Data transfer

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo

mov S, D
reg, reg
reg, mem
mem, reg
imd, reg
imd, mem
	D = S	---------	
			mov %rcx, %rax
			mov %rdx, i
			movw table(%rdi), %cx
			mov $stack_top, %rsp
			movq $1<<3, mask

movasb I, R			movabsq $0xFEDCBA9876543210, %r10

movsx S, R
reg8, reg16
reg8, reg32
reg8, reg64
reg16, reg32
reg16, reg64
mem8, reg16
mem8, reg32
mem8, reg64
mem16, reg32
mem16, reg64
	low(R) = low(S)
	if (high_bit(S) == 0)
	high(R) = 0
	else
	high(R) = ~0
		--------- 	
			movsbw %dl, %cx
			movsbl %al, %eax
			movsbq %dl, %rax
			movswl %ax, %eax
			movswq %dx, %rax
			movsbw char, %r10w
			movsbl char, %r10d
			movsbq (%rbx),%r10
			movswl mask, %eax
			movswq i, %r12

movslq S32, R64 **movsxd S32, R64** reg32, reg64 mem32, reg64		---------	 movslq %eax, %r15 movsxd %eax, %r15 movsxd var, %r10
movzx S, R reg8, reg16 reg8, reg32 reg8, reg64 reg16, reg32 reg16, reg64 mem8, reg16 mem8, reg32 mem8, reg64 mem16, reg32 mem16, reg64	low(S) = low(S) high(D) = 0	---------	 movsbw %dl, %cx movsbl %al, %eax movsbq %dl, %rax movswl %ax, %eax movswq %dx, %rax movsbw char, %r10w movsbl char, %r10d movsbq (%rbx),%r10 movswl mask, %eax movswq i, %r12
push S reg64 mem64 imd	rsp = rsp - 8 [rsp] = S	---------	 push %rax push (%rbx) push $0
pop D reg64 mem64	D = [rsp] rsp = rsp + 8	---------	 pop %rdx pop i
xchg D, R mem, reg reg, reg	temp = D D = S S = temp	---------	 xchg var, %rsi xchg %al, %bl
lea M, D mem, reg	Load effective address D = address(M)	---------	 lea i, %rax lea base(%rbp, %rsi, 8), %rbx
cmovXX S, R reg, reg mem, reg	Conditional move R = (XX is true) ? S : R	---------	 cmovXX %rax, %rdx cmovXX var, % ax
in port, acc imd8, acc dx, acc	input byte, word or dword AL = [port]	---------	 in $0xfa, %al in %dx, %ax
out acc, port acc, imd8 acc, dx	output byte, word or dword [port] = acc	---------	 out %ax, $0x44 out %eax, %dx

Flag Manipulation

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
lahf	AH = EFLAGS & 0x1f 7 6 4 2 0 = S Z A P C	---------	lahf
sahf	EFLAGS |= AH & 0xd5 S Z A P C = 7 6 4 2 0	----MMMMM	sahf
pushf	rsp = rsp – 8 ; [rsp] = RFLAGS	---------	pushf
popf	RFLAGS = [rsp]; rsp = rsp + 8	MMMMMMMMM	popf
clc	CF = 0	--------0	clc
cmc	CF = ~CF	--------M	cmc
stc	CF = 1	--------1	stc
cld	DF = 0	-0-------	cld
std	DF = 1	-1-------	std
cli	IF = 0	--0------	cli
sti	IF = 1 depois de executar a próxima instrução	--1------	sti
setXX dst reg8 mem8	Conditional byte set dst = XX is true	---------	 setXX %al setXX res

Arithmetic

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo

add S, D
reg, reg
mem, reg
reg, mem
imd, reg
imd, mem
	D = D + S	M---MMMMM	
			add %rcx, %rax
			add name(%rbx), %r8
			add %bl, temp
			add $1, %cl
			addq $2, alpha

adc S, D reg, reg mem, reg reg, mem imd, reg imd, mem	Adição com carry D = D + S + CF	M---MMMMM	 adc %rsi, %rax adc beta(%rsi), %rdx adc %rdi, key(%rsi) adc $256, %rbx adcq $0x30, gamma
inc D reg mem	D = D + 1	M---MMMMM	 inc %rbx incq alpha(%rdi)

