# 以软件形式添加LDB/STB

.include "beta.uasm"

.include "checkoff.uasm"

regs:

RESERVE(32) // Array used to store register contents

UI:

save\_all\_regs(regs)

LD(xp,-4,r0) // illegal instruction

extract\_field(r0,31,26,r1) // extract opcode, bits 31:26

CMPEQC(r1,0x10,r2) // OPCODE=16?

BT(r2, LDB) // yes, handle the swapreg instruction.

CMPEQC(r1,0x11,r2) // OPCODE=17?

BT(r2, STB)

LD(r31,regs,r0) // Its something else. Restore regs

LD(r31,regs+4,r1) // we've used, and go to the system's

LD(r31,regs+8,r2) // Illegal Instruction handler.

BR(\_IllegalInstruction)

LDB:

extract\_field(r0,20,16,r2) // extract ra field from trapped instruction

MULC(r2, 4, r2) // convert to byte offset into regs array

LD(r2, regs, r4) // r4 <- regs[ra]

extract\_field(r0,25,21,r1) // extract rc field from trapped instruction

MULC(r1, 4, r1) // convert to byte offset into regs array

LD(r1, regs, r3) // r3 <- regs[rc]

SHLC(r0, 16, r2)

SRAC(r2, 16, r2)

ADD(r4, r2, r4)

LD(r4, 0, r5) // MDATA

extract\_field(r4,1,0,r4) // EA1:0

CMPEQC(r4, 0, r8)

BT(r8, L0)

CMPEQC(r4, 1, r8)

BT(r8, L1)

CMPEQC(r4, 2, r8)

BT(r8, L2)

CMPEQC(r4, 3, r8)

BT(r8, L3)

L0:

extract\_field(r5,7,0,r7)

BR(L4)

L1:

extract\_field(r5,15,8,r7)

BR(L4)

L2:

extract\_field(r5,23,16,r7)

BR(L4)

L3:

extract\_field(r5,31,24,r7)

L4:

ANDC(r3, 0, r3)

OR(r3, r7, r3)

ST(r3, regs, r1)

restore\_all\_regs(regs)

JMP(xp)

STB:

extract\_field(r0,20,16,r2) // extract ra field from trapped instruction

MULC(r2, 4, r2) // convert to byte offset into regs array

LD(r2, regs, r4) // r4 <- regs[ra]

extract\_field(r0,25,21,r1) // extract rc field from trapped instruction

MULC(r1, 4, r1) // convert to byte offset into regs array

LD(r1, regs, r3) // r3 <- regs[rc]

SHLC(r0, 16, r2)

SRAC(r2, 16, r2)

ADD(r4, r2, r4) // EA

LD(r4, 0, r5) // MDATA

extract\_field(r3, 7, 0, r3) // RC7-0

extract\_field(r4, 1, 0, r9) // EA1:0

CMPEQC(r9, 0, r8)

BT(r8, A0) // yes, handle the swapreg instruction.

CMPEQC(r9, 1, r8)

BT(r8, A1) // yes, handle the swapreg instruction.

CMPEQC(r9, 2, r8)

BT(r8, A2) // yes, handle the swapreg instruction.

CMPEQC(r9, 3, r8)

BT(r8, A3) // yes, handle the swapreg instruction.

A0:

ORC(r9, 0xffff, r9)

ANDC(r9, 0xff00, r9)

BR(A4)

A1:

SHLC(r3, 8, r3)

ORC(r9, 0xffff, r9)

SHLC(r9, 16, r9)

ORC(r9, 0x00ff, r9)

BR(A4)

A2:

SHLC(r3, 16, r3)

ORC(r9, 0xff00, r9)

ANDC(r9, 0xff00, r9)

SHLC(r9, 16, r9)

ORC(r10, 0x00ff, r10)

SHLC(r10, 8, r10)

ORC(r10, 0x00ff, r10)

OR(r9, r10, r9)

BR(A4)

A3:

SHLC(r3, 24, r3)

ORC(r9, 0x00ff, r9)

SHLC(r9, 16, r9)

ORC(r10, 0x00ff, r10)

SHLC(r10, 8, r10)

ORC(r10, 0x00ff, r10)

OR(r9, r10, r9)

A4:

AND(r5, r9, r5)

OR(r5, r3, r5)

ST(r5, 0, r4)

restore\_all\_regs(regs)

JMP(xp)