MIPS reference card

add rd, rs, rt	Add	rd = rs + rt	R 0 / 20	registers
sub rd, rs, rt	Subtract	rd = rs - rt	R 0 / 22	\$0 \$zero
addi rt, rs, imm	Add Imm.	$rt = rs + imm \pm$	I 8	\$1 \$at
addu rd, rs, rt	Add Unsigned	rd = rs + rt	R 0 / 21	\$2-\$3 \$v0-\$v1
subu rd, rs, rt	Subtract Unsigned	rd = rs - rt	R 0 / 23	\$4-\$7 \$a0-\$a3
addiu rt, rs, imm	Add Imm. Unsigned	$rt = rs + imm_{\pm}$	I 9	\$8-\$15 \$t0-\$t7
mult rs, rt	Multiply	{hi, lo} = rs * rt	R 0 / 18	\$16-\$23 \$s0-\$s7
div rs, rt	Divide	lo = rs / rt; hi = rs % rt	R 0 / 1a	\$24-\$25 \$t8-\$t9
multu rs, rt	Multiply Unsigned	$\{hi, lo\} = rs * rt$	R 0 / 19	\$26-\$27 \$k0-\$k1
divu rs, rt	Divide Unsigned	lo = rs / rt; hi = rs % rt	R 0 / 1b	\$28 \$gp
mfhi rd	Move From Hi	rd = hi	R 0 / 10	\$29 \$sp
mflo rd	Move From Lo	rd = lo	R 0 / 12	\$30 \$fp
and rd, rs, rt	And	rd = rs & rt	R 0 / 24	\$31 \$ra
or rd, rs, rt	Or	rd = rs rt	R 0 / 25	hi —
nor rd, rs, rt	Nor	rd = ~(rs rt)	R 0 / 27	lo —
xor rd, rs, rt	eXclusive Or	rd = rs ^ rt	R 0 / 26	PC —
andi rt, rs, imm	And Imm.	rt = rs & imm ₀	Ιc	co \$13 c0_cause
ori rt, rs, imm	Or Imm.	rt = rs imm ₀	I d	co \$14 c0_epc
xori rt, rs, imm	eXclusive Or Imm.	rt = rs ^ imm ₀	I e	-
sll rd, rt, sh	Shift Left Logical	rd = rt << sh	R 0 / 0	syscall codes
srl rd, rt, sh	Shift Right Logical	rd = rt >>> sh	R 0 / 2	for MARS/SPIM
sra rd, rt, sh	Shift Right Arithmetic	rd = rt >> sh	R 0 / 3	1 print integer
sllv rd, rt, rs	Shift Left Logical Variable	rd = rt << rs	R 0 / 4	2 print float
srlv rd, rt, rs	Shift Right Logical Variable	rd = rt >>> rs	R 0 / 6	3 print double
srav rd, rt, rs	Shift Right Arithmetic Variable	rd = rt >> rs	R 0 / 7	4 print string
slt rd, rs, rt	Set if Less Than	rd = rs < rt ? 1 : 0	R 0 / 2a	5 read integer
sltu rd, rs, rt	Set if Less Than Unsigned	rd = rs < rt ? 1 : 0	R 0 / 2b	6 read float
slti rt, rs, imm	Set if Less Than Imm.	rt = rs < imm+? 1 : 0	I a	7 read double
sltiu rt, rs, imm	Set if Less Than Imm. Unsigned	$rt = rs < imm \pm ? 1 : 0$	I b	8 read string
j addr	Jump	PC = PC&0xF0000000 (addrn<< 2)		9 sbrk/alloc. mem.
		10 10001110000000 (000010 : 2)	· -) SULK/alloc. Hichi.
_	÷	$$ra = PC + 8$: $PC = PC & 0 \times F00000000 \mid (addross 2)$		
jal addr	Jump And Link	\$ra = PC + 8; PC = PC&0xF0000000 (addr ₀ << 2) PC = rs	J 3	10 exit
jal addr jr rs	Jump And Link Jump Register	PC = rs	J 3 R 0 / 8	10 exit 11 print character
jal addr jr rs jalr rs	Jump And Link Jump Register Jump And Link Register	PC = rs \$ra = PC + 8; PC = rs	J 3 R 0/8 R 0/9	10 exit 11 print character 12 read character
<pre>jal addr jr rs jalr rs beq rt, rs, imm</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal	PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2)	J 3 R 0 / 8 R 0 / 9 I 4	10 exit 11 print character 12 read character 13 open file
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal	PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2)	J 3 R 0/8 R 0/9 I 4 I 5	10 exit 11 print character 12 read character 13 open file 14 read file
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call	PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm.	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16</pre>	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c	10 exit 11 print character 12 read character 13 open file 14 read file
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±])</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Load Byte Unsigned	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) nt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Load Half	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±])</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFFF</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFFF rt = M4[rs + imm±]</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) rt, imm(rs) rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Store Byte Store Half	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt M1[rs + imm±] = rt M2[rs + imm±] = rt</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) sw rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt M4[rs + imm±] = rt</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] = rt rt = M4[rs + imm±]</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) sw rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt M4[rs + imm±] = rt</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] = rt rt = M4[rs + imm±]</pre>	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lh rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) sw rt, imm(rs) srt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked Store Conditional	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] & 0xFFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] = rt M4[rs + imm±] = rt; rt = atomic ? 1 : 0</pre>	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break
jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm(rs) lbu rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) srt, imm(rs)	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked Store Conditional	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] M1[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt; rt = atomic ? 1 : 0</pre> 6 bits 5 bits 5 bits 5 bits 5 bits R Op rs rt rd sh	J 3 R 0/8 R 0/9 I 4 I 5 R 0/c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break a reserved instr.
jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm(rs) lbu rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) sc rt, imm(rs) sc rt, imm(rs) sc rt, imm(rs)	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked Store Conditional	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] M1[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt; rt = atomic ? 1 : 0</pre> 6 bits 5 bits 5 bits 5 bits 5 bits R OP rs rt rd sh	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break a reserved instr. b coproc. unusable
<pre>jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm(rs) lbu rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) st rt, imm(rs) st rt, imm(rs) st rt, imm(rs) sc rt, imm(rs) sc rt, imm(rs) sc rt, imm(rs)</pre>	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Store Byte Store Half Store Word Load Linked Store Conditional o-instructions Branch if Greater or Equal Branch if Less or Equal Branch if Less Than	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] M1[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt; rt = atomic ? 1 : 0</pre> 6 bits 5 bits 5 bits 5 bits 5 bits R Op rs rt rd sh	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break a reserved instr. b coproc. unusable c arith. overflow
jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm lb rt, imm(rs) lbu rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) srt, imm(rs)	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Load Word Store Byte Store Half Store Word Load Linked Store Conditional lo-instructions Branch if Greater or Equal Branch if Less or Equal Branch if Less Than Load Address	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFF rt = M4[rs + imm±] M1[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt; rt = atomic ? 1 : 0</pre> 6 bits 5 bits 5 bits 5 bits 5 bits R OP rs rt rd sh	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break a reserved instr. b coproc. unusable c arith. overflow F: fetch instr.
jal addr jr rs jalr rs beq rt, rs, imm bne rt, rs, imm syscall lui rt, imm(rs) lbu rt, imm(rs) lhu rt, imm(rs) lw rt, imm(rs) sb rt, imm(rs) sh rt, imm(rs) st rt, imm(rs) srt, imm(rs)	Jump And Link Jump Register Jump And Link Register Branch if Equal Branch if Not Equal System Call Load Upper Imm. Load Byte Load Byte Unsigned Load Half Load Half Unsigned Store Byte Store Half Store Word Load Linked Store Conditional o-instructions Branch if Greater or Equal Branch if Less or Equal Branch if Less Than	<pre>PC = rs \$ra = PC + 8; PC = rs if (rs == rt) PC += 4 + (imm±<< 2) if (rs != rt) PC += 4 + (imm±<< 2) co_cause = 8 << 2; co_epc = PC; PC = 0x80000080 rt = imm << 16 rt = SignExt(M1[rs + imm±]) rt = M1[rs + imm±] & 0xFF rt = SignExt(M2[rs + imm±]) rt = M2[rs + imm±] & 0xFFFF rt = M4[rs + imm±] = rt M2[rs + imm±] = rt M4[rs + imm±] = rt rt = M4[rs + imm±] = rt rt = M4[rs + imm±] M4[rs + imm±] = rt; rt = atomic ? 1 : 0 6 bits 5 bits 5 bits 5 bits 5 bits R</pre>	J 3 R 0 / 8 R 0 / 9 I 4 I 5 R 0 / c I f I 20 I 24 I 21 I 25 I 23 I 28 I 29 I 2b I 30 I 38	10 exit 11 print character 12 read character 13 open file 14 read file 15 write to file 16 close file exception causes 0 interrupt 1 TLB protection 2 TLB miss L/F 3 TLB miss S 4 bad address L/F 5 bad address S 6 bus error F 7 bus error L/S 8 syscall 9 break a reserved instr. b coproc. unusable c arith. overflow

No Operation

nop