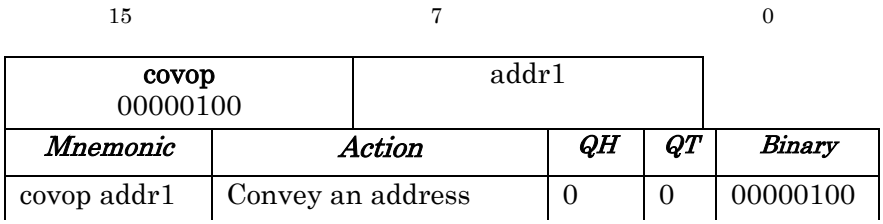
The Queue Computer Project

Instruction Set Architecture

© Abderazek Ben Abdallah
Graduate School of Information Systems
The University of Electro-communications, Tokyo
January 2003

COVOP

covop



Mnemonic	*Action*	*QH*	*QT*	*Binary*
covop addr1	Convey an address	0	0	00000100

Instruction format

8	8

Description:

Convey an 8-bit address to a load or store instruction to extend the addressing bits from 8 to 16 bits.

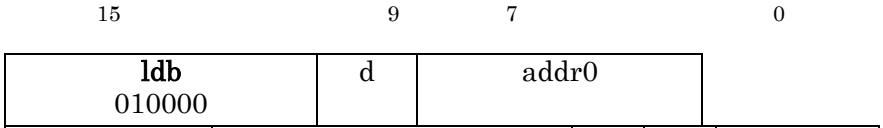
Here, QH = 0 & QT = 0

Load & Store

Instruction Format

6	2	8

Load byte ldb



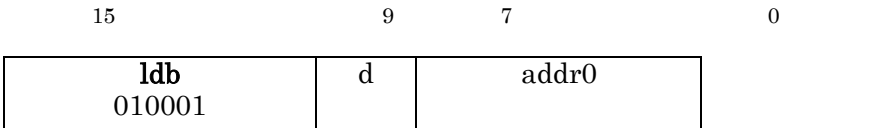
Format	*Action*	*QH*	*QT*	*Binary*
ldb addr0(d)	qtw←m((d)+addr1.addr0)	0	1	010000

Description:

Load byte to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr1.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Load byte unsigned ldbu



Format	*Action*	*QH*	*QT*	*Binary*
ldbu addr0(d)	qtw←m((d)+addr1.addr0)	0	1	010001

Description:

Load byte unsigned to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr1.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Load string lds

15	9	7	0
lds 010010	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
lds addr0(d)	qtw←m((d)+addr1.addr0*2)	0	1	010010

Description:

Load string to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr1.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Load string unsigned ldsu

15	9	7	0
ldsu 010011	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
ldsu addr0(d)	qtw←m((d)+addr1.addr0*2)	0	1	010011

Description:

Load string unsigned to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr1.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Load word ldw

15	9	7	0
ldw 010100	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
ldw addr0(d)	qtw←m((d)+addr1.addr0*4)	0	1	010100

Description:

Load word to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr1.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Store byte stb

15	9	7	0
stb 010110	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
stb addr(d)	qhw→m((d)+addr1.addr0)	1	0	010110

Description:

Store byte to the operand queue pointed by the memory address ((d)+ addr0) or ((d) + addr0.addr1) from QH if the store instruction follows a convey instruction.

So, here QH = 1 & QT = 0

Store byte unsigned **stbu**

15	9	7	0
stbu 001110	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
stbu addr(d)	qhw→m((d)+addr1.addr0)	1	0	001110

Description:

Store byte unsigned to the operand queue pointed by the memory address ((d)+ addr0) or ((d) + addr0.addr1) from QH if the store instruction follows a convey instruction.

So, here QH = 1 & QT = 0

Store string **sts**

15	9	7	0
sts 010111	d	addr0	

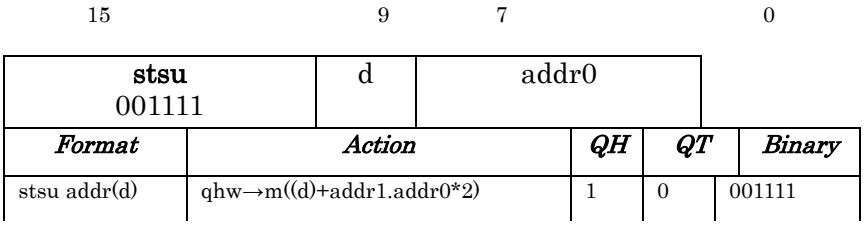
Description:

Store string to the operand queue pointed by the memory address ((d)+ addr0) or ((d) + addr0.addr1) from QH if the store instruction follows a convey instruction.

