MIPS project

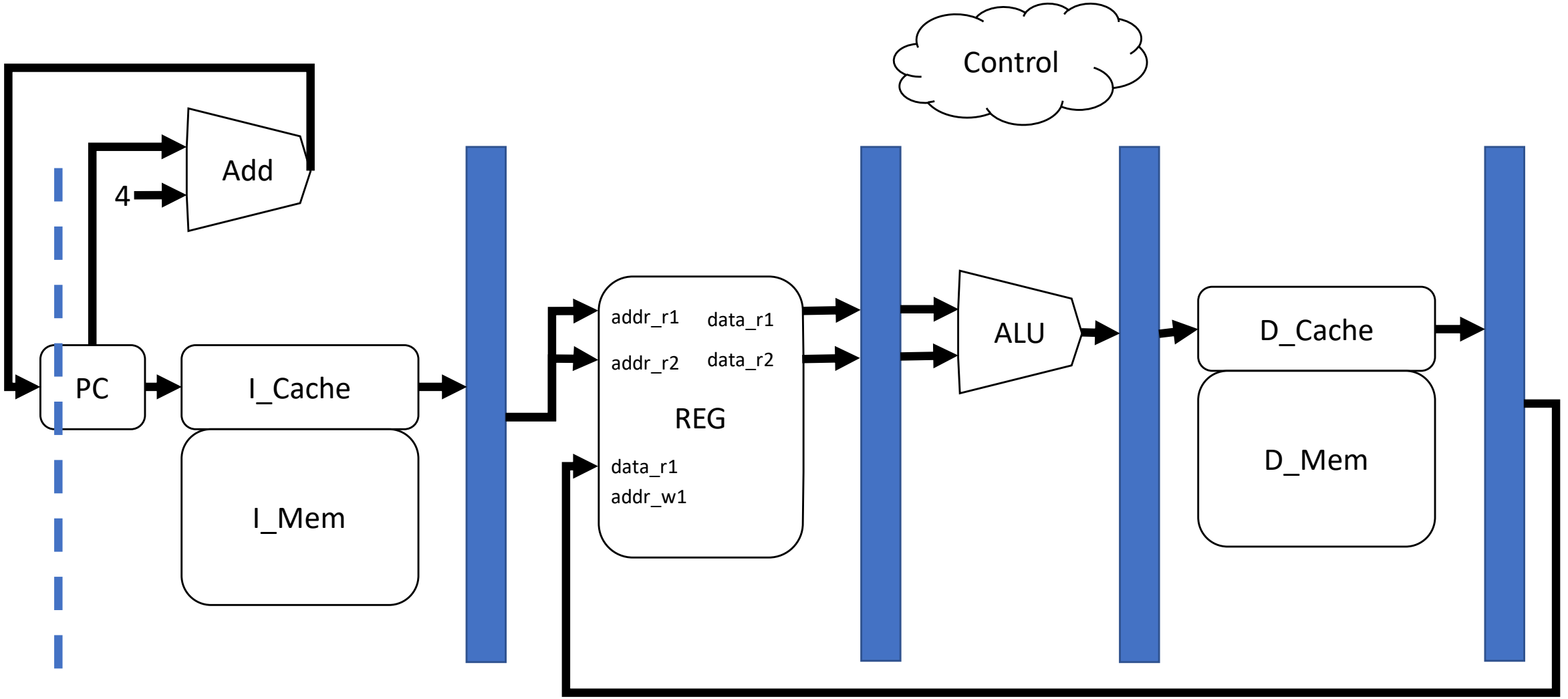
Outline

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

Instructions supported

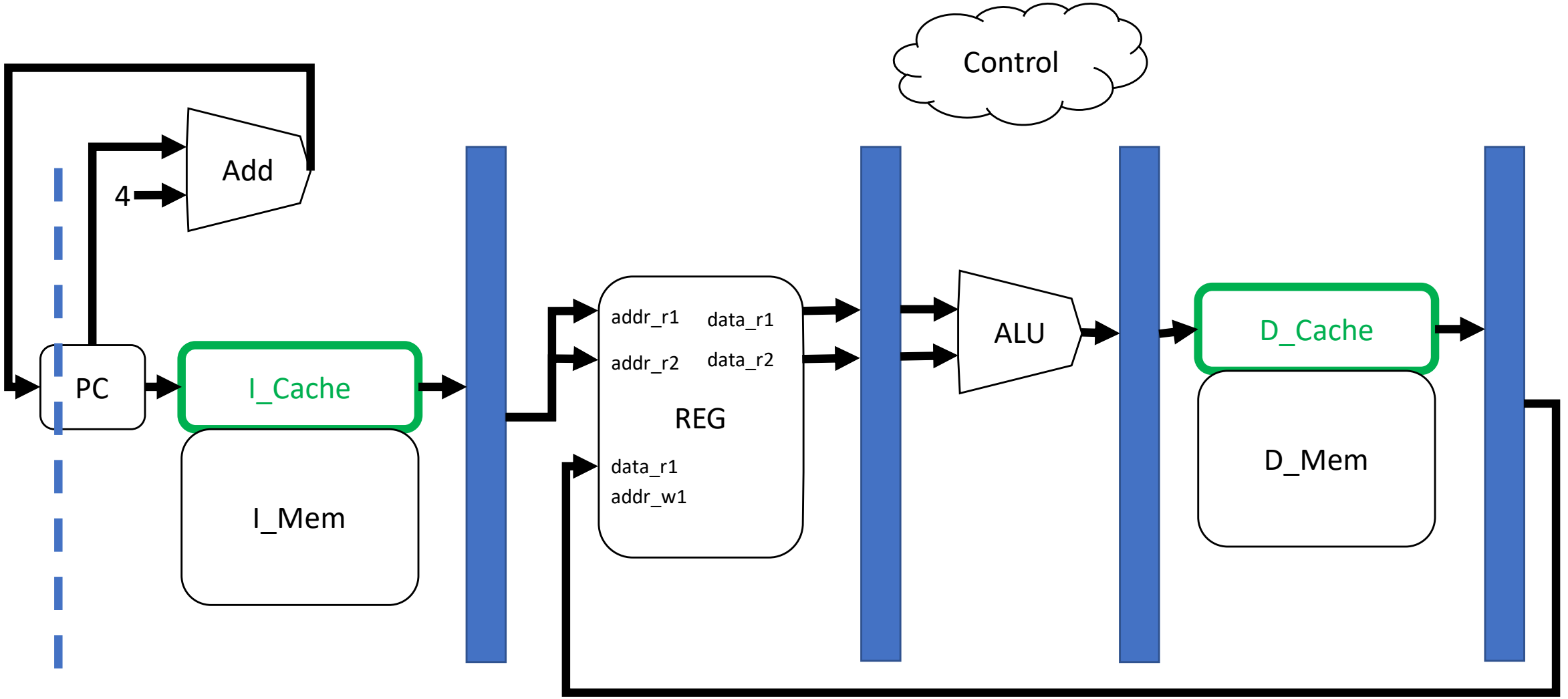
Name	Description
ADD	Addition, overflow detection for signed operand is not required*
ADDI	Addition immediate with sign-extension, without overflow detection*
SUB	Subtract, overflow detection for signed operand is not required*
AND	Boolean logic operation
ANDI	Boolean logic operation, zero-extension 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



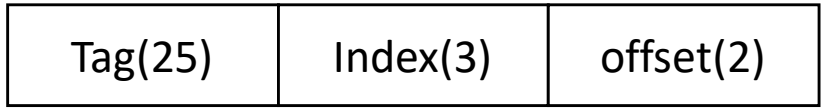
Outline

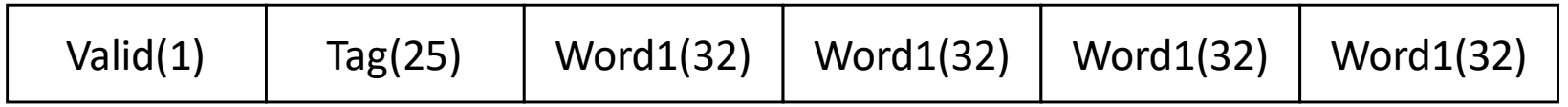
- Design Specification
- Hardware
  - Cache Implementation
  - MIPS Implementation
    - Instructions
    - Hazard Handling
  - Performance


Control
Add
4
PC
I_Cache
I_Mem
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
ALU
D_Cache
D_Mem

