AMD64 SYSTEM V ABI CALLING CONVENTION

- First 6 integer/pointer parameters: **RDI, RSI, RDX, RCX, R8, R9**
- First 8 floating point parameters: **XMM0…XMM7**
- Further parameters: **through the stack, in reversed order**
- Function has to preserve: **RBX, RBP, RSP, R12…R15**
- Integer/pointer return value: **RAX**
- Floating point return value: **XMM0**
- Variadic functions: number of floating point parameters in **AL**
- Before calling a function, **RSP** has to be 16-byte aligned

SYSTEM CALLS

0	sys_read	Reads from a file_descriptor. Number of bytes read is returned (ENTER included). Parameters: file_descriptor, buffer_pointer, buffer_size
1	sys_write	Writes into a file_descriptor. Number of bytes written is returned. Parameters: file_descriptor, buffer_pointer, buffer_size
60	sys_exit	Terminates the process. Parameters: error_code

INTEGER ARITHMETIC INSTRUCTIONS

MUL	op	op: reg8, mem8, reg16, mem16, reg32, mem32, reg64, mem64	
	8 bit	AX = AL ∗ op	
	16 bit	DX:AX = AX ∗ op	
	32 bit	EDX:EAX = EAX ∗ op	
	64 bit	RDX:RAX = RAX ∗ op	
IMUL	op	See MUL above	
IMUL	op1, op2	op1 = op1 * op2	
IMUL	op1, op2, op3	op1 = op2 * op3	
DIV	op	op: reg8, mem8, reg16, mem16, reg32, mem32, reg64, mem64	
	8 bit	AL = AX / op	AH = remainder
	16 bit	AX = DX:AX / op	DX = remainder
	32 bit	EAX = EDX:EAX / op	EDX = remainder
	64 bit	RAX = RDX:RAX / op	RDX = remainder
IDIV	op	See DIV above	

FLOAT DATA MOVEMENT INSTRUCTIONS

movsd	op1, op2	op2 = op1	op1, op2: xmm, mem64
movd	op1, op2	op2 = op1	op1, op2: xmm, reg32, mem32
movq	op1, op2	op2 = op1	op1, op2: xmm, reg64, mem64

FLOAT ARITHMETIC INSTRUCTIONS

addsd	op1, op2	op1 = op1 + op2	op1, op2: xmm, mem64
subsd	op1, op2	op1 = op1 - op2	op1, op2: xmm, mem64
mulsd	op1, op2	op1 = op1 * op2	op1, op2: xmm, mem64
divsd	op1, op2	op1 = op1 / op2	op1, op2: xmm, mem64
sqrtsd	op1, op2	op1 = $\sqrt{op2}$	op1, op2: xmm, mem64

SSE CONVERSION INSTRUCTIONS

cvtsi2sd	op1, op2	op1: xmm	op2: reg32/64, mem32/64
	Converts 32/64 bit integer to double		