So, here QH = 1 & QT = 0

Format	*Action*	*QH*	*QT*	*Binary*

Store string unsigned **stsu**



15	9	7	0
stsu 001111	d	addr0	

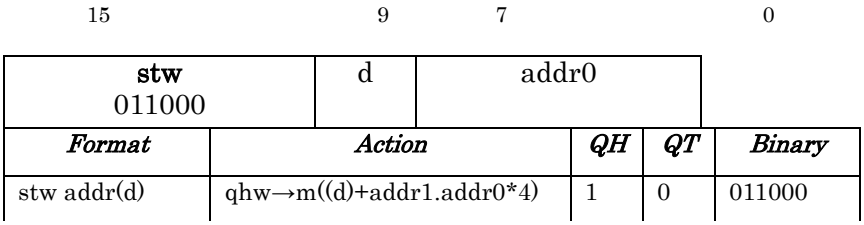
Format	*Action*	*QH*	*QT*	*Binary*
stsu addr(d)	qhw→m((d)+addr1.addr0*2)	1	0	001111

Description:

Store string unsigned to the operand queue pointed by the memory address ((d)+ addr0) or ((d) + addr0.addr1) from QH if the store instruction follows a convey instruction.

So, here QH = 1 & QT = 0

Store word **stw**



15	9	7	0
stw 011000	d	addr0	

Format	*Action*	*QH*	*QT*	*Binary*
stw addr(d)	qhw→m((d)+addr1.addr0*4)	1	0	011000

Description:

Store word to the operand queue pointed by the memory address ((d)+ addr0) or ((d) + addr0.addr1) from QH if the store instruction follows a convey instruction.

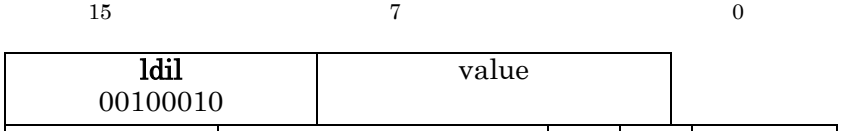
So, here QH = 1 & QT = 0

Immediate

Instruction Format:

8	8

Load immediate value ldil



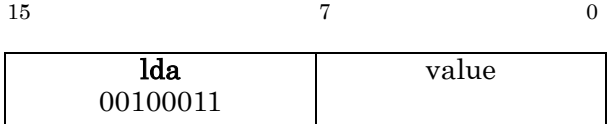
Format	*Action*	*QH*	*QT*	*Binary*
ldil value	qtw(0-7bit)←value)	0	1	00100010

Description:

Load immediate value to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr0.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

Load immediate address lda



Format	*Action*	*PQPcfo*	*PQPcf+*	*QH*	*QT*	*Binary*
ldia value	qtw← address of this instruction+value	Yes	Yes	0	1	00100011

Description:

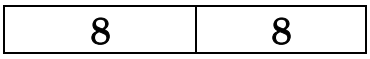
Load immediate address to the operand queue pointed by the QT from memory address ((d)+ addr0) or from ((d) + addr0.addr1) if the load instruction follows a convey instruction.

Here, QH = 0 & QT = 1

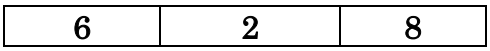
Control

Instruction Format

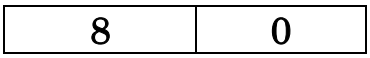
Branch



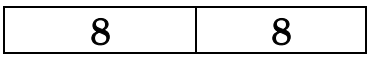
Jump



Interrupt, Barrier



Queue Control



Branch target **b t**

15	7	0
b 00000101	t	

Format	*Action*	*QH*	*QT*	*Binary*
b t	fc←(pc)+2(addr1.t)	0	0	00000101

Description:

This instruction to the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0

Branch equal **beq t**

15	7	0
beq 00000110	t	

Format	*Action*	*QH*	*QT*	*Binary*
beq t	fc←(pc)+2(addr1.t),if cc=eq	0	0	00000110

Description:

This instruction to the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 , '**CC**' means conditional code.

