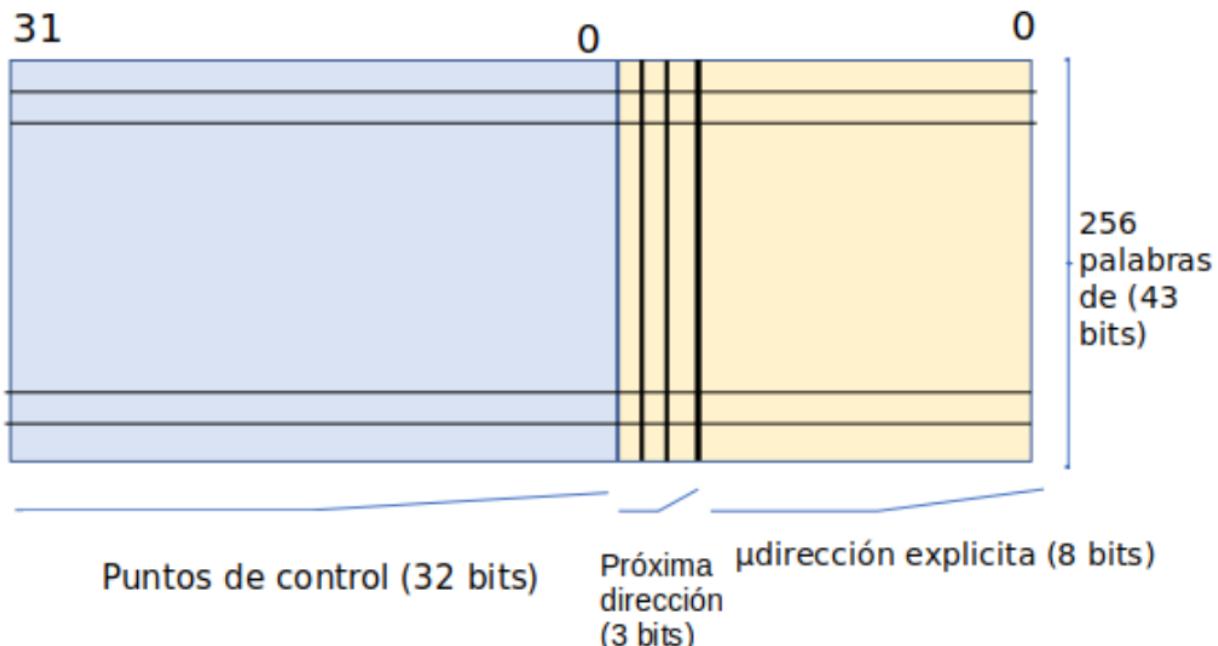


Figura 4: Micro Memoria implementada por Jaureguibehere

| Bit 10 | Bit 9 | Bit 8 | Próxima μdirección |
|--------|-------|-------|-----------------------------|
| 0 | x | x | IR |
| 1 | 0 | 0 | μPC+1 (implicito) |
| 1 | 0 | 1 | Si C=0 μPC+1 (implicito) |
| | | | Si C=1 μdirección expl. |
| 1 | 1 | 0 | Si Z=0 μPC+1 (implicito) |
| | | | Si Z=1 μdirección expl. |
| 1 | 1 | 1 | μdirección explícita |

Figura 5: Descripción de los tres bits para indicar la próxima dirección

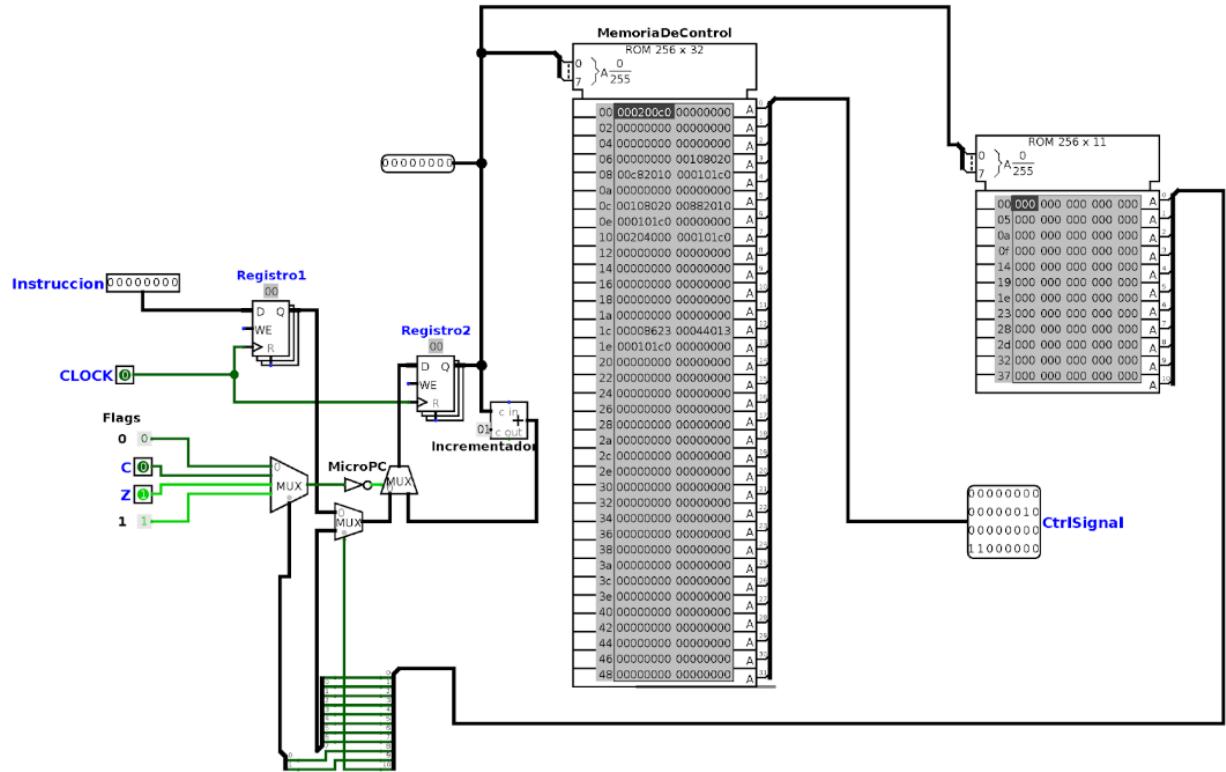


Figura 6: Unidad de Control de la Máquina plus de Jaureguibehere

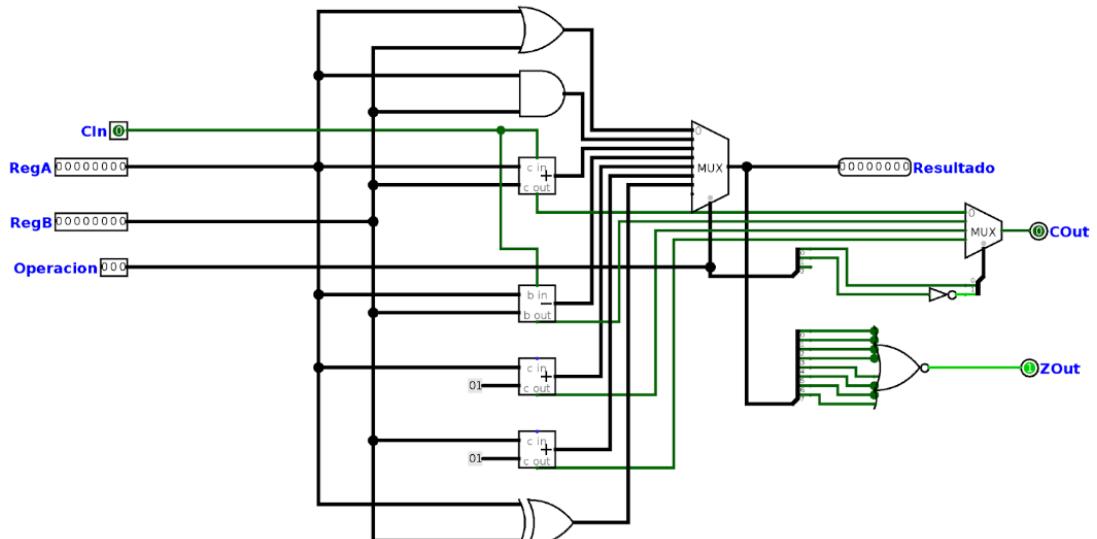


Figura 7: ALU de la Máquina plus de Jaureguibehere

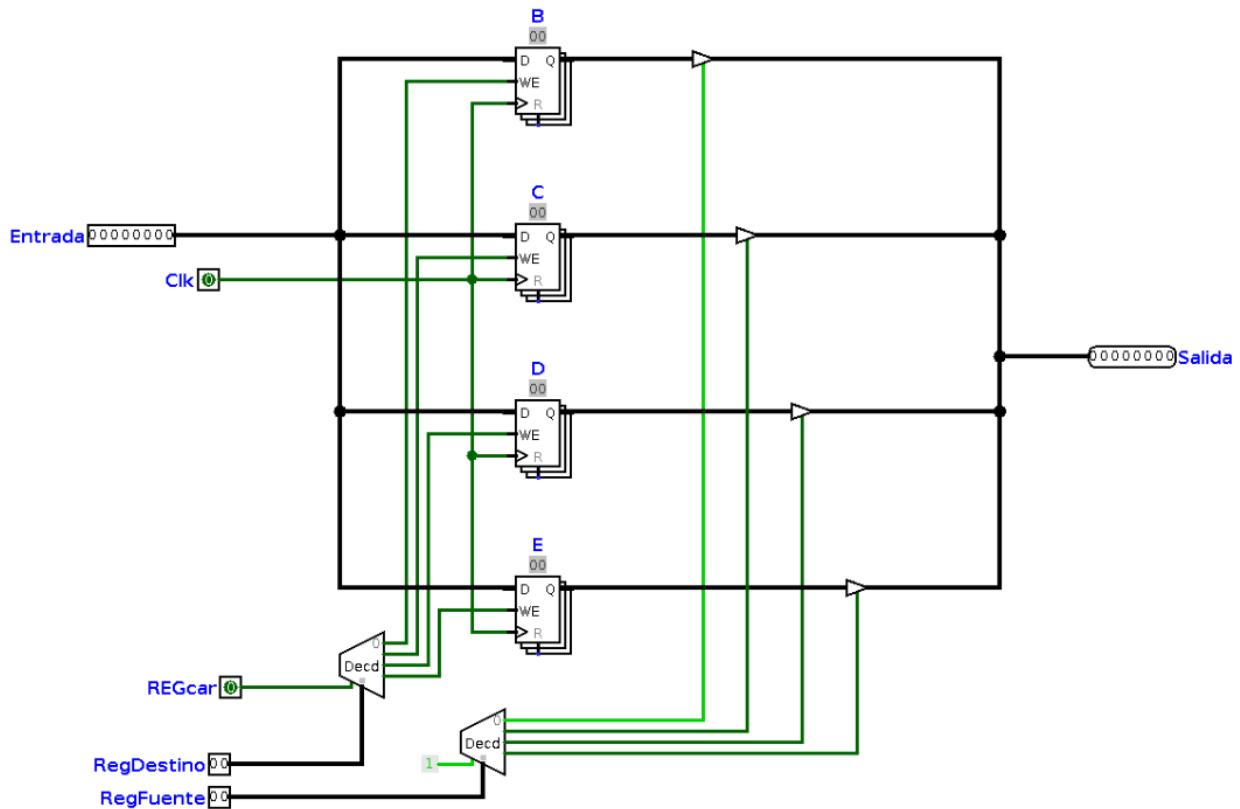


Figura 8: Banco de Registros de la Máquina plus de Jaureguibehere

El siguiente paso estuvo en manos de los estudiantes Borda, Paula y Verdi, Lautaro que agregaron los primeros microprogramas a la memoria de control de la unidad de control de las siguientes instrucciones:

- LDA dir
- STAX
- MOV B, A
- ANI imm
- ADD C
- SUB C
- LDAX
- JMP dir

- CMP B
- BEQ dir
- STA dir

Para ejecutar una instrucción se entraba directo a la memoria de control y desde allí saltaban al inicio de su microprograma, ya que aún no existía la tabla de decodificación. En esta versión esta era la descripción de las señales de control:

| BIT | DESCRIPCIÓN |
|----------|---|
| 0 - 6 | Sin definir, para uso futuro. |
| 7 | Habilita la escritura de datos en la memoria principal. |
| 8 | Habilita la salida o lectura de datos en la memoria principal. |
| 9 - 10 | En conjunto seleccionan el registro destino. |
| 11 - 12 | En conjunto seleccionan el registro fuente. |
| 13 | Habilita la escritura en el registro MDR. |
| 14 | Habilita al registro MDR para transmitir datos por el bus. |
| 15 | Habilita el registro de carga. |
| 16 | Habilita la escritura en el registro temporal data high. |
| 17 | Habilita la salida de un dato proveniente del banco de registros. |
| 18 | Indica la selección del multiplexor, si es 0 el registro MAR toma el valor del PC, si está en 1 toma la dirección formada por los registros data high y data low. |
| 19 | Habilita la escritura en el registro MAR. |
| 20 | Habilita la escritura en el registro temporal data low. |
| 21 | Habilita la escritura en el registro OpB (operando B). |
| 22 | Habilita la escritura en el registro PC (contador de programa). |
| 23 | Habilita la escritura en el registro ACC (acumulador). |
| 24 | Habilita al registro ACC para transmitir datos por el bus. |
| 25 | Indica si en la operación habrá carry. |
| 26 al 28 | En conjunto forman el código de la operación a realizarse. |
| 29 | Habilita a la ALU a la transmisión de datos (resultado) por el bus. |
| 30 | Es usado para la selección de un multiplexor, si es 0 el contador de programa (PC) se incrementa, si es 1 se produce un salto ya que el contador de programa (PC) toma la dirección formada por los registros data high y data low. |
| 31 | Habilita la escritura en el registro IR. |

Cuadro 1: Señales de control de la versión de Borda, Paula y Verdi, Lautaro

Penchulef, Daiana y Sanabra, Alejandro hicieron un trabajo similar al de Verdi, Lautaro y Borda, Paula microprogramado de sólo algunas instrucciones a la Máquina plus.

Con la primer versión funcional de la Máquina plus pero con el conjunto de instrucciones incompletas, tuvieron el trabajo de completarlas los estudiantes Fernandez, Teodoro y Alvarado, Nicole quienes se encargaron de cargar en la Unidad de Control (UC) todas las instrucciones de la Máquina plus y sus microprogramas.

En esta versión se agregó en la UC una memoria para usarla como tabla de decodificación de las instrucciones, que segun el código de operación de una instrucción ésta indica la dirección del microprograma en la memoria de control, también ampliaron la memoria de control y la memoria de direccionamiento explícito de 256 palabras a 1k con lo cual se necesitan 10 bits para direccionamiento. Hubo otro cambio en la memoria de direccionamiento explícito, pasó de representar 11 bits a 16, dos bits más de la dirección explícita, y tres bits más para uso futuro.

La tabla de instrucciones es de 256 palabras de 10 bits ya que esto alcanza y sobra para representar los códigos de operación de las instrucciones y los 10 bits para indicar la dirección del microprograma en la memoria de control.

En esta versión se habilitaron los 2 primeros bits de las señales de control y el resto sigue igual que antes.

| BIT | DESCRIPCIÓN |
|--------|-------------------------------------|
| 0 | Habilita la escritura en el flag Z. |
| 1 | Habilita la escritura en el flag C. |
| 2 al 6 | Sin definir, para uso futuro. |

Micro memoria de 1024 palabras de 48 bits

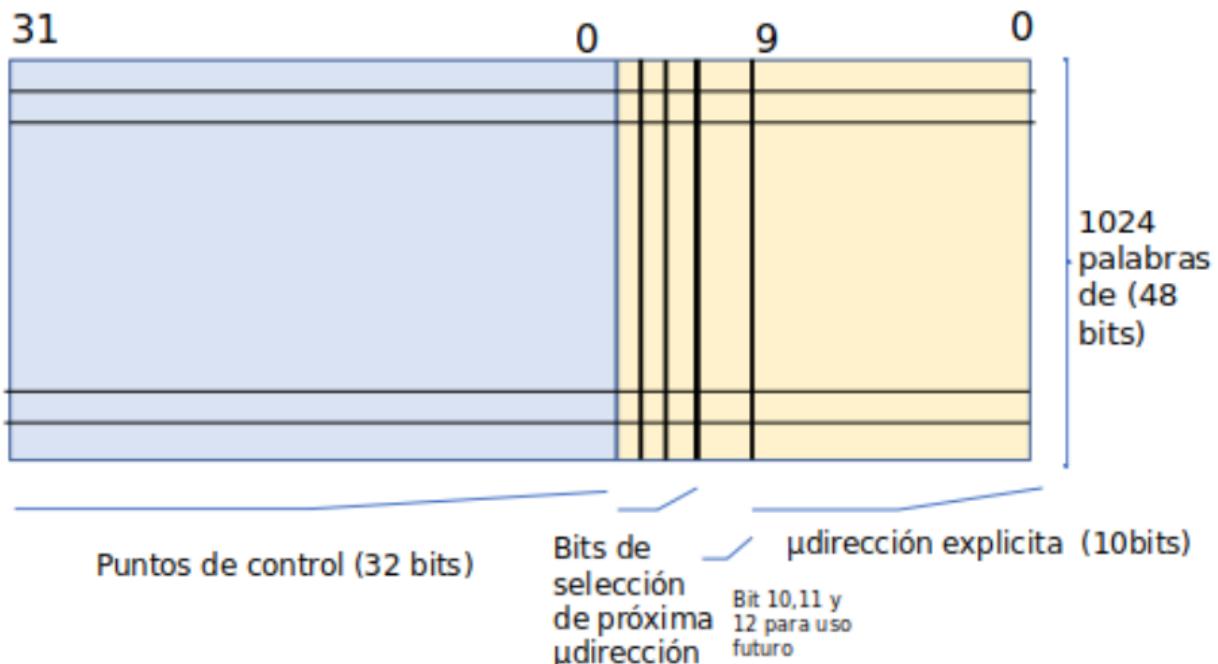


Figura 9: Micro Memoria mejorada por Alvarado y Fernandez

| Bit 15 | Bit 14 | Bit 13 | Próxima μdirección |
|--------|--------|--------|---------------------------------------|
| 0 | x | x | IR |
| 1 | 0 | 0 | $\mu\text{PC}+1$ (implícito) |
| 1 | 0 | 1 | Si $C=0$ $\mu\text{PC}+1$ (implícito) |
| | | | Si $C=1$ μdirección expl. |
| 1 | 1 | 0 | Si $Z=0$ $\mu\text{PC}+1$ (implícito) |
| | | | Si $Z=1$ μdirección expl. |
| 1 | 1 | 1 | μdirección explícita |

Figura 10: Descripción de los tres bits que indican la próxima dirección

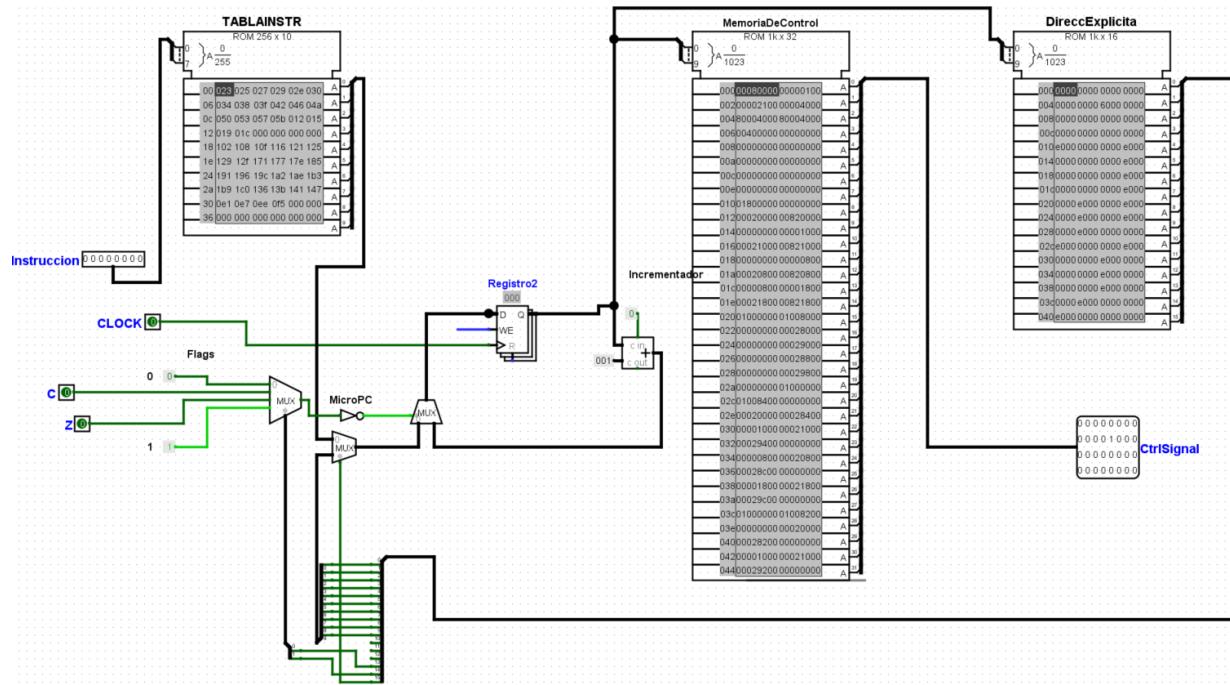


Figura 11: Unidad de Control mejorada de Alvarado y Fernandez

También mejoraron la ALU agregando la operación NOT y registros para las banderas de Carry y Zero de las operaciones aritméticas.

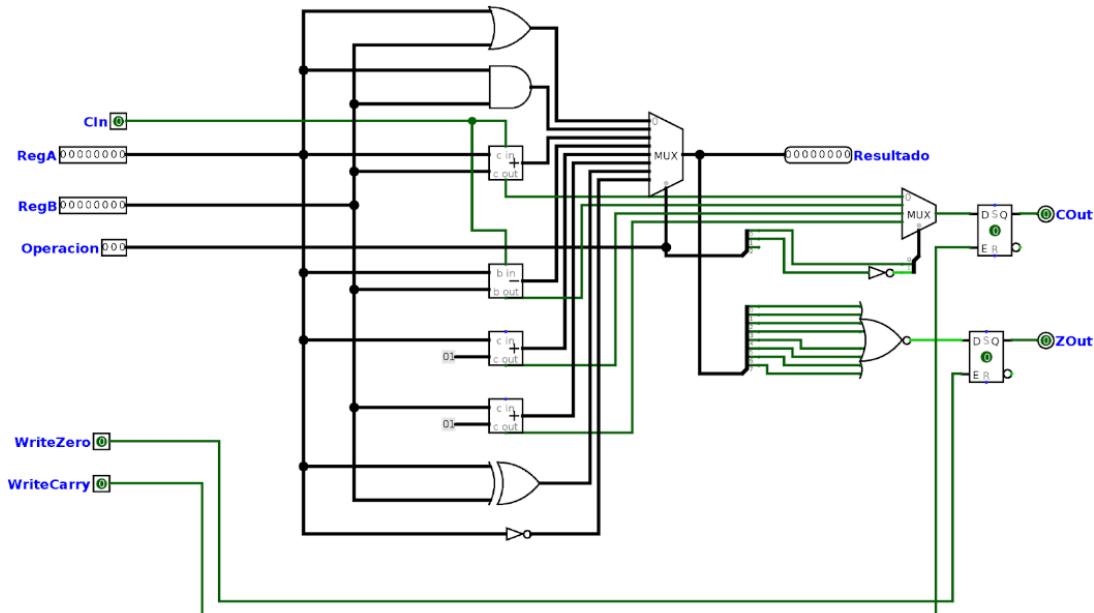


Figura 12: ALU mejorada por Alvarado y Fernandez.

Operaciones de la ALU con la nueva incorporación de la operación NOT:

| BIT | OPERACIÓN |
|-----|-----------|
| 000 | OR |
| 001 | AND |
| 010 | SUMA |
| 011 | RESTA |
| 100 | RegA + 1 |
| 101 | RegB + 1 |
| 110 | XOR |
| 111 | NOT |

Cuadro 2: Incorporación de la operación NOT en la Alu

En el estado del circuito de la Máquina plus de Alvarado, Nicole y Fernandez, Teodoro, el estudiante Herrera, Axel hizo un trabajo de reprogramación de las instrucciones con sus respectivos microprogramas.

El trabajo de Herrera, Axel pasó a Cotrena, Federico quien desarrolló e implementó en el circuito de la Máquina plus un sistema de fases de reloj con el objetivo de optimizar la ejecución de una instrucción, esta se ejecutará en menos ciclos de reloj.

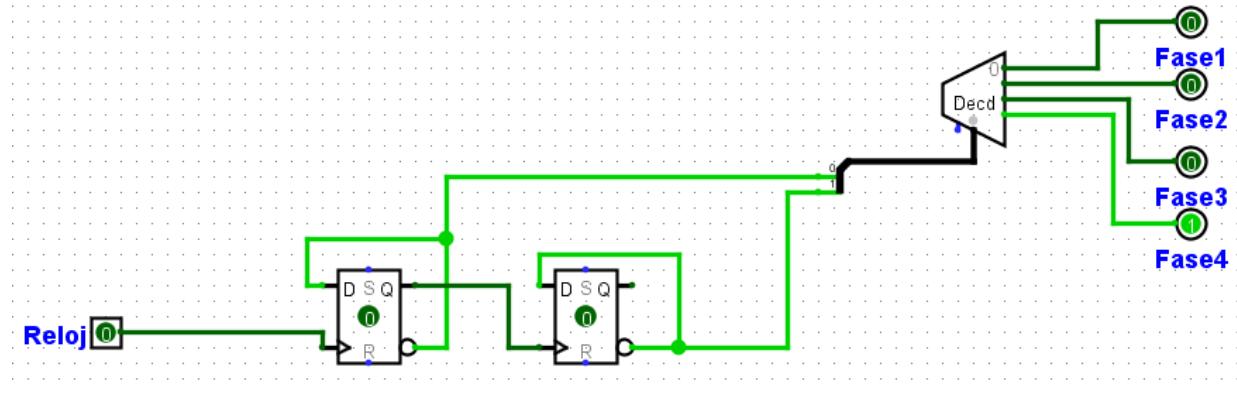


Figura 13: Vista del sistema de fases de reloj de Cotrena, Federico

La frecuencia del bloque de reloj está dada por el reloj principal que va activando las distintas fases.

