Common RISC-V instructions

Notes: op, funct, rd, rs1, rs2, imm, address, shamt refer to fields in the instruction format
PC is assumed to point to the next instruction; Mem is the byte addressed main memory

Assembly Instruction	Instr. Format	op/funct3/funct7	Comments	Pseudo
add t1,t2,t3	R	51/0/0	Addition: set t1 to (t2 plus t3)	
addi t1,t2,imm	I	19/0/0	Addition immediate: set t1 to (t2 plus signed 12-bit immediate)	
sub t1,t2,t3	R	51/0/16	Subtraction: set t1 to (t2 minus t3)	
mul t1,t2,t3	R	51/0/1	Multiplication: set t1 to the lower 32 bits of t2*t3	
mulh t1,t2,t3	R	51/1/1	Multiplication: set t1 to the upper 32 bits of t2*t3 using signed multiplication	
mulhsu t1,t2,t3	R	51/2/1	Multiplication: set t1 to the upper 32 bits of t2*t3 where t2 is signed and t3 is unsigned	
mulhu t1,t2,t3	R	51/3/1	Multiplication: set t1 to the upper 32 bits of t2*t3 using unsigned multiplication	
div t1,t2,t3	R	51/4/1	Division: set t1 to the result of t2/t3	
divu t1,t2,t3	R	51/5/1	Division: set t1 to the result of t2/t3 using unsigned division	
rem t1,t2,t3	R	51/6/1	Remainder: set t1 to the remainder of t2/t3	
remu t1,t2,t3	R	51/7/1	Remainder: set t1 to the remainder of t2/t3 using unsigned division	
and t1,t2,t3	R	51/7/0	Bitwise AND : Set t1 to bitwise AND of t2 and t3	
andi t1,t2,imm	I	19/4	Bitwise AND immediate : Set t1 to bitwise AND of t2 and sign-extended 12-bit immediate	
or t1,t2,t3	R	51/6/0	Bitwise OR : Set t1 to bitwise OR of t2 and t3	
ori t1,t2,imm	I	19/3	Bitwise OR immediate : Set t1 to bitwise OR of t2 and sign-extended 12-bit immediate	
xor t1,t2,t3	R	51/4/0	Bitwise XOR : Set t1 to bitwise XOR of t2 and t3	
xori t1,t2,imm	I	19/3	Bitwise XOR immediate : Set t1 to bitwise XOR of t2 and sign-extended 12-bit immediate	
sll t1,t2,t3	R	51/1/0	Shift left logical: Set t1 to result of shifting t2 left by number of bits specified by value in low-order 5 bits of t3	
slli t1,t2,imm	I	19/5	Shift left logical : Set t1 to result of shifting t2 left by number of bits specified by immediate	
srl t1,t2,t3	R	51/5/0	Shift right logical: Set t1 to result of shifting t2 right by number of bits specified by value in low-order 5 bits of t3	
srli t1,t2,imm	I	19/6	Shift right logical : Set t1 to result of shifting t2 right by number of bits specified by immediate	
sra t1,t2,t3	R	51/5/16	Shift right arithmetic: Set t1 to result of sign-extended shifting t2 right by number of bits specified by value in low-order 5 bits of t3	
srai t1,t2,imm	I	19/6/32	Shift right arithmetic : Set t1 to result of sign-extended shifting t2 right by number of bits specified by immediate	
lw t1, offset(t2)	I	3/2	Set t1 to contents of effective memory word address	
lui t1,imm	U	55	Load upper immediate: set t1 to 20-bit followed by 12 0s	
sw t1, offset(t2)	S	35/2	Store word : Store contents of t1 into effective memory word address	
sb t1, offset(t2)	S	35/0	Store byte : Store the low-order 8 bits of t1 into the effective memory byte address	
beq t1,t2,label	SB	99/0	Branch if equal : Branch to statement at label's address if t1 and t2 are equal	
bge t1,t2,label	SB	99/5	Branch if greater than or equal: Branch to statement at label's address if t1 is greater than or equal to t2	
slt t1,t2,t3	R	51/2/0	Set less than : If t2 is less than t3, then set t1 to 1 else set t1 to 0	
slti t1,t2,imm	I	19/2	Set less than immediate : If t2 is less than sign-extended 12-bit immediate, then set t1 to 1 else set t1 to 0	
sltu t1,t2,t3	R	51/3/0	Set less than : If t2 is less than t3 using unsigned comparision, then set t1 to 1 else set t1 to 0	
jal t1, target	UJ	111	Jump and link : Set t1 to Program Counter (return address) then jump to statement at target address	
bne t1,t2,label	SB	99/1	Branch if not equal : Branch to statement at label's address if t1 and t2 are not equal	
la t1, label	-	-	Load Address: Set t1 to label's address	X
li t1, imm	-	-	Load Immediate: Set t1 to 12-bit immediate (sign-extended)	X
li t1, imm	-	-	Load Immediate: set t1 to 32-bit immediate	X
mv t1, t2	-	-	MoVe: Set t1 to contents of t2	X

Register		Usage	Caller-	Callee-
Nr.	Name		saved	
x0	zero	Konstante mit 0	-	-
x1	ra	Rücksprungadresse	X	(X)
x2	sp	Stack-Pointer	-	-
x3	gp	Global-Pointer	-	-
x4	tp	Thread-Pointer	-	-
x5-7	t0-2	Temporär Variablen	X	-
x8	s0/fp	Langlebige Variablen/Frame-Pointer		X
x9	s1	Langlebige Variablen		X
x10-11	a0-1	Argumente/Rückgabe	X	
x12-17	a2-7	Argumente	X	
x18-27	s2-11	Langlebige Variablen		X
x28-31	t3-6	Temporäre Variable	X	

Systemaufrufe			
Funktion	**Code in a7**	**Eingabe**	**Ausgabe**
PrintInt	1	Integer in a0	-
PrintFloat	2	Float in fa0	-
PrintDouble	3	Double in fa0	-
PrintString	4	Adresse des Strings in a0	-
ReadInt	5	-	Integer in a0
ReadFloat	6	-	Float in fa0
ReadDouble	7	-	Double in fa0
ReadString	8	Adresse des Strings in a0; Max. Länge in a1	-
Sbrk	9	Byte-Anzahl in a0	Anfangsadresse in a0
Exit	10	-	-
Exit2	93	Exit-Code in a0	-

Direktiven	
Name	**Beschreibung**
.align	Align next data item on specified byte boundary (0=byte, 1=half, 2=word, 3=double)
.ascii	Store the string in the Data segment but do not add null terminator
.asciz	Store the string in the Data segment and add null terminator
.byte	Store the listed value(s) as 8 bit bytes
.data	Subsequent items stored in Data segment at next available address
.double	Store the listed value(s) as double precision floating point
.extern	Declare the listed label and byte length to be a global data field
.float	Store the listed value(s) as single precision floating point
.globl	Declare the listed label(s) as global to enable referencing from other files
.half	.half Store the listed value(s) as 16 bit halfwords on halfword boundary
.space	Reserve the next specified number of bytes in Data segment
.text	Subsequent items (instructions) stored in Text segment at next available address
.word	Store the listed value(s) as 32 bit words on word boundary