Branch not equal **bne**

15	7	0
bne 00000111	t	

Format	*Action*	*QH*	*QT*	*Binary*
bne t	fc←(pc)+2(addr1.t),if cc=ne	0	0	00000111

Description:

Branch not equal to target the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**CC**' means conditional code

Branch less than **blt t**

15	7	0
blt 00001000	t	

Format	*Action*	*QH*	*QT*	*Binary*
blt t	fc←(pc)+2(addr1.t),if cc=lt	0	0	00001000

Description:

This instruction to target the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**CC**' means conditional code

Branch less than or equal **ble t**

15	7	0
ble 00001001	t	

Format	*Action*	*QH*	*QT*	*Binary*
ble t	fc←(pc)+2(addr1.t),if cc=lte	0	0	00001001

Description:

This instruction to target the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**CC**' means conditional code

Branch greater than **bgt t**

15	7	0
bgt 00001010	t	

Format	*Action*	*QH*	*QT*	*Binary*
bgt t	fc←(pc)+2(addr1.t),if cc=gt	0	0	00001010

Description:

This instruction to target the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**CC**' means conditional code

Branch greater than or equal **bge t**

15	7	0
bge 00001011	t	

Format	*Action*	*QH*	*QT*	*Binary*
bge t	fc←(pc)+2(addr1.t),if cc=gt	0	0	00001011

Description:

This instruction to target the operand queue pointed by the program counter ((pc)+ t) or ((pc) + addr1.t) to FC if the branch instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**CC**' means conditional code

Stop QH move stpqh n

15	7	0
stpqh 01011100	n	

Format	*Action*	*QH*	*QT*	*Binary*
stpqh n	Stop QH moving	9	0	01011100

Description:

Stop QH move to target the operand queue pointed that the QH stop to move from n^{th} position.

Here QH = 9 & QT = 0

Stop LQH move stplqh n

15	7	0
stplqh 00000101	n	

Format	*Action*	*QH*	*QT*	*Binary*
stplqh n	Stop LQH moving	0	0	00000101

Description:

Stop LQH move to target the operand queue pointed that the LQH stop to move from n^{th} position.

Here QH = 0 & QT = 0

Fixed QH automatically autqh n

15	7	0
autqh 01011101	n	

Format	*Action*	*QH*	*QT*	*Binary*
autqh n	Fixed QH automatically	0	0	01011101

Description:

This instruction to target the operand queue pointed that the QH will be fixed in the current n^{th} position.

Here QH = 9 & QT = 0

Fixed LQH automatically autlqh n

15	7	0
autlqh 00000110	n	

Format	*Action*	*QH*	*QT*	*Binary*
autlqh n	Fixed LQH automatically	0	0	00000110

Description:

This instruction to target the operand queue pointed that the LQH will be fixed in the current n^{th} position.

Here QH = 9 & QT = 0

Jump jump t(a)

15	9	7	0
jump 00001100	a	t	

Format	*Action*	*QH*	*QT*	*Binary*
jump t(a)	fc←(a)+2(addr1.t)	0	0	00001100

Description:

This instruction to target the operand queue pointed by the memory address ((a)+ t) or ((a) + addr1.t) to FC if the jump instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**(a)**' means the content of address register '**a**'

Call call t(a)

15	9	7	0
call 00001101	a	t	

Format	*Action*	*QH*	*QT*	*Binary*
call t(a)	fc←(a)+2(addr1.t)	0	0	00001101

Description:

This instruction to target the operand queue pointed by the memory address ((a)+ t) or ((a) + addr1.t) to FC if the call instruction follows a convey instruction.

Here QH = 0 & QT = 0 '**(a)**' means the content of address register '**a**'

Return from call rfc

15	7	0
rfc 01010010		

Format	*Action*	*QH*	*QT*	*Binary*
rfc	fc←cra	0	0	01010010

Description:

This instruction to target the operand queue pointed by the current return address to FC.

Here QH = 0 & QT = 0

Return from interrupt rfi

15 7 0

rfi 01010001	

Format	*Action*	*PQPcfo*	*PQPcf+*	*QH*	*QT*	*Binary*
rfi	fc←(ira),Set	Yes	Yes	0	0	01010001

Description:

This instruction to target the operand queue pointed by the interrupt return address (ira) to FC .

