The Queue Computer Project

Instruction Set Architecture

Ben Abdallah Abderazek,
Jan. 2003
COVOP

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>covop</td>
<td>Convey an address</td>
<td>0</td>
<td>0</td>
<td>00000100</td>
</tr>
<tr>
<td>addr1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Instruction format

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
Load byte

\[ \text{ldb} \quad \text{ldb} \]

\[
\begin{array}{c|c|c|c|c|c}
\text{Format} & \text{Action} & \text{QH} & \text{QT} & \text{Binary} \\
\hline
\text{ldb addr0(d)} & \text{qtw} ← \text{m}((d)+\text{addr0}) & 0 & 1 & 010000 \\
\end{array}
\]

**Description:**
Load byte to the operand queue pointed by the QT from memory address \((d)+\text{addr0})\) or from \((d) + \text{addr1.addr1}) if the load instruction follows a convey instruction.
Here, \(\text{QH} = 0 \& \text{QT} = 1\)

Load byte unsigned

\[ \text{ldbu} \quad \text{ldbu} \]

\[
\begin{array}{c|c|c|c|c|c}
\text{Format} & \text{Action} & \text{QH} & \text{QT} & \text{Binary} \\
\hline
\text{ldbu addr0(d)} & \text{qtw} ← \text{m}((d)+\text{addr1.addr0}) & 0 & 1 & 010001 \\
\end{array}
\]

**Description:**
Load byte unsigned to the operand queue pointed by the QT from memory address \((d)+\text{addr0})\) or from \((d) + \text{addr1.addr1}) if the load instruction follows a convey instruction.
Here, \(\text{QH} = 0 \& \text{QT} = 1\)
Load string

<table>
<thead>
<tr>
<th>lds</th>
<th>d</th>
<th>addr0</th>
</tr>
</thead>
<tbody>
<tr>
<td>010010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>ldsu</th>
<th>d</th>
<th>addr0</th>
</tr>
</thead>
<tbody>
<tr>
<td>010011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

\[ \text{ldw} \]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldw addr0(d)</td>
<td>qtw←m((d)+addr1.addr0*4)</td>
<td>0</td>
<td>1</td>
<td>010100</td>
</tr>
</tbody>
</table>

**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

\[ \text{stb} \]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>stb addr(d)</td>
<td>qhw→m((d)+addr1.addr0)</td>
<td>1</td>
<td>0</td>
<td>010110</td>
</tr>
</tbody>
</table>

**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 \(QH = 1\) & \(QT = 0\)
**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**

<table>
<thead>
<tr>
<th><strong>Format</strong></th>
<th><strong>Action</strong></th>
<th><strong>QH</strong></th>
<th><strong>QT</strong></th>
<th><strong>Binary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>stbu addr(d)</td>
<td>qhw→m((d)+addr1.addr0)</td>
<td>1</td>
<td>0</td>
<td>001110</td>
</tr>
</tbody>
</table>

**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

**Store string unsigned**

<table>
<thead>
<tr>
<th><strong>Format</strong></th>
<th><strong>Action</strong></th>
<th><strong>QH</strong></th>
<th><strong>QT</strong></th>
<th><strong>Binary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>stsu addr(d)</td>
<td>qhw→m((d)+addr1.addr0*2)</td>
<td>1</td>
<td>0</td>
<td>010111</td>
</tr>
</tbody>
</table>
**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**

<table>
<thead>
<tr>
<th>15</th>
<th>9</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>stw</td>
<td>d</td>
<td>addr0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>stw addr(d)</td>
<td>qhw→m((d)+addr1.addr0*2)</td>
<td>1</td>
<td>0</td>
<td>001111</td>
</tr>
</tbody>
</table>

**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 \(QH = 1 \& QT = 0\)
Load immediate value

**ldil**

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldil value</td>
<td>qtw(0-7bit)</td>
<td>→</td>
<td>value)</td>
<td>0</td>
</tr>
</tbody>
</table>

**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**
**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

8 8

Jump

6 2 8

Interrupt, Barrier

8 0

Queue Control

8 8

Branch target bt
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

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.
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

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>bne t</td>
<td>fc←(pc)+2(addr1.t),if cc=ne 0 0 0000111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>blt t</td>
<td>fc←(pc)+2(addr1.t),if cc=lt 0 0 00001000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

\[
\begin{array}{c|c|c|c|c}
\textbf{ble} & \textbf{t} & \textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
00001001 & & fc←(pc)+2(addr1.t),if cc=lte & 0 & 0 & 00001001 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
\textbf{bgt} & \textbf{t} & \textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
00001010 & & fc←(pc)+2(addr1.t),if cc=gt & 0 & 0 & 00001010 \\
\end{array}
\]

