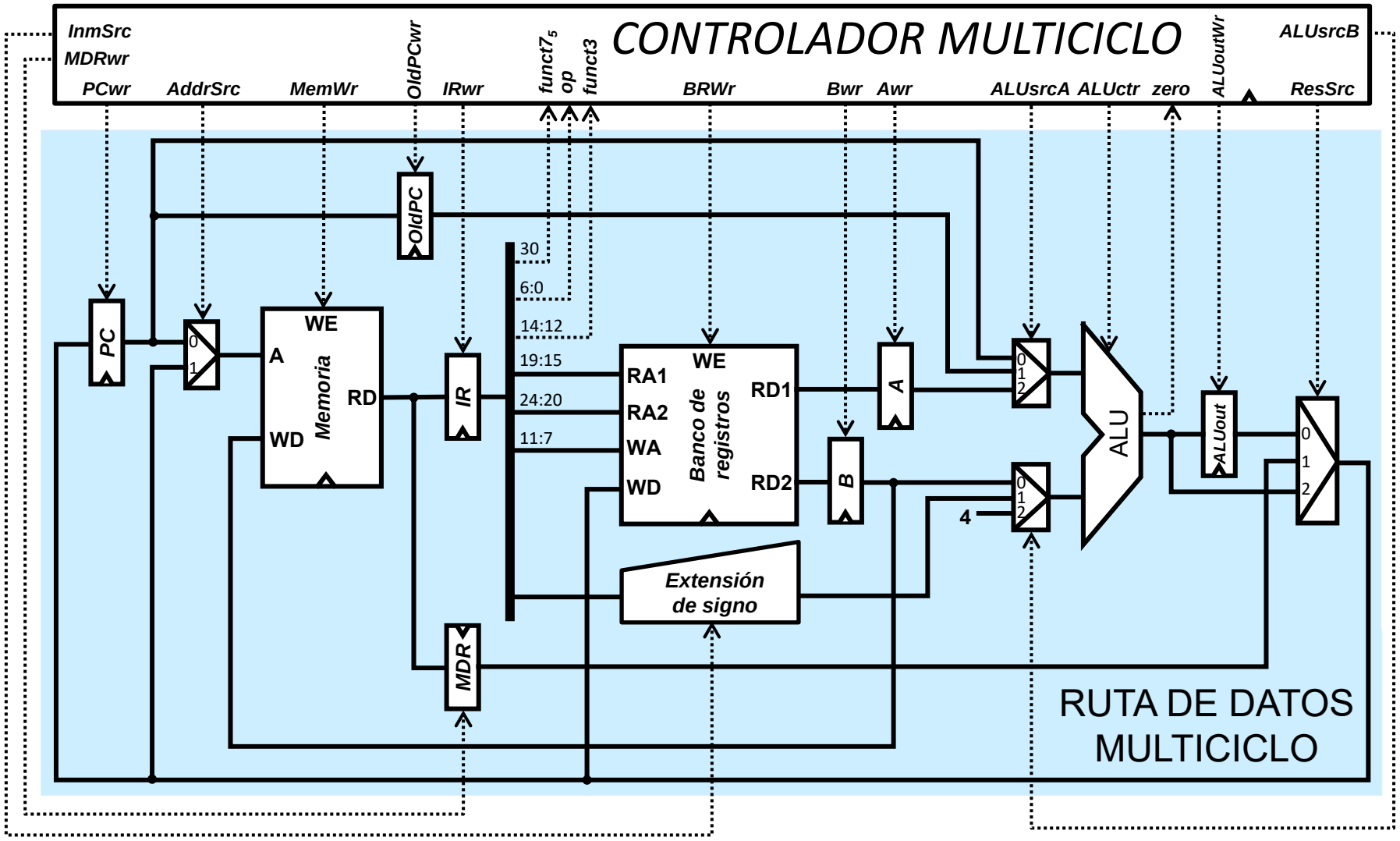
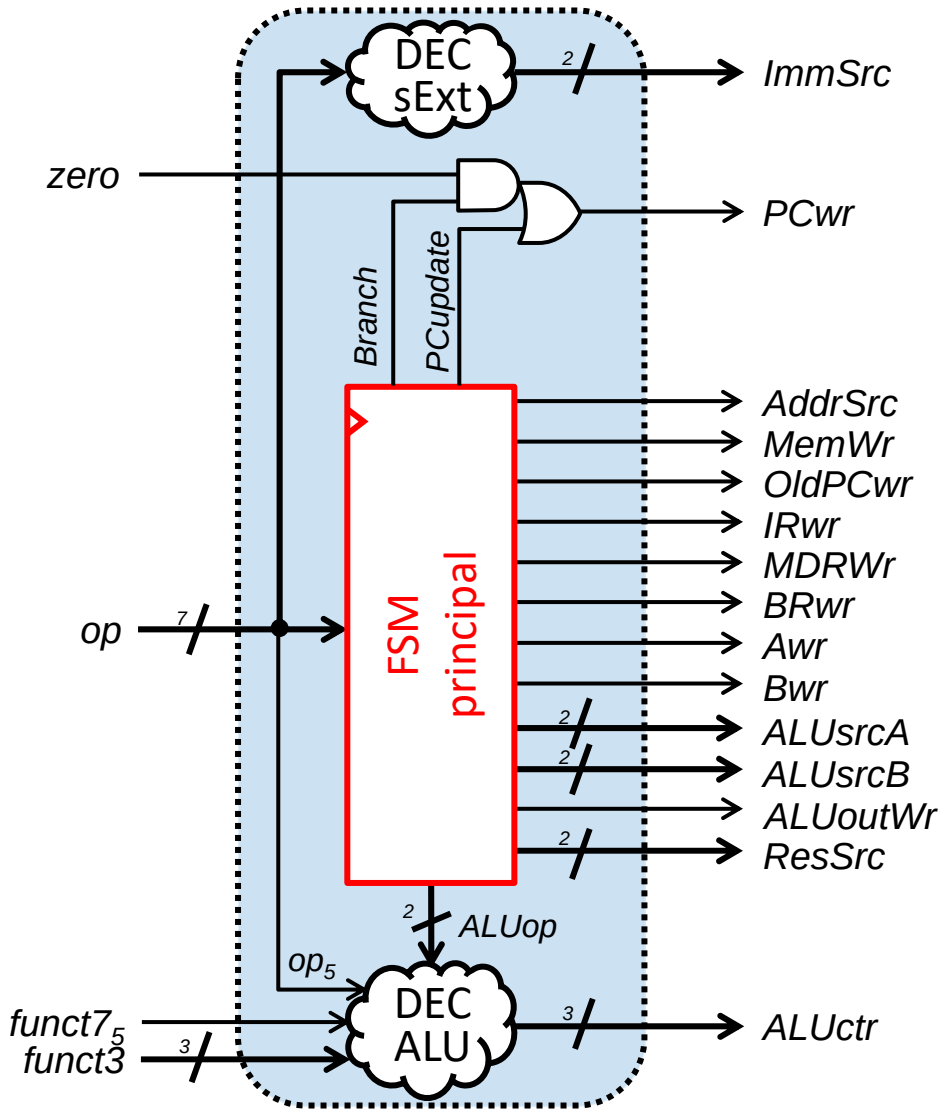
Problemas Tema 6:

# Diseño multiciclo del procesador

**Fernando Castro Rodríguez**
**José Manuel Mendías Cuadros**
*Dpto. Arquitectura de Computadores y Automática*
*Universidad Complutense de Madrid*

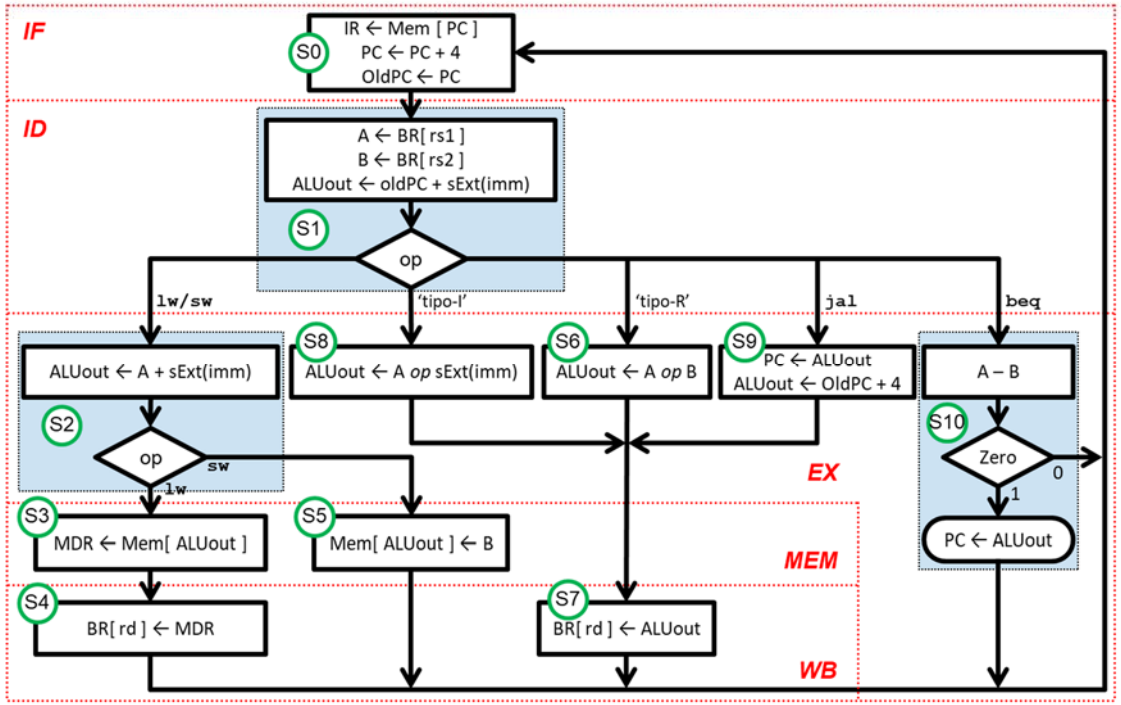
**1)** En el procesador **multiciclo** estudiado en clase, justifica los valores que toman las señales de control para la ejecución de las instrucciones “sw”.



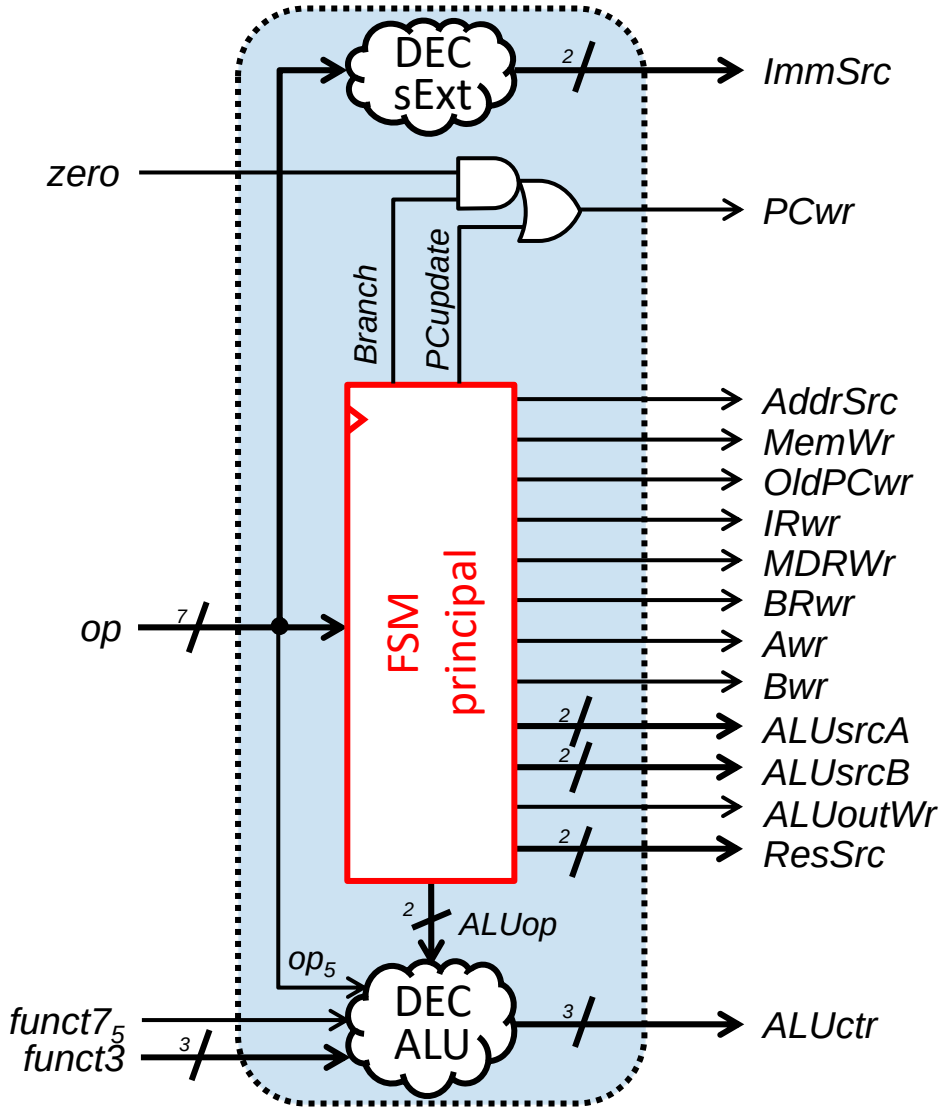



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

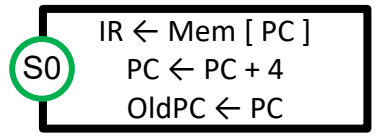



**Las instrucciones sw siguen la secuencia de estados S0-S1-S2-S5**

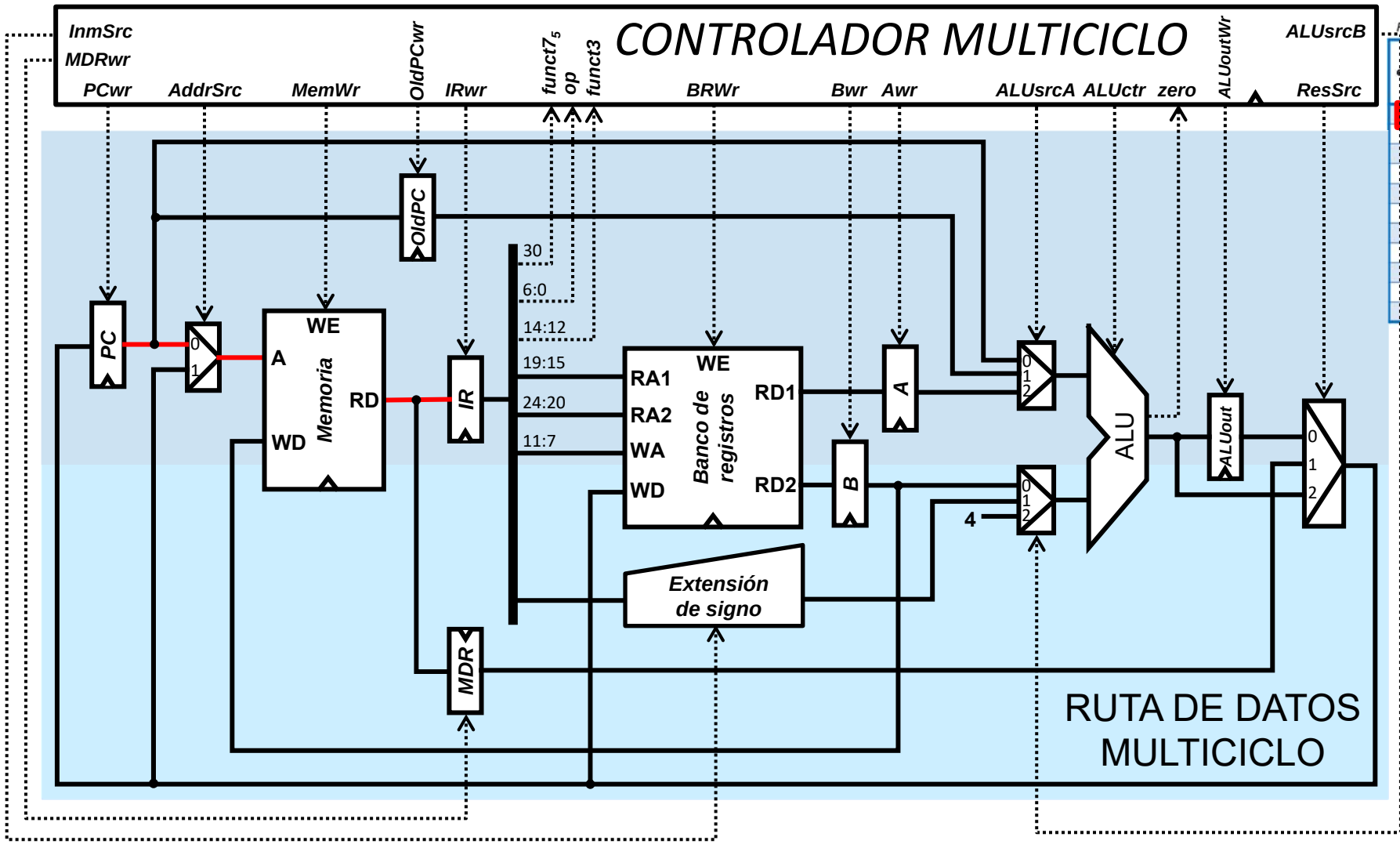



*Función de salida*

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

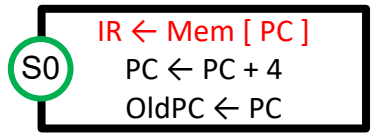



- La señal Branch está a 0 ya que no se trata de una instrucción beq
- La señal PCupdate está a 1 ya que el PC se actualiza con el valor PC+4




Función de salida

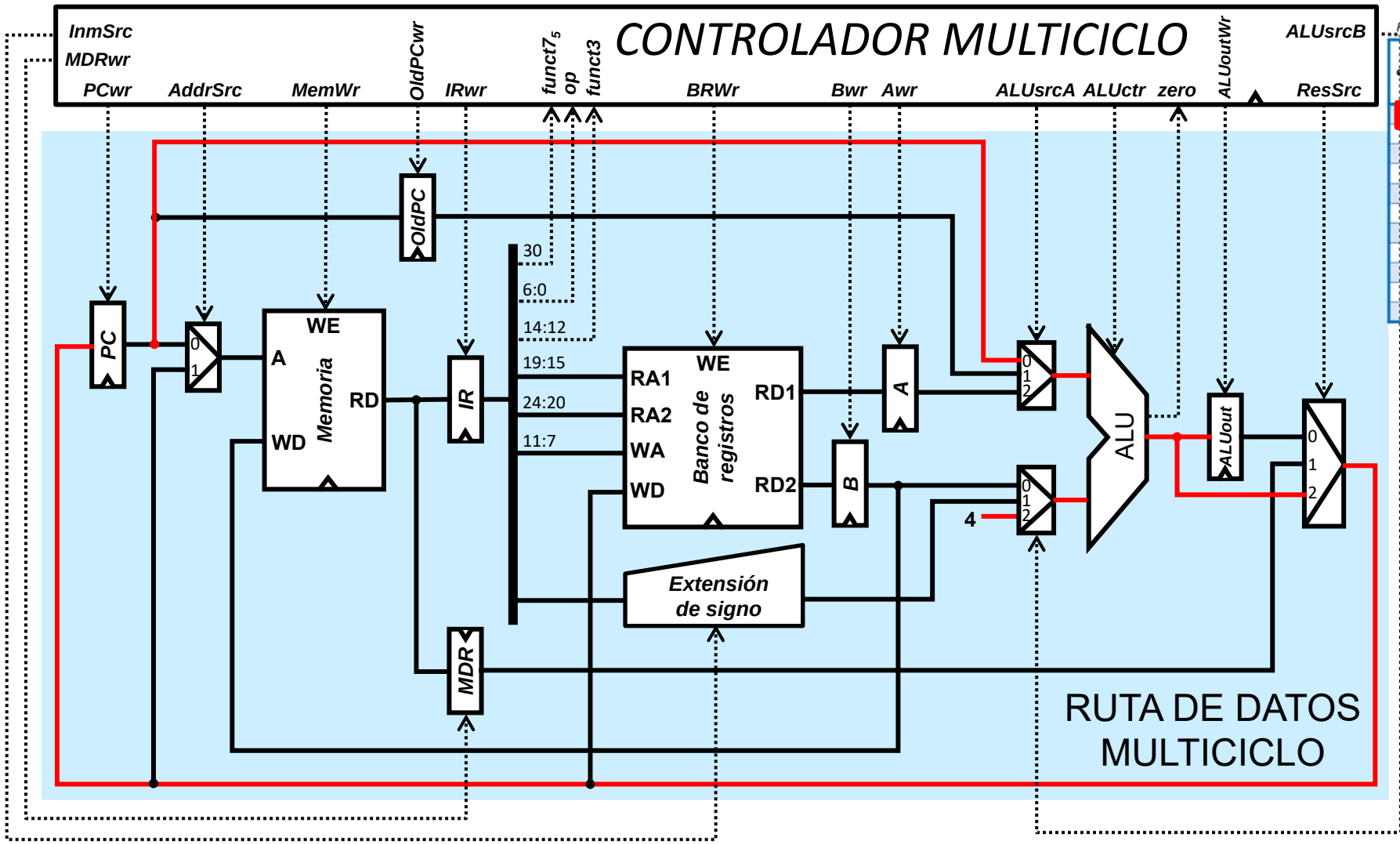
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- AddrSrc toma el valor 0 ya que se lee la dirección de memoria indicada por el PC para obtener la instrucción

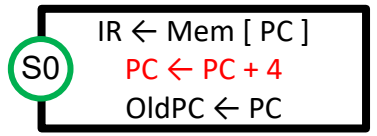
- IRwr toma el valor 1 ya que la instrucción leída de memoria se escribe en el registro IR

- MemWr toma el valor 0 ya que en este ciclo la memoria se lee, pero no se escribe

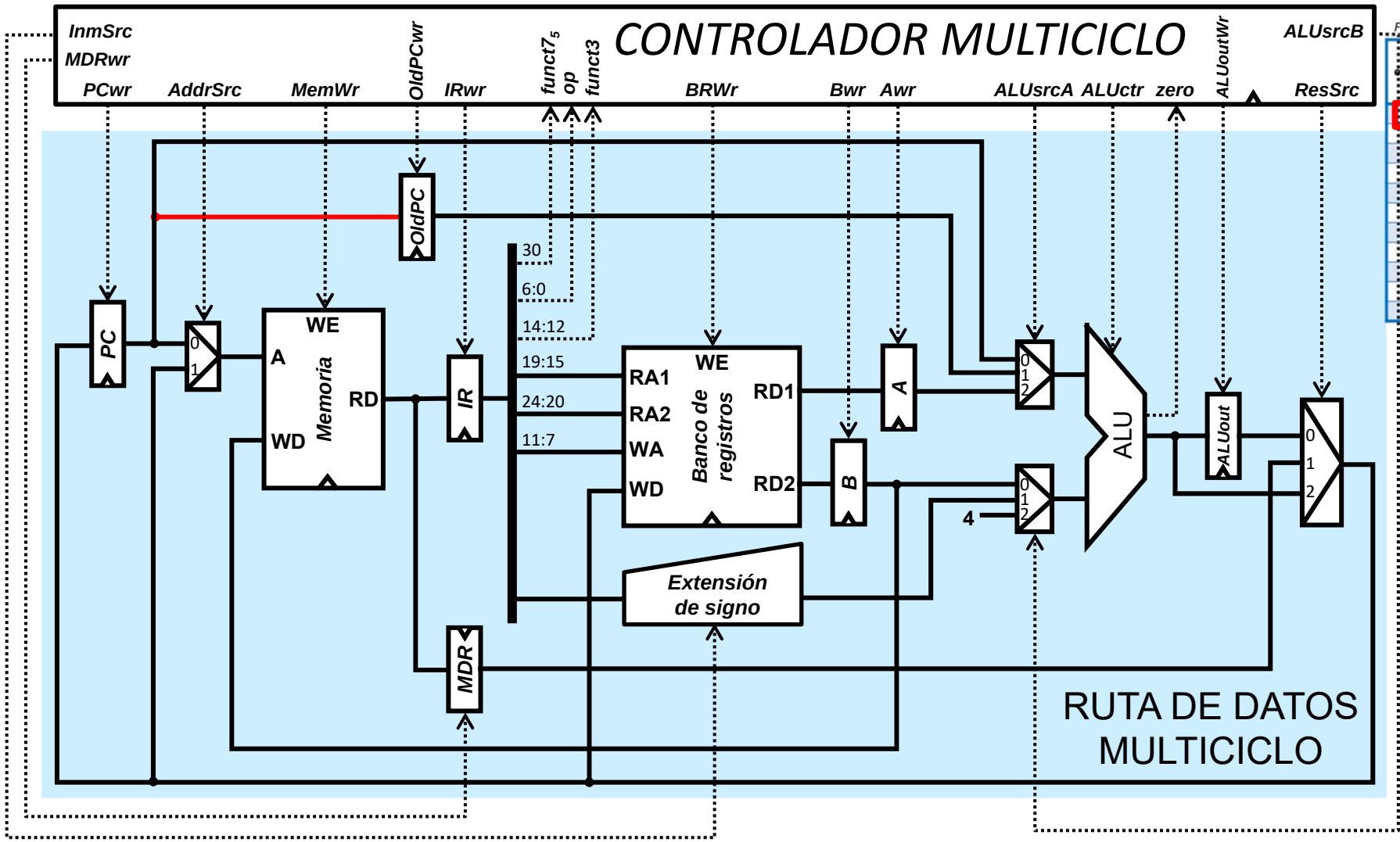



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

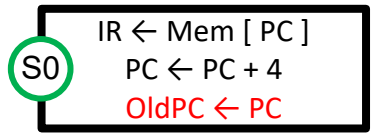



- En la ALU se realiza la operación PC+4 (ALUop=00). El valor del PC llega a la entrada superior de la ALU a través de la entrada 0 del mux correspondiente, por lo que ALUsrcA vale 00. El valor 4 entra a la entrada inferior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcB toma el valor 10. El resultado no se escribe en el registro ALUout (por tanto la señal ALUoutWr vale 0), sino que se lleva directamente al registro PC para actualizar su valor, a través de la entrada 2 del multiplexor controlado por ResSrc (10)

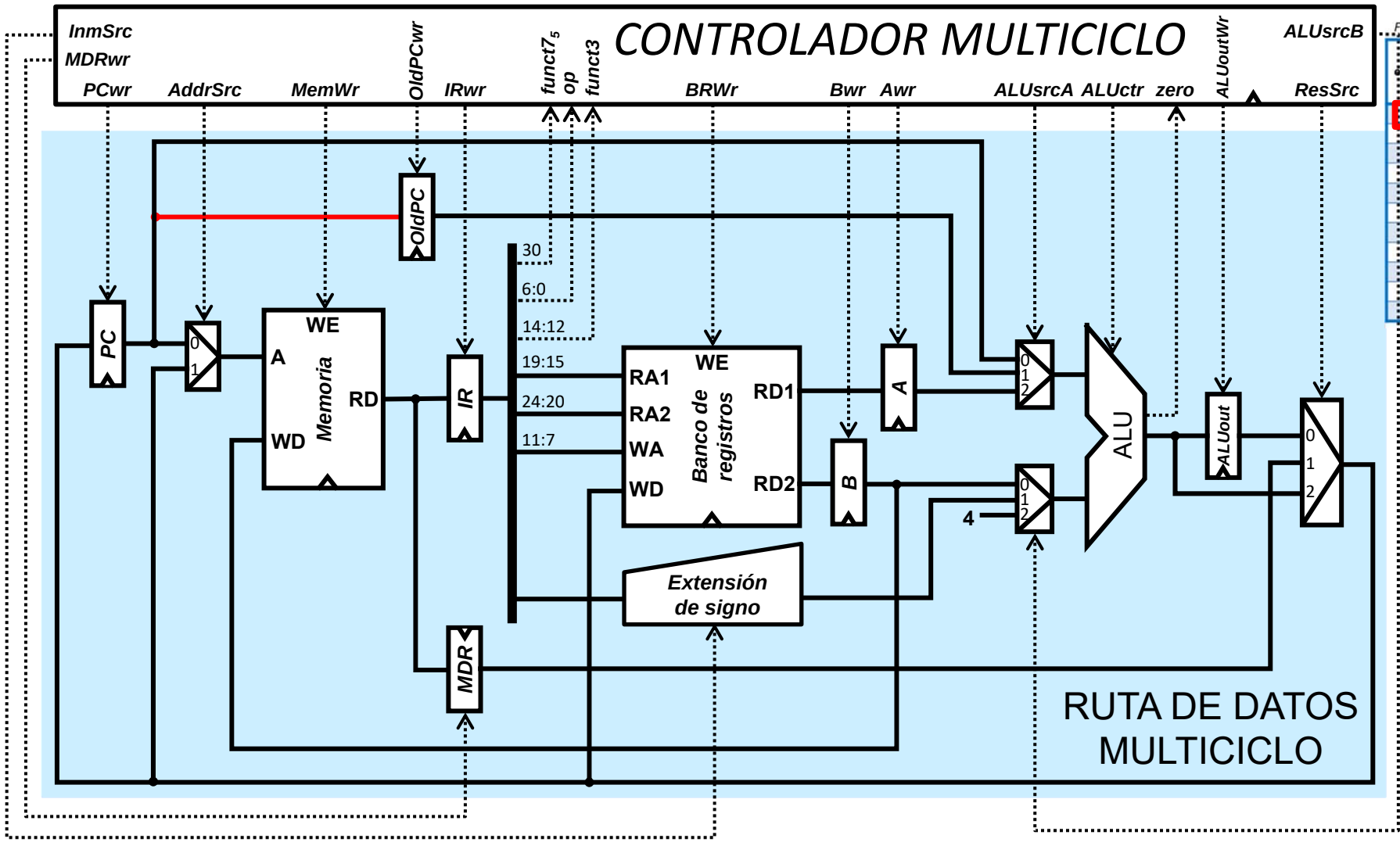



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

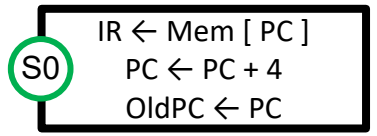



- OldPCwr toma el valor 1 ya que escribimos sobre el registro OldPC el valor del PC correspondiente a instrucción que hemos leído de memoria

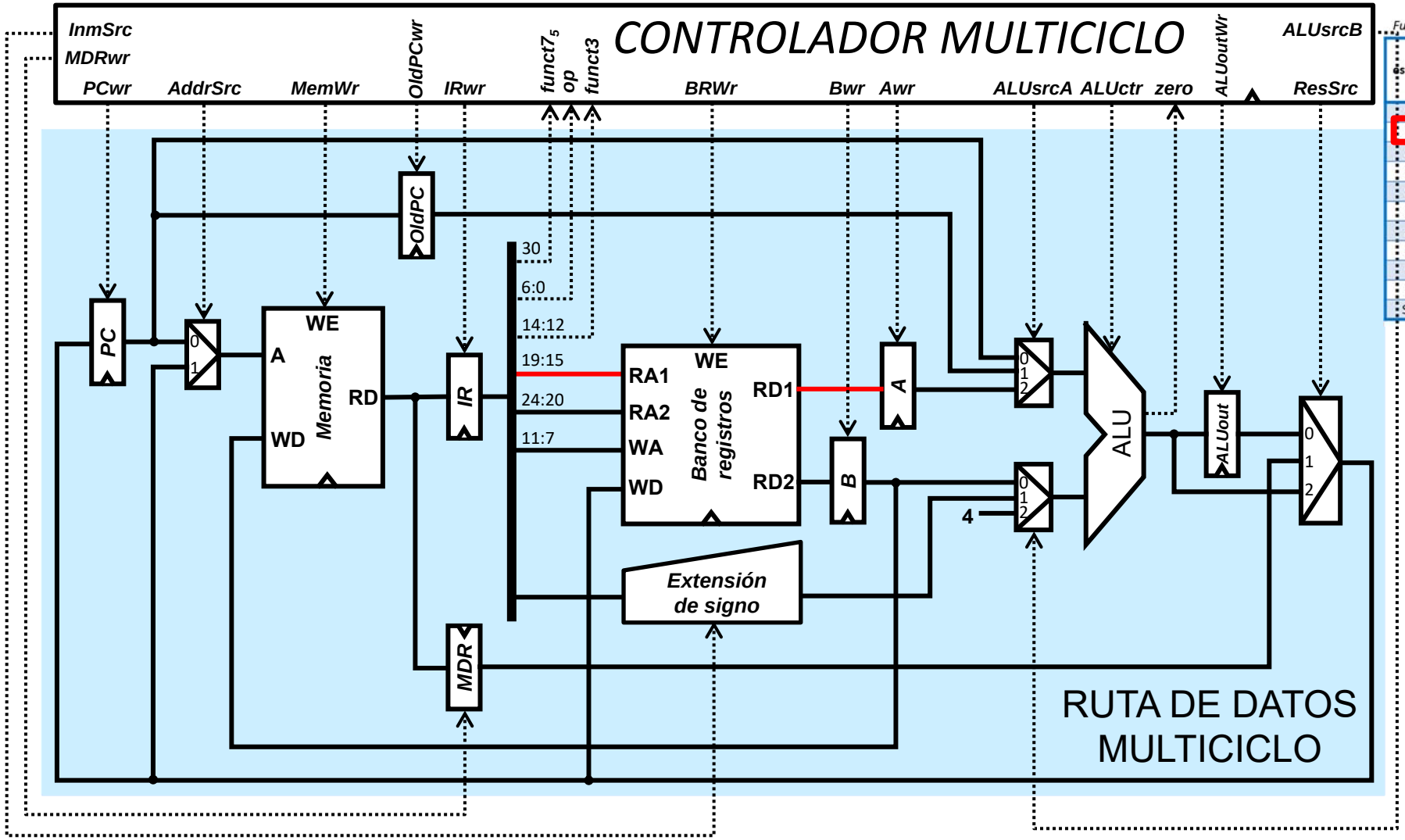



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Las señales MDRwr, BRwr, Awr y Bwr están a 0 porque no se escribe sobre el registro MDR, ni sobre el banco de registros, ni sobre los registros A y B



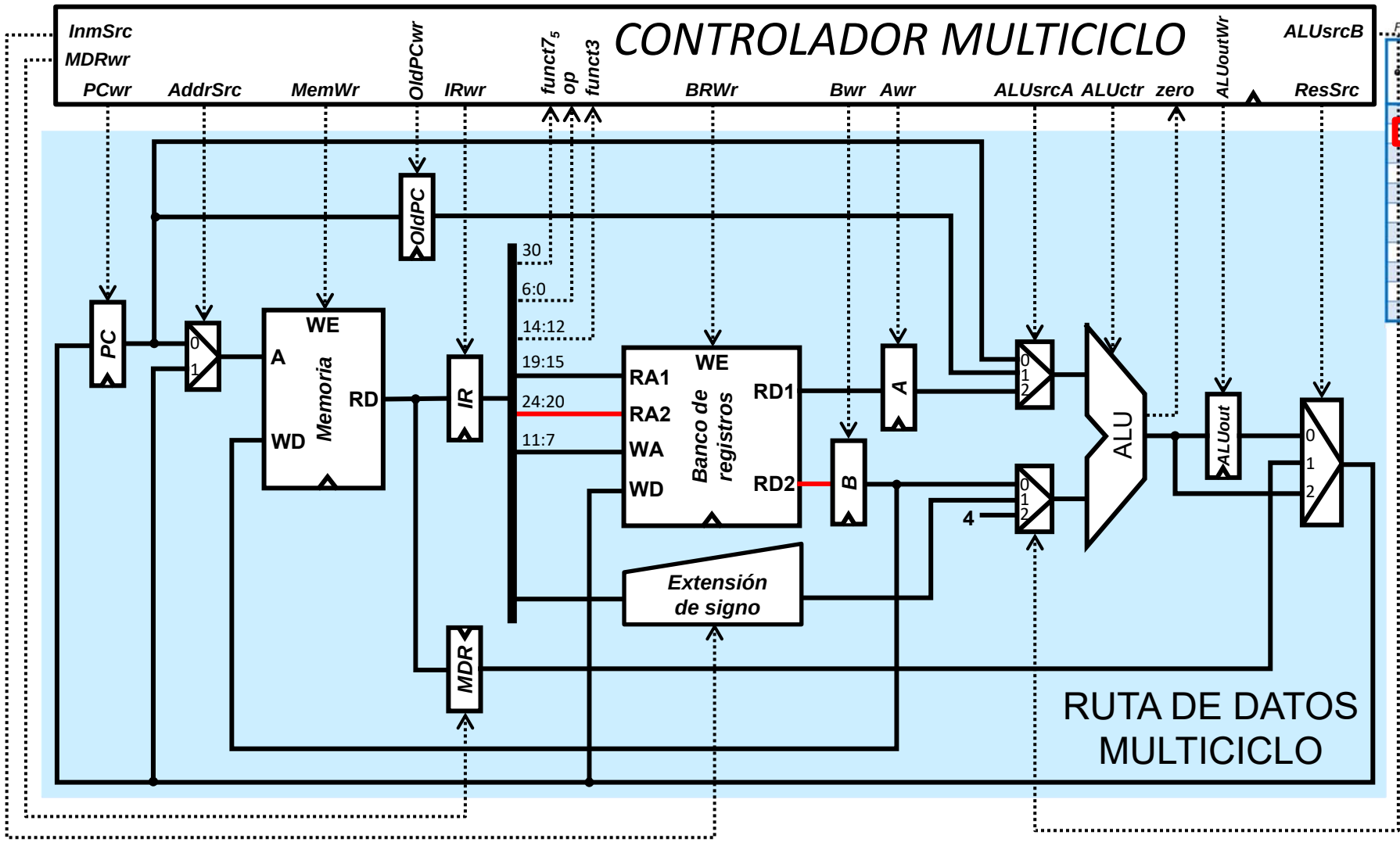

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 1 se escribe en el registro A, por lo que Awr=1



