.

Theia assembly language specification

Version 0.1
Last update Friday, August 03, 2012

Diego Valverde
2012

Table of Contents

1.1. Introduction

This document is dedicated to specify the grammar, instruction layout and general statement format of the T-ASM compiler.

1.2. Instruction specification

THEIA instructions are 64 bits wide. Each instruction is divided into the following sections as depicted in Figure 1: operation section, destination section, source 1 and source 0 sections or immediate value section. The source 0 and source 1 sections are mutually exclusive with the immediate value section.

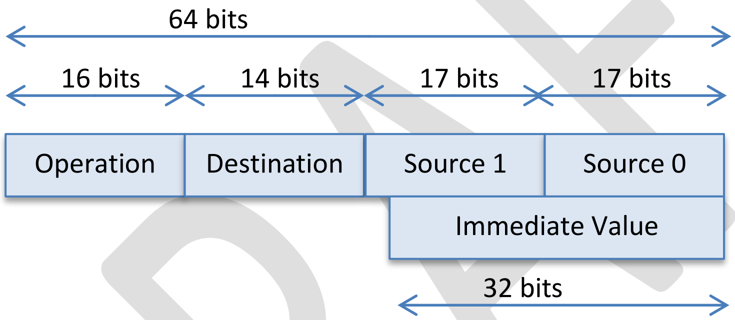


Figure 1 Instruction Layout

Each instruction section has special fields that modify the ALU behavior in various ways. A very important field is the IMM field. The IMM field tells the ALU whether it has to interpret the lowest 32 bits of the instruction as an immediate (literal) value, called IMMV, or as part of the register source sections. Figure 2 illustrates how the ALU interprets the instruction depending on the IMM bit.

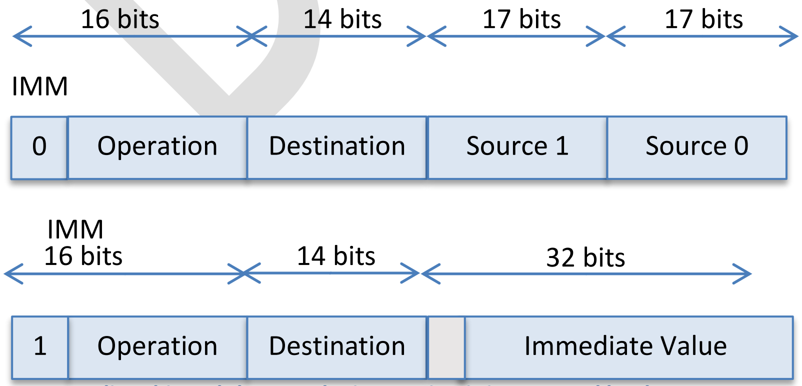
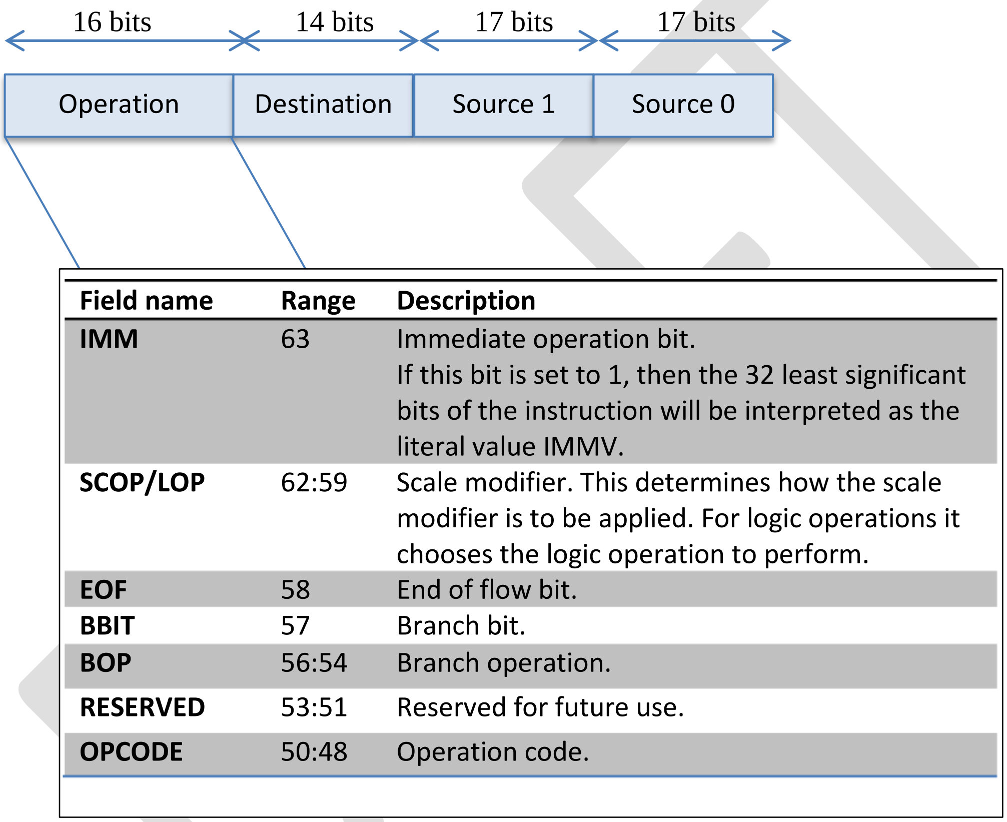


