	Normal return	Error return	Error propagation	Error catching

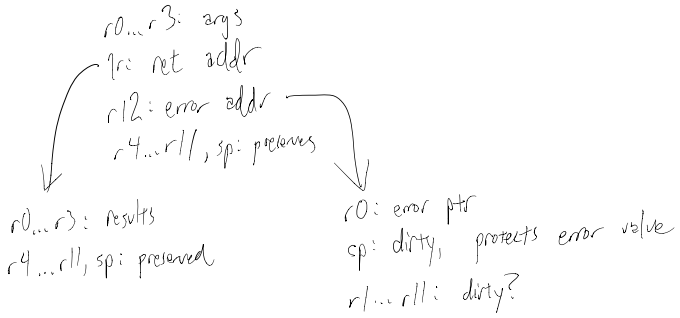
Result	returns_result_reg:
	mov rax, rValue
	xor edx, edx
	ret
	
	returns_result_mem32:
	mov rax, rSRet
	<store result in [rax]>
	movb [rax+32], 0
	ret
		returns_result_reg:
		mov rax, rError
		mov edx, 1
		ret
		
		returns_result_mem32:
		mov rax rSRet
		mov [rax], rError
		movb [rax+32], 0
		ret
			returns_result_reg:
			call returns_result_mem32
			cmpb [rax+32], 0
			jne propagate
			...success...
			propagate:
			mov rax, [rax]
			mov edx, 1
			ret
			
			returns_result_mem32:
			call returns_result_reg
			test rdx, rdx
			jne propagate
			...success...
			propagate:
			mov [rSRet], rax
			movb [rSRet+32], 1
			mov rax, rSRet
			ret
				call returns_result_mem32
				cmpb [rax+32], 0
				jne catch
				...success...
				catch:
				mov rax, [rax]
				...handle error in rax...
				
				call returns_result_reg
				test rdx, rdx
				jne catch
				...success...
				catch:
				...handle error in rax...

Swift4	mov rax, rValue ret	mov rErrorRet, rError ret	call returns_result test rErrorRet, rErrorRet jne propagate ...success... propagate: ret	call returns_result test rErrorRet, rErrorRet jne catch ...success... catch: ...handle error in rE...
CPS	mov rax, rValue ret	mov rax, rError jmp rErrorDest	mov rErrorDest, rSavedErrorDest call returns_result Different register conventions for return and error jmp?	mov rErrorDest, [rip+catch] call returns_result ...success... catch:

JSError	mov rax, rValue
	ret
		xor rax, rax
		mov [rErrorOut], rError
		ret
			mov rErrorOut', rErrorOut
			call returns_result
			test rax, rax
			je propagate
			...success...
			propagate:
			ret ; rax already >= 0;
			[rErrorOut] already has error
				lea rErrorOut, [rsp+n]
				call returns_result
				test rax, rax
				je catch
				...success...
				catch:
				mov rError, [rsp+n]
				...handle...

Swift3	mov rax, rValue
	ret
		mov [rErrorOut], 0
		ret
			mov rErrorOut', rErrorOut
			call returns_result
			cmp [rErrorOut], 0
			je propagate
				lea rErrorOut, [rsp+n]
				call returns_result
				cmp [rsp+n], 0
				je catch

		...success... propagate: ret	...success... catch: mov rError, [rsp+n] ...handle...
HRESULT mov [rOut], rValue xor eax, eax ret	mov rax, rError ret	lea rOut, [rsp+n] call returns_result test rax, rax jne propagate ...success... propagate: ret	lea rOut, [rsp+n] call returns_result test rax, rax jne catch ...success... catch: ...handle error in rax...


r0...r3: args
lr: ret addr
r12: error addr
r4...r11, sp: preserves
r0...r3): results
r4...r11, sp: preserved
r0: error ptr
cp: dirty, protects error value
r1...r11: dirty?