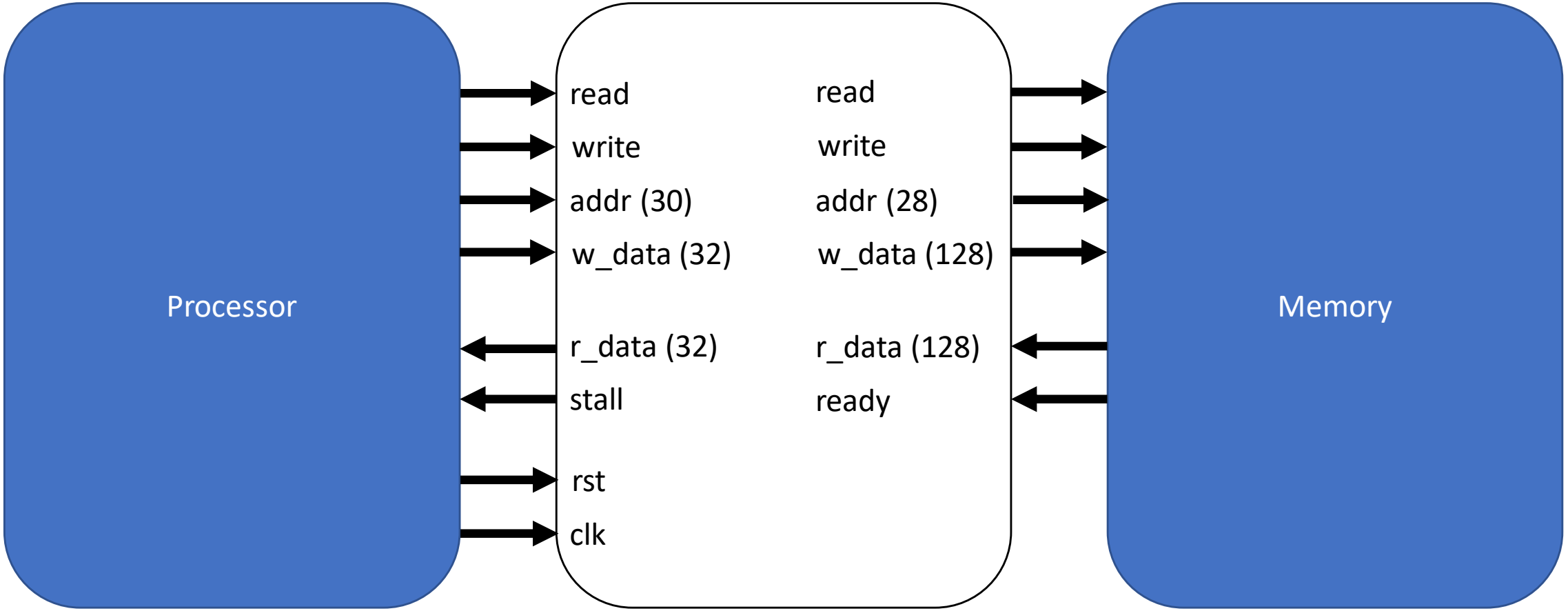
Cache

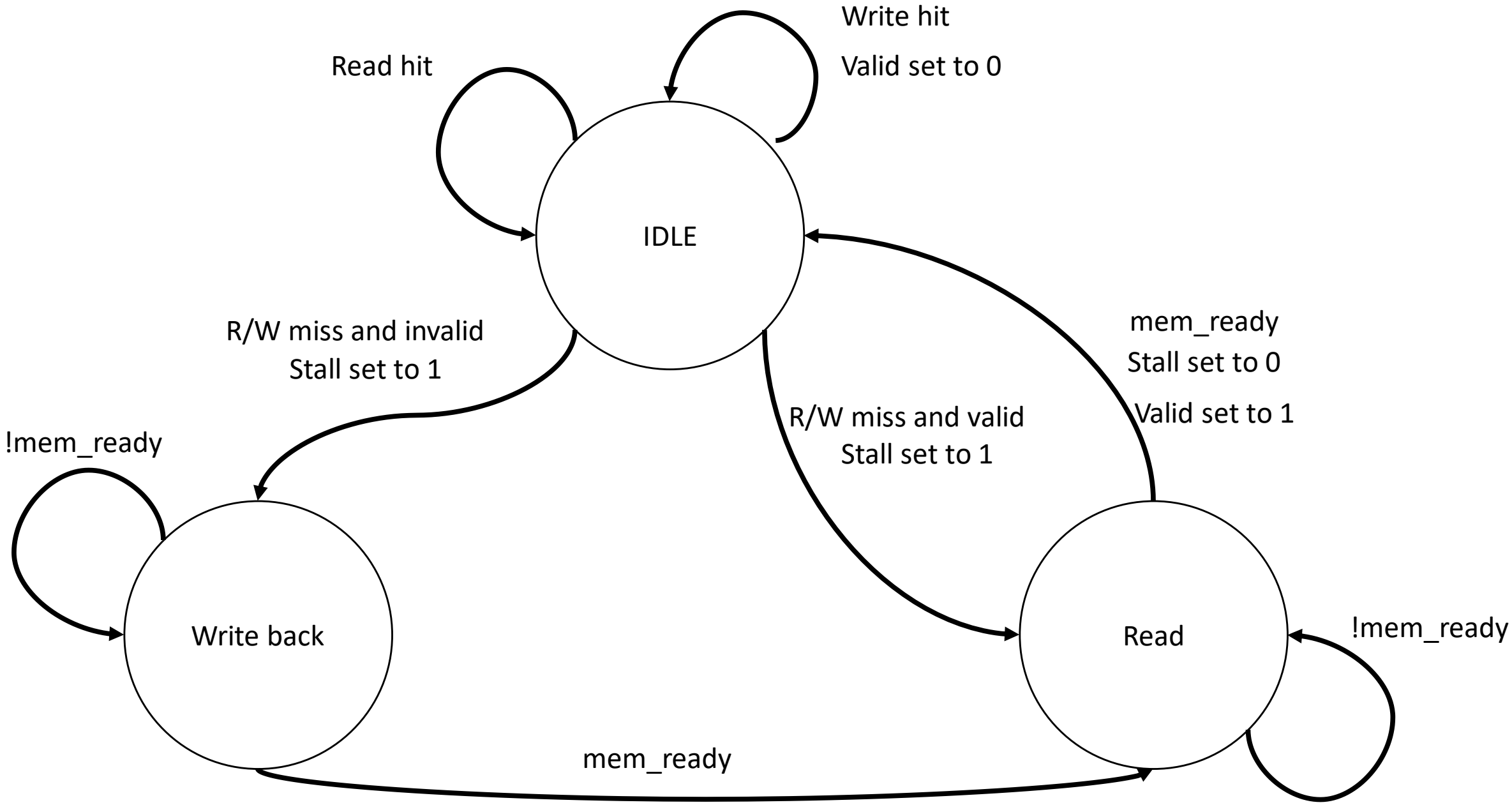
- Arrangement
  - 32-bit word
  - 4-word block, 8 blocks
  - 1-way cache
  - Write back, write allocate
    - Reset: set valid to 1, avoid writing back invalid data to memory
    - Known issue: need a dirty bit





Cache

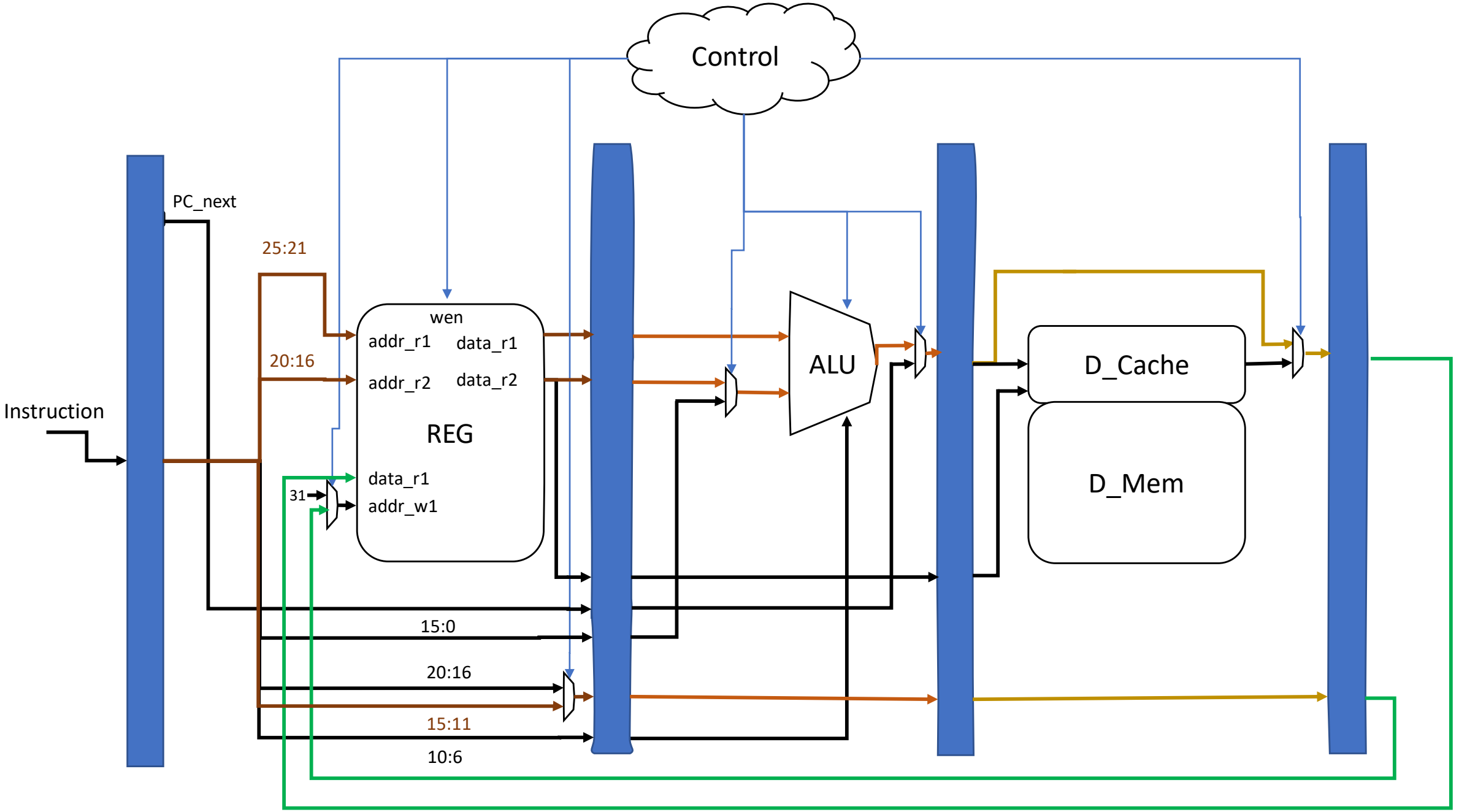

Processor
read
write
addr (30)
w_data (32)
r_data (32)
stall
rst
clk
read
write
addr (28)
w_data (128)
r_data (128)
ready
Memory


Write hit
Valid set to 0
Read hit
IDLE
R/W miss and invalid
Stall set to 1
mem_ready
Stall set to 0
Valid set to 1
R/W miss and valid
Stall set to 1
!mem_ready
Write back
Read
!mem_ready
mem_ready

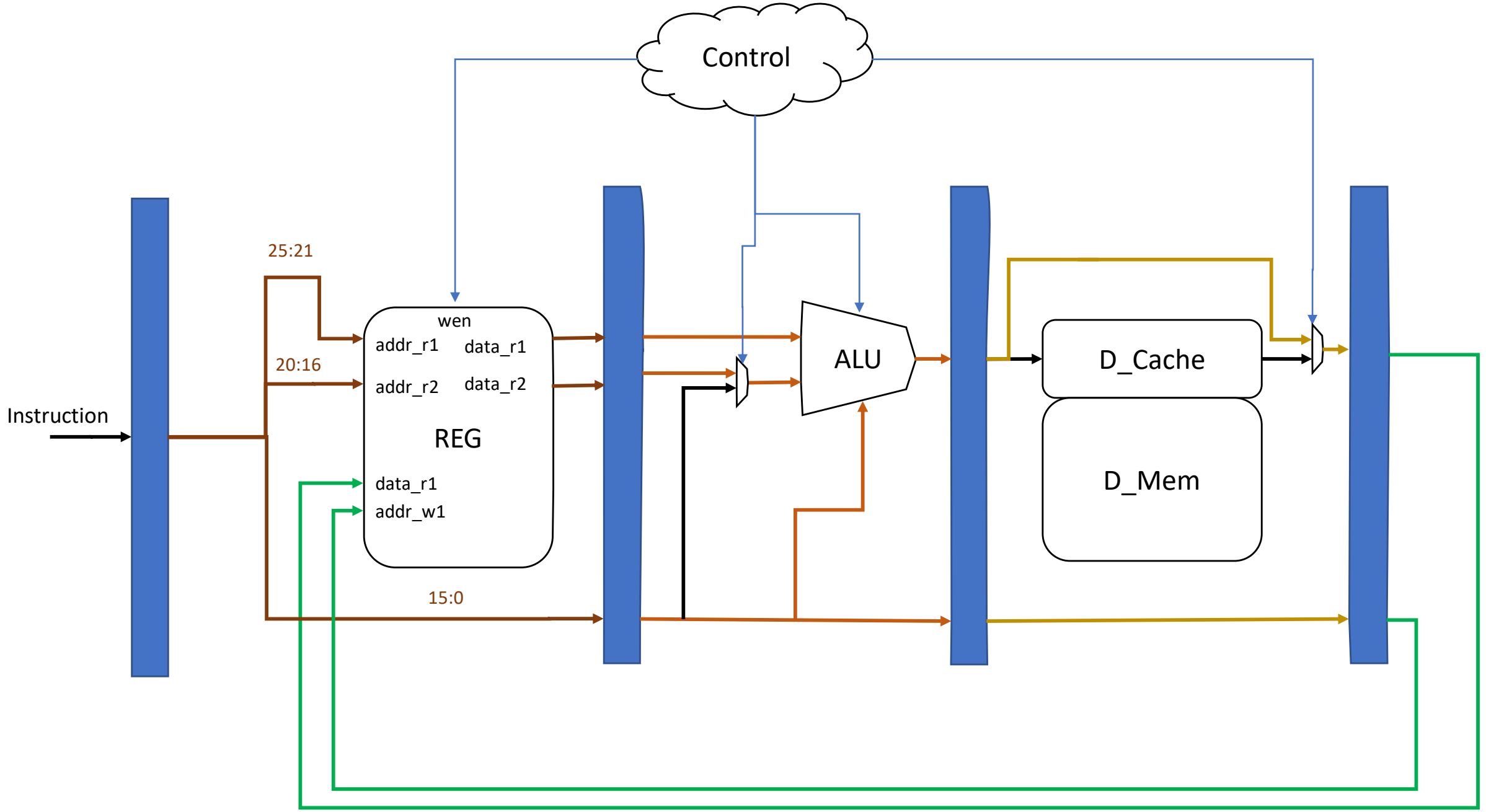
Outline

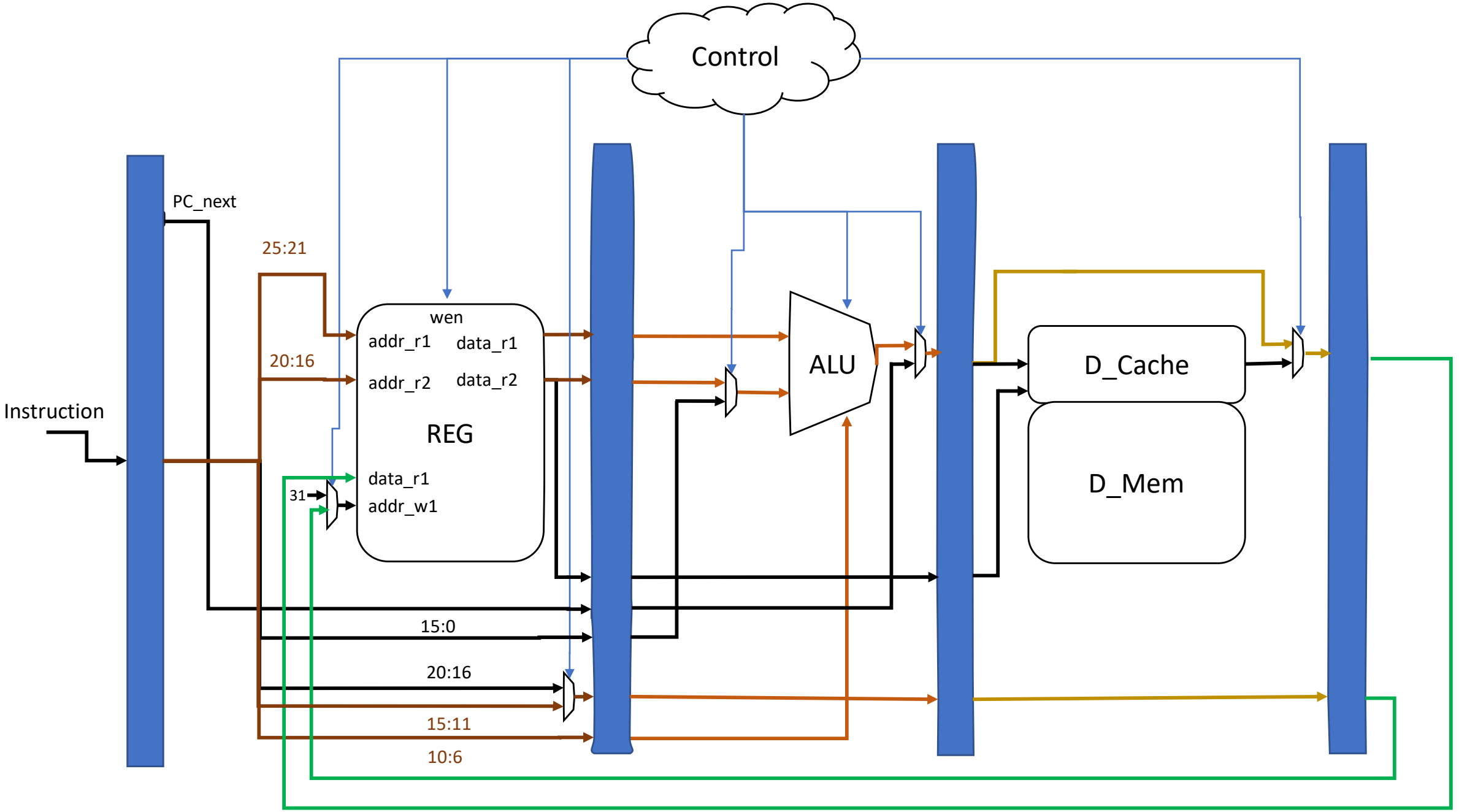
- Design Specification
- Hardware
  - Cache Implementation
  - MIPS Implementation
    - Instructions
    - Hazard Handling
  - Performance

