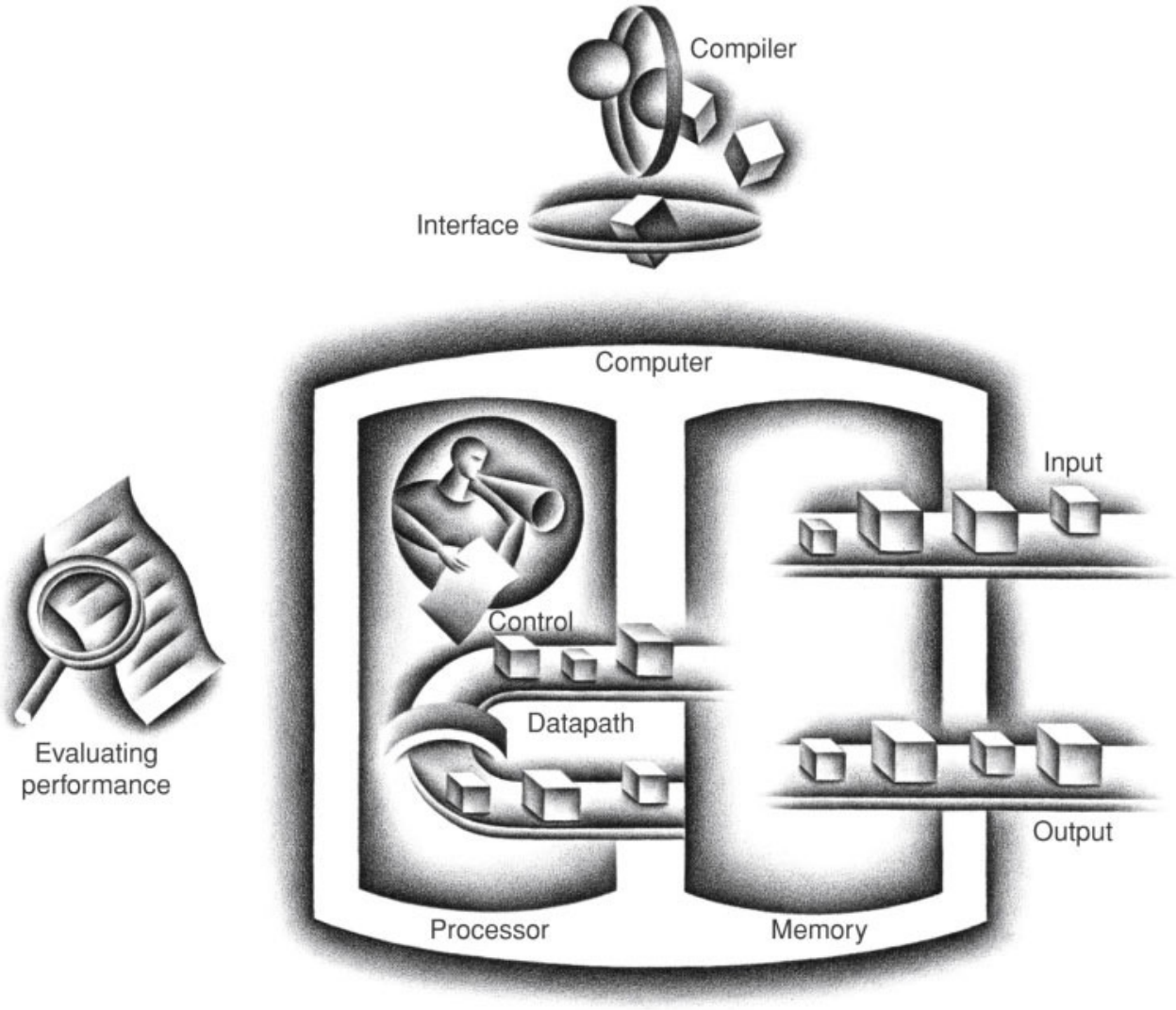
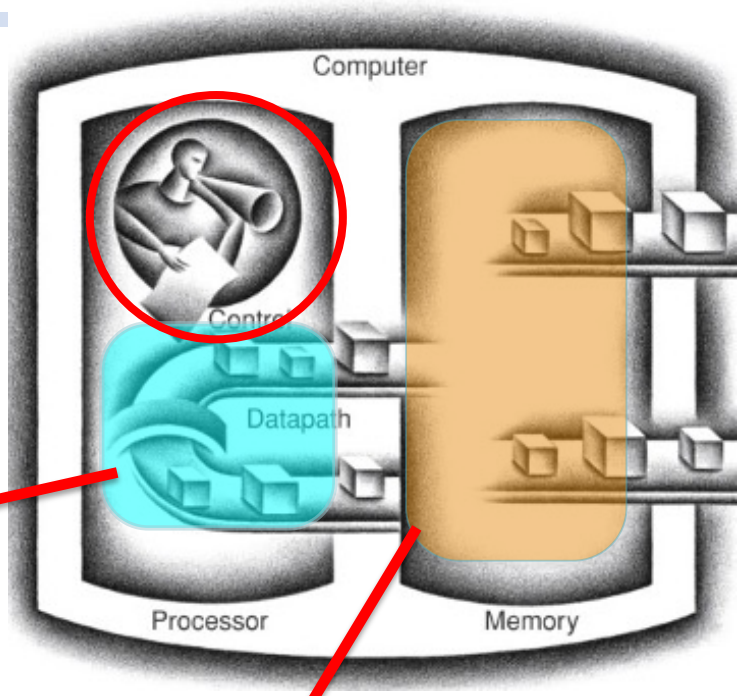


FIGURE 1.5 The organization of a computer, showing the five classic components. The processor gets instructions and data from memory. Input writes data to memory, and output reads data from memory. Control sends the signals that determine the operations of the datapath, memory, input, and output.