Figure 2 Immediate bit and the way the instruction is interpreted by the IIU

Each operation section is divided into several section “fields”. The section fields provide further granularity for each instruction section behavior.

The next figure presents an example of the various fields from the operation section.



Field name	Range	Description
IMM	63	Immediate operation bit. If this bit is set to 1, then the 32 least significant bits of the instruction will be interpreted as the literal value IMMV.
SCOP/LOP	62:59	Scale modifier. This determines how the scale modifier is to be applied. For logic operations it chooses the logic operation to perform.
EOF	58	End of flow bit.
BBIT	57	Branch bit.
BOP	56:54	Branch operation.
RESERVED	53:51	Reserved for future use.
OPCODE	50:48	Operation code.

Figure 3 Operation section fields

The fields from the other instruction sections will not be mentioned here but those are specified in full detail under the “THEIA Architecture Specification” Document [TBD].

1.3. Assembly language grammar

The general syntax of the assembly compiler in BNF[1] notation is specified on this section. A T-ASM file is defined as a list of statements <statement-list>, where each <statement> is given specified as follows:

Table 1 T-ASM BNF Grammar specification[2]

```
<statement-list> ::= <statement>|<statement-list>

<statement> ::=
[[<label>] [“\n”]][<scale>] <nemonic> [<branch-type>] (<dst-reg>|”@”<label>)  (<src-reg>  <src-reg>) |
(<literal>  [“0”] ) “\n”

Given that:

<nemonic>       ::= “NOP”|”ADD”|”DIV”|”MUL”|”SQRT”|”LOGIC”|”IO”
<dst-reg>       ::= [“@”[”*”]](“R”|”r”)”[“ <digit>* [ “+” “offset”  [“+” <src-reg>] ]“]” “.” <dst-coords>
<src-reg>       ::=  (“R”|”r”)”[“ <digit>* [ “+” “offset” ][“+” “index”] ]“]” “.” <src-coords>
<literal>       ::= “I(“ <digit>+ “)”
<label>         ::= <character>(<character>|<digit>)*”:”
<dst-coords>    ::= (<coord>|”_”)(<coord>|”_”)(<coord>|”_”)
<src-coords>    ::= [“-”]<coord>[“-”]<coord> [“-”]<coord>
<coord>         ::=”x”|”y”|”z”
<scale>         ::= ”unscaled”|”scaled”
<branch-type>  ::= “<” ”BRANCH”  ”.” ("ALWAYS"|"ZERO"|"NOT_ZERO"|"SIGN"|"NOT_SIGN"|
"ZERO_OR_SIGN"|"ZERO_OR_NOT_SIGN" ) “>”
```

The next table presents a series examples of T-ASM valid statements (both grammatically and semantically correct):

Table 2 Example of grammatically valid T-ASM statements

```
//No operation
NOP R0.___ R0.xyz    R0.xyz

//Addition one channel: R[12 + offset].x = 0xcafe  + 0
ADD R[12 + offset ].x__    I(0xcafe) 0

//Addition 3 channels: R[10 + offset].x = R[10 + offset].y = R[10 + offset].z =  0xbabe  + 0
ADD R[10 + offset ].xyz    I(0xbabe) 0
```

[1] Backus-Naur Form

[2] Please note that not all grammatically valid expressions may be semantically valid expressions. For example, the “@” and “*” register qualifiers are only valid for the destination register and only for branching operations.

```
//Subtraction:  R[13 + offset] = R[12 + offset] – R[11 + offset]
ADD R[13+ offset].xyz  R[12 + offset].xyz   R[11 + offset].-x-y-z

//Branch to absolute address 36  if R[55] != R[56]
ADD <BRANCH.NOT_ZERO> @36.___ R[55].xyz R[56].-x-y-z

//Add x components or R3 and R0 and store in y component of R3
ADD R[3]._y_ R[3].xxx R[0].xxx

//Branch into a label called "GenerateRay"
ADD <BRANCH.ALWAYS> @GenerateRay.___ R[0].xyz R[0].xyz

//IO operation
OUT R[0 + offset].xyz    R[10 + offset].xyz   R[12 + offset].xyz
```

1.4. List of assembly language statements (UNDER CONSTRUCTION)

This section will specify the full set of T-ASM valid statements and their respective codification in machine language.

