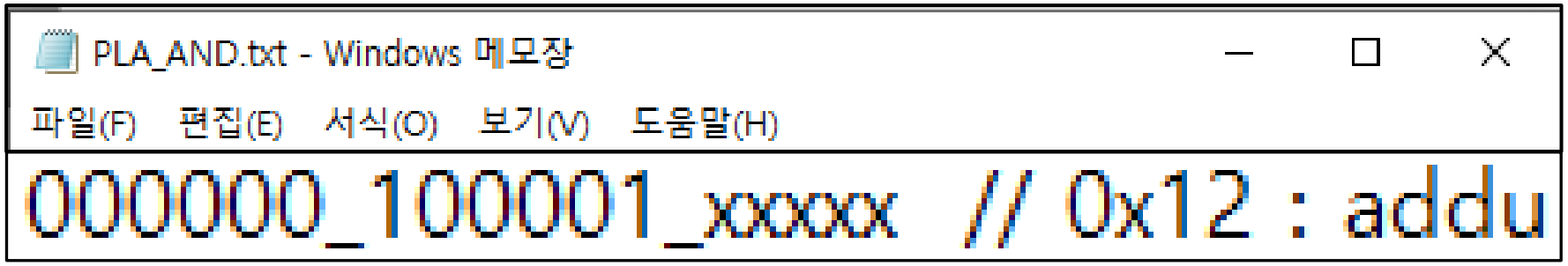
Computer Architecture Lab

05. Week #6

Ex) ADDU ($d = $s + $t, R-type)

Instruction	Opcode/Function	Syntax	Operation
addu	100001	f $d, $s, $t	$d = $s + $t



Opcode_Function_Regimm

Ex) ADDU ($d = $s + $t, R-type)

01

RegDst	Selection for Destination Register
00	$rt
01	$rd
10	$31

00

RegDatSel	To Select Data Source for Register Write
00	Write ALU/MEM Result to Register file
01	Write LO to Register file
10	Write HI to Register file
11	Write PC to Register file

1

RegWrite	Write Enable to Register File
0	Do not write to Register file
1	Write to Register file

Ex) ADDU ($d = $s + $t, R-type)

x

EXTmode	Immediate Data Extension Mode
0	Zero Extension
1	Sign Extension

00

ALUsrcB	ALU Input B Source Selection
00	Register file port B
01	Immediate value
10	zero

00

ALUctrl[1:0]	Extra ALU Control Signal
ALUctrl[0]=0	Shift = Shift Amount
ALUctrl[0]=1	Shift = $rs
ALUctrl[1]=0	Normal ALU input (a,b)
ALUctrl[1]=1	Exchanged ALU input (b,a)

Ex) ADDU ($d = $s + $t, R-type)

ALUop	ALU Operation Code		
00000	Bitwise AND	01001	a × b
00001	Bitwise OR	01010	Unsigned a × b
00010	Bitwise NOR	01011	a / b
00011	Bitwise XOR	01100	Unsigned a / b
00100	a + b	01101	b << a
00101	Unsigned a + b	01110	b >> a
00110	a – b	01111	b >>> a
00111	Unsigned a – b	10000	Set Less Than
01000	Reserved	10001	Unsigned SLT
10010	HI = a	10011	LO = a

Ex) ADDU ($d = $s + $t, R-type)

xxx

DatWidth	Data Width for Memory Access
000	32-bit Word
010	16-bit Halfword
011	8-bit Byte
110	16-bit Halfword /w Sign Ext
111	8-bit Byte /w Sign Ext

0

MemWrite	Memory Write Enable
0	Do not write Data to Memory
1	Write Data to Memory

0

MemtoReg	Write Memory Data to Register File
0	Write ALU data to Register file
1	Write Memory data to Register file

Ex) ADDU ($d = $s + $t, R-type)

