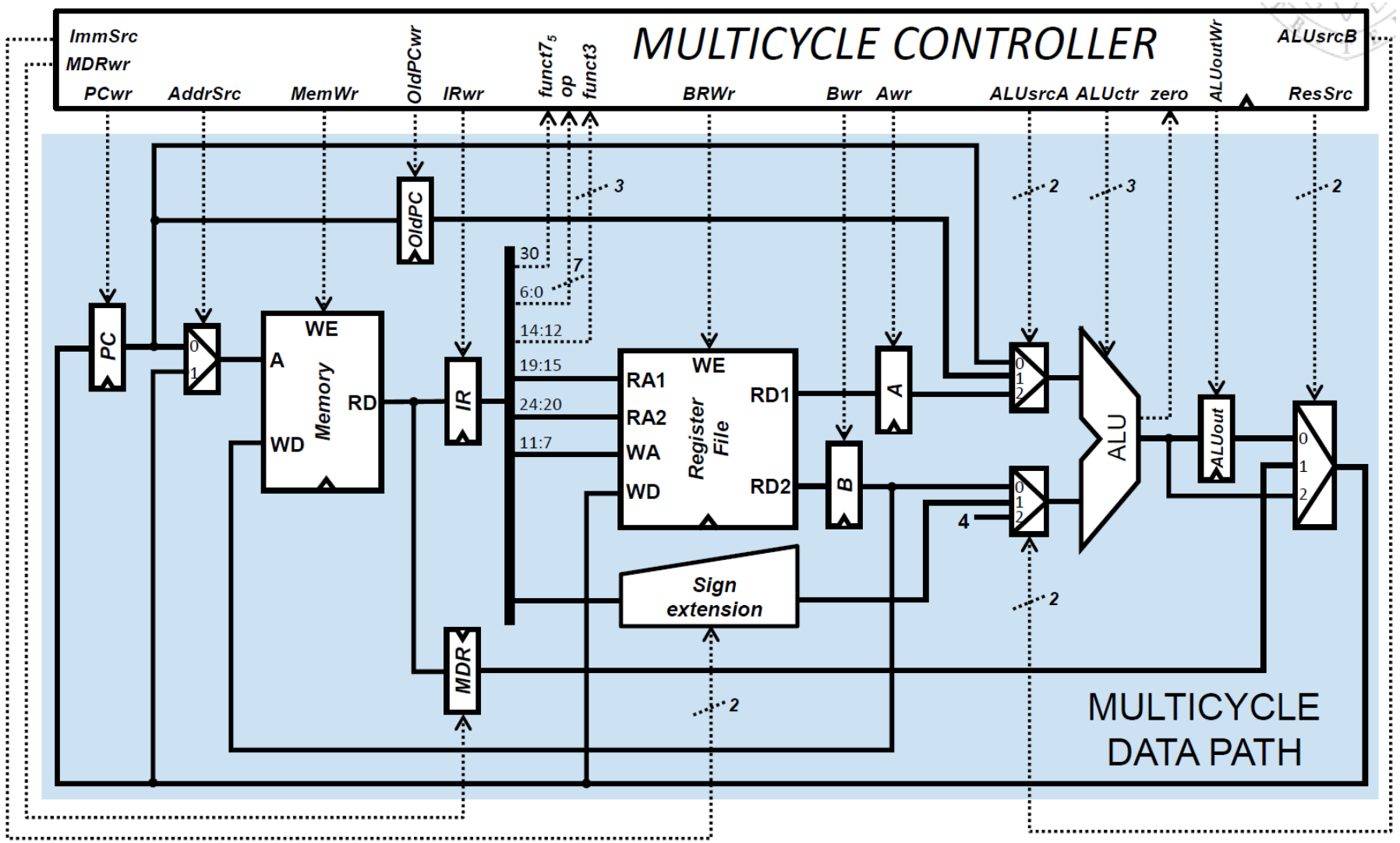
Module 6 - Problems:

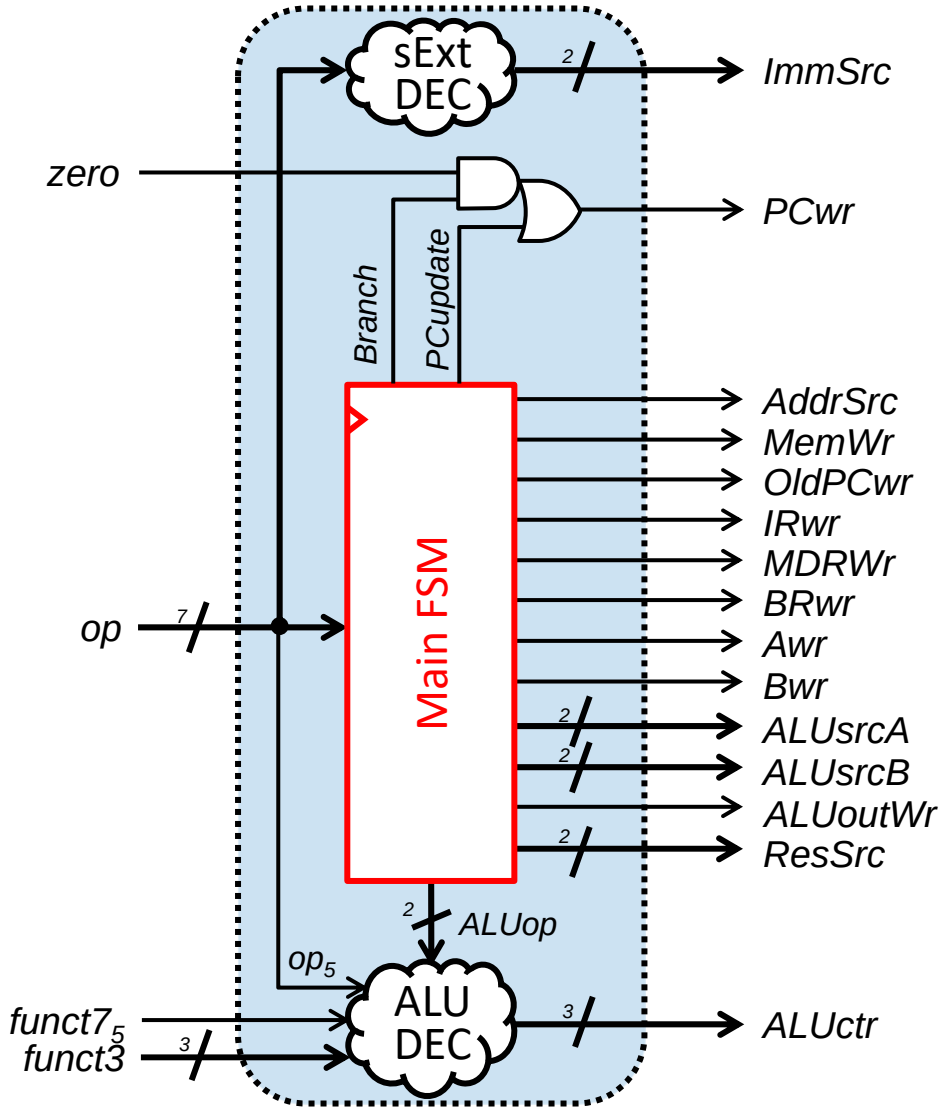
Multicycle processor design

Introduction to Computers II

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

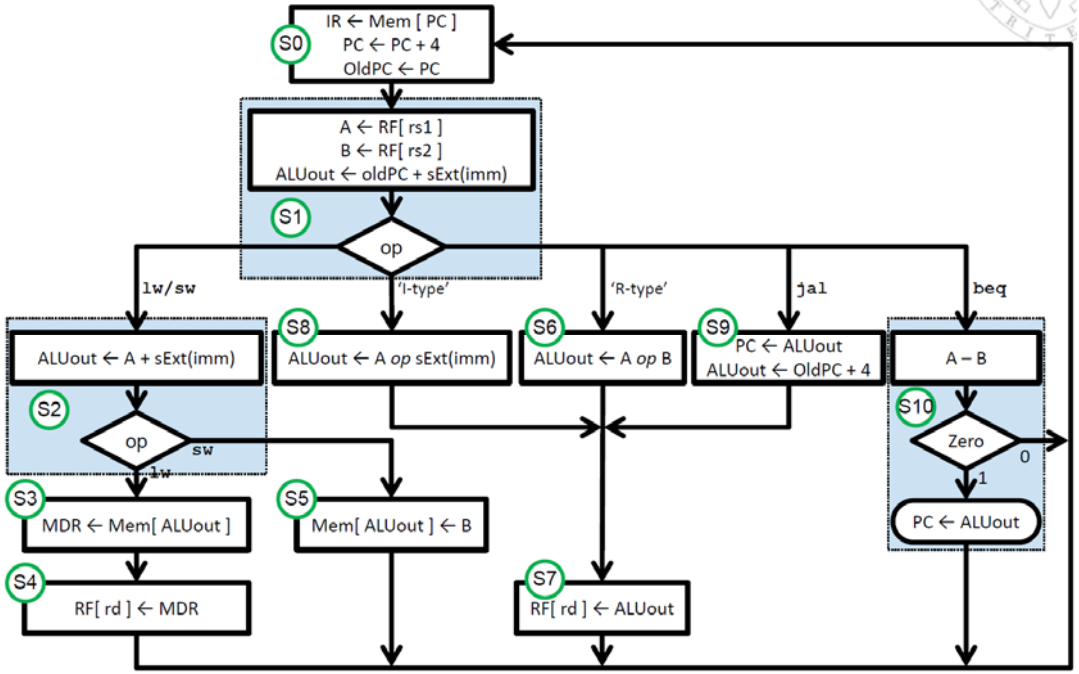
1) Provide the value of the control signals produced in a multicycle RISC-V when executing a **`SW`** instruction.



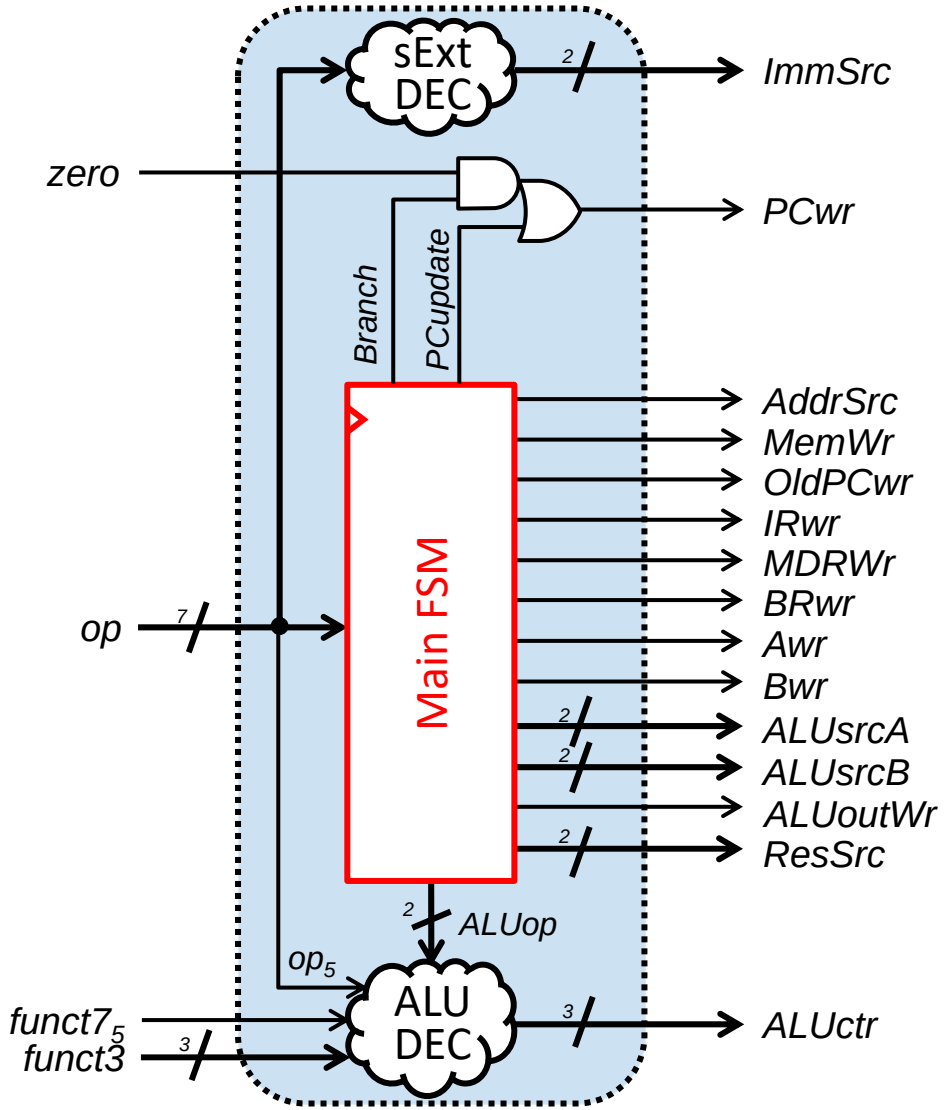


Output function

state	Branch	PCupdate	AddrSrc	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

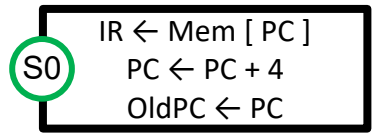


"sw" instructions follow the state sequence S0-S1-S2-S5

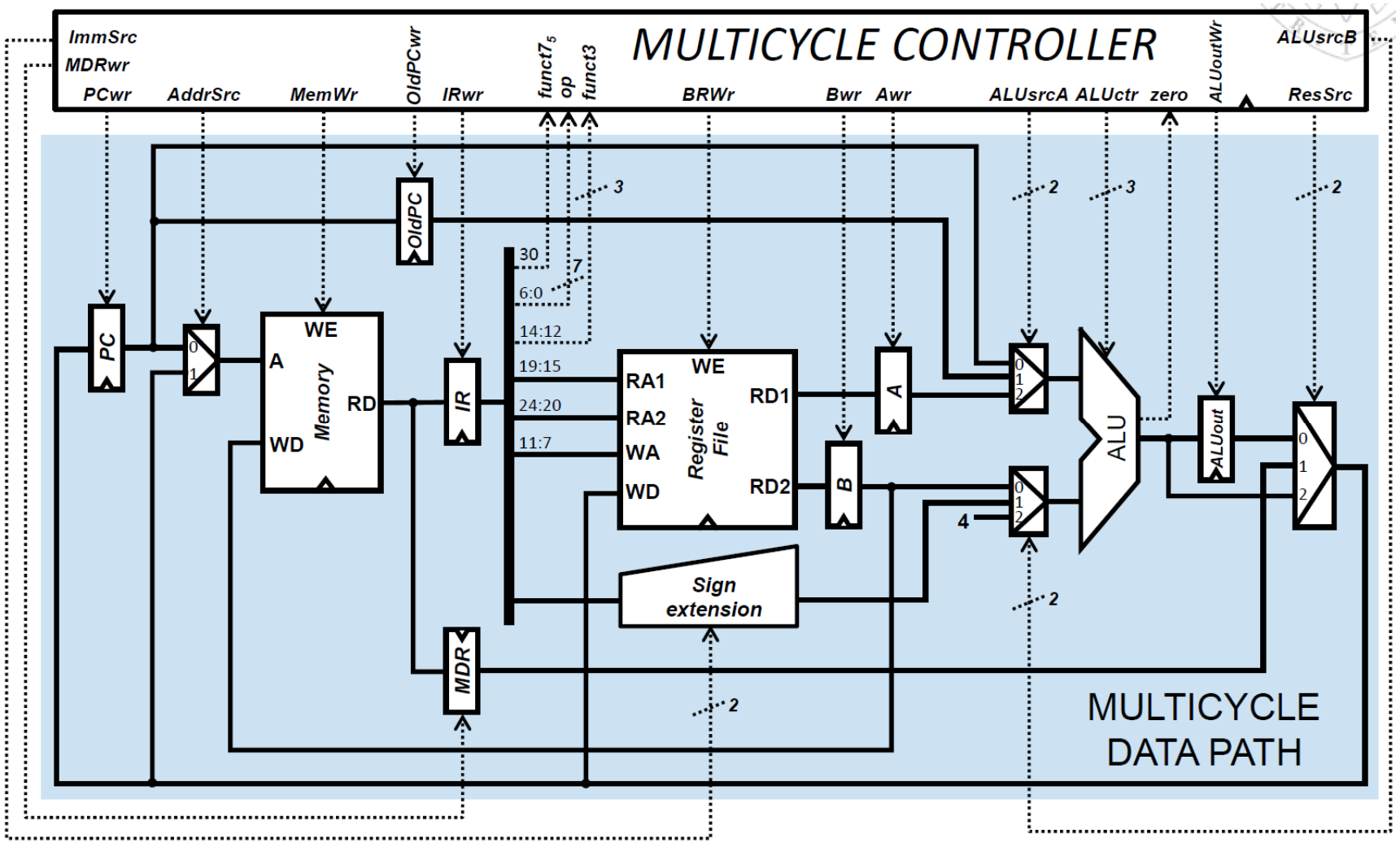


Output function

state	Branch	PCupdate	AddrSrc	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

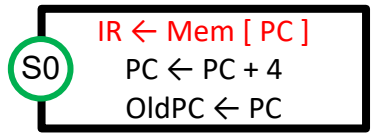


- The Branch signal is 0 because this is not a beq instruction
- The PCupdate signal is 1 because the PC is updated with PC+4



Output function

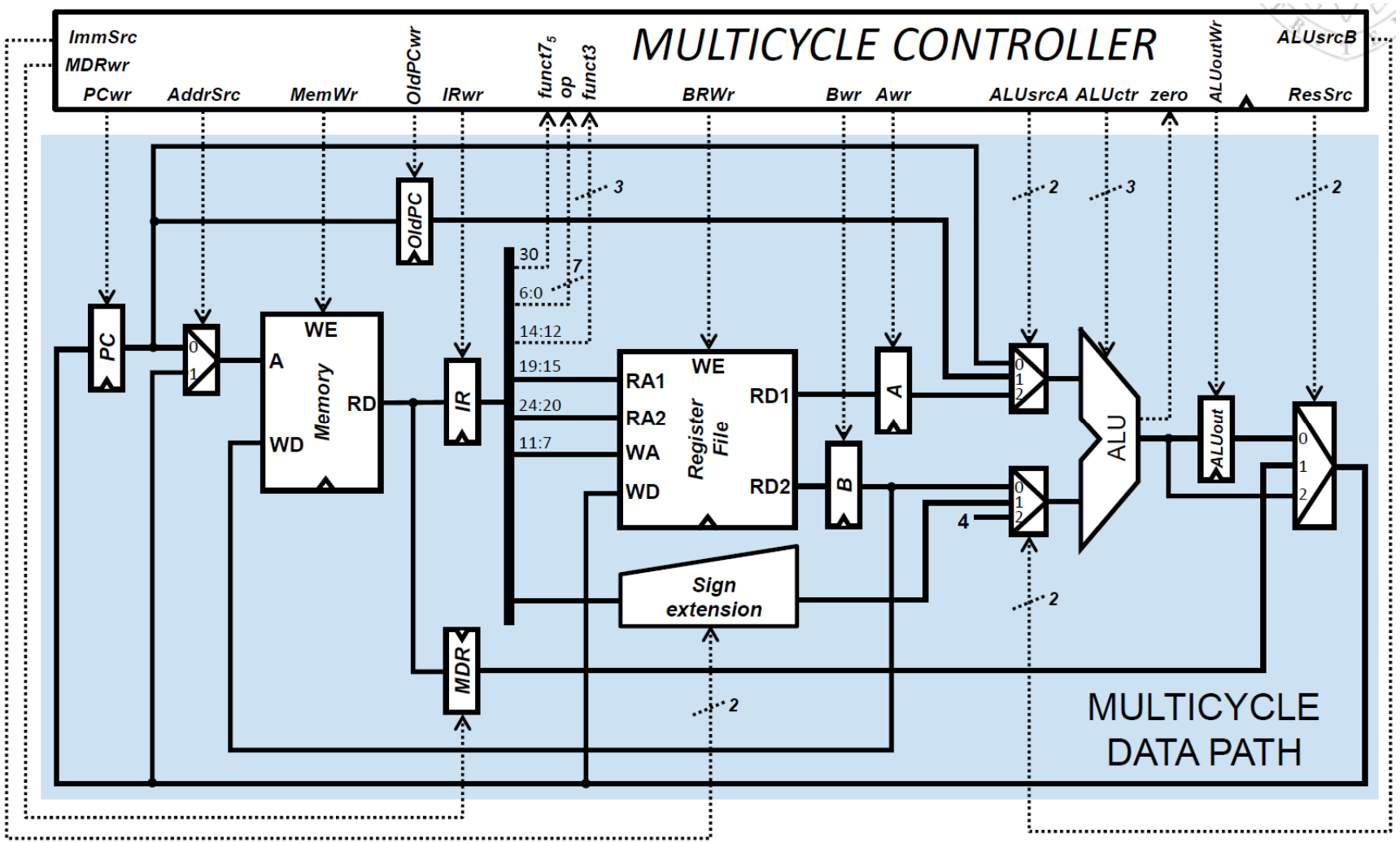
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 is 0 because the memory address to read is the one provided by the PC

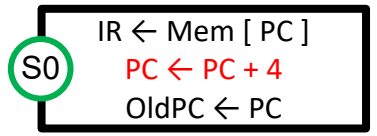
- IRwr is 1 because the instruction fetched from memory is written in the IR

- MemWr is 0 because in this cycle the memory is read, but it is not written

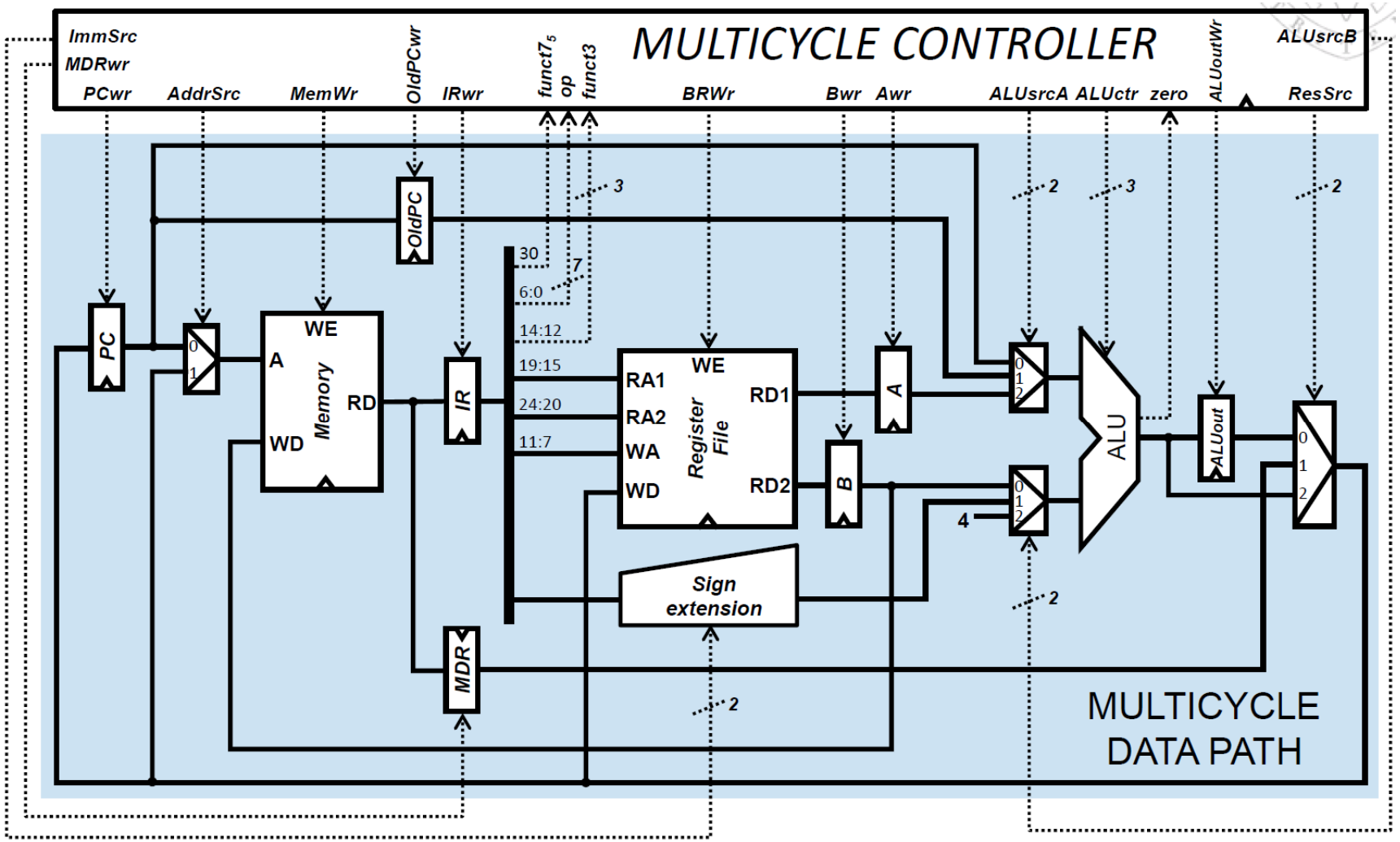


Output function

state	Branch	PCupdate	AddrSrc	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

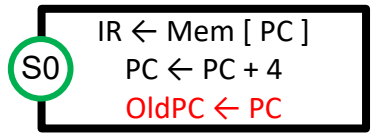


- The ALU performs PC+4 (ALUop=00). The PC value arrives at the ALU upper input through channel 0 of the corresponding MUX, thus ALUsrcA is 00. The immediate 4 arrives at the ALU lower input through channel 2 of the corresponding MUX, thus ALUsrcB is 10. The result is not written in the ALUout register (thus ALUoutWr is 0), but it is directly routed to the PC, in order to update its value, through channel 2 of the MUX controlled by ResSrc (10).

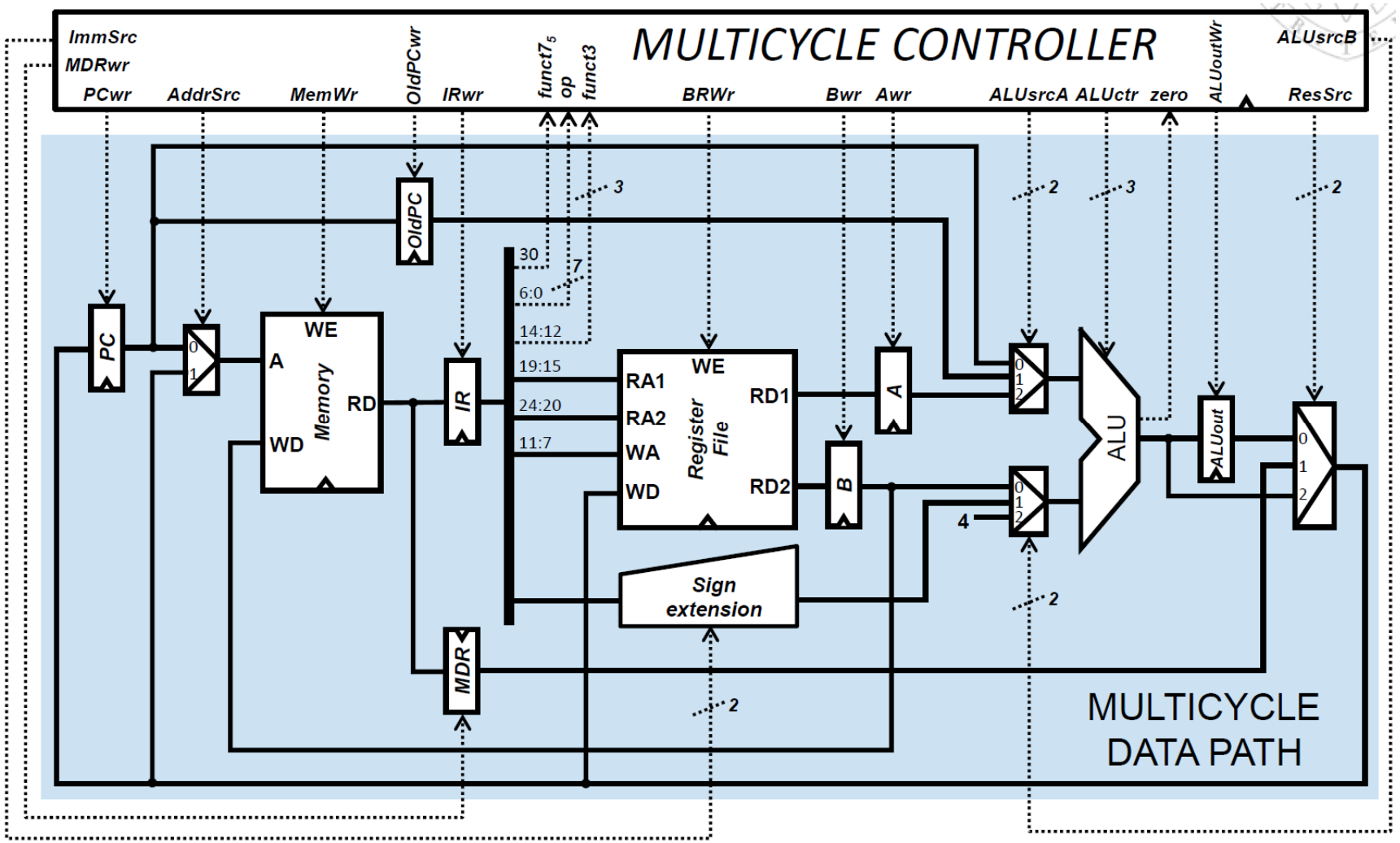


Output function

state	Branch	PCupdate	AddrSrc	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

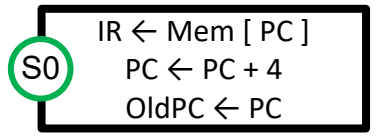


- OldPCwr is 1 because we write the current PC value in the OldPC register.



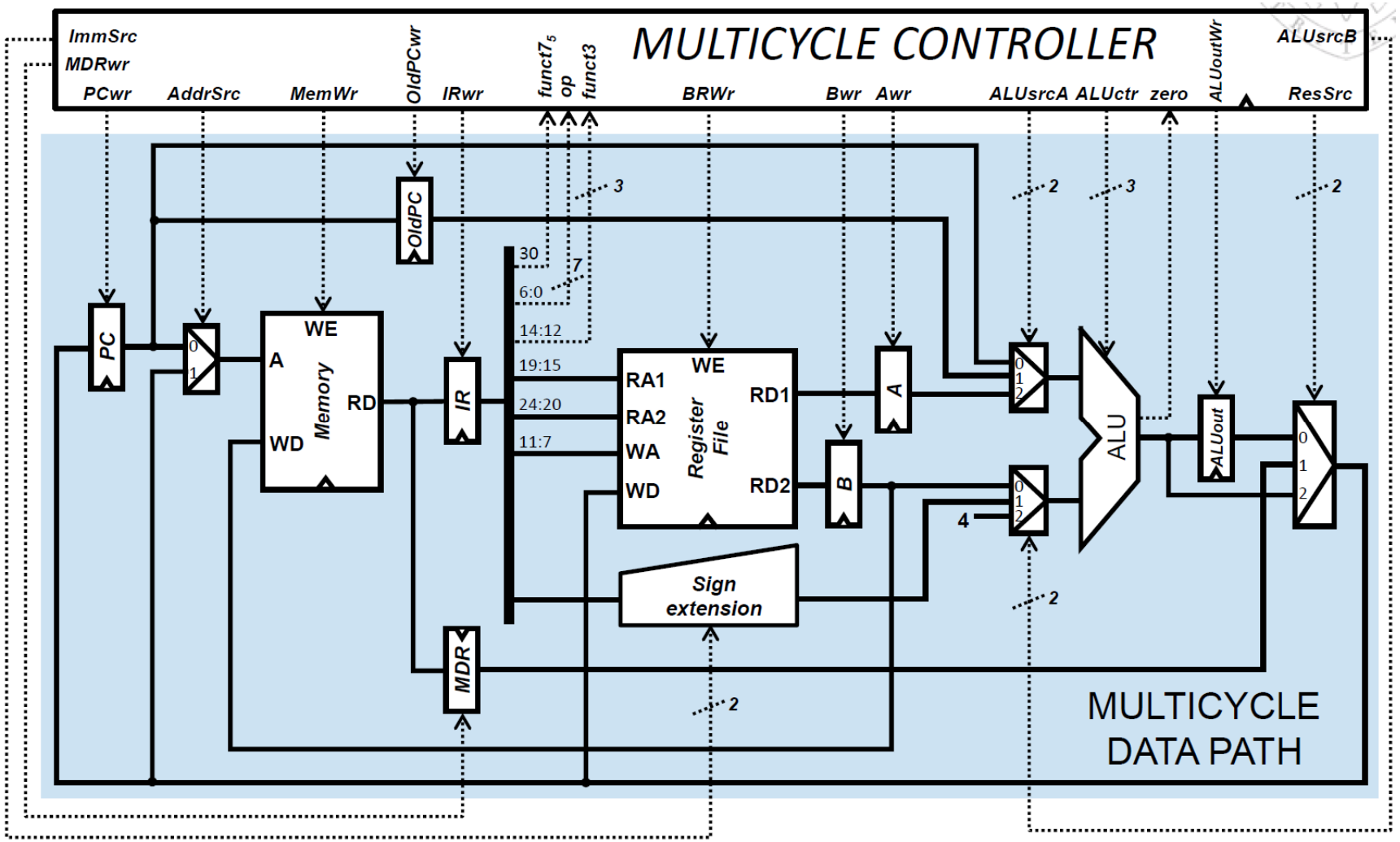
Output function

state	Branch	PCupdate	AddrSrc	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



Other signals:

- Signals MDRwr, BRwr, Awr y Bwr are 0 because the MDR register, the register file and the A and B registers are not written.



