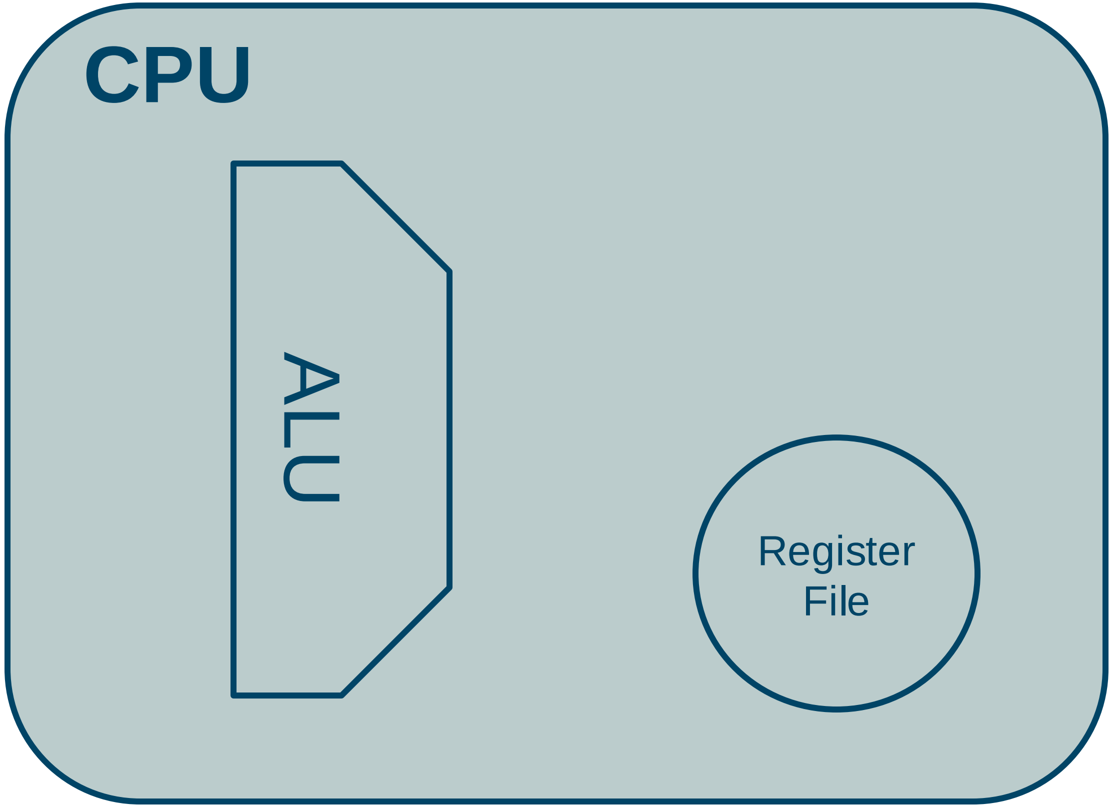
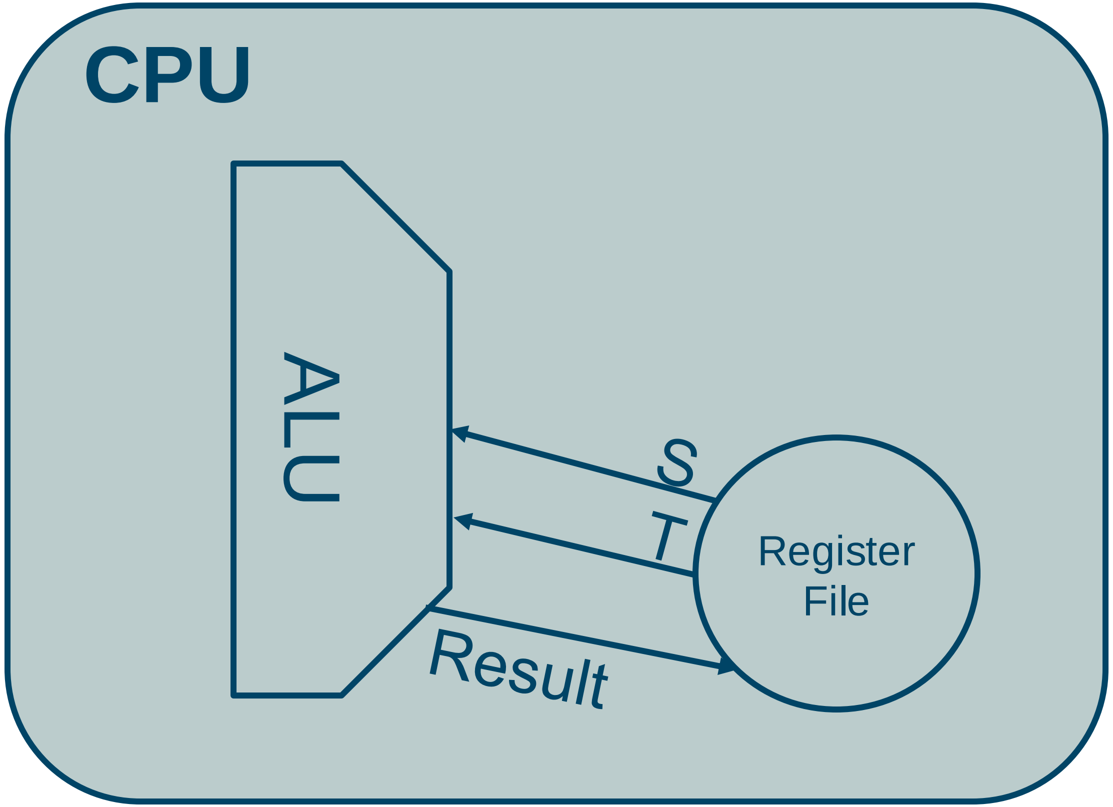
Computer Architecture

