

MIPS project

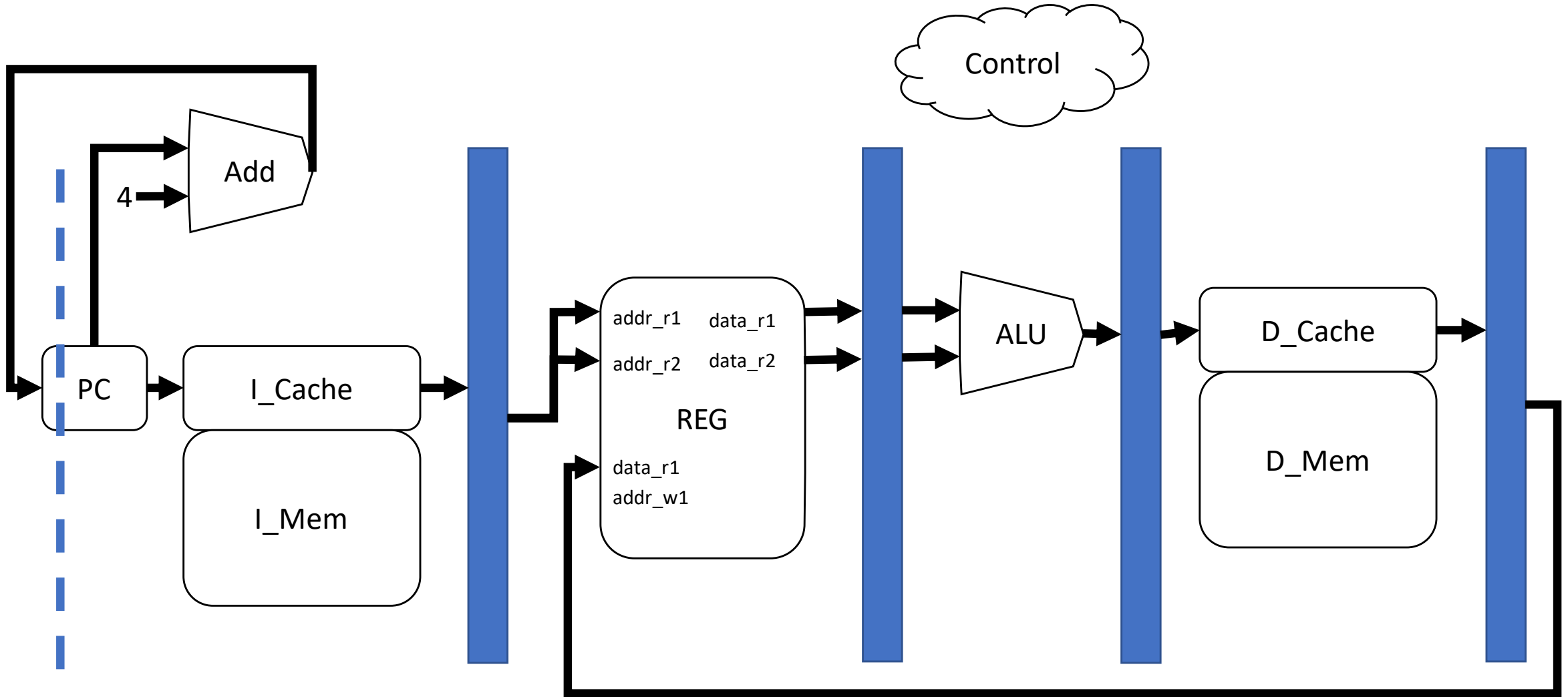
Outline

- Design Specification
- Hardware
 - Cache Implementation
 - MIPS Implementation
 - Instructions
 - Hazard Handling
 - Performance

Instructions supported

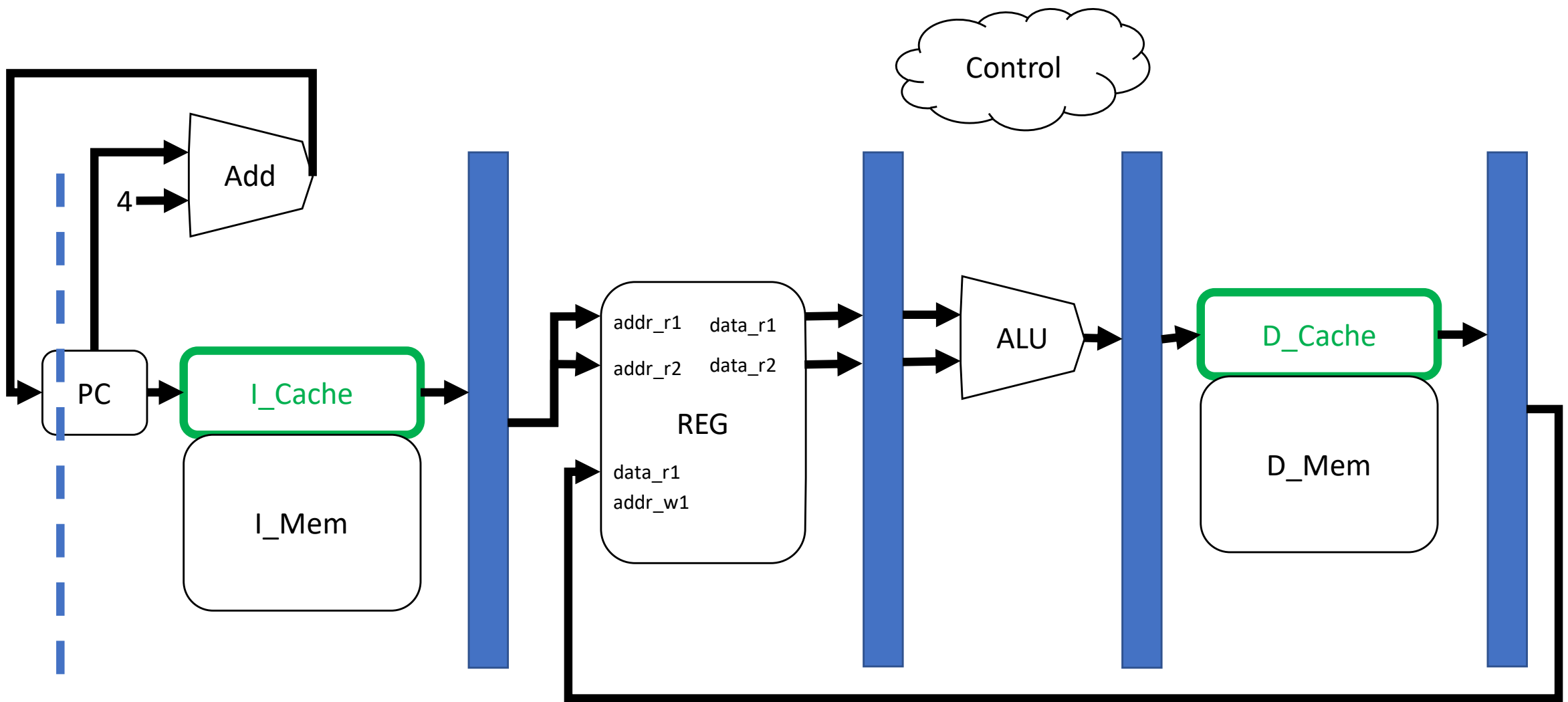
Name	Description
ADD	Addition, overflow detection for signed operand is not required*
ADDI	Addition immediate with sign-extension, without overflow detection*
SUB	Subtract, overflow detection for signed operand is not required*
AND	Boolean logic operation
ANDI	Boolean logic operation, zero-extension for upper 16bit of immediate
OR	Boolean logic operation
ORI	Boolean logic operation, zero-extension for upper 16bit of immediate
XOR	Boolean logic operation
XORI	Boolean logic operation, zero-extension for upper 16bit of immediate
NOR	Boolean logic operation
SLL	Shift left logical (zero padding)
SRA	Shift right arithmetic (sign-digit padding)
SRL	Shift right logical (zero padding)
SLT	Set less than, comparison instruction
SLTI	Set less than variable, comparison instruction
BEQ	Branch on equal, conditional branch instruction
J	Unconditionally jump
JAL	Unconditionally jump and link (Save next PC in \$r31)
JR	Unconditionally jump to the instruction whose address is in \$rs
JALR	Jump and link register
LW	Load word from data memory (assign word-aligned)
SW	Store word to data memory (assign word-aligned)
NOP	No operation

Pipeline Stages



Outline

- Design Specification
- Hardware
 - Cache Implementation
 - MIPS Implementation
 - Instructions
 - Hazard Handling
 - Performance