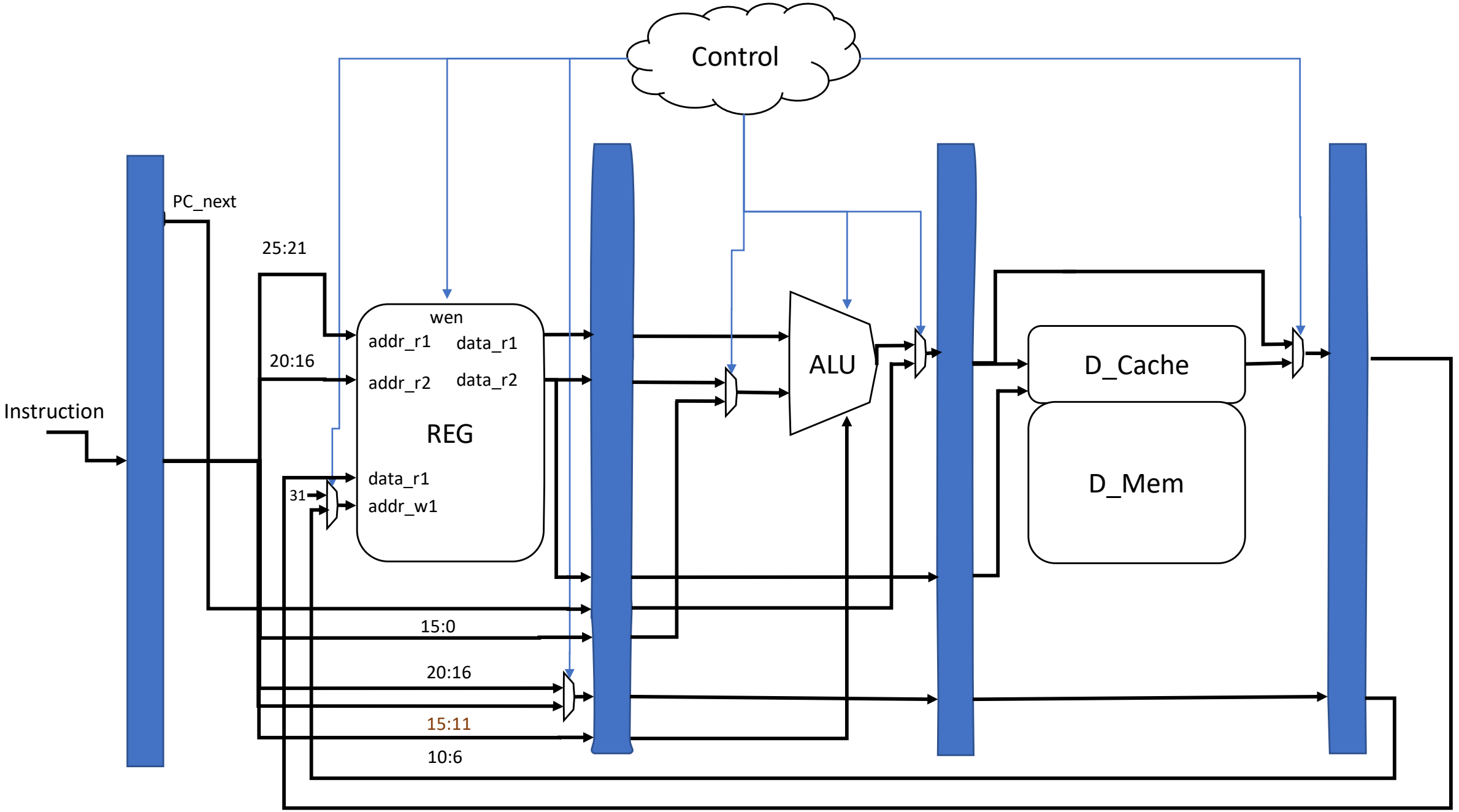
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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


Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

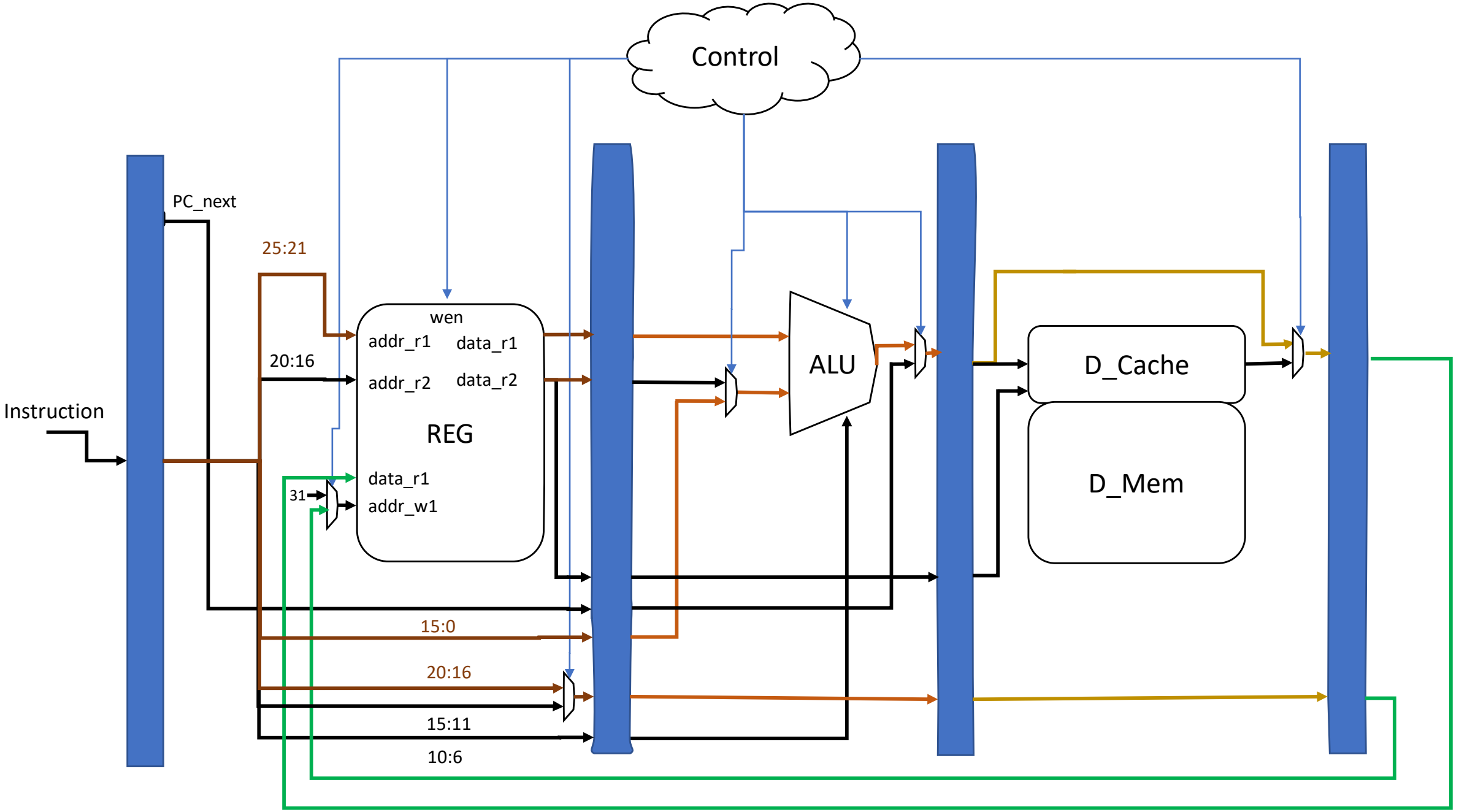
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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


