

# SOMMAIRE

SOMMAIRE	1
UNIT	2

# UNIT CPU

Unit ID	Unit Name			
	Instruction 0	Instruction 1	Instruction 2	Instruction 3
0	BRU/CMP	AGU	----	----
1	LSU1	LSU2	----	----
2	ALU1/DIV	ALU2	ALU3	ALU4
3	VPU1/PDIV	VPU2	----	----

## **LDDMA : Load DMA**

---

Load 32 bytes into RAM or L2 / L3 cache and put it in D-SRAM

### **Operation Code :**

31	20	9	8	5	4	4	3	2	1	0
Source 4 Immediate	Source 3 Immediate	Source 2 Register	Source 1 Register	LDDMA 00	Unit Value					
11 bits	12 bits	4 bits	1 bits	2 bits	2 bits					

### **Operation :**

SP\_MEMORY( (IMM3x32) +RS1) = RAM( (IMM4x32)+RS2)

### **Latency :**

--- cycles

### **Example :**

lddma 0xFFFF[r0],100[r15]

## STDMA : Store DMA

---

Sends 32 bytes in D-SRAM in L2 / L3 cache or in RAM

### Operation Code :

31	20	9	8	5	4	4	3	2	1	0
Source 4 Immediate	Source 3 Immediate	Source 2 Register	Source 1 Register	STDMA 00	Unit Value					
11 bits	12 bits	4 bits	1 bits	2 bits	2 bits					

### Operation :

$\text{RAM}(\text{IMM4} + \text{RS2}) = \text{SP\_MEMORY}(\text{IMM3} + \text{RS1})$

### Latency :

--- cycles

### Example :

stdma 0xFFFF[r0],100[r15]

## **LDDMAR : Load DMA Register**

---

Load 32\*n bytes into RAM or L2 / L3 cache and put it in D-SRAM

### **Operation Code :**

31	26	25	20	19	8	7	2	1	0
Source 3 Register	Source 2 Register	Source 1 Immediate	LDDMAR 0000 01	Unit Value					
6 bits	6 bits	12 bits	6 bits	2 bits					

### **Operation :**

SP\_MEMORY(RS3) = (IMMx32) RAM(RS2)

### **Latency :**

--- cycles

### **Example :**

lddmar r0,r1,100

## STDMAR : Store DMA Register

---

Sends 32\*n bytes in D-SRAM in L2 / L3 cache or in RAM

### Operation Code :

31	26	25	20	19	8	7	2	1	0
Source 3 Register	Source 2 Register	Source 1 Immediate	STDMAR 0001 01	Unit Value					
6 bits	6 bits	12 bits	6 bits	2 bits					

### Operation :

RAM(RS3) = (IMMx32) SP\_MEMORY(RS2)

### Latency :

--- cycles

### Example :

stdmar r0,r1,100

## **DMAIR : DMA Instruction Register**

---

Load 32\*n bytes into RAM or L2 / L3 cache and put it in I-SRAM

### **Operation Code :**

31	26	25	20	19	8	7	2	1	0
Source 3 Register	Source 2 Register	Source 1 Immediate	DMAIR 0010 01	Unit Value					
6 bits	6 bits	12 bits	6 bits	2 bits					

### **Operation :**

ISRAM(RS3) = (IMMx32) RAM(RS2)

### **Latency :**

--- cycles

### **Example :**

dmair r0,r1,100

## **LDDMAL : Load DMA command List**

---

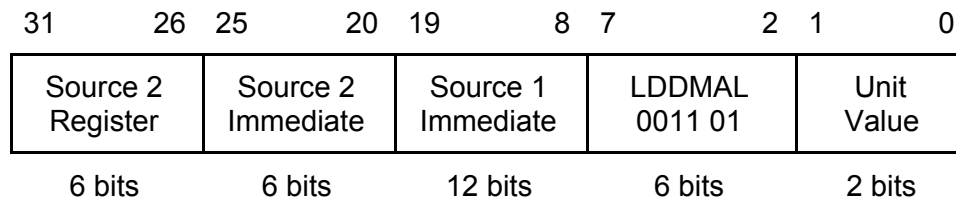
Sends a command list to DMA, the command list is in D-SRAM, it is structured like this:

size 4byte

dst 4byte

adr 8byte

### **Operation Code :**



### **Operation :**

LOOP IMM1

LDDMAR = SP\_MEMORY(RS3+IMM2)

### **Latency :**

--- cycles

### **Example :**

lddmal 12[r0],100



## STDMAL : Store DMA command List

---

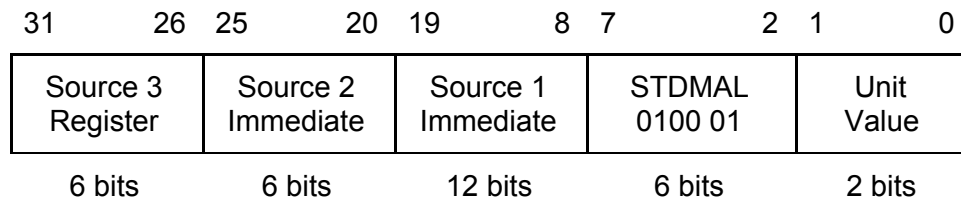
Sends a command list to DMA, the command list is in D-SRAM, it is structured like this:

size 4byte

adr 4byte

dst 8byte

### Operation Code :



### Operation :

LOOP IMM1

STDMAR = SP\_MEMORY(RS3+IMM2)

### Latency :

--- cycles

### Example :

stdmal 42[r0],7

## **CLEARC : Clear Cache**

---

Clear the entire cache (send all data to their corresponding addresses)

### **Operation Code :**

31                      8   7                      2   1                      0

0	CLEARC 1010 01	Unit Value
---	-------------------	---------------

24 bits

6 bits

2 bits

### **Operation :**

...

