

RISC-V: guía de referencia (CREATOR)

Llamada al sistema (ecall)			
Servicio	Código	Argumentos	Resultados
print_init	1	a0 = entero	
print_float	2	fa0 = float	
print_double	3	fa0 = double	
print_string	4	a0 = dir. string	
read_int	5		entero en a0
read_float	6		float en fa0
read_double	7		double in fa0
read_string	8	a0 = dir. string a1 = longitud	
sbrk	9	a0 = # bytes	dir. en a0
exit	10		
print_char	11	a0 = código ASCII	
read_char	12		char en a0

Registro de enteros	
Nombre registro	Uso
zero	Constante 0
ra	Dirección retorno (rutinas/funciones)
sp	Puntero de pila
gp	Puntero global
tp	Puntero de hilo
t0..t6	Temporal (NO se preserva en llamadas)
s0..s11	Temp. Guardados (se preserva en llam.)
a0, a1	Arg. para funciones / valor retorno
a2..a7	Argumentos para funciones
Registros de coma flotante	
ft0..ft11	Temporal (NO se preserva en llamadas)
fs0..fs11	Temp. guardados (se preserva en llam.)
fa0, fa1	Arg. para funciones / valor retorno
fa2..fa7	Argumentos para funciones

Transferencia de datos				Aritméticas (coma flotante, .s/.d)	
li rd, n	rd = n	(PseudoInst, n-> 32 bits)		fmv.s	rd = rs
mv rd, rs	rd = rs			fadd.s rd, rs1, rs2	rd = rs1 + rs2
lui rd, inm	rd = inm[31:12] <<12	(extensión signo)		fsub.s rd, rs1, rs2	rd = rs1 - rs2
Aritméticas (enteros)				fmul.s rd, rs1, rs2	rd = rs1 * rs2
add rd, rs1, rs2	rd = rs1 + rs2			fdiv.s rd, rs1, rs2	rd = rs1 / rs2
addi rd, rs1, n	rd = rs1 + n			fmin.s rd, rs1, rs2	rd = min(rs1,rs2)
sub rd, rs1, rs2	rd = rs1 - rs2			fmax.s rd, rs1, rs2	rd = max(rs1,rs2)
mul rd, rs1, rs2	rd = rs1 * rs2			fsqrt.s rd, rs	rd = sqrt(rs)
div rd, rs1, rs2	rd = rs1 / rs2			fmadd.s rd, rs1, rs2, rs3	rd = rs1 * rs2 + rs3
rem rd, rs1, rs2	rd = rs1 % rs2			fmsub.s rd, rs1, rs2, rs3	rd = rs1 * rs2 - rs3
Lógicas (entero)				fabs.s rd, rs	rd = rs
and rd, rs1, rs2	rd = rs1 AND rs2			fneg.s rd, rs	rd = -rs
andi rd, rs, n	rd = rs1 AND n	(n-> 12 bits)		Entero ↔ Coma flotante	
or rd, rs1, rs2	rd = rs1 OR rs2			fmv.w.x rd, rs	rd = rs simple = entero
ori rd, rs1, n	rd = rs1 OR n	(n-> 12 bits)		fmv.x.w rd, rs	rd = rs entero = simple
not rd, rs1	rd = !rs1	(complemento a uno)		Comparación (enteros), n-> 12 bits	
neg rd, rs1	rd = !rs1 + 1	(complemento a dos)		slt rd, rs1, rs2	if (s(rs1) < s(rs2)) rd = 1; else rd = 0
xor rd, rs1, rs2	rd = rs1 XOR rs2			sltu rd, rs1, rs2	if (u(rs1) < u(rs2)) rd = 1; else rd = 0
srli rd, rs1, n	rd = rs1 >> n	(derecha lógico)		slti rd, rs1, n	if (s(rs1) < s(n)) rd = 1; else rd = 0
slli rd, rs1, n	rd = rs1 << n	(n-> 5 bits)		sltiu rd, rs1, n	if (u(rs1) < u(5)) rd = 1; else rd = 0
srai rd, rs1, n	rd = rs1 >> n	(derecha aritmético)		seqz rd, rs1	if (rs1 == 0) rd = 1; else rd = 0
