vmx20 Virtual Machine Instructions

---

halt

Operation

Halt the processor.

Format

halt

Encoding

halt = 0 (0x00)

Description

The processor executing the instruction is halted. When all processors have halted, the VM terminates.

---

load

Operation

Load value from memory.

Format

load reg,addr

Encoding

load = 1 (0x01)

Description

reg = *(pc+*addr)*. If the effective address is out of range of the available memory, then the executing processor halts with an error.

---

store

Operation

Store value to memory.

Format

store reg,addr

Encoding

store = 2 (0x02)

Description

*(pc+*addr*) = *reg*. If the effective address is out of range of the available memory, then the executing processor halts with an error.

---

ldimm

Operation

Load immediate value.

Format

ldimm reg,const

Encoding

ldimm = 3 (0x03)

Description

reg = const.

---

ldaddr

Operation

Load address.

Format

ldaddr reg,addr

Encoding

ldaddr = 4 (0x04)

Description

reg = pc + *addr*.

---

ldind

Operation

Load indirect with offset.

Format

ldind reg1,offset(reg2)

Encoding

ldind = 5 (0x05)

Description

reg1 = *(*reg2* + *offset)*. If the effective address is out of range of the available memory, then the executing processor halts with an error.

---

stind

Operation

Store indirect with offset.

Format

stind reg1,offset(reg2)

Encoding

stind = 6 (0x06)

Description

*(*reg2 + offset*) = *reg1*. If the effective address is out of range of the available memory, then the executing processor halts with an error.

---

addf

Operation

Floating-point addition.

Format

addf reg1,reg2

Encoding

addf = 7 (0x07)

Description

reg1 = reg1 + reg2.

---

subf

Operation

Floating-point subtraction.

Format

subf reg1,reg2

Encoding

subf = 8 (0x08)

Description

reg1 = reg1 - reg2.

---

divf

Operation

Floating-point division.

Format

divf reg1,reg2

Encoding

divf = 9 (0x09)

Description

reg1 = reg1 / *reg2*. If *reg2* contains zero, then the executing processor halts with an error.

---

mulf

Operation

Floating-point multiplication.

Format

mulf reg1,reg2

Encoding

mulf = 10 (0x0A)

Description

*reg1 = reg1 * reg2*.

---

addi

Operation

Integer addition.

Format

addi reg1,reg2

Encoding

addi = 11 (0x0B)

Description

reg1 = reg1 + reg2.

---

subi

Operation

Integer subtraction.

Format

subi reg1,reg2

Encoding

subi = 12 (0x0C)

Description

reg1 = reg1 - reg2.

---

divi

Operation

Integer division.

Format

divi reg1,reg2

Encoding

divi = 13 (0x0D)

Description

reg1 = reg1 / *reg2*. If *reg2* contains zero, then the executing processor halts with an error.

---

muli

Operation

Integer multiplication.

Format

muli reg1,reg2

Encoding

muli = 14 (0x0E)

Description

*reg1 = reg1 * reg2*.

---

call

Operation

Call a function.

Format

call addr

Encoding

call = 15 (0x0F)

Description

sp -= 1; *sp = pc; pc += *addr*; sp -= 1; *sp = fp; fp = sp; sp -= 1; *sp = 0. If sp is out of range of the available memory, then the executing processor halts with an error.

---

ret

Operation

Return from a function.

Format

ret

Encoding

ret = 16 (0x10)

Description

returnValue = *sp; sp += 1; savedFP = *sp; sp += 1; returnAddress = *sp; sp += 1; fp = savedFP; pc = returnAddress; *(fp-1) = returnValue. If sp or fp is out of range of the available memory, then the executing processor halts with an error.

---

blt

Operation

Branch if less than.

Format

blt reg1,reg2,addr

Encoding

blt = 17 (0x11)

Description

if *reg1* < *reg2* then pc += *addr*. This is an integer comparison.

---

bgt

Operation

Branch if greater than.

Format

bgt reg1,reg2,addr

Encoding

bgt = 18 (0x12)

Description

if *reg1* > *reg2* then pc += *addr*. This is an integer comparison.

---

beq

Operation

Branch if equal to.

Format

beq reg1,reg2,addr

Encoding

beq = 19 (0x13)

Description

if *reg1 == reg2* then pc += *addr*. This is an integer comparison.

---

jmp

Operation

Unconditional jump.

Format

jmp addr

Encoding

jmp = 20 (0x14)

Description

pc += *addr*.

---

cmpxchg

Operation

Compare and exchange.

Format

cmpxchg reg1,reg2,addr

Encoding

cmpxchg = 21 (0x15)

Description

if *reg1* == *(pc + *addr*) then *(pc + *addr*) = *reg2* else *reg1* = *(pc + *addr*).
Note: this is done atomically by locking the memory bus for the duration of the instruction. If the effective address is out of range of the available memory, then the executing processor halts with an error. The comparison is an integer comparison.

---

getpid

Operation

Get processor ID.

Format

getpid reg

Encoding

getpid = 22 (0x16)

Description

reg is assigned the processor id of the processor executing the instruction.
Note: processors id begin at 0 and run to n-1, where n is the number of processors in the VM.

---

getpn

Operation

Get number of processors.

Format

getpn reg

Encoding

getpn = 23 (0x17)

Description

reg is assigned the number of processors in the VM.

---

push

Operation

Push contents of register.

Format

push reg

Encoding

push = 24 (0x18)

Description

sp -= 1; *sp = *reg*. If sp is out of range of the available memory, then the executing processor halts with an error.

---

pop

Operation

Pop into register.

Format

pop reg

Encoding

pop = 25 (0x19)

Description

reg = *sp; sp += 1. If sp is out of range of the available memory, then the executing processor halts with an error.