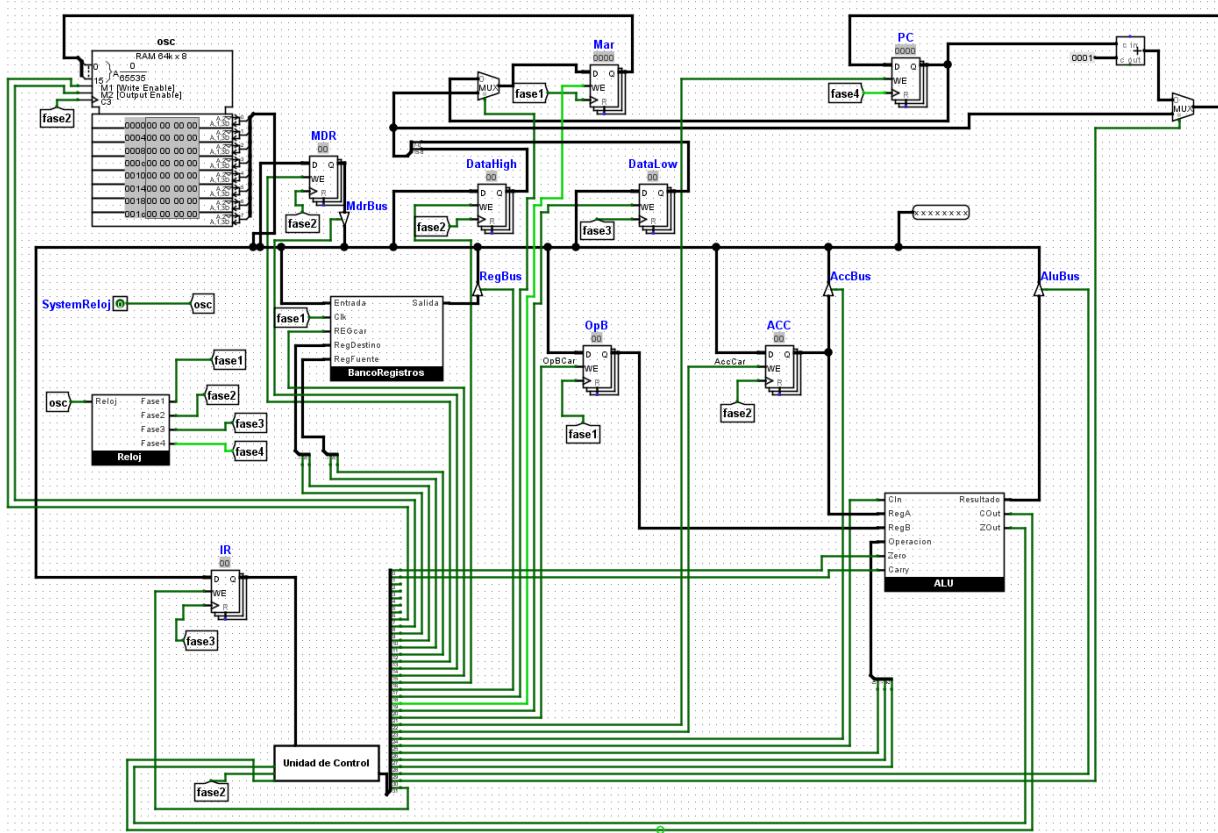


Figura 14: Vista general de la Máquina plus con el sistema de fases de reloj

Las fases de reloj en la máquina plus fueron asignadas de la siguiente manera:

- Fase 1: fue para aquellas transferencias entre registro que ocupen la utilización del registro MAR para escritura, como por ejemplo MAR <- PC, también para las distintas transferencias entre los bancos de registro.
- Fase 2: fue para las transferencia entre registros que ocupen la escritura del Registro MDR como por ejemplo MDR <- MP[00.00] , MDR <- Acc , como a su vez para el registro de Acc.
- Fase 3: fue utilizada para el registro de instrucciones y el registro DataLow este último para aprovechar el secuenciamiento del reloj.
- Fase 4: se utilizó para el contador de programa para solucionar el problema de inestabilidad en los distintos registros y también en el bus de datos que ocasionó la implementación de las fases de reloj.

4. Unidad de Control Microprogramada

La idea principal de una unidad de control microprogramada es la simplicidad conceptual, junto al hecho de que la información de control reside en una memoria, la que supone una fácil modificación y ampliación. Esta idea fue propuesta por Maurice Vincent Wilkes y otros científicos a principios de 1950.

Utilizando esta técnica se genera en la memoria de control una secuencia de microinstrucciones (microprograma), por cada macroinstrucción de una computadora. El microprograma es el conjunto de microinstrucciones que conforman el cronograma de una instrucción. Ejecutar un microprograma consiste en ir leyendo cada una de las microinstrucciones que lo componen, enviando información leída a los elementos del computador como señales de control. La cadencia de lectura está controlada por el reloj principal de la máquina.

La microprogramación permite construir computadoras capaces de soportar varios conjuntos de instrucciones, con tan solo cambiar el contenido de la memoria de control que está en el interior de la unidad de control. Aunque sea un poco más lenta que una unidad de control cableada, es menos costosa y tiene menos posibilidades de error, frente al complejo circuito lógico del tipo cableado.

La unidad de control microprogramada puede tener dos tipos de secuenciamiento, explícito, consiste en señalar en cada microinstrucción la dirección de la siguiente. El encadenamiento con una nueva instrucción se resuelve mediante una señal de control que toma como dirección de la siguiente microinstrucción el nuevo código de operación. También existen dos tipos de microinstrucciones, verticales y horizontales. Se llama microprogramación horizontal cuando las microinstrucciones no están codificadas. Por el contrario, recibe el nombre de microprogramación vertical cuando las microinstrucciones están muy codificadas. Según el grado de codificación, existen microprogramas más o menos horizontales o verticales.

5. Descripción del Circuito

5.1. Unidad de Control

La unidad de control es uno de los tres bloques fundamentales de la computadora, mantiene el control general y el envío de información a todos los elementos mediante buses (dirección, datos y control).

En más detalles esta unidad es la encargada de interpretar, controlar la ejecución de las instrucciones recibidas desde la memoria principal y resolver situaciones anómalas o de conflicto que pueden ocurrir en la máquina.

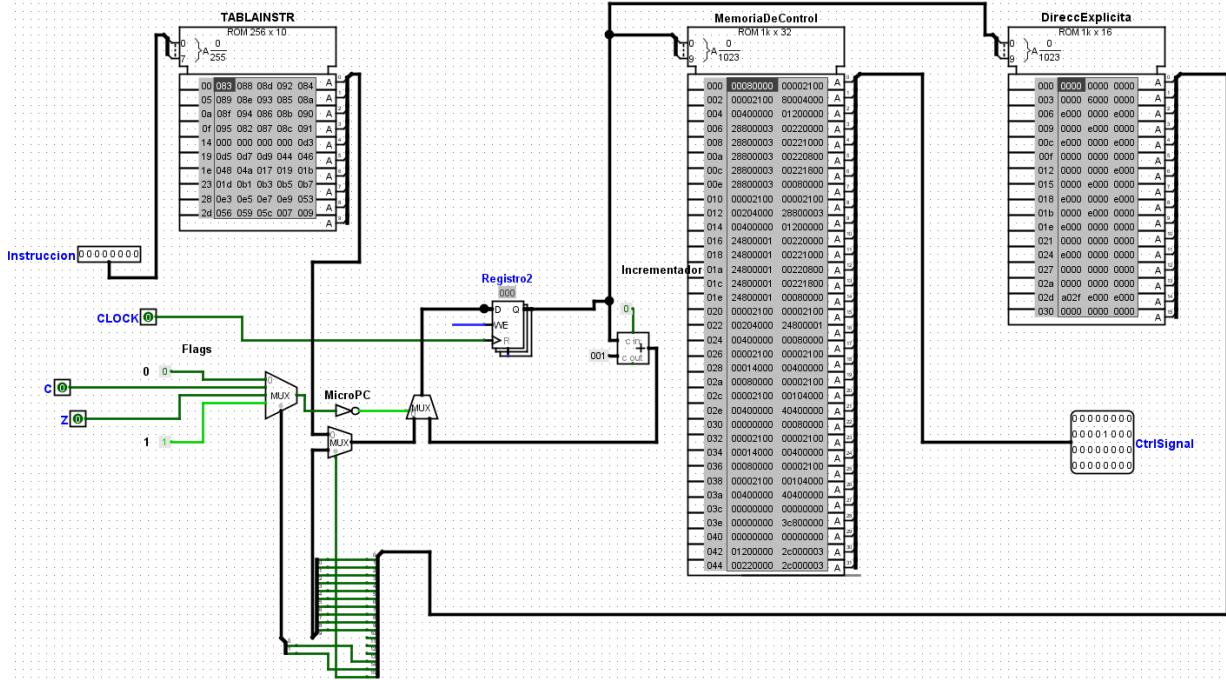


Figura 15: Unidad de Control de la Máquina plus

5.2. Tabla de Instrucciones

En la unidad de control la tabla de instrucciones es una memoria ROM que indica según el código de operación de una instrucción proveniente de un programa en memoria principal, la dirección en la memoria de control en donde comienza su microprograma. Se redirecciona con 8 bits, y tiene 10 bits para los datos de cada dirección.

| TABLA INSTR | | | | | | | | | |
|--------------------|-----|-----|-----|-----|-----|---|--|--|--|
| .ROM 256 x 10. | | | | | | | | | |
| 0 } A 0 7 } 255 | | | | | | | | | |
| 00 | 083 | 088 | 08d | 092 | 084 | A | | | |
| 05 | 089 | 08e | 093 | 085 | 08a | A | | | |
| 0a | 08f | 094 | 086 | 08b | 090 | A | | | |
| 0f | 095 | 082 | 087 | 08c | 091 | A | | | |
| 14 | 000 | 000 | 000 | 000 | 0d3 | A | | | |
| 19 | 0d5 | 0d7 | 0d9 | 044 | 046 | A | | | |
| 1e | 048 | 04a | 017 | 019 | 01b | A | | | |
| 23 | 01d | 0b1 | 0b3 | 0b5 | 0b7 | A | | | |
| 28 | 0e3 | 0e5 | 0e7 | 0e9 | 053 | A | | | |
| 2d | 056 | 059 | 05c | 007 | 009 | A | | | |

Figura 16: Vista parcial de la Tabla de Instrucciones

```

00 [083]088 08d 092 084 089 08e 093 085 08a 08f 094 086 08b 090 095
10 082 087 08c 091 000 000 000 000 0d3 0d5 0d7 0d9 044 046 048 04a
20 017 019 01b 01d 0b1 0b3 0b5 0b7 0e3 0e5 0e7 0e9 053 056 059 05c
30 007 009 00b 00d 000 000 000 000 000 000 000 000 000 000 000 000
40 07e 07f 080 081 07d 005 0d1 042 015 0af 0e1 052 000 000 000 000
50 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
60 09b 0a0 0a5 0aa 096 00f 0db 04c 01f 0b9 0eb 000 000 000 000 000
70 069 0bf 031 025 05f 000 000 000 000 000 000 000 000 000 000 000
80 03f 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
90 0cc 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
a0 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
b0 077 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
c0 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
d0 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
e0 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
f0 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

```

Figura 17: Vista completa de la Tabla de Instrucciones en Editor Hex de Logisim

Cada código de operación del conjunto de instrucciones implementado en la Máquina plus está mapeado directamente con la dirección de esta memoria, esto quiere decir que si el código de operación de la instrucción por ejemplo es 1C, debemos ver esa dirección en la tabla de instrucciones para conocer la dirección de su microprograma, 1C es la instrucción CMP B y la dirección de su microprograma en la memoria de control es 44.

5.2.1. Descripción del Contenido

| OPERACIÓN | CÓDIGO DE OPERACIÓN/DIRECCIÓN | DIRECCIÓN DEL MICROPROGRAMA |
|-----------|-------------------------------|-----------------------------|
| MOV B, B | 00 | 83 |
| MOV C, B | 01 | 88 |
| MOV D, B | 02 | 8d |
| MOV E, B | 03 | 92 |
| MOV B, C | 04 | 84 |
| MOV C, C | 05 | 89 |
| MOV D, C | 06 | 8e |
| MOV E, C | 07 | 93 |
| MOV B, D | 08 | 85 |
| MOV C, D | 09 | 8a |
| MOV D, D | 0a | 8f |
| MOV E, D | 0b | 94 |
| MOV B, E | 0c | 86 |
| MOV C, E | 0d | 8b |
| MOV D, E | 0e | 90 |
| MOV E, E | 0f | 95 |
| MOV B, A | 10 | 82 |
| MOV C, A | 11 | 87 |
| MOV D, A | 12 | 8c |
| MOV E, A | 13 | 91 |
| SUB B | 18 | d3 |
| SUB C | 19 | d5 |
| SUB D | 1a | d7 |
| SUB E | 1b | d9 |
| CMP B | 1c | 44 |
| CMP C | 1d | 46 |
| CMP D | 1e | 48 |
| CMP E | 1f | 4a |
| ANA B | 20 | 17 |
| ANA C | 21 | 19 |
| ANA D | 22 | 1b |
| ANA E | 23 | 1d |
| ORA B | 24 | b1 |
| ORA C | 25 | b3 |
| ORA D | 26 | b5 |
| ORA E | 27 | b7 |

Cuadro 3: Tabla de instrucciones 1/3

| OPERACIÓN | CÓDIGO DE OPERACIÓN/DIRECCIÓN | DIRECCIÓN DEL MICROPROGRAMA |
|------------|-------------------------------|-----------------------------|
| XRA B | 28 | e3 |
| XRA C | 29 | e5 |
| XRA D | 2a | e7 |
| XRA E | 2b | e9 |
| INR B | 2c | 53 |
| INR C | 2d | 56 |
| INR D | 2e | 59 |
| INR E | 2f | 5c |
| ADD B | 30 | 07 |
| ADD C | 31 | 09 |
| ADD D | 32 | 0b |
| ADD E | 33 | 0d |
| MOV A, B | 40 | 7e |
| MOV A, C | 41 | 7f |
| MOV A, D | 42 | 80 |
| MOV A, E | 43 | 81 |
| MOV A, A | 44 | 7d |
| ADD A | 45 | 05 |
| SUB A | 46 | d1 |
| CMP A | 47 | 42 |
| ANA A | 48 | 15 |
| ORA A | 49 | af |
| XRA A | 4a | e1 |
| INR A | 4b | 52 |
| MVI inm, B | 60 | 9b |
| MVI inm, C | 61 | a0 |
| MVI inm, D | 62 | a5 |
| MVI inm, E | 63 | aa |
| MVI inm, A | 64 | 96 |
| ADI inm | 65 | 0f |
| SUI inm | 66 | db |
| CPI inm | 67 | 4c |
| ANI inm | 68 | 1f |
| ORI inm | 69 | b9 |
| XRI inm | 6a | eb |

Cuadro 4: Tabla de instrucciones 2/3

| OPERACIÓN | CÓDIGO DE OPERACIÓN/DIRECCIÓN | DIRECCIÓN DEL MICROPROGRAMA |
|-----------|-------------------------------|-----------------------------|
| LDA dir | 70 | 69 |
| STA dir | 71 | bf |
| BEQ dir | 72 | 31 |
| BC dir | 73 | 25 |
| JMP dir | 74 | 5f |
| CMA | 80 | 3f |
| STAX | 90 | cc |
| LDAX | b0 | 77 |

Cuadro 5: Tabla de instrucciones 3/3

5.3. Memoria de Control

La memoria de control contiene el microprograma de cada instrucción de la Máquina plus.

5.3.1. Señales de Control

En la memoria de control están grabados los códigos de las señales que controlan cada uno de los pasos elementales en que se descompone una instrucción y que reciben la denominación de microinstrucciones, y este conjunto es el microprograma de una instrucción.

En la Máquina plus cada microinstrucción tiene asociada 32 señales de control, que se leen ordenadamente en binario o hexadecimal y que según la posición maneja una señal particular.

Estas son las diferentes señales de control:

| BIT | DESCRIPCIÓN |
|----------|---|
| 0 | Habilita la escritura en el flag Z. |
| 1 | Habilita la escritura en el flag C. |
| 2 al 6 | Sin definir, para uso futuro. |
| 7 | Habilita la escritura de datos en la memoria principal. |
| 8 | Habilita la salida o lectura de datos desde la memoria principal. |
| 9 | Selecciona el registro destino del banco de registros, pone en 1 el bit 1. |
| 10 | Selecciona el registro destino del banco de registros, pone en 1 el bit 0. |
| 11 | Selecciona el registro fuente del banco de registros, pone en 1 el bit 1. |
| 12 | Selecciona el registro fuente del banco de registros, pone en 1 el bit 0. |
| 13 | Habilita la escritura en el registro MDR. |
| 14 | Habilita la salida del registro MDR hacia el bus. |
| 15 | Habilita la carga de registros en el banco de registros. |
| 16 | Habilita la escritura en el registro DATAHIGH. |
| 17 | Habilita la salida del banco de registros hacia el bus. |
| 18 | Selecciona el contenido de los registros DATAHIGH y DATALOW o del registro PC, para entrada al registro MAR. |
| 19 | Habilita la escritura en el registro MAR. |
| 20 | Habilita la escritura en el registro temporal DATALOW. |
| 21 | Habilita la escritura en el registro OpB (operando B). |
| 22 | Habilita la escritura en el registro PC (contador de programa). |
| 23 | Habilita la escritura en el registro ACC (acumulador). |
| 24 | Habilita la salida del registro ACC hacia el bus. |
| 25 | Indica si en la operación habrá carry. |
| 26 al 28 | Seleccionan el código de operación de la ALU. |
| 29 | Habilita la salida de la ALU hacia el bus. |
| 30 | Selecciona la siguiente dirección del registro PC, si es cero PC se incrementa y si es uno PC toma la dirección formada por los registros DATAHIGH y DATALOW. |
| 31 | Habilita la escritura en el registro IR. |

Cuadro 6: Señales de control

5.3.2. Descripción del Contenido

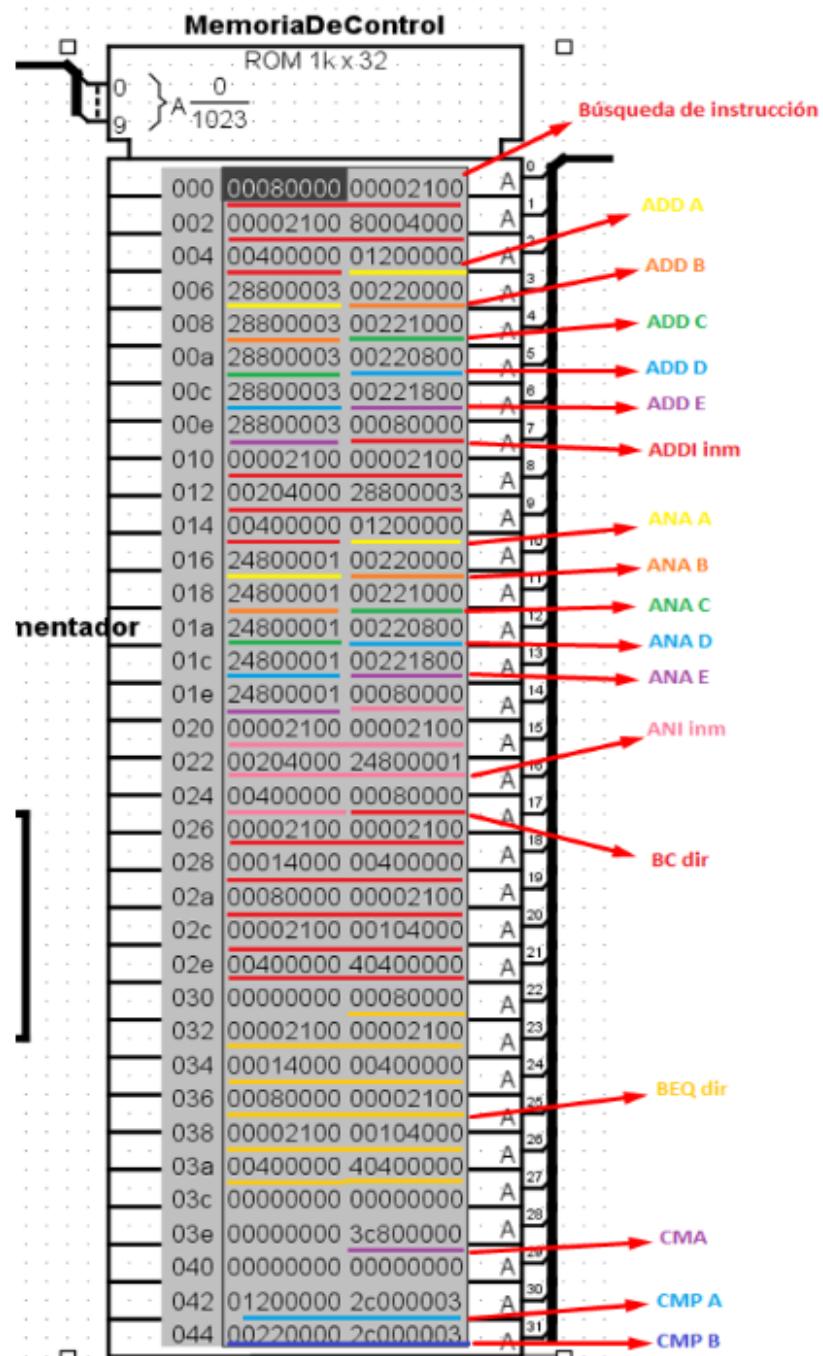


Figura 18: Descripción visual de la Memoria de Control. 1/4.

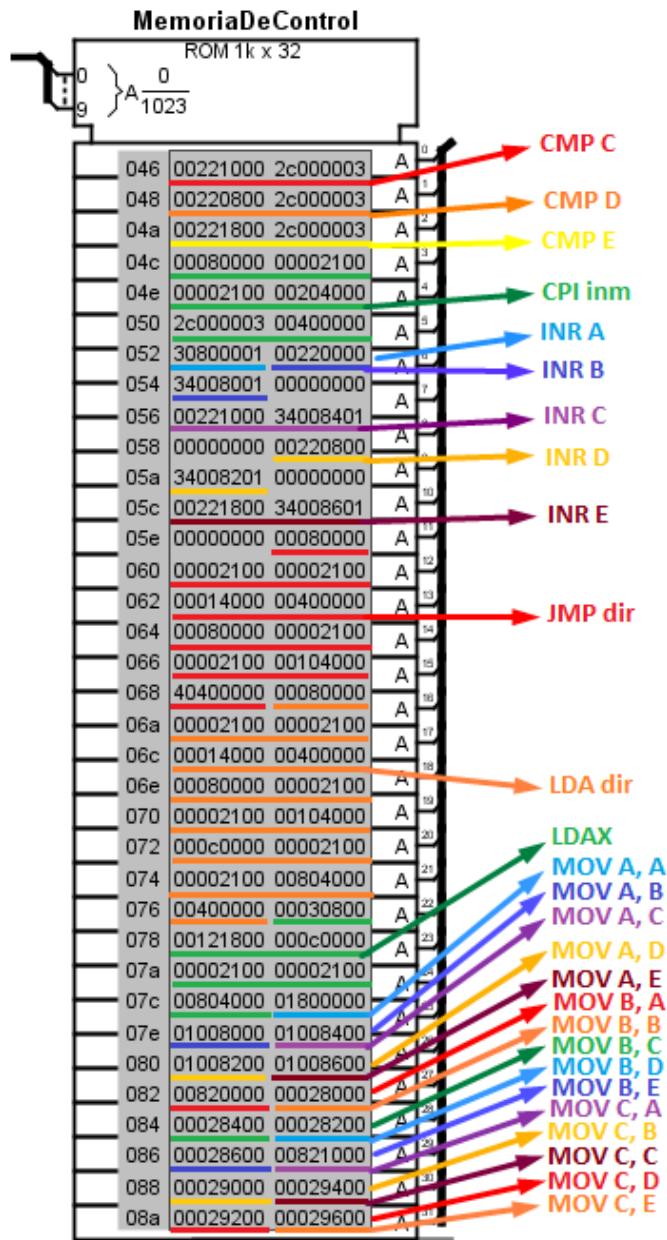


Figura 19: Descripción visual de la Memoria de Control. 2/4.

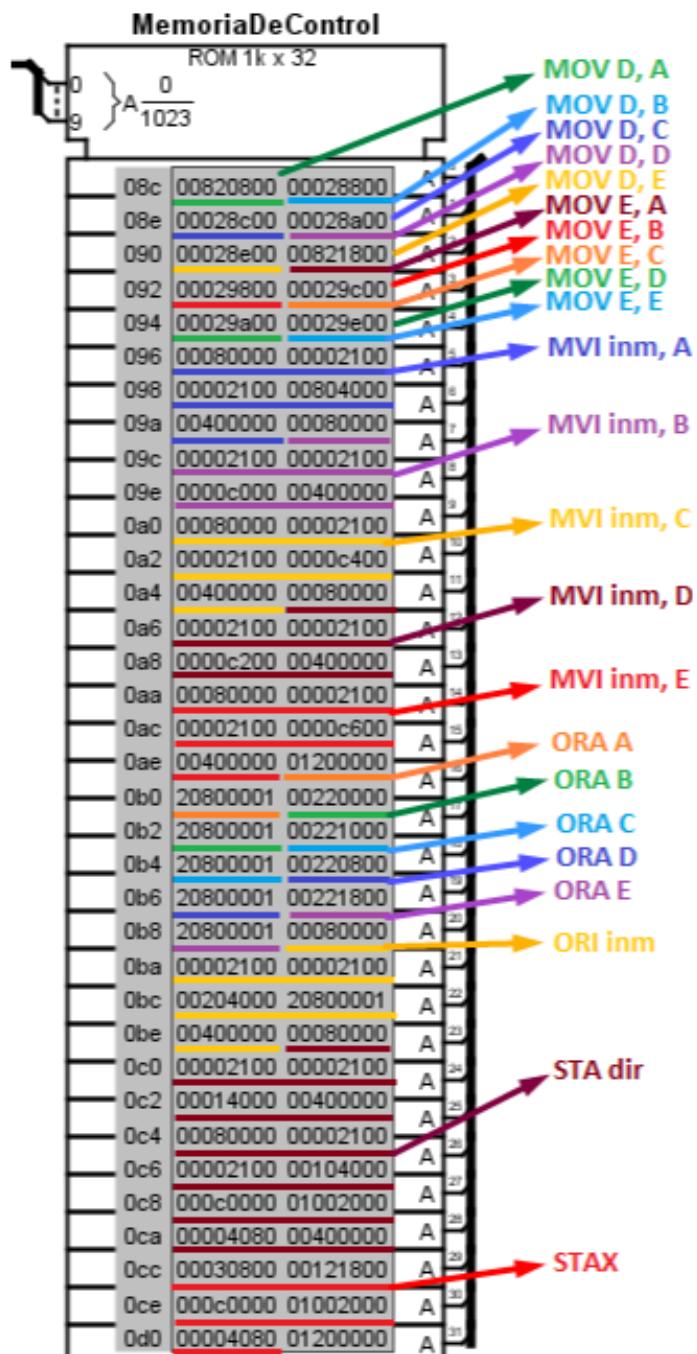


Figura 20: Descripción visual de la Memoria de Control. 3/4.