The next 3 tables will specify the values of **OPCODE**, **dst_sel**, **src1_sel** and **src0_sel** which will be used through the rest of the tables in this document section.

Table 3 Instruction OPCODE field values

OPCODE	nemonic	Description
000	NOP	A NOP operation is issued by IIU.
001	ADD	Integer Addition.
010	DIV	Integer division.
011	MUL	Integer multiplication.
100	SQRT	Integer square root.
101	LOGIC	Bitwise logic operations.
110	IO	Input/Output operations
111	RSVR2	RESERVED[3].

[3] The compiler should never assign this value

Table 4 dst_sel encoding

dst_sel string	WEX	WEY	WEZ	Description
___	0	0	0	No write
__z	0	0	1	Z only
_y_	0	1	0	Y only
_yz	0	1	1	Y and Z
x__	1	0	0	X only
x_z	1	0	1	X and Z
xy_	1	1	0	X and Y
Xyz	1	1	1	All

Table 5 src1_sel and src0_sel swizzle encoding[4]

src*_sel string * can be 0 or 1	SWZZ*X	SWZZ*Y	SWZZ*Z
Xyz	00	00	00
Xyy	00	00	01
Xyx	00	00	10
Xzz	00	01	00
Xzy	00	01	01
Xzx	00	01	10
Xxz	00	10	00
Xxy	00	10	01
Xxx	00	10	10
Zyz	01	00	00
Zyy	01	00	01
Zyx	01	00	10
Zzz	01	01	00
Zzy	01	01	01
Zxz	01	10	00
Yyz	10	00	00
Yyy	10	00	01
Yyx	10	00	10
Yzz	10	01	00
Yzy	10	01	01
Yzx	10	01	10

[4] An example how to use this table is provided next:
ADD R[1].xyz R[2].**xyy** R[3].xyz //Use row xyy, thus SWZZ0X=00, SWZZ0X =00, SWZZ0X =01.
ADD R[1].xyz R[2].xyz R[3].**yyz** //Use row yyz, thus SWZZ0X=10, SWZZ0X =00, SWZZ0X =00.

Yxz	10	10	00
Yxy	10	10	01
Yxx	10	10	10

Table 6 src0_sel and scr1_sel sign encoding[5]

src{0|1}_sel string * can be x, y or z	SIGN{0|1}X	SIGN{0|1}Y	SIGN{0|1}Z
*******	0	0	0
****_***	0	0	1
_*	0	1	0
_*_	0	1	1
-***	1	0	0
-_***	1	0	1
-*_**	1	1	0
-*_*_*	1	1	1

The next series of tables specify all of the possible operation combinations and their codification.

Table 7

STATEMENT ID: **A-1**

Statement:

NOP R[i] R[j] R[k]

Description:
No operation

Example:

NOP R0.___ R0.xyz   R0.xyz

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
NOP	n/a	n/a	n/a

[5] An example how to use this table is provided next:
ADD R[1].xyz R[2].**-xyz** R[3].xyz   //Use row -***, thus SIGN1X=1, SIGN1Y=0, SIGN1Z=0.
ADD R[1].xyz R[2].xyz   R[3].**-x-y-z**   //Use row -*-*-*, thus SIGN0X=1, SIGN0Y=0, SIGN0Z=0.