### **Throughput/Latency :**

1/3+ cycles

### **Example :**

clearc

## **WAIT : WAIT Transfer DMA**

---

Wait for the end of the transfer of all DMAs

### **Operation Code :**

31                      8   7                      2   1                      0

0	WAIT 1011 01	Unit Value
---	-----------------	---------------

24 bits

6 bits

2 bits

### **Operation :**

...

### **Throughput/Latency :**

1/3+ cycles

### **Example :**

wait

## ADD : Addition

---

Add two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	ADD 0000 0000 00			Unit Value				
6 bits	6 bits	6 bits	2 bits	10 bits			2 bits				

### Operation :

RD.size = RS1 + RS2

### Throughput/Latency :

1/3 cycles

### Example :

add.w r0,r1,r2

## **SUB : Subtraction**

---

Subtract two registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	SUB 0001 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### **Operation :**

RD.size = RS1 - RS2

### **Throughput/Latency :**

1/3 cycles

### **Example :**

sub.w r0,r1,r2

## **MULS : Multiplication Signed**

---

Multiply two signed registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	MULS 0010 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### **Operation :**

RD.size = RS1 x RS2

### **Throughput/Latency :**

1/(3-5) cycles

### **Example :**

mults.w r0,r1,r2

## MULU : Multiplication Unsigned

---

Multiply two unsigned registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	MULU 0011 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 x RS2

### Throughput/Latency :

1/(3-5) cycles

### Example :

mulu.w r0,r1,r2

## **DIVS : Division Signed**

---

Divide two signed registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	DIVS 0100 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### **Operation :**

RD.size = RS1 / RS2

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divs.l r0,r1,r2



## **DIVU : Division Unsigned**

---

Divide two unsigned registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	DIVU 0101 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### **Operation :**

RD.size = RS1 / RS2

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divu.b r0,r1,r2

## AND : And

---

AND operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	AND 0110 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 & RS2

### Throughput/Latency :

1/3 cycles

### Example :

and.b r0,r1,r2

## OR : Or

---

OR operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	OR 0111 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 | RS2

### Throughput/Latency :

1/3 cycles

### Example :

or.l r0,r1,r2

## **XOR : Exclusive Or**

---

XOR operation on two registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	XOR 1000 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### **Operation :**

RD.size = RS1 ~ RS2

### **Throughput/Latency :**

1/3 cycles

### **Example :**

xor.q r0,r1,r

## ASL : Arithmetic Shift Left

---

ASL operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	ASL 1001 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 << RS2

### Throughput/Latency :

1/3 cycles

### Example :

asl.b r0,r1,r2

## LSL : Logical Shift Left

---

LSL operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	LSL 1010 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

$RD.size = RS1 \ll RS2$

### Throughput/Latency :

1/3 cycles

### Example :

lsl.l r0,r1,r2

## ASR : Arithmetic Shift Right

---

ASR operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	ASR 1011 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 >> RS2

### Throughput/Latency :

1/3 cycles

### Example :

asr.q r0,r1,r2

## LSR : Logical Shift Right

---

LSR operation on two registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	2	1	0
Destination Register	Source 2 Register	Source 1 Register	Size Value	LSR 1100 0000 00				Unit Value			
6 bits	6 bits	6 bits	2 bits	10 bits				2 bits			

### Operation :

RD.size = RS1 >> RS2

### Throughput/Latency :

1/3 cycles

### Example :

lsl.b r0,r1,r2



## **ADDI : Addition Immediate**

---

Add a register with an immediate value and puts the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	ADDI 0000 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### **Operation :**

RD.size = RS2 + IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

addi.w r0,r1,100

## **SUBI : Subtraction Immediate**

---

Subtract a register with an immediate value and puts the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	SUBI 0001 01		Unit Value					
6 bits	6 bits	10 bits	2 bits	6 bits		2 bits					

### **Operation :**

RD.size = RS2 - IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

addi.w r0,r1,100

## **MULSI : Multiplication Signed Immediate**

---

Multiply an signed register by an immediate signed value and put the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	MULSI 0010 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### **Operation :**

RD.size = RS2 x IMM

### **Throughput/Latency :**

1/(3-5) cycles

### **Example :**

mulsi.b r0,r1,100

## MULUI : Multiplication Unsigned Immediate

---

Multiply an unsigned register by an immediate unsigned value and put the result on another register

### Operation Code :

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	MULUI 001101	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### Operation :

RD.size = RS2 x IMM

### Throughput/Latency :

1/(3-5) cycles

### Example :

mului.w r0,r1,100

## **DIVSI : Division Signed Immediate**

---

Divide an signed register by an immediate signed value and put the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7		2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	DIVSI 0100 01		Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits		2 bits						

### **Operation :**

RD.size = RS2 / IMM

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divs.l r0,r1,100

## **DIVUI : Division Unsigned Immediate**

---

Divide an unsigned register by an immediate unsigned value and put the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	DIVUI 0101 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### **Operation :**

