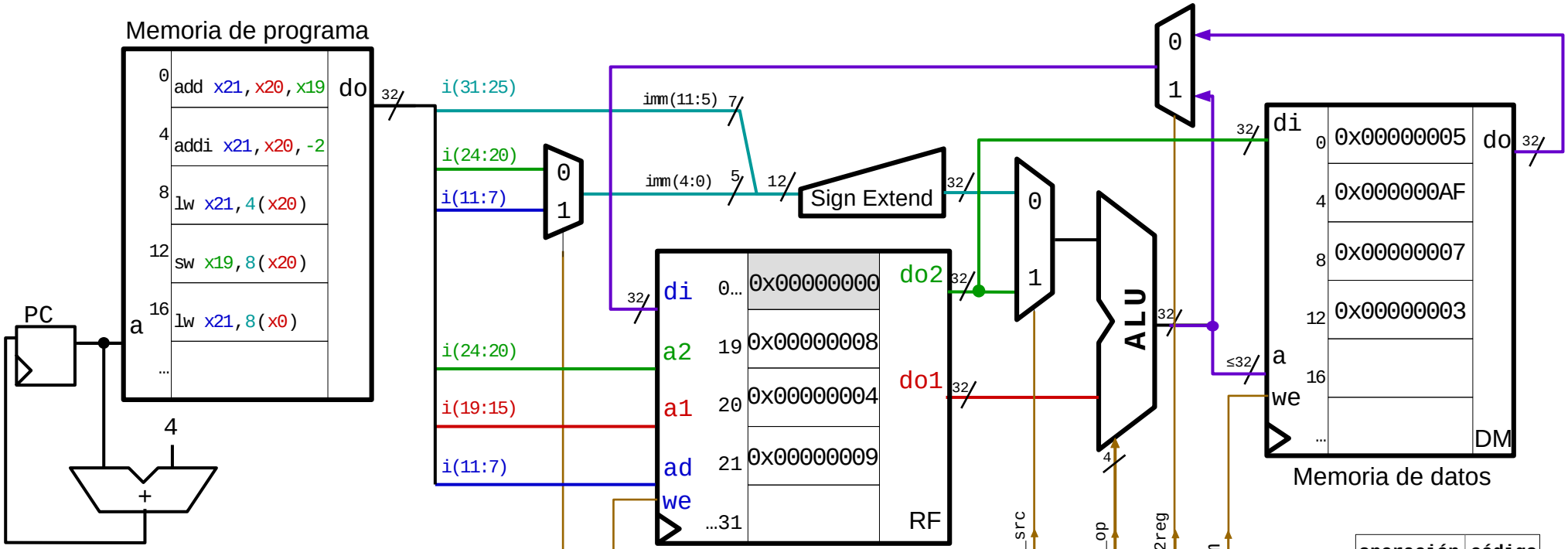
Diseño básico de un procesador RISC-V

Instrucciones de control (branches)

Diseño de Sistemas Electrónicos

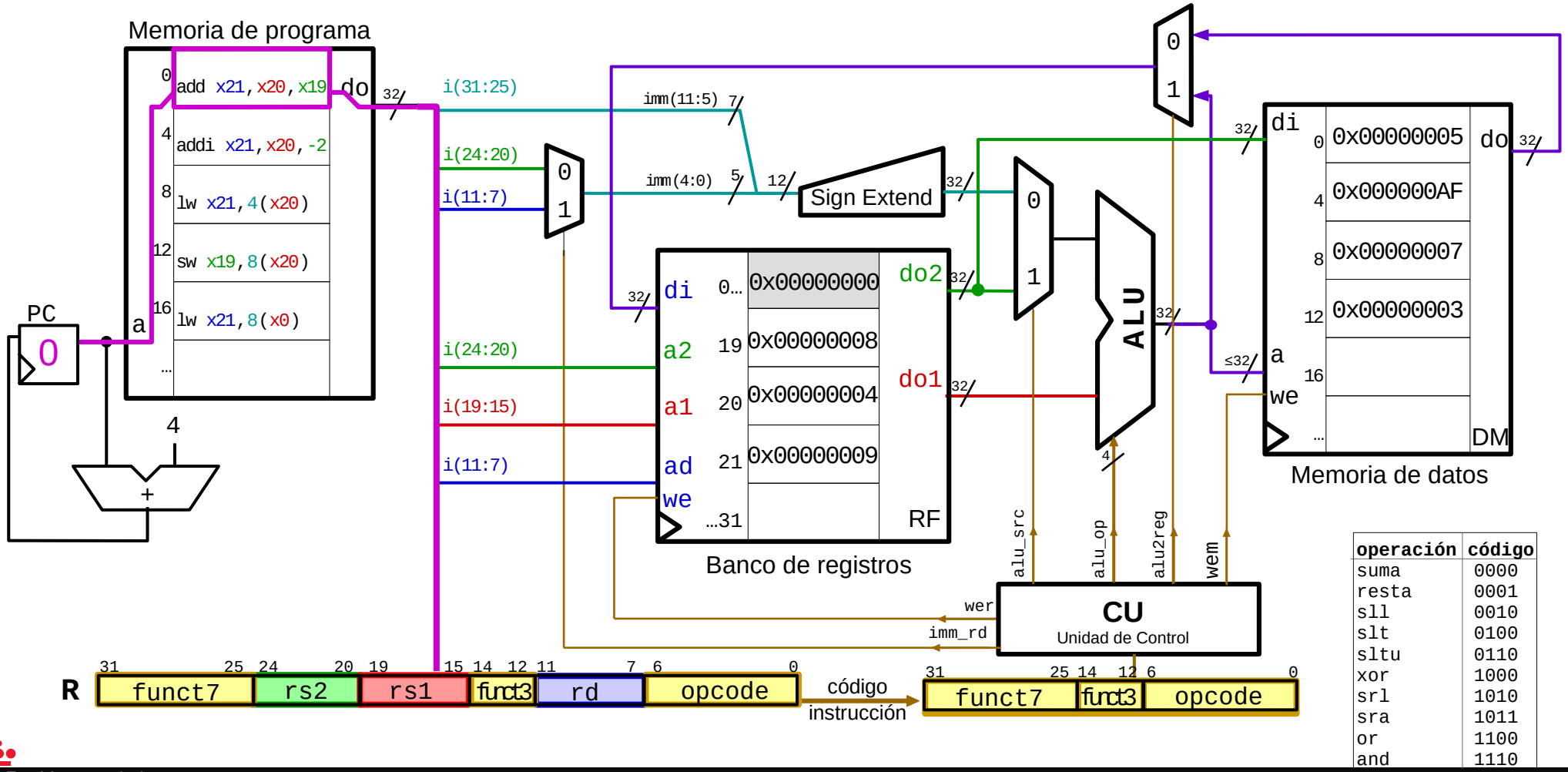
Felipe Machado

Procesador monociclo (R, I, *load*, *store*)



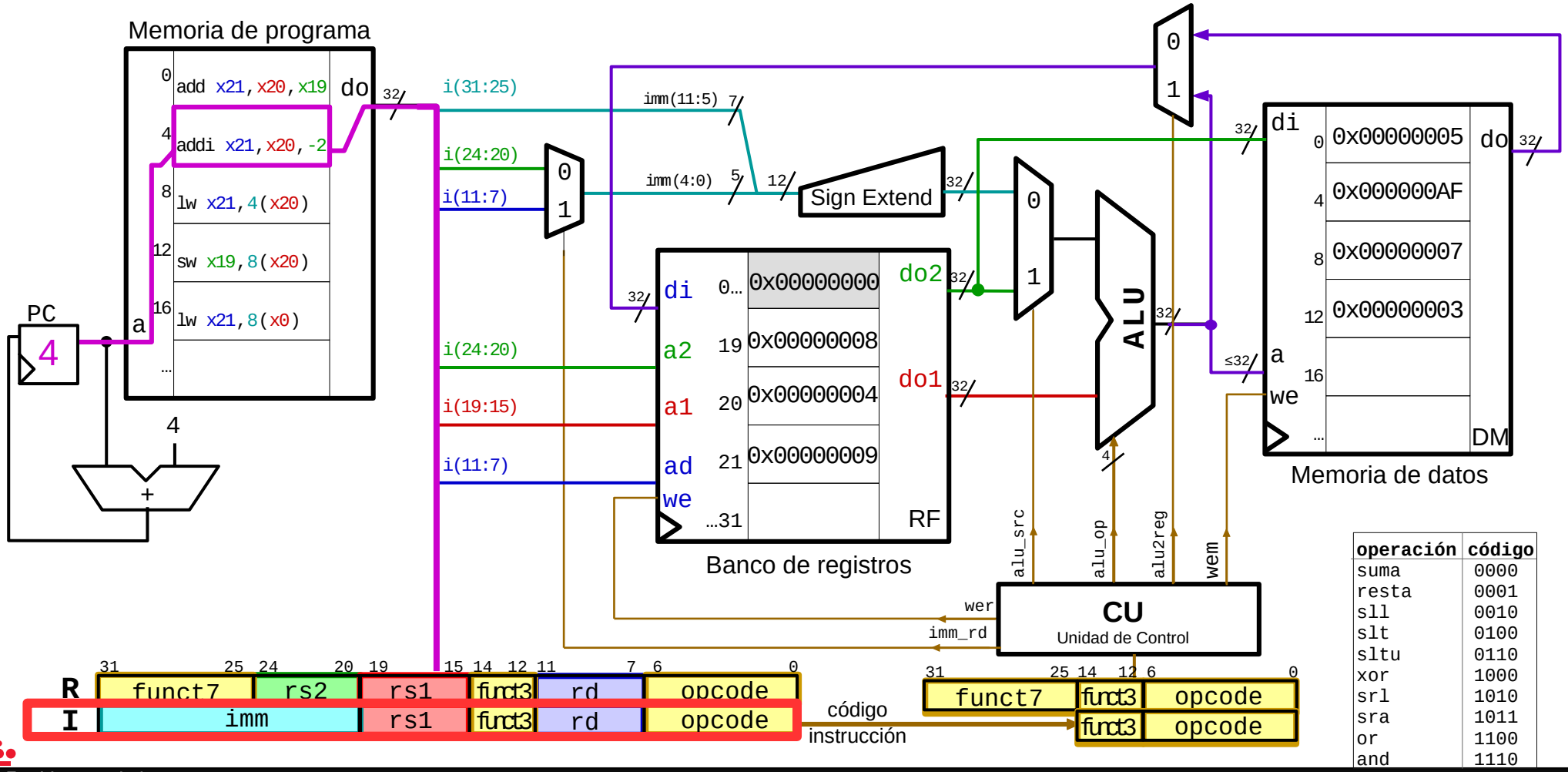
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Procesador monociclo (R, I, *load*, *store*)



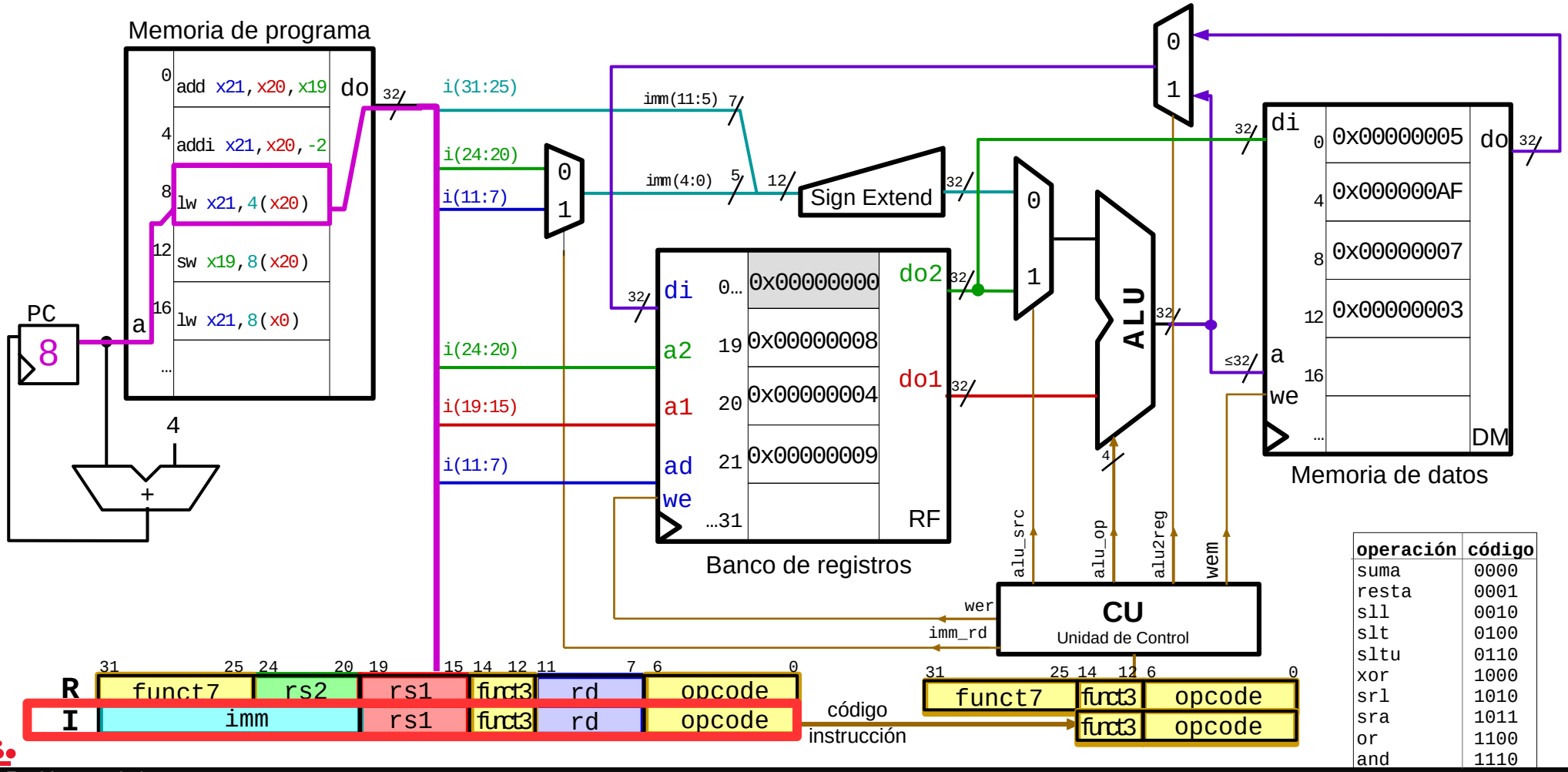
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Procesador monociclo (R, I, *load*, *store*)



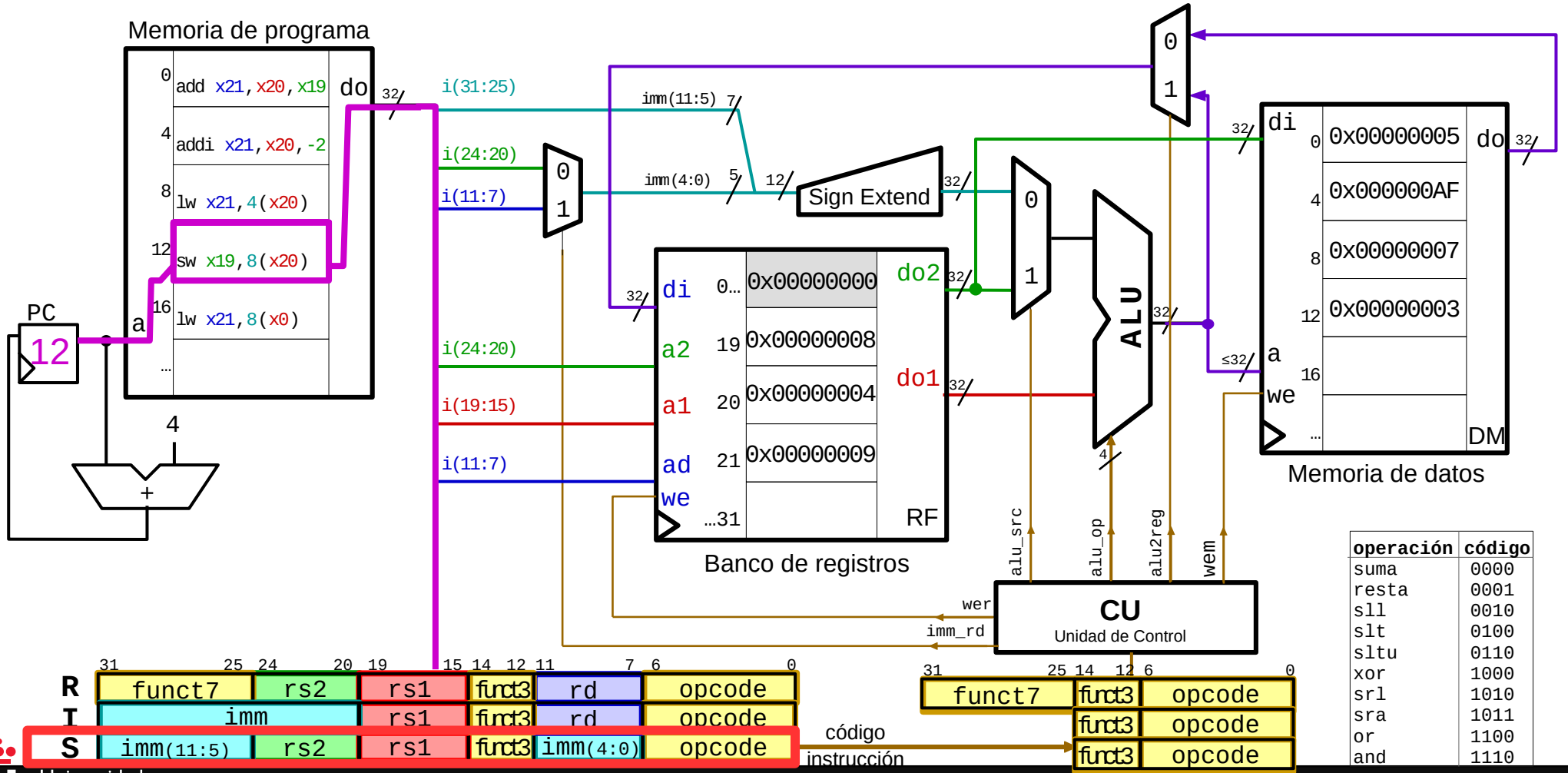
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Procesador monociclo (R, I, *load*, *store*)