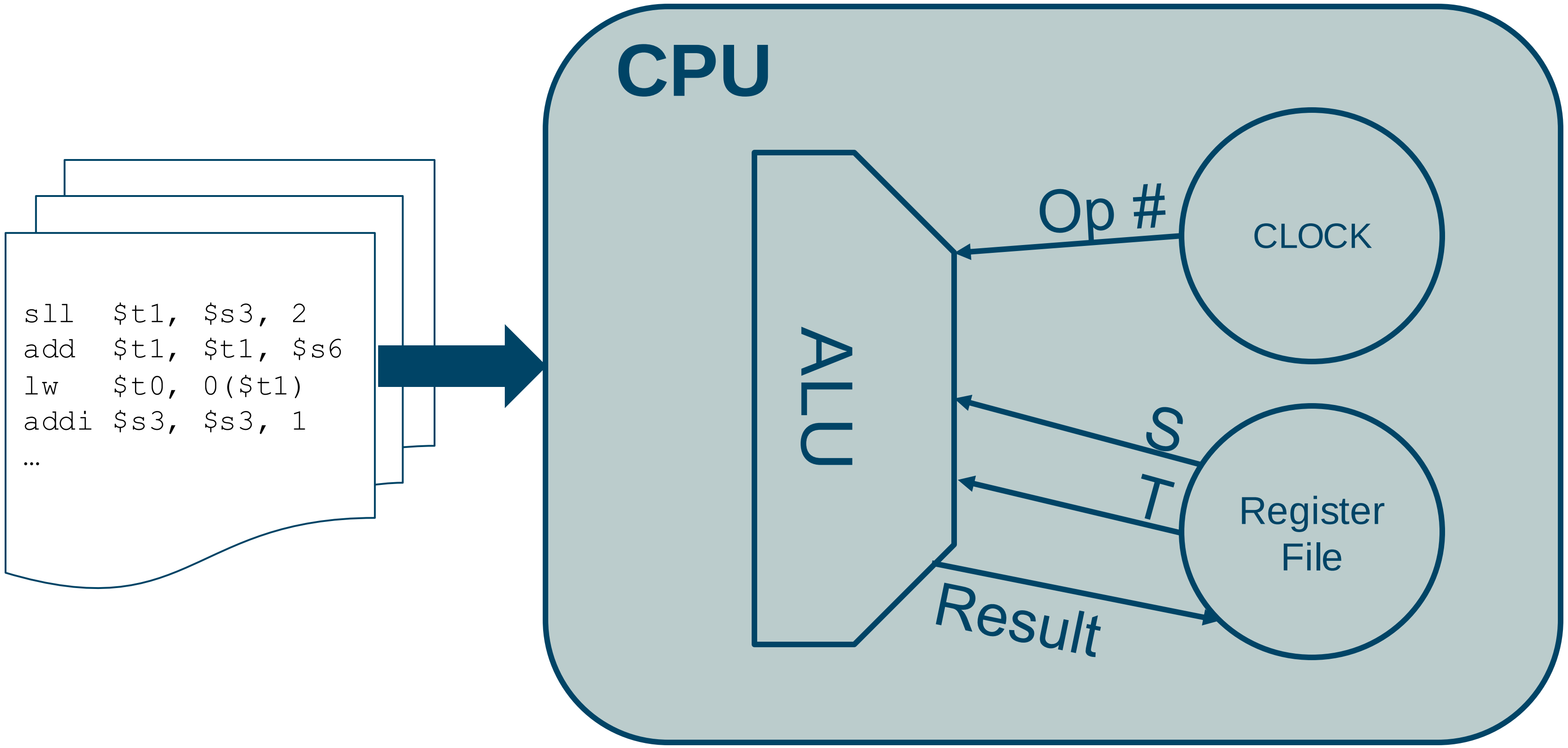
Kasper Engelen

Laurens De Wachter

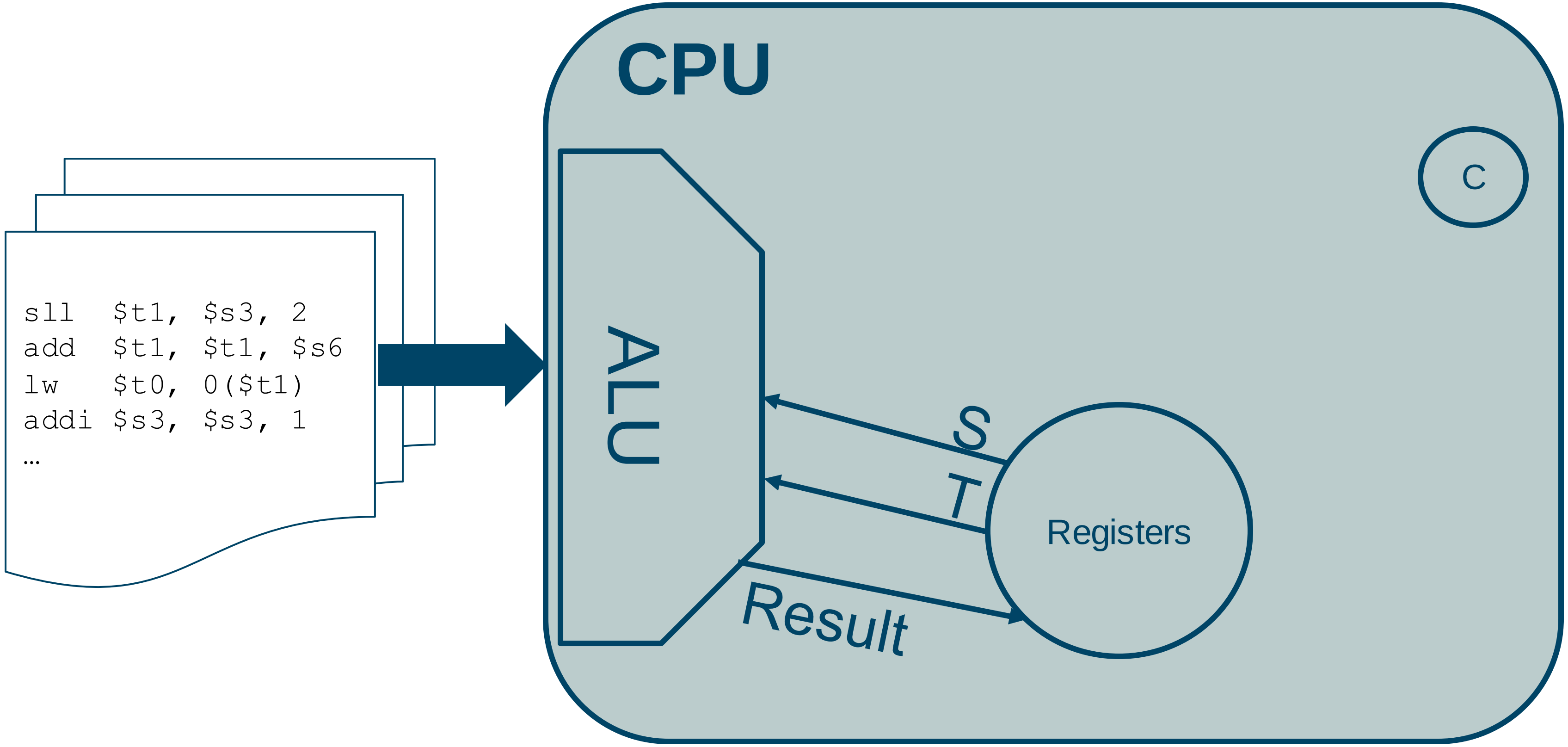