CPU Overview



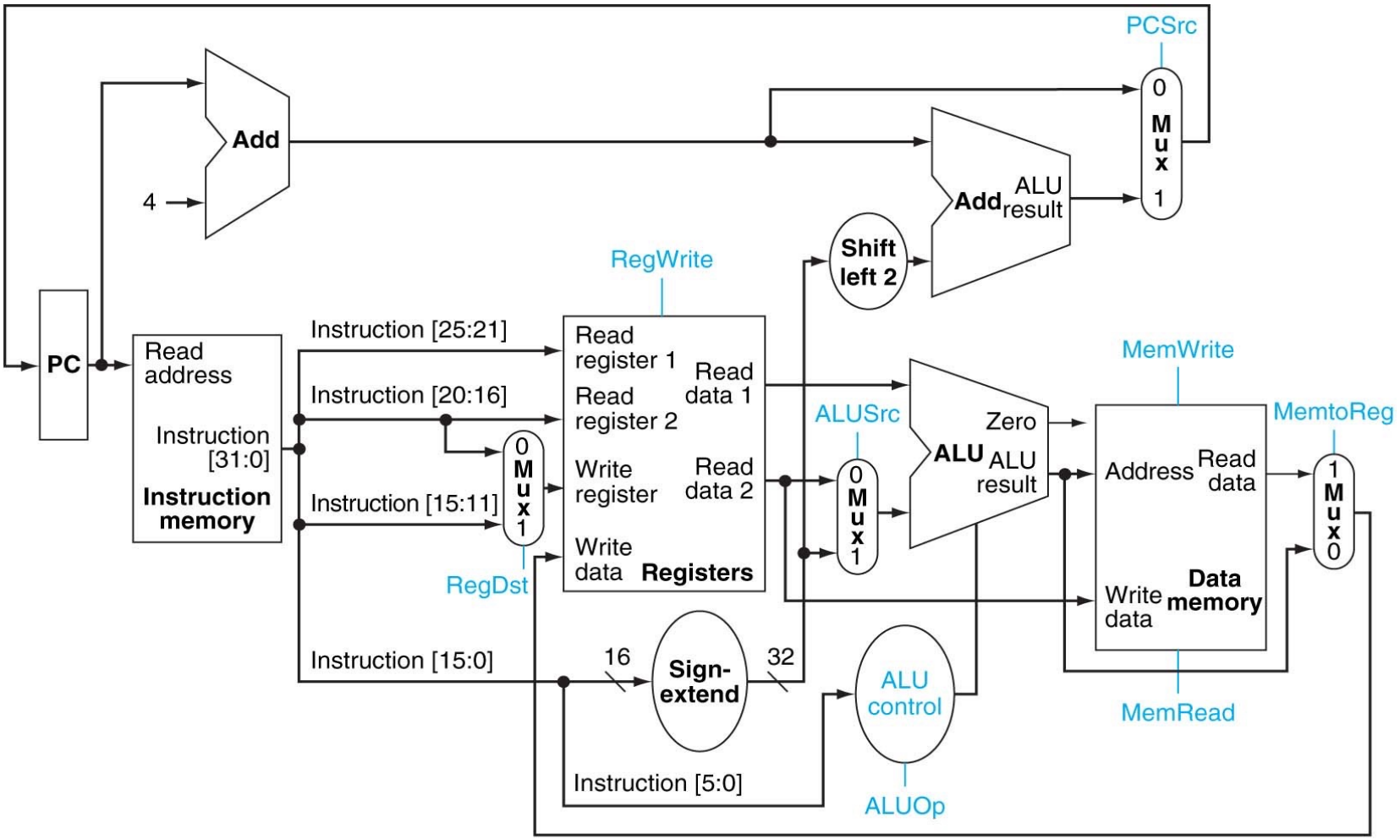
Processor

4
Add
Add
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
ALU
Address
Data memory
Data
Memory

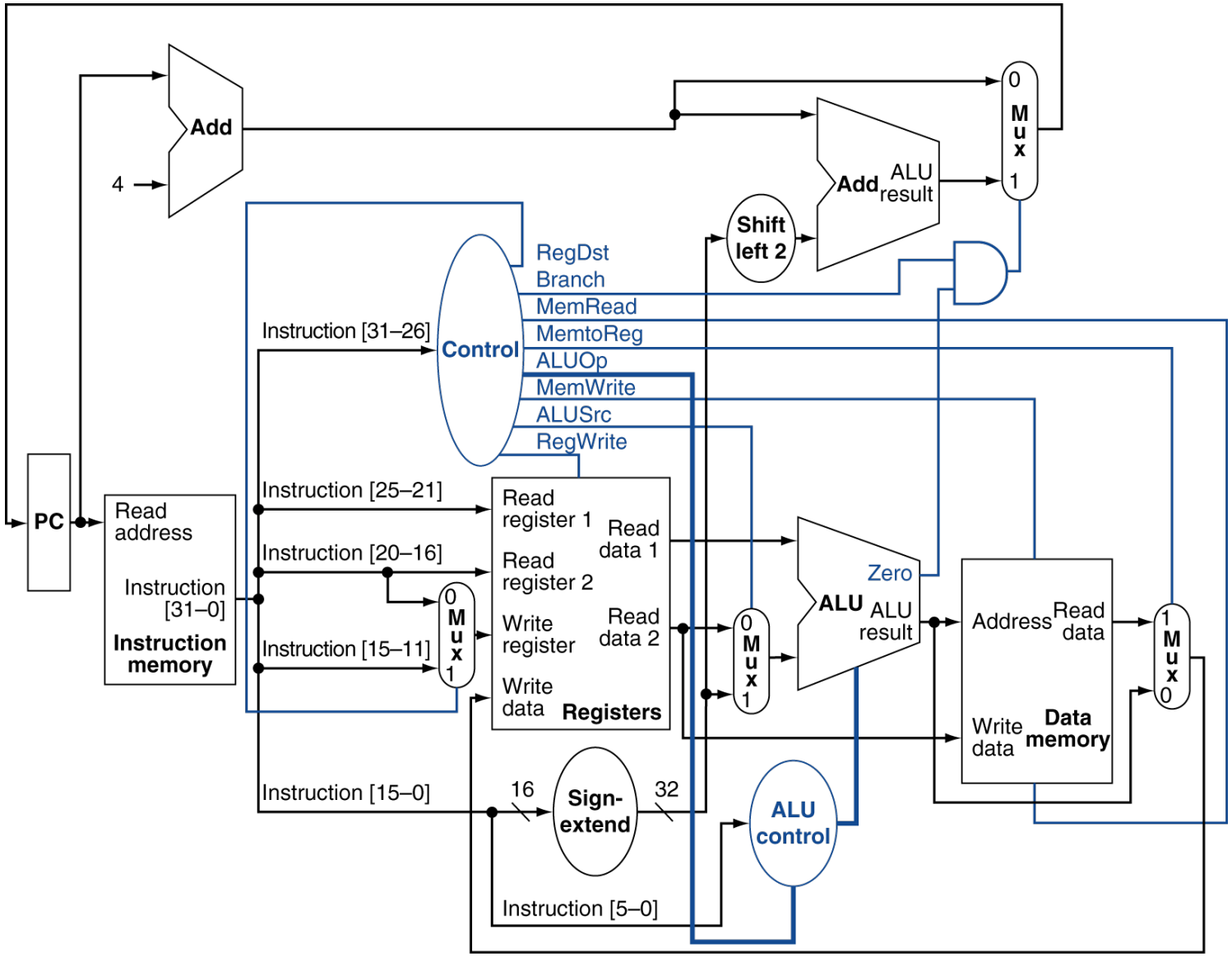
Harvard Architecture vs. Von Neumann Architecture

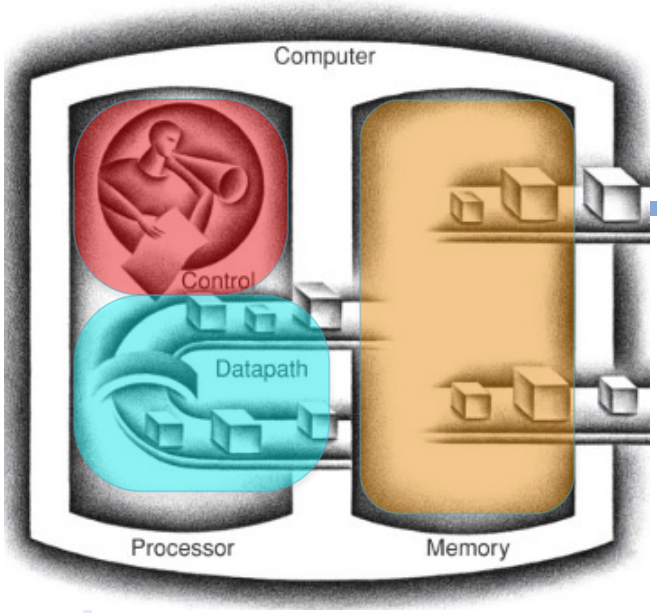
- von Neumann Architecture
  - Same memory holds data, instructions.
  - A single set of address/data buses between CPU and memory
- Harvard Architecture
  - Separate memories for data and instructions.
  - Two sets of address/data buses between CPU and memory

Datapath


PCSrc
Add
4
0
M
u
x
1
ALU
Add result
Shift
left 2
RegWrite
PC
Read
address
Instruction
[31:0]
Instruction
memory
Instruction [25:21]
Instruction [20:16]
Instruction [15:11]
Instruction [15:0]
Instruction [5:0]
Read
register 1
Read
register 2
Write
register
Write
data
Read
data 1
Read
data 2
Registers
RegDst
ALUSrc
Zero
ALU
ALU
result
MemWrite
MemtoReg
Address
Read
data
Write
data
Data
memory
MemRead
16
Sign-
extend
32
ALU
control
ALUOp

Datapath With Control

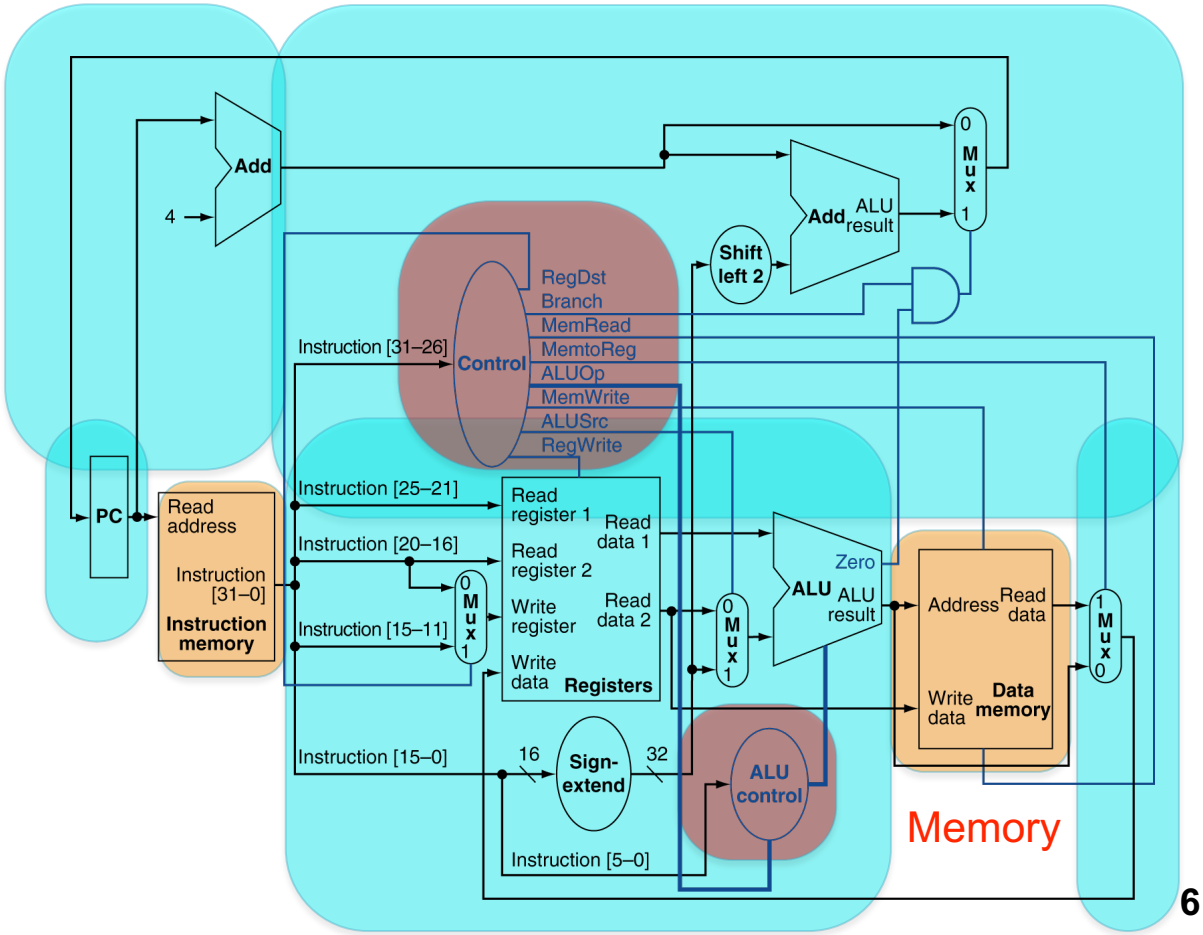




control 회로는 조합회로
→ 진리표만 정하면 된다.