RD.size = RS2 / IMM

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divs.l r0,r1,100

## ANDI : And Immediate

---

AND operation on a register and an immediate value and puts the result on another register

### Operation Code :

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	ANDI 0110 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### Operation :

RD.size = RS2 & IMM

### Throughput/Latency :

1/3 cycles

### Example :

andi.w r0,r1,100

## **ORI : Or Immediate**

---

OR operation on a register and an immediate value and puts the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	ORI 0111 01		Unit Value					
6 bits	6 bits	10 bits	2 bits	6 bits		2 bits					

### **Operation :**

RD.size = RS2 | IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

ori.l r0,r1,100



## **XORI : Exclusive Or Immediate**

---

XOR operation on a register and an immediate value and puts the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	XORI 1000 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### **Operation :**

RD.size = RS2 ~ IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

xori.q r0,r1,100

## ASLI : Arithmetic Shift Left Immediate

---

ASL operation on a register and an immediate value and puts the result on another register

### Operation Code :

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	ASLI 1001 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### Operation :

RD.size = RS2 << IMM

### Throughput/Latency :

1/3 cycles

### Example :

asli.w r0,r1,100

## **LSLI : Logical Shift Left Immediate**

---

LSL operation on a register and an immediate value and puts the result on another register

### **Operation Code :**

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	LSLI 1010 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### **Operation :**

RD.size = RS2 << IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

lsl.l r0,r1,100

## ASRI : Arithmetic Shift Right Immediate

---

ASR operation on a register and an immediate value and puts the result on another register

### Operation Code :

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	ASRI 1011 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### Operation :

RD.size = RS2 >> IMM

### Throughput/Latency :

1/3 cycles

### Example :

asri.w r0,r1,100

## LSRI : Logical Shift Right Immediate

---

LSR operation on a register and an immediate value and puts the result on another register

### Operation Code :

31	26	25	20	19	10	9	8	7	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	LSRI 1100 01	Unit Value						
6 bits	6 bits	10 bits	2 bits	6 bits	2 bits						

### Operation :

RD.size = RS2 >> IMM

### Throughput/Latency :

1/3 cycles

### Example :

lsri.w r0,r1,100

## **ADDQ : Addition Quick immediate**

---

Add a register with an immediate value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0	
Destination Register			Source Immediate			Size Value		ADDQ 0000 10		Unit Value
6 bits			16 bits			2 bits		6 bits		2 bits

### **Operation :**

RD.size += IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

addq.w r0,100

## **SUBQ : Subtraction Quick immediate**

---

Subtract a register with an immediate value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	SUBQ 0001 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size -= IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

subq.w r1,100

## **MULSQ : Multiplication Signed Quick immediate**

---

Multiply an signed register with an immediate signed value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	MULSQ 0010 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size \*= IMM

### **Throughput/Latency :**

1/(3-5) cycles

### **Example :**

mulsq.w r1,100



## MULUQ : Multiplication Unsigned Quick immediate

---

Multiply an unsigned register with an immediate unsigned value and put the result in the same register

### Operation Code :

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	MULUQ 0011 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### Operation :

RD.size \*= IMM

### Throughput/Latency :

1/(3-5) cycles

### Example :

muluq.w r1,100

## **DIVSQ : Divide Signed Quick immediate**

---

Divide an signed register with an immediate signed value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	DIVLSQ 0100 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size /= IMM

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divsq.w r1,100

## **DIVUQ : Divide Unsigned Quick immediate**

---

Divide an unsigned register with an immediate unsigned value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	DIVLSQ 0101 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size /= IMM

### **Throughput/Latency :**

1/(3-10) cycles

### **Example :**

divuq.w r1,100

## **ANDQ : AND Quick immediate**

---

AND a register with an immediate value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	ANDQ 0110 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size /= IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

andq.w r1,100

## ORQ : OR Quick immediate

---

OR a register with an immediate value and put the result in the same register

### Operation Code :

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	ORQ 0111 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### Operation :

RD.size /= IMM

### Throughput/Latency :

1/3 cycles

### Example :

orq.w r1,100

## **XORQ : XOR Quick immediate**

---

XOR a register with an immediate value and put the result in the same register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Register	Source Immediate	Size Value	XORQ 1000 10	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

RD.size /= IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

xorq.w r1,100

## **MOVEI : Move Immediate**

---

Copy an immediate value to a register

### **Operation Code :**

31	26	25	6	5	4	3	2	1	0
Destination Register	Source Immediate	Size Value	MOVEI 11	Unit Value					
6 bits	20 bits	2 bits	2 bits	2 bits					

### **Operation :**

RD.size = IMM

### **Throughput/Latency :**

1/3 cycles

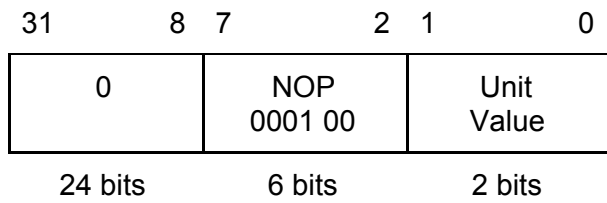
### **Example :**

movei.w r0,100

## **NOP : No Operation**

---

### **Operation Code :**



### **Operation :**

...