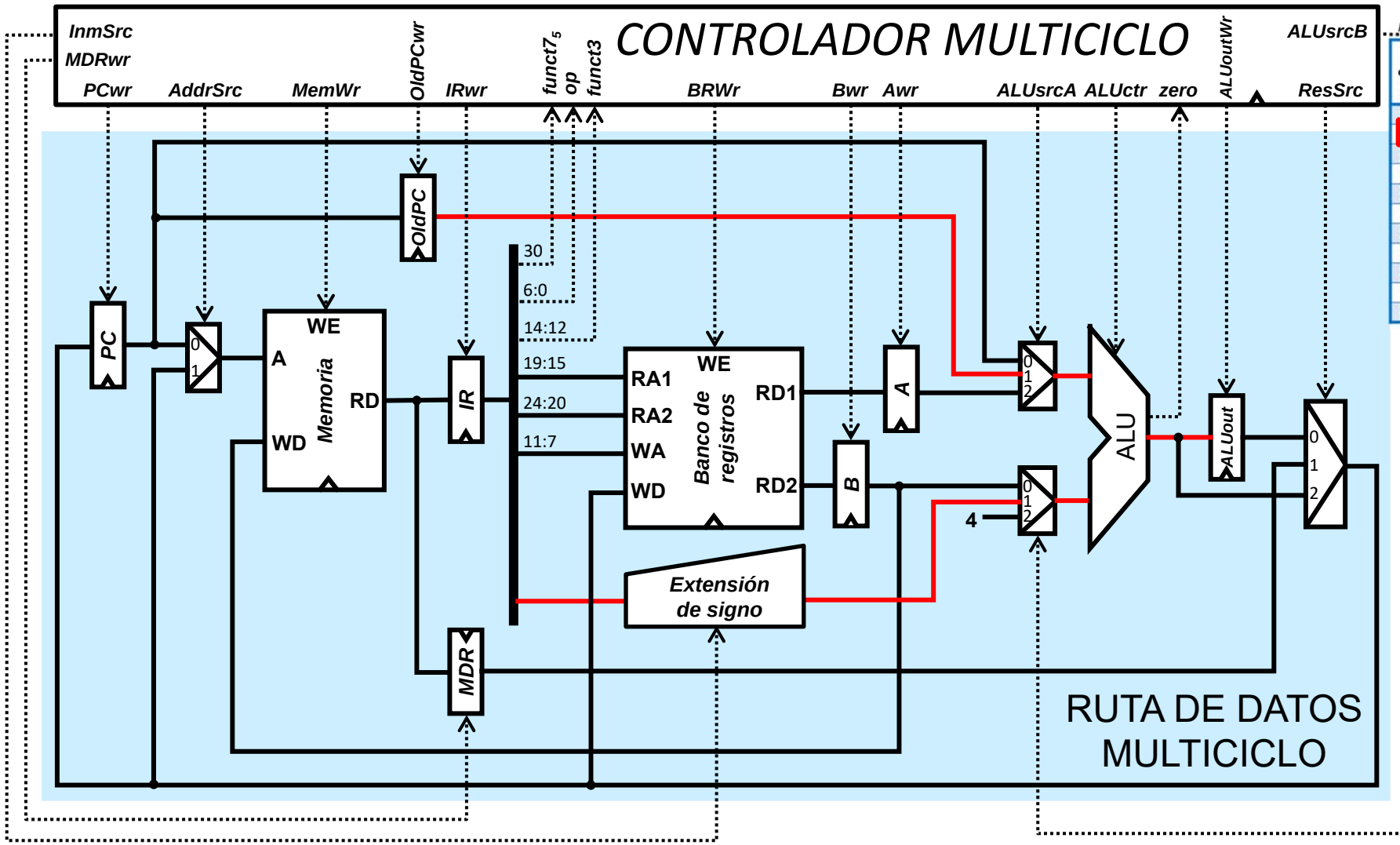

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 2 se escribe en el registro B, por lo que Bwr=1



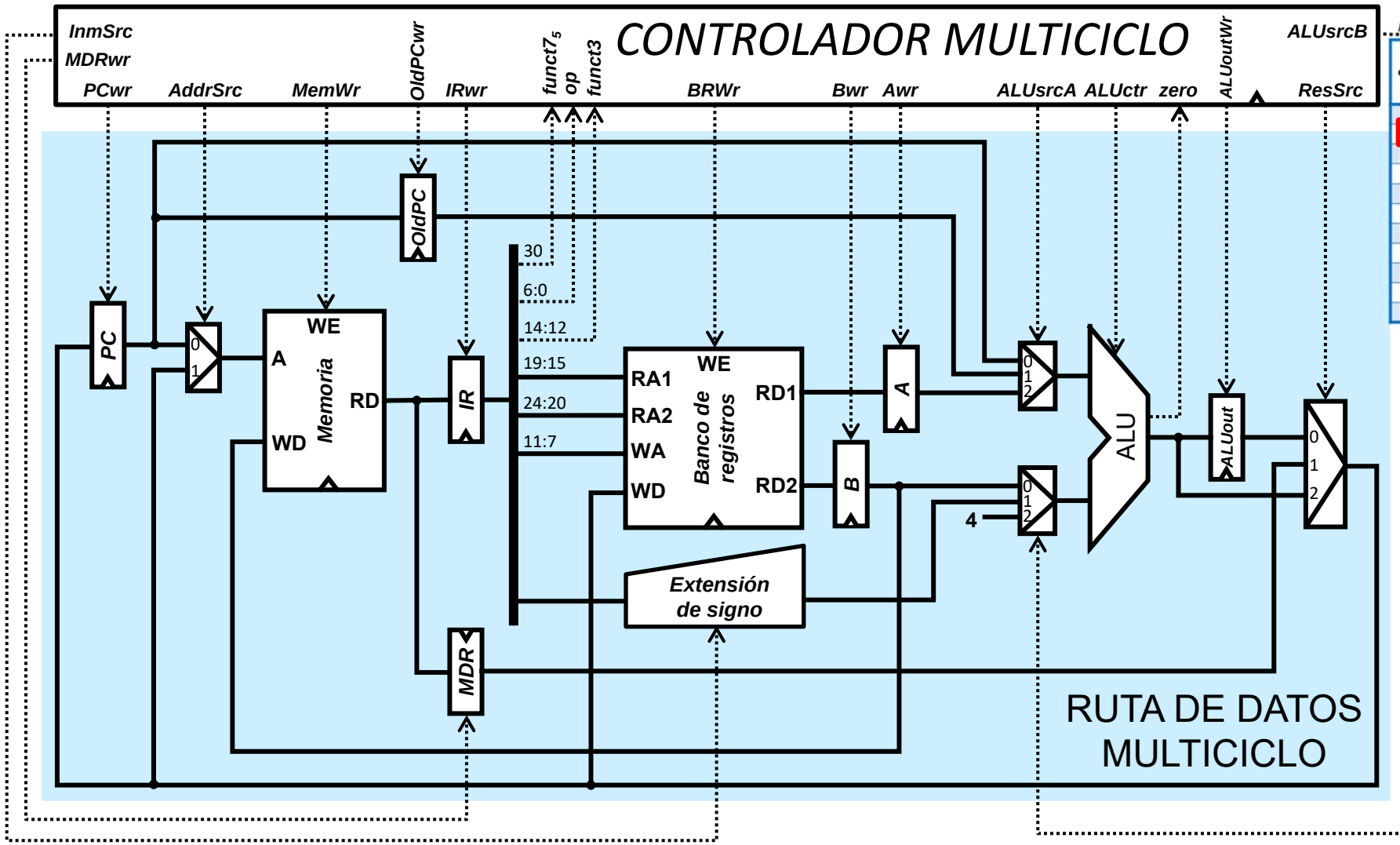

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- En la ALU se realiza la suma (por tanto ALUop=00) de oldPC + sExt(imm). El valor contenido en OldPC llega a la entrada superior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcA=01. El valor del inmediato con el signo extendido llega a la entrada inferior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcB=01. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.



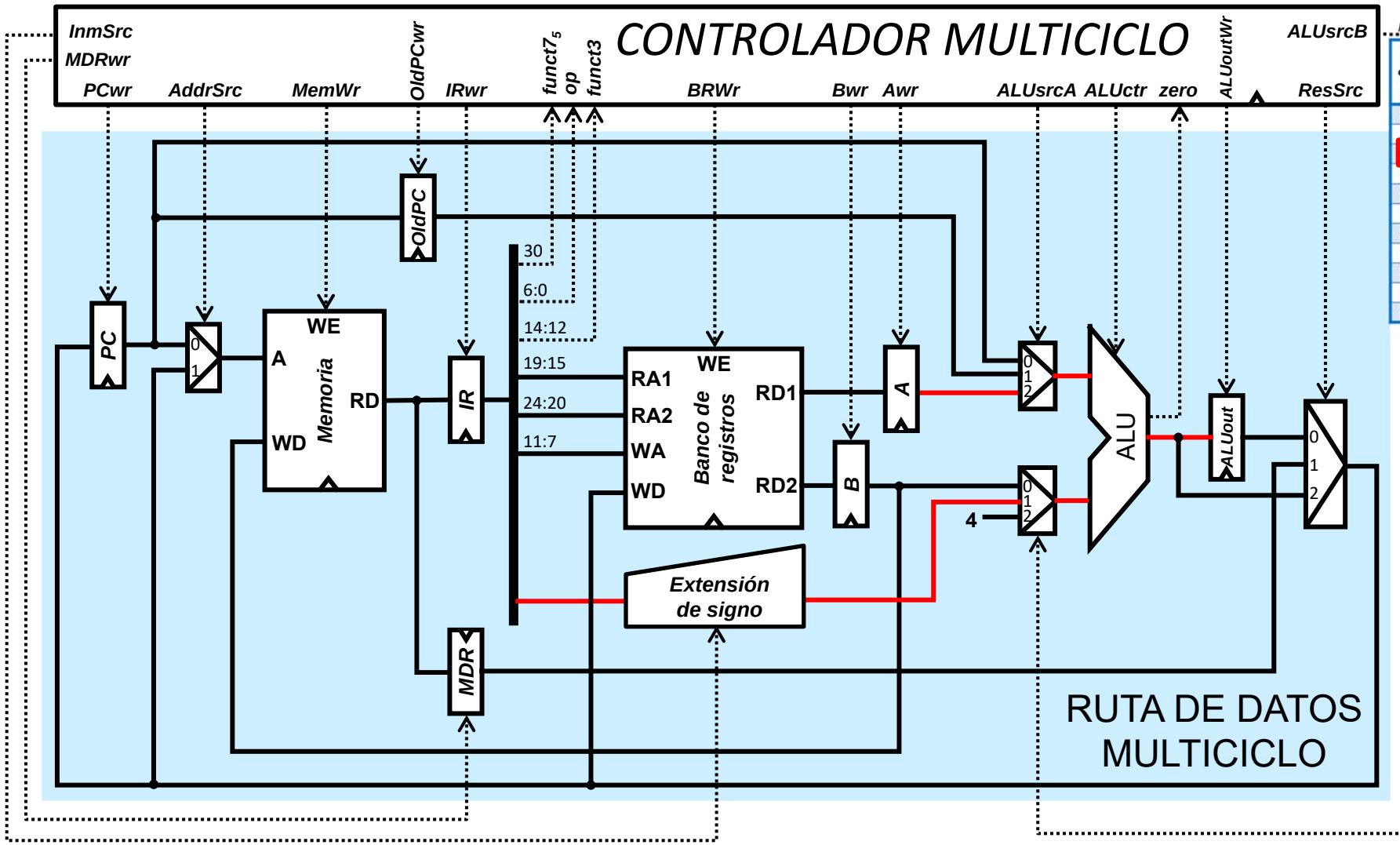

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

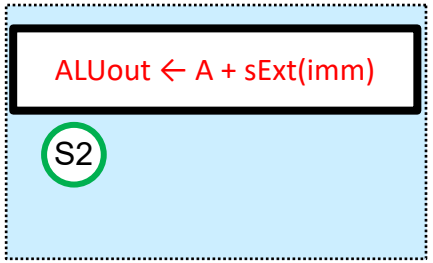
S1

- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr y BRwr están a 0 porque en este ciclo no se escribe sobre el registro OldPC, ni sobre el banco de registros
- El valor de ResSrc es indiferente porque no vamos a escribir en este ciclo ni en el banco de registros ni en el PC

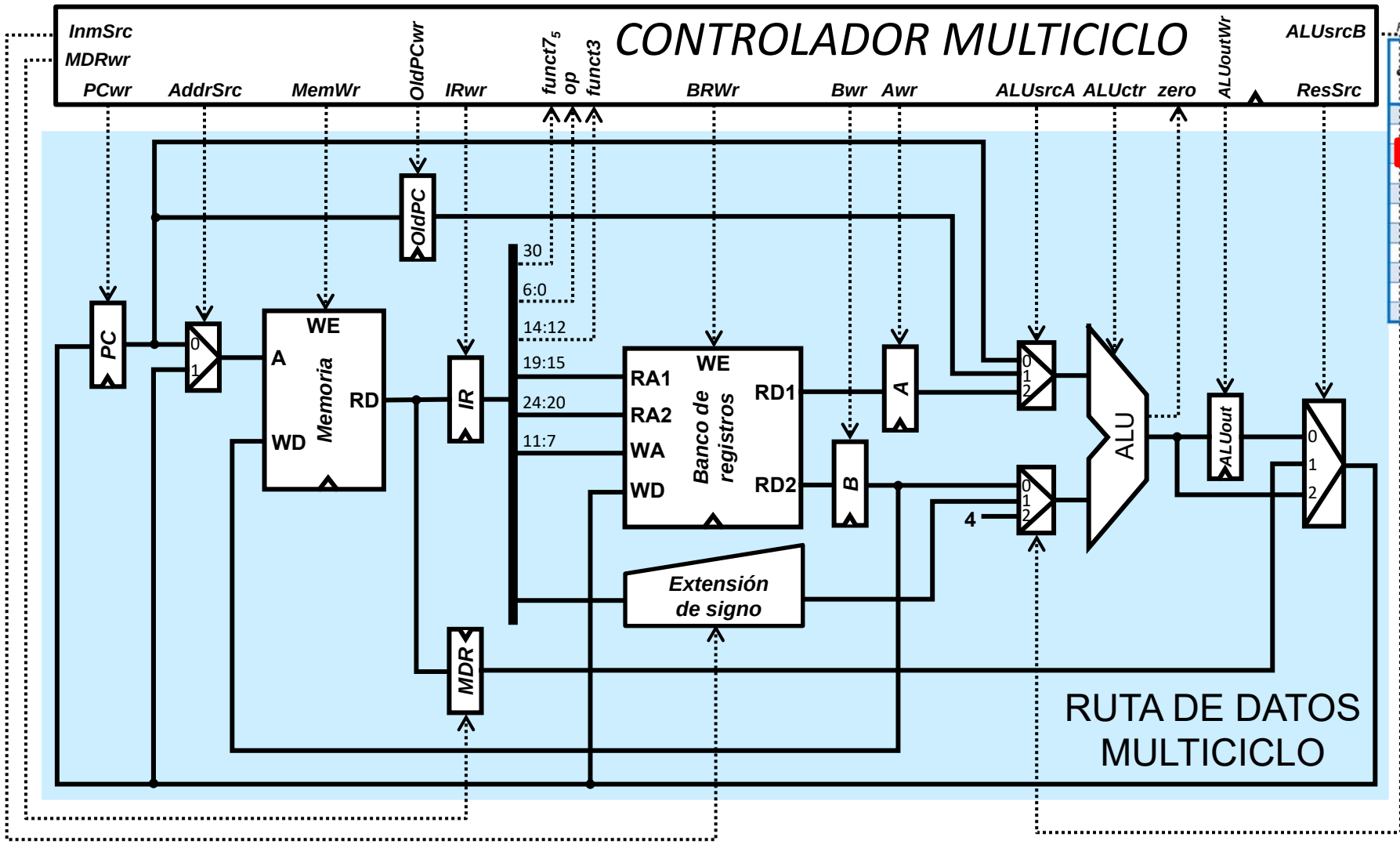



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

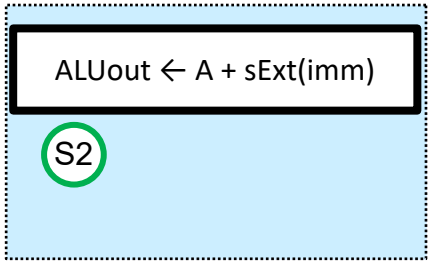



- En la ALU se realiza la suma (por tanto ALUop=00) de A + sExt(imm). El valor contenido en A llega a la entrada superior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcA=10. El valor del inmediato con el signo extendido llega a la entrada inferior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcB=01. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.

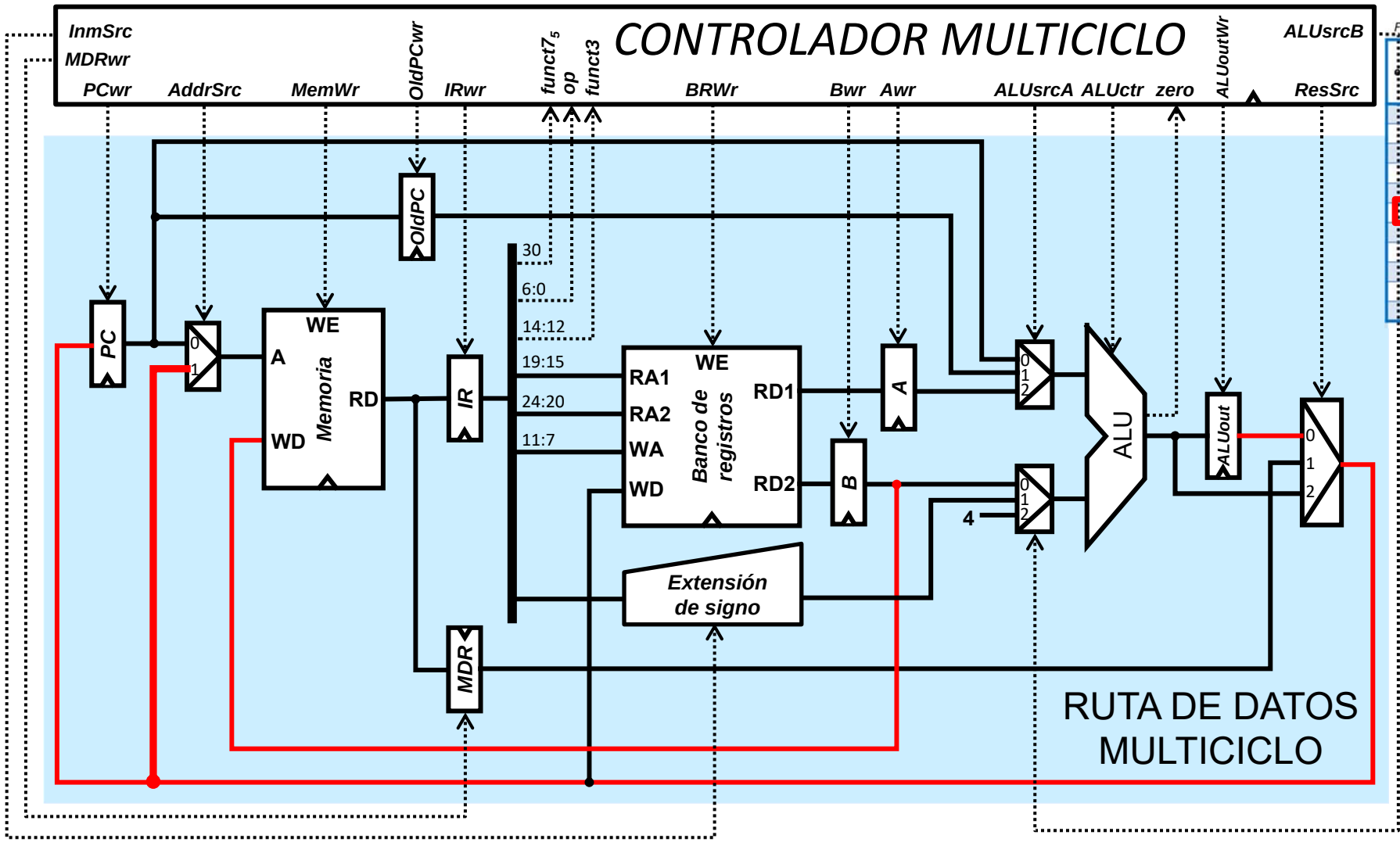



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |



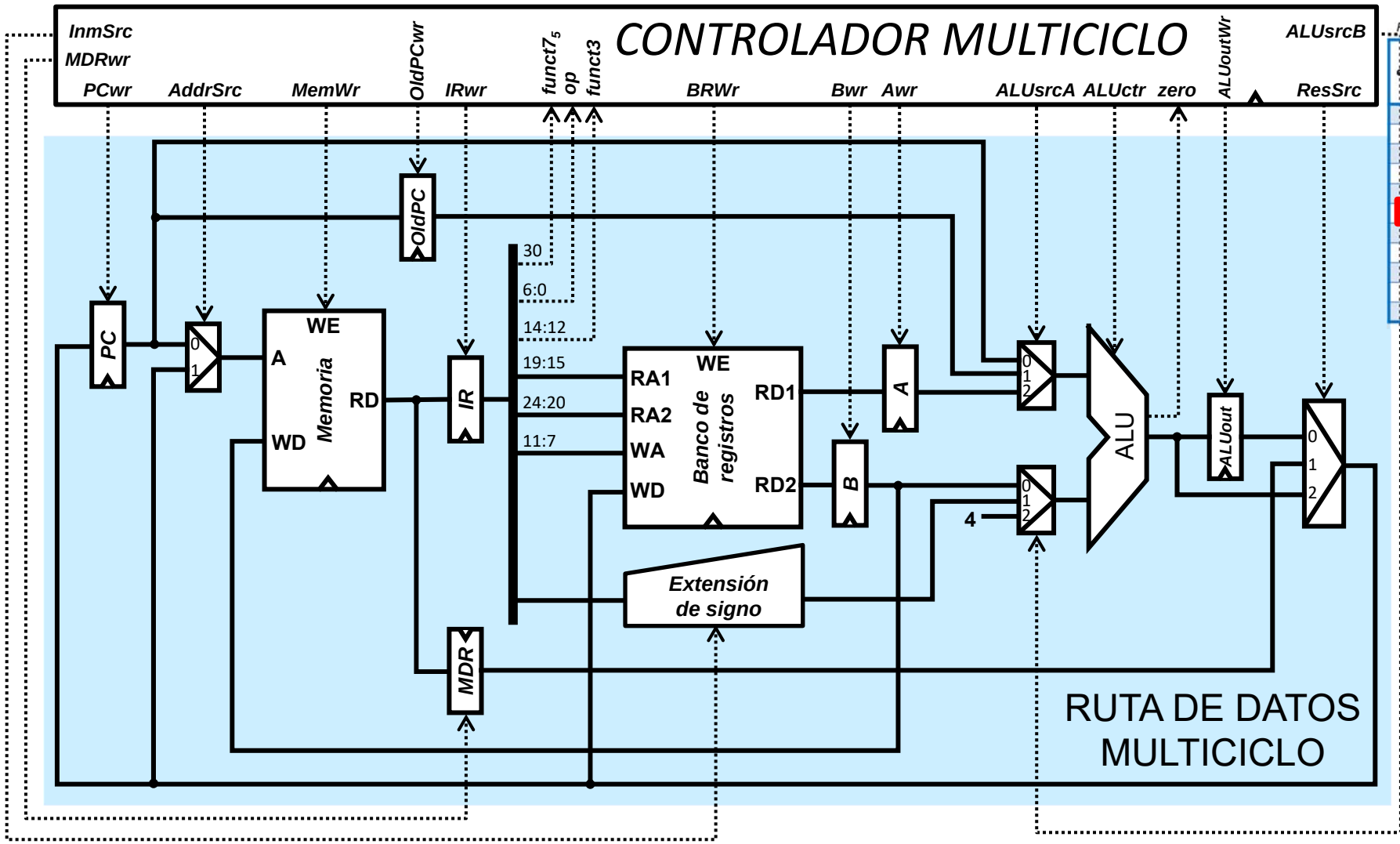

- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr, BRwr, Awr y Bwr están a 0 porque en este ciclo no se escribe sobre el registro OldPC, ni sobre el banco de registros ni sobre los registros A y B
- El valor de ResSrc es indiferente porque no vamos a escribir en este ciclo ni en el banco de registros ni en el PC




Función de salida

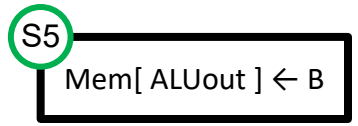
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |


- El contenido de B se escribe sobre la memoria, por lo que MemWr=1. La dirección de memoria sobre la que se escribe la proporciona el registro ALUout, por lo que la señal ResSrc toma el valor 00 para llevar esa dirección al multiplexor de entrada de la memoria a través de la entra 1 de dicho mux, por lo que AddrSrc=1



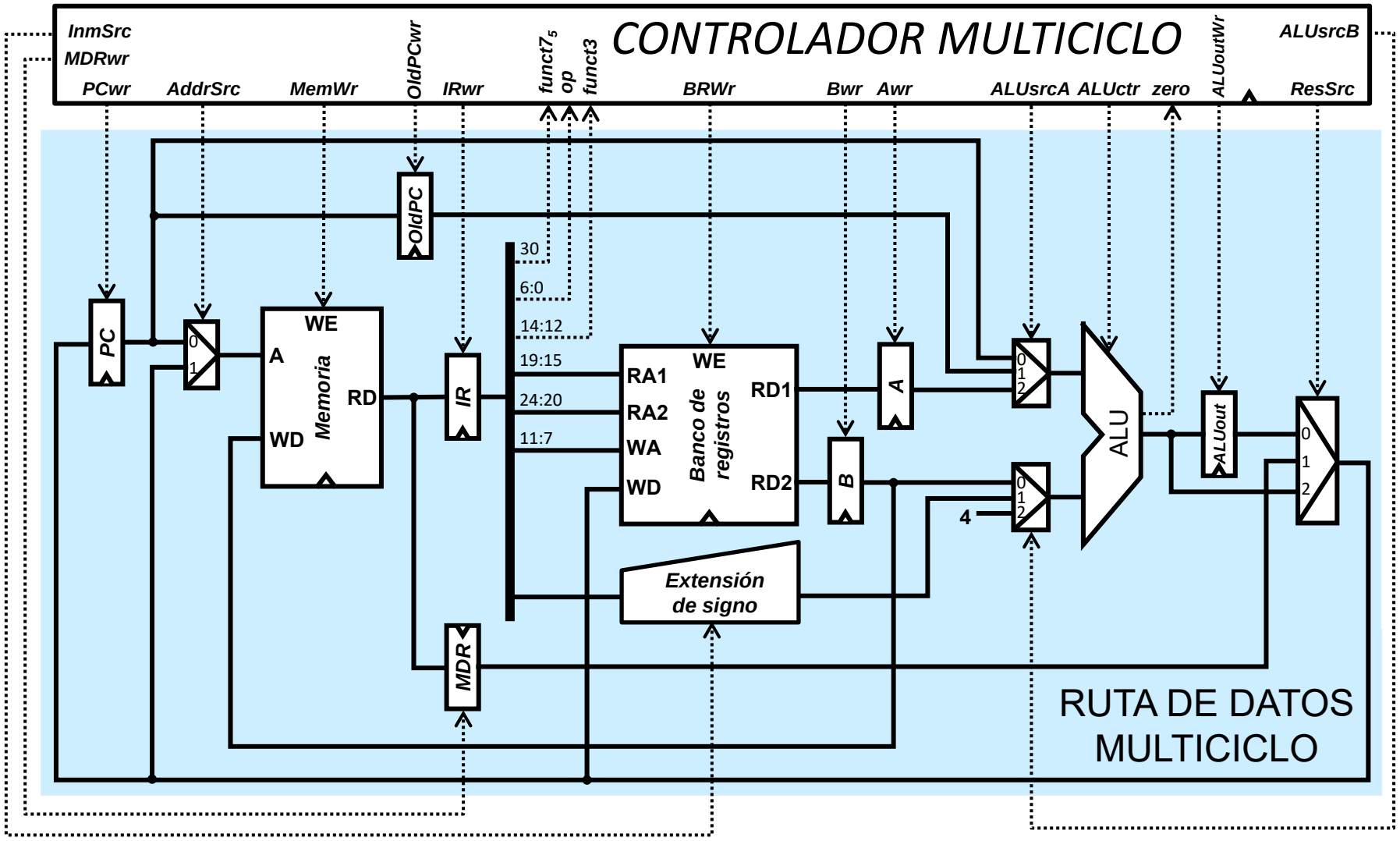

Función de salida

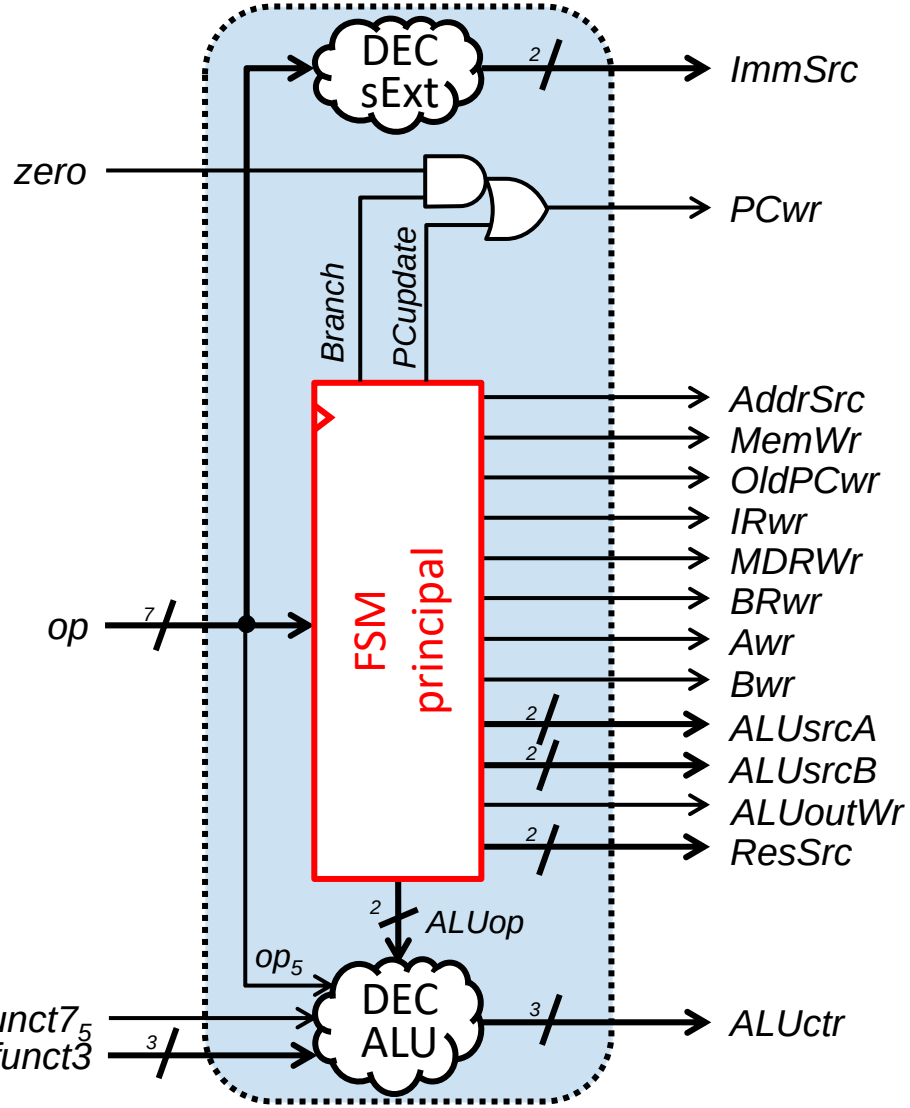
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- OldPCwr, IRwr, MDRwr, BRwr, Awr y Bwr están a 0 porque en este ciclo no se escribe sobre los registros OldPC, IR, MDR, A, B ni sobre el banco de registros
- La ALU no se utiliza en este ciclo (no se escribe nada sobre su registro de salida, por eso ALUoutWr=0), por lo que los valores de las señales que controlan los muxes que seleccionan los valores de las entradas de la ALU (ALUsrcA y ALUsrcB) son indiferentes. Asimismo, por el mismo motivo, el valor de ALUop también es indiferente

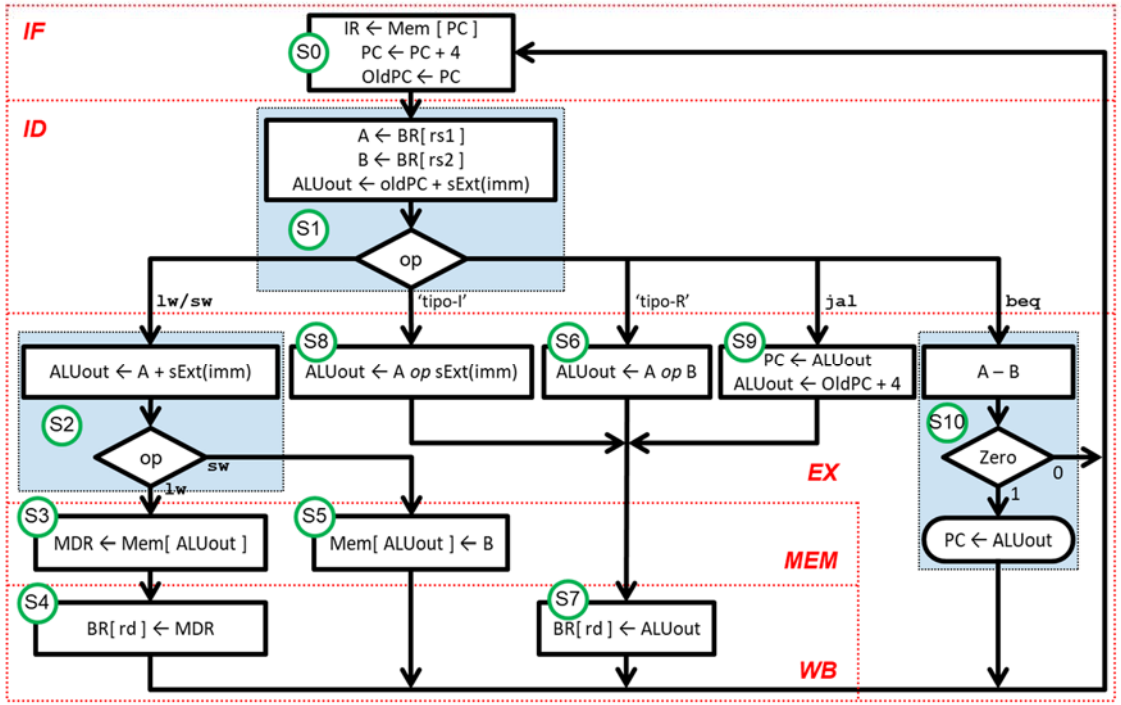
**2)** En el procesador **multiciclo** estudiado en clase, justifica los valores que toman las señales de control para la ejecución de las instrucciones “add”.