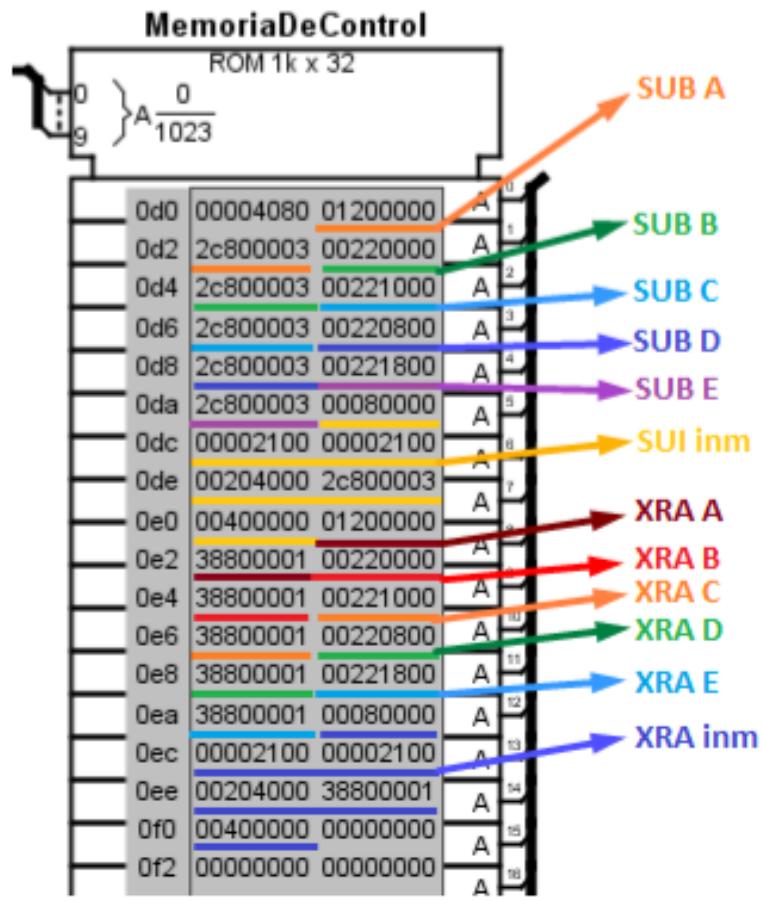


Figura 21: Descripción visual de la Memoria de Control. 4/4.

5.3.3. Descripción detallada de la memoria de control

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|--|---|
| 00 | 04 | Búsqueda de una instrucción. Bloque común a todas las instrucciones. | <ol style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 80004000->bits 14,31 5. 400000->bit 22 |
| 05 | 06 | 45: ADD A | <ol style="list-style-type: none"> 1. 1200000->bits 24, 21 2. 28800003->bits 23, 27, 29, 0, 1 |
| 07 | 08 | 30: ADD B | <ol style="list-style-type: none"> 1. 220000->bits 21, 17 2. 28800003->bits 23, 27, 29, 0, 1 |
| 09 | 0a | 31: ADD C | <ol style="list-style-type: none"> 1. 221000->bits 21, 17, 12 2. 28800003->bits 23, 27, 29, 0, 1 |

Cuadro 7: Memoria de control detallada 1/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| 0b | 0c | 32: ADD D | <ul style="list-style-type: none"> 1. 220800->bits 21, 17, 11 2. 28800003->bits 23, 27, 29, 0, 1 |
| 0d | 0e | 33: ADD E | <ul style="list-style-type: none"> 1. 221800->bits 21, 17, 11, 12 2. 28800003->bits 23, 27, 29, 0, 1 |
| 0f | 14 | 65: ADDI inm | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 28800003->bits 23, 27, 29, 0, 1 6. 400000->bit 22 |
| 15 | 16 | 48: ANA A | <ul style="list-style-type: none"> 1. 1200000->bits 24, 21 2. 24800001->bits 23, 29, 26, 0 |

Cuadro 8: Memoria de control detallada 2/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| 17 | 18 | 20: ANA B | <ul style="list-style-type: none"> 1. 220000->bits 17, 21 2. 24800001->bits 23, 29, 26, 0 |
| 19 | 1a | 21: ANA C | <ul style="list-style-type: none"> 1. 221000->bits 12, 17, 21 2. 24800001->bits 23, 29, 26, 0 |
| 1b | 1c | 22: ANA D | <ul style="list-style-type: none"> 1. 220800->bits 11, 17, 21 2. 24800001->bits 23, 29, 26, 0 |
| 1d | 1e | 23: ANA E | <ul style="list-style-type: none"> 1. 221800->bits 11, 12, 17, 21 2. 24800001->bits 23, 29, 26, 0 |

Cuadro 9: Memoria de control detallada 3/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| 1f | 24 | 68: ANI inm | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 24800001->bits 23, 29, 26, 0 6. 400000->bit 22 |
| 25 | 30 | 73: BC dir | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 14000->bits 16, 14 5. 400000->bit 22 6. 80000->bit 19 7. 2100->bits 8, 13 8. 2100->bits 8, 13 9. 104000->bits 14, 20 10. 400000->bit 22 11. 40400000->bits 30, 22 |

Cuadro 10: Memoria de control detallada 4/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| 31 | 3b | 72: BEQ dir | <ol style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 14000->bits 16, 14 5. 400000->bit 22 6. 80000->bit 19 7. 2100->bits 8, 13 8. 2100->bits 8, 13 9. 104000->bits 14, 20 10. 400000->bit 22 11. 40400000->bits 30, 22 |
| 3f | 3f | 80: CMA | <ol style="list-style-type: none"> 1. 3C800000->bits 26, 27, 28, 23, 29 |
| 42 | 43 | 47: CMP A | <ol style="list-style-type: none"> 1. 1200000->bits 24, 21 2. 2C000003->bits 26, 27, 29, 0, 1 |

Cuadro 11: Memoria de control detallada 5/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| 44 | 45 | 1C: CMP B | <ul style="list-style-type: none"> 1. 220000->bits 17, 21 2. 2C000003->bits 26, 27, 29, 0, 1 |
| 46 | 47 | 1D: CMP C | <ul style="list-style-type: none"> 1. 221000->bits 17, 21, 12 2. 2C000003->bits 26, 27, 29, 0, 1 |
| 48 | 49 | 1E: CMP D | <ul style="list-style-type: none"> 1. 220800->bits 17, 21, 11 2. 2C000003->bits 26, 27, 29, 0, 1 |
| 4a | 4b | 1F: CMP E | <ul style="list-style-type: none"> 1. 221800->bits 17, 21, 11, 12 2. 2C000003->bits 26, 27, 29, 0, 1 |

Cuadro 12: Memoria de control detallada 6/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| 4c | 51 | 67: CPI inm | <ol style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 2C000003->bits 26, 27, 29, 0, 1 6. 400000->bit 22 |
| 52 | 52 | 4B: INR A | <ol style="list-style-type: none"> 1. 30800001->bits 28, 29, 23, 0 |
| 53 | 54 | 2C: INR B | <ol style="list-style-type: none"> 1. 220000 bits 21, 17 2. 34008001->bits 26, 28, 29, 0, 15 |
| 56 | 57 | 2D: INR C | <ol style="list-style-type: none"> 1. 221000->bits 21, 17, 12 2. 34008401->bits 26, 28, 29, 0, 10, 15 |

Cuadro 13: Memoria de control detallada 7/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| 59 | 5a | 2E: INR D | 1. 220800->bits 21,17, 11 2. 34008201->bits 26, 28, 29, 0, 9, 15 |
| 5c | 5d | 2F: INR E | 1. 221800->bits 21, 17, 11, 12 2. 34008601->bits 26, 28, 29, 0, 9, 10, 15 |
| 5f | 68 | 74: JMP dir | 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 14000->bits 16, 14 5. 400000->bit 22 6. 80000->bit 19 7. 2100->bits 8, 13 8. 2100->bits 8, 13 9. 104000->bits 14, 20 10. 40400000->bits 30, 22 |

Cuadro 14: Memoria de control detallada 8/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| 69 | 76 | 70: LDA dir | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 14000->bits 16, 14 5. 400000->bit 22 6. 80000->bit 19 7. 2100->bits 8, 13 8. 2100->bits 8, 13 9. 104000->bits 14, 20 10. C0000->bits 18, 19 11. 2100->bits 8, 13 12. 2100->bits 8, 13 13. 804000->bits 14, 23 14. 400000->bit 22 |
| 77 | 7c | B0: LDAX | <ul style="list-style-type: none"> 1. 30800->bits 11, 17, 16 2. 121800->bits 11, 12, 17, 20 3. C0000->bits 18, 19 4. 2100->bits 8, 13 5. 2100->bits 8, 13 6. 804000->bits 14, 23 |

Cuadro 15: Memoria de control detallada 9/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|-----------------------------------|
| 7d | 7d | 44: MOV A, A | 1. 1800000->bits 23, 24 |
| 7e | 7e | 40: MOV A, B | 1. 1008000->bits 15, 24 |
| 7f | 7f | 41: MOV A, C | 1. 1008400->bits 15, 24, 10 |
| 80 | 80 | 42: MOV A, D | 1. 1008200->bits 15, 24, 9 |
| 81 | 81 | 43: MOV A, E | 1. 1008600->bits 15, 24, 9, 10 |
| 82 | 82 | 10: MOV B, A | 1. 820000->bits 23, 17 |
| 83 | 83 | 00: MOV B, B | 1. 28000->bits 15, 17 |
| 84 | 84 | 04: MOV B, C | 1. 28400->bits 10, 15, 17 |
| 85 | 85 | 08: MOV B, D | 1. 28200->bits 9, 15, 17 |
| 86 | 86 | 0C: MOV B, E | 1. 28600->bits 9, 10, 15, 17 |

Cuadro 16: Memoria de control detallada 10/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|----------------------------------|
| 87 | 87 | 11: MOV C, A | 1. 821000->bits 23, 17, 12 |
| 88 | 88 | 01: MOV C, B | 1. 29000->bits 15, 17, 12 |
| 89 | 89 | 05: MOV C, C | 1. 29400->bits 12, 10, 17, 15 |
| 8a | 8a | 09: MOV C, D | 1. 29200->bits 9, 15, 17, 12 |
| 8b | 8b | 0D: MOV C, E | 1. 29600->bits 9, 10, 12, 15, 17 |
| 8c | 8c | 12: MOV D, A | 1. 820800->bits 23, 11, 17 |
| 8d | 8d | 02: MOV D, B | 1. 28800->bits 15, 17, 11 |
| 8e | 8e | 06: MOV D, C | 1. 28C00->bits 10, 15, 17, 11 |
| 8f | 8f | 0A: MOV D, D | 1. 28A00->bits 9, 11, 15, 17 |
| 90 | 90 | 0E: MOV D, E | 1. 28E00->bits 9, 10, 11, 15, 17 |

Cuadro 17: Memoria de control detallada 11/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| 91 | 91 | 13: MOV E, A | 1. 821800->bits 23, 11, 12, 17 |
| 92 | 92 | 03: MOV E, B | 1. 29800->bits 15, 17, 11, 12 |
| 93 | 93 | 07: MOV E, C | 1. 29C00->bits 10, 11, 12, 15, 17 |
| 94 | 94 | 0B: MOV E, D | 1. 29A00->bits 9, 11, 12, 15, 17 |
| 95 | 95 | 0F: MOV E, E | 1. 29E00->bits 11, 12, 9, 10, 15, 17 |
| 96 | 9a | 64: MVI inm, A | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 804000->bits 14, 23 5. 400000->bit 22 |

Cuadro 18: Memoria de control detallada 12/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| 9b | 9f | 60: MVI inm, B | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. C000->bits 14, 15 5. 400000->bit 22 |
| a0 | a4 | 61: MVI inm, C | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. C400->bits 14, 15, 10 5. 400000->bit 22 |
| a5 | a9 | 62: MVI inm, D | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. C200->bits 14, 15, 9 5. 400000->bit 22 |

Cuadro 19: Memoria de control detallada 13/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| aa | ae | 63: MVI imm, E | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. C600->bits 14, 15, 9, 10 5. 400000->bit 22 |
| af | bo | 49: ORA A | <ul style="list-style-type: none"> 1. 1200000->bits 24, 21 2. 20800001->bits 23, 29, 0 |
| b1 | b2 | 24: ORA B | <ul style="list-style-type: none"> 1. 220000->bits 17, 21 2. 20800001->bits 23, 29, 0 |
| b3 | b4 | 25: ORA C | <ul style="list-style-type: none"> 1. 221000->bits 17, 21, 12 2. 20800001->bits 23, 29, 0 |

Cuadro 20: Memoria de control detallada 14/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| b5 | b6 | 26: ORA D | <ul style="list-style-type: none"> 1. 220800->bits 17, 21, 11 2. 20800001->bits 23, 29, 0 |
| b7 | b8 | 27: ORA E | <ul style="list-style-type: none"> 1. 221800->bits 17, 21, 11, 12 2. 20800001->bits 23, 29, 0 |
| b9 | be | 69: ORI imm | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 20800001->bits 23, 29, 0 6. 400000->bit 22 |

Cuadro 21: Memoria de control detallada 15/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| bf | cb | 71: STA dir | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 14000->bits 16, 14 5. 400000->bit 22 6. 80000->bit 19 7. 2100->bits 8, 13 8. 2100->bits 8, 13 9. 104000->bits 14, 20 10. C0000->bits 18, 19 11. 1002000->bits 13, 24 12. 4080->bits 14, 7 13. 400000->bit 22 |
| cc | d0 | 90: STAX | <ul style="list-style-type: none"> 1. 30800->bits 11, 17, 16 2. 121800->bits 11, 12, 17, 20 3. C0000->bits 18, 19 4. 1002000->bits 24, 13 5. 4080->bits 14, 7 |

Cuadro 22: Memoria de control detallada 16/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| d1 | d2 | 46: SUB A | 1. 1200000->bits 24, 21 2. 2C800003->bits 23, 29, 26, 27, 0, 1 |
| d3 | d4 | 18: SUB B | 1. 220000->bits 21, 17 2. 2C800003->bits 23, 29, 26, 27, 0, 1 |
| d5 | d6 | 19: SUB C | 1. 221000->bits 21, 17, 12 2. 2C800003->bits 23, 29, 26, 27, 0, 1 |
| d7 | d8 | 1A: SUB D | 1. 220800->bits 21, 17, 11 2. 2C800003->bits 23, 29, 26, 27, 0, 1 |

Cuadro 23: Memoria de control detallada 17/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|---|
| d9 | da | 1B: SUB E | <ul style="list-style-type: none"> 1. 221800->bits 21, 17, 11, 12 2. 2C800003->bits 23, 29, 26, 27, 0, 1 |
| db | e0 | 66: SUI imm | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 2C800003->bits 23, 29, 26, 27, 0, 1 6. 400000->bit 22 |
| e1 | e2 | 4A: XRA A | <ul style="list-style-type: none"> 1. 1200000->bits 24, 21 2. 38800001->bits 0, 27, 28, 29, 23 |
| e3 | e4 | 28: XRA B | <ul style="list-style-type: none"> 1. 220000->bits 21, 17 2. 38800001->bits 0, 27, 28, 29, 23 |

Cuadro 24: Memoria de control detallada 18/19

| DESDE (DIR) | HASTA (DIR) | CÓDIGO E INSTRUCCIÓN | SEÑALES DE CONTROL |
|----------------|----------------|----------------------|--|
| e5 | e6 | 29: XRA C | <ul style="list-style-type: none"> 1. 221000->bits 21, 17, 12 2. 38800001->bits 0, 27, 28, 29, 23 |
| e7 | e8 | 2A: XRA D | <ul style="list-style-type: none"> 1. 220800->bits 21, 17, 11 2. 38800001->bits 0, 27, 28, 29, 23 |
| e9 | ea | 2B: XRA E | <ul style="list-style-type: none"> 1. 221800->bits 21, 17, 11, 12 2. 38800001->bits 0, 27, 28, 29, 23 |
| eb | f0 | 6A: XRI inm | <ul style="list-style-type: none"> 1. 80000->bit 19 2. 2100->bits 8, 13 3. 2100->bits 8, 13 4. 204000->bits 21, 14 5. 38800001->bits 0, 27, 28, 29, 23 6. 400000->bit 22 |

Cuadro 25: Memoria de control detallada 19/19

5.4. Memoria de Direccionamiento Explícito

La memoria de direccionamiento explícito determina lo que se debe ejecutar después de la ejecución de una microinstrucción, la implementación es de secuenciamiento implícito, es decir que todas las microinstrucciones están ordenadas por cada microprograma.

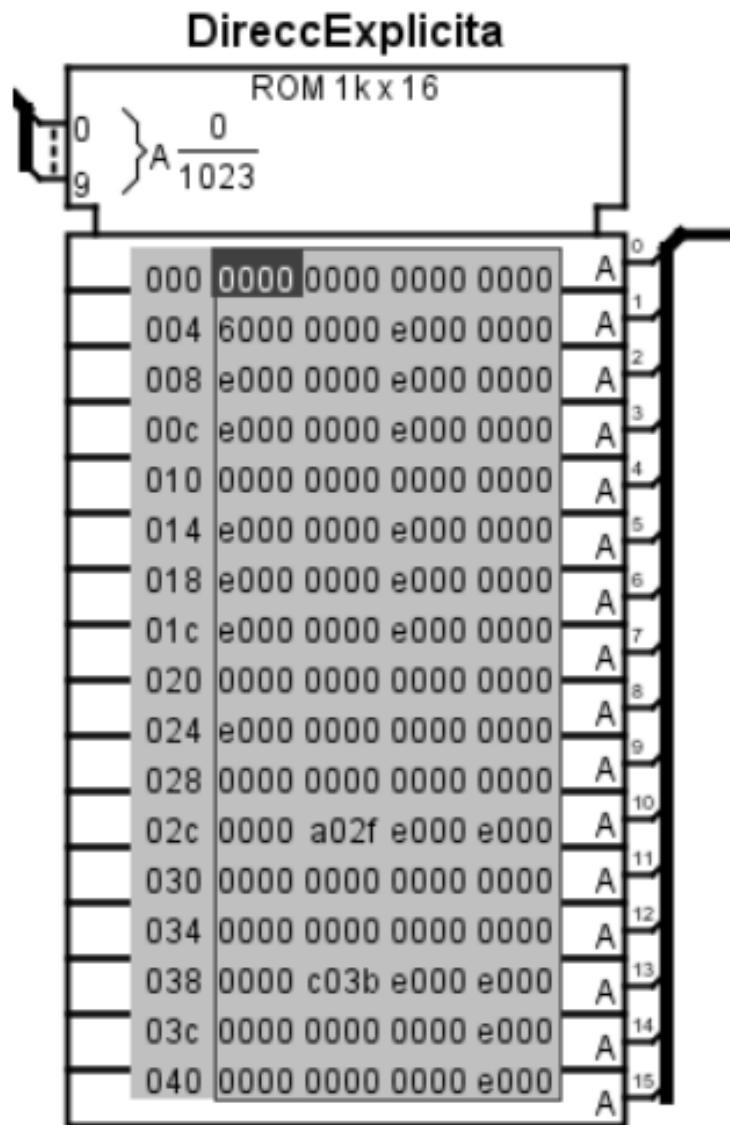


Figura 22: Memoria de Direccionamiento Explícito

| CÓDIGO | DESCRIPCIÓN |
|--------|--|
| 0000 | Avanza a la siguiente microinstrucción. |
| 6000 | Luego de la búsqueda de una instrucción elige la próxima instrucción de IR para ser decodificada. |
| E000 | Vuelve a la búsqueda y decodificación de la siguiente instrucción. Todas las instrucciones terminan con esta dirección explícita. |
| AXXX | Si C = 1 salta a la dirección especificada por los 10 primeros bits de la dirección explícita. Si C = 0 avanza hacia la próxima microinstrucción. |
| CXXX | Si Z = 1 salta a la dirección especificada por los 10 primeros bits de la dirección explícita. Si Z = 0 avanza hacia la próxima microinstrucción. |

Cuadro 26: Descripción de los códigos de memoria de direccionamiento explícito

6. ALU

La ALU (Unidad Aritmética Lógica) es una de las tres unidades principales de una computadora, encargada de realizar las operaciones aritméticas y lógicas.

Esta recibe la operación a realizar y los operandos involucrados, se genera un resultado que luego puede ser almacenado, y también en ciertos casos se levantan ciertas banderas para dar aviso a la unidad de control del estado de la operación.

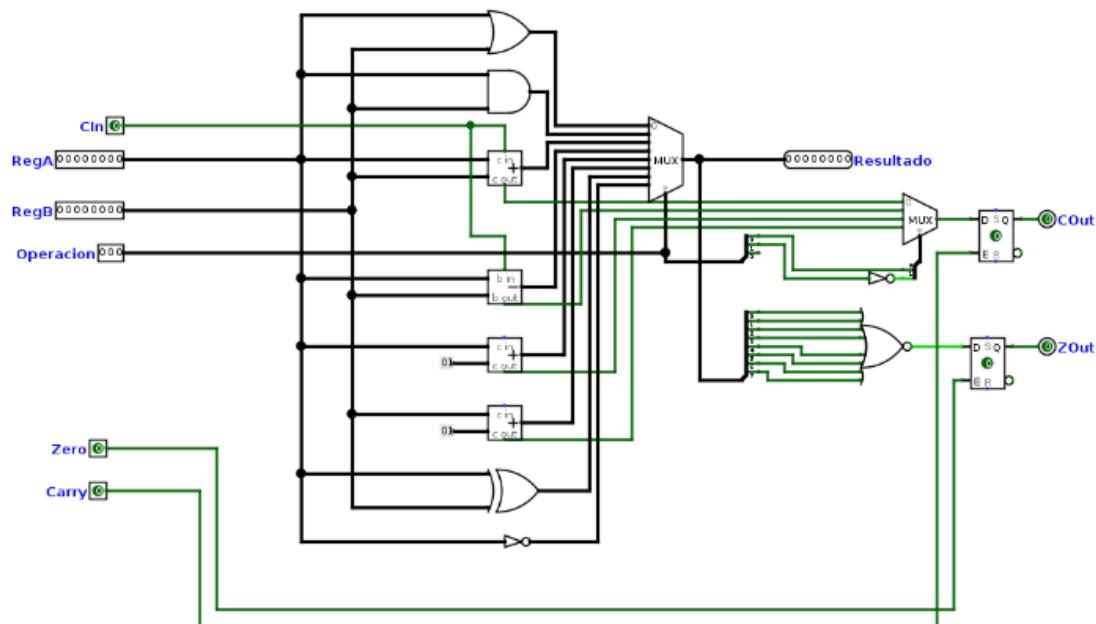


Figura 23: Unidad Aritmética Lógica (ALU)

Registro de banderas (flags): Los flags registran el estado del procesador después de una operación.

- CF (Carry). Indica acarreo en las operaciones aritméticas.
- ZF (Zero). Se activa cuando el resultado de una operación es cero.

6.1. Códigos de Operación de la ALU

| CÓDIGO | OPERACIÓN |
|--------|-----------|
| 000 | OR |
| 001 | AND |
| 010 | SUMA |
| 011 | RESTA |
| 100 | RegA + 1 |
| 101 | RegB + 1 |
| 110 | XOR |
| 111 | NOT |

Cuadro 27: Códigos de Operación de la ALU

7. Banco de Registros

El banco de registros es una serie de posiciones especiales de memoria, que permiten un acceso a operadores y lugares de almacenamiento de resultados mucho más veloz que si estuvieran en el sistema de memoria principal.

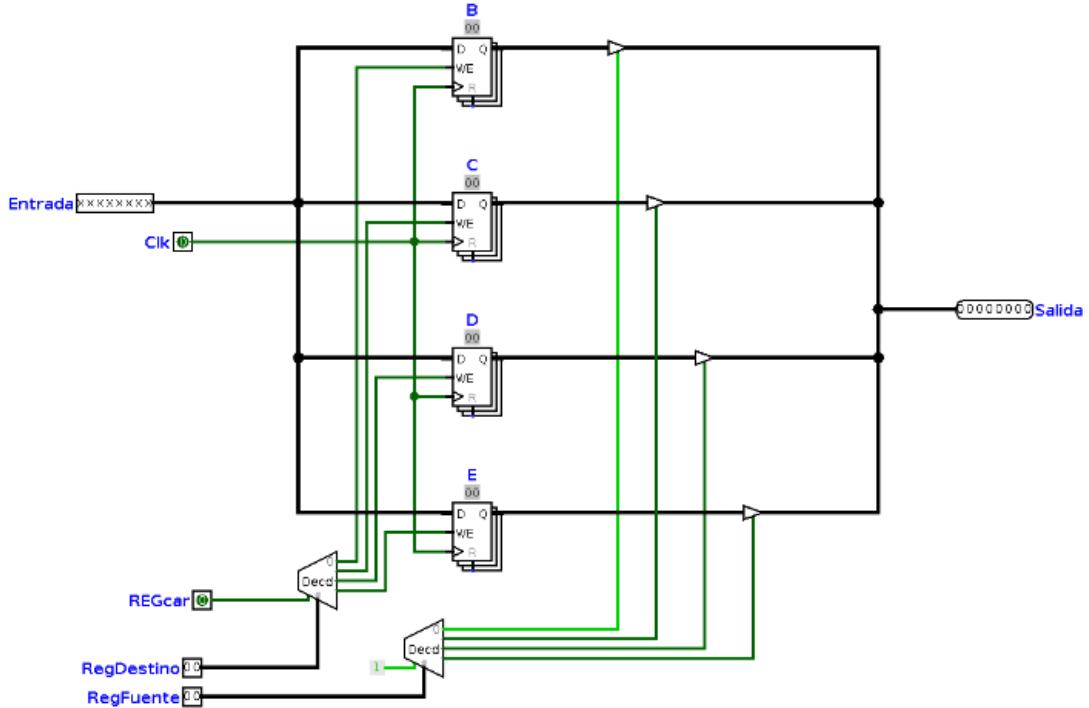


Figura 24: Banco de Registros

Registros Generales: Su función es el almacenamiento temporal de datos.

- AX (acumulador): Es utilizado en las instrucciones aritméticas.
- BX (base): Se usa generalmente para indicar un desplazamiento.
- CX (contador): Se utiliza en bucles.
- DX (datos): Se utiliza también en operaciones aritméticas.

8. IR

Registro de Instrucciones (RI): Almacena el código de operación de la instrucción que se está ejecutando.

