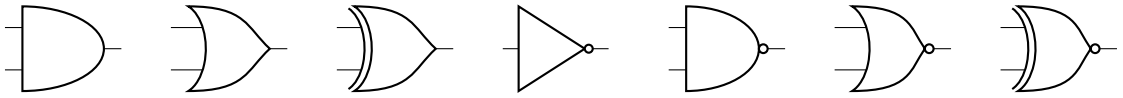
Below is a reminder of the common logic gate symbols.



Below is a reminder of the common IJVM commands.

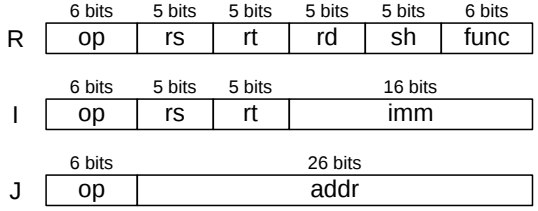
Op-code (Hex)	Assembly Language Mnemonic	Operands	Description
0x10	BIPUSH	byte	Push a byte onto stack
0x59	DUP	N/A	Copy top word on stack and push onto stack
0xA7	GOTO	label name	Unconditional jump
0x60	IADD	N/A	Pop two words from stack; push their sum
0x7E	IAND	N/A	Pop two words from stack; push Boolean AND
0x99	IFEQ	label name	Pop word from stack and branch if it is zero
0x9B	IFLT	label name	Pop word from stack and branch if it is less than zero
0x9F	IF_ICMPEQ	label name	Pop two words from stack and branch if they are equal
0x84	IINC	variable name, byte	Add a constant value to a local variable
0x15	ILOAD	variable name	Push local variable onto stack
0xB6	INVOKEVIRTUAL	method name	Invoke a method
0x80	IOR	N/A	Pop two words from stack; push Boolean OR
0xAC	IRETURN	N/A	Return from method with integer value
0x36	ISTORE	variable name	Pop word from stack and store in local variable
0x64	ISUB	N/A	Pop two words from stack; push their difference
0x13	LDC_W	constant name	Push constant from constant pool onto stack
0x00	NOP	N/A	Do nothing
0x57	POP	N/A	Delete word from top of stack
0x5F	SWAP	N/A	Swap the two top words on the stack
0xC4	WIDE	N/A	Prefix instruction; next instruction has a 16-bit index