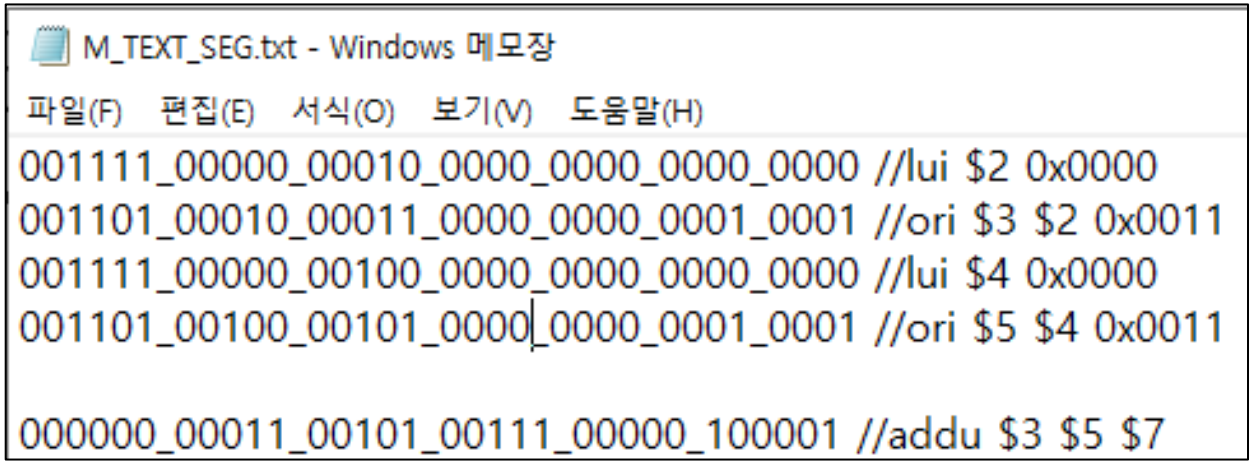
000

Branch	Branch Option & Address Source
000	Use PC+4
011	Unconditional Branch to PC+imm16
100	Branch if zero
101	Branch if not zero
110	Branch if not positive
111	Branch if positive