sra rd, rs1, rs2	rd = rs1 >> rs2	(derecha aritmético)		snez rd, rs1	if (rs1 != 0) rd = 1; else rd = 0
sll rd, rs1, rs2	rd = rs1 << rs2			sgtz rd, rs1	if (rs1 > 0) rd = 1; else rd = 0
srl rd, rs1, rs2	rd = rs1 >> rs2	(derecha lógico)		sltz rd, rs1	if (rs1 < 0) rd = 1; else rd = 0
Instrucciones de salto (registros de enteros)				Comparación (coma flotante) (rd=reg. entero, rs1 y rs2 reg. de coma flotante)	
beq t0 t1 etiq	Jump to etiq if t0==t1			feq.s rd, rs1, rs2	if (rs1== rs2) rd= 1;else rd = 0 (float)
bne t0 t1 etiq	Jump to etiq if t0!=t1			fle.s rd, rs1, rs2	if (rs1<= rs2) rd= 1;else rd = 0 (float)
blt t0 t1 etiq	Jump to etiq if t0<t1			flt.s rd, rs1, rs2	if (rs1< rs2) rd= 1;else rd = 0 (float)
bltu t0 t1 etiq	Jump to etiq if t0<t1 (unsigned)			feq.d rd, rs1, rs2	if (rs1== rs2) rd= 1;else rd = 0 (double)
bge t0 t1 etiq	Jump to etiq if t0>=t1			fle.d rd, rs1, rs2	if (rs1<= rs2) rd= 1;else rd = 0 (double)
bgeu t0 t1 etiq	Jump to etiq if t0>=t1 (unsigned)			flt.d rd, rs1, rs2	if (rs1< rs2) rd= 1;else rd = 0 (double)
bgt t0 t1 etiq	Jump to etiq if t0>t1			Llamadas a función	
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 = número de ciclos de reloj usados
Acceso a memoria (registro de enteros)				Acceso a memoria (coma flotante)	
la rd, address	rd = dirección	dirección->32 bits		flw rd, n(rs1)	rd = Memoria[n+rs1] load float
lb rd, n(rs1)	rd = Memory[n+rs1]	load byte		fsw rd, n(rs1)	Memoria[n+rs1] = rd store float
lbu rd, n(rs1)	rd = Memory[n+rs1]	load byte unsigned		fld rd, n(rs1)	rd = Memoria[n+rs1] load double
lw rd, n(rs1)	rd = Memory[n+rs1]	load word		fsd rd, n(rs1)	Memoria[n+rs1] = rd store double
sb rd, n(rs1)	Memory[n+rs1] = rd	store byte			
sw rd, n(rs1)	Memory[n+rs1] = sw	store word			
Operaciones de conversión				Clasificación de coma flotante	
fcvt.w.s rd, rs1	De simple precisión (fs1) a entero (rd) con signo			fclass.s rd, rs1	Clasifica simple precisión
fcvt.wu.s rd, rs1	De simple precisión (fs1) a entero (rd) sin signo			fclass.d rd, rs1	Clasifica doble precisión
fcvt.s.w rd, rs1	De entero con signo (rs1) a simple precisión (rd)			Valor en rd	
fcvt.s.wu rd, rs1	De entero sin signo (rs1) a simple precisión (rd)			0, 7	-Inf, +Inf
fcvt.w.d rd, rs1	De doble precisión (fs1) a entero (rd) con signo			1	Normalizado negativo
fcvt.wu.d rd, rs1	De doble precisión (fs1) a entero (rd) sin signo			2	No normalizado negative
fcvt.d.w rd, rs1	De entero con signo (rs1) a double precisión (rd)			3, 4	-0, +0
fcvt.d.wu rd, rs1	De entero sin signo (rs1) a double precisión (rd)			5	Normalizado positivo
fcvt.s.d rd, rs1	De doble precisión (rs1) a simple precisión (rd)			6	Not normalizado positivo
fcvt.d.s rd, rs1	De simple precisión (rs1) a double precisión(rd)			8, 9	NaN