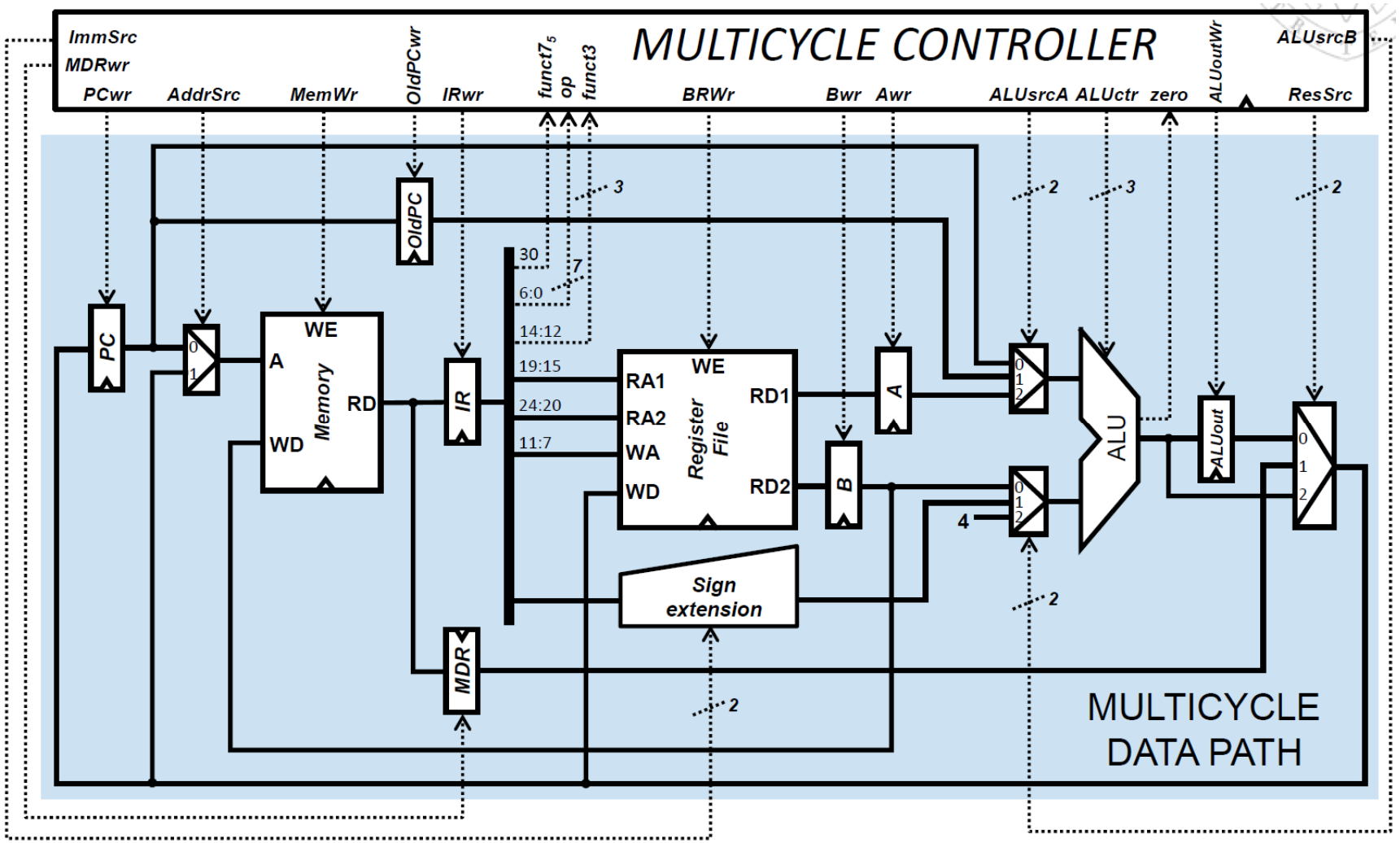
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 1 is written into the A register, thus Awr=1



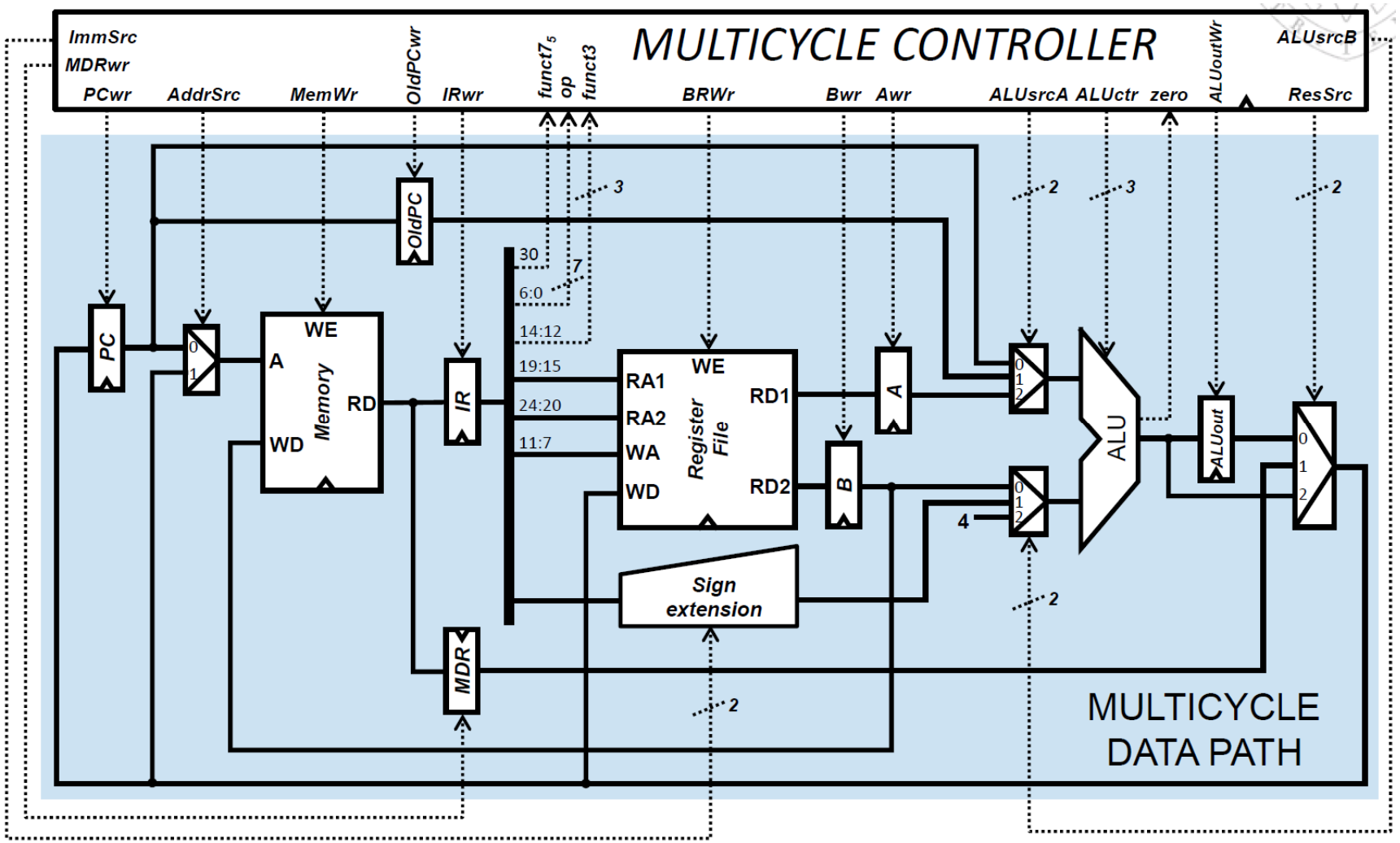
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 2 is written into the B register, thus Bwr=1



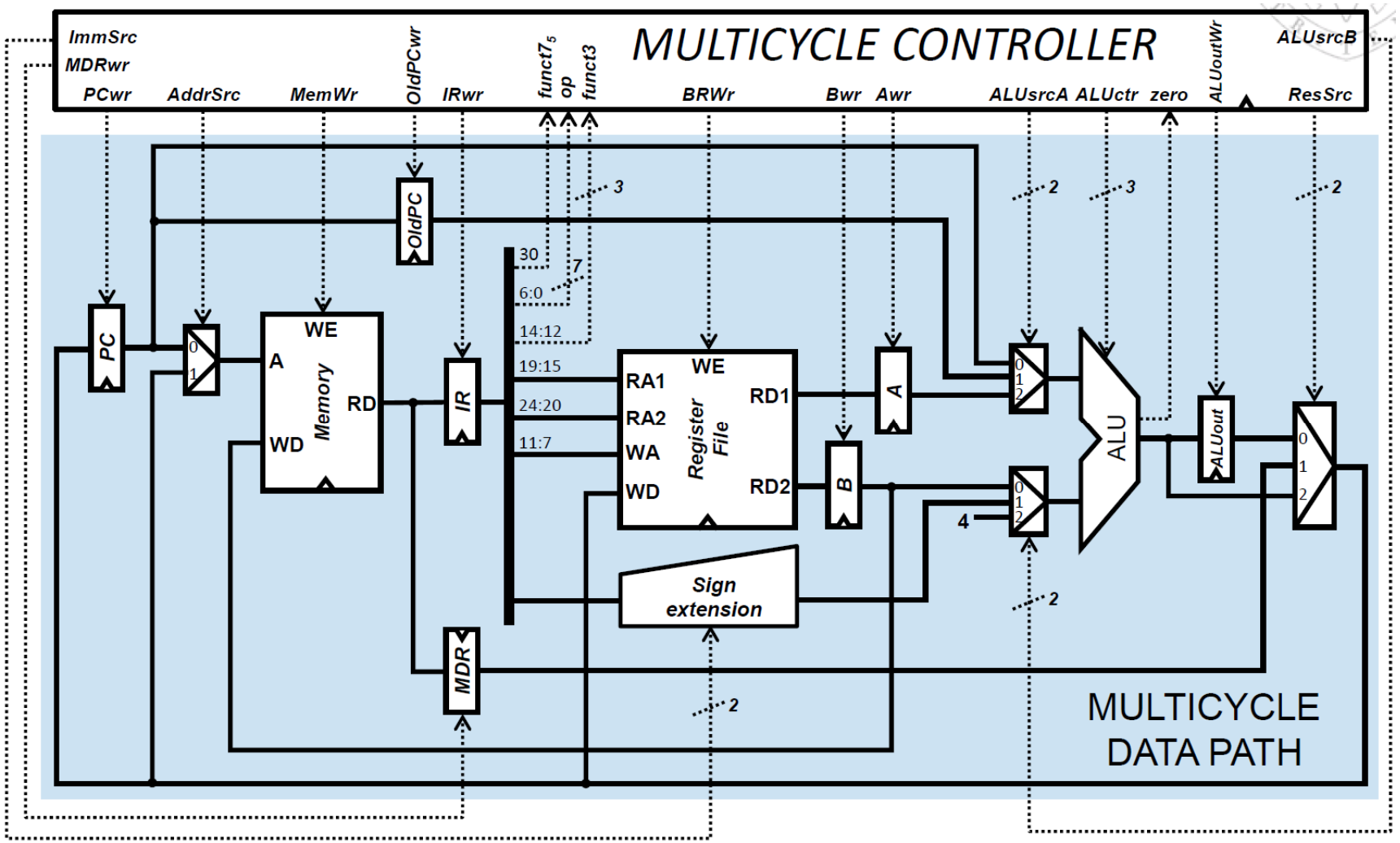
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The ALU performs the addition (ALUop=00) of OldPC + sExt(imm). The OldPC value arrives at the ALU upper input through channel 1 of the corresponding MUX, thus ALUsrcA is 01. The signed extended immediate arrives at the ALU lower input through channel 1 of the corresponding MUX, thus ALUsrcB is 01. The result of the addition is written in ALUout, thus ALUoutWr is 1.



Output function

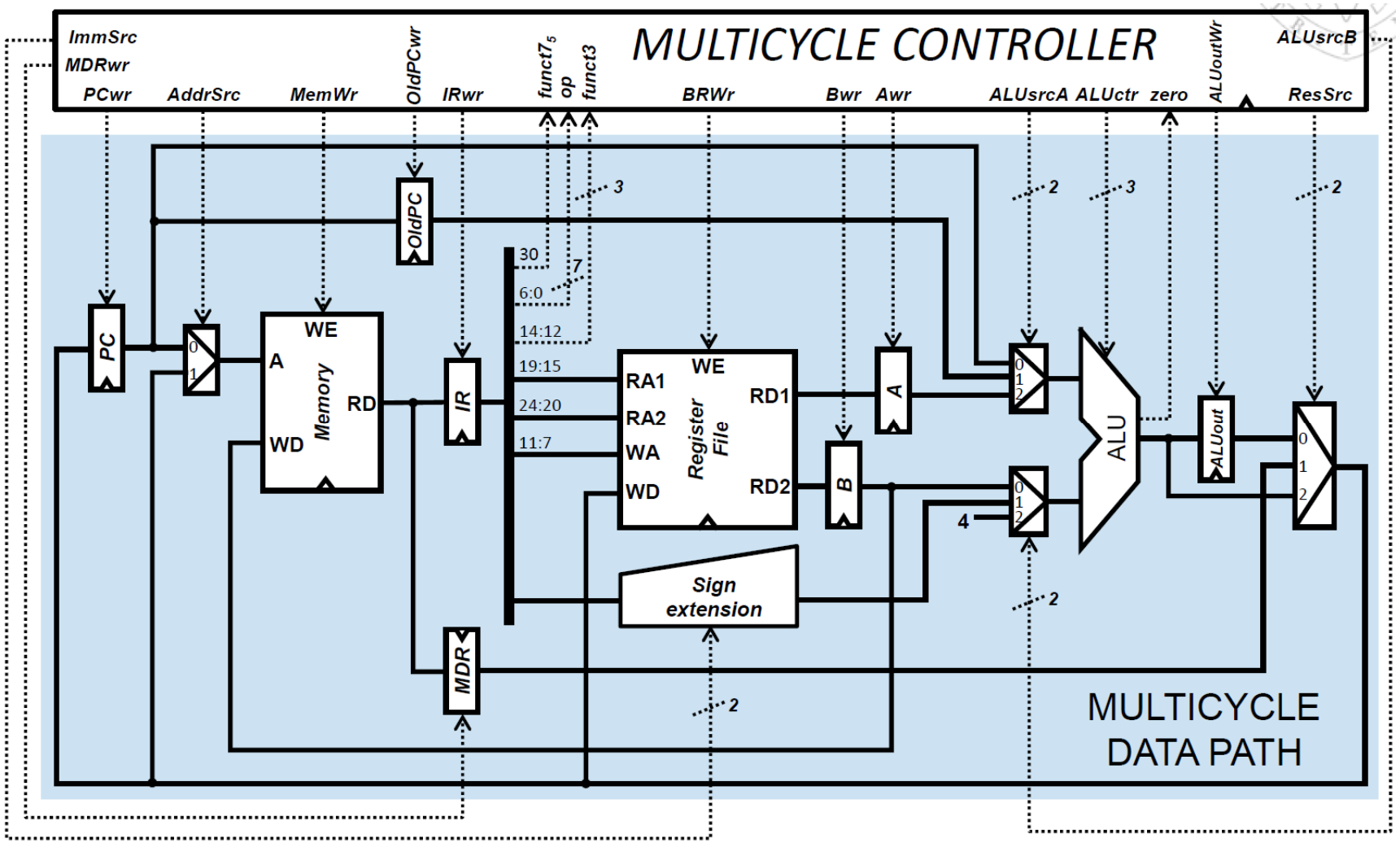
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

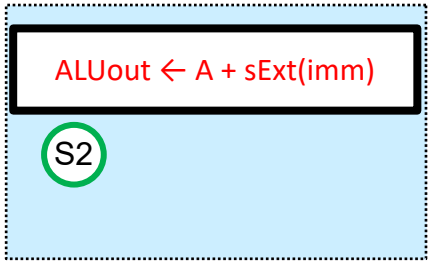
Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr and BRwr are 0 because the OldPC register, and the register file are not written in this cycle.
- ResSrc is a “don’t care” because the register file and the PC are not written in this cycle.

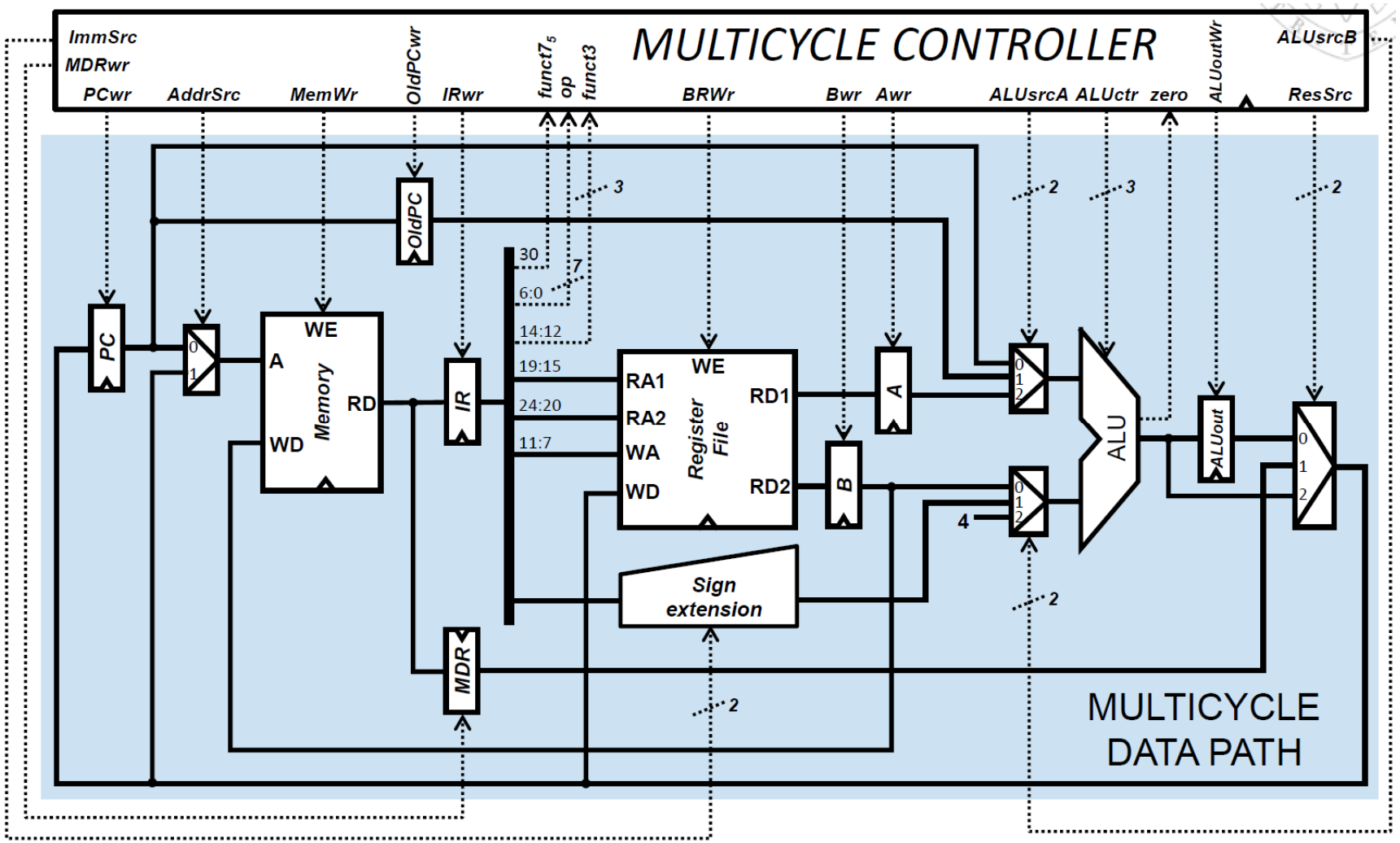


Output function

state	Branch	PCupdate	AddrSrc	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

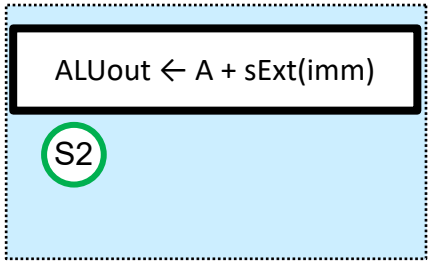


- The ALU performs the addition (ALUop=00) of A + sExt(imm). The A value arrives at the ALU upper input through channel 2 of the corresponding MUX, thus ALUsrcA is 10. The signed extended immediate arrives at the ALU lower input through channel 1 of the corresponding MUX, thus ALUsrcB is 01. The result of the addition is written in ALUout, thus ALUoutWr is 1.



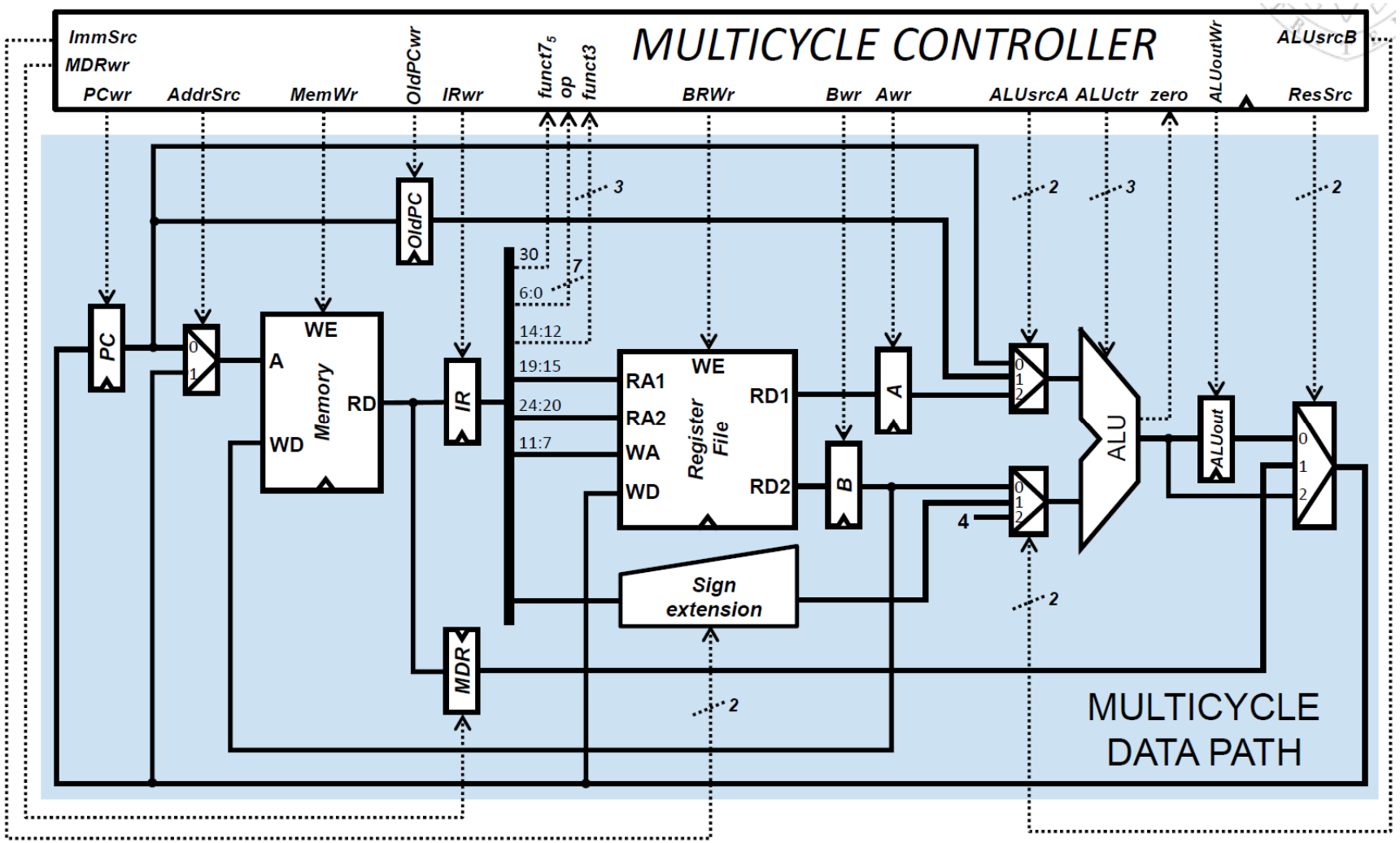
Output function

state	Branch	PCupdate	AddrSrc	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



Other signals:

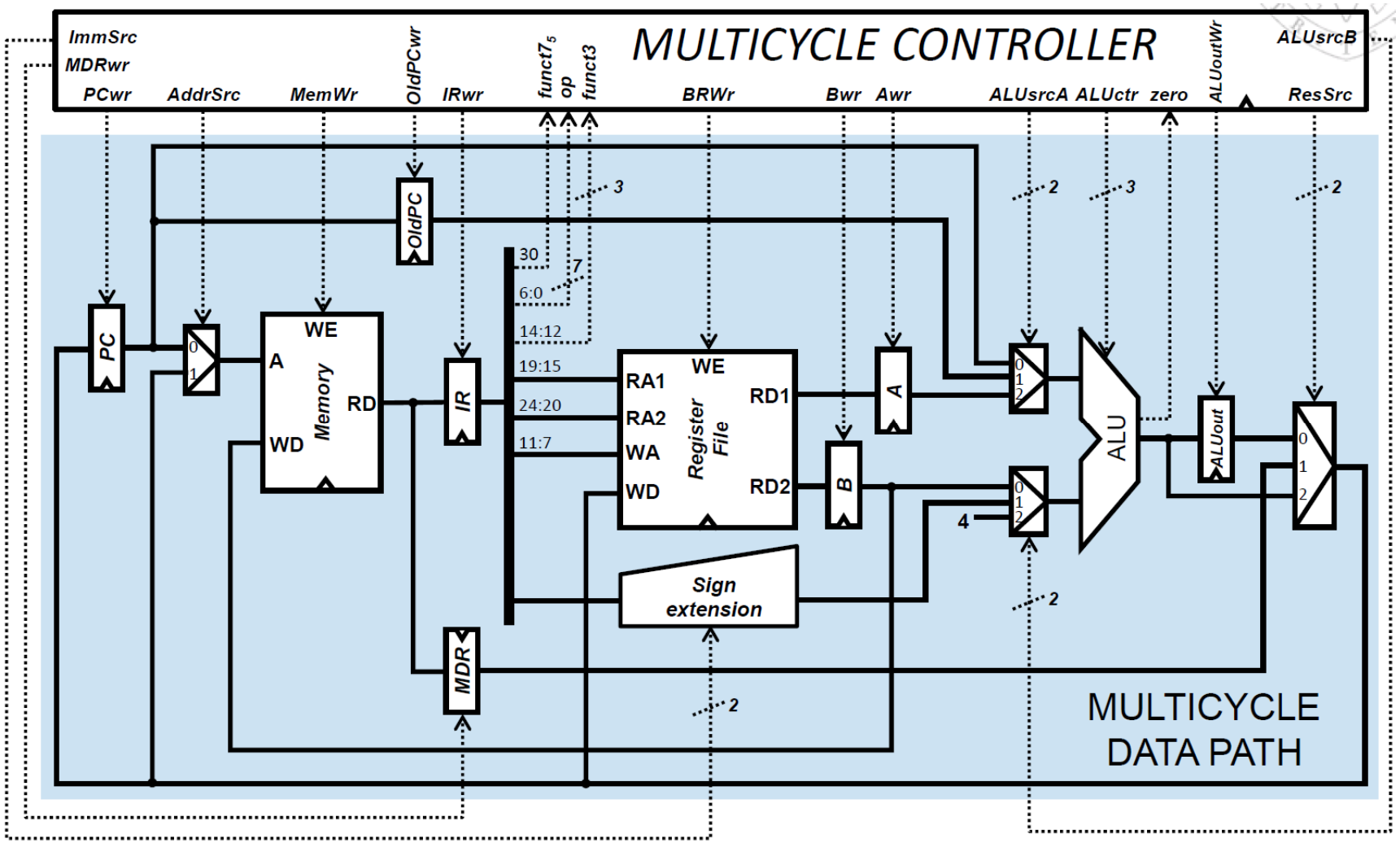
- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr, BRwr, Awr and Bwr are 0 because the OldPC register, the register file and the A and B registers are not written in this cycle.
- ResSrc is a “don’t care” because the register file and the PC are not written in this cycle.



Output function

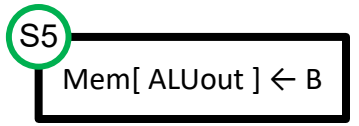
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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

- The B value has to be written in the memory, thus MemWr=1. The memory address for the write operation is the one contained in ALUout, and therefore the ResSrc signal is 00 in order to route this value to the memory input MUX. The value will arrive at the memory through channel 1 of that MUX, thus AddrSrc=1



Output function

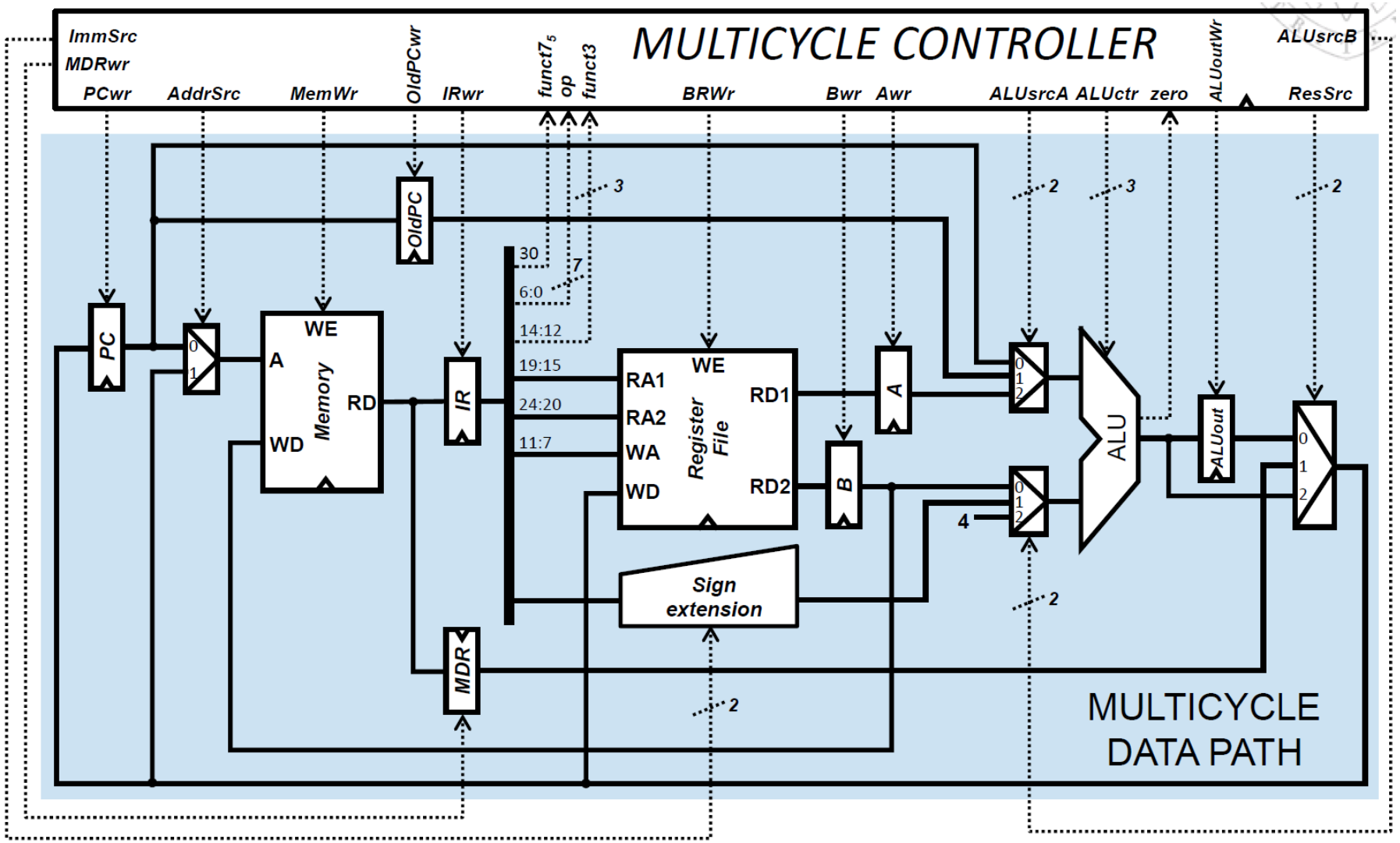
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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

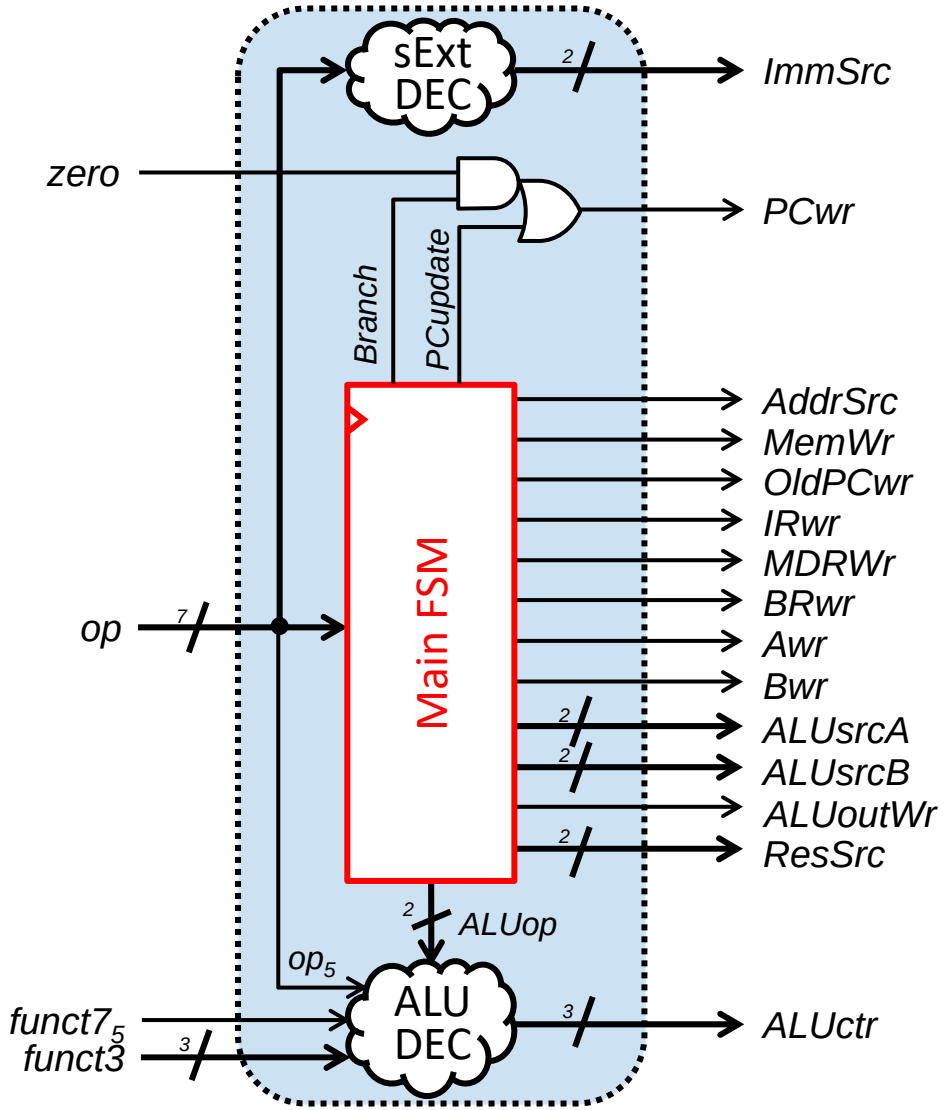


Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- OldPCwr, IRwr, MDRwr, BRwr, Awr and Bwr are 0 because the OldPC register, the IR register, the MDR register, the register file and the A and B registers are not written in this cycle.
- The ALU is not used in this cycle (nothing is written in its output register, thus ALUoutWr=0), and therefore the ALU input MUX control signals (ALUsrcA y ALUsrcB) are “don’t care”. For the same reason, the ALUop value is also a “don’t care”.

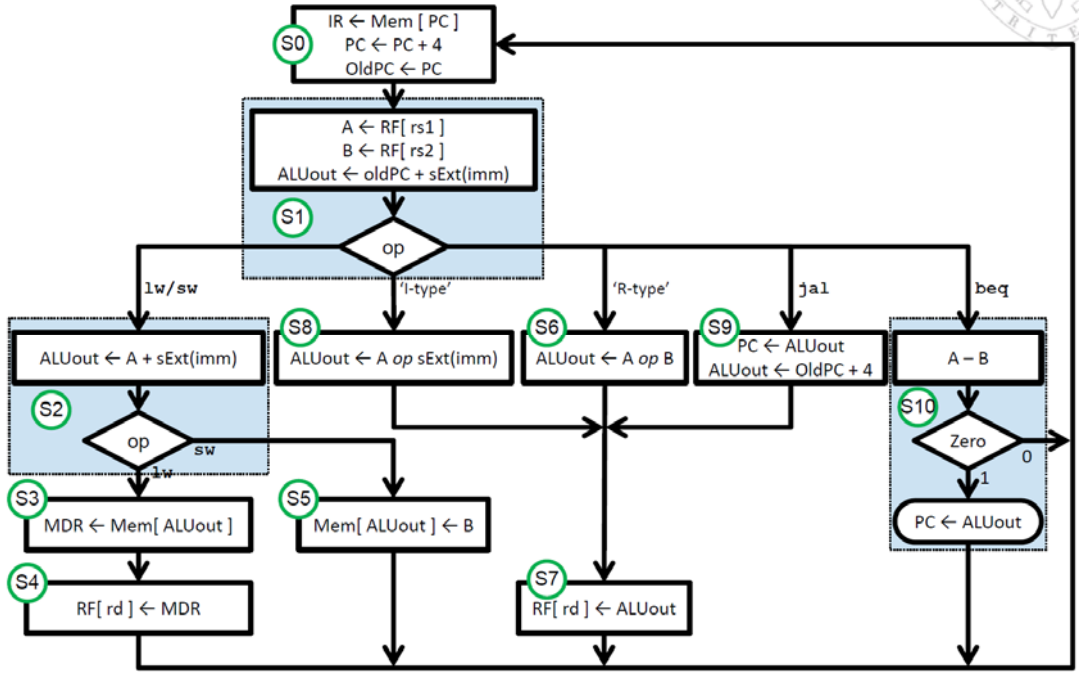
2) Provide the value of the control signals produced in a multicycle RISC-V when executing an **add** instruction.