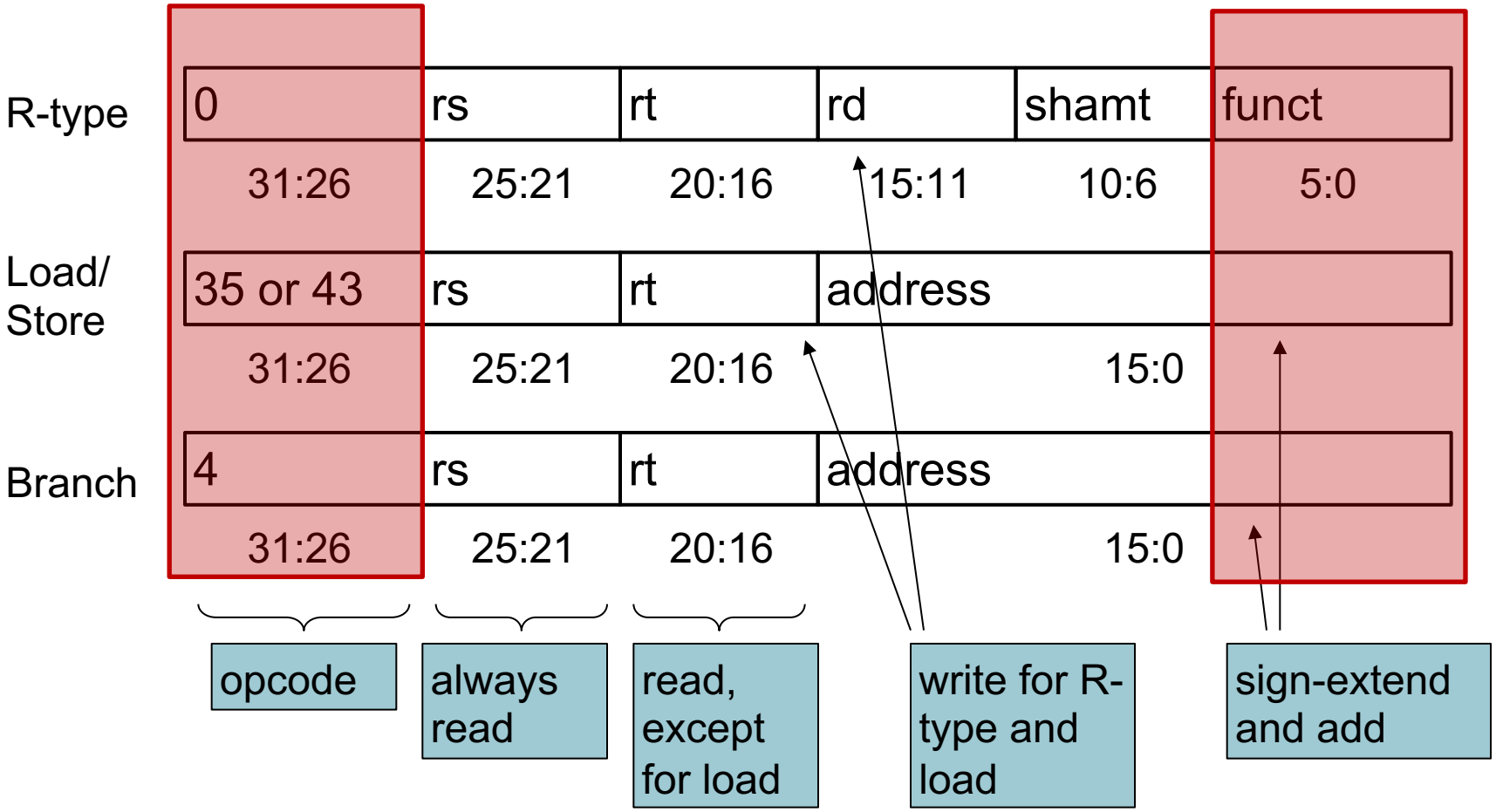
입력?
출력?

Processor

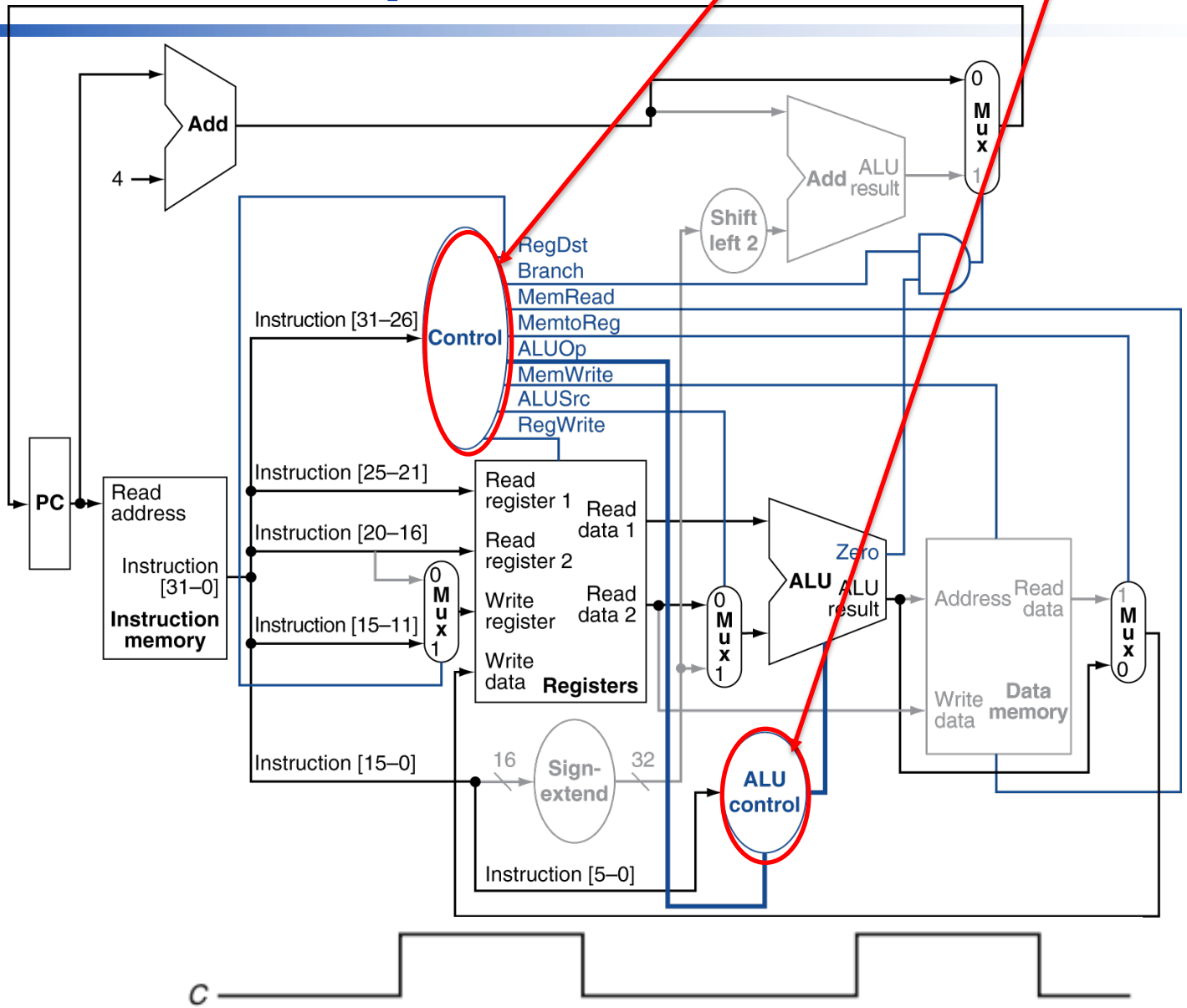


Roles of Instruction Fields

- Control signals derived from instruction



Input and Output of Control/ALU control

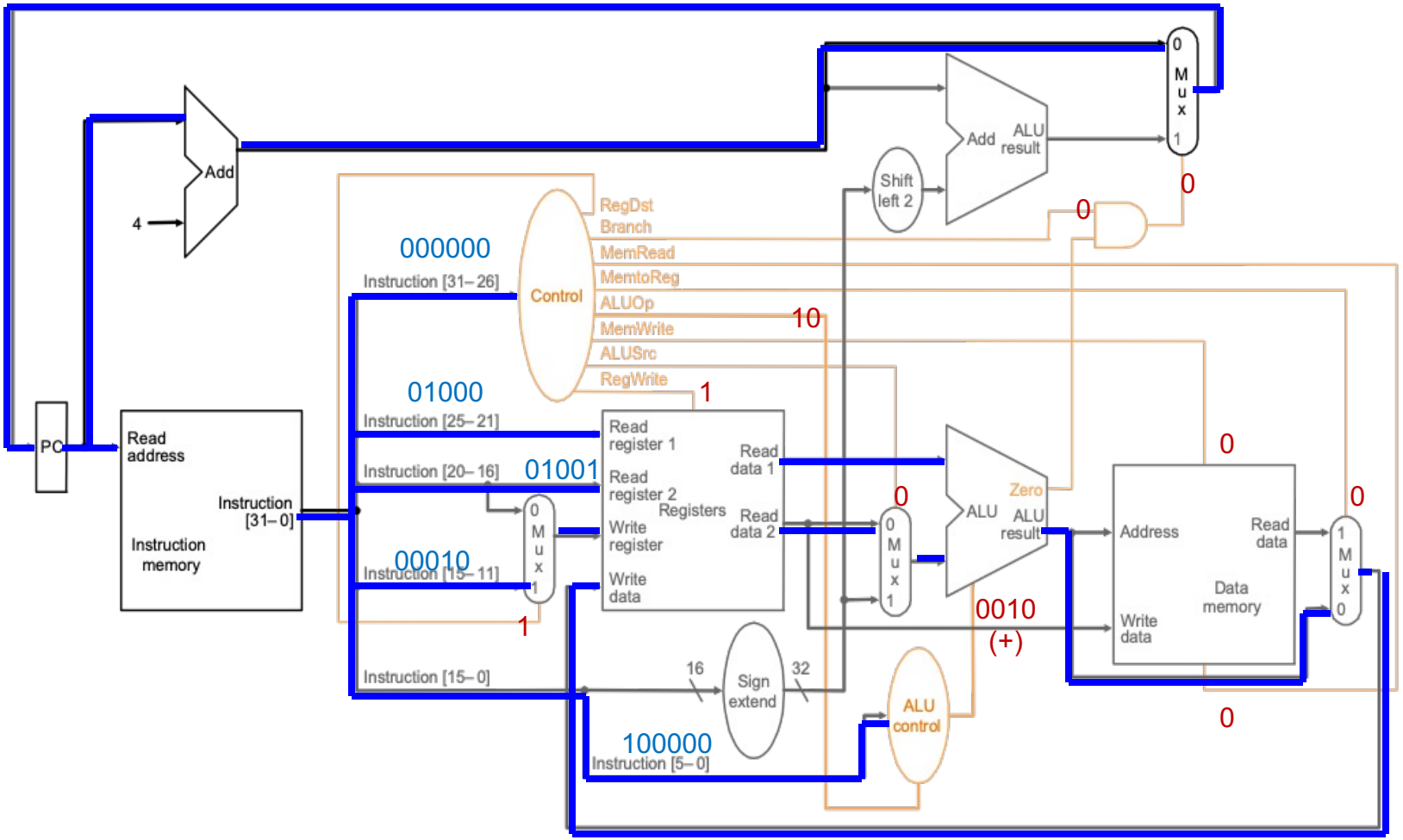


Meaning of Control Signals

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

FIGURE 4.16 The effect of each of the seven control signals. When the 1-bit control to a two-way multiplexor is asserted, the multiplexor selects the input corresponding to 1. Otherwise, if the control is deasserted, the multiplexor selects the 0 input. Remember that the state elements all have the clock as an implicit input and that the clock is used in controlling writes. Gating the clock externally to a state element can create timing problems. (See **Appendix B** for further discussion of this problem.)

