Arithmetic, Logical Operation, Test-and-Set						
001 000	addi	addi rt rs imm	rt = rs + sxt(imm)			
001 001	addiu	addiu <i>rt rs imm</i>	rt = rs + sxt(imm)			
001 010	slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$			
001 011	sltiu	sltiu <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$			
001 100	andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$			
001 101	ori	ori <i>rt rs imm</i>	$rt = rs \lor zxt(imm)$			
001 110	xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$			
001 111	lui	lui <i>rt imm</i>	$rt = imm0^{16}$			

Arithmetic, Logical Operation							
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt				
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt				
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt				
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt				
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$				
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$				
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$				
000000 100 111	nor	nor <i>rd rs rt</i>	$rd = rs \lor rt$				
Test-and-Set Operation							
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				

$$lop(c) = c.gpr(rs(c)).$$

all ALU operations

Arithmetic, Logical Operation, Test-and-Set					
001 000	addi	addi <i>r</i>	rsi	пm	rt = rs + sxt(imm)
001 001	addiu	addiu i	t rs	mm	rt = rs + sxt(imm)
001 010	slti	slti <i>rt</i>	rs i	пm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$
001 011	sltiu	sltiu <i>r</i>	rsi	mm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$
001 100	andi	andi <i>r</i>	rsi	mm	$rt = rs \wedge zxt(imm)$
001 101	ori	ori <i>rt</i>	rs ii	$\imath m$	$rt = rs \lor zxt(imm)$
001 110	xori	xori r	rs i	nm	$rt = rs \oplus zxt(imm)$
001 111	lui	lui <i>i</i>	t im	m	$rt = imm0^{16}$

Arithmetic, Logical Ope	ration				
000000 100 000	add	add <i>r</i>	l rs	t	rd = rs + rt
000000 100 001	addu	addu <i>i</i>	d rs	rt	rd = rs + rt
000000 100 010	sub	sub <i>r</i>	l rs	t	rd = rs - rt
000000 100 011	subu	subu <i>i</i>	d rs	rt	rd = rs - rt
000000 100 100	and	and <i>r</i>	l rs	t	$rd = rs \wedge rt$
000000 100 101	or	or <i>ra</i>	rs 1		$rd = rs \lor rt$
000000 100 110	xor	xor r	rs	t	$rd = rs \oplus rt$
000000 100 111	nor	nor <i>r</i>	rs	t	$rd = \overline{rs \vee rt}$
Test-and-Set Operation					
000000 101 010	slt	slt <i>ra</i>	rs 1	ţ	$rd = (rs < rt ? 1_{32} : 0_{32})$
000000 101 011	sltu	sltu <i>r</i>	l rs	't	$rd = (rs < rt ? 1_{32} : 0_{32})$

$$lop(c) = c.gpr(rs(c)).$$

right operand: gpr(rt) or extended immediate

$$rop(c) \equiv \begin{cases} c.gpr(rt(c)), & rtype(c), \\ xtimm(c), & \text{otherwise.} \end{cases}$$

all ALU operations

Arithmetic, Logical Operation, Test-and-Set					
001 000	addi	addi <i>rt</i>	rs imm	rt = rs + sxt(imm)	
001 001	addiu	addiu <i>rt</i>	rs imm	rt = rs + sxt(imm)	
001 010	slti	slti <i>rt 1</i>	s imm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$	
001 011	sltiu	sltiu <i>rt</i>	rs imm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$	
001 100	andi	andi <i>rt</i>	rs imm	$rt = rs \wedge zxt(imm)$	
001 101	ori	ori <i>rt 1</i>	s imm	$rt = rs \lor zxt(imm)$	
001 110	xori	xori <i>rt</i>	's imm	$rt = rs \oplus zxt(imm)$	
001 111	lui	lui <i>rt</i>	imm	$rt = imm0^{16}$	

Arithmetic, Logical Operation						
000000 100 000	add	add <i>rd 1</i>	s rt	rd = rs + rt		
000000 100 001	addu	addu <i>rd</i>	s rt	rd = rs + rt		
000000 100 010	sub	sub <i>rd r</i>	s rt	rd = rs - rt		
000000 100 011	subu	subu <i>rd</i> .	s rt	rd = rs - rt		
000000 100 100	and	and <i>rd r</i>	s rt	$rd = rs \wedge rt$		
000000 100 101	or	or <i>rd rs</i>	rt	$rd = rs \lor rt$		
000000 100 110	xor	xor <i>rd r</i>	rt	$rd = rs \oplus rt$		
000000 100 111	nor	nor <i>rd r</i>	rt	$rd = \overline{rs \vee rt}$		
Test-and-Set Operation						
000000 101 010	slt	slt <i>rd r</i> :	rt	$rd = (rs < rt ? 1_{32} : 0_{32})$		
000000 101 011	sltu	sltu <i>rd 1</i>	s rt	$rd = (rs < rt ? 1_{32} : 0_{32})$		