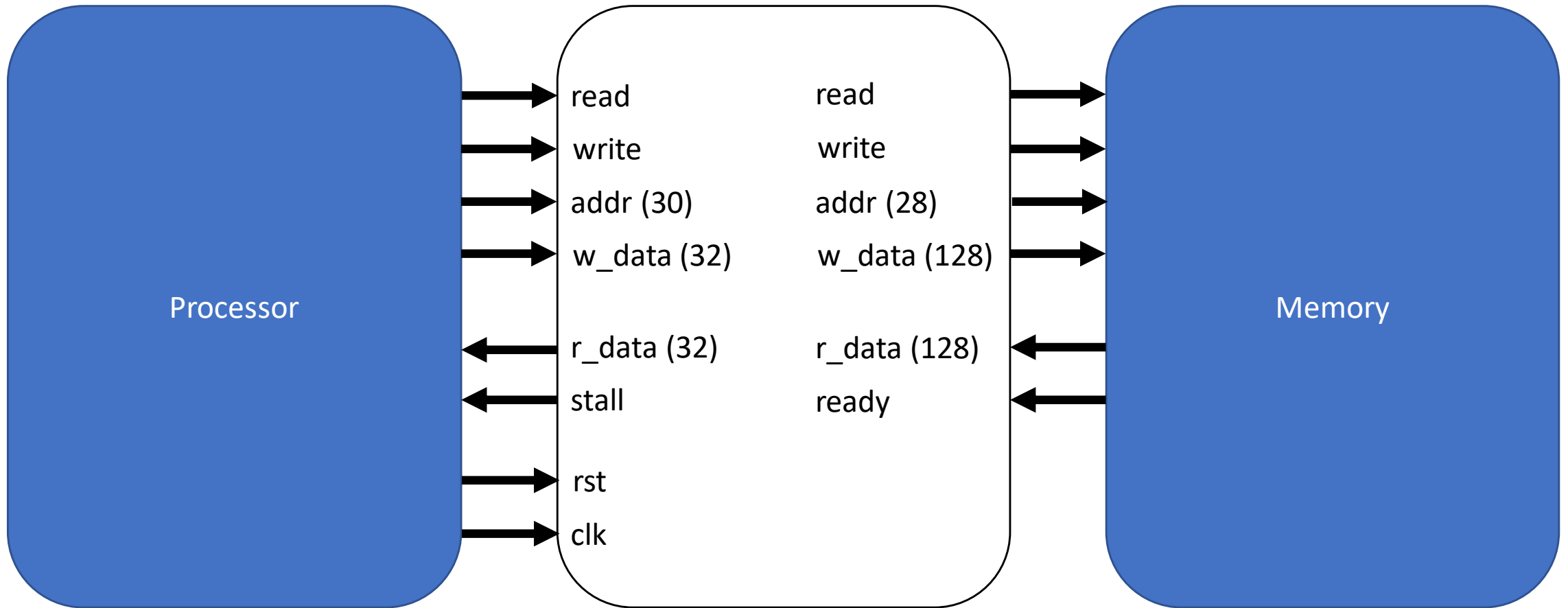
Cache

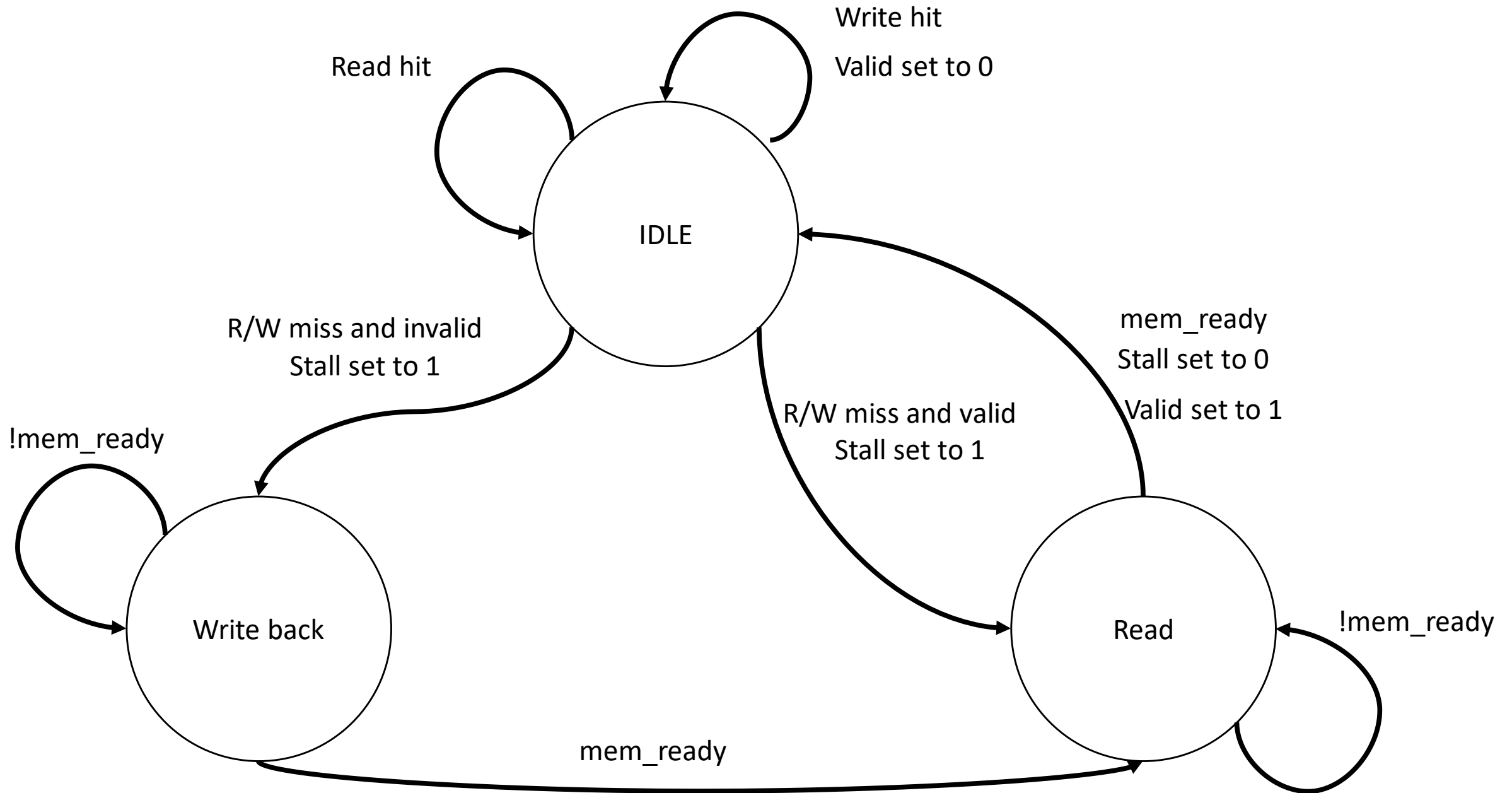
- Arrangement
 - 32-bit word
 - 4-word block, 8 blocks
 - 1-way cache
 - Write back, write allocate
 - Reset: set valid to 1, avoid writing back invalid data to memory
 - Known issue: need a dirty bit

Tag(25)	Index(3)	offset(2)
---------	----------	-----------

Valid(1)	Tag(25)	Word1(32)	Word1(32)	Word1(32)	Word1(32)
----------	---------	-----------	-----------	-----------	-----------