Here QH = 0 & QT = 0

No operation **nop**

15 7 0

nop 00000000	

Format	*Action*	*PQPcfo*	*PQPcf+*	*QH*	*QT*	*Binary*
nop		Yes	Yes	0	0	00000000

Description:

nop is a dummy instruction that has no effect. It can be useful as an explicit 'do nothing' instruction.

Halt (stop) Stop **halt**

15 7 0

hlt 00000010				
Format	***Action***	***QH***	***QT***	***Binary***
hlt	Stop fetch,decode	0	0	00000010

Description:

This barrier instruction to target the operand queue pointed to stop fetching, to stop decoding.

Here QH = 0 & QT = 0

Barrier **bar**

15 7 0

bar 01100000				
Format	***Action***	***QH***	***QT***	***Binary***
bar		0	0	01100000

Description:

This barrier instruction to target the operand queue pointed that 'wait' until all the previous instructions are executed.

Here QH = 0 & QT = 0

Serial on/off **sonf**

15 7 0

sonf 01100001	on/off			
Format	***Action***	***QH***	***QT***	***Binary***
sonf	Serial On/Off Begin & end serial execution	0	0	01100001

Description:

This barrier instruction to target the operand queue pointed to execute serially on for serial on and multiple out of order for serial off.

Here QH = 0 & QT = 0

Switch (Program mode selctor) **type mode**

15 7 0

test 00000001	mode	
Format	***Action***	***Binary***
type mode	type 0 → Queue type 1 → Stack	00000001

Description:

Switch the execution mode by following the "mode" bit (0 indicates the Queue program and 1 indicates the Stack program).

ALU for Single Word

opcode	offset (n)
8	8

Add **add n**

15 7 0

add 00110000	n

Format	*Action*	*QH*	*QT*	*Binary*
add n	qtw0←qhw0+(qh+n)w Consumed instr. if c=0	1	1	00110000

Description:

Add two operands to the operand queue pointed by the QT from (qhw0+(qh+n)w).

Here QH = 1 & QT = 1

Add Unsigned **addu n**

15 7 0

addu 00110001	n

Format	*Action*	*QH*	*QT*	*Binary*
addu n	qtw0←qhw0+(qh+n)w	1	1	00110001

Description:

Add two unsigned operands to the operand queue pointed by the QT from (qhw0+(qh+n)w).

Here QH = 1 & QT = 1

Sub **sub n**

15	7	0
sub 00110010	n	

Format	*Action*	*QH*	*QT*	*Binary*
sub n	qtw0←qhw0-(qh+n)w	1	1	00110010

Description:

Subtract two operands to the operand queue pointed by the QT from (qhw0-(qh+n)w).

Here QH = 1 & QT = 1

Sub unsigned **subu n**

15	7	0
subu 00110011	n	

Format	*Action*	*QH*	*QT*	*Binary*
subu n	qtw0←qhw0-(qh+n)w	1	1	00110011

Description:

Subtract two unsigned operands to the operand queue pointed by the QT from (qhw0-(qh+n)w).

Here QH = 1 & QT = 1

Sub by order **subo n**

15 7 0

subo 00110100	n			
Format	***Action***	***QH***	***QT***	***Binary***
subo n	qtw0←(qh+n)w-qhw0	1	1	00110100

Description:
This instruction operands to the operand queue pointed by the QT subtract ((qh+n)-qhw0)w.
Here QH = 1 & QT = 1

Sub unsigned by order **subuo n**

15 7 0

subuo 00110101	n			
Format	***Action***	***QH***	***QT***	***Binary***
subuo n	qtw0←(qh+n)w - qhw0	1	1	00110101

Description:
This instruction operands to the operand queue pointed by the QT subtract from ((qh+n)- qhw0)w.
Here QH = 1 & QT = 1

Multiply **mul n**

15 7 0

Format	*Action*	*QH*	*QT*	*Binary*
mul n	qtw0←qhw0*(qh+n)w	1	1	00111011

mul 00111011	n

Description:

Multiply two operands to the operand queue pointed by the QT from (qhw0*(qh+n)w).

Multiply unsigned **mulu n**

15 7 0

mulu 00111100	n

Format	*Action*	*QH*	*QT*	*Binary*
mulu n	qtw0←qhw0+(qh+n)w	1	1	00111100

Description:

Multiply two unsigned operands to the operand queue pointed by the QT from (qhw0*(qh+n)w.