Universiteit Antwerpen


CPU
ALU
Register File

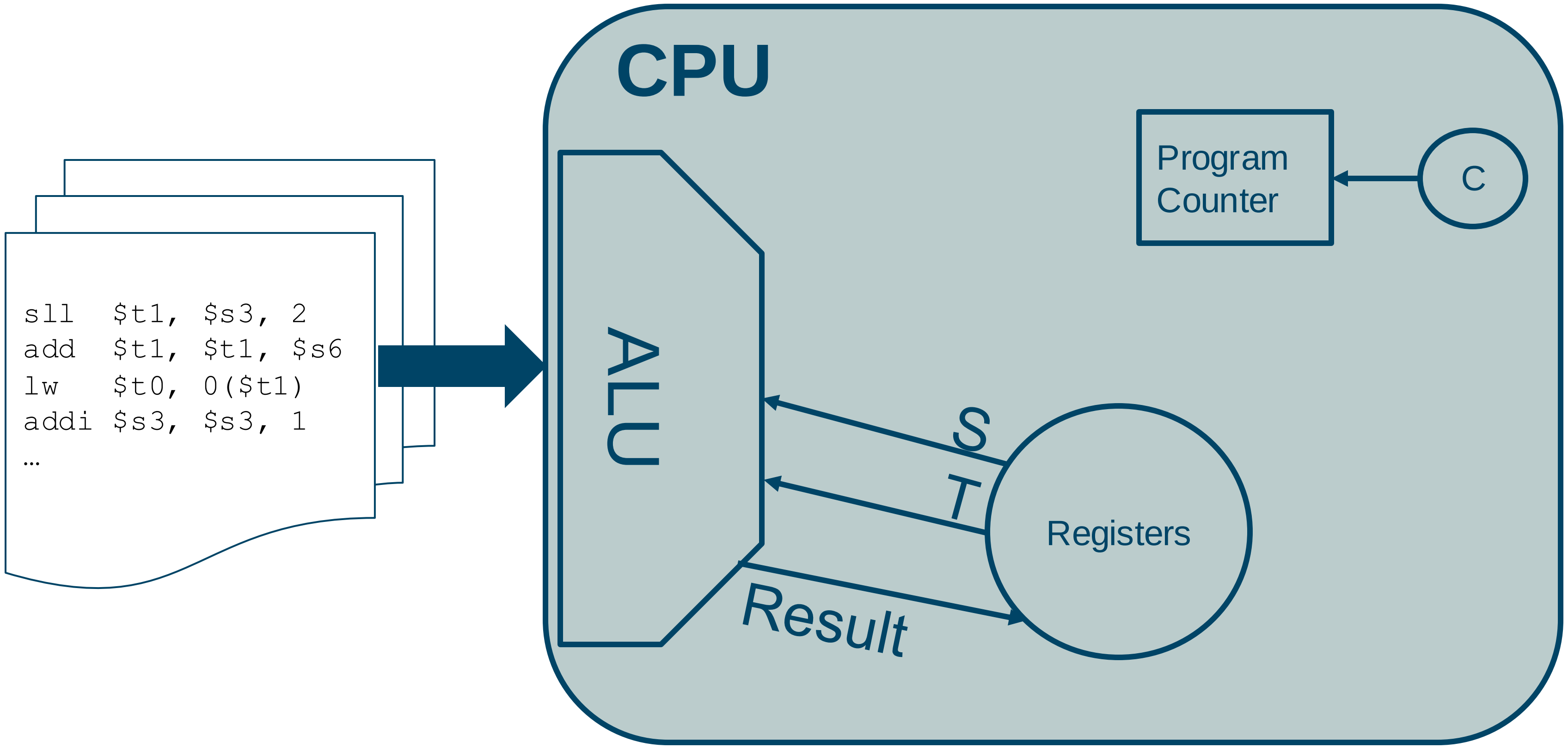

CPU
ALU
S
T
Result
Register File