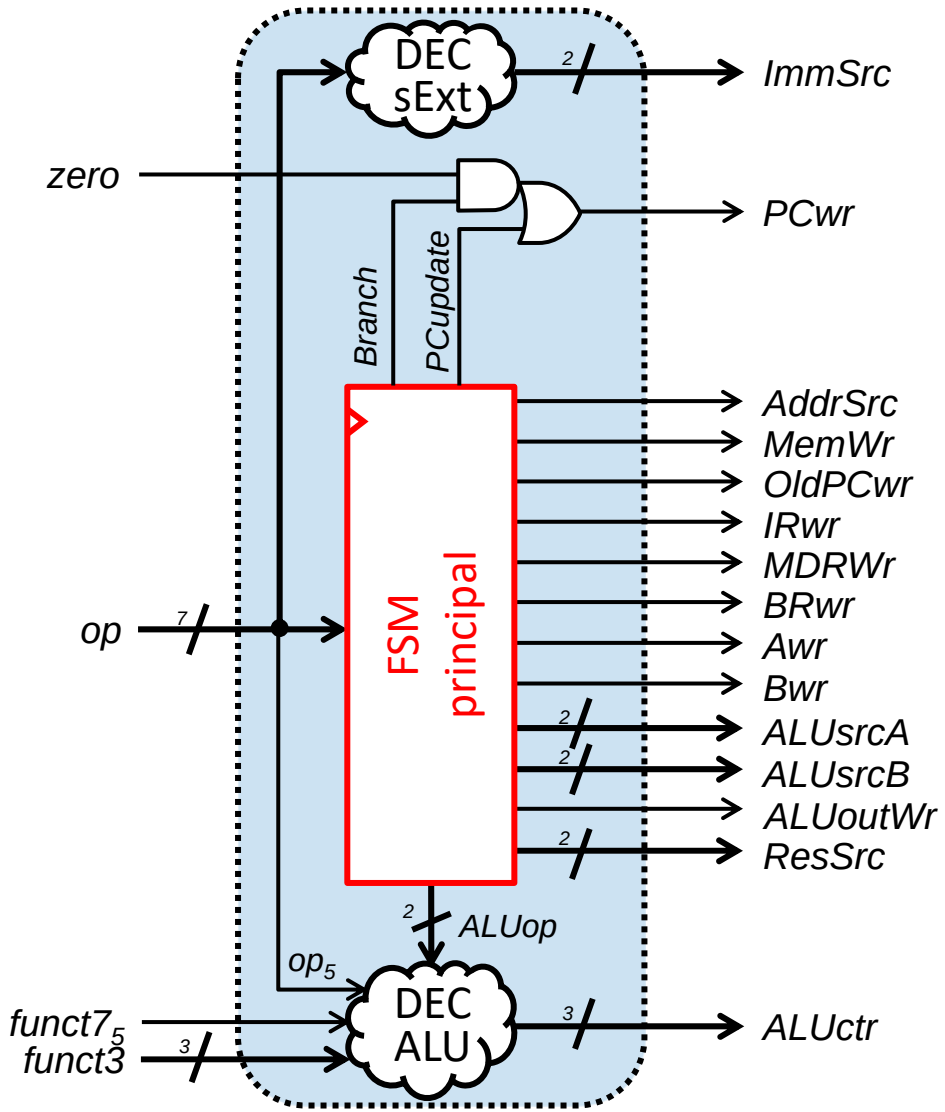



*Función de salida*

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

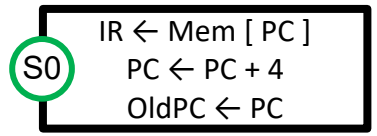



**Las instrucciones add siguen la secuencia de estados S0-S1-S6-S7**

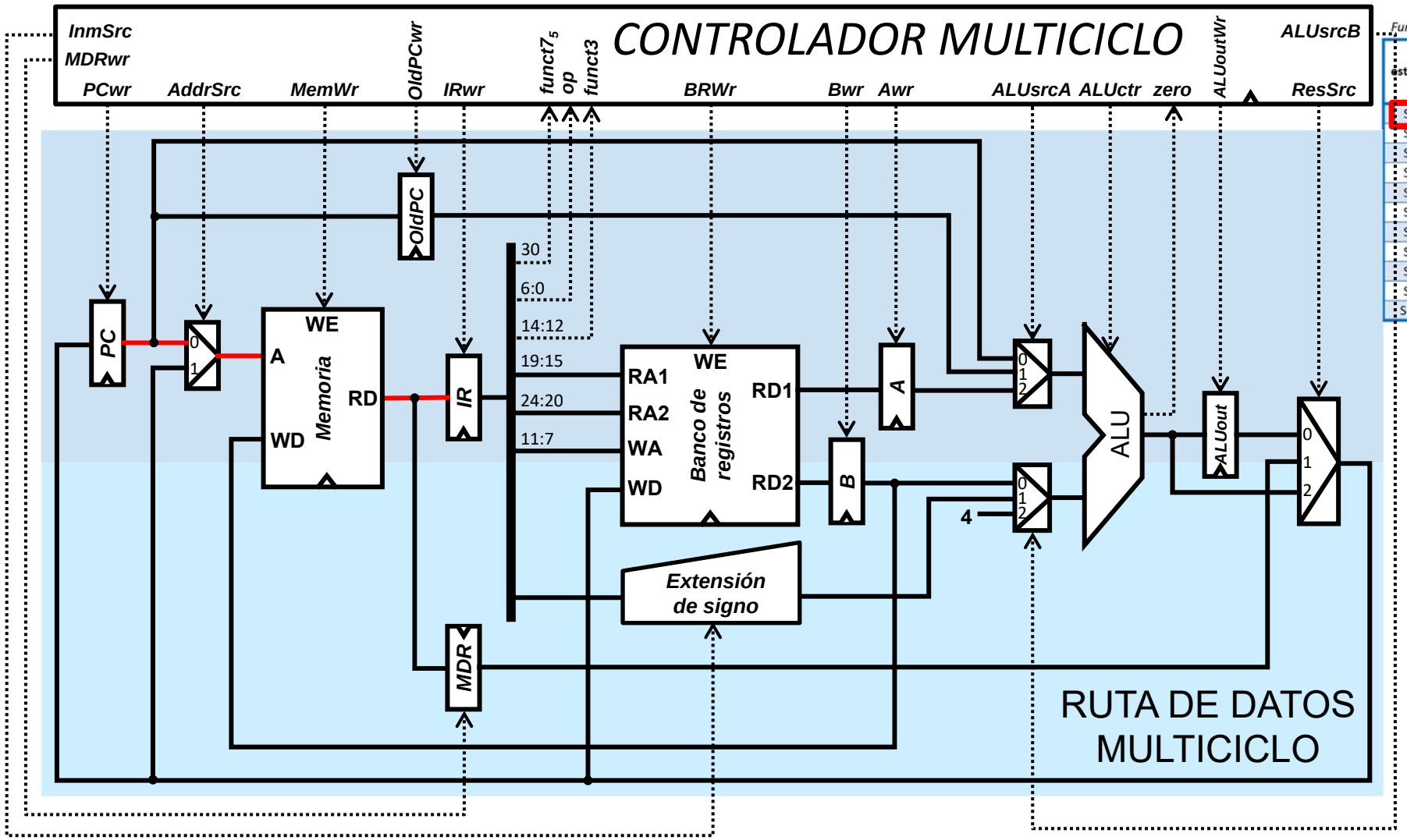



*Función de salida*

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

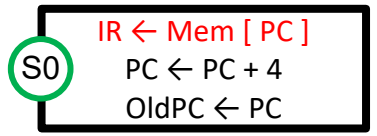



- La señal Branch está a 0 ya que no se trata de una instrucción beq
- La señal PCupdate está a 1 ya que el PC se actualiza con el valor PC+4




Función de salida

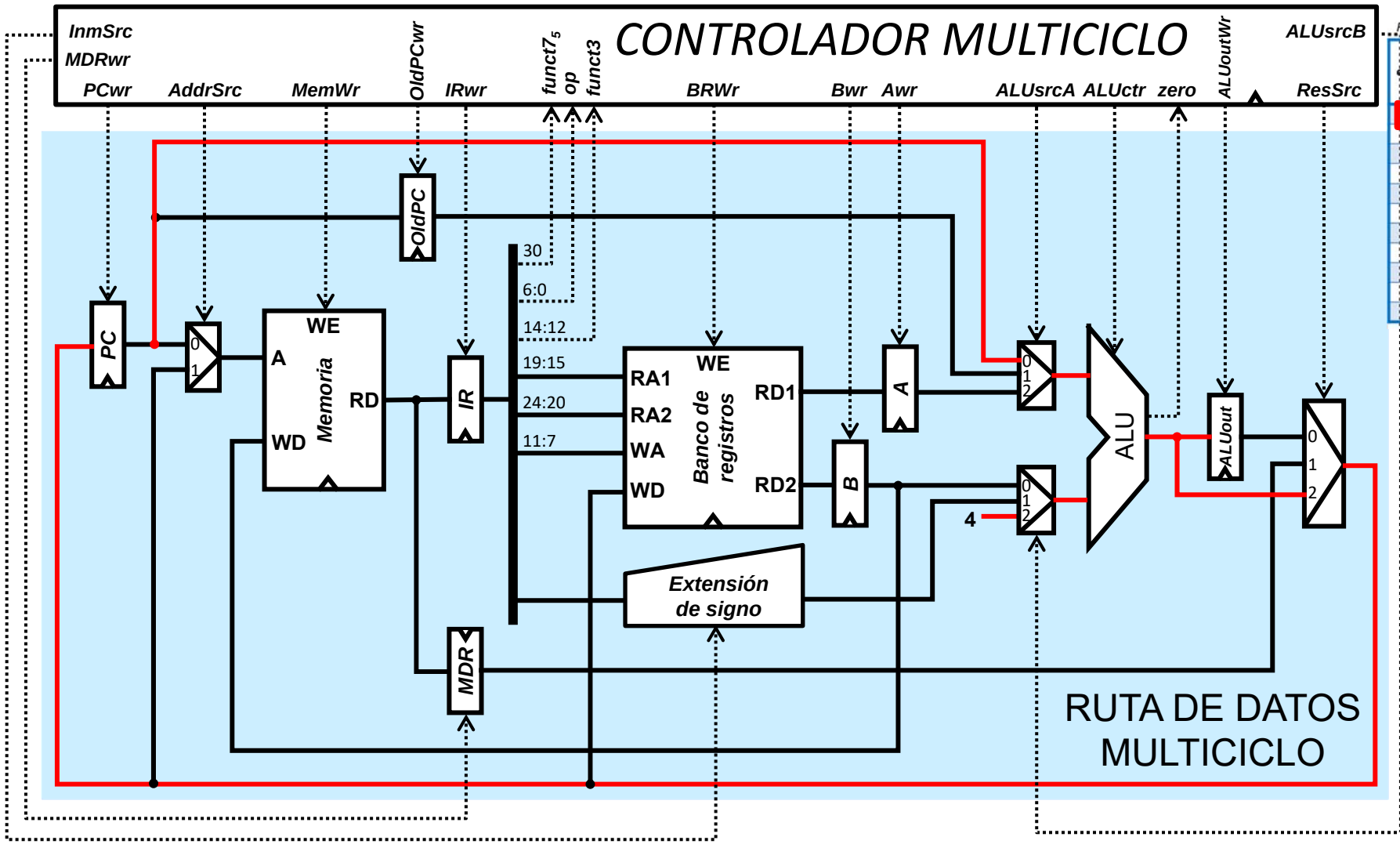
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- AddrSrc toma el valor 0 ya que se lee la dirección de memoria indicada por el PC para obtener la instrucción

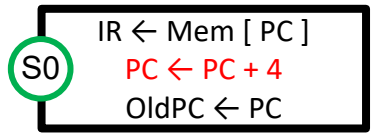
- IRwr toma el valor 1 ya que la instrucción leída de memoria se escribe en el registro IR

- MemWr toma el valor 0 ya que en este ciclo la memoria se lee, pero no se escribe

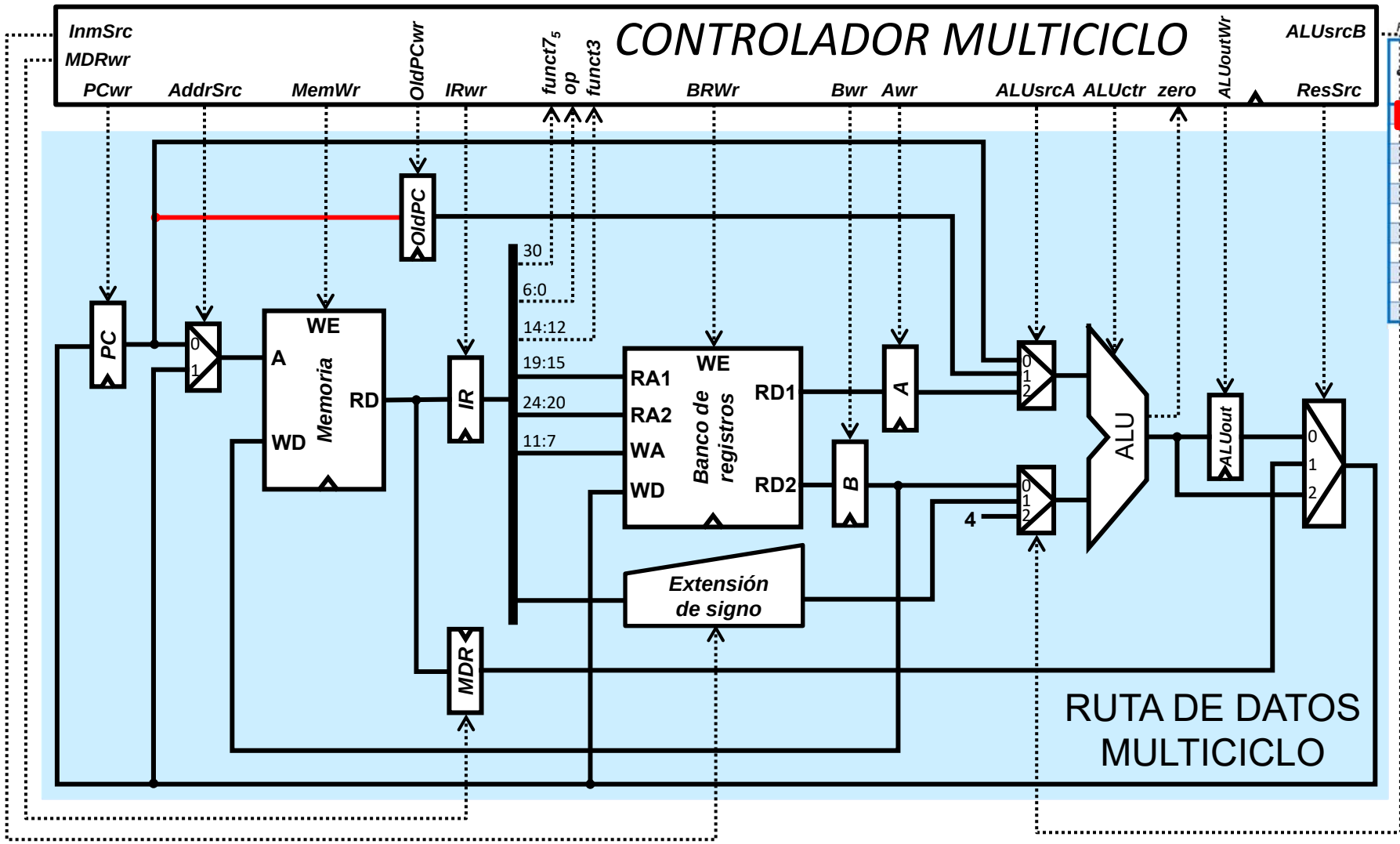



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

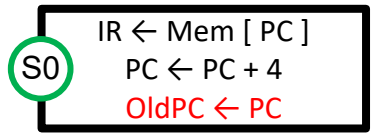



- En la ALU se realiza la operación PC+4 (ALUop=00). El valor del PC llega a la entrada superior de la ALU a través de la entrada 0 del mux correspondiente, por lo que ALUsrcA vale 00. El valor 4 entra a la entrada inferior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcB toma el valor 10. El resultado no se escribe en el registro ALUout (por tanto la señal ALUoutWr vale 0), sino que se lleva directamente al registro PC para actualizar su valor, a través de la entrada 2 del multiplexor controlado por ResSrc (10)

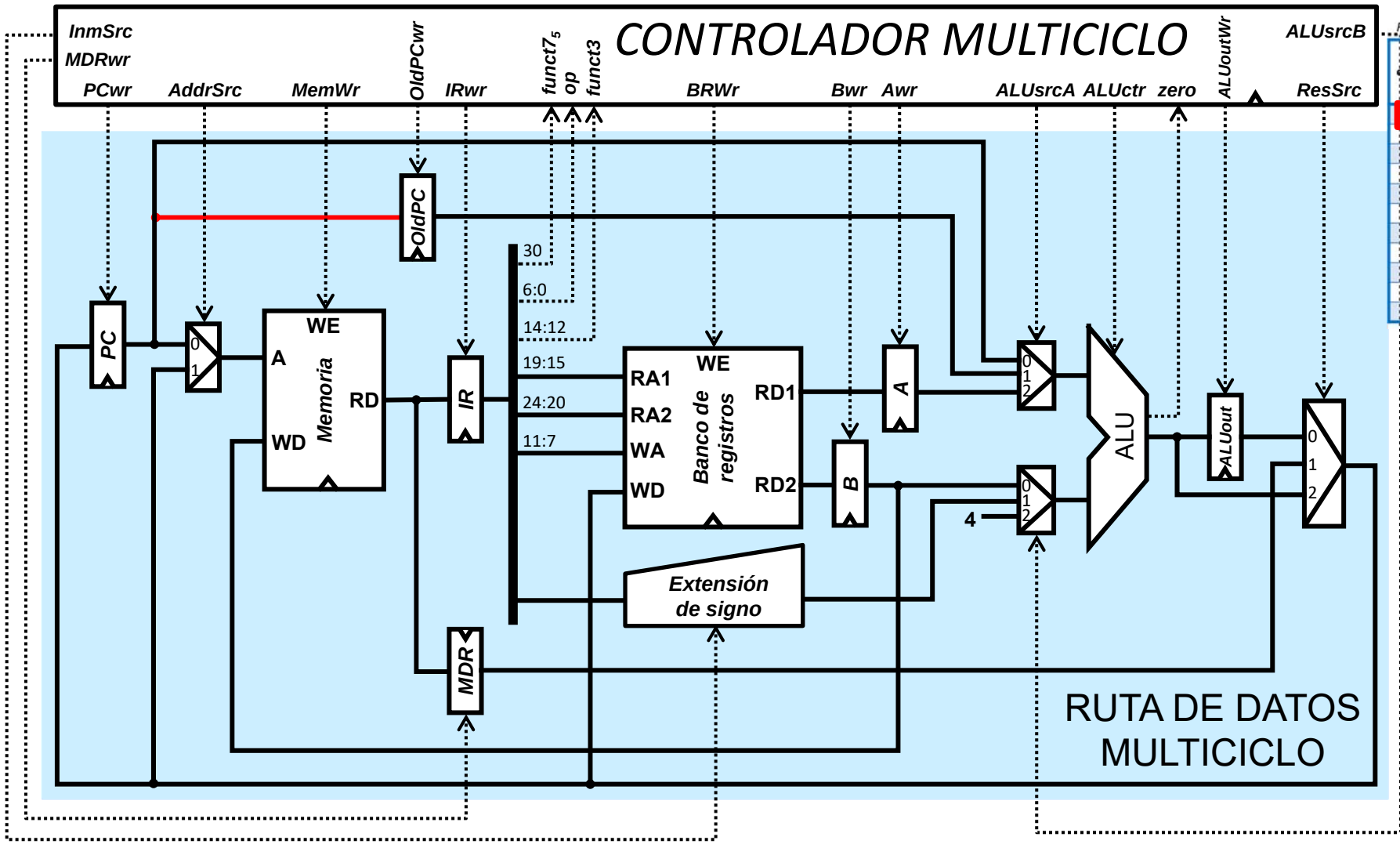



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

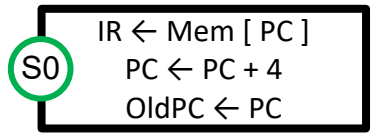



- OldPCwr toma el valor 1 ya que escribimos sobre el registro OldPC el valor del PC correspondiente a instrucción que hemos leído de memoria

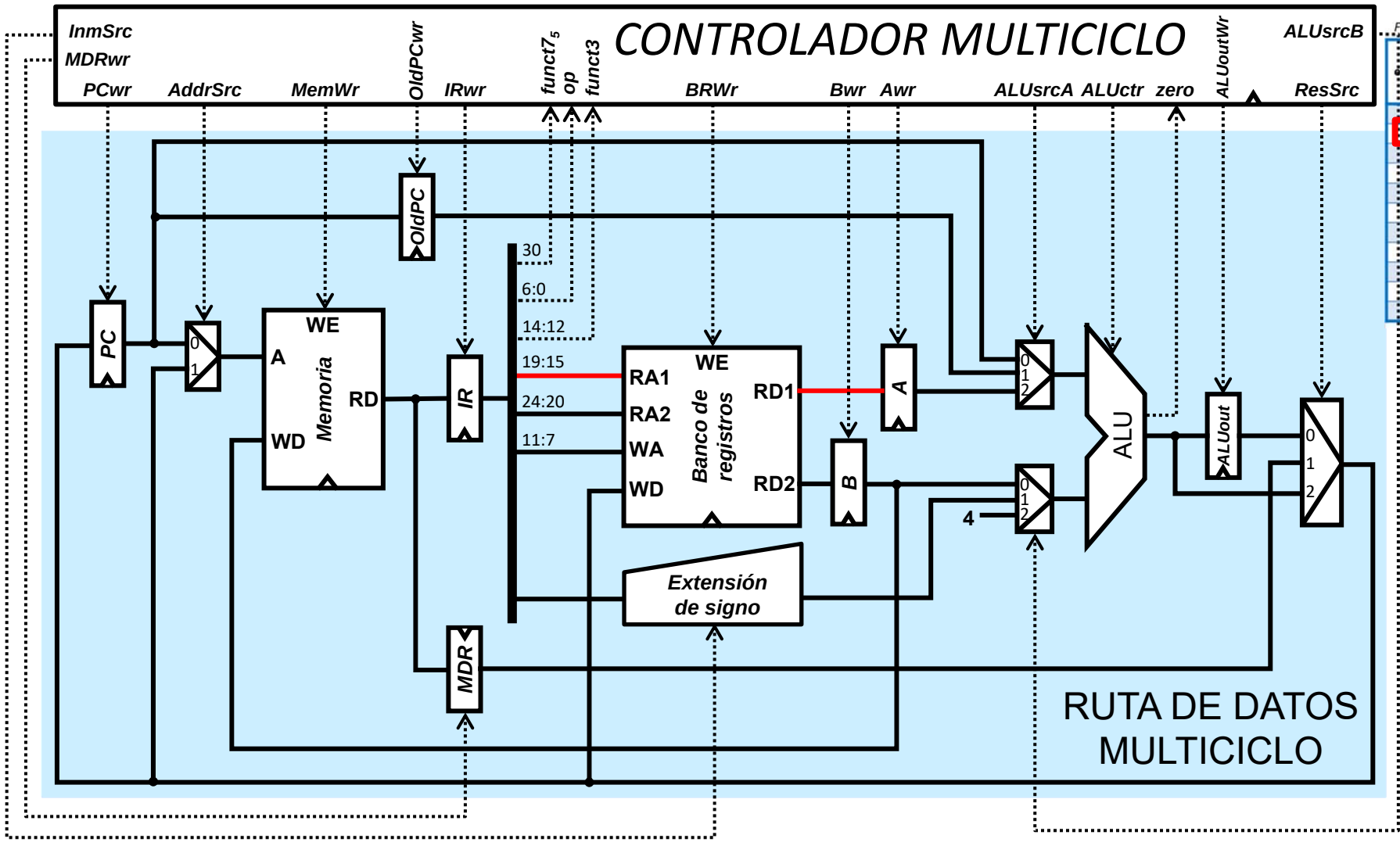



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Las señales MDRwr, BRwr, Awr y Bwr están a 0 porque no se escribe sobre el registro MDR, ni sobre el banco de registros, ni sobre los registros A y B



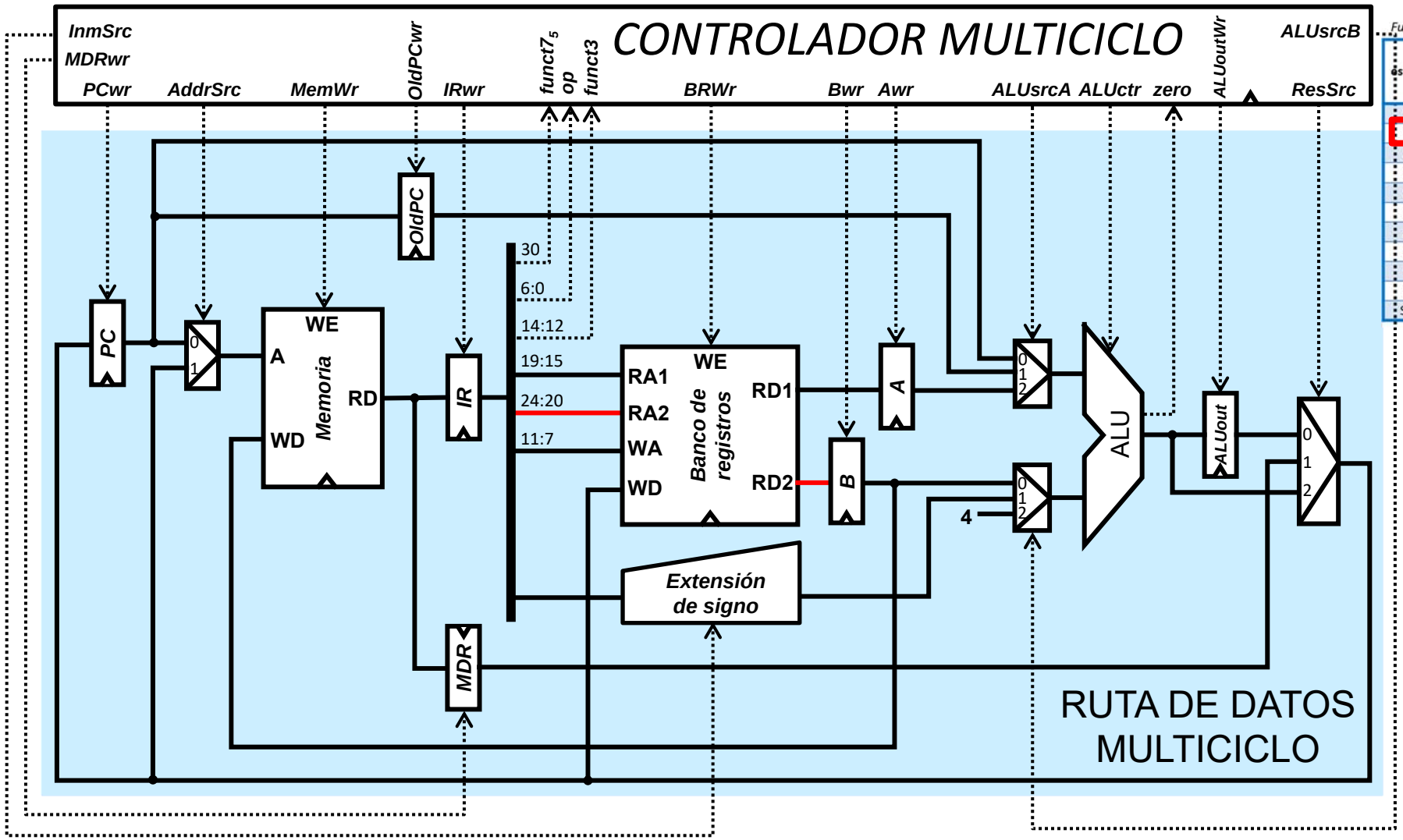

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 1 se escribe en el registro A, por lo que Awr=1



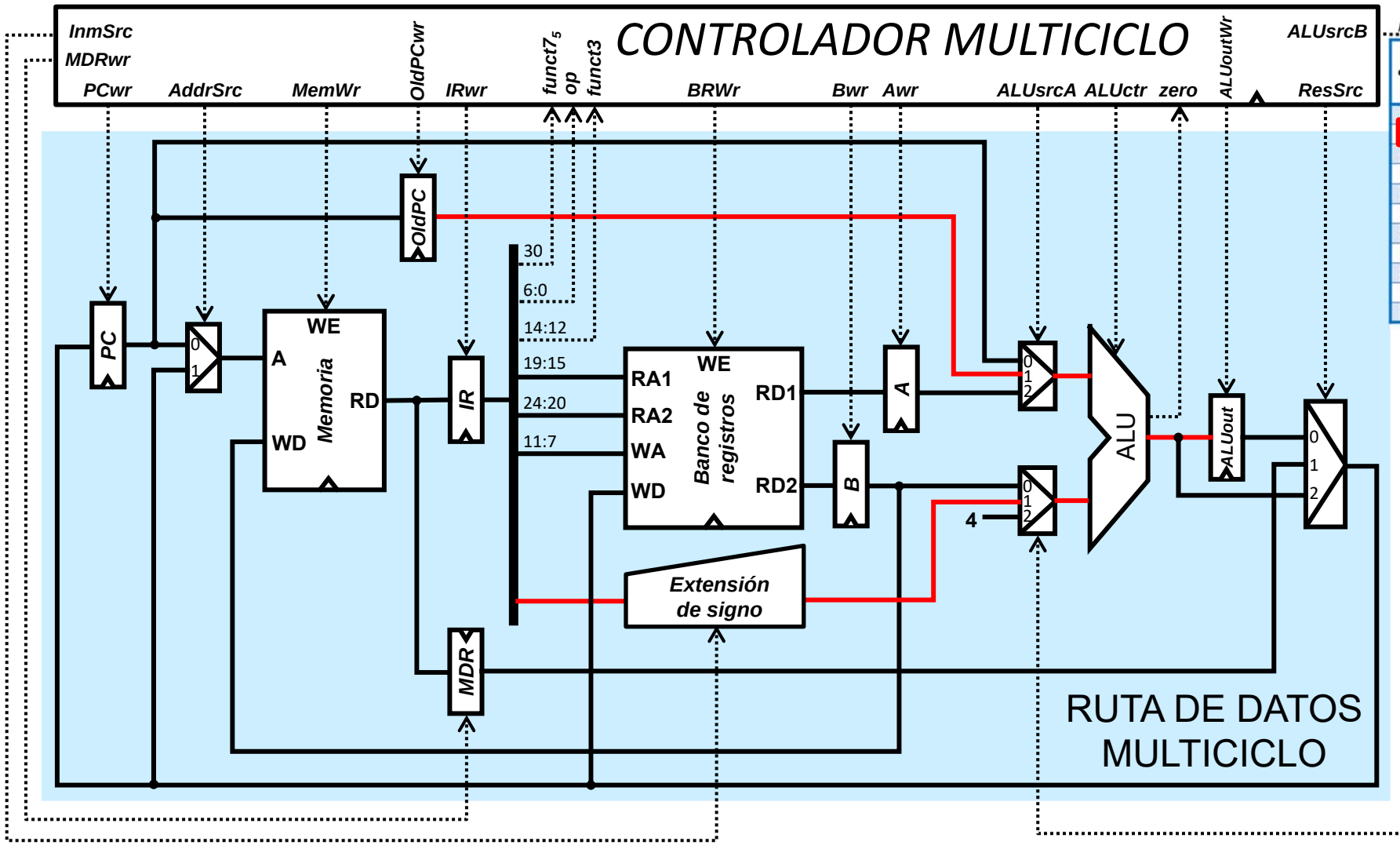

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 2 se escribe en el registro B, por lo que Bwr=1



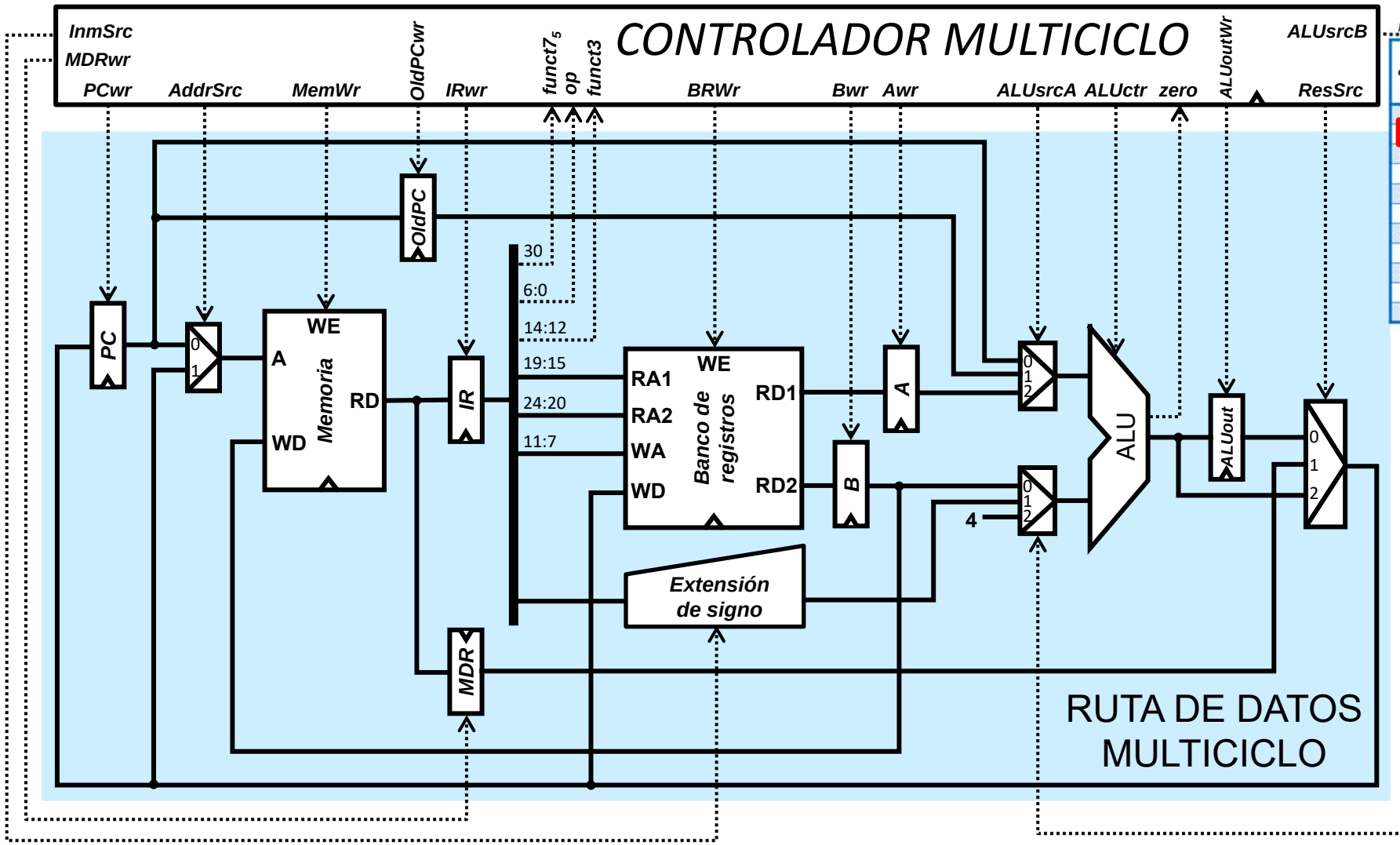

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- En la ALU se realiza la suma (por tanto ALUop=00) de oldPC + sExt(imm). El valor contenido en OldPC llega a la entrada superior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcA=01. El valor del inmediato con el signo extendido llega a la entrada inferior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcB=01. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.



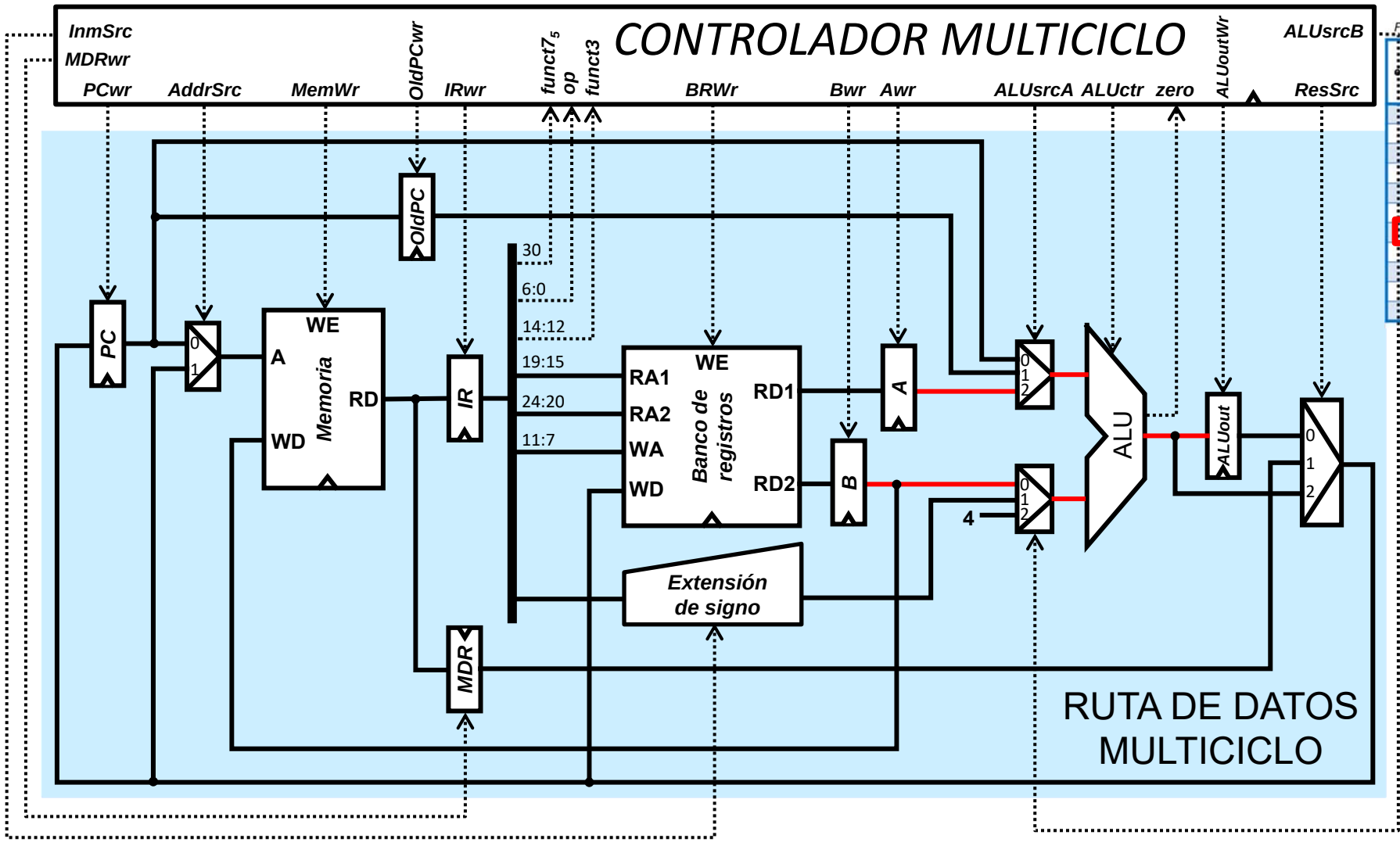

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

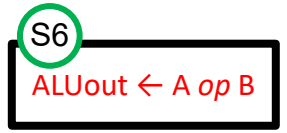
S1

- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr y BRwr están a 0 porque en este ciclo no se escribe sobre el registro OldPC, ni sobre el banco de registros
- El valor de ResSrc es indiferente porque no vamos a escribir en este ciclo ni en el banco de registros ni en el PC

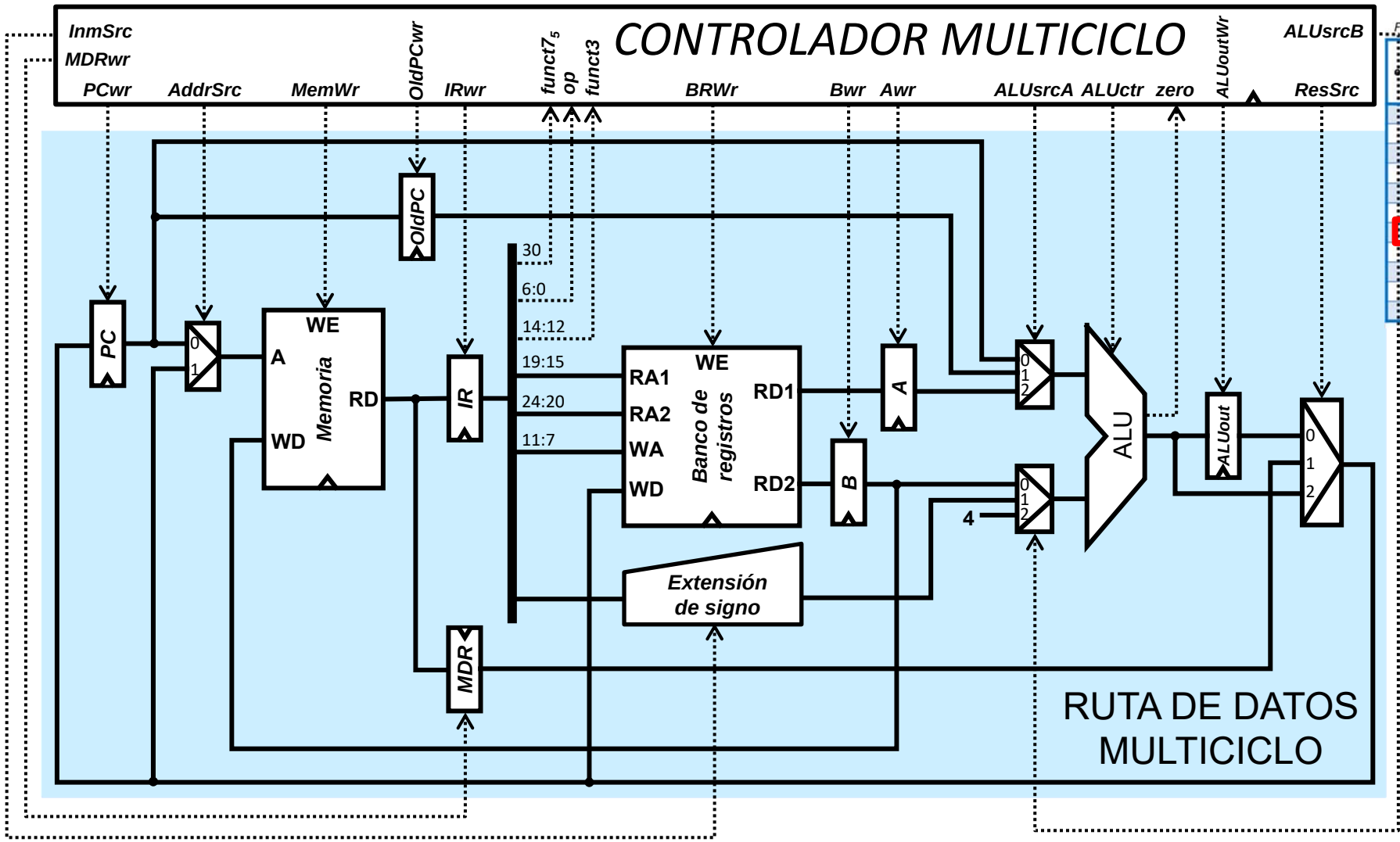



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

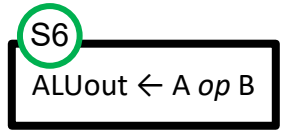



- En la ALU se realiza la operación A + B (en este caso una operación aritmético-lógica de suma, por tanto ALUop=10). El valor contenido en A llega a la entrada superior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcA=10. El valor de B llega a la entrada inferior de la ALU a través de la entrada 0 del mux correspondiente, por lo que ALUsrcB=00. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.