Execution of R-type (Arithmetic/Logic) instruction

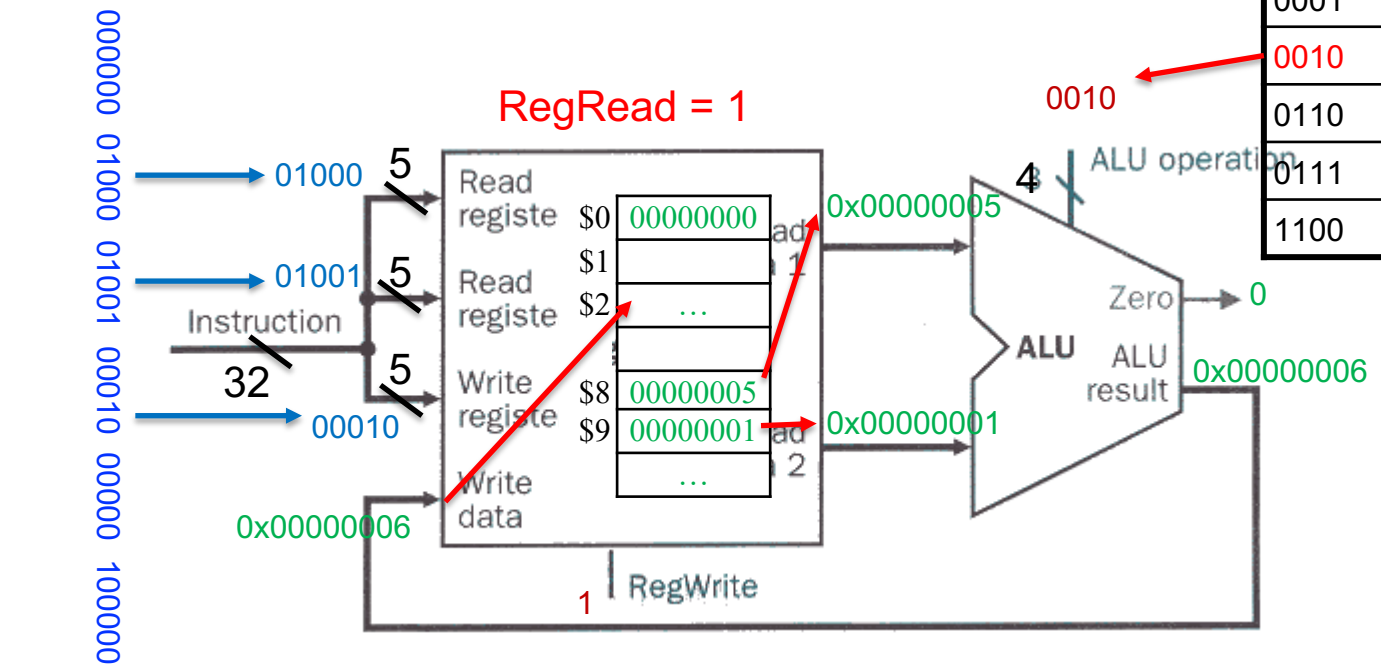


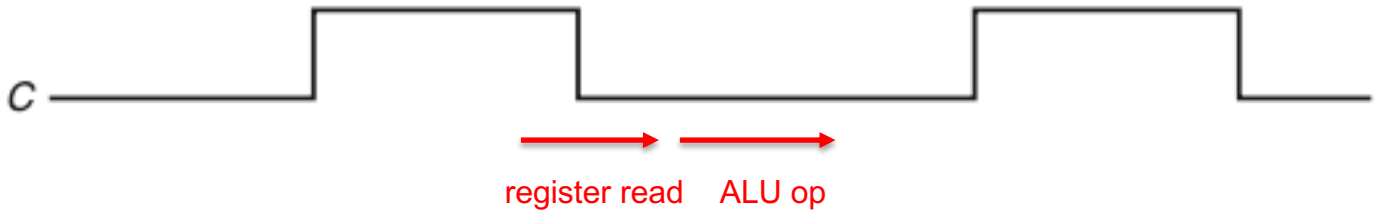
add $2, $8,$9 000000 01000 01001 00010 00000 100000 10

Datapath part 2: Execution of A/L Instructions

add $2, $8,$9 000000 01000 01001 00010 00000 100000

ALU operation	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	slt
1100	NOR





ALU Control

- ALU used for
  - R-type: F depends on funct field
  - Load/Store: F = add
  - Branch: F = subtract

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

ALU control 의 입력과 출력

ALU control 회로의 input

ALU control 회로의 output

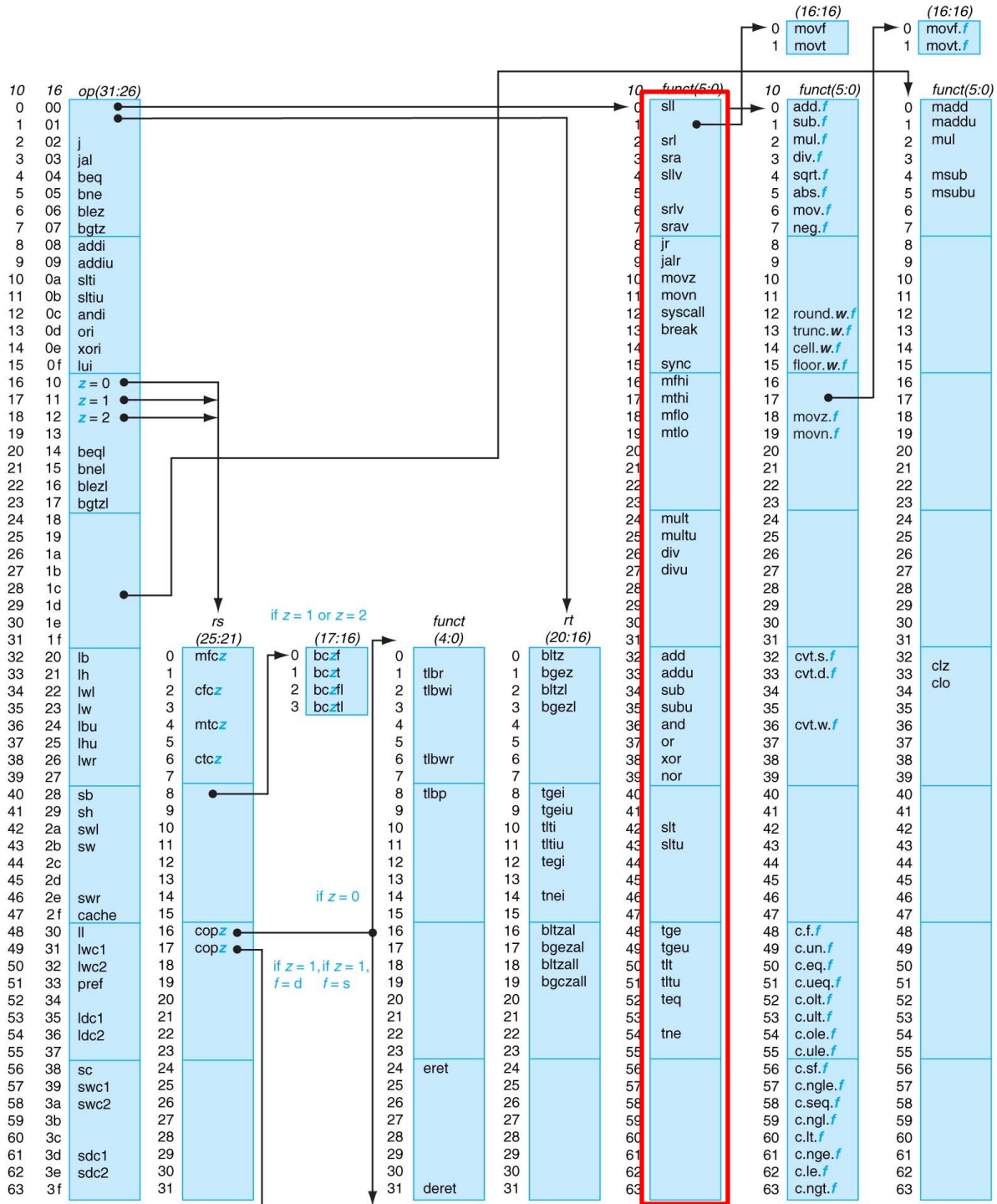
opcode	ALUOp	Operation	funct	ALU function	ALU control
	00		XXXXXX	add	0010
	00		XXXXXX	add	0010
	01		XXXXXX	subtract	0110
	10		100000	add	0010
			100010	subtract	0110
			100100	AND	0000
			100101	OR	0001
			101010	set-on-less-than	0111