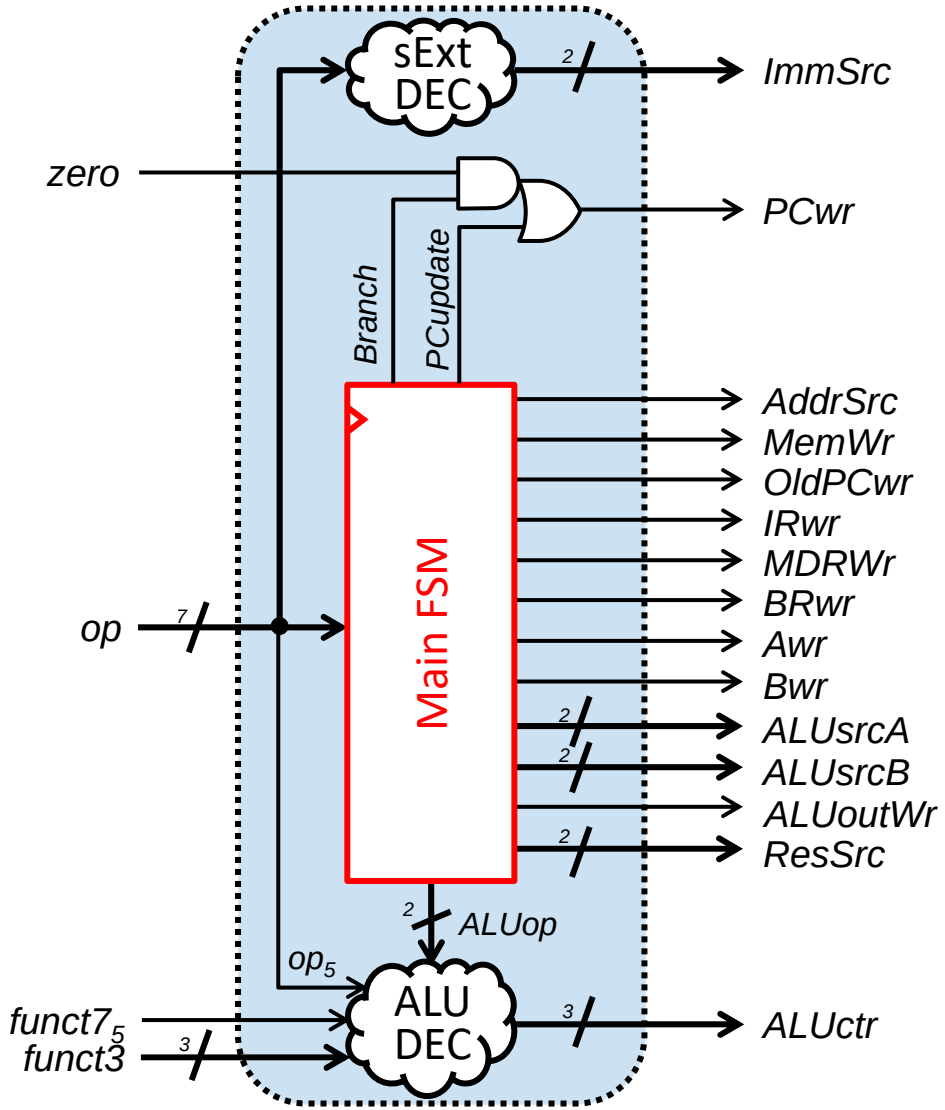


Función de salida

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

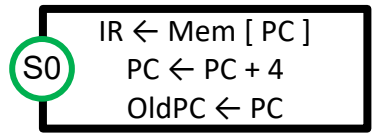


"add" instructions follow the state sequence S0-S1-S6-S7

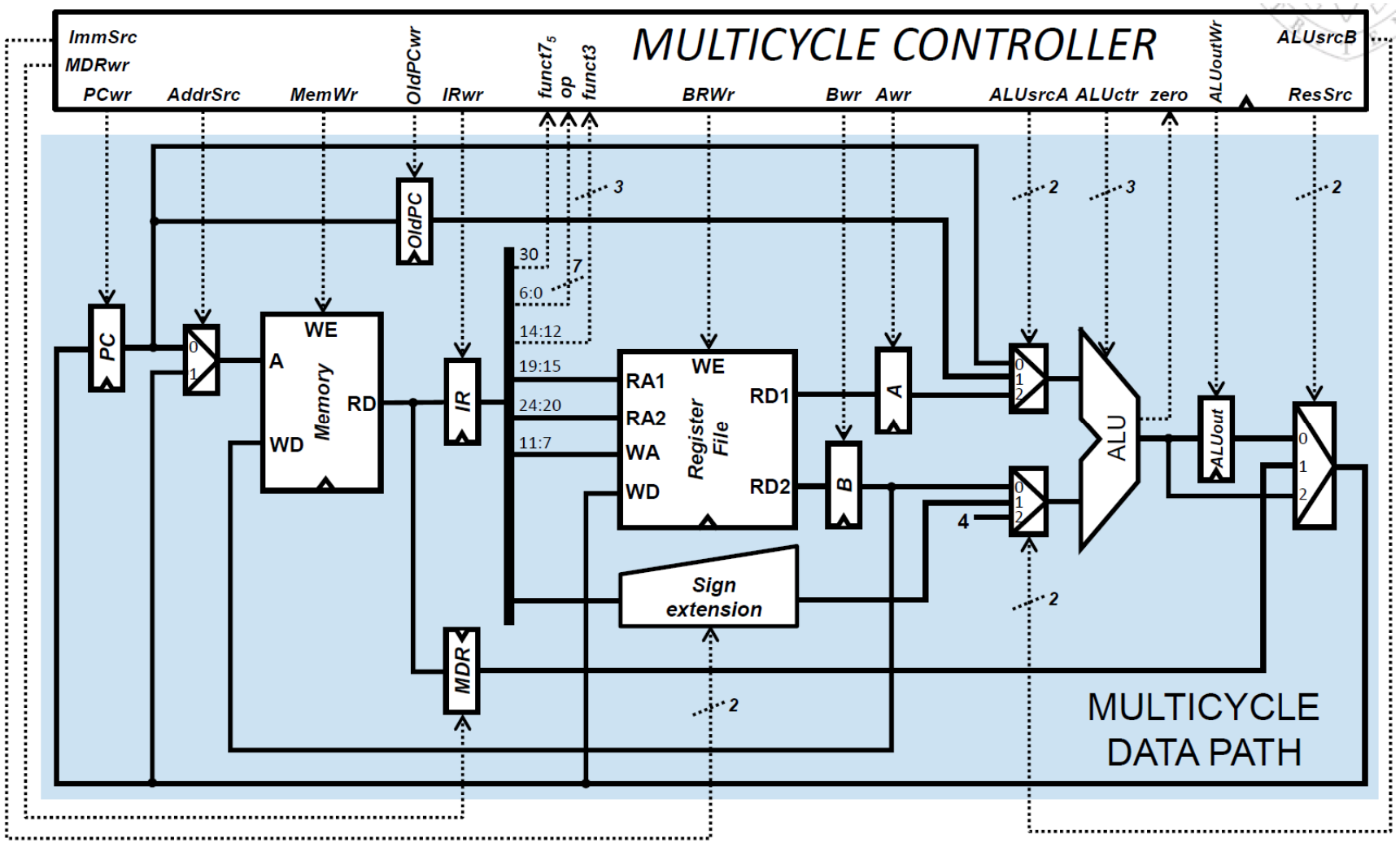


Función de salida

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

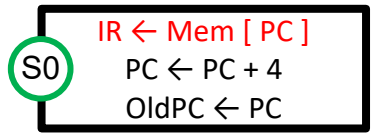


- The Branch signal is 0 because this is not a beq instruction
- The PCupdate signal is 1 because the PC is updated with PC+4



Output function

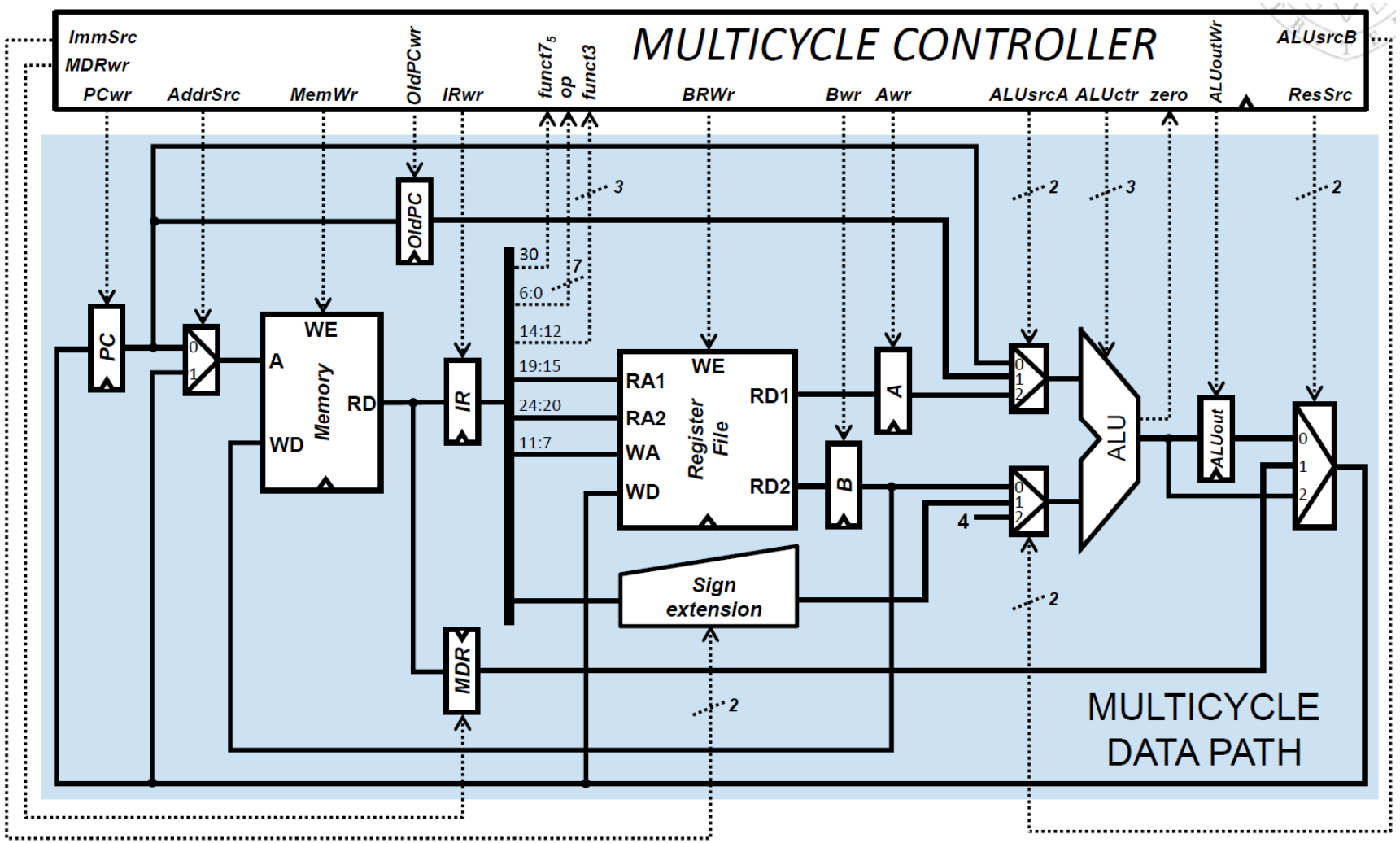
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 is 0 because the memory address to read is the one provided by the PC

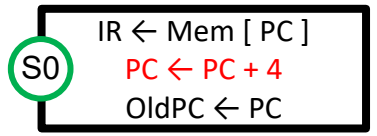
- IRwr is 1 because the instruction fetched from memory is written in the IR

- MemWr is 0 because in this cycle the memory is read, but it is not written

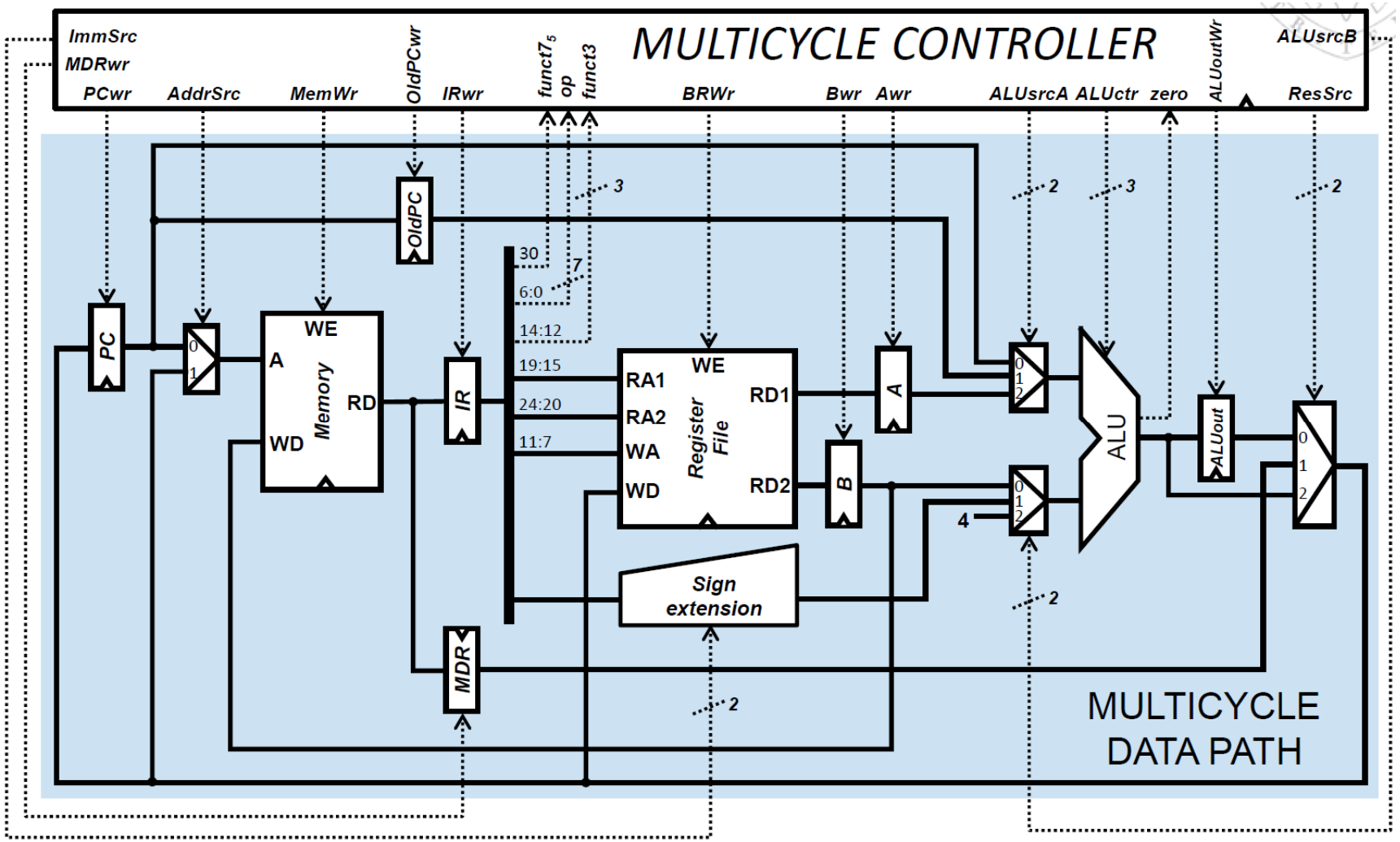


Output function

state	Branch	PCupdate	AddrSrc	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

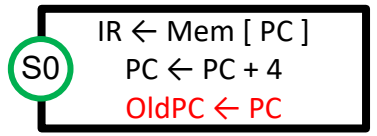


- The ALU performs PC+4 (ALUop=00). The PC value arrives at the ALU upper input through channel 0 of the corresponding MUX, thus ALUsrcA is 00. The immediate 4 arrives at the ALU lower input through channel 2 of the corresponding MUX, thus ALUsrcB is 10. The result is not written in the ALUout register (thus ALUoutWr is 0), but it is directly routed to the PC, in order to update its value, through channel 2 of the MUX controlled by ResSrc (10).

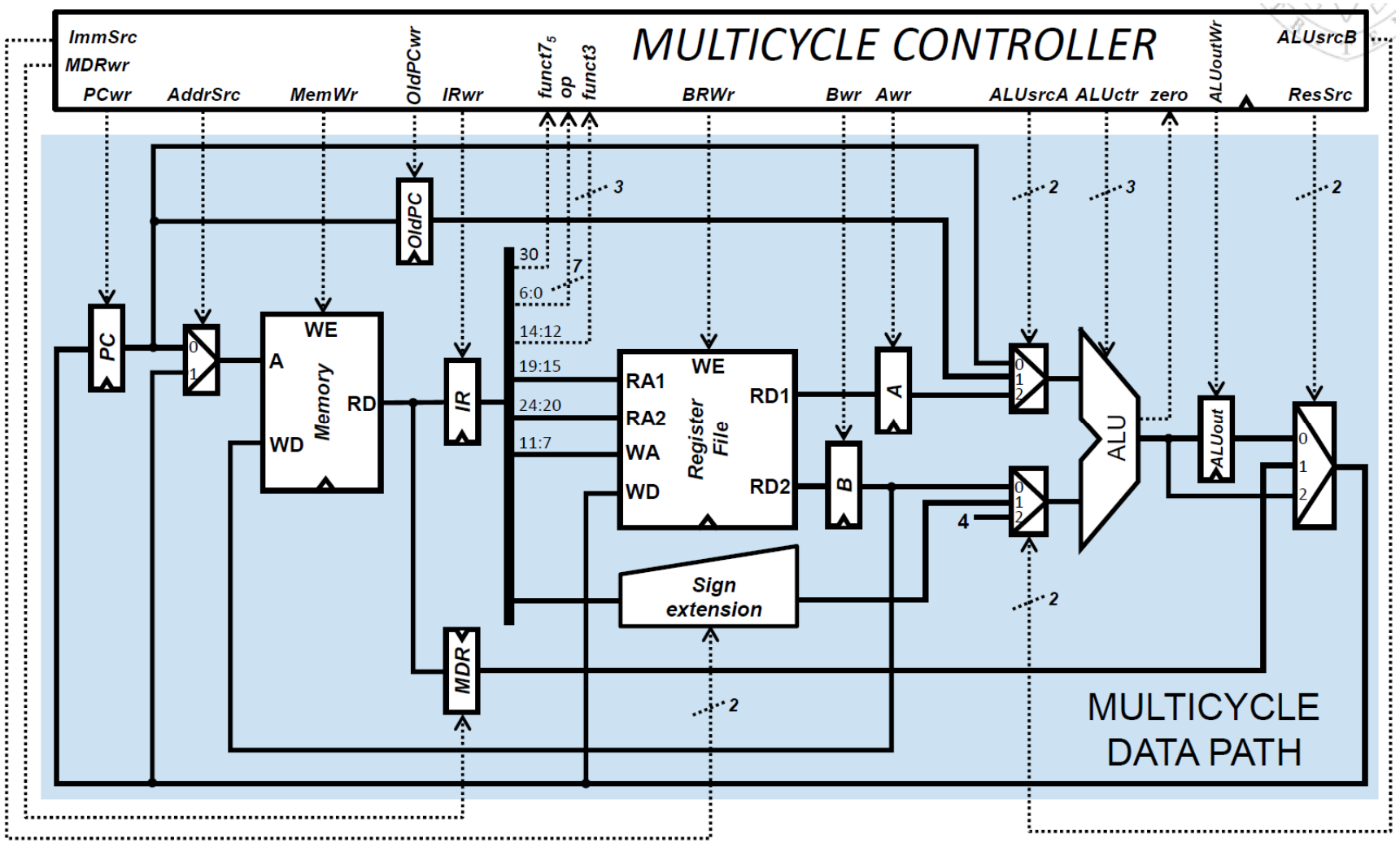


Output function

state	Branch	PCupdate	AddrSrc	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

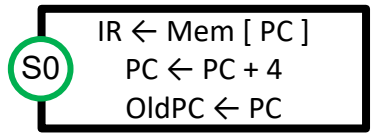


- OldPCwr is 1 because we write the current PC value in the OldPC register.



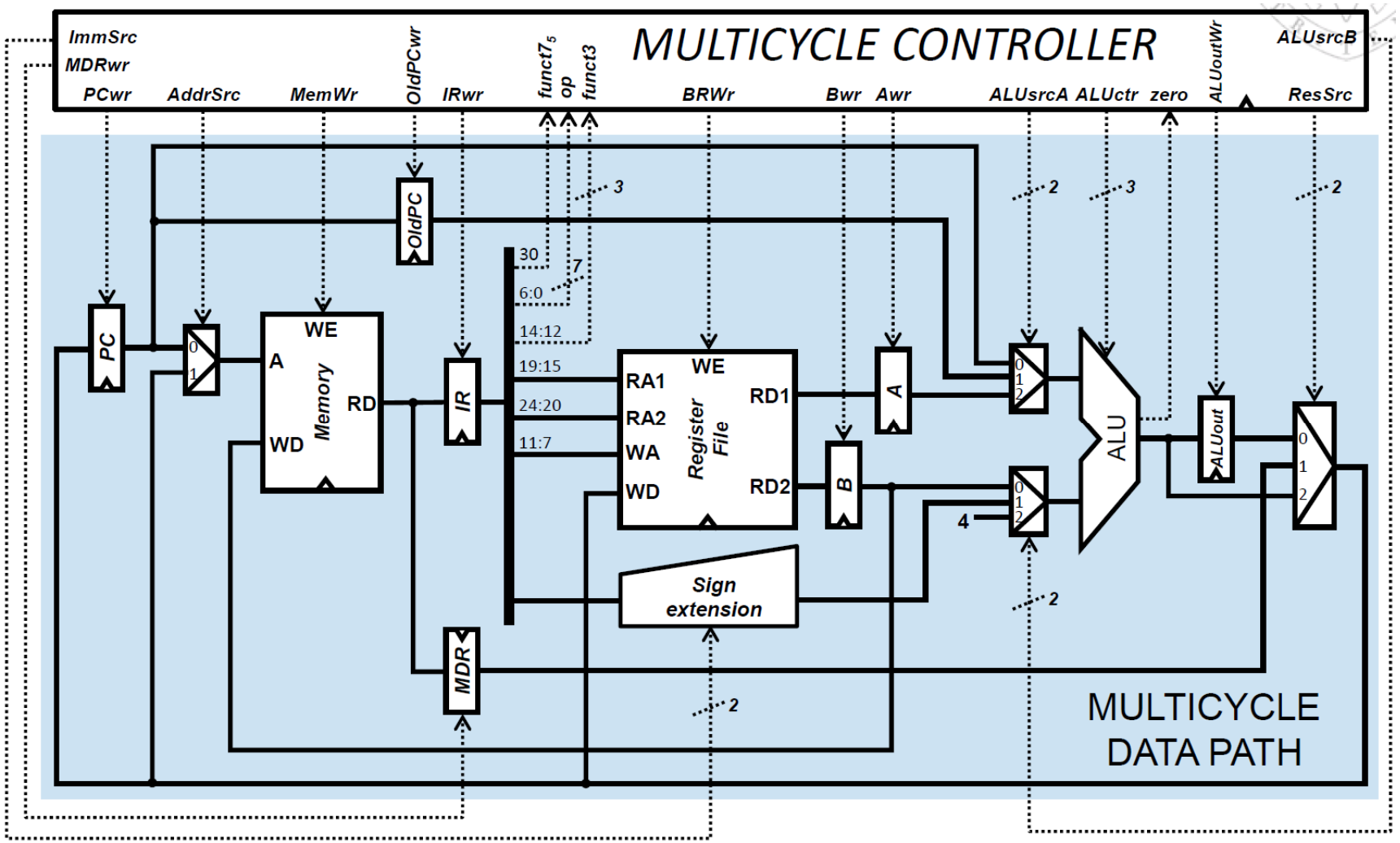
Output function

state	Branch	PCupdate	AddrSrc	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



Other signals:

- Signals MDRwr, BRwr, Awr y Bwr are 0 because the MDR register, the register file and the A and B registers are not written.



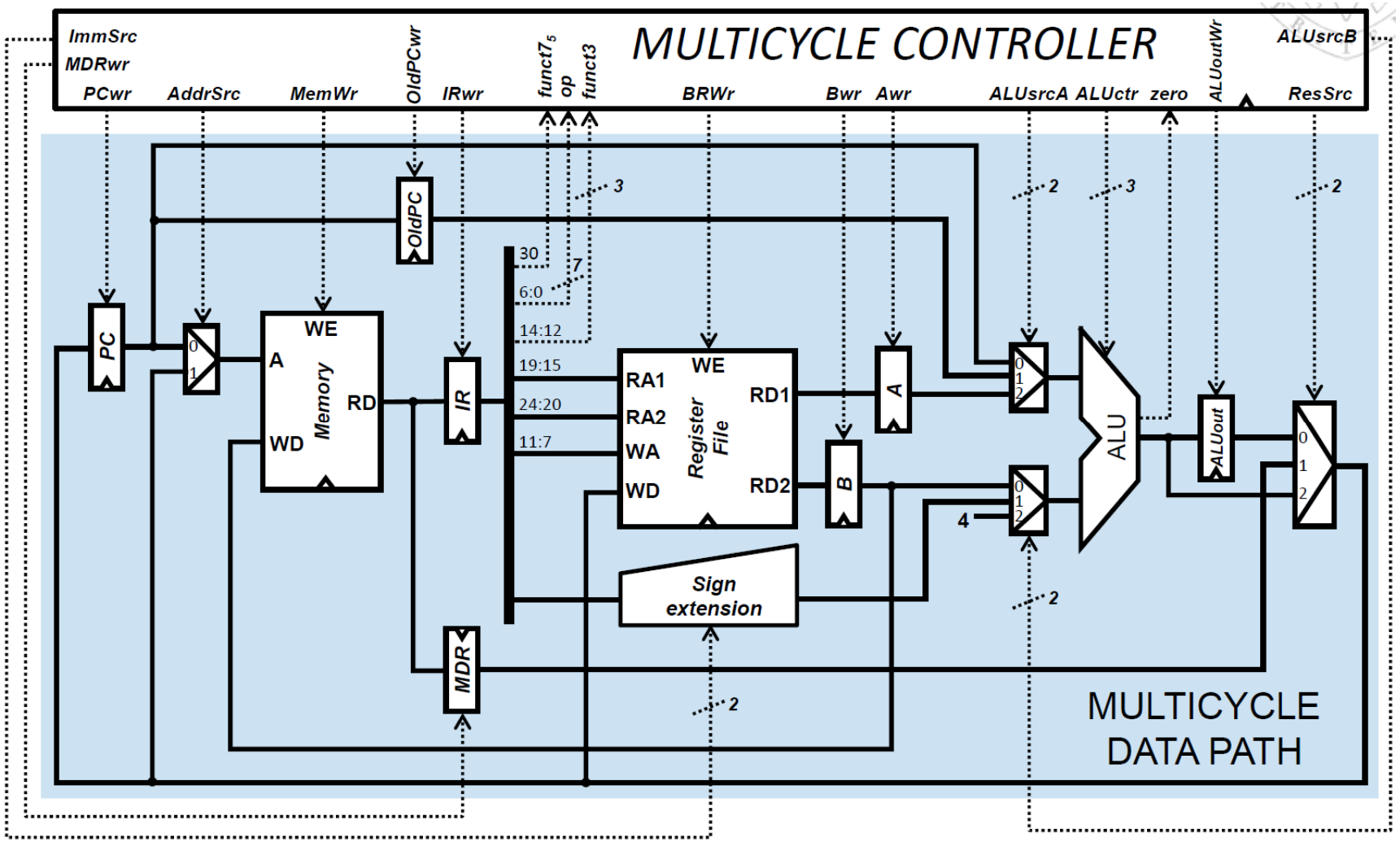
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 1 is written into the A register, thus Awr=1



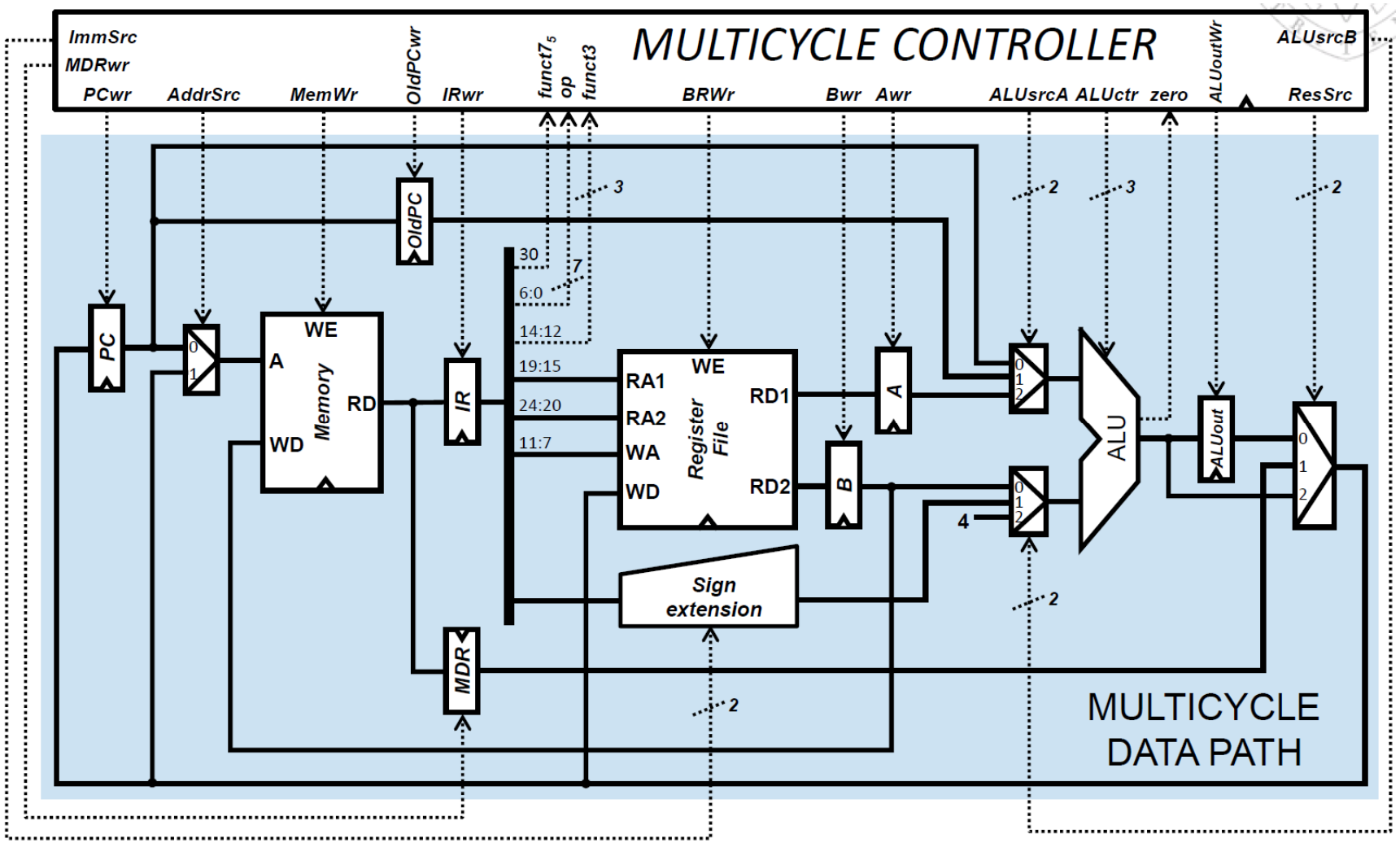
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 2 is written into the B register, thus Bwr=1



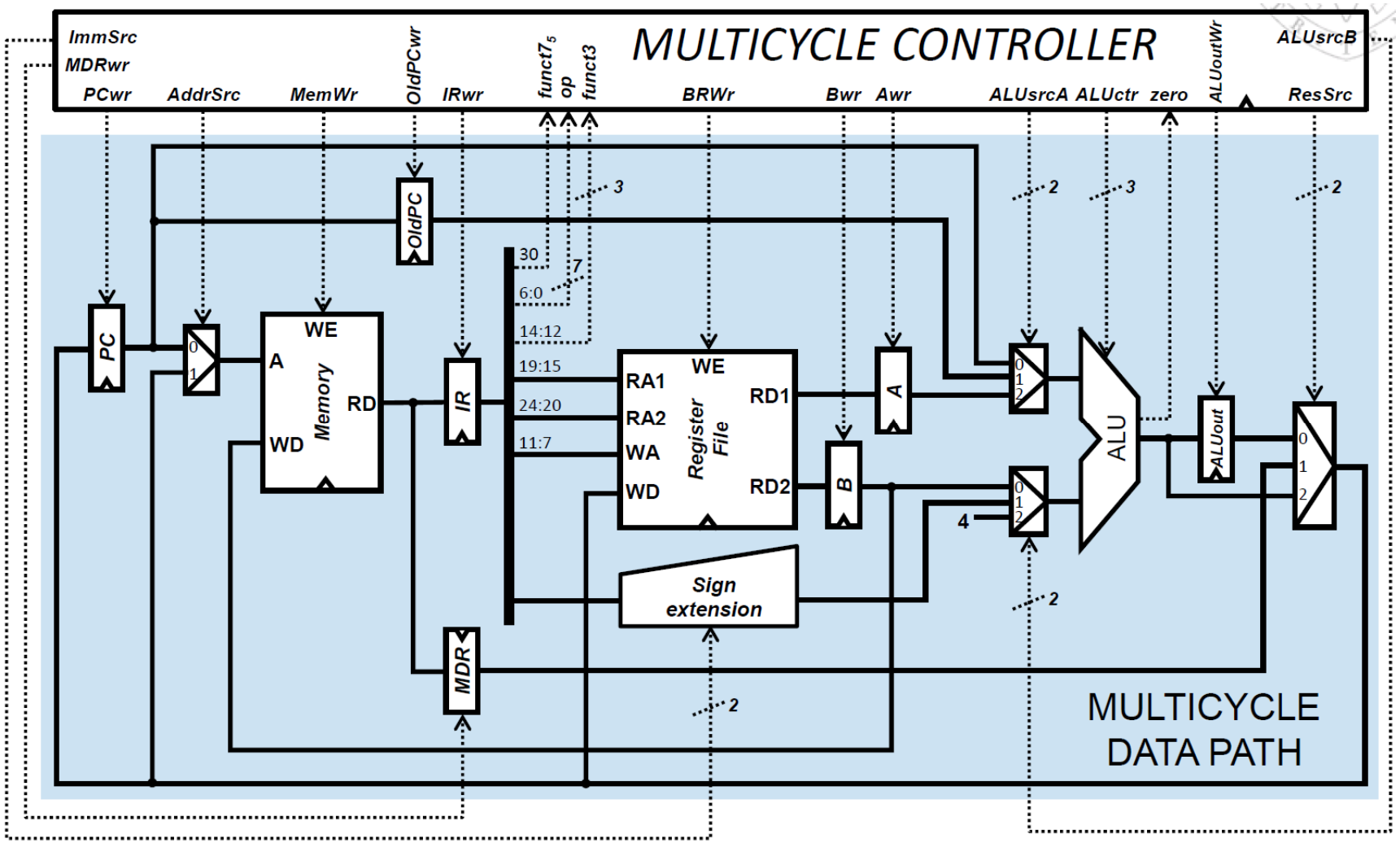
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The ALU performs the addition (ALUop=00) of OldPC + sExt(imm). The OldPC value arrives at the ALU upper input through channel 1 of the corresponding MUX, thus ALUsrcA is 01. The signed extended immediate arrives at the ALU lower input through channel 1 of the corresponding MUX, thus ALUsrcB is 01. The result of the addition is written in ALUout, thus ALUoutWr is 1.