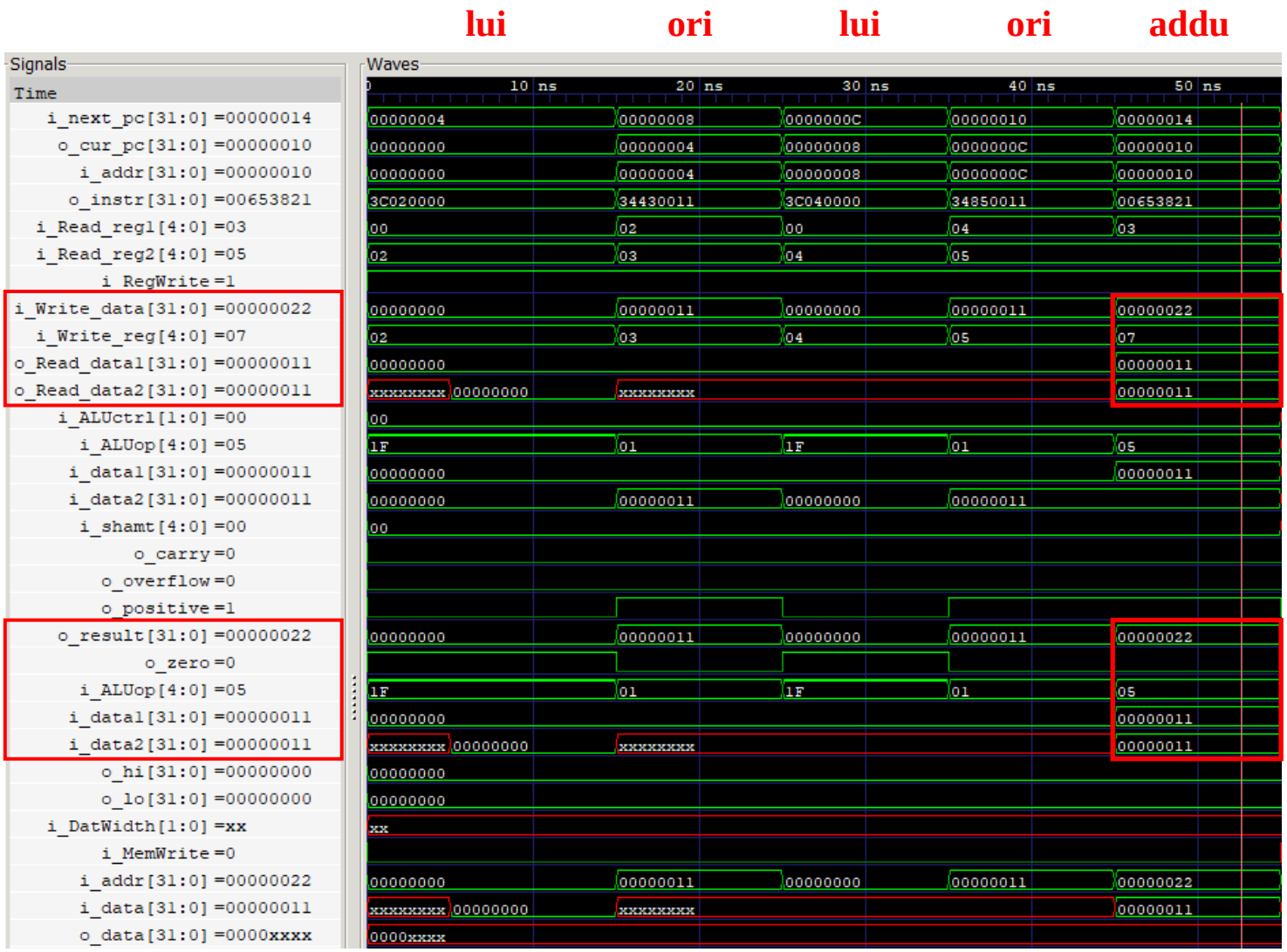
00

Jump	Jump Address Source
00	Do not use Jump address
01	Use Pseudo Jump address format
10	Use $rs

Ex) ADDU ($d = $s + $t, R-type)

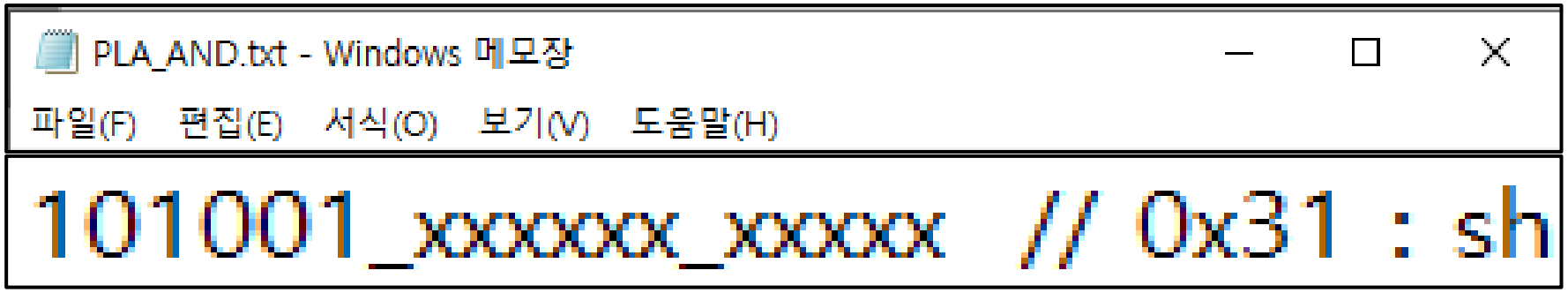


Ex) ADDU ($d = $s + $t, R-type)



Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

Instruction	Opcode/Function	Syntax	Operation
sh	101001	o $t, i ($s)	MEM [$s + i]:2 = LH ($t)