ALU control 의 입력과 출력

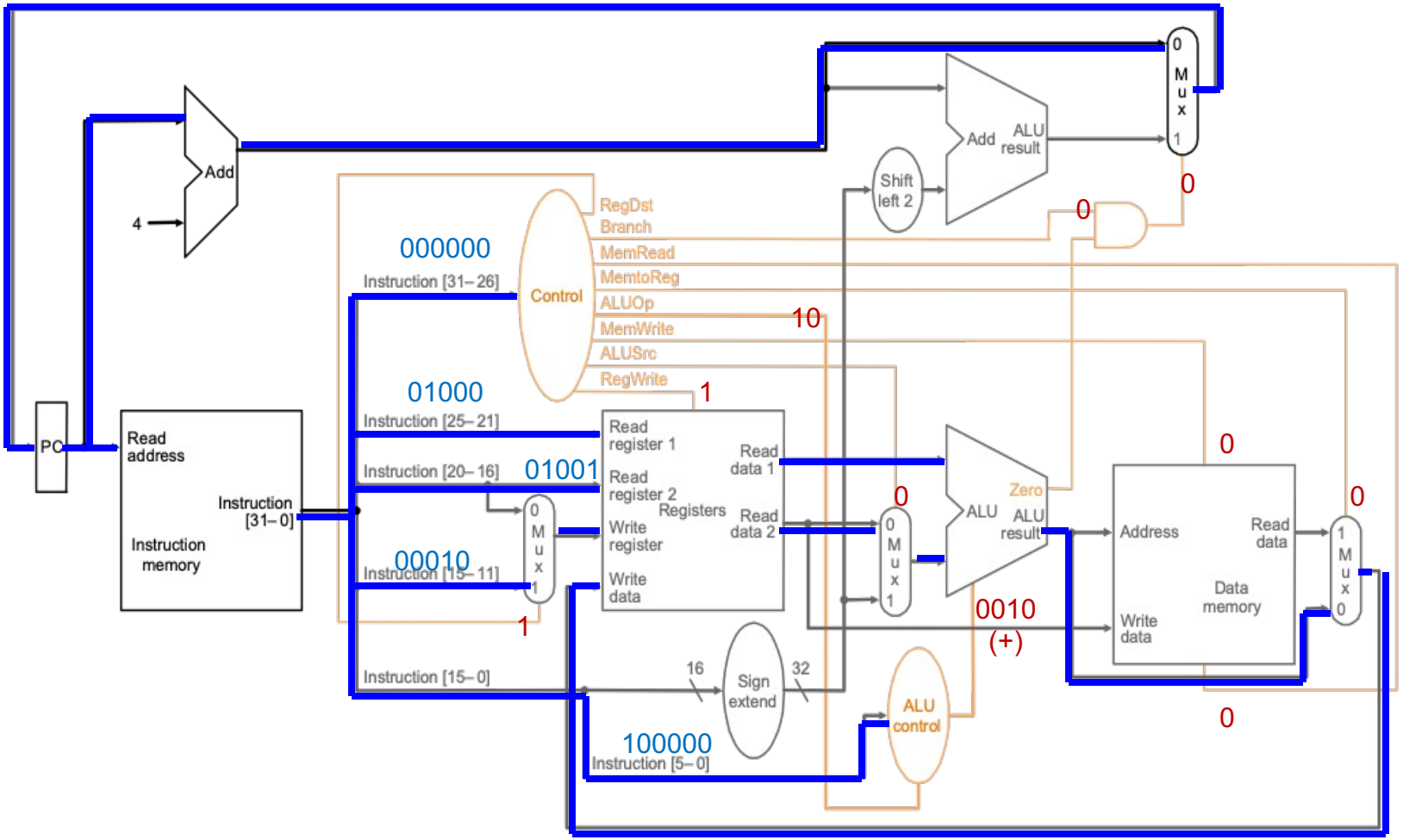
ALU control 회로의 input

ALU control 회로의 output

opcode	ALUOp	Operation	funct INSTR[5:0]	ALU function	ALU control
	00		XXXXXX	add	0010
	00		XXXXXX	add	0010
	01		XXXXXX	subtract	0110
R-type (0x00)	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111


op(31:26)
rs (25:21)
rt (20:16)
funct (4:0)
funct(5:0)
(16:16)
if z = 1 or z = 2
if z = 0
if z = 1, if z = 1, f = d f = s
j
jal
beq
bne
blez
bgtz
addi
addiu
slti
sltiu
andi
ori
xori
lui
z = 0
z = 1
z = 2
beql
bnel
blezl
bgtzl
lb
lh
lwl
lw
lbu
lhu
lwr
sb
sh
swl
sw
swr
cache
ll
lwc1
lwc2
pref
ldc1
ldc2
sc
swc1
swc2
sdc1
sdc2
mfcz
cfcz
mtcz
ctcz
copz
copz
bczf
bczt
bczfl
bczfl
bczt
bczfl
bczfl
bczfl
tlbr
tlbwi
tlbwr
tlbp
eret
deret
bltz
bgez
bltzl
bgezl
tgei
tgeiu
tlti
tltiu
tegi
tnei
bltzal
bgezal
bltzall
bgczall
sll
srl
sra
sllv
srlv
srav
jr
jalr
movz
movn
syscall
break
sync
mfhi
mthi
mflo
mtlo
mult
multu
div
divu
add
addu
sub
subu
and
or
xor
nor
slt
sltu
tge
tgeu
tlt
tltu
teq
tne
movf
movt
add.f
sub.f
mul.f
div.f
sqrt.f
abs.f
mov.f
neg.f
round.w.f
trunc.w.f
cell.w.f
floor.w.f
movz.f
movn.f
cvt.s.f
cvt.d.f
cvt.w.f
c.f.f
c.un.f
c.eq.f
c.ueq.f
c.olt.f
c.ult.f
c.ole.f
c.ule.f
c.sf.f
c.ngle.f
c.seq.f
c.ngl.f
c.lt.f
c.nge.f
c.le.f
c.ngt.f
movf.f
movt.f
madd
maddu
mul
msub
msubu
clz
clo

Execution of R-type (Arithmetic/Logic) instruction



add $2, $8,$9 000000 01000 01001 00010 00000 100000

ALU control 의 입력과 출력

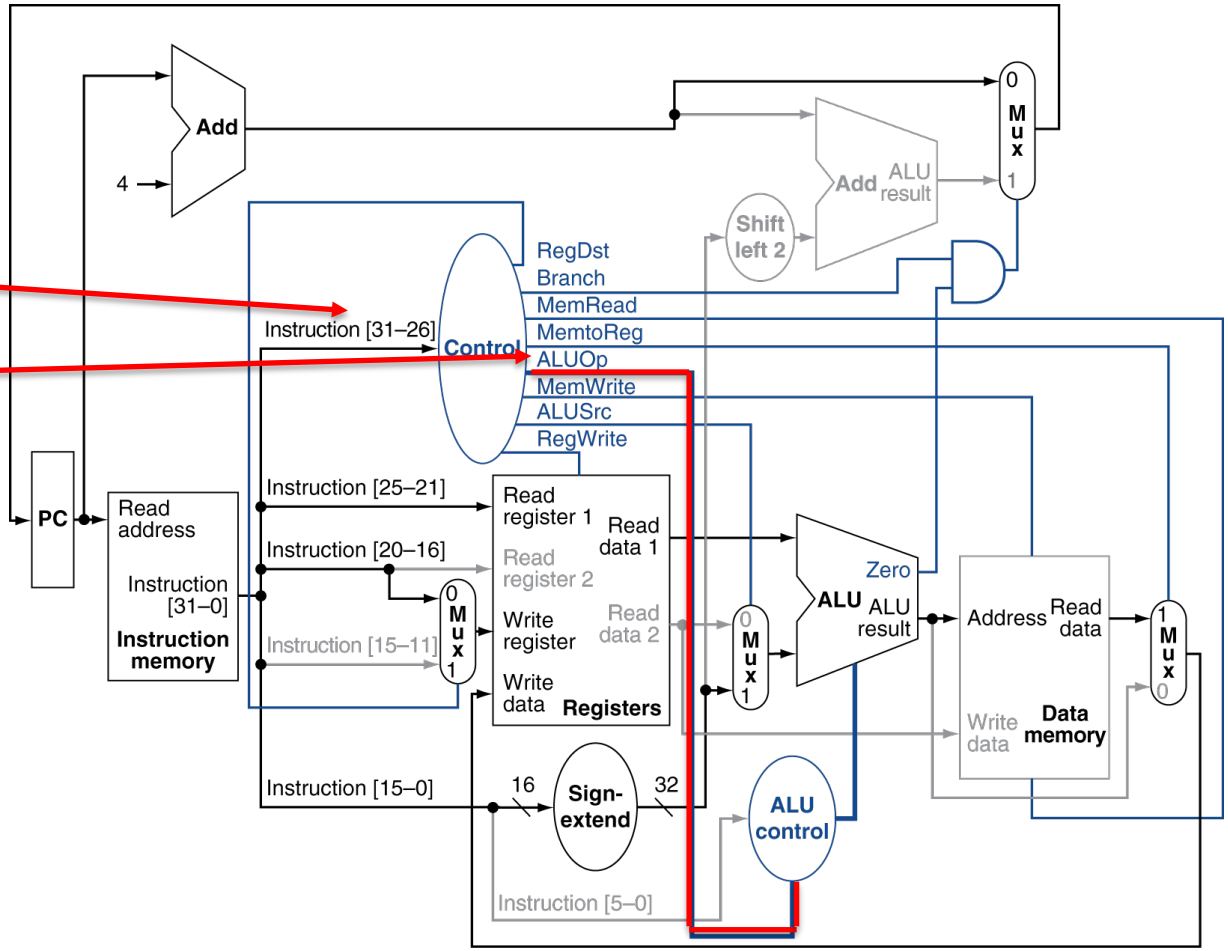
I format instructions

ALU control 회로의 input

ALU control 회로의 output

opcode INSTR[31:26]	ALUOp →2bit code로 변환	Operation	funct INSTR[5:0]	ALU function	ALU control
lw(0x23)	00	load word	XXXXXX	add	0010
sw(0x2b)	00	store word	XXXXXX	add	0010
beq(0x04)	01	branch equal	XXXXXX	subtract	0110
R-type (0x00)	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

	opcode INSTR[31:26]	ALUOp →2bit code로 변환
lw	100011	00
sw	101011	00
beq	000100	01
	000000	10