Cache

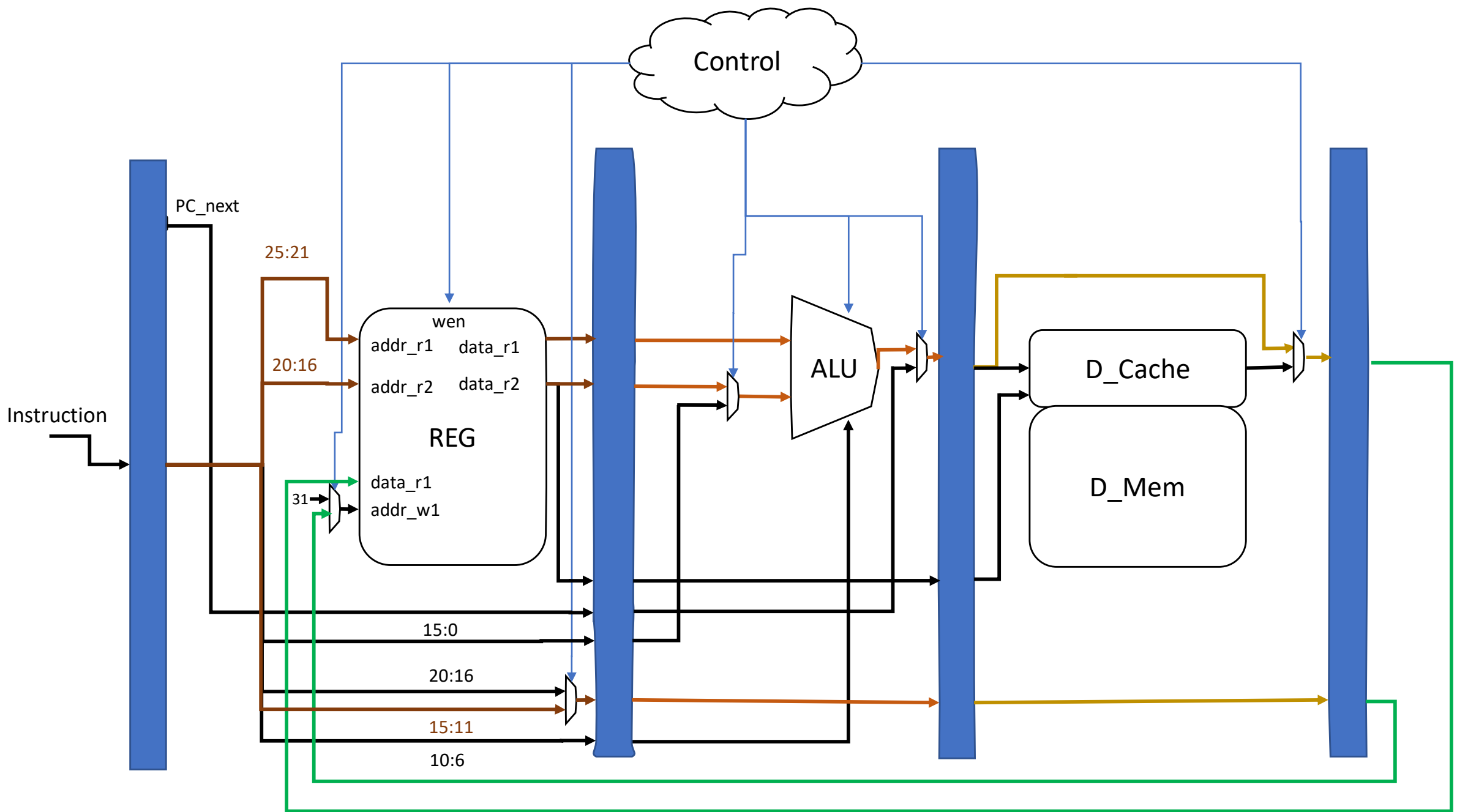




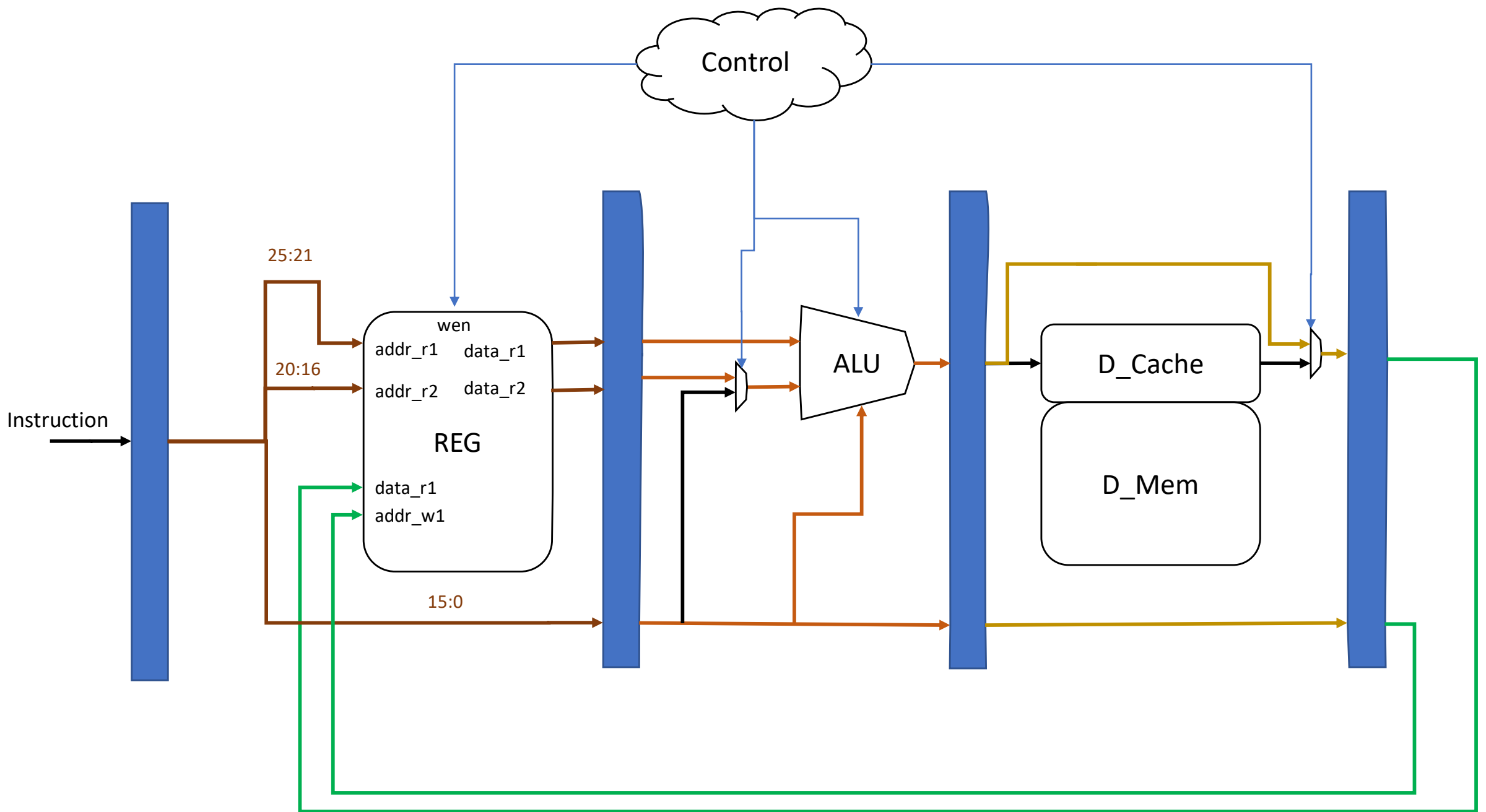
Outline

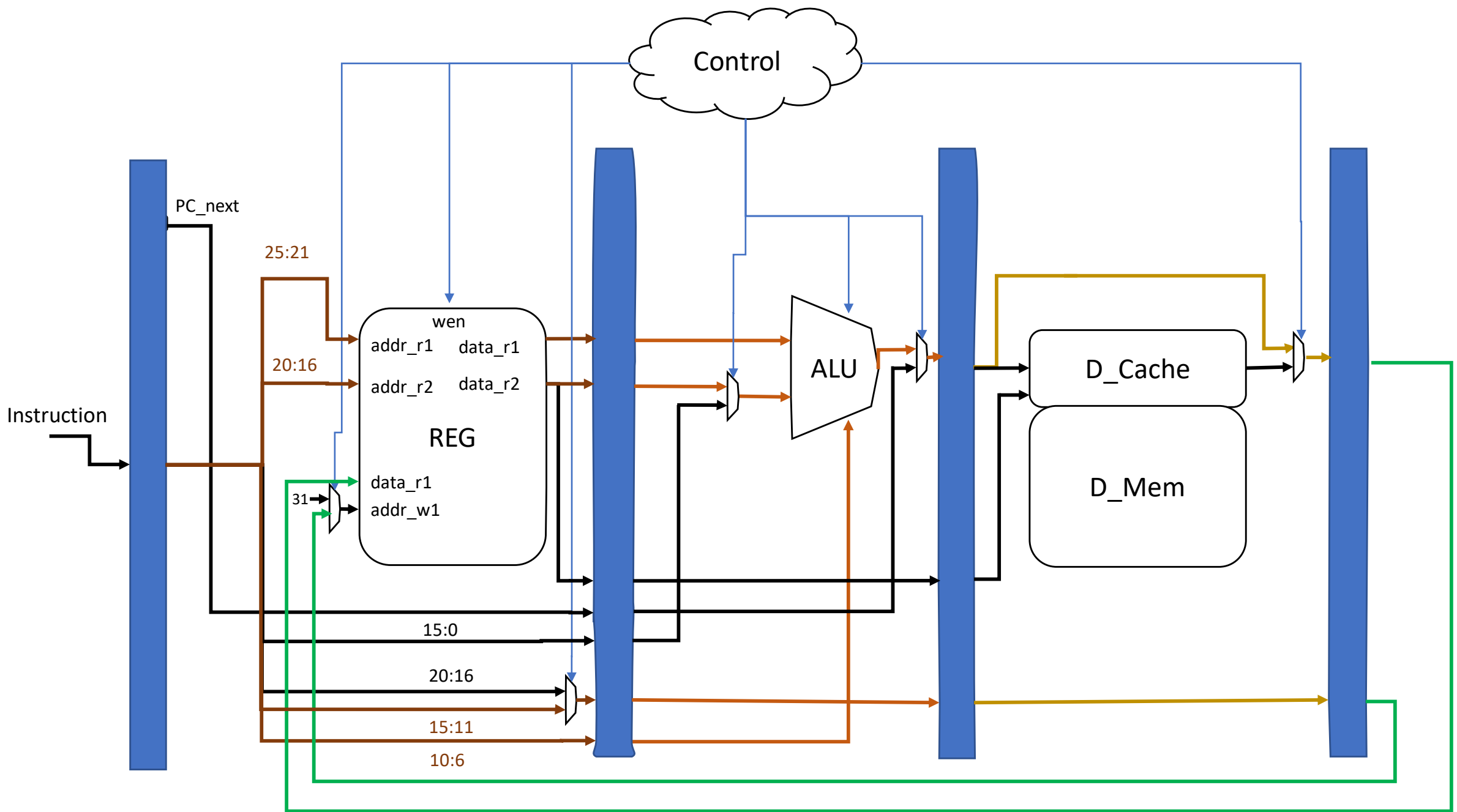
- Design Specification
- Hardware
 - Cache Implementation
 - MIPS Implementation
 - Instructions
 - Hazard Handling
 - Performance

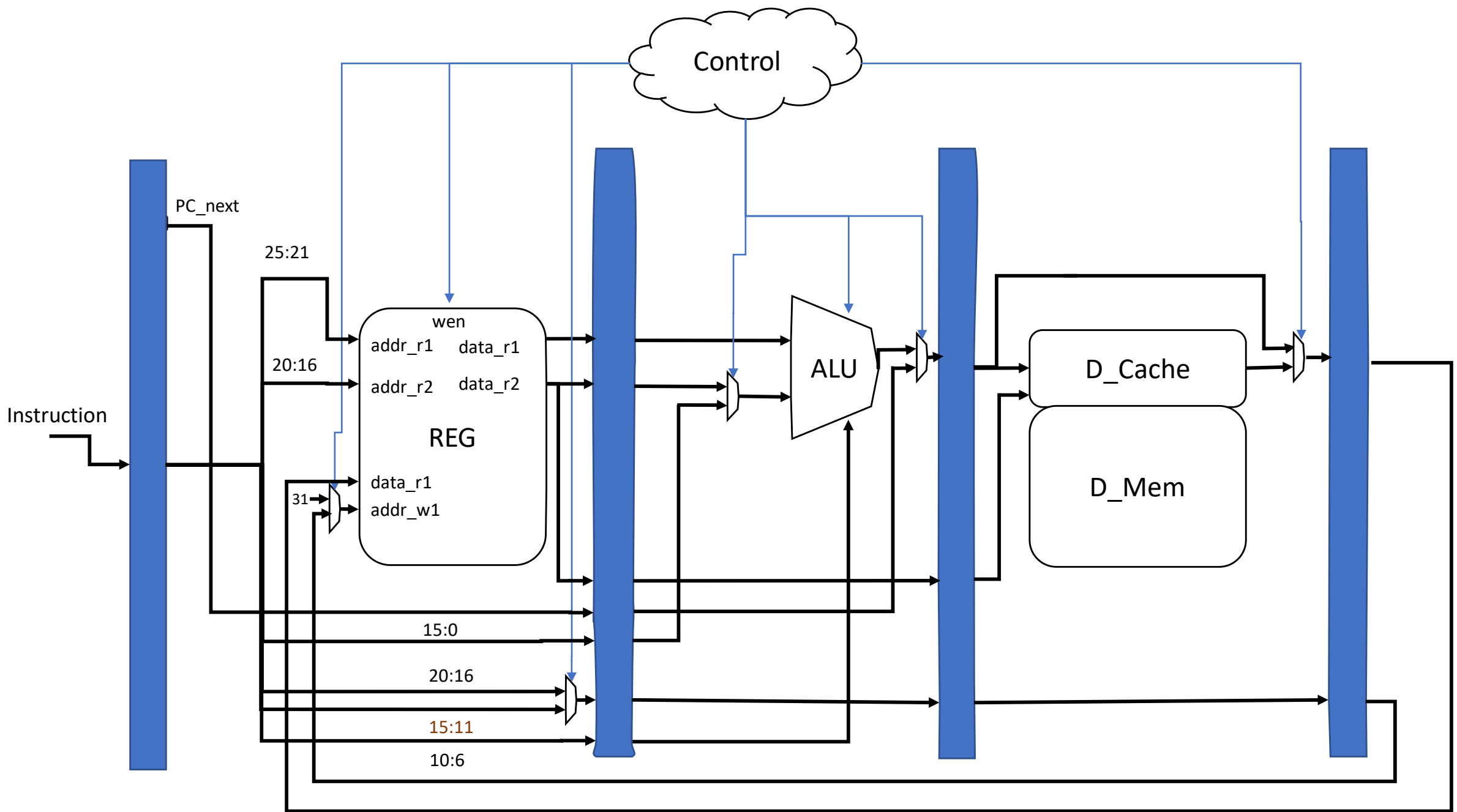
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, conditional branch instruction	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=rt)	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



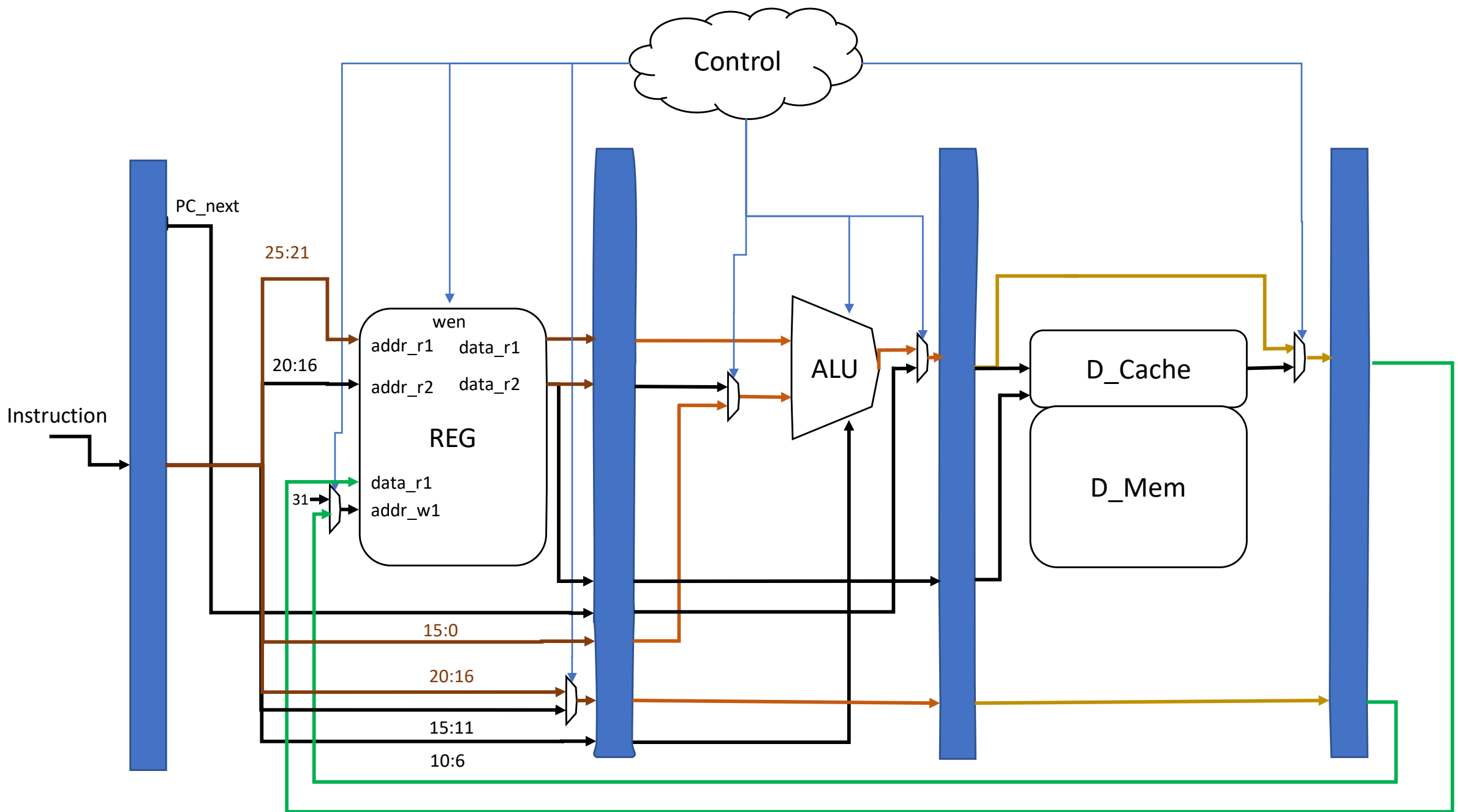
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, conditional branch instruction	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=rt)	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



