

RISC-V Reference Guide (CREATOR Simulator)

System Calls (ecall)			
Service	Call Code (a7)	Arguments	Result
Print_int	1	a0 = integer	
Print_float	2	fa0 = float	
Print_double	3	fa0 = double	
Print_string	4	a0 = string addr	
Read_int	5		Integer in a0
Read_float	6		Float in fa0
Read_double	7		Double in fa0
Read_string	8	a0 = string addr a1 = length	
Sbrk	9	a0 = length	Address in a0
Exit	10		
Print_char	11	a0 = ASCII code	
Read_char	12		Char in a0

Integer Registers	
Register Name	Usage
zero	Constant 0
ra	Return address (routines/functions)
sp	Stack pointer
gp	Global pointer
tp	Thread pointer
t0..t6	Temporary (NOT preserved across calls)
s0..s11	Saved temporary (preserved across calls)
a0, a1	Arguments for functions / return value
a2..a7	Arguments for functions
Floating-point registers	
ft0..ft11	Temporary (NOT preserved across calls)
fs0..fs11	Saved temporary (preserved across calls)
fa0, fa1	Arguments for functions / return value
fa2..fa7	Arguments for functions

Data transfer			Arithmetic (floating-point, s/d)		
li rd, n	rd = n	(PseudoInst, n-> 32 bits)	fmv.s fd, fs1	fd = fs1	
mv rd, rs	rd = rs		fadd.s fd, fs1, fs2	fd = fs1+fs2	
lui rd, inm	rd = inm[31:12] <<12 (extend the sign)		fsub.s fd, fs1, fs2	fd = fs1-fs2	
Arithmetic (Integer)			fmul.s fd, fs1, fs2	fd = fs1*fs2	
add rd, rs1, rs2	rd = rs1+rs2		fdiv.s fd, fs1, fs2	fd = fs1/fs2	
addi rd, rs1, n	rd = rs1 + n	(n-> 12 bits)	fmin.s fd, fs1, fs2	fd = min(fs1,fs2)	
sub rd, rs1, rs2	rd = rs1- rs2		fmax.s fd, fs1, fs2	fd = max(fs1,fs2)	
mul rd, rs1, rs2	rd = rs1* rs2		fsqrt.s fd, fs1	fd = sqrt(fs1)	
div rd, rs1, rs2	rd = rs1/rs2		fmadd.s fd, fs1, fs2, fs3	fd = fs1*fs2+fs3	
rem rd, rs1, rs2	rd = rs1% rs2		fmsub.s fd, ff1, fs2, fs3	fd = fs1*fs2-fs3	
Logical (Integer)			fabs.s fd, fs1	fd = fs	
and rd, rs1, rs2	rd = rs1 AND rs2		fneg.s fd, fs1	fd = -fs	
andi rd, rs1, n	rd = rs1 AND n	(n-> 12 bits)	Integer ↔ Floating point		
or rd, rs1, rs2	rd = rs1 OR rs2		fmv.w.x fd, rs	fd = rs	single = integer
ori rd, rs1, n	rd = rs1 OR n	(n-> 12 bits)	fmv.x.w rd, fs	rd = fs	integer = single
Comparison (Integer), n→ 12 bits			Comparison (integer), n→ 12 bits		
not rd, rs1	rd = !rs1	(one's complement)	slt rd, rs1, rs2	if (s(rs1) < s(rs2)) rd = 1; else rd = 0	
neg rd, rs1	rd = !rs1 + 1	(two's complement)	sltu rd, rs1, rs2	if (u(rs1) < u(rs2)) rd = 1; else rd = 0	
xor rd, rs1, rs2	rd = rs1 XOT rs2		slti rd, rs1, n	if (s(rs1) < s(n)) rd = 1; else rd = 0	
srli rd, rs1, n	rd = rs1 >> n	logical, n-> 5 bits	sltiu rd, rs1, n	if (u(rs1) < u(n)) rd = 1; else rd = 0	
slli rd, rs1, n	rd = rs1 << n	n-> 5 bits	seqz rd, rs1	if (rs1 == 0) rd = 1; else rd = 0	
srai rd, rs1, n	rd = rs1 >> n	arithmetic, n-> 5 bits	snez rd, rs1	if (rs1 != 0) rd = 1; else rd = 0	
sra rd, rs1, rs2	rd = rs1 >> rs2	arithmetic	sgtz rd, rs1	if (rs1 > 0) rd = 1; else rd = 0	
sll rd, rs1, rs2	rd = rs1 << rs2		sltz rd, rs1	if (rs1 < 0) rd = 1; else rd = 0	