Here QH = 1 & QT = 1

Divide **div n**

15	7	0
div 00111101	n	

Format	*Action*	*QH*	*QT*	*Binary*
div n	qtw0←qhw0/(qh+n)w	1	1	00111101

Description:

Divide two operands to the operand queue pointed by the QT from (qhw0/(qh+n)w).

Here QH = 1 & QT = 1

Divide unsigned **divu n**

15	7	0
divu 00111110	n	

Format	*Action*	*QH*	*QT*	*Binary*
divu n	qtw0←qhw0/(qh+n)w	1	1	00111110

Description:

Divide two unsigned operands to the operand queue pointed by the QT from (qhw0/(qh+n)w).

Here QH = 1 & QT = 1

Divide by order **divo n**

15	7	0
divo 00111111	n	

Format	*Action*	*QH*	*QT*	*Binary*
divo n	qtw0←(qh+n)w / qhw0	1	1	00111111

Description:

This instruction to the operand queue pointed by the QT that divide qhw1 by qhw0 for consumed order instruction or ((qh+n)w/ qhw0).

Here QH = 1 & QT = 1

Divide unsigned by order **divuo n**

15	7	0
divuo 01000000	n	

Format	*Action*	*QH*	*QT*	*Binary*
divuo n	qtw0←(qh+n)w/qhw0	1	1	01000000

Description:

This instruction to the operand queue pointed by the QT that divide (qh+n)w by qhw0 .

Here QH = 1 & QT = 1

Modular **mod n**

15 7 0

mod 01000001	n			
Format	***Action***	***QH***	***QT***	***Binary***
mod n	rem(qhw0/(qh+n)w)	1	1	01000001

Description:

This instruction operands to the operand queue pointed that reminder of two operands to the QT from (qhw0/(qh+n)w).

Here QH = 1 & QT = 1

Modular by order **modo n**

15 7 0

modo n 01000011	n			
Format	***Action***	***QH***	***QT***	***Binary***
modo n	rem((qh+n)w/qhw0)	1	1	01000011

Description:

This instruction operands to the operand queue pointed that reminder of two operands to the QT ((qh+n)w/ qhw0).

Here QH = 1 & QT = 1

Modular unsigned **modu n**

15	7	0
modu 01000010	n	

Format	*Action*	*QH*	*QT*	*Binary*
modu p,n	rem(qhw0/(qh+n)w)	1	1	01000010

Description:

This instruction operands to the operand queue pointed that reminder of two unsigned operands to the QT from (qhw0/(qh+n)w).

Here QH = 1 & QT = 1

Modular unsigned by order **moduo n**

15	7	0
moduo 01000010	n	

Format	*Action*	*QH*	*QT*	*Binary*
moduo n	rem((qh+n)w/qhw0)	1	1	01000010

Description:

This instruction operands to the operand queue pointed that reminder of two unsigned operands to the QT from ((qh+n)w/ qhw0).

Here QH = 1 & QT = 1

And **and n**

15	7	0
and 00110111	n	

Format	*Action*	*QH*	*QT*	*Binary*
and n	qtw0←qhw0 and (qh+n)w	2	1	00110111

Description:

And two operands to the operand queue pointed by the QT from (qhw0 and (qh+n)w.

Here QH = 2 & QT = 1

Or **or n**

15	7	0
or 00111000	n	

Format	*Action*	*QH*	*QT*	*Binary*
or n	qtw0←qhw0 or (qh+n)w	1	1	00111000

Description:

Or two operands to the operand queue pointed by the QT from (qhw0 or (qh+n)w .

Here QH = 1 & QT = 1

Negative **neg n**

15 7 0

neg 00110110	n			
Format	***Action***	***QH***	***QT***	***Binary***
neg n	qtw0←-(qh+n)w	1	1	00110110

Description:

This instruction to the operand queue pointed by the QT from (qhw0← -(qh+n)w).

Here QH = 1 & QT = 1

Xor **xor n**

15 7 0

xor 00111001	n			
Format	***Action***	***QH***	***QT***	***Binary***
xor n	qtw0←qhw0 xor (qh+n)w	1	1	00111001

Description:

Xor two operands to the operand queue pointed by the QT from (qhw0 xor (qh+n)w).