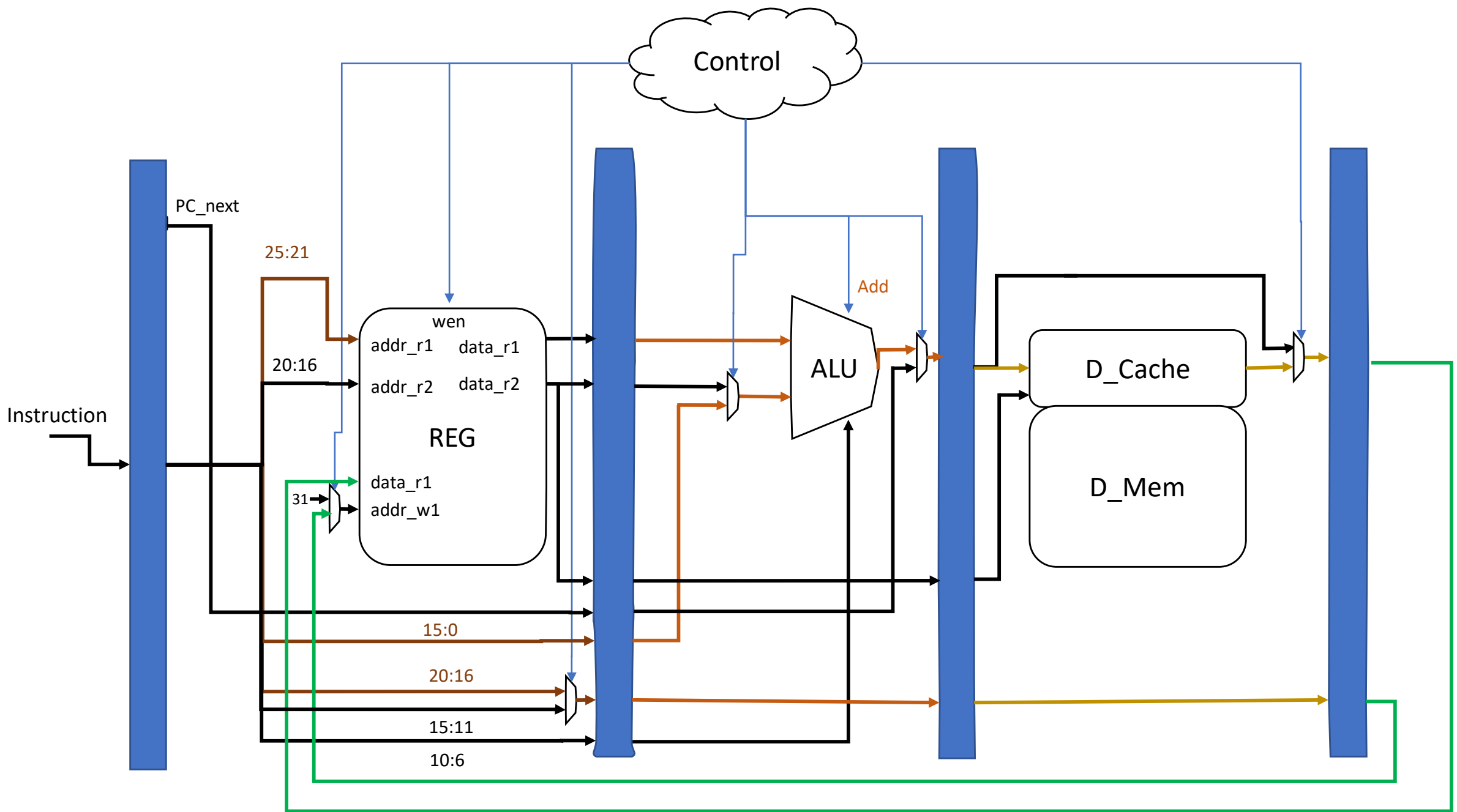




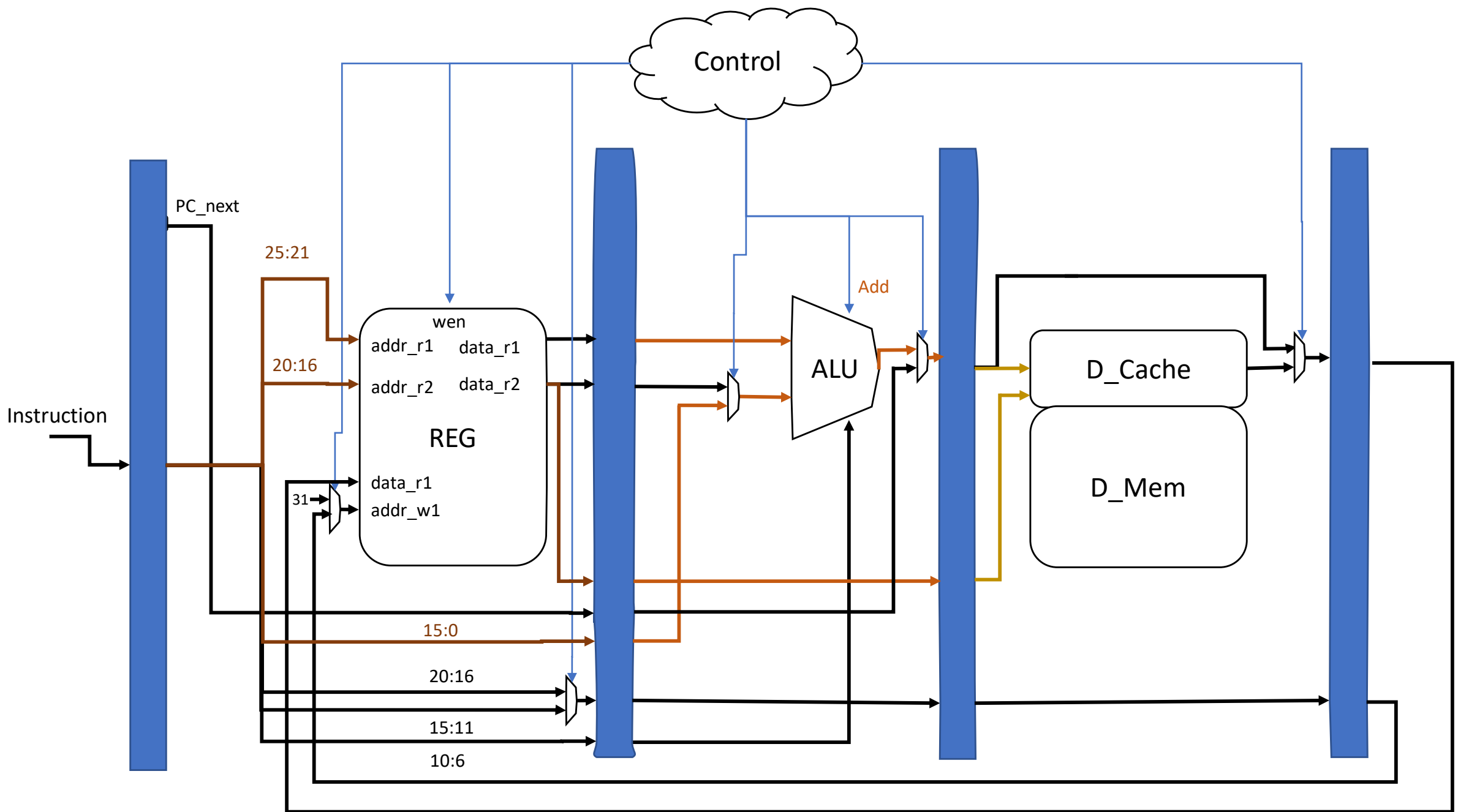
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, conditional branch instruction	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=rt)	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



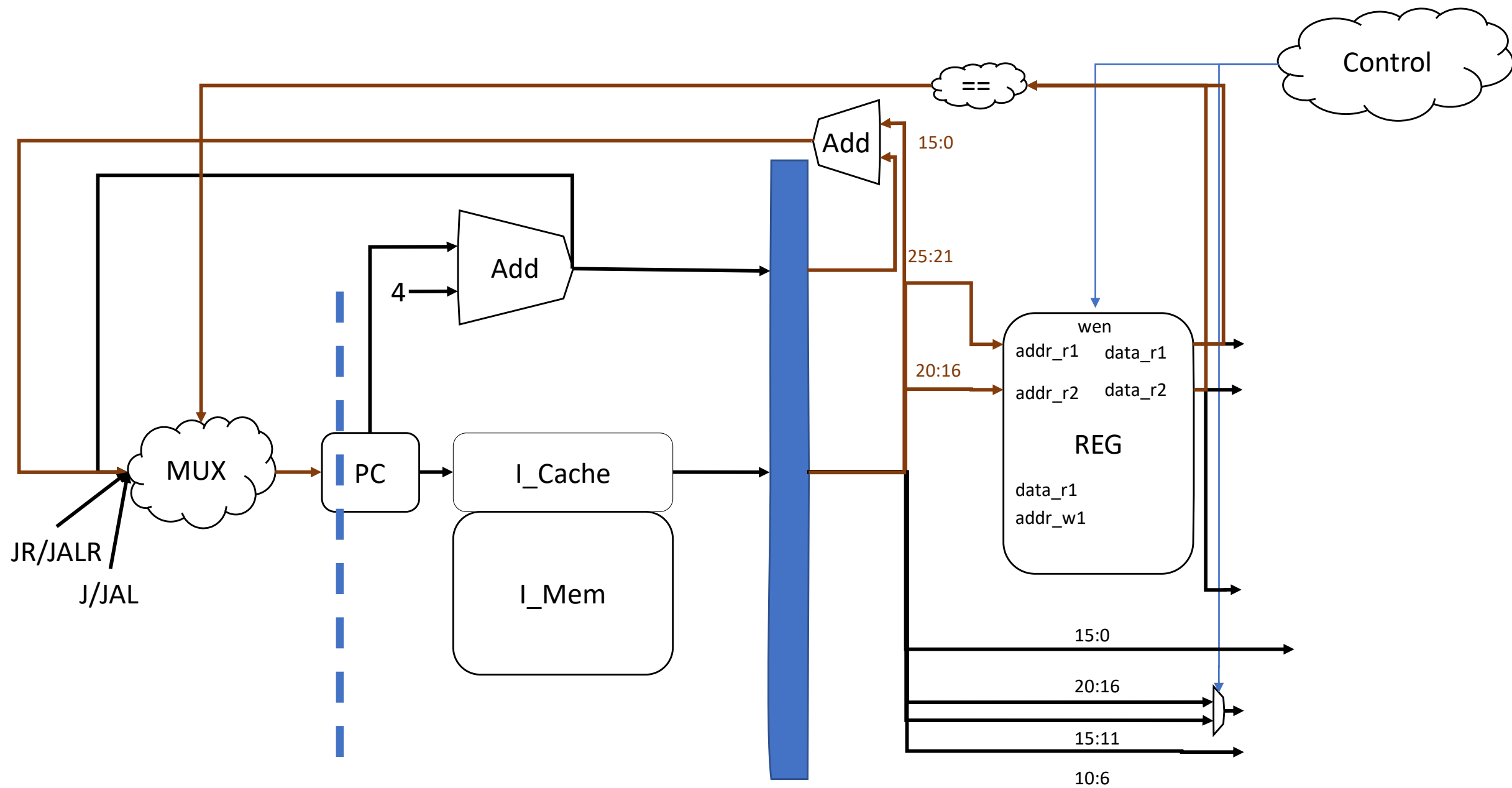
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, conditional branch instruction	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



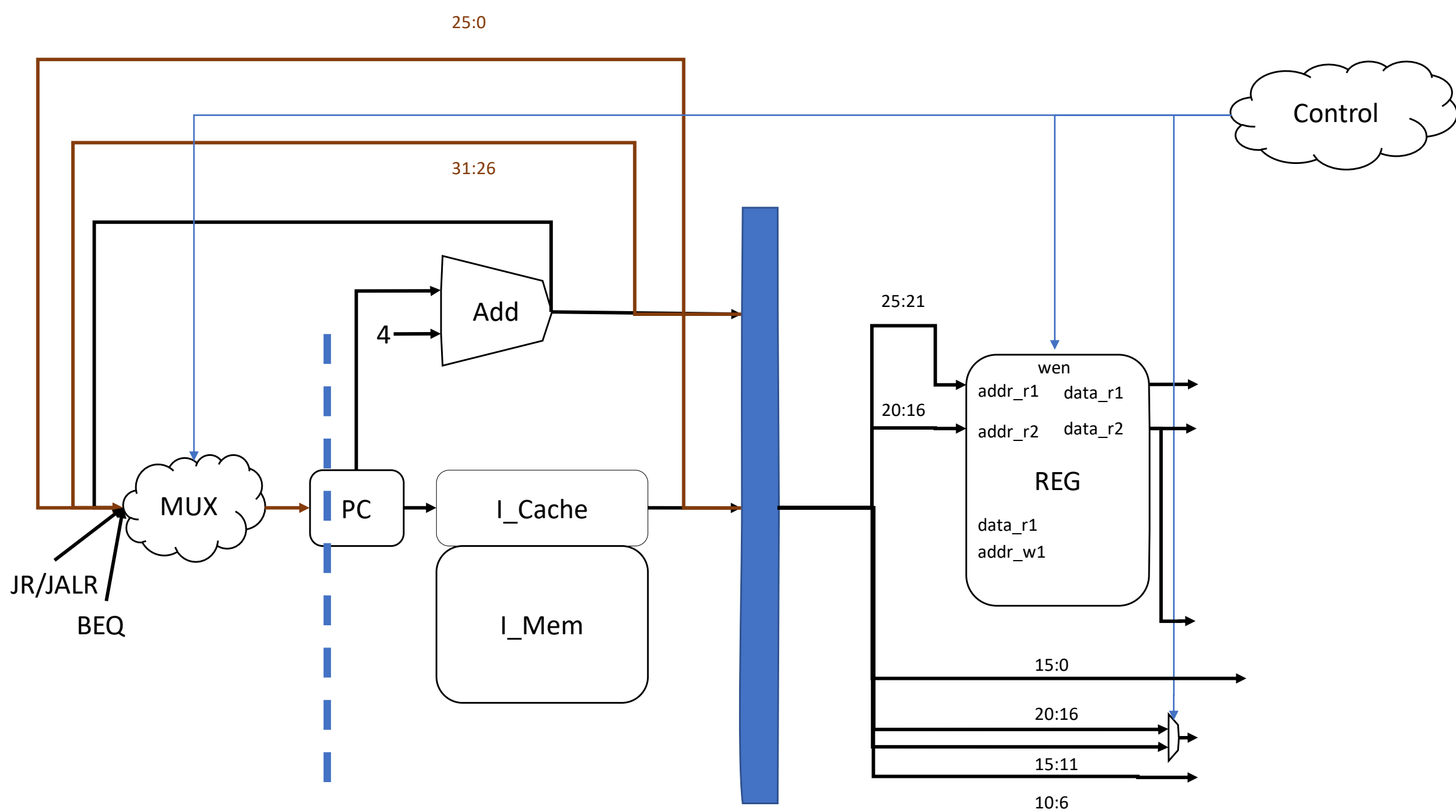
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, conditional branch instruction	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