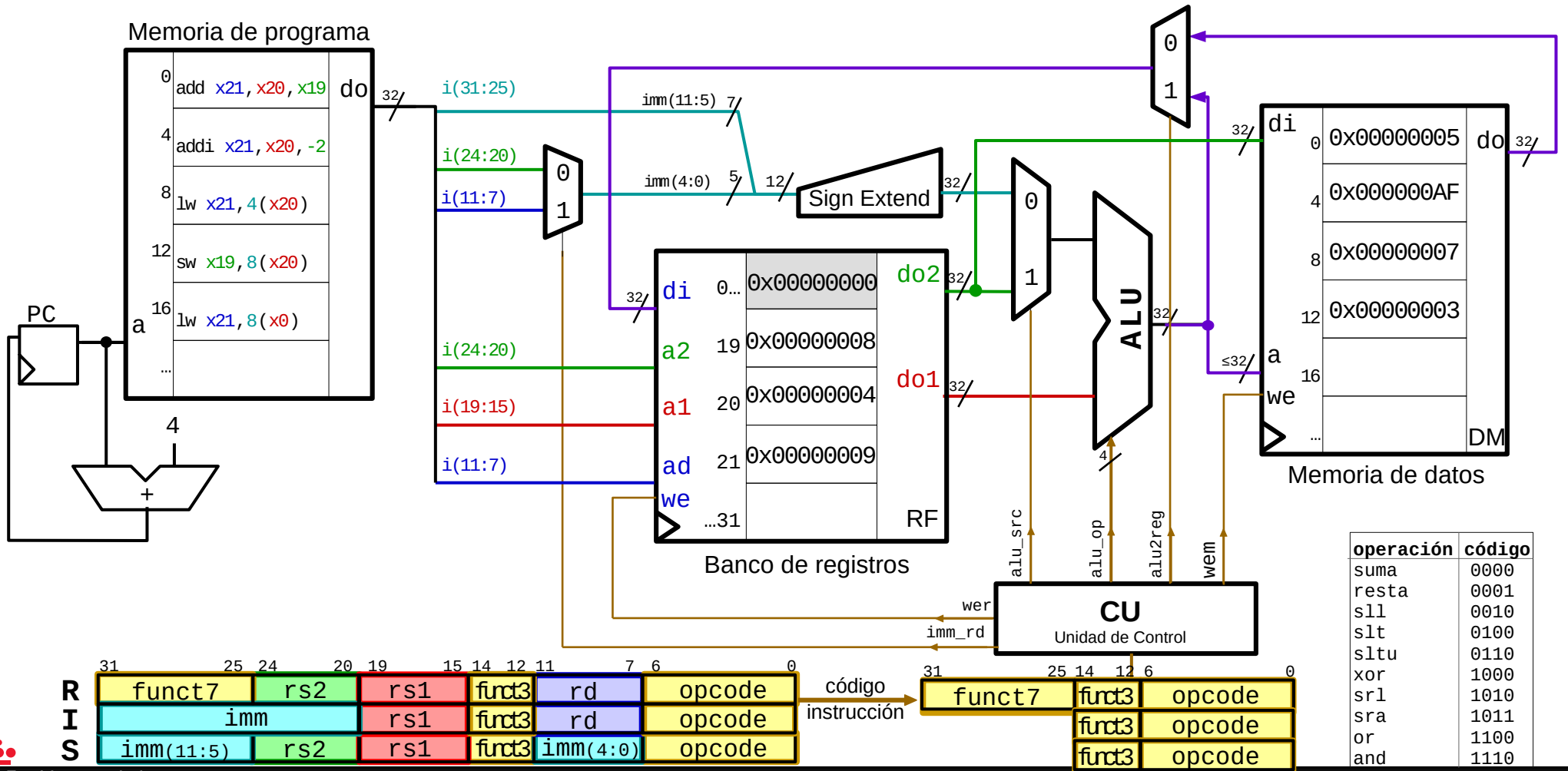
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Procesador monociclo (R, I, *load*, *store*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Procesador monociclo (R, I, *load*, *store*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones tipo R e I

	Tipo R	Tipo I	
Add	`add`	`addi`	Aritméticas
Subtract	`sub`	—	
AND	`and`	`andi`	A nivel de bits
OR	`or`	`ori`	
Exclusive OR	`xor`	`xori`	
Set if Less Than	`slt`	`slti`	Comparaciones
Set if Less Than Unsigned	`sltu`	`sltiu`	
Shift Left Logic	`sll`	`slli`	Desplazamientos
Shift Right Logic	`srl`	`srli`	
Shift Right Arithmetic	`sra`	`srai`	

19 instrucciones

Instrucciones de transferencia de datos

Tipo I: Carga de memoria a registro	
`lw`	*Load Word*
`lhu`	*Load Halfword, Unsigned*
`lbu`	*Load Byte, Unsigned*
`lh`	*Load Halfword*
`lb`	*Load Byte*

Tipo S: Almacenamiento, de registro a memoria	
`sw`	*Store Word*
`sh`	*Store Halfword*
`sb`	*Store Byte*

8 instrucciones

Instrucciones de control

6 instrucciones

Tipo B		
`beq`	*Branch if Equal*	`rs1 == rs2`
`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`
`blt`	*Branch if Less Than*	$rs1 <_s rs2$
`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$
`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$
`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$

No son necesarias:

`bgt`	*Branch if Greater Than*	$rs1 >_s rs2$
`ble`	*Branch if Less than or Equal*	$rs1 \leq_s rs2$
`bgtu`	*Branch if Greater Than, Unsigned*	$rs1 >_u rs2$
`bleu`	*Branch if Less than or Equal, Unsigned*	$rs1 \leq_u rs2$

Instrucciones de control (*branch*)

6 instrucciones

Tipo B		
`beq`	*Branch if Equal*	`rs1 == rs2`
`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`
`blt`	*Branch if Less Than*	$rs1 <_s rs2$
`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$
`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$
`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$

```
blt x19, x20, offset
```
si x19 $<_s$ x20 → salta

```
blt x20, x19, offset
```
si x19 $>_s$ x20 → salta

No hace falta: `bgt` *Branch if Greater Than*

Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1
Registro fuente 2

si rs1 menor → salto del programa
si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

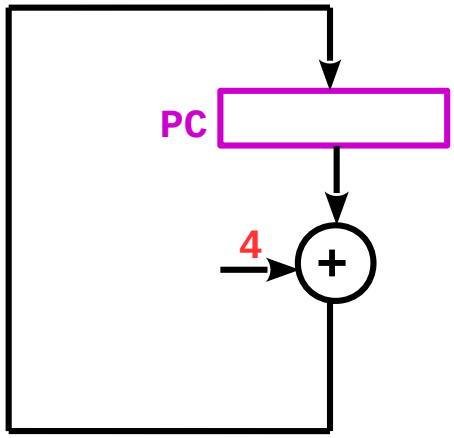
desplazamiento

\+ Contador programa (PC)

Dirección de memoria del salto

si rs1 menor

se guarda en PC



Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1
Registro fuente 2

si rs1 menor → salto del programa
si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

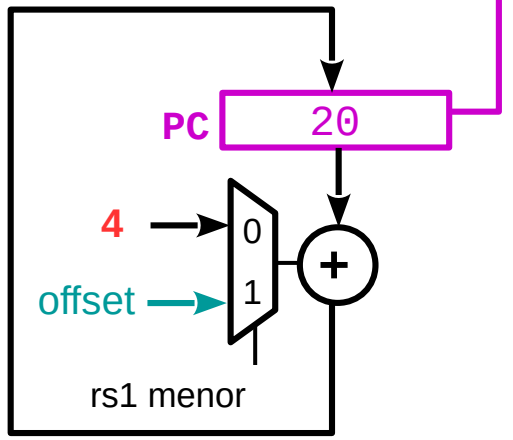
desplazamiento

\+ Contador programa (PC)

Dirección de memoria del salto

si rs1 menor

se guarda en PC



Memoria

20	`blt x22, x23, Else`
24	`add x19, x20, x21`
28	`beq x0, x0, Exit`
32	`sub x19, x20, x21`
36	
40	
...	

Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1
Registro fuente 2

si rs1 menor → salto del programa
si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

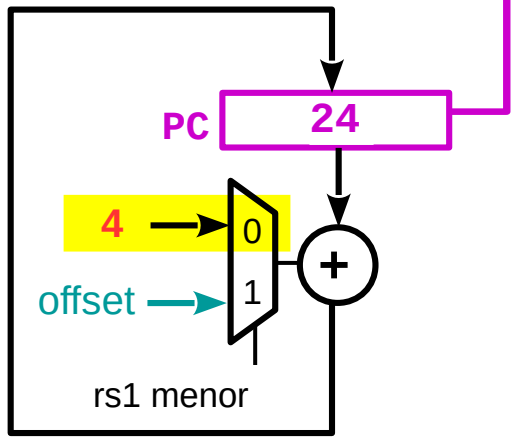
desplazamiento

\+ Contador programa (PC)

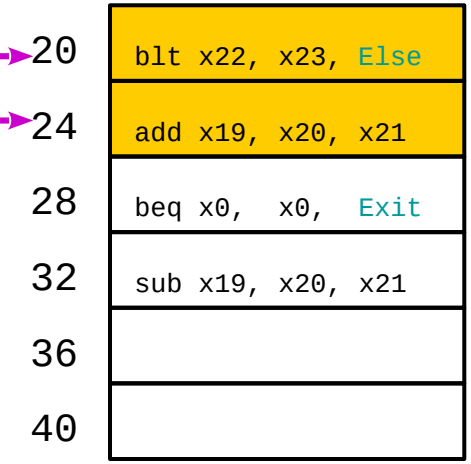
Dirección de memoria del salto

si rs1 menor

se guarda en PC



Memoria



Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1

Registro fuente 2

si rs1 menor → salto del programa

si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

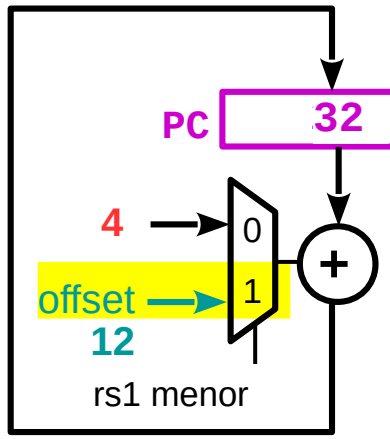
desplazamiento

\+ Contador programa (PC)

Dirección de memoria del salto

si rs1 menor

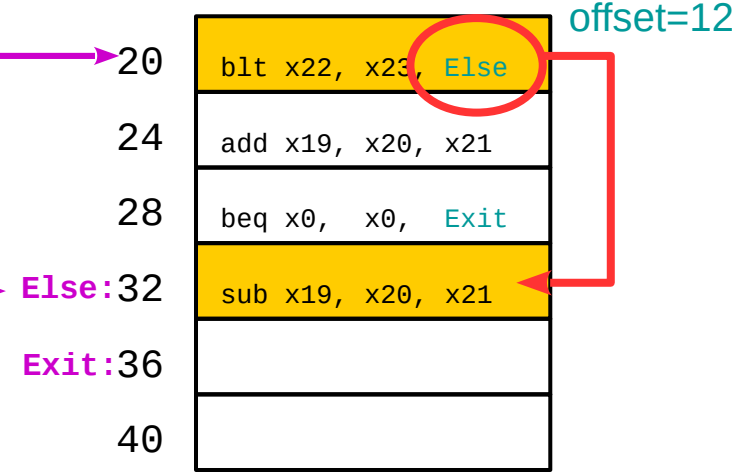
se guarda en PC



Memoria

3 instrucciones

offset=12



Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1

Registro fuente 2

si rs1 menor → salto del programa

si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

desplazamiento

\+ Contador programa (PC)

Dirección de memoria del salto

si rs1 menor

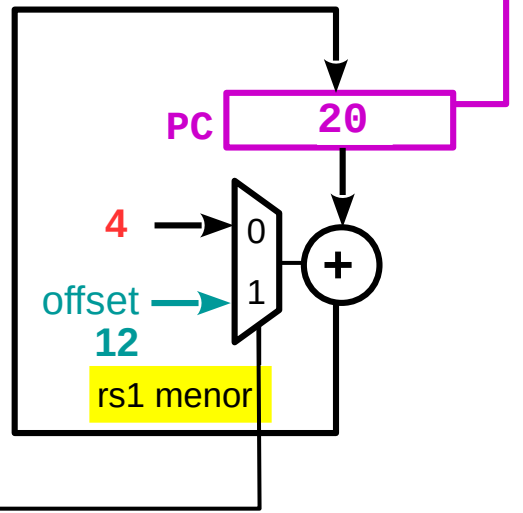
se guarda en PC

rs1 rs2

comparador (con signo)

rs1 menor

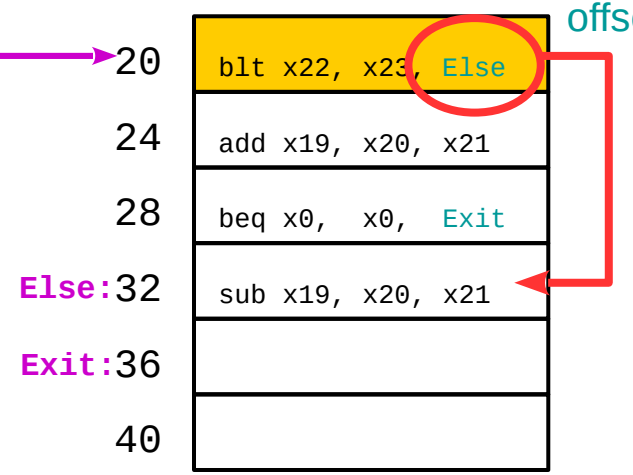
Necesito un comparador



Memoria

3 instrucciones

offset=12



Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1
Registro fuente 2

si rs1 menor → salto del programa
si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

desplazamiento

\+ Contador programa (PC)

Dirección de memoria del salto

si rs1 menor

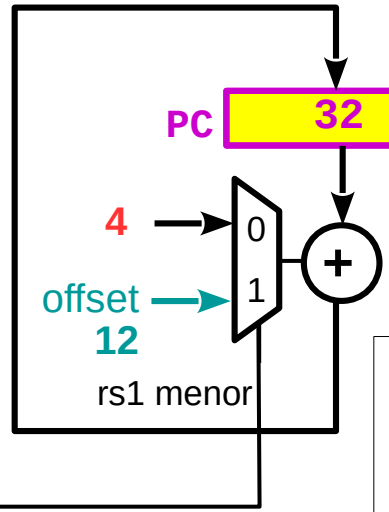
rs1 rs2

comparador (con signo)

rs1 menor

se guarda en PC

Necesito un comparador



Memoria

3 instrucciones

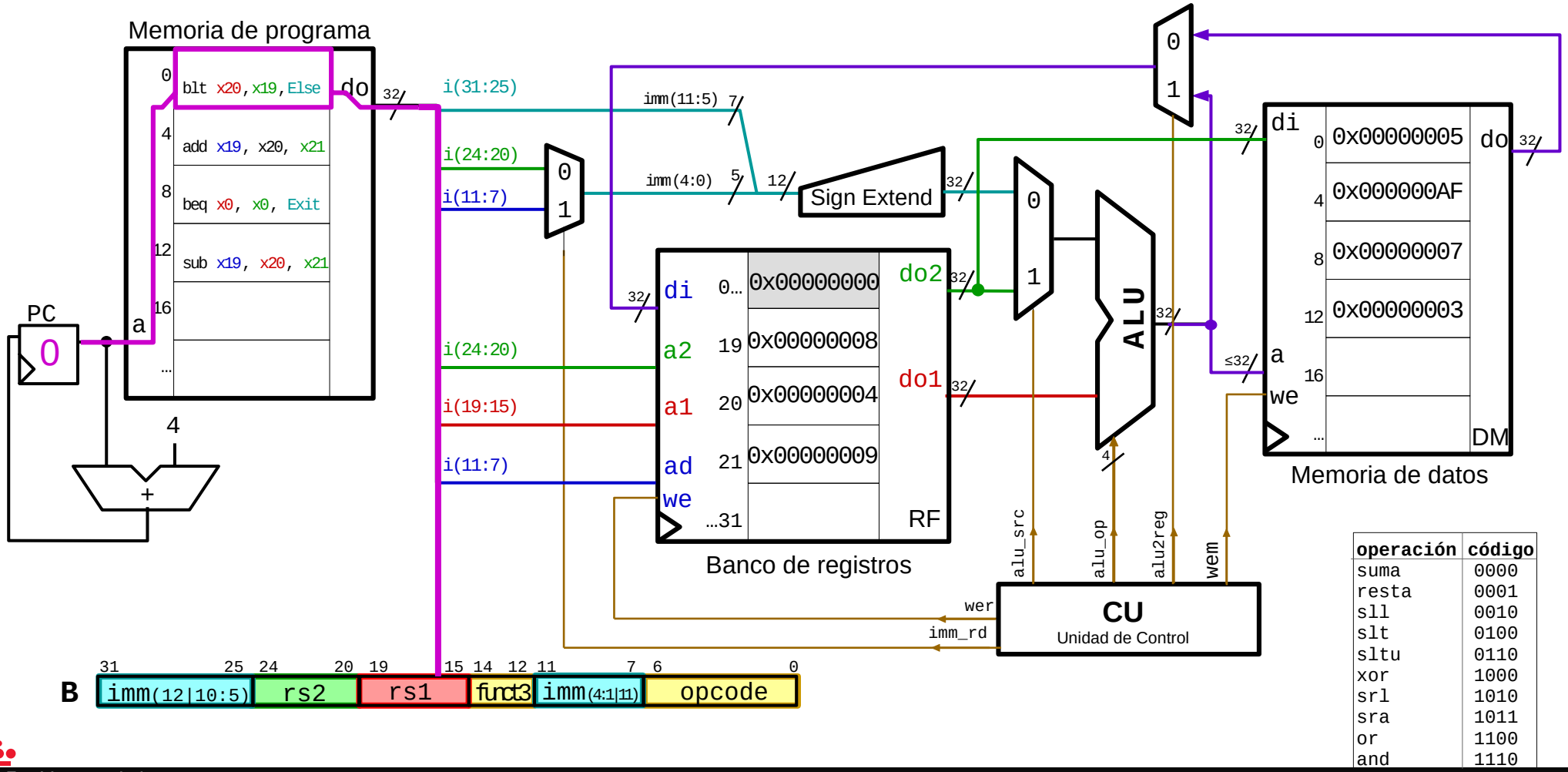
offset=12

20	blt x22, x23, Else
24	add x19, x20, x21
28	beq x0, x0, Exit
Else: 32	sub x19, x20, x21
Exit: 36	
40	

Modifica PC

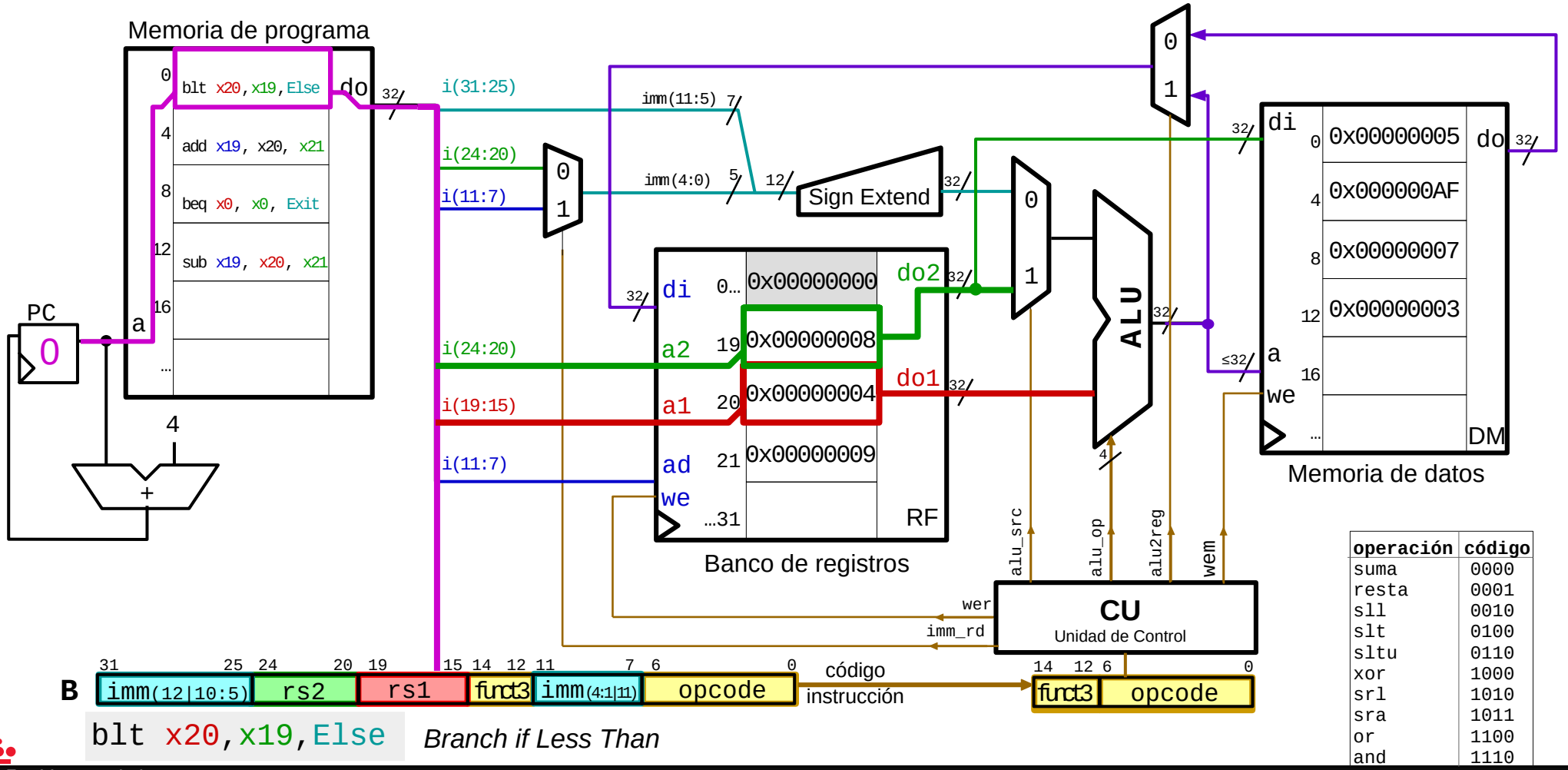
- **Sin registro destino** — *Igual que instrucciones tipo S (store)*
- **No almacena en memoria** — *Diferencia con instrucciones tipo S (store)*
- **No usa memoria** — *Ni carga ni almacena*

Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

`blt x20,x19,Else` *Branch if Less Than*

Instrucciones de control (*branch*)

Saltar si rs1 menor (con signo)

Registro fuente 1
Registro fuente 2

si rs1 menor → salto del programa
si rs1 no menor → siguiente instrucción (+4)

```
blt rs1, rs2, offset
```

Branch if Less Than

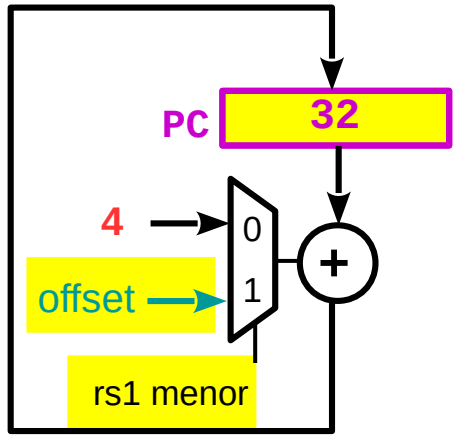
desplazamiento

\+ Contador programa (PC)

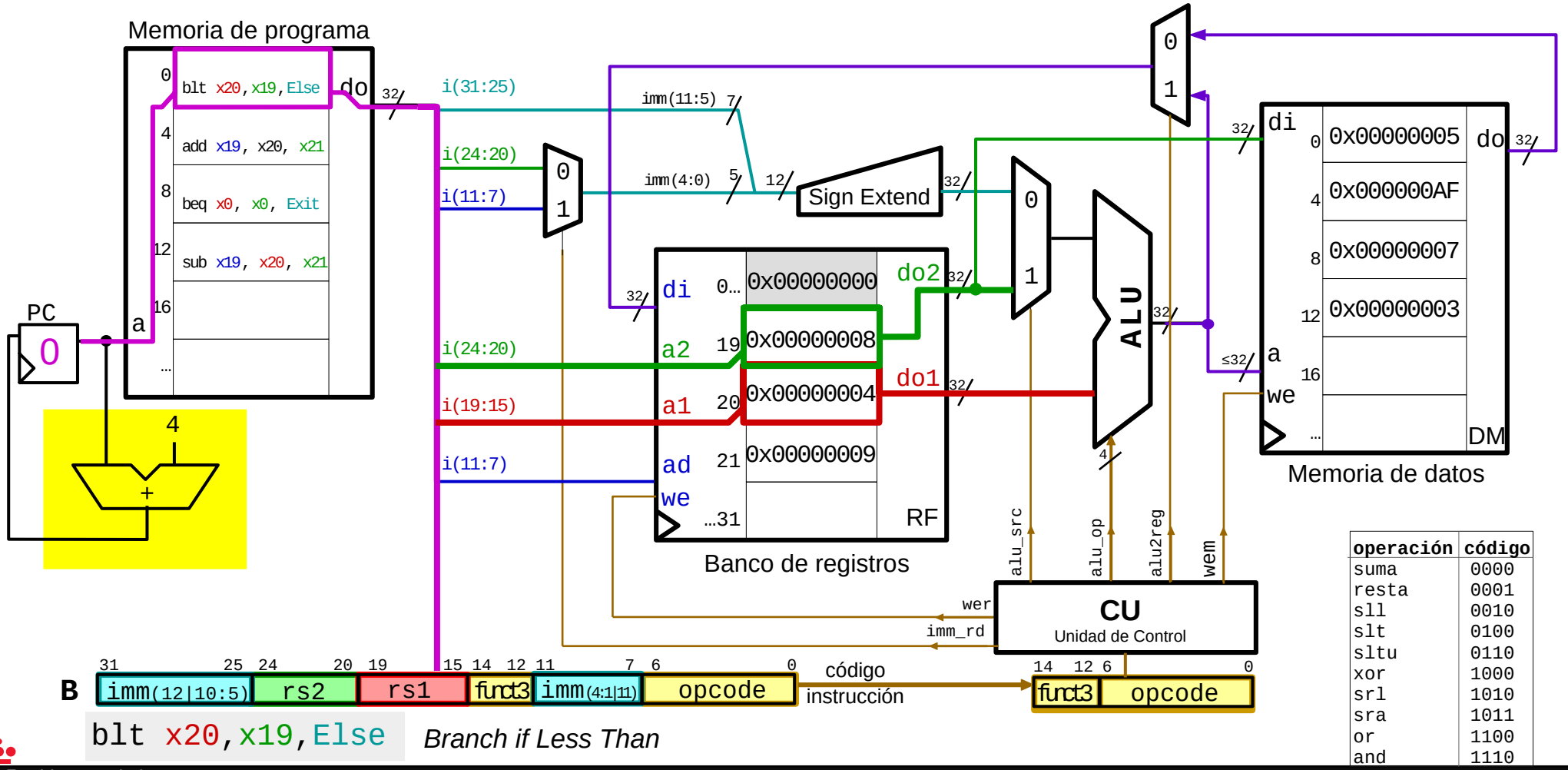
Dirección de memoria del salto

si rs1 menor

se guarda en PC



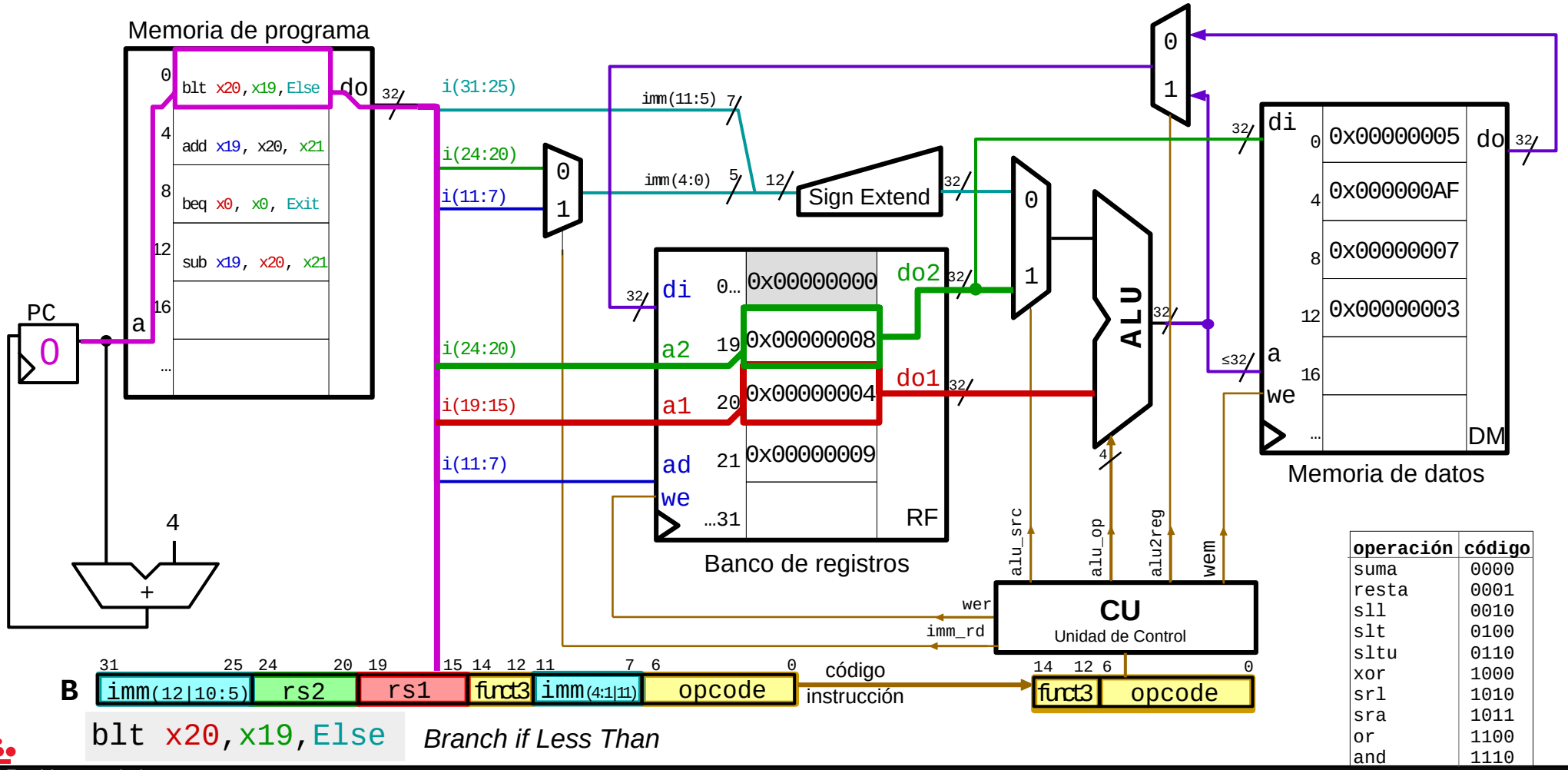
Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

`blt x20,x19,Else` *Branch if Less Than*

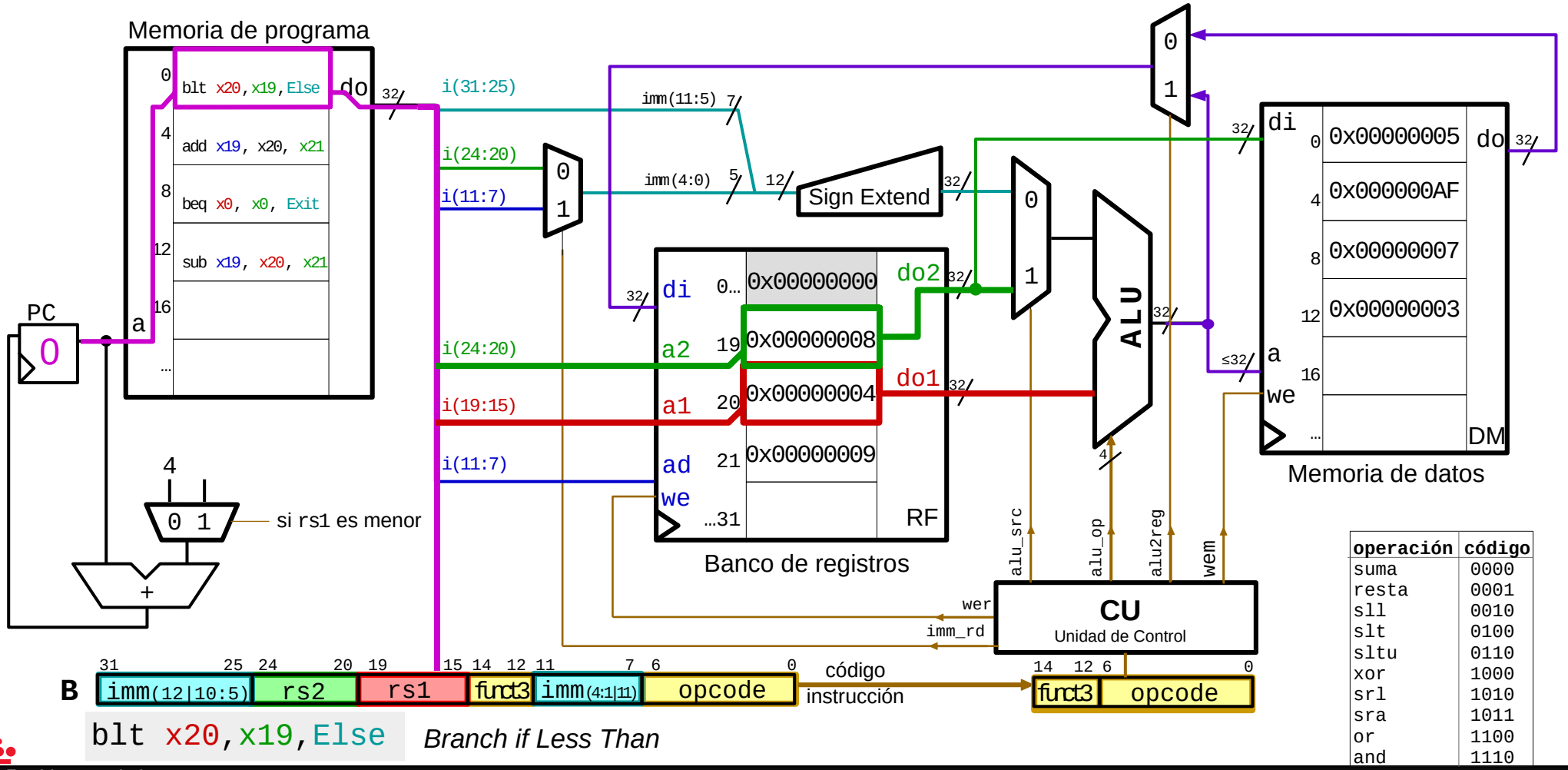
Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

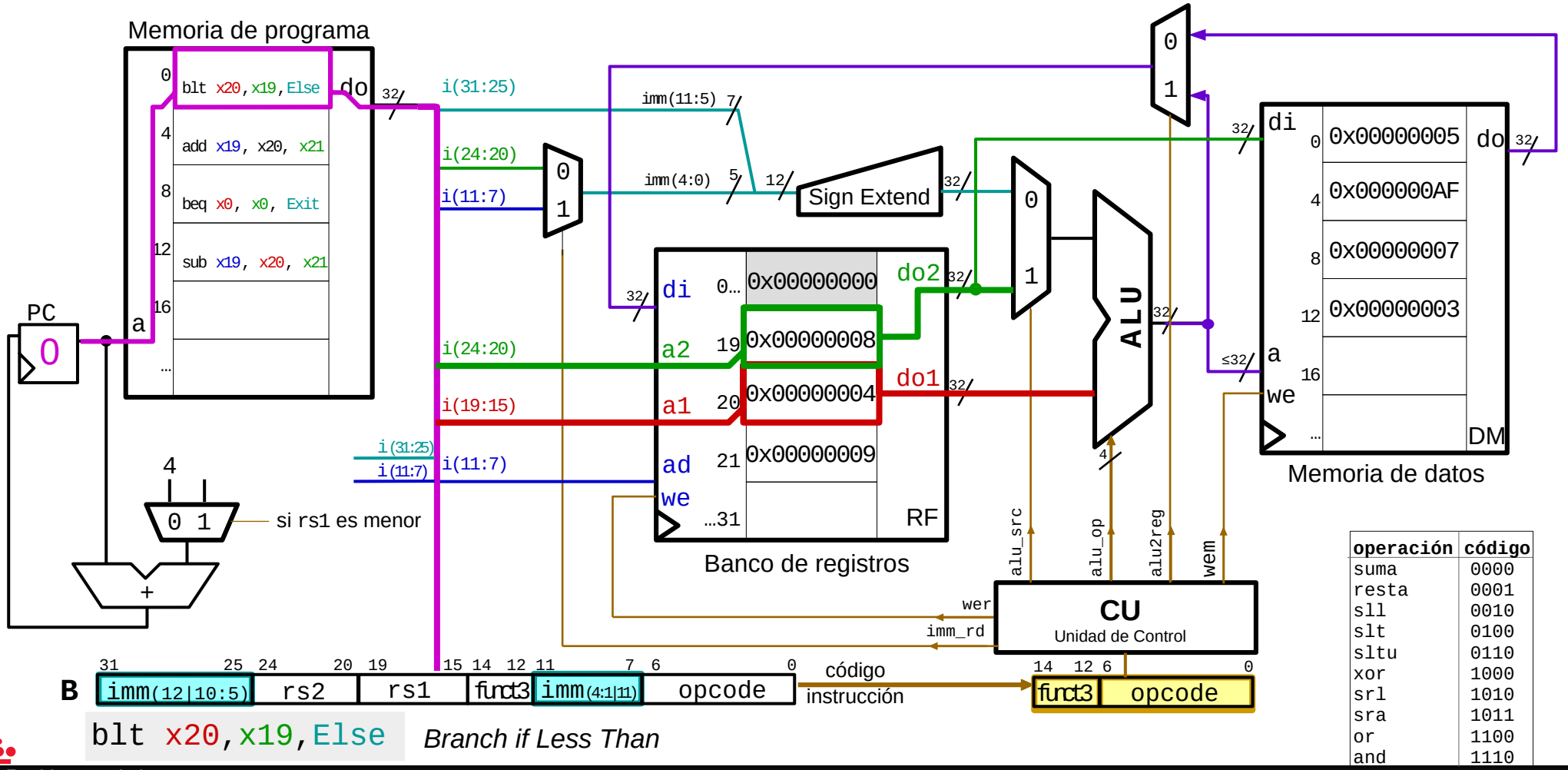
`blt x20,x19,Else` *Branch if Less Than*

Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

```
blt x20,x19,Else
```

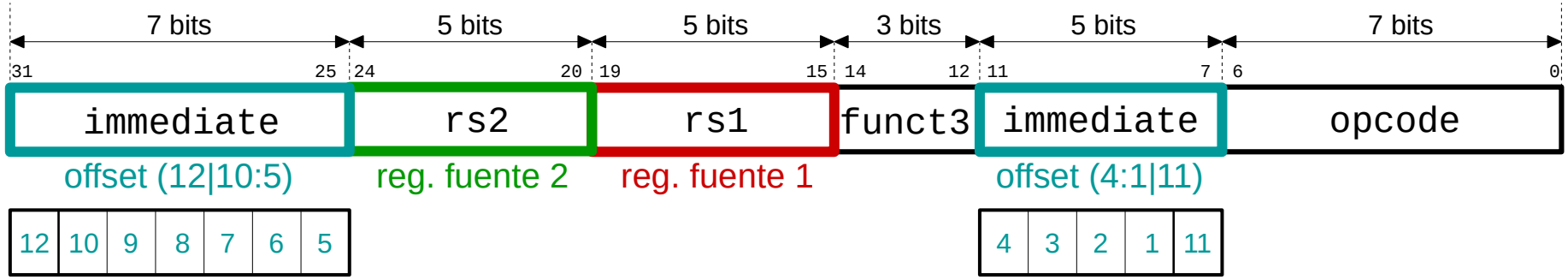
Branch if Less Than

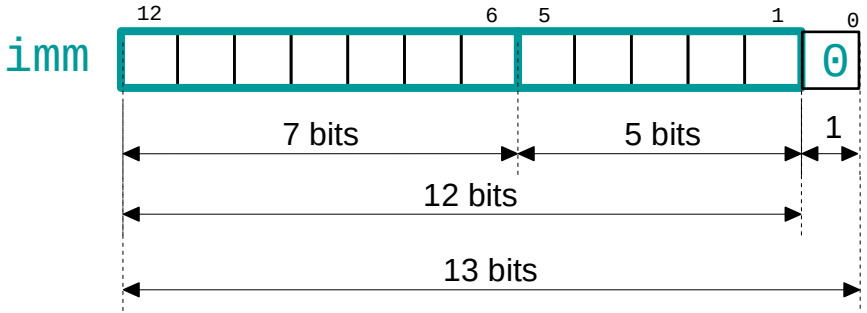
Instrucciones de control

`blt rs1, rs2, offset`

Branch if Less Than

Formato instrucción tipo B

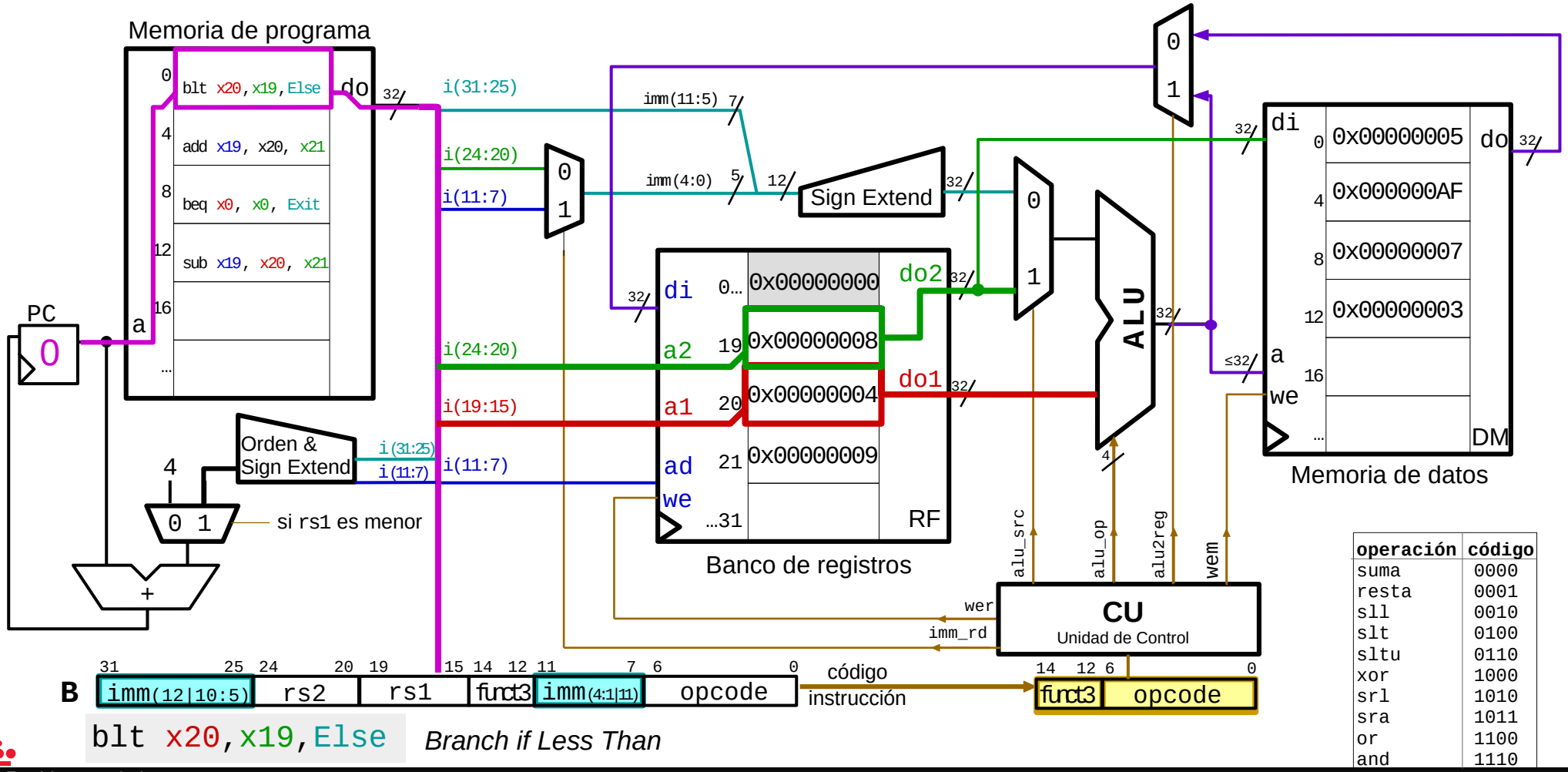




en C'2 → $[-2^{12}, 2^{12}-1] = [-4096, 4095]$

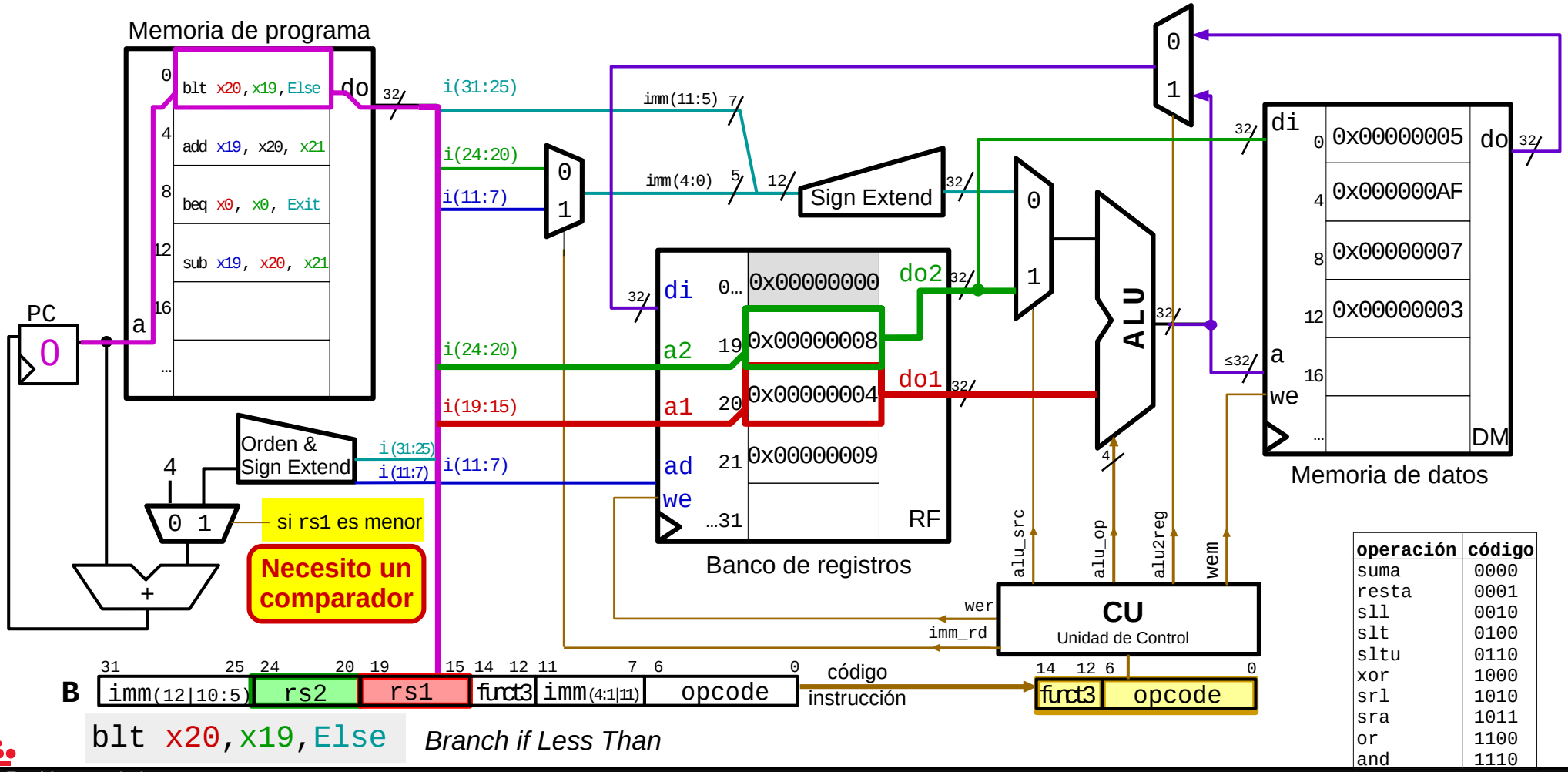
saltos relativos

Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

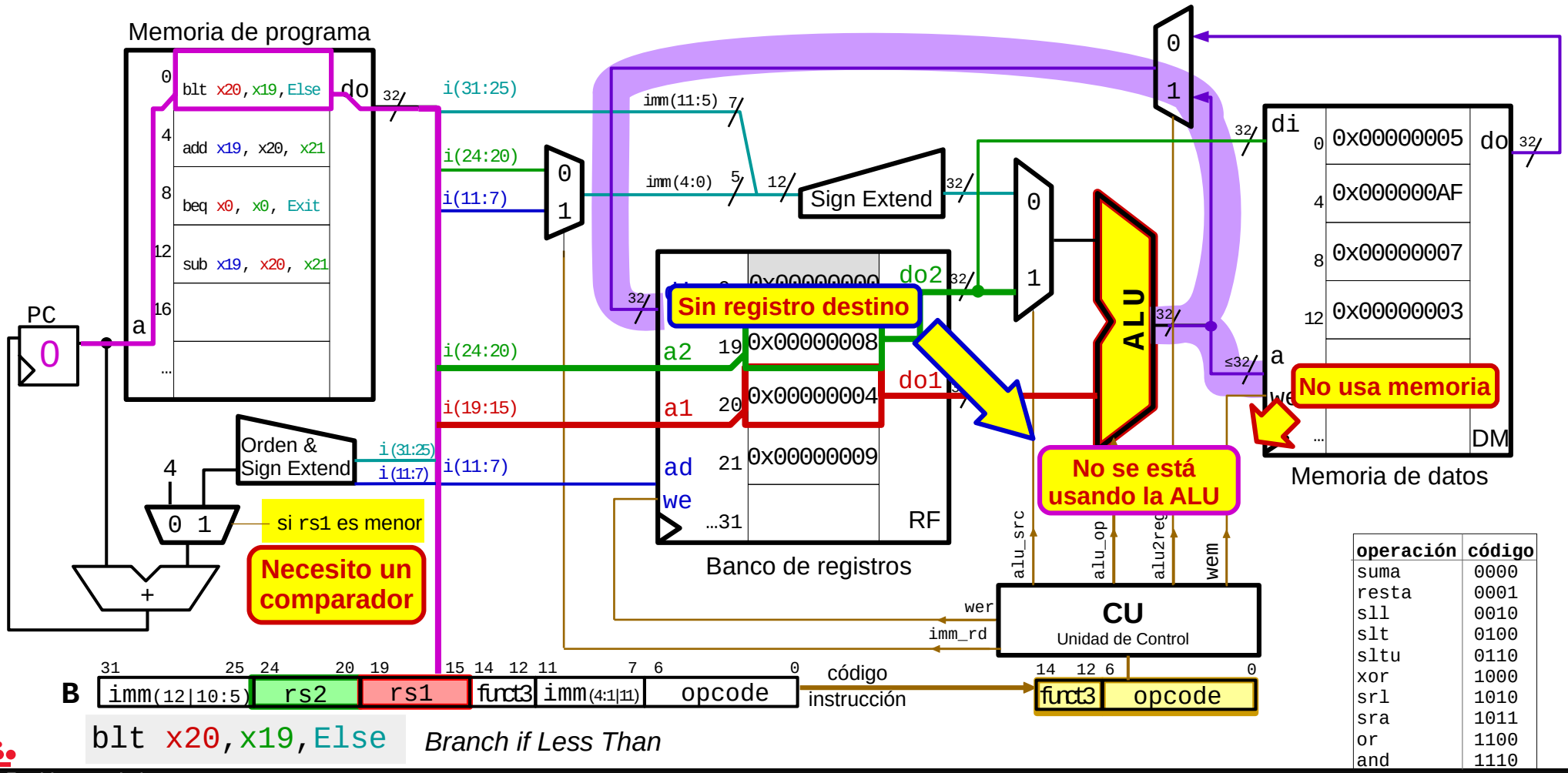
Instrucciones de control (*branch*)



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

```
blt x20,x19,Else
```

Branch if Less Than

Instrucciones tipo R e I

	Tipo R	Tipo I	
Add	`add`	`addi`	Aritméticas
Subtract	`sub`	—	
AND	`and`	`andi`	A nivel de bits
OR	`or`	`ori`	
Exclusive OR	`xor`	`xori`	
Set if Less Than	`slt`	`slti`	Comparaciones
Set if Less Than Unsigned	`sltu`	`sltiu`	
Shift Left Logic	`sll`	`slli`	Desplazamientos
Shift Right Logic	`srl`	`srli`	
Shift Right Arithmetic	`sra`	`srai`	

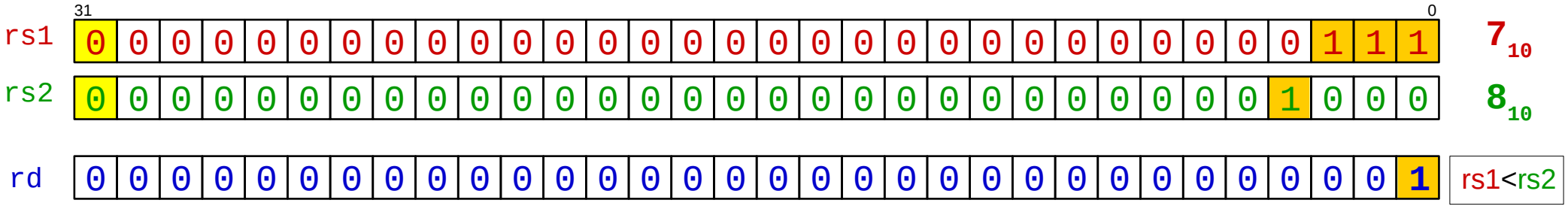
Comparador

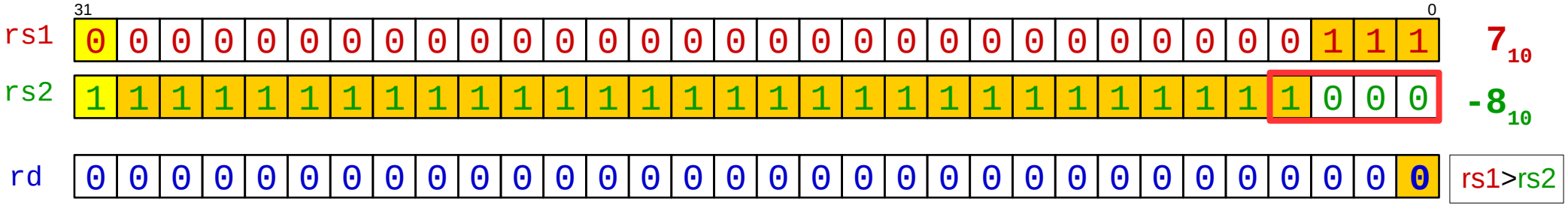
Menor que

`slt rd, rs1, rs2 #rd ← 1 si rs1 < rs2 si no 0`

Set if Less Than

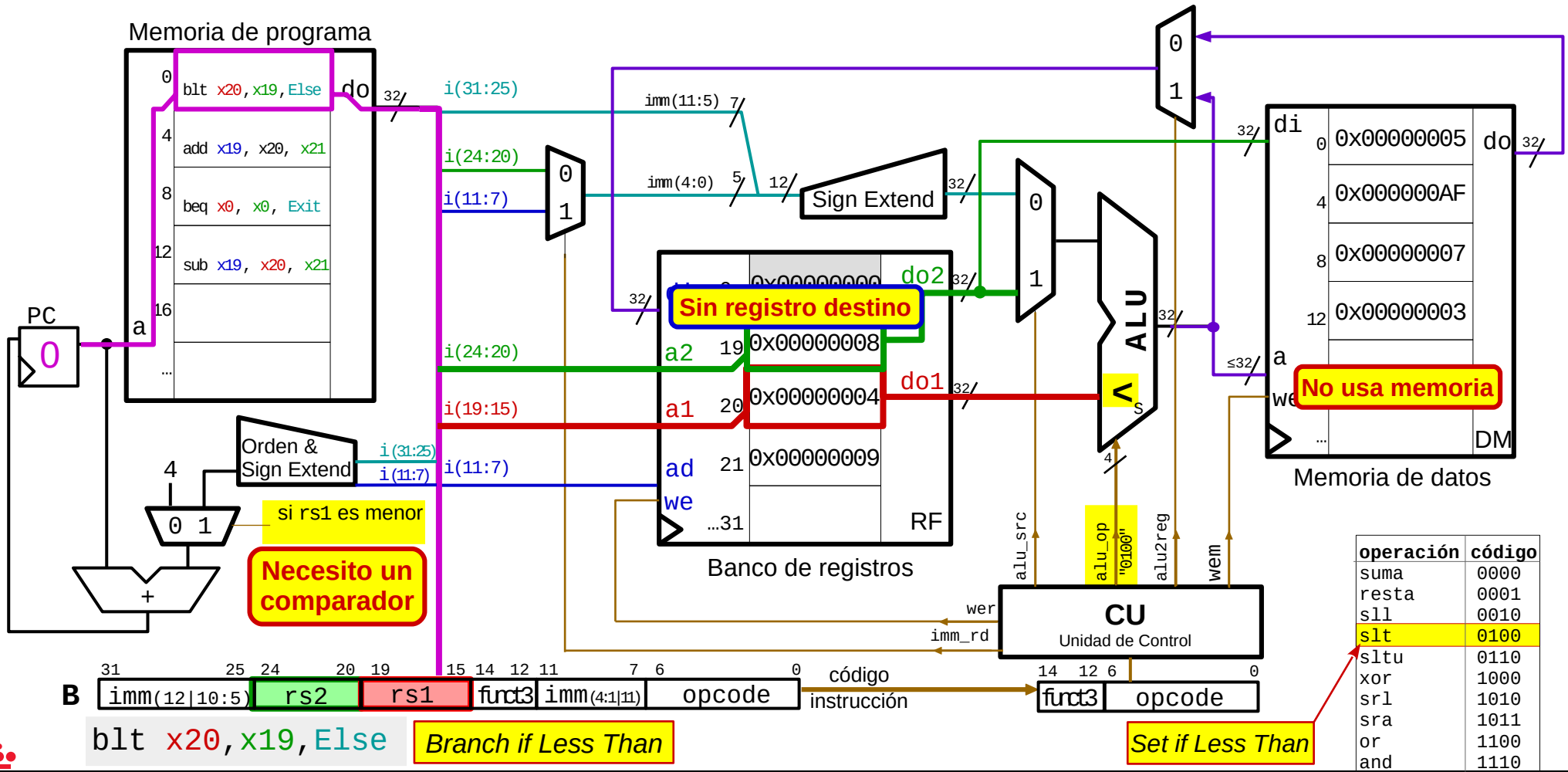
Compara rs1 y rs2 como números en complemento a 2