srl rd, rs1, rs2	rd = rs1 >> rs2	logical	Comparison (floating point) (rd=int register, fs1 and fs2 floating point register)		
Branch Instructions (integer registers)			Comparison (floating point) (rd=int register, fs1 and fs2 floating point register)		
beq t0 t1 etiq	Jump to etiq if t0==t1		feq.s rd, fs1, fs2	if (fs1== fs2) rd= 1;else rd = 0 (float)	
bne t0 t1 etiq	Jump to etiq if t0!=t1		fle.s rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (float)	
blt t0 t1 etiq	Jump to etiq if t0<t1		flt.s rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (float)	
bltu t0 t1 etiq	Jump to etiq if t0<t1 (unsigned)		feq.d rd, fs1, fs2	if (fs1== fs2) rd= 1;else rd = 0 (double)	
bge t0 t1 etiq	Jump to etiq if t0>t1		fle.d rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (double)	
bgeu t0 t1 etiq	Jump to etiq if t0>=t1 (unsigned)		flt.d rd, fs1, fs2	if (fs1< fs2) rd= 1;else rd = 0 (double)	
bgt t0 t1 etiq	Jump to etiq if t0>t1		Function Calls		
bgtu t0 t1 etiq	Jump to etiq if t0>t1 (unsigned)		jal ra, address	ra = PC; PC = address	
ble t0 t1 etiq	Jump to etiq if t0<=t1		jr ra	PC = ra	
bleu t0 t1 etiq	Jump to etiq if t0<=t1 (unsigned)		Hardware Counter		
j etiq	PC = PC + etiq		rdcycle rd	rd = number of elapsed cycles	
Memory Access (Integer registers), n→12 bits			Memory access (floating point), n→12bits		
la rd, address	rd = address	address->32 bits	flw fd, n(rs1)	fd = Memory[n+rs1]	load float
lb rd, n(rs1)	rd = Memory[n+rs1]	load byte	fsw fd, n(rs1)	Memory[n+rs1] = fd	store float
lbu rd, n(rs1)	rd = Memory[n+rs1]	load byte unsigned	fld fd, n(rs1)	fd = Memory[n+rs1]	load double
lw rd, n(rs1)	rd = Memory[n+rs1]	load word	fsd fd, n(rs1)	Memory[n+rs1] = fd	store double
sb rd, n(rs1)	Memory[n+rs1] = rd	store byte			
sw rd, n(rs1)	Memory[n+rs1] = rd	store word			
Conversion Operations			Floating-point Classification		
fcvt.w.s rd, fs1	From single precision (fs1) to integer (rd) with sign		fclass.s rd, fs1	Classify single precision	
fcvt.wu.s rd, fs1	From single precision (fs1) to integer (rd) without sign		fclass.d rd, fs1	Classify double precision	
fcvt.s.w fd, rs1	From integer with sign (rs1) to single precision (fd)		Bit position in rd		
fcvt.s.wu fd, rs1	From integer without sign (rs1) to single precision (fd)		0, 7	-Inf, +Inf	
fcvt.w.d rd, fs1	From rom double precision (fs1) to integer (rd) with sign		1	Normalized negative	
fcvt.wu.d rd, fs1	From double precision (fs1) to integer (rd) without sign		2	Not normalized negative	
fcvt.d.w fd, rs1	From integer with sign (rs1) to double precision (fd)		3, 4	-0, +0	
fcvt.d.wu fd, rs1	From integer without sign (rs1) to double precision (fd)		5	Non normalized positive	
fcvt.s.d fd, fs1	From double (fs1) to single precision (fd)		6	Normalized positive	
fcvt.d.s fd, fs1	From single (fs1) to double precision (fd)		8, 9	Signaling NaN, Quiet NaN	