Control
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
ALU
D_Cache
D_Mem

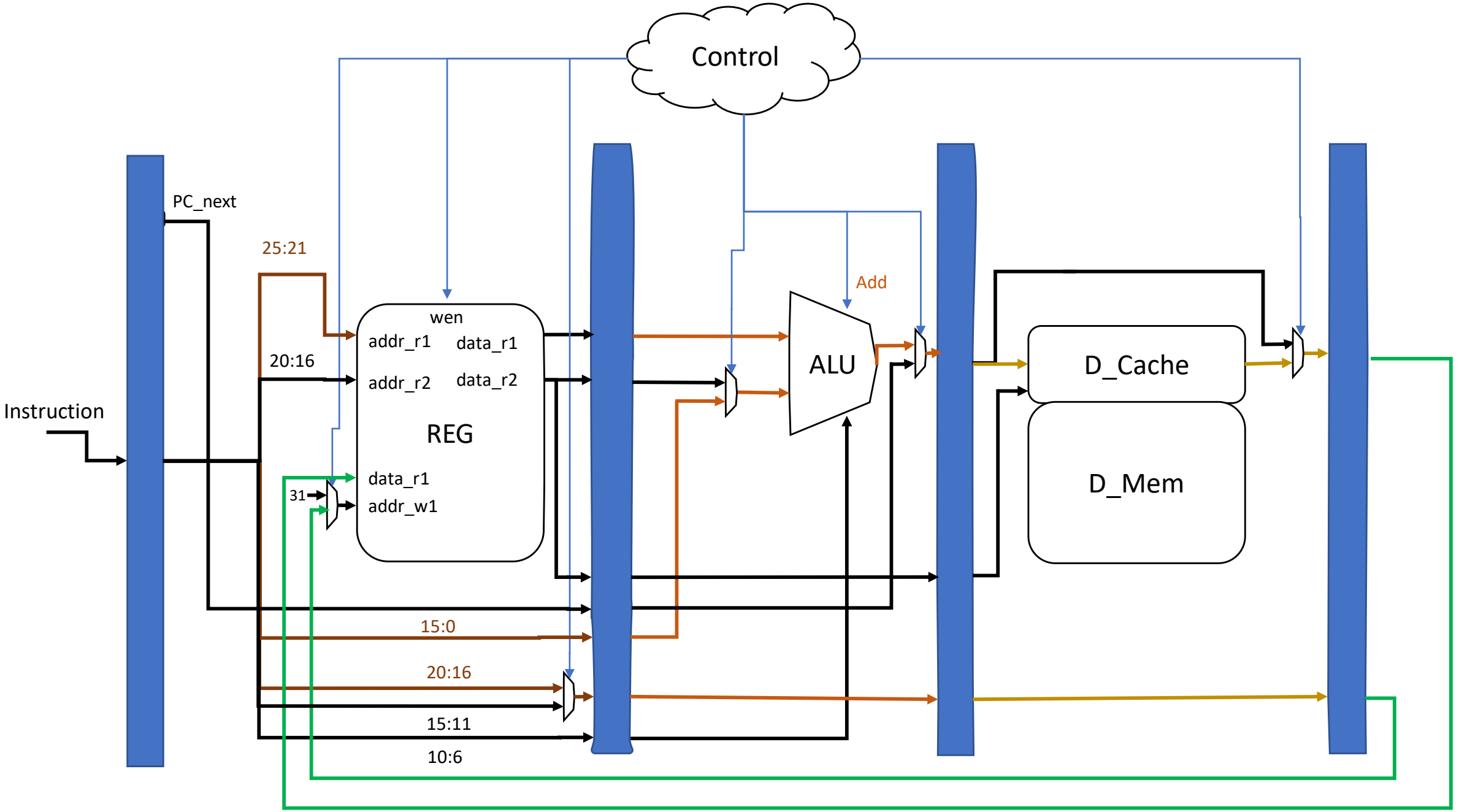

Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6


Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

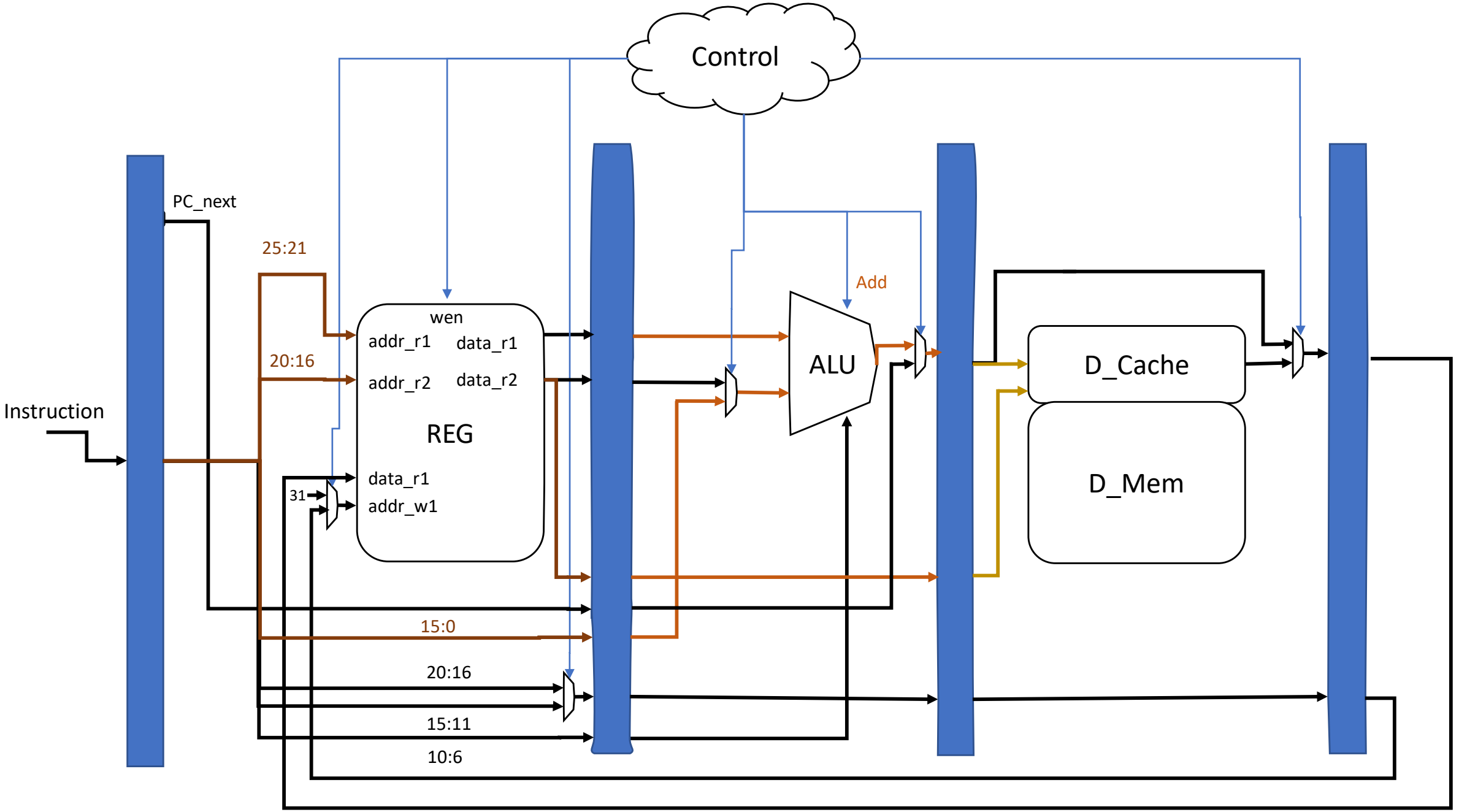
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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


Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

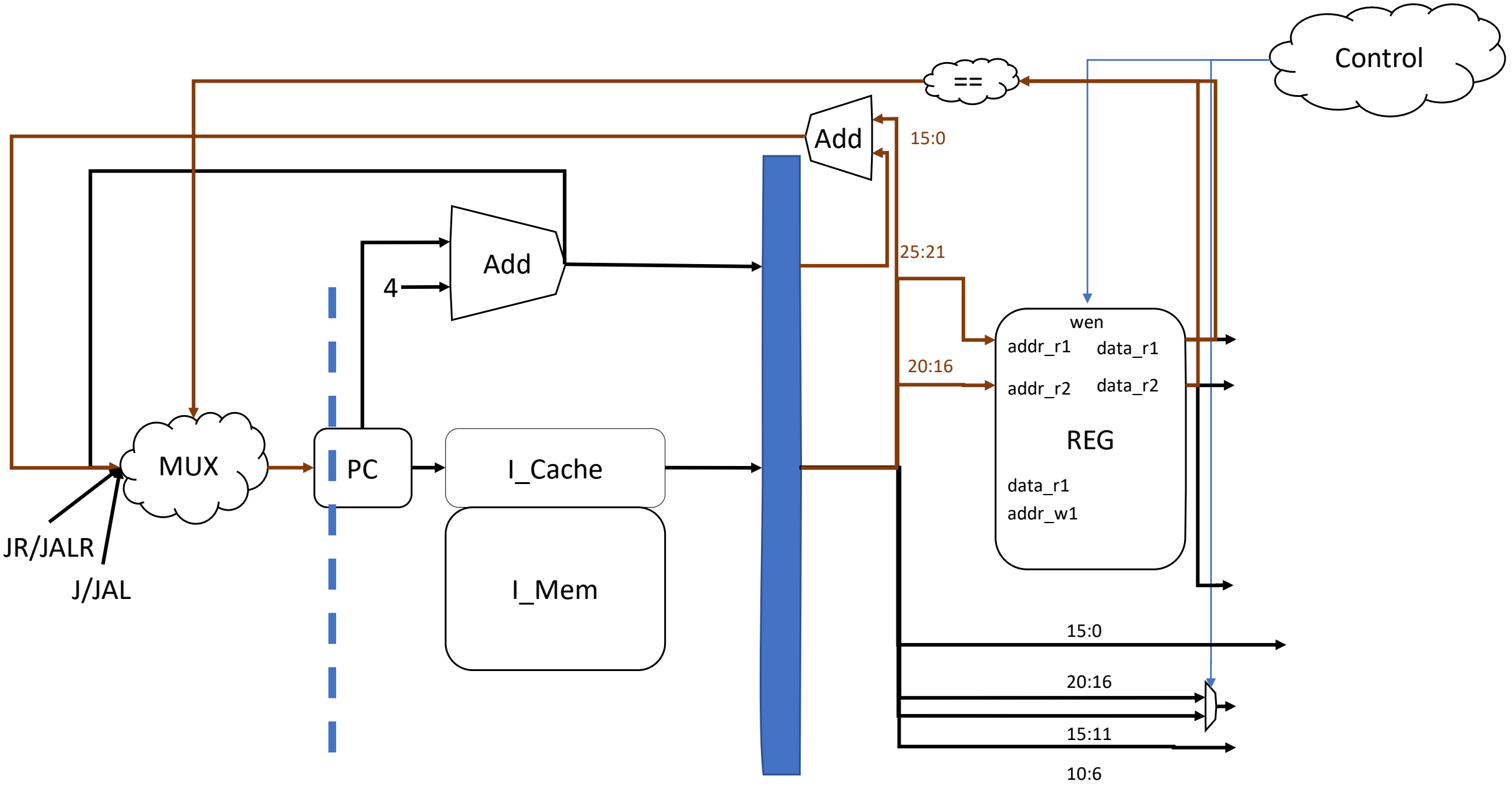
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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


Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
Add
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

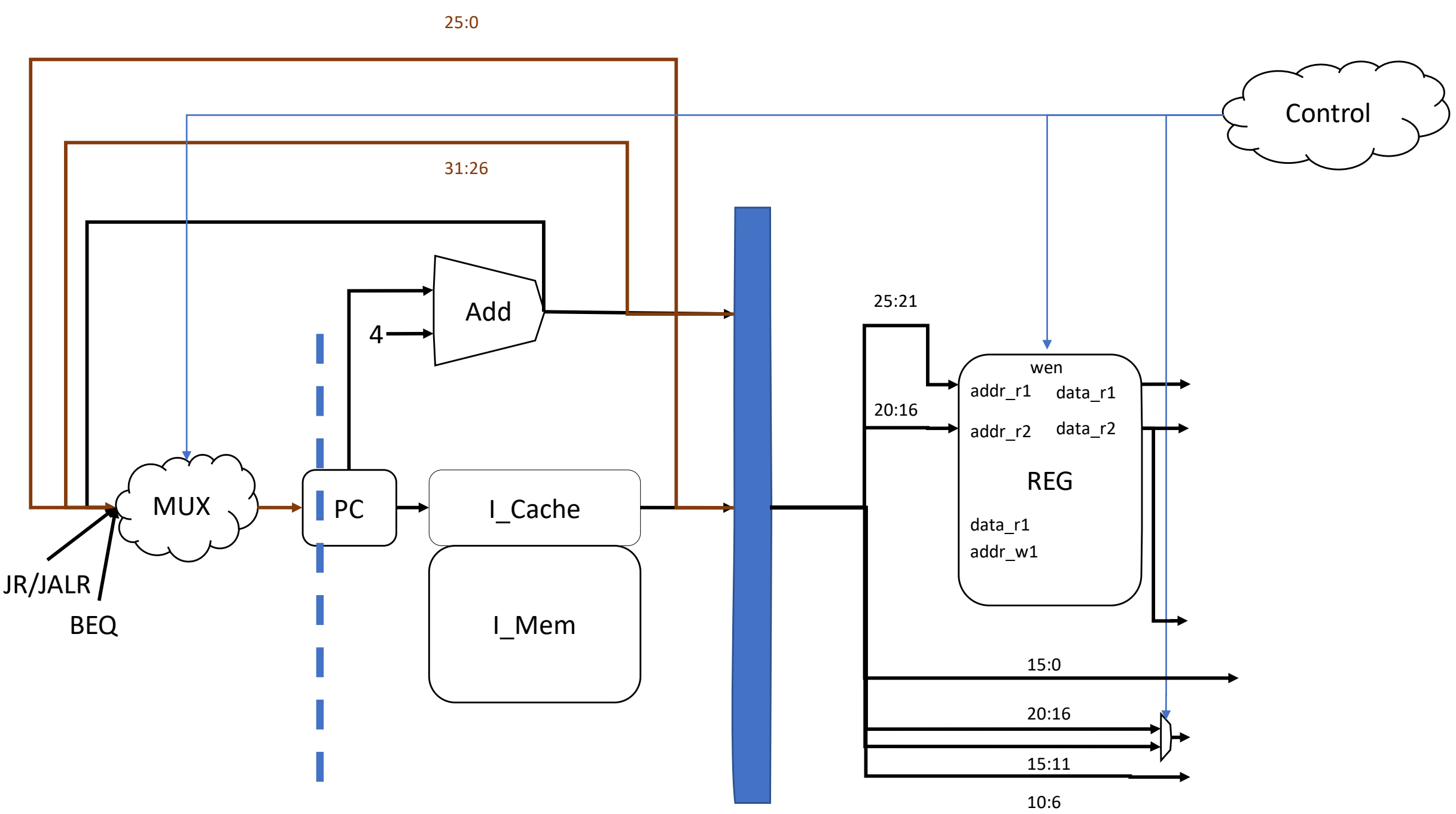
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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


Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
Add
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