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

\[
\begin{array}{c|c|c|c|c}
\textbf{bge} & \textbf{t} & \textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
00001010 & & fc←(pc)+2(addr1.t),if cc=lte & 0 & 0 & 00001010 \\
\end{array}
\]
**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

```
 15 7 0
```

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>stpqh n</td>
<td>Stop QH moving</td>
<td>9</td>
<td>0</td>
<td>01011100</td>
</tr>
</tbody>
</table>

**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

```
 15 7 0
```

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>stplqh n</td>
<td>Stop LQH moving</td>
<td>9</td>
<td>0</td>
<td>01011100</td>
</tr>
</tbody>
</table>

---
### 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  
\textbf{autqh n}

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>autqh</td>
<td>Fixed QH automatically</td>
<td>0</td>
<td>0</td>
<td>01011101</td>
</tr>
</tbody>
</table>

### 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  
\textbf{autlqh n}

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>stplqh</td>
<td>Stop LQH moving</td>
<td>0</td>
<td>0</td>
<td>00000101</td>
</tr>
</tbody>
</table>
Description:
This instruction to target the operand queue pointed that the LQH will be fixed in the current n\textsuperscript{th} position.
Here QH = 9 & QT = 0

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
\hline
\textbf{autlqh n} & Fixed LQH automatically & 0 & 0 & 00000110 \\
\hline
\end{tabular}
\end{center}

Jump

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
\hline
\textbf{jump t(a)} & fc←(a)+2(addr1.t) & 0 & 0 & 00001100 \\
\hline
\end{tabular}
\end{center}

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)
Description:
This instruction to target the operand queue pointed by the memory address \((a) + t\) or \((a) + \text{addr1.t}\) 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

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Format} & \text{Action} & QH & QT \\
\hline
\text{call } t(a) & fc\leftarrow (a)+2(\text{addr1.t}) & 0 & 0 \\
\hline
\end{array}
\]

\text{Return from interrupt}

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Format} & \text{Action} & QH & QT \\
\hline
\text{rfi} & fc\leftarrow \text{rca} & 0 & 0 \\
\hline
\end{array}
\]
**Description:**
This instruction to target the operand queue pointed by the interrupt return address (ira) to FC.
Here \( QH = 0 \) & \( QT = 0 \)

**No operation**

**Description:**
\textbf{nop} is a dummy instruction that has no effect. It can be useful as an explicit 'do nothing' instruction.

**Halt (stop)**

Stop

\textbf{halt}
Description:
This barrier instruction to target the operand queue pointed to stop fetching, to stop decoding.
Here QH = 0 & QT = 0

Serial on/off

Description:
This barrier instruction to target the operand queue pointed that ‘wait’ until all the previous instructions are executed.
Here QH = 0 & QT = 0
**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 selector)**

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonf</td>
<td>Serial On/Off Begin &amp; end serial execution</td>
<td>0</td>
<td>0</td>
<td>01100001</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>opcode</th>
<th>offset (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Add \( \text{add} \ n \)
**Description:**
Add two operands to the operand queue pointed by the QT from \((qhw0+(qh+n)w)\).
Here \(QH = 1\) & \(QT = 1\).

### Add Unsigned  **\texttt{addu n}**

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>(QH)</th>
<th>(QT)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>add u</td>
<td>qtw0←qhw0+(qh+n)w Consumed instr. if (c=0)</td>
<td>1</td>
<td>1</td>
<td>00110000</td>
</tr>
</tbody>
</table>

### Description:
Add two unsigned operands to the operand queue pointed by the QT from \((qhw0+(qh+n)w)\).
Here \(QH = 1\) & \(QT = 1\).
Description:
Subtract two operands to the operand queue pointed by the QT from \((qhw0 - (qh+n)w)\).
Here \(QH = 1\) & \(QT = 1\)

Sub unsigned \(\text{subu } n\)

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

Here \(QH = 1\) & \(QT = 1\)

**Sub unsigned by order**

\[
\text{subuo } n
\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>subuo n</td>
<td>qtw0—(qh+n)w - qhw0</td>
<td>1</td>
<td>1</td>
<td>00110101</td>
</tr>
</tbody>
</table>

**Description:**
This instruction operands to the operand queue pointed by the QT subtract from \((qh+n) - qhw0\).

Here \(QH = 1\) & \(QT = 1\)

**Multiply**

\[
\text{mul } n
\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>mul n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiply unsigned

\[ \text{mulu } n \]

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

Here \(QH = 1\) & \(QT = 1\)