Opcode_Function_Regimm

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

xx

RegDst	Selection for Destination Register
00	$rt
01	$rd
10	$31

xx

RegDatSel	To Select Data Source for Register Write
00	Write ALU/MEM Result to Register file
01	Write LO to Register file
10	Write HI to Register file
11	Write PC to Register file

0

RegWrite	Write Enable to Register File
0	Do not write to Register file
1	Write to Register file

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

1

EXTmode	Immediate Data Extension Mode
0	Zero Extension
1	Sign Extension

01

ALUsrcB	ALU Input B Source Selection
00	Register file port B
01	Immediate value
10	zero

00

ALUctrl[1:0]	Extra ALU Control Signal
ALUctrl[0]=0	Shift = Shift Amount
ALUctrl[0]=1	Shift = $rs
ALUctrl[1]=0	Normal ALU input (a,b)
ALUctrl[1]=1	Exchanged ALU input (b,a)

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

ALUop	ALU Operation Code		
00000	Bitwise AND	01001	a × b
00001	Bitwise OR	01010	Unsigned a × b
00010	Bitwise NOR	01011	a / b
00011	Bitwise XOR	01100	Unsigned a / b
00100	a + b	01101	b << a
00101	Unsigned a + b	01110	b >> a
00110	a – b	01111	b >>> a
00111	Unsigned a – b	10000	Set Less Than
01000	Reserved	10001	Unsigned SLT
10010	HI = a	10011	LO = a

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

010

DatWidth	Data Width for Memory Access
000	32-bit Word
010	16-bit Halfword
011	8-bit Byte
110	16-bit Halfword /w Sign Ext
111	8-bit Byte /w Sign Ext

1

MemWrite	Memory Write Enable
0	Do not write Data to Memory
1	Write Data to Memory

x

MemtoReg	Write Memory Data to Register File
0	Write ALU data to Register file
1	Write Memory data to Register file

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

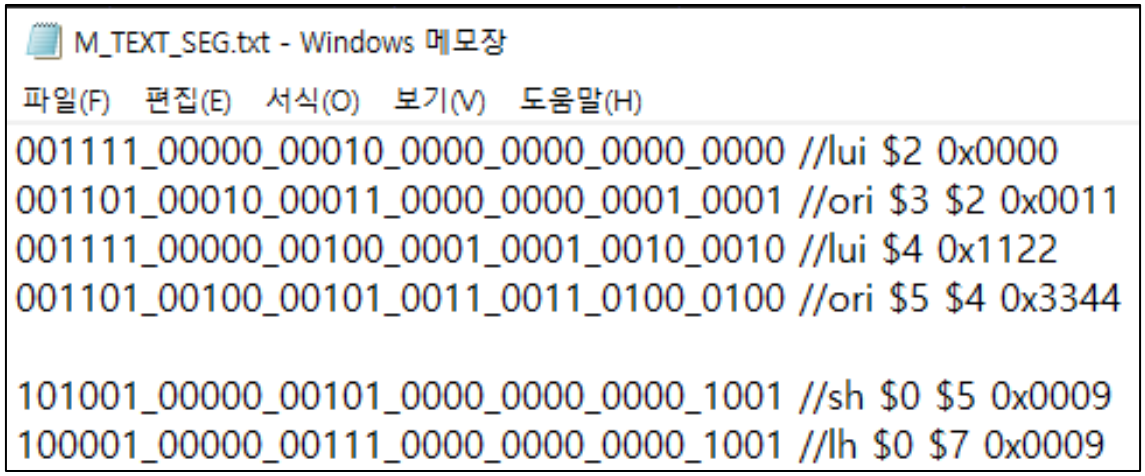
000

Branch	Branch Option & Address Source
000	Use PC+4
011	Unconditional Branch to PC+imm16
100	Branch if zero
101	Branch if not zero
110	Branch if not positive
111	Branch if positive