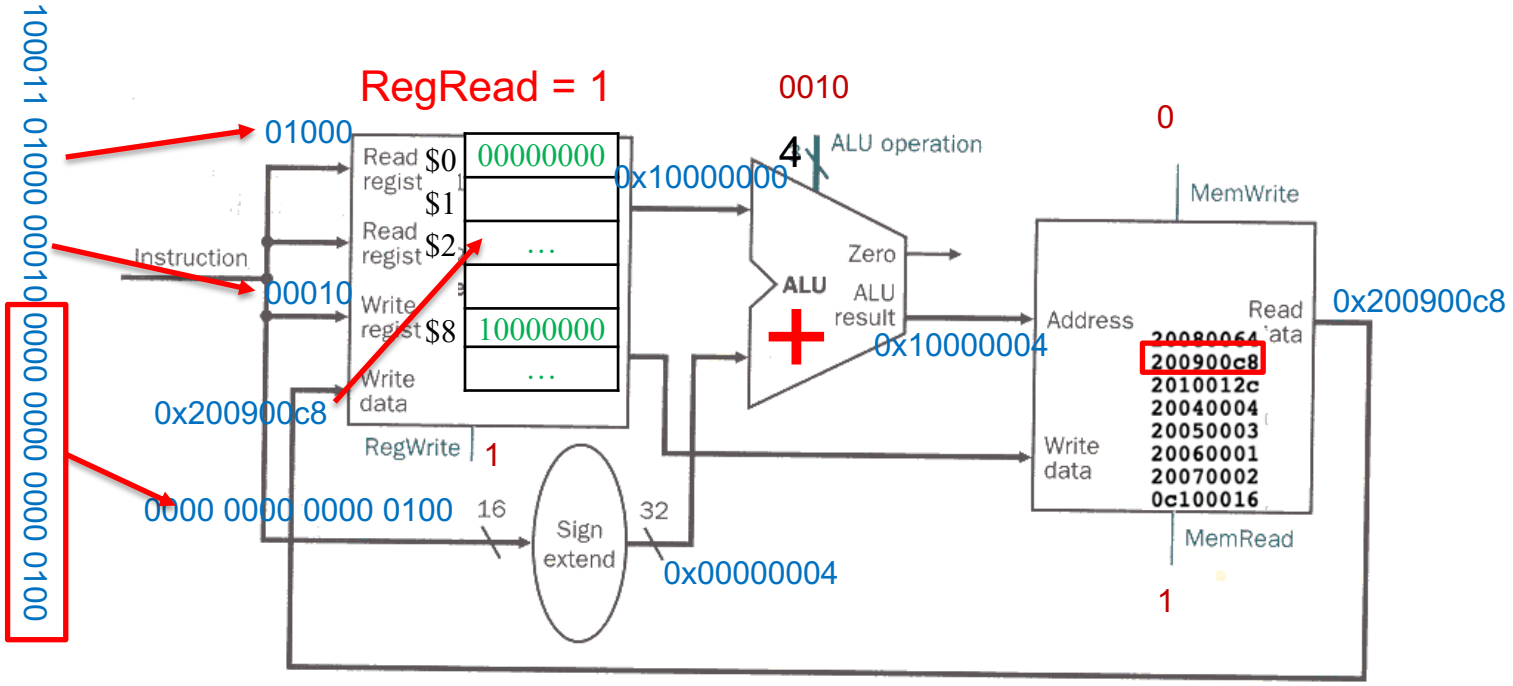
ALU operation 을 위한 진리표

don’t care term (X) 가 많을수록 회로가 단순해짐

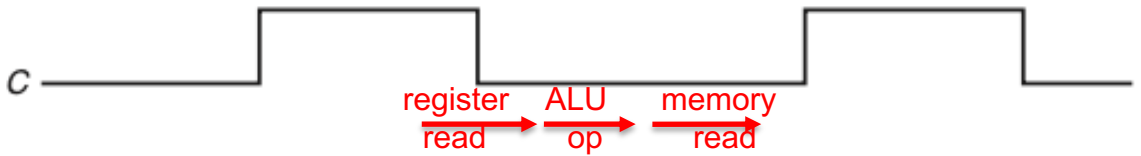
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

FIGURE 4.13 The truth table for the 4 ALU control bits (called Operation). The inputs are the ALUOp and function code field. Only the entries for which the ALU control is asserted are shown. Some don’t-care entries have been added. For example, the ALUOp does not use the encoding 11, so the truth table can contain entries 1X and X1, rather than 10 and 01. Note that when the function field is used, the first 2 bits (F5 and F4) of these instructions are always 10, so they are don’t-care terms and are replaced with XX in the truth table.

Datapath part 3: Execution of Data Transfer Instructions (lw)

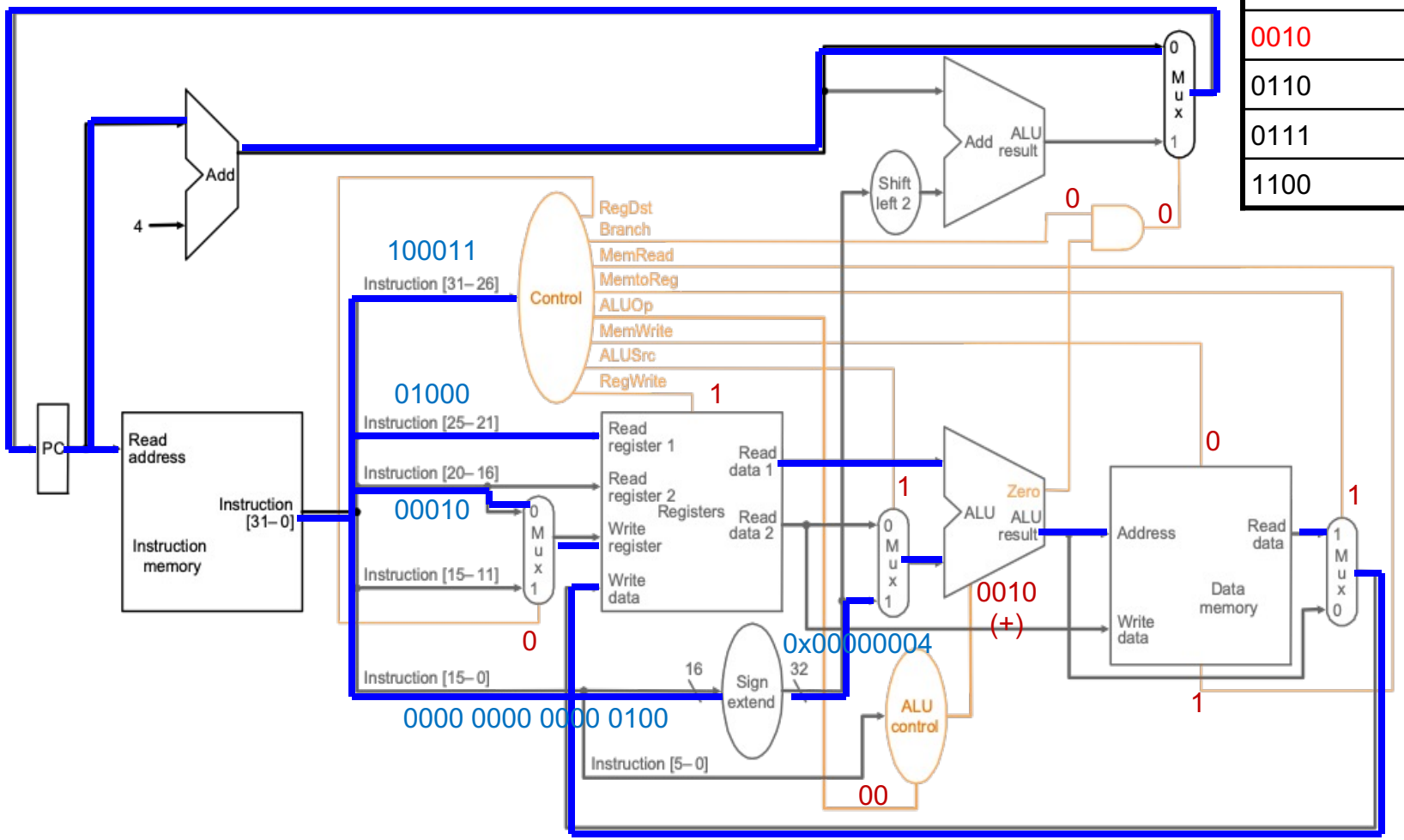


lw $2, 4($8) 100011 01000 00010 0000 0000 0000 0100



Execution of lw instruction

ALU operation	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	slt
1100	NOR



lw $2, 4($8) 100011 01000 00010 0000 0000 0000 0100

ALU operation 을 위한 진리표

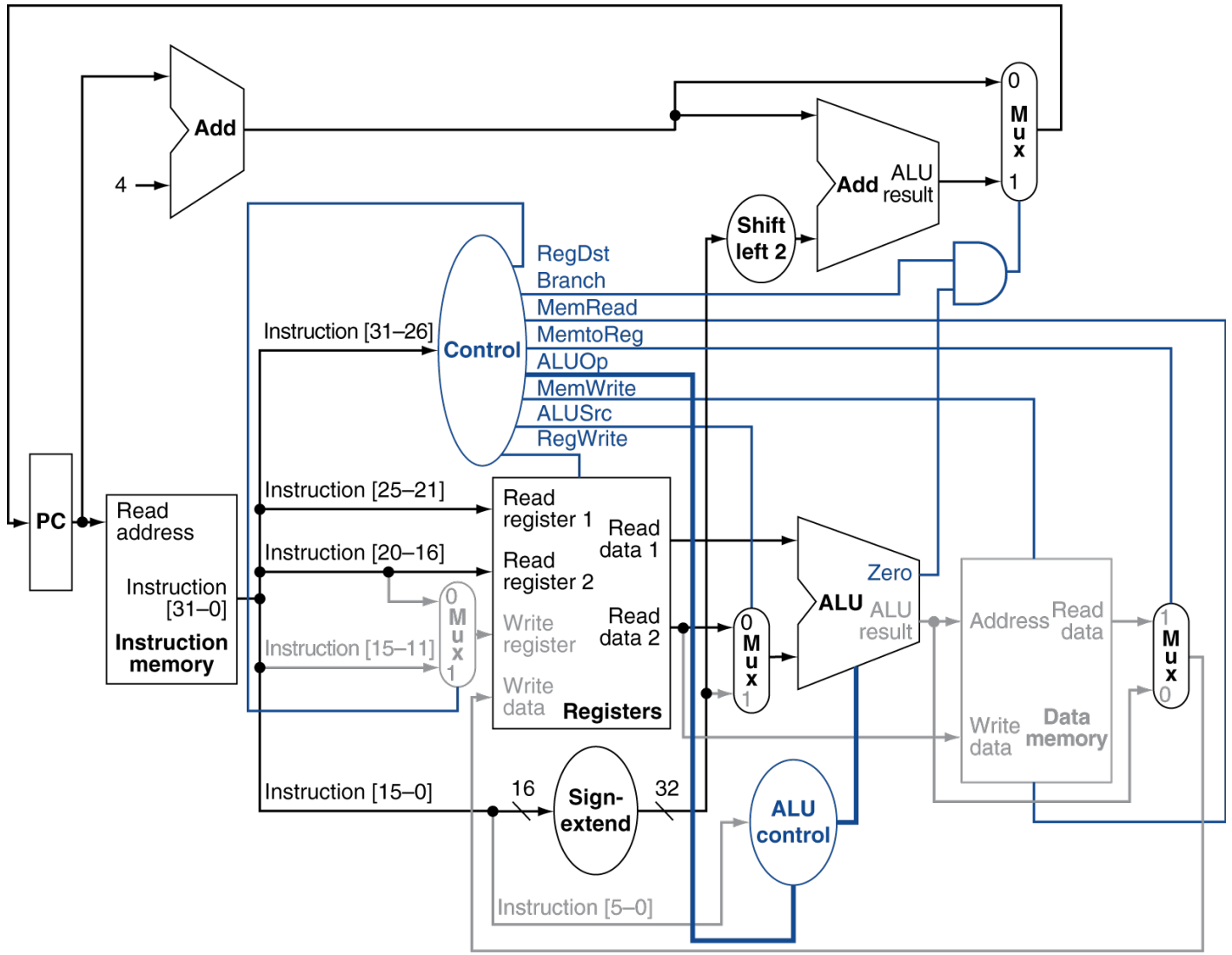
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

FIGURE 4.13 The truth table for the 4 ALU control bits (called Operation). The inputs are the ALUOp and function code field. Only the entries for which the ALU control is asserted are shown. Some don't-care entries have been added. For example, the ALUOp does not use the encoding 11, so the truth table can contain entries 1X and X1, rather than 10 and 01. Note that when the function field is used, the first 2 bits (F5 and F4) of these instructions are always 10, so they are don't-care terms and are replaced with XX in the truth table.