Output function

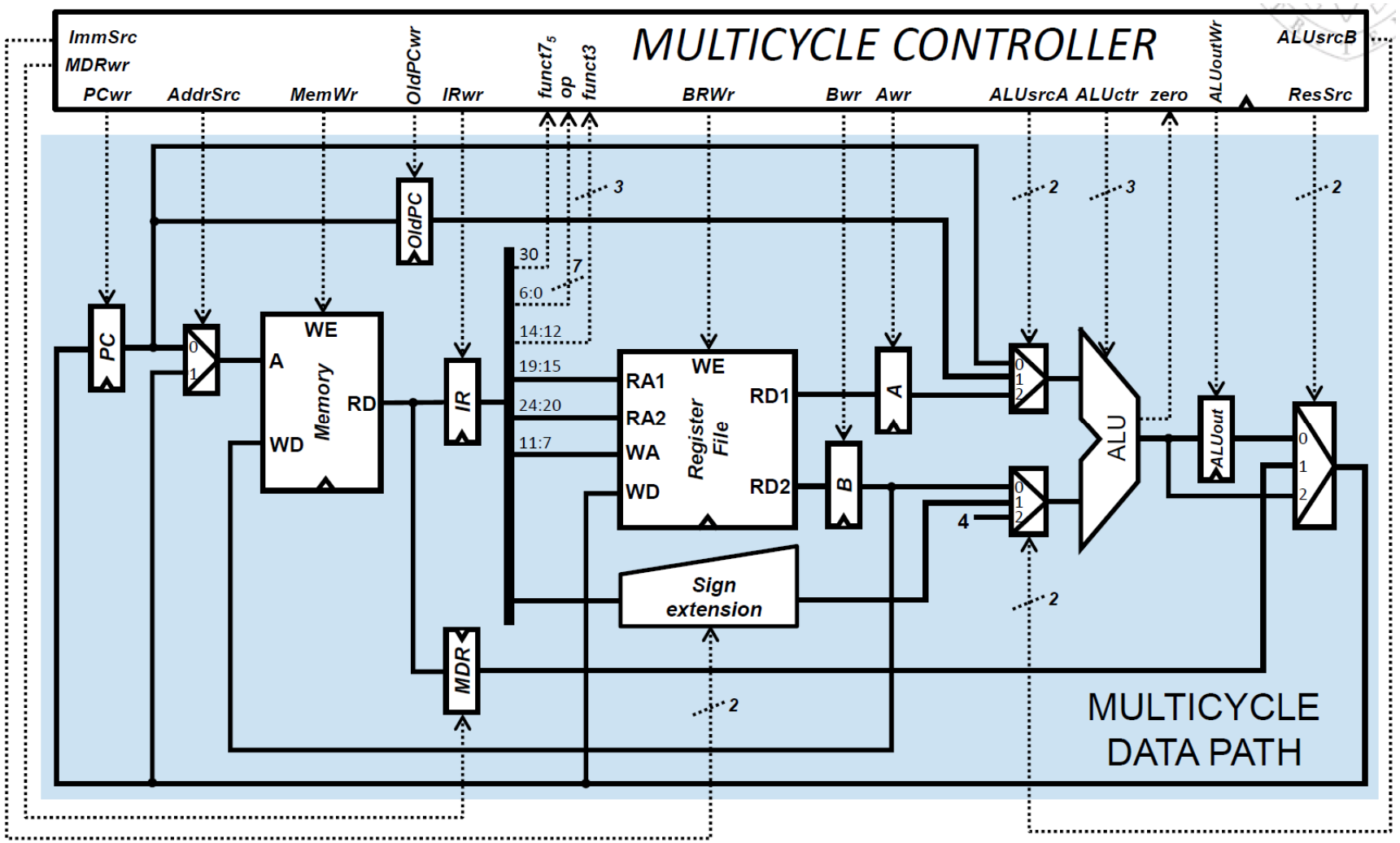
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

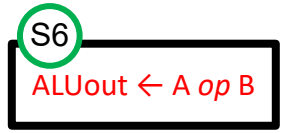
Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr and BRwr are 0 because the OldPC register, and the register file are not written in this cycle.
- ResSrc is a “don’t care” because the register file and the PC are not written in this cycle.

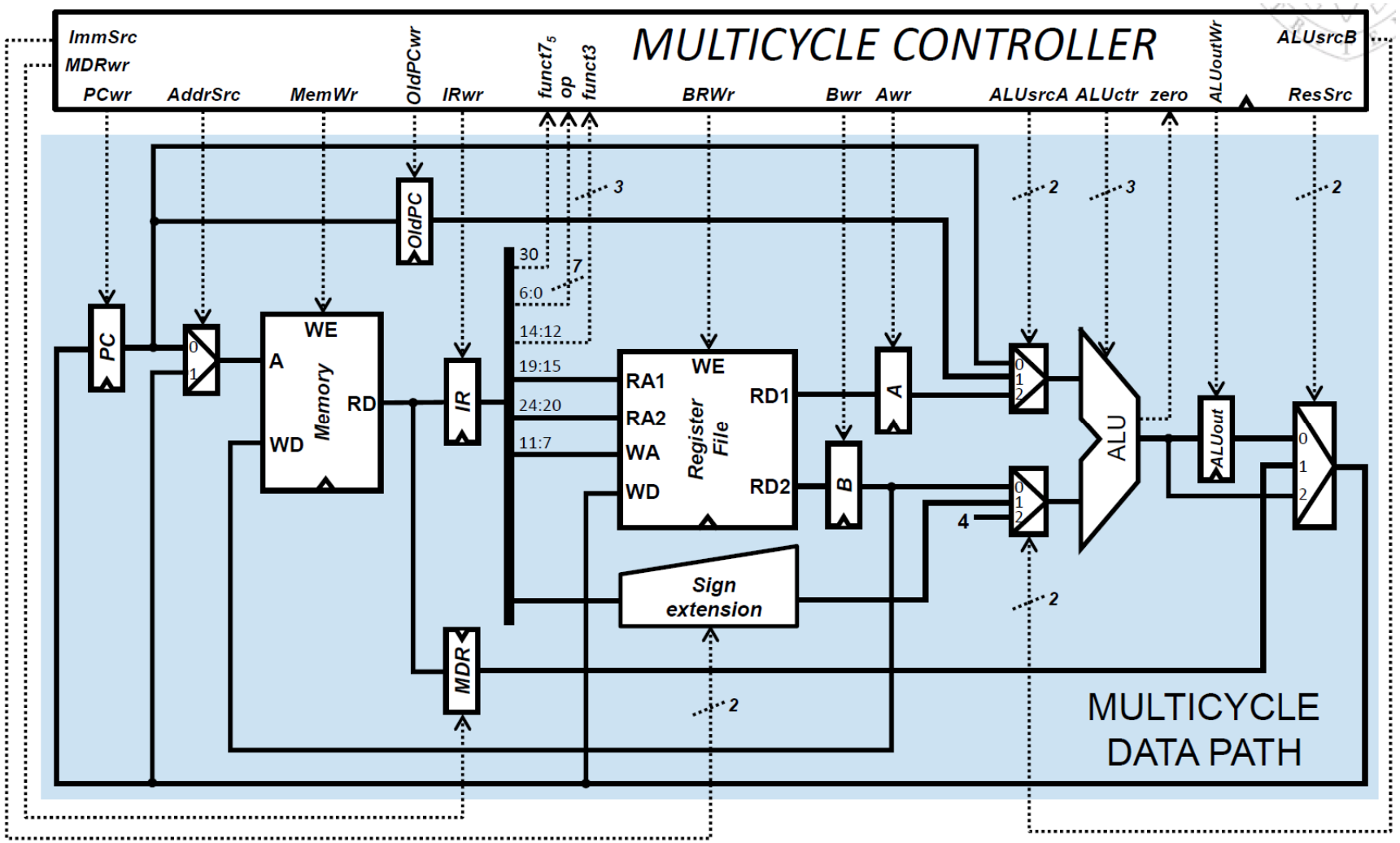


Output function

state	Branch	PCupdate	AddrSrc	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

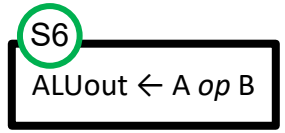


- The ALU performs the addition A+B (this is an arithmetic-logic addition, thus ALUop=00). The value of A arrives at the ALU upper input through channel 2 of the corresponding MUX, thus ALUsrcA is 10. The value of B arrives at the ALU lower input through channel 0 of the corresponding MUX, thus ALUsrcB is 00. The result of the addition is written in ALUout, thus ALUoutWr is 1.



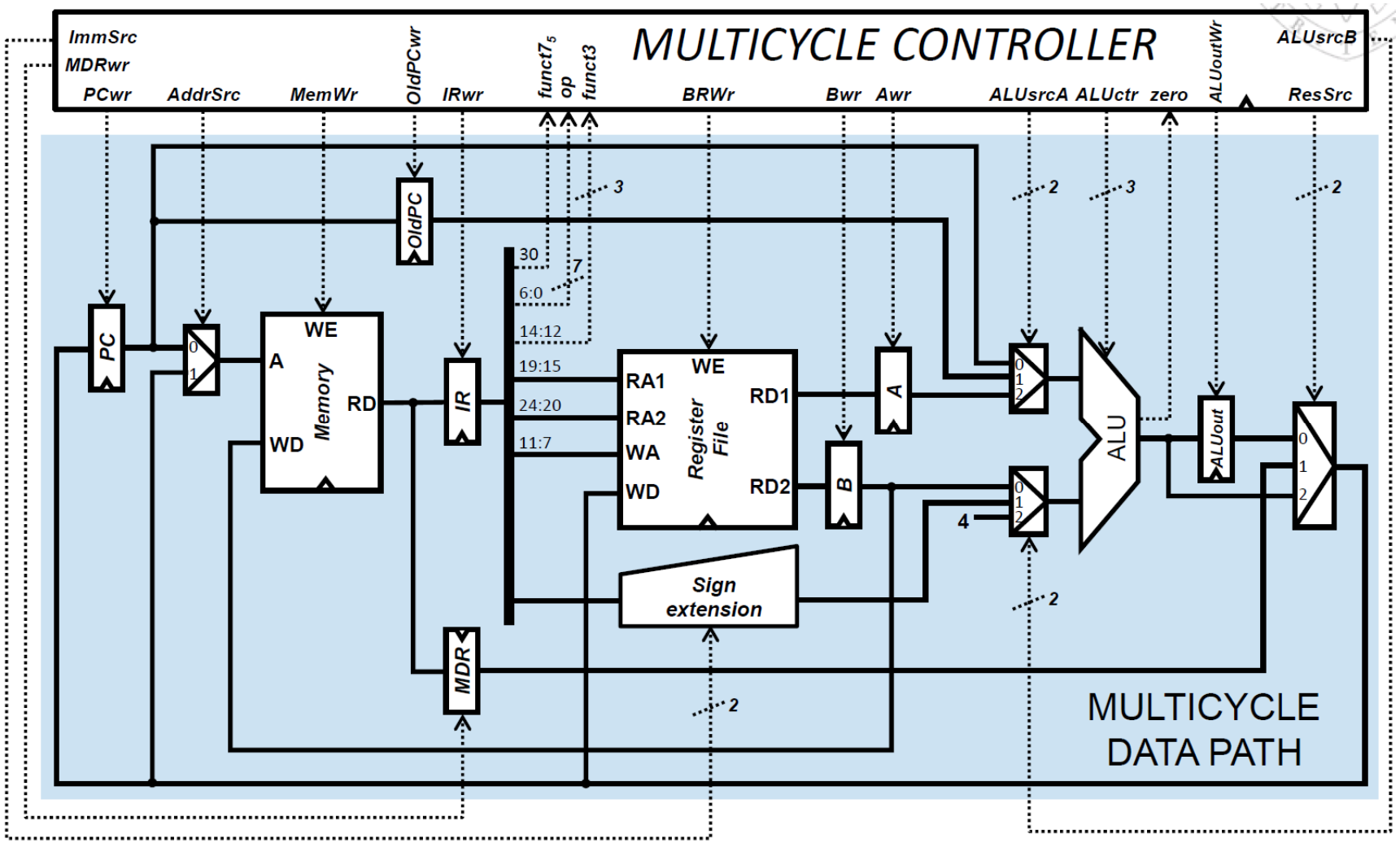
Output function

state	Branch	PCupdate	AddrSrc	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



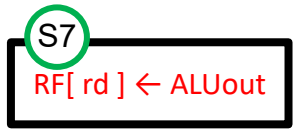
Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr, BRwr, Awr and Bwr are 0 because the OldPC register, the register file and the A and B registers are not written in this cycle.
- ResSrc is a “don’t care” because the register file and the PC are not written in this cycle.

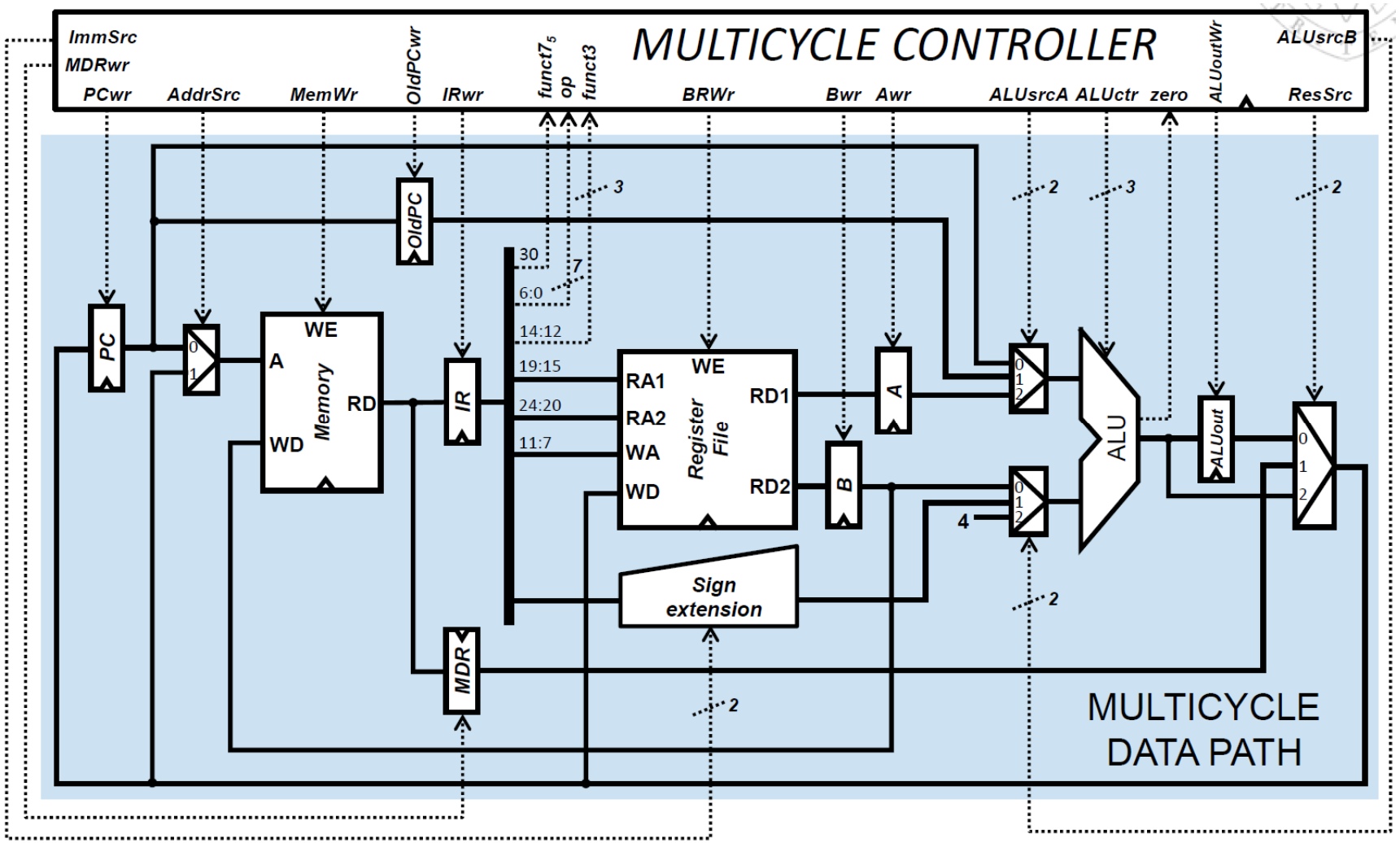


Output function

state	Branch	PCupdate	AddrSrc	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

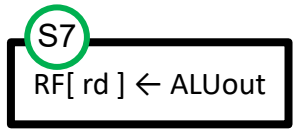


- The content of ALUout (which is the result of the addition performed in the previous cycle) is routed to the register file through channel 0 of the MUX controlled by ResSrc, and therefore this signal is 00. This value is written in the destination register (rd), determined by bits 11:7 of the instruction. Since the register file is written the BRwr signal is 1.



Output function

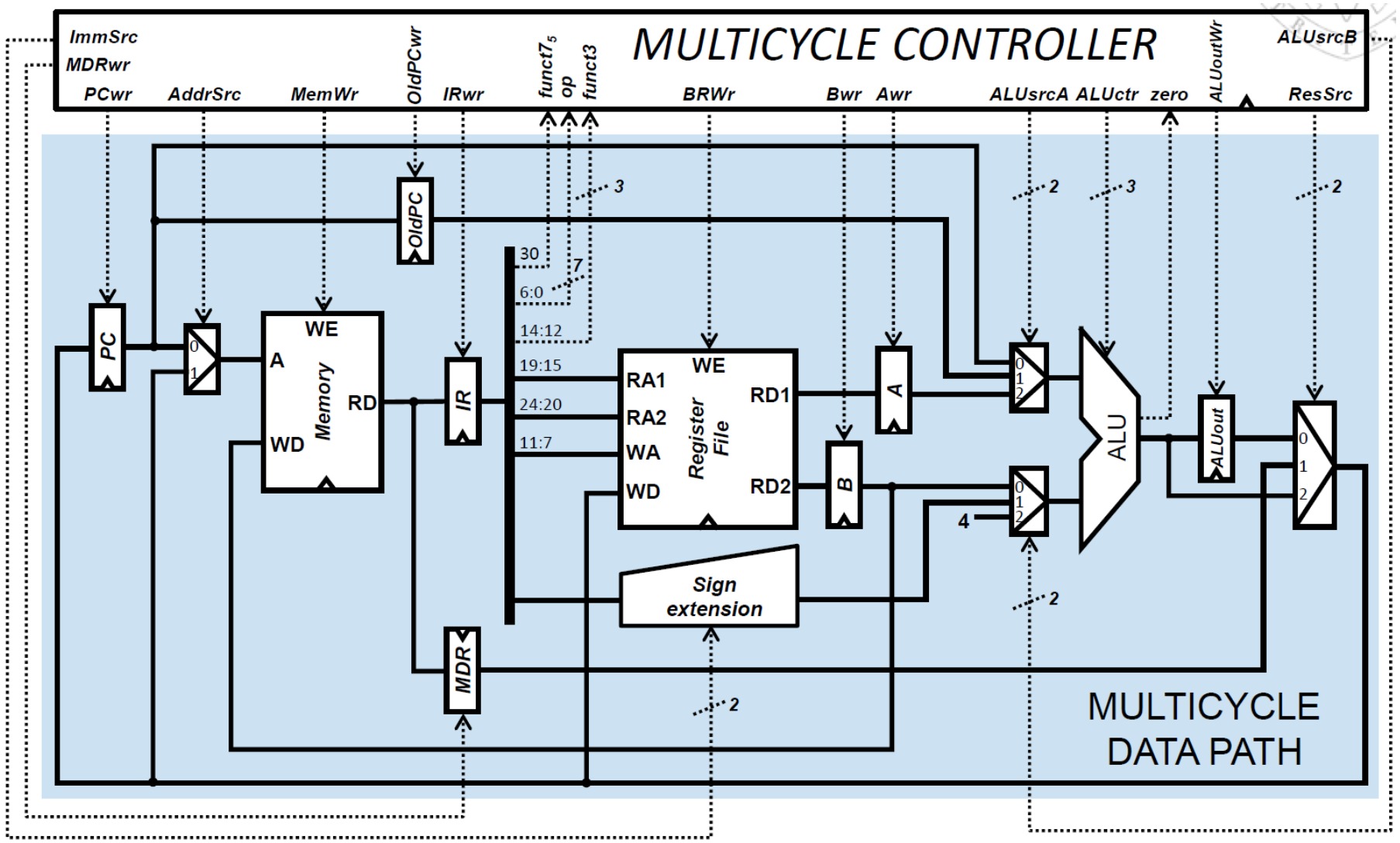
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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

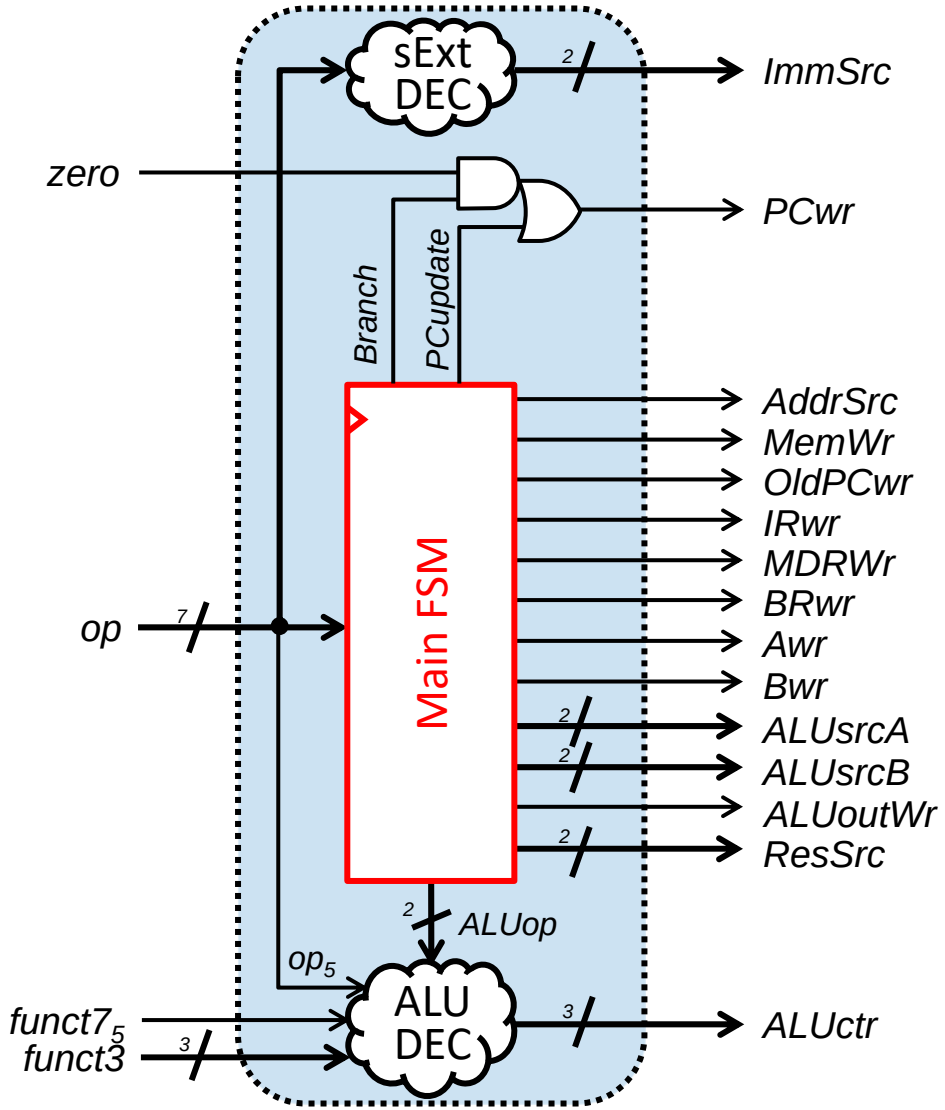


Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- OldPCwr, Awr, Bwr and MemWr are 0 because the OldPC register, the A and B registers and the memory are not written in this cycle.
- The ALU is not used in this cycle (nothing is written in its output register, thus ALUoutWr=0) and therefore the signals that control the ALU input MUX (ALUsrcA y ALUsrcB) are “don’t care”. For the same reason, ALUop is also a “don’t care”.

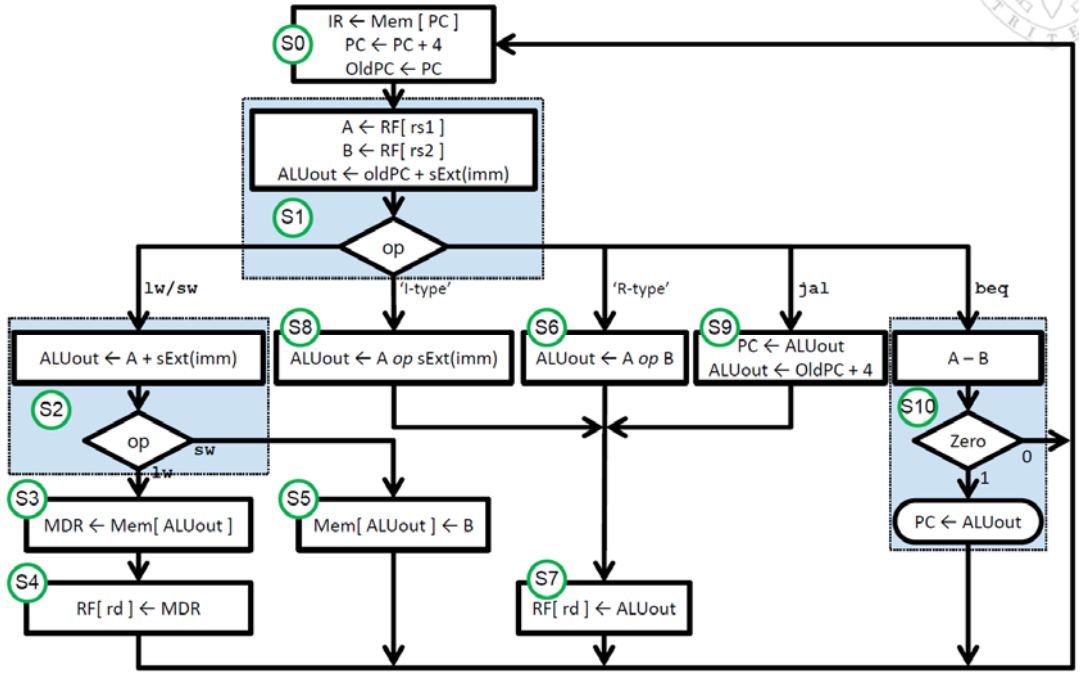
3) Provide the value of the control signals produced in a multicycle RISC-V when executing a **`jal`** instruction.



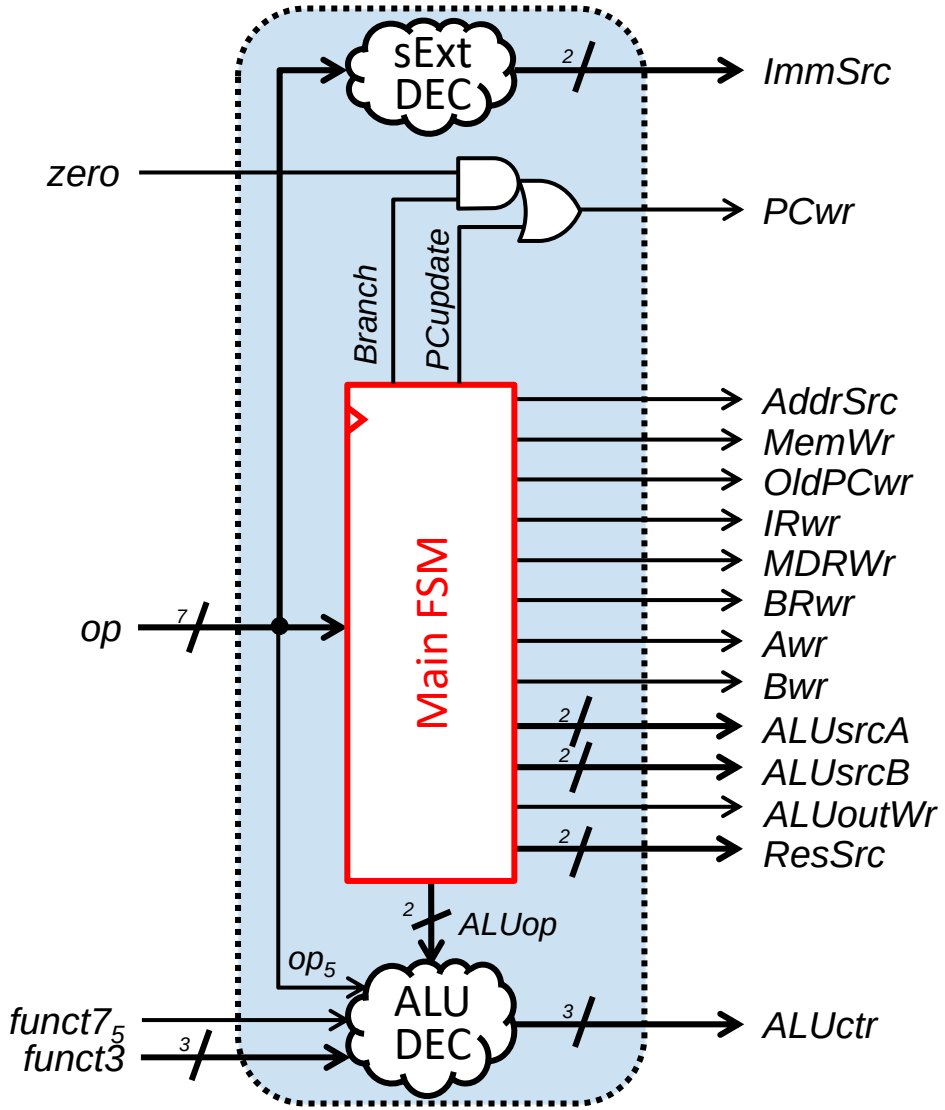


Función de salida

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

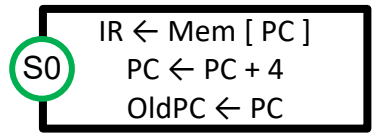


"jal" instructions follow the state sequence S0-S1-S9-S7

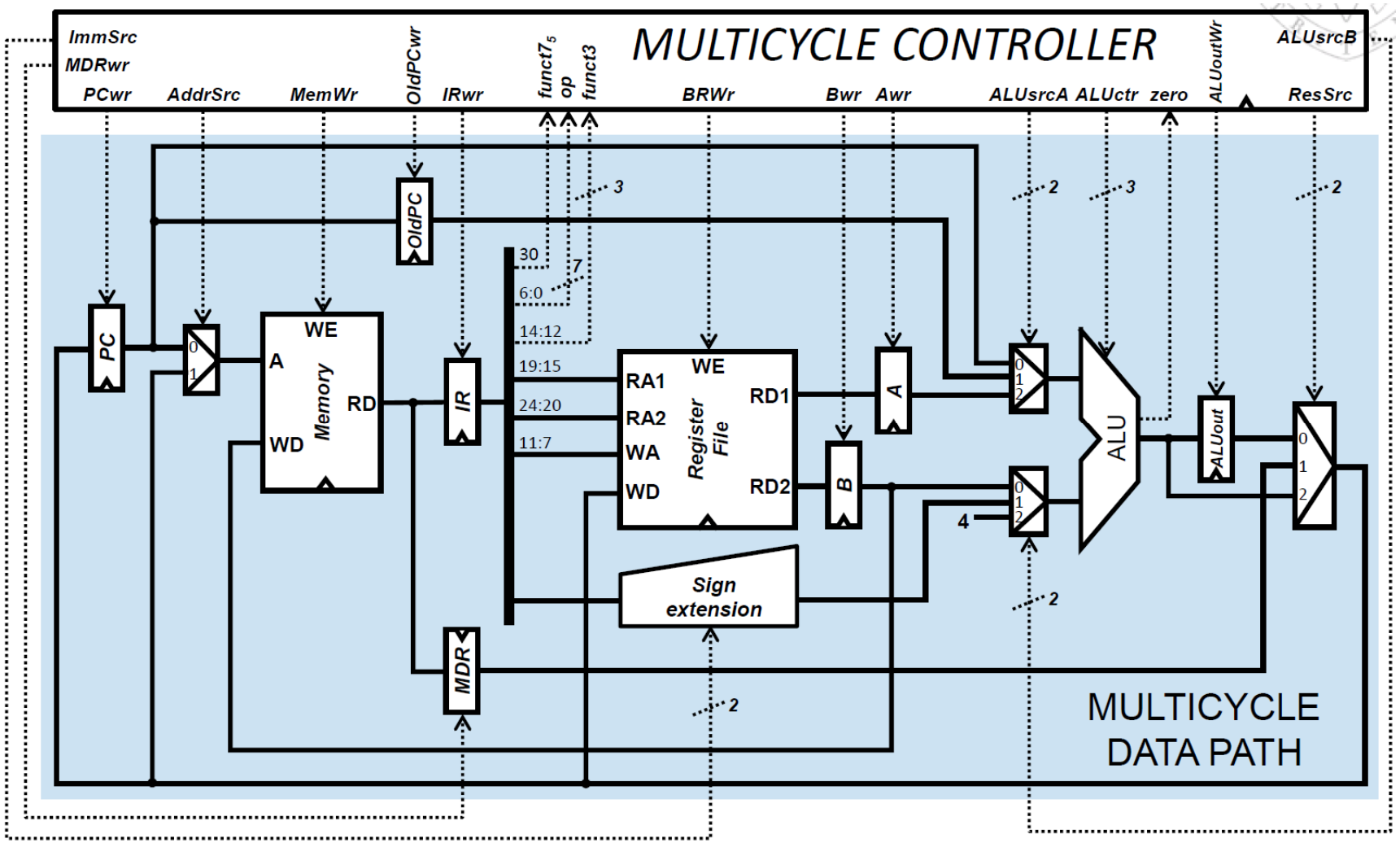


Función de salida

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

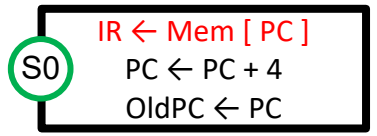


- The Branch signal is 0 because this is not a beq instruction
- The PCupdate signal is 1 because the PC is updated with PC+4