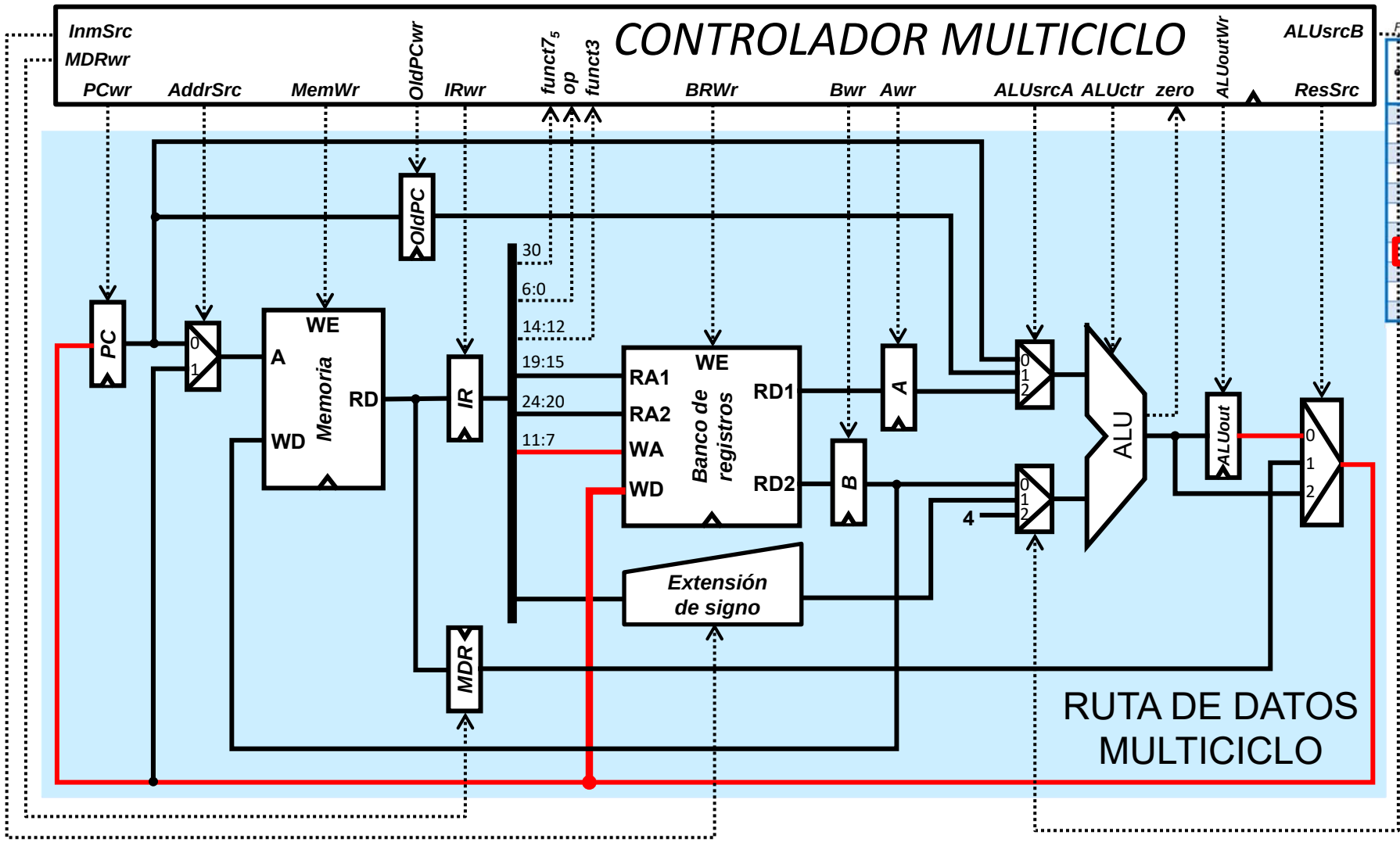



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

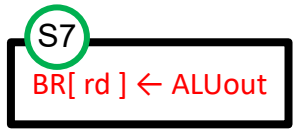



- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr, BRwr, Awr y Bwr están a 0 porque en este ciclo no se escribe sobre los registros OldPC, A, B ni sobre el banco de registros
- El valor de ResSrc es indiferente porque no vamos a escribir en este ciclo ni en el banco de registros ni en el PC

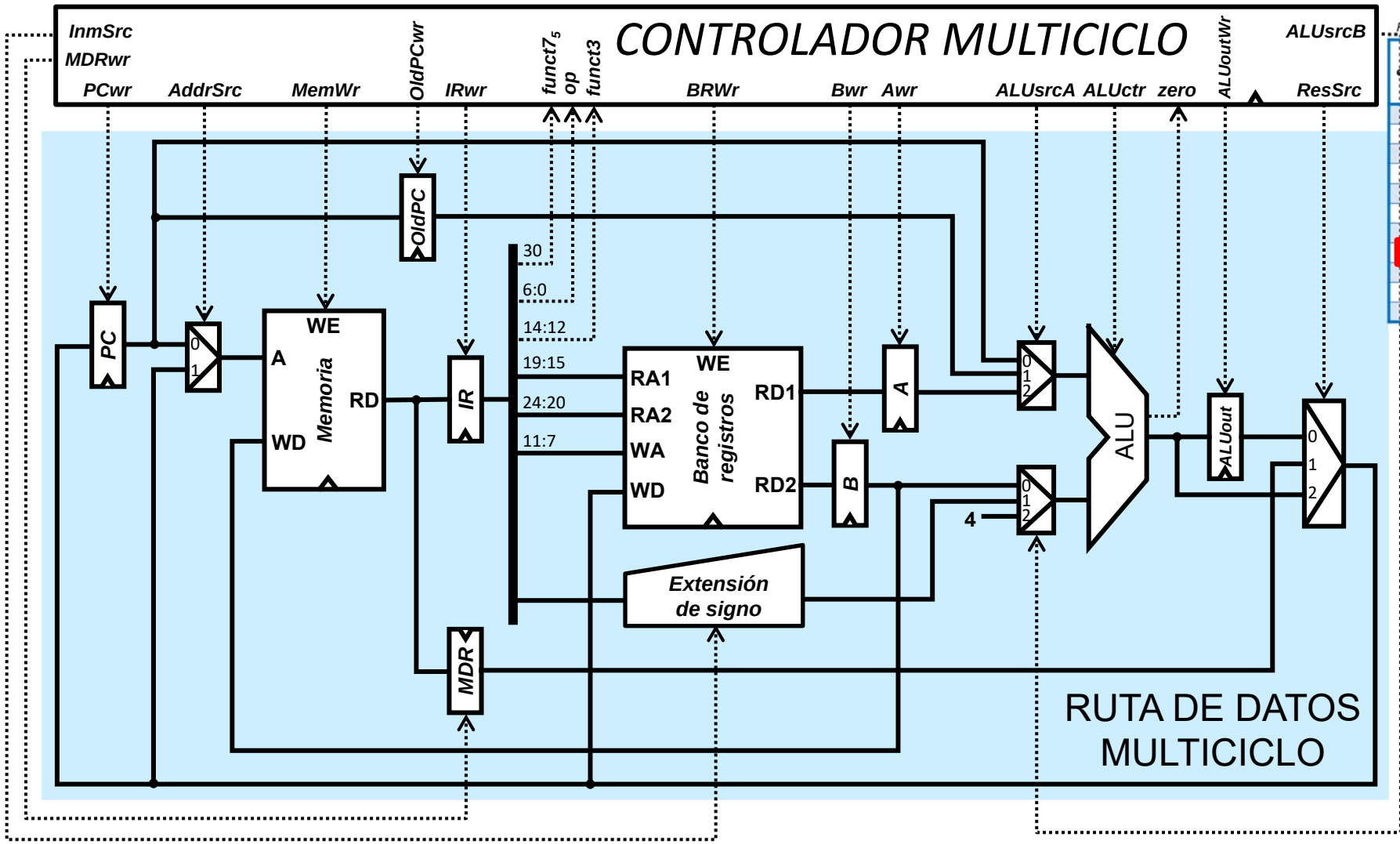



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

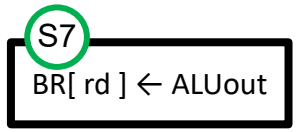



- El contenido de ALUout (que es el resultado de la suma realizada en el ciclo anterior) se lleva al banco de registros a través de la entrada 0 del multiplexor controlado por ResSrc, por lo que esta señal toma el valor 00. Dicho valor se escribe en el registro destino (rd) de la instrucción de suma, determinado por los bits 11:7 de la instrucción. Dado que se escribe en el banco de registros, la señal BRwr toma el valor 1.



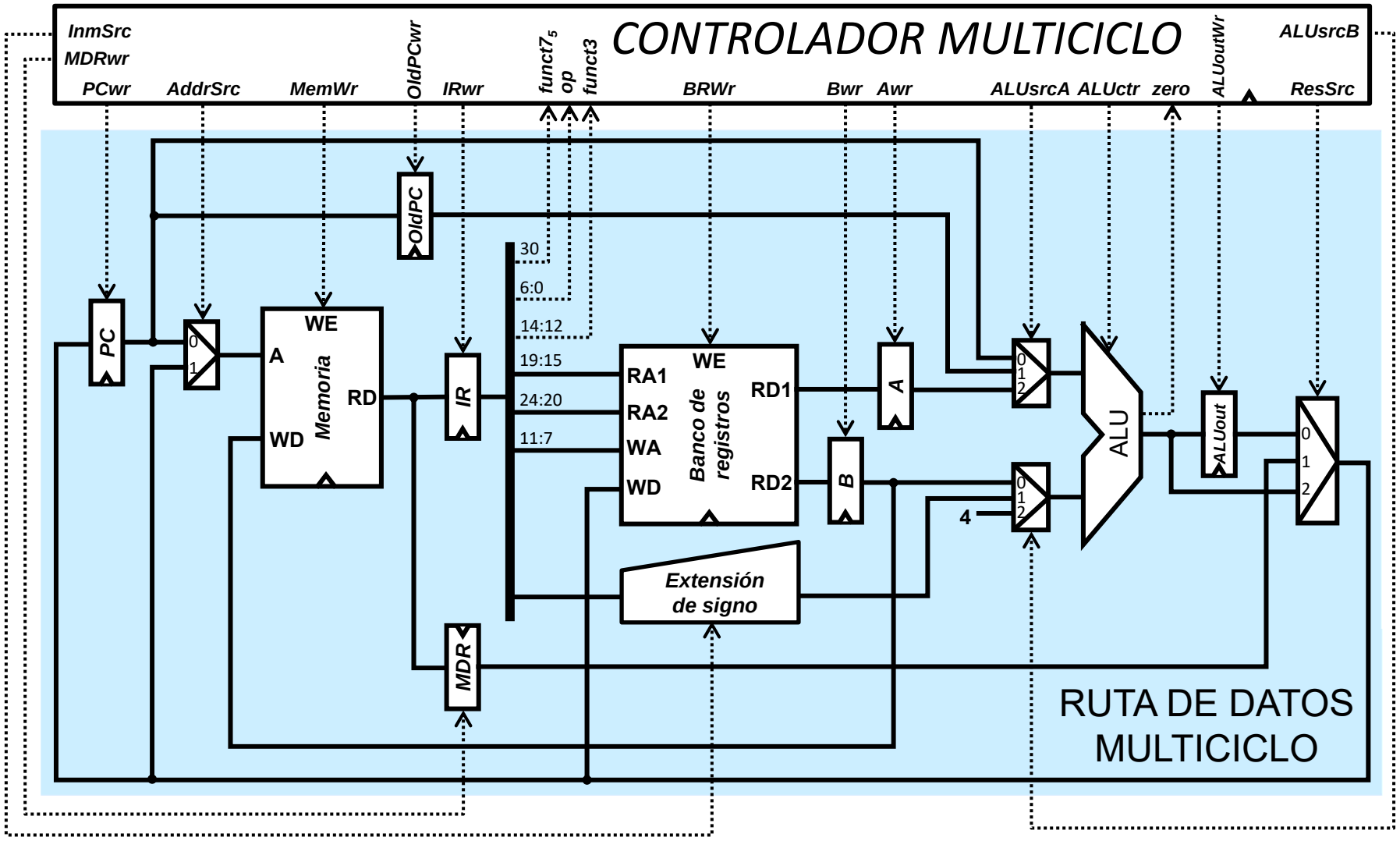

Función de salida

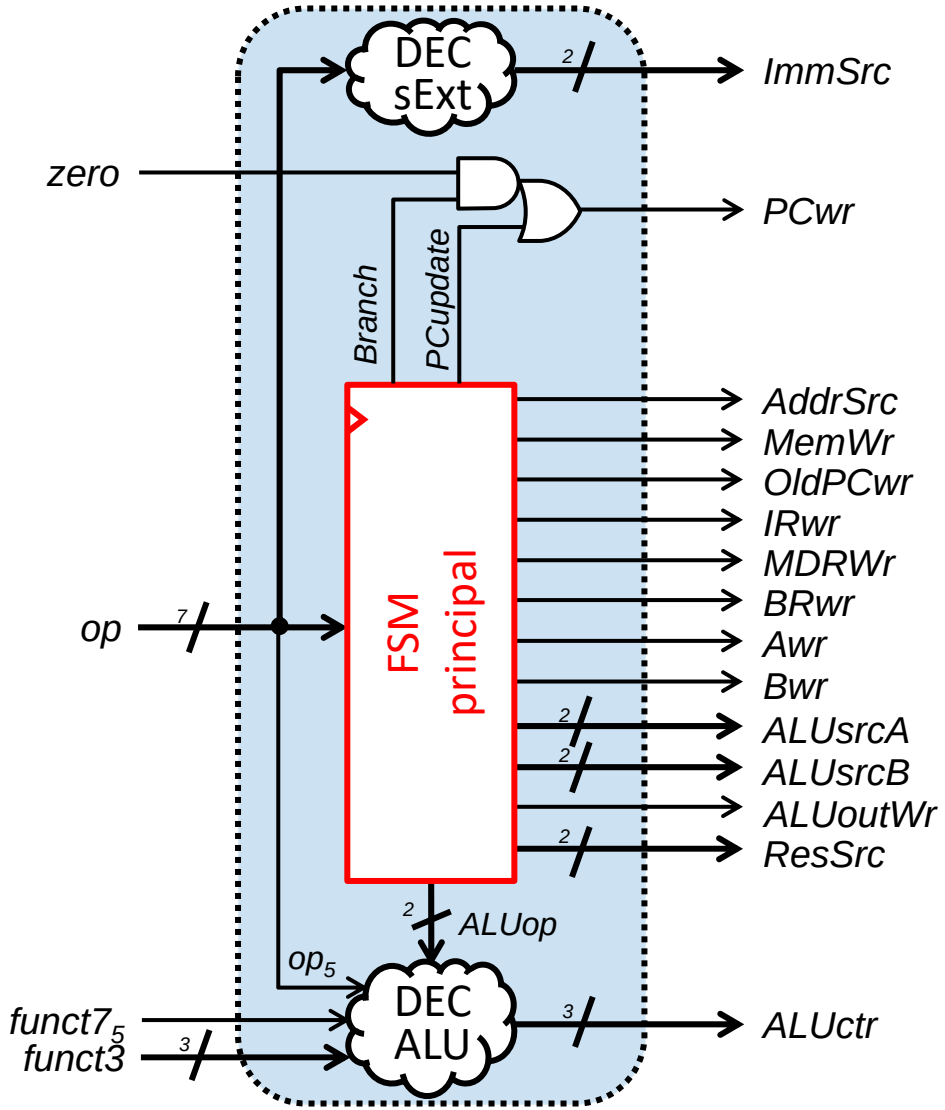
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- OldPCwr, Awr, Bwr y MemWr están a 0 porque en este ciclo no se escribe sobre los registros OldPC, A y B ni Memoria
- La ALU no se utiliza en este ciclo (no se escribe nada sobre su registro de salida, por eso ALUoutWr=0), por lo que los valores de las señales que controlan los muxes que seleccionan los valores de las entradas de la ALU (ALUsrcA y ALUsrcB) son indiferentes. Asimismo, por el mismo motivo, el valor de ALUop también es indiferente

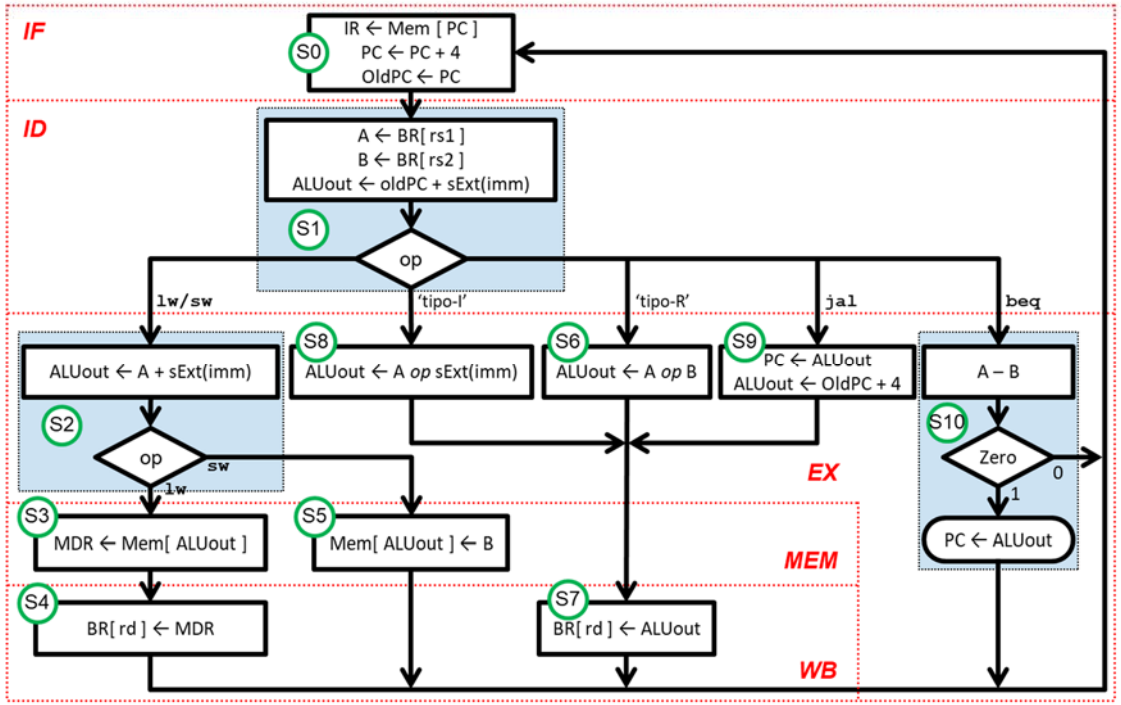
**3)** En el procesador **multiciclo** estudiado en clase, justifica los valores que toman las señales de control para la ejecución de las instrucciones “jal”.



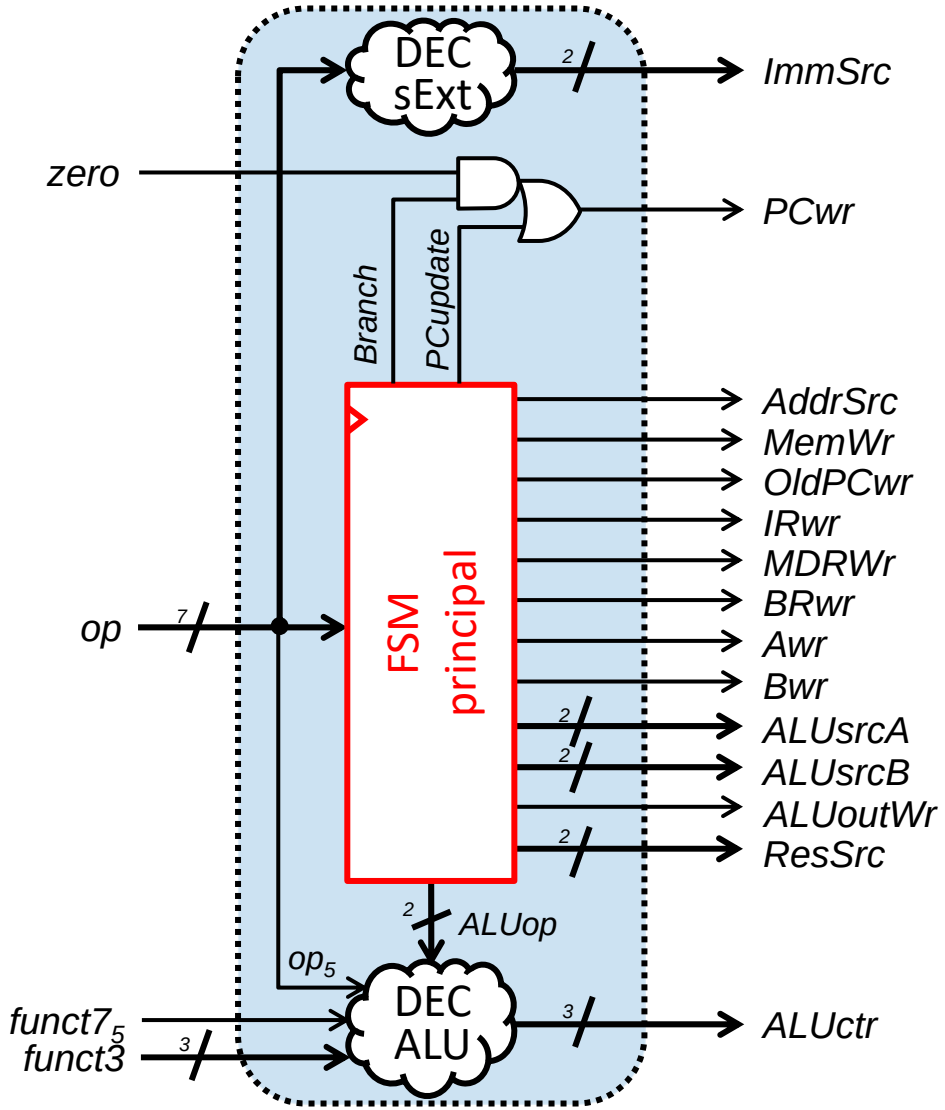



*Función de salida*

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

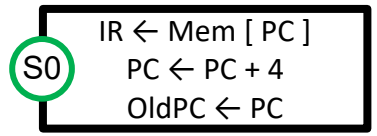



**Las instrucciones jal siguen la secuencia de estados S0-S1-S9-S7**

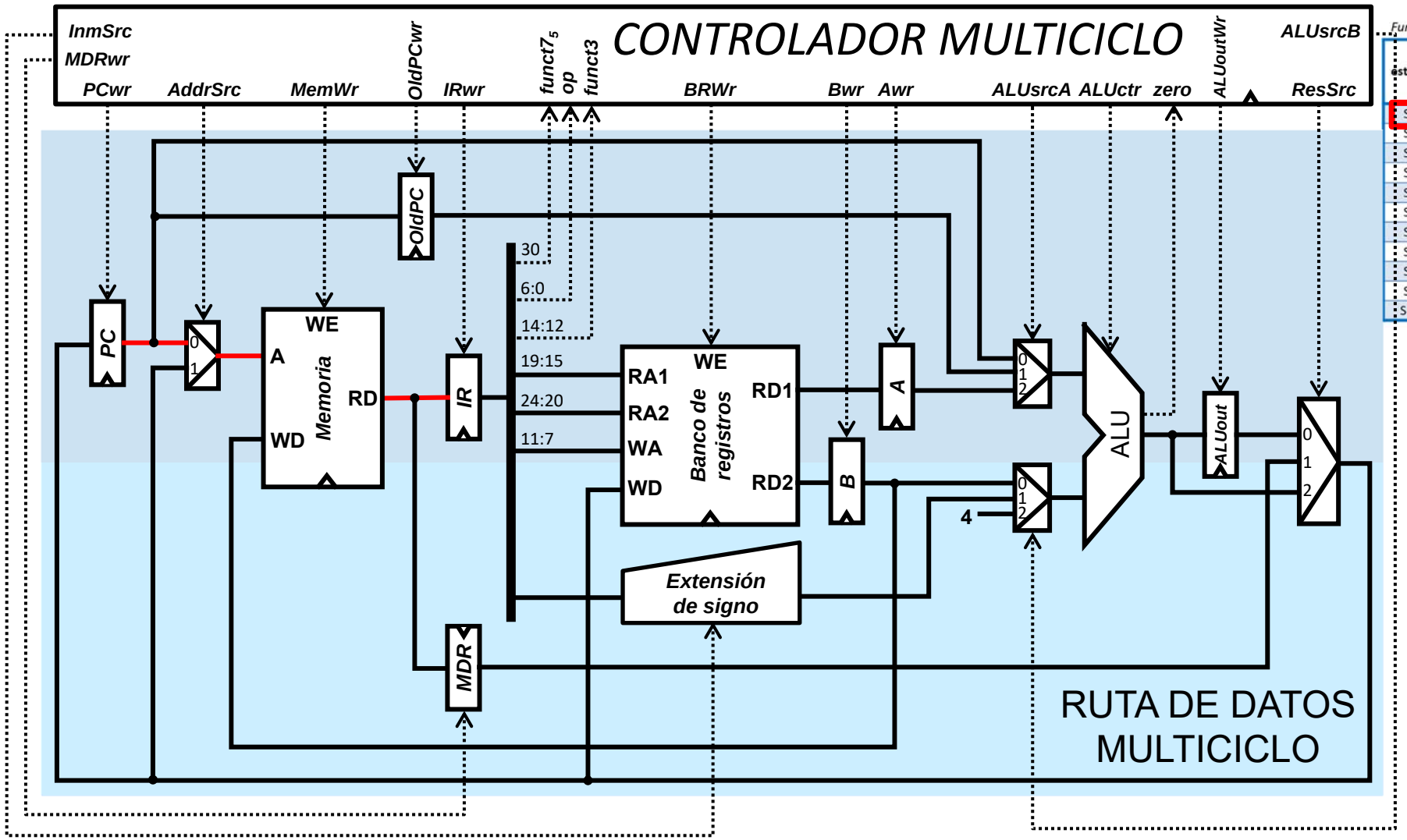



*Función de salida*

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

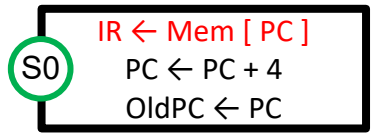



- La señal Branch está a 0 ya que no se trata de una instrucción beq
- La señal PCupdate está a 1 ya que el PC se actualiza con el valor PC+4




Función de salida

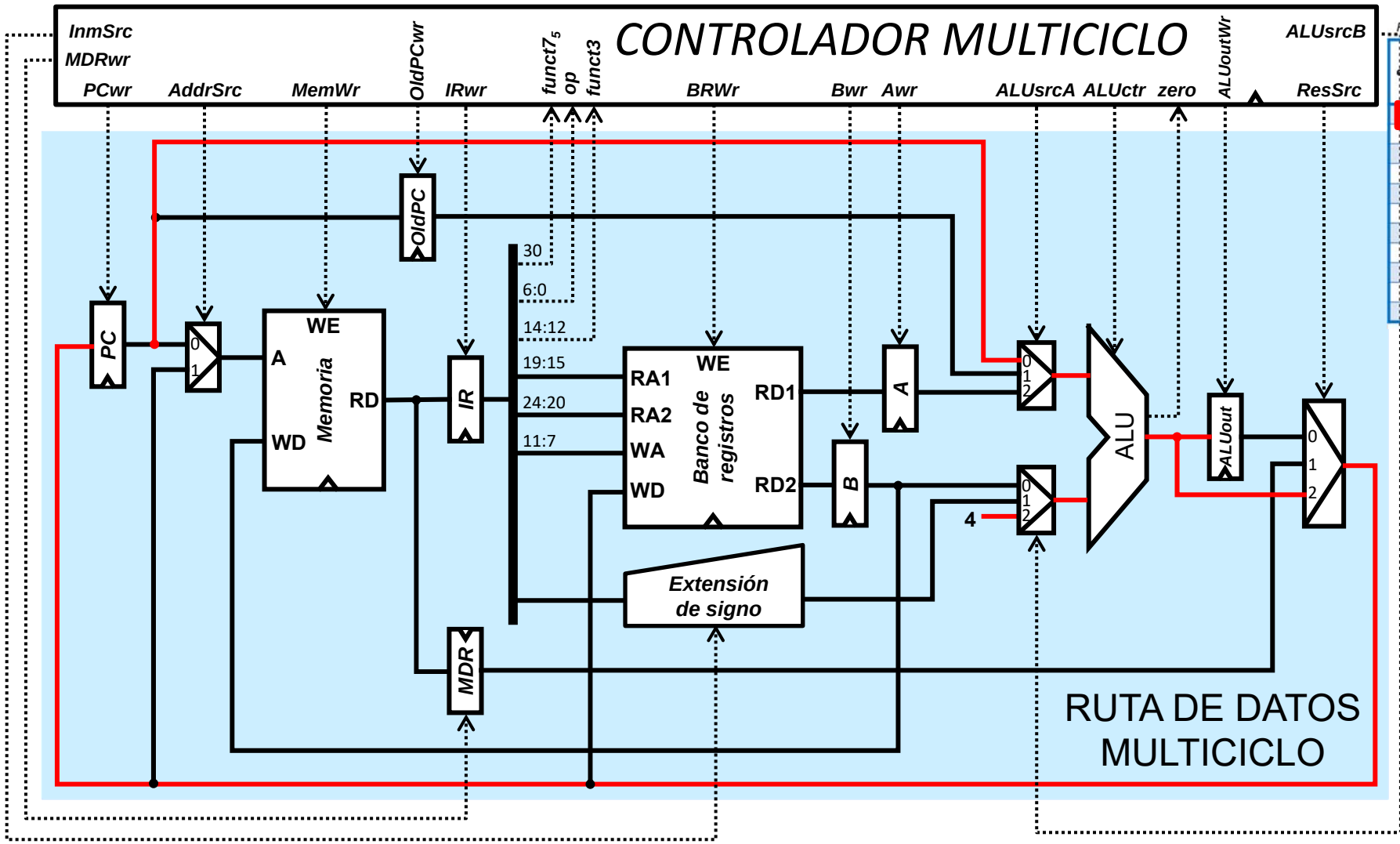
| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- AddrSrc toma el valor 0 ya que se lee la dirección de memoria indicada por el PC para obtener la instrucción

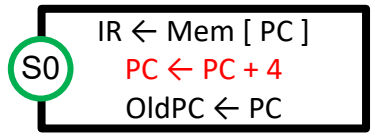
- IRwr toma el valor 1 ya que la instrucción leída de memoria se escribe en el registro IR

- MemWr toma el valor 0 ya que en este ciclo la memoria se lee, pero no se escribe

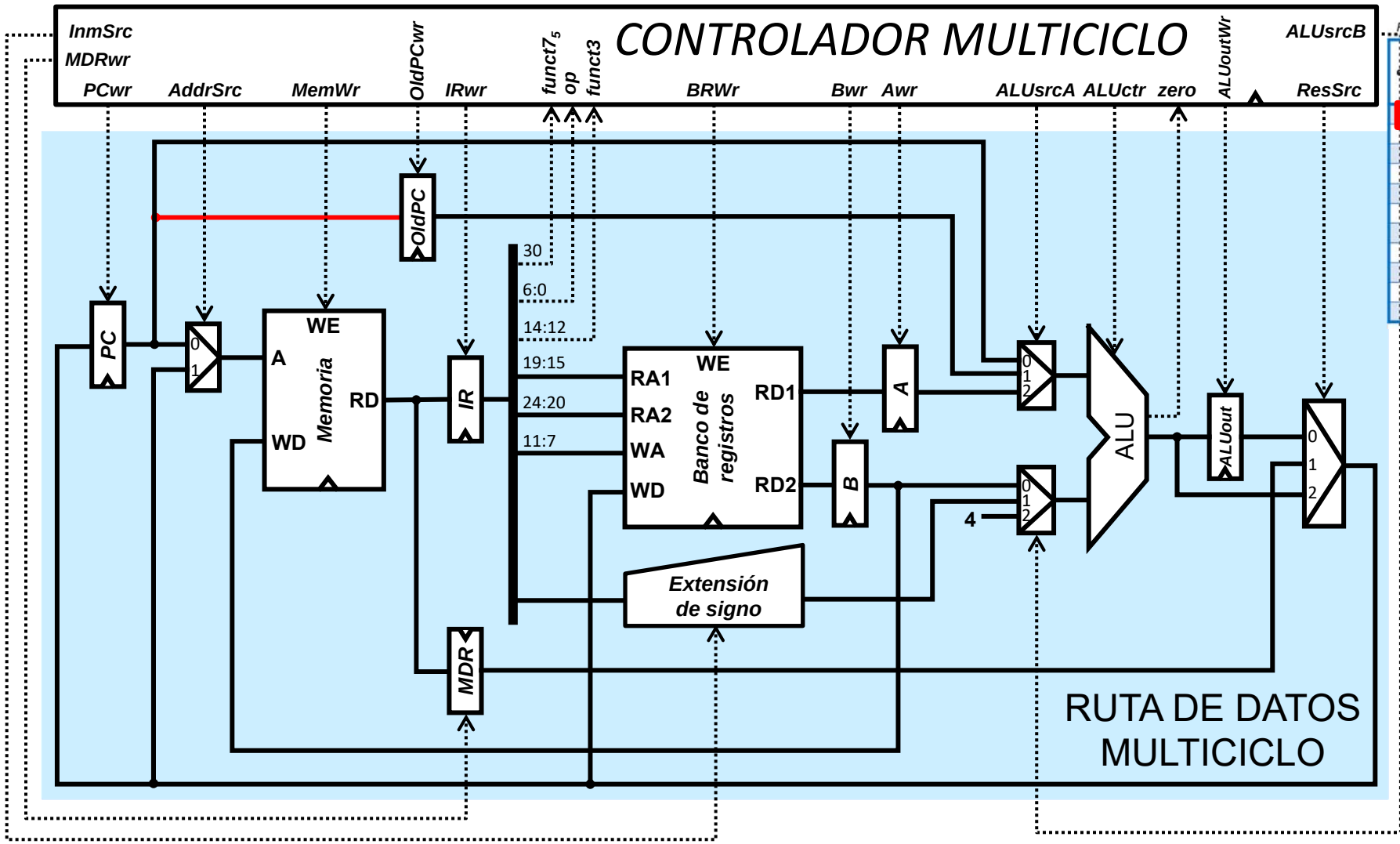



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

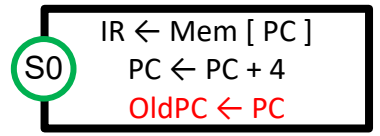



- En la ALU se realiza la operación PC+4 (ALUop=00). El valor del PC llega a la entrada superior de la ALU a través de la entrada 0 del mux correspondiente, por lo que ALUsrcA vale 00. El valor 4 entra a la entrada inferior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcB toma el valor 10. El resultado no se escribe en el registro ALUout (por tanto la señal ALUoutWr vale 0), sino que se lleva directamente al registro PC para actualizar su valor, a través de la entrada 2 del multiplexor controlado por ResSrc (10)