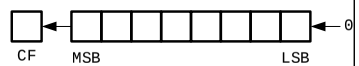
sub S, D
reg, reg
mem, reg
reg, mem
imd, reg
imd, mem
	D = D - S	M---MMMMM	
			sub %rcx, %rdx
			sub math(%rsi,%rbx,2), %r10
			sub %cl, 2(%rbx)
			sub $5280, %r14
			subq $1000, amount

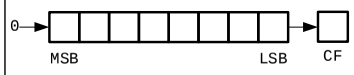
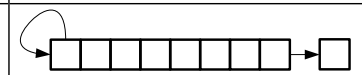
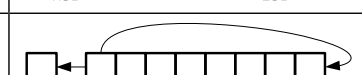
sbb S, D reg, reg mem, reg reg, mem imd, reg imd, mem	Subtracção com borrow D = D - S - CF	M---MMMMM	 sbb %r12, %r11 sbb pay(,%rsi, 4), %rdi sbb %rax, balance sbb $1, %cl sbbb $10, count(%rsi)
dec D Reg mem	D = D - 1	M---MMMMM	 dec %al decw array(%rdi)
neg D reg mem	D = -D	M---MMMMM	 neg %al negl multiplier

cmp S, D
reg, reg
mem, reg
reg, mem
imd, reg
imd, mem
	Flags modificadas de acordo com o resultado da operação D - S	M---MMMMM	
			cmp %cx, %bx
			cmp alpha, %dl
			cmp %si, 2(%rbp)
			cmp $2, %bl
			cmpq $0x3420, x(%rbx)

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
div op **idiv op** reg8 mem8 reg16 mem16 reg32 mem32 reg64 mem64	Divisão de números sem sinal Divisão de números com sinal AL = AX / byte AH = AX % byte AX = DX:AX / word DX = DX:AX % word EAX = EDX:EAX / dword EDX = EDX:EAX % dword RAX = RDX:RAX / qword RDX = RDX:RAX % qword	U---UUUUU	 div %cl divb alpha div %bx divw table(%rsi) div %ebx divl (%rsi) div %rbx divq (%rsi)
mul op reg8 mem8 reg16 mem16 reg32 mem32 reg64 mem64	Multiplicação de números sem sinal AX = AL * op (byte) DX:AX = AX * op (word) EDX:EAX = EAX * op (dword) RDX:RAX = RAX * op (qword)	M---UUUUM	 mul %bl mulb month(%rsi) mul %cx mulw baund_rate mul %ebx mull (%rsi) mul %rbx mulq (%rsi)
imul **[[op3],op2],op1** reg8 mem8 reg16 mem16 reg32 mem32 reg64 mem64 reg, reg mem, reg imd, reg imd, reg, reg imd, mem, reg	Multiplicação de números com sinal AL = AL * op1 (byte) DX:AX = AX * op1 (word) EDX:EAX = EAX * op1 (dword) RDX:RAX = RAX * op1 (qword) op1 = op1 * op2 op1 = op2 * op3	M---UUUUM	 imul %cl imulb rate imul %bx imulw red(%rbp, %rdi) imul %ebx imulw (%rsi) imul %r10 imulq (%r10) imul %rax, %rbx imul m, %r14 imul $5, %r12 imul $54, %ax, %bx imul $3, n, %r13
cbw	Estende o sinal de AL para AX	---------	cbw
cwde	Estende o sinal de AX para EAX	---------	cwde
cdqe/cltq	Estende o sinal de EAX para RAX	---------	cdqe
cwd	Estende o sinal de AX para DX:AX	---------	cwd
cdq/cltd	Estende o sinal de EAX para EDX:EAX	---------	cdq
cqo/cqto	Estende o sinal de RAX para RDX:RAX	---------	cqo

Shift

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
shld count, R, D imd, reg, reg imd, reg, mem CL, reg, reg CL, reg, mem	temp = count & 1fh value = concatenate(D, S) value = value << temp D = value	---------	 mov (%rsi), %rax shld $7, %rax, 8(%rsi)
shrd count, R, D imd, reg, reg imd, reg, mem CL, reg, reg CL, reg, mem	temp = count & 1fh value = concatenate(S, D) value = value >> temp D = value	---------	 shrd $23, %r10, %r11 shrd %cl, %r10, var
sal/shl count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 sal %cl, %rdi shl $5, %ax sal %cl, stor_cnt shlq $3, status(%rbx)

