Table of Contents

[1. Command 5](#_Toc530474513)

[2. Input Specifications 12](#_Toc530474514)

[3. Output Specifications 13](#_Toc530474515)

[3.1 IMP Format 13](#_Toc530474516)

[3.2 ELF Format 15](#_Toc530474517)

[4. Assembly Language Specifications 16](#_Toc530474518)

[4.1 Program Elements 16](#_Toc530474519)

[4.1.1 Source statements 16](#_Toc530474520)

[4.1.2 Keywords 17](#_Toc530474521)

[4.1.3 Symbols 17](#_Toc530474522)

[(1) Role of symbols 17](#_Toc530474523)

[(2) Naming a symbol 17](#_Toc530474524)

[4.1.4 Character strings 18](#_Toc530474525)

[4.1.5 Constants 18](#_Toc530474526)

[(1) Integer constants 18](#_Toc530474527)

[(2) Floating-point constants 19](#_Toc530474528)

[(3) Label constants 19](#_Toc530474529)

[(4) Access to constant data 19](#_Toc530474530)

[4.1.6 Expressions and Operators 20](#_Toc530474531)

[(1) Expression of integer constants and expression of an integer constant and a label constant 21](#_Toc530474532)

[(2) Expression of floating-point constants 22](#_Toc530474533)

[4.2 Instructions 23](#_Toc530474534)

[4.2.1 Control instructions 24](#_Toc530474535)

[4.2.2 Data transfer instructions 28](#_Toc530474536)

[4.2.3 Arithmetic/logical instructions 31](#_Toc530474537)

[4.2.4 Instructions to be replaced 37](#_Toc530474538)

[4.2.5 Instructions using register pairs 38](#_Toc530474539)

[4.2.6 Conditional instruction execution 41](#_Toc530474540)

[4.3 Assembler Control Instructions 42](#_Toc530474541)

[4.3.1 List 42](#_Toc530474542)

[4.3.2 Details 42](#_Toc530474543)

[(1) SECTION 42](#_Toc530474544)

[(2) DATA 46](#_Toc530474545)

[(3) DCW 47](#_Toc530474546)

[(4) DCH 48](#_Toc530474547)

[(5) DCB 48](#_Toc530474548)

[(6) DCF 49](#_Toc530474549)

[(7) DS 49](#_Toc530474550)

[(8) DSB 49](#_Toc530474551)

[(9) ARGSIZE 50](#_Toc530474552)

[(10) SRC 50](#_Toc530474553)

[(11) DST 50](#_Toc530474554)

[(12) ENTRY 51](#_Toc530474555)

[(13) ALIGN 51](#_Toc530474556)

[(14) .LINE 53](#_Toc530474557)

[(15) .\_LINE\_TOP/.\_LINE\_END 54](#_Toc530474558)

[(16) PUBLIC 56](#_Toc530474559)

[(17) EXTERN 56](#_Toc530474560)

[(18) \_\_\_stack/.DS/\_\_\_stackEnd (three consecutive instructions) 56](#_Toc530474561)

[4.4 Extended Instructions 59](#_Toc530474562)

[5. Assembler Errors 67](#_Toc530474563)

[5.1 Message Format 67](#_Toc530474564)

[5.2 Messages 67](#_Toc530474565)

[5.3 Error Messages for Determining Licenses 70](#_Toc530474566)

* Notes on notation

Symbols used in the description in the specifications have the meanings below.

< > Indicates that the item enclosed in it is to be specified.

[ ] Indicates an optional item.

... Indicates that the item immediately preceding it is to be specified one or more times.

Δ Indicates one or more spaces (and tabs).

| Indicates that a choice can be made from the items delimited by it.

In this document, "Shaders" are defined as follows.

VTX Shader: IMP-X4 VTX Shader

PX Shader: IMP-X4 PX Shader

Unified Shader: IMP-X5-H3, IMP-X5-V3M, IMP-X5-V3M2, IMP-X5-V3H and IMP-X6-V3U.

# Command

The command name is sasm (Shader Assembler).

Usage:

sasm [options]Δ<filename>

For <filename>, specify an input file name.

