Instruction	Functionality	Encoding Forma	MC Example	asm example
set Rg	Rg = result	0000 00 gg	0000 0000	set $0
add1 Rg	Rg = Rg + 1	0000 01 gg	0000 0100	add1 $0
sub1 Rg	Rg = Rg - 1	0000 10 gg	0000 1000	sub1 $0

seqc gg	if mem(4 +gg) == mem(8 +gg)
	status = 1;
	mem(gg) = 0
	else:
	status = 0;
	mem(gg) += 1;
		0000 11 gg	0000 1100	seqc 0

add1c gg	mem(8+gg) += 1	0001 01 gg	0001 0100	add1c 0
swf Rg	mem(gg) = $0	0001 10gg	0001 1000	swf 0
lwf Rg	$0 = mem(gg)	0001 11gg	0001 1100	lwf 0

sneg Rg	if Rg < 0:
	status = 1
	else:
	status = 0
		0010 00gg	0010 0010	sneg $2

snotneg Rg	if Rg > 0:
	status = 1
	else:
	status = 0
		0010 01gg	0010 0110	snotneg $2

lwfaddreg	$0 = $0 + mem(gg)	0010 10gg	0010 1000	lwfaddreg 0
lwfsubreg	$0 = $0 - mem(gg)	0010 11gg	0010 1100	lwfsubreg 0
lwmult	$0 = $0 * mem(Rg);	0011 00gg	0011 0010	lwmultinc $2
lwprev	$0 = mem(Rg-1)	0011 01gg	0011 0110	lwprev $2
lwmult	$0 = $0 * mem(Rg)	0011 10gg	0011 1010	lwmult $2
hold	result = $0	0011 1100	0011 1100	hold
addback	$0 = result	0011 1101	0011 1101	addback
add Ra, Rb	Ra = Ra + Rb	0100 aabb	0100 0001	add $0, $1

swinc Ra, R	mem(Ra) = Rb;
	Ra++;
		0101 aabb	0101 0110	swinc $2, $1

init aabb	init = init | (aabb << 4 * counter)
	if counter == 3:
	result = temp;
	init = 0;
	counter = 0;
	else:
	counter++;
		0110 aabb	0110 1111	init F

loop imm	LUT[imm] = pc + 4	100j jjjj	1000 0000	loop:

jif imm	if (status):
	pc = LUT[imm];
		110j jjjj	1010 0000	jif loop

j imm	pc = LUT[imm]	111j jjjj	1100 0000	j loop

Instruction	Op	InputControl[2:0]	tputAControl[2:0]	tputBControl[2:0]	RegWrite	MemRegF	FlagControl	SetStatus	IncCounter	ALUControl	yALUControl[2:0]	MemWrite	BypassMem	MemInc
set Rg	0000 00XX	10	10	xx	1	0	XX	0	1	0000	1111	0	1	X
add1 Rg	0000 01XX	10	10	xx	1	0	XX	0	0	0100	1111	0	1	X
sub1 Rg	0000 10XX	10	10	XX	1	0	XX	0	0	0101	1111	0	1	X
seqc gg	0000 11XX	XX	XX	XX	0	1	01	1	0	0000	XXXX	0	1	0
add1c gg	0001 01XX	XX	XX	XX	0	1	XX	0	0	1000	XXXX	1	X	1
swf Rg	0001 10XX	XX	XX	XX	0	1	XX	0	0	0000	XXXX	1	X	0
lwf Rg	0001 11XX	00	XX	XX	1	0	XX	0	0	0000	1111	0	0	X
sneg Rg	0010 00XX	XX	10	XX	0	0	10	1	0	1001	XXXX	0	X	X
snotneg Rg	0010 01XX	XX	10	XX	0	0	11	1	0	1001	XXXX	0	X	X
lwfaddreg	0010 10XX	00	XX	00	1	1	XX	0	0	0000	0001	0	0	X
lwfsubreg	0010 11XX	00	XX	00	1	1	XX	0	0	0000	0010	0	0	X
lwmultinc	0011 00XX	00	10	00	1	0	XX	0	0	0000	0011	0	0	X
lwprev	0011 01XX	00	10	00	1	0	XX	0	0	0000	0101	0	0	X
lwmult	0011 10XX	00	10	00	1	0	XX	0	0	0000	0011	0	0	X
hold	0011 1100	11	00	XX	1	X	XX	0	0	0000	1111	0	1	X
addback	0011 1101	00	11	XX	1	X	XX	0	0	0000	1111	0	1	X
add Ra, Rb	0100 XXXX	01	01	01	1	0	XX	0	0	0001	1111	0	1	X
swinc Ra, Rb	0101 XXXX	10	01	01	1	0	XX	0	0	0000	1110	1	1	0
init aabb	0110 XXXX	XX	XX	XX		X	XX	0	1	XXXX	XXXX	0	x	X
loop imm	100X XXXX	XX	XX	xx		X	XX	0	0	XXXX	XXXX	0	X	X
jif imm	110X XXXX	XX	XX	xx		X	XX	0	0	XXXX	XXXX	0	X	X
j imm	1100 0000	XX	XX	xx		X	XX	0	0	XXXX	XXXX	0	X	X

Registers		**InputControl[2:0]**		**OutputControl[2:0]**		**ALUControl[3:0]**	
Name	Number	**Encoding**	**Meaning**	**Encoding**	**Meaning**	A	0000
$0	0	00	$0	00	$0	A+B	0001
$1	1	01	$Ra	01	$Rb	A-B	0010
$2	2	10	$Rg	10	$Rg	A*B	0011
$3	3	11	result	11	result	A+1	0100
result	4					A-1	0101
						A<<B	0110
						B+1	1110
						A+8	1000
						0-A	1001
						B	1111

	InputControl[2:0]		**OutputControl[2:0]**	
	Encoding	**Meaning**	**Encoding**	**Meaning**
	00	$0	00	$0
	01	$Ra	01	$Rb
	10	$Rg	10	$Rg
	11	result	11	result