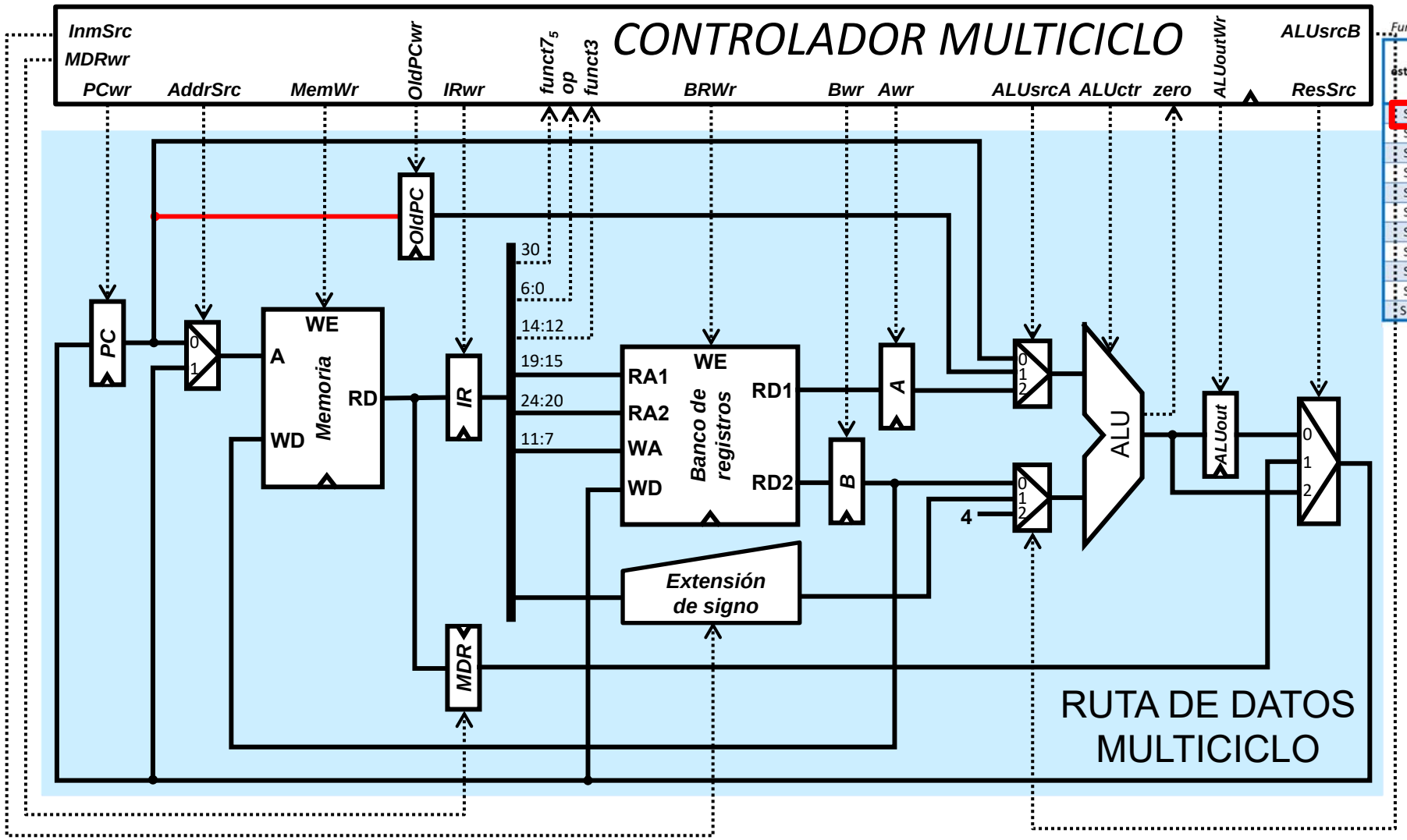



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

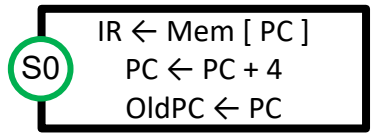



- OldPCwr toma el valor 1 ya que escribimos sobre el registro OldPC el valor del PC correspondiente a instrucción que hemos leído de memoria

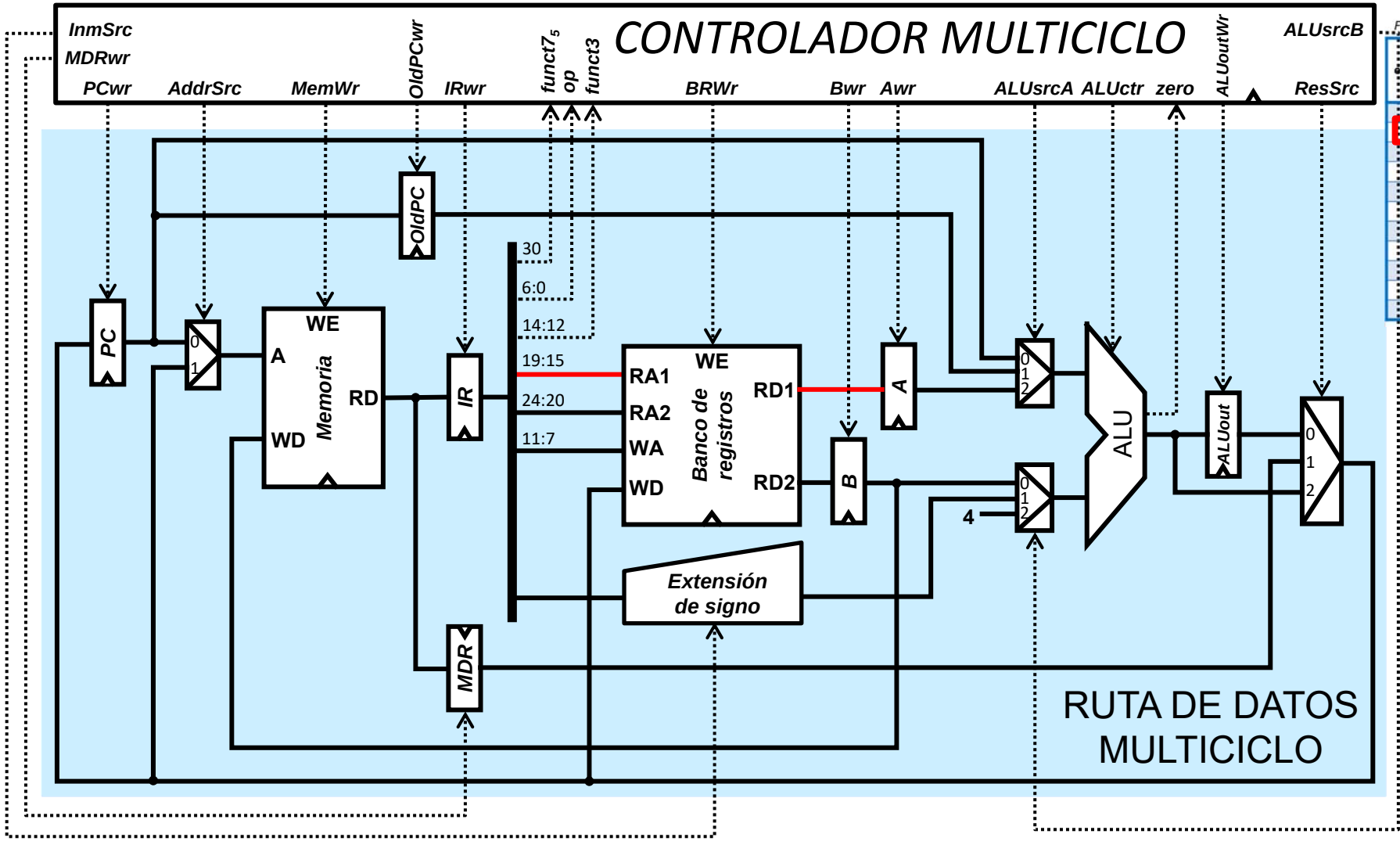



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Las señales MDRwr, BRwr, Awr y Bwr están a 0 porque no se escribe sobre el registro MDR, ni sobre el banco de registros, ni sobre los registros A y B



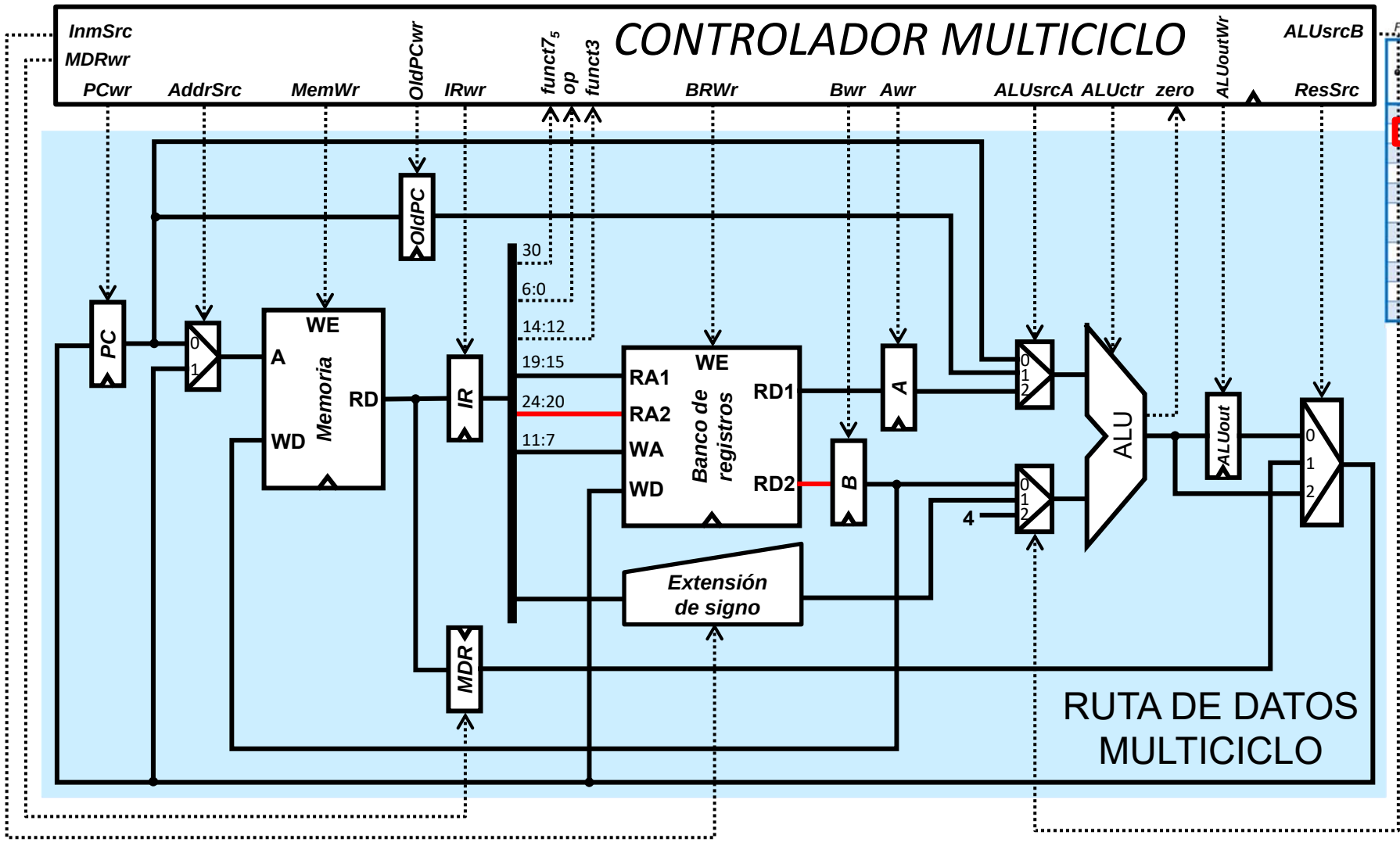

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 1 se escribe en el registro A, por lo que Awr=1



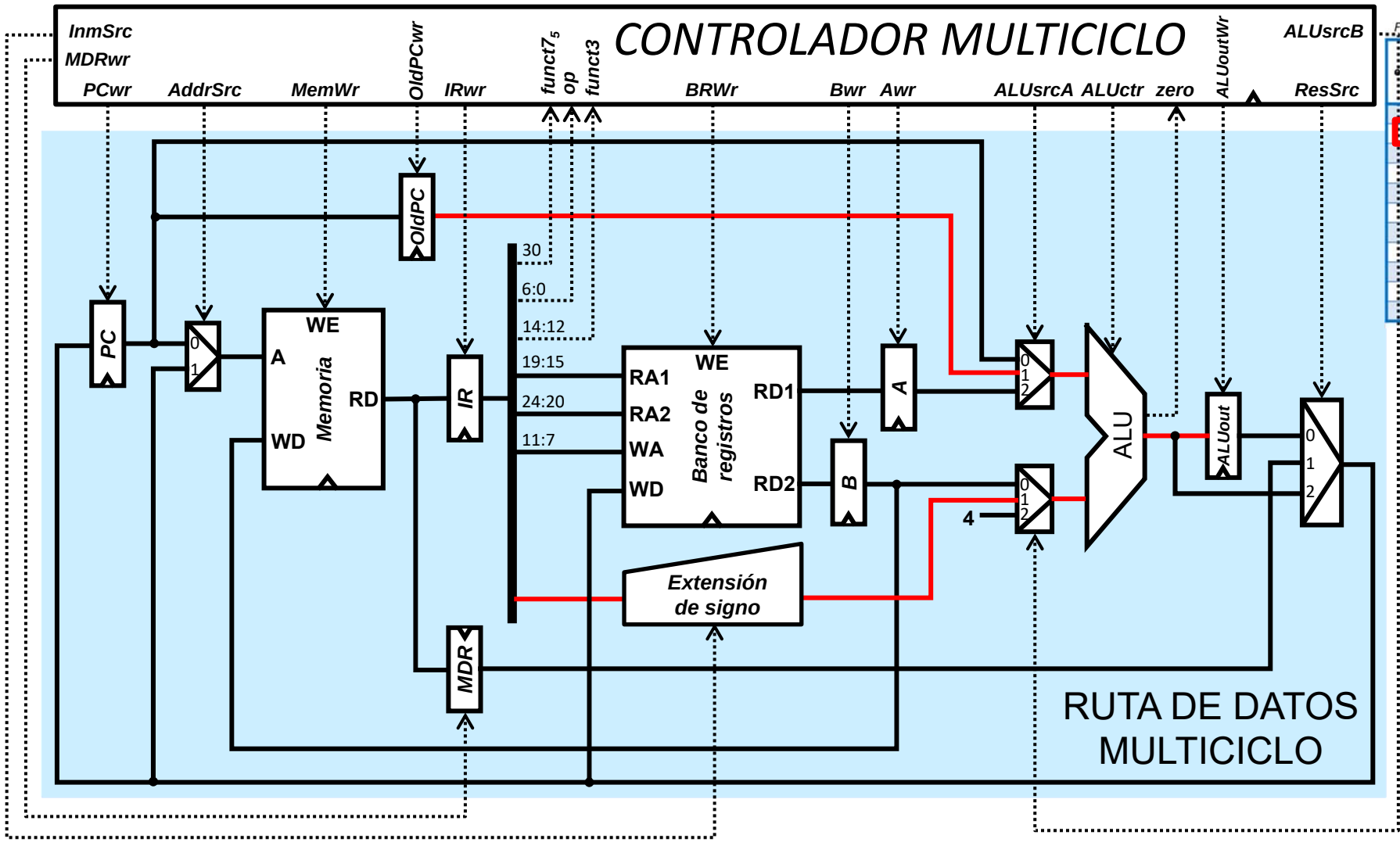

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- El contenido del registro fuente 2 se escribe en el registro B, por lo que Bwr=1



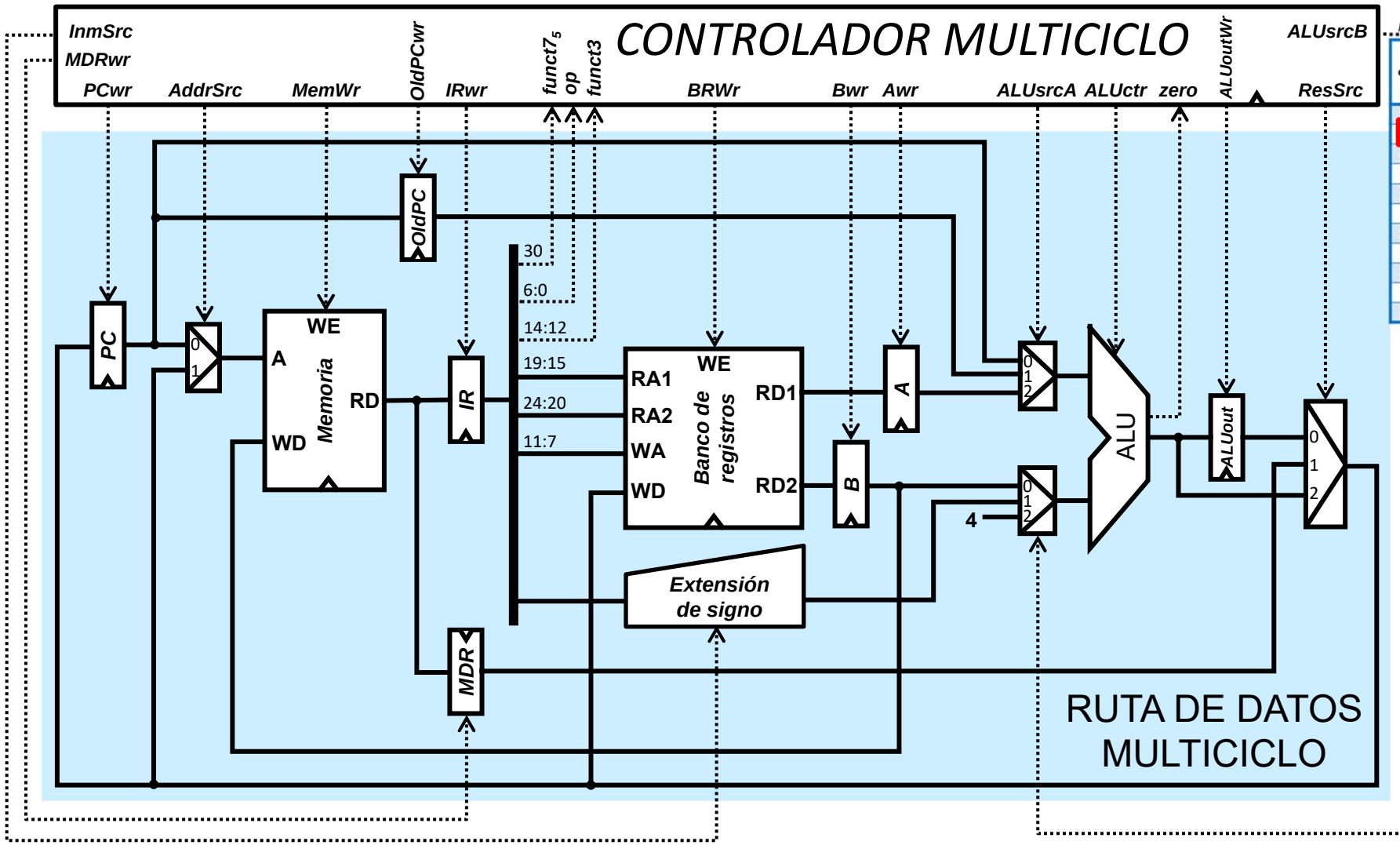

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- En la ALU se realiza la suma (por tanto ALUop=00) de oldPC + sExt(imm). El valor contenido en OldPC llega a la entrada superior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcA=01. El valor del inmediato con el signo extendido llega a la entrada inferior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcB=01. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.



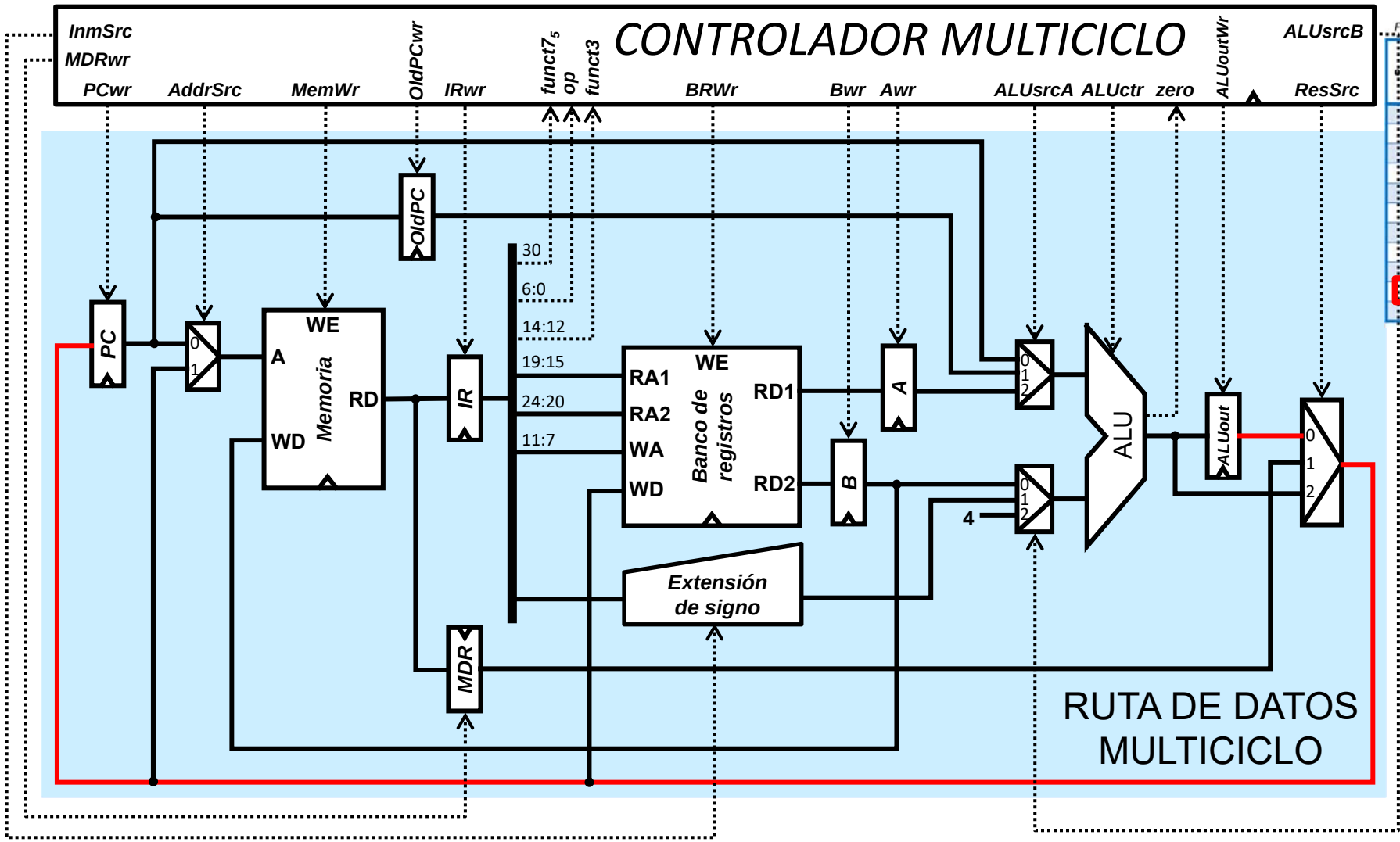

Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

A ← BR[ rs1 ]
B ← BR[ rs2 ]
ALUout ← oldPC + sExt(imm)

S1

- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr y BRwr están a 0 porque en este ciclo no se escribe sobre el registro OldPC, ni sobre el banco de registros
- El valor de ResSrc es indiferente porque no vamos a escribir en este ciclo ni en el banco de registros ni en el PC

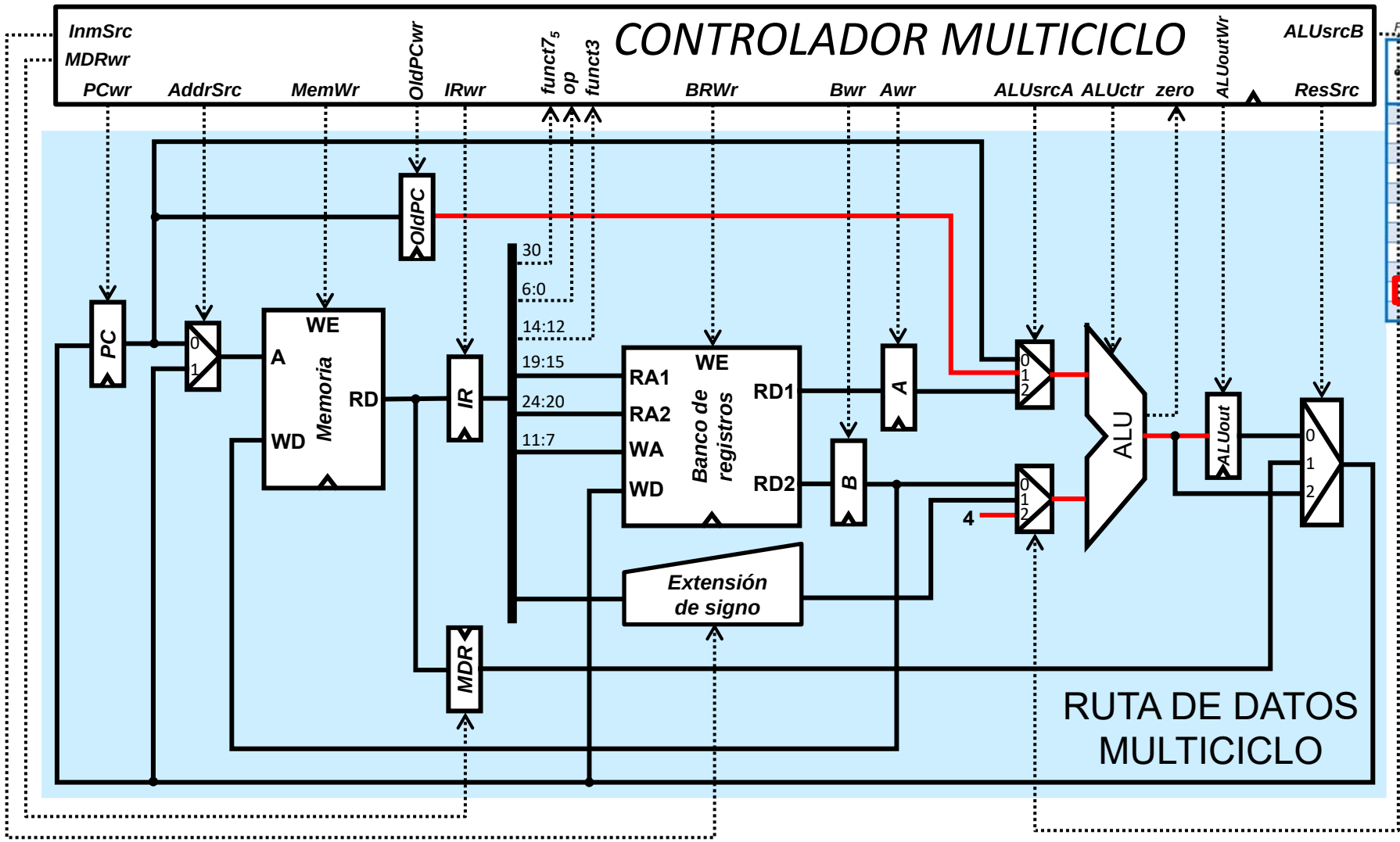



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

S9
PC ← ALUout
ALUout ← OldPC + 4

- El contenido de ALUout (que es el resultado de la suma realizada en el ciclo anterior, es decir la dirección a la que debe saltar la instrucción jal) se lleva al registro PC a través de la entrada 0 del multiplexor controlado por ResSrc, por lo que esta señal toma el valor 00. Asimismo al modificarse el valor del PC, la señala PCupdate toma el valor 1.

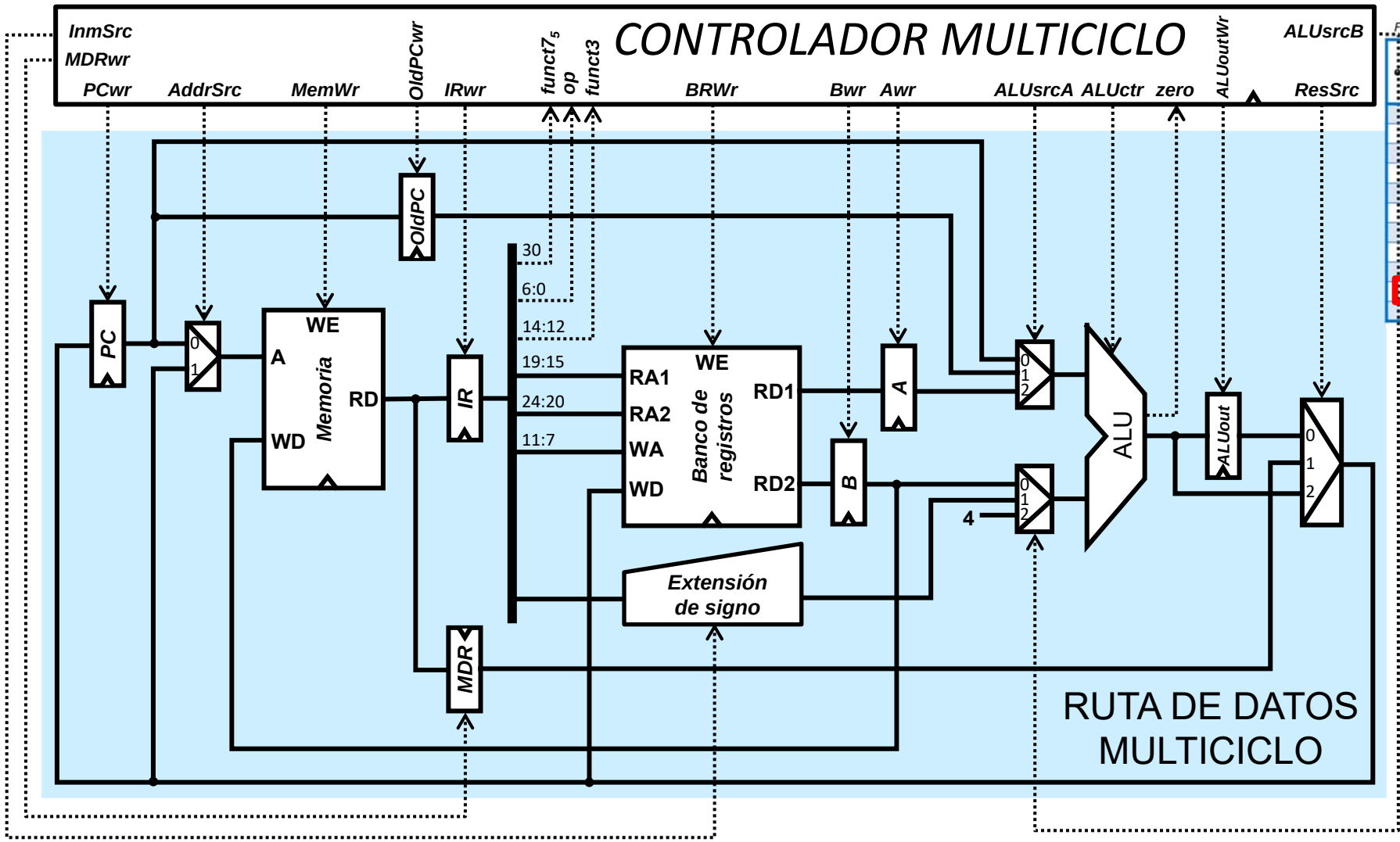



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

S9
PC ← ALUout
ALUout ← OldPC + 4

- En la ALU se realiza la suma (por tanto ALUop=00) de oldPC + 4 (para obtener la dirección de retorno de la instrucción jal). El valor contenido en OldPC llega a la entrada superior de la ALU a través de la entrada 1 del mux correspondiente, por lo que ALUsrcA=01. El valor de la constante 4 llega a la entrada inferior de la ALU a través de la entrada 2 del mux correspondiente, por lo que ALUsrcB=10. El resultado de la suma se guarda en ALUout, por lo que la señal ALUoutWr=1.

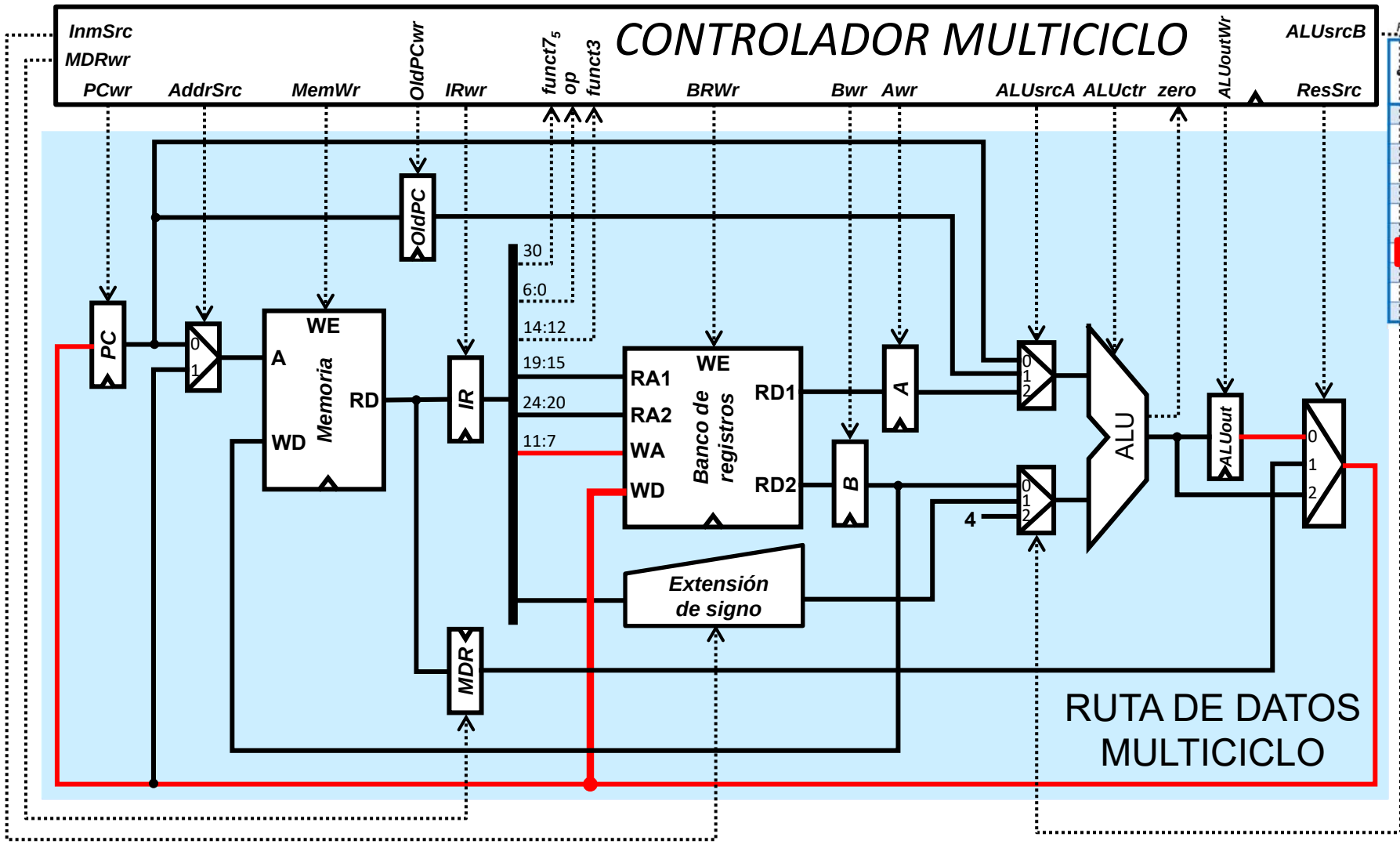



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

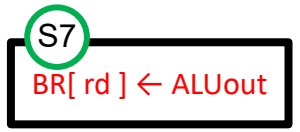
S9
PC ← ALUout
ALUout ← OldPC + 4

- Resto de señales:
- Branch está a 0 ya que no se trata de una instrucción beq
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- MemWr está a 0 ya que no escribimos sobre la memoria
- OldPCwr, BRwr, Awr y Bwr toman el valor 0 dado que no se escribe sobre los registros OldPc, A, B ni sobre el banco de registros

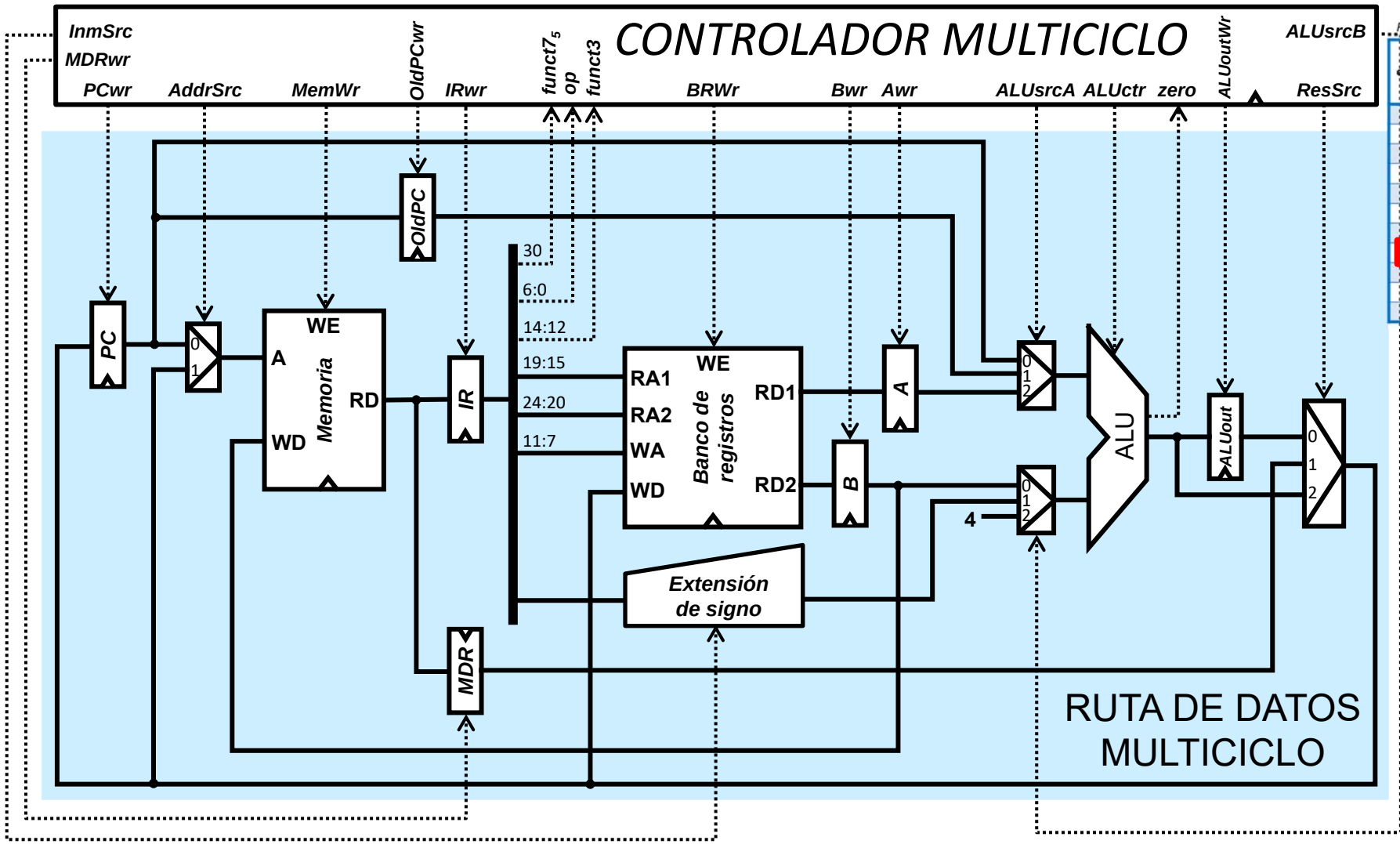



Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |

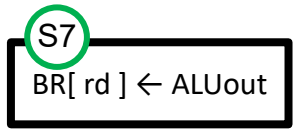



- El contenido de ALUout (que es el resultado de la suma realizada en el ciclo anterior, es decir la dirección de retorno de la instrucción jal) se lleva al banco de registros a través de la entrada 0 del multiplexor controlado por ResSrc, por lo que esta señal toma el valor 00. Dicho valor se escribe en el registro destino (rd) de la instrucción jal, determinado por los bits 11:7 de la instrucción. Dado que se escribe en el banco de registros, la señal BRwr toma el valor 1.