If neither options nor input file name is specified, sasm terminates after outputting option descriptions to the standard error output. (This has the same effect as specifying the -h option.)

* Options

When an option is specified multiple times, the last one specified in the command line is valid. No error message is output in this case.

* Output (object) file name specification: -oΔ<filename>

If the output file name specification is omitted, the current folder is used as the output destination.

In this case, the output file name will be the input file name whose suffix is replaced by ".o".

Not open to users

Endian specification: -eΔ{big|little}

When the endian specification is omitted, little is assumed.

* Version display: -V

sasm terminates after outputting version information to the standard error output.

* Option description display: -h

sasm terminates after outputting option descriptions to the standard error output.

The following descriptions are output.

Command Line Format

sasm [option] assembly-source-filename

Option Function

-V Display sasm version number.

-h Display this message.

-o <filename> Specify output file name.

-O {IMP|ELF} Specify output file format.

-T {ABS|REL} Specify output file type.

-c {X4|X5|V3M|V3M2|V3H|V3U} Specify shader processor type.

-P <size> Specify value of program check size.

-U <size> Specify value of uniform check size.

-D [L:] <address> Specify uniform address.

-W <size> Specify value of workmemory check size.

-L <address> Specify workmemory address (non-common).

-X <address> Specify workmemory address (common).

-G <size> Specify value of global workmemory check size.

-B <address> Specify global workmemory address (non-common).

-Y <address> Specify global workmemory address (common).

-S <size> Specify value of stack check size.

-C <address> Specify stack address (non-common).

-Z <address> Specify stack address (common).

-M Display label information.

-d Outputs debugging information.

-l <filename> Specify license file name. (\*1)

-p Display license information. (\*1)

\*1: Only displayed when a license is available.

IMP Shader Processor specification: -cΔ{X4|X5|V3M|V3M2|V3H|V3U}

When this option is omitted, an error occurs.

When the -V or -h option is specified, no error occurs.

X4: The functions for the VTX and PX Shaders are valid.

X5: The functions for the Unified Shader in the IMP-X5-H3 core are valid.

V3M: The functions for the Unified Shader in the IMP-X5-V3M core are valid.

V3M2: The functions for the Unified Shader in the IMP-X5-V3M2 core are valid.

V3H: The functions for the Unified Shader in the IMP-X5-V3H core are valid.

V3U: The functions for the Unified Shader in the IMP-X6-V3U core are valid.

Table 1.1 shows the differences in functions depending on this option setting.

Table 1.1 Differences in Functions Depending on -c Option

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Function | | X4 | X5 | V3M | V3M2 | V3H | V3U |
| Program sections that can be used | | VTX  PX | UNIFIED  VTX  PX | .text | .text | .text | .text |
| Instructions that can be used | | VTX  PX | UNIFIED  VTX  PX | V3M | V3M2 | V3H | V3U |
| Options that can be used (options with underlines when the respective options are omitted) | | | | | | |  |
|  | Output format specification | -O IMP | -O IMP  -O ELF | -O ELF | -O ELF | -O ELF | -O ELF |
|  | Output file type specification | -T ABS | -T ABS | -T ABS  -T REL | -T REL | -T REL | -T REL |
| Memory area size options when the respective options are omitted | | | | | | |  |
|  | Program | 65536 | 65536 | 65536 | 65536 | 65536 | 65536 |
|  | uniform | 32 | 64 | 256 | 256 | 256 | 256 |
|  | lwm (workmemory) | 256 | 1024 | 16384 | 16384 | 16384 | 32768 |
|  | gwm (globalworkmemory) | 1024 | 2048 | 4096 | 4096 | 4096 | 8192 |
|  | impc (stack) | 8192 | 8192 | 917504 | 917504 | 2097152 | 16777216 |
| Start address options when the respective options are omitted | | | | | | |  |
|  | uniform | - | - | 0xf0018000 | 0xf0018000 | 0xf0018000 | 0xf0018000 |
|  | lwm | - | - | 0xf8000000 | 0xf8000000 | 0xf8000000 | 0xf8000000 |
|  | lwm\_comm | - | - | 0xf0000000 | 0xf0000000 | 0xf0000000 | 0xf0000000 |
|  | gwm (globalworkmemory) | 0x400 | 0x2000 | 0xf8008000 | 0xf8008000 | 0xf8008000 | 0xf8008000 |
|  | gwm\_comm | - | - | 0xf0008000 | 0xf0008000 | 0xf0008000 | 0xf0008000 |
|  | impc | - | - | 0xed800000 | 0xed800000 | 0xed800000 | 0xed800000 |
|  | impc\_comm | - | - | 0xed000000 | 0xed000000 | 0xed000000 | 0xee000000 |
| Revision for IMP format output | | 3 | 4 | - | - | - | - |