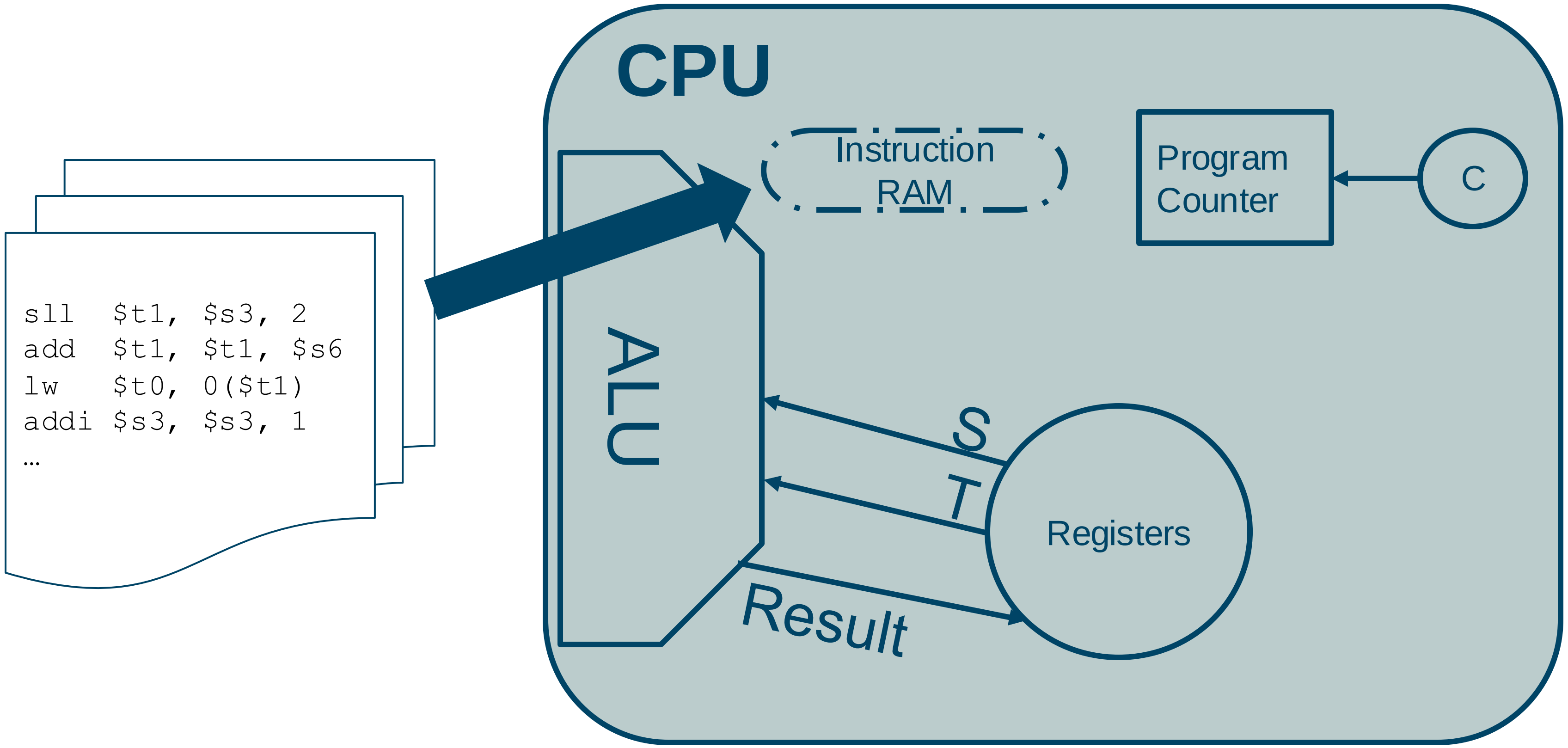
```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```

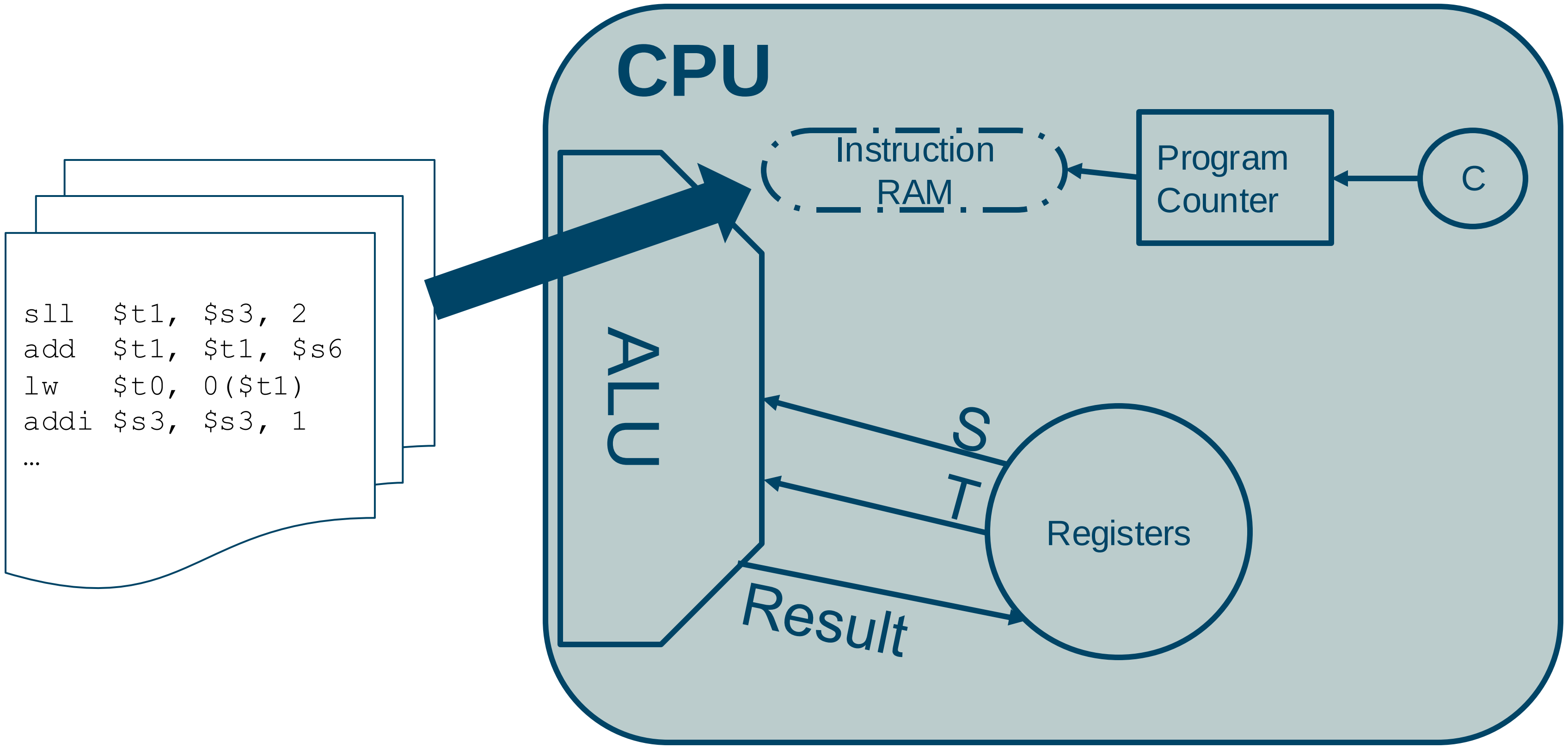