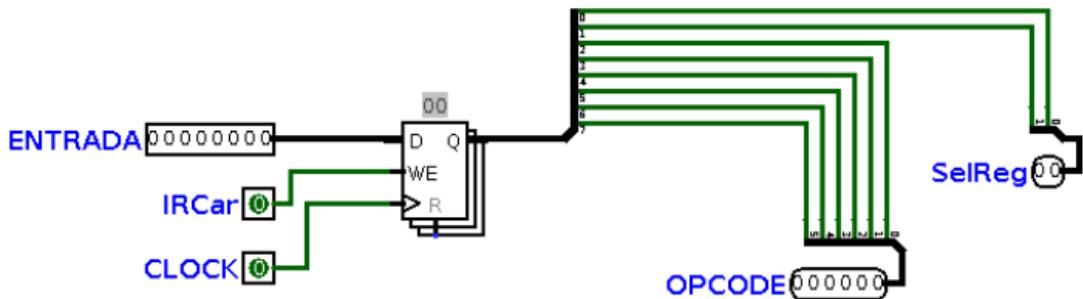
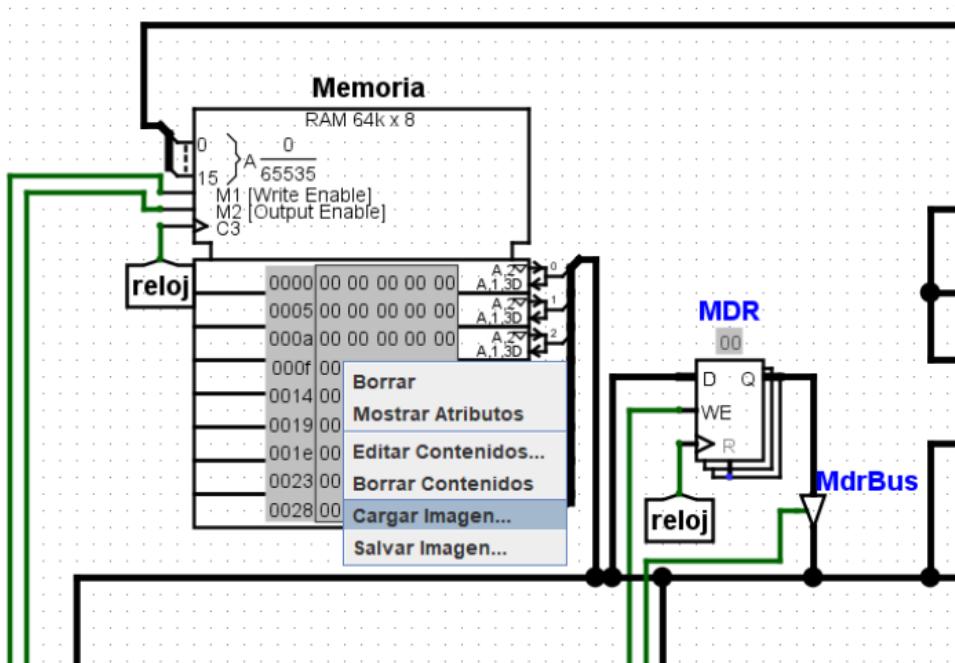


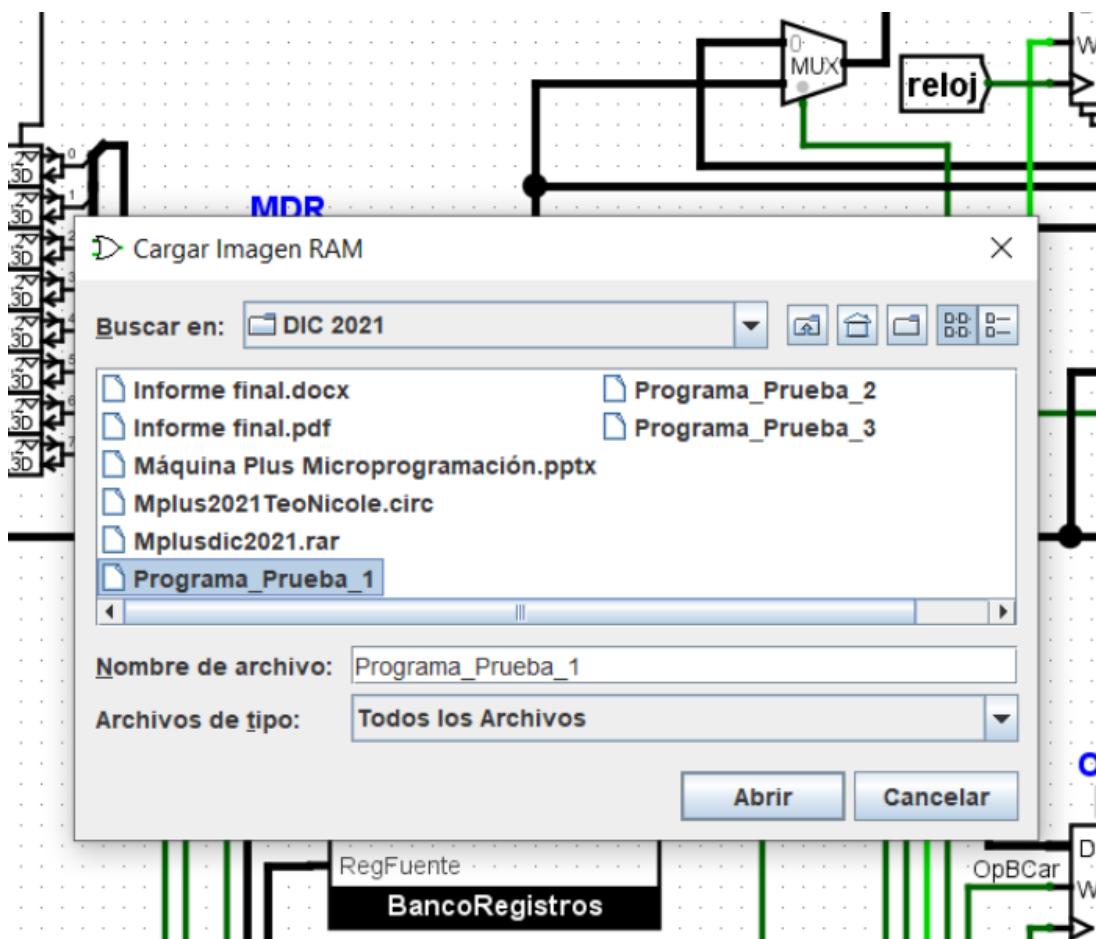
Figura 25: IR

9. Simulación de un programa

La simulación de un programa está hecha en el estado del circuito realizado por el estudiante Herrera, Axel.

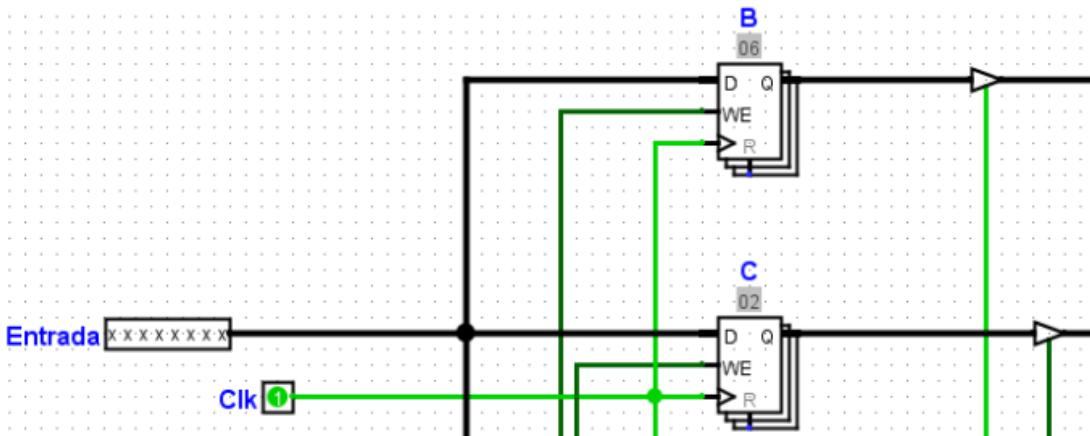


Cargamos un programa máquina en memoria principal para observar cómo es el flujo de la ejecución.



Elegimos el programa de prueba que queramos cargar. Programa de Prueba 1:

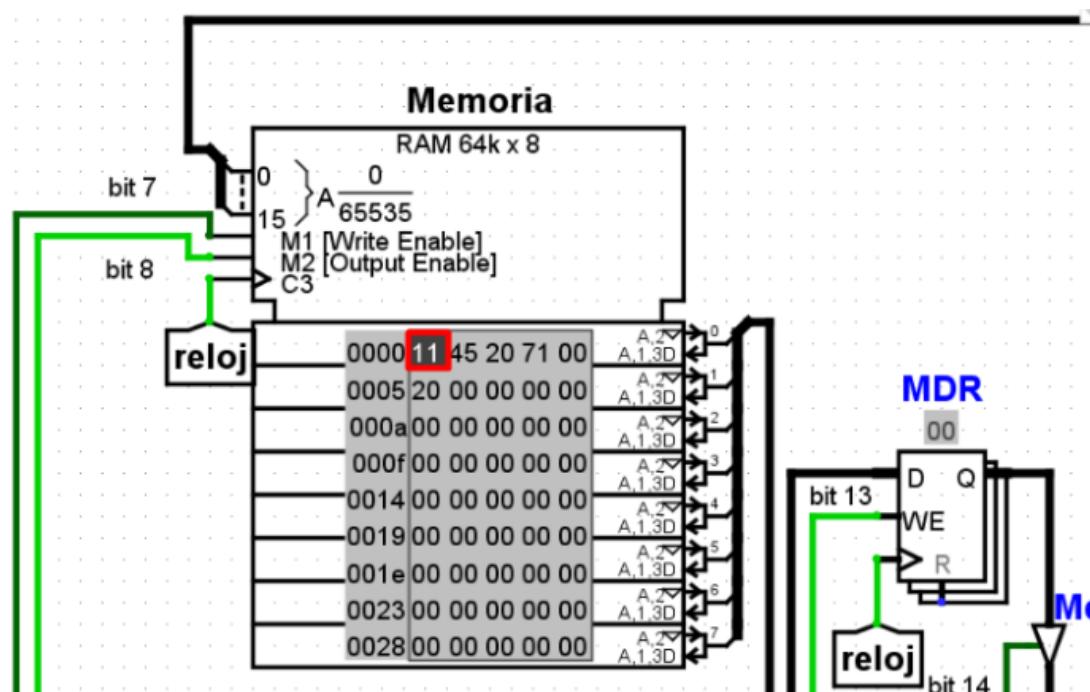
1. MOV C, A =>ACC <- C
2. ADD A =>ACC <- ACC + ACC
3. ANA B =>ACC <- ACC and B
4. STA 0020 =>MEM[0020] <- ACC



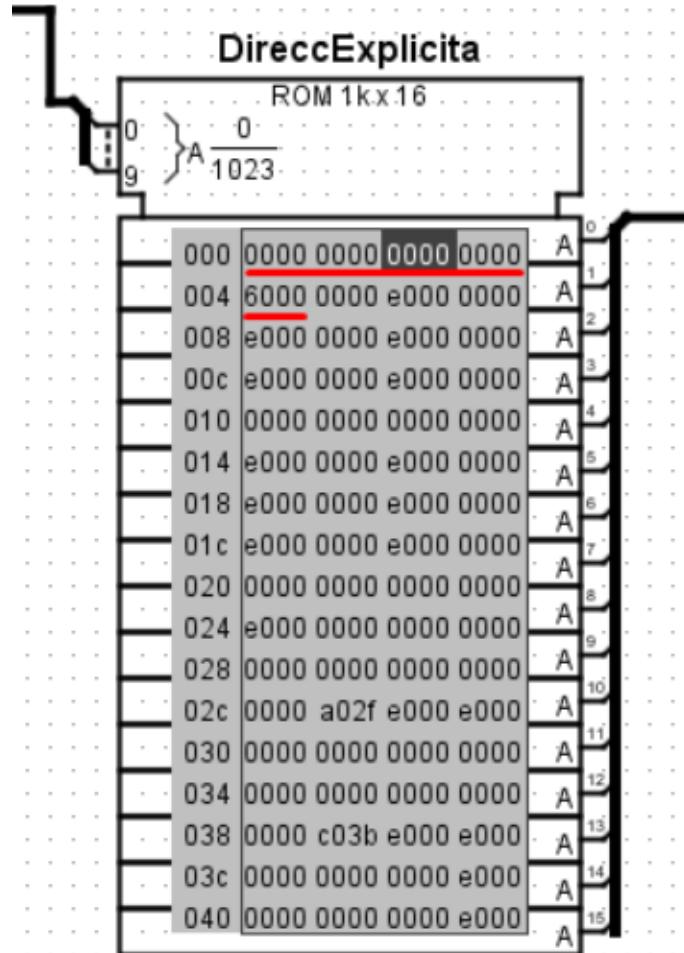
Cargamos los registros con los siguientes valores: B = 06 C = 02.

9.1. MOV C

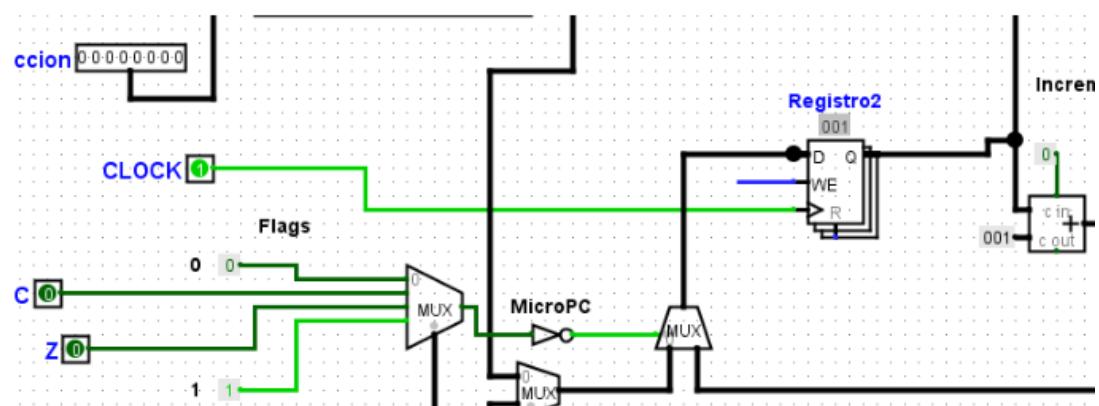
MOV C, A =>ACC <- C



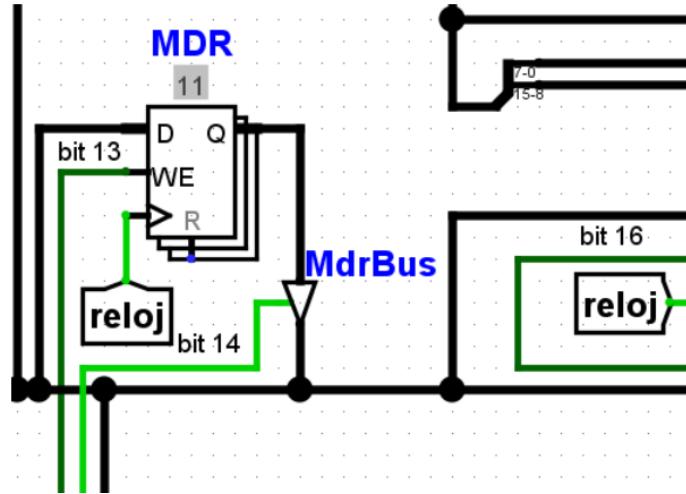
Con el primer pulso de reloj, apunta a la primera instrucción del programa que se va a ejecutar.



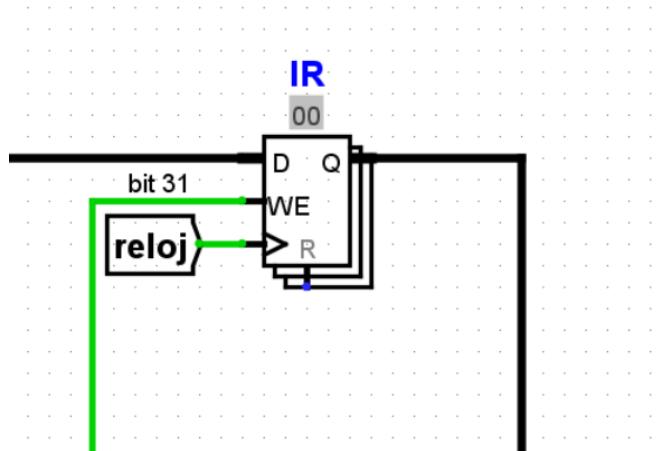
Avanza hacia la siguiente microinstrucción.



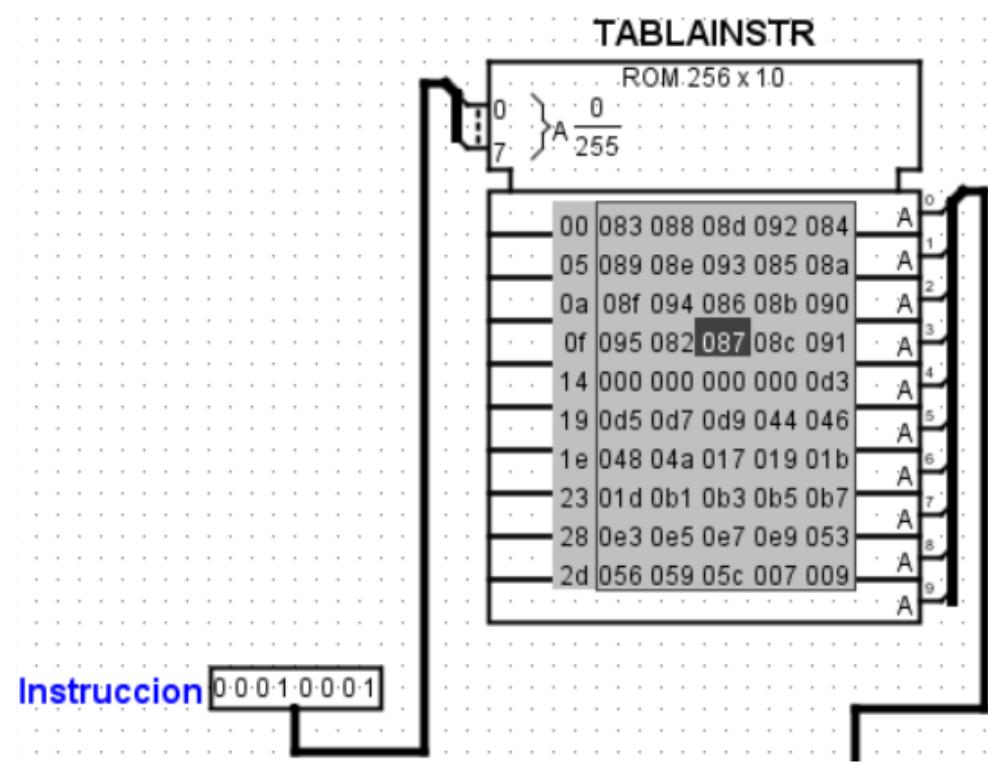
Registro 2 comienza con en 1 y se va a ir incrementando.



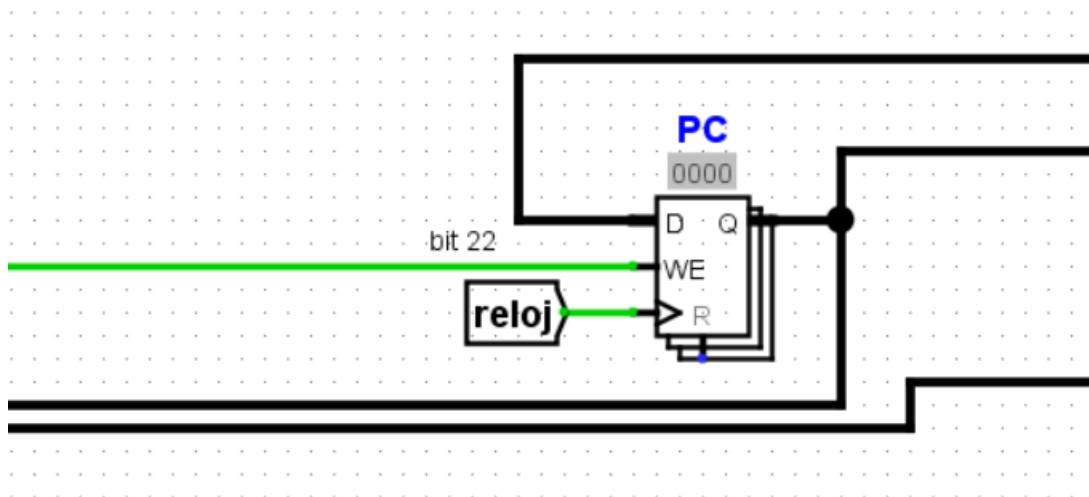
El Registro de Datos de Memoria escribe el código de operación que se está ejecutando $MDR \leftarrow mp[MAR]$.



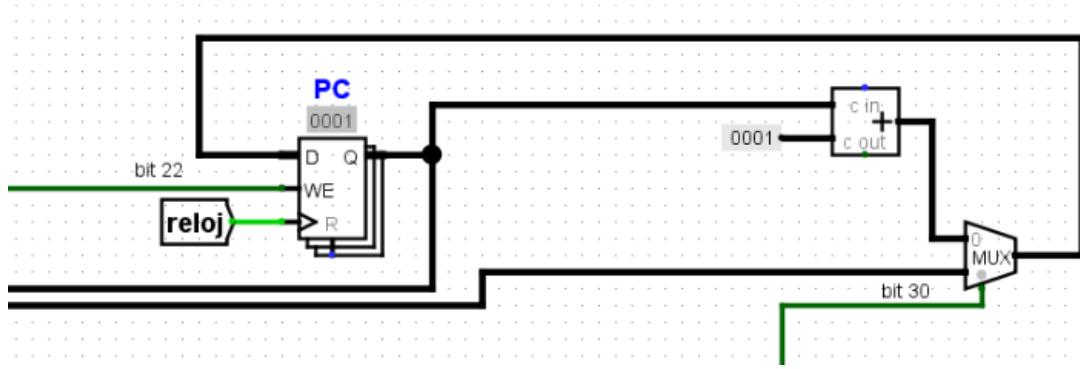
El bit 31 habilita la escritura en el registro IR. $IR \leftarrow MDR$.



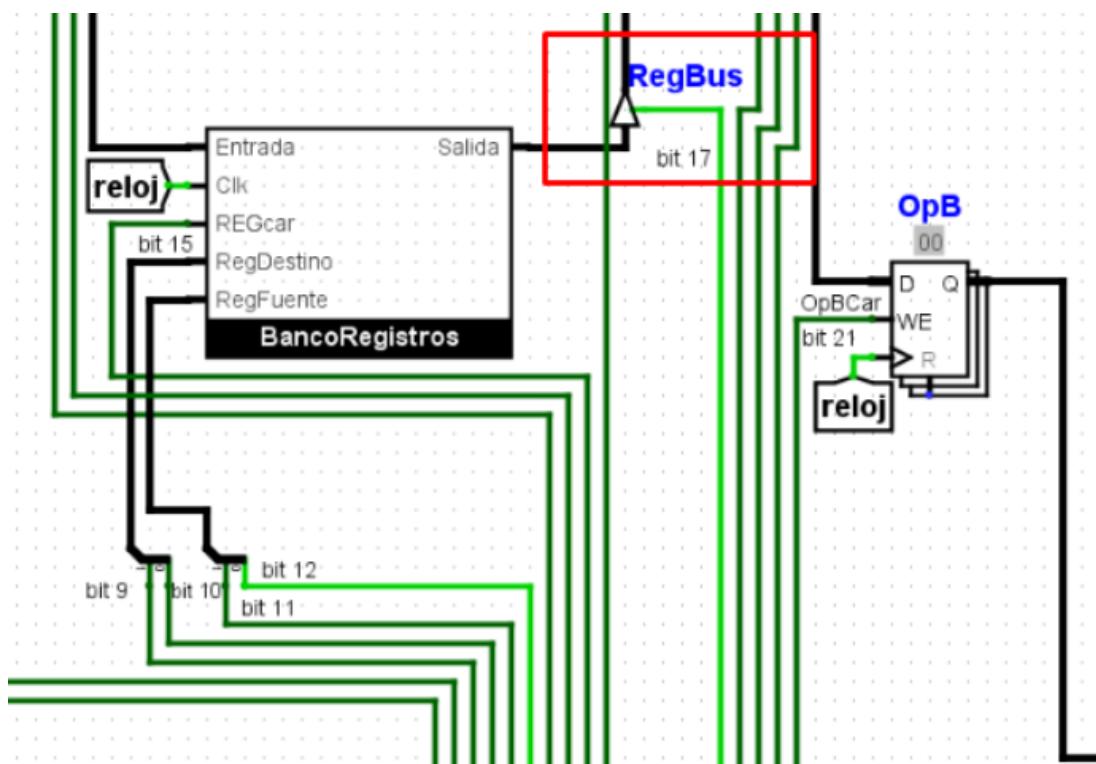
Decodificación de la dirección de microprograma MOV C, A 11 87.



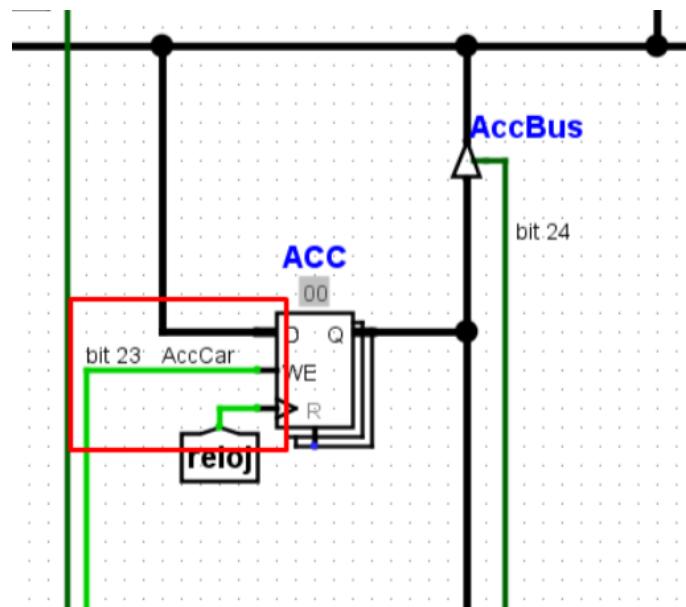
El bit 22 habilita la escritura en el registro PC (contador de programa).



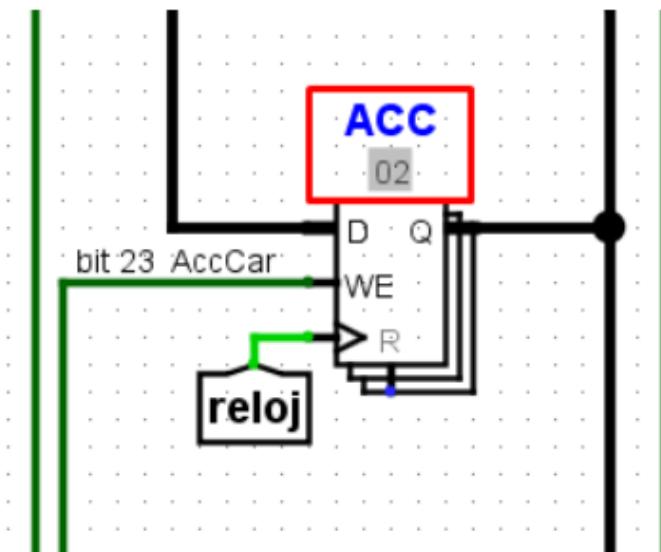
Se incrementa en 1 el contador de programa.