Instrucciones de control (*branch*)

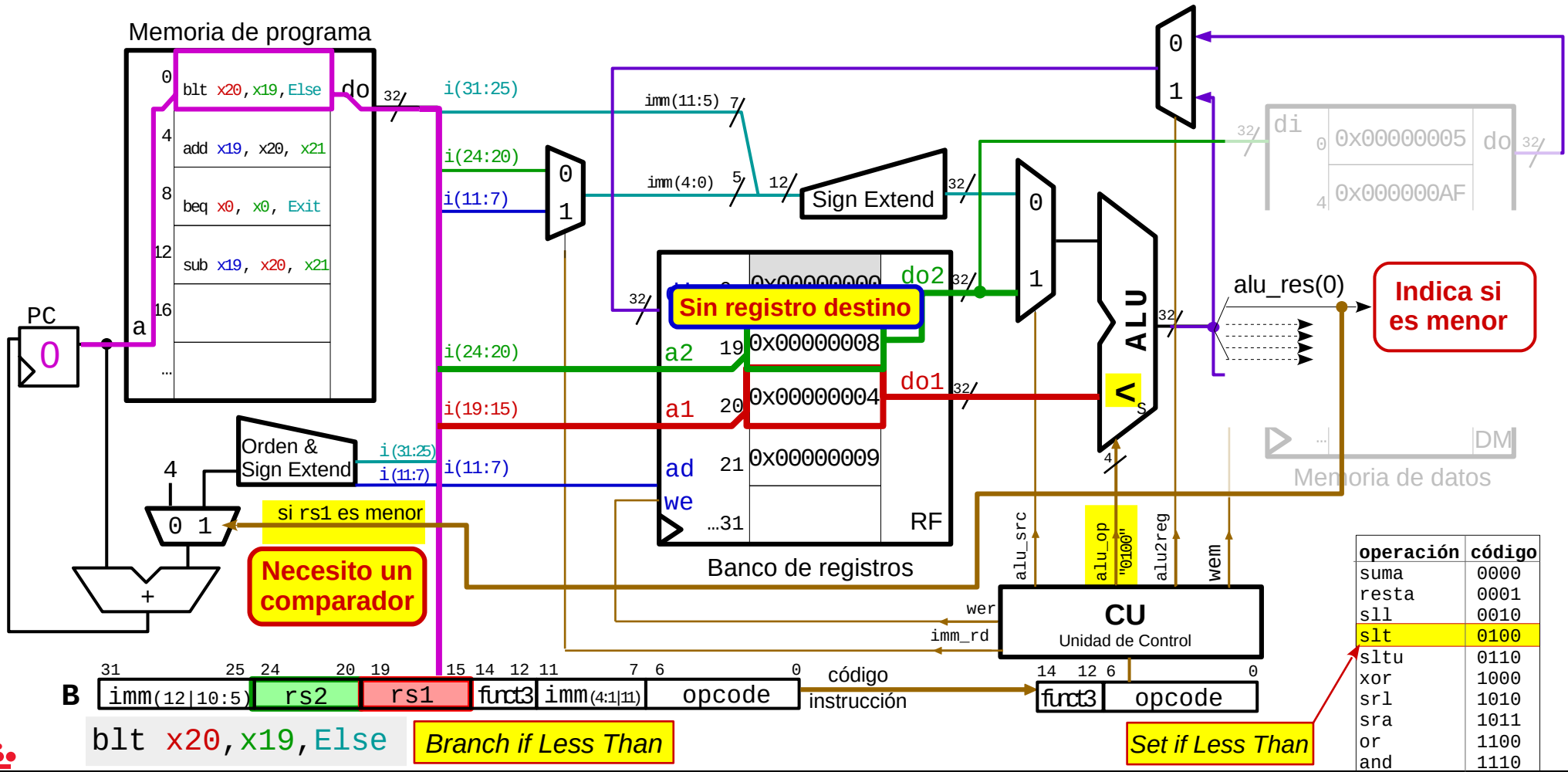
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

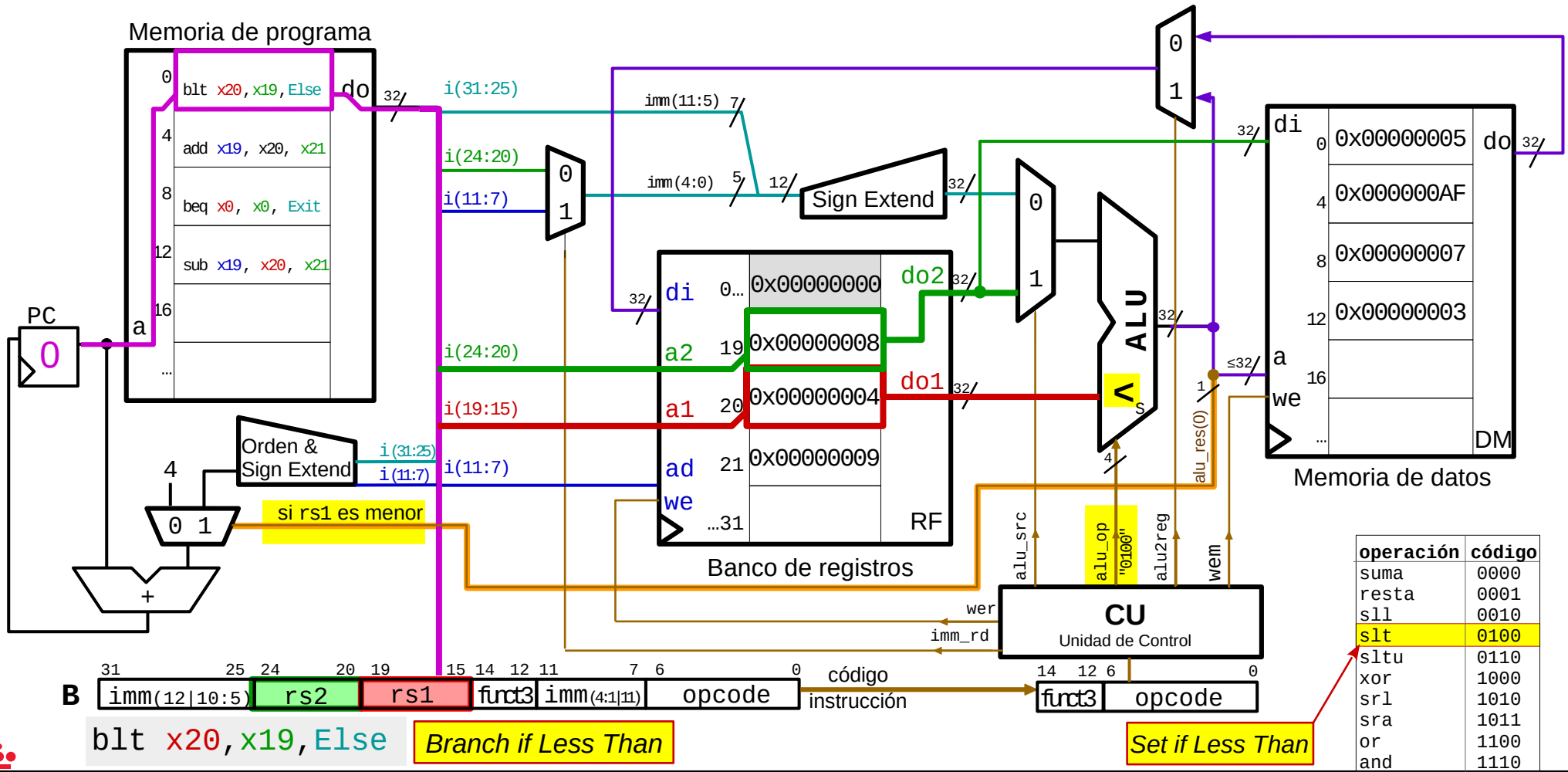
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

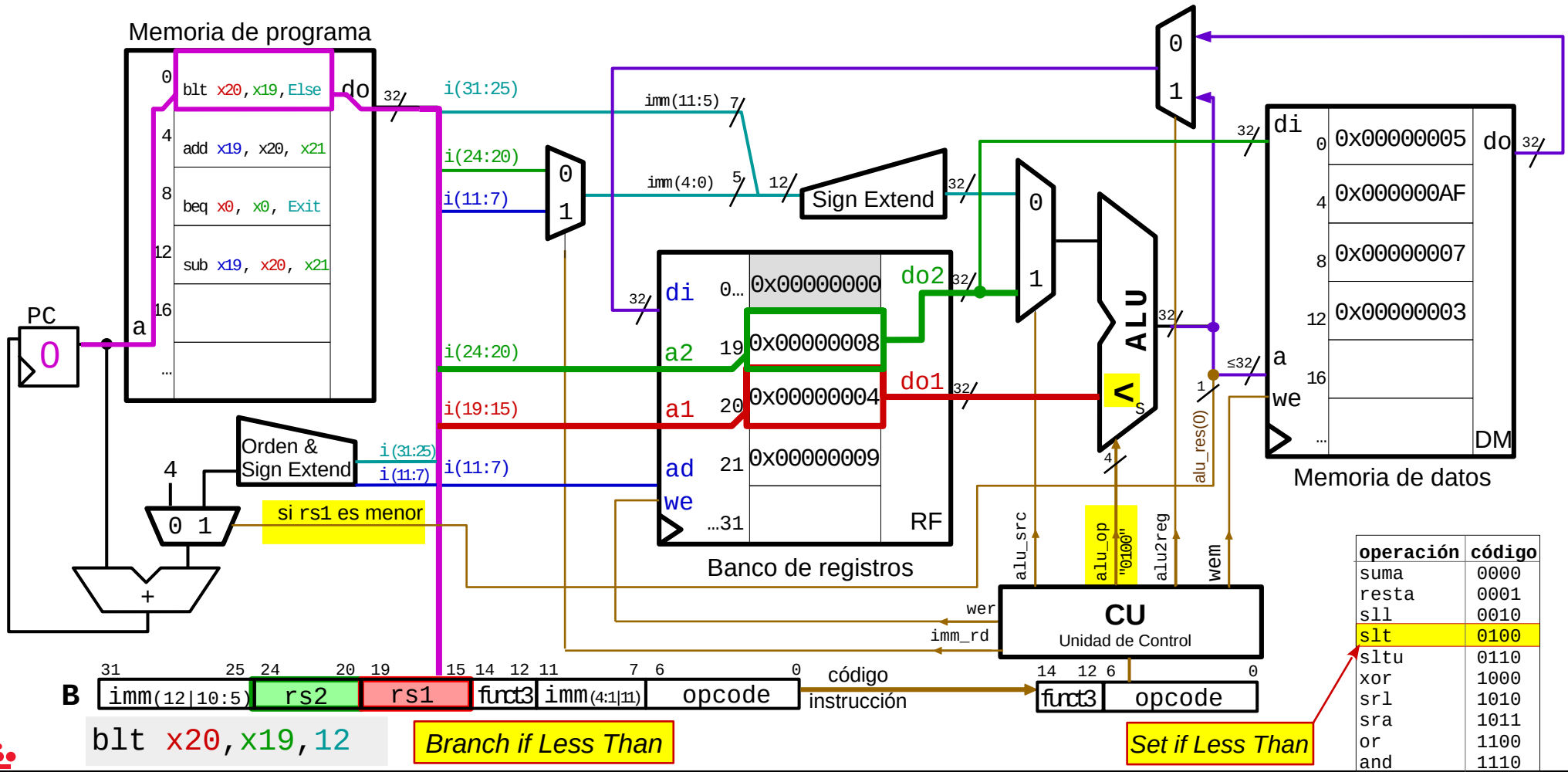
Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



Instrucciones de control (*branch*)

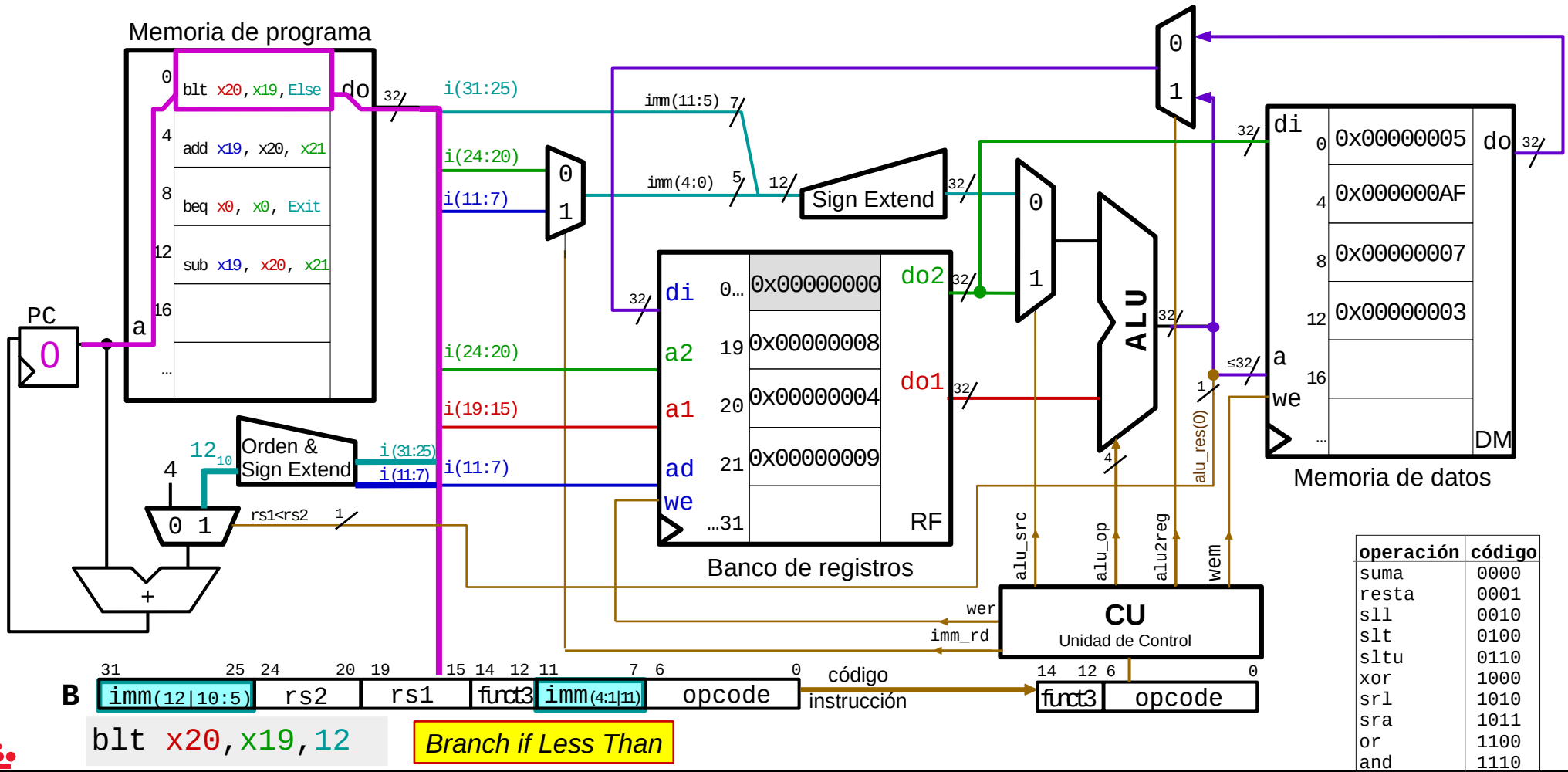
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



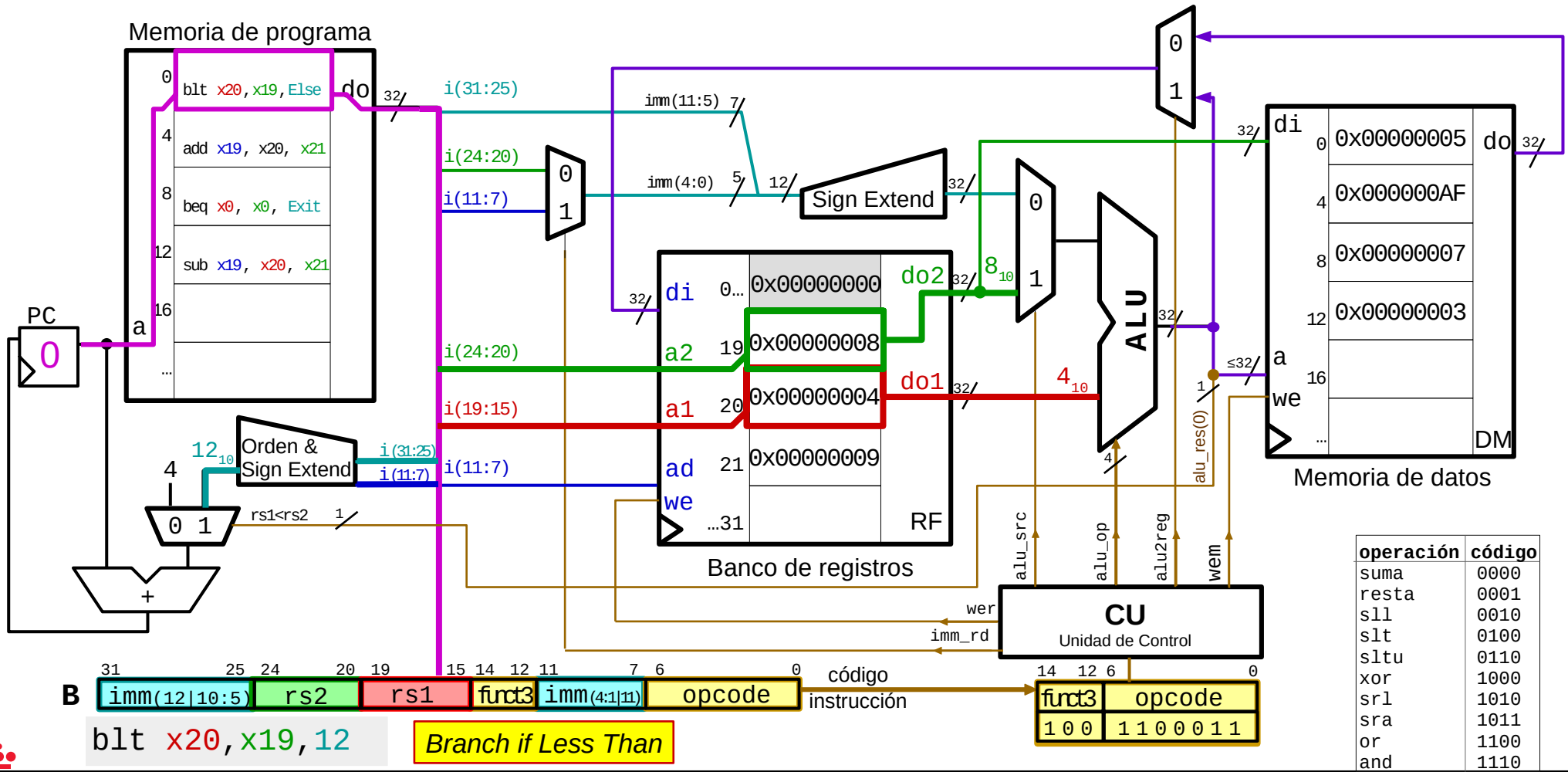
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

```
blt x20,x19,12
```

Branch if Less Than

Instrucciones de control (*branch*)