### **Throughput/Latency :**

1/3 cycles

### **Example :**

nop



## MOVELR : Move Link Register Load

---

Only the first instruction

Copy the LR register to a general register

### Operation Code :

31	26	25	6	7	2	1	0
Source Register	----	0	MOVELR	0010 00	Unit Value		
6 bits	18 bits		6 bits		2 bits		

### Operation :

RSrc = LR

### Throughput/Latency :

1/3 cycles

### Example :

move r0,lr

## MOVELRS : Move Link Register Store

---

Only the first instruction

Copy a general register to the LR register

### Operation Code :

31	26	25	6	7	2	1	0
Source Register	----	0	MOVELRS 0011 00				Unit Value
6 bits	18 bits		6 bits				2 bits

### Operation :

LR = RSrc

### Throughput/Latency :

1/3 cycles

### Example :

move lr,r0

## **MOVEBR : Move Buffer Register**

---

Only the first instruction

Copy a general register to the BR register

### **Operation Code :**

31	26	25	6	7	2	1	0
Source Register	----	0	MOVEBR 0100 00				Unit Value
6 bits	18 bits		6 bits				2 bits

### **Operation :**

BR = RSrc

### **Throughput/Latency :**

1/3 cycles

### **Example :**

move br,r0

## **CMP : Compare**

---

Compare two registers

### **Operation Code :**

31	26	25	20	19	8	7	6	5	2	1	0
Source 2 Register	Source 1 Register	0	Size Value	CMP 0000	Unit Value						
6 bits	6 bits	12 bits	2 bits	4 bits	2 bits						

### **Operation :**

RF = RS1 ? RS2

### **Throughput/Latency :**

1/3 cycles

### **Example :**

cmp.b r1,r2

## PCMP.H : Posits Half Compare

---

Compare two 16-bit posit registers

### Operation Code :

31	26	25	20	19	10	9	8	7	6	5	2	1	0
Source 2 Posit Register	Source 1 Posit Register	0	Size 2 Value	Size 1 Value	PCMPH 0100	Unit Value							
6 bits	6 bits	10 bits	2 bits	2 bits	4 bits	2 bits							

### Operation :

RF = PH1  $\Leftrightarrow$  PH2

### Throughput/Latency :

1/3 cycles

### Example :

pcmp.h vp0x, vp0y

## PCMP.S : Posits Single Compare

---

Compare two 32-bit posit registers

### Operation Code :

31	26	25	20	19	6	5	2	1	0
Source 2 Posit Register		Source 1 Posit Register		0		PCMPS 1000		Unit Value	
6 bits		6 bits		14 bits		4 bits		2 bits	

### Operation :

RF = PS1  $\Leftrightarrow$  PS2

### Throughput/Latency :

1/3 cycles

### Example :

pcmp.s vp0, vp1

## CMPI : Compare Immediate

---

Compare a register with an immediate value

### Operation Code :

31	26	25	6	5	4	3	2	1	0
Source 2 Register	Source 1 Immediate	Size	CMPI 01	Unit Value					
6 bits	20 bits	2 bits	2 bits	2 bits					

### Operation :

RF = RS1  $\Leftrightarrow$  IMM

### Throughput/Latency :

1/3 cycles

### Example :

cmpi.b r1,42

## PCMPI.H : Posits Half Compare Immediate

---

Compare a 16-bit posits register with an immediate value

### Operation Code :

31	26	25	10	9	6	5	4	3	2	1	0
Source 2 Posit Register	Source 1 Immediate	0	Size Value	PCMPH 10	Unit Value						
6 bits	16 bits	4 bits	2 bits	2 bits	2 bits						

### Operation :

$RF = RS1 \Leftrightarrow IMM \ll 3$

### Throughput/Latency :

1/3 cycles

### Example :

pcmpi.h vp0x,3.41



## PCMPI.S : Posits Single Compare Immediate

---

Compare a 32-bit posits register with an immediate value

### Operation Code :

31	26	25	4	3	2	1	0
Source 2 Posit Register	Source 1 Immediate	PCMPIS 11	Unit Value				
6 bits	22 bits	2 bits	2 bits				

### Operation :

$RF = RS2 \Leftrightarrow IMM \ll 3$

### Throughput/Latency :

1/3 cycles

### Example :

pcmpi.s vp0,3.41

## **BNE : Branch Not Equal**

---

Make a relative jump if there is no equal

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----				BNE 0000 0011 00		Unit Value
14 bits	6 bits				10 bits		2 bits

### **Operation :**

if( $Z == 0$ )  $PC += IMM \times 8$

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bne label

## BEQ : Branch Equal

---

Make a relative jump if there is equal

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		BEQ			Unit Value	
	00 0000		0001 0011 00				
14 bits	6 bits		10 bits			2 bits	

### Operation :

if(Z == 1) PC += IMM x 8

### Throughput/Latency :

1/2 cycles

### Example :

beq label

## BL : Branch Less

---

Make a relative jump if there is less

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		BL				Unit Value
	00 0000		0010 0011 00				
14 bits	6 bits		10 bits				2 bits

### Operation :

if() PC += IMM x 8

### Throughput/Latency :

1/2 cycles

### Example :

bl label

## BLE : Branch Less or Equal

---

Make a relative jump if there is less or equal

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		BLE			Unit Value	
	00 0000		0011 0011 00				
14 bits	6 bits		10 bits			2 bits	

### Operation :

if() PC += IMM x 8

### Throughput/Latency :

1/2 cycles

### Example :

ble label

## BG: Branch Greater

---

Make a relative jump if there is greater

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		BG			Unit Value	
	00 0000		0100 0011 00				
14 bits	6 bits		10 bits			2 bits	