Here QH = 1 & QT = 1

Not **not n**

15 7 0

not 00111010	n

Format	*Action*	*QH*	*QT*	*Binary*
not n	qtw0←not((qh+n)w)	1	1	00111010

Description:

This instruction to the operand queue pointed by the QT from (qhw0←not((qh+n)w)).

Here QH = 1 & QT = 1

Shift and rotate Instruction

Shift Instruction Format:

8	4	4

Rotate Instruction Format:

8	8

Right Shift **sru s,n**

15	7	3	0
sru 01001011	s	n	

Format	*Action*	*QH*	*QT*	*Binary*
sru p,s,n	qtw0←logical right shift(qh+n)w	1	1	01001011

Description:

This instruction to the operand queue pointed by the QT that logically right shift (qhw0←(qh+n)w.

Here QH = 1 & QT = 1

Left Shift **slu s,n**

15	7	3	0
slu 01001100	s	n	

Format	*Action*	*QH*	*QT*	*Binary*
slu s,n	qtw0←logical right shift(qh+n)w	1	1	01001100

Description:

This instruction to the operand queue pointed by the QT that logically left shift from (qhw0←(qh+n)w.

Here QH = 1 & QT = 1

Right Shift (Arithmetically) **sr n**

15	7	0
sr 01001101	n	

Format	*Action*	*QH*	*QT*	*Binary*
sr n	qtw0←Arithmetic right shift(qh+n)w	1	1	01001101

Description:

This instruction to the operand queue pointed by the QT that arithmetically from (qhw0←(qh+n)w.

Here QH = 1 & QT = 1

Rotate left **rol n**

15	7	0
rol 01001111	n	

Format	*Action*	*QH*	*QT*	*Binary*
rot n	Rotate left	1	1	01001111

Description:

This instruction to the operand queue pointed by the QT that rotates left.

Here QH = 1 & QT = 1

Rotate Right **ror n**

15	7	0
ror 01001110	n	

Format	*Action*	*QH*	*QT*	*Binary*
ror p,s,n	Rotate right	1	1	01001110

Description:

This instruction to the operand queue pointed by the QT that rotates right.

Here QH = 1 & QT = 1

Duplicate **dup n**

15	7	0
dup 00100111	n	

Format	*Action*	*QH*	*QT*	*Binary*
dup n	qtw0,,,,qtw(2c-1)←q(qh+n)	1	1	00100111

Description:

This instruction to the operand queue pointed by the QT that rotate & duplicate from (qhw0,,,,qtw(2c-1) ← q(qh+n)).

Here QH = 1 & QT = 1

Move **mov n**

15 7 0

mov 00101000	n

Format	*Action*	*QH*	*QT*	*Binary*
dup n	qtw0,,,,qtw(2c-1)←q(qh+n)	1	1	00101000

Description:

This instruction to the operand queue pointed by the QT that move from ((qh+n)w) for produced order instruction.

Here QH = 1 & QT = 1

ALU for Double Word

opcode	offset (n)
8	8

Add double **addd n**

15 7 0

addd n 11100000	n

Format	*Action*	*QH*	*QT*	*Binary*
addd n	qtd0←qhd0+(qh+n)d	2	2	11100000

Description:

Add two double operands to the operand queue pointed by the QT from (qhd0+(qh+n)d).

Here QH = 2 & QT = 2

Add double Unsigned **adddu n**

15 7 0

adddu 11100001	n

Format	*Action*	*QH*	*QT*	*Binary*
adddu n	qtd0←qhd0+(qh+n)d	2	2	11100001

Description:

Add two double unsigned operands to the operand queue pointed by the QT from (qhd0+(qh+n)d).

Here QH = 2 & QT = 2

Sub double **subd n**

15	7	0
subd 11100010	n	

Format	*Action*	*QH*	*QT*	*Binary*
subd n	qtd0←qhd0-(qh+n)d	2	2	11100010

Description:

Subtract two double operands to the operand queue pointed by the QT from (qhd0-(qh+n)d).

Here QH = 2 & QT = 2

Sub double unsigned **subdu n**

15	7	0
subdu 11100000	n	

Format	*Action*	*QH*	*QT*	*Binary*
subdu n	qtd0←qhd0-(qh+n)d	2	2	11100000

Description:

Subtract two unsigned operands to the operand queue pointed by the QT (qhd0-(qh+n)d).

