a7address registers (a7 is reserved for the stack pointer)

spstack pointer (a7)

d0d7data registers

pcprogram counter

zpchack for program space, but 0 addressing (GAS)

cc or ccrcondition code register

srstatus register

fp0fp7floating point registers

fpi or fpirfloating point instruction register

fpc or fpcrfloating point control register

fps or fpsrfloating point status register

movw a3@,d2
Displacementan@(d)movw a3@(24),d2
Word Indexan@(d,ri:w)movw a3@(16, d2:w),d3
Long Indexan@(d,ri:l)movw a3@(16, d2:l),d3
Absolute Shortxxx:wmovw 14:w,d2
Absolute Longxxx:lmovw 14:l,d2
PC Displacementpc@(d)movw pc@(20),d3

Post-Indexed
Normalidentifiermovw var,d3
Immediate#xxxmovw #27+3,d3

An)An@(An)100reg. number:An
*(An+d16)An@(d16)(d16,An)101reg. number:An
*(An+d8+Xn)An@(d8,Xn)(d8,An,Xn)110reg. number:An
*(An+bd+Xn)An@(bd,Xn)(bd,An,Xn)110reg. number:An
((An+bd)+od+Xn)An@(bd)@(od,Xn)([bd,An],Xn,od)110reg. number:An
((An+bd+Xn)+od)An@(bd,Xn)@(od)([bd,An,Xn],od)110reg. number:An
*(xxx)xxx:w(xxx).W111000
*(xxx)xxx:l(xxx).L111001
data#data#<data>111100
Ð Ð Ð Ð Ð
Ð Ð Ð Ð Ð
*(pc+d16)pc@(d16)(d16,pc)111010
*(pc+d8+Xn)pc@(d8,Xn)(d8,pc,Xn)111011
*(pc+bd+Xn)pc@(bd,Xn)(bd,pc,Xn)111011
((pc+bd)+od+Xn)pc@(bd)@(od,Xn)([bd,pc],Xn,od)111011
((pc+bd+Xn)+od)pc@(bd,Xn)@(od)([bd,pc,Xn],od)111011

,An@

addqaddq#data,<ea>Add Quick

addxaddxDn,DnAdd Extended
addxAn@,An@

andand<ea>,DnAND Logical
andDn,<ea>

andiandi#data,<ea>AND Immediate
and#data,<ea>

andi to ccrandi#data,ccrAND Immediate
and#data,ccrto Condition Codes

andi to srandi#data,srAND Immediate
and#data,srto Status Register

aslasl#data,DnArithmetic Shift Left
aslDn,Dn
asl<ea>

asrasr#data,DnArithmetic Shift Right
asrDn,Dn
asr<ea>

bcc (and bccs)Branch Conditionally
bcc<ea>carry clear
bcs<ea>carry set
beq<ea>equal
bge<ea>greater or equal
bgt<ea>greater than
bhi<ea>high
ble<ea>less or equal
bls<ea>low or same
blt<ea>less than
bmi<ea>minus
bne<ea>not equal
bpl<ea>plus
bvc<ea>overflow clear
bvs<ea>overflow set

bchgbchgDn,<ea>Test a Bit and Change
bchg#data,<ea>

bclrbclrDn,<ea>Test a Bit and Clear
bclr#data,<ea>

bfextsbfexts<ea>DnExtract Bit Field Signed
bfexts<ea>Dn
bfexts<ea>Dn
bfexts<ea>Dn

bfextubfextu<ea>DnExtract Bit Field Unsigned
bfextu<ea>Dn
bfextu<ea>Dn
bfextu<ea>Dn

bfffobfffo<ea>DnFind First One in Bit Field
bfffo<ea>Dn
bfffo<ea>Dn
bfffo<ea>Dn

bfinsbfinsDn,<ea>Insert Bit Field
bfinsDn,<ea>
bfinsDn,<ea>
bfinsDn,<ea>

bfsetbfset<ea>Test Bit Field and Set
bfset<ea>Dn
bfset<ea>Dn
bfset<ea>Dn

bftstbftst<ea>Test Bit Field
bftst<ea>Dn
bftst<ea>Dn
bftst<ea>Dn

bkptbkpt#dataBreakpoint

brabra<ea>Branch Always
bras<ea>(short form)

bsetbsetDn,<ea>Test a Bit and Set
bset#data,<ea>

bsrbsr<ea>Branch to Subroutine
bsrs<ea>(short form)

