all ALU operations

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs $\wedge$ rt
000000	100 101		or	or *rd rs rt*	rd = rs $\vee$ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs $\oplus$ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

R-type

left operand

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

all ALU operations

I-type

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

R-type

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs $\wedge$ rt
000000	100 101		or	or *rd rs rt*	rd = rs $\vee$ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs $\oplus$ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

left operand

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

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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs $\wedge$ rt
000000	100 101		or	or *rd rs rt*	rd = rs $\vee$ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs $\oplus$ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

R-type

left operand

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

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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{rs \vee rt}$
Test-and-Set 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})

R-type

left operand

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

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

I-type

Arithmetic, Logical Operation, Test-and-Set					
001 0 00		addi	addi *rt rs imm*	rt = rs + sxt(imm)
001 0 01		addiu	addiu *rt rs imm*	rt = rs + sxt(imm)
001 0 10		slti	slti *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 0 11		sltiu	sltiu *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 1 00		andi	andi *rt rs imm*	rt = rs ∧ zxt(imm)
001 1 01		ori	ori *rt rs imm*	rt = rs ∨ zxt(imm)
001 1 10		xori	xori *rt rs imm*	rt = rs ⊕ zxt(imm)
001 1 11		lui	lui *rt imm*	rt = imm0^{16}

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{rs \vee rt}$
Test-and-Set 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})

R-type

left operand

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

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

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

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

all ALU operations

I-type

Arithmetic, Logical Operation, Test-and-Set					
001 0 00		addi	addi *rt rs imm*	rt = rs + sxt(imm)
001 0 01		addiu	addiu *rt rs imm*	rt = rs + sxt(imm)
001 0 10		slti	slti *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 0 11		sltiu	sltiu *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 1 00		andi	andi *rt rs imm*	rt = rs ∧ zxt(imm)
001 1 01		ori	ori *rt rs imm*	rt = rs ∨ zxt(imm)
001 1 10		xori	xori *rt rs imm*	rt = rs ⊕ zxt(imm)
001 1 11		lui	lui *rt imm*	rt = $imm0^{16}$

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{rs \vee rt}$
Test-and-Set 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})

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	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	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

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

I-type

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

Arithmetic, Logical Operation					
000000	100	000	add	add *rd rs rt*	rd = rs + rt
000000	100	001	addu	addu *rd rs rt*	rd = rs + rt
000000	100	010	sub	sub *rd rs rt*	rd = rs − rt
000000	100	011	subu	subu *rd rs rt*	rd = rs − rt
000000	100	100	and	and *rd rs rt*	rd = rs ∧ rt
000000	100	101	or	or *rd rs rt*	rd = rs ∨ rt
000000	100	110	xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100	111	nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

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	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	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

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]

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

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

all ALU operations

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

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	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

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	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

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	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

R-type

all ALU operations summary

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

I-type

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

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs $\wedge$ rt
000000	100 101		or	or *rd rs rt*	rd = rs $\vee$ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs $\oplus$ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set 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})

R-type