Here QH = 2 & QT = 2

Sub double by order **subdo n**

15 7 0

subdo 11100100	n			
Format	***Action***	***QH***	***QT***	***Binary***
subdo n	qtd0←(qh+n)d - qhd0	2	2	11100100

Description:

This instruction operands to the operand queue pointed by the QT subtract from (qh+n)d- qhd0).

Here QH = 2 & QT = 2

Sub unsigned double by order **subduo n**

15 7 0

subduo 11100101	n			
Format	***Action***	***QH***	***QT***	***Binary***
subduo n	qtd0←(qh+n)d-qhd0	2	2	11100101

Description:

This instruction operands to the operand queue pointed by the QT subtract from ((qh+n)d- qhd0).

Here QH = 2 & QT = 2

Multiply double **muld n**

15	7	0
muld 11100110	n	

Format	*Action*	*QH*	*QT*	*Binary*
muld n	qtd0←qhd0+(qh+n)d	2	2	11100110

Description:

Multiply two operands to the operand queue pointed by the QT from (qhd0*(qh+n)d) for produced order instruction.

Here QH = 2 & QT = 2

Multiply unsigned double **muldu n**

15	7	0
muldu 11100111	n	

Format	*Action*	*QH*	*QT*	*Binary*
muldu n	qtd0←qhd0+(qh+n)d	2	2	11100111

Description:

Multiply two unsigned double operands to the operand queue pointed by the QT from (qhd0*(qh+n)d).

Here QH = 2 & QT = 2

Divide double **divd n**

15 7 0

divd p,n 11101000				
Format	***Action***	***QH***	***QT***	***Binary***
divd n	qtd0←qhd0/(qh+n)d	2	2	11101000

Description:
Divide two operands to the operand queue pointed by the QT from (qhd0/(qh+n)d).

Here QH = 2 & QT = 2

Divide double unsigned **divdu n**

15 7 0

divu 11100000	n			
Format	***Action***	***QH***	***QT***	***Binary***
divdu n	qtd0←qhd0/(qh+n)d	2	2	11100000

Description:
Divide two unsigned double operands to the operand queue pointed by the QT from (qhd0/qhd1) for consumed order instruction or (qhd0/(qh+n)d) for produced order instruction.
Here QH = 2 & QT = 2

Divide double by order **divdo n**

15 7 0

divdo 11101010	n

Format	*Action*	*QH*	*QT*	*Binary*
divdo n	qtd0←(qh+n)d/ qhd0	2	2	11101010

Description:

This instruction to the operand queue pointed by the QT that divide (qh+n)d by qhd0.

Here QH = 2 & QT = 2

Divide unsigned double by order **divduo n**

15 7 0

divduo 11101011	n

Format	*Action*	*QH*	*QT*	*Binary*
divduo n	qtd0←(qh+n)d/qhd0	2	2	11101011

Description:

This instruction to the operand queue pointed by the QT that divide (qh+n)d by qhd0.

Here QH = 2 & QT = 2

Modular double **modd n**

15 7 0

modd 11101100	n			
Format	***Action***	***QH***	***QT***	***Binary***
modd n	rem(qhd0/(qh+n)d)	2	2	11101100

Description:

This instruction operands to the operand queue pointed that reminder of two operands to the QT from (qhd0/(qh+n)d).

Here QH = 2 & QT = 2

Modular double by order **moddo n**

15 7 0

moddo 11101110	n			
Format	***Action***	***QH***	***QT***	***Binary***
moddo n	rem((qh+n)d/qhd0)	2	2	11101110

Description:

This instruction operands to the operand queue pointed that reminder of two operands to the QT from ((qh+n)d/ qhd0).

Here QH = 4 & QT = 2

Modular double unsigned **moddu n**

15 7 0

moddu 11101101	n			
Format	***Action***	***QH***	***QT***	***Binary***
moddu n	rem(qhd0/(qh+n)d)	2	2	11101101

Description:

This instruction operands to the operand queue pointed that reminder of two unsigned operands to the QT from (qhd0/(qh+n)d).

Here QH = 2 & QT = 2

Modular unsigned double by order **modduo n**

15 7 0

modduo 11101111	n			
Format	***Action***	***QH***	***QT***	***Binary***
modduo n	rem((qh+n)d/qhd0)	2	2	11101111

Description:

This instruction operands to the operand queue pointed that reminder of two unsigned operands to the QT from ((qh+n)d/ qhd0).