```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```

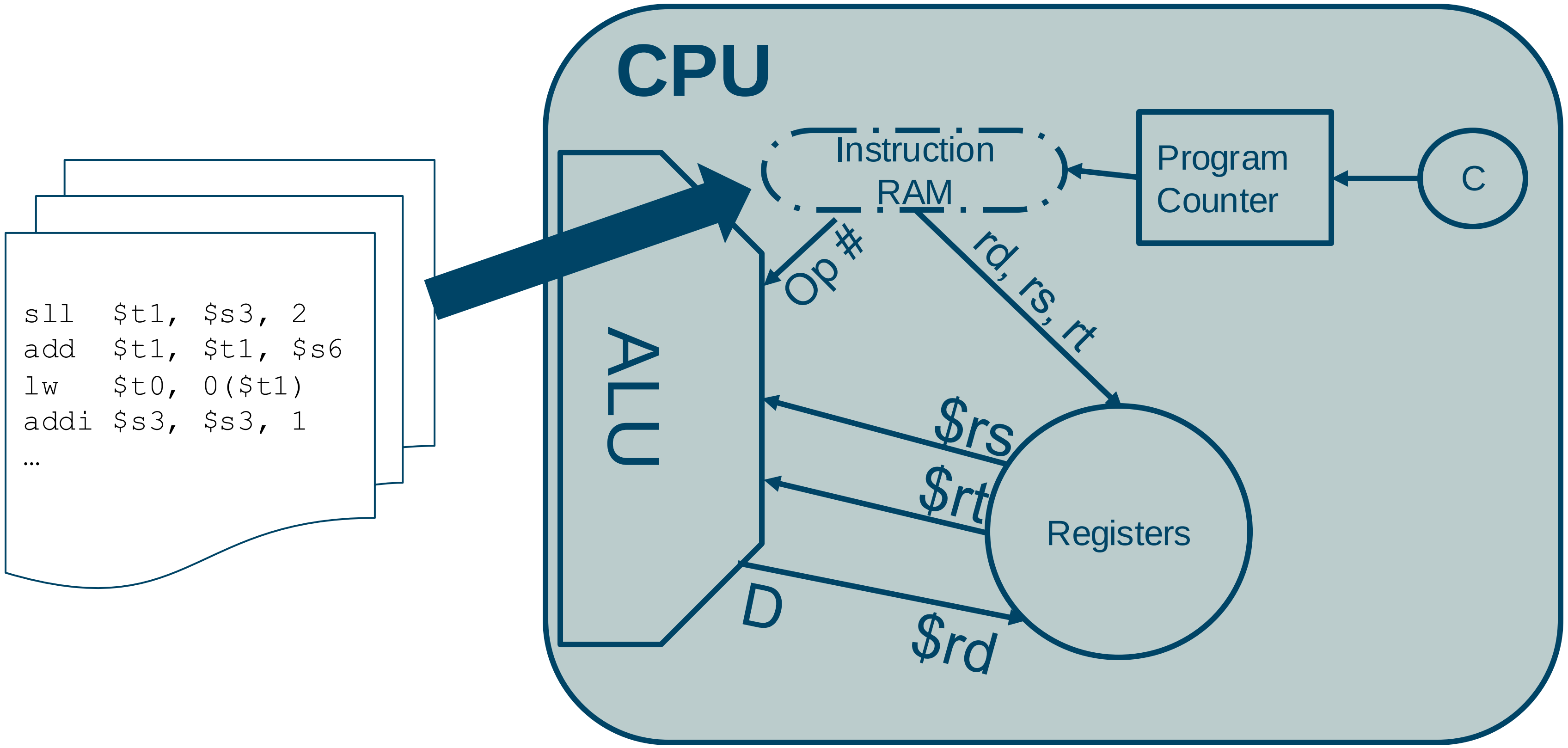


```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```


```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```



```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```