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
shr count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 shr %cl, %rsi shr $1, %si shrb %cl, input shrq $1, by(%rsi, %rbx)
sar count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 sar %cl, %di sar $1, %dx sarw %cl, n_blocks sarb $2, n_blocks
rol count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 rol %cl, %di rol $1, %bx rolq %cl, alpha rolb $2, byte(%rdi)
rcl count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 rcl %cl, %al rcl $1, %cl rclq %cl, parm(%r13) rclq $4, alpha
ror count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 ror %cl, %bx ror $2, %al rorb %cl, cmd_word rorl $2, port_stat
rcr count,D CL, reg imd8, reg CL, mem imd8, mem		M-------M	 rcr %cl, %bl rcr $10, %bx rcr %cl, array(%r14) rcrq $24, (%r12)

Logic

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
and S, D reg, reg mem, reg reg, mem imd, reg imd, mem	D = D & S (and bit a bit)	0---MMUM0	 and %al, %bl and flag_word, %rcx and %al, ascii(%rdi) and $0xf0, %cl andq $3, beta
test S, D reg, reg mem,reg reg, mem imd, reg imd, mem	Flags modificadas de acordo com a operação D & S (and bit a bit)	0---MMUM0	 test %si, %di test end_cnt, %rax testw $0xCC4, (%r15) testq $1, retcode
or S, D reg, reg mem, reg reg, mem imd, reg imd, mem	D = D | S (or bit a bit)	0---MMUM0	 or %dl, %al or prtid(%rdi), %r14 or %cl, flag_byte or $1, %cx orq $0xcf, car(%rbx)
or S,D reg, reg mem, reg reg, mem imd, reg imd, mem	D = D ^ S (xor bit a bit)	0---MMUM0	 xor %rbx, %r10 xor mask_byte, %dl xor %rdx, alpha(%rsi) xor $0xc2, %rsi xorq $0xff, retcode
not D reg mem	D = ~D (inverte bit a bit)	---------	 not %rax notw charater

String manipulation

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
rep	CX = CX - 1 Repete operação de string enquanto CX <> 0	---------	rep movsq
repe/repz	CX = CX - 1 Repete operação de string enquanto CX <> 0 && ZF == 1	---------	repe cmpsq
repne/repnz	CX = CX - 1 Repete operação de string enquanto CX <> 0 && ZF == 0	---------	repne cmpsb
movs movsb movsw movsd movsq	Move string b? n=1 : w? n=2 : d? n=4 : q? n=8 [RDI] = [RSI] if (DF == 0) {ESI += n; EDI += n} else {ESI -= n; EDI -= n}	---------	rep movsb
cmps cmpsb cmpsw cmpsd cmpsq	Compara strings b? n=1 : w? n=2 : d? n=4 : q? n=8 [RDI] - [RSI] if (DF == 0) {RSI += n; RDI += n} else {RSI -= n; RDI -= n}	M---MMMMM	rep cmpsb
scas scasb scasw scasd scasq	Scan string scasb n = 1; scasw n = 2; scasd n = 4; scasq n = 8 al, ax, eax or rax - [RDI] if (DF == 0) RDI += n; else RDI -= n	M---MMMMM	repne scasq
lods lodsb lodsw lodsd lodsq	Load string b? n=1 : w? n=2 : d? n=4 : q? n=8 al, ax, eax or rax = [RSI] if (DF == 0) RSI += n; else RSI -= n	---------	rep lods
stos stosb stosw stosd stosq	Store string stosb n = 1; stosw n = 2; stosd n = 4; stosq n = 8 ES:[EDI] = al, ax, eax or rax if (DF == 0) EDI += n; else EDI -= n	---------	rep stos
ins insb insw insd insq	Input string from I/O port b? n=1 : w? n=2 : d? n=4 : q? n=8 [RDI] = port(DX) if (DF == 0) RDI += n; else RDI -= n	---------	rep insb
outs outsb outsw outsd outsq	Output string to I/O port b? n=1 : w? n=2 : d? n=4 : q? n=8 port(DX) = [RSI] if (DF == 0) RSI += n; else RSI -= n	---------	rep outsb
xlat xlatb	AL = [EBX + AL]	---------	xlatb