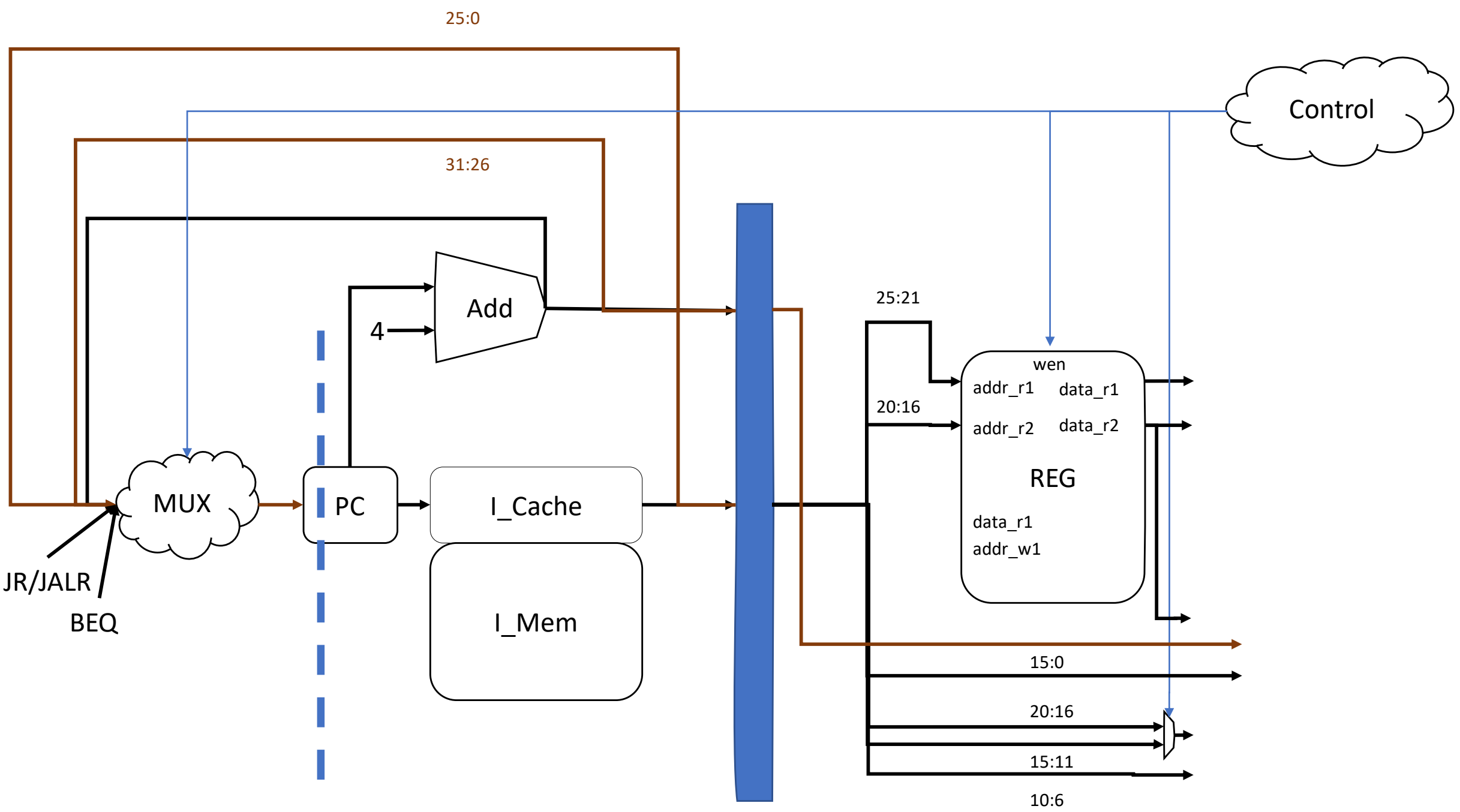
				addr(26)(J type)				
						const(16)(I type)		
Name	Type	Description	OP(6)	rs(5)	rt(5)	rd(5)	shamt(5)	funct(6)
ADD	R	Addition, overflow detection for signed operand is not required	000000	rs	rt	rd	-	100000
ADDI	I	Addition immediate with sign-extension, without overflow detection	001000	rs	rt	const		
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

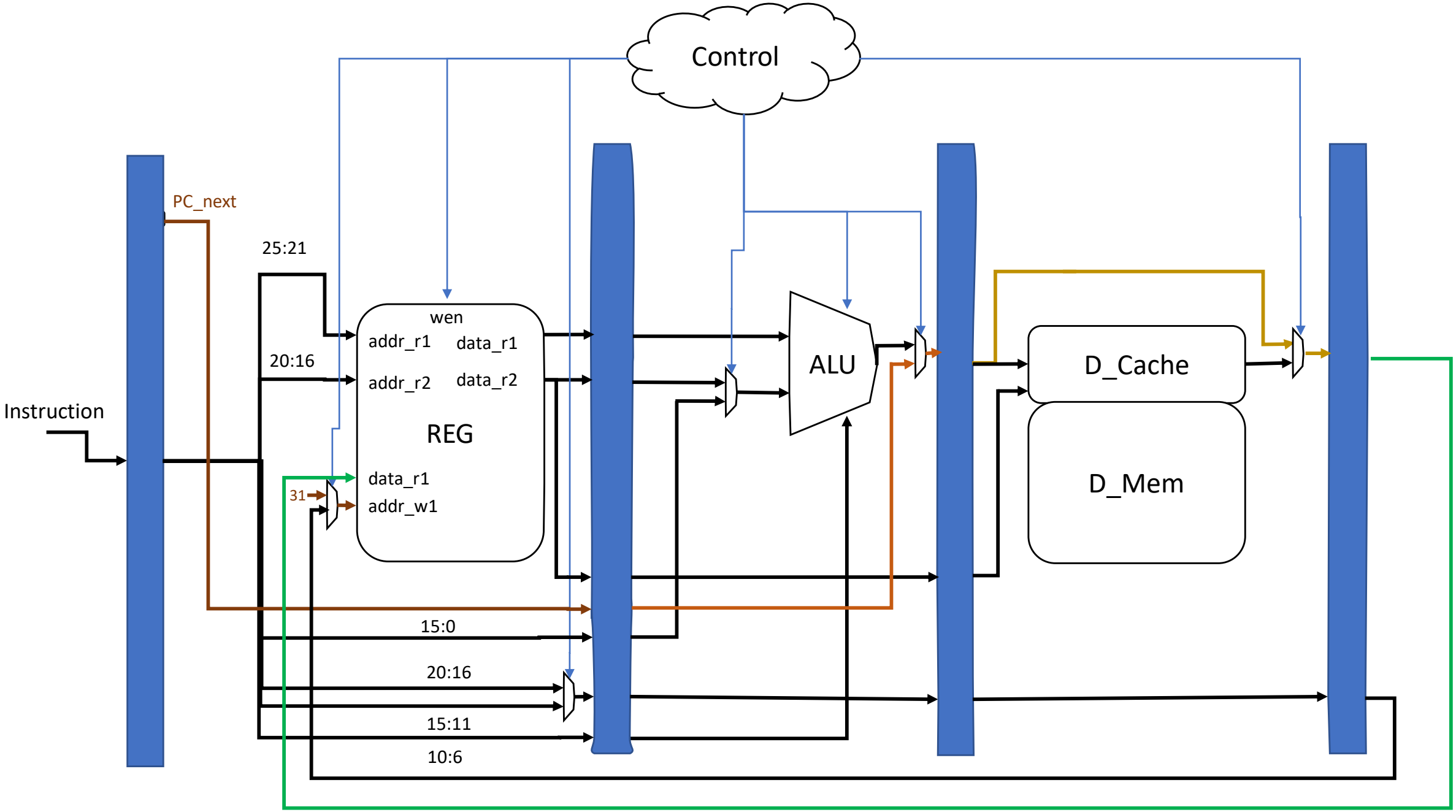

Control
==
Add
15:0
Add
4
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
MUX
PC
I_Cache
I_Mem
JR/JALR
J/JAL
15:0
20:16
15:11
10:6

				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

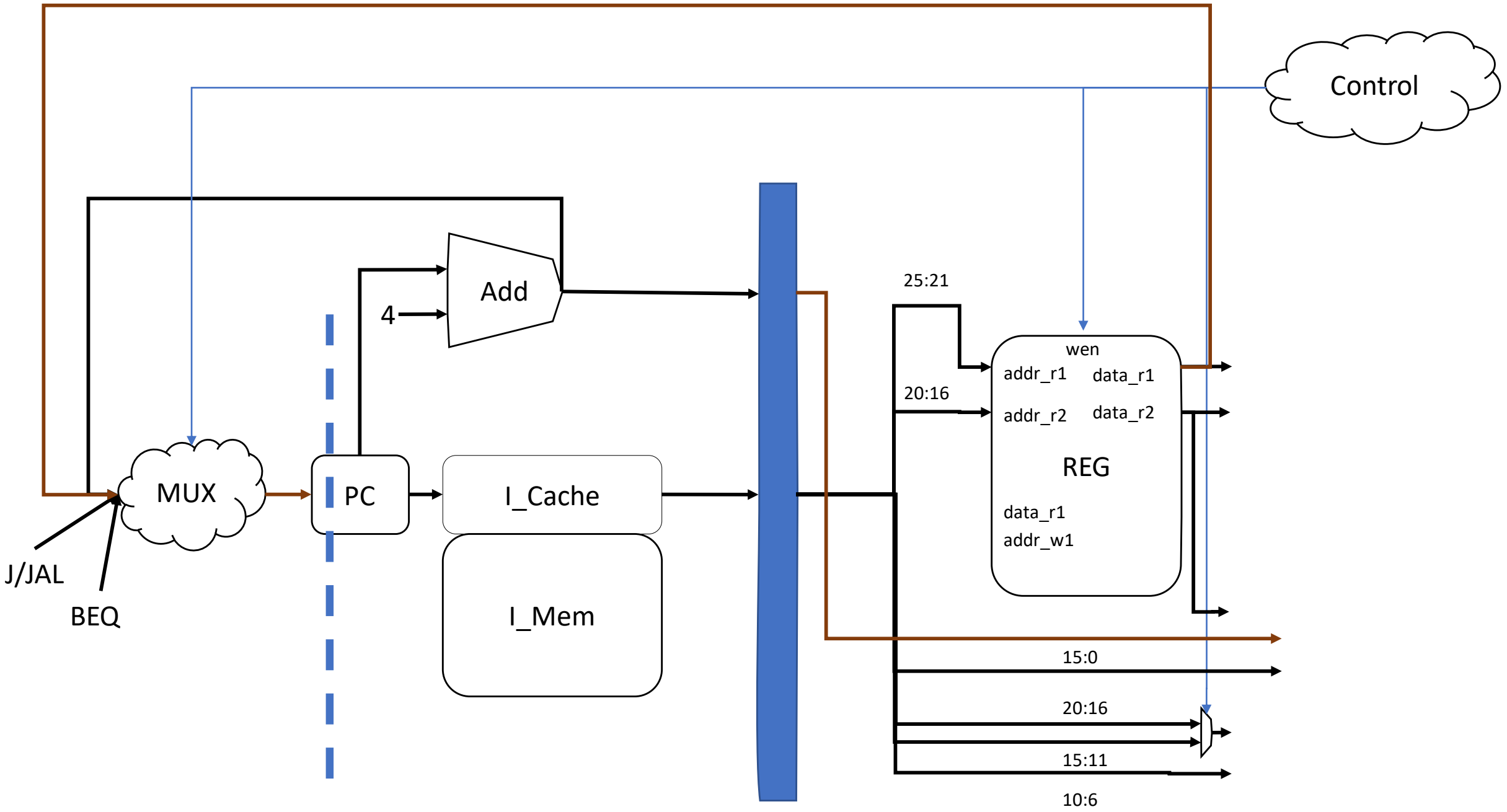

25:0
Control
31:26
Add
4
25:21
wen
addr_r1
data_r1
20:16
addr_r2
data_r2
REG
data_r1
addr_w1
MUX
PC
I_Cache
JR/JALR
BEQ
I_Mem
15:0
20:16
15:11
10:6