Ejecución de una instrucción: `MOV C, A`. Bit 17: habilita la salida del banco de registros hacia el bus.



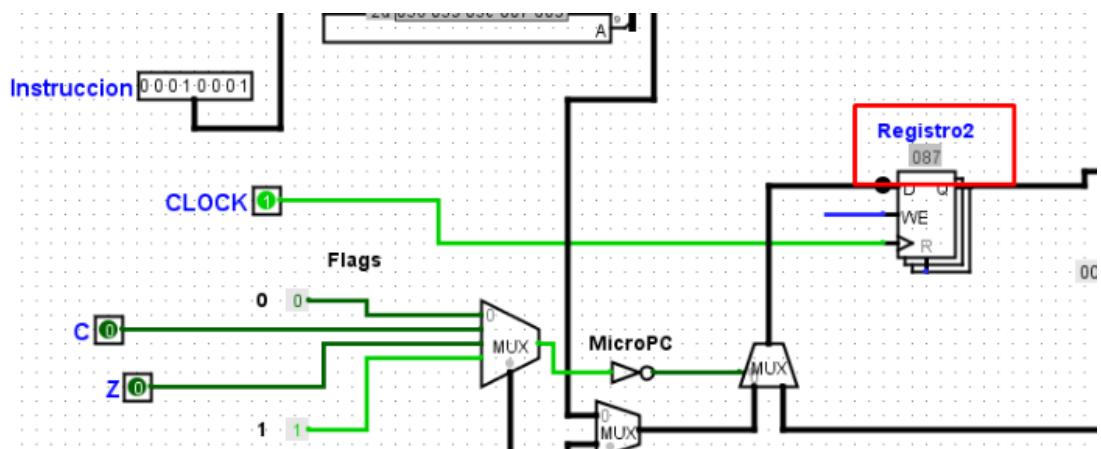
Bit 23: habilita la escritura en ACC.



El acumulador tiene cargado el inicial de C=02.

| | | |
|-----|-------------------|---|
| 070 | 00002100 00104000 | A |
| 072 | 000c0000 00002100 | A |
| 074 | 00002100 00804000 | A |
| 076 | 00400000 00030800 | A |
| 078 | 00121800 000c0000 | A |
| 07a | 00002100 00002100 | A |
| 07c | 00804000 01800000 | A |
| 07e | 01008000 01008400 | A |
| 080 | 01008200 01008600 | A |
| 082 | 00820000 00028000 | A |
| 084 | 00028400 00028200 | A |
| 086 | 00028600 00821000 | A |

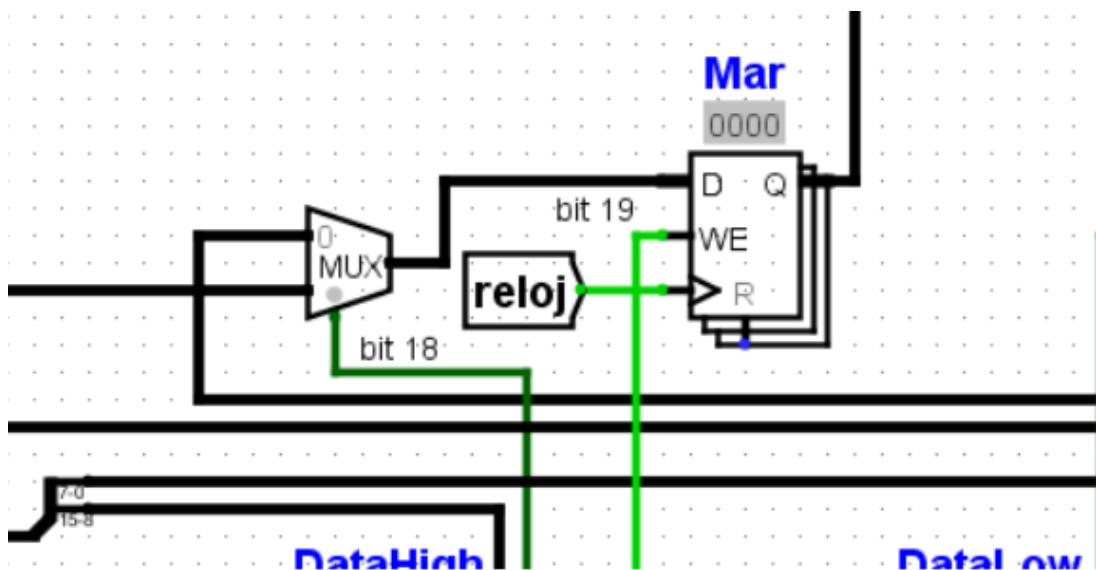
Memoria de control: comienza con el contenido de la instrucción.



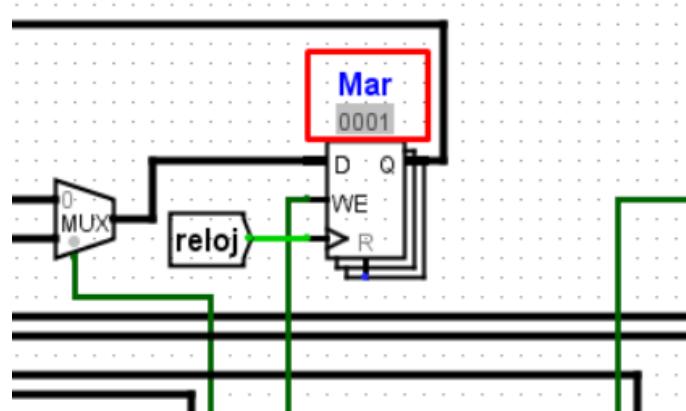
Registro 2 carga la dirección 087 leída en TABLAINSTR.

| | | | |
|--|-----|---------------------|-----------|
| | 068 | e000 0000 0000 0000 | A 9 |
| | 06c | 0000 0000 0000 0000 | A 10 |
| | 070 | 0000 0000 0000 0000 | A 11 |
| | 074 | 0000 0000 e000 0000 | A 12 |
| | 078 | 0000 0000 0000 0000 | A 13 |
| | 07c | e000 e000 e000 e000 | A 14 |
| | 080 | e000 e000 e000 e000 | A 15 |
| | 084 | e000 e000 e000 e000 | A e000 |

Vuelve a la búsqueda y decodificación de la siguiente instrucción; todas las instrucciones terminan con esta dirección explícita.



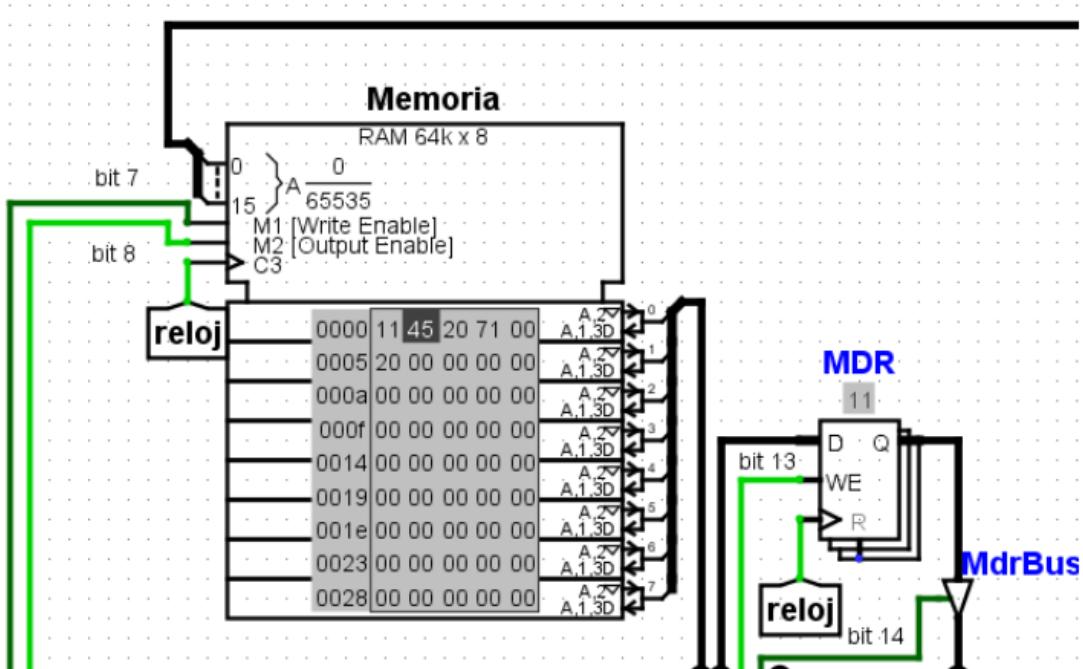
Bit 19: habilita la escritura en MAR.



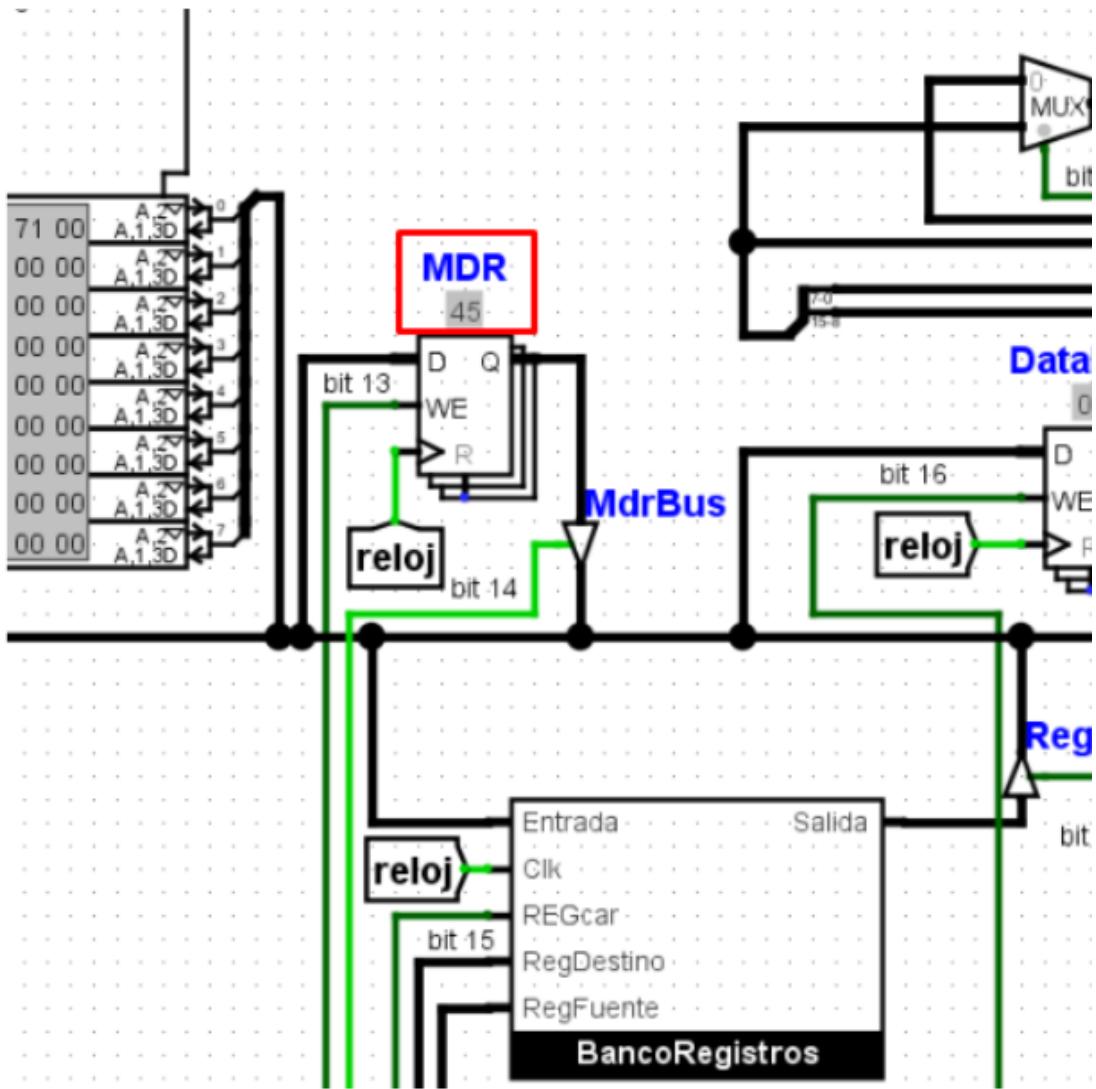
Proceso de búsqueda: MAR <- PC.

9.2. ADD A

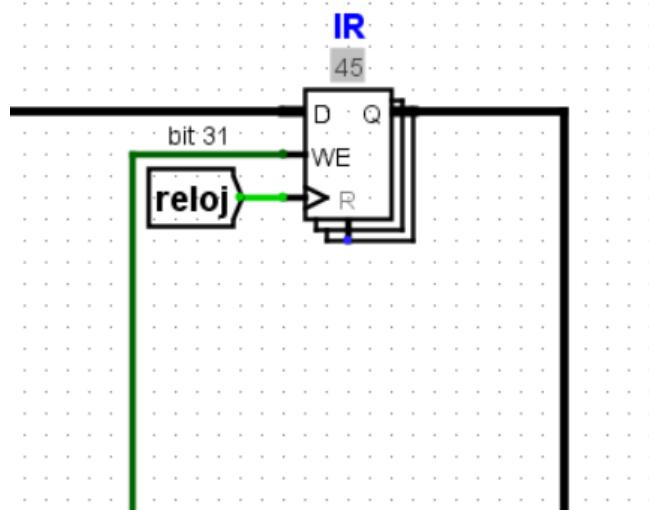
ADD A=>ACC <-ACC + ACC



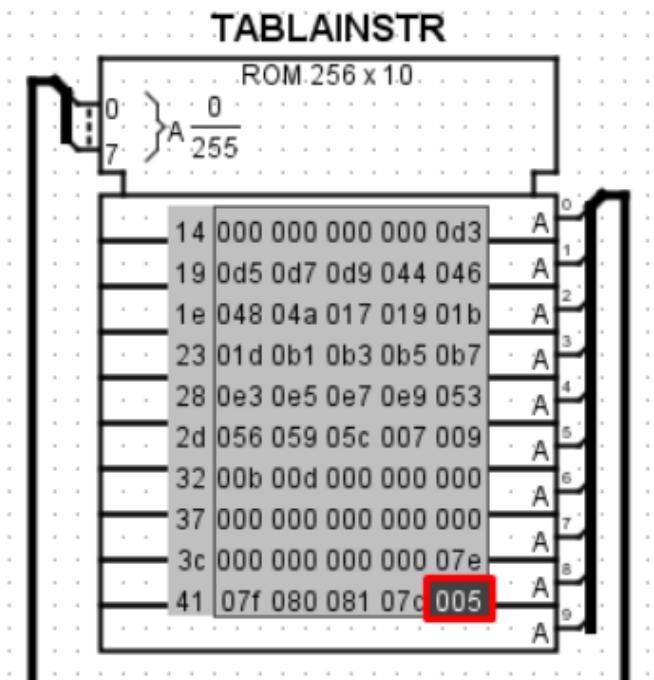
Apunta a la siguiente instrucción del programa que se va a ejecutar.



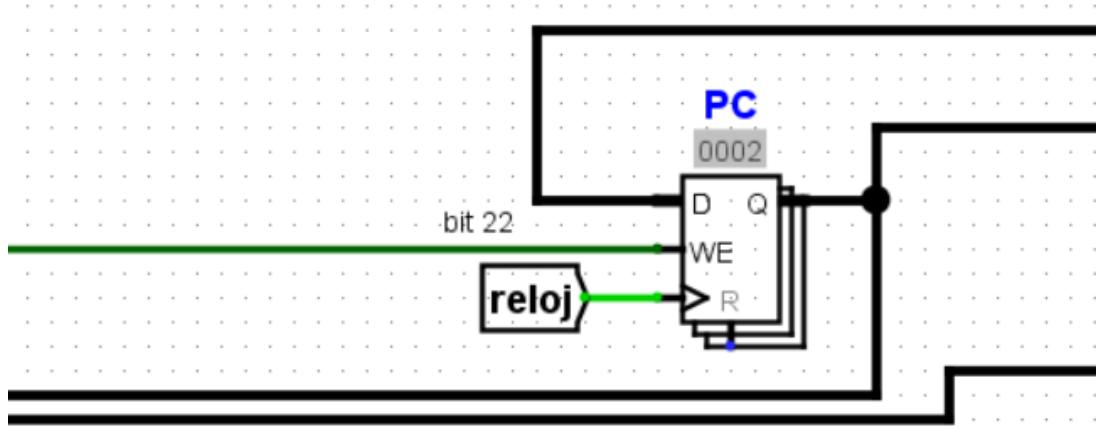
El Registro de Datos de Memoria escribe el código de operación que se está ejecutando
 $MDR \leftarrow mp[MAR]$.



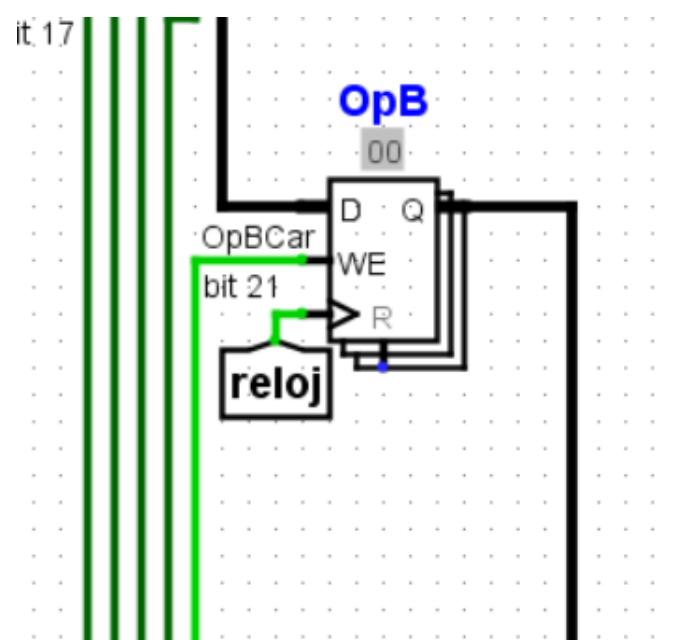
Carga en el registro de instrucción la instrucción que vamos a ejecutar. El código de operación de esta instrucción debe ser decodificado durante la fase de decodificación.



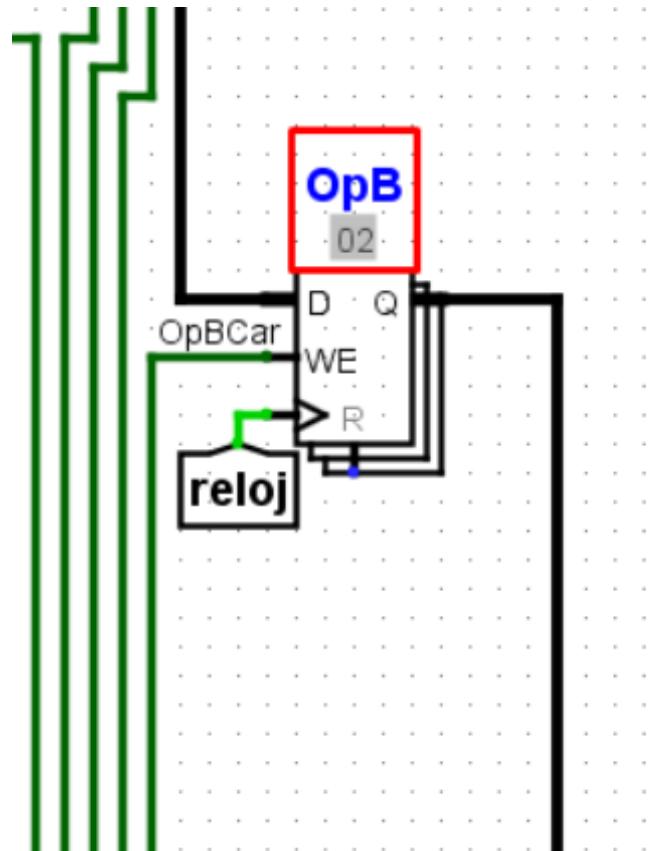
Decodificación de la dirección de microprograma ADD A 45 05.



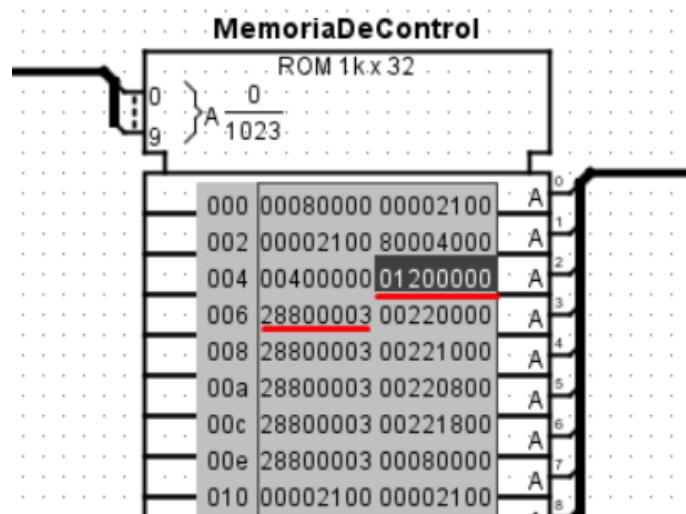
Se incrementa en 1 el contador de programa.



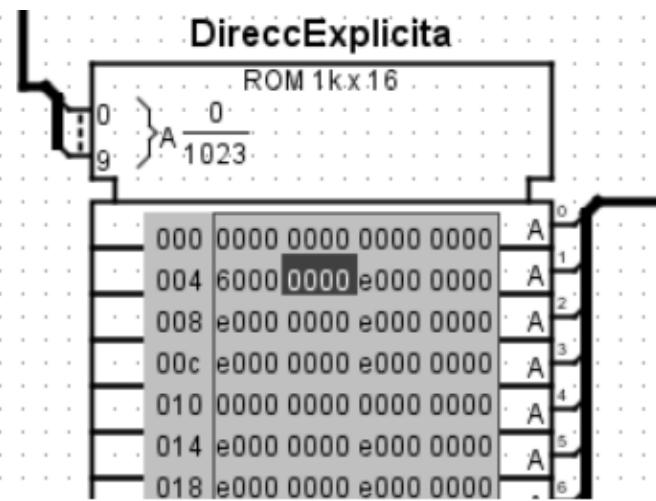
Bit 21: habilita la escritura en el registro OpB.



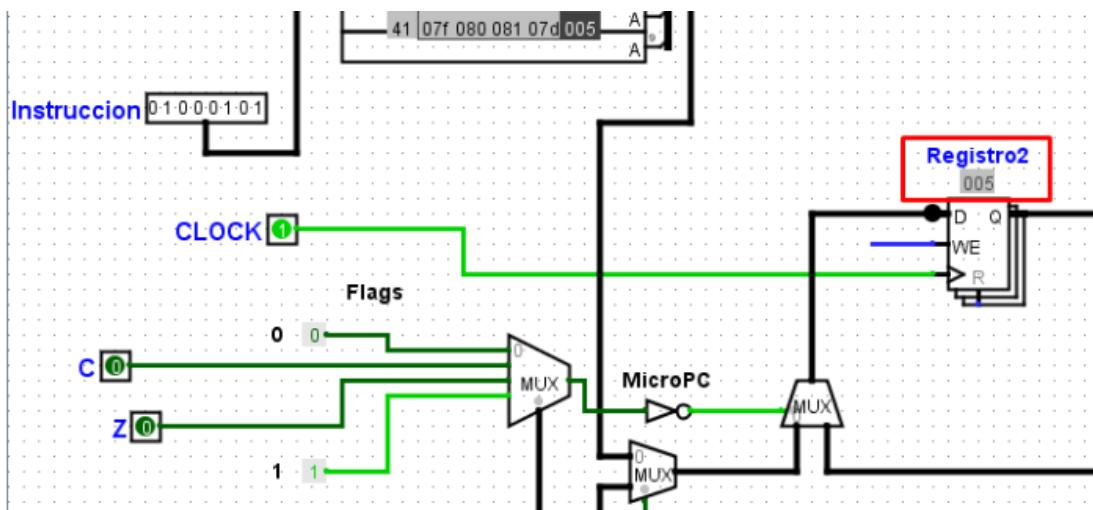
El registro OpB carga el valor que sale del banco de registro.



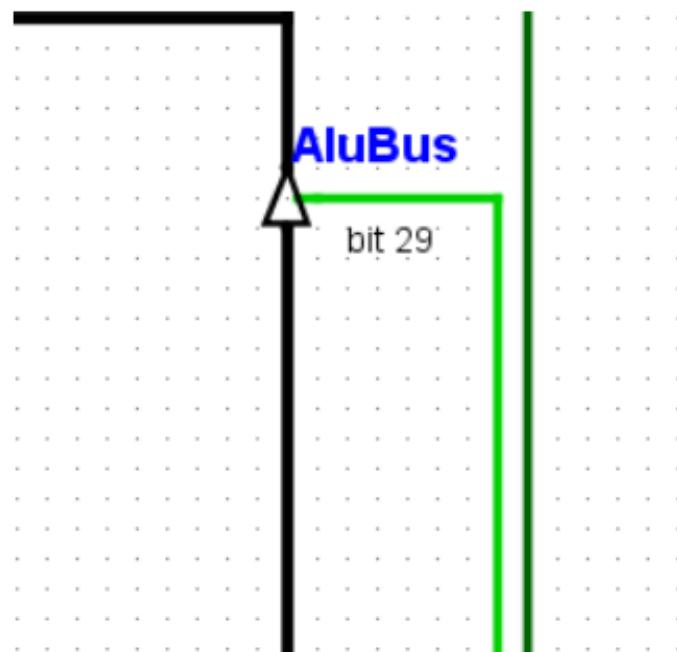
Memoria de control: comienza con el contenido de la instrucción.



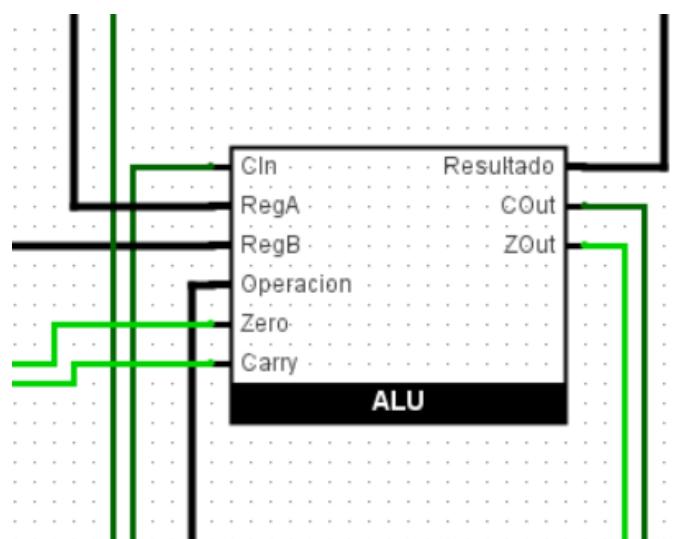
Avanza hacia la siguiente microinstrucción.