Divide

\[ \text{div } n \]
Description:
Divide two operands to the operand queue pointed by the QT from \((qhw0/(qh+n)w)\).
Here \(QH = 1\) & \(QT = 1\)

Divide unsigned \(\text{divu } n\)

\[
\begin{array}{c|c|c|c|c}
\text{divu} & n \\
00111110 & \\
\hline
\text{Format} & \text{Action} & QH & QT & Binary \\
\hline
\text{divu } n & qtw0\leftarrow qhw0/(qh+n)w & 1 & 1 & 00111110 \\
\end{array}
\]

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 \(\text{divo } n\)

\[
\begin{array}{c|c|c|c|c}
\text{divo} & n \\
00111101 & \\
\hline
\text{Format} & \text{Action} & QH & QT & Binary \\
\hline
\text{divo } n & qtw0\leftarrow qhw0/(qh+n)w & 1 & 1 & 00111101 \\
\end{array}
\]

**Description:**
This instruction to the operand queue pointed by the QT that divide qhw1 by qhw0 for consumed order instruction or \((qhw+1)/qhw0\).

Here QH = 1 & QT = 1

**Divide unsigned by order**

\[\texttt{divuo n}\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>divuo n</td>
<td>qtw0(\rightarrow) qhw0</td>
<td>1</td>
<td>1</td>
<td>01000000</td>
</tr>
</tbody>
</table>

**Description:**
This instruction to the operand queue pointed by the QT that divide (qhw+1)w by qhw0.
Here QH = 1 & QT = 1

**Modular**

\[\texttt{mod n}\]
Description:
This instruction operands to the operand queue pointed that reminder of two operands to the QT from \((qh\times n)w\). Here \(QH = 1\) & \(QT = 1\)

Modular by order \(\text{modo } n\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>(QH)</th>
<th>(QT)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{modo } n) (01000011)</td>
<td>(\text{rem}(qhw0/(qh\times n)w))</td>
<td>1</td>
<td>1</td>
<td>(01000001)</td>
</tr>
</tbody>
</table>

Description:
This instruction operands to the operand queue pointed that reminder of two operands to the QT \((qh\times n)w\). Here \(QH = 1\) & \(QT = 1\)

Modular unsigned \(\text{modu } n\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>(QH)</th>
<th>(QT)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{modu } n) (01000011)</td>
<td>(\text{rem}((qh+n)w/qhw0))</td>
<td>1</td>
<td>1</td>
<td>(01000011)</td>
</tr>
</tbody>
</table>

Description:
This instruction operates on 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

\[\text{moduo } n\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>modulo n</td>
<td>rem(qhw0/(qh+n)w)</td>
<td>1</td>
<td>1</td>
<td>01000010</td>
</tr>
</tbody>
</table>

Description:
This instruction operates on the operand queue pointed that reminder of two unsigned operands to the QT from (qh+n)w/qhw0).
Here QH = 1 & QT = 1

And

\[\text{and } n\]
Description:
And two operands to the operand queue pointed by the QT from (qhw0 and (qh+n)w.
Here QH = 2 & QT = 1

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>and n</td>
<td>qtw0←qhw0 and (qh+n)w</td>
<td>2</td>
<td>1</td>
<td>00110111</td>
</tr>
</tbody>
</table>

Description:
Or two operands to the operand queue pointed by the QT from (qhw0 or (qh+n)w.
Here QH = 1 & QT = 1

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>or n</td>
<td>qtw0←qhw0 or (qh+n)w</td>
<td>1</td>
<td>1</td>
<td>00111000</td>
</tr>
</tbody>
</table>

Negative

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

Xor

Description:
Xor two operands to the operand queue pointed by the QT from (qh0 xor (qh+n)w).
Here QH = 1 & QT = 1

Not
**Description:**
This instruction to the operand queue pointed by the QT from \( q_{hw0} \leftarrow \text{not}((q_{h}+n)w) \).
Here \( QH = 1 \) & \( QT = 1 \)

### Shift and rotate Instruction

#### Shift Instruction Format:

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>not n</td>
<td>qtw0←not((qh+n)w)</td>
<td>1</td>
<td>1</td>
<td>00111010</td>
</tr>
</tbody>
</table>

#### Rotate Instruction Format:

Right Shift \( \text{sru s,n} \)
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

\[ \texttt{slu s,n} \]

\[
\begin{array}{c|c|c|c|c}
<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>slu s,n</td>
<td>qtw0←logical shift(qh+n)w</td>
<td>1</td>
<td>1</td>
<td>01001100</td>
</tr>
</tbody>
</table>
\end{array}
\]