$$lop(c) = c.gpr(rs(c)).$$

right operand: gpr(rt) or extended immediate

$$rop(c) \equiv \begin{cases} c.gpr(rt(c)), & rtype(c), \\ xtimm(c), & \text{otherwise.} \end{cases}$$

how to extend (u=?)

$$xtimm(c) \equiv \begin{cases} sxtimm(c), & opc(c)[2] = 0, \\ zxtimm(c), & opc(c)[2] = 1 \end{cases}$$

all ALU operations

Arithmetic, Logical Operation, Test-and-Set							
001 0 0	addi	addi <i>rt rs imm</i>	rt = rs +	sxt(imm)			
001 0 1	addiu	addiu <i>rt rs imm</i>	rt = rs +	sxt(imm)			
001 0 0	slti	slti <i>rt rs imm</i>	rt = (rs <	sxt(imm)	$? 1_{32} : 0_{32})$		
001 0 1	sltiu	sltiu <i>rt rs imm</i>	rt = (rs <	sxt(imm)	$? 1_{32} : 0_{32})$		
001 1 00	andi	andi <i>rt rs imm</i>	$rt = rs \wedge$	zxt(imm)			
001 1 11	ori	ori <i>rt rs imm</i>		zxt(imm)			
001 1 0	xori	xori rt rs imm		zxt(imm)			
001 111	lui	lui <i>rt imm</i>	rt = imm	10^{16}			

Arithmetic, Logical Operation						
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt			
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt			
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt			
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt			
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$			
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$			
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$			
000000 100 111	nor	nor <i>rd rs rt</i>	$rd = rs \vee rt$			
Test-and-Set Operation						
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			

$$lop(c) = c.gpr(rs(c)).$$

right operand: gpr(rt) or extended immediate

$$rop(c) \equiv \begin{cases} c.gpr(rt(c)), & rtype(c), \\ xtimm(c), & \text{otherwise.} \end{cases}$$

how to extend (u=?)

$$xtimm(c) \equiv \begin{cases} sxtimm(c), & opc(c)[2] = 0, \\ zxtimm(c), & opc(c)[2] = 1 \end{cases}$$

MIPS instruction manual acknowledges this as 'misnomer'

all ALU operations

Arithme	Arithmetic, Logical Operation, Test-and-Set							
001 0 0		addi	addi rt rs imm	rt = rs + sxt(imm)				
001 0)1	1	addiu	addiu rt rs imm	rt = rs + sxt(imm)				
001 0 0)	slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$				
001 0 1	1	sltiu	sltiu rt rs imm	$rt = (rs < sxt(imm ? 1_{32} : 0_{32})$				
001 1 0)	andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$				
001 1)1	1	ori	ori rt rs imm	$rt = rs \lor zxt(imm)$				
001 1 0		xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$				
001 1 1	1	lui	lui <i>rt imm</i>	$rt = imm0^{16}$				

Arithmetic, Logical Operation						
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt			
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt			
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt			
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt			
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$			
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$			
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$			
000000 100 111	nor	nor rd rs rt	$rd = \overline{rs \vee rt}$			
Test-and-Set Operation						
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			

$$lop(c) = c.gpr(rs(c)).$$

right operand: gpr(rt) or extended immediate

$$rop(c) \equiv \begin{cases} c.gpr(rt(c)), & rtype(c), \\ xtimm(c), & \text{otherwise.} \end{cases}$$

how to extend (u=?)

$$xtimm(c) \equiv \begin{cases} sxtimm(c), & opc(c)[2] = 0, \\ zxtimm(c), & opc(c)[2] = 1 \end{cases}$$

$$\equiv ifill(c)^{16}imm(c)$$
fill bit

$$ifill(c) \equiv \begin{cases} 0, & opc(c)[2] = 1, \\ imm(c)[15], & \text{otherwise} \end{cases}$$

