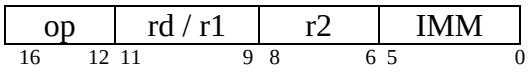
Team2-4
Design Document
January 8th, 2004

Assembly Language Instructions

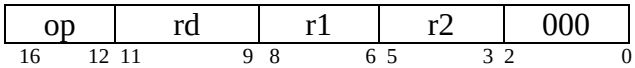
Below is a revised listing of commands in our assembly language.

Instruction Name	Command Form	Meaning
add	add $1, $2	$1=$1+$2
sub	sub $1, $2	$1=$2-$1
ori	ori $1, $2, IMM	$1=$2 | IMM
sll	sll $1, $2, IMM	$1=$2 << IMM
lw	lw $2	$2=Memory[$2]
sw	sw $2	Memory[$2] = $2
beq	beq $1, $2, IMM	If ($1==$2) go to PC+2+IMM
slt	slt $1, $2, $3	If($2<$3) $1=1; else $1=0
j	j IMM	Jumps to IMM
swap	swap $t1, $t2	Swap the values of $1 and $2
or	or $1, $2, $3	$2 or $3, placed in $1
lu	lu $1, IMM	$1 = IMM * 2^8
jr	jr $1	Jumps to the address in $1
jal	jal IMM	Jumps to IMM and stores the current PC in JALReg

Machine Language Instruction Format

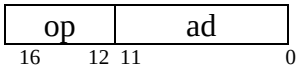


- *op*: Operation instruction code
- *rd*: The destination register, where the results are stored
- *r1*: The first register source operand
- *r2*: The second register source operand
- *IMM:* The last 6 bits are either used for the immediate value, or are not used in this operations and are set to all 0's



- *op*: Operation instruction code
- *rd*: The destination register, where the results are stored
- *r1*: The first register source operand
- *r2*: The second register source operand

- The last 3 bits are always set to 0's



- *op*: Operation instruction code
- *ad*: Address to jump to

Rules for translating assembly language into machine language

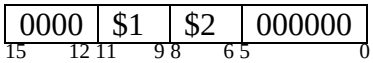
Look at the 4 most significant bits, and from the operation command, and determine what to do from there.

Machine Language Specifications

Below are the Machine Language Specifications, listing all of our commands in machine language. The first four digits in all of our commands are comprised of the op-code. This can be a number ranging from 0000 to 1111.

Add

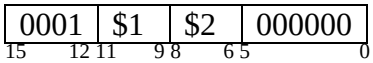
add $1, $2



This command adds the contents of $1 and $2 and writes that number back in $1.

Subtract

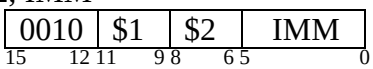
sub $1, $2



This command subtracts the contents of $2 from $1 and writes that number back in $1.

OR Immediate

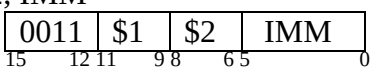
ori $1, $2, IMM



This command or's the contents of $2 and the zero-extended immediate value then stores that value in $1.

Shift Left Logical

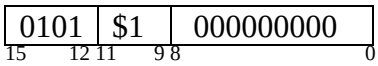
sll $1, $2, IMM



This command shifts the value of $2 the distance of the immediate value to the left then writes the new value in $1.

Load Word

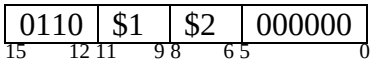
lw $1



This command goes to the address in memory that is contained in $1, it takes the contents of that address and writes that value back into $1.

Store Word

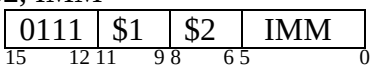
sw $1, $2



This command goes to the address in memory that is contained in $2 and writes the value of $1 into it.

Branch Equal To

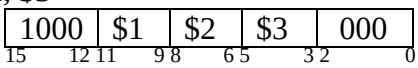
beq $1, $2, IMM



This command compares the values of $1 and $2 and if they are equal, it sets the PC equal to the PC plus four plus the immediate value.

Set Less Than

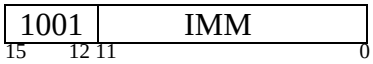
slt $1, $2, $3



This command checks to see if $2 is less than $3, if it is it sets the value of $1 to 1, otherwise, $1 is set to 0.

Jump

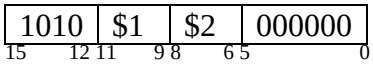
j IMM



This command jumps to the location of the immediate value.

Swap

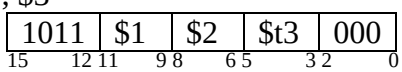
Swap $1, $2



This command copies the contents of $2 into $r0 then puts the value of $1 in $2, then puts the value of $r0 into $1.

Or

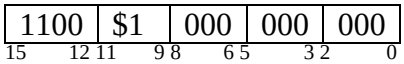
or $1, $2, $3



This command or’s the values of $2 and $3, and then it writes that value into $1.

Jump Register

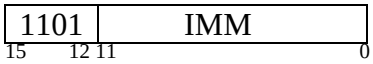
jr $1



This command jumps to the address specified by the value in $1.

Jump and Link

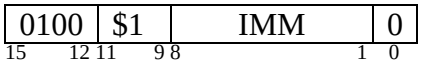
jal IMM



This command jumps to the location of the immediate value and stores the value of the PC in the special purpose register JALReg.

Load Upper

lu $1, IMM



This command loads the immediate value into the upper 8 bits of $1.

Register Usage Conventions

Register Name	Register Usage
$1	temporary (not preserved)
$2	temporary (not preserved)
$3	temporary (not preserved)
$4	temporary (not preserved)
$5	argument 1
$6	argument 2
$7	saved temporary 1 (preserved)
$8	saved temporary 2 (preserved)

- Registers $1 through $4 are used to store temporary values which do not necessarily needed to be saved across procedure calls.
- Registers $5 and $6 are used to store values that are passed to a procedure.
- Register $7 is used to store values which need to be saved across multiple register calls.
- Register $display only stores the value which is being saved in the display

Special Register Usage Conventions

Register Name	Register Usage
A	Stores the value of the register described by IR[11:9]
B	Stores the value of the register described by IR[8:6]
C	Stores the value of the register described by IR[5:3]
ALUReg	Temporary storage for the output of the ALU
EPC	Stores the previous PC while interrupts are being handled
PC	The program line counter
IR	Temporary storage that saves the value of the current instruction
INPUT	Stores the value of the 16-bit input bus
OUTPUT	Stores the value of the 16-bit output, to be displayed
FLAG	Stores the information that specifies interrupts that must be handled