btstbtstDn,<ea>Test a Bit
btst#data,<ea>

callmcallm#data,<ea>CALL Module

cascasDn,Dn,<ea>Compare and Swap with

cmpacmpa<ea>,AnCompare Address
cmp<ea>,An

cmpicmpi#data,<ea>Compare Immediate
cmp#data,<ea>

cmpmcmpmAn@+,An@+Compare Memory

cmp2cmp2<ea>,RnCompare Register Against
Bounds

dbccTest Cond., Decrement, Branch
dbccDn,<ea>carry clear
dbcsDn,<ea>carry set
dbeqDn,<ea>equal
dbfDn,<ea>never equal
dbgeDn,<ea>greater or equal
dbgtDn,<ea>greater than
dbhiDn,<ea>high
dbleDn,<ea>less or equal
dblsDn,<ea>low or same
dbltDn,<ea>less than
dbmiDn,<ea>minus
dbneDn,<ea>not equal
dbplDn,<ea>plus
dbraDn,<ea>always true
dbtDn,<ea>always true
dbvcDn,<ea>overflow clear
dbvsDn,<ea>overflow set

divsdivs<ea>,DnSigned Divide
divs<ea>,Dn(same as divsw)
divsl<ea>,Dn,Dn *

divudivu<ea>,DnUnsigned Divide
divu<ea>,Dn(same as divsw)
divul<ea>,Dn,Dn *

eoreorDn,<ea>Exclusive-OR Logical

eorieori#data,<ea>Exclusive-OR Immediate
eor#data,<ea>

eori to ccreori#data,ccrExclusive-OR Immediate
eor#data,ccrto Condition Code

eori to sreori#data,srExclusive-OR Immediate
eor#data,srto Status Register

illegalillegalTake Illegal Instruction Trap

jcc (and jccs)Jump or Branch Conditionally
jhi<ea>high
jls<ea>low or same
jcc<ea>carry clear
jcs<ea>carry set
jne<ea>not equal
jeq<ea>equal
jvc<ea>overflow clear
jvs<ea>overflow set
jpl<ea>plus
jmi<ea>minus
jge<ea>greater or equal
jlt<ea>less than
jgt<ea>greater than
jle<ea>less or equal

jmpjmp<ea>Jump
jra<ea>Jump or Branch
jras<ea>Jump or Branch (short form)

jsrjsr<ea>Jump to Subroutine
jbsr<ea>Jump or Branch to Subroutine
jbsrs<ea> (short form)

lealea<ea>,AnLoad Effective Address

linklinkAn,#dataLink and Allocate
linkAn,#data(same as linkw)

lsllsl#data,DnLogical Shift Left
lslDn,Dn
lsl<ea>

lsrlsr#data,DnLogical Shift Right
lsrDn,Dn
lsr<ea>

movemove<ea>,<ea>Move Data from Source
mov<ea>,<ea>to Destination

movemovesr,<ea>Move from the
  from srmovsr,<ea>Status Register

movemoveccr,<ea>Move from the
  from ccrmovccr,<ea>Condition Code Register

moveamovea<ea>,AnMove Address
move<ea>,An(alternate form)
mova<ea>,An("")
mov<ea>,An("")

movecmovecRn,RcMove Control Register
movecRc,Rn
movecRn,#data
movec#data,Rn
movcRn,Rc(alternate form)
movcRc,Rn("")
movcRn,#data("")
movc#data,Rn("")

movemmovemreglist,<ea>Move Multiple Registers
movem<ea>,reglist
movmreglist,<ea>(alternate form)
movm<ea>,reglist("")
movem#data,<ea>(reglist can be immediate
movem<ea>,#data data, as shown here)

movepmovep(d,An),DnMove Peripheral Data
movepDn,(d,An)
movp(d,An),Dn(alternate form)
movpDn,(d,An)("")

moveqmoveq#data,DnMove Quick
movq#data,Dn(alternate form)
movel#data,Dn("")
movl#data,Dn("")
moveql#data,Dn("")
movql#data,Dn("")

movesmoves<ea>,RnMove Address Space
movesRn,<ea>
movs<ea>,Rn(alternate form)
movsRn,<ea>("")

mulsmuls<ea>,DnSigned Multiply
muls<ea>,Dn,Dn *
muls<ea>,Dn(same as mulsw)

mulumulu<ea>,DnUnsigned Multiply
mulu<ea>,Dn,Dn *
mulu<ea>,Dn(same as muluw)

nbcdnbcd<ea>Negate Decimal with Extend

oror<ea>,DnInclusive-OR Logical
orDn,<ea>

oriori#data,<ea>Inclusive-OR Immediate
or#data,<ea>

ori to ccrori#data,ccrInclusive-OR Immediate
or#data,ccrto Condition Codes

ori to ssrori#data,srInclusive-OR Immediate
or#data,srto the Status Register

packpackDn,Dn,#dataPack
packAn@,An@,#data

peapea<ea>Push Effective Address

resetresetReset External Devices

rolrol#data,DnRotate Left without Extend
rolDn,Dn
rol<ea>

rorror#data,DnRotate Right without Extend
rorDn,Dn
ror<ea>

roxlroxl#data,DnRotate Left with Extend
roxlDn,Dn
roxl<ea>

roxrroxr#data,DnRotate Right with Extend
roxrDn,Dn
roxr<ea>

rtdrtd#dataReturn and Deallocate

rterteReturn from Exception

rtmrtmRnReturn from Module

rtrrtrReturn and Restore
Condition Codes

rtsrtsReturn from Subroutine

sbcdsbcdDn,DnSubtract Decimal with Extend
sbcdAn@,An@

sle<ea>less or equal
sls<ea>low or same
slt<ea>less than
smi<ea>minus
sne<ea>not equal
spl<ea>plus
st<ea>always true
svc<ea>overflow clear
svs<ea>overflow set