Not open to users

There are no differences in functions of V3M2 and V3H depending on the -c option.

However, there is only a difference that ELF e\_flags for output files differs.

Not open to users

Input db2 file specification: -bΔ<filename>

This option specifies the name of the .db2 file output by the compiler.

When this option is specified, it is assumed that the input assembler source file was generated by the compiler and the symbol attributes in the .db2 file are reflected to the symbol information in the output object file.

This option is valid when ELF is specified for the -O option and V3M, V3M2, or V3H is specified for the -c option.

* Output format specification: -OΔ{IMP|ELF}

This option specifies the output format.

See Table 1.1 for the setting when the specification of the output format is omitted.

* Output file type specification: -TΔ{ABS|REL}

This option specifies the output file type.

The REL specification of this option is only valid when ELF is specified for the -O option.

See Table 1.1 for the setting when the specification of the output file type is omitted.

* Program area size: -PΔ<size>

See Table 1.1 for the setting when the specification of the program area size is omitted.

Unit: 32 bits

* Uniform area size: -UΔ<size>

See Table 1.1 for the setting when the specification of the uniform area size is omitted.

Unit: 32 bits

* Uniform start address: -DΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the uniform start address is omitted.

This option specifies the start address of the uniform section.

Unit: 8 bits

* Uniform LMA (Load Memory Address): -DΔL:<address>

This option is only valid when the following conditions are all satisfied.

* X5 is specified for the -c option.
* ELF is specified for the -O option.
* ABS is specified for the -T option.

This option specifies the physical address of uniform in the ELF program header (p\_paddr).

Unit: 8 bits

* lwm (workmemory) area size: -WΔ<size>

See Table 1.1 for the setting when the specification of the lwm (workmemory) area size is omitted.

X4 and X5: workmemory area size

V3M, V3M2, V3H and V3U: Total size of lwm and lwm\_comm areas

Unit: X4 and X5: 32 bits; V3M, V3M2, V3H and V3U: 8 bits

* lwm start address: -LΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the lwm start address is omitted.

This option specifies the start address of the lwm section.

Unit: 8 bits

* lwm\_comm start address: -XΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the lwm\_comm start address is omitted.

This option specifies the start address of the lwm\_comm section.

Unit: 8 bits

* gwm (globalworkmemory) area size: -GΔ<size>

See Table 1.1 for the setting when the specification of the gwm (globalworkmemory) area size is omitted.

X4 and X5: globalworkmemory area size

V3M, V3M2, V3H and V3U: Total size of gwm and gwm\_comm areas

Unit: X4 and X5: 32 bits; V3M, V3M2, V3H and V3U: 8 bits

* gwm (globalworkmemory): -BΔ<address>

This option is only valid when ABS is specified for the -T option.

See Table 1.1 for the setting when the specification of the gwm (globalworkmemory) start address is omitted.

X4 and X5: Start address of the globalworkmemory section

V3M, V3M2, V3H and V3U: Start address of the gwm section

Unit: X4 and X5: 32 bits; V3M, V3M2, V3H and V3U: 8 bits

* gwm\_comm start address: -YΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the gwm\_comm start address is omitted.

This option specifies the start address of the gwm\_comm section.

Unit: 8 bits

* impc (stack) area size: -SΔ<size>

See Table 1.1 for the setting when the specification of the impc (stack) area size is omitted.

X4 and X5: Stack area size

V3M, V3M2, V3H and V3U: Total size of impc and impc\_comm areas

Unit: X4 and X5: 32 bits; V3M, V3M2, V3H and V3U: 8 bits

* impc start address: -CΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the impc start address is omitted.