$$\equiv I(c)[15] \wedge \overline{I(c)[28]},$$

Arithmetic, Logical Operation, Test-and-Set 001 0 0 addi rt rs imm addi rt = rs + sxt(imm)001 0 1 addiu addiu rt rs imm |rt = rs + sxt(imm)|001 0 0 slti slti rt rs imm $|rt = (rs < sxt(imm) ? 1_{32} : 0_{32})|$ 001 0 1 sltiu sltiu *rt rs imm* $|rt = (rs < sxt(imm) ? 1_{32} : 0_{32})|$ 001 1 0 andi rt rs imm andi $rt = rs \wedge zxt(imm)$ 001 1 11 $rt = rs \lor zxt(imm)$ ori rt rs imm ori 001 1 0 xori rt rs imm $|rt = rs \oplus zxt(imm)|$ xori $rt = imm0^{16}$ 001 1 1 lui lui *rt imm*

all ALU operations

Arithmetic, Logical Operation						
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt			
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt			
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt			
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt			
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$			
000000 100 101	or	or rd rs rt	$rd = rs \lor rt$			
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$			
000000 100 111	nor	nor rd rs rt	$rd = rs \lor rt$			
Test-and-Set Operation						
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			

af[3:0]] i	alures[n-1:0]	ovfalu
000	*	$a +_{n} b$	$[a] + [b] \notin T_n$
0001	*	$a+_{n}b$	0
0010	*	an b	$[a] - [b] \notin T_n$
0011	*	an b	0
0 100	*	$a \wedge b$	0
0101	*	$a \lor b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \lor b}$	0
0 111	1	$b[n/2-1:0]0^{n/2}$	0
1010	*	$0^{n-1}([a] < [b] ? 1 : 0)$	0
1 D1 1	*	$0^{n-1}(\langle a\rangle < \langle b\rangle ? 1:0)$	0

Table 7. Specification of ALU operations

function bits[2:0]

$$af(c)[2:0] \equiv \begin{cases} fun(c)[2:0], & rtype(c), \\ opc(c)[2:0], & \text{otherwise} \end{cases}$$

$$\equiv \begin{cases} I(c)[2:0], & rtype(c), \\ I(c)[28:26], & \text{otherwise.} \end{cases}$$

all ALU operations

Ari	Arithmetic, Logical Operation, Test-and-Set								
001	000		addi	addi <i>rt rs imm</i>	rt = rs + sxt(imm)				
001	001		addiu	addiu <i>rt rs imm</i>	rt = rs + sxt(imm)				
001	010		slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$				
001	011		sltiu	sltiu <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$				
001	100		andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$				
001	101		ori	ori <i>rt rs imm</i>	$rt = rs \lor zxt(imm)$				
001	110		xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$				
001	111		lui	lui <i>rt imm</i>	$rt = imm0^{16}$				

Arithmetic, Logical Operation							
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt				
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt				
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt				
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt				
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$				
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$				
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$				
000000 100 111	nor	nor rd rs rt	$rd = \overline{rs \vee rt}$				
Test-and-Se Operation							
000000 101 010	slt	slt rd rs rt	$rd = (rs < rt ? 1_{32} : 0_{32})$				
000000 101 011	sltu	sltu rd rs rt	$rd = (rs < rt ? 1_{32} : 0_{32})$				

af[3:0]	i	alures[n-1:0]	ovfalu
0000	*	$a+_n b$	$[a] + [b] \notin T_n$
0001	*	$a+_n b$	0
0010	*	an b	$[a] - [b] \notin T_n$
0011	*	an b	0
0100	*	$a \wedge b$	0
0101	*	$a \lor b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \lor b}$	0
0111	1	$b[n/2-1:0]0^{n/2}$	0
1 10	*	$0^{n-1}([a] < [b] ? 1 : 0)$	0
1 11	*	$0^{n-1}(\langle a \rangle < \langle b \rangle ? 1 : 0)$	0

Table 7. Specification of ALU operations

function bits[2:0]

$$af(c)[2:0] \equiv \begin{cases} fun(c)[2:0], & rtype(c), \\ opc(c)[2:0], & \text{otherwise} \end{cases}$$

$$\equiv \begin{cases} I(c)[2:0], & rtype(c), \\ I(c)[28:26], & \text{otherwise.} \end{cases}$$

function bit[3]

$$af(c)[3] \equiv \begin{cases} \frac{fun(c)[3]}{opc(c)[2]} & \text{rtype}(c), \\ \frac{opc(c)[2]}{opc(c)[1]}, & \text{otherwise} \end{cases}$$

$$\equiv \begin{cases} \frac{I(c)[3]}{I(c)[28]} \wedge I(c)[27], & \text{otherwise}. \end{cases}$$

all ALU operations

I-type

Arithmetic, Logical Operation, Test-and-Set						
001 000	addi	addi <i>rt rs imm</i>	rt = rs + sxt(imm)			
001_001	addiu	addiu <i>rt rs imm</i>	rt = rs + sxt(imm)			
001 01)	slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$			
001 01 1	sltiu	sltiu <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$			
001 100	andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$			
001 101	ori	ori <i>rt rs imm</i>	$rt = rs \lor zxt(imm)$			
001 110	xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$			
001 111	lui	lui <i>rt imm</i>	$rt = imm0^{16}$			