```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```

Instruction Set

Binary (12-bits)	Name	Assembly	Semantics
`<opcode> <Rd> <Rs> <Rt> <Func>`			
`000 ddd 000 000 0000`	zero	zero rd	rd = 0
`001 ddd sss ttt 0000`	add	add rd rs rt	rd = rs + rt
`001 ddd sss ttt 0001`	sub	sub rd rs rt	rd = rs - rt
`001 ddd sss ttt 0010`	and	and rd rs rt	rd = rs & rt (bitwise)
`001 ddd sss ttt 0011`	or	or rd rs rt	rd = rs | rt (bitwise)
`001 ddd sss ttt 0100`	lt	lt rd rs rt	rd = (rs < rt)
`001 ddd sss ttt 0101`	gt	gt rd rs rt	rd = (rs > rt)
`001 ddd sss ttt 0110`	eq	eq rd rs rt	rd = (rs == rt)
`001 ddd sss ttt 0111`	neq	neq rd rs rt	rd = (rs != rt)
`010 ddd sss 000 0000`	not	not rd rs	rd = !rs (bitwise)
`010 ddd sss 000 0001`	inv	inv rd rs	rd = -rs
`010 ddd sss 000 0010`	sll	sll rd rs	rd = (rs << 1)
`010 ddd sss 000 0011`	srl	srl rd rs	rd = (rs >> 1)
`010 ddd sss 000 0100`	sla	sla rd rs	rd = rs * 2
`010 ddd sss 000 0101`	sra	sra rd rs	rd = rs // 2 (integer division)
`011 ddd iii iii iii 0`	ldi	ldi rd imm (signed)	rd = imm
`011 ddd iii iii iii 1`	lui	lui rd imm (unsigned)	rd = imm << 6
`100 ddd iii iii iii 0`	addi	addi rd imm (unsigned)	rd = rs + imm
`100 ddd iii iii iii 1`	subi	subi rd imm (unsigned)	rd = rs - imm
`111 ddd sss iii iii 0`	lw	lw rd rs imm (unsigned)	rd = MEM[rs + imm]
`111 ddd sss iii iii 1`	sw	sw rd rs imm (unsigned)	MEM[rs + imm] = rd

Instruction Set

Binary (12-bits)	Name	Assembly	Semantics
<opcode> <Rd> <Rs> <Rt> <Func>			
000 ddd 000 000 0000	zero	zero rd	rd = 0
001 ddd sss ttt 0000	add	add rd rs rt	rd = rs + rt
001 ddd sss ttt 0001	sub	sub rd rs rt	rd = rs - rt
001 ddd sss ttt 0010	and	and rd rs rt	rd = rs & rt (bitwise)
001 ddd sss ttt 0011	or	or rd rs rt	rd = rs | rt (bitwise)
001 ddd sss ttt 0100	lt	lt rd rs rt	rd = (rs < rt)
001 ddd sss ttt 0101	gt	gt rd rs rt	rd = (rs > rt)
001 ddd sss ttt 0110	eq	eq rd rs rt	rd = (rs == rt)
001 ddd sss ttt 0111	neq	neq rd rs rt	rd = (rs != rt)
010 ddd sss 000 0000	not	not rd rs	rd = !rs (bitwise)
010 ddd sss 000 0001	inv	inv rd rs	rd = -rs
010 ddd sss 000 0010	sll	sll rd rs	rd = (rs << 1)
010 ddd sss 000 0011	srl	srl rd rs	rd = (rs >> 1)
010 ddd sss 000 0100	sla	sla rd rs	rd = rs * 2
010 ddd sss 000 0101	sra	sra rd rs	rd = rs // 2 (integer division)
011 ddd iii iii iii 0	ldi	ldi rd imm (signed)	rd = imm
011 ddd iii iii iii 1	lui	lui rd imm (unsigned)	rd = imm << 6
100 ddd iii iii iii 0	addi	addi rd imm (unsigned)	rd = rs + imm
100 ddd iii iii iii 1	subi	subi rd imm (unsigned)	rd = rs - imm
111 ddd sss iii iii 0			= MEM[rs + imm]
111 ddd sss iii iii 1			[rs + imm] = rd

Zero Instruction

Instruction Set

Binary (12-bits)	Name	Assembly	Semantics
`<opcode> <Rd> <Rs> <Rt> <Func>`			
`000 ddd 000 000 0000`	zero	zero rd	rd = 0
`001 ddd sss ttt 0000`	add	add rd rs rt	rd = rs + rt
`001 ddd sss ttt 0001`	sub	sub rd rs rt	rd = rs - rt
`001 ddd sss ttt 0010`	and	and rd rs rt	rd = rs & rt (bitwise)
`001 ddd sss ttt 0011`	or	or rd rs rt	rd = rs | rt (bitwise)
`001 ddd sss ttt 0100`	lt	lt rd rs rt	rd = (rs < rt)
`001 ddd sss ttt 0101`	gt	gt rd rs rt	rd = (rs > rt)
`001 ddd sss ttt 0110`	eq	eq rd rs rt	rd = (rs == rt)
`001 ddd sss ttt 0111`	neq	neq rd rs rt	rd = (rs != rt)
`010 ddd sss 000 0000`	not	not rd rs	rd = !rs (bitwise)
`010 ddd sss 000 0001`	inv	inv rd rs	rd = -rs
`010 ddd sss 000 0010`	sll	sll rd rs	rd = (rs << 1)
`010 ddd sss 000 0011`	srl	srl rd rs	rd = (rs >> 1)
`010 ddd sss 000 0100`	sla	sla rd rs	rd = rs * 2
`010 ddd sss 000 0101`	sra	sra rd rs	rd = rs // 2 (integer division)
`011 ddd iii iii iii 0`	ldi	ldi rd imm (signed)	rd = imm
`011 ddd iii iii iii 1`	lui	lui rd imm (unsigned)	rd = imm << 6
`100 ddd iii iii iii 0`	addi	addi rd imm (unsigned)	rd = rs + imm
`100 ddd iii iii iii 1`	subi	subi rd imm (unsigned)	rd = rs - imm
`111 ddd sss iii iii 0`			= MEM[rs + imm]
`111 ddd sss iii iii 1`			[rs + imm] = rd