stopstop#dataLoad Status Register and Stop

subsub<ea>,DnSubtract
subDn,<ea>

subasuba<ea>,AnSubtract Address
sub<ea>,An

subisubi#data,<ea>Subtract Immediate
sub#data,<ea>

subqsubq#data,<ea>Subtract Quick
sub#data,<ea>

subxsubxDn,DnSubtract with Extend
subxAn@,An@

swapswapDnSwap Register Halves

tastas<ea>Test and Set an Operand

traptrap#dataTrap

trapccTrap on Condition (Unsized)
trapcccarry clear
trapcscarry set
trapeqequal
trapfnever equal
trapgegreater or equal
trapgtgreater than
traphihigh
trapleless or equal
traplslow or same
trapltless than
trapmiminus
trapnenot equal
trapplplus
traptalways true

trapgt.#datagreater than
traphi.#datahigh
traple.#dataless or equal
trapls.#datalow or same
traplt.#dataless than
trapmi.#dataminus
trapne.#datanot equal
trappl.#dataplus
trapt.#dataalways true
trapvc.#dataoverflow clear
trapvs.#dataoverflow set

trapvtrapvTrap on Overflow

tsttst<ea>Test an Operand

unlkunlkAnUnlink

unpkunpkDn,Dn,#dataUnpack BCD
unpkAn@,An@,#data

1
fetoxm1FPn,FPn
fetoxm1FPn

fgetexpfgetexp<ea>,FPnGet Exponent
fgetexpFPn,FPn
fgetexpFPn

fgetmanfgetman<ea>,FPnGet Mantissa
fgetmanFPn,FPn
fgetmanFPn

fintfint<ea>,FPnInteger Part
fintFPn,FPn
fintFPn

fintrzfintrz<ea>,FPnInteger Part, Round-to-Zero
fintrzFPn,FPn
fintrzFPn

flog10flog10<ea>,FPnLog10
flog10FPn,FPn
flog10FPn

flog2flog2<ea>,FPnLog2
flog2FPn,FPn
flog2FPn

flognflogn<ea>,FPnLoge
flognFPn,FPn
flognFPn

flognp1flognp1<ea>,FPnLoge(x+1)
flognp1FPn,FPn
flognp1FPn

fmodfmod<ea>,FPnModulo Remainder
fmodFPn,FPn
fmodFPn

fmovefmove<ea>,FPnMove Floating-Point
fmoveFPn,<ea>Data Register
fmoveFPn,<ea>
fmoveFPn,<ea>,
fmoveFPn,FPn
fmoveFPn

fmove fcrfmoveFPcr,<ea>Move Floating-Point

fpmovemfmovemFPcrlist,<ea>Move Multiple Floating-Point
ccrfmovem<ea>,FPcrlistControl Registers

fmulfmul<ea>,FPnFloating-Point Multiply
fmulFPn,FPn
fmulFPn

fnegfneg<ea>,FPnFloating-Point Negate
fnegFPn,FPn
fnegFPn

fnopfnopNo Operation

fremfrem<ea>,FPnIEEE Remainder
fremFPn,FPn
fremFPn

frestorefrestore<ea>Restore Internal Floating-Point
State

fsavefsave<ea>Save Internal Floating-Point
State

fscalefscale<ea>,FPnScale Exponent
fscaleFPn,FPn
fscaleFPn

fsccSet According to Floating-Point
fs<ea>Condition

fsgldivfsgldiv<ea>,FPnSingle Precision Divide
fsgldivFPn,FPn
fsgldivFPn

fsglmulfsglmul<ea>,FPnSingle Precision Multiply
fsglmulFPn,FPn
fsglmulFPn

fsinfsin<ea>,FPnSine
fsinFPn,FPn
fsinFPn

fsincosfsincos<ea>,FPn,FPn *Simultaneous Sine and Cosine
fsincosFPn,FPn,FPn *

fsinhfsinh<ea>,FPnHyperbolic Sine
fsinhFPn,FPn

ftanftan<ea>,FPnTangent
ftanFPn,FPn
ftanFPn

ftanhftanh<ea>,FPnHyperbolic Tangent
ftanhFPn,FPn
ftanhFPn

ftentoxftentox<ea>,FPn10x
ftentoxFPn,FPn
ftentoxFPn

ftrapccTrap on Floating-Point
ftrapCondition
ftrapw#data
ftrapl#data

ftstftst<ea>Test Floating-Point Operand
ftstFPn

ftwotoxftwotox<ea>,FPn2x
ftwotoxFPn,FPn
ftwotoxFPn