Team 2-1
Register Conventions/Descriptions

Name	Numerical Value	Description
$0	0	Save for value of 0
$temp0-$temp4	1-5	Temporary Registers
$store0-$store2	7-9	Storing Registers
$arg0-$arg1	10-11	Argument for Procedure
$ret0-$ret1	12-13	Return Values
$tempa	14	Reserved for the Assembler
$ra	15	Return Address
$spr0	6	Input Register

Team 2-1
Assembly to Machine Language translations

Assembly	Binary
add	0000
sub	0001
or	0010
N/A	0011
and	0100
lsh	0101
slt	0110
goal	0111
go	1000
gore	1001
ceq	1010
cne	1011
lv	1100
sv	1101
luc	1110
llc	1111

Register	Binary
$0	0000
$temp0	0001
$temp1	0010
$temp2	0011
$temp3	0100
$temp4	0101
$spr0	0110
$store0	0111
$store1	1000
$store2	1001
$arg0	1010
$arg1	1011
$ret0	1100
$ret1	1101
$tempa	1110
$ra	1111

Refer to the syntax instruction document for the parameter order of the commands

Team 2-1
Machine Language Instruction Format

Register type Commands (Ex. add)

op	rs	rt	rd
4	4	4	4

Jump type Commands (Ex. goal)

op	Label
4	12

Immediate Type Commands (Ex. lv)

op	rd	const
4	4	8

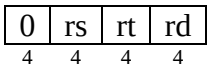
or

op	rs	rd	const
4	4	4	4

op: instruction code
rs: source register
rt: second source register
rd: destination register
const: constant value
Label: address

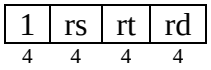
Team 2-1
Syntax of each instruction

Addition
`add rd, rs, rt`



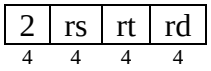
This command adds `rt` to `rs` and stores it in `rd`.

Subtraction
`sub rd, rs, rt`



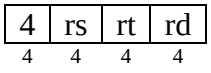
This command subtracts `rt` from `rs` and stores it in `rd`.

Logical Or
`or rd, rs, rt`



This command ORs `rs` and `rt` and stores the value in `rd`.

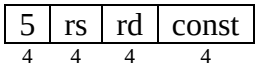
Logical And
`and rd, rs, rt`



This command ANDs `rs` and `rt` and stores the value in `rd`.

Logical Left Shift

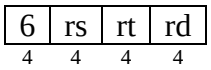
`lsh rd, rs, const`



This command shifts all the bits left from `rs` to the left one bit per `const` and stores the value in `rd`.

Set Less Than

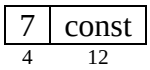
`slt rd, rs, rt`



If `rs` is less than `rt`, `rd` is 1, else `rd` is 0.

GoTo Label and Link

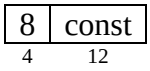
`goal const`



This command stores the address following the current address into `ra`, then goes to `Label`.

GoTo Label

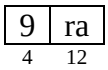
`go const`



This command goes to address represented by `const`

GoTo Return Address

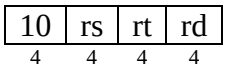
`gore ra`



This command goes to address stored in `ra`.

Conditional GoTo if Equal

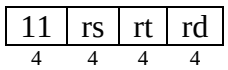
`ceq rs, rt, rd`



This command goes to the address stored in `rd` if `rs` is equal to `rt`.
NOTE: This should be executed from the pseudo command ceq

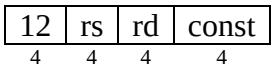
Conditional GoTo if Not Equal

`cne rs, rt, rd`



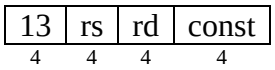
This command goes to the address stored in `rd` if `rs` in not equal to `rt`.
NOTE: This should be executed from the pseudo command cne

Load Value
`lv rd, const(rs)`



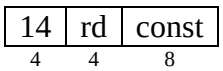
This command loads the value stored in `(rs + const)`, and stores it to `rd`.

Store Value
`sv rs, const(rd)`



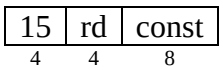
This command stores `rs` into `rd + const`.

Load Upper Constant
`luc rd, const`



This command loads `const` into upper 8 bits of `rd`.
NOTE: to load a big constant use the pseudo command lc

Load Lower Constant
`llw rd, const`



This command loads `const` into lower 8 bits part of `rd`.
NOTE: to load a big constant use the pseudo command lc

Logical Or Constant
`orc rd, rs, const` *~pseudocommand~*

This command logical ORs `rs` and `const` and stores the value in `rd`.

Load Address
`la rd, Label` *~pseudocommand~*

This command loads `Label` into `rd`.

Load Constant
`lc rd, const` *~pseudocommand~*

This command loads `const` into `rd`.

Move
`move rd, rs` *~pseudocommand~*

This command stores `rs` to `rd`.

Conditional GoTo if Equal (pseudo)

`ceq rs, rt, Label` *~pseudocommand~*

This command stores `rs` to `rd`.

Conditional GoTo if Not Equal (pseudo)

`cnq rs, rt, Label` *~pseudocommand~*

This command stores `rs` to `rd`.