Función de salida

| estado | Branch | PCupdate | AdrSrc | MemWr | OldPCwr | IRwr | MDRwr | BRwr | Awr | Bwr | ALUsrcA | ALUsrcB | ALUop | ALUoutWr | ResSrc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 10 | 00 | 0 | 10 |
| S1 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 01 | 01 | 00 | 1 | – |
| S2 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 00 | 1 | – |
| S3 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S4 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 01 |
| S5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | 00 |
| S6 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 10 | 1 | – |
| S7 | 0 | 0 | – | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | – | – | 0 | 00 |
| S8 | 0 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 01 | 10 | 1 | – |
| S9 | 0 | 1 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 00 | 1 | 00 |
| S10 | 1 | 0 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 01 | 0 | 00 |




- Resto de señales:
- Branch y PCupdate están a 0 ya que no se trata de una instrucción beq ni se actualiza el PC en este ciclo
- El valor de AddrSrc es indiferente ya que lo que se lea de memoria no se escribe ni en IR ni en MDR (IRwr y MDRwr=0)
- OldPCwr, Awr, Bwr y MemWr están a 0 porque en este ciclo no se escribe sobre los registros OldPC, A y B ni Memoria
- La ALU no se utiliza en este ciclo (no se escribe nada sobre su registro de salida, por eso ALUoutWr=0), por lo que los valores de las señales que controlan los muxes que seleccionan los valores de las entradas de la ALU (ALUsrcA y ALUsrcB) son indiferentes. Asimismo, por el mismo motivo, el valor de ALUop también es indiferente

**4)** En el procesador RISC-V **multiciclo** estudiado en clase, un determinado programa se ejecuta a razón de un promedio de 4.25 ciclos por instrucción. El programa está formado por 10000 instrucciones, de las cuales un 30% son instrucciones de load, un 15% son instrucciones de store, un 20% instrucciones aritmético-lógicas y el resto son instrucciones jal y beq. Determina cuántas instrucciones de tipo jal y cuantas instrucciones de tipo beq integran dicho programa.

| Tipo instr | fracción inst | nºciclos/inst |
|---|---|---|
| Load | 30% | 5 |
| Store | 15% | 4 |
| Aritmético-lógicas | 20% | 4 |
| jal | fjal | 4 |
| beq | fbeq | 3 |

CPI = 4.25 = 0.3 x 5 + 0.15 x 4 + 0.2 x 4 +fjal x 4 + fbeq x 3

4.25 – 1.5 – 0.6 -0.8 = 4fjal + 3 fbeq → 1.35 = 4fjal + 3 fbeq

Por otro lado las instrucciones de jal y beq representan conjuntamente el 35% de las instrucciones del programa, por lo que fjal + fbeq = 0.35

1.35 = 4 (0.35-fbeq) + 3 fbeq → fbeq = 0.05 → 5% de instrucciones de salto y 30% de instrucciones de jal

**3000 instrucciones de jal y 500 instrucciones de beq**

**5)** En cierto procesador **multiciclo** se ejecuta un programa que consta de 140 instrucciones, de las cuales 70 tardan en ejecutarse en dicho procesador cuatro ciclos, 35 tardan cinco ciclos, 20 tardan tres ciclos y las 15 restantes tardan siete ciclos. Calcule el CPI promedio para dicho programa. Si el procesador funciona a una frecuencia de 2.0 GHz, determine el tiempo de ejecución del programa.

| Tipo instr | nºinst | nºciclos/inst |
|---|---|---|
| A | 70 | 4 |
| B | 35 | 5 |
| C | 20 | 3 |
| D | 15 | 7 |

Nº total inst (N) = 140

CPI = (70*4+35*5+20*3+15*7) / 140 = **4.4285 ciclos**

Tiempo_ejecución = N*CPI/frecuencia = $140*4.4285/2*10^9$ = **310ns**

**6)** Se dispone de los siguientes datos de dos procesadores **multiciclo** y de su rendimiento en la ejecución de una determinada tarea:

- PowerPC que funciona a una frecuencia de 1.8 GHz y obtiene 700 MIPS.
- Pentium 4 que funciona a 1.6 GHz y 850 MIPS.

Calcule el CPI de cada procesador.

| Procesador | Frecuencia | MIPS |
|---|---|---|
| PowerPC | 1.8GHz | 700 |
| Pentium4 | 1.6GHz | 850 |

$MIPS = (N/T)*10^{-6} = [N/(N*CPI*tc)] *10^{-6} = [(1/(CPI*tc)] *10^{-6} = (f/CPI) *10^{-6}$

$\rightarrow CPI = (f/MIPS) *10^{-6}$

$CPI_{PowerPC} = 1.8*10^9$ [ciclos/s] / $700*10^6$ [instr/s] = **2.57 [ciclos/instr]**

$CPI_{Pentium4} = 1.6*10^9$ [ciclos/s] / $850*10^6$ [instr/s] = **1.88 [ciclos/instr]**

**7)** Considere los dos procesadores del ejercicio anterior. En la ejecución de un determinado programa los procesadores obtienen un CPI de 5.5 (PowerPC) y 7 (Pentium 4). El compilador genera un código máquina para dicho programa que tiene 9 millones de instrucciones (PowerPC) y 7.2 millones de instrucciones (Pentium) ¿Qué computador ejecutará más rápidamente la tarea?

$Tiempo_{PowerPC}$ = $9*10^6$ [instr] * 5.5 [ciclos/instr] / $1.8*10^9$ [ciclos/s] = 0.0275s = **27.5ms**

$Tiempo_{Pentium4}$ = $7.2*10^6$ [instr] * 7 [ciclos/instr] / $1.6*10^9$ [ciclos/s] = 0.0315s = **31.5ms**

**8)** Un mismo programa se ejecuta en dos computadores **multiciclo** (A y B) que tienen frecuencias de reloj de 1 GHz y 1.5 GHz, respectivamente. Para ejecutar el programa en A es necesario ejecutar un cierto número de instrucciones repartidas de la siguiente manera:

| | Aritmética | Load | Store | beq |
|---|---|---|---|---|
| Frecuencia | 50% | 25% | 10% | 15% |
| Ciclos | 4 | 5 | 4 | 3 |

a) Calcula el CPI del programa en el computador A.
b) En el computador B el número de instrucciones ejecutadas es el 60% de las ejecutadas en A y el tiempo de ejecución es la mitad que en A. ¿Cuál es el CPI obtenido en la ejecución del programa en el computador B?

**a)** **CPI(A)** = 4x0,5 + 5x0,25 + 4x0,10 + 3x0,15 = **4,1**

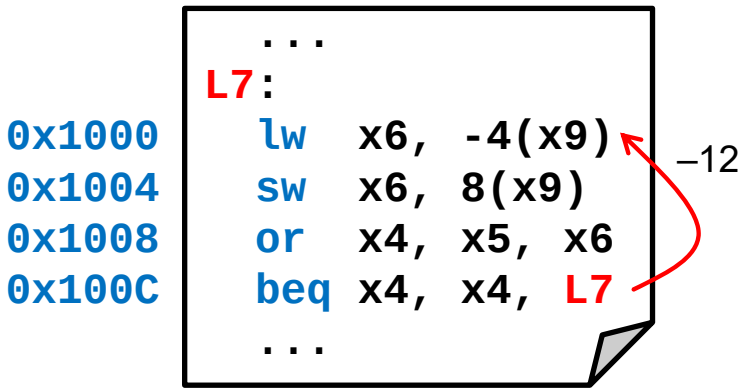
b) Datos: NI(B) = 0,6xNI(A); T(B) = 0,5xT(A);

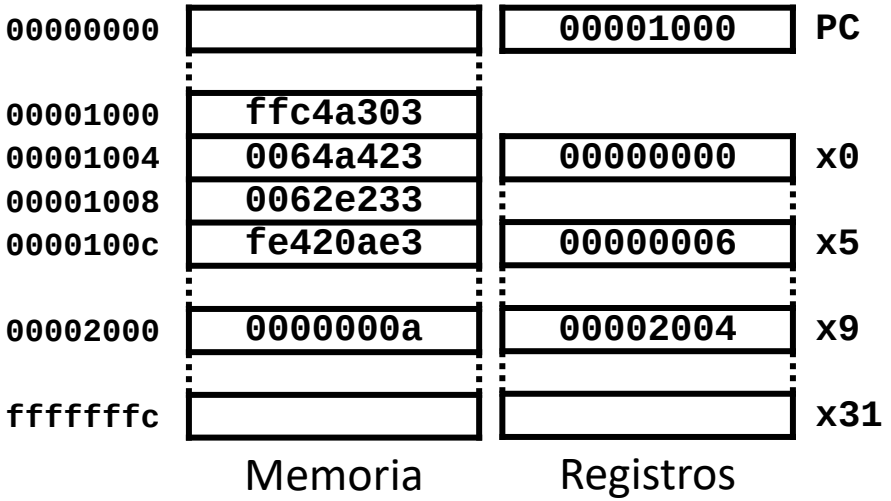
Tiempo en A: T(A) = NI(A) x CPI(A) / f(A)
Tiempo en B: 0,5xT(A) = 0,6xNI(A) x CPI(B) / f(B)

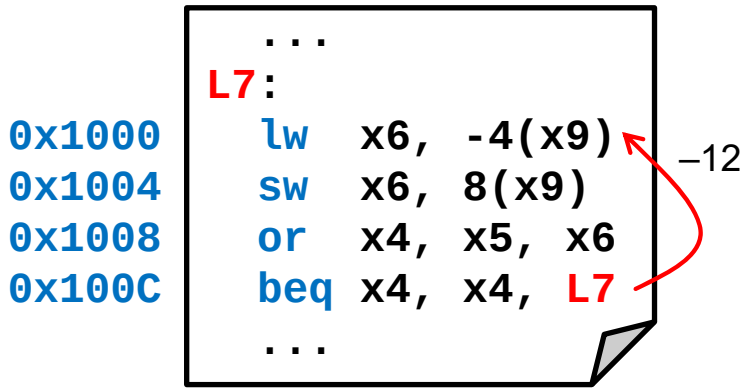
Dividiendo Tiempo en B / Tiempo en A:

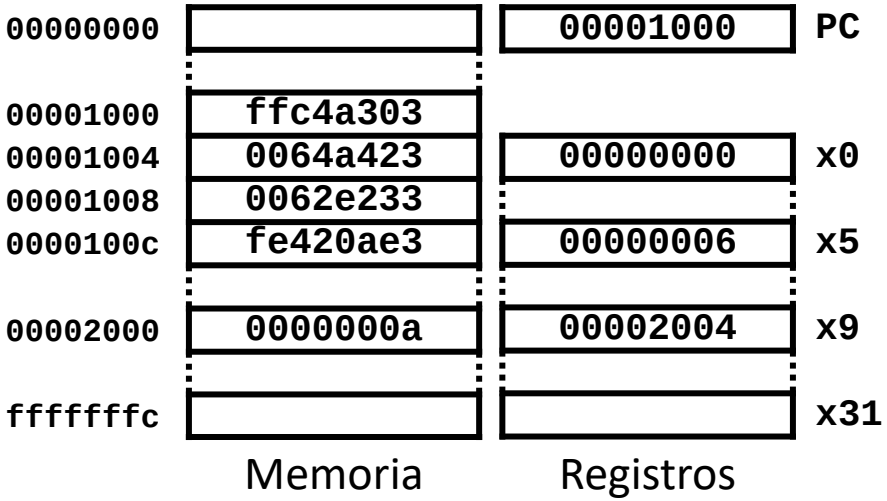
0,5 = [0,6xCPI(B) / f(B)] / [CPI(A) / f(A)] = 0,6 x CPI(B) x f(A) / CPI(A) x f(B)
**CPI(B)** = 0,5 x CPI(A) x f(B) / 0,6 x f(A) = 0,5x4,1x1,5x$10^9$ / 0,6x1x$10^9$ = 3,075 / 0,6 = **5,125**

**9)** Supongamos que sobre el procesador **multiciclo** estudiado en clase se quiere ejecutar el programa que se muestra a continuación, siendo el contenido inicial de los registros y la memoria el mostrado en la figura. Representa los diagramas de ejecución para los registros y la memoria, así como para las señales de estado y de control, de modo que se visualicen sus valores en cada ciclo de reloj correspondiente a la ejecución del programa.

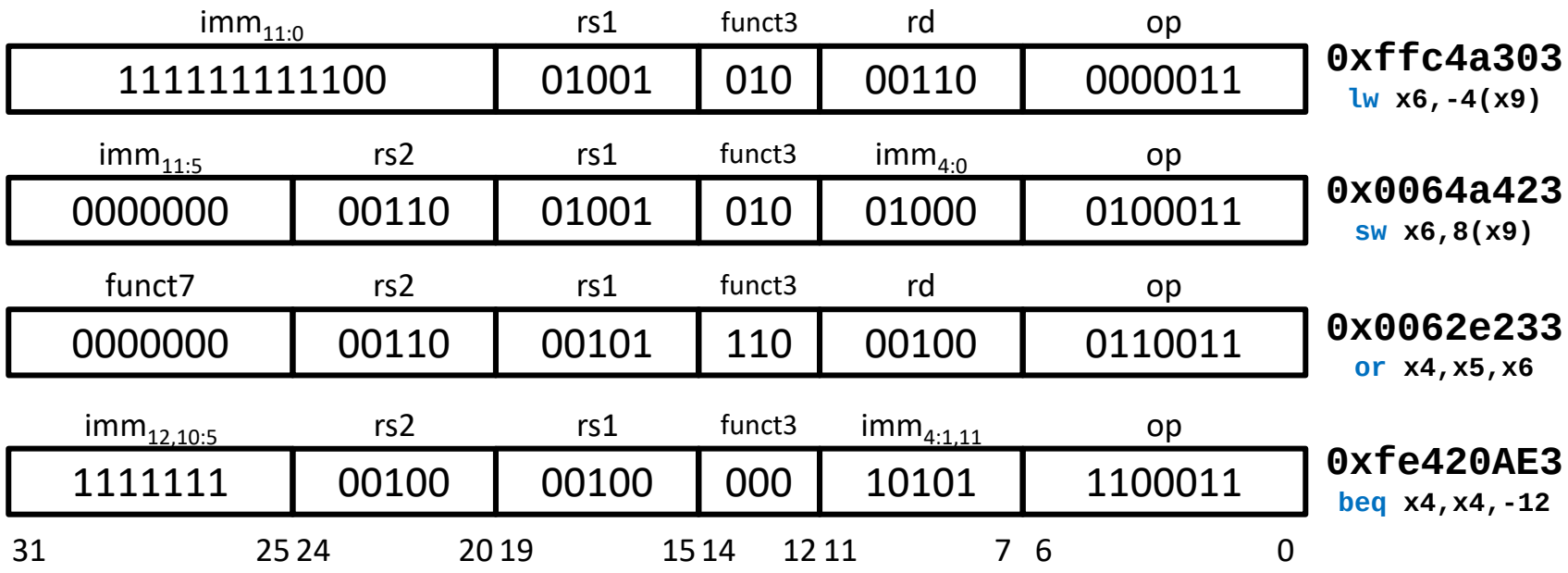





La representación en código máquina de las instrucciones que conforman el programa es la siguiente:



```
          ...
L7:
0x1000    lw  x6, -4(x9)
0x1004    sw  x6, 8(x9)
0x1008    or  x4, x5, x6
0x100C    beq x4, x4, L7
          ...
```

–12

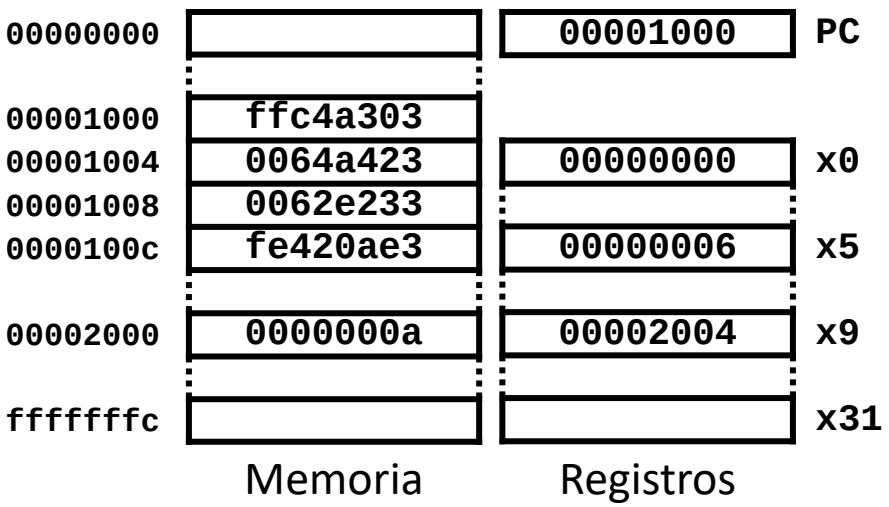
| | Memoria | | Registros | |
|---|---|---|---|---|
| 00000000 | | PC | 00001000 | PC |
| 00001000 | ffc4a303 | | | |
| 00001004 | 0064a423 | | 00000000 | x0 |
| 00001008 | 0062e233 | | | |
| 0000100c | fe420ae3 | | 00000006 | x5 |
| 00002000 | 0000000a | | 00002004 | x9 |
| fffffffc | | | | x31 |

# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | | | | | | | | |
| PC | 00001000 | | | | | | | | |
| OldPC | ???????? | | | | | | | | |
| IR | ???????? | | | | | | | | |
| MDR | ???????? | | | | | | | | |
| A | ???????? | | | | | | | | |
| B | ???????? | | | | | | | | |
| ALUout | ???????? | | | | | | | | |
| x4 | ???????? | | | | | | | | |
| x5 | 00000006 | | | | | | | | |
| x6 | ???????? | | | | | | | | |
| x9 | 00002004 | | | | | | | | |
| MEM[0x2000] | 0000000a | | | | | | | | |
| MEM[0x200c] | ???????? | | | | | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

−12

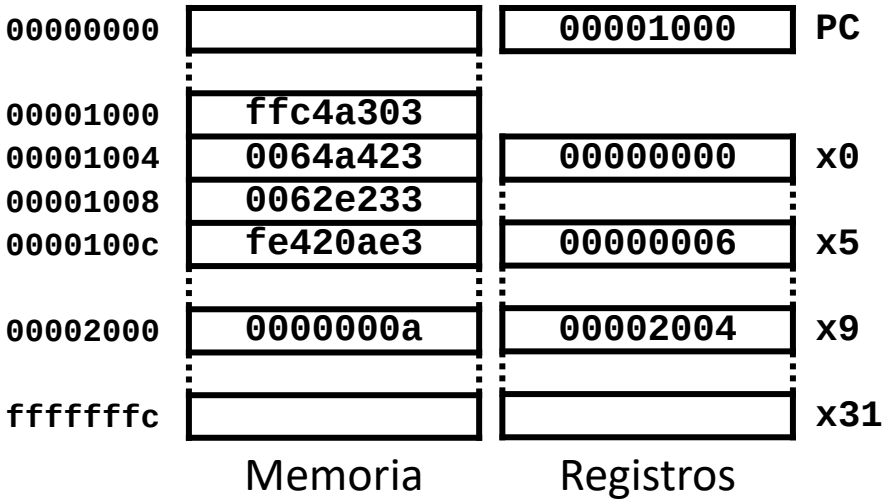



# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | | | | | | | |
| PC | 00001000 | 00001004 | | | | | | | |
| OldPC | ???????? | 00001000 | | | | | | | |
| IR | ???????? | ffc4a303 | | | | | | | |
| MDR | ???????? | = | | | | | | | |
| A | ???????? | = | | | | | | | |
| B | ???????? | = | | | | | | | |
| ALUout | ???????? | = | | | | | | | |
| x4 | ???????? | = | | | | | | | |
| x5 | 00000006 | = | | | | | | | |
| x6 | ???????? | = | | | | | | | |
| x9 | 00002004 | = | | | | | | | |
| MEM[0x2000] | 0000000a | = | | | | | | | |
| MEM[0x200c] | ???????? | = | | | | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

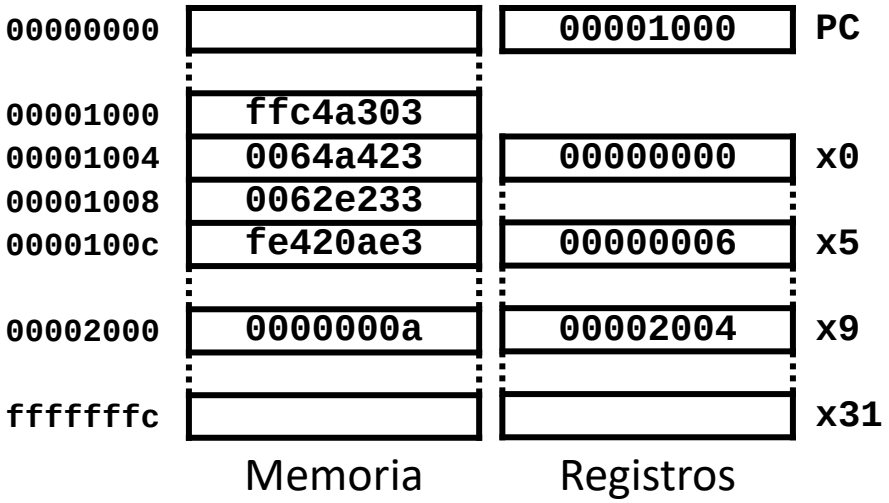
| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | | | | | | |
| PC | 00001000 | 00001004 | = | | | | | | |
| OldPC | ???????? | 00001000 | = | | | | | | |
| IR | ???????? | ffc4a303 | = | | | | | | |
| MDR | ???????? | ???????? | = | | | | | | |
| A | ???????? | ???????? | 00002004 | | | | | | |
| B | ???????? | ???????? | basura¹ | | | | | | |
| ALUout | ???????? | ???????? | basura² | | | | | | |
| x4 | ???????? | ???????? | = | | | | | | |
| x5 | 00000006 | 00000006 | = | | | | | | |
| x6 | ???????? | ???????? | = | | | | | | |
| x9 | 00002004 | 00002004 | = | | | | | | |
| MEM[0x2000] | 0000000a | 0000000a | = | | | | | | |
| MEM[0x200c] | ???????? | ???????? | = | | | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

basura¹ = BR[$IR_{24..20}$] = x28 = 0x00000000

basura² = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

```
        ...
L7:
0x1000  lw  x6, -4(x9)
0x1004  sw  x6, 8(x9)
0x1008  or  x4, x5, x6
0x100C  beq x4, x4, L7
        ...
```

−12

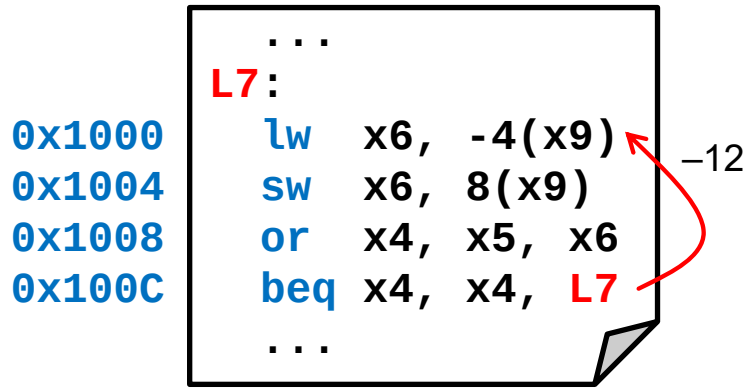


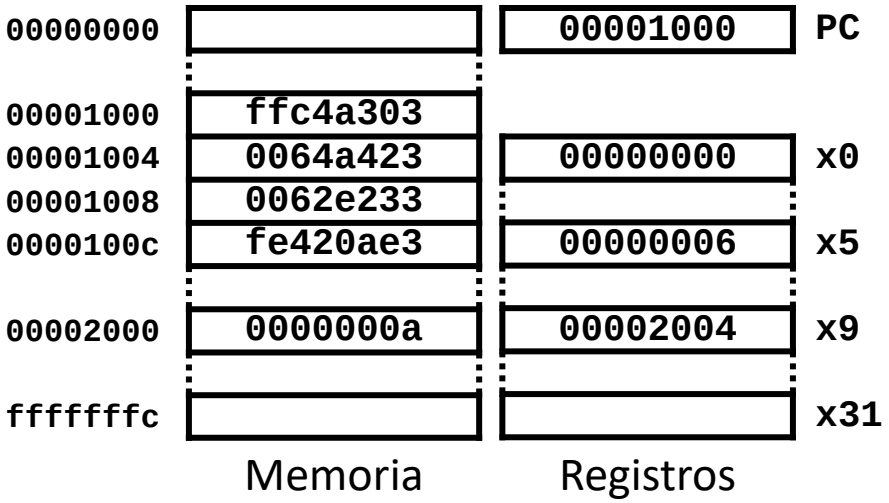

# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | | | | | |
| PC | 00001000 | 00001004 | 00001004 | = | | | | | |
| OldPC | ???????? | 00001000 | 00001000 | = | | | | | |
| IR | ???????? | ffc4a303 | ffc4a303 | = | | | | | |
| MDR | ???????? | ???????? | ???????? | = | | | | | |
| A | ???????? | ???????? | 00002004 | = | | | | | |
| B | ???????? | ???????? | basura¹ | = | | | | | |
| ALUout | ???????? | ???????? | basura² | 00002000 | | | | | |
| x4 | ???????? | ???????? | ???????? | = | | | | | |
| x5 | 00000006 | 00000006 | 00000006 | = | | | | | |
| x6 | ???????? | ???????? | ???????? | = | | | | | |
| x9 | 00002004 | 00002004 | 00002004 | = | | | | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | = | | | | | |
| MEM[0x200c] | ???????? | ???????? | ???????? | = | | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

basura¹ = BR[$IR_{24..20}$] = x28 = 0x00000000

basura² = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc




# Diagrama de ejecución: registros y memoria (i)

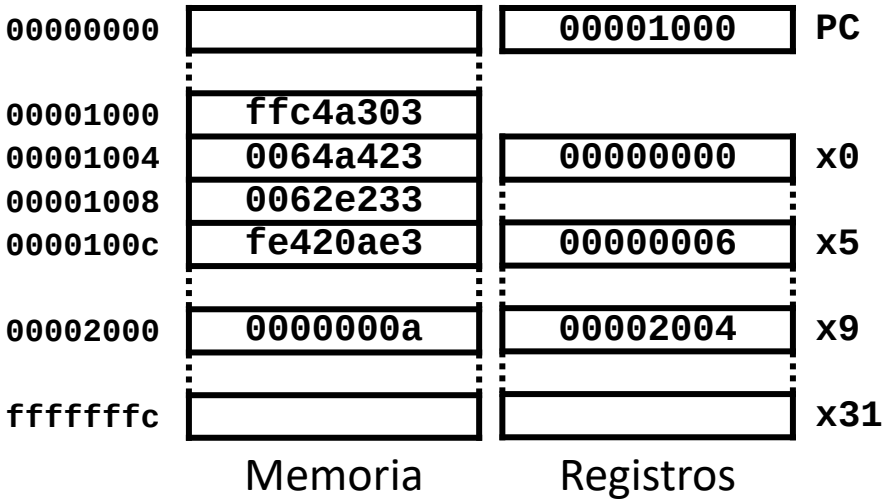
| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | | | | |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | = | | | | |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | = | | | | |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | = | | | | |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | | | | |
| A | ???????? | ???????? | 00002004 | 00002004 | = | | | | |
| B | ???????? | ???????? | basura[1] | basura | = | | | | |
| ALUout | ???????? | ???????? | basura[2] | 00002000 | = | | | | |
| x4 | ???????? | ???????? | ???????? | ???????? | = | | | | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | = | | | | |
| x6 | ???????? | ???????? | ???????? | ???????? | = | | | | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | = | | | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | = | | | | |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | = | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

basura[1] = BR[IR$_{24..20}$] = x28 = ???

basura[2] = OldPC + SExt(imm) = 0x1000 + (-4) = 0x00000ffc

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

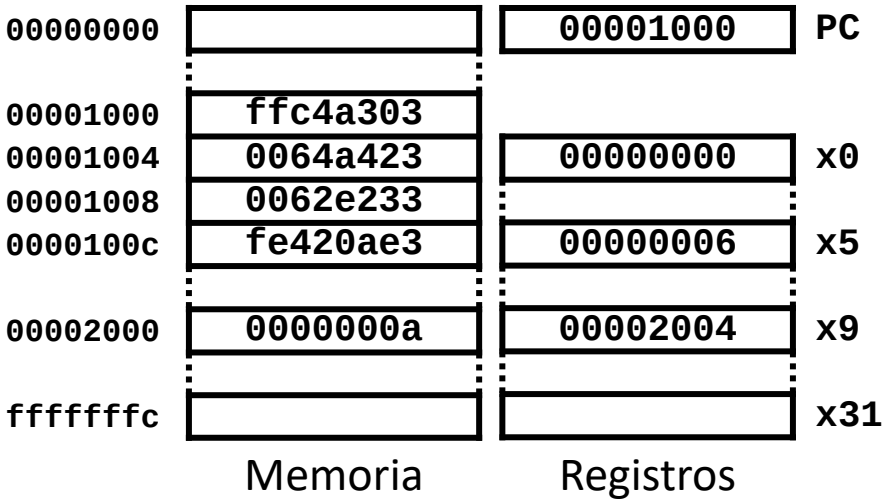
| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | | | |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | = | | | |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | = | | | |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | = | | | |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | = | | | |
| A | ???????? | ???????? | 00002004 | 00002004 | 00002004 | = | | | |
| B | ???????? | ???????? | basura$^1$ | basura | basura | = | | | |
| ALUout | ???????? | ???????? | basura$^2$ | 00002000 | 00002000 | = | | | |
| x4 | ???????? | ???????? | ???????? | ???????? | ???????? | = | | | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = | | | |
| x6 | ???????? | ???????? | ???????? | ???????? | ???????? | 0000000a | | | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = | | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | ???????? | = | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

basura$^1$ = BR[IR$_{24..20}$] = x28 = ???

basura$^2$ = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

```
   ...
L7:
0x1000  lw  x6, -4(x9)
0x1004  sw  x6, 8(x9)
0x1008  or  x4, x5, x6
0x100C  beq x4, x4, L7
   ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

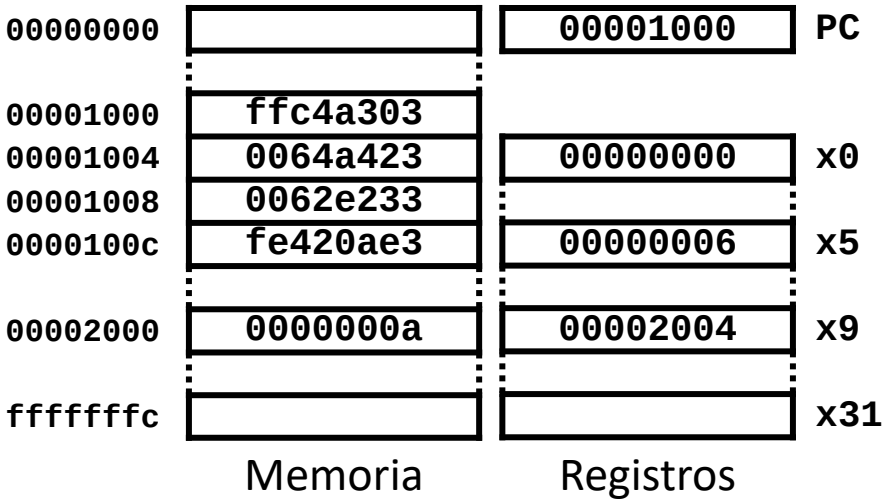
| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | | |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | 00001004 | 00001008 | | |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | 00001000 | 00001004 | | |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | | |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | = | | |
| A | ???????? | ???????? | 00002004 | 00002004 | 00002004 | 00002004 | = | | |
| B | ???????? | ???????? | basura[1] | basura | basura | basura | = | | |
| ALUout | ???????? | ???????? | basura[2] | 00002000 | 00002000 | 00002000 | = | | |
| x4 | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = | | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = | | |
| x6 | ???????? | ???????? | ???????? | ???????? | ???????? | 0000000a | = | | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

basura[1] = BR[IR$_{24..20}$] = x28 = ???

basura[2] = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | S2 | |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | 00001004 | 00001008 | = | |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | 00001000 | 00001004 | = | |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | = | |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | 0000000a | = | |
| A | ???????? | ???????? | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | |
| B | ???????? | ???????? | basura[1] | basura | basura | basura | basura | 0000000a | |
| ALUout | ???????? | ???????? | basura[2] | 00002000 | 00002000 | 00002000 | 00002000 | basura[3] | |
| x4 | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = | |
| x6 | ???????? | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | = | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

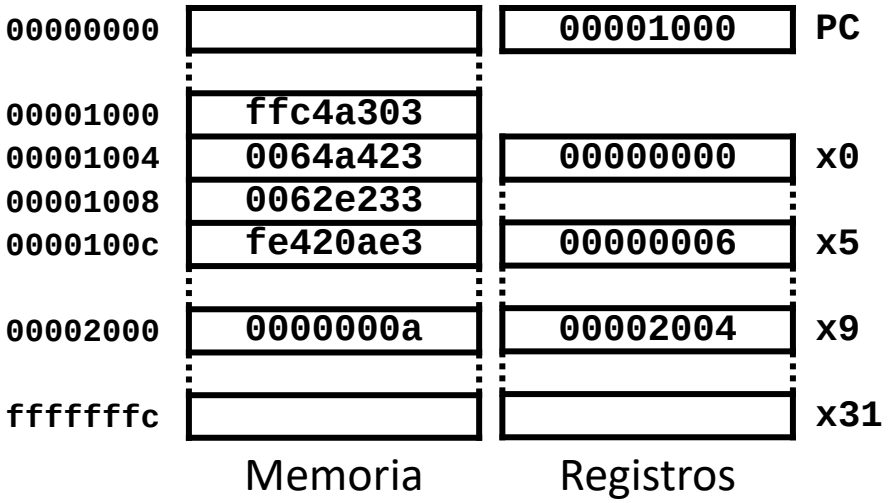
basura[1] = BR[IR$_{24..20}$] = x28 = ???

basura[2] = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

basura[3] = OldPC + SExt(imm) = 0x1004 + 8 = 0x0000100c

```
        ...
L7:
0x1000  lw  x6, -4(x9)
0x1004  sw  x6, 8(x9)
0x1008  or  x4, x5, x6
0x100C  beq x4, x4, L7
        ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | S2 | S5 |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | 00001004 | 00001008 | 00001008 | = |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | 00001000 | 00001004 | 00001004 | = |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | 0064a423 | = |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | 0000000a | 0000000a | = |
| A | ???????? | ???????? | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = |
| B | ???????? | ???????? | basura$^1$ | basura | basura | basura | basura | 0000000a | = |
| ALUout | ???????? | ???????? | basura$^2$ | 00002000 | 00002000 | 00002000 | 00002000 | basura$^3$ | 0000200c |
| x4 | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = |
| x6 | ???????? | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | 0000000a | = |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | = |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

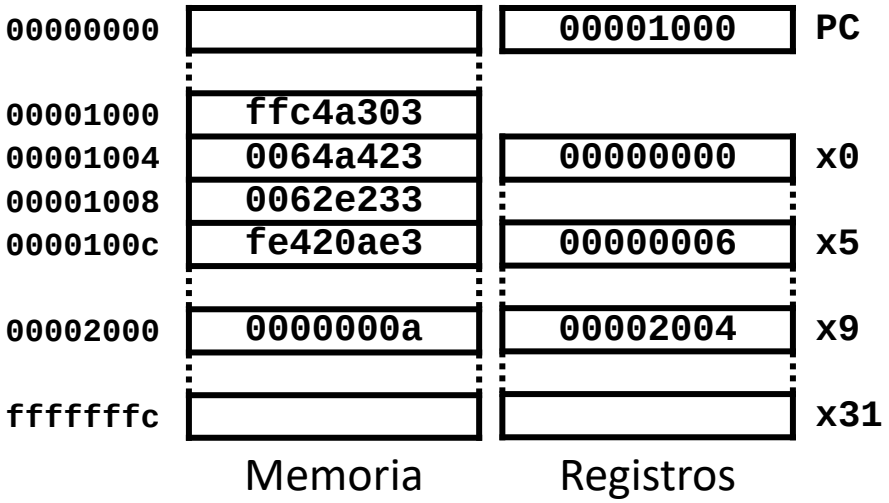
basura$^1$ = BR[IR$_{24..20}$] = x28 = ???

basura$^2$ = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

basura$^3$ = OldPC + SExt(imm) = 0x1004 + 8 = 0x0000100c

```
        ...
L7:
0x1000  lw  x6, -4(x9)
0x1004  sw  x6, 8(x9)
0x1008  or  x4, x5, x6
0x100C  beq x4, x4, L7
        ...
```

−12




# Diagrama de ejecución: registros y memoria (i)

| | lw x6, -4(x9) | | | | | sw x6, 8(x9) | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | S2 | S5 |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | 00001004 | 00001008 | 00001008 | 00001008 |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | 00001000 | 00001004 | 00001004 | 00001004 |
| IR | ???????? | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | 0064a423 | 0064a423 |
| MDR | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| A | ???????? | ???????? | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 |
| B | ???????? | ???????? | basura[1] | basura | basura | basura | basura | 0000000a | 0000000a |
| ALUout | ???????? | ???????? | basura[2] | 00002000 | 00002000 | 00002000 | 00002000 | basura[3] | 0000200c |
| x4 | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x6 | ???????? | ???????? | ???????? | ???????? | ???????? | 0000000a | 0000000a | 0000000a | 0000000a |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| MEM[0x200c] | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

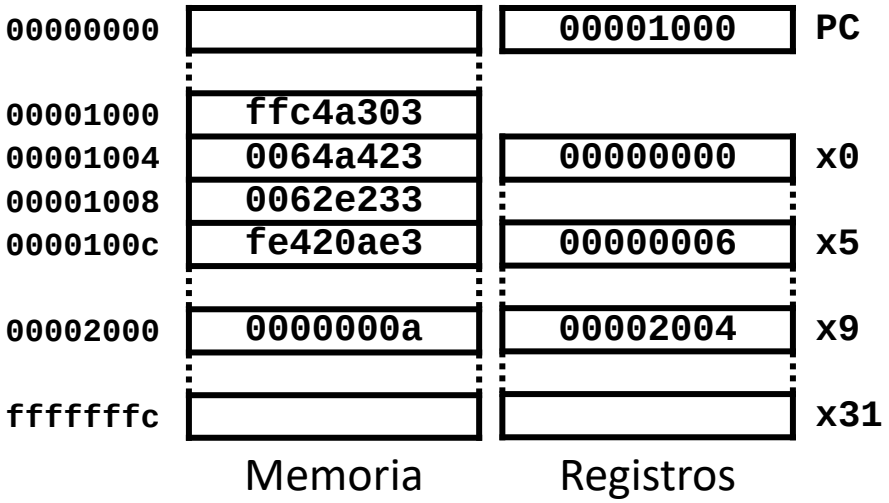
$basura^1$ = BR[$IR_{24..20}$] = x28 = ???