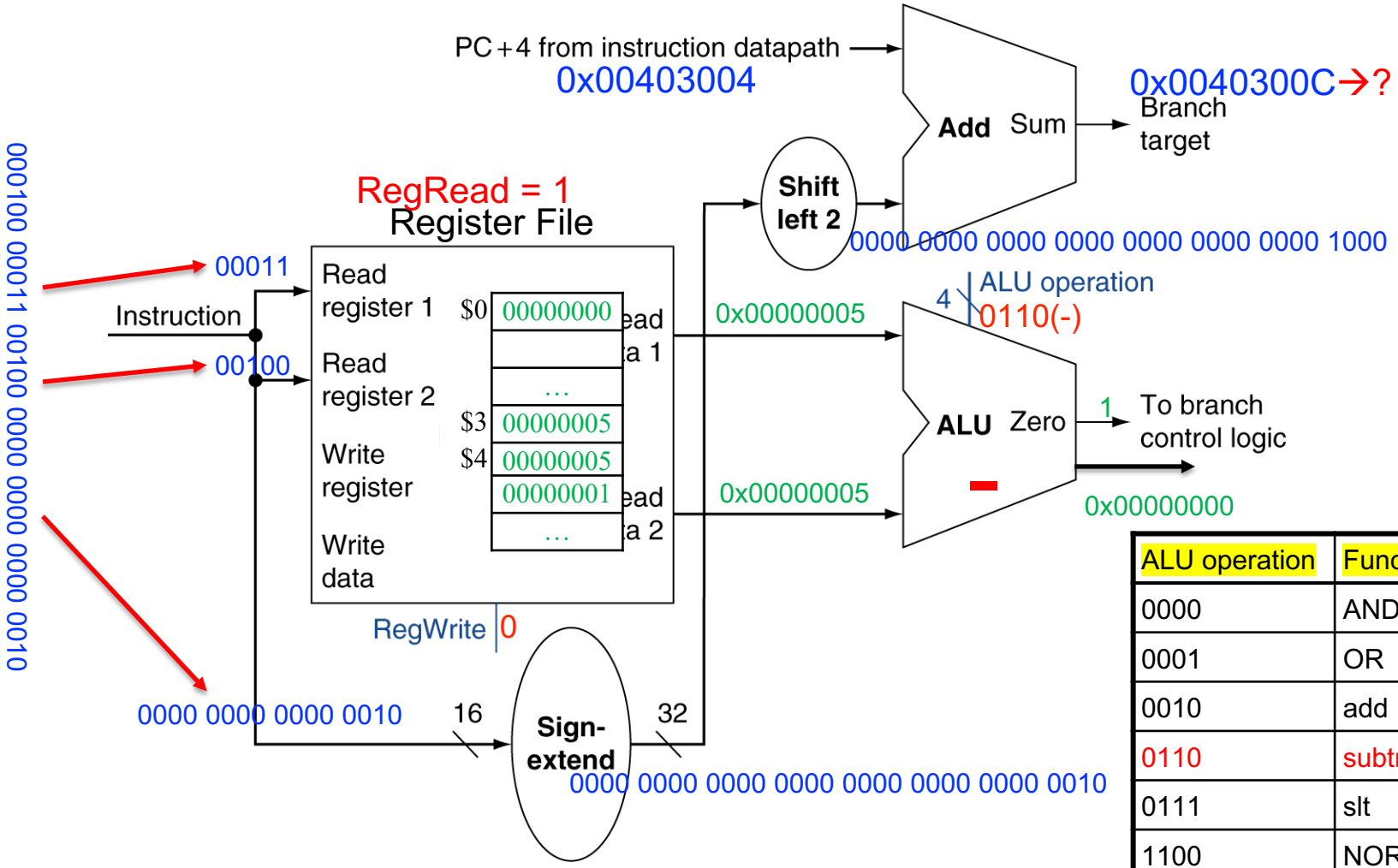
ALU control

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw(0x23)	00	load word	XXXXXX	add	0010
sw(0x2b)	00	store word	XXXXXX	add	0010
beq(0x04)	01	branch equal	XXXXXX	subtract	0110
R-type (0x00)	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