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	0	
MODE	47:45	0	
WEX	44	0	
WEY	43	0	
WEZ	42	0	
DSTINDEX	41:34	0	
SIGN1X	33	0	
SIGN1Y	32	0	
IMMV	31:0	0	

Table 8

STATEMENT ID: **A-1**

Statement:

OPCODE R[index].<dst_sel>   I(literal)   0

Description:

Operates zero plus specified literal value I(literal) and store in R[index]

Example:

```
//      8001880000000001      ADD R[0]._y_ I(1) 0
//      8001840000000002      ADD R[0].__z I(2) 0
//      8001840800000000      ADD R[2].__z I(0) 0
```

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[index]	I(literal)	n/a

Field	Position	Value (Hex)	Notes
IMM	63	1	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	0b100	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	Index	
SIGN1X	33	0	

SIGN1Y	32	0	
IMMV	31:0	Literal	

Table 9

STATEMENT ID: **A-8**

Statement:

OPCODE R[index + offset].<dst_select>   I(literal)   0

Description:

Operates on zero and the specified literal value I(literal) and store in R[index + offset]

Example:

// 8001B02800000004 ADD R[10 + offset].x__ I(4) 0

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[index]	I(literal)	n/a

Field	Position	Value (Hex)	Notes
IMM	63	1	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	0b101	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	index	
SIGN1X	33	0	
SIGN1Y	32	0	
IMMV	31:0	Literal	

Table 10

STATEMENT ID: **A-14**

Statement:

OPCODE R[i].<dst_select> R[j].<scr1_sel> R[k].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	**0b000**	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	I	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	
SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	
SWZZ1Z	26:25	See Table 5	
SRC1ADDR	17:24	J	
SIGN0X	16	See Table 6	
SIGN0Y	15	See Table 6	
SIGN0Z	14	See Table 6	
SWZZ0X	13:12	See Table 5	
SWZZ0Y	11:10	See Table 5	
SWZZ0Z	9:8	See Table 5	
SRC0ADRR	7:0	K	

Table 11

STATEMENT ID: **A-14**

Statement:

OPCODE R[i].<dst_select> R[j].<scr1_sel> R[k + offset].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	**0b001**	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	I	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	
SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	
SWZZ1Z	26:25	See Table 5	
SRC1ADDR	17:24	J	
SIGN0X	16	See Table 6	
SIGN0Y	15	See Table 6	
SIGN0Z	14	See Table 6	
SWZZ0X	13:12	See Table 5	
SWZZ0Y	11:10	See Table 5	
SWZZ0Z	9:8	See Table 5	
SRC0ADRR	7:0	K	

Table 12

STATEMENT ID: **A-14**

Statement:

OPCODE R[i].<dst_select> R[j + offset].<scr1_sel> R[k].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	**0b010**	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	I	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	
SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	
SWZZ1Z	26:25	See Table 5	
SRC1ADDR	17:24	J	
SIGN0X	16	See Table 6	
SIGN0Y	15	See Table 6	
SIGN0Z	14	See Table 6	
SWZZ0X	13:12	See Table 5	
SWZZ0Y	11:10	See Table 5	
SWZZ0Z	9:8	See Table 5	
SRC0ADRR	7:0	K	

Table 13

STATEMENT ID: **A-14**

Statement:

OPCODE R[i].<dst_select> R[j + offset].<scr1_sel> R[k + offset].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	

SCOP/LOP	62:59	0
EOF	58	0
BBIT	57	0
BOP	56:54	0
OPCODE	50:48	See Table 3
MODE	47:45	0b011
WEX	44	See Table 4
WEY	43	See Table 4
WEZ	42	See Table 4
DSTINDEX	41:34	I
SIGN1X	33	See Table 6
SIGN1Y	32	See Table 6
SIGN1Z	31	See Table 6
SWZZ1X	30:29	See Table 5
SWZZ1Y	28:27	See Table 5
SWZZ1Z	26:25	See Table 5
SRC1ADDR	17:24	J
SIGN0X	16	See Table 6
SIGN0Y	15	See Table 6
SIGN0Z	14	See Table 6
SWZZ0X	13:12	See Table 5
SWZZ0Y	11:10	See Table 5
SWZZ0Z	9:8	See Table 5
SRC0ADRR	7:0	K

Table 14

STATEMENT ID: **A-14**

Statement:

OPCODE R[i + offset].<dst_select>   R[j].<scr1_sel>   R[k].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	

OPCODE	50:48	See Table 3
MODE	47:45	0b100
WEX	44	See Table 4
WEY	43	See Table 4
WEZ	42	See Table 4
DSTINDEX	41:34	I
SIGN1X	33	See Table 6
SIGN1Y	32	See Table 6
SIGN1Z	31	See Table 6
SWZZ1X	30:29	See Table 5
SWZZ1Y	28:27	See Table 5
SWZZ1Z	26:25	See Table 5
SRC1ADDR	17:24	J
SIGN0X	16	See Table 6
SIGN0Y	15	See Table 6
SIGN0Z	14	See Table 6
SWZZ0X	13:12	See Table 5
SWZZ0Y	11:10	See Table 5
SWZZ0Z	9:8	See Table 5
SRC0ADRR	7:0	K