Output function

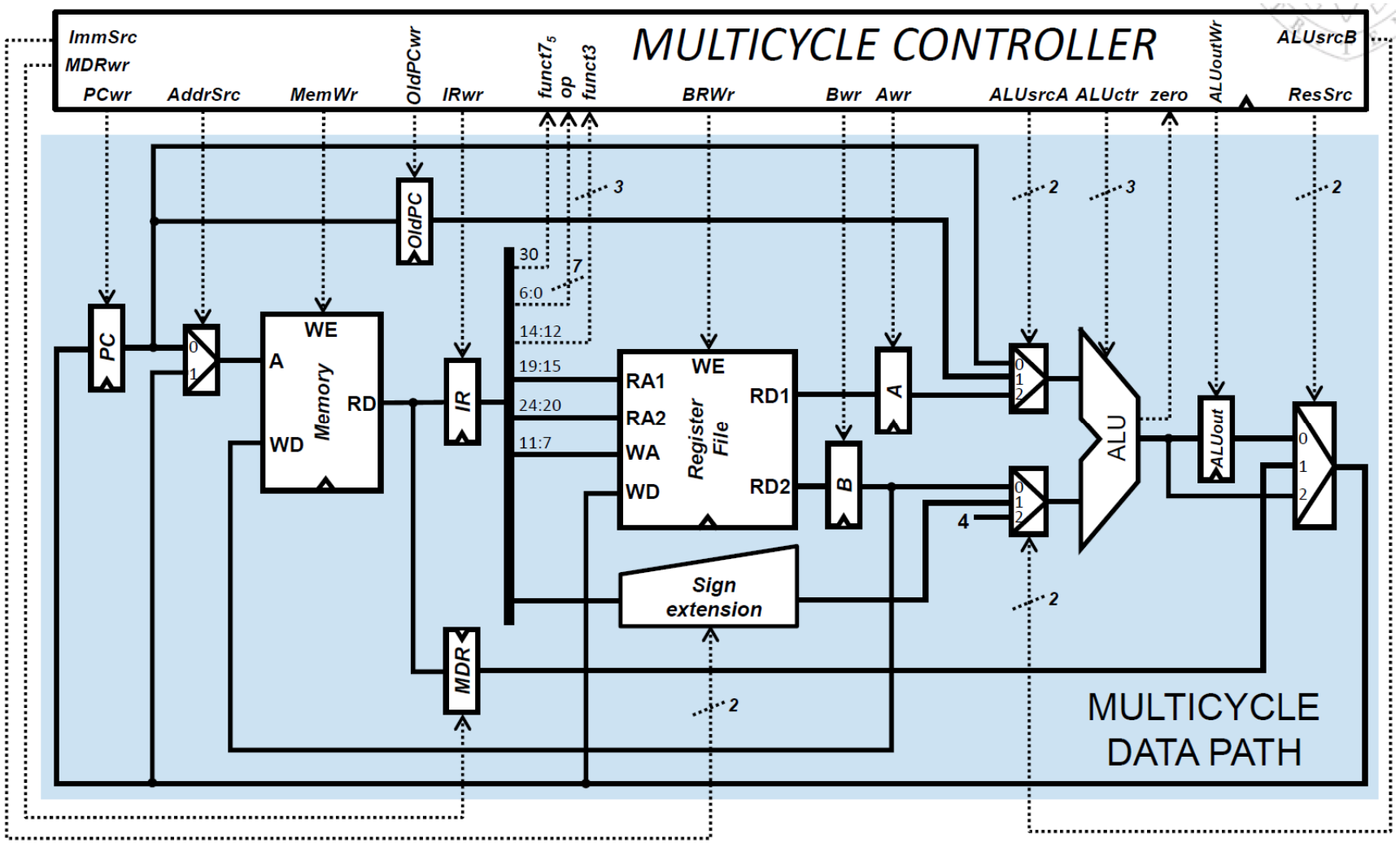
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 is 0 because the memory address to read is the one provided by the PC

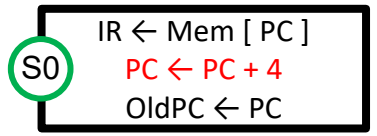
- IRwr is 1 because the instruction fetched from memory is written in the IR

- MemWr is 0 because in this cycle the memory is read, but it is not written

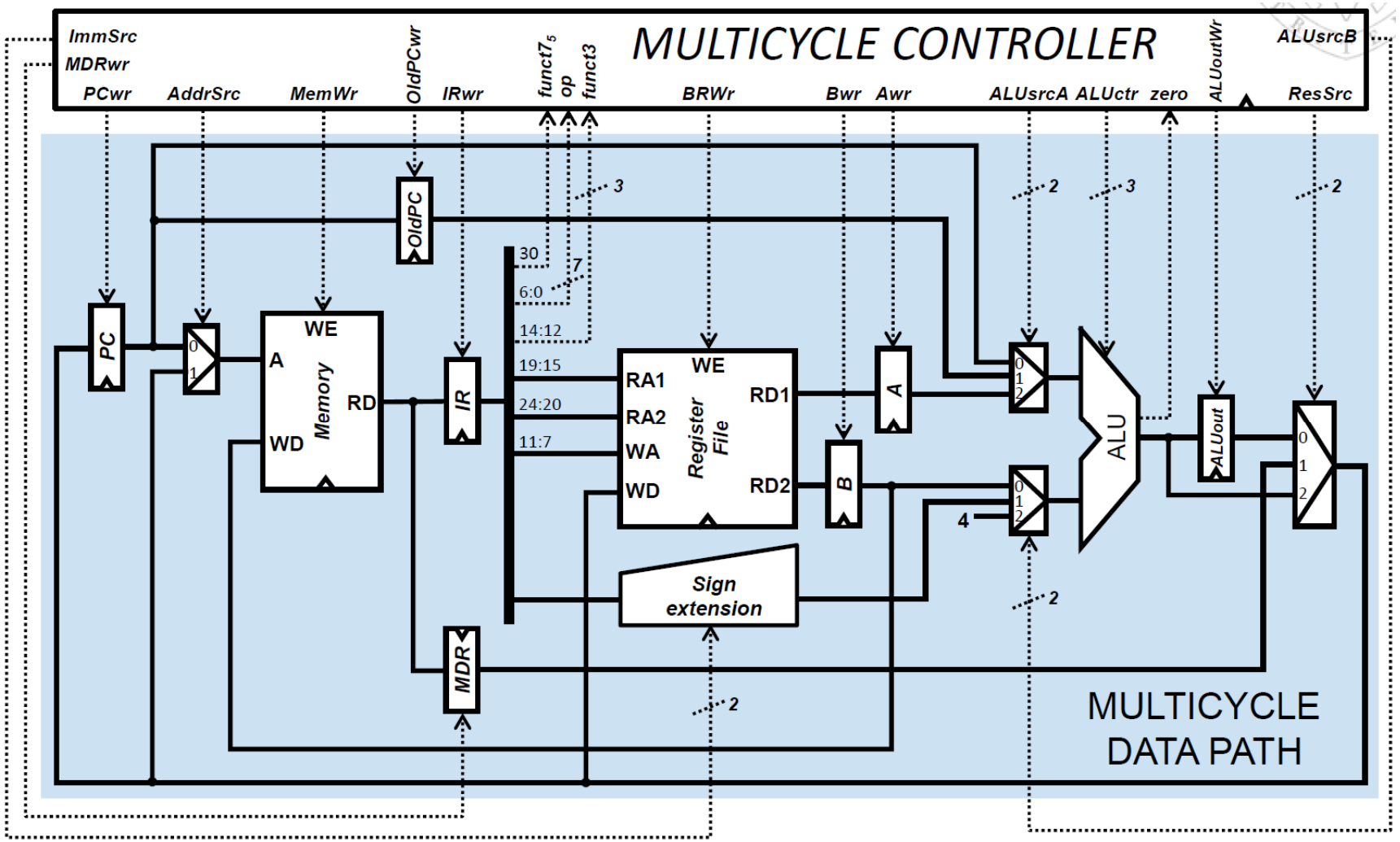


Output function

state	Branch	PCupdate	AddrSrc	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

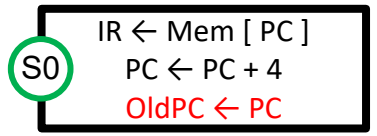


- The ALU performs PC+4 (ALUop=00). The PC value arrives at the ALU upper input through channel 0 of the corresponding MUX, thus ALUsrcA is 00. The immediate 4 arrives at the ALU lower input through channel 2 of the corresponding MUX, thus ALUsrcB is 10. The result is not written in the ALUout register (thus ALUoutWr is 0), but it is directly routed to the PC, in order to update its value, through channel 2 of the MUX controlled by ResSrc (10).

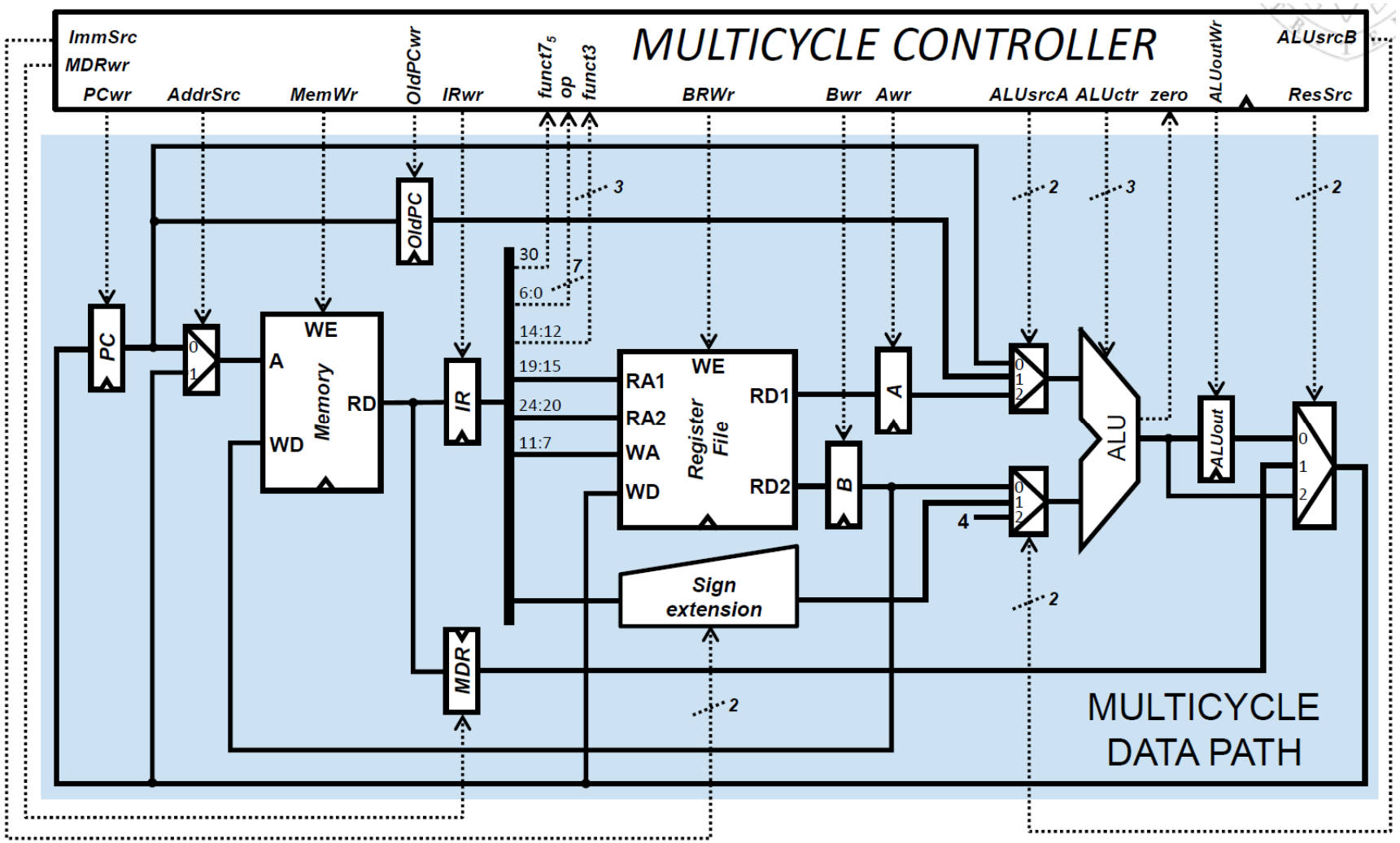


Output function

state	Branch	PCupdate	AddrSrc	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

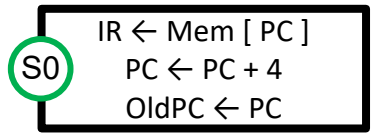


- OldPCwr is 1 because we write the current PC value in the OldPC register.



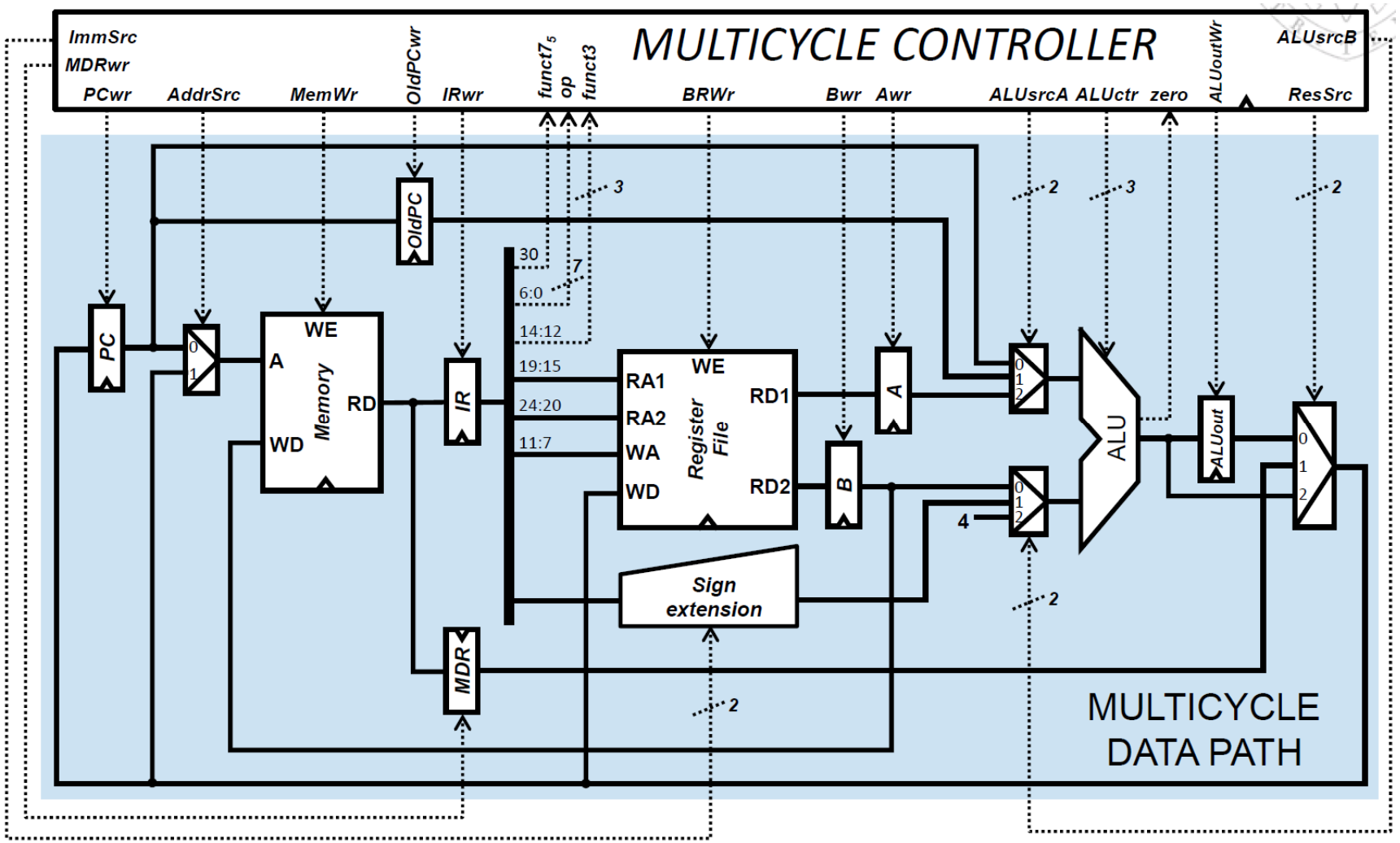
Output function

state	Branch	PCupdate	AddrSrc	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



Other signals:

- Signals MDRwr, BRwr, Awr y Bwr are 0 because the MDR register, the register file and the A and B registers are not written.



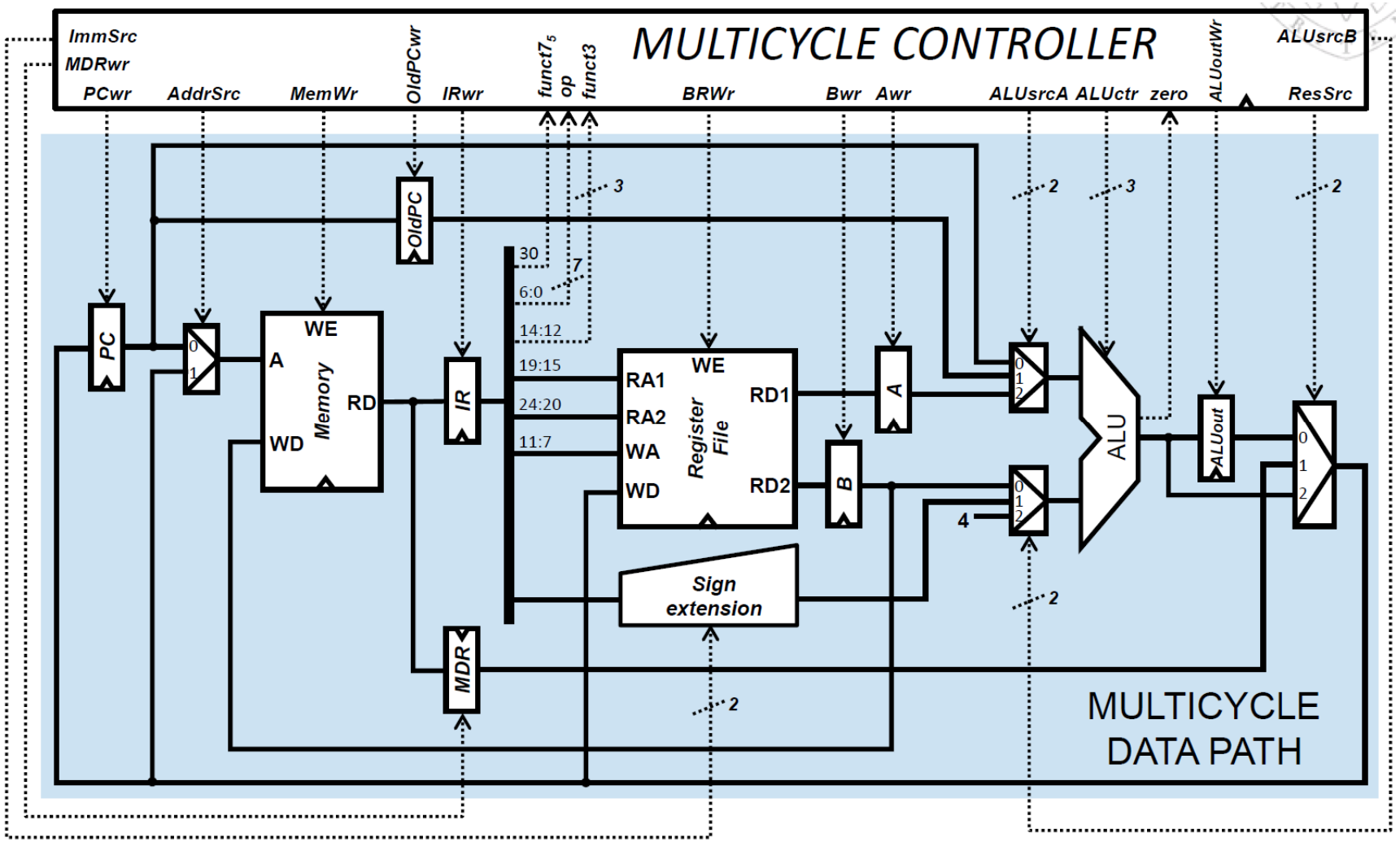
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 1 is written into the A register, thus Awr=1



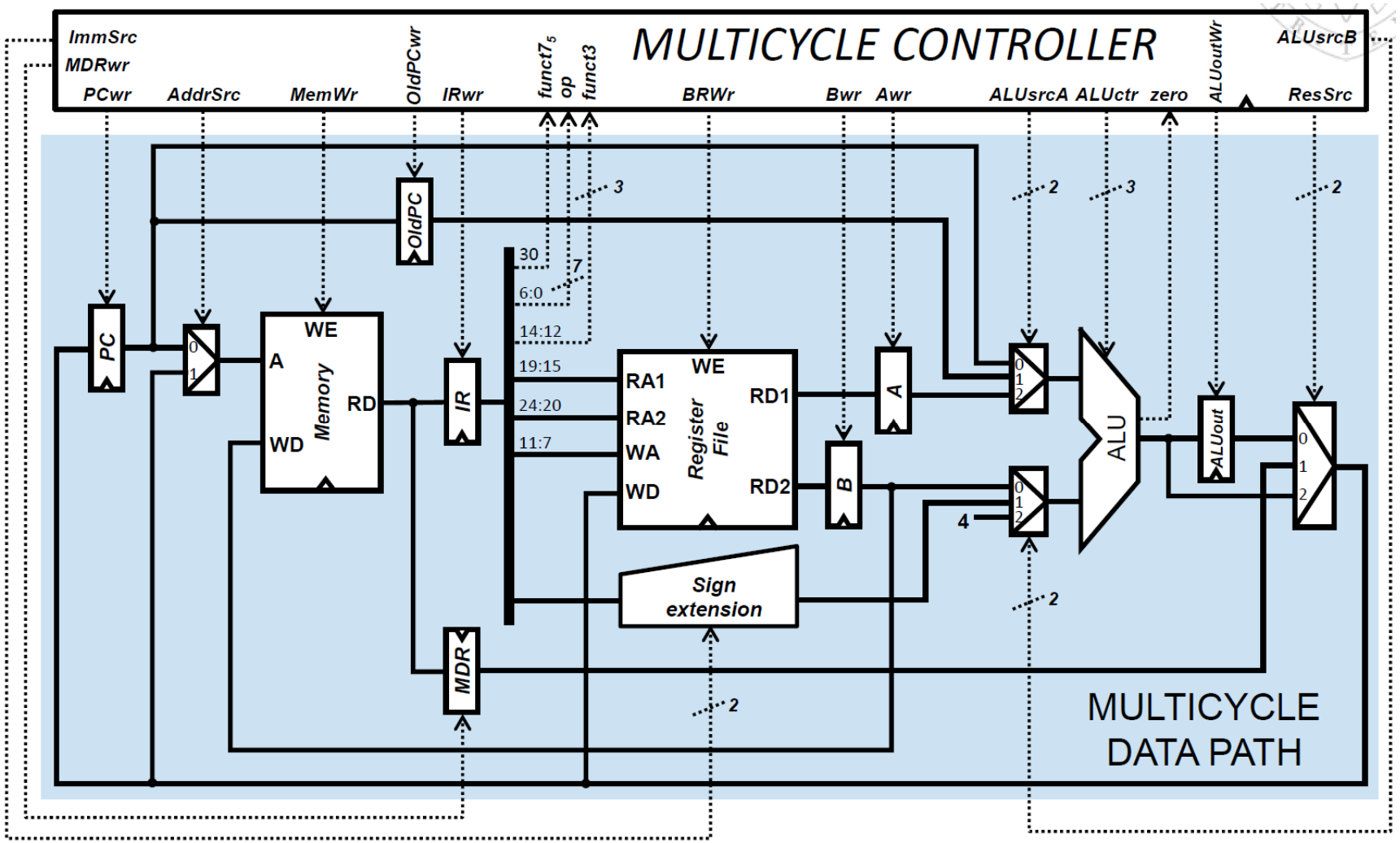
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The content of source register 2 is written into the B register, thus Bwr=1



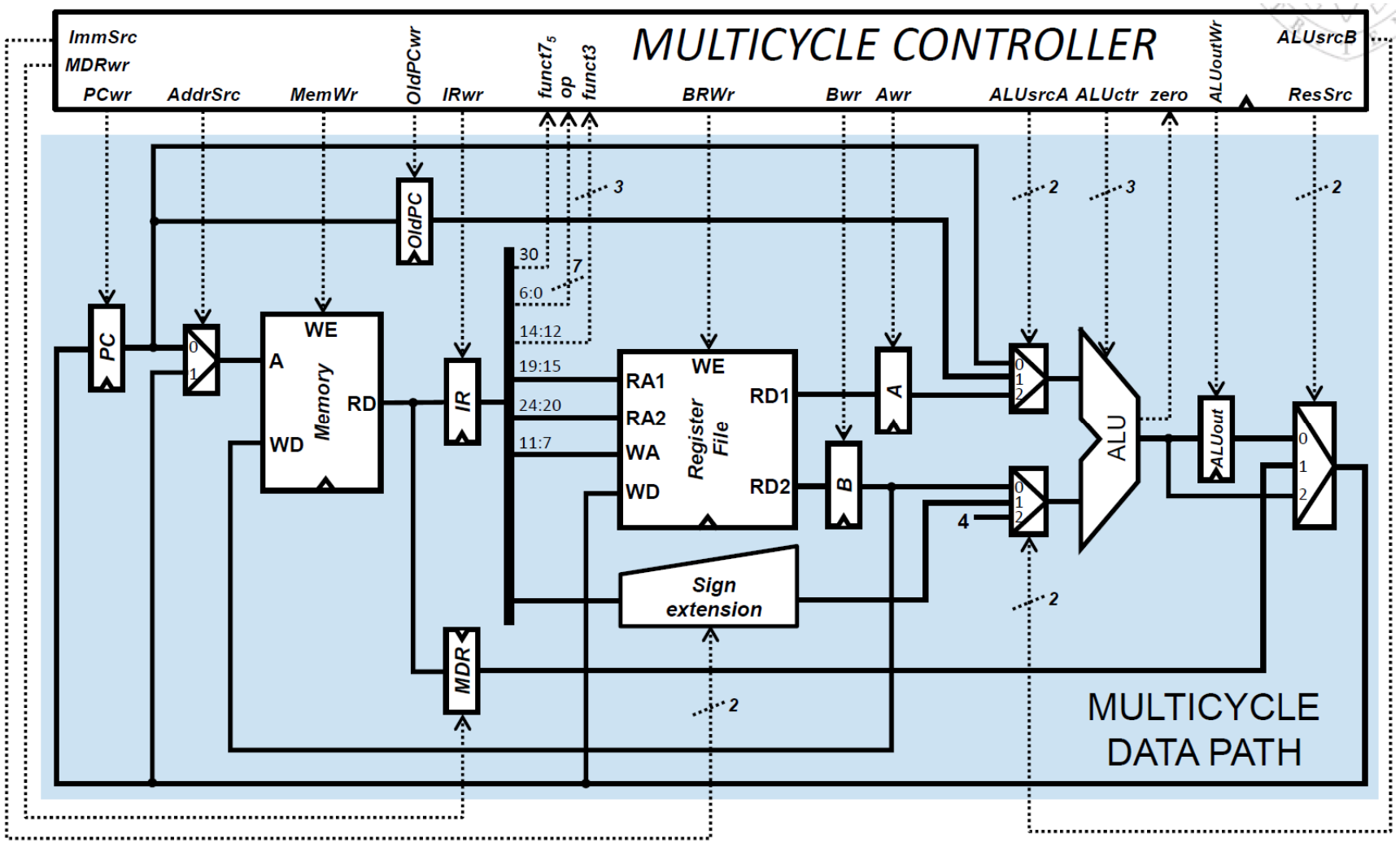
Output function

state	Branch	PCupdate	AddrSrc	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

- The ALU performs the addition (ALUop=00) of OldPC + sExt(imm). The OldPC value arrives at the ALU upper input through channel 1 of the corresponding MUX, thus ALUsrcA is 01. The signed extended immediate arrives at the ALU lower input through channel 1 of the corresponding MUX, thus ALUsrcB is 01. The result of the addition is written in ALUout, thus ALUoutWr is 1.



Output function

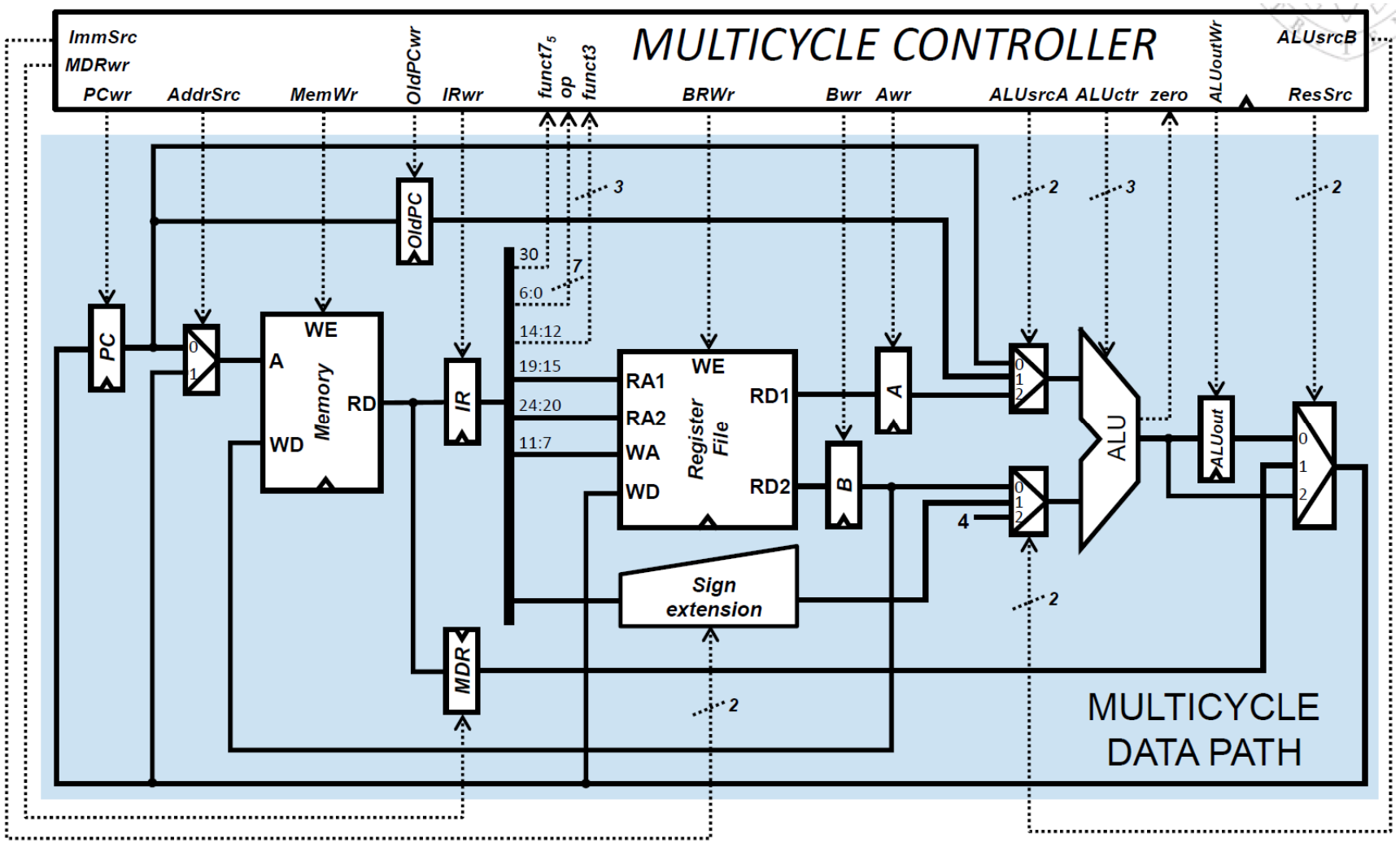
state	Branch	PCupdate	AddrSrc	MemWr	OldPCwr	IRwr	MDRwr	BRwr	Awr	Bwr	ALUsrcA	ALUsrcB	ALUop	ALUoutWr	ResSrc
S0	0	1	0	0	1	1	0	0	0	0	00	10	00	0	10
S1	0	0	–	0	0	0	0	0	1	1	01	01	00	1	–
S2	0	0	–	0	0	0	0	0	0	0	10	01	00	1	–
S3	0	0	1	0	0	0	1	0	0	0	–	–	–	0	00
S4	0	0	–	0	0	0	0	1	0	0	–	–	–	0	01
S5	0	0	1	1	0	0	0	0	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 ← RF[rs1]
B ← RF[rs2]
ALUout ← oldPC + sExt(imm)

S1

Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr and BRwr are 0 because the OldPC register, and the register file are not written in this cycle.
- ResSrc is a “don’t care” because the register file and the PC are not written in this cycle.

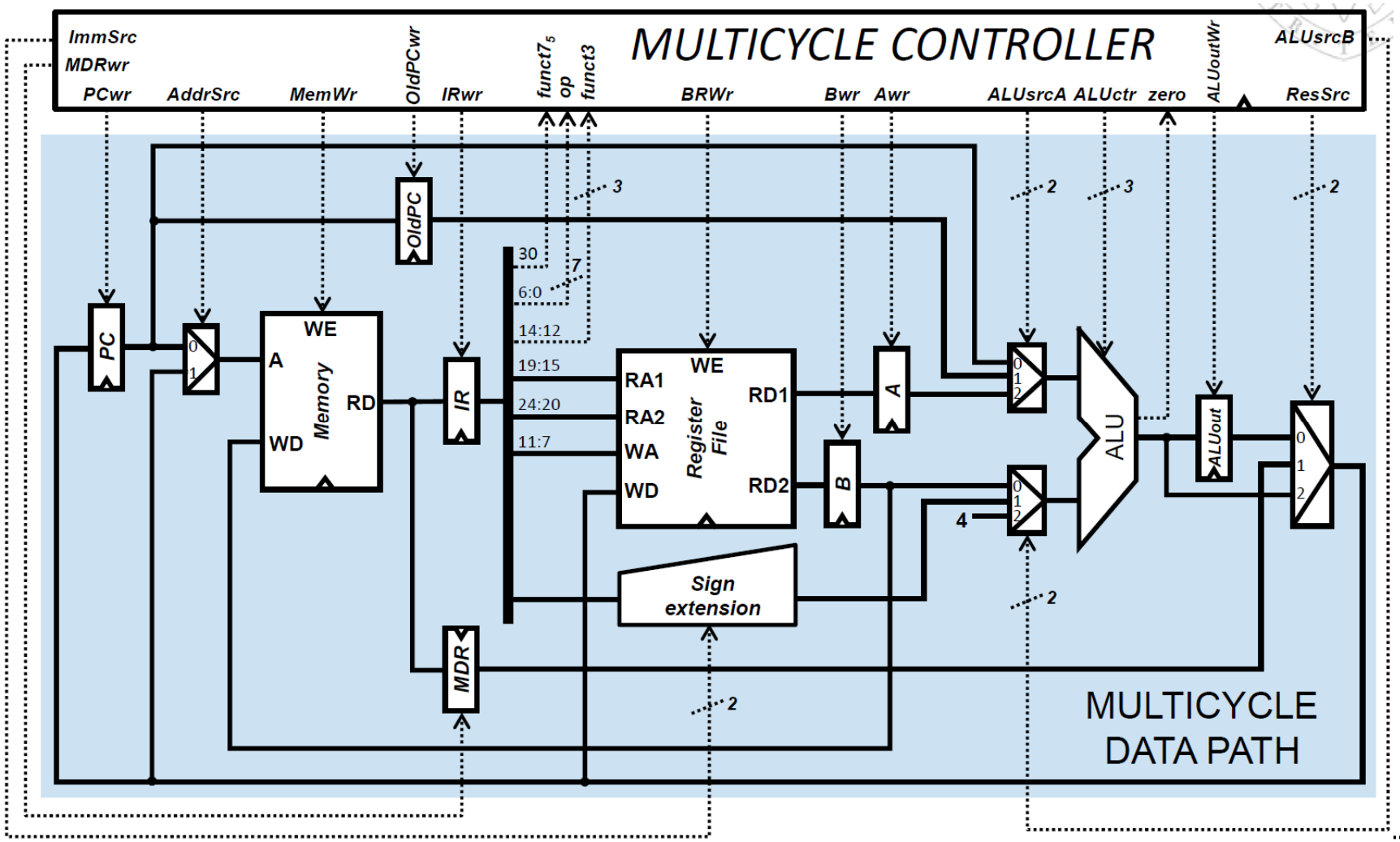


Output function

state	Branch	PCupdate	AddrSrc	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

- The content of ALUout (which is the result of the addition performed in the previous cycle, i.e., the target address of the jal instruction) is routed to the PC through channel 0 of the MUX controlled by ResSrc, and therefore this signal is 00. Also, since the value of PC is updated, the PCupdate signal is 1.

