

Reference Materials

1. RTN Description of SRC

Unified RTN for SRC

Below is the entire RTN description for SRC with reset and exception handling integrated into it.

Memory

Processor state

PC 31..0 :	program counter (address of the next instruction)
IR 31..0 :	instruction register
Run:	one bit run/halt indicator
Strt:	start and hard reset signal
Rst:	soft reset signal
R[0..31] 31..0 :	general purpose registers

Processor interrupt mechanism

ireq:	interrupt request signal
iack:	interrupt acknowledge signal
IE:	one bit interrupt enable flag
IPC 31..0 :	storage for PC saved upon interrupt
II 31..0 :	interrupt info.: about source of last interrupt
Isrc_info 15..0 :	information from interrupt source
Isrc_vect 7..0 :	type code from interrupt source
Ivect 31..0 := 20@0#Isrc_vect 7..0 #4@0:	

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]:	

Formats

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 :	2nd operand, conditional test, 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

Branch condition format

cond := (c3 2..0 =0 0:	never
c3 2..0 =1 1:	always
c3 2..0 =2 R[rc]=0:	if register is zero
c3 2..0 =3 R[rc] 0:	if register is nonzero
c3 2..0 =4 R[rc] 31 =0:	if register is positive or zero
c3 2..0 =5 R[rc] 31 =1):	if register is negative

Shift count format

n := ((c3 4..0 =0) R[rc] 4..0 :	shift count is register or
(c3 4..0 0) c3 4..0):	constant field of instruction

Effective address calculations

disp 31..0 := ((rb=0) c2 16..0 {sign extend}:	disp.
(rb 0) R[rb] + c2 16..0 {sign extend, 2's complement}):	addr.
rel 31..0 := PC 31..0 + c1 21..0 {sign extend, 2's comp.}:	rel. adr.

Instruction interpretation

instruction_interpretation :=

(~Run Strt (Run 1: PC, R[0..31] 0;	Hard reset
instruction_interpretation):	
Run Rst (Rst 0: IE 0: PC 0;	Soft reset
instruction_interpretation):	
Run ~ Rst (ireq IE) (IPC PC 31..0 :	Interrupt
II 15..0 Isrc_info 15..0 :	
IE 0: PC Ivect 31..0 :	
iack 1; iack 0;	
instruction_interpretation):	
Run ~ Rst ~ (ireq IE) (IR M[PC]:	Normal fetch
PC PC + 4; instruction_execution):	

Instruction execution instruction_execution := (

Load and store instructions

ld (:= op= 1)	R[ra]	M[disp]:	load register
ldr (:= op= 2)	R[ra]	M[rel]:	load register relative
st (:= op= 3)	M[disp]	R[ra]:	store register
str (:= op= 4)	M[rel]	R[ra]:	store register relative
la (:= op= 5)	R[ra]	disp:	load displacement address
lar (:= op= 6)	R[ra]	rel:	load relative address

Branch instructions

br (:= op= 8)	(cond	PC	R[rb]):	cond. branch
brl (:= op= 9)	(R[ra]	PC: cond	(PC	R[rb])): branch & link

Arithmetic instructions (assumed to be 2's complement arithmetic)

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

shr (:= op= 26)	R[ra]	31..0	(n @ 0) # R[rb] 31..n :	right
shra (:= op= 27)	R[ra]	31..0	(n @ R[rb] 31) # R[rb] 31..n :	arith.
shl (:= op= 28)	R[ra]	31..0	R[rb] 31-n..0 #(n @ 0):	left
shc (:= op= 29)	R[ra]	31..0	R[rb] 31-n..0 #R[rb] 31..32-n :	circ.

Interrupt instructions

een (:= op = 10)	(IE	1):	exception enable
edi (:= op = 11)	(IE	0):	exception disable
rfi (:= op = 30)	(PC	IPC: IE	1): return from interrupt
svi (:= op = 16)	(R[ra]	15..0	II 15..0 : R[rb] IPC 31..0): save interrupt state
ri (:= op = 17)	(II	15..0	R[ra] 15..0 : IPC 31..0 R[rb]): restore interrupt state

Miscellaneous instructions

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

instruction_interpretation.