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)

\[ \texttt{sr n} \]

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

\[ \text{sr n} \]
\[
\begin{array}{|c|c|c|c|}
\hline
\text{Format} & \text{Action} & \text{QH} & \text{QT} \\
\hline
\text{sr n} & \text{Arithmetic right shift(qh+n)w} & 1 & 1 \\
\hline
\end{array}
\]

Description:
This instruction to the operand queue pointed by the QT that rotates left.
Here QH = 1 & QT = 1

Rotate Right

\[ \text{rol n} \]
\[
\begin{array}{|c|c|c|c|}
\hline
\text{Format} & \text{Action} & \text{QH} & \text{QT} \\
\hline
\text{rol n} & \text{Rotate left} & 1 & 1 \\
\hline
\end{array}
\]
Description:
This instruction to the operand queue pointed by the QT that rotates right.
Here QH = 1 & QT = 1

Duplicate

dup n

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
**Description:**
This instruction to the operand queue pointed by the QT that move from \((qh+n)w\) for produced order instruction.
Here \(QH = 1\) & \(QT = 1\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>(QH)</th>
<th>(QT)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>dup (n)</td>
<td>(qtw0,\ldots,qtw(2c-1)\rightarrow q(qh+n))</td>
<td>1</td>
<td>1</td>
<td>00101000</td>
</tr>
</tbody>
</table>
ALU for Double Word

<table>
<thead>
<tr>
<th>opcode</th>
<th>offset (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Add double addd n
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 \( \text{addu} \ n \)

\[
\begin{array}{cccc}
\text{addu} & 11100001 & n \\
\hline
\text{Format} & \text{Action} & QH & QT & \text{Binary} \\
\hline
\text{addu} n & \text{qtd}0\rightarrow\text{qhd}0+(\text{qh}+n)d & 2 & 2 & 11100001
\end{array}
\]

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 \( \text{subd} \ n \)
Description:
Subtract two double operands to the operand queue pointed by the QT from \((\text{qhd}_0 - (\text{qh}+n)_d)\).
Here \(\text{QH} = 2\) & \(\text{QT} = 2\)

Sub double unsigned \(\text{subdu n}\)

Description:
Subtract two unsigned operands to the operand queue pointed by the QT from \((\text{qhd}_0 - (\text{qh}+n)_d)\).
Here \(\text{QH} = 2\) & \(\text{QT} = 2\)

Sub double by order \(\text{subdo n}\)
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

\texttt{subdo n}

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
\texttt{Format} & \texttt{Action} & \texttt{QH} & \texttt{QT} & \texttt{Binary} \\
\hline
\texttt{subdo n} & qtd0 - (qh+n)d - qhd0 & 2 & 2 & 11100100 \\
\hline
\end{tabular}
\end{center}

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

\texttt{muld n}

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 \(\text{muldu } n\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>muldu n</td>
<td>qtd0→qhd0+(qh+n)d</td>
<td>2</td>
<td>2</td>
<td>11100111</td>
</tr>
</tbody>
</table>

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 \(\text{divd } n\)
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 \(\text{divdu } n\)

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

Divide double by order \(\text{divdo } n\)
Description:
This instruction to the operand queue pointed by the QT that divide \((q_h+n)d\) by \(q_{hd0}\).

Here \(Q_H = 2\) & \(Q_T = 2\)

Divide unsigned double by order \textbf{divduo n}

\[
\begin{array}{c|c|c|c|c}
\text{Format} & \text{Action} & Q_H & Q_T & \text{Binary} \\
\hline
\text{divduo n} & qtd0\leftarrow(q_h+n)d/q_{hd0} & 2 & 2 & 11101011 \\
\end{array}
\]

Description:
This instruction to the operand queue pointed by the QT that divide \((q_h+n)d\) by \(q_{hd0}\).

Here \(Q_H = 2\) & \(Q_T = 2\)

Modular double \textbf{modd n}
**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**  \[moddo\ n\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>moddo</td>
<td>(n)</td>
<td>2</td>
<td>2</td>
<td>11101110</td>
</tr>
</tbody>
</table>

**Description:**
This instruction operands to the operand queue pointed that reminder of two operands to the QT from \((qh+n)/qhd0)\).
Here \(QH = 4 \& QT = 2\)

**Modular double unsigned**  \[moddu\ n\]

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>moddu</td>
<td>(n)</td>
<td>2</td>
<td>2</td>
<td>11101110</td>
</tr>
</tbody>
</table>
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 \(\text{modduo } n\)

\[
\begin{array}{c|c|c|c|c}
\text{Format} & \text{Action} & QH & QT & \text{Binary} \\
\hline
\text{modduo } n & \text{rem}(qhd0/(qh+n)d) & 2 & 2 & 11101101 \\
\end{array}
\]