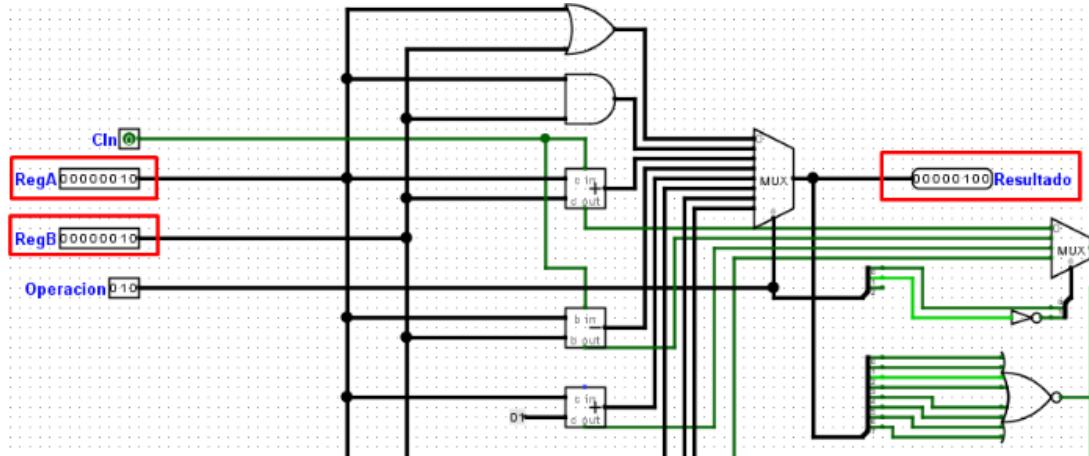


Registro 2 carga la dirección 005 leída en TABLA INSTR.

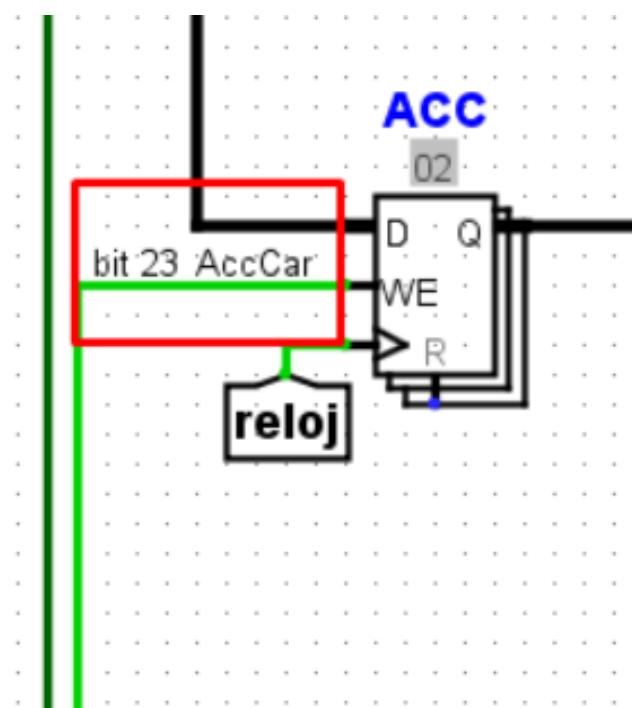


Bit 29: habilita la salida de la ALU hacia el bus.

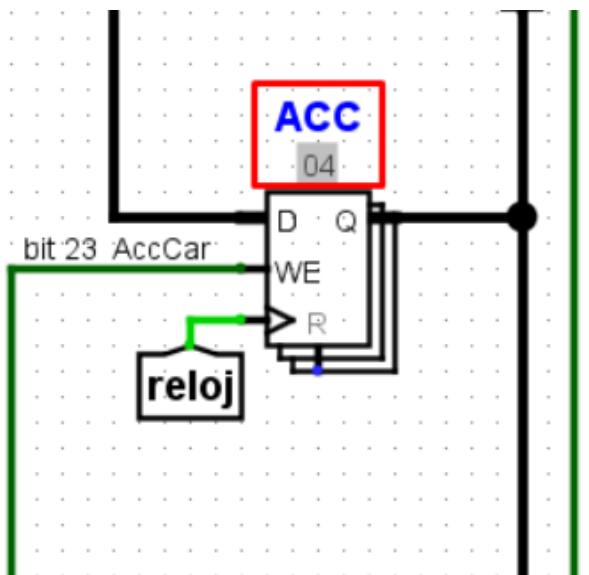




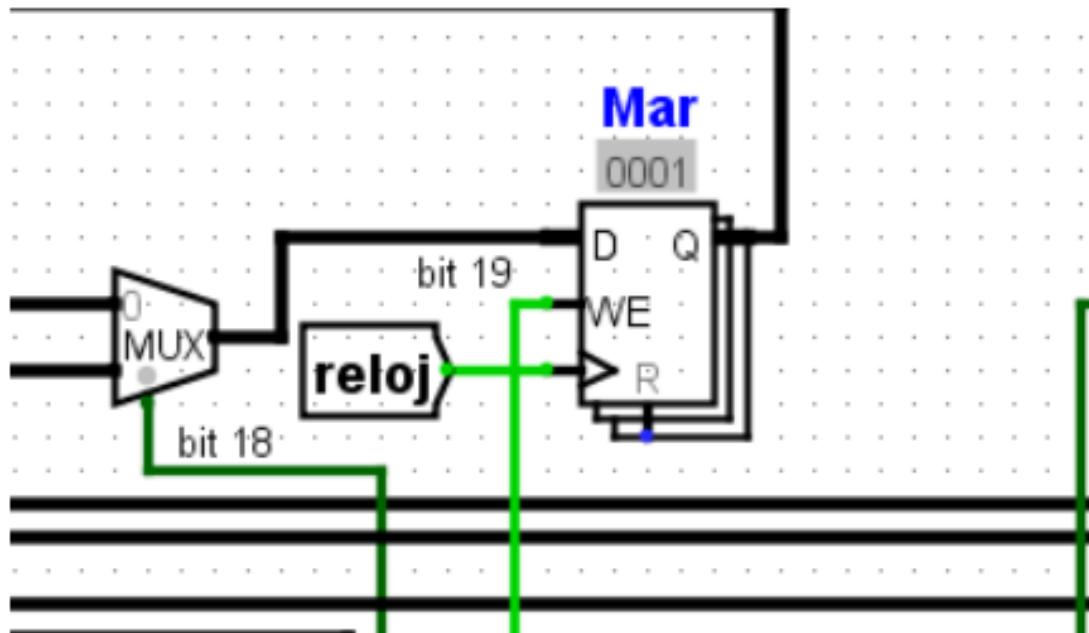
La Alu realiza la suma ADD A.



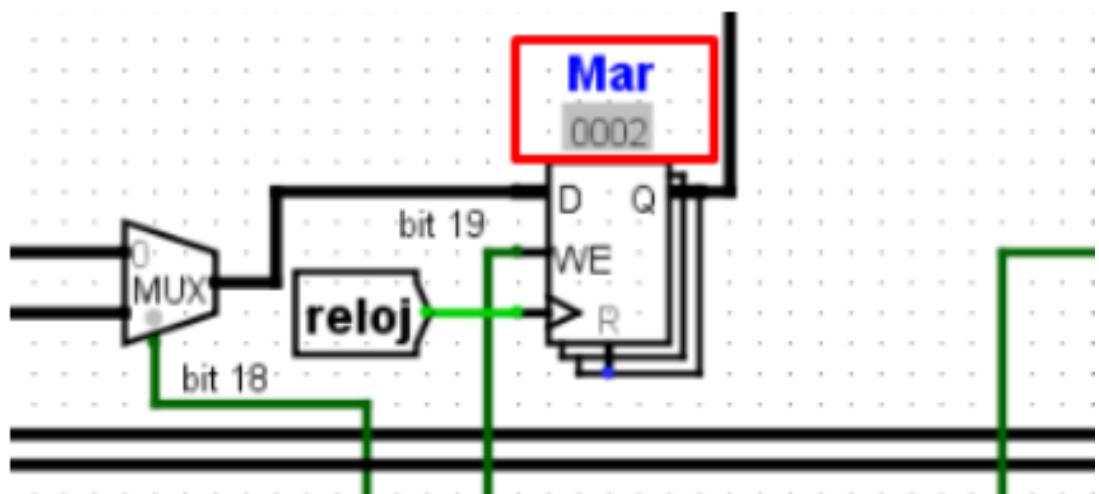
Bit 23: habilita la escritura en ACC.



El acumulador guarda la suma ADD A.



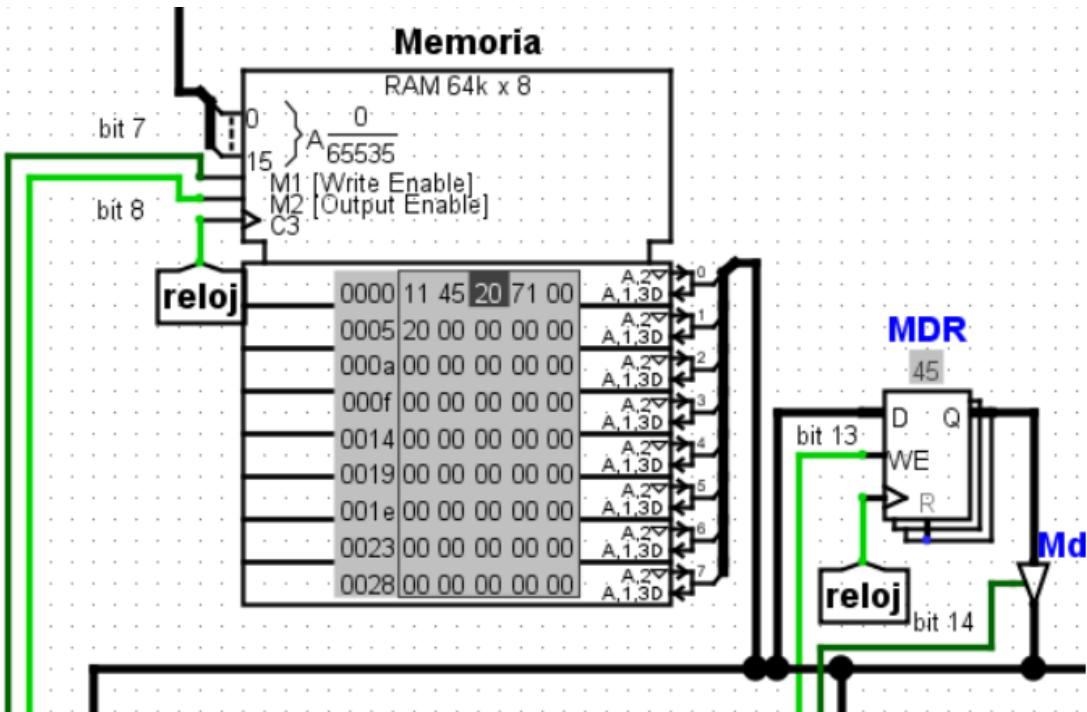
Bit 19: habilita la escritura en MAR.



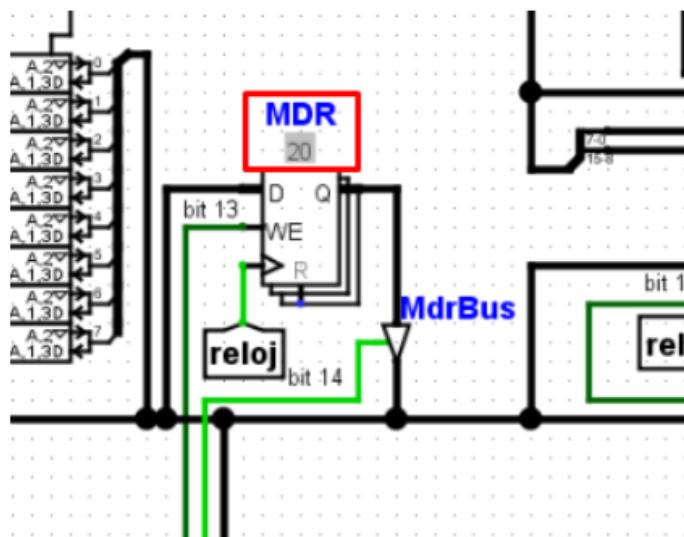
Proceso de búsqueda: MAR <- PC.

9.3. ANA B

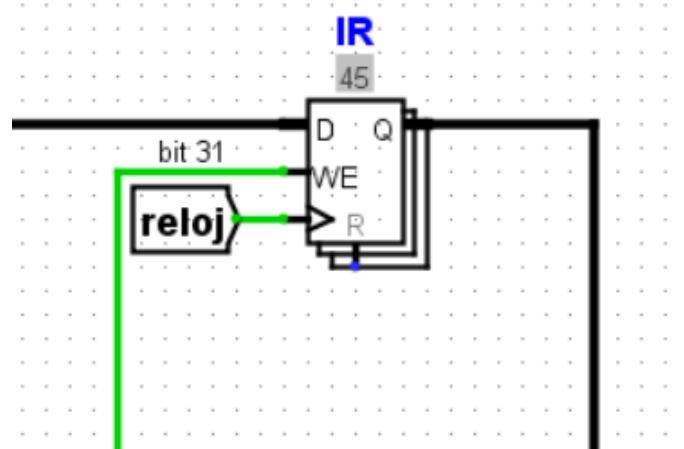
ANA B=>ACC<-ACC and B



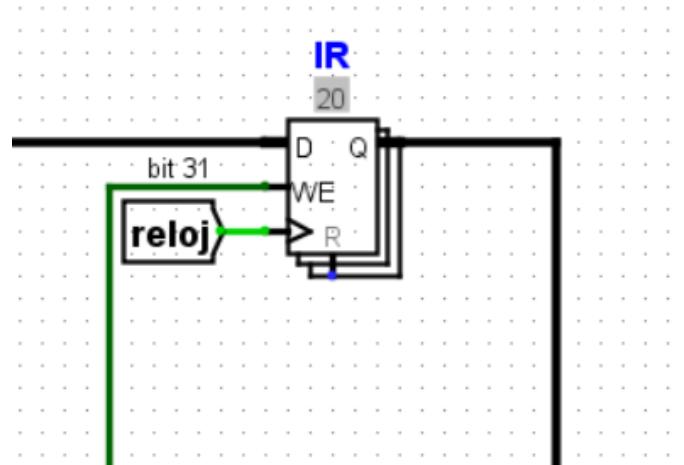
Apunta a la siguiente instrucción del programa que se va a ejecutar.



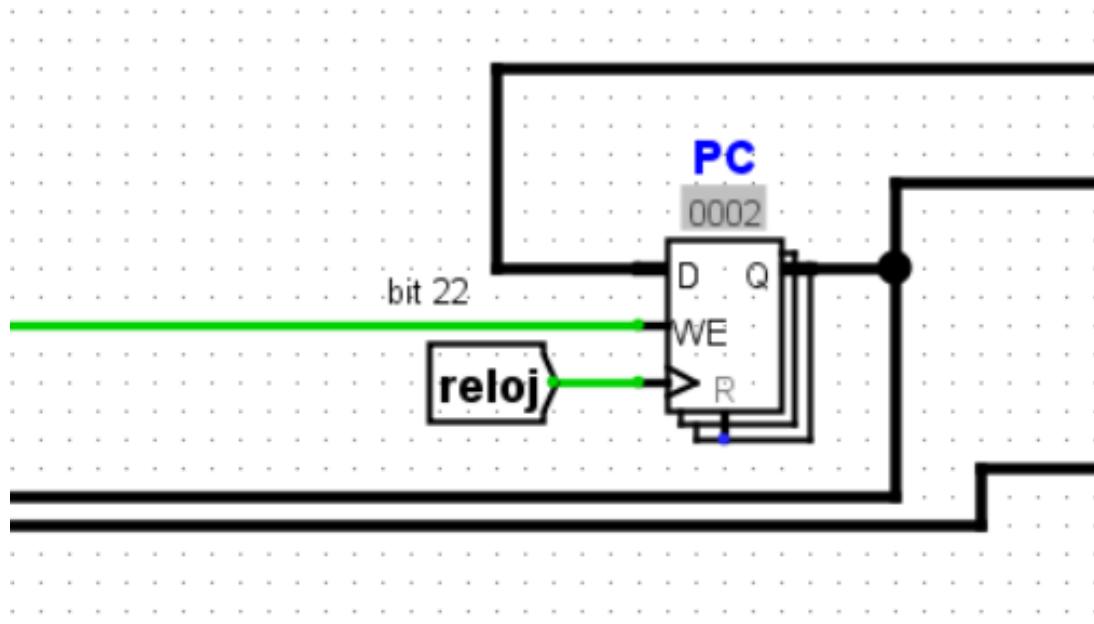
El Registro de Datos de Memoria escribe el código de operación que se está ejecutando
MDR <- mp[MAR].



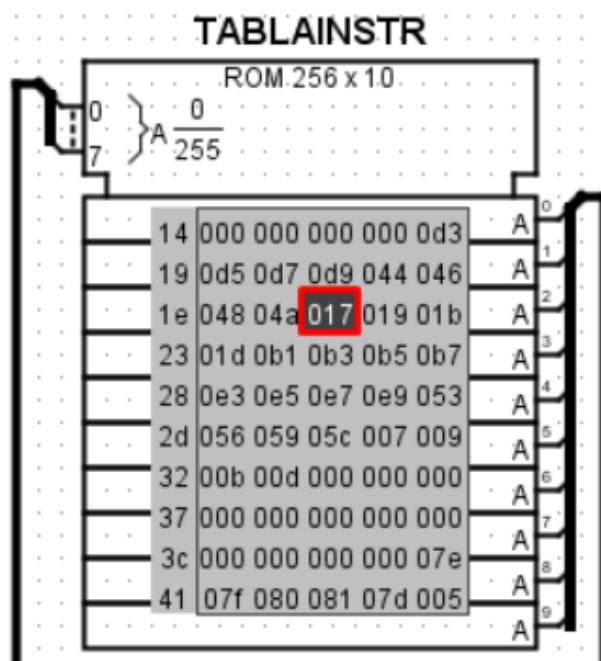
Bit 31: habilita la escritura en el registro IR. IR <- MDR.



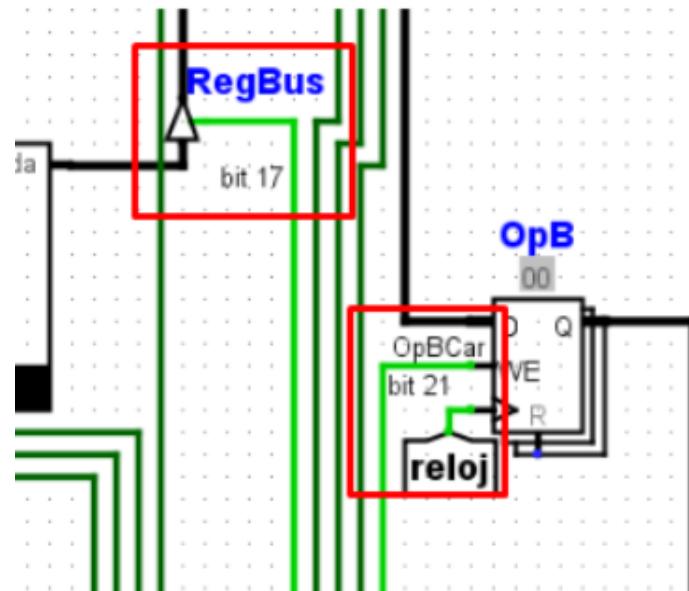
Carga en el registro de instrucción la instrucción que vamos a ejecutar. El código de operación de esta instrucción debe ser decodificado durante la fase de decodificación.



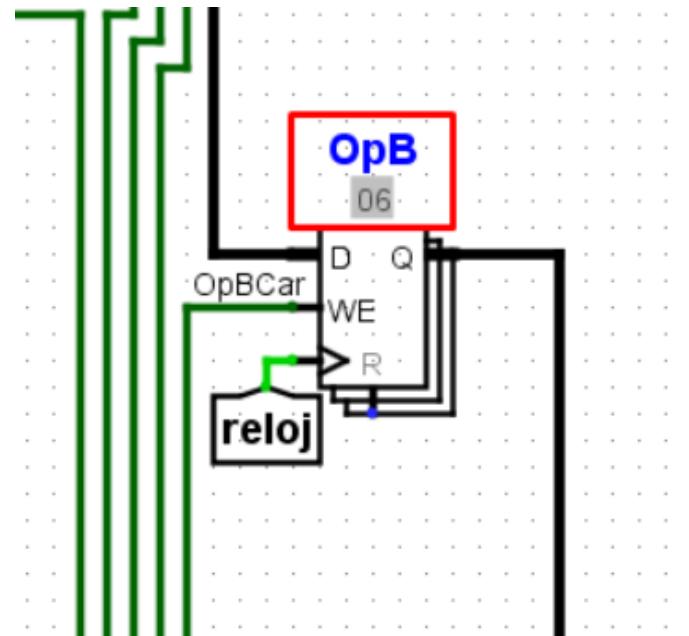
El bit 22 habilita la escritura en el registro PC (contador de programa).



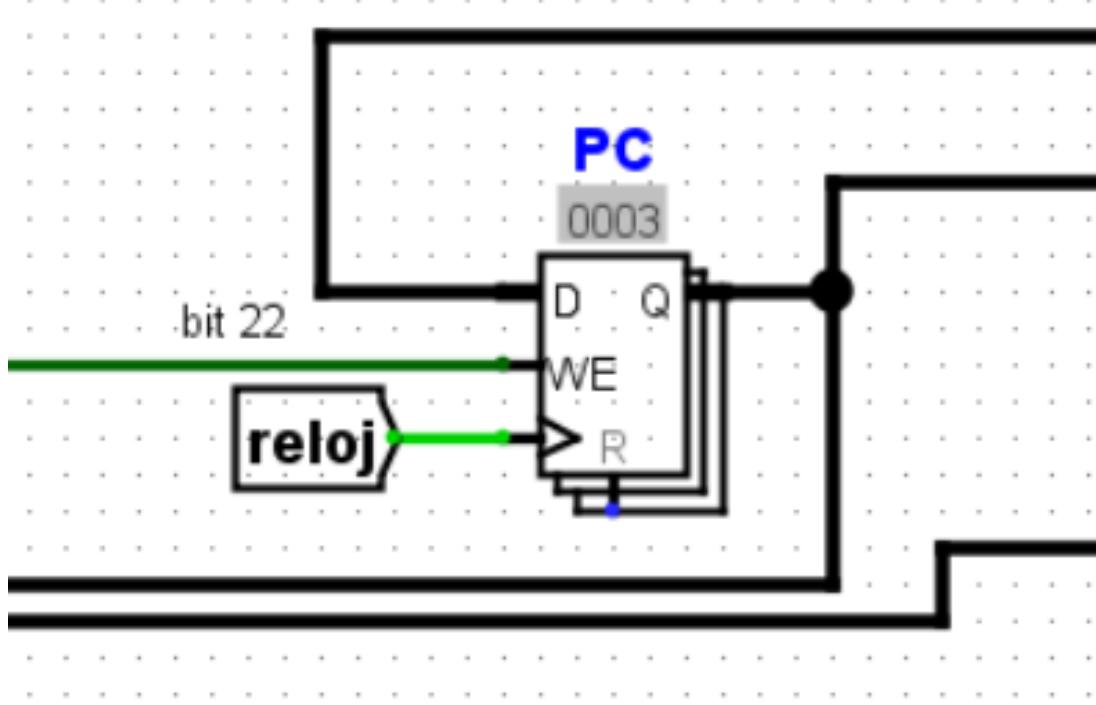
Decodificación de la dirección de microprograma ANA B 20 17.



Bit 17: habilita la salida del banco de registros hacia el bus. Bit 21: habilita la escritura en el registro OpB.



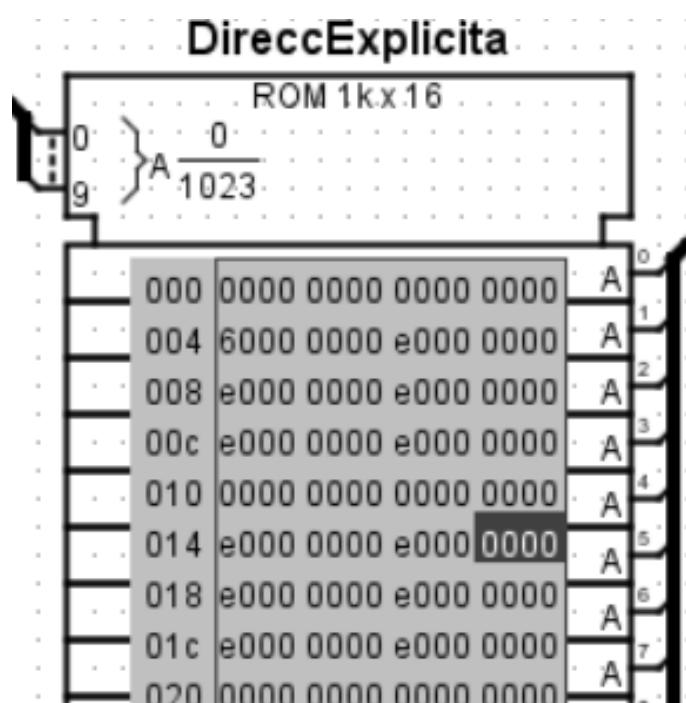
El registro Op B toma el valor del banco de registro.