Table 15

STATEMENT ID: **A-14**

Statement:

OPCODE R[i + offset].<dst_select>   R[j].<scr1_sel>   R[k + offset].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	0b101	
WEX	44	See Table 4	
WEY	43	See Table 4	

WEZ	42	See Table 4
DSTINDEX	41:34	I
SIGN1X	33	See Table 6
SIGN1Y	32	See Table 6
SIGN1Z	31	See Table 6
SWZZ1X	30:29	See Table 5
SWZZ1Y	28:27	See Table 5
SWZZ1Z	26:25	See Table 5
SRC1ADDR	17:24	J
SIGN0X	16	See Table 6
SIGN0Y	15	See Table 6
SIGN0Z	14	See Table 6
SWZZ0X	13:12	See Table 5
SWZZ0Y	11:10	See Table 5
SWZZ0Z	9:8	See Table 5
SRC0ADRR	7:0	K

Table 16

STATEMENT ID: **A-14**

Statement:

OPCODE R[i + offset].<dst_select>   R[j + offset].<scr1_sel>   R[k].<scr0_sel>

Description:

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	0b110	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	I	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	

SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	
SWZZ1Z	26:25	See Table 5	
SRC1ADDR	17:24	J	
SIGN0X	16	See Table 6	
SIGN0Y	15	See Table 6	
SIGN0Z	14	See Table 6	
SWZZ0X	13:12	See Table 5	
SWZZ0Y	11:10	See Table 5	
SWZZ0Z	9:8	See Table 5	
SRC0ADRR	7:0	K	

Table 17

STATEMENT ID:

Statement:

OPCODE R[i + offset].<dst_select> R[j + offset].<scr1_sel> R[k + offset].<scr0_sel>

Description:

Adds R[j + offset] plus R[k + offset] and store result in R[i + offset]

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC
OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	0	
BOP	56:54	0	
OPCODE	50:48	See Table 3	
MODE	47:45	**0b111**	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	i	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	
SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	

SWZZ1Z	26:25	See Table 5
SRC1ADDR	17:24	j
SIGN0X	16	See Table 6
SIGN0Y	15	See Table 6
SIGN0Z	14	See Table 6
SWZZ0X	13:12	See Table 5
SWZZ0Y	11:10	See Table 5
SWZZ0Z	9:8	See Table 5
SRC0ADRR	7:0	k

The following tables will use the <branch_type> field as defined next:

Table 18 Branch operation BBIT/BOP values

BBIT/ BOP	BRANCH NEMONIC
1 000	“ALWAYS”
1 001	“ZERO”
1 010	“NOT_ZERO”
1 011	“SIGN”
1 100	"NOT_SIGN"
1 101	"ZERO_OR_SIGN"
1 110	"ZERO_OR_NOT_SIGN"

Table 19

STATEMENT ID:

Statement:

OPCODE “<BRANCH.”<branch_type>”>” @<literal>.<dst_select> R[j + offset].<scr1_sel> R[k + offset].<scr0_sel>

Description:

Branches into the absolute address <literal> depending on <branch_type> and the results of the given OPCODE.

Example:

ADD <BRANCH.NOT_ZERO> @36.___ R55.xyz R56.-x-y-z

Encoding (Hex)
02810090006FC038

OP-NEMONIC	DST-NEMONIC	SRC1-NEMONIC	SCR0-NEMONIC

OPCODE	R[i]	R[j+offset]	R[k + offset]

Field	Position	Value (Hex)	Notes
IMM	63	0	
SCOP/LOP	62:59	0	
EOF	58	0	
BBIT	57	1	
BOP	56:54	See Table 18	
OPCODE	50:48	See Table 3	
MODE	47:45	**0b111**	
WEX	44	See Table 4	
WEY	43	See Table 4	
WEZ	42	See Table 4	
DSTINDEX	41:34	I	
SIGN1X	33	See Table 6	
SIGN1Y	32	See Table 6	
SIGN1Z	31	See Table 6	
SWZZ1X	30:29	See Table 5	
SWZZ1Y	28:27	See Table 5	
SWZZ1Z	26:25	See Table 5	
SRC1ADDR	17:24	J	
SIGN0X	16	See Table 6	
SIGN0Y	15	See Table 6	
SIGN0Z	14	See Table 6	
SWZZ0X	13:12	See Table 5	
SWZZ0Y	11:10	See Table 5	
SWZZ0Z	9:8	See Table 5	
SRC0ADRR	7:0	K	