Arithmetic, Logical Operation							
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt				
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt				
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt				
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt				
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$				
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$				
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$				
000000 100 111	nor	nor rd rs rt	$rd = rs \lor rt$				
Test-and-Set Operation							
000000 10 1 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				

R-type

af[3:0]	i	alures[n-1:0]	ovfalu
0000	*	$a +_n b$	$[a] + [b] \notin T_n$
0001	*	$a+_n b$	0
0010	*	an b	$[a] - [b] \notin T_n$
0011	*	an b	0
0100	*	$a \wedge b$	0
0101	*	$a \lor b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \lor b}$	0
0111	1	$b[n/2 - 1:0]0^{n/2}$	0
1010	*	$0^{n-1}([a] < [b] ? 1 : 0)$	0
1011	*	$0^{n-1}(\langle a \rangle < \langle b \rangle ? 1 : 0)$	0

Table 7. Specification of ALU operations

ALU input i

$$i = itype(c)$$

all ALU operations

Arithmetic, Log	Arithmetic, Logical Operation, Test-and-Set						
001 000	addi	addi <i>rt rs imm</i>	rt = rs + sxt(imm)				
001 001	addiu	addiu rt rs imm	rt = rs + sxt(imm)				
001 010	slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$				
001 011	sltiu	sltiu <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$				
001 100	andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$				
001 101	ori	ori <i>rt rs imm</i>	$rt = rs \lor zxt(imm)$				
001 110	xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$				
001 111	lui	lui <i>rt imm</i>	$rt = imm0^{16}$				

Arithmetic, Logical Operation							
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt				
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt				
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt				
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt				
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$				
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$				
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$				
000000 100 111	nor	nor <i>rd rs rt</i>	$rd = \overline{rs \vee rt}$				
Test-and-Set Operation							
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				

af[3:0]	i	alures[n-1:0]	ovfalu
0000	*	$a +_n b$	$[a] + [b] \notin T_n$
0001	*	$a +_n b$	0
0010	*	an b	$[a] - [b] \notin T_n$
0011	*	an b	0
0100	*	$a \wedge b$	0
0101	*	$a \lor b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \lor b}$	0
0111	1	$b[n/2 - 1:0]0^{n/2}$	0
1010	*	$0^{n-1}([a] < [b] ? 1 : 0)$	0
1011	*	$0^{n-1}(\langle a \rangle < \langle b \rangle ? 1 : 0)$	0

 Table 7. Specification of ALU operations

ALU input i

i = itype(c)

ALU result

 $ares(c) \equiv alures(lop(c), rop(c), af(c), itype(c)).$

all ALU operations

Arithmetic, Logical Operation, Test-and-Set						
001 000	addi	addi rt rs imm	rt =	rs + sxt(imm)		
001 001	addiu	addiu <i>rt rs imm</i>	rt =	rs + sxt(imm)		
001 010	slti	slti <i>rt rs imm</i>	rt =	$(rs < sxt(imm) ? 1_{32} : 0_{32})$		
001 011	sltiu	sltiu <i>rt rs imm</i>	rt =	$(rs < sxt(imm) ? 1_{32} : 0_{32})$		
001 100	andi	andi <i>rt rs imm</i>	rt =	rs ∧ zxt(imm)		
001 101	ori	ori <i>rt rs imm</i>	rt =	rs ∨ zxt(imm)		
001 110	xori	xori rt rs imm	1	$rs \oplus zxt(imm)$		
001 111	lui	lui <i>rt imm</i>	rt =	imm0 ¹⁶		

Arithmetic, Logical Operation							
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt				
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt				
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt				
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt				
000000 100 100	and	and <i>rd rs rt</i>	$rd = rs \wedge rt$				
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$				
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$				
000000 100 111	nor	nor rd rs rt	$rd = rs \vee rt$				
Test-and-Set Operation							
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$				

af[3:0]	i	alures[n-1:0]	ovfalu
0000	*	$a +_n b$	$[a] + [b] \notin T_n$
0001	*	$a +_n b$	0
0010	*	an b	$[a] - [b] \notin T_n$
0011	*	an b	0
0100	*	$a \wedge b$	0
0101	*	$a \lor b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \lor b}$	0
0111	1	$b[n/2-1:0]0^{n/2}$	0
1010	*	$0^{n-1}([a] < [b] ? 1 : 0)$	0
1011	*	$0^{n-1}(\langle a\rangle < \langle b\rangle ? 1:0)$	0