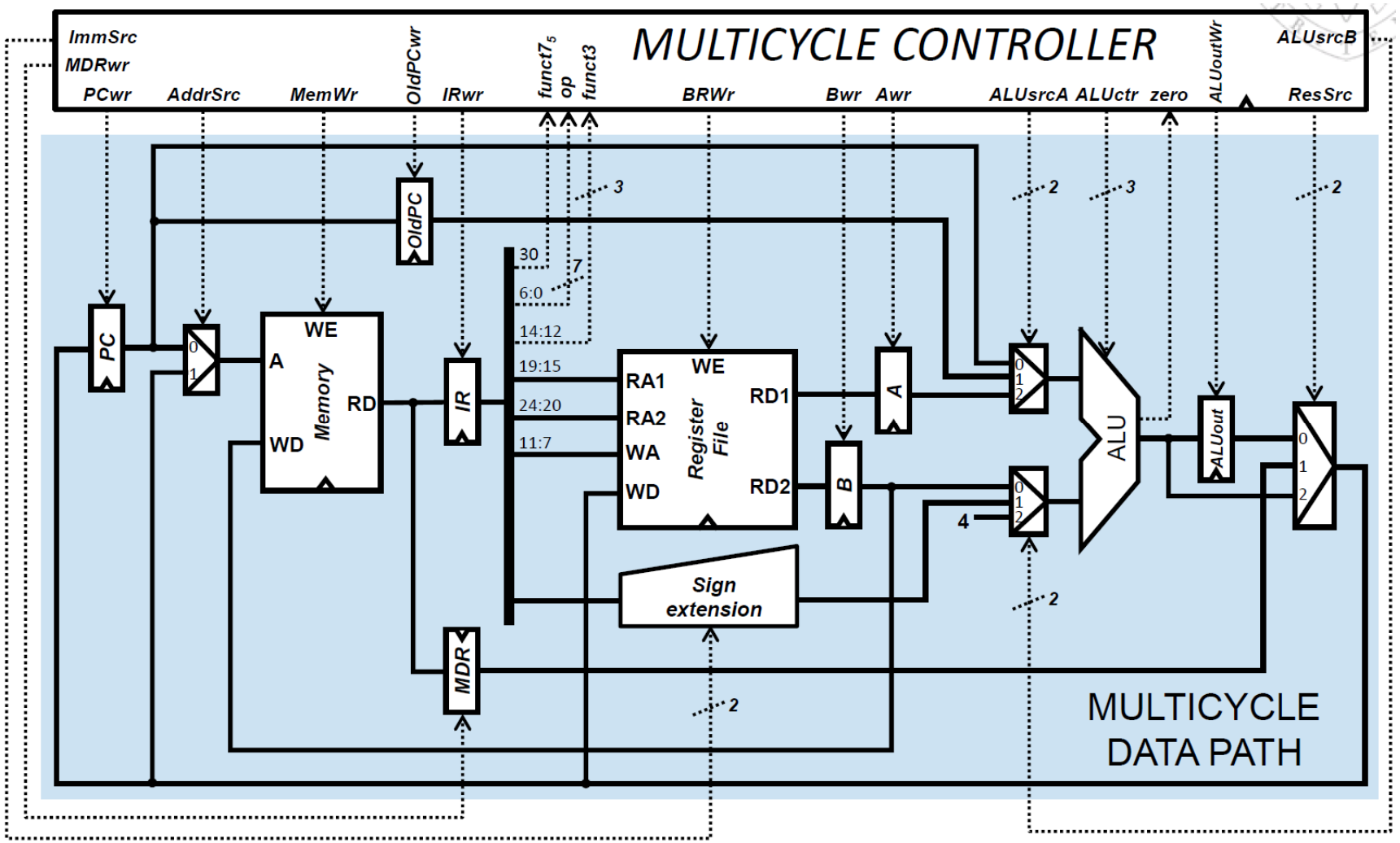


Output function

state	Branch	PCupdate	AddrSrc	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

- The ALU performs the addition (ALUop=00) of OldPC + 4 (to obtain the return address of the jal instruction). The OldPC value arrives at the ALU upper input through channel 1 of the corresponding MUX, thus ALUsrcA is 01. The immediate 4 arrives at the ALU lower input through channel 2 of the corresponding MUX, thus ALUsrcB is 10. The result of the addition is written in ALUout, thus ALUoutWr is 1.



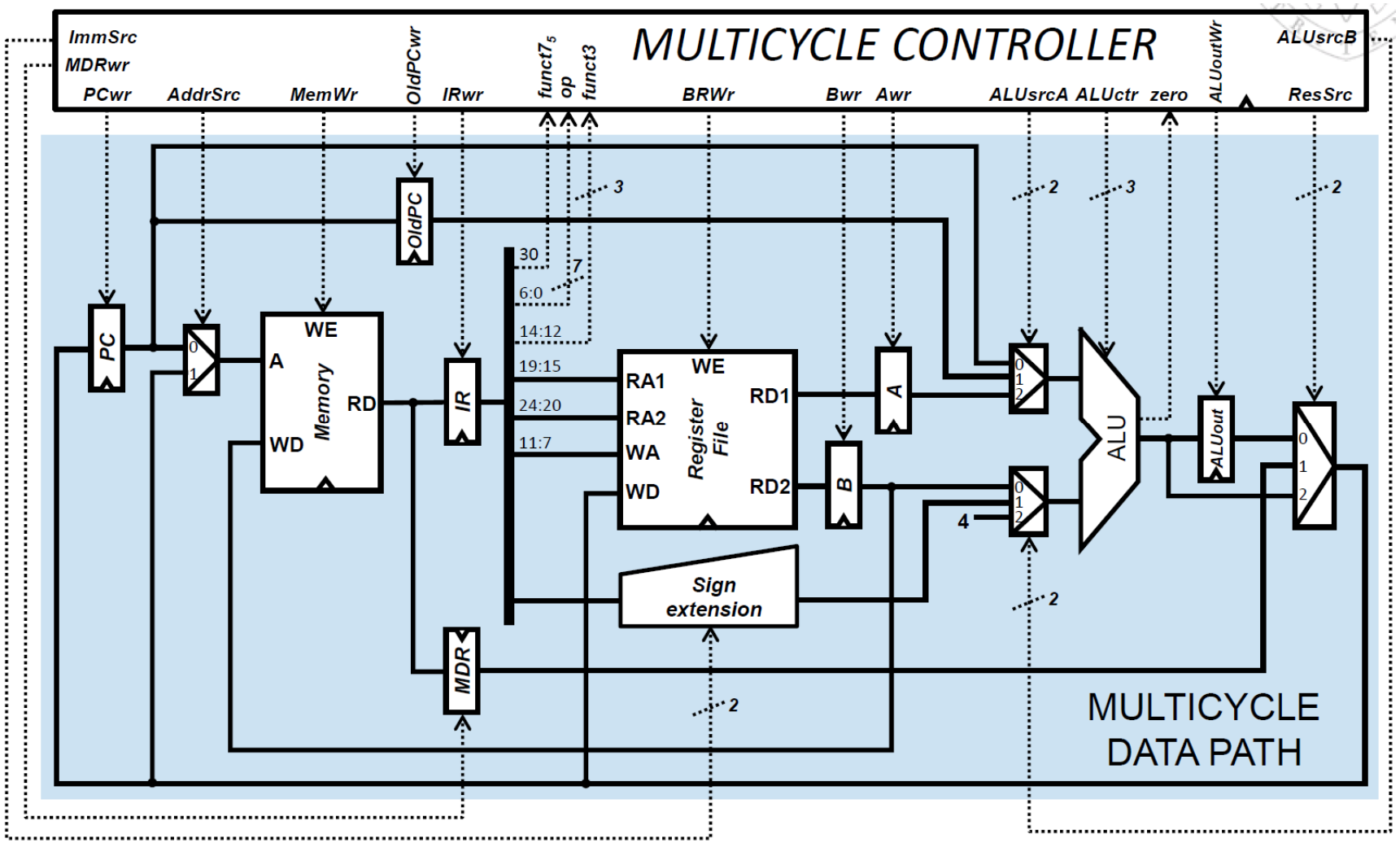
Output function

state	Branch	PCupdate	AddrSrc	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

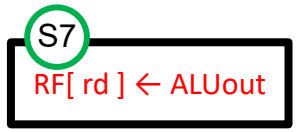
Other signals:

- Branch is 0 because this is not a beq instruction.
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- MemWr is 0 because the memory is not written.
- OldPCwr, BRwr, Awr and Bwr are 0 because the OldPC register, the register file and the A and B registers are not written in this cycle.

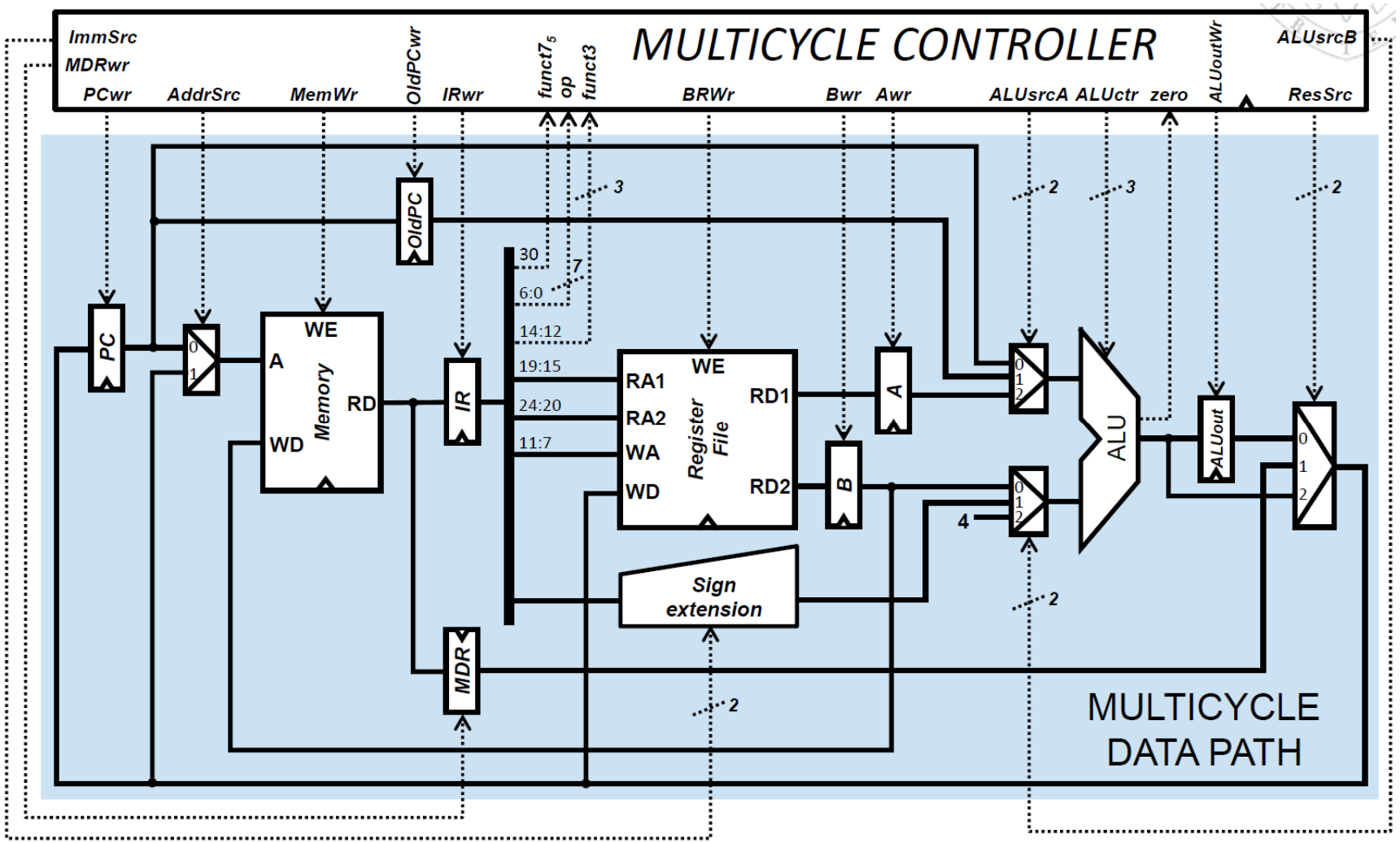


Output function

state	Branch	PCupdate	AddrSrc	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

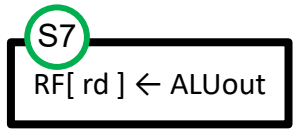


- The content of ALUout (which is the result of the addition performed in the previous cycle, i.e., the return address of the jal instruction) is routed to the register file through channel 0 of the MUX controlled by ResSrc, and therefore this signal is 00. This value is written in the destination register (rd), determined by bits 11:7 of the instruction. Since the register file is written the BRwr signal is 1.



Output function

state	Branch	PCupdate	AddrSrc	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



Other signals:

- Branch and PCupdate are 0 because this is not a beq instruction and the PC is not updated in this cycle
- The AddrSrc is a “don’t care” because nothing is written in IR or MDR (IRwr and MDRwr=0).
- OldPCwr, Awr, Bwr and MemWr are 0 because the OldPC register, the A and B registers and the memory, are not written in this cycle.
- The ALU is not used in this cycle (nothing is written in its output register, thus ALUoutWr=0) and therefore the signals that control the ALU input MUX (ALUsrcA y ALUsrcB) are “don’t care”. For the same reason, ALUop is also a “don’t care”.

4) A given program, which is running on a multicycle processor, needs an average of 4.25 cycles per instruction. The program is formed by 10,000 instructions, with the following distribution: 30% are **`lw`**, 15% are **`sw`**, 20% are arithmetic-logic and the rest are **`jal`** and **`beq`**. Determine how many **`jal`** instructions and how many **`beq`** instructions are in such program.

Instr. type	Inst. %	Cycles/inst.
Load	30%	5
Store	15%	4
Arithmetic-logic	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

Also, instructions jal and beq represent, together, 35% of all instructions of the program, thus fjal + fbeq = 0.35

1.35 = 4 (0.35-fbeq) + 3 fbeq → fbeq = 0.05 → 5% are beq and 30% are jal

3,000 instructions are jal and 500 instructions are beq

5) A certain multicycle processor is running a program with 140 instructions. 70 of these instructions take 4 cycles to execute, 35 take 5 cycles, 20 take 3 cycles and the other 15 take 7 cycles. Calculate the CPI of this program. If the processor works with a 2 GHz frequency, calculate the execution time of the program.

Instr. Type	Num. Inst.	Cycles/inst
A	70	4
B	35	5
C	20	3
D	15	7

Total num. inst. (N) = 140

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

Execution_time = N*CPI/frequency = $140*4.4285/2*10^9$ = **310 ns**

6) Consider two multicycle processors with the following features

- PowerPC, which works with a 1.8 GHz frequency and 700 MIPS.
- Pentium 4, which works with a 1.6 GHz frequency and 850 MIPS.

Calculate the CPI of each processor.

Processor	Frequency	MIPS
PowerPC	1.8GHz	700
Pentium4	1.6GHz	850

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

→ $CPI = (f/MIPS) *10^{-6}$

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

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

7) Consider the two processors of the previous exercise. When running a certain program, the processors obtain a CPI of 5.5 (PowerPC) and 7 (Pentium 4). The machine code generated by the compiler has 9 million instructions (PowerPC) and 7.2 million instructions (Pentium). Which computer will run the program faster?

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

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

8) A given program runs on two multicycle processors (A and B), which work with 1 GHz and 1.5 GHz frequencies, respectively. The distribution of the program instructions running on A is as follows:

	arithmetic	load	store	branch
frequency	50%	25%	10%	15%
cycles	4	5	4	3

a) Calculate the CPI of the program running on processor A.
b) The number of instructions running on B is 60% of the executed in A, and its execution time is half the time in A. Calculate the CPI of the program running on processor B.

a) **CPI(A)** = 4x0.5 + 5x0.25 + 4x0.10 + 3x0.15 = **4.1**

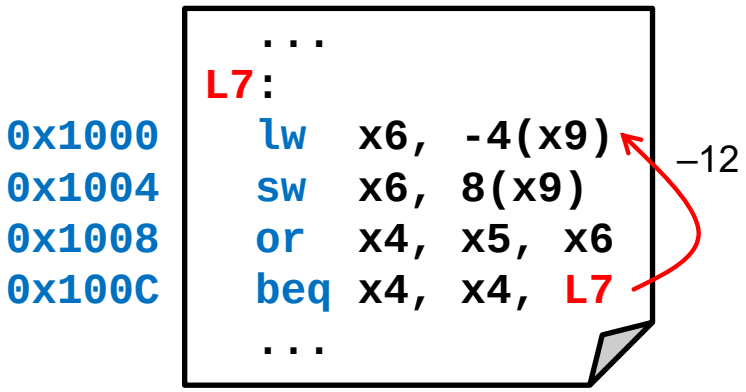
b) NI(B) = 0.6xNI(A); T(B) = 0.5xT(A);

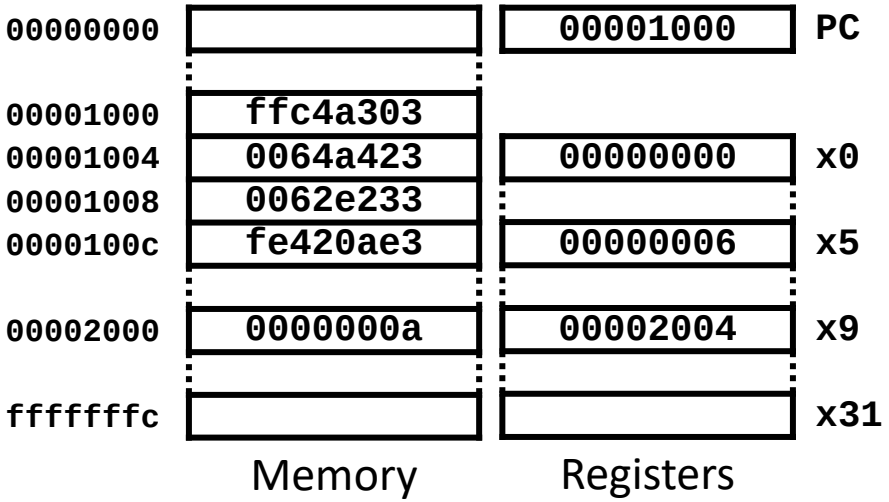
Time in A: T(A) = NI(A) x CPI(A) / f(A)
Time in B: 0.5xT(A) = 0.6xNI(A) x CPI(B) / f(B)

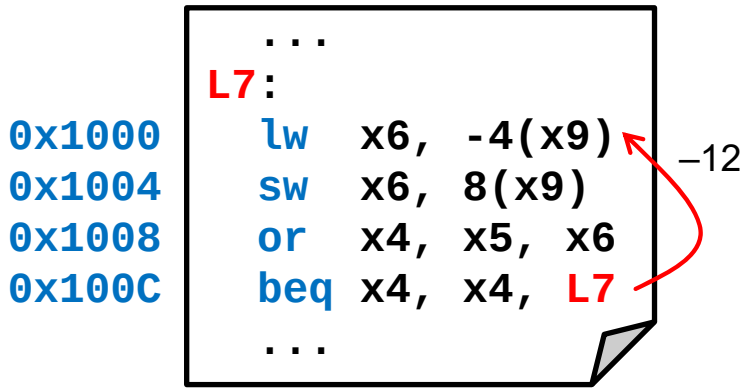
Dividing Time in B / Time in A:

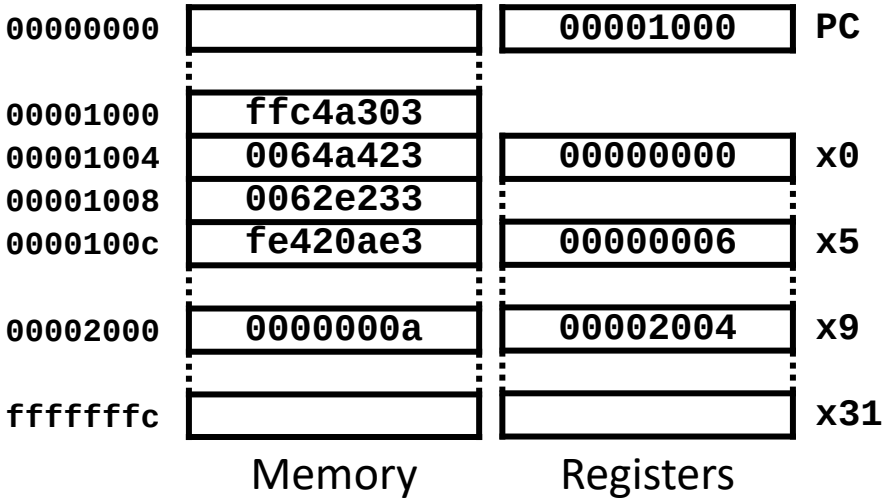
0.5 = [0.6xCPI(B) / f(B)] / [CPI(A) / f(A)] = 0.6 x CPI(B) x f(A) / CPI(A) x f(B)
CPI(B) = 0.5 x CPI(A) x f(B) / 0.6 x f(A) = $0.5x4.1x1.5x10^9$ / $0.6x1x10^9$ = 3.075 / 0.6 = **5.125**

9) Assume that the following program is running on a multicycle RISC-V, with the initial value of the memory and registers shown in the picture. Represent the execution diagrams for the registers and the memory, as well as for the control and status signals, so that their values are shown for each clock cycle.

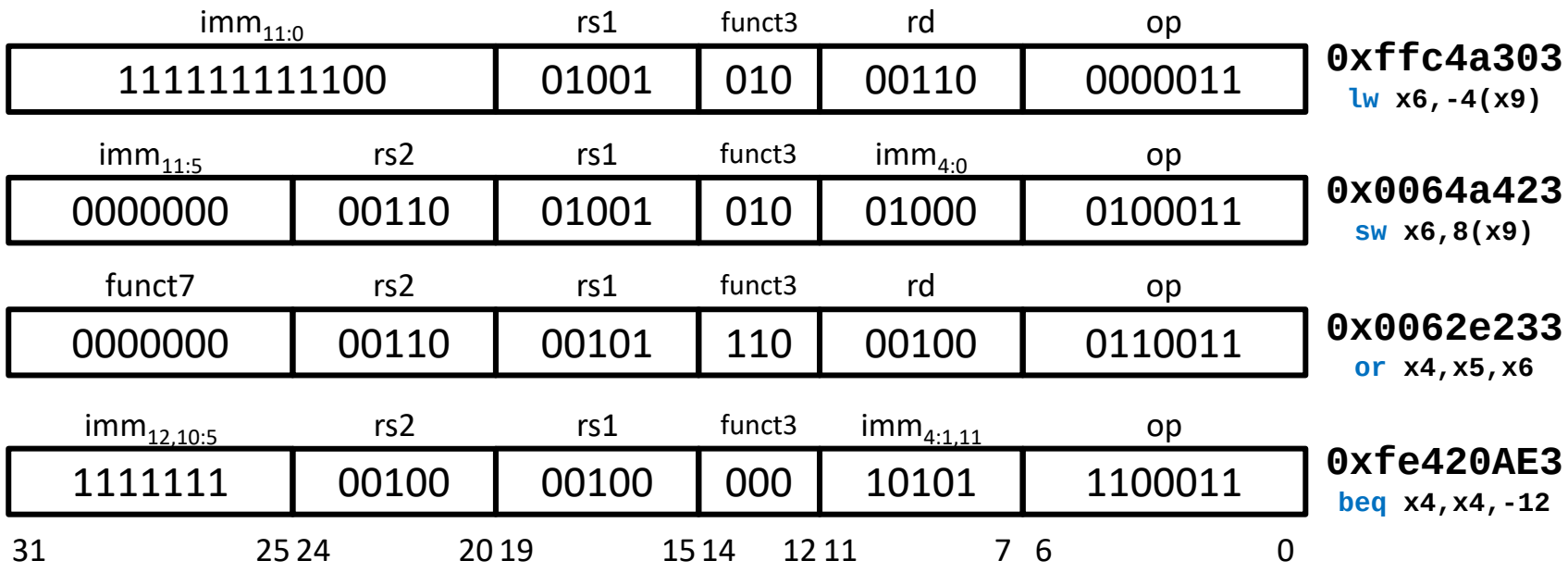








The machine code representation of the program instructions is the following:



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

−12

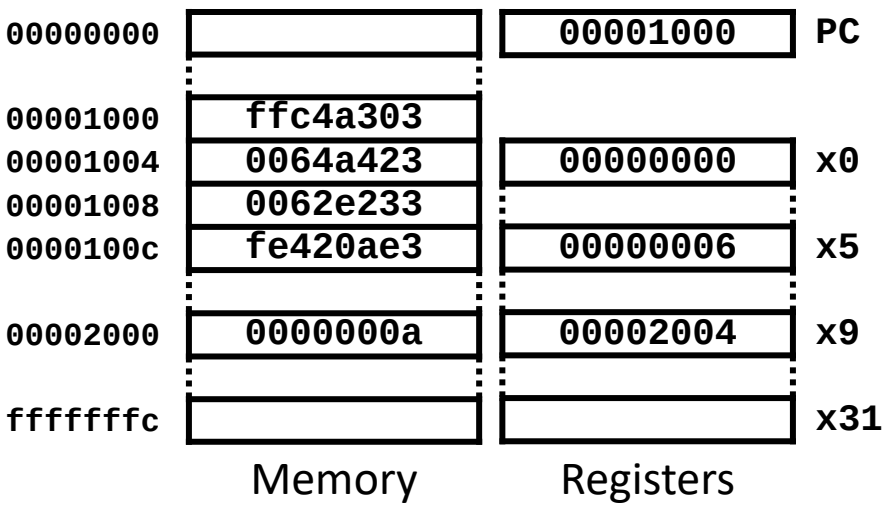
	Memory		Registers
00000000		00001000	PC
00001000	ffc4a303		
00001004	0064a423	00000000	x0
00001008	0062e233		
0000100c	fe420ae3	00000006	x5
00002000	0000000a	00002004	x9
fffffffc			x31

Execution diagram: registers and memory (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	S0								
PC	00001000								
OldPC	????????								
IR	????????								
MDR	????????								
A	????????								
B	????????								
ALUout	????????								
x4	????????								
x5	00000006								
x6	????????								
x9	00002004								
MEM[0x2000]	0000000a								
MEM[0x200c]	????????								
clk									
# cycle	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

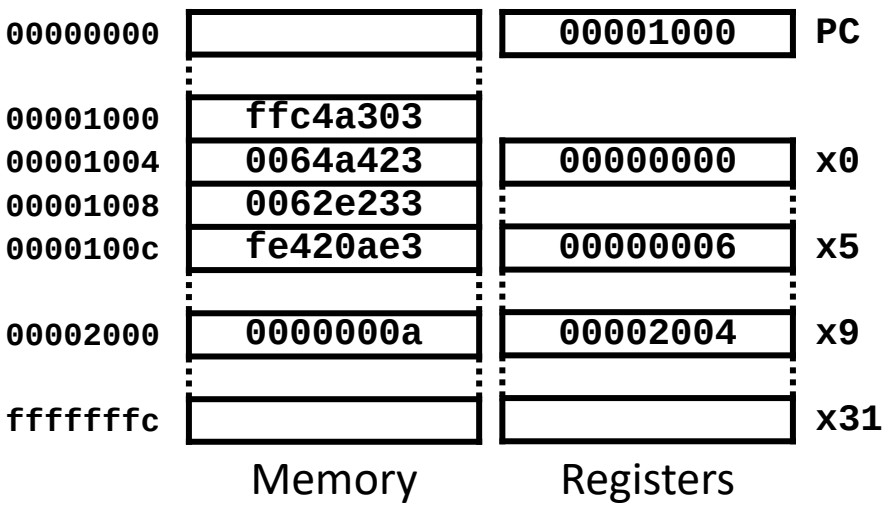


Execution diagram: registers and memory (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	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									
# cycle	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



Execution diagram: registers and memory (i)

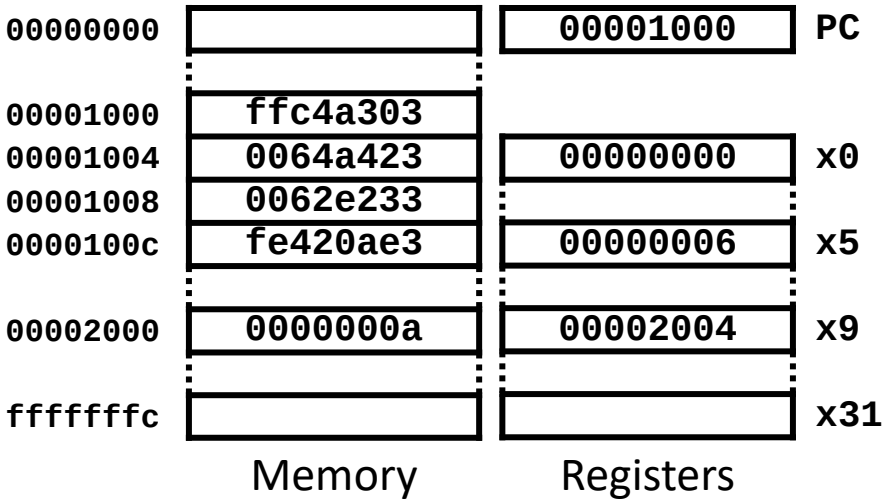
	lw x6, -4(x9)					sw x6, 8(x9)			
state	S0	S1	S2						
PC	00001000	00001004	=						
OldPC	????????	00001000	=						
IR	????????	ffc4a303	=						
MDR	????????	????????	=						
A	????????	????????	00002004						
B	????????	????????	garbage^1						
ALUout	????????	????????	garbage^2						
x4	????????	????????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

garbage^1 = RF[IR$_{24..20}$] = x28 = ???

garbage^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



Execution diagram: registers and memory (i)

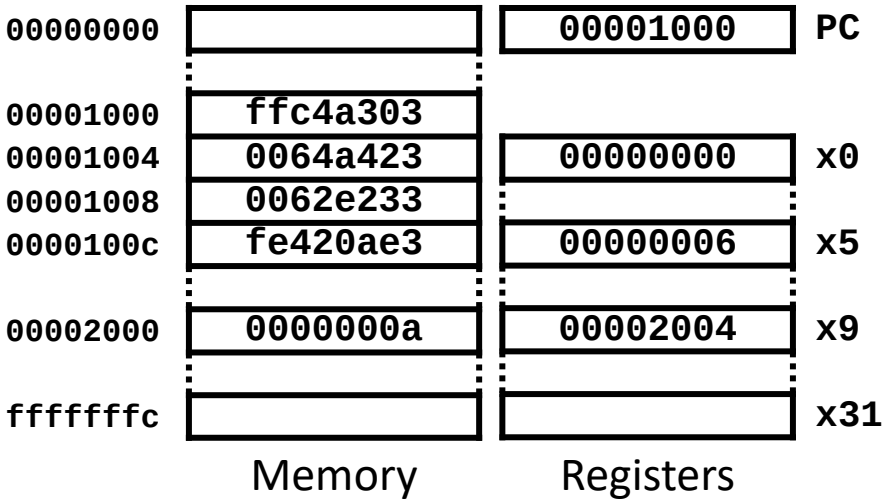
	lw x6, -4(x9)					sw x6, 8(x9)			
state	S0	S1	S2	S3					
PC	00001000	00001004	00001004	=					
OldPC	????????	00001000	00001000	=					
IR	????????	ffc4a303	ffc4a303	=					
MDR	????????	????????	????????	=					
A	????????	????????	00002004	=					
B	????????	????????	garbage[1]	=					
ALUout	????????	????????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

garbage[1] = RF[IR$_{24..20}$] = x28 = ???

garbage[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

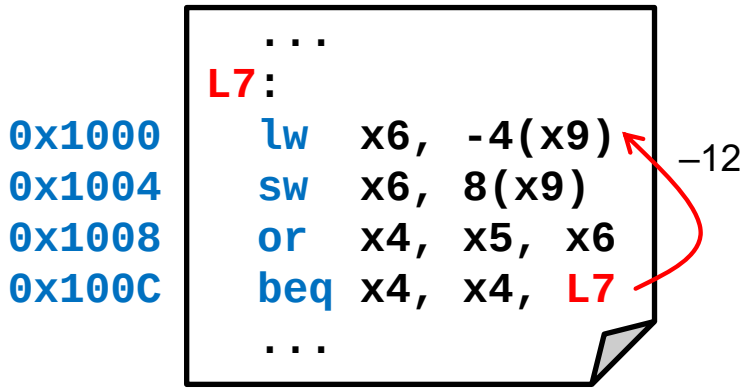


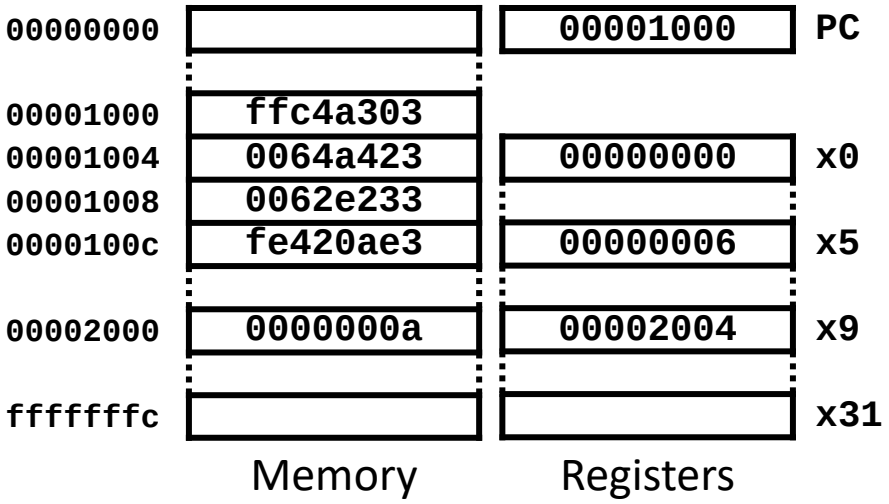
Execution diagram: registers and memory (i)

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

garbage[1] = RF[IR$_{24..20}$] = x28 = ???