Register Instructions

Instruction Set

Binary (12-bits)	Name	Assembly	Semantics
`<opcode> <Rd> <Rs> <Rt> <Func>`			
`000 ddd 000 000 0000`	`zero`	`zero rd`	`rd = 0`
`001 ddd sss ttt 0000`	`add`	`add rd rs rt`	`rd = rs + rt`
`001 ddd sss ttt 0001`	`sub`	`sub rd rs rt`	`rd = rs - rt`
`001 ddd sss ttt 0010`	`and`	`and rd rs rt`	`rd = rs & rt` (bitwise)
`001 ddd sss ttt 0011`	`or`	`or rd rs rt`	`rd = rs | rt` (bitwise)
`001 ddd sss ttt 0100`	`lt`	`lt rd rs rt`	`rd = (rs < rt)`
`001 ddd sss ttt 0101`	`gt`	`gt rd rs rt`	`rd = (rs > rt)`
`001 ddd sss ttt 0110`	`eq`	`eq rd rs rt`	`rd = (rs == rt)`
`001 ddd sss ttt 0111`	`neq`	`neq rd rs rt`	`rd = (rs != rt)`
`010 ddd sss 000 0000`	`not`	`not rd rs`	`rd = !rs` (bitwise)
`010 ddd sss 000 0001`	`inv`	`inv rd rs`	`rd = -rs`
`010 ddd sss 000 0010`	`sll`	`sll rd rs`	`rd = (rs << 1)`
`010 ddd sss 000 0011`	`srl`	`srl rd rs`	`rd = (rs >> 1)`
`010 ddd sss 000 0100`	`sla`	`sla rd rs`	`rd = rs * 2`
`010 ddd sss 000 0101`	`sra`	`sra rd rs`	`rd = rs // 2` (integer division)
`011 ddd iii iii iii 0`	`ldi`	`ldi rd imm` (signed)	`rd = imm`
`011 ddd iii iii iii 1`	`lui`	`lui rd imm` (unsigned)	`rd = imm << 6`
`100 ddd iii iii iii 0`	`addi`	`addi rd imm` (unsigned)	`rd = rs + imm`
`100 ddd iii iii iii 1`	`subi`	`subi rd imm` (unsigned)	`rd = rs - imm`
`111 ddd sss iii iii 0`			`= MEM[rs + imm]`
`111 ddd sss iii iii 1`			`[rs + imm] = rd`

