Type	Operation	Format	Function
ALU	add	add ra rb rd	rd = ra + rb
	sub	sub ra rb rd	rd = ra - rb
	lsh	lsh ra rb rd	rd = ra << rb
	rsh	rsh ra rb rd	rd = ra >> rb
	and	and ra rb rd	rd = ra & rb
	or	or ra rb rd	rd = ra | rb
	xor	xor ra rb rd	rd = ra ^ rb
	neg	neg rd	rd = ~rd, rd = ~ra
	addi	addi ra imm rd	rd = ra + imm(x)
	nop	nop	r0 = r0 + 0
	inc	inc rd	rd = rd + imm(1)
	subi	subi ra imm rd	rd = ra - imm(x)
	dec	dec rd	rd = rd - imm(1)
	lshi	lshi ra imm rd	rd = ra << imm
	rshi	rshi ra imm rd	rd = ra >> imm
	not	not rd	rd = !rd, rd = !ra
Branch	beq	beq ra rb tgt [pt]	pc = tgt if ra == rb
	bi	bi tgt	pc = tgt
	bne	bne ra rb tgt [pt]	pc = tgt if ra != rb
	blt	blt ra rb tgt [pt]	pc = tgt if ra < rb
	bgt	bgt ra rb tgt [pt]	pc = tgt if ra > rb
	ble	ble ra rb tgt [pt]	pc = tgt if ra <= rb
	bge	bge ra rb tgt [pt]	pc = tgt if ra >= rb
	bs	bs rspc rb tgt [pt]	pc = tgt if rspc == rb
	bss	bss rspc_a rspc_b tgt [pt]	pc = tgt if rspc_a == rspc_b
	beqi	beqi ra imm tgt [pt]	pc = tgt if ra == imm
	bz	bz ra tgt [pt]	pc = tgt if ra == 0
	bneqi	bneqi ra imm tgt [pt]	pc = tgt if ra != imm
	bnz	bnz ra tgt [pt]	pc = tgt if ra != 0
	bsi	bsi rspc imm tgt [pt]	pc = tgt if rspc == imm
Data	mov	mov ra rd	rd = ra
	movsg	mov rspc rd	rd = rspc
	movgs	mov ra rspc	rspc = ra
	ldflags	ldflags ra	MSHR.flags = ra[0+:num_flags]; dst = e_opd_flags
	movfg	mov flag rd	rd[0] = flag
	movgf	mov ra flag	flag = ra[0]
	movpg	mov param gpr	gpr = param
	movgp	mov gpr param	param = gpr
	movi	movi imm rd	rd = imm
	movis	movi imm rspc	rspc = imm
	ldflagsi	ldflagsi imm	MSHR.flags = imm[0+:num_flags]
	clf	clf	clear MSHR.flags; dst = e_opd_flags
	movip	movip imm param	param = imm
	clm	clm	clear MSHR
Flag	sf	sf flag	rdflag = 1
	sfz	sfz flag	rdflag = 0
	andf	andf flag flag rd	rd = raflag & rbflag
	orf	orf flag flag rd	rd = raflag | rbflag
	nandf	nandf flag flag rd	rd = !(raflag & rbflag)
	norf	norf flag flag rd	rd = !(raflag | rbflag)
	notf	notf flag rd	rd = ~flag
	bf	bf tgt flag [flag…] [pt]	pc = tgt if all flags 1
	bfz	bfz tgt flag [flag…] [pt]	pc = tgt if all flags 0
	bfnz	bfnz tgt flag [flag…] [pt]	pc = tgt if any flag 1
	bfnot	bfnot tgt flag [flag…] [pt]	pc = tgt if any flag not 1
Directory	rdp	rdp addr=<a>	pf = pending_bits[addr];
	rdw	rdw addr=<a> lce=<l> lru_way=<w> [src=<ra>]	produce sharers, lru info, etc.
	rde	rde addr=<a> lce=<l> way=<w> [src=<ra>] dst=<rd>	rd = addr, sharers_states[lce] = state
	wdp	wdp addr=<a> p=<0,1>	pending_bits[addr] +/- 1
	clp	clp addr=<a>	pending_bits[addr] = 0
	clr	clr addr=<a> lce=<l>	clears physical directory row for [addr, lce] pair
	wde	wde addr=<a> lce=<l> way=<w> [src=<ra>] state=<s> [state_imm]	dir[addr, lce] = [tag, state]
	wds	wds addr=<a> lce=<l> way=<w> [src=<ra>] state=<s> [state_imm]	dir[addr, lce] = […, state]

	gad	gad	execute GAD unit
Queue	wfq	wfq queue [queue…]	wait for one or more input queues to have valid data
	pushq	pushq queue cmd addr=<a> lce=<l> way=<w> [src=<ra>] wp=<0,1> spec=<0,1>	push BedRock message on outbound queue
	popq	popq queue [wp]	dequeue message from queue, optionally write pending bit
	poph	poph queue rd	capture fields from message header to CCE state
	popd	popd queue rd	capture 64-bits data from message to rd
	specq	specq spec_cmd addr_sel [state]	speculation bits operation
	inv	inv	send INV cmd to all LCE's with block (MSHR.paddr) in S)

Notes

The Format column describes what the programmer should write in the microcode assembly source file. Arguments surrounded by brackets, e.g., [arg], are optional.