Instruction	Stack before	Stack after
`Const(c)`	s	$c :: s$
`Add`	$c_2 :: c_1 :: s$	$(c_1 + c_2) :: s$
`Sub`	$c_2 :: c_1 :: s$	$(c_1 - c_2) :: s$
`Mul`	$c_2 :: c_1 :: s$	$(c_1 \cdot c_2) :: s$
`Div`	$c_2 :: c_1 :: s$	$(c_1 / c_2) :: s$
`Neg`	$c :: s$	$-c :: s$

Instruction	Operand stack before	Operand stack after
`Eq`	$c_2 :: c_1 :: s$	$1 :: s$ if $c_1 = c_2$, $0 :: s$ otherwise
`Lt`	$c_2 :: c_1 :: s$	$1 :: s$ if $c_1 < c_2$, $0 :: s$ otherwise
`Leq`	$c_2 :: c_1 :: s$	$1 :: s$ if $c_1 \leq c_2$, $0 :: s$ otherwise
`And`	$c_2 :: c_1 :: s$	$1 :: s$ if $c_1 = 1$ and $c_2 = 1$, $0 :: s$ otherwise
`Or`	$c_2 :: c_1 :: s$	$1 :: s$ if $c_1 = 1$ or $c_2 = 1$, $0 :: s$ otherwise
`Not`	$c :: s$	$1 :: s$ if $c = 0$, $0 :: s$ otherwise

Instructions before	Operand stack before	(Instructions, Operand stack) after
`Branch(thencode, elsecode)` :: morecode	$c :: s$	(thencode ++ morecode, s) if $c = 1$, (elsecode ++ morecode, s) otherwise

Instruction	Operand stack before	Environment stack before	Operand stack after	Environment stack after
`Enter`	$c :: s$	e	s	$c :: e$
`Read(index)`	s	$c_0 :: c_1 \ldots$	$c_{index} :: s$	$c_0 :: c_1 \ldots$
`Exit(num)`	s	$c_1 :: c_2 \ldots :: c_{num} :: e$	s	e

Instruction	Operand stack before	Store before	Operand stack after	Store after
`Alloc`	s	σ where $\ell \notin \text{dom}(\sigma)$	ℓ :: s	$\sigma[\ell \mapsto 0]$
`Load`	ℓ :: s	σ	$\sigma(\ell)$ :: s	σ
`Store`	c :: ℓ :: s	σ	s	$\sigma[\ell \mapsto c]$
`Dup`	c :: s	σ	c :: c :: s	σ
`Pop`	c :: s	σ	s	σ
`Unit`	s	σ	0 :: s	σ

Instructions before	Instructions after
`Loop(`condcode`,` bodycode`)` :: morecode	condcode ++ `Branch(`bodycode ++ `List(Loop(`condcode`,` bodycode`)),` `Nil)` :: morecode

Instruction	Operand stack before	Environment stack	Operand stack after
`Lambda(freeids, body)`	s	e	(body, env) :: s

Instruction	Operand stack before	Environment stack before	Call stack before	Operand stack after	Environment stack after	Call stack after
`Call(arity)`	c_{arity} :: … :: c_1 :: (body, env) :: s	e	cs	Nil	(see below)	F :: cs

Instruction	Operand stack before	Environment stack before	Call stack before	Operand stack after	Environment stack after	Call stack after
`Return`	c :: s	e	F :: cs	c :: s’	e’	cs

Instruction	Operand stack before	Environment stack before	Operand stack after	Environment stack after
`EnterDefs(num)`	$(body_{num}, env_{num})$ :: … :: $(body_1, env_1)$ :: s	e	s	$(body_{num}, env_{num}')$ :: … :: $(body_1, env_1')$ :: e