### Operation :

if() PC += IMM x 8

### Throughput/Latency :

1/2 cycles

### Example :

bg label

## **BGE : Branch Greater or Equal**

---

Make a relative jump if there is greater or equal

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----				BGE 0101 0011 00		Unit Value
14 bits	6 bits				10 bits		2 bits

### **Operation :**

if() PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bge label

## **BLS : Branch Less Signed**

---

Make a relative jump if there is less signed

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----		BLS				Unit Value
	00 0000		0110 0011 00				
14 bits	6 bits		10 bits				2 bits

### **Operation :**

if() PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bls label



## **BLES : Branch Less or Equal Signed**

---

Make a relative jump if there is equal signed

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----				BLES 0111 0011 00		Unit Value
14 bits	6 bits				10 bits		2 bits

### **Operation :**

if() PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bles label

## **BGS: Branch Greater Signed**

---

Make a relative jump if there is greater signed

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----		BGS			Unit Value	
	00 0000		1000 0011 00				
14 bits	6 bits		10 bits			2 bits	

### **Operation :**

if() PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bgs label

## **BGES : Branch Greater or Equal Signed**

---

Make a relative jump if there is greater signed or equal

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----			BGES			Unit Value
	00 0000			1001 0011 00			
14 bits	6 bits			10 bits			2 bits

### **Operation :**

if() PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bges label

## **BRA : Branch Unconditional**

---

Make a relative unconditional jump

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----		B				Unit Value
	00 0000		1010 0011 00				
14 bits	6 bits		10 bits				2 bits

### **Operation :**

PC += IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

bges label

## **JMP : Jump**

---

Make a unconditional jump

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----				JMP 0000 1011 00		Unit Value
14 bits	6 bits			10 bits			2 bits

### **Operation :**

PC = IMM x 8

### **Throughput/Latency :**

1/2 cycles

### **Example :**

jmp label

## JMPR : Jump Relative

---

Make a unconditional jump with BR

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		JMPR				Unit Value
	00 0000		0001 1011 00				
14 bits	6 bits		10 bits				2 bits

### Operation :

$$PC = BR + IMM \times 8$$

### Throughput/Latency :

1/2 cycles

### Example :

jmp r label

## CALL : Call

---

Make a unconditional jump, and store the PC address in LR

### Operation Code :

31	18	17	12	11	2	1	0
Source Immediate	----		CALL				Unit Value
	00 0000		0010 1011 00				
14 bits	6 bits		10 bits				2 bits

### Operation :

LR = PC + 8

PC = IMM x 8

### Throughput/Latency :

1/2 cycles

### Example :

call label

## **CALLR : Call Relative**

---

Make a unconditional jump with BR, and store the PC address in LR

### **Operation Code :**

31	18	17	12	11	2	1	0
Source Immediate	----		CALLR				Unit Value
	00 0000		0011 1011 00				
14 bits	6 bits		10 bits				2 bits

### **Operation :**

$$LR = PC + 8$$

$$PC = BR + IMM \times 8$$

### **Throughput/Latency :**

1/2 cycles

### **Example :**

callr label



## ENDP : End of Program

---

Stop the program, the PC does not move, and sends a stop signal

### Operation Code :

31	12	11	2	1	0
0	END 0100 1011 00				Unit Value
20 bits		10 bits			2 bits

### Operation :

...

### Throughput/Latency :

1/2 cycles

### Example :

end

## SWT : Switch

---

Change the number of instructions/cycle by 2 or 4

*SWITCH*: if 7th bit is 1, switch to 4 instructions/cycle mode.

### Operation Code :

31	12	11	2	1	0
----	SWT				Unit
0	011c 1011 00				Value
20 bits		10 bits			2 bits

### Operation :

$RF \& 0xFE = c$

### Throughput/Latency :

1/2 cycles

### Example :

swt 0

## RET : Return

---

Make a jump from the LR value

### Operation Code :

31	12	11	2	1	0
----	RET			Unit	
0	0000 1111 00			Value	
20 bits		10 bits		2 bits	

### Operation :

PC = LR

### Throughput/Latency :

1/2 cycles

### Example :

ret

## **LDM : Load Memory**

---

Load in D-SRAM in a register, the size varies between 8 bits and 64 bits

### **Operation Code :**

31	26	25	23	22	8	7	6	5	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	LDM 0000	Unit Value						
6 bits	3 bits	15 bits	2 bits	4 bits	2 bits						

### **Operation :**

RD.size = SP\_MEMORY(RS2 + IMM)

### **Throughput/Latency :**

1/4 cycles

### **Example :**

ldm.w r0,42[r1]

## STM : Store Memory

---

Store in D-SRAM in a register, the size varies between 8 bits and 64 bits

### Operation Code :

31	26	25	23	22	8	7	6	5	2	1	0
Source 3 Register	Source 2 Register	Source 1 Immediate	Size Value	STM 0100	Unit Value						
6 bits	3 bits	15 bits	2 bits	4 bits	2 bits						

### Operation :

$SP\_MEMORY(RS2 + IMM) = RS3.size$

### Throughput/Latency :

1/4 cycles

### Example :

stm.w r0,42[r1]

## **LDC : Load Cache**

---

Load into cache in a registry, size varies between 8 bit and 64 bit

### **Operation Code :**

31	26	25	23	22	8	7	6	5	2	1	0
Destination Register	Source 2 Register	Source 1 Immediate	Size Value	LDC 0001	Unit Value						
6 bits	3 bits	15 bits	2 bits	4 bits	2 bits						