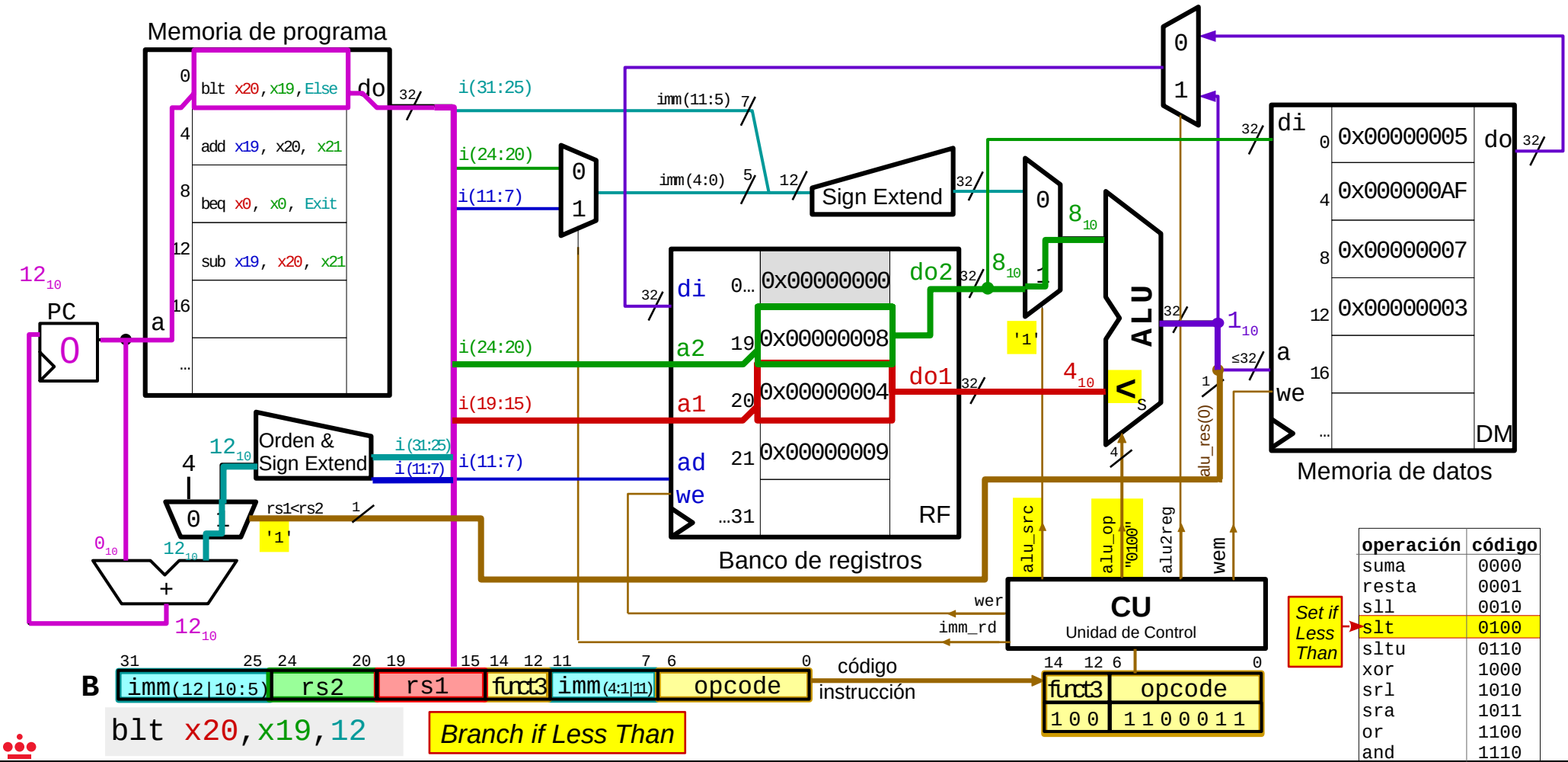
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

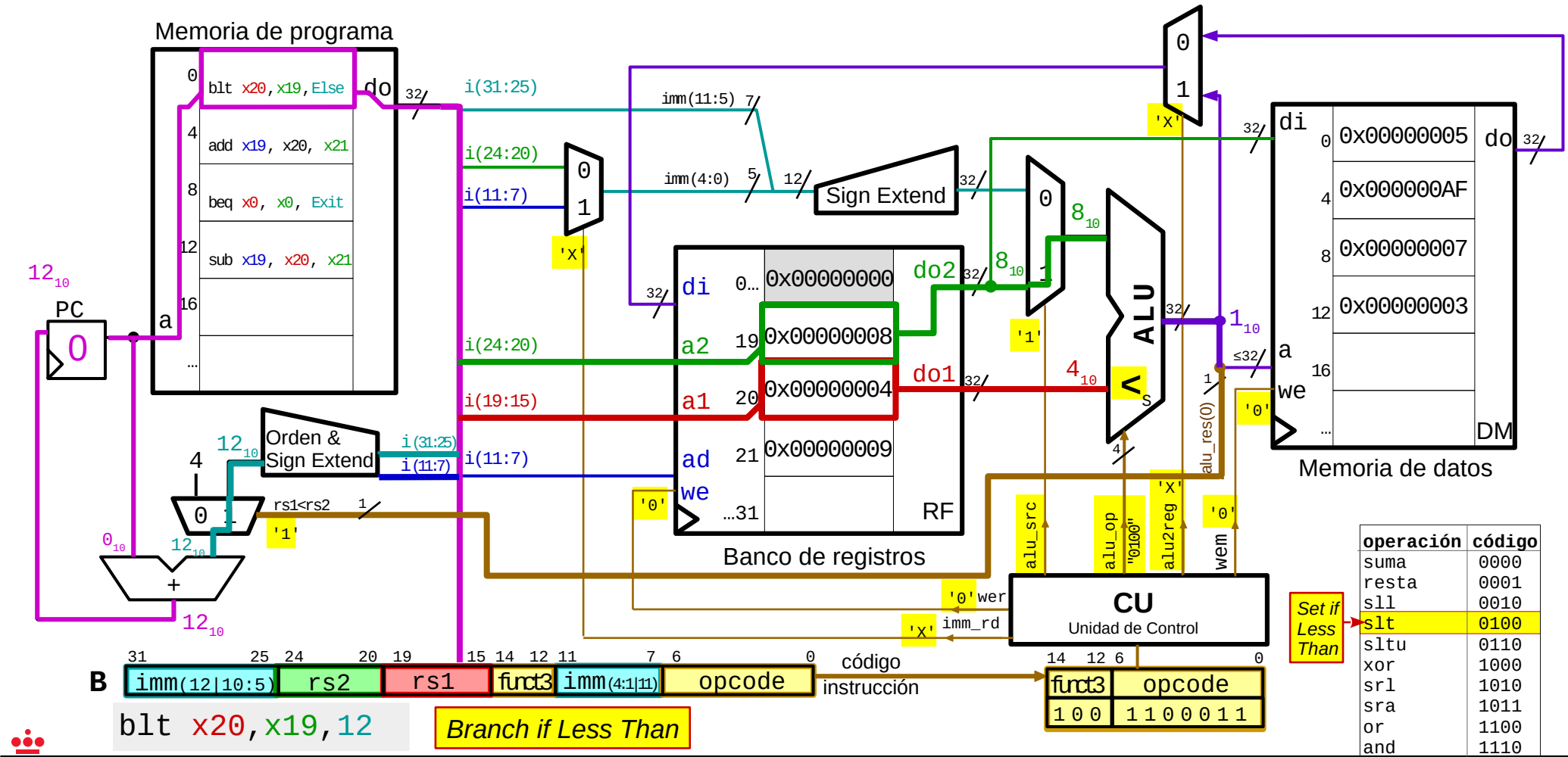
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

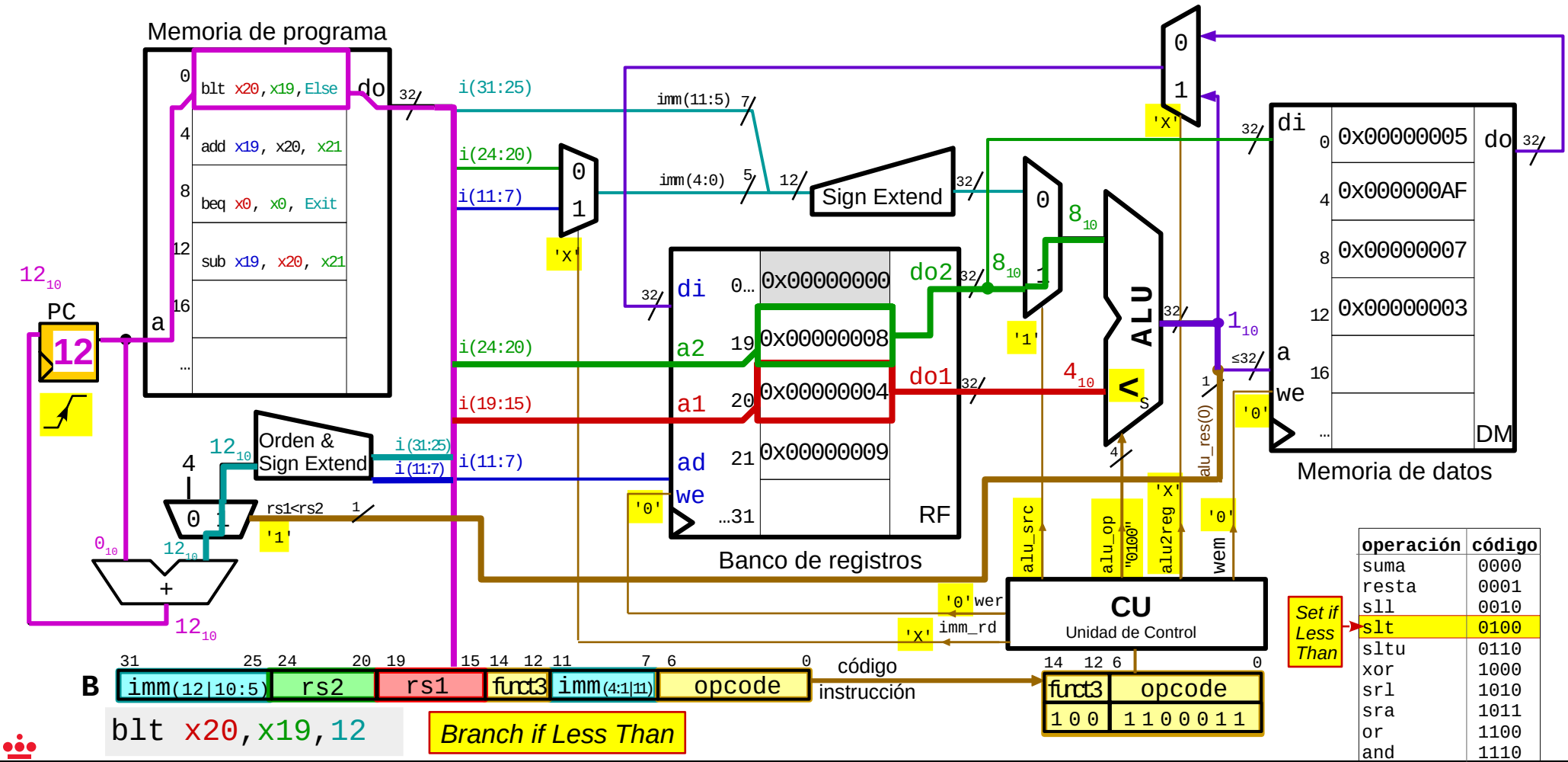
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

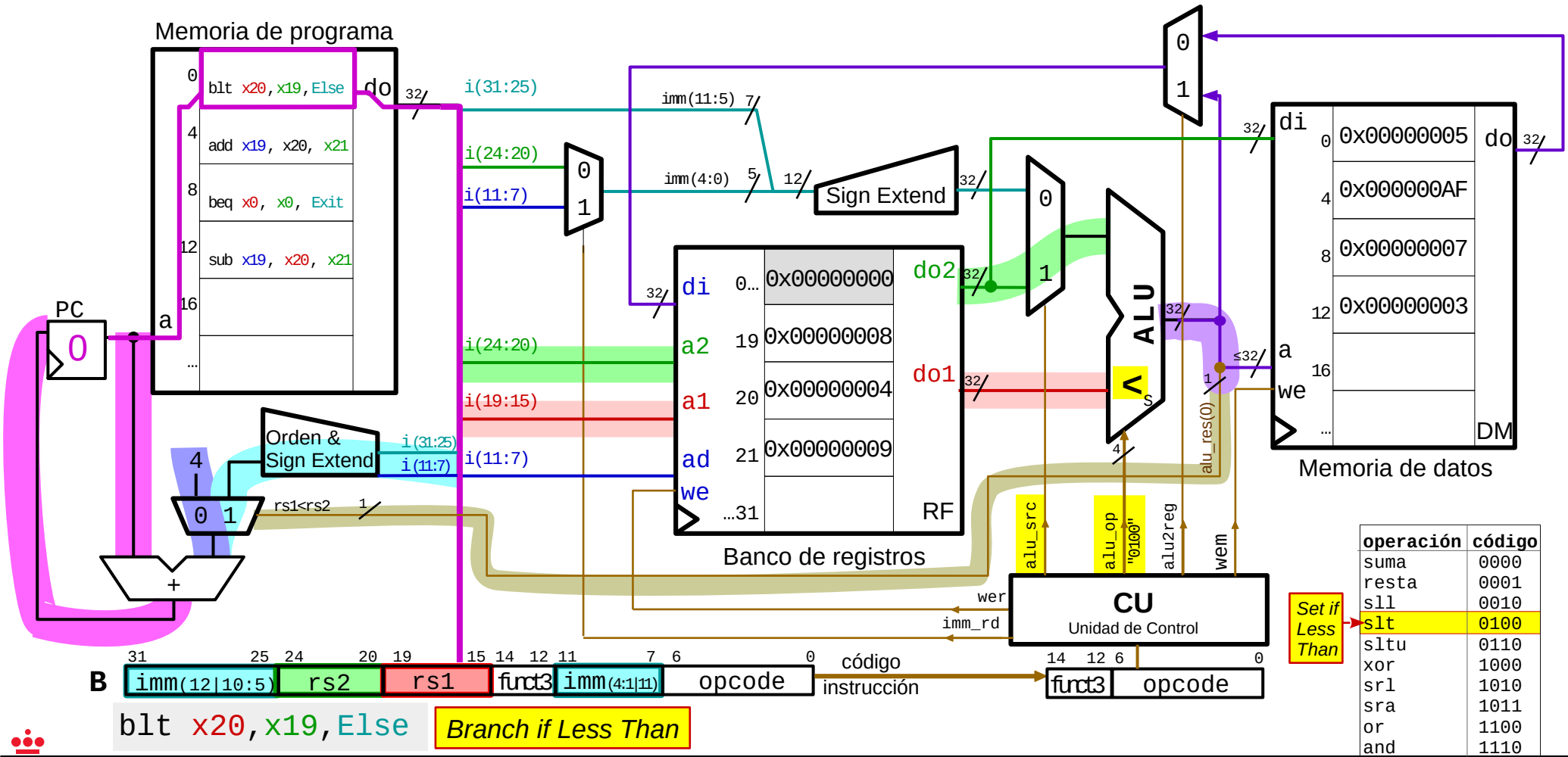
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

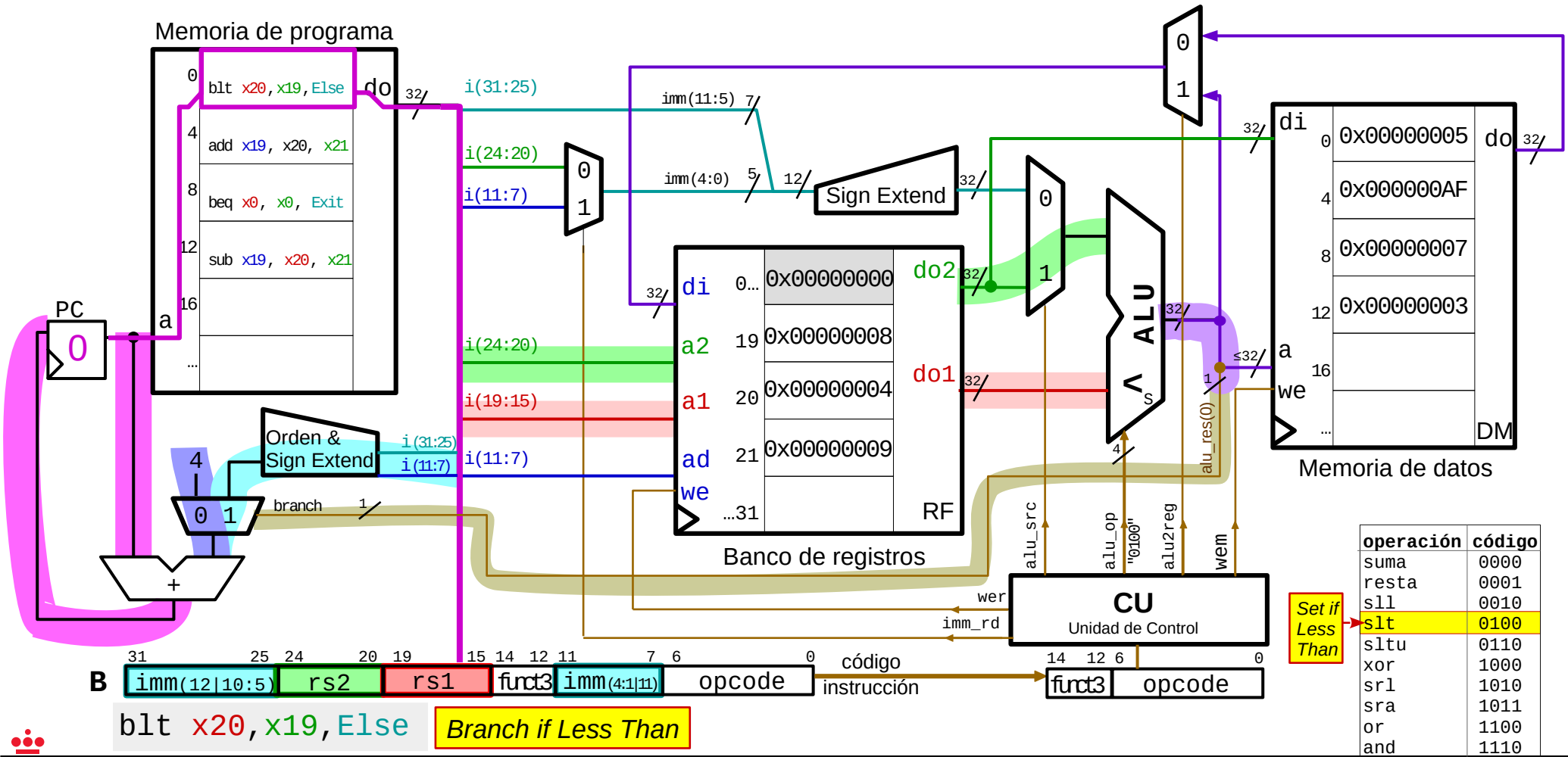
Instrucciones de control

6 instrucciones

	Tipo B			
	`beq`	*Branch if Equal*	`rs1 == rs2`	
	`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`	
☑	`blt`	*Branch if Less Than*	$rs1 <_s rs2$	*Set if Less Than*
	`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$	
	`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$	*Set if Less Than Unsigned*
	`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$	

Instrucciones de control (*branch*)