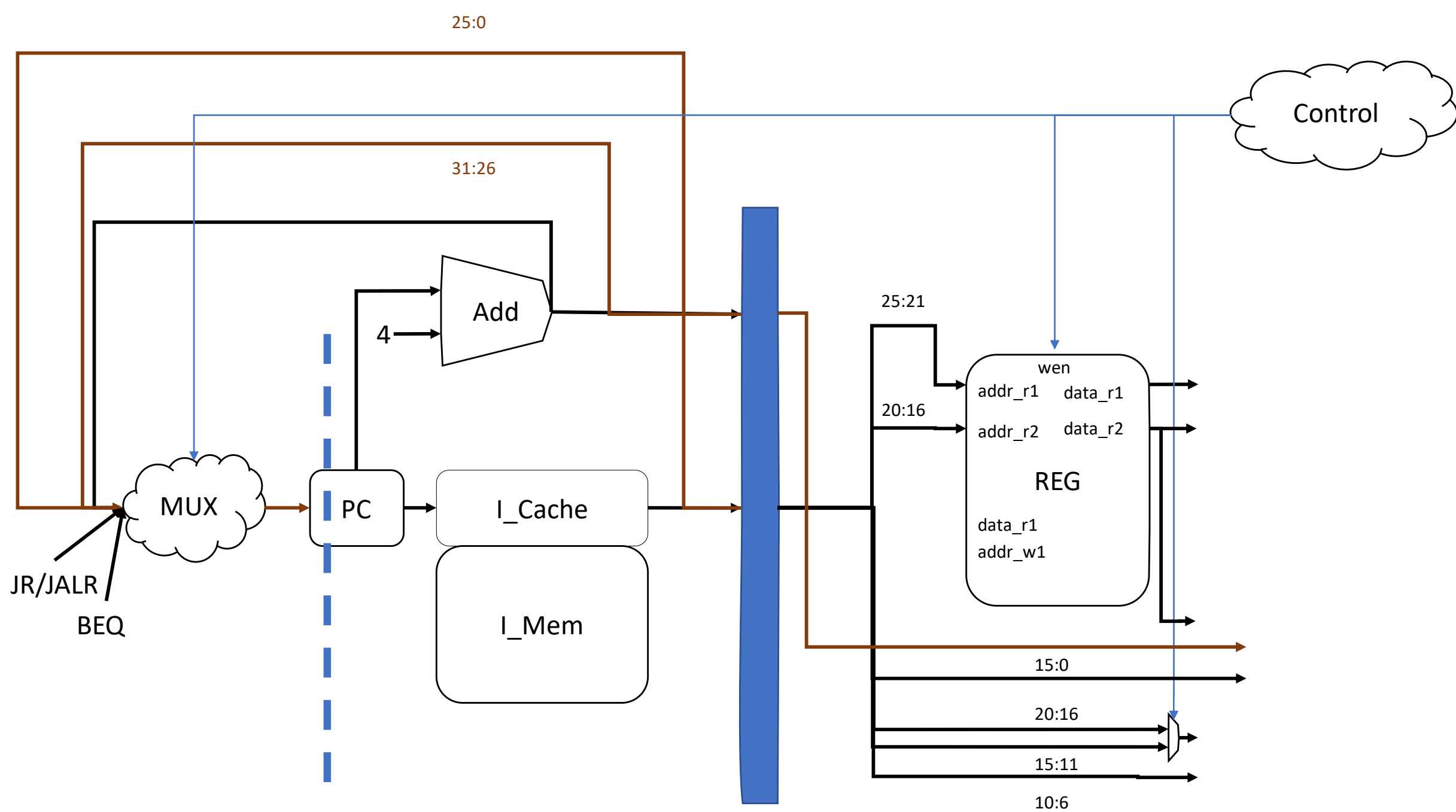
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, jump to PC+4+4*const	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000

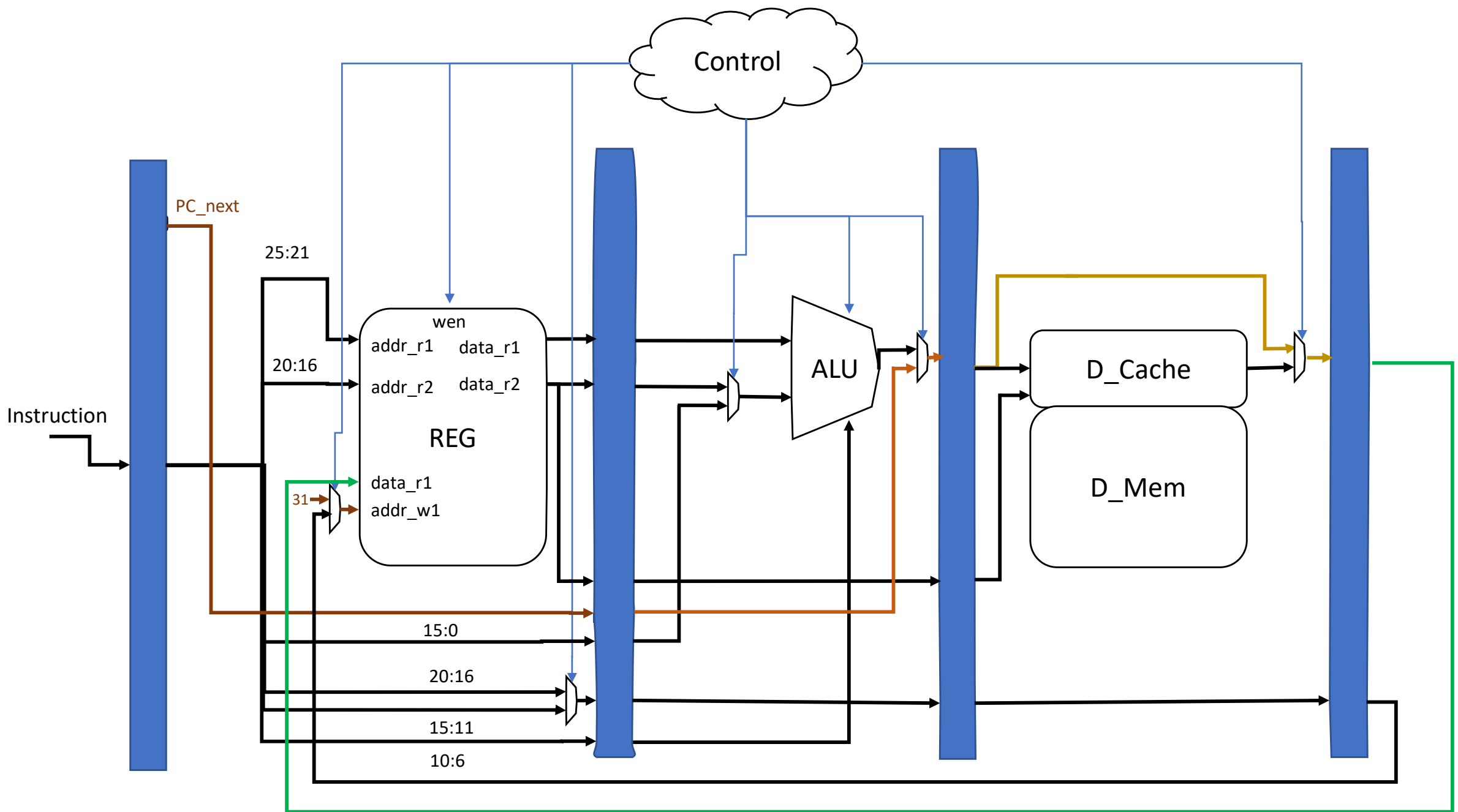


				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, jump to PC+4+4*const	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000

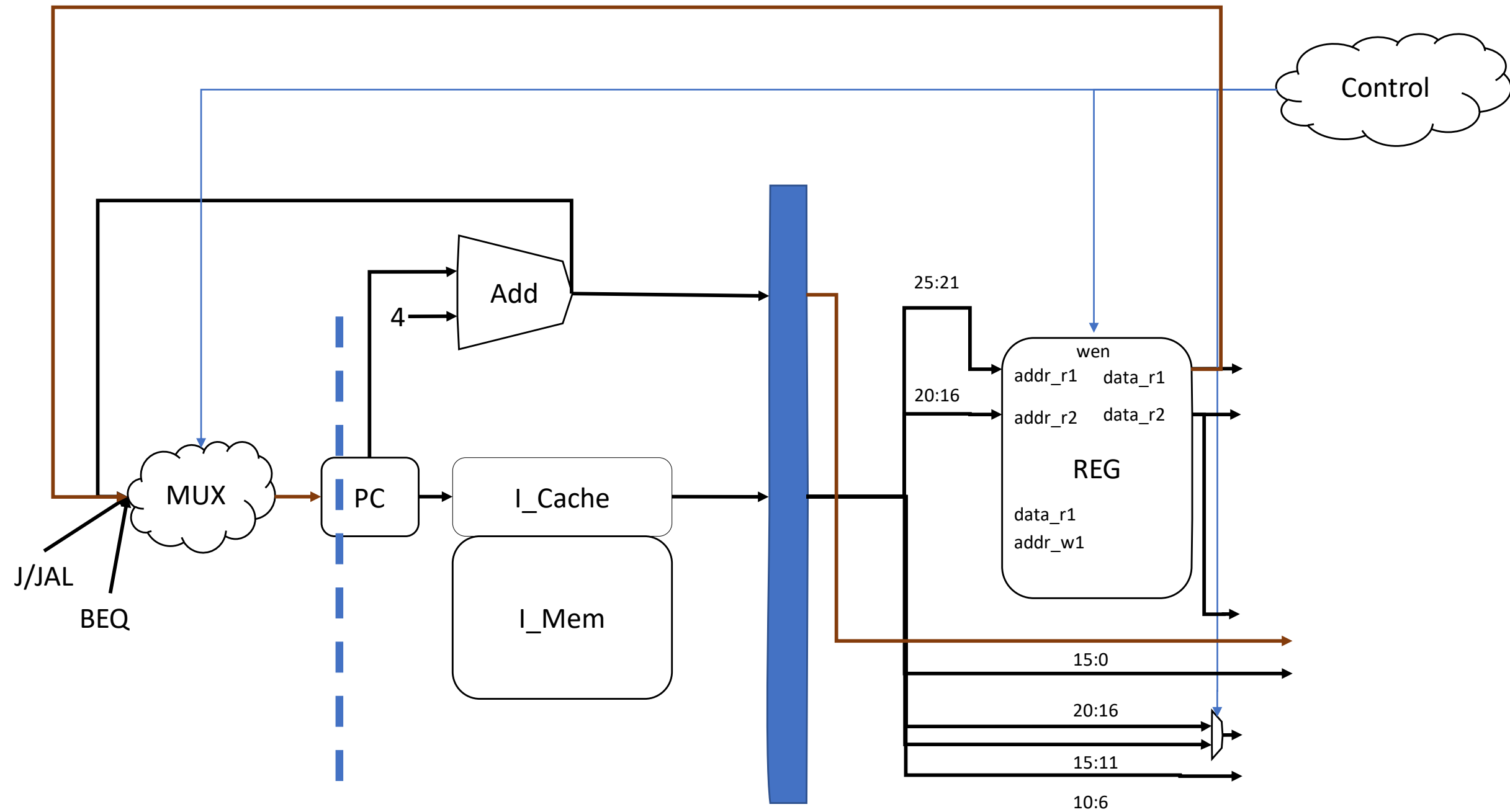


				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, jump to PC+4+4*const	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000