				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

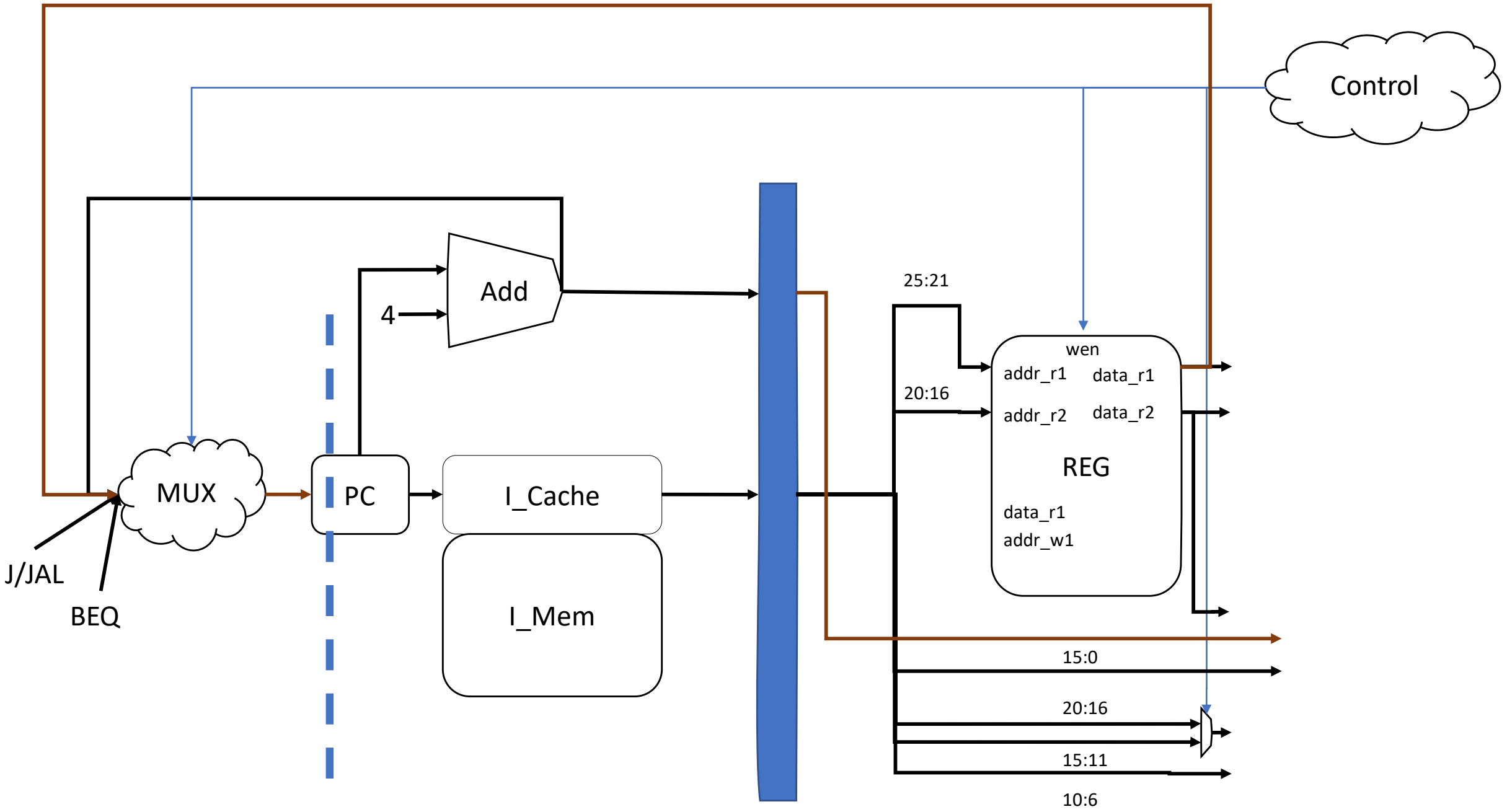

25:0
Control
31:26
Add
4
25:21
wen
addr_r1
data_r1
20:16
addr_r2
data_r2
REG
data_r1
addr_w1
MUX
PC
I_Cache
I_Mem
JR/JALR
BEQ
15:0
20:16
15:11
10:6

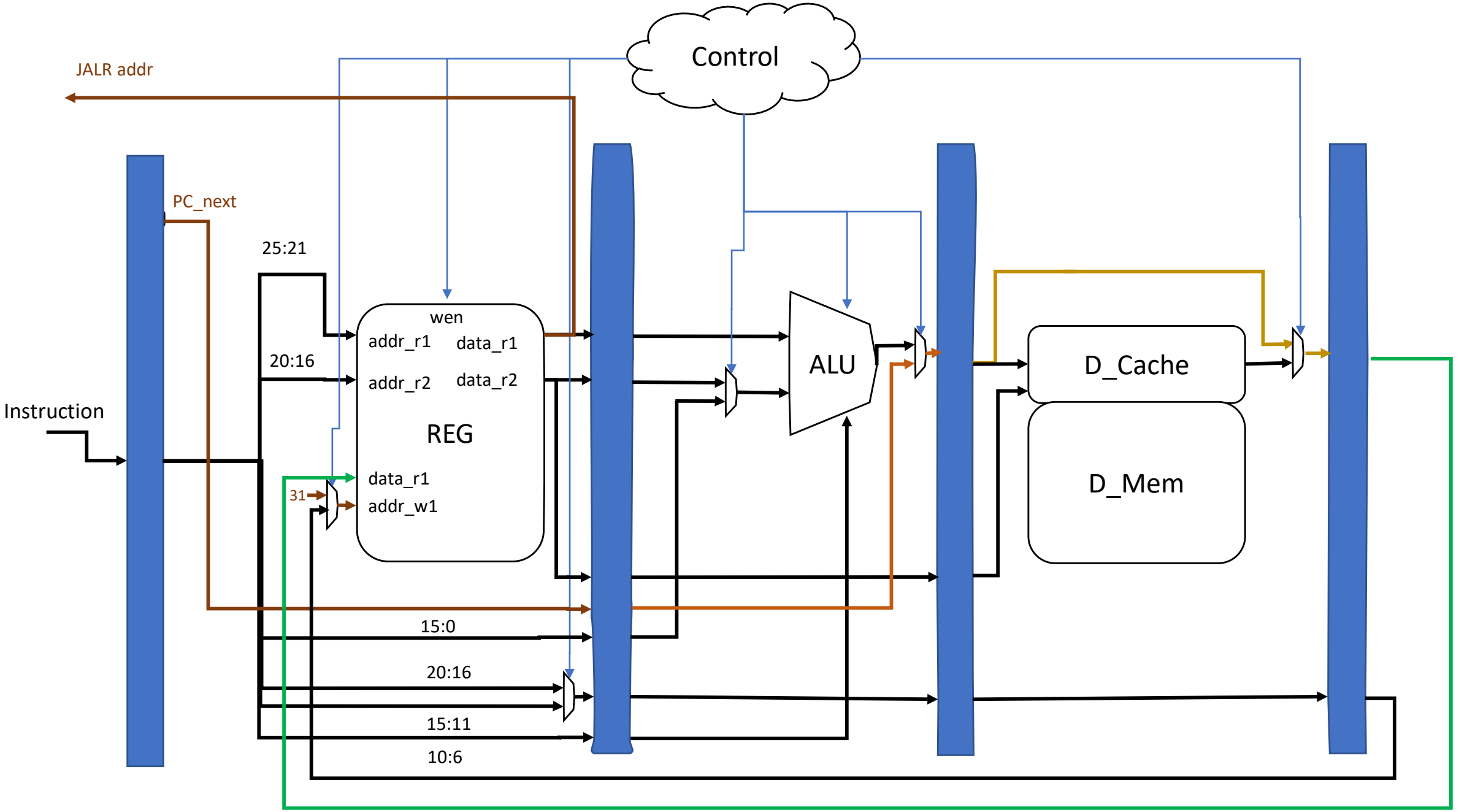

Control
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

				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


Control
Add
4
MUX
PC
I_Cache
I_Mem
J/JAL
BEQ
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
20:16
15:11
10:6

				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

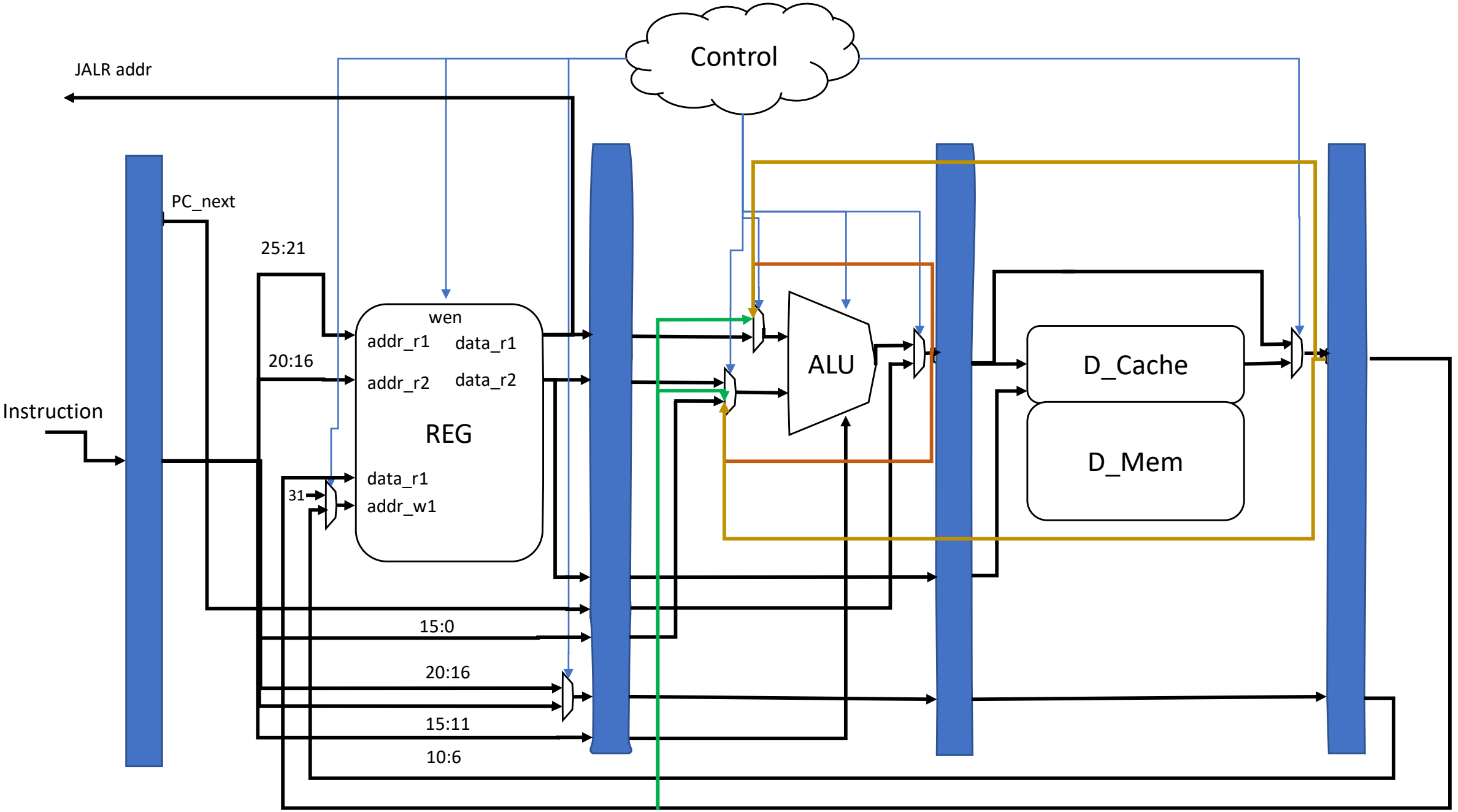

Control
Add
4
MUX
PC
I_Cache
I_Mem
J/JAL
BEQ
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
20:16
15:11
10:6


