VMIPS instructions

addvv.d	Vd, Vs, Vt	Add elements of Vs and Vt, result in Vd
addvs.d	Vd, Vs, Ft	Add elements of Vs to Ft, result in Vd
subvv.d	Vd, Vs, Vt	Subtract elements of Vt from Vs, result in Vd
subvs.d	Vd, Vs, Ft	Subtract Ft from elements of Vs, result in Vd
subsv.d	Vd, Fs, Vt	Subtract elements of Vt from Fs, result in Vd
mulvv.d	Vd, Vs, Vt	Multiply elements of Vs and Vt, result in Vd
mulvs.d	Vd, Vs, Ft	Multiply elements of Vs by Ft, result in Vd
divvv.d	Vd, Vs, Vt	Divide elements of Vs and Vt, result in Vd
divvs.d	Vd, Vs, Ft	Divide elements of Vs by Ft, result in Vd
divsv.d	Vd, Fs, Vt	Divide Fs by elements of Vt, result in Vd
lv	Vd, (Rs)	Load into Vd from memory starting at address Rs
sv	Vd, (Rs)	Store Vd into memory starting at address Rs
lvws	Vd, (Rs, Rt)	Load into Vd starting from Rs with stride Rt (i.e., addresses Rs, $\mathtt{Rs} + \mathtt{Rt}$, $\mathtt{Rs} + 2 \cdot \mathtt{Rt}, \ldots$)
svws	Vd, (Rs, Rt)	Store Vd starting from Rs with stride Rt (i.e., addresses Rs, $\mathtt{Rs} + \mathtt{Rt}$, $\mathtt{Rs} + 2 \cdot \mathtt{Rt}, \ldots$)
lvi	Vd, (Rs + Vt)	Load into Vd at addresses $\mathtt{Rs} + \mathtt{Vt}_i$
svi	Vd, (Rs + Vt)	Store Vd at addresses $\mathtt{Rs} + \mathtt{Vt}_i$
cvi	Vd, Rs	Set Vd to hold the indices 0, Rs, $2 \cdot \mathtt{Rs}, \ldots$
sXXvv.d	Vs, Vt	Compare elements of Vs with Vt using **XX** = EQ/NE/LT/LE/GT/GE; 0/1 results into VM
sXXvs.d	Vs, Ft	Compare elements of Vs with Ft using **XX** = EQ/NE/LT/LE/GT/GE; 0/1 results into VM
pop	Rd, VM	Place into Rd the number the 1s in VM
cvm		Set VM to all 1s.
mtc1	VLR, Rs	Move Rs into vector length register VLR
mfc1	Rd, VLR	Move vector length register VLR into Rd
mvtm	VM, Fs	Move Fs into vector mask register VM
mvfm	Fd, VM	Move vector mask register VM into Fd