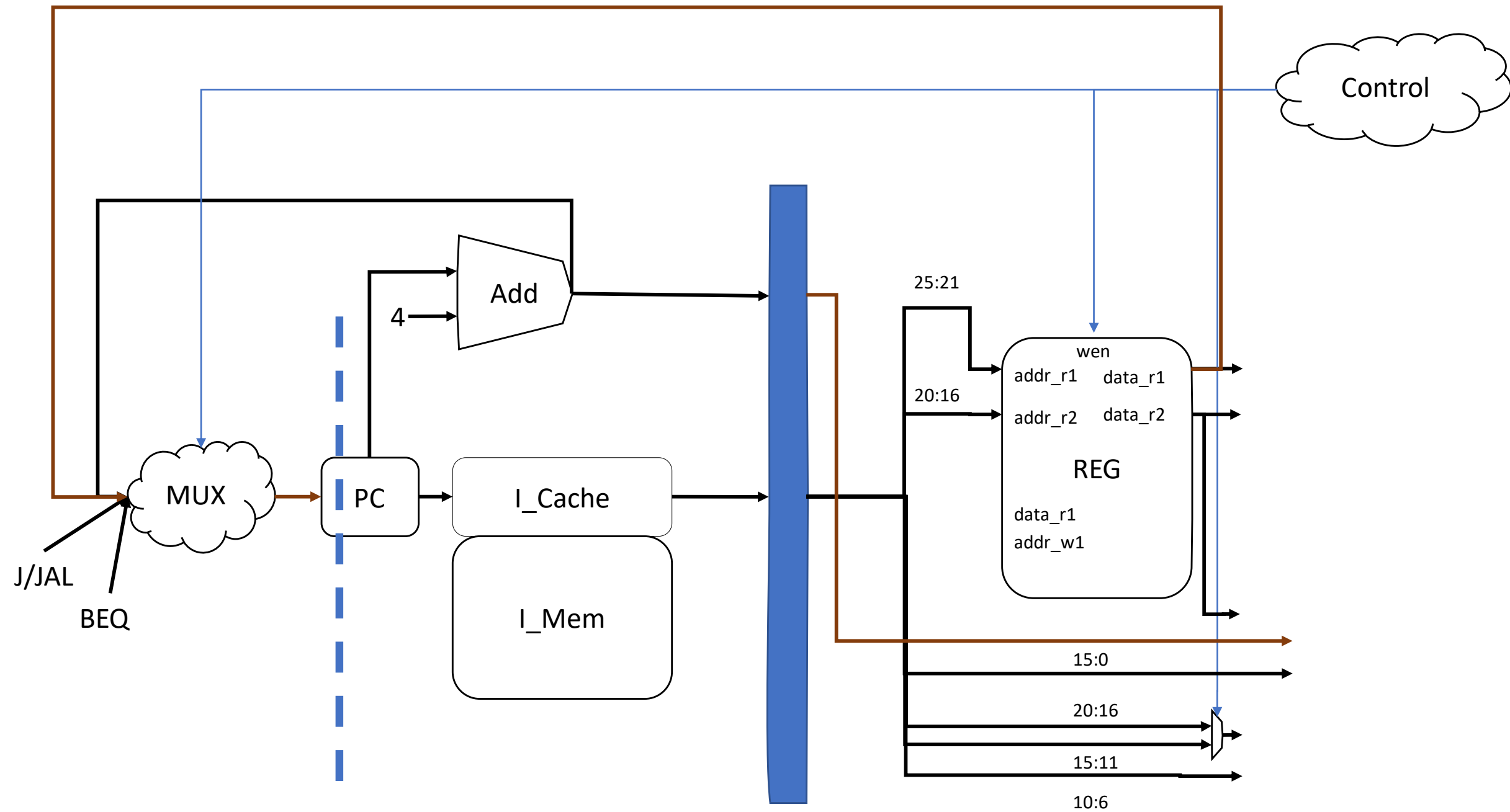


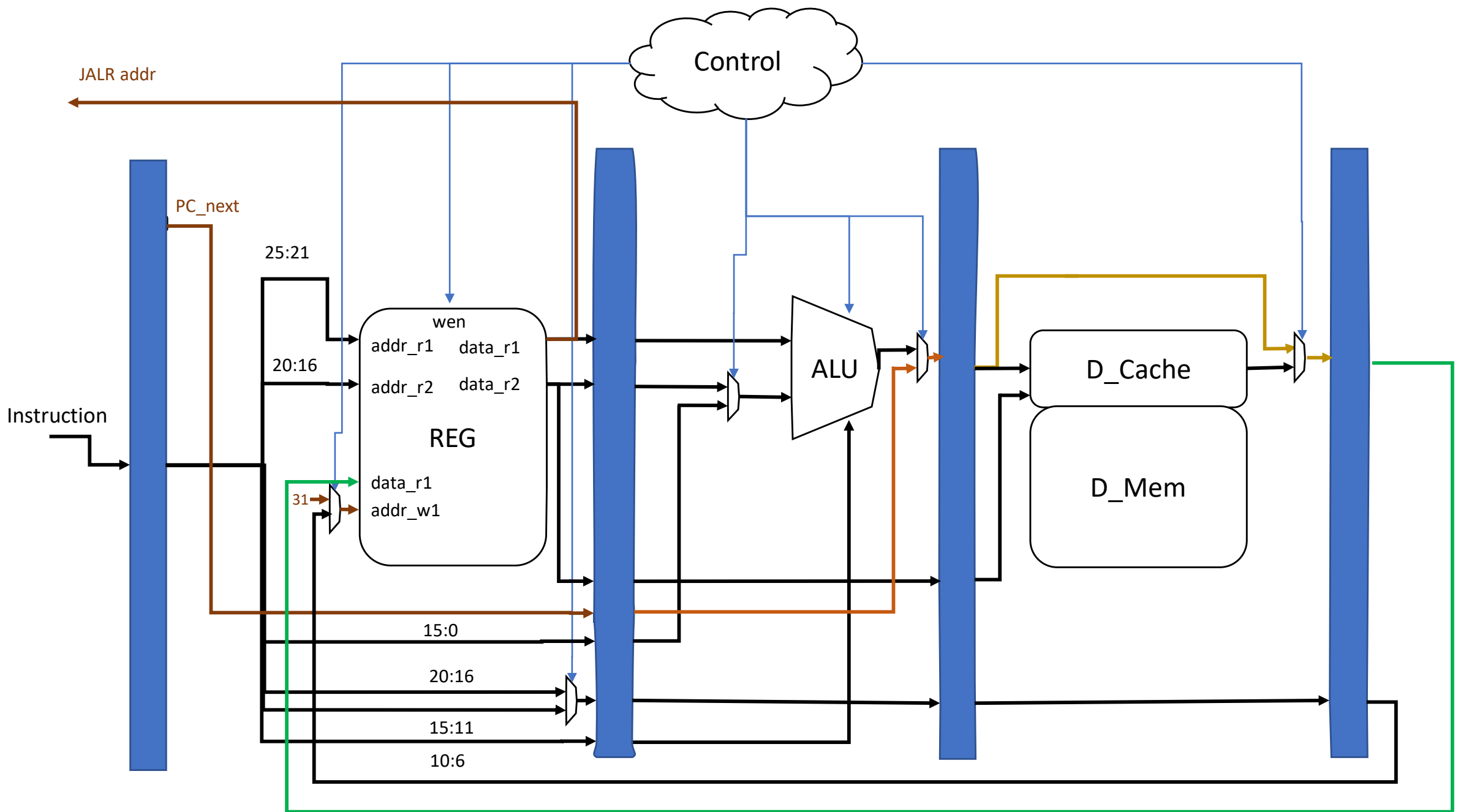


				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, jump to PC+4+4*const	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000



				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
SUB	R	Subtract, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100010
AND	R	Boolean logic operation	000000	rs	rt	rd	-	100100
ANDI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001100	rs	rt	const		
OR	R	Boolean logic operation	000000	rs	rt	rd	-	100101
ORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
XOR	R	Boolean logic operation	000000	rs	rt	rd	-	101000
XORI	I	Boolean logic operation, zero-extension for upper 16bit of immediate	001000	rs	rt	const		
NOR	R	Boolean logic operation	000000	rs	rt	rd	-	100111
SLL	R	Shift left logical (zero padding)	000000	-	rt	rd	shift	000000
SRA	R	Shift right arithmetic (sign-digit padding)	000000	-	rt	rd	shift	000011
SRL	R	Shift right logical (zero padding)	000000	-	rt	rd	shift	000010
SLT	R	Set less than, comparison instruction	000000	rs	rt	rd	-	101010
SLTI	I	Set less than variable, comparison instruction	001010	rs	rt	const		
BEQ	I	Branch on equal, jump to PC+4+4*const	000100	rs	rt	const		
J	J	Unconditionally jump	000010	addr				
JAL	J	Unconditionally jump and link (Save next PC in \$r31)	000011	addr				
JR	R	Unconditionally jump to the instruction whose address is in \$rs	000000	rs	-	-	-	001000
JALR	R	Jump and link register(save next PC to rd)	000000	rs	-	rd	-	001001
LW	I	Load word from data memory (rt=MEM([rs]+const))	100011	rs	rt	const		
SW	I	Store word to data memory (MEM([rs]+const)=[rt])	101011	rs	rt	const		
NOP	R	No operation	000000	-	-	-	-	000000

