Processor State

PC⟨31..0⟩:	Program counter (address of next instruction)
IR⟨31..0⟩:	Instruction register
Run:	1-bit run/halt indicator
Strt:	Start signal
R[0..31]⟨31..0⟩:	General purpose registers

Main Memory State

Mem[0..2^{32} – 1]⟨7..0⟩: 2^{32} addressable bytes of memory
M[x]⟨31..0⟩ := Mem[x]#Mem[x+1]#Mem[x+2]#Mem[x+3]:

Instruction Formats

op⟨4..0⟩ := IR⟨31..27⟩:	Operation code field
ra⟨4..0⟩ := IR⟨26..22⟩:	Target register field
rb⟨4..0⟩ := IR⟨21..17⟩:	Operand, address index, or branch target
rc⟨4..0⟩ := IR⟨16..12⟩:	Second operand, condition, or shift count
c1⟨21..0⟩ := IR⟨21..0⟩:	Long displacement field
c2⟨16..0⟩ := IR⟨16..0⟩:	Short displacement or immediate field
c3⟨11..0⟩ := IR⟨11..0⟩:	Count or modifier field

Effective Address Calculations

disp⟨31..0⟩ := ((rb=0) → c2⟨16..0⟩ {sign extend}:
Displacement
(rb≠0) → R[rb] + c2⟨16..0⟩ {sign ext., 2's comp.}): address
rel⟨31..0⟩ := PC⟨31..0⟩ + c1⟨21..0⟩ {sign ext., 2's comp.}: Rel. address

(instruction_interpretation := (
¬Run∧Strt → Run ← 1; instruction_interpretation):
Run → (IR ← M[PC]: PC ← PC + 4; instruction_execution):

instruction_execution := (

ld (:= op= 1) → R[ra] ← M[disp]:	ld ra, offset(rb)	Load register
ldr (:= op= 2) → R[ra] ← M[rel]:	ldr ra, offset	Load reg. rel.
st (:= op= 3) → M[disp] ← R[ra]:	st ra, offset(rb)	Store register
str (:= op= 4) → M[rel] ← R[ra]:	str ra, offset(rb)	Store reg. rel.
la (:= op= 5) → R[ra] ← disp:	la ra, offset(rb)	Load disp. adr.
lar (:= op= 6) → R[ra] ← rel:	lar ra, offset	Load rel. adr.

Branch Instructions

cond := (c3⟨2..0⟩=0 → 0:	br(l)nv (ra)	Never
c3⟨2..0⟩=1 → 1:	br(l) (ra,) rb	Always
c3⟨2..0⟩=2 → R[rc]=0:	brzr(l) (ra,) rb, rc	R[rc] = 0
c3⟨2..0⟩=3 → R[rc]≠0:	brnz(l) (ra,) rb, rc	R[rc] ≠ 0
c3⟨2..0⟩=4 → R[rc]⟨31⟩=0:	brpl(l) (ra,) rb, rc	R[rc] ≥ 0
c3⟨2..0⟩=5 → R[rc]⟨31⟩=1):	brmi(l) (ra,) rb, rc	R[rc] < 0
br (:= op= 8) → (cond → PC ← R[rb]):		Conditional branch
brl (:= op= 9) → (R[ra] ← PC: cond → (PC ← R[rb])):		Branch and link

Arithmetic Instructions (Assumed to Be 2's Complement Arithmetic)

add (:= op= 12) → R[ra] ← R[rb] + R[rc]:	add ra, rb, rc
addi (:= op= 13) → R[ra] ← R[rb] + c2⟨16..0⟩ {2's comp. sign ext.}:	addi ra, rb, imm
sub (:= op= 14) → R[ra] ← R[rb] – R[rc]:	
neg (:= op= 15) → R[ra] ← – R[rc]:	neg ra, rc

and (:= op= 20) → R[ra] ← R[rb] ∧ R[rc]:
andi (:= op= 21) → R[ra] ← R[rb] ∧ c2⟨16..0⟩ {sign-extend}:
or (:= op= 22) → R[ra] ← R[rb] ∨ R[rc]:
ori (:= op= 23) → R[ra] ← R[rb] ∨ c2⟨16..0⟩ {sign-extend}:
not (:= op= 24) → R[ra] ← ¬R[rc]:

Shift Instructions

n := ((c3⟨4..0⟩=0) → R[rc]⟨4..0⟩: Shift count in a register or
(c3⟨4..0⟩≠0) → c3⟨4..0⟩): constant field of the inst.
shr (:= op= 26)→R[ra]⟨31..0⟩ ← (n @ 0) # R[rb]⟨31..n⟩:.
shr ra, rb, cnt; shr ra, rb, rc
shra (:= op= 27) → R[ra]⟨31..0⟩ ← (n @ R[rb]⟨31⟩) # R[rb]⟨31..n⟩:
shl (:= op= 28) → R[ra]⟨31..0⟩ ← R[rb]⟨31–n..0⟩ # (n @ 0):
shc (:= op= 29) → R[ra]⟨31..0⟩ ← R[rb]⟨31–n..0⟩ # R[rb]⟨31..32 – n⟩:

Miscellaneous Instructions

nop (:= op= 0) → :	No operation
stop (:= op= 31) → Run ← 0):	Stop instruction
instruction_interpretation):	End of instruction_execution