00

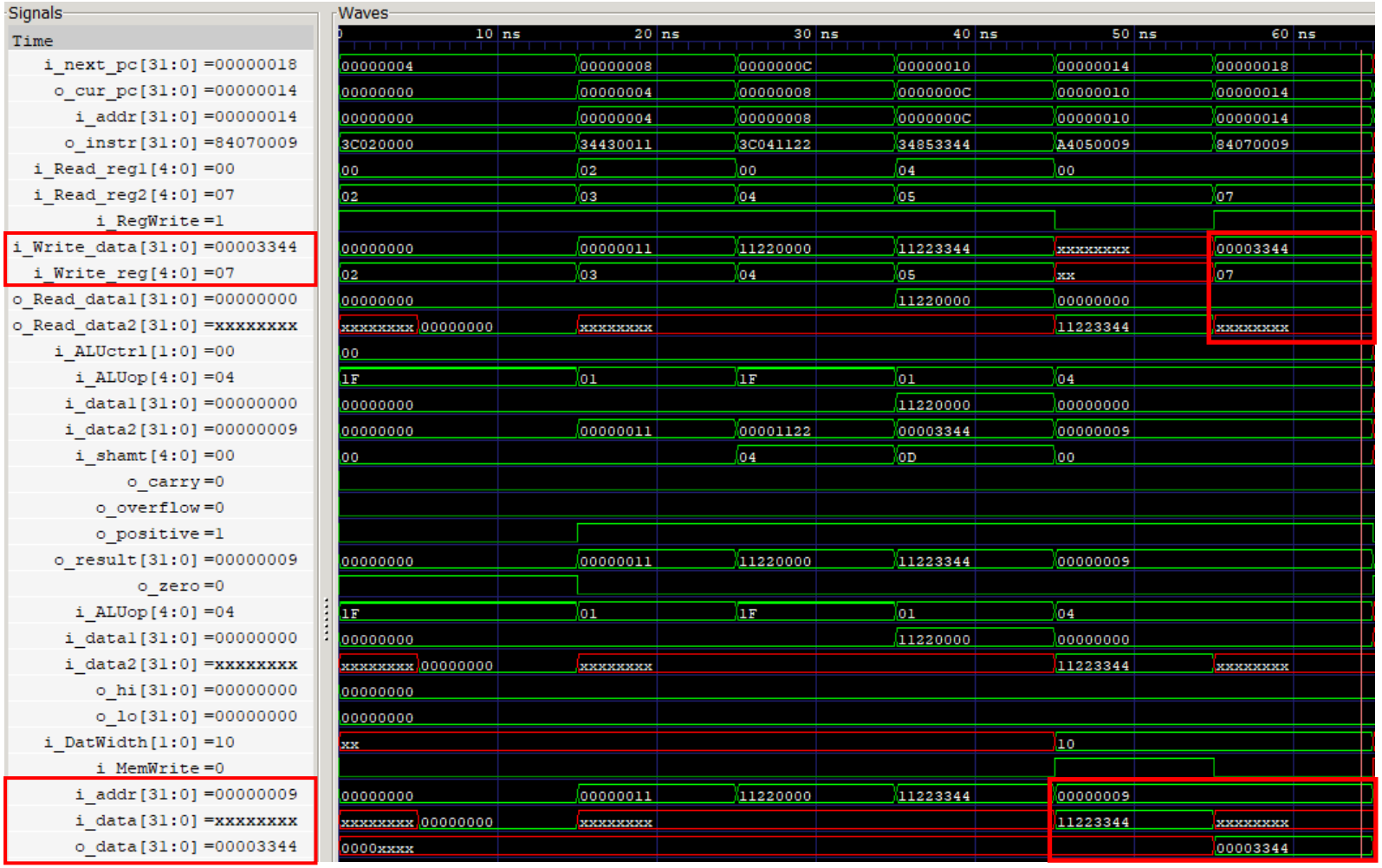
Jump	Jump Address Source
00	Do not use Jump address
01	Use Pseudo Jump address format
10	Use $rs

Ex) SH (MEM[$s+i]:2 = LH($t), I-type)



Ex) SH (MEM[$s+i]:2 = LH($t), I-type)

lui **ori** **lui** **ori** **sh** **lh**



END