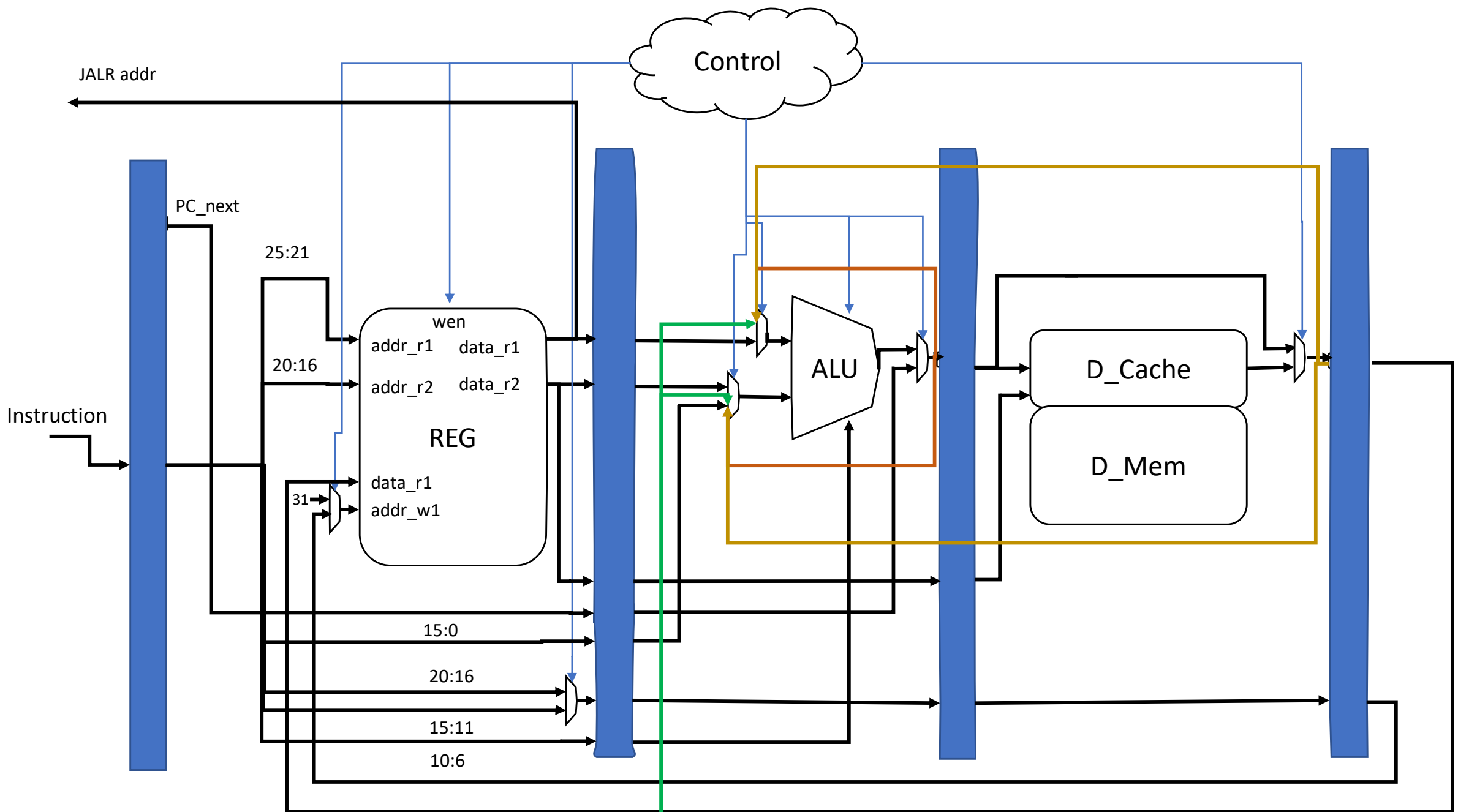




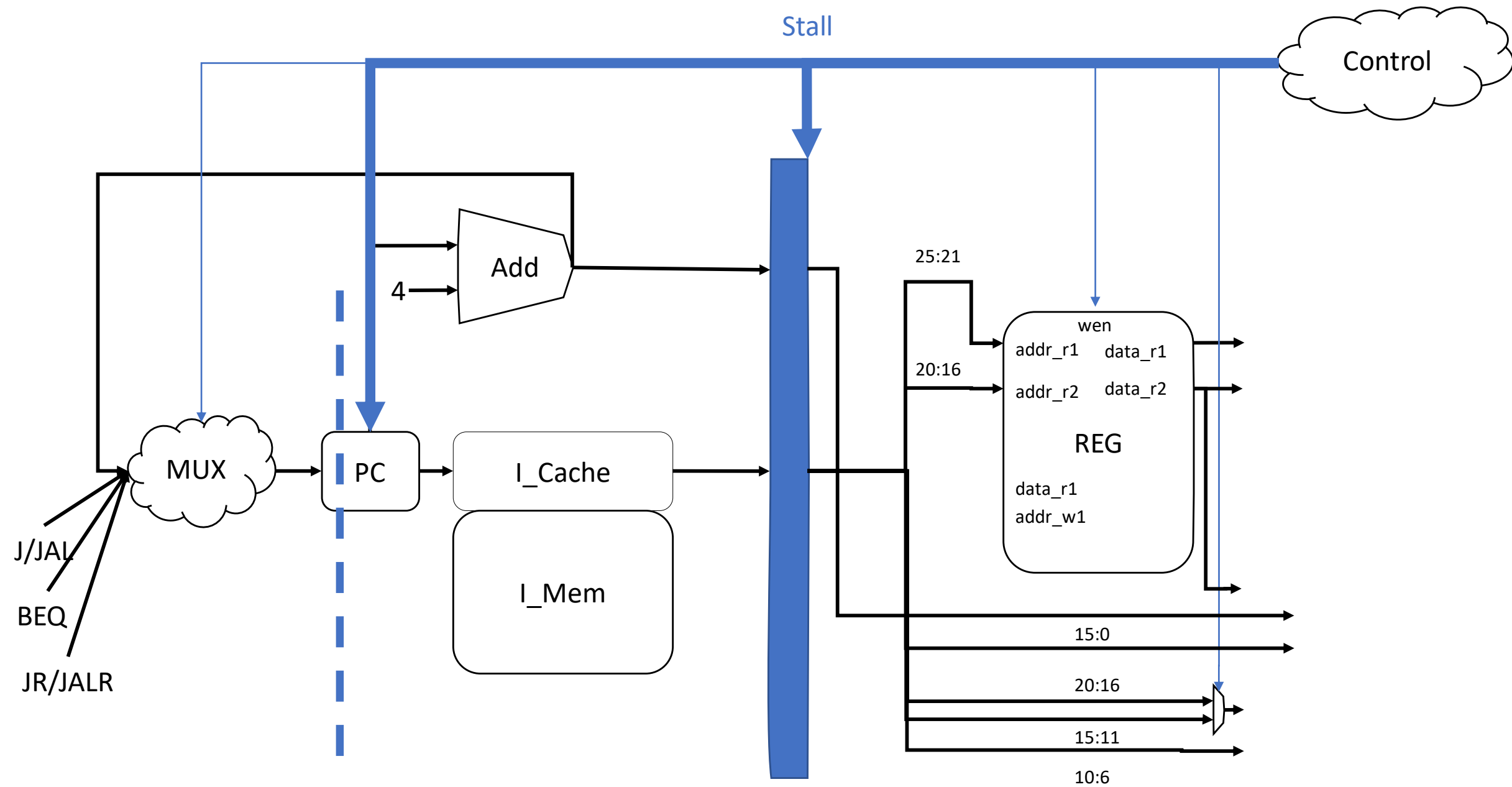
Outline

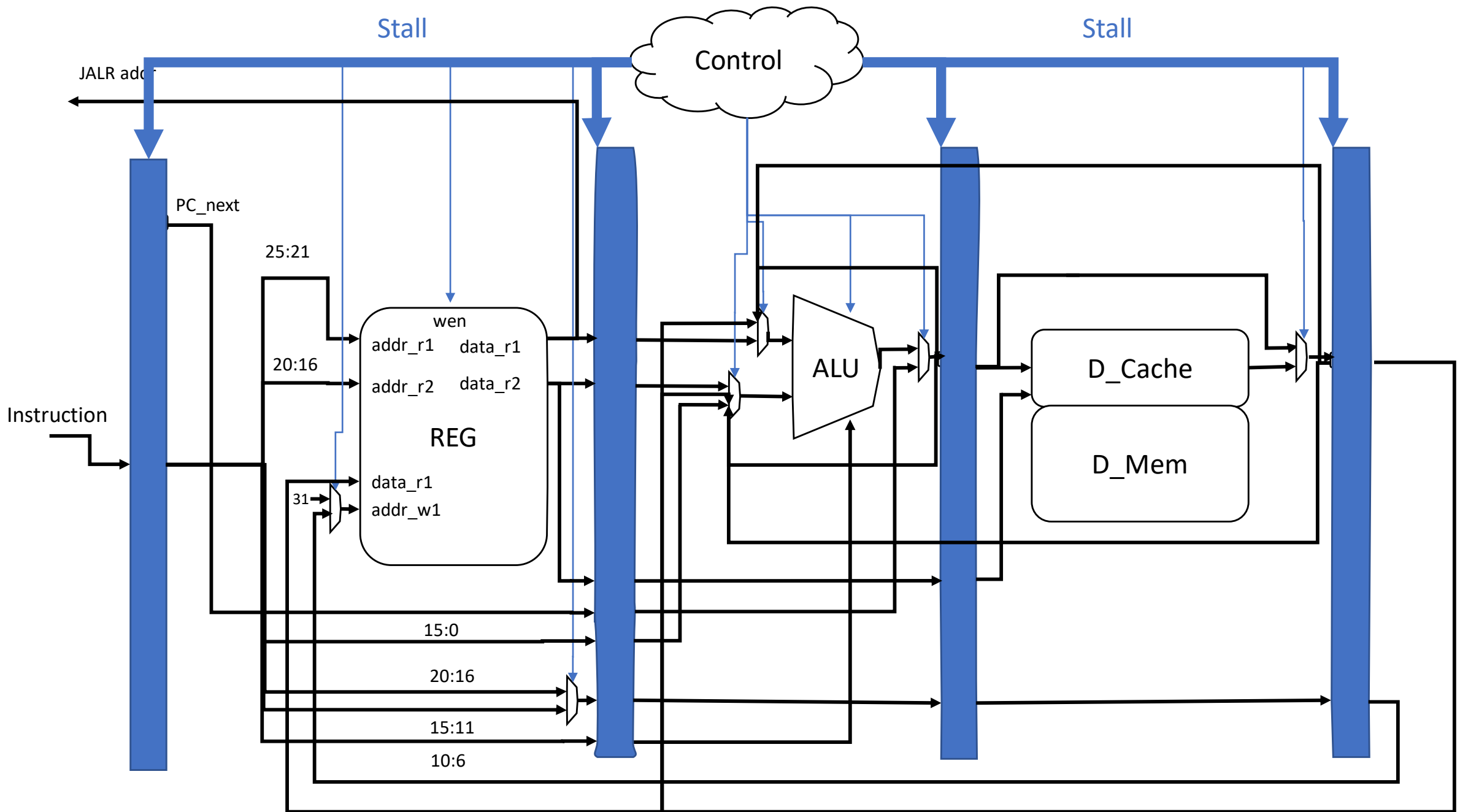
- Design Specification
- Hardware
 - Cache Implementation
 - MIPS Implementation
 - Instructions
 - Hazard Handling
 - Performance

Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Insert Bubble & Stall
Jump/Branch	Flush