### **Operation :**

RD.size = CACHE\_MEMORY(RS2 + IMM)

### **Throughput/Latency :**

1/4 cycles

### **Example :**

ldc.w r0,42[r1]

## STC : Store Cache

---

Store into cache in a registry, size varies between 8 bit and 64 bit

### Operation Code :

31	26	25	23	22	8	7	6	5	2	1	0
Source 3 Register	Source 2 Register	Source 1 Immediate	Size Value	STC 0101	Unit Value						
6 bits	3 bits	15 bits	2 bits	4 bits	2 bits						

### Operation :

CACHE\_MEMORY(RS2 + IMM) = RS3.size

### Throughput/Latency :

1/4 cycles

### Example :

stc.w r0,42[r1]

## IN : In I/O Memory

---

Load into the IO D-SRAM in a register

### Operation Code :

31	26	25	18	17	10	9	8	7	2	1	0
Destination Register	Source 1 Immediate	---- 0000 0000		Size Value		IN 0000 10		Unit Value			
6 bits		8 bits		8 bits		2 bits		6 bits		2 bits	

### Operation :

RD.size = SPIO\_MEMORY(IMM)

### Throughput/Latency :

1/4 cycles

### Example :

in.w 42,r0



## OUT : Out I/O Memory

---

Store into the IO D-SRAM in a register

### Operation Code :

31	26	25	18	17	10	9	8	7	2	1	0
Source 2 Destination	Source 1 Immediate	----	0000 0000	Size Value	OUT 0001 10	Unit Value					
6 bits	8 bits	8 bits	2 bits	6 bits	2 bits						

### Operation :

SPIO\_MEMORY(IMM) = RS2.size

### Throughput/Latency :

1/4 cycles

### Example :

out.q 42,r0

## OUTIB : Out I/O Memory Immediate Byte

---

Store into the IO D-SRAM in a immediate value byte

### Operation Code :

31	24	23	8	7	2	1	0
Source 2 Immediate		Source 1 Immediate		OUTI 0010 10		Unit Value	
8 bits		16 bits		6 bits		2 bits	

### Operation :

SPIO\_MEMORY(IMM2) = IMM1

### Throughput/Latency :

1/4 cycles

### Example :

outi.b 42,13

## OUTIW : Out I/O Memory Immediate Word

---

Store into the IO D-SRAM in a immediate value word

### Operation Code :

31	24	23	8	7	2	1	0
Source 2 Immediate	Source 1 Immediate	OUTI 0011 10	Unit Value				
8 bits	16 bits	6 bits	2 bits				

### Operation :

SPIO\_MEMORY(IMM2) = IMM1

### Throughput/Latency :

1/4 cycles

### Example :

outi.w 13,4200

## **LDMV : Load Memory Vector Posits**

---

Load in D-SRAM in a vector posit register, the size varies between 16 bits and 64 bits

### **Operation Code :**

31	26	25	23	22	10	9	8	7	2	1	0
Destination Posit Register	Source 2 Register	Source 1 Immediate	Size Value	LDMV 0000 11	Unit Value						
6 bits	3 bits	13 bits	2 bits	6 bits	2 bits						

### **Operation :**

VPD.size = SP\_MEMORY(RS2 + IMM)x8

### **Throughput/Latency :**

1/4 cycles

### **Example :**

ldmv.xy vp0,42[r1]

## STMV : Store Memory Vector Posits

---

Store in D-SRAM in a vector posit register, the size varies between 16 bits and 64 bits

### Operation Code :

31	26	25	23	22	10	9	8	7	2	1	0	
Source 3 Posit Register			Source 2 Register		Source 1 Immediate			Size Value		STMV 0001 11		Unit Value
6 bits			3 bits		13 bits			2 bits		6 bits		2 bits

### Operation :

$SP\_MEMORY(RS2 + IMM) \times 8 = VPS3.size$

### Throughput/Latency :

1/4 cycles

### Example :

stmv.xy vp0,42[r1]

## **LDCV : Load Cache Vector Posits**

---

Load in Cache in a vector posit register, the size varies between 16 bits and 64 bits

### **Operation Code :**

31	26	25	23	22	10	9	8	7	2	1	0
Destination Posit Register	Source 2 Register	Source 1 Immediate	Size Value	LDCV 0010 11	Unit Value						
6 bits	3 bits	13 bits	2 bits	6 bits	2 bits						

**Operation :**  $VPD.size = CACHE\_MEMORY(RS2 + IMM) \times 8$

**Throughput/Latency :**

1/4 cycles

**Example :**

ldcv.xy vp0,42[r1]

## STCV : Store Cache Vector Posits

---

Store in Cache in a vector posit register, the size varies between 16 bits and 64 bits

### Operation Code :

31	26	25	23	22	10	9	8	7	2	1	0
Source 3 Posit Register	Source 2 Register	Source 1 Immediate	Size Value	STCV 0011 11	Unit Value						
6 bits	3 bits	13 bits	2 bits	6 bits	2 bits						

### Operation :

CACHE\_MEMORY(RS2 + IMM)x8 = VPS3.size

### Throughput/Latency :

1/4 cycles

### Example :

stcv.xy vp0,42[r1]