Branch-on-Equal Instruction



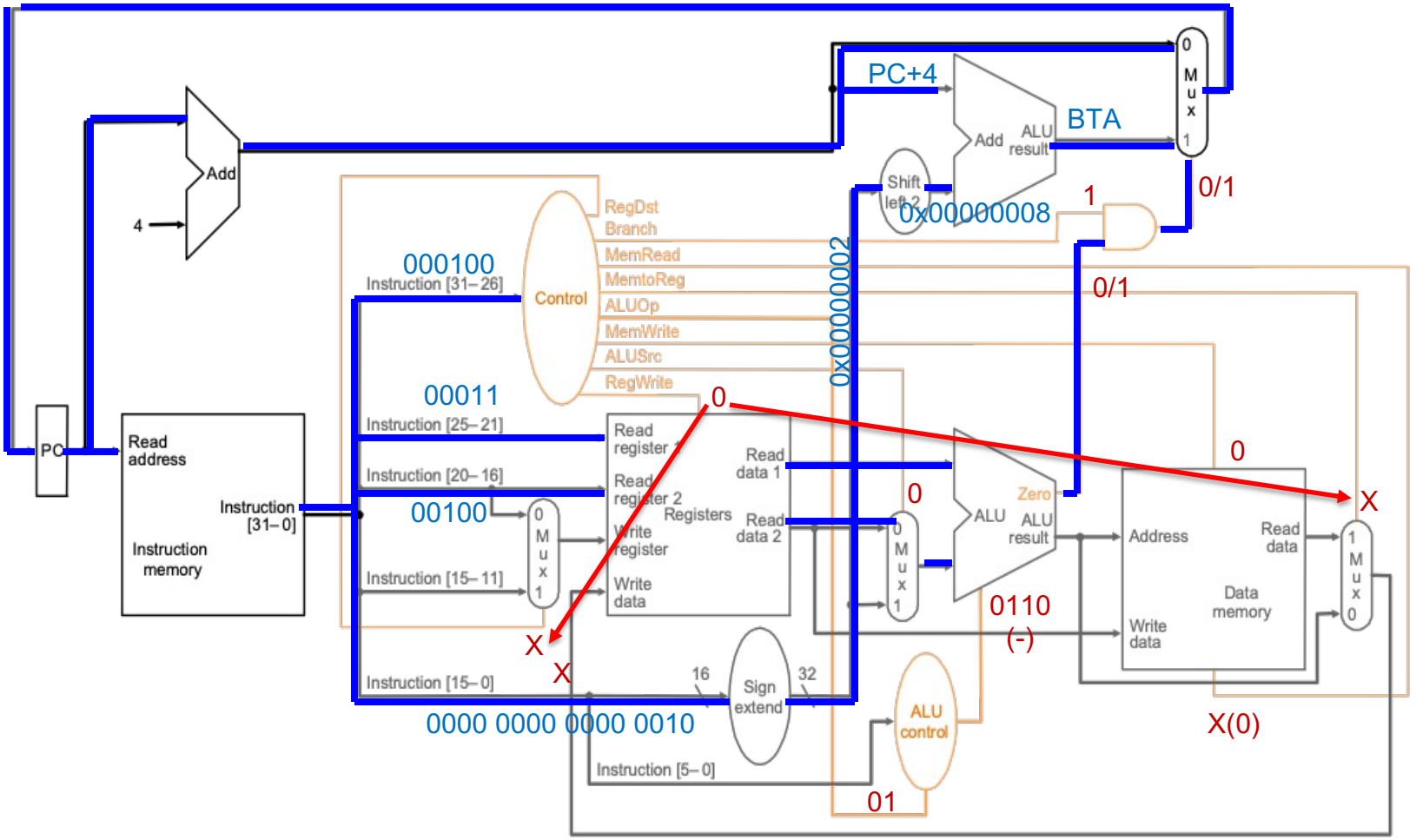
part 4: Branch Instructions



ALU operation	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	slt
1100	NOR

beq $3, $4, L 000100 00011 00100 0000 0000 0000 0010

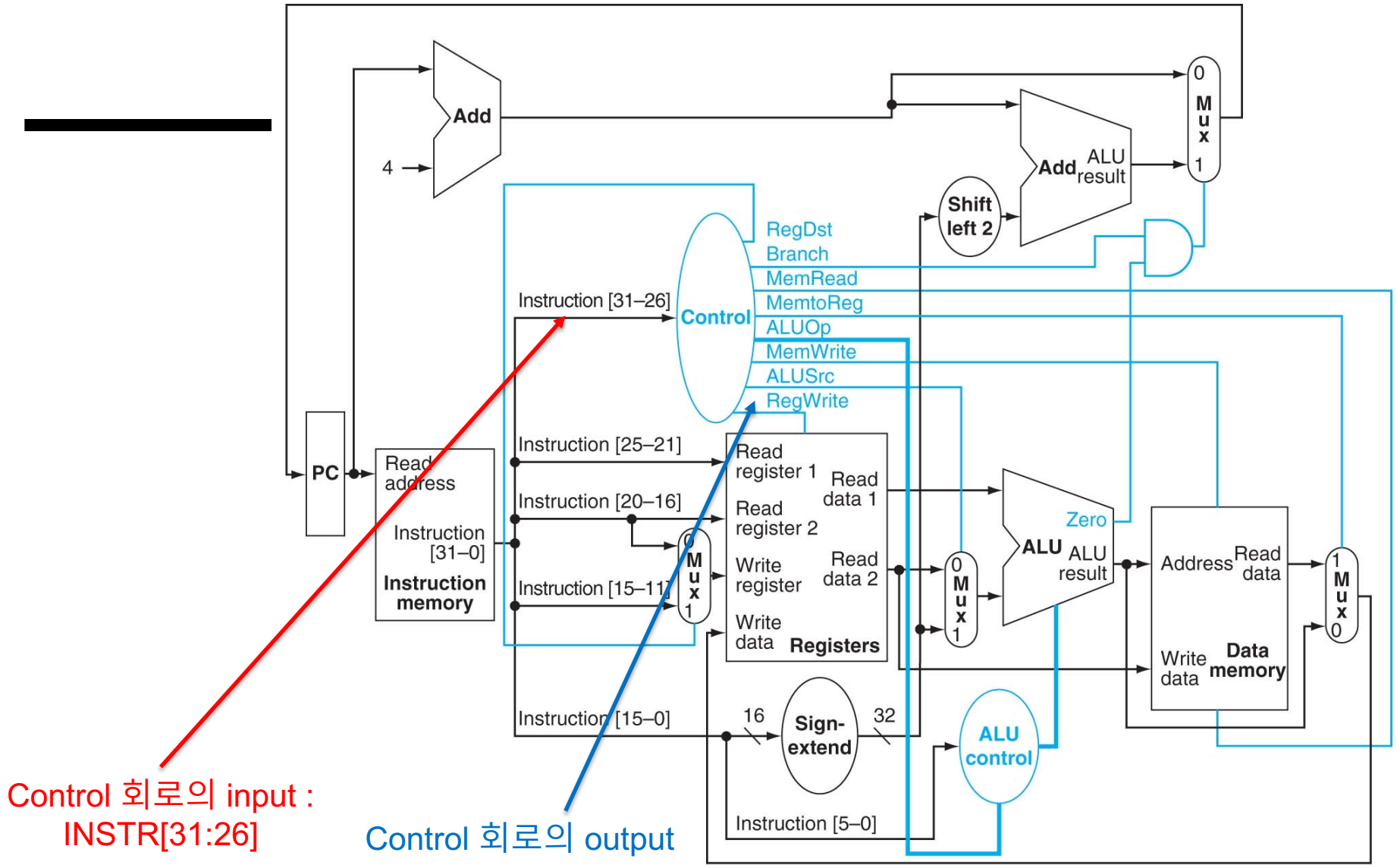
Execution of branch instruction



beq $3, $4, L 000100 00011 00100 0000 0000 0000 0010

ALU control

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw(0x23)	00	load word	XXXXXX	add	0010
sw(0x2b)	00	store word	XXXXXX	add	0010
beq(0x04)	01	branch equal	XXXXXX	subtract	0110
R-type (0x00)	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111



Control 회로의 input : INSTR[31:26]

Control 회로의 output

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Truth table for the control

Input

Name	Opcode in decimal	Opcode in binary					
		Op5	Op4	Op3	Op2	Op1	Op0
R-format	0_{ten}	0	0	0	0	0	0
lw	35_{ten}	1	0	0	0	1	1
sw	43_{ten}	1	0	1	0	1	1
beq	4_{ten}	0	0	0	1	0	0

Output

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

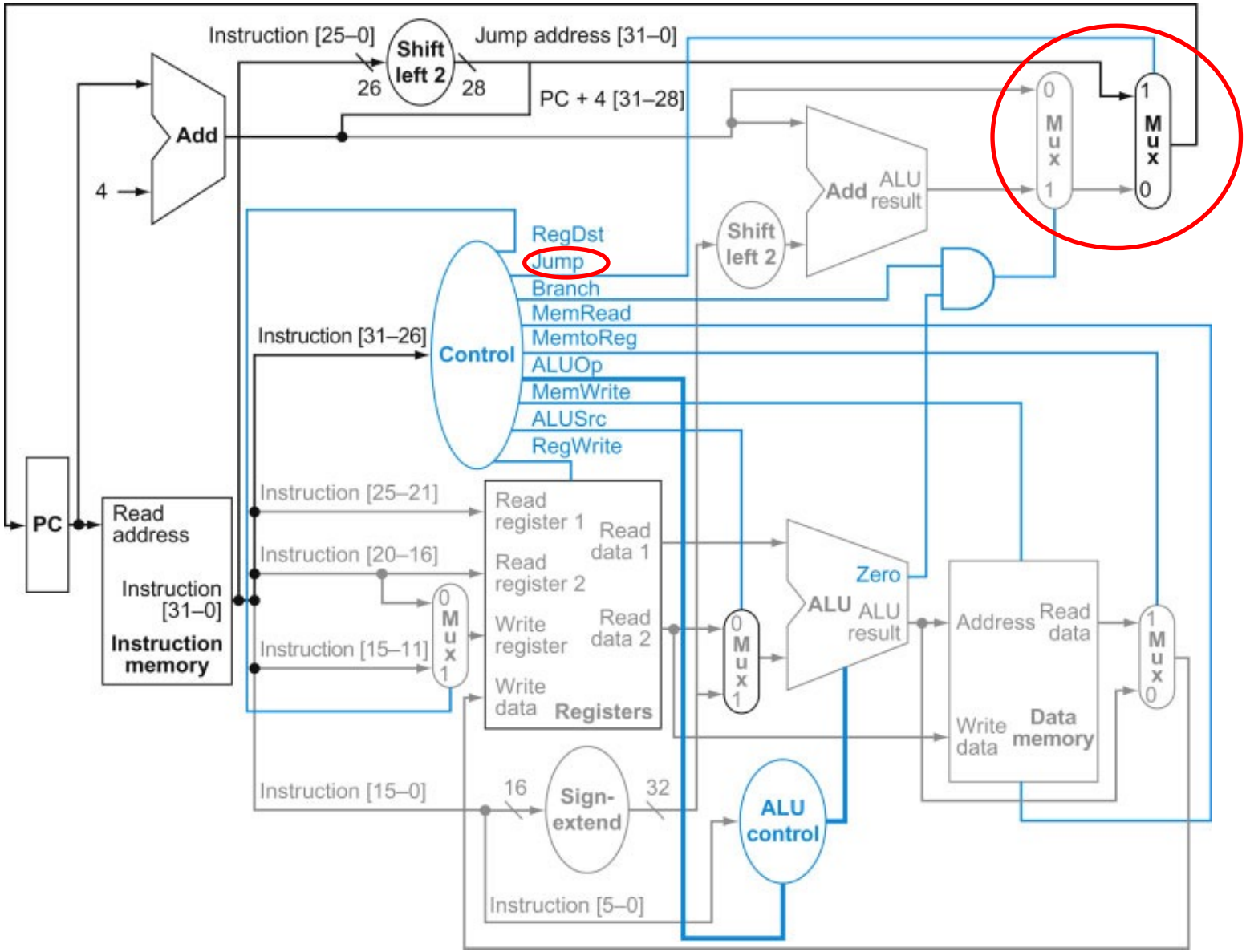
ALU operation 을 위한 진리표

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

FIGURE 4.13 The truth table for the 4 ALU control bits (called Operation). The inputs are the ALUOp and function code field. Only the entries for which the ALU control is asserted are shown. Some don't-care entries have been added. For example, the ALUOp does not use the encoding 11, so the truth table can contain entries 1X and X1, rather than 10 and 01. Note that when the function field is used, the first 2 bits (F5 and F4) of these instructions are always 10, so they are don't-care terms and are replaced with XX in the truth table.

J instruction

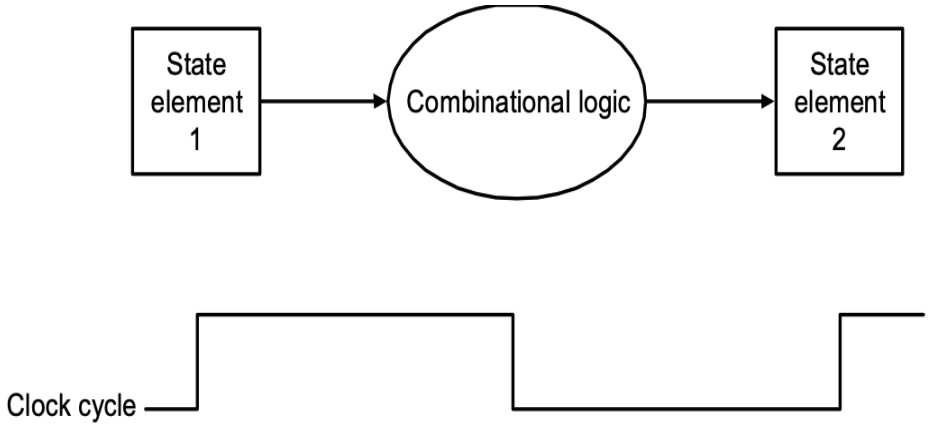
000010	address
31:26	25:0

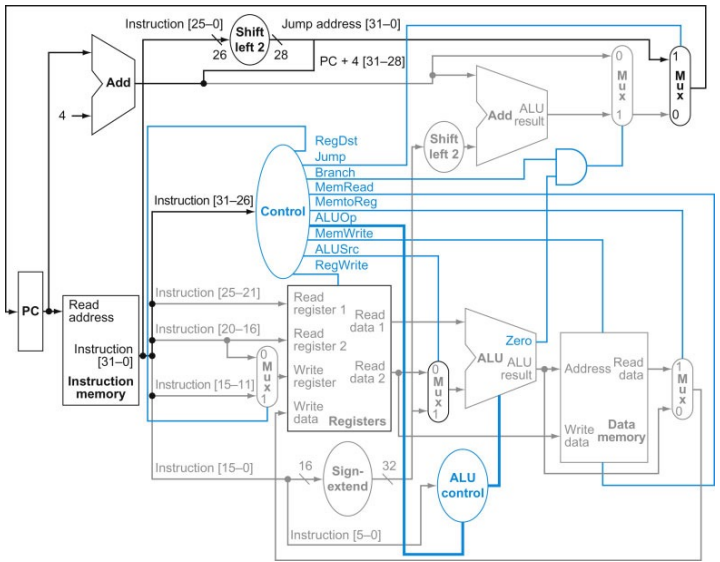


Our Simple Control Structure

- All of the control logic is combinational
- We wait for everything to settle down, and the right thing to be done
  - ALU might not produce right answer right away
  - we use write signals along with clock to determine when to write

  (registers and memory are written at the end of the cycle)
- Cycle time determined by length of the longest path





Performance Issues

- Longest delay determines clock period
  - Critical path: load instruction
  - Instruction memory → register file → ALU → data memory → register file
- Not feasible to vary period for different instructions
- We will improve the performance by pipelining