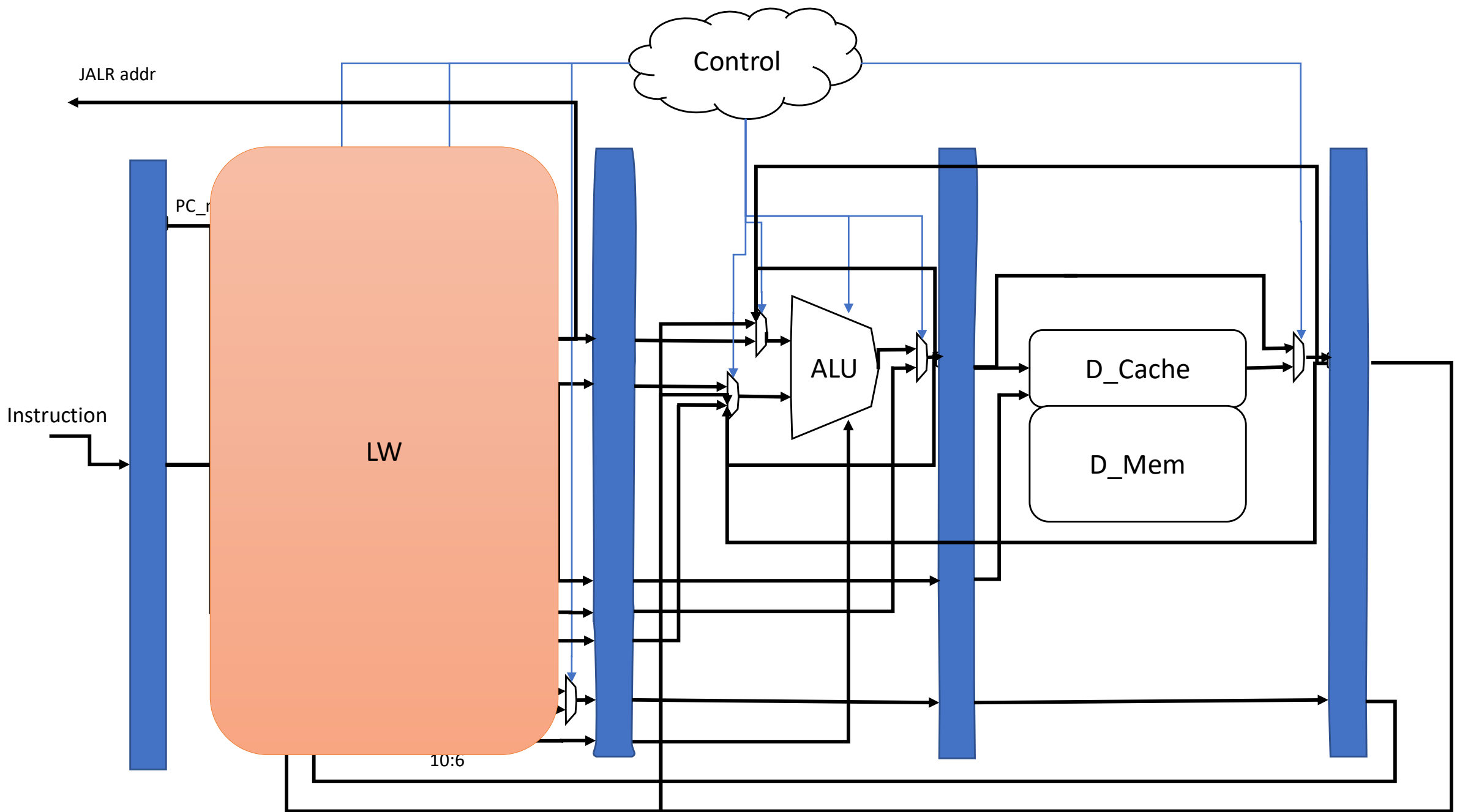


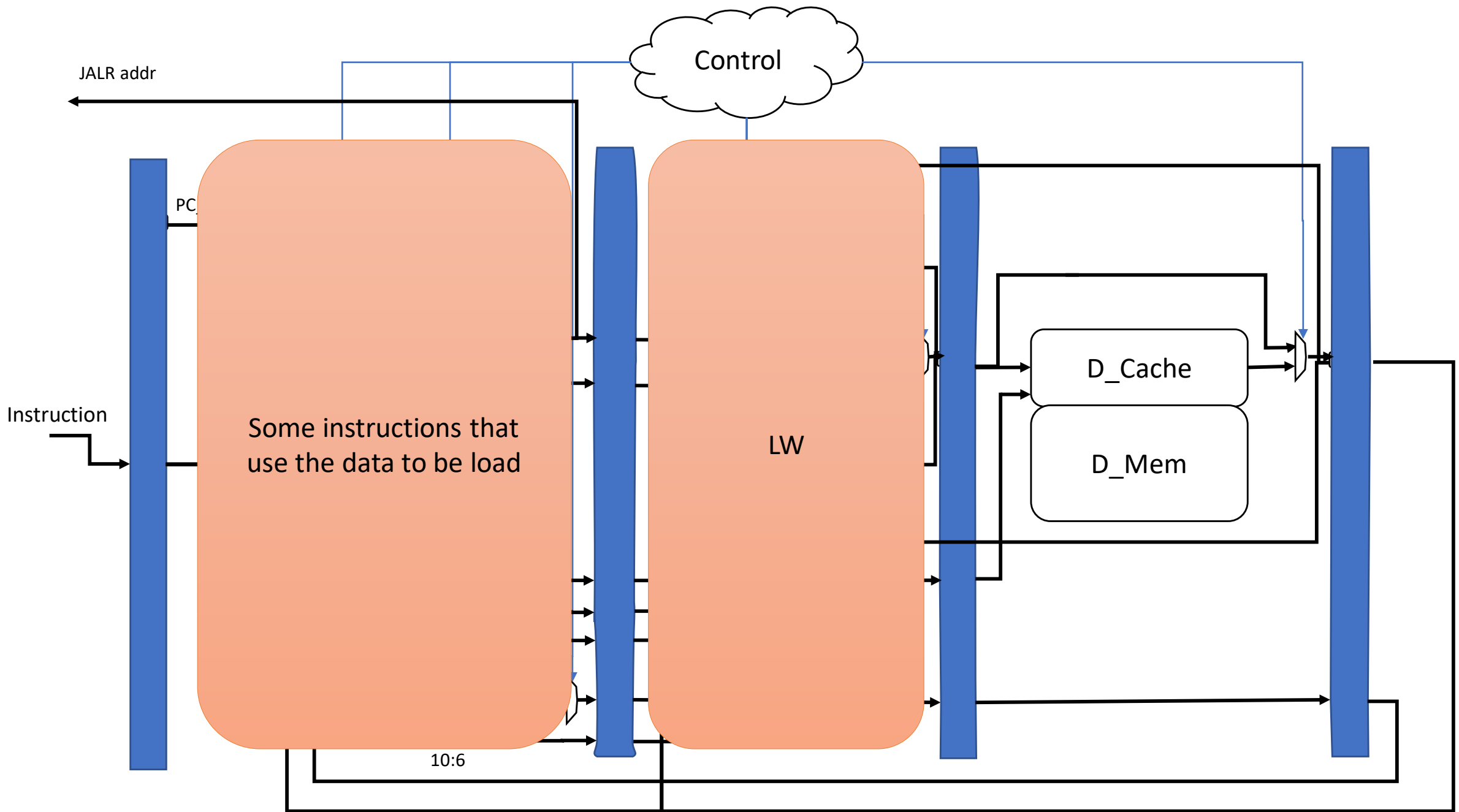
Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Stall & Insert Bubble
Jump/Branch	Flush

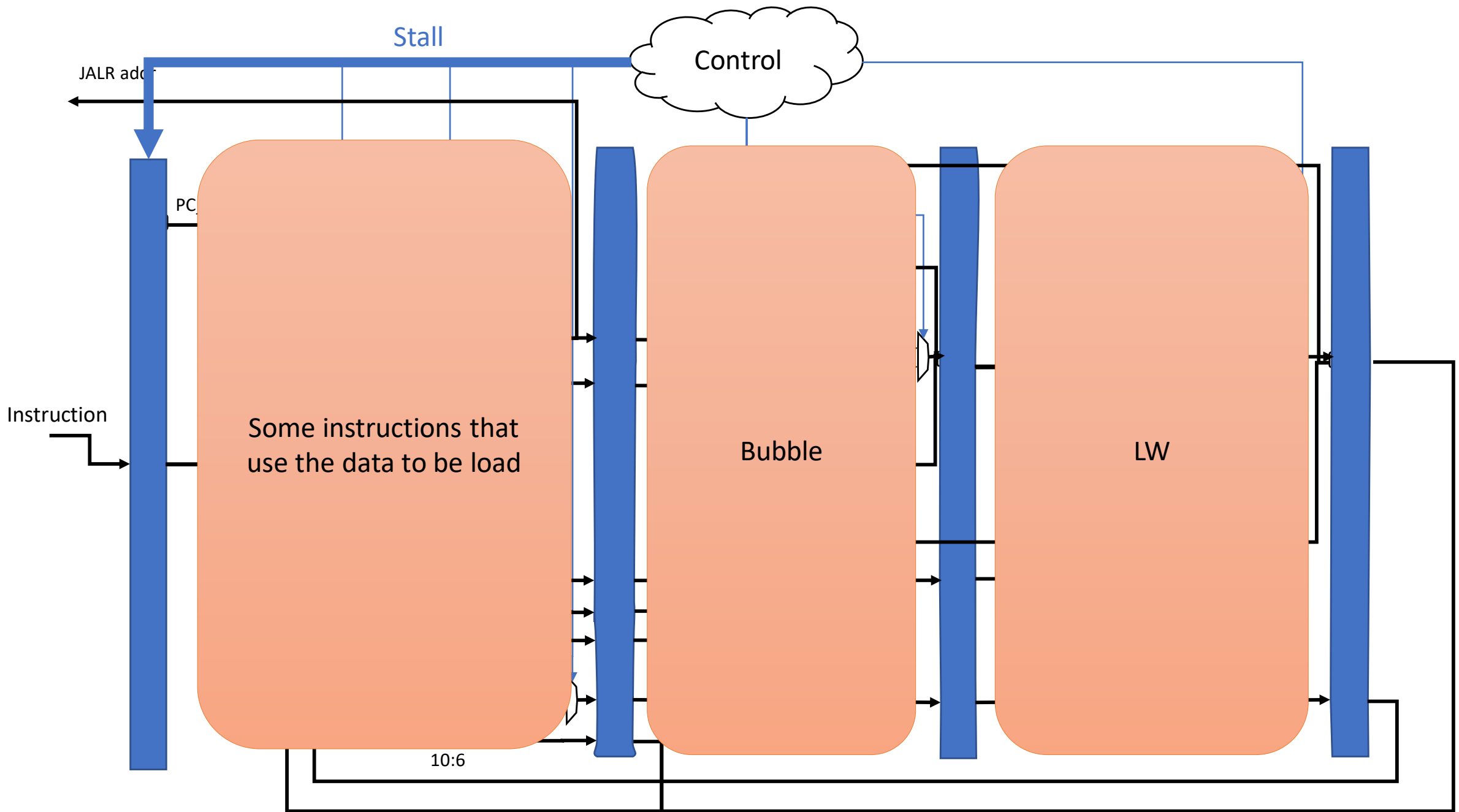




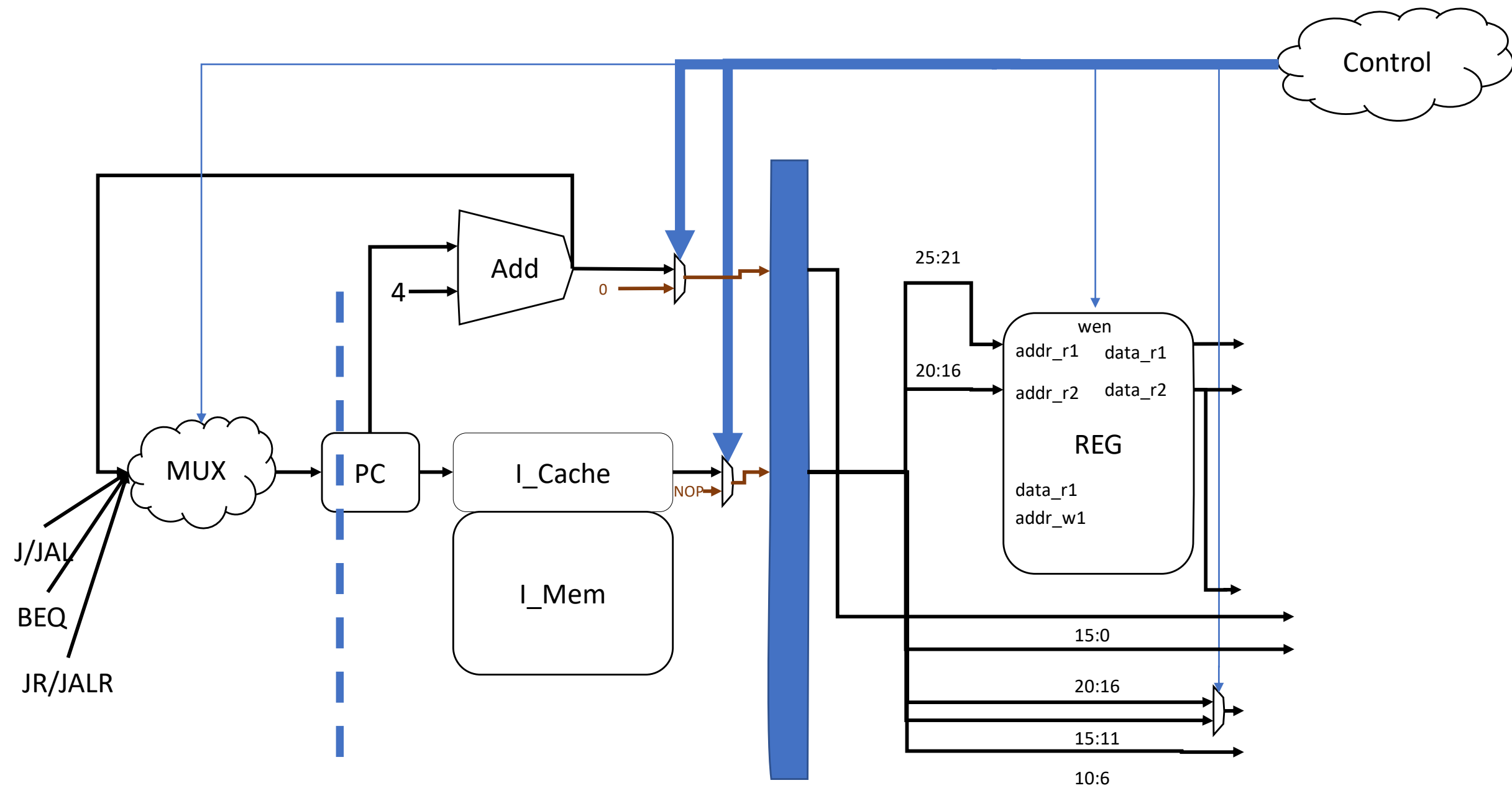
Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Insert Bubble & Stall
Jump/Branch	Flush







Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Insert Bubble & Stall
Branch Hazard (BEQ/JR/JALR)	Flush



Outline

- Design Specification
- **Hardware**
 - Cache Implementation
 - MIPS Implementation
 - Instructions
 - Hazard Handling
 - **Performance**

Synthesis Result

- Using TSMC 0.13um technology
 - Minimum clock cycle: 3.5ns
 - Total simulation time of the testbench: 7145.25 ns
 - 286617um²

