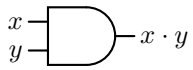
Introduction to Computer Architecture
Reference Pages

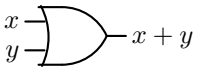
Argument	Register
first	`rdi`
second	`rsi`
third	`rdx`
fourth	`rcx`
fifth	`r8`
sixth	`r9`

AND gate



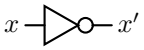
x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR gate



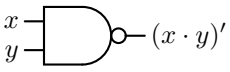
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate



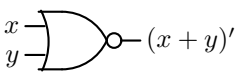
x	x'
0	0
0	1

NAND gate



x	y	$(x \cdot y)'$
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate



x	y	$(x + y)'$
0	0	1
0	1	0
1	0	0
1	1	0

arithmetic/logic:

opcode	***source***	***destination***	***action***
add*s*	$*imm*/%*reg*	%*reg*/*mem*	add
add*s*	*mem*	%*reg*	add
and*s*	$*imm*/%*reg*	%*reg*/*mem*	bit-wise and
and*s*	*mem*	%*reg*	bit-wise and
cmp*s*	$*imm*/%*reg*	%*reg*/*mem*	compare
cmp*s*	*mem*	%*reg*	compare
dec*s*	%*reg*/*mem*		decrement
div*s*	%*reg*/*mem*		unsigned divide
idiv*s*	%*reg*/*mem*		signed divide
imul*s*	%*reg*/*mem*		signed multiply
inc*s*	%*reg*/*mem*		increment
lea*w*	*mem*	%*reg*	load effective address
mul*s*	%*reg*/*mem*		unsigned multiply
neg*s*	%*reg*/*mem*		negate
or*s*	$*imm*/%*reg*	%*reg*/*mem*	bit-wise inclusive or
or*s*	*mem*	%*reg*	bit-wise inclusive or
sal*s*	$*imm*/%cl	%*reg*/*mem*	shift arithmetic left
sar*s*	$*imm*/%cl	%*reg*/*mem*	shift arithmetic right
shl*s*	$*imm*/%cl	%*reg*/*mem*	shift left
shr*s*	$*imm*/%cl	%*reg*/*mem*	shift right
sub*s*	$*imm*/%*reg*	%*reg*/*mem*	subtract
sub*s*	*mem*	%*reg*	subtract
test*s*	$*imm*/%*reg*	%*reg*/*mem*	test bits
test*s*	*mem*	%*reg*	test bits
xor*s*	$*imm*/%*reg*	%*reg*/*mem*	bit-wise exclusive or
xor*s*	*mem*	%*reg*	bit-wise exclusive or

s = b, w, l, q; *w* = l, q

conditional jumps:

opcode	***location***	***action***
ja	*label*	jump above (unsigned)
jae	*label*	jump above/equal (unsigned)
jb	*label*	jump below (unsigned)
jbe	*label*	jump below/equal (unsigned)
je	*label*	jump equal
jg	*label*	jump greater than (signed)
jge	*label*	jump greater than/equal (signed)
jl	*label*	jump less than (signed)
jle	*label*	jump less than/equal (signed)
jne	*label*	jump not equal
jno	*label*	jump no overflow