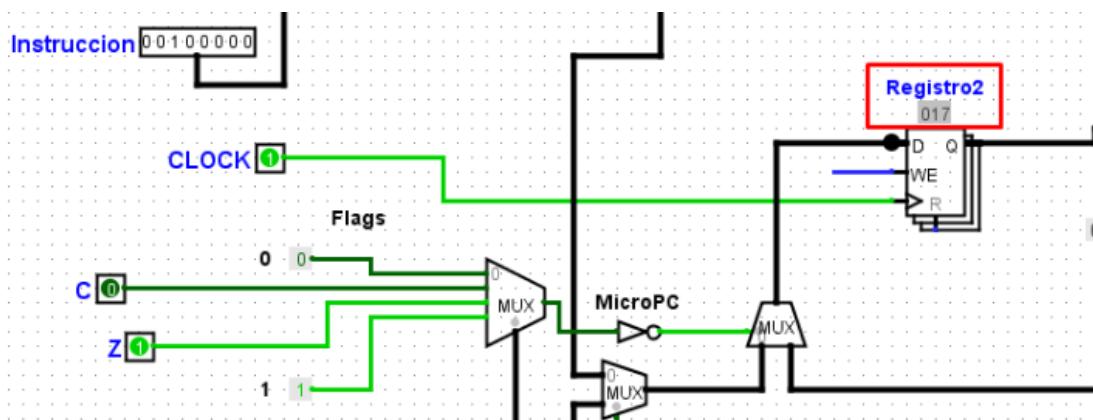
Se incrementa en 1 el contador de programa.

| | | | |
|----------|-----|-----------------------|-----------------|
| | | | A ₈ |
| | | | A ₉ |
| | | | A ₁₀ |
| | | | A ₁₁ |
| | | 016 24800001 00220000 | A ₁₂ |
| mentador | 018 | 24800001 00221000 | A ₁₃ |
| | 01a | 24800001 00220800 | A ₁₄ |
| | 01c | 24800001 00221800 | A ₁₅ |
| | 01e | 24800001 00080000 | A ₁₆ |
| | 020 | 00002100 00002100 | |
| | 022 | 00204000 24800001 | |
| | 024 | 00400000 00080000 | |

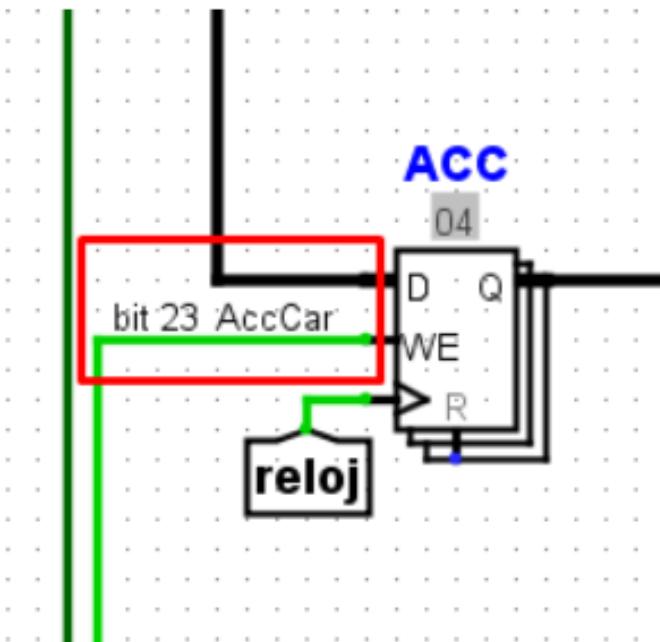
Memoria de control: comienza con el contenido de la instrucción.



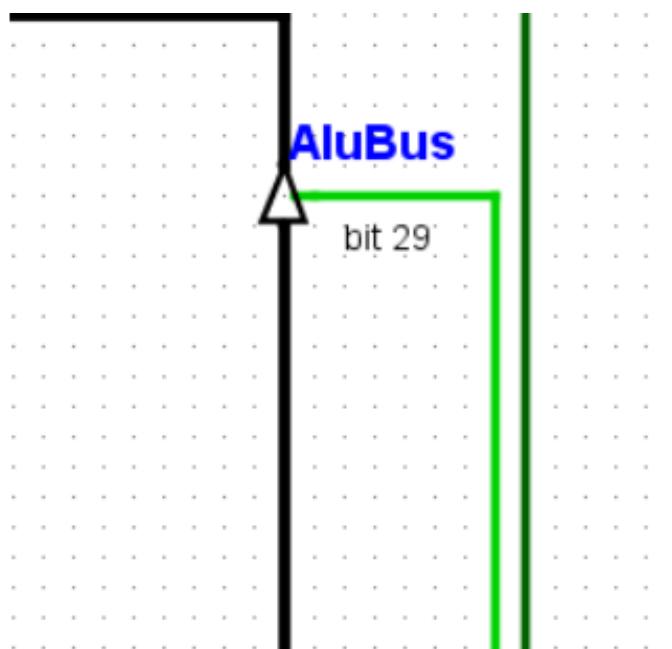
Avanza hacia la siguiente microinstrucción.



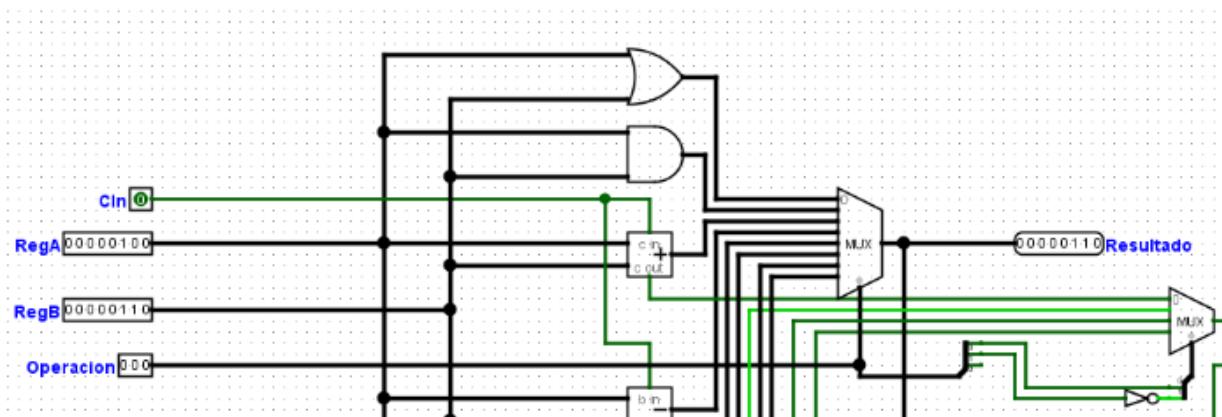
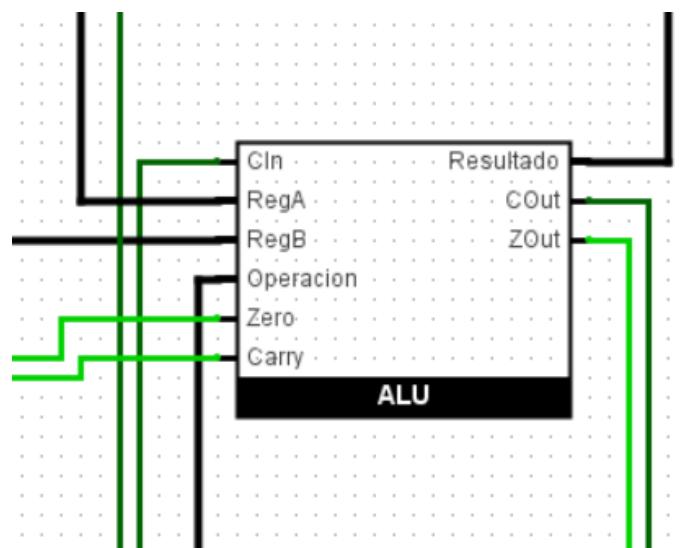
Registro 2 carga la dirección 017 leída en TABLAInstr.

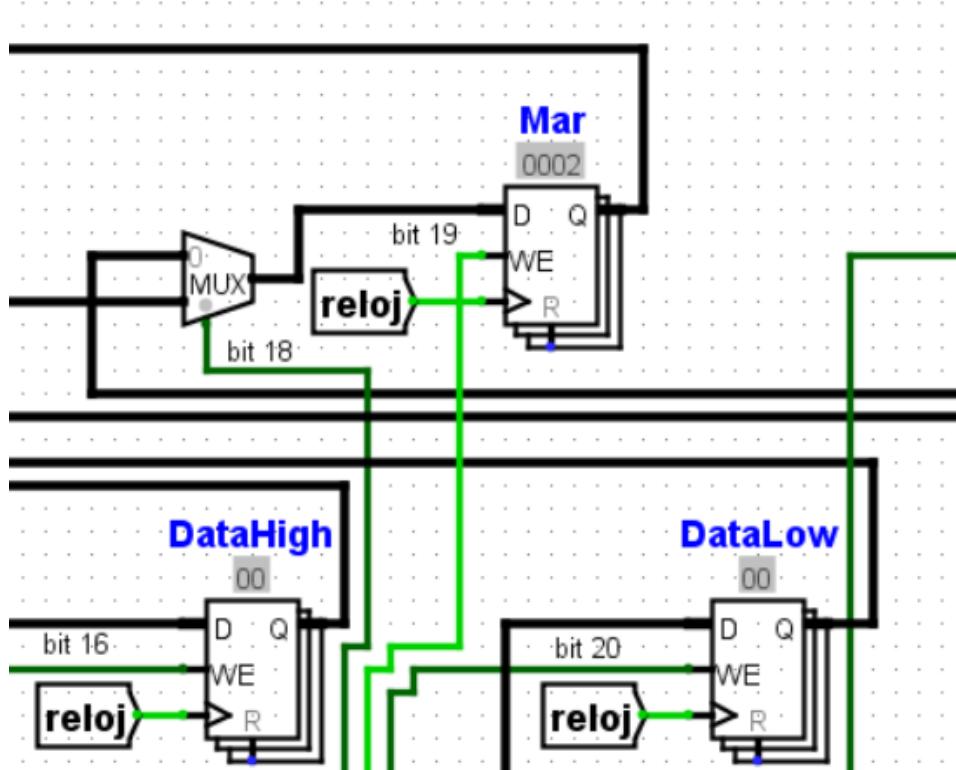


Bit 23: habilita la escritura en ACC.

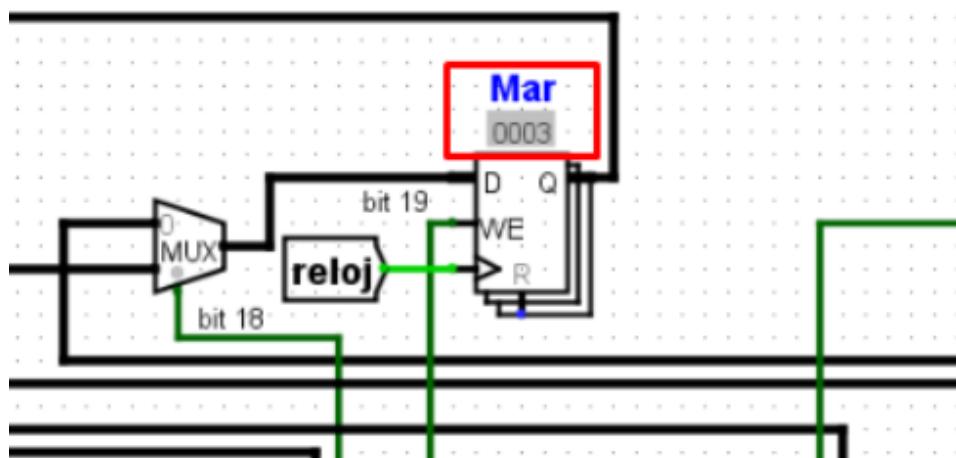


Bit 29: habilita la salida de la ALU hacia el bus.





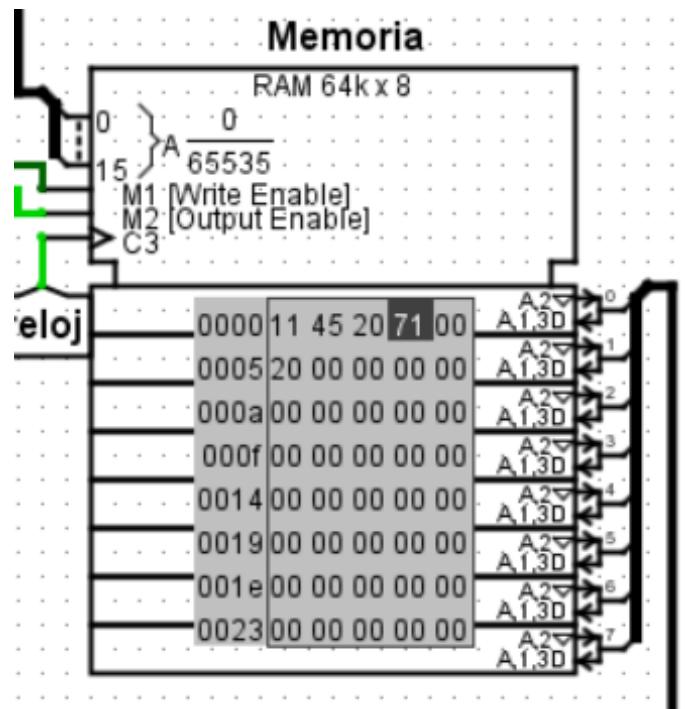
Bit 19: habilita la escritura en MAR.



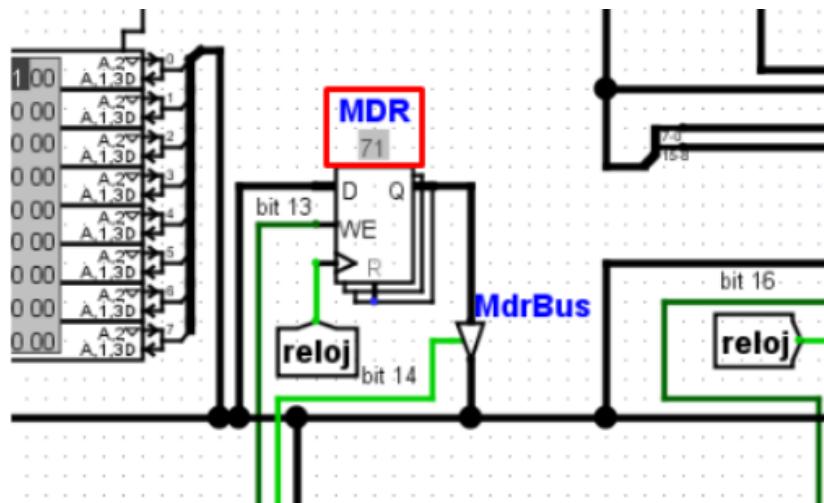
Proceso de búsqueda: MAR <- PC.

9.4. STA 0020

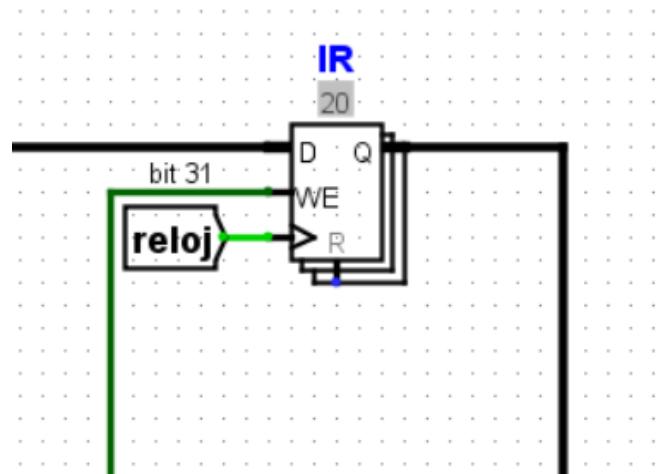
STA 0020 =>MEM[0020] <- ACC



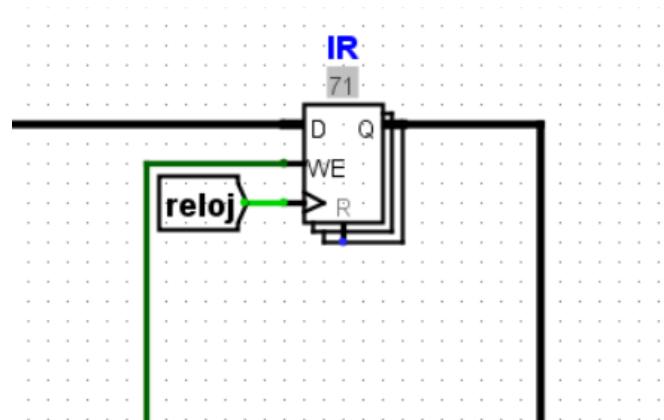
Apunta a la siguiente instrucción del programa que se va a ejecutar.



El Registro de Datos de Memoria escribe el código de operación que se está ejecutando
MDR <- mp[MAR].

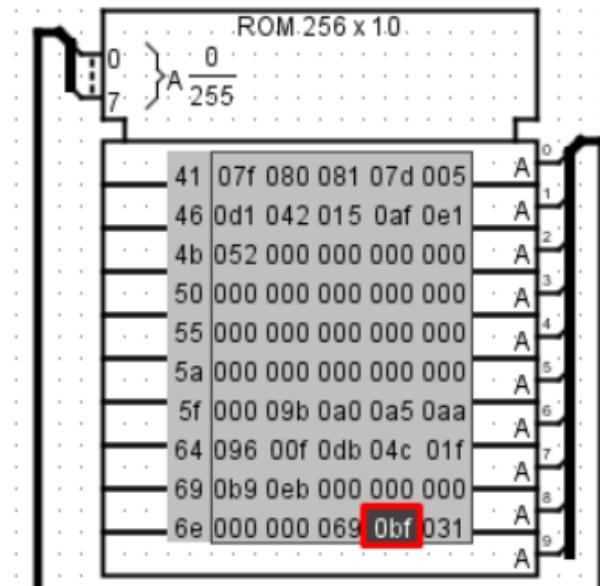


Bit 31: habilita la escritura en el registro IR. IR <- MDR.

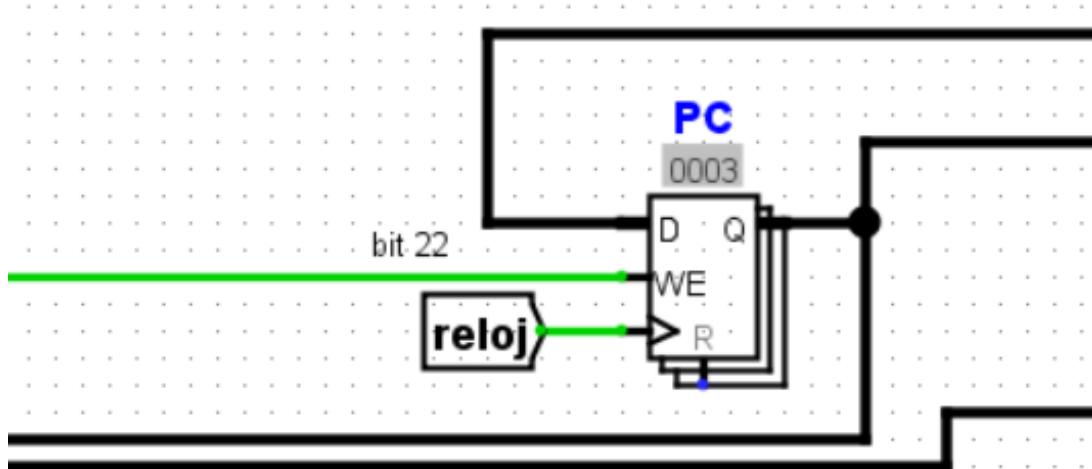


Carga en el registro de instrucción la instrucción que vamos a ejecutar. El código de operación de esta instrucción debe ser decodificado durante la fase de decodificación.

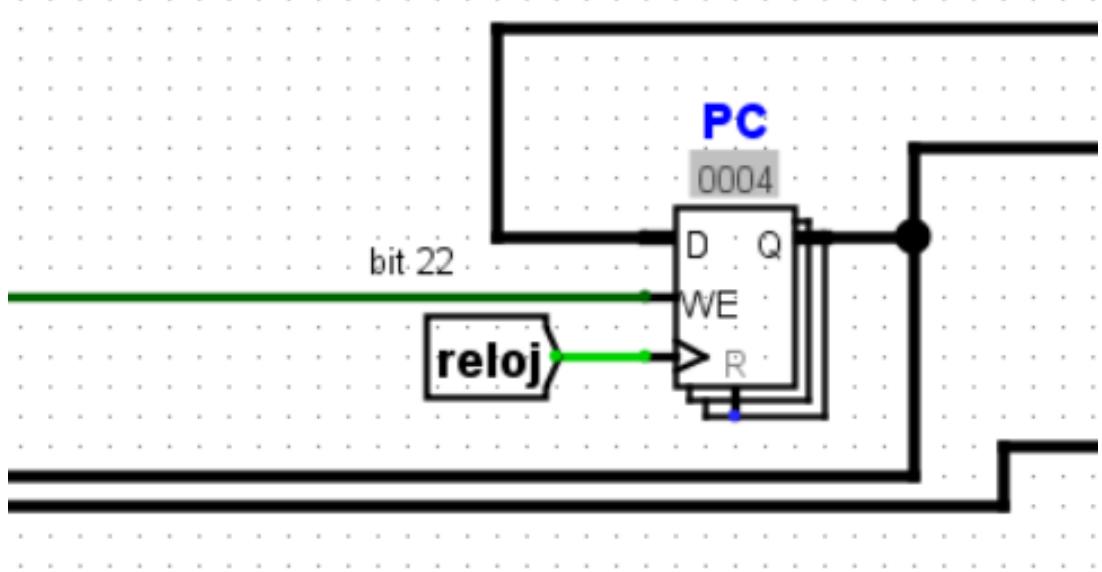
TABLAISTR



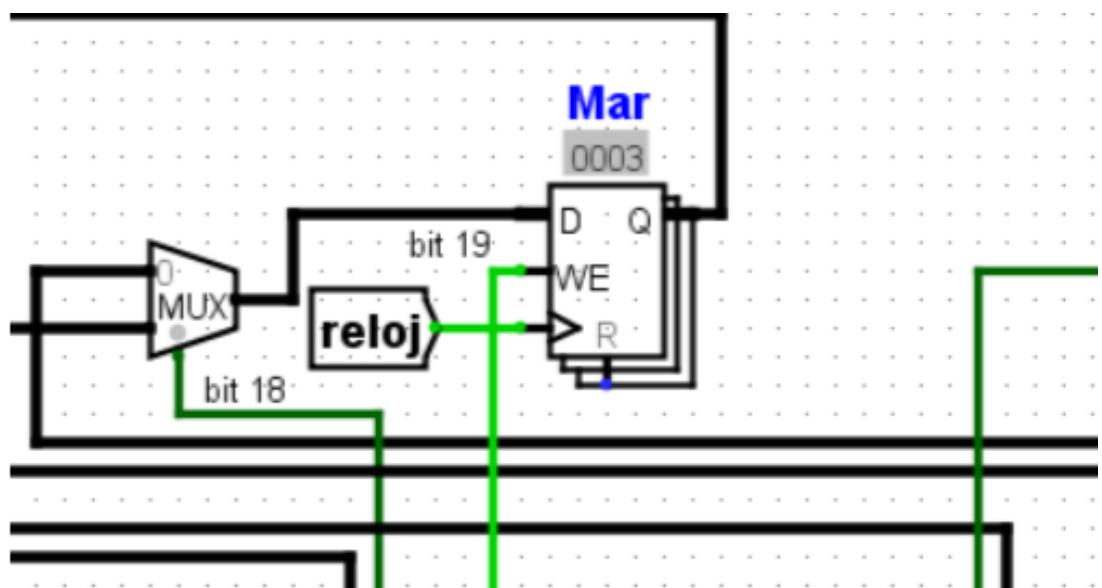
Decodificación de la dirección de microprograma STA dir 71 bf.



El bit 22 habilita la escritura en el registro PC (contador de programa).



Se incrementa en 1 el contador de programa.



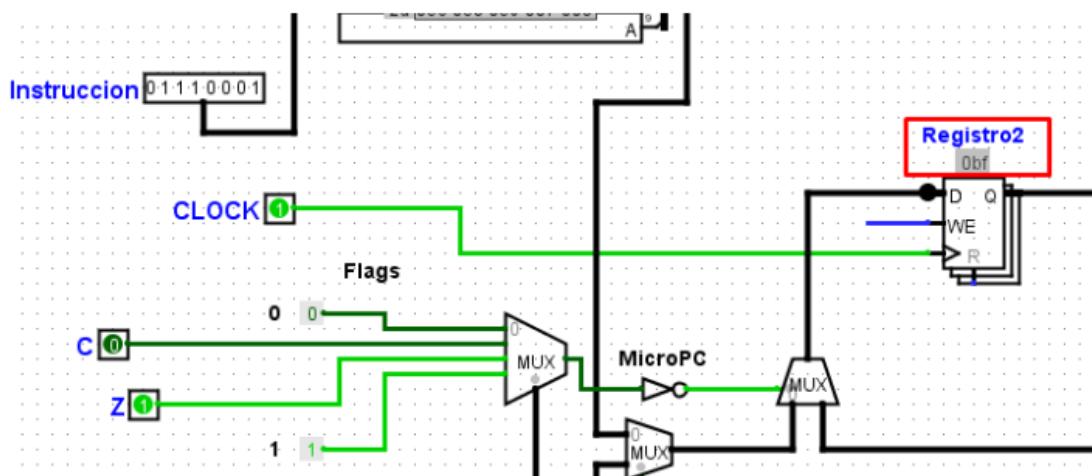
Bit 19: habilita la escritura en MAR.

| | | |
|-----|-------------------|------|
| 0b0 | 20800001 00220000 | A 25 |
| 0b2 | 20800001 00221000 | A 26 |
| 0b4 | 20800001 00220800 | A 27 |
| 0b6 | 20800001 00221800 | A 28 |
| 0b8 | 20800001 00080000 | A 29 |
| 0ba | 00002100 00002100 | A 30 |
| 0bc | 00204000 20800001 | A 31 |
| 0be | 00400000 00080000 | A |

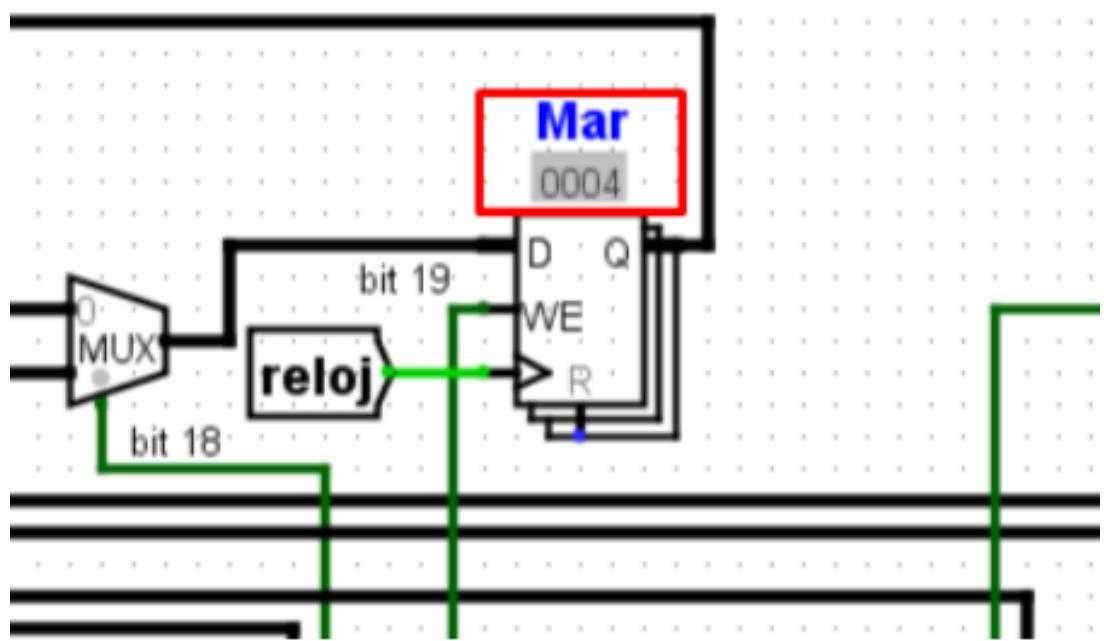
Memoria de control: comienza con el contenido de la instrucción.

| | | |
|-----|---------------------|------|
| 0a8 | 0000 e000 0000 0000 | A 10 |
| 0ac | 0000 0000 e000 0000 | A 11 |
| 0b0 | e000 0000 e000 0000 | A 12 |
| 0b4 | e000 0000 e000 0000 | A 13 |
| 0b8 | e000 0000 0000 0000 | A 14 |
| 0bc | 0000 0000 e000 0000 | A 15 |

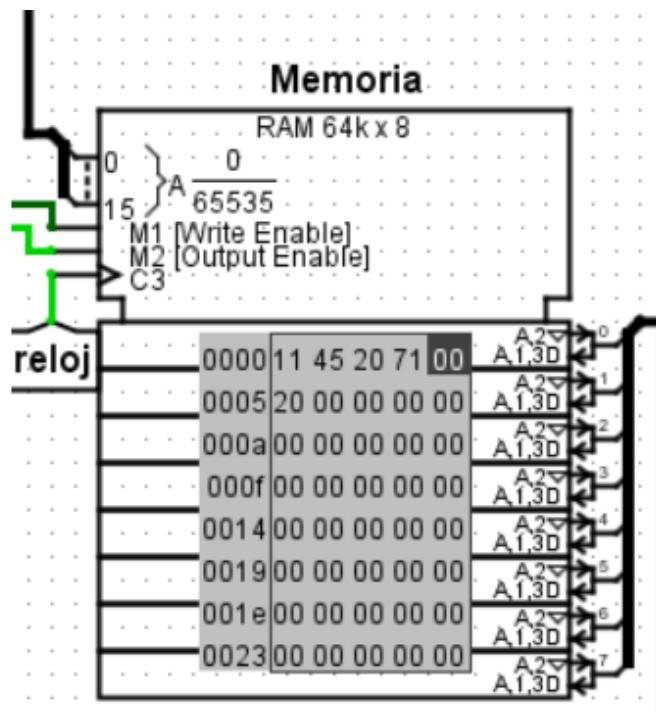
Avanza hacia la siguiente microinstrucción.