Control
JALR addr
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

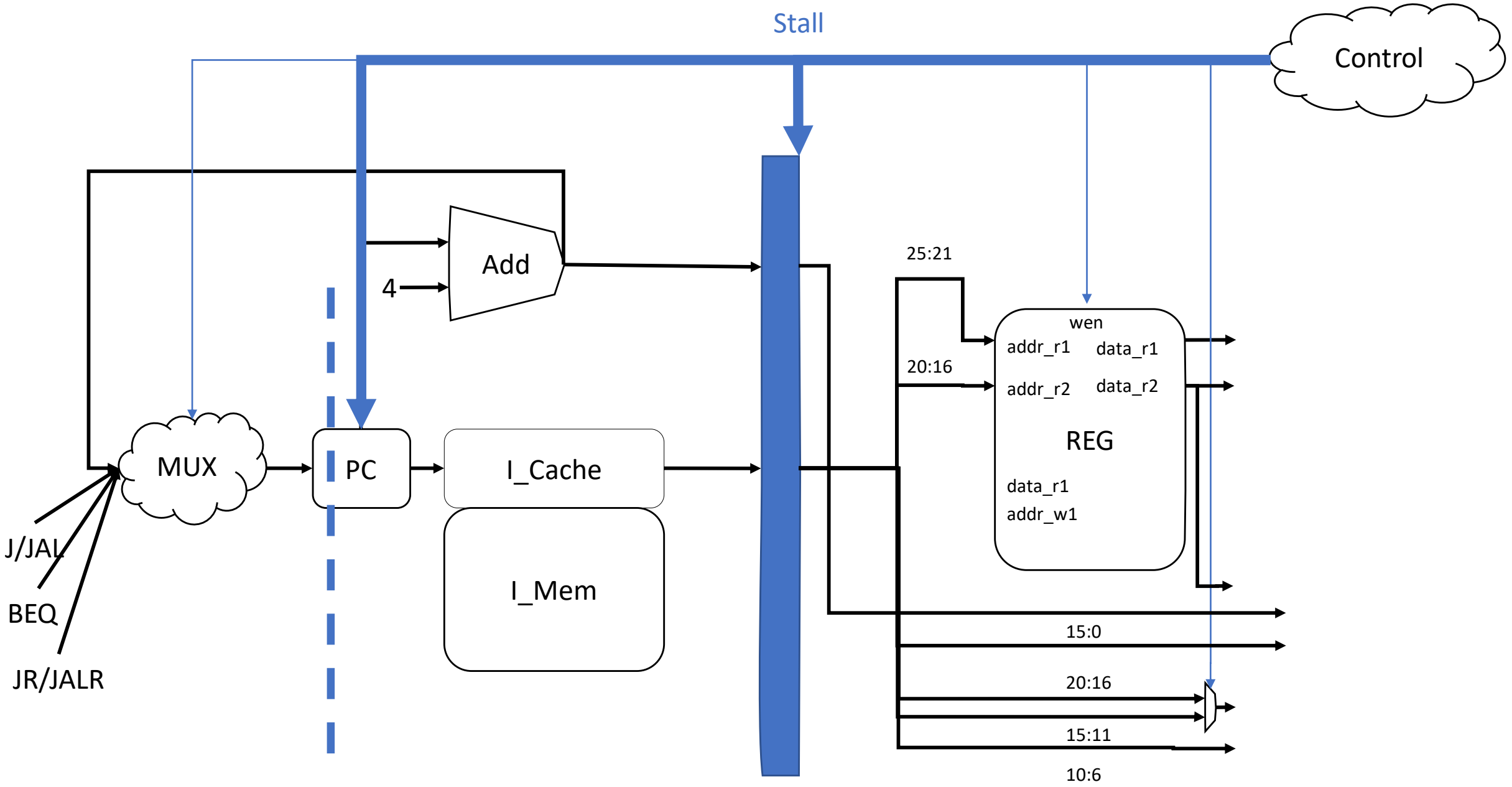
Outline

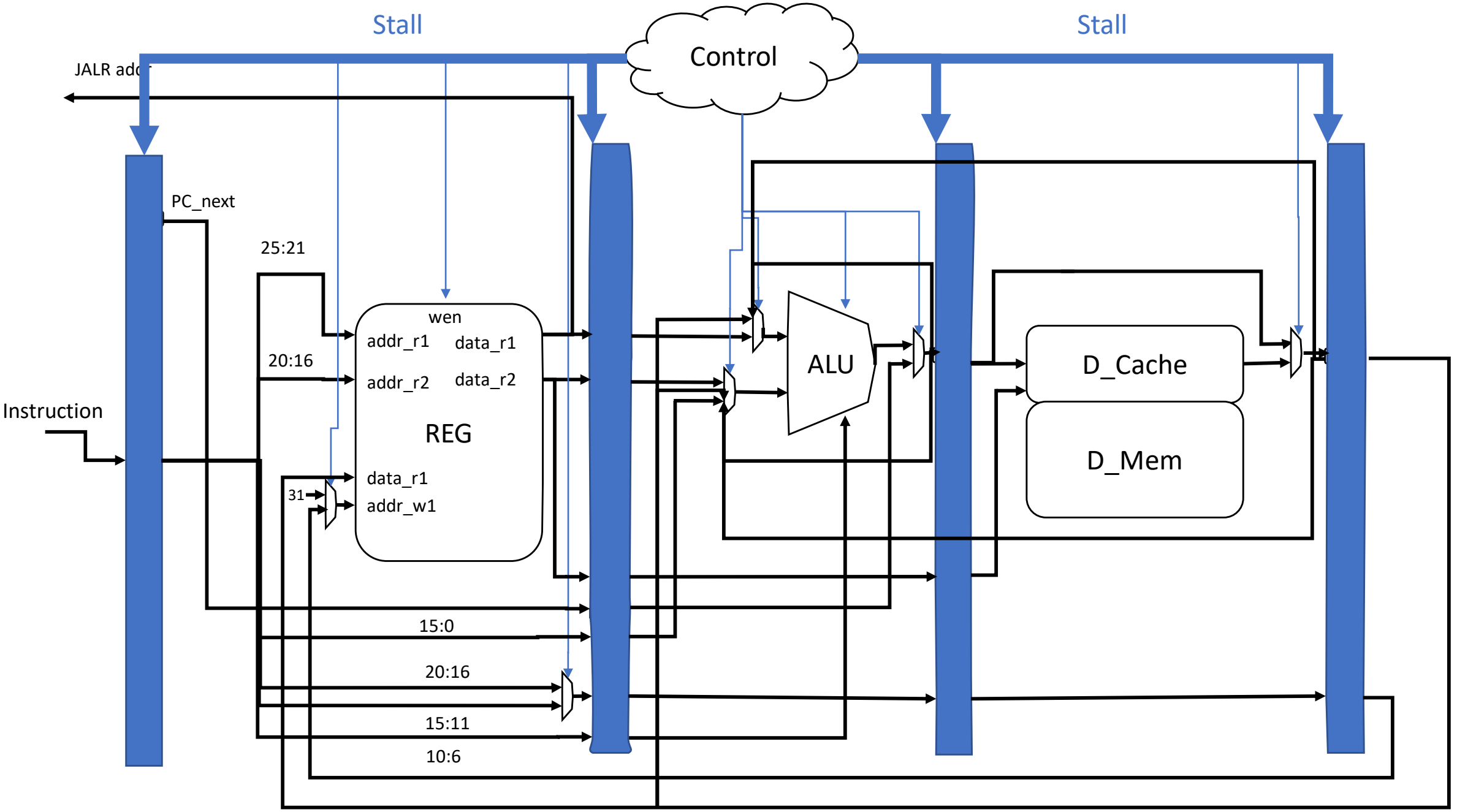
- Design Specification
- Hardware
  - Cache Implementation
  - MIPS Implementation
    - Instructions
    - Hazard Handling
  - Performance

Type	Solution
Data Hazard	Forwarding
Load-use Hazard	Stall
Jump/Branch	Flush

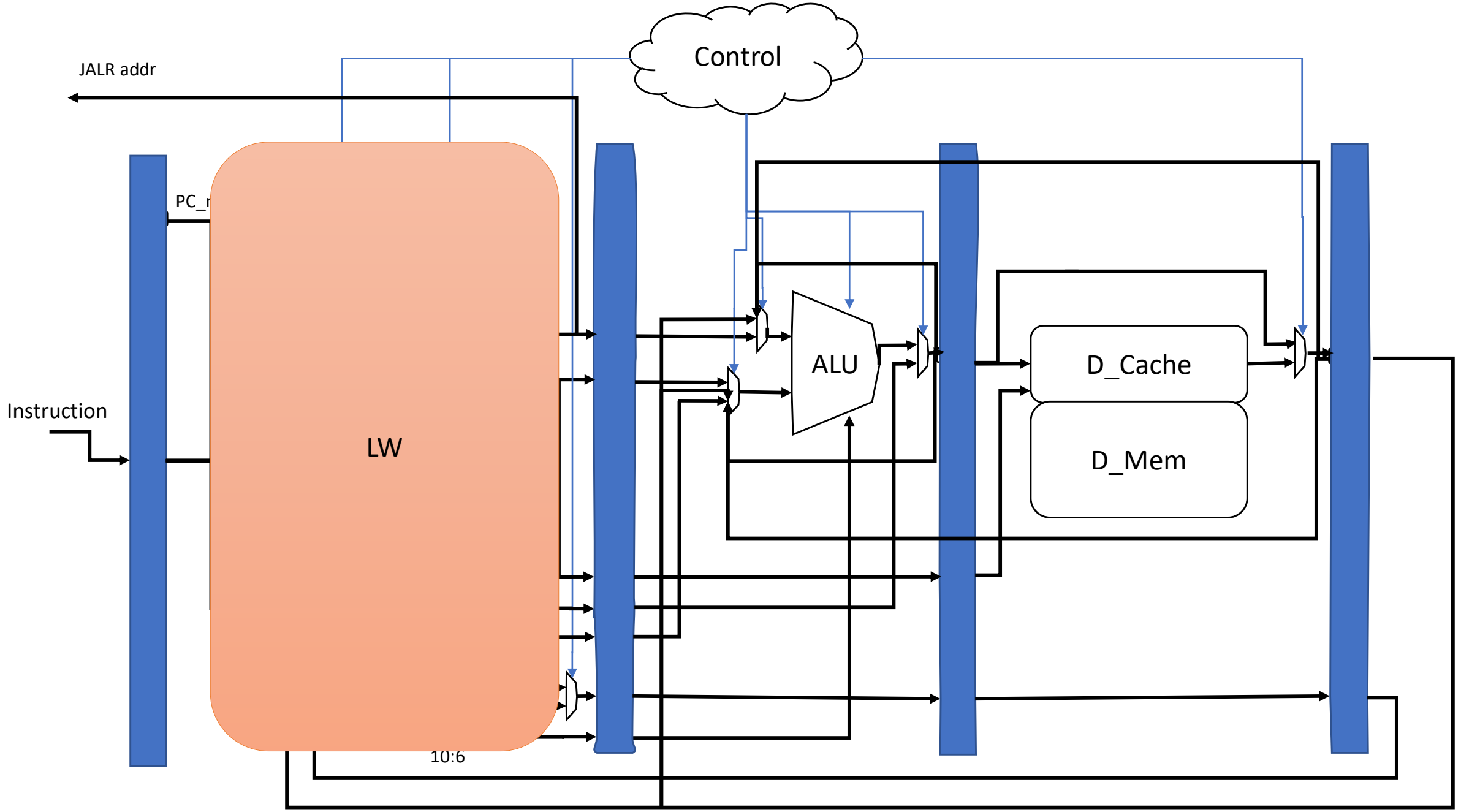

Control
JALR addr
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

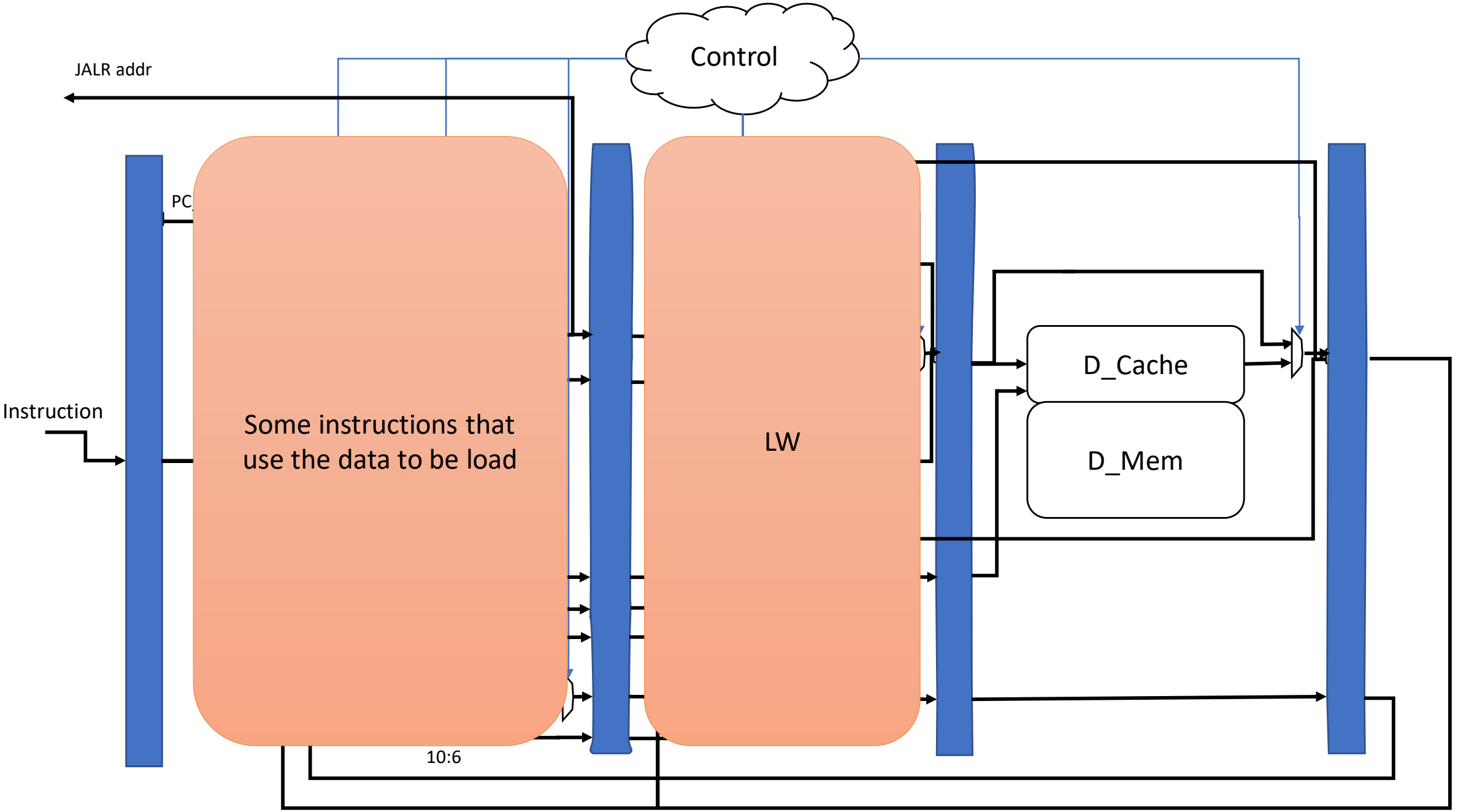
Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Stall & Insert Bubble
Jump/Branch	Flush