## **PADD.H : Posits Half Addition**

---

Add two posits 16 bits registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PADD.H 000 000 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### **Operation :**

$$VPH = PS1 + PS2$$

### **Throughput/Latency :**

1/4 cycles

### **Example :**

padd.h vp0, vp0y, vp0z



## PSUB.H : Posits Half Subtraction

---

Subtract two posits 16 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PSUB.H 001 000 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$$VPH = PS1 - PS2$$

### Throughput/Latency :

1/4 cycles

### Example :

psub.h vp0, vp0y, vp0z

## PMUL.H : Posits Half Multiplication

---

Multiply two posits 16 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PMUL.H 010 000 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$$VPH = PS1 \times PS2$$

### Throughput/Latency :

1/4 cycles

### Example :

pmul.h vp0, vp0y, vp0z

## PMULADD.H : Posits Half Multiplication Addition

Multiply and add two posits 16 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PMADD.H 011 000 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$$VPH = PS1 + (PS1 \times PS2)$$

### Throughput/Latency :

1/4 cycles

### Example :

pmuladd.h vp0, vp0y, vp0z

## **PADD.S : Posits Single Addition**

---

Add two posits 32 bits registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	10	9	2	1	0
Destination Posit Register		Source 2 Posit Register		Source 1 Posit Register		---- 0000		PADD.S 100 000 00		Unit Value	
6 bits		6 bits		6 bits		4 bits		8 bits		2 bits	

### **Operation :**

$$PD = PS1 + PS2$$

### **Throughput/Latency :**

1/5 cycles

### **Example :**

padd.s vp0, vp0, vp0

## PSUB.S : Posits Single Subtraction

---

Subtract two posits 32 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	10	9	2	1	0
Destination Posit Register		Source 2 Posit Register		Source 1 Posit Register		---- 0000		PSUB.S 101 000 00		Unit Value	
6 bits		6 bits		6 bits		4 bits		8 bits		2 bits	

### Operation :

$$PD = PS1 - PS2$$

### Throughput/Latency :

1/5 cycles

### Example :

psub.s vp30, vp10, vp20

## PMUL.S : Posits Single Multiplication

---

Multiply two posits 32 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	10	9	2	1	0			
Destination Posit Register			Source 2 Posit Register			Source 1 Posit Register			---- 0000		PMUL.S 110 000 00		Unit Value	
6 bits			6 bits			6 bits			4 bits		8 bits		2 bits	

### Operation :

$$PD = PS1 \times PS2$$

### Throughput/Latency :

1/5 cycles

### Example :

pmul.s vp1, vp2, vp3

## PMADD.S : Posits Single Multiplication Addition

---

Multiply and add two posits 32 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	10	9	2	1	0			
Destination Posit Register			Source 2 Posit Register			Source 1 Posit Register			---- 0000		PMADD.S 111 000 00		Unit Value	
6 bits			6 bits			6 bits			4 bits		8 bits		2 bits	

### Operation :

$$PD = PS1 + (PS1 \times PS2)$$

### Latency :

5 cycles

### Example :

pmuladd.s vp0, vp8, vp7

## **PADD.XYZW : Vector Posits Half Addition**

---

Add two vector posits 64 bits registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	-- 00	Size Value	PADD.XYZW 000 001 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### **Operation :**

$VPD.size = VPS1.size + VPS2.size$

### **Throughput/Latency :**

1/4 cycles

### **Example :**

padd.xyzw vp0, vp1, vp2



## PSUB.XYZW : Vector Posits Half Subtraction

Subtract two vector posits 64 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	-- 00	Size Value	PSUB.XYZW 001 001 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$VPD.size = VPS1.size - VPS2.size$

### Throughput/Latency :

1/4 cycles

### Example :

psub.xyzw vp0, vp1, vp2

## PMUL.XYZW : Vector Posits Half Multiplication

Multiply two vector posits 64 bits registers and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	-- 00	Size Value	PMUL.XYZW 010 001 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$VPD.size = VPS1.size \times VPS2.size$

### Throughput/Latency :

1/4 cycles

### Example :

pmul.xyzw vp0, vp1, vp2

## **PMULADD.XYZW : Vector Posits Half Multiplication Addition**

Multiply and add two vector posits 64 bits registers and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	-- 00	Size Value	PMADD.XYZW 011 001 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### **Operation :**

$VPD.size = VPS1.size + (VPS1.size \times VPS2.size)$

### **Throughput/Latency :**

1/4 cycles

### **Example :**

psub.xyzw vp0, vp1, vp2

## **PADDS : Vector Posits Half Addition with Scalar**

Add vector posits 64 bits registers with posits 16 bits register and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PADDS 000 010 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### **Operation :**

$VPD.size = VPS1.size + PS2$

### **Throughput/Latency :**

1/4 cycles

### **Example :**

padds.xyzw vp0, vp1, vp2x

## PSUBS : Vector Posits Half Subtraction with Scalar

Subtract vector posits 64 bits registers with posits 16 bits register and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PSUBS 001 010 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$VPD.size = VPS1.size - PS2$

### Throughput/Latency :

1/4 cycles

### Example :

psubs.xyzw vp0, vp1, vp2x

## PMULS : Vector Posits Half Multiplication with Scalar

Multiply vector posits 64 bits registers with posits 16 bits register and puts the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PMULS 010 010 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$VPD.size = VPS1.size \times PS2$