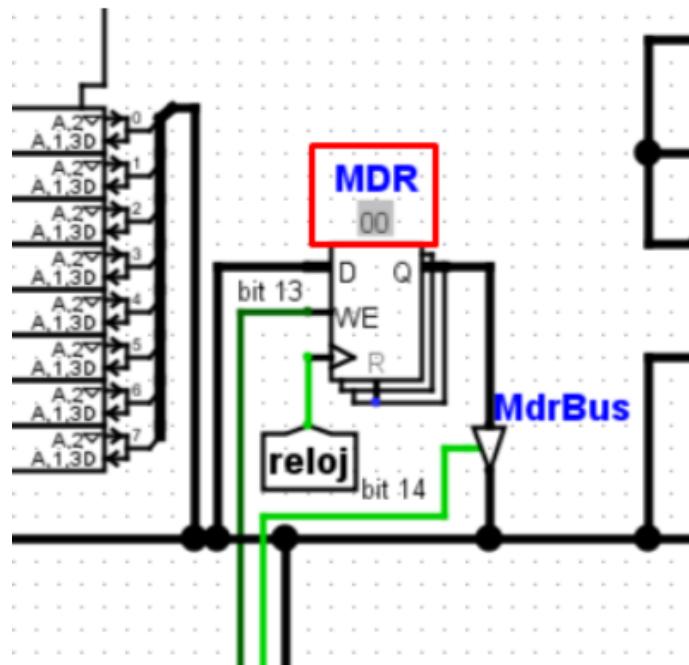
Registro 2 carga la dirección 0bf.



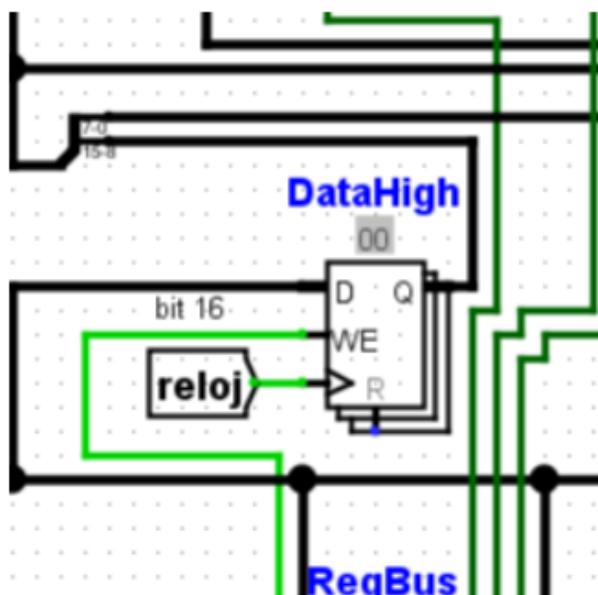
Proceso de búsqueda: MAR <- PC.



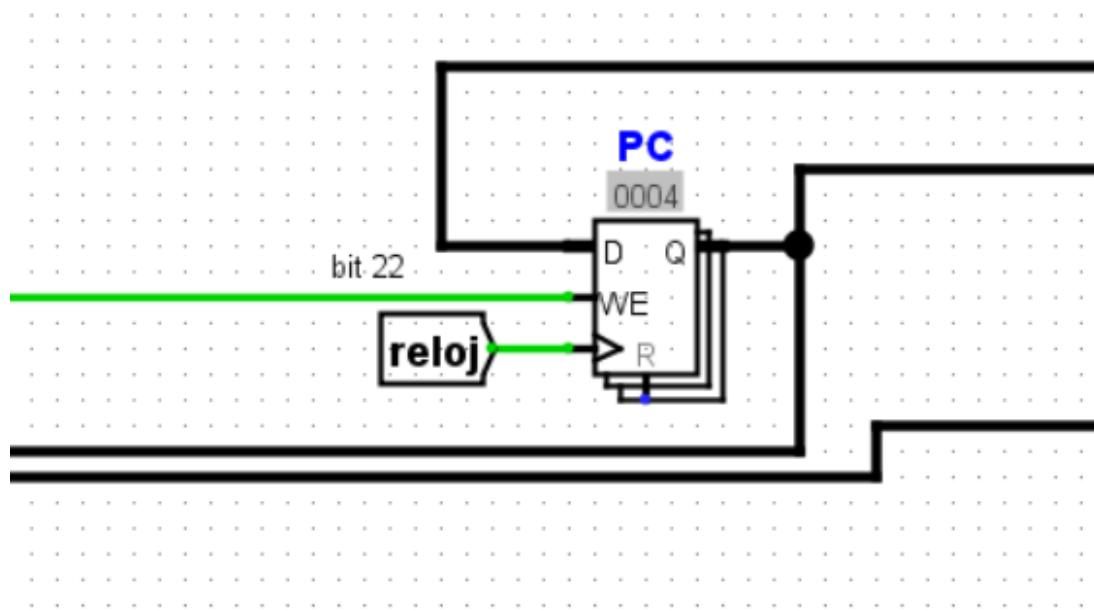
Apunta a la siguiente instrucción del programa que se va a ejecutar.



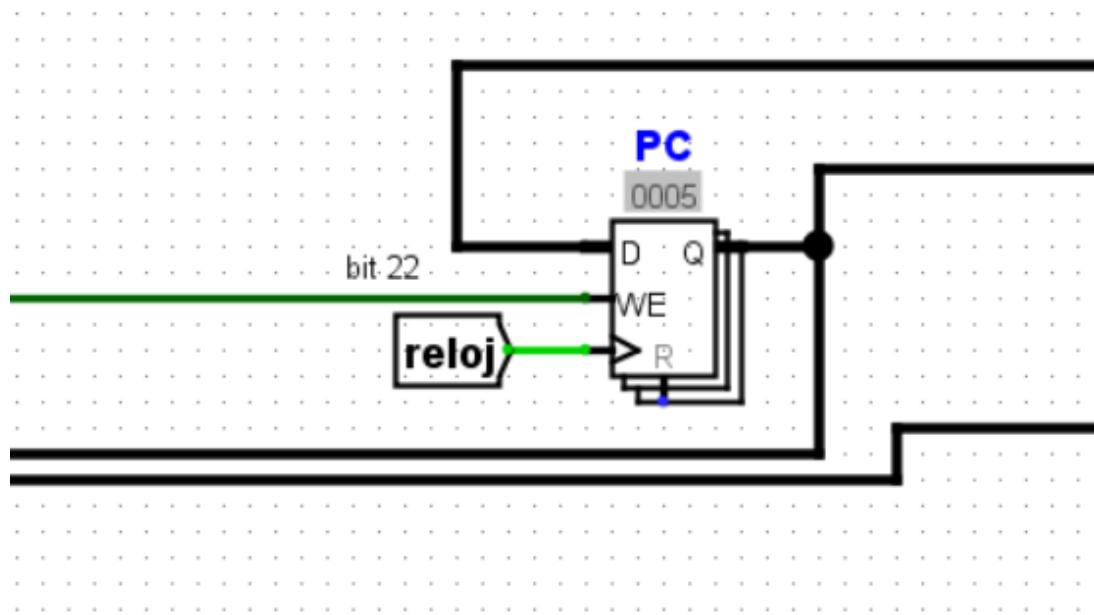
El Registro de Datos de Memoria escribe el código de operación que se está ejecutando
 $MDR \leftarrow mp[MAR]$.



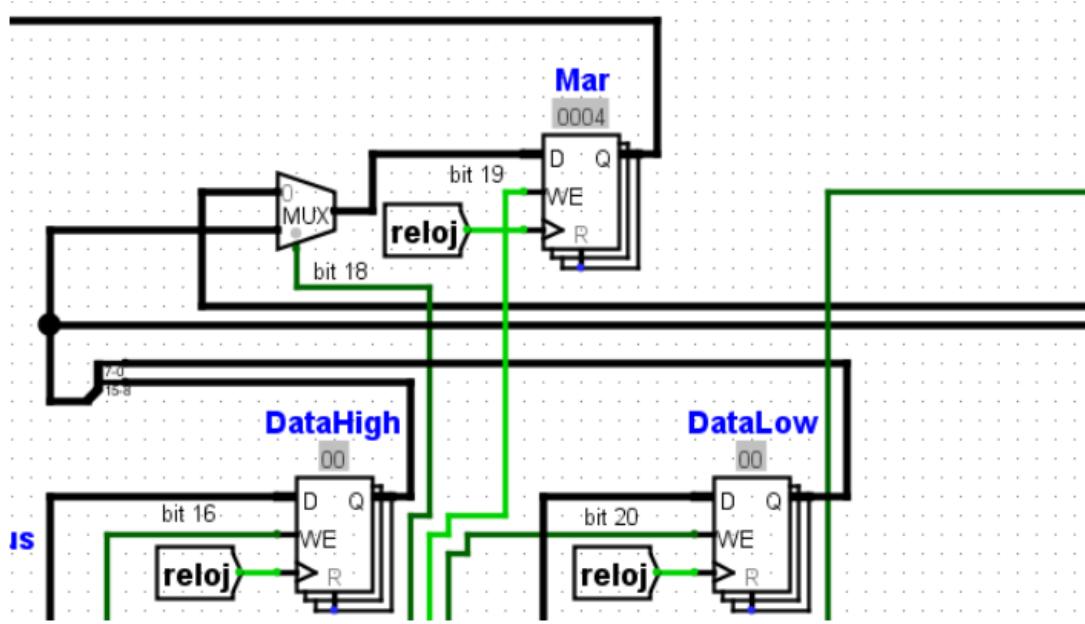
Bit 16: habilita la escritura en el registro DATAHIGH.



El bit 22 habilita la escritura en el registro PC (contador de programa).



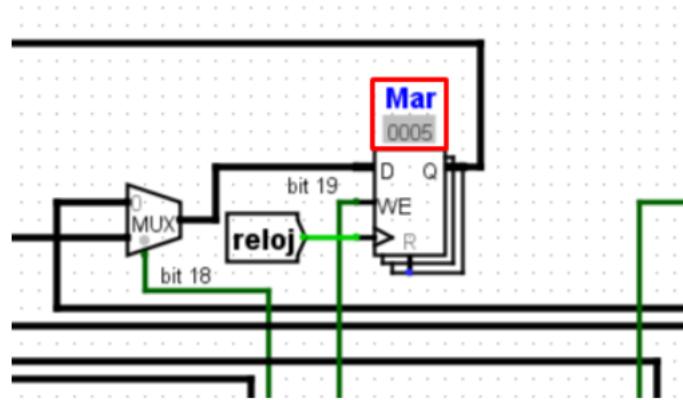
Se incrementa en 1 el contador de programa.



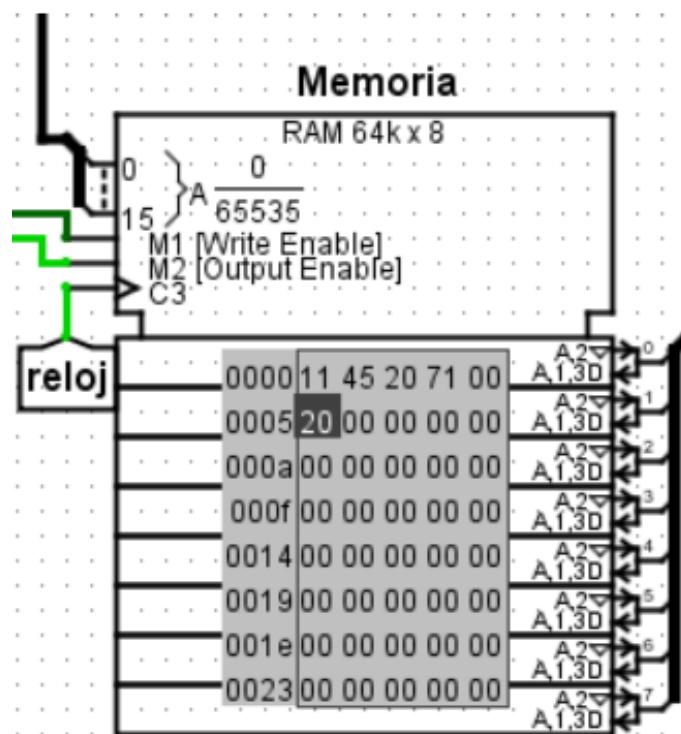
Bit 19: habilita la escritura en MAR.

| | | | |
|-----|-----|-------------------|-----------------|
| ... | 0b2 | 20800001 00221000 | A ₂₁ |
| ... | 0b4 | 20800001 00220800 | A ₂₂ |
| ... | 0b6 | 20800001 00221800 | A ₂₃ |
| ... | 0b8 | 20800001 00080000 | A ₂₄ |
| ... | 0ba | 00002100 00002100 | A ₂₅ |
| ... | 0bc | 00204000 20800001 | A ₂₆ |
| ... | 0be | 00400000 00080000 | A ₂₇ |
| ... | 0c0 | 00002100 00002100 | A ₂₈ |
| ... | 0c2 | 00014000 00400000 | A ₂₉ |
| ... | 0c4 | 00080000 00002100 | A ₃₀ |
| ... | 0c6 | 00002100 00104000 | A ₃₁ |
| ... | 0c8 | 000c0000 01002000 | |
| ... | 0ca | 00004080 00400000 | |

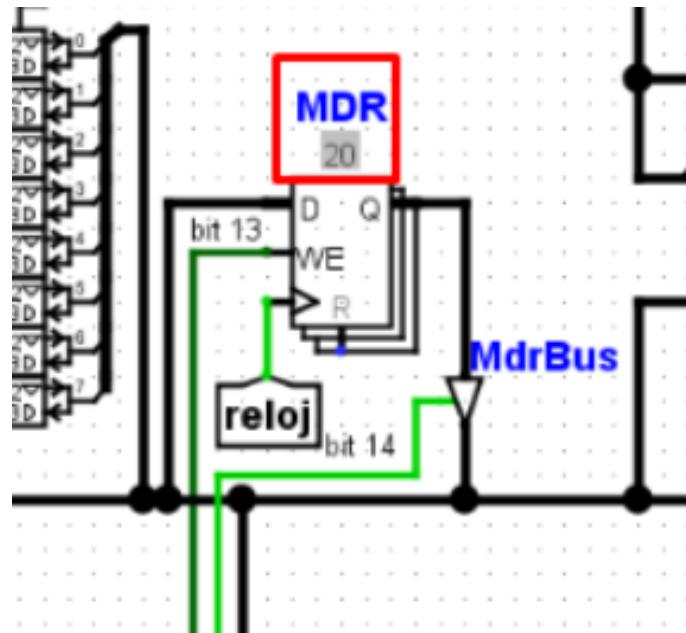
Memoria de control: contenido de la instrucción STA dir 71 bf.



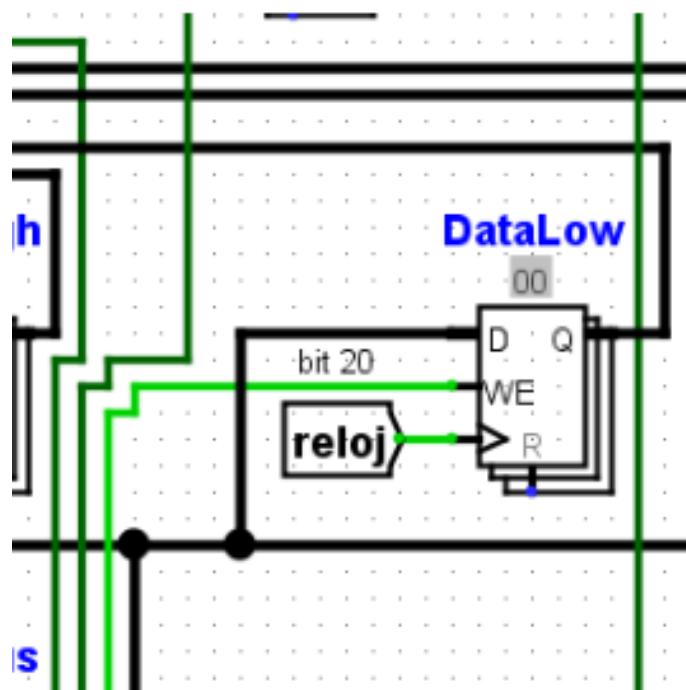
Proceso de búsqueda: MAR <- PC.



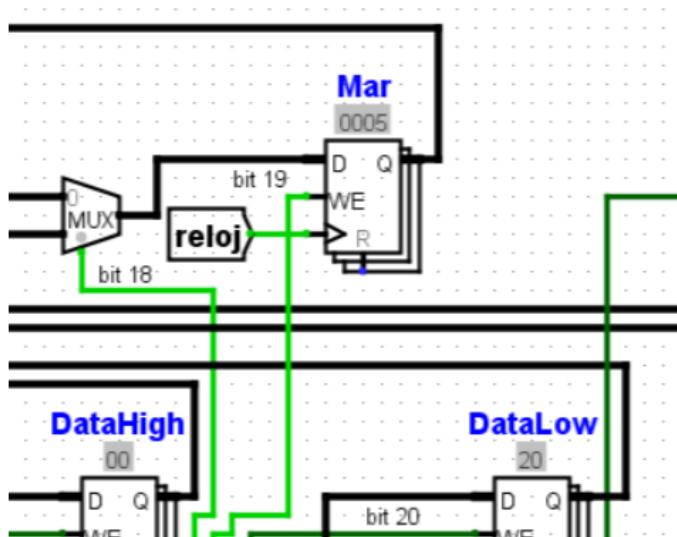
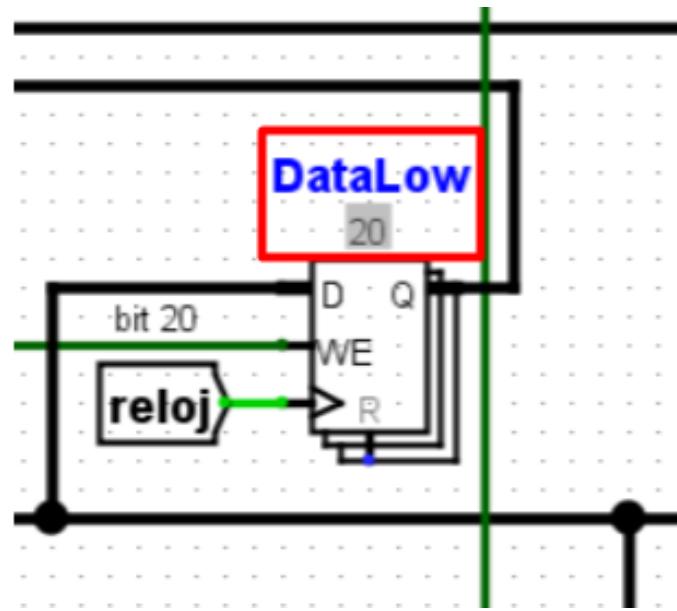
Apunta a la siguiente instrucción del programa que se va a ejecutar.



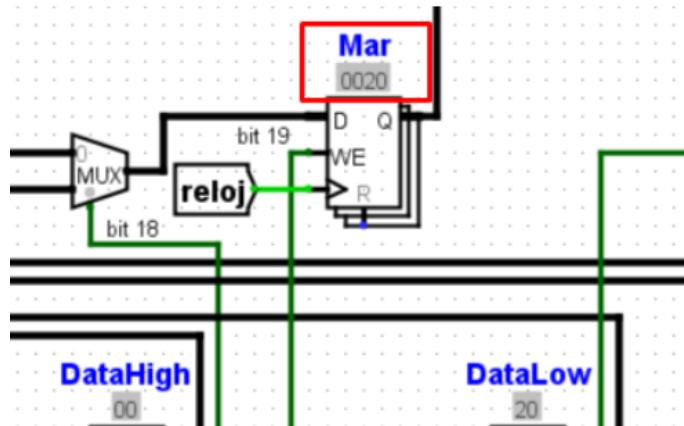
El Registro de Datos de Memoria escribe el código de operación que se está ejecutando
 $MDR \leftarrow mp[MAR]$.



Bit 20: habilita la escritura en DATALOW.



Bit 18: selecciona el contenido de DATAHIGH-DATALOW ó el contenido de PC como entrada al registro MAR. Bit 19: habilita la escritura en MAR.



Bit 29: habilita la salida de la ALU hacia el bus.

10. Apéndice

10.1. Logisim

Logisim es una herramienta educativa de libre distribución, que permite diseñar y simular circuitos lógicos digitales por medio de una sencilla interfaz gráfica de usuario. Por eso permite aprender con facilidad los conceptos básicos relacionados con la lógica de circuitos digitales. Incluso puede emplearse para diseñar y simular CPUs enteros con propósitos educativos.

Es muy utilizado en universidades de todo el mundo, en materias relacionadas a la introducción a la informática u organización y arquitectura de computadoras.

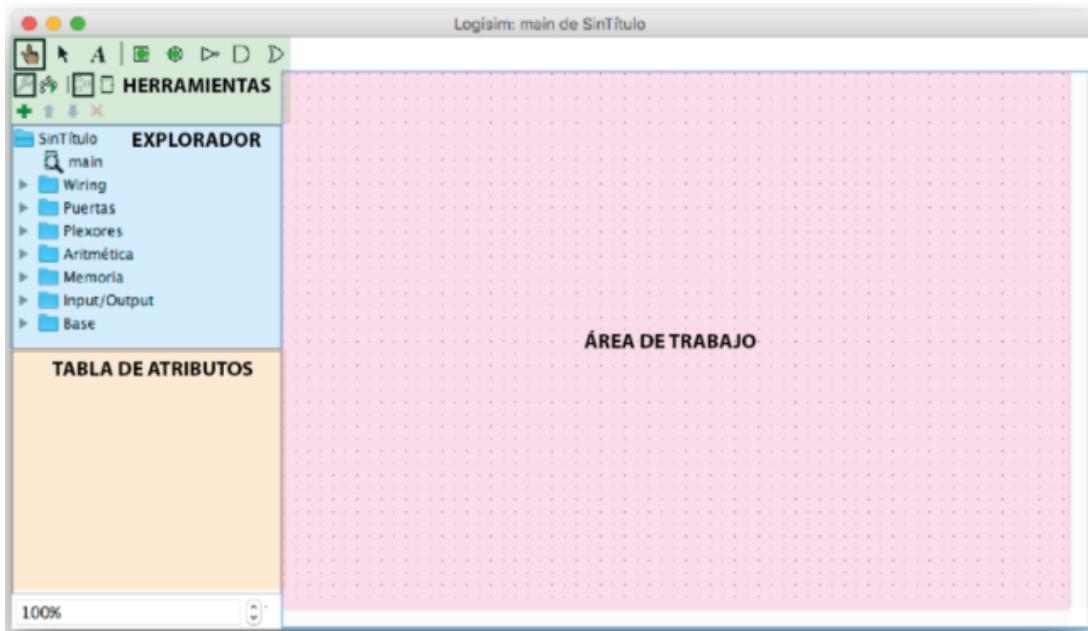


Figura 26: Interfaz de Logisim

11. Conclusiones

Concluimos en que es una buena forma de reforzar los temas vistos en las asignaturas Elementos de Informática y Arquitectura de Computadoras. Nos parece interesante que este circuito de la Máquina plus hecho por varios estudiantes nombrados en el documento, guiados por la cátedra, pueda ser utilizado en futuras clases o cursos de las materias de modo didáctico. Facilitando la comprensión sobre el funcionamiento de un procesador a los estudiantes y que de esta manera puedan ver cómo funciona la Máquina plus internamente e incluso hacer pruebas mediante el cargado de programas en memoria principal.

12. Referencias

1. Documentación de cada estudiante que trabajó sobre el circuito de la Máquina plus y material dado por las cátedras Elementos de Informática y Arquitectura de Computadoras.
2. Logisim (<http://www.cburch.com/logisim/>)
3. Logisim Evolution (<https://github.com/logisim-evolution/logisim-evolution>)
4. Rafael Asenjo Plaza, Eladio Gutiérrez Carrasco y Julián Ramos Córzar (2001). Fundamentos de los Computadores. Universidad de Málaga.

5. Jose M Angulo Usategui y Pedro de Miguel Anasagasti (1987). Arquitectura de Computadores. Paraninfo.
6. Enrique Mandado (1998). Sistemas Electrónicos Digitales. Marcombo.
7. William Stallings (2006). Organización y Arquitectura de Computadores. Pearson.
8. Andrew S. Tanenbaum (1999). Organización de Computadoras, un enfoque estructurado. Pearson.
9. Stalling Appendices. (<https://www.ecs.csun.edu/~cputnam/Comp546/Stallings-Appendices/20-Micropogrammed.pdf>)