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





Execution diagram: registers and memory (i)

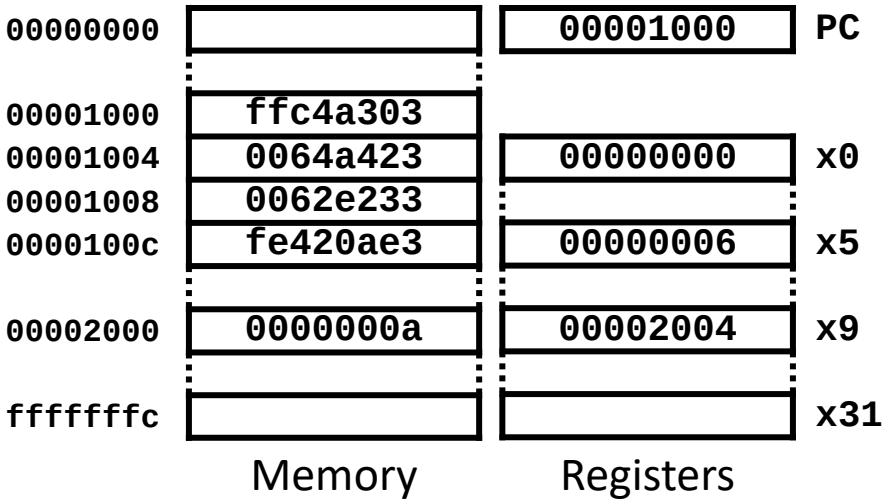
	lw x6, -4(x9)					sw x6, 8(x9)			
state	S0	S1	S2	S3	S4	S0			
PC	00001000	00001004	00001004	00001004	00001004	=			
OldPC	????????	00001000	00001000	00001000	00001000	=			
IR	????????	ffc4a303	ffc4a303	ffc4a303	ffc4a303	=			
MDR	????????	????????	????????	????????	0000000a	=			
A	????????	????????	00002004	00002004	00002004	=			
B	????????	????????	garbage[1]	garbage	garbage	=			
ALUout	????????	????????	garbage[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									
# cycle	1	2	3	4	5	6	7	8	9

garbage[1] = RF[IR$_{24..20}$] = x28 = ???

garbage[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

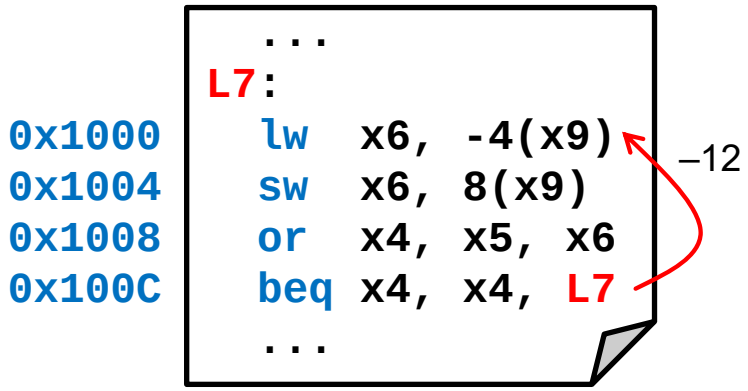


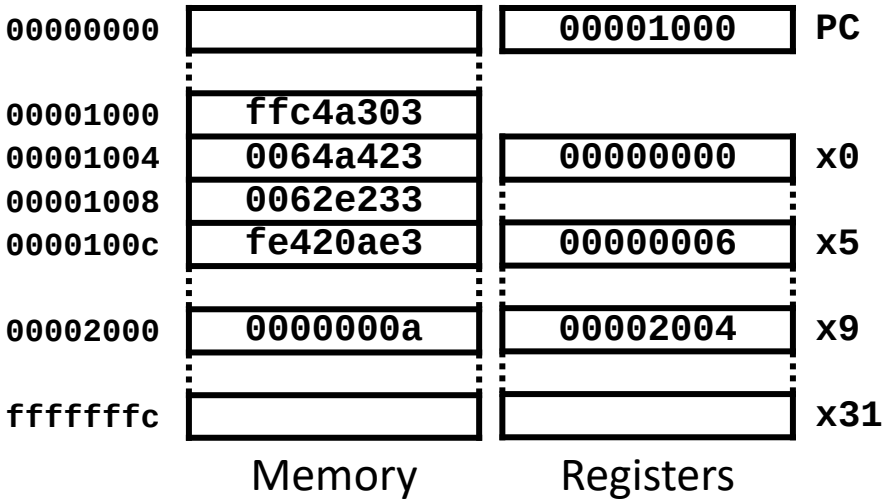
Execution diagram: registers and memory (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	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	????????	????????	garbage[1]	garbage	garbage	garbage	=		
ALUout	????????	????????	garbage[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									
# cycle	1	2	3	4	5	6	7	8	9

garbage[1] = RF[IR$_{24..20}$] = x28 = ???

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





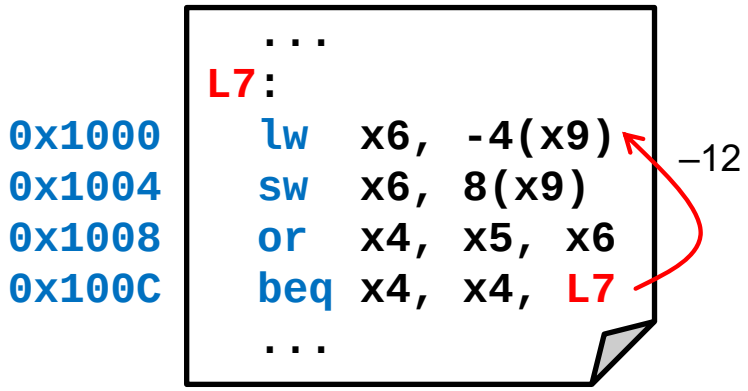
Execution diagram: registers and memory (i)

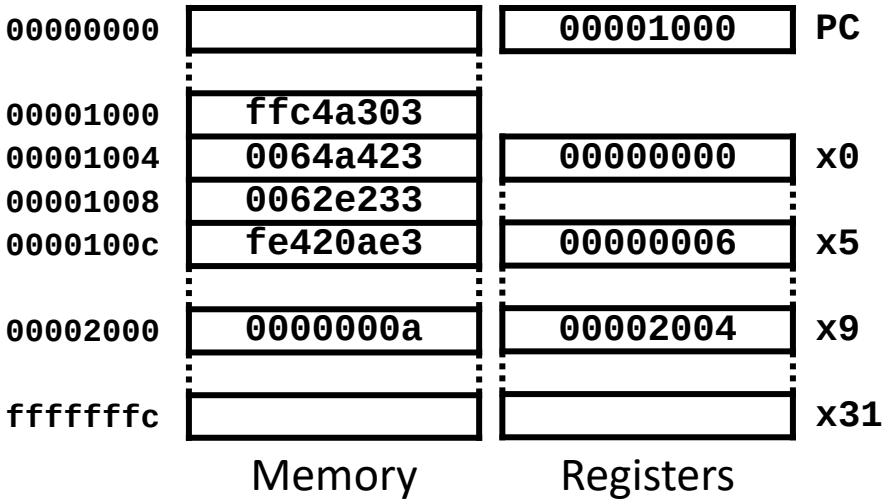
	lw x6, -4(x9)					sw x6, 8(x9)			
state	S0	S1	S2	S3	S4	S0	S1	S2	
PC	00001000	00001004	00001004	00001004	00001004	00001004	00001008	=	
OldPC	????????	00001000	00001000	00001000	00001000	00001000	00001004	=	
IR	????????	ffc4a303	ffc4a303	ffc4a303	ffc4a303	ffc4a303	0064a423	=	
MDR	????????	????????	????????	????????	0000000a	0000000a	0000000a	=	
A	????????	????????	00002004	00002004	00002004	00002004	00002004	00002004	
B	????????	????????	garbage[1]	garbage	garbage	garbage	garbage	0000000a	
ALUout	????????	????????	garbage[2]	00002000	00002000	00002000	00002000	garbage[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									
# cycle	1	2	3	4	5	6	7	8	9

garbage[1] = RF[IR$_{24..20}$] = x28 = ???

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

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





Execution diagram: registers and memory (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	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	????????	????????	garbage^1	garbage	garbage	garbage	garbage	0000000a	=
ALUout	????????	????????	garbage^2	00002000	00002000	00002000	00002000	garbage^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									
# cycle	1	2	3	4	5	6	7	8	9

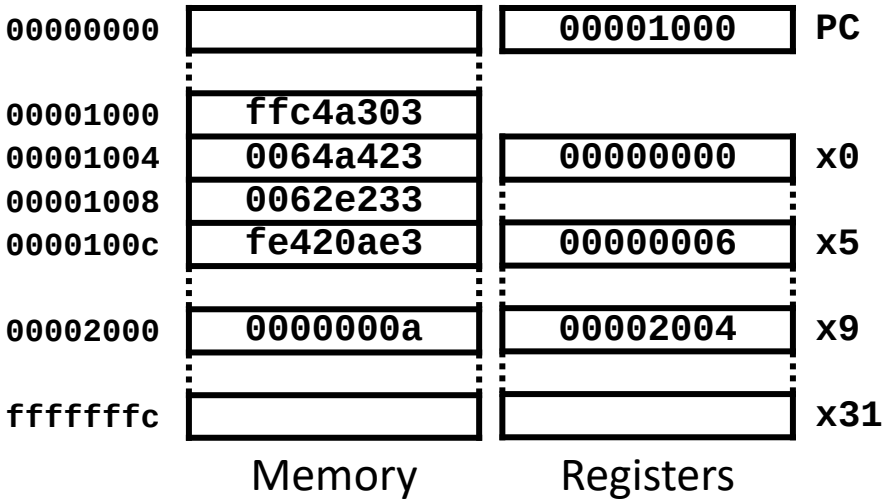
garbage^1 = RF[IR$_{24..20}$] = x28 = ???

garbage^2 = OldPC + SExt(imm) = 0x1000 + (-4) = 0x00000ffc

garbage^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



Execution diagram: registers and memory (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	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	????????	????????	garbage[1]	garbage	garbage	garbage	garbage	0000000a	0000000a
ALUout	????????	????????	garbage[2]	00002000	00002000	00002000	00002000	garbage[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									
# cycle	1	2	3	4	5	6	7	8	9

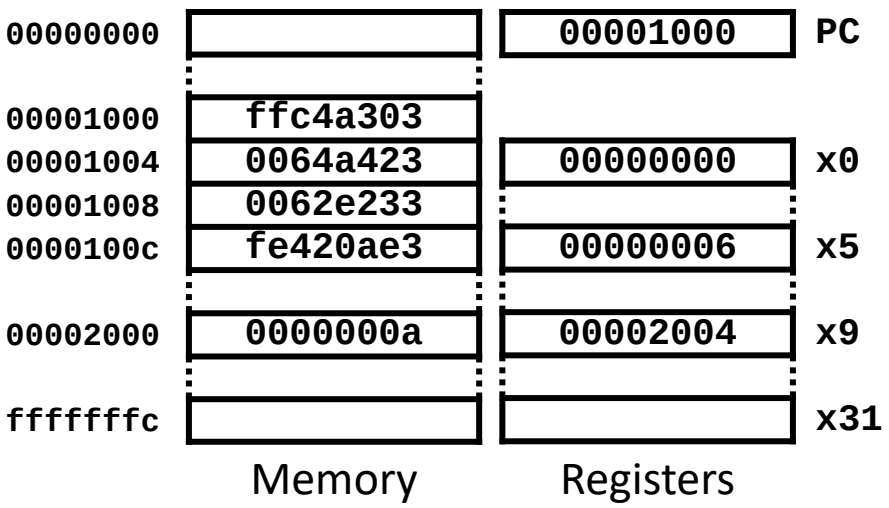
garbage[1] = RF[IR$_{24..20}$] = x28 = ???

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

garbage[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
          ...
```

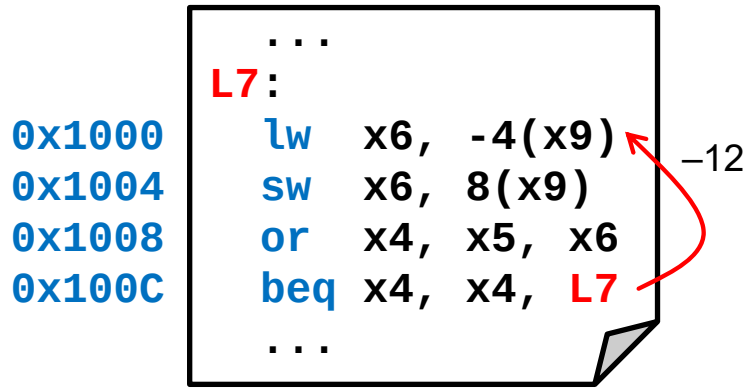
–12

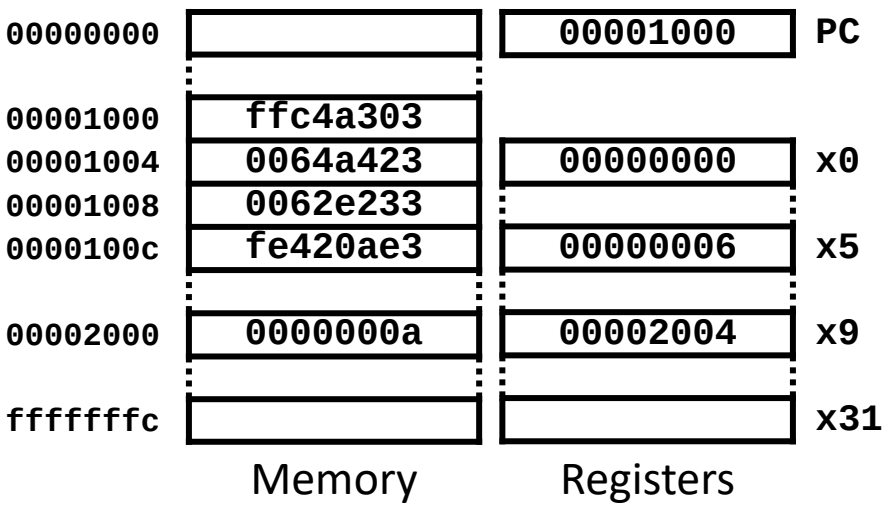


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	S0							
PC	=							
OldPC	=							
IR	=							
MDR	=							
A	=							
B	=							
ALUout	=							
x4	=							
x5	=							
x6	=							
x9	=							
MEM[0x2000]	=							
MEM[0x200c]	0000000a							
clk								
# cycle	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
          ...
```



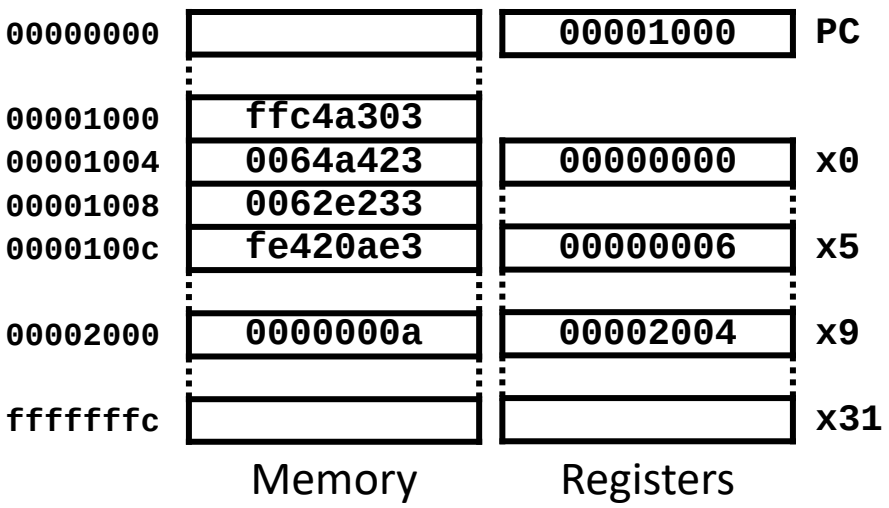


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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								
# cycle	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
          ...
```

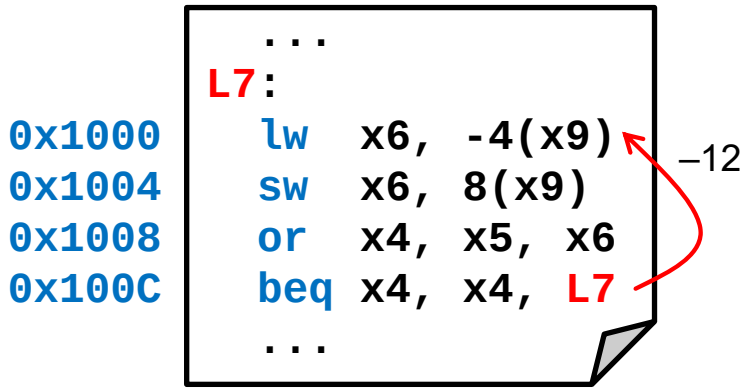
–12

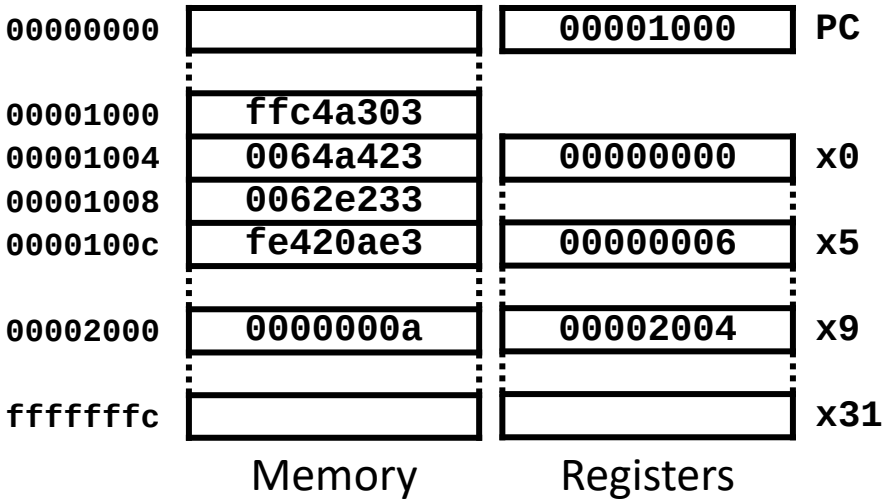


Execution diagram: registers and memory (ii)

	or x4, x5, x6			beq x4, x4, -12			...
state	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	garbage[4]				
x4	????????	????????	=				
x5	00000006	00000006	=				
x6	0000000a	0000000a	=				
x9	00002004	00002004	=				
MEM[0x2000]	0000000a	0000000a	=				
MEM[0x200c]	0000000a	0000000a	=				
clk							
# cycle	10	11	12	13	14	15	16

garbage[4] = depends on the sExt DEC implementation

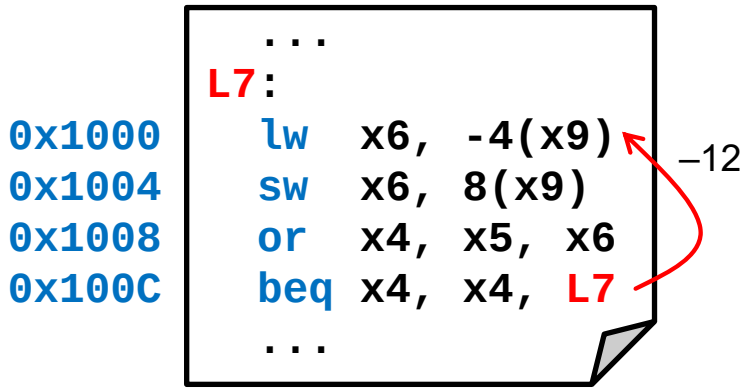


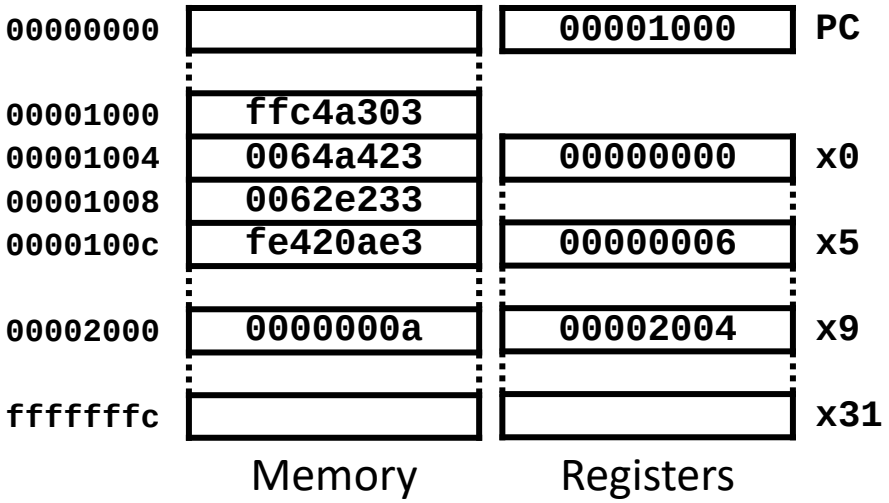


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation

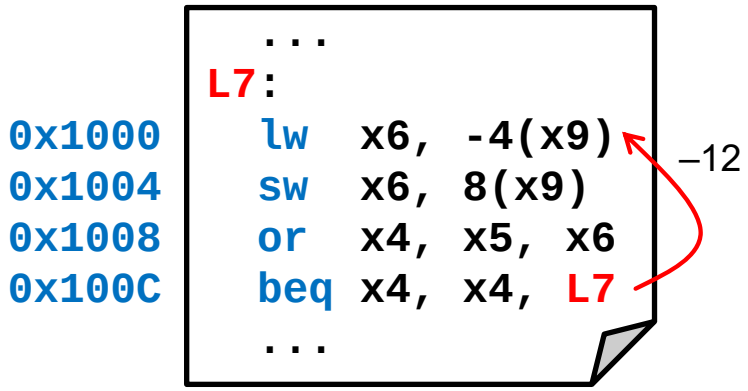


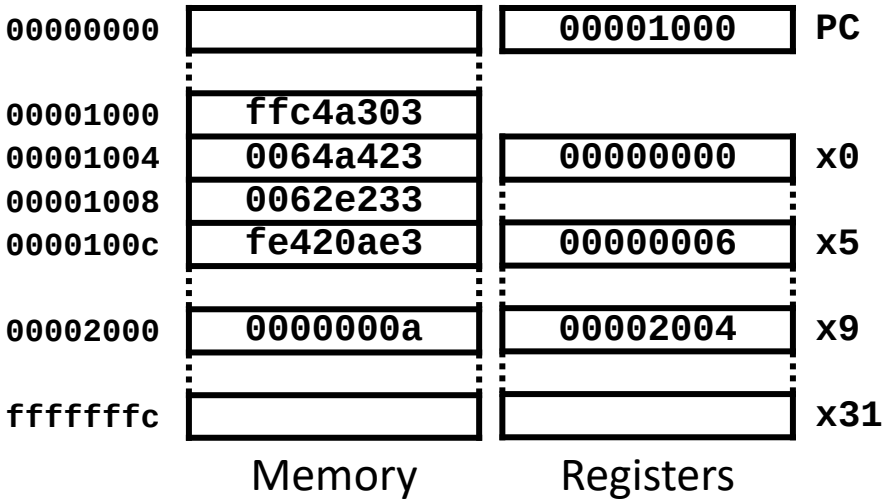


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation

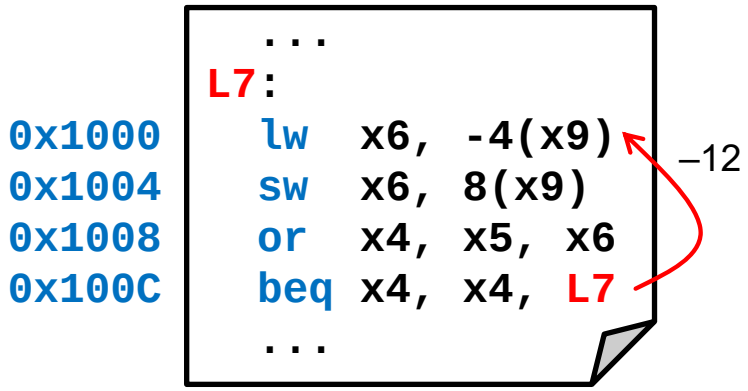


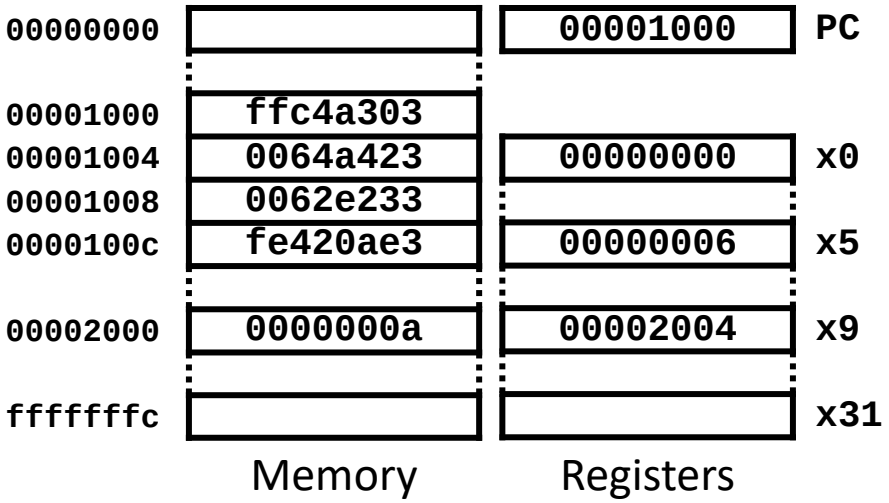


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation

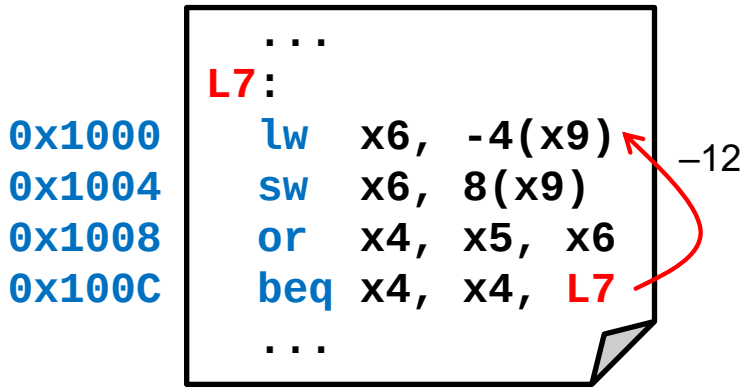


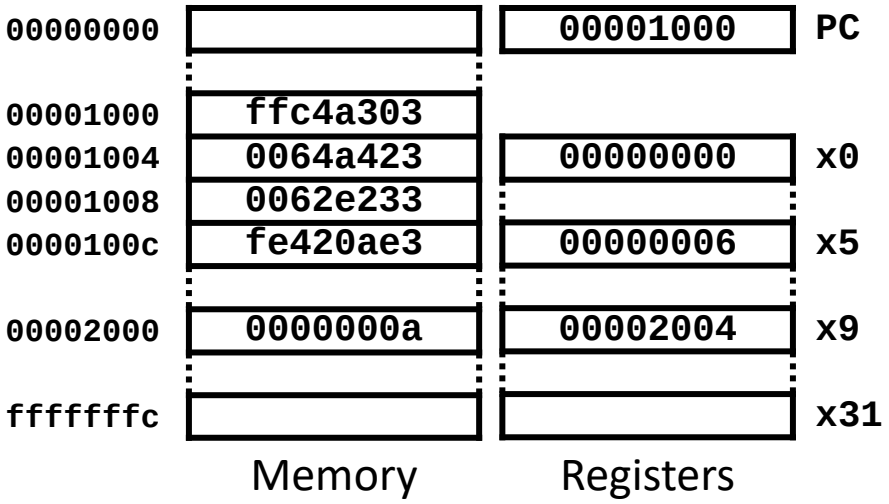


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation

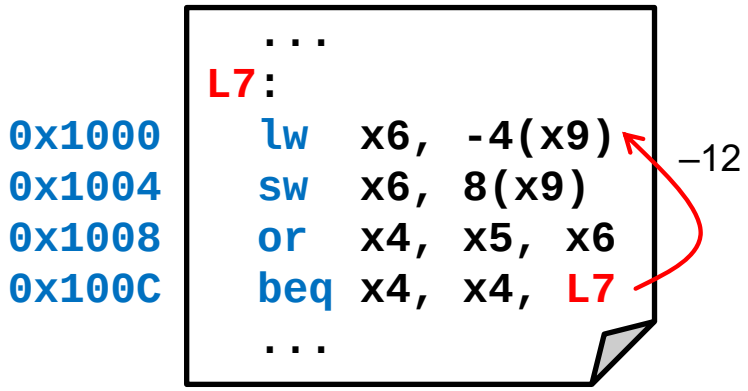


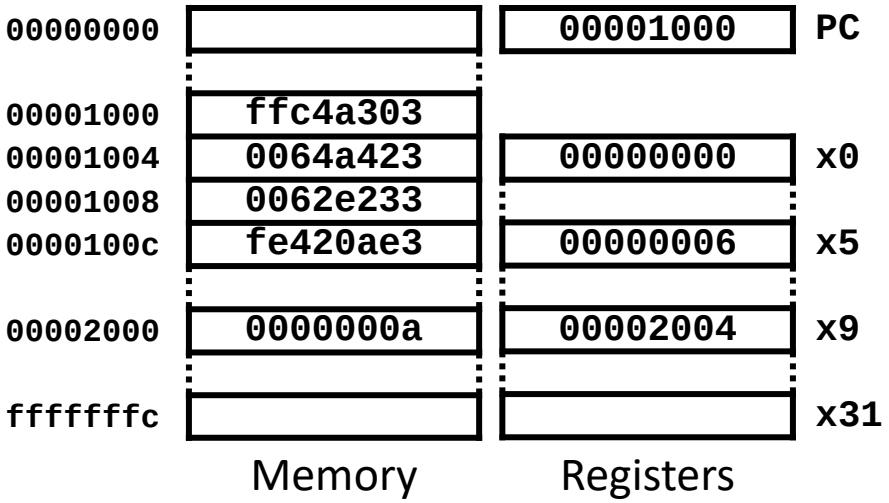


Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation





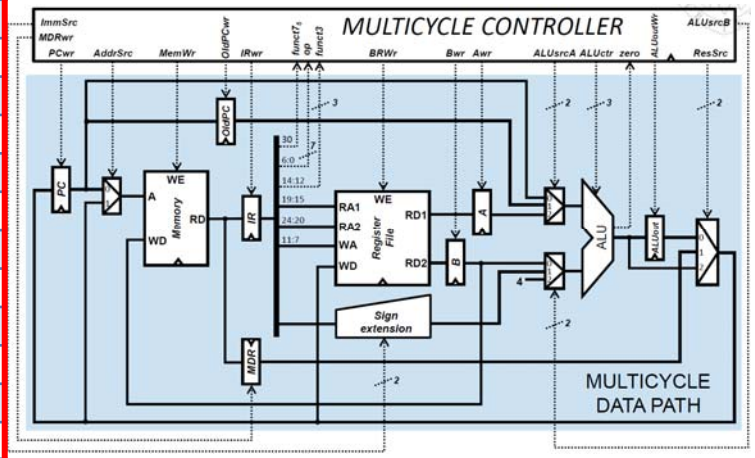
Execution diagram: registers and memory (ii)

	or x4, x5, x6				beq x4, x4, -12			...
state	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	garbage[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								
# cycle	10	11	12	13	14	15	16	

garbage[4] = depends on the sExt DEC implementation

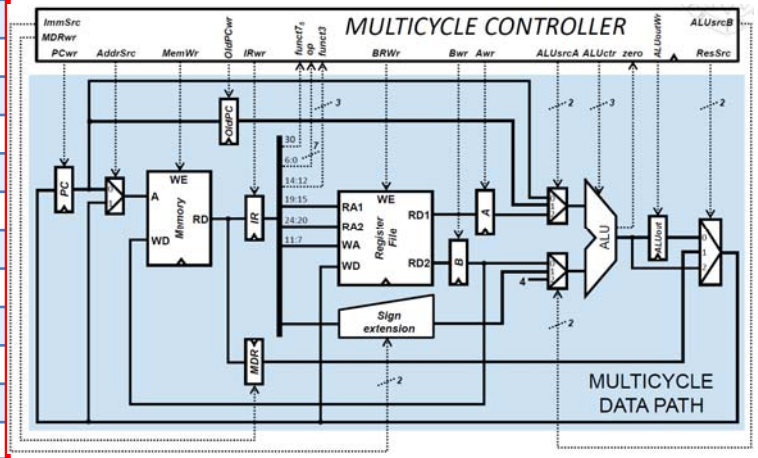
Execution diagram: status and control signals (i)

	`lw x6, -4(x9)`					`sw x6, 8(x9)`			
state	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									
# cycle	1	2	3	4	5	6	7	8	9



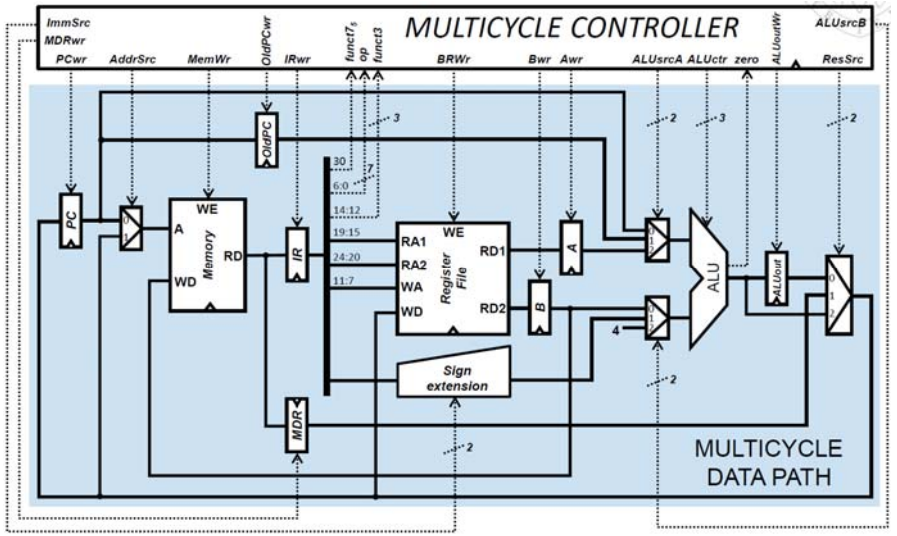
Execution diagram: status and control signals (i)

	lw x6, -4(x9)					sw x6, 8(x9)			
state	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									
# cycle	1	2	3	4	5	6	7	8	9



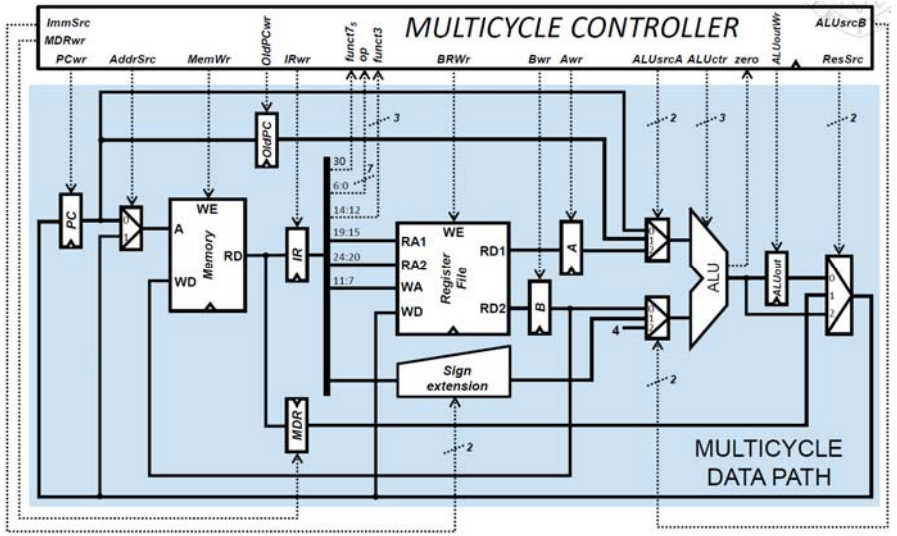
Execution diagram: status and control signals (ii)

	or x4, x5, x6				beq x4, x4, -12			
state	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								
# cycle	10	11	12	13	14	15	16	17



Execution diagram: status and control signals (ii)

	or x4, x5, x6				beq x4, x4, -12			
state	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								
# cycle	10	11	12	13	14	15	16	17



10) Assume that the following program is running on a multicycle RISC-V, with the initial value of the memory and registers shown in the picture. Represent the execution diagrams for the registers and the memory, as well as for the control and status signals, so that their values are shown for each clock cycle.

C/C++

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

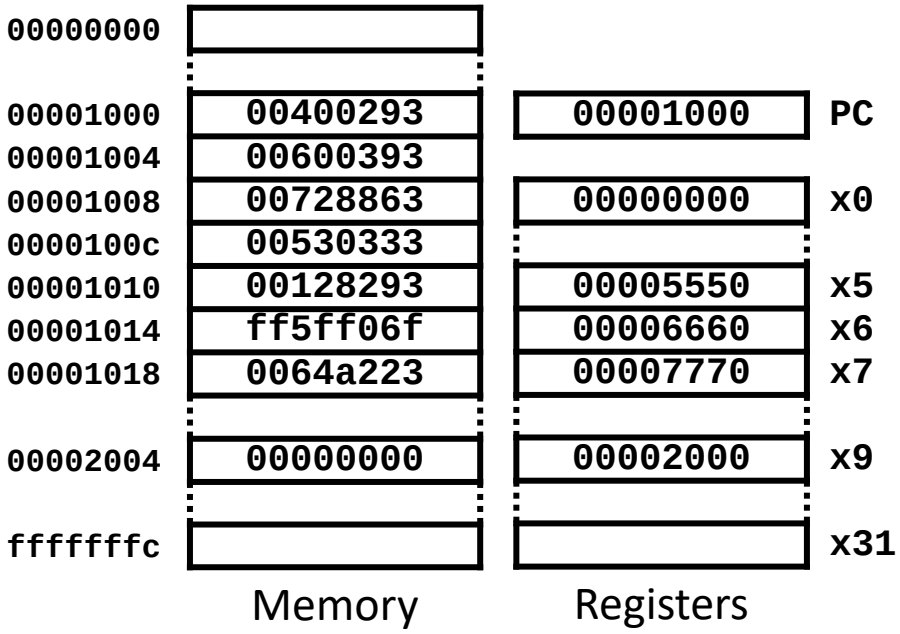
i → x5
a → x6

ASM

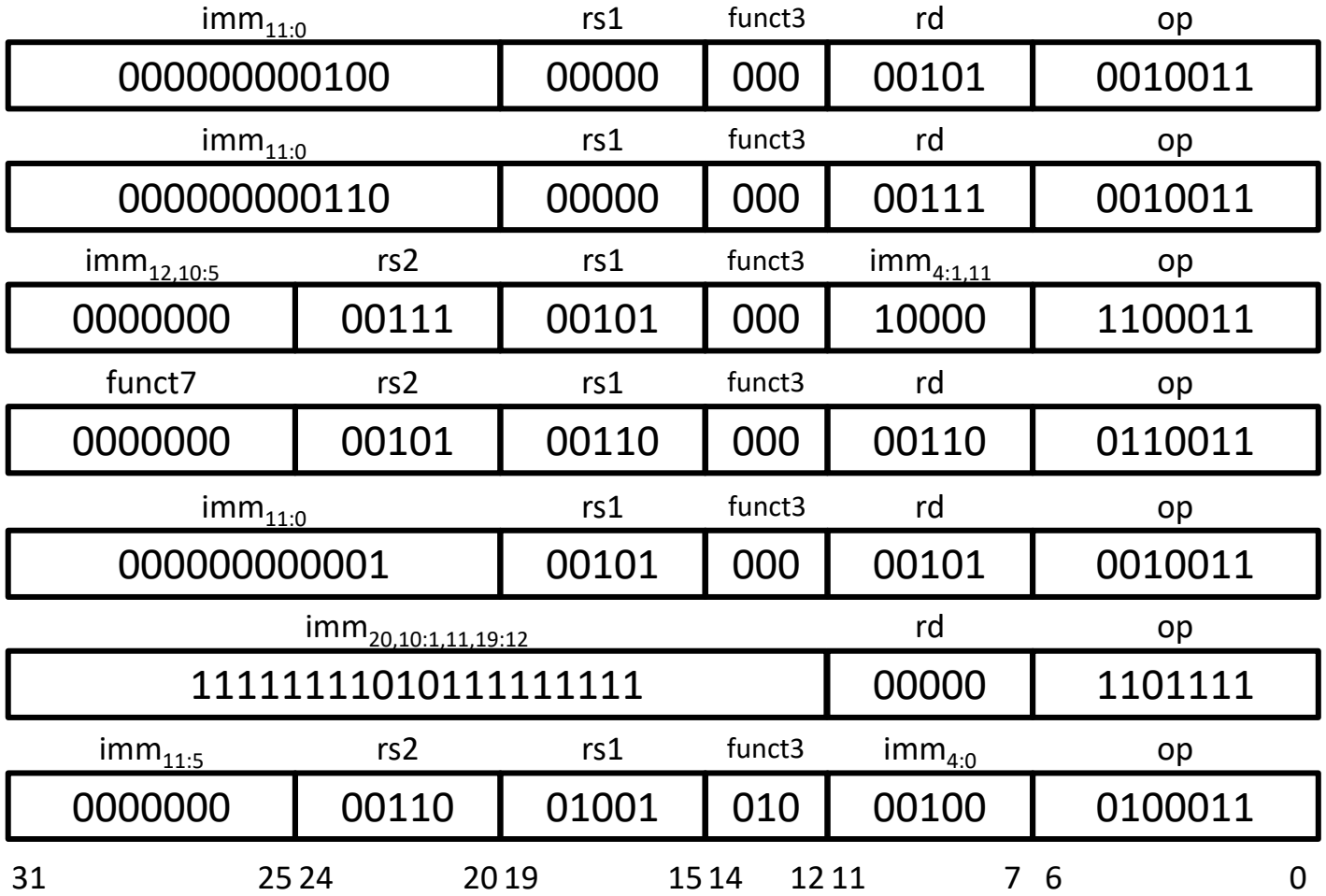
```
...
0x1000   addi x5, x0, 4
0x1004   addi x7, x0, 6
for:
0x1008   beq  x5, x7, efor
0x100c   add  x6, x6, x5
0x1010   addi x5, x5, 1
0x1014   jal  x0, for
efor:
0x1018   sw   x6, 4(x9)
...
```

+16

–12



The machine code representation of the program instructions is the following:



0x00400293
`addi x5,x0,4`

0x00600393
`addi x7,x0,6`

0x00728863
`beq x5,x7,16`

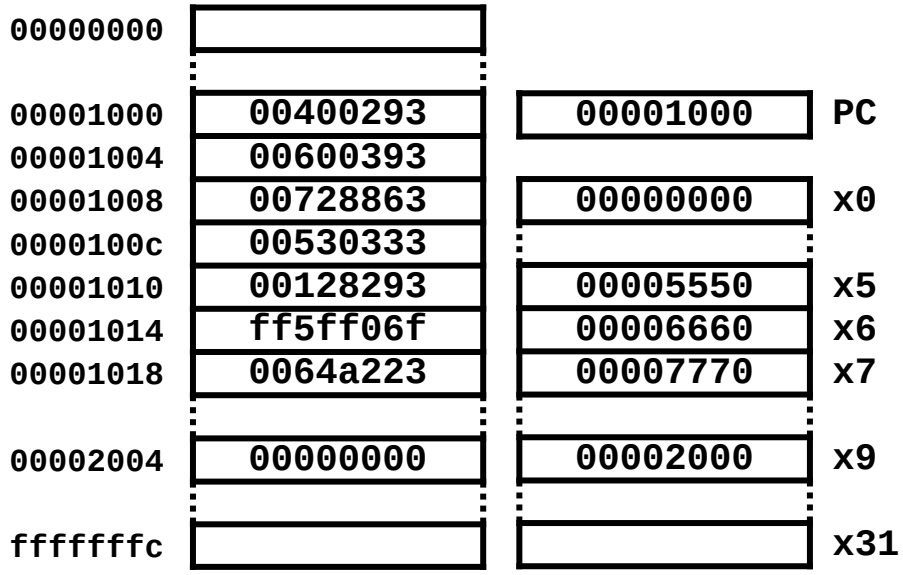
0x00530333
`add x6,x6,x5`

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)
            ...
```