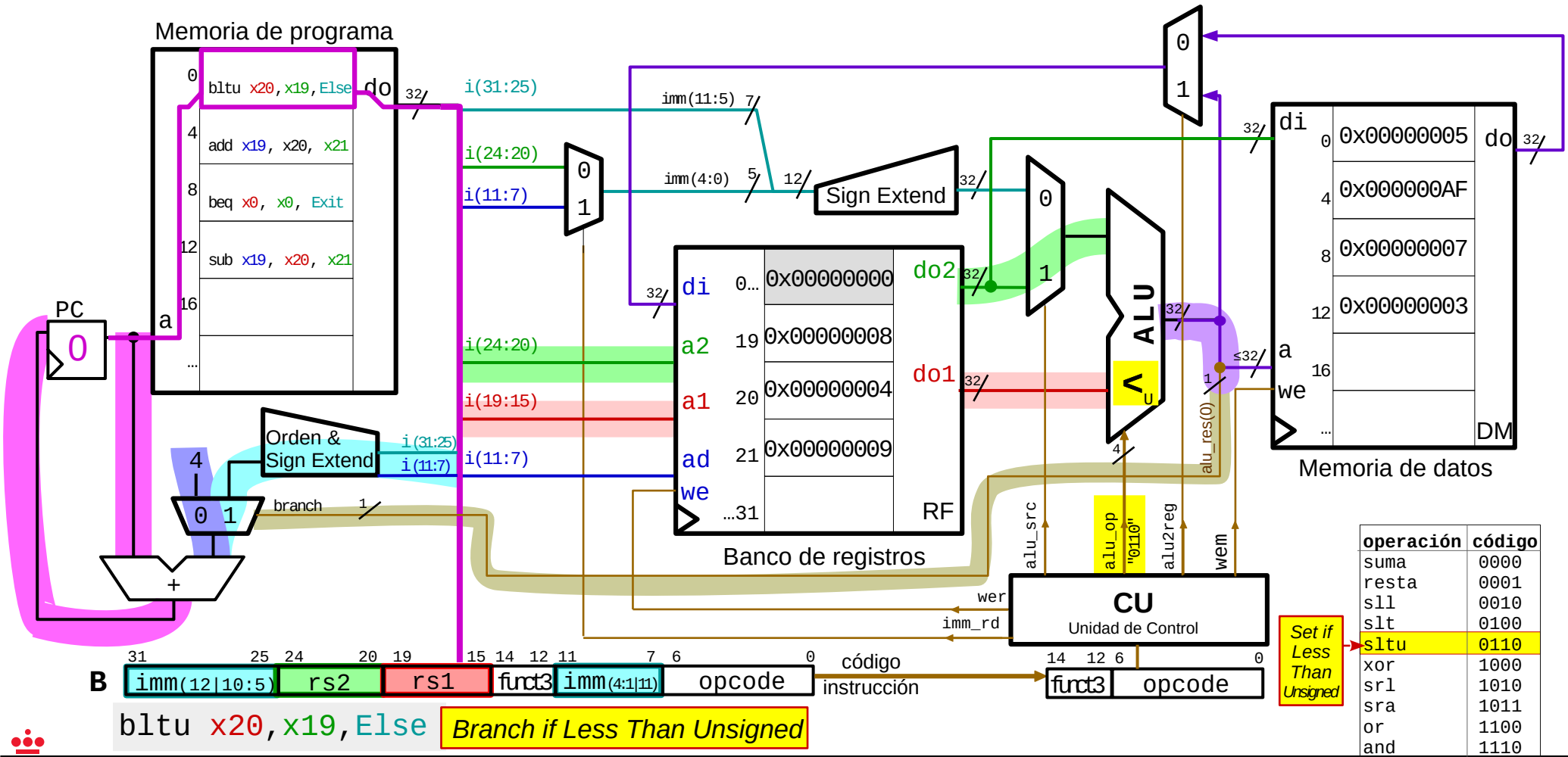
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control

6 instrucciones

Creamos operación nueva en la ALU

	Tipo B			
	beq	*Branch if Equal*	rs1 == rs2	*Set if Equal*
	bne	*Branch if Not Equal*	rs1 ≠ rs2	
☑	blt	*Branch if Less Than*	$rs1 <_s rs2$	*Set if Less Than*
	bge	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$	
☑	bltu	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$	*Set if Less Than Unsigned*
	bgeu	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$	

No hay instrucción *Set if Equal*

Necesitaría 2 instrucciones:

```
xor x21, x19, x20   → # x21 ← 0 si x19==x20
                      # x21 ≠ 0 si x19!=x20
sltiu x21, x21, 1   → # x21 ← 1 si x21 <u 1
                                   x21 == 0
```

Set if Less Than Immediate Unsigned

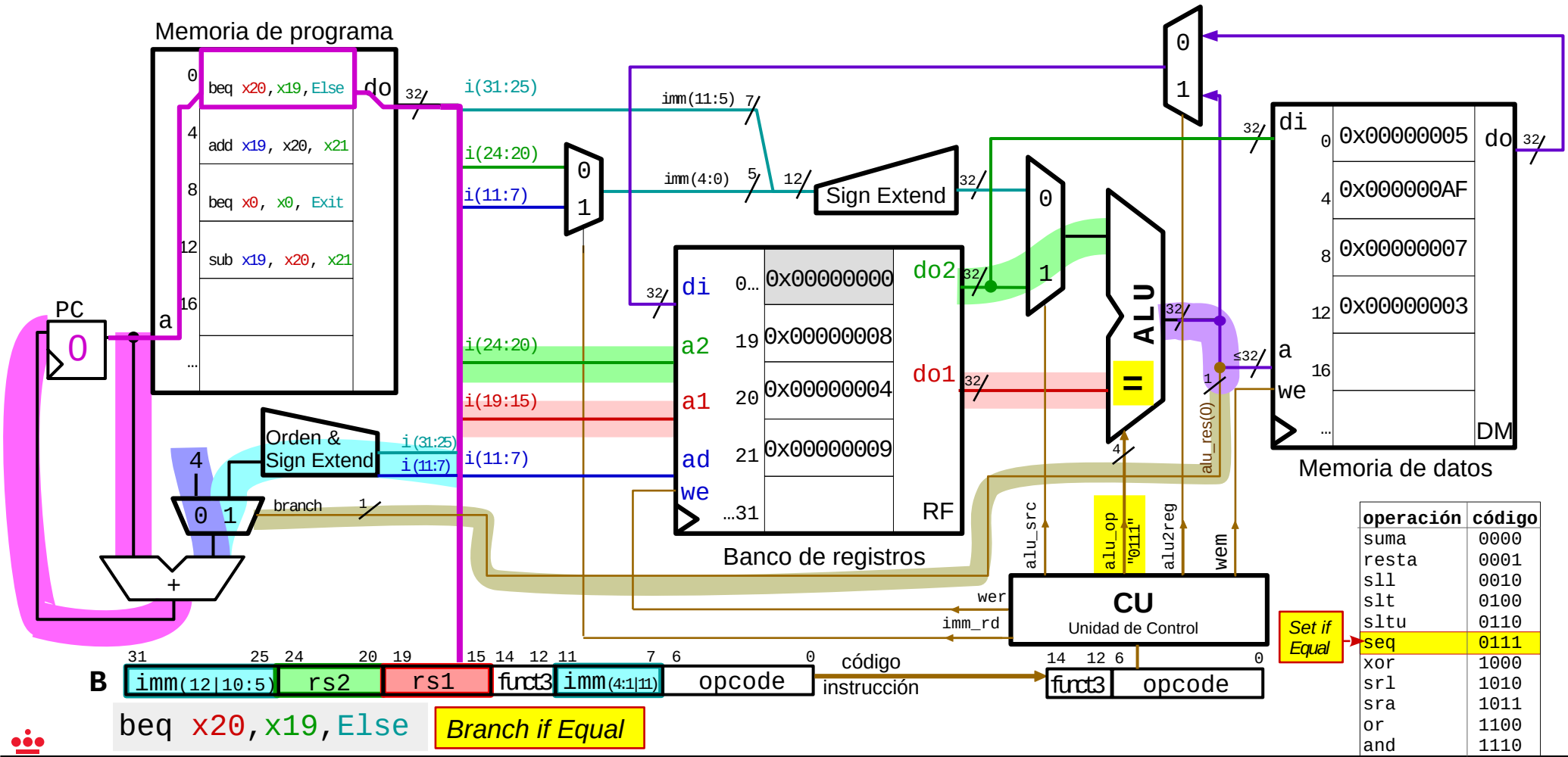
Set if Not Equal

```
xor x21, x19, x20
sltu x21, x0, x21   → # x21 ← 1 si 0 <u x21
                                   x21 != 0
```

Set if Less Than Unsigned

Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control

6 instrucciones

	Tipo B			
☑	`beq`	*Branch if Equal*	`rs1 == rs2`	operación nueva *Set if Equal*
	`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`	
☑	`blt`	*Branch if Less Than*	$rs1 <_s rs2$	*Set if Less Than*
	`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$	
☑	`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$	*Set if Less Than Unsigned*
	`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$	

Instrucciones de control

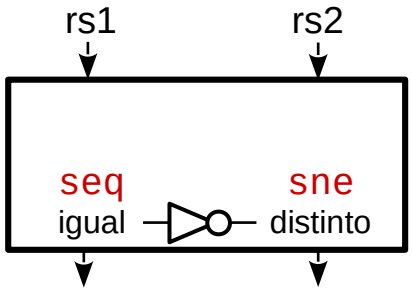
6 instrucciones

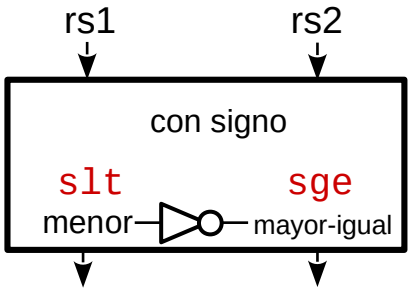
	Tipo B		
☑	`beq`	*Branch if Equal*	`rs1 == rs2`
	`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`
☑	`blt`	*Branch if Less Than*	$rs1 <_s rs2$
	`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$
☑	`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$
	`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$

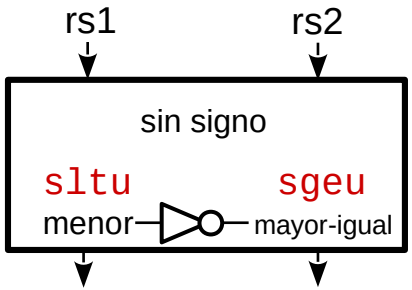
Set if Equal ↔ opuestos ↔ *Set if **NOT** Equal*

Set if Less Than ↔ opuestos ↔ *Set if **NOT** Less Than*

Set if Less Than Unsigned ↔ opuestos ↔ *Set if **NOT** Less Than Unsigned*

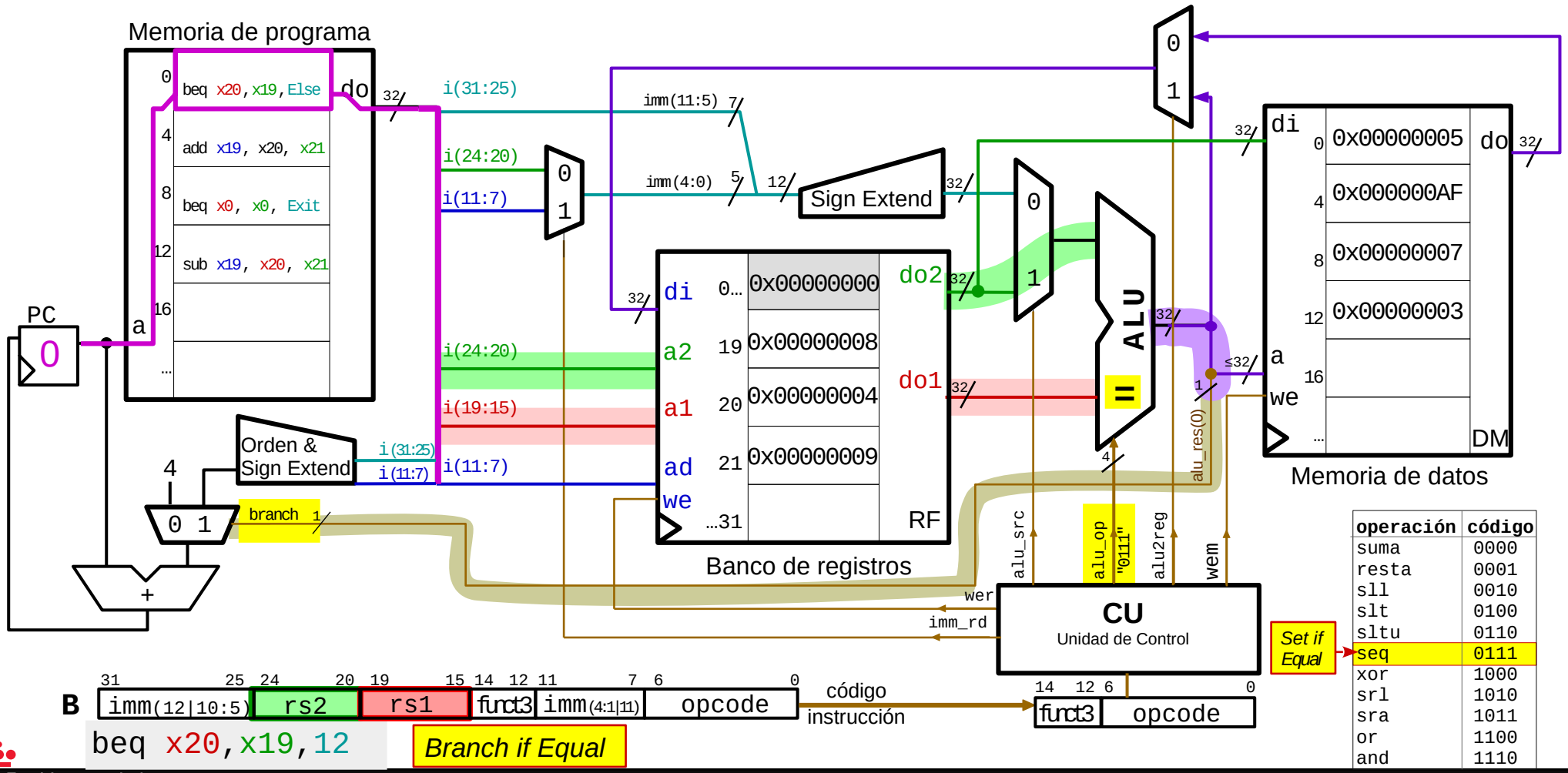






Instrucciones de control (*branch*)

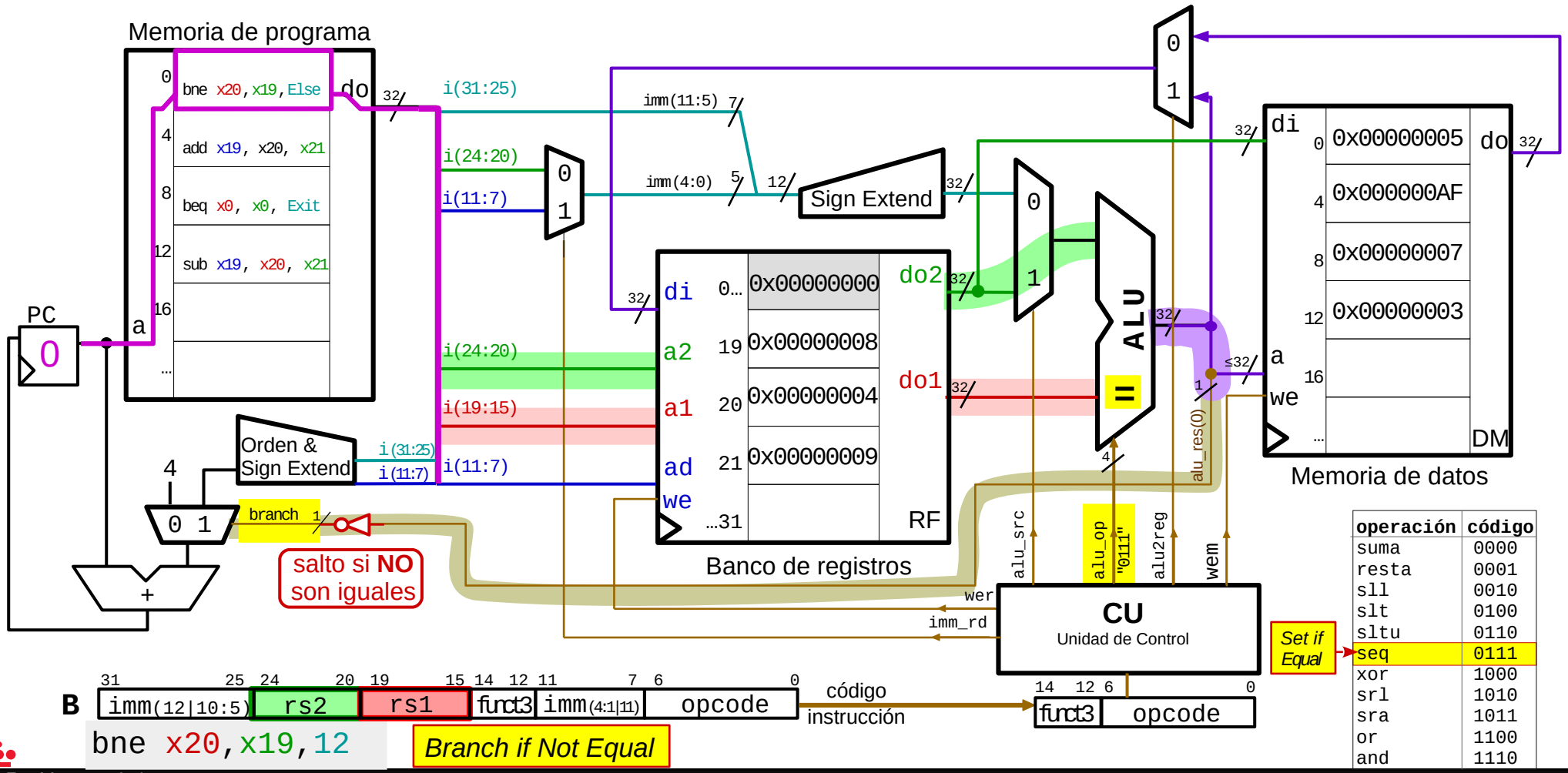
Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