Here QH = 4 & QT = 2

And double **andd n**

15 7 0

andd 11110000	n			
Format	***Action***	***QH***	***QT***	***Binary***
andd n	qtd0←qhd0 and (qh+n)d	2	2	11110000

Description:

And two double operands to the operand queue pointed by the QT from (qhd0 and (qh+n)d.

Here QH = 2 & QT = 2

Or double **ord n**

15 7 0

ord 11110001	n			
Format	***Action***	***QH***	***QT***	***Binary***
ord n	qtd0←qhd0 or (qh+n)d	2	2	11110001

Description:

Or two double operands to the operand queue pointed by the QT from (qh+n)d.

Here QH = 2 & QT = 2

Negative double **negd n**

15	7	0
negd 10110000	n	

Format	*Action*	*QH*	*QT*	*Binary*
negd n	qtd0←-(qh+n)d	2	2	10110000

Description:

This instruction to the operand queue pointed by the QT from (qhd0← -(qh+n)d) .

Here QH = 2 & QT = 2

Xor double **xord n**

15	7	0
xord 11110010	n	

Format	*Action*	*QH*	*QT*	*Binary*
xord n	qtd0←qhd0 xor (qh+n)d	2	2	11110010

Description:

Xor two double operands to the operand queue pointed by the QT from (qhw0 xor (qh+n)w).

Here QH = 2 & QT = 2

Not double **notd n**

15 7 0

notd 10110001	n

Format	*Action*	*QH*	*QT*	*Binary*
notd n	qtd0←not((qh+n)d)	2	2	10110001

Description:

This instruction to the operand queue pointed by the QT from (qhw0←not((qh+n)w)).

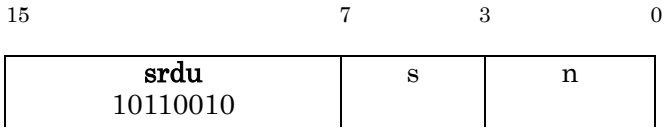
Here QH = 2 & QT = 2

Shift Instruction

Instruction Format:

8	4	4

Right Shift for double **srdu s,n**



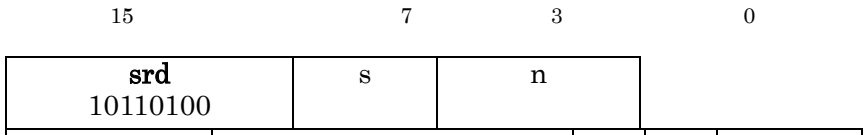
Format	*Action*	*QH*	*QT*	*Binary*
srdu s,n	qtd0←logical right shift(qh+n)d	2	2	10110010

Description:

This instruction to the operand queue pointed by the QT that logically right shift from (qhd0←(qh+n)d .

Here QH = 2 & QT = 2

Right Shift (Arithmetically) for double **srd s,n**



Format	*Action*	*QH*	*QT*	*Binary*
srd s,n	qtd0←Arithmetic right shift(qh+n)d	2	2	10110100

Description:

This instruction to the operand queue pointed by the QT that arithmetically from (qhd0←(qh+n)d).

Here QH = 2 & QT = 2

Rotate left for double **rold n**

15 7 0

rold 10110101	n

Format	*Action*	*QH*	*QT*	*Binary*
rold n	Rotate left	2	2	10110101

Description:

This instruction to the operand queue pointed by the QT that rotate left.

Here QH = 2 & QT = 2

Rotate Right for double **rord n**

15 7 0

rord 10111010	n

Format	*Action*	*QH*	*QT*	*Binary*
rord n	Rotate right	2	2	10111010

Description:

This instruction to the operand queue pointed by the QT that rotate right.

Here QH = 2 & QT = 2

Rotate & duplicate for double **dupd n**

15 7 0

dup 01011001	n

Format	*Action*	*QH*	*QT*	*Binary*
dupd n	qtd0,,,,qtd(2c-1)←q(qhd+n)	2	2	01011001

Description:

This instruction to the operand queue pointed by the QT that rotate & duplicate from (qhd0,,,,qtd(2c-1) ← q(qh+n)).

Here QH = 2 & QT = 2