Execution diagram: registers and memory (i)

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

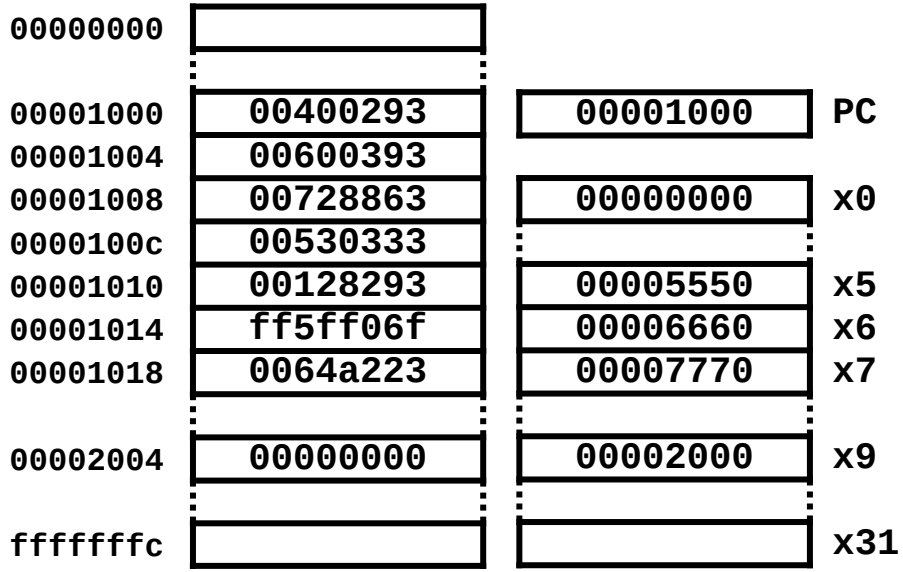
garbage^1 = RF[IR$_{24..20}$] = x4 = ???

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

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

garbage^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)
            ...
```



Execution diagram: registers and memory (i)

	addi x5, x0, 4				addi x7, x0, 6			
state	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	????????	????????	garbage^1	garbage	garbage	garbage	garbage^3	garbage
ALUout	????????	????????	garbage^2	00000004	00000004	00000004	garbage^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								
# cycle	1	2	3	4	5	6	7	8

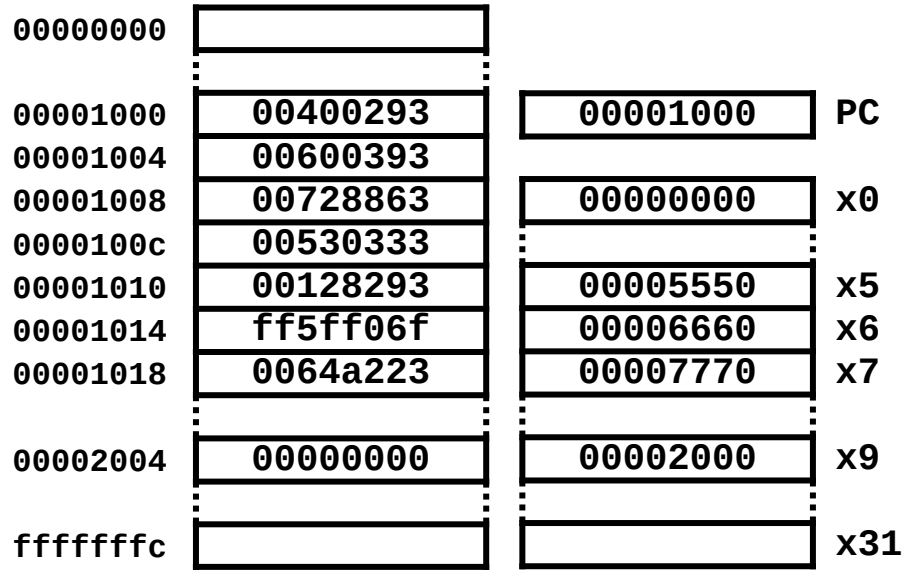
$garbage^1 = RF[IR_{24..20}] = x4 = ???$

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

$garbage^3 = RF[IR_{24..20}] = x6 = 0x00006660$

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

```
...
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)
...
```

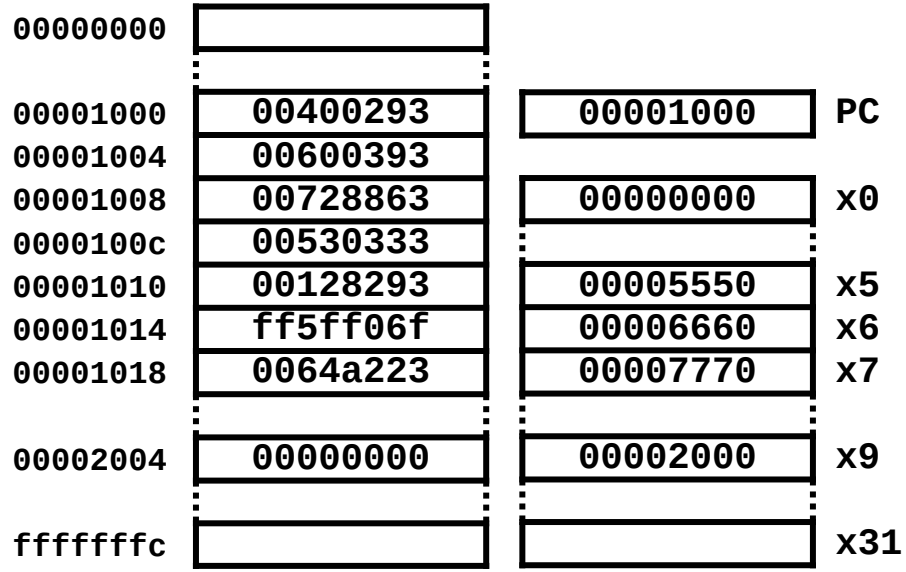


Execution diagram: registers and memory (ii)

	beq x5, x7, 16			add x6, x6, x5			
state	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	garbage	=	00000006	=	=	00000004	=
ALUout	00000006	=	00001018	=	=	garbage[5]	00006664
x5	00000004	=	=	=	=	=	=
x6	00006660	=	=	=	=	=	=
x7	00000006	=	=	=	=	=	=
x9	00002000	=	=	=	=	=	=
MEM[0x2004]	00000000	=	=	=	=	=	=
clk							
# cycle	9	10	11	12	13	14	15

garbage[5] = depends on the sExt DEC implementation

```
...
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)
...
```

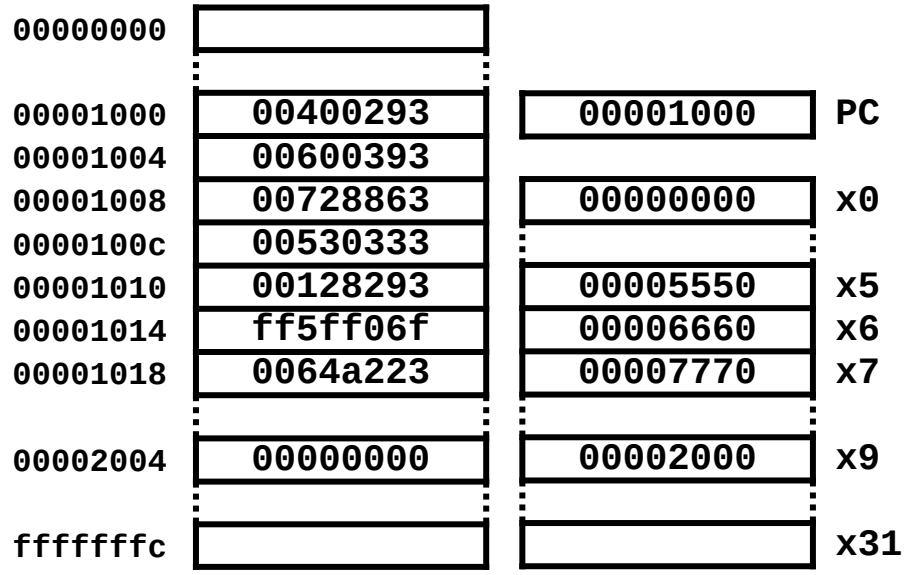


Execution diagram: registers and memory (ii)

	beq x5, x7, 16			add x6, x6, x5			
state	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	garbage	garbage	00000006	00000006	00000006	00000004	00000004
ALUout	00000006	00000006	00001018	00001018	00001018	garbage[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							
# cycle	9	10	11	12	13	14	15

garbage[5] = depends on the sExt DEC implementation

```
...
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)
...
```



Execution diagram: registers and memory (iii)

	addi x5, x5, 1				jal x0, -12			
state	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	=	=	=	garbage^{8}	=
B	00000004	=	garbage^{6}	=	=	=	garbage^{9}	=
ALUout	00006664	=	garbage^{7}	00000005	=	=	00001008	00001018
x5	00000004	=	=	=	00000005	=	=	=
x6	00006664	=	=	=	=	=	=	=
x7	00000006	=	=	=	=	=	=	=
x9	00002000	=	=	=	=	=	=	=
MEM[0x2004]	00000000	=	=	=	=	=	=	=
clk								
# cycle	16	17	18	19	20	21	22	23

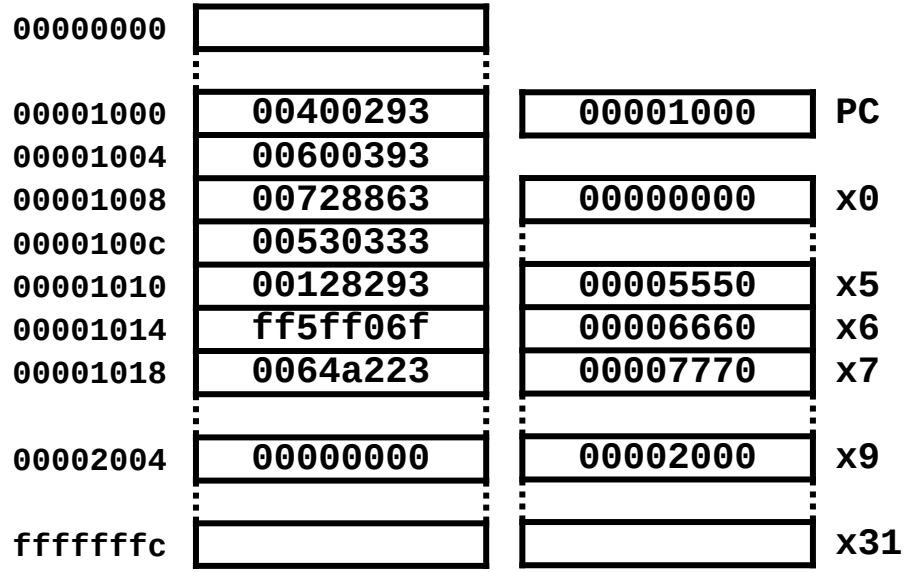
garbage^{6} = RF[IR$_{24..20}$]
= x21 = ???

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

garbage^{8} = RF[IR$_{19..15}$]
= x31 = ???

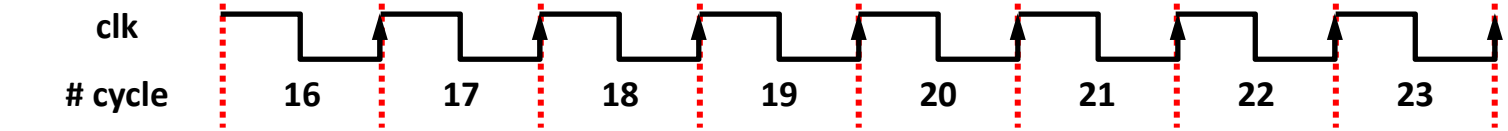
garbage^{9} = RF[IR$_{24..20}$]
= 0x1004 + 6 = 0x0000100a

```
...
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)
...
```



Execution diagram: registers and memory (iii)

	addi x5, x5, 1				jal x0, -12			
state	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	garbage^{8}	garbage
B	00000004	00000004	garbage^{6}	garbage	garbage	garbage	garbage^{9}	garbage
ALUout	00006664	00006664	garbage^{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



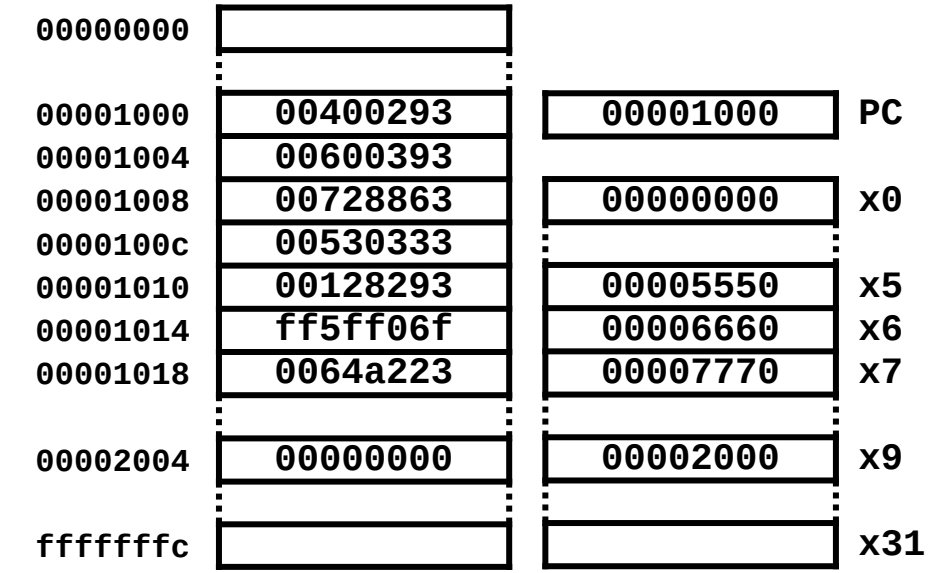
garbage^{6} = RF[IR$_{24..20}$]
= x21 = ???

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

garbage^{8} = RF[IR$_{19..15}$]
= x31 = ???

garbage^{9} = RF[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)
...
```

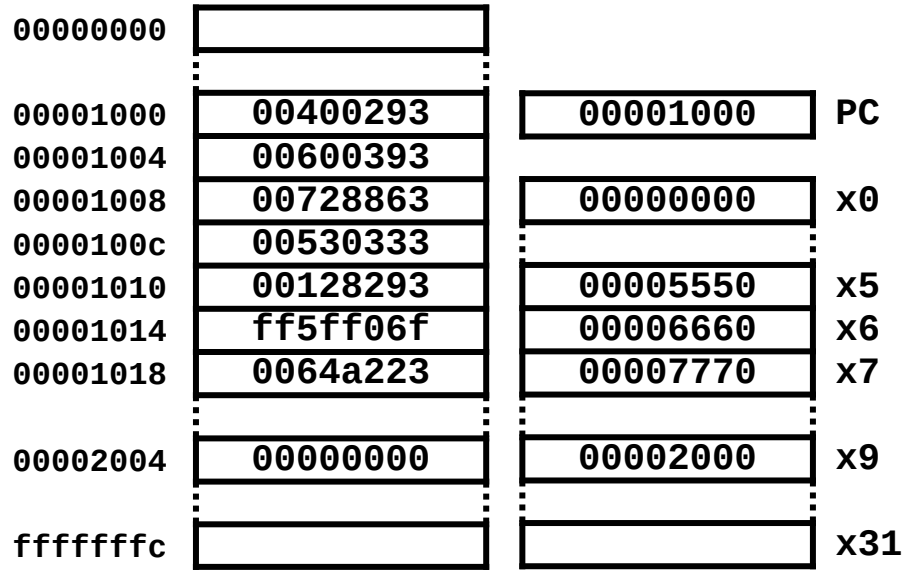


Execution diagram: registers and memory (iv)

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

garbage^A = depends on the sExt DEC implementation

```
...
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)
...
```

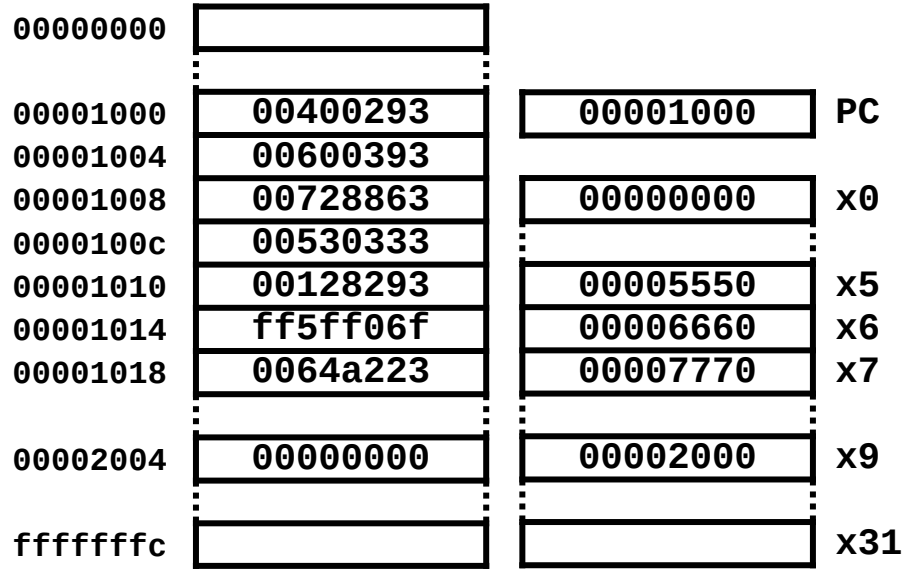


Execution diagram: registers and memory (iv)

	beq x5, x7, 16			add x6, x6, x5			
state	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	garbage	garbage	00000005	00000005	00000005	00006664	00006664
B	garbage	garbage	00000006	00000006	00000006	00000005	00000005
ALUout	00001018	00001018	00001018	00001018	00001018	garbage^A	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							
# cycle	24	25	26	27	28	29	30

garbage^A = depends on the sExt DEC implementation

```
...
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)
...
```



Execution diagram: registers and memory (v)

	addi x5, x5, 1				jal x0, -12			
state	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	=	=	=	$garbage^D$	=
B	00000005	=	$garbage^B$	=	=	=	$garbage^E$	=
ALUout	00006669	=	$garbage^C$	00000006	=	=	00001008	00001018
x5	00000005	=	=	=	00000006	=	=	=
x6	00006669	=	=	=	=	=	=	=
x7	00000006	=	=	=	=	=	=	=
x9	00002000	=	=	=	=	=	=	=
MEM[0x2004]	00000000	=	=	=	=	=	=	=
clk								
# cycle	31	32	33	34	35	36	37	38

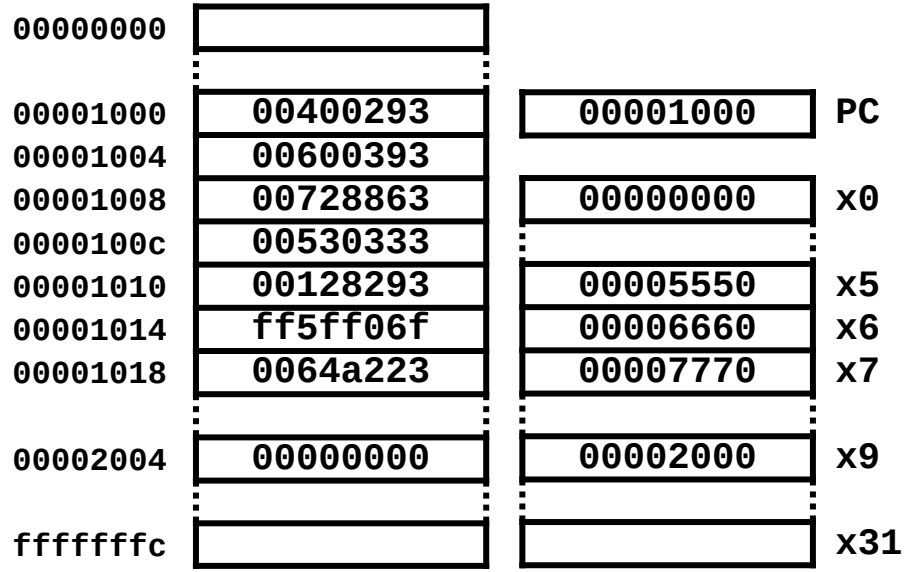
$garbage^B$ = $RF[IR_{24..20}]$ = x21 = ???

$garbage^C$ = OldPC + SExt(imm) = 0x1010 + 1 = 0x00001011

$garbage^D$ = $RF[IR_{19..15}]$ = x31 = ???

$garbage^E$ = $RF[IR_{24..20}]$ = 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)
...
```



Execution diagram: registers and memory (v)

	addi x5, x5, 1				jal x0, -12			
state	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	garbage^D	garbage
B	00000005	00000005	garbage^B	garbage	garbage	garbage	garbage^E	garbage
ALUout	00006669	00006669	garbage^C	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								
# cycle	31	32	33	34	35	36	37	38

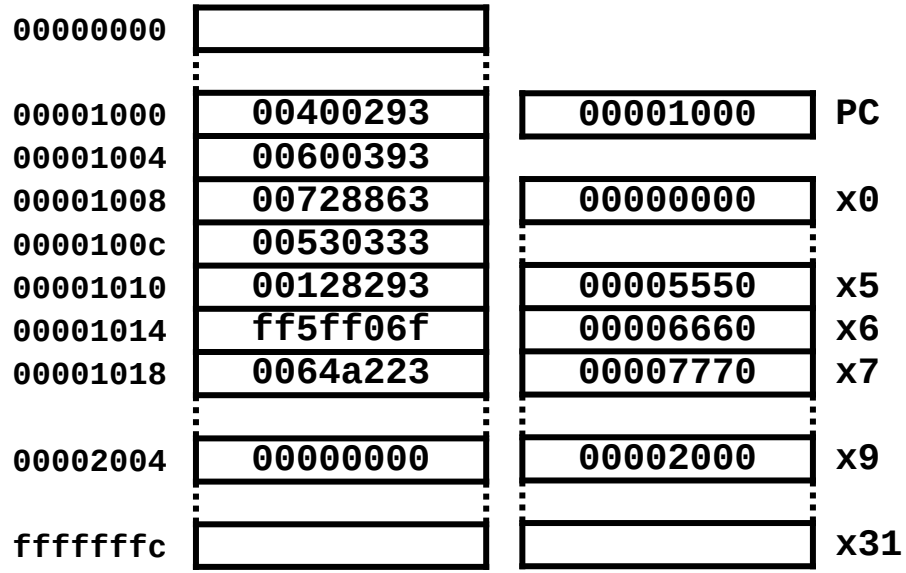
$garbage^B = RF[IR_{24..20}] = x21 = ???$

$garbage^C = OldPC + SExt(imm) = 0x1010 + 1 = 0x00001011$

$garbage^D = RF[IR_{19..15}] = x31 = ???$

$garbage^E = RF[IR_{24..20}] = 0x1004 + 6 = 0x0000100a$

```
...
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)
...
```

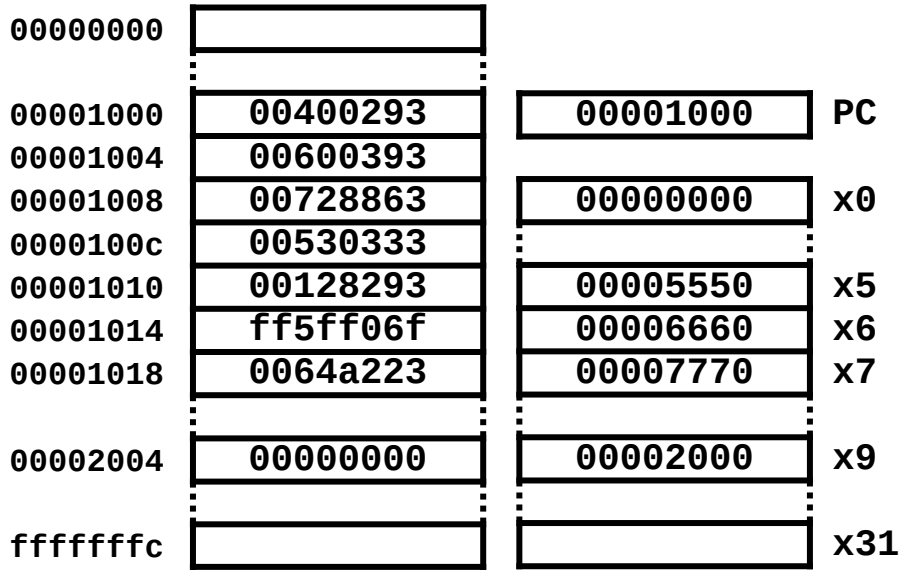


Execution diagram: registers and memory (vi)

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

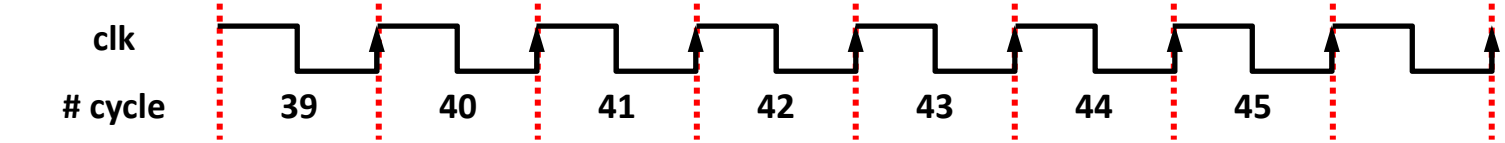
garbage^F = OldPC + SExt(imm) =
0x1018 + 4 = 0x0000101c

```
...
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)
...
```



Execution diagram: registers and memory (vi)

	beq x5, x7, 16			sw x6, 4(x9)				...
state	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	garbage	garbage	00000006	00000006	00000006	00002000	00002000	00002000
B	garbage	garbage	00000006	00000006	00000006	00006669	00006669	00006669
ALUout	00001018	00001018	00001018	00001018	00001018	garbage^F	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



$garbage^F$ = OldPC + SExt(imm) = 0x1018 + 4 = 0x0000101c

24) (Adapted September 2016) The program fragment shown below is running on a multicycle RISC-V, which has a 1 GHz frequency, producing an execution time of 3,474 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:
 ... (rest of the program)
```

a) Calculate the CPI of this program fragment.
b) Calculate the value of the MIPS metric.

f = 1 GHz = 10^9 cycles/s T = 3474 ns = 3,474x10^{-9} s

The loop has 6 instructions, and it is executed from s4=127 down to 0 (i.e. 128 times)

a) T = NI x CPI / f → CPI = T x f / NI
In this code: NI = 4 + 6x128 = 772 instructions. Thus, **CPI =** (3,474x10^{-9} s x 10^9 cycles/s) / 772 instr. = **4.5**

b) **MIPS =** f / (CPIx10^6) = 10^9 / (4.5x10^6) = **222.2**

About *Creative Commons*

- CC license (Creative Commons)

  - This license enables reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator. If you remix, adapt, or build upon the material, you must license the modified material under identical terms:

Attribution:
Credit must be given to the creator.

Non commercial:
Only noncommercial uses of the work are permitted.

Share alike:
Adaptations must be shared under the same terms.

More information: https://creativecommons.org/licenses/by-nc-sa/4.0/