Bit manipulation

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo

bsf target, index
reg, reg
mem, reg
	Scan bit forward
	for(i = 0; target[i] == 0 && i <= 15(31)(63); i++);
	index = i;
		U---UMUUU	bsf %rcx, %rax
			bsf var, %ax

bsr target, index
reg, reg
mem, reg
	Scan bit reverse
	for(i = 15(31)(63); target[i] == 0 && i >= 0; i--);
	index = i;
		U---UMUUU	bsr %rcx, %rax
			bsr var, %ax

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
bt index, target imd8, reg imd8, mem reg, reg reg, mem	Test bit CF = target[index]	U---UUUUM	
btc index, target imd8, reg imd8, mem reg, reg reg, mem	Test bit and complement CF = target[index] target[index] = ~ target[index]	U---UUUUM	
btr index, target imd8, reg imd8, mem reg, reg reg, mem	Test bit and reset CF = target[index] target[index] = 0	U---UUUUM	
bts index, target imd8, reg imd8, mem reg, reg reg, mem	Test bit and set CF = target[index] target[index] = 1	U---UUUUM	

Control transfer

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
jmp target label reg mem	RIP += offset8(16)(32) RIP = reg RIP = [mem]	---------	jmp .L1 jmp *%rbx jmp *switch(%rsi)
call target label reg mem	push RIP; RIP += offset16(32) push RIP; RIP = reg push RIP; RIP = [mem]	---------	call strcpy call *%rbx call *table(%rsi)
ret [count]	pop RIP pop RIP; RSP = RSP + count	---------	ret ret $4
jXX disp disp8 disp64	if (XX is true) RIP += disp	---------	jXX label
jcxz disp disp8	Jump if CX is zero if (CX == 0) RIP += disp	---------	jcxz count_done
jecxz disp disp8	Jump if ECX is zero if (ECX == 0) RIP += disp	---------	jecxz count_done
jrcxz disp disp8	Jump if RCX is zero if (RCX == 0) RIP += disp	---------	jrcxz count_done
loop disp disp8	RCX = RCX - 1; if (RCX != 0) RIP += disp	---------	loop again
loope/loopz disp disp8	RCX = RCX - 1; if (RCX != 0 && ZF == 1) RIP += disp	---------	loope again
loopne/loopnz disp disp8	rCX = RCX - 1; If (RCX != 0 && ZF == 0) RIP += disp	---------	loopne again

enter level, size
imd8, imd16
	level = level & 0x1f
	push rbp
	temp = rsp
	if (level > 0) {
	for (i = 1; i < level; i++) {
	rbp = rbp – 8
	push [rbp]
	}
	push temp
	}
	rbp = temp
	esp = esp – size
		--------- 		

Sintaxe	Descrição	Flags ODITSZAPC	Exemplo
leave	mov RSP, RBP pop RBP	---------	

Condições

Mnemónica	Descrição	Condição
g / nle	greater / not less or equal (operandos com sinal)	CF == OF && ZF == 0
ge / nl	greater or equal / not less (operandos com sinal)	CF == OF
l / nge	less / not greater nor equal (operandos com sinal)	CF != OF
le / ng	less or equal / not greater (operandos com sinal)	CF != OF || ZF == 1
a / nbe	above / not below nor equal (operandos sem sinal)	CF == 0 && ZF == 0
ae / nb	above or equal / not below (operandos sem sinal)	CF == 0
b / nae	below / not above nor equal (operandos sem sinal)	CF == 1
be / na	below or equal / not above (operandos sem sinal)	CF == 1 || ZF == 1
p / pe	parity / parity even	PF == 1
np / po	not parity / parity odd	PF == 0
o	overflow	OF == 1
no	not overflow	OF == 0
s	sign	SF == 1
ns	not sign	SF == 0
e / z	equal / zero	ZF == 1
ne / nz	not equal / not zero	ZF == 0
c	carry	CF == 1
nc	not carry	CF == 0

Formato do registo EFlags