

## Repertorio de instrucciones Y86

Instrucción	Descripción	Ejemplo
nop	No operación	nop
halt	Detiene el simulador	halt
rrmovl rA, rB	Copia desde el registro rA hasta el registro rB	rrmovl %eax, %ecx rrmovl %esp, %ebp
irmovl V, rB	Mueve un valor inmediato (de 32 bits) al registro rB	irmovl \$1234, %eax
rmmovl rA D(rB)	Mueve el valor del registro rA hasta la posición de memoria dada por [rB+D]	rmmovl %eax, 8(%ebp) rmmovl %edx, 0x100(%esi)
mrmovl D(rB), rA	Mueve el contenido de la memoria dada por [rB+D] al registro rA	mrmovl 8(%ebp), %eax mrmovl 0(%ecx), %edx
addl rA, rB	Suma: $rB = rB + rA$	addl %eax, %edx
subl rA, rB	Diferencia: $rB = rB - rA$	subl %eax, %edx
andl rA, rB	Condición lógica.	andl %eax, %edx
xorl rA, rB	Or exclusivo.	xorl %eax, %eax
jmp Dest	Salto incondicional a dirección Dest (de 32 bits)	jmp 0x1000000 jmp foo
jle Dest	Salta si el resultado previo fue menor o igual a cero. based on signed arithmetic (to determine $\leq$ )	jle foo
jl Dest	Salta si el resultado previo fue menor a cero. based on signed arithmetic (to determine $<$ )	jl foo
je Dest	Salta si el resultado previo fue igual a cero.	je foo
jne Dest	Salta si el resultado previo fue distinto de cero.	jne foo
jge Dest	Salta si el resultado previo fue mayor o igual a cero. based on signed arithmetic (to determine $\geq$ )	jge foo

jg Dest	Salta si el resultado previo fue mayor a cero. based on signed arithmetic (to determine >)	jg foo
call	Inicia subrutina, guarda próxima dirección en stack.	call L1
ret	Retorno desde sub-rutina	ret
pushl rA	push rA en stack. Mem[%esp-4] = rA, %esp=%esp-4	pushl %eax
popl rA	pop de stack a rA rA = Mem[%esp], %esp = %esp+4	popl %ebp