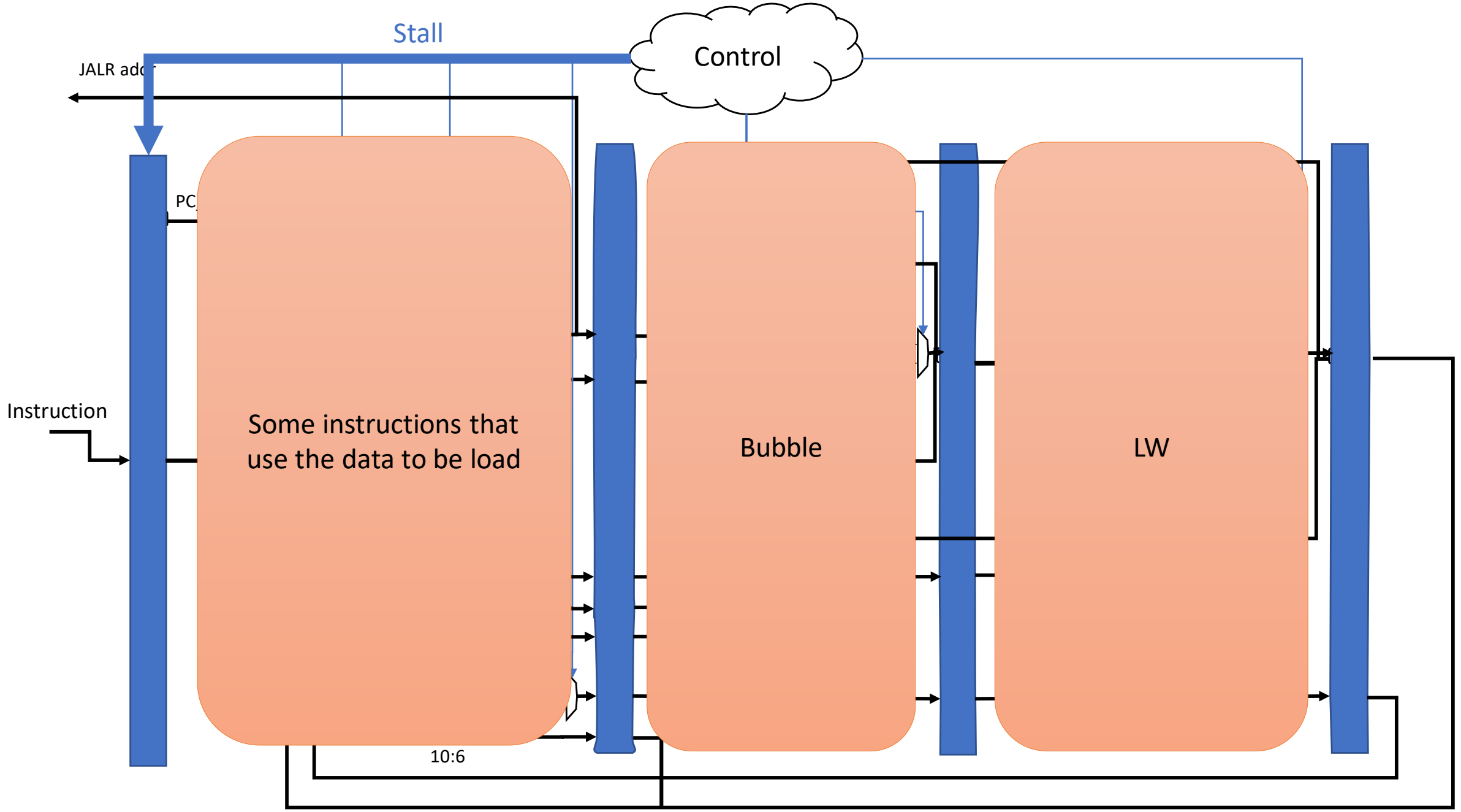

Stall
Control
Add
4
MUX
PC
I_Cache
I_Mem
J/JAL
BEQ
JR/JALR
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
20:16
15:11
10:6


Stall
Control
Stall
JALR addr
PC_next
25:21
20:16
Instruction
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
31
ALU
D_Cache
D_Mem
15:0
20:16
15:11
10:6

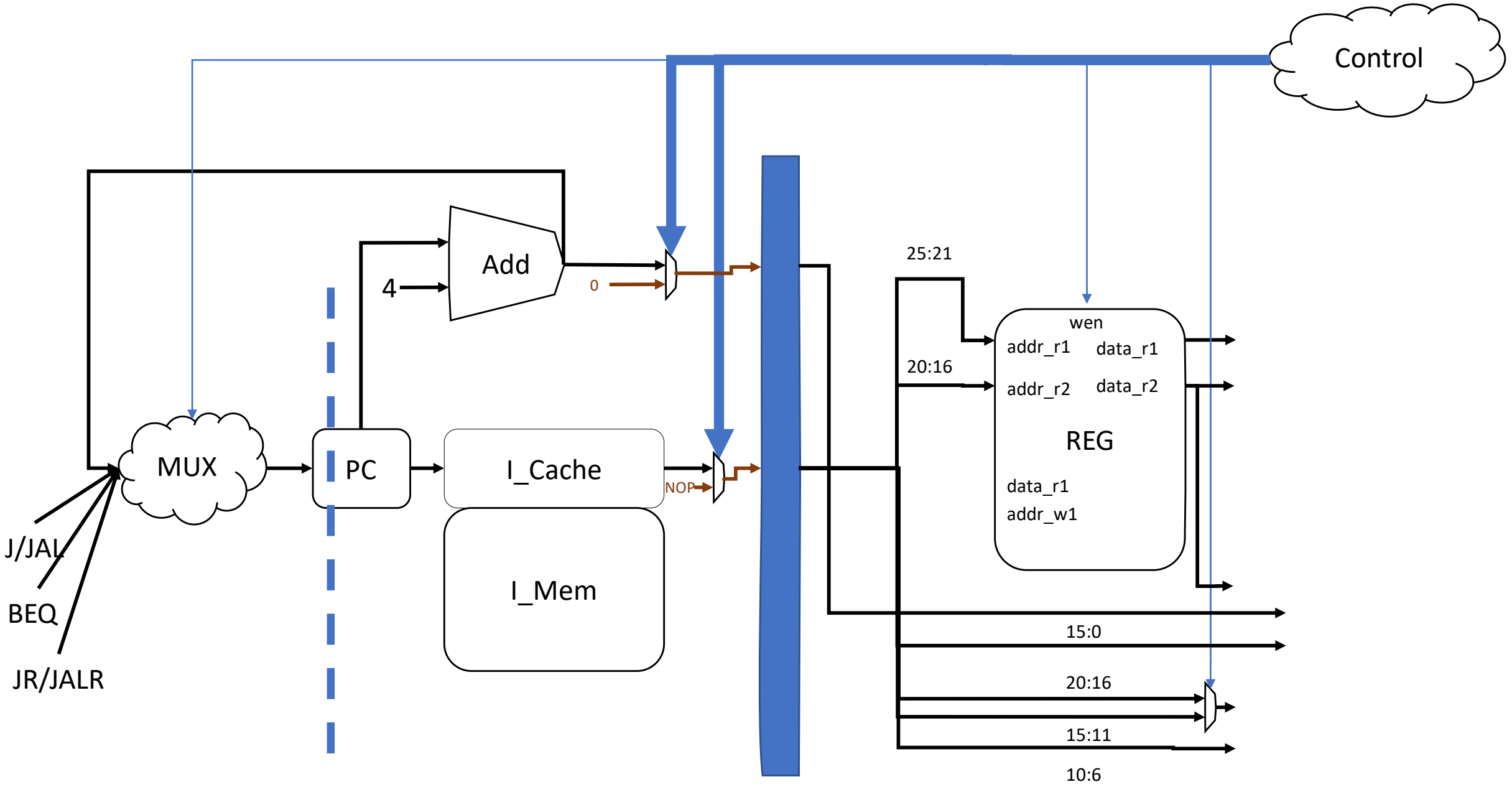
Type	Solution
Data Hazard	Forwarding
Memory Stall	Stall
Load-use Hazard	Insert Bubble & Stall
Jump/Branch	Flush


Control
JALR addr
PC_r
Instruction
LW
ALU
D_Cache
D_Mem
10:6


Control
JALR addr
PC
Instruction
Some instructions that use the data to be load
LW
D_Cache
D_Mem
10:6


Stall
Control
JALR add
PC
Instruction
Some instructions that use the data to be load
Bubble
LW
10:6

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

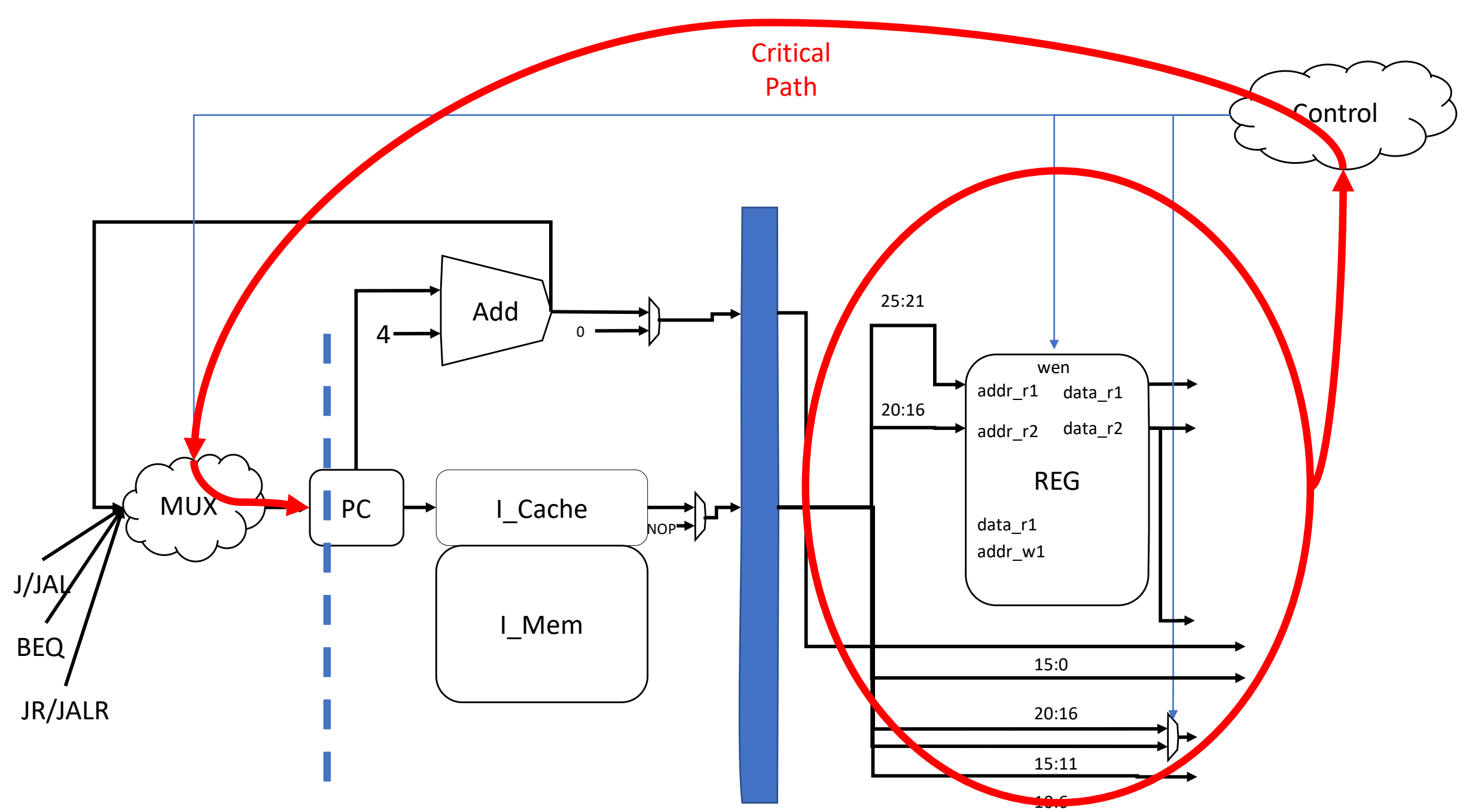

Control
Add
4
0
MUX
PC
I_Cache
NOP
I_Mem
J/JAL
BEQ
JR/JALR
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
20:16
15:11
10:6

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
  - 286617um^2


Critical
Path
Control
Add
4
0
MUX
PC
I_Cache
NOP
I_Mem
J/JAL
BEQ
JR/JALR
25:21
20:16
wen
addr_r1
data_r1
addr_r2
data_r2
REG
data_r1
addr_w1
15:0
20:16
15:11