Unary Operations

Instruction Set

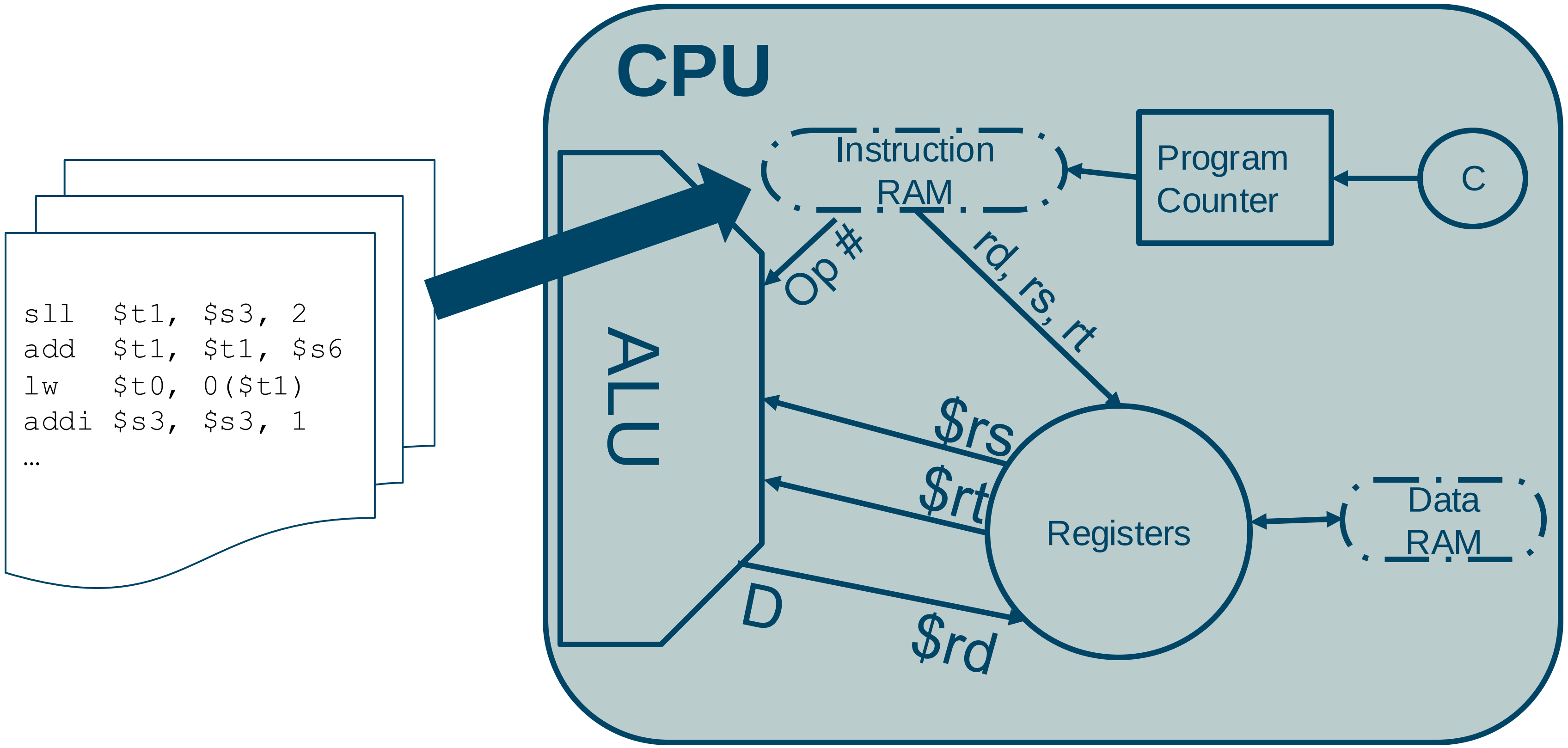
Binary (12-bits)	Name	Assembly	Semantics
`<opcode> <Rd> <Rs> <Rt> <Func>`			
`000 ddd 000 000 0000`	`zero`	`zero rd`	`rd = 0`
`001 ddd sss ttt 0000`	`add`	`add rd rs rt`	`rd = rs + rt`
`001 ddd sss ttt 0001`	`sub`	`sub rd rs rt`	`rd = rs - rt`
`001 ddd sss ttt 0010`	`and`	`and rd rs rt`	`rd = rs & rt` (bitwise)
`001 ddd sss ttt 0011`	`or`	`or rd rs rt`	`rd = rs | rt` (bitwise)
`001 ddd sss ttt 0100`	`lt`	`lt rd rs rt`	`rd = (rs < rt)`
`001 ddd sss ttt 0101`	`gt`	`gt rd rs rt`	`rd = (rs > rt)`
`001 ddd sss ttt 0110`	`eq`	`eq rd rs rt`	`rd = (rs == rt)`
`001 ddd sss ttt 0111`	`neq`	`neq rd rs rt`	`rd = (rs != rt)`
`010 ddd sss 000 0000`	`not`	`not rd rs`	`rd = !rs` (bitwise)
`010 ddd sss 000 0001`	`inv`	`inv rd rs`	`rd = -rs`
`010 ddd sss 000 0010`	`sll`	`sll rd rs`	`rd = (rs << 1)`
`010 ddd sss 000 0011`	`srl`	`srl rd rs`	`rd = (rs >> 1)`
`010 ddd sss 000 0100`	`sla`	`sla rd rs`	`rd = rs * 2`
`010 ddd sss 000 0101`	`sra`	`sra rd rs`	`rd = rs // 2` (integer division)
`011 ddd iii iii iii 0`	`ldi`	`ldi rd imm` (signed)	`rd = imm`
`011 ddd iii iii iii 1`	`lui`	`lui rd imm` (unsigned)	`rd = imm << 6`
`100 ddd iii iii iii 0`	`addi`	`addi rd imm` (unsigned)	`rd = rs + imm`
`100 ddd iii iii iii 1`	`subi`	`subi rd imm` (unsigned)	`rd = rs - imm`
`111 ddd sss iii iii 0`			`= MEM[rs + imm]`
`111 ddd sss iii iii 1`			`[rs + imm] = rd`

Binary Operations

Instruction Set

Binary (12-bits)	Name	Assembly	Semantics
`<opcode> <Rd> <Rs> <Rt> <Func>`			
`000 ddd 000 000 0000`	`zero`	`zero rd`	`rd = 0`
`001 ddd sss ttt 0000`	`add`	`add rd rs rt`	`rd = rs + rt`
`001 ddd sss ttt 0001`	`sub`	`sub rd rs rt`	`rd = rs - rt`
`001 ddd sss ttt 0010`	`and`	`and rd rs rt`	`rd = rs & rt` (bitwise)
`001 ddd sss ttt 0011`	`or`	`or rd rs rt`	`rd = rs | rt` (bitwise)
`001 ddd sss ttt 0100`	`lt`	`lt rd rs rt`	`rd = (rs < rt)`
`001 ddd sss ttt 0101`	`gt`	`gt rd rs rt`	`rd = (rs > rt)`
`001 ddd sss ttt 0110`	`eq`	`eq rd rs rt`	`rd = (rs == rt)`
`001 ddd sss ttt 0111`	`neq`	`neq rd rs rt`	`rd = (rs != rt)`
`010 ddd sss 000 0000`	`not`	`not rd rs`	`rd = !rs` (bitwise)
`010 ddd sss 000 0001`	`inv`	`inv rd rs`	`rd = -rs`
`010 ddd sss 000 0010`	`sll`	`sll rd rs`	`rd = (rs << 1)`
`010 ddd sss 000 0011`	`srl`	`srl rd rs`	`rd = (rs >> 1)`
`010 ddd sss 000 0100`	`sla`	`sla rd rs`	`rd = rs * 2`
`010 ddd sss 000 0101`	`sra`	`sra rd rs`	`rd = rs // 2` (integer division)
`011 ddd iii iii iii 0`	`ldi`	`ldi rd imm` (signed)	`rd = imm`
`011 ddd iii iii iii 1`			`= imm << 6`
`100 ddd iii iii iii 0`			`= rs + imm`
`100 ddd iii iii iii 1`			`= rs - imm`
`111 ddd sss iii iii 0`	`lw`	`lw rd rs imm` (unsigned)	`rd = MEM[rs + imm]`
`111 ddd sss iii iii 1`	`sw`	`sw rd rs imm` (unsigned)	`MEM[rs + imm] = rd`

Memory Instructions



```
sll  $t1, $s3, 2
add  $t1, $t1, $s6
lw   $t0, 0($t1)
addi $s3, $s3, 1
…
```