$basura^2$ = OldPC + SExt(imm) = 0x1000 + (−4) = 0x00000ffc

$basura^3$ = OldPC + SExt(imm) = 0x1004 + 8 = 0x0000100c

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12

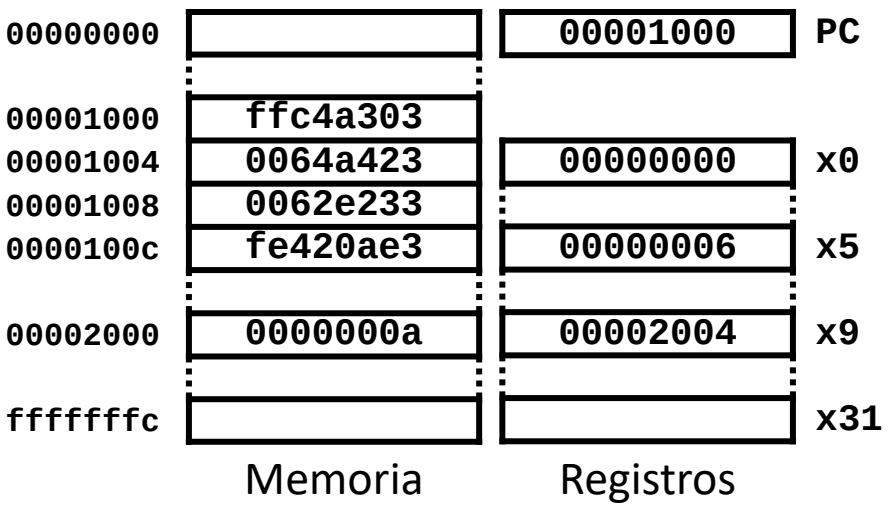



# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | | | | | | | |
| PC | = | | | | | | | |
| OldPC | = | | | | | | | |
| IR | = | | | | | | | |
| MDR | = | | | | | | | |
| A | = | | | | | | | |
| B | = | | | | | | | |
| ALUout | = | | | | | | | |
| x4 | = | | | | | | | |
| x5 | = | | | | | | | |
| x6 | = | | | | | | | |
| x9 | = | | | | | | | |
| MEM[0x2000] | = | | | | | | | |
| MEM[0x200c] | 0000000a | | | | | | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12

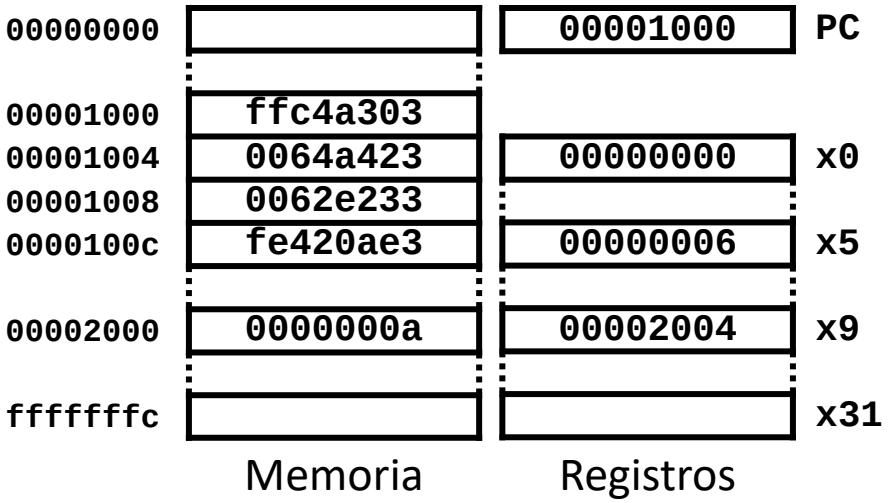



# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | | | | | | |
| PC | 00001008 | 0000100c | | | | | | |
| OldPC | 00001004 | 00001008 | | | | | | |
| IR | 0064a423 | 0062e233 | | | | | | |
| MDR | 0000000a | = | | | | | | |
| A | 00002004 | = | | | | | | |
| B | 0000000a | = | | | | | | |
| ALUout | 0000200c | = | | | | | | |
| x4 | ???????? | = | | | | | | |
| x5 | 00000006 | = | | | | | | |
| x6 | 0000000a | = | | | | | | |
| x9 | 00002004 | = | | | | | | |
| MEM[0x2000] | 0000000a | = | | | | | | |
| MEM[0x200c] | 0000000a | = | | | | | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12



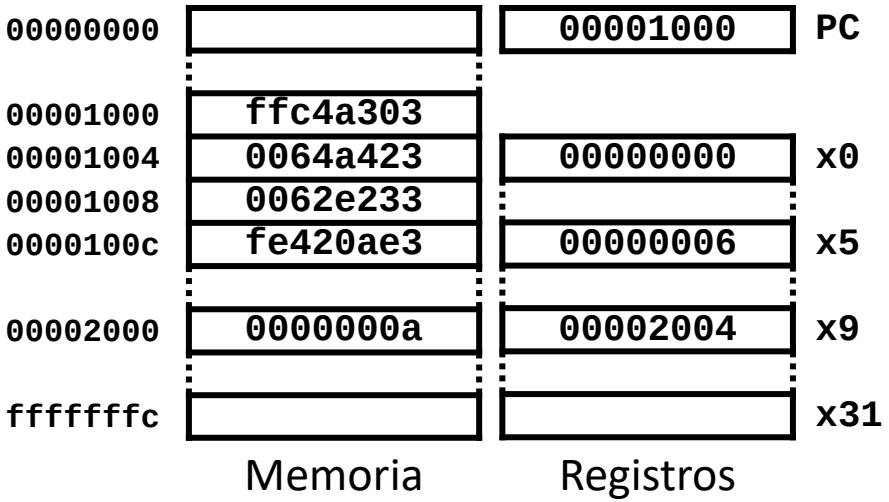

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | | | | |
| PC | 00001008 | 0000100c | = | | | | |
| OldPC | 00001004 | 00001008 | = | | | | |
| IR | 0064a423 | 0062e233 | = | | | | |
| MDR | 0000000a | 0000000a | = | | | | |
| A | 00002004 | 00002004 | 00000006 | | | | |
| B | 0000000a | 0000000a | 0000000a | | | | |
| ALUout | 0000200c | 0000200c | basura[4] | | | | |
| x4 | ???????? | ???????? | = | | | | |
| x5 | 00000006 | 00000006 | = | | | | |
| x6 | 0000000a | 0000000a | = | | | | |
| x9 | 00002004 | 00002004 | = | | | | |
| MEM[0x2000] | 0000000a | 0000000a | = | | | | |
| MEM[0x200c] | 0000000a | 0000000a | = | | | | |
| clk | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

basura[4] = depende de la implementación del DEC sEXt

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12



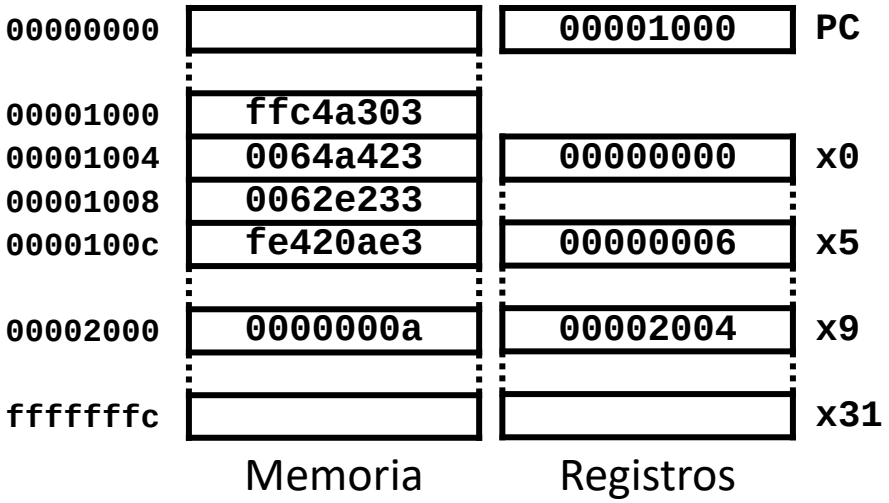

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | | | | |
| PC | 00001008 | 0000100c | 0000100c | = | | | | |
| OldPC | 00001004 | 00001008 | 00001008 | = | | | | |
| IR | 0064a423 | 0062e233 | 0062e233 | = | | | | |
| MDR | 0000000a | 0000000a | 0000000a | = | | | | |
| A | 00002004 | 00002004 | 00000006 | = | | | | |
| B | 0000000a | 0000000a | 0000000a | = | | | | |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | | | | |
| x4 | ???????? | ???????? | ???????? | = | | | | |
| x5 | 00000006 | 00000006 | 00000006 | = | | | | |
| x6 | 0000000a | 0000000a | 0000000a | = | | | | |
| x9 | 00002004 | 00002004 | 00002004 | = | | | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | = | | | | |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | = | | | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12



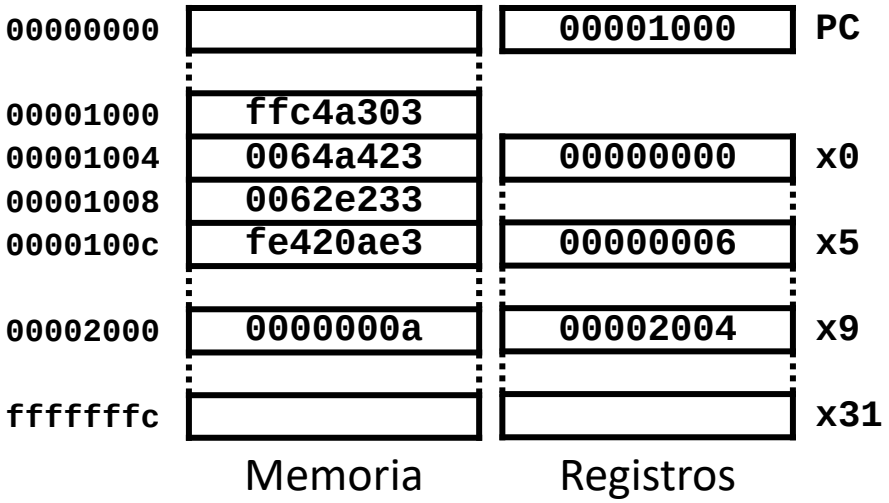

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | | | |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | = | | | |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | = | | | |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | = | | | |
| MDR | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| A | 00002004 | 00002004 | 00000006 | 00000006 | = | | | |
| B | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | = | | | |
| x4 | ???????? | ???????? | ???????? | ???????? | 0000000e | | | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | = | | | |
| x6 | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | = | | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | 0000000a | = | | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

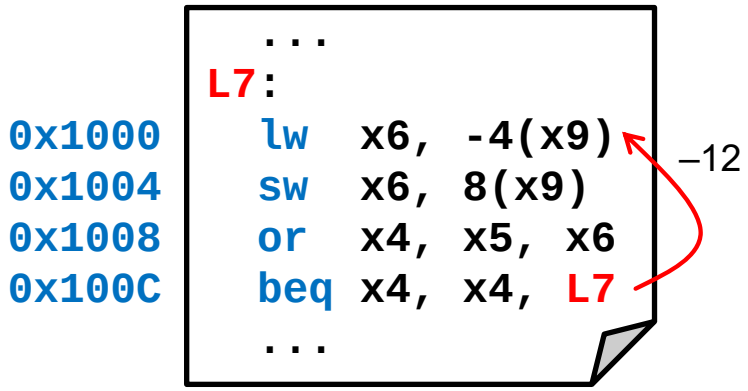
–12

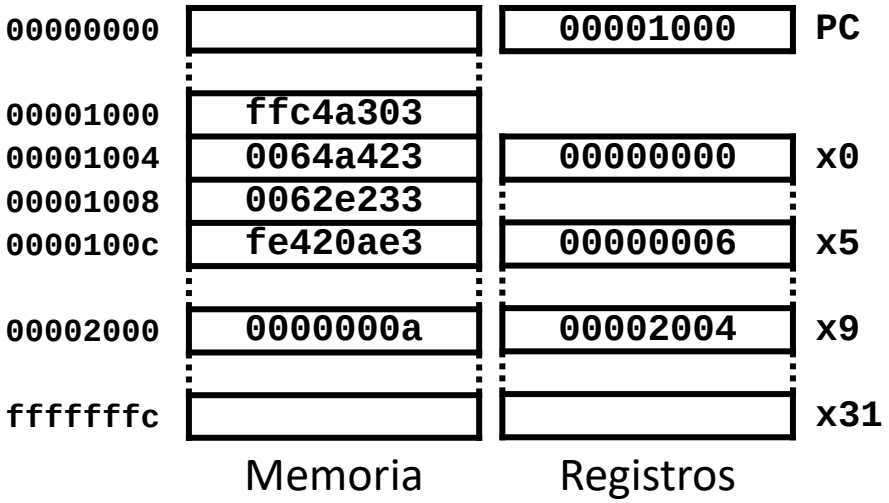



# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | | |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 0000100c | 00001010 | | |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | 00001008 | 0000100c | | |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | | |
| MDR | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| A | 00002004 | 00002004 | 00000006 | 00000006 | 00000006 | = | | |
| B | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | 0000000e | = | | |
| x4 | ???????? | ???????? | ???????? | ???????? | 0000000e | = | | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = | | |
| x6 | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = | | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt



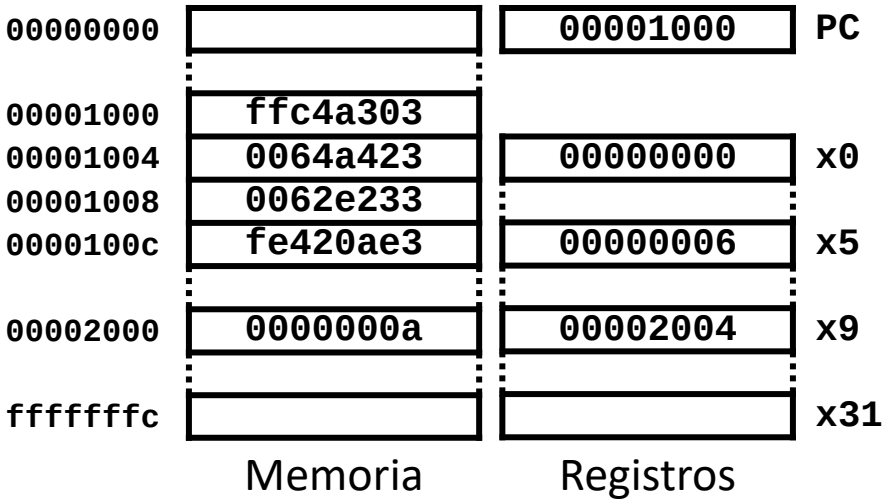

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | S10 | |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 0000100c | 00001010 | = | |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | 00001008 | 0000100c | = | |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | = | |
| MDR | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | |
| A | 00002004 | 00002004 | 00000006 | 00000006 | 00000006 | 00000006 | 0000000e | |
| B | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000e | |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | 0000000e | 0000000e | 00001000 | |
| x4 | ???????? | ???????? | ???????? | ???????? | 0000000e | 0000000e | = | |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = | |
| x6 | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = | |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt

```
          ...
L7:
0x1000    lw   x6, -4(x9)
0x1004    sw   x6, 8(x9)
0x1008    or   x4, x5, x6
0x100C    beq  x4, x4, L7
          ...
```

–12



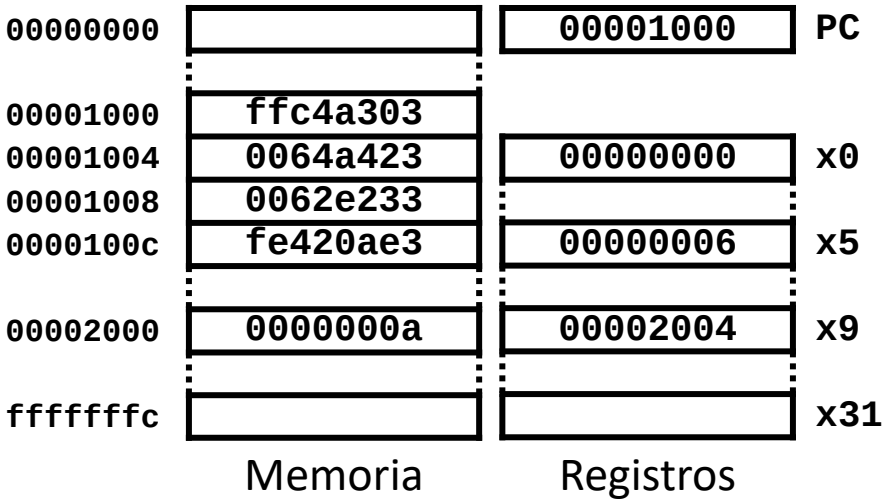

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | S10 | S0 |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 0000100c | 00001010 | 00001010 | 00001000 |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | 00001008 | 0000100c | 0000100c | = |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | fe420ae3 | = |
| MDR | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = |
| A | 00002004 | 00002004 | 00000006 | 00000006 | 00000006 | 00000006 | 0000000e | = |
| B | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000e | = |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | 0000000e | 0000000e | 00001000 | = |
| x4 | ???????? | ???????? | ???????? | ???????? | 0000000e | 0000000e | 0000000e | = |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | = |
| x6 | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | = |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | = |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt

```
        ...
L7:
0x1000  lw  x6, -4(x9)
0x1004  sw  x6, 8(x9)
0x1008  or  x4, x5, x6
0x100C  beq x4, x4, L7
        ...
```

−12



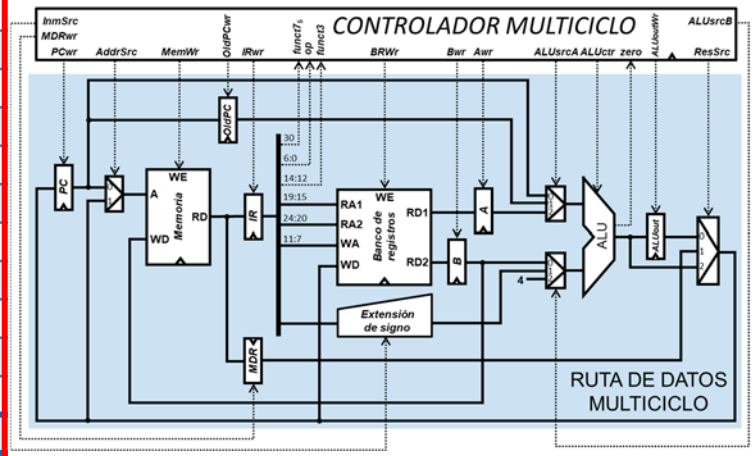

# Diagrama de ejecución: registros y memoria (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | S10 | S0 |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 0000100c | 00001010 | 00001010 | 00001000 |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | 00001008 | 0000100c | 0000100c | 0000100c |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | fe420ae3 | fe420ae3 |
| MDR | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| A | 00002004 | 00002004 | 00000006 | 00000006 | 00000006 | 00000006 | 0000000e | 0000000e |
| B | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000e | 0000000e |
| ALUout | 0000200c | 0000200c | basura[4] | 0000000e | 0000000e | 0000000e | 00001000 | 00001000 |
| x4 | ???????? | ???????? | ???????? | ???????? | 0000000e | 0000000e | 0000000e | 0000000e |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x6 | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| x9 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 | 00002004 |
| MEM[0x2000] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| MEM[0x200c] | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a | 0000000a |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |

basura[4] = depende de la implementación del DEC sEXt

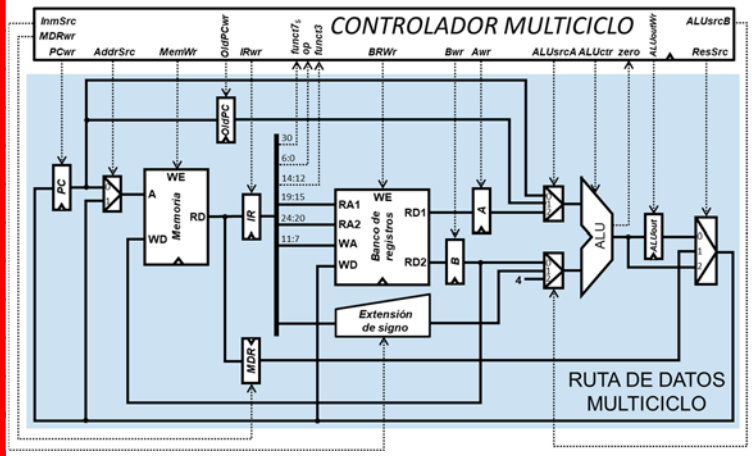
# Diagrama de ejecución: señales de estado y control (i)

| | `lw x6, -4(x9)` | | | | | `sw x6, 8(x9)` | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | S2 | S5 |
| IR | 00000000 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | 0064a423 | 0064a423 |
| op | 0000000 | 0000011 | 0000011 | 0000011 | 0000011 | 0000011 | 0100011 | 0100011 | 0100011 |
| $funct7_5$ | 0 | – | – | – | – | – | – | – | – |
| funct3 | 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 |
| zero | 0 | – | – | – | – | – | – | – | – |
| Branch | | | | | | | | | |
| PCupdate | | | | | | | | | |
| AddrSrc | | | | | | | | | |
| MemWr | | | | | | | | | |
| OldPCwr | | | | | | | | | |
| IRwr | | | | | | | | | |
| MDRwr | | | | | | | | | |
| BRwr | | | | | | | | | |
| Awr | | | | | | | | | |
| Bwr | | | | | | | | | |
| ALUsrcA | | | | | | | | | |
| ALUsrcB | | | | | | | | | |
| ALUop | | | | | | | | | |
| ALUoutWr | | | | | | | | | |
| ResSrc | | | | | | | | | |
| InmSrc | | | | | | | | | |
| ALUctr | | | | | | | | | |
| PCwr | | | | | | | | | |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |



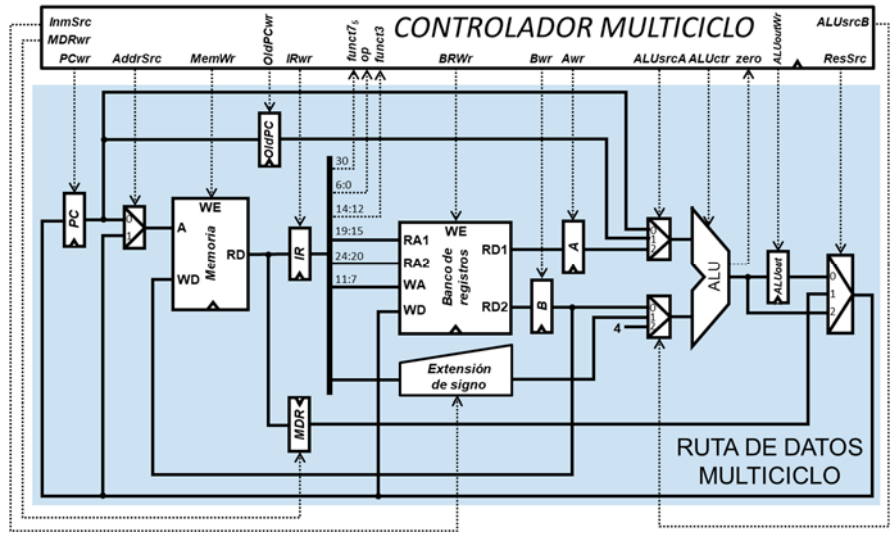
# Diagrama de ejecución: señales de estado y control (i)

| | `lw x6, -4(x9)` | | | | | `sw x6, 8(x9)` | | | |
|---|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S2 | S3 | S4 | S0 | S1 | S2 | S5 |
| IR | 00000000 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | ffc4a303 | 0064a423 | 0064a423 | 0064a423 |
| op | 0000000 | 0000011 | 0000011 | 0000011 | 0000011 | 0000011 | 0100011 | 0100011 | 0100011 |
| $funct7_5$ | 0 | – | – | – | – | – | – | – | – |
| funct3 | 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 | 010 |
| zero | 0 | – | – | – | – | – | – | – | – |
| Branch | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCupdate | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| AddrSrc | 0 | – | – | 1 | – | 0 | – | – | 1 |
| MemWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| OldPCwr | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| IRwr | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| MDRwr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BRwr | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Awr | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Bwr | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ALUsrcA | 00 | 01 | 10 | – | – | 00 | 01 | 10 | – |
| ALUsrcB | 10 | 01 | 01 | – | – | 10 | 01 | 01 | – |
| ALUop | 00 | 00 | 00 | – | – | 00 | 00 | 00 | – |
| ALUoutWr | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| ResSrc | 10 | – | – | 00 | 01 | 10 | – | – | 00 |
| InmSrc | – | 00 | 00 | 00 | 00 | 00 | 01 | 01 | 01 |
| ALUctr | 000 | 000 | 000 | – | – | 000 | 000 | 000 | – |
| PCwr | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| clk | | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |



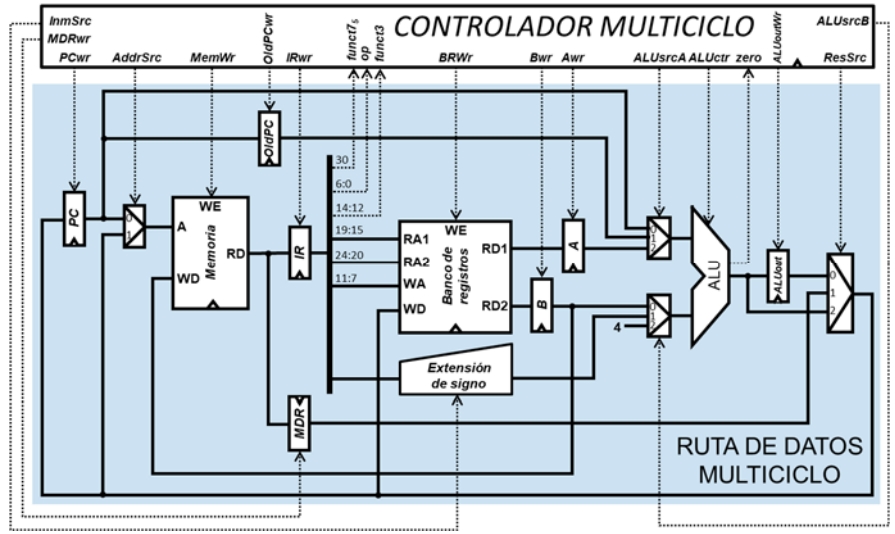
# Diagrama de ejecución: señales de estado y control (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | S10 | S0 |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | fe420ae3 | fe420ae3 |
| op | 0100011 | 0110011 | 0110011 | 0110011 | 0110011 | 1100011 | 1100011 | 1100011 |
| $funct7_5$ | - | 0 | 0 | 0 | 0 | - | - | - |
| funct3 | 010 | 110 | 110 | 110 | 110 | 000 | 000 | 000 |
| zero | - | - | - | - | - | 1 | 1 | 1 |
| Branch | | | | | | | | |
| PCupdate | | | | | | | | |
| AddrSrc | | | | | | | | |
| MemWr | | | | | | | | |
| OldPCwr | | | | | | | | |
| IRwr | | | | | | | | |
| MDRwr | | | | | | | | |
| BRwr | | | | | | | | |
| Awr | | | | | | | | |
| Bwr | | | | | | | | |
| ALUsrcA | | | | | | | | |
| ALUsrcB | | | | | | | | |
| ALUop | | | | | | | | |
| ALUoutWr | | | | | | | | |
| ResSrc | | | | | | | | |
| InmSrc | | | | | | | | |
| ALUctr | | | | | | | | |
| PCwr | | | | | | | | |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |



# Diagrama de ejecución: señales de estado y control (ii)

| | or x4, x5, x6 | | | | beq x4, x4, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S6 | S7 | S0 | S1 | S10 | S0 |
| IR | 0064a423 | 0062e233 | 0062e233 | 0062e233 | 0062e233 | fe420ae3 | fe420ae3 | fe420ae3 |
| op | 0100011 | 0110011 | 0110011 | 0110011 | 0110011 | 1100011 | 1100011 | 1100011 |
| $funct7_5$ | – | 0 | 0 | 0 | 0 | – | – | – |
| funct3 | 010 | 110 | 110 | 110 | 110 | 000 | 000 | 000 |
| zero | – | – | – | – | – | 1 | 1 | 1 |
| Branch | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PCupdate | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| AddrSrc | 0 | – | – | – | 0 | – | – | 0 |
| MemWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OldPCwr | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| IRwr | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| MDRwr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BRwr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Awr | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Bwr | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| ALUsrcA | 00 | 01 | 10 | – | 00 | 01 | 10 | 00 |
| ALUsrcB | 10 | 01 | 00 | – | 10 | 01 | 00 | 10 |
| ALUop | 00 | 00 | 10 | – | 00 | 00 | 01 | 00 |
| ALUoutWr | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| ResSrc | 10 | – | – | 00 | 10 | – | 00 | 10 |
| InmSrc | 01 | – | – | – | – | 10 | 10 | 10 |
| ALUctr | 000 | 000 | 011 | – | 000 | 000 | 001 | 000 |
| PCwr | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| clk | | | | | | | | |
| # ciclo | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |



**10)** Supongamos que sobre el procesador **multiciclo** estudiado en clase se quiere ejecutar el programa que se muestra a continuación, siendo el contenido inicial de los registros y la memoria el mostrado en la figura. Representa los diagramas de ejecución para los registros y la memoria, así como para las señales de estado y de control, de modo que se visualicen sus valores en cada ciclo de reloj correspondiente a la ejecución del programa.

C/C++

```
...
for( i=4; i!=6; i=i+1 )
{
  a = a + i;
}
...
```

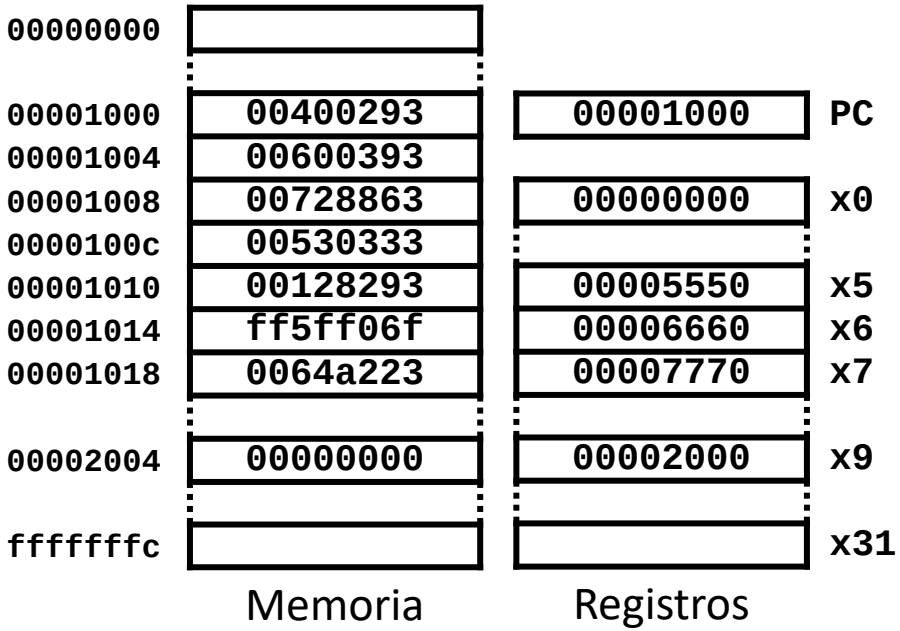
*i* → *x5*
*a* → *x6*

ASM

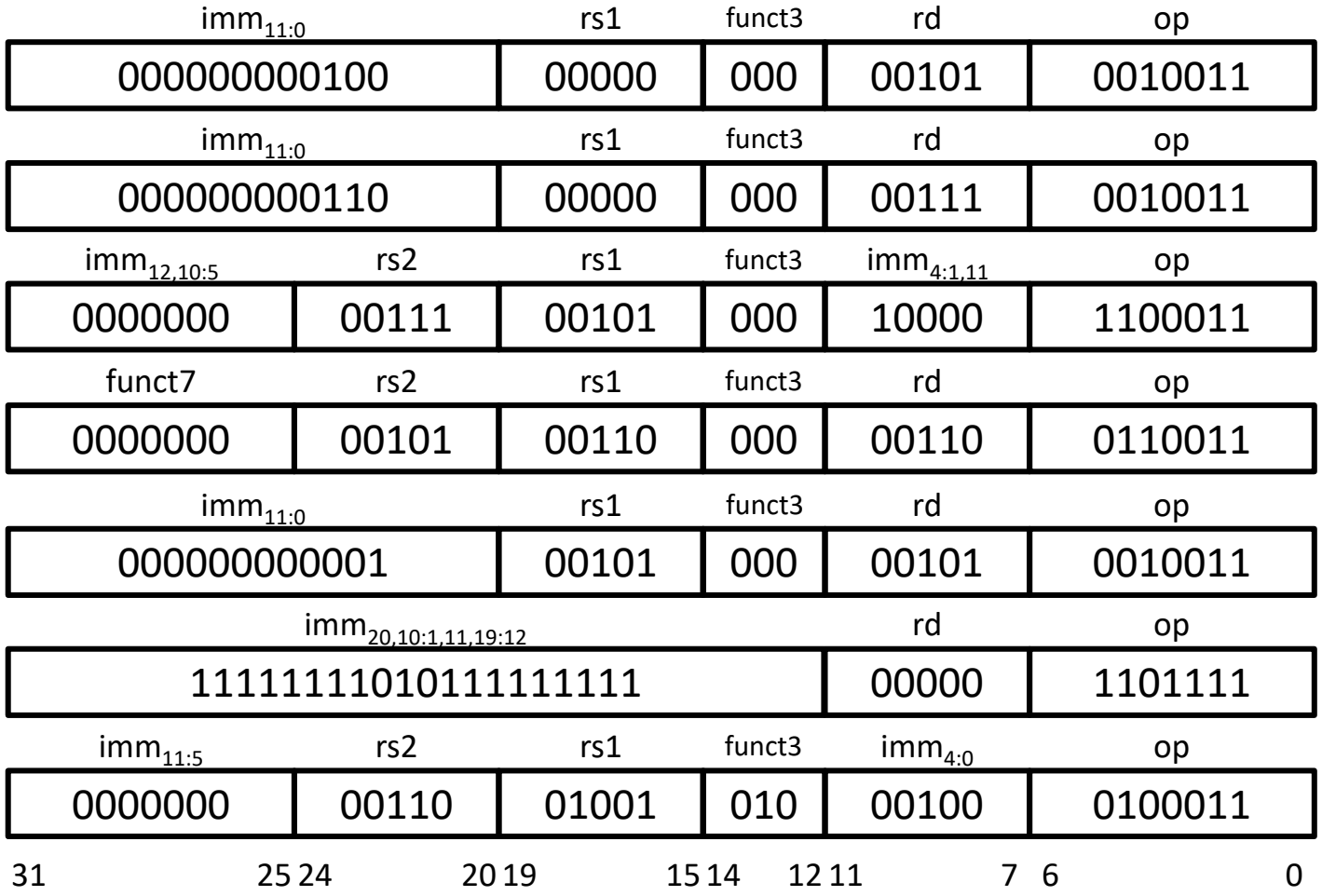
```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```

+16

−12



La representación en código máquina de las instrucciones que conforman el programa es la siguiente:




**0x00400293**
`addi x5,x0,4`

**0x00600393**
`addi x7,x0,6`

**0x00728863**
`beq x5,x7,16`

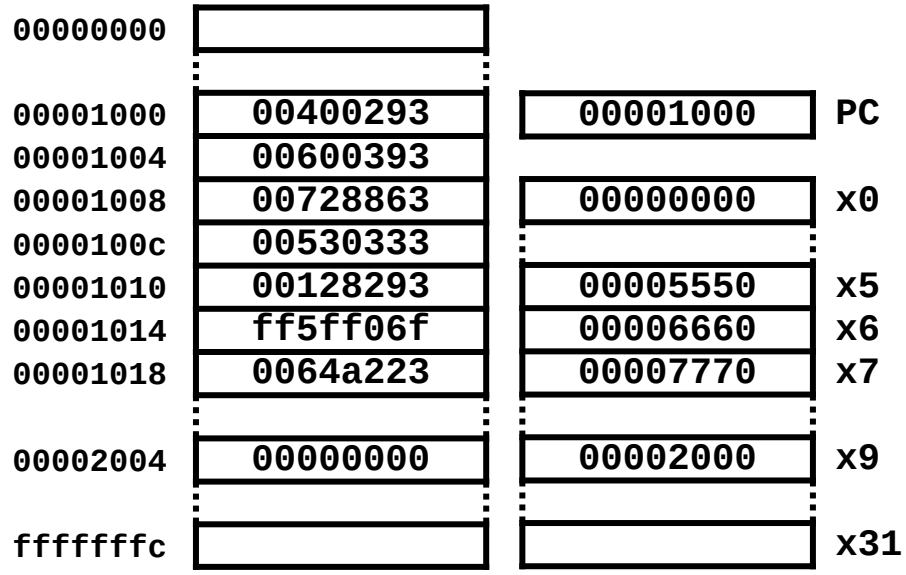
**0x00530333**
`add x6,x6,x5`

**0x00128293**
`addi x5,x5,1`

**0xff5ff06f**
`jal x0,-12`

**0x0064a223**
`sw x6,4(x9)`

