r_n – register
i_n – immediate
l_n – label
s_n – id
cc – condition code register

Arithmetic

add	$r_1, r_2 \Rightarrow r_3$	$r_1 + r_2 \Rightarrow r_3$
addi	$r_1, i_1 \Rightarrow r_2$	$r_1 + i_1 \Rightarrow r_2$
div	$r_1, r_2 \Rightarrow r_3$	$r_1 \,/\, r_2 \Rightarrow r_3$
mult	$r_1, r_2 \Rightarrow r_3$	$r_1 * r_2 \Rightarrow r_3$
sub	$r_1, r_2 \Rightarrow r_3$	$r_1 - r_2 \Rightarrow r_3$
rsubi	$r_1, i_1 \Rightarrow r_2$	$i_1 - r_1 \Rightarrow r_2$

Boolean

and	$r_1, r_2 \Rightarrow r_3$	$r_1 \wedge r_2 \Rightarrow r_3$
or	$r_1, r_2 \Rightarrow r_3$	$r_1 \vee i_1 \Rightarrow r_2$
xori	$r_1, i_1 \Rightarrow r_2$	$r_1 \otimes i_1 \Rightarrow r_2$

Comparison and Branching

comp	$r_1, r_2 \Rightarrow$ cc	set cc
compi	$r_1, i_1 \Rightarrow$ cc	set cc
cbreq	cc, l_1, l_2	cc == EQ $\Rightarrow l_1 \rightarrow$ PC otherwise $l_2 \rightarrow$ PC
cbrge	cc, l_1, l_2	cc == GE $\Rightarrow l_1 \rightarrow$ PC
cbrgt	cc, l_1, l_2	cc == GT $\Rightarrow l_1 \rightarrow$ PC
cbrle	cc, l_1, l_2	cc == LE $\Rightarrow l_1 \rightarrow$ PC
cbrlt	cc, l_1, l_2	cc == LT $\Rightarrow l_1 \rightarrow$ PC
cbrne	cc, l_1, l_2	cc == NE $\Rightarrow l_1 \rightarrow$ PC
jumpi	l_1	$l_1 \rightarrow$ PC

Loads

loadi	$i_1 \Rightarrow r_1$	$i_1 \rightarrow r_1$
loadai	$r_1, i_1 \Rightarrow r_2$	MEMORY($r_1 + i_1$) $\rightarrow r_2$
loadglobal	$s_1 \Rightarrow r_1$	MEMORY(@s_1) $\rightarrow r_1$
loadinargument	$s_1, i_1 \Rightarrow r_1$	MEMORY(@arg(i_1)) $\rightarrow r_1$
loadret	$\Rightarrow r_1$	MEMORY(ret) $\rightarrow r_1$
computeformaladdress	$s_1, i_1 \Rightarrow r_1$	@arg(i_1) $\rightarrow r_1$
restoreformal	s_1, i_1	flag that the value at a formal's actual location has changed
computeglobaladdress	$s_1 \Rightarrow r_1$	@$s_1 \rightarrow r_1$

Stores

storeai	$r_1 \Rightarrow r_2, i_1$	$r_1 \rightarrow \text{MEMORY}(r_2 + i_1)$
storeglobal	$r_1 \Rightarrow s_1$	$r_1 \rightarrow \text{MEMORY}(@s_1)$
storeinargument	$r_1 \Rightarrow s_1, i_1$	$r_1 \rightarrow \text{MEMORY}(@\text{arg}(i_1))$
storeoutargument	$r_1 \Rightarrow i_1$	$r_1 \rightarrow \text{MEMORY}(@\text{outarg}(i_1))$
storeret	r_1	$r_1 \rightarrow \text{MEMORY}(\text{ret})$

Invocation

call	l_1	$r_1 \rightarrow \text{PC} \rightarrow \text{retaddr}, l_1 \rightarrow \text{PC}$
ret		$\text{retaddr} \rightarrow \text{PC}$

Allocation

new	$i_1 \Rightarrow r_1$	allocate and store address in r_1
del	r_1	deallocate memory at address r_1

I/O

print	r_1	output integer in r_1
println	r_1	output integer in r_1
read	r_1	store integer at address in r_1 ($\rightarrow \text{MEMORY}(r_1)$)

Moves

mov	r1, r2	r1 $=\rangle$ r2
moveq	i1, r1	if CC == EQ then i1 $=\rangle$ r1
movge	i1, r1	if CC == GE then i1 $=\rangle$ r1
movgt	i1, r1	if CC == GT then i1 $=\rangle$ r1
movle	i1, r1	if CC == LE then i1 $=\rangle$ r1
movlt	i1, r1	if CC == LT then i1 $=\rangle$ r1
movne	i1, r1	if CC == NE then i1 $=\rangle$ r1