Table 7. Specification of ALU operations

ALU input i

$$i = itype(c)$$

ALU result

$$ares(c) \equiv alures(lop(c), rop(c), af(c), itype(c)).$$

destination register

$$rdes(c) \equiv \begin{cases} rd(c), & rtype(c), \\ rt(c), & \text{otherwise.} \end{cases}$$

all ALU operations

Arithmetic, Logical Operation, Test-and-Set					
001 000	addi	add	rt	s imm	rt = rs + sxt(imm)
001 001	addiu	addi	1 <i>rt</i>	rs imm	rt = rs + sxt(imm)
001 010	slti	slti	rt 1	s imm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$
001 011	sltiu	sltiı	rt	rs imm	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$
001 100	andi	and	rt	rs imm	$rt = rs \wedge zxt(imm)$
001 101	ori	ori	rt 1	s imm	$rt = rs \lor zxt(imm)$
001 110	xori	xor	rt	's imm	$rt = rs \oplus zxt(imm)$
001 111	lui	lu	rt	imm	$rt = imm0^{16}$

Arithmetic, Logical Operation						
000000 100 000	add	ado	rd	rs rt	rd = rs + rt	
000000 100 001	addu	add	ı rd	rs rt	rd = rs + rt	
000000 100 010	sub	sul	rd	rs rt	rd = rs - rt	
000000 100 011	subu	sub	ı rd	rs rt	rd = rs - rt	
000000 100 100	and	and	rd	rs rt	$rd = rs \wedge rt$	
000000 100 101	or	or	rd 1	s rt	$rd = rs \lor rt$	
000000 100 110	xor	XO	rd	rs rt	$rd = rs \oplus rt$	
000000 100 111	nor	no	rd	s rt	$rd = rs \vee rt$	
Test-and-Set Operation						
000000 101 010	slt	slt	rd i	s rt	$rd = (rs < rt ? 1_{32} : 0_{32})$	
000000 101 011	sltu	sltı	rd	rs rt	$rd = (rs < rt ? 1_{32} : 0_{32})$	

$$alu(c) \rightarrow \begin{cases} c'.gpr(x) = \begin{cases} ares(c), & x = rdes(c), \\ c.gpr(x), & \text{otherwise,} \end{cases} \\ c'.m = c.m, \\ c'.pc = c.pc +_{32} 4_{32}. \end{cases}$$

Arithmetic, Logical Operation, Test-and-Set					
001 000	addi	addi rt rs imm	rt = rs + sxt(imm)		
001 001	addiu	addiu <i>rt rs imm</i>	rt = rs + sxt(imm)		
001 010	slti	slti <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$		
001 011	sltiu	sltiu <i>rt rs imm</i>	$rt = (rs < sxt(imm) ? 1_{32} : 0_{32})$		
001 100	andi	andi <i>rt rs imm</i>	$rt = rs \wedge zxt(imm)$		
001 101	ori	ori <i>rt rs imm</i>	$rt = rs \lor zxt(imm)$		
001 110	xori	xori rt rs imm	$rt = rs \oplus zxt(imm)$		
001 111	lui	lui <i>rt imm</i>	$rt = imm0^{16}$		

Arithmetic, Logical Operation						
000000 100 000	add	add <i>rd rs rt</i>	rd = rs + rt			
000000 100 001	addu	addu <i>rd rs rt</i>	rd = rs + rt			
000000 100 010	sub	sub <i>rd rs rt</i>	rd = rs - rt			
000000 100 011	subu	subu <i>rd rs rt</i>	rd = rs - rt			
000000 100 100	and	and rd rs rt	$rd = rs \wedge rt$			
000000 100 101	or	or <i>rd rs rt</i>	$rd = rs \lor rt$			
000000 100 110	xor	xor rd rs rt	$rd = rs \oplus rt$			
000000 100 111	nor	nor <i>rd rs rt</i>	$rd = \overline{rs \vee rt}$			
Test-and-Set Operation						
000000 101 010	slt	slt <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			
000000 101 011	sltu	sltu <i>rd rs rt</i>	$rd = (rs < rt ? 1_{32} : 0_{32})$			
	•					