```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```




# Diagrama de ejecución: registros y memoria (i)

| | addi x5, x0, 4 | | | | addi x7, x0, 6 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S8 | S7 |
| PC | 00001000 | 00001004 | = | = | = | 00001008 | = | = |
| OldPC | ???????? | 00001000 | = | = | = | 00001004 | = | = |
| IR | ???????? | 00400293 | = | = | = | 00600393 | = | = |
| MDR | ???????? | = | = | = | = | = | = | = |
| A | ???????? | = | 00000000 | = | = | = | 00000000 | = |
| B | ???????? | = | basura$^1$ | = | = | = | basura$^3$ | = |
| ALUout | ???????? | = | basura$^2$ | 00000004 | = | = | basura$^4$ | 00000006 |
| x5 | 00005550 | = | = | = | 00000004 | = | = | = |
| x6 | 00006660 | = | = | = | = | = | = | = |
| x7 | 00007770 | = | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = | = |
| clk | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

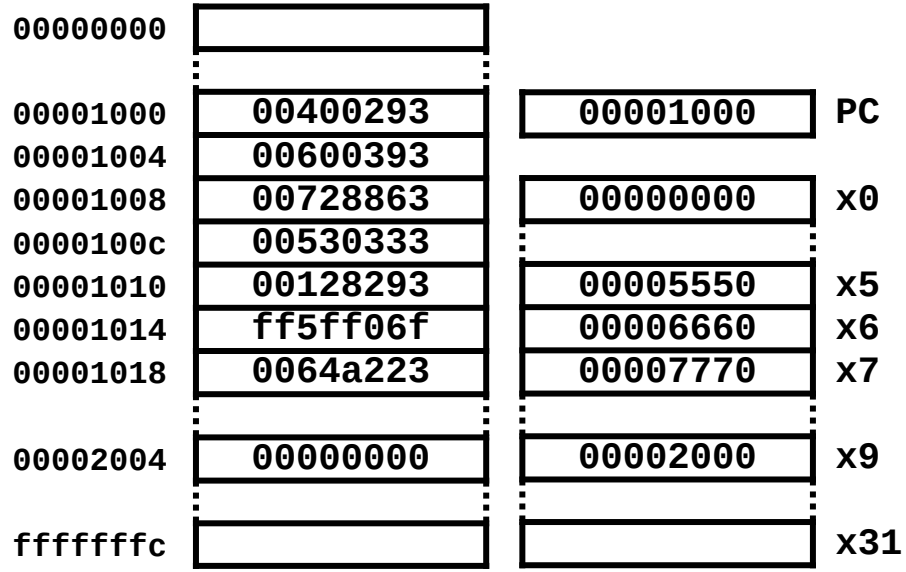
basura$^1$ = BR[IR$_{24..20}$] = x4 = ???

basura$^2$ = OldPC + SExt(imm) = 0x1000 + 4 = 0x00001004

basura$^3$ = BR[IR$_{24..20}$] = x6 = 0x00006660

basura$^4$ = OldPC + SExt(imm) = 0x1004 + 6 = 0x0000100a

```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```




# Diagrama de ejecución: registros y memoria (i)

| | addi x5, x0, 4 | | | | addi x7, x0, 6 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S8 | S7 |
| PC | 00001000 | 00001004 | 00001004 | 00001004 | 00001004 | 00001008 | 00001008 | 00001008 |
| OldPC | ???????? | 00001000 | 00001000 | 00001000 | 00001000 | 00001004 | 00001004 | 00001004 |
| IR | ???????? | 00400293 | 00400293 | 00400293 | 00400293 | 00600393 | 00600393 | 00600393 |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | ???????? | ???????? | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| B | ???????? | ???????? | basura$^1$ | basura | basura | basura | basura$^3$ | basura |
| ALUout | ???????? | ???????? | basura$^2$ | 00000004 | 00000004 | 00000004 | basura$^4$ | 00000006 |
| x5 | 00005550 | 00005550 | 00005550 | 00005550 | 00000004 | 00000004 | 00000004 | 00000004 |
| x6 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 |
| x7 | 00007770 | 00007770 | 00007770 | 00007770 | 00007770 | 00007770 | 00007770 | 00007770 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| clk | | | | | | | | |
| # ciclo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

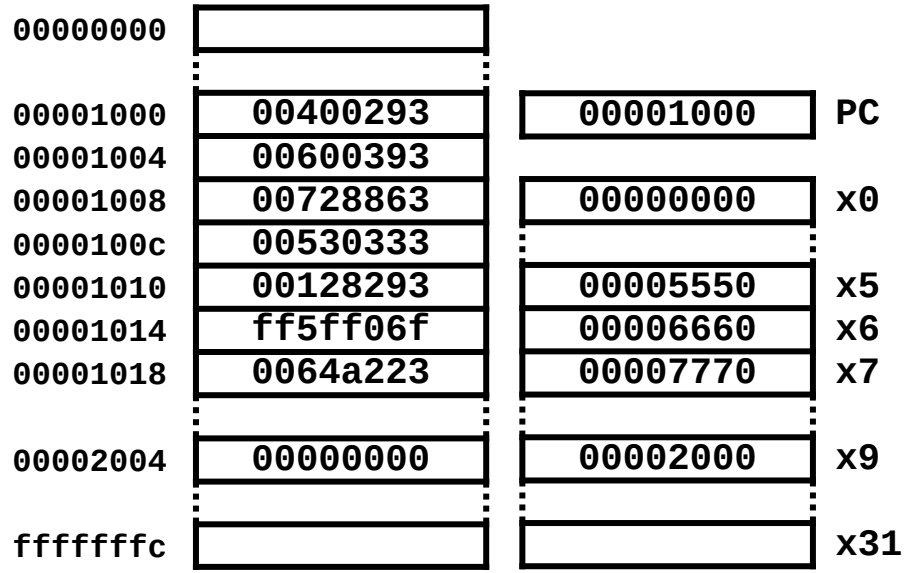
basura$^1$ = BR[IR$_{24..20}$] = x4 = ???

basura$^2$ = OldPC + SExt(imm) = 0x1000 + 4 = 0x00001004

basura$^3$ = BR[IR$_{24..20}$] = x6 = 0x00006660

basura$^4$ = OldPC + SExt(imm) = 0x1004 + 6 = 0x0000100a

```
...
0x1000    addi x5, x0, 4
0x1004    addi x7, x0, 6
for:
0x1008    beq  x5, x7, efor
0x100c    add  x6, x6, x5
0x1010    addi x5, x5, 1
0x1014    jal  x0, for
efor:
0x1018    sw   x6, 4(x9)
...
```

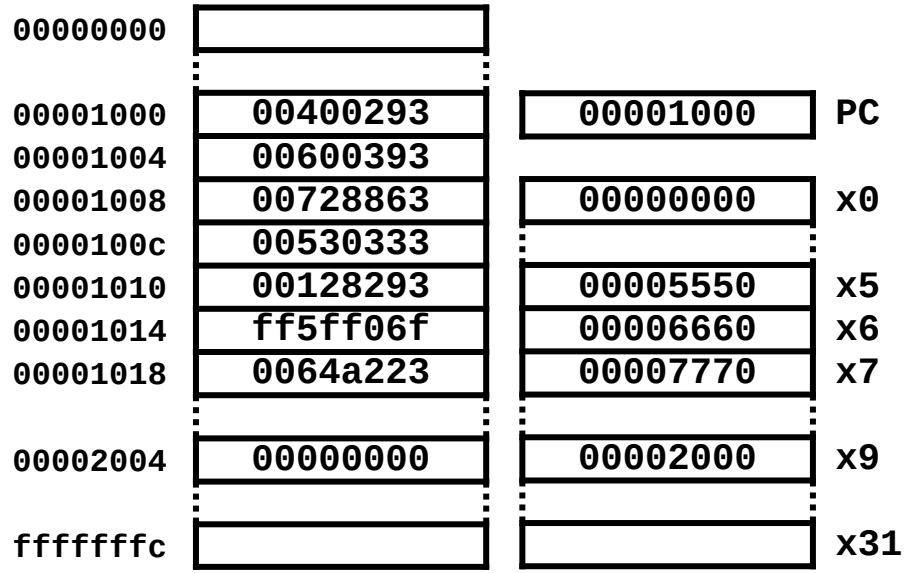



# Diagrama de ejecución: registros y memoria (ii)

| | beq x5, x7, 16 | | | add x6, x6, x5 | | | |
|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S6 | S7 |
| PC | 00001008 | 0000100c | = | = | 00001010 | = | = |
| OldPC | 00001004 | 00001008 | = | = | 0000100c | = | = |
| IR | 00600393 | 00728863 | = | = | 00530333 | = | = |
| MDR | ???????? | = | = | = | = | = | = |
| A | 00000000 | = | 00000004 | = | = | 00006660 | = |
| B | basura | = | 00000006 | = | = | 00000004 | = |
| ALUout | 00000006 | = | 00001018 | = | = | basura[5] | 00006664 |
| x5 | 00000004 | = | = | = | = | = | = |
| x6 | 00006660 | = | = | = | = | = | = |
| x7 | 00000006 | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = |
| clk | | | | | | | |
| # ciclo | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

basura[5] = depende de la implementación del DEC sEXt

```
...
0x1000    addi x5, x0, 4
0x1004    addi x7, x0, 6
for:
0x1008    beq  x5, x7, efor
0x100c    add  x6, x6, x5
0x1010    addi x5, x5, 1
0x1014    jal  x0, for
efor:
0x1018    sw   x6, 4(x9)
...
```

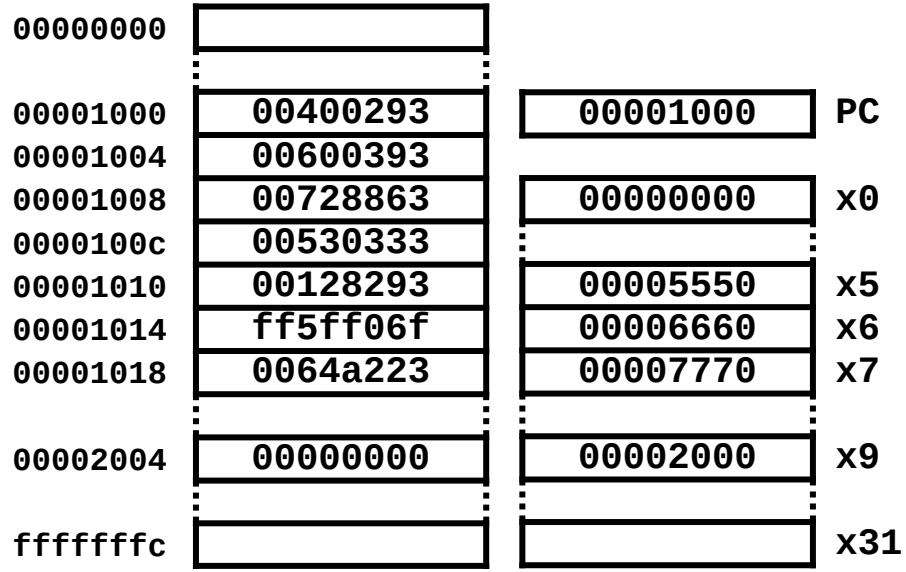



# Diagrama de ejecución: registros y memoria (ii)

| | beq x5, x7, 16 | | | add x6, x6, x5 | | | |
|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S6 | S7 |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 00001010 | 00001010 | 00001010 |
| OldPC | 00001004 | 00001008 | 00001008 | 00001008 | 0000100c | 0000100c | 0000100c |
| IR | 00600393 | 00728863 | 00728863 | 00728863 | 00530333 | 00530333 | 00530333 |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | 00000000 | 00000000 | 00000004 | 00000004 | 00000004 | 00006660 | 00006660 |
| B | basura | basura | 00000006 | 00000006 | 00000006 | 00000004 | 00000004 |
| ALUout | 00000006 | 00000006 | 00001018 | 00001018 | 00001018 | basura[5] | 00006664 |
| x5 | 00000004 | 00000004 | 00000004 | 00000004 | 00000004 | 00000004 | 00000004 |
| x6 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 | 00006660 |
| x7 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| clk | | | | | | | |
| # ciclo | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

basura[5] = depende de la implementación del DEC sEXt

```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```




# Diagrama de ejecución: registros y memoria (iii)

| | addi x5, x5, 1 | | | | jal x0, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S9 | S7 |
| PC | 00001010 | 00001014 | = | = | = | 00001018 | = | 00001008 |
| OldPC | 0000100c | 00001010 | = | = | = | 00001014 | = | = |
| IR | 00530333 | 00128293 | = | = | = | ff5ff06f | = | = |
| MDR | ???????? | = | = | = | = | = | = | = |
| A | 00006660 | = | 00000004 | = | = | = | basura$^{8}$ | = |
| B | 00000004 | = | basura$^{6}$ | = | = | = | basura$^{9}$ | = |
| ALUout | 00006664 | = | basura$^{7}$ | 00000005 | = | = | 00001008 | 00001018 |
| x5 | 00000004 | = | = | = | 00000005 | = | = | = |
| x6 | 00006664 | = | = | = | = | = | = | = |
| x7 | 00000006 | = | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = | = |
| clk | | | | | | | | |
| # ciclo | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

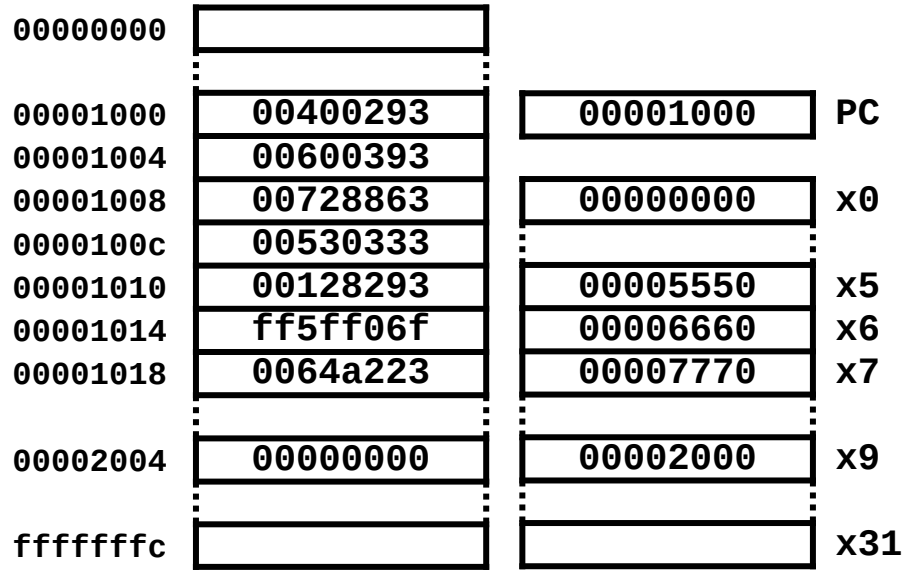
basura$^{6}$ = BR[IR$_{24..20}$]
= x21 = ???

basura$^{7}$ = OldPC + SExt(imm)
= 0x1010 + 1 = 0x00001011

basura$^{8}$ = BR[IR$_{19..15}$]
= x31 = ???

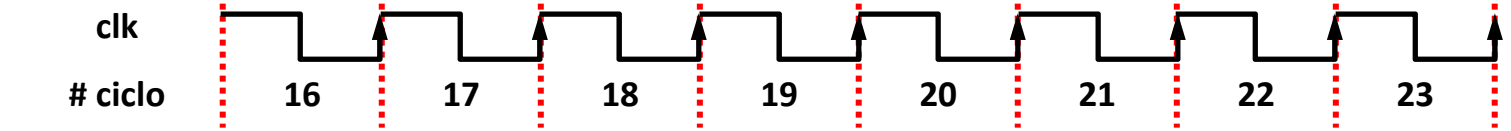
basura$^{9}$ = BR[IR$_{24..20}$]
= 0x1004 + 6 = 0x0000100a

```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```




# Diagrama de ejecución: registros y memoria (iii)

| | addi x5, x5, 1 | | | | jal x0, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S9 | S7 |
| PC | 00001010 | 00001014 | 00001010 | 00001014 | 00001010 | 00001018 | 00001018 | 00001008 |
| OldPC | 0000100c | 00001010 | 0000100c | 00001010 | 0000100c | 00001014 | 00001014 | 00001014 |
| IR | 00530333 | 00128293 | 00530333 | 00128293 | 00530333 | ff5ff06f | ff5ff06f | ff5ff06f |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | 00006660 | 00006660 | 00000004 | 00000004 | 00000004 | 00000004 | basura$^{8}$ | basura |
| B | 00000004 | 00000004 | basura$^{6}$ | basura | basura | basura | basura$^{9}$ | basura |
| ALUout | 00006664 | 00006664 | basura$^{7}$ | 00000005 | 00000005 | 00000005 | 00001008 | 00001018 |
| x5 | 00000004 | 00000004 | 00000004 | 00000004 | 00000005 | 00000005 | 00000005 | 00000005 |
| x6 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 |
| x7 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |



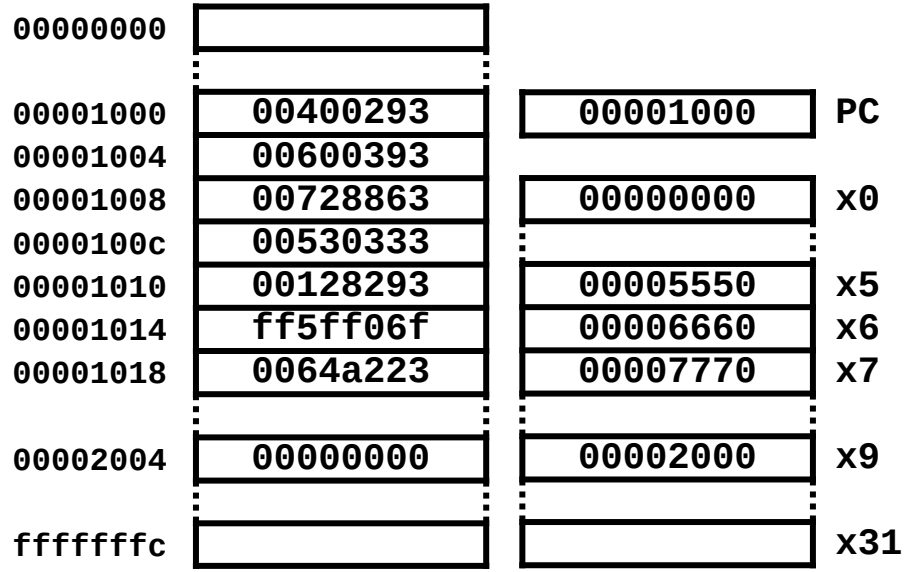

$basura^{6}$ = BR[$IR_{24..20}$]
= x21 = ???

$basura^{7}$ = OldPC + SExt(imm)
= 0x1010 + 1 = 0x00001011

$basura^{8}$ = BR[$IR_{19..15}$]
= x31 = ???

$basura^{9}$ = BR[$IR_{24..20}$]
= 0x1004 + 6 = 0x0000100a

```
...
0x1000    addi x5, x0, 4
0x1004    addi x7, x0, 6
for:
0x1008    beq  x5, x7, efor
0x100c    add  x6, x6, x5
0x1010    addi x5, x5, 1
0x1014    jal  x0, for
efor:
0x1018    sw   x6, 4(x9)
...
```

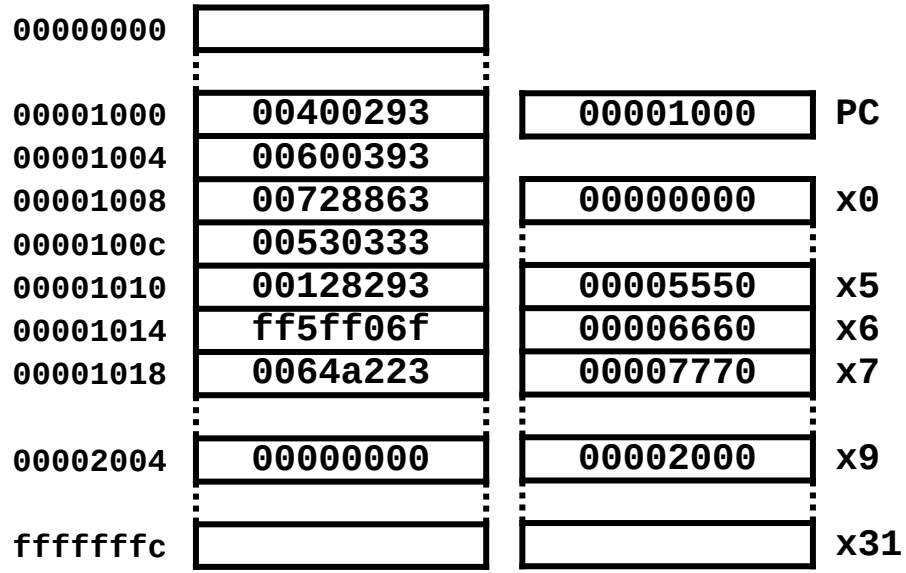



# Diagrama de ejecución: registros y memoria (iv)

| | beq x5, x7, 16 | | | add x6, x6, x5 | | | |
|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S6 | S7 |
| PC | 00001008 | 0000100c | = | = | 00001010 | = | = |
| OldPC | 00001014 | 00001008 | = | = | 0000100c | = | = |
| IR | ff5ff06f | 00728863 | = | = | 00530333 | = | = |
| MDR | ???????? | = | = | = | = | = | = |
| A | basura | = | 00000005 | = | = | 00006664 | = |
| B | basura | = | 00000006 | = | = | 00000005 | = |
| ALUout | 00001018 | = | 00001018 | = | = | basura[10] | 00006669 |
| x5 | 00000005 | = | = | = | = | = | = |
| x6 | 00006664 | = | = | = | = | = | = |
| x7 | 00000006 | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = |
| clk | | | | | | | |
| # ciclo | 24 | 25 | 26 | 27 | 28 | 29 | 30 |

basura[10] = depende de la implementación del DEC sEXt

```
0x1000    ...
            addi x5, x0, 4
0x1004      addi x7, x0, 6
          for:
0x1008      beq  x5, x7, efor
0x100c      add  x6, x6, x5
0x1010      addi x5, x5, 1
0x1014      jal  x0, for
          efor:
0x1018      sw   x6, 4(x9)
            ...
```

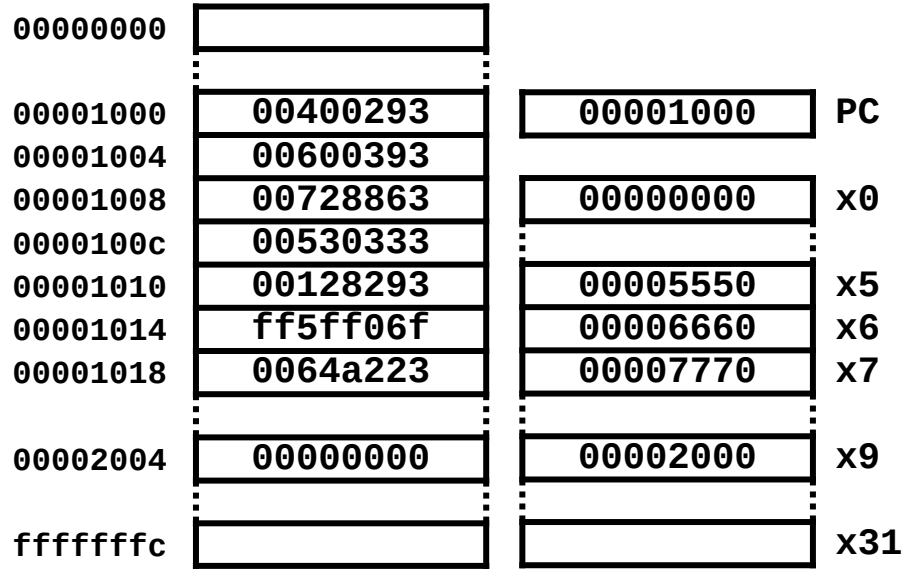



# Diagrama de ejecución: registros y memoria (iv)

| | beq x5, x7, 16 | | | add x6, x6, x5 | | | |
|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S6 | S7 |
| PC | 00001008 | 0000100c | 0000100c | 0000100c | 00001010 | 00001010 | 00001010 |
| OldPC | 00001014 | 00001008 | 00001008 | 00001008 | 0000100c | 0000100c | 0000100c |
| IR | ff5ff06f | 00728863 | 00728863 | 00728863 | 00530333 | 00530333 | 00530333 |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | basura | basura | 00000005 | 00000005 | 00000005 | 00006664 | 00006664 |
| B | basura | basura | 00000006 | 00000006 | 00000006 | 00000005 | 00000005 |
| ALUout | 00001018 | 00001018 | 00001018 | 00001018 | 00001018 | basura$^{10}$ | 00006669 |
| x5 | 00000005 | 00000005 | 00000005 | 00000005 | 00000005 | 00000005 | 00000005 |
| x6 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 | 00006664 |
| x7 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| clk | | | | | | | |
| # ciclo | 24 | 25 | 26 | 27 | 28 | 29 | 30 |

basura$^{10}$ = depende de la implementación del DEC sEXt

```
0x1000    ...
            addi x5, x0, 4
0x1004      addi x7, x0, 6
          for:
0x1008      beq  x5, x7, efor
0x100c      add  x6, x6, x5
0x1010      addi x5, x5, 1
0x1014      jal  x0, for
          efor:
0x1018      sw   x6, 4(x9)
            ...
```




# Diagrama de ejecución: registros y memoria (v)

| | addi x5, x5, 1 | | | | jal x0, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S9 | S7 |
| PC | 00001010 | 00001014 | = | = | = | 00001018 | = | 00001008 |
| OldPC | 0000100c | 00001010 | = | = | = | 00001014 | = | = |
| IR | 00530333 | 00128293 | = | = | = | ff5ff06f | = | = |
| MDR | ???????? | = | = | = | = | = | = | = |
| A | 00006664 | = | 00000005 | = | = | = | basura$^{13}$ | = |
| B | 00000005 | = | basura$^{11}$ | = | = | = | basura$^{14}$ | = |
| ALUout | 00006669 | = | basura$^{12}$ | 00000006 | = | = | 00001008 | 00001018 |
| x5 | 00000005 | = | = | = | 00000006 | = | = | = |
| x6 | 00006669 | = | = | = | = | = | = | = |
| x7 | 00000006 | = | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = | = |
| clk | | | | | | | | |
| # ciclo | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |

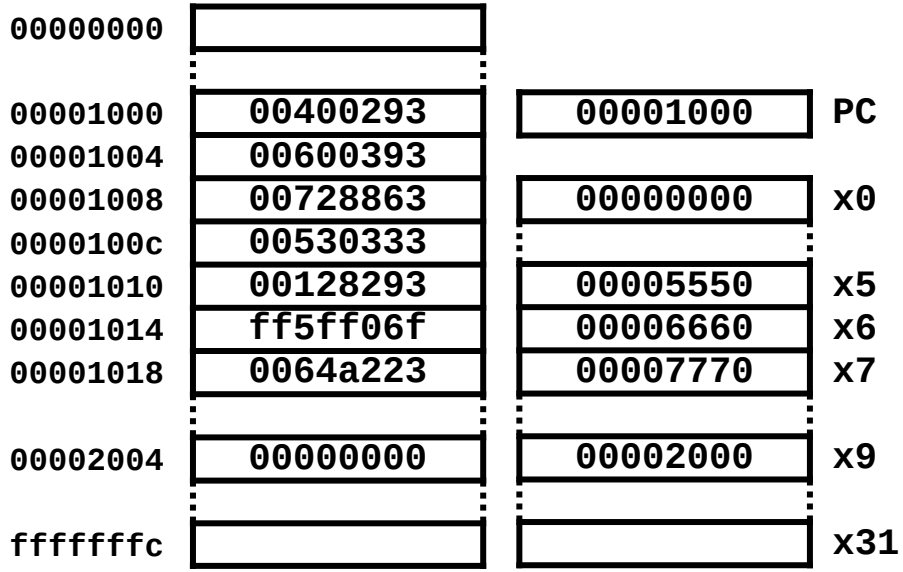
basura$^{11}$ = BR[IR$_{24..20}$] = x21 = ???

basura$^{12}$ = OldPC + SExt(imm) = 0x1010 + 1 = 0x00001011

basura$^{13}$ = BR[IR$_{19..15}$] = x31 = ???

basura$^{14}$ = BR[IR$_{24..20}$] = 0x1004 + 6 = 0x0000100a

```
0x1000    ...
            addi x5, x0, 4
0x1004      addi x7, x0, 6
          for:
0x1008      beq  x5, x7, efor
0x100c      add  x6, x6, x5
0x1010      addi x5, x5, 1
0x1014      jal  x0, for
          efor:
0x1018      sw   x6, 4(x9)
            ...
```




# Diagrama de ejecución: registros y memoria (v)

| | addi x5, x5, 1 | | | | jal x0, -12 | | | |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S8 | S7 | S0 | S1 | S9 | S7 |
| PC | 00001010 | 00001014 | 00001014 | 00001014 | 00001014 | 00001018 | 00001018 | 00001008 |
| OldPC | 0000100c | 00001010 | 00001010 | 00001010 | 00001010 | 00001014 | 00001014 | 00001014 |
| IR | 00530333 | 00128293 | 00128293 | 00128293 | 00128293 | ff5ff06f | ff5ff06f | ff5ff06f |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | 00006664 | 00006664 | 00000005 | 00000005 | 00000005 | 00000005 | $basura^{13}$ | basura |
| B | 00000005 | 00000005 | $basura^{11}$ | basura | basura | basura | $basura^{14}$ | basura |
| ALUout | 00006669 | 00006669 | $basura^{12}$ | 00000006 | 00000006 | 00000006 | 00001008 | 00001018 |
| x5 | 00000005 | 00000005 | 00000005 | 00000005 | 00000006 | 00000006 | 00000006 | 00000006 |
| x6 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 |
| x7 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |
| clk | | | | | | | | |
| # ciclo | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |

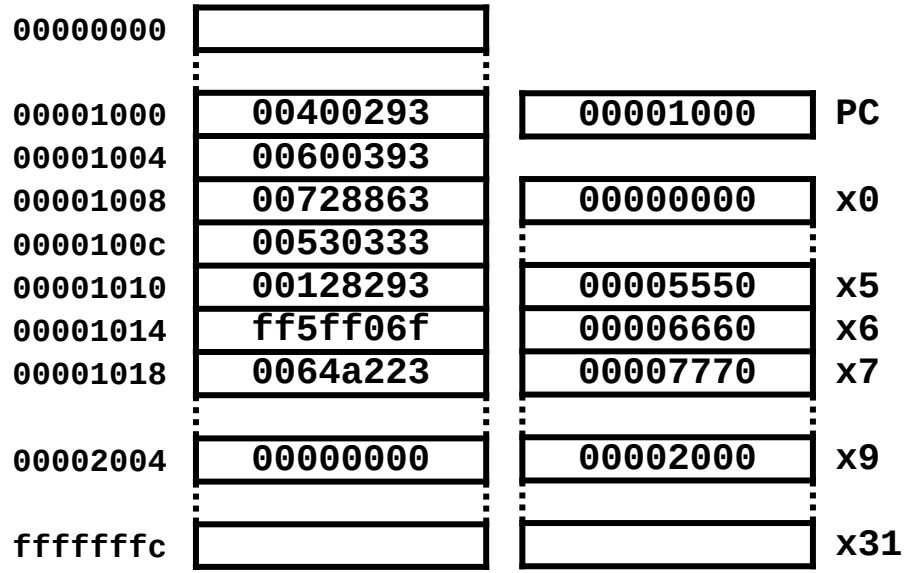
$basura^{11}$ = BR[$IR_{24..20}$] = x21 = ???

$basura^{12}$ = OldPC + SExt(imm) = 0x1010 + 1 = 0x00001011

$basura^{13}$ = BR[$IR_{19..15}$] = x31 = ???

$basura^{14}$ = BR[$IR_{24..20}$] = 0x1004 + 6 = 0x0000100a

```
...
0x1000    addi x5, x0, 4
0x1004    addi x7, x0, 6
for:
0x1008    beq  x5, x7, efor
0x100c    add  x6, x6, x5
0x1010    addi x5, x5, 1
0x1014    jal  x0, for
efor:
0x1018    sw   x6, 4(x9)
...
```

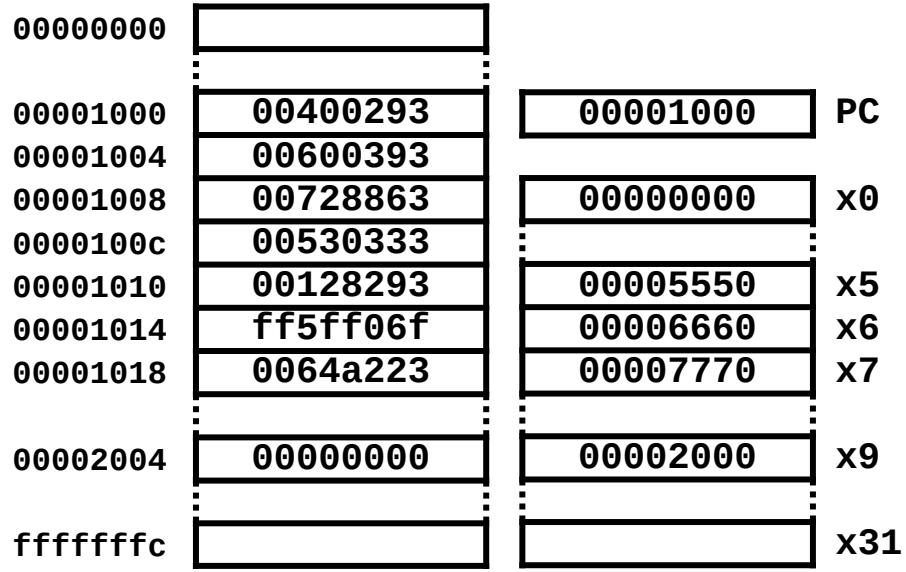



# Diagrama de ejecución: registros y memoria (vi)

| | beq x5, x7, 16 | | | sw x6, 4(x9) | | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S2 | S5 | S0 |
| PC | 00001008 | 0000100c | = | 00001018 | 0000101c | = | = | = |
| OldPC | 00001014 | 00001008 | = | = | 00001018 | = | = | = |
| IR | ff5ff06f | 00728863 | = | = | 0064a223 | = | = | = |
| MDR | ???????? | = | = | = | = | = | = | = |
| A | basura | = | 00000006 | = | = | 00002000 | = | = |
| B | basura | = | 00000006 | = | = | 00006669 | = | = |
| ALUout | 00001018 | = | 00001018 | = | = | basura[15] | 00002004 | = |
| x5 | 00000006 | = | = | = | = | = | = | = |
| x6 | 00006669 | = | = | = | = | = | = | = |
| x7 | 00000006 | = | = | = | = | = | = | = |
| x9 | 00002000 | = | = | = | = | = | = | = |
| MEM[0x2004] | 00000000 | = | = | = | = | = | = | 00006669 |
| clk | | | | | | | | |
| # ciclo | 39 | 40 | 41 | 42 | 43 | 44 | 45 | |

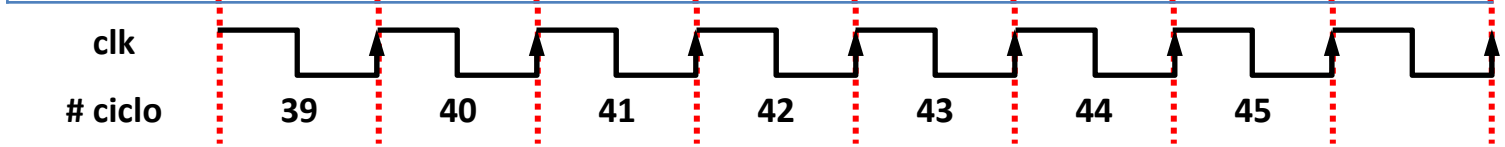
$basura^{15}$ = OldPC + SExt(imm) =
0x1018 + 4 = 0x0000101c

```
...
0x1000    addi x5, x0, 4
0x1004    addi x7, x0, 6
for:
0x1008    beq  x5, x7, efor
0x100c    add  x6, x6, x5
0x1010    addi x5, x5, 1
0x1014    jal  x0, for
efor:
0x1018    sw   x6, 4(x9)
...
```




# Diagrama de ejecución: registros y memoria (vi)

| | beq x5, x7, 16 | | | sw x6, 4(x9) | | | | ... |
|---|---|---|---|---|---|---|---|---|
| estado | S0 | S1 | S10 | S0 | S1 | S2 | S5 | S0 |
| PC | 00001008 | 0000100c | 0000100c | 00001018 | 0000101c | 0000101c | 0000101c | 0000101c |
| OldPC | 00001014 | 00001008 | 00001008 | 00001008 | 00001018 | 00001018 | 00001018 | 00001018 |
| IR | ff5ff06f | 00728863 | 00728863 | 00728863 | 0064a223 | 0064a223 | 0064a223 | 0064a223 |
| MDR | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? | ???????? |
| A | basura | basura | 00000006 | 00000006 | 00000006 | 00002000 | 00002000 | 00002000 |
| B | basura | basura | 00000006 | 00000006 | 00000006 | 00006669 | 00006669 | 00006669 |
| ALUout | 00001018 | 00001018 | 00001018 | 00001018 | 00001018 | basura[15] | 00002004 | 00002004 |
| x5 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x6 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 | 00006669 |
| x7 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 | 00000006 |
| x9 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 | 00002000 |
| MEM[0x2004] | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00006669 |




$basura^{15}$ = OldPC + SExt(imm) =
0x1018 + 4 = 0x0000101c

**24)** (Adaptado de examen Septiembre 2016) El fragmento de programa mostrado debajo se ejecuta en un procesador RISC-V **multiciclo** con una frecuencia de reloj de 1 GHz, resultando un tiempo de ejecución de 3474 ns.

```
        mv s4, zero
        la s5, A
        mv s6, zero
        li s4, 127
L1:     slli s3, s4, 2
        add s3,s3,s5
        lw s0, 0(s3)
        add s6, s6, s0
        add s4, s4, -1
        bge s4,zero,L1
        L2:
        ... (resto del programa)
```

a) Hallar el valor del CPI para este fragmento de programa.
b) Hallar el valor de la métrica MIPS obtenido.

Datos: f = 1 GHz = $10^9$ ciclos/s  T = 3474 ns = 3474x$10^{-9}$ s

Observación: el bucle tiene 6 instrucciones, y se ejecuta para s4 = 127..0 (i.e. 128 veces)

a) T = NI x CPI / f → CPI = T x f / NI
Del código: NI = 4 + 6x128 = 772 instrucciones. Así, **CPI =** (3474x$10^{-9}$ s x $10^9$ ciclos/s) / 772 instr. = **4.5**

b) **MIPS =** f / (CPIx$10^6$ ) = $10^9$ / (4.5x$10^6$ ) = **222.2**

# Acerca de *Creative Commons*

- Licencia CC (Creative Commons)


  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/