MIPS reference card

add	rd, rs, rt	Add	rd = rs + rt	R 0 / 20
sub	rd, rs, rt	Subtract	rd = rs − rt	R 0 / 22
addi	rt, rs, imm	Add Imm.	rt = rs + $imm_{\pm}$	I 8
addu	rd, rs, rt	Add Unsigned	rd = rs + rt	R 0 / 21
subu	rd, rs, rt	Subtract Unsigned	rd = rs − rt	R 0 / 23
addiu	rt, rs, imm	Add Imm. Unsigned	rt = rs + $imm_{\pm}$	I 9
mult	rs, rt	Multiply	{hi, lo} = rs * rt	R 0 / 18
div	rs, rt	Divide	lo = rs / rt; hi = rs % rt	R 0 / 1a
multu	rs, rt	Multiply Unsigned	{hi, lo} = rs * rt	R 0 / 19
divu	rs, rt	Divide Unsigned	lo = rs / rt; hi = rs % rt	R 0 / 1b
mfhi	rd	Move From Hi	rd = hi	R 0 / 10
mflo	rd	Move From Lo	rd = lo	R 0 / 12
and	rd, rs, rt	And	rd = rs & rt	R 0 / 24
or	rd, rs, rt	Or	rd = rs | rt	R 0 / 25
nor	rd, rs, rt	Nor	rd = ~(rs | rt)	R 0 / 27
xor	rd, rs, rt	eXclusive Or	rd = rs ^ rt	R 0 / 26
andi	rt, rs, imm	And Imm.	rt = rs & imm_0	I c
ori	rt, rs, imm	Or Imm.	rt = rs | imm_0	I d
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
srl	rd, rt, sh	Shift Right Logical	rd = rt >>> sh	R 0 / 2
sra	rd, rt, sh	Shift Right Arithmetic	rd = rt >> sh	R 0 / 3
sllv	rd, rt, rs	Shift Left Logical Variable	rd = rt << rs	R 0 / 4
srlv	rd, rt, rs	Shift Right Logical Variable	rd = rt >>> rs	R 0 / 6
srav	rd, rt, rs	Shift Right Arithmetic Variable	rd = rt >> rs	R 0 / 7
slt	rd, rs, rt	Set if Less Than	rd = rs < rt ? 1 : 0	R 0 / 2a
sltu	rd, rs, rt	Set if Less Than Unsigned	rd = rs < rt ? 1 : 0	R 0 / 2b
slti	rt, rs, imm	Set if Less Than Imm.	rt = rs < $imm_{\pm}$? 1 : 0	I a
sltiu	rt, rs, imm	Set if Less Than Imm. Unsigned	rt = rs < $imm_{\pm}$? 1 : 0	I b
j	addr	Jump	PC = PC&0xF0000000 | ($addr_0$<< 2)	J 2
jal	addr	Jump And Link	$ra = PC + 8; PC = PC&0xF0000000 \| ($addr_0$<< 2)	J 3
jr	rs	Jump Register	PC = rs	R 0 / 8
jalr	rs	Jump And Link Register	$ra = PC + 8; PC = rs	R 0 / 9
beq	rt, rs, imm	Branch if Equal	if (rs == rt) PC += 4 + ($imm_{\pm}$<< 2)	I 4
bne	rt, rs, imm	Branch if Not Equal	if (rs != rt) PC += 4 + ($imm_{\pm}$<< 2)	I 5
syscall		System Call	c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080	R 0 / c
lui	rt, imm	Load Upper Imm.	rt = imm << 16	I f
lb	rt, imm(rs)	Load Byte	rt = SignExt(M_1[rs + $imm_{\pm}$])	I 20
lbu	rt, imm(rs)	Load Byte Unsigned	rt = M_1[rs + $imm_{\pm}$] & 0xFF	I 24
lh	rt, imm(rs)	Load Half	rt = SignExt(M_2[rs + $imm_{\pm}$])	I 21
lhu	rt, imm(rs)	Load Half Unsigned	rt = M_2[rs + $imm_{\pm}$] & 0xFFFF	I 25
lw	rt, imm(rs)	Load Word	rt = M_4[rs + $imm_{\pm}$]	I 23
sb	rt, imm(rs)	Store Byte	M_1[rs + $imm_{\pm}$] = rt	I 28
sh	rt, imm(rs)	Store Half	M_2[rs + $imm_{\pm}$] = rt	I 29
sw	rt, imm(rs)	Store Word	M_4[rs + $imm_{\pm}$] = rt	I 2b
ll	rt, imm(rs)	Load Linked	rt = M_4[rs + $imm_{\pm}$]	I 30
sc	rt, imm(rs)	Store Conditional	M_4[rs + $imm_{\pm}$] = rt; rt = atomic ? 1 : 0	I 38

pseudo-instructions

bge	rx, ry, imm	Branch if Greater or Equal
bgt	rx, ry, imm	Branch if Greater Than
ble	rx, ry, imm	Branch if Less or Equal
blt	rx, ry, imm	Branch if Less Than
la	rx, label	Load Address
li	rx, imm	Load Immediate
move	rx, ry	Move register
nop		No Operation



registers

$0	$zero
$1	$at
$2–$3	$v0–$v1
$4–$7	$a0–$a3
$8–$15	$t0–$t7
$16–$23	$s0–$s7
$24–$25	$t8–$t9
$26–$27	$k0–$k1
$28	$gp
$29	$sp
$30	$fp
$31	$ra
hi	—
lo	—
PC	—
co $13	c0_cause
co $14	c0_epc

syscall codes for MARS/SPIM

1 print integer
2 print float
3 print double
4 print string
5 read integer
6 read float
7 read double
8 read string
9 sbrk/alloc. mem.
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.
L: load data
S: store data