### Throughput/Latency :

1/4 cycles

### Example :

psubs.xyzw vp0, vp1, vp2x

## **PMULADDS : Vector Posits Half Multiplication and Addition with Scalar**

Multiply and add vector posits 64 bits registers with posits 16 bits register and puts the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PMADDS 011 010 00	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### **Operation :**

$$VPD.size = VPS1.size + (VPS1.size \times PS2)$$

### **Throughput/Latency :**

1/4 cycles

### **Example :**

psubs.xyzw vp0, vp1, vp2x

## PIPR : Posits Half Inner Product

---

Make a cross product on a 64 bit vector posits register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PIPR 011 011 00		Unit Value						
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits		2 bits						

### Operation :

$PD = VPS1.size \cdot VPS2.size$

### Throughput/Latency :

1/4 cycles

### Example :

pipr.xyz vp0, vp0y, vp0z



## PMOVE : Posits Move

---

Copy a 16 bit posit register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source Posit Register	----- 0000 00	Size 2 Value	Size 1 Value	PMOVE 00 0000 01	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

VPDst.select2 = VPSrc.select1

### Throughput/Latency :

1/3 cycles

### Example :

pmove vp0y, vp0z

## **VPMOVE : Vector Posits Move**

---

copy a vector 64 bit posit register

### **Operation Code :**

31	26	25	20	19	12	11	10	9	2	1	0
Destination Posit Register	Source Posit Register	----- 0000 0000	Size Value	VPMOVE 00 0001 01	Unit Value						
6 bits	6 bits	8 bits	2 bits	8 bits	2 bits						

### **Operation :**

VPDst.size = VPSrc.size

### **Throughput/Latency :**

1/3 cycles

### **Example :**

vpmove vp0, vp1

## **PMOVEI.H : Posits Half Move Immediate**

---

Copy a immediate value posit 16 bit in a posits register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Posit Register	Source Immediate	Size Value	PMOVEI.H 0100 01	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

PDst.select = IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

pmovei.h vp0y, 42.7

## **PMOVEI.S : Posits Single Move Immediate**

---

Copy a immediate value posit 32 bit in a posits register

### **Operation Code :**

31	26	25	8	7	2	1	0
Destination Posit Register	Source Immediate	PMOVEI.S 0101 01	Unit Value				
6 bits	18 bits	6 bits	2 bits				

### **Operation :**

PDst = IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

pmovei.h vp0, 42.3

## **VPMOVEI.H : Vector Posits Half Move Immediate**

---

Copy a immediate value posit 16 bit in a vector posits register

### **Operation Code :**

31	26	25	10	9	8	7	2	1	0
Destination Posit Register	Source Immediate	Size Value	VPMOVEI.H 0110 01	Unit Value					
6 bits	16 bits	2 bits	6 bits	2 bits					

### **Operation :**

VPDst.size = IMM

### **Throughput/Latency :**

1/3 cycles

### **Example :**

vpmovei.xyz vp0y, 42.7

## PMOVERO : Posits Move Register Out

---

Copy a posits register in general register

### Operation Code :

31	26	25	20	19	12	11	10	9	2	1	0
Destination Register	Source Posit Register	----- 0000 0000	Size Value	PMOVERO 00 0010 01	Unit Value						
6 bits	6 bits	8 bits	2 bits	8 bits	2 bits						

### Operation :

RDst = VPSrc.size

### Throughput/Latency :

1/3 cycles

### Example :

pmovero r0,vp1

## PMOVERI : Posits Move Register In

---

Copy a general register in posits register

### Operation Code :

31	26	25	20	19	12	11	10	9	2	1	0
Destination Posit Register	Source Register	----- 0000 0000	Size Value	PMOVERI 00 0011 01	Unit Value						
6 bits	6 bits	8 bits	2 bits	8 bits	2 bits						

### Operation :

VPSrc.size = RDst

### Throughput/Latency :

1/3 cycles

### Example :

pmoveri vp1,r0

## **PDIV.H : Posits Half Division**

---

Divide two register 16 bits posits and put the result on another register

### **Operation Code :**

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PDIV.H 00 0000 10			Unit Value					
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits			2 bits					

### **Operation :**

$PDst = Psrc1.select1 / Psrc2.select1$

### **Throughput/Latency :**

1/6 cycles

### **Example :**

pdiv.h vp0, vp0y, vp0z



## PDIV.S : Posits Single Division

---

Divide two register 32 bits posits and put the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source 2 Posit Register	Source 1 Posit Register	Size 2 Value	Size 1 Value	PDIV.S 00 0001 10	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

$PDst = PSrc1 / PSrc2$

### Throughput/Latency :

1/8 cycles

### Example :

pdiv.s vp0, vp1, vp2

## PSQRT.H : Posits Half Square Root

Square Root register 16 bits posits and put the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source Posit Register	----- 0000 00	Size 2 Value	Size 1 Value	PSQRT.H 00 0010 10	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

PDst = SQRT (VPSrc.select)

### Throughput/Latency :

1/10 cycles

### Example :

psqr.h vp1y,vp1x

## PSQRT.S : Posits Half Square Root

Square Root register 32 bits posits and put the result on another register

### Operation Code :

31	26	25	20	19	14	13	12	11	10	9	2	1	0
Destination Posit Register	Source Posit Register	----- 0000 00	Size 2 Value	Size 1 Value	PSQRT.S 00 0011 10	Unit Value							
6 bits	6 bits	6 bits	2 bits	2 bits	8 bits	2 bits							

### Operation :

PDst = SQRT(VPSrc)

### Throughput/Latency :

1/14 cycles

### Example :

psqrt.s vp0, vp1