Description:
This instruction operands to the operand queue pointed that reminder of two unsigned operands to the QT from \((qh0/(qh+n)d)\).
Here \(QH = 4\) & \(QT = 2\)

And double \(\text{andd } n\)
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

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ord n</td>
<td>qtd0←qhd0 or (qh+n)d</td>
<td>2</td>
<td>2</td>
<td>11110001</td>
</tr>
</tbody>
</table>

Description:
Or two double operands to the operand queue pointed by the QT from (qh+n)d.
Here QH = 2 & QT = 2

Negative double

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>negd n</td>
<td></td>
<td>2</td>
<td>2</td>
<td>11110000</td>
</tr>
</tbody>
</table>
Description:
This instruction to the operand queue pointed by the QT from \((qhd_0 ← (qh+n)d)\).

Here \(QH = 2\) & \(QT = 2\)

Xor double \hspace{1cm} \textbf{xord} \hspace{1cm} n

\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
\hline
\text{negd} n & qtd0 ← (qh+n)d & 2 & 2 & \text{10110000} \\
\text{xord} n & qtd0 ← qhd0 xor (qh+n)d & 2 & 2 & \text{11110010} \\
\hline
\end{tabular}
\end{center}

Description:
Xor two double operands to the operand queue pointed by the QT from \((qhw_0 \ xor \ (qh+n)w)\).
Here \(QH = 2\) & \(QT = 2\)

Not double \hspace{1cm} \textbf{notd} \hspace{1cm} n

\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Format} & \textbf{Action} & \textbf{QH} & \textbf{QT} & \textbf{Binary} \\
\hline
\text{negd} n & qtd0 ← (qh+n)d & 2 & 2 & \text{10110000} \\
\text{xord} n & qtd0 ← qhd0 xor (qh+n)d & 2 & 2 & \text{11110010} \\
\hline
\end{tabular}
\end{center}
### Description:
This instruction to the operand queue pointed by the QT from \((qhw0\leftarrow\text{not}((qh+n)w))\). Here \(QH = 2\) & \(QT = 2\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>notd n</td>
<td>qtd0←not((qh+n)d)</td>
<td>2</td>
<td>2</td>
<td>10110001</td>
</tr>
</tbody>
</table>
Shift Instruction

Instruction Format:

| 8 | 4 | 4 |

Right Shift for double \textit{srdu s,n}

5

Description:
This instruction to the operand queue pointed by the QT that logically right shift from \((\text{qhd}0 \leftarrow (\text{qh}+\text{n})d)\).
Here \(\text{QH} = 2\) & \(\text{QT} = 2\)

**Right Shift (Arithmetically) for double** \(\text{srd \ s, n}\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>srd s,n</td>
<td>qtd0(\leftarrow)Arithmetic right shift((\text{qh}+\text{n})d)</td>
<td>2</td>
<td>2</td>
<td>10110100</td>
</tr>
</tbody>
</table>

Description:
This instruction to the operand queue pointed by the QT that arithmetically from \((\text{qhd}0 \leftarrow (\text{qh}+\text{n})d)\).
Here \(\text{QH} = 2\) & \(\text{QT} = 2\)

**Rotate left for double** \(\text{rold \ n}\)

**Description:**
This instruction to the operand queue pointed by the QT that rotate left. Here QH = 2 & QT = 2

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>rold n</td>
<td>Rotate left</td>
<td>2</td>
<td>2</td>
<td>10110101</td>
</tr>
</tbody>
</table>

**Rotate Right for double**

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>rord n</td>
<td>Rotate right</td>
<td>2</td>
<td>2</td>
<td>10110101</td>
</tr>
</tbody>
</table>

**Description:**
This instruction to the operand queue pointed by the QT that rotate right. Here QH = 2 & QT = 2

**Rotate & duplicate for double**

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>QH</th>
<th>QT</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>dupd n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Description:
This instruction to the operand queue pointed by the QT that rotate & duplicate from \((q_{hd0}, \ldots, q_{td(2c-1)} ← q(qh+n))\).

Here \(QH = 2\) & \(QT = 2\)

<table>
<thead>
<tr>
<th>Format</th>
<th>Action</th>
<th>(QH)</th>
<th>(QT)</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>dupd n</td>
<td>qtd0,\ldots,qtd(2c-1)←q(qhd+n)</td>
<td>2</td>
<td>2</td>
<td>01011001</td>
</tr>
</tbody>
</table>