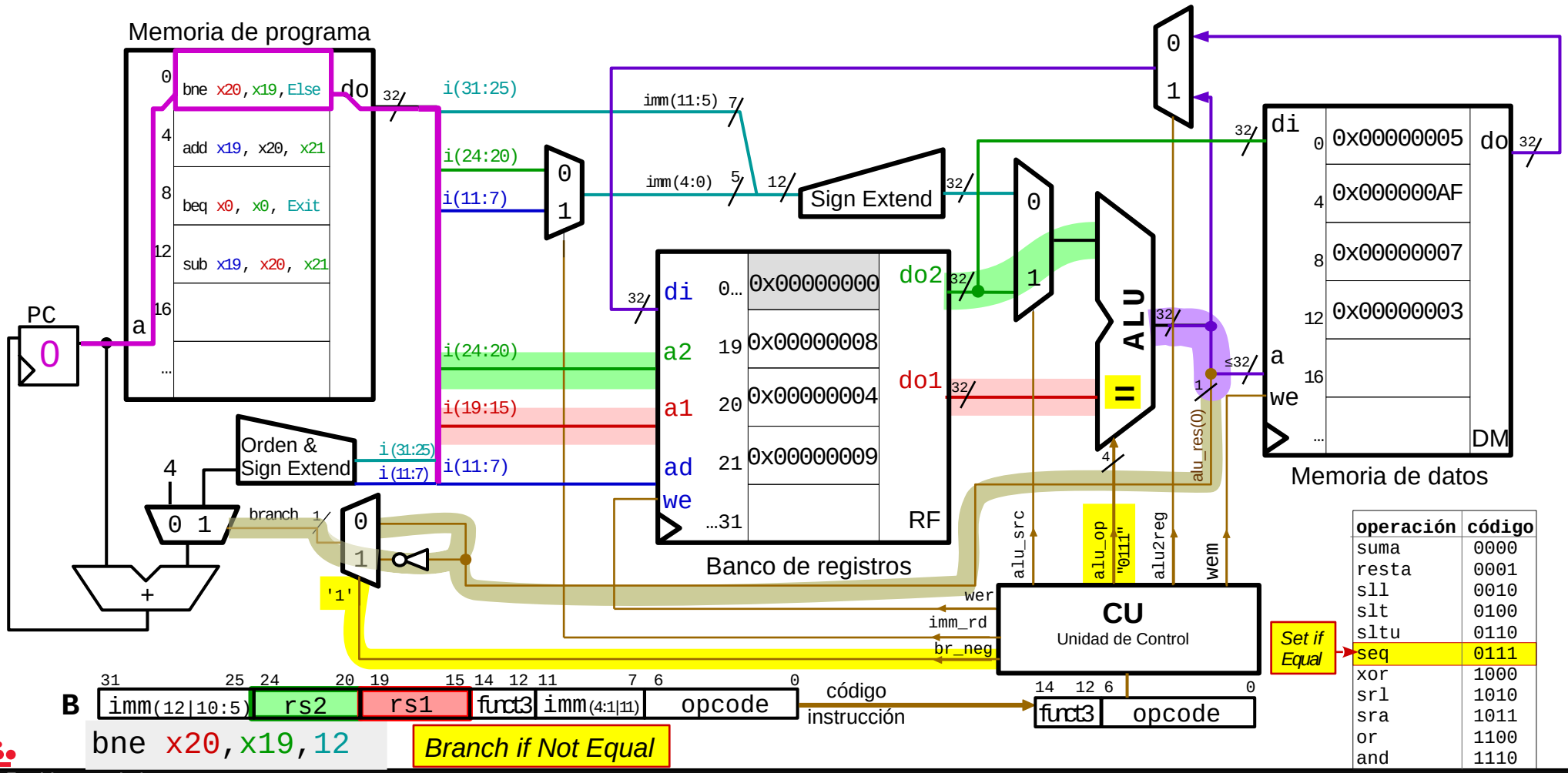
Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



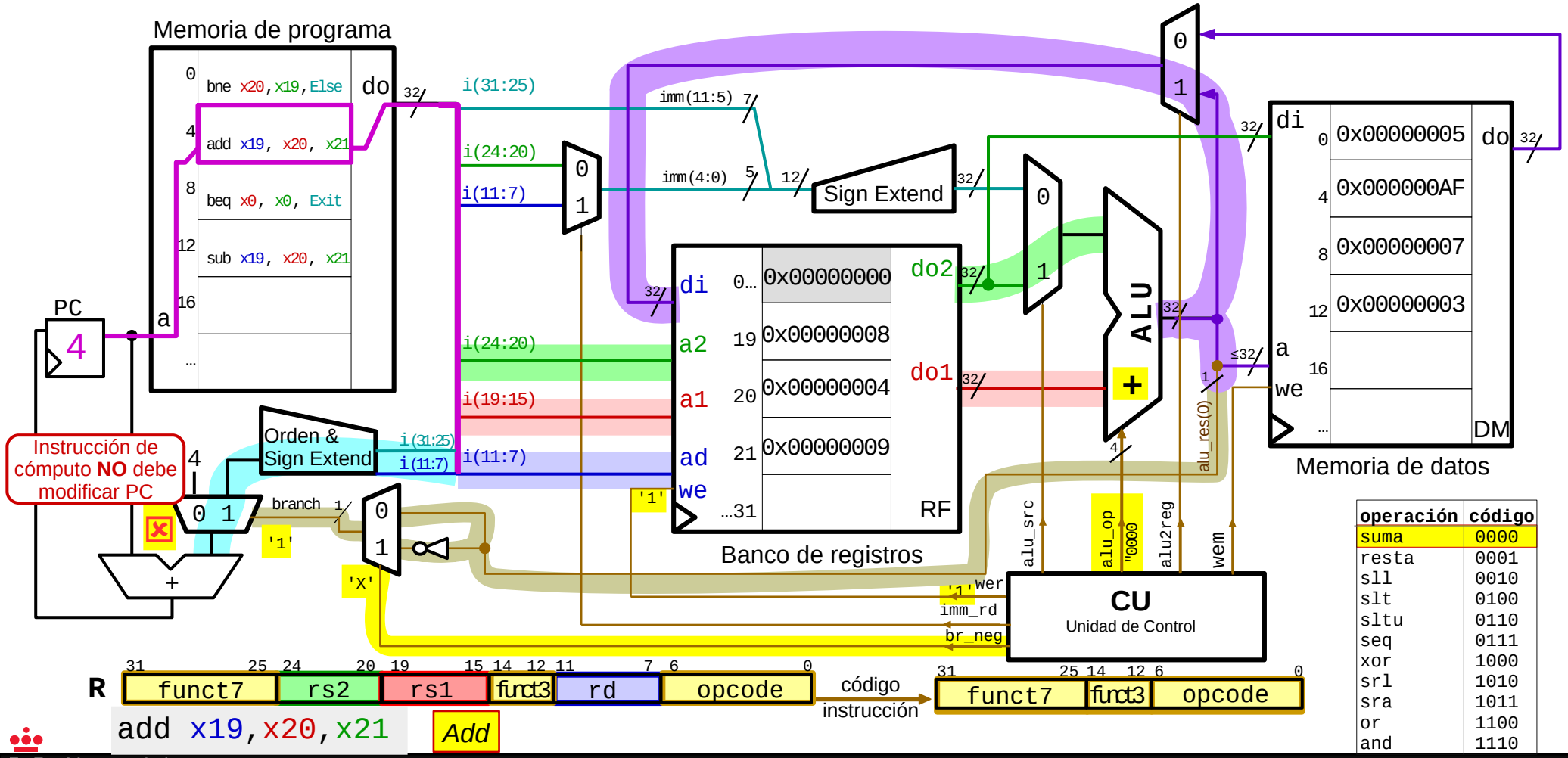
Instrucciones de control (*branch*)

Modifica PC { **Sin registro destino** / **No usa memoria** *Ni carga ni almacena*



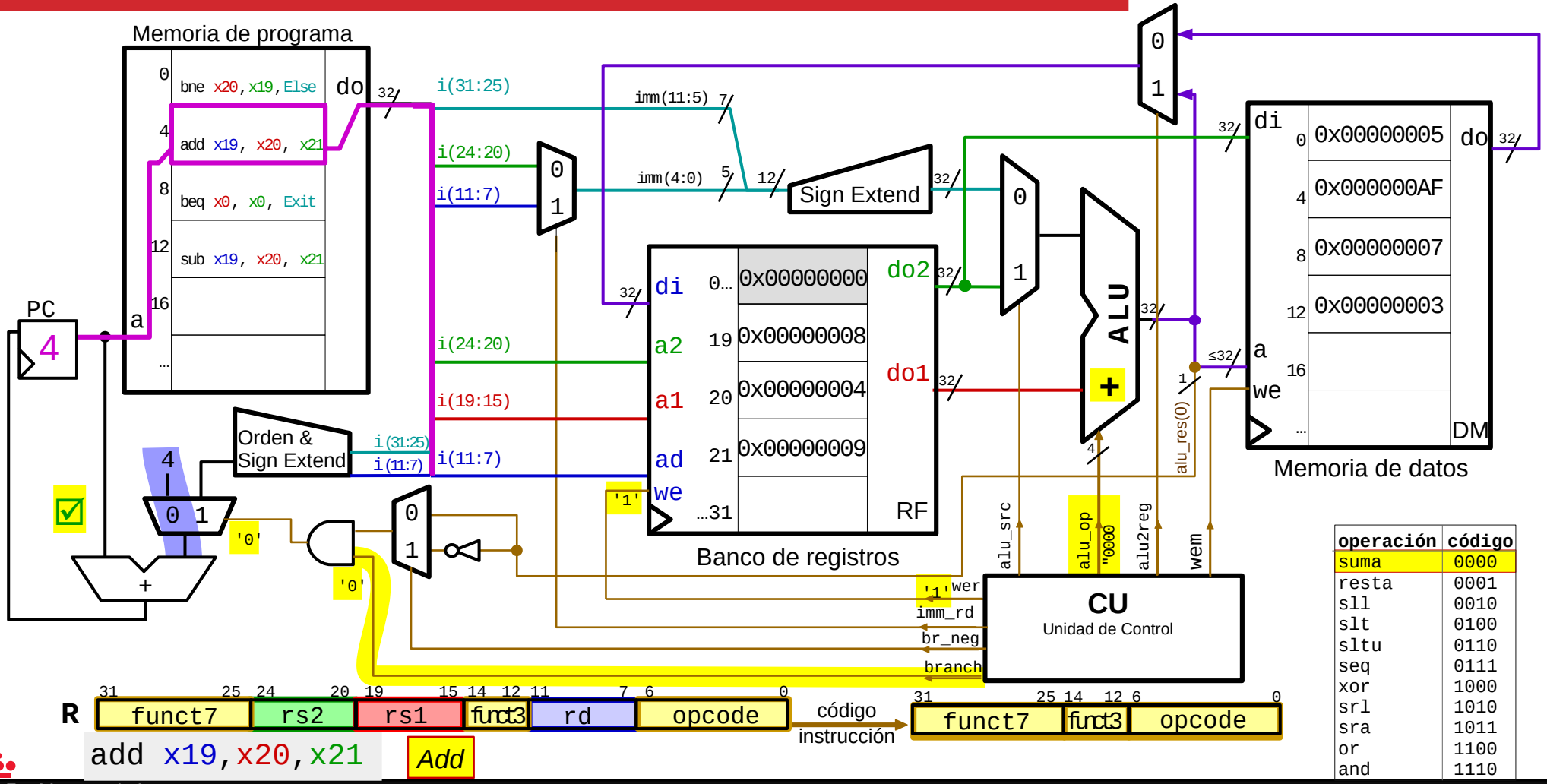
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de cómputo



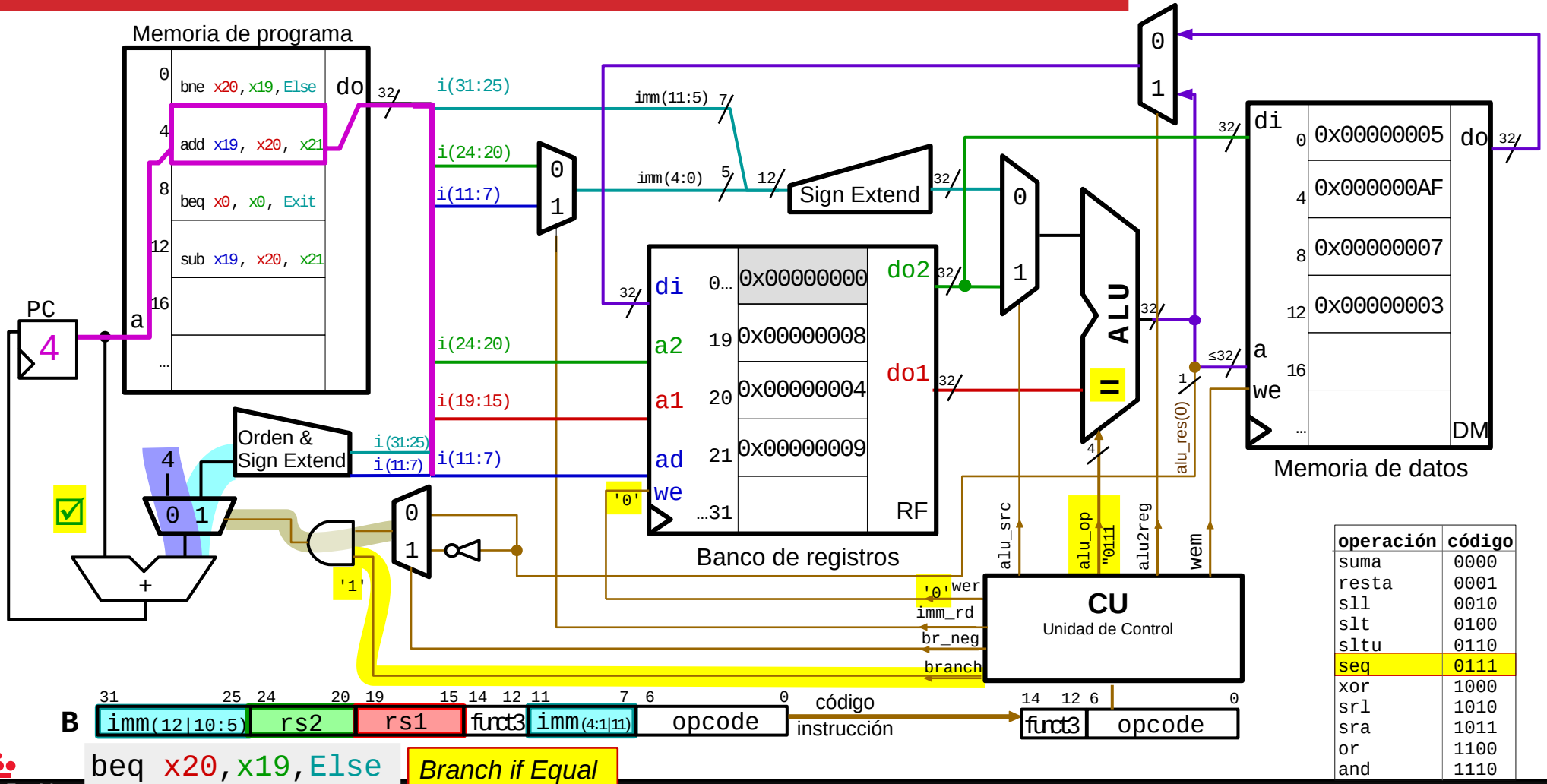
operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de cómputo



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

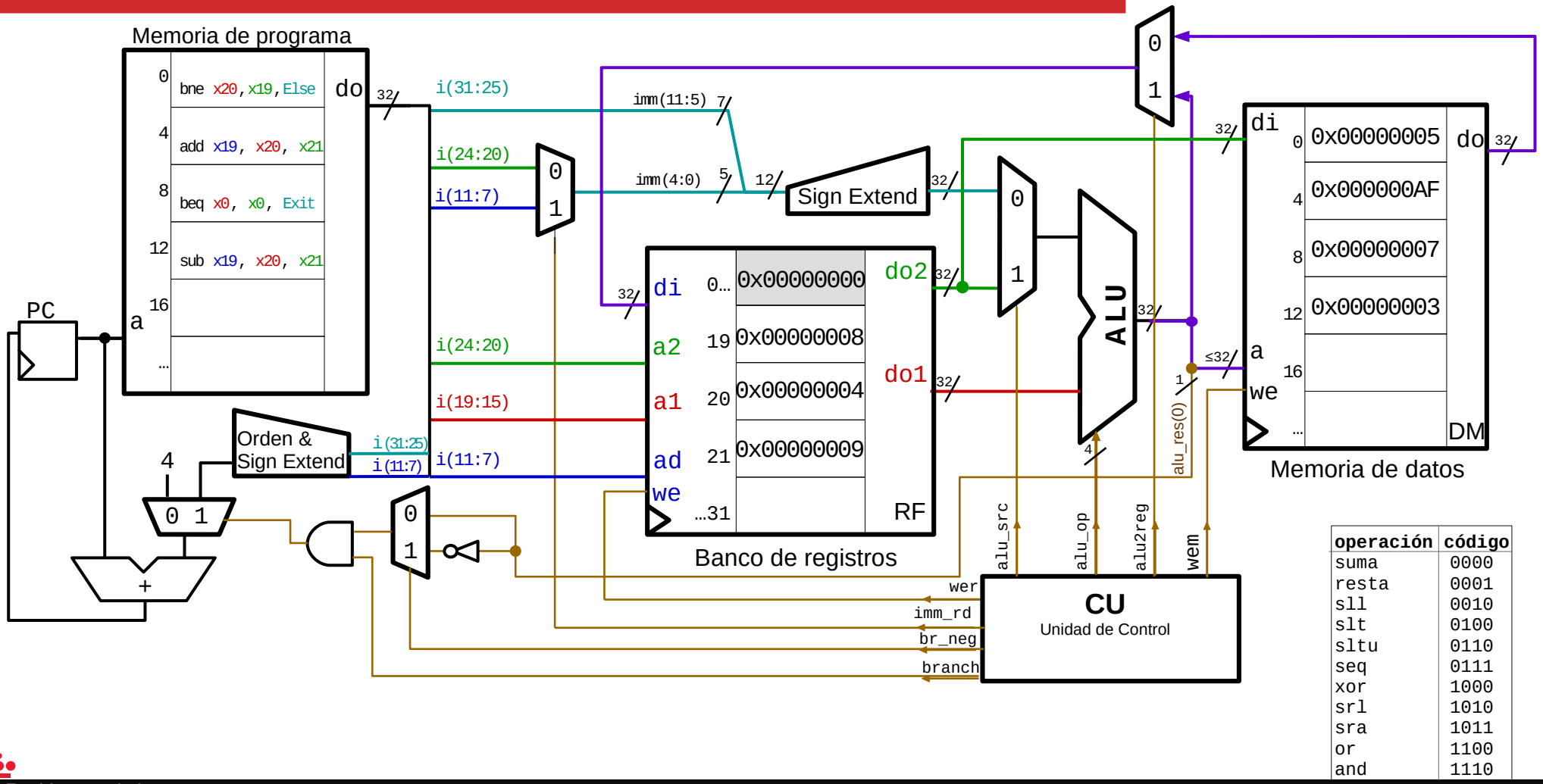
Instrucciones de cómputo



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

`beq x20,x19,Else` *Branch if Equal*

Instrucciones de cómputo



operación	código
suma	0000
resta	0001
sll	0010
slt	0100
sltu	0110
seq	0111
xor	1000
srl	1010
sra	1011
or	1100
and	1110

Instrucciones de control

6 instrucciones

	Tipo B		
☑	`beq`	*Branch if Equal*	`rs1 == rs2`
☑	`bne`	*Branch if Not Equal*	`rs1 ≠ rs2`
☑	`blt`	*Branch if Less Than*	$rs1 <_s rs2$
☑	`bge`	*Branch if Greater than or Equal*	$rs1 \geq_s rs2$
☑	`bltu`	*Branch if Less Than, Unsigned*	$rs1 <_u rs2$
☑	`bgeu`	*Branch if Greater than or Equal, Unsigned*	$rs1 \geq_u rs2$

Set if Equal ↔ opuestos ↔ *Set if **NOT** Equal*

Set if Less Than ↔ opuestos ↔ *Set if **NOT** Less Than*

Set if Less Than Unsigned ↔ opuestos ↔ *Set if **NOT** Less Than Unsigned*

Bibliografía y referencias

- Guía Práctica de RISC-V: El Atlas de una Arquitectura Abierta
  David Patterson, Andrew Waterman
  http://riscvbook.com/espanol/
- Digital Design and Computer Architecture
  David Money Harris, Sarah L. Harris
- Computer Organization and Design. RISC-V Edition
  David Patterson, John Hennessy
- The RISC-V Instruction Set Manual.
  Vol I: User-Level ISA. v2.2
  Andrew Waterman, Krste Asanovic
  https://riscv.org/specifications/
- Design of the RISC-V Instruction Set Architecture
  Andrew Waterman
  http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html
- https://riscv.org/
- https://github.com/kvakil/venus

doi:10.5281/zenodo.4426635
versión 1.0
Video disponible: https://youtu.be/UzCLsPhydBY