This option specifies the start address of the impc section.

Unit: 8 bits

* impc\_comm address: -ZΔ<address>

This option is only valid when ABS is specified for the -T option and V3M is specified for the -c option.

See Table 1.1 for the setting when the specification of the impc\_comm start address is omitted.

This option specifies the start address of the impc\_comm section.

Unit: 8 bits

* Label information output: -M

This option is only valid when ABS is specified for the -T option.

After assembly, label names and addresses are sent to the standard output.

When the label information output option is omitted, no label information is output.

Information is output as shown below.

# <section name>

<symbol>Δ<address (hex)>

:

# <section name>

<symbol>Δ<address (hex)>

:

Information is sorted by <symbol> for each section.

<address (hex)> is output in eight hexadecimal digits without prefix 0x.

Output example:

# VTX

\_f0 00000009

\_f1.1 00000000

\_main 00000012

# WORKMEMORY

\_a 00000000

\_b 00000001

* Debugging information specification: -d

When this option is specified, the debugging information is output.

When the -b option is specified and the debugging information is included in the .db2 file, the debugging information corresponding to the C source file is output.

This option is only valid when ELF is specified for the -O option and V3M, V3M2, V3H or V3U is specified for the -c option.

When this option is omitted, no debugging information is output.

* License file specification: -lΔ<filename>

This option is only valid for the build version that the license function is enabled.

When this option is omitted, the "license.txt" file which is in the same folder as for sasm.exe is specified.

* License information display: -p

This option is only valid for the build version that the license function is enabled.

The license information is sent to the standard output.

The following shows the output contents.

Detect MAC address:

00:00:00:00:00:00

License information:

GEN3 IMP-Compiler debugger:

LICENSEE=XXXXXXX

VERSION\_MAJOR=X

VERSION\_MINOR=X

EXPIRED\_DATE=yyyy/m/d

MACHINE=00:00:00:00:00:00

LICENSED\_NO=X

SERIAL\_NO=XXXXXXXX

OPTION1=

OPTION2=

OPTION3=

<size> and <address> must be specified for options in the following format within the range shown below.

[+] [0 [{ x | X }]] digits

"digits" must be one or more numeric digits or alphabetic characters ("a" to "f" or "A" to "F"). When the first character is 0 and the second character is neither "x" nor "X", the specified character string is regarded as an octal number. When the first character is 0 and the second character is "x" or "X", the character string is regarded as a hexadecimal number. When the first character is a digit from "1" to "9", the character string is regarded as a decimal number. The number can be prefixed with a plus sign.

Range of "digits": 0 to 0xfffffffe

Note that the -B <address> must be specified within the following range when -c V3M, V3M2, V3H or V3U is not specified.

Range of "digits": 0 to 0x3fffffff

When -c V3M, V3M2, V3H or V3U is specified, the address must be specified with a multiple of 4 (section alignment).

# Input Specifications

Two types of file can be input; the assembler source file and the .db2 file output by the compiler.

The input assembler source file must be a file written for the VTX Shader, PX Shader, or Unified Shader.

The .db2 file is in the ELF format. In this file, the compiler outputs symbol information and debug information, which cannot be passed to the assembler through the assembler source file alone. For details of the contents of the .db2 file, refer to the CVengine Compiler (ccimp) - db2 File Specifications.

# Output Specifications

The output file is in either of the following two formats: the conventional IMP format and the ELF format.

The following two types of object file can be output in the ELF format: absolute files and relocatable files.

## IMP Format

Data in each data area can be output together with the program code of the VTX Shader, PX Shader, or Unified Shader, as the same object file. (This assembler does not support a file containing the program codes of the VTX Shader, PX Shader, and Unified Shader together.)

An output file contains header information for managing data and code. Header information contains the positions and sizes of each data area, the VTX Shader area, the PX Shader area, and the Unified Shader area in the file, and also contains an external data size and input/output D$ numbers. It also contains, as the start address of the program, the addresses of the \_main label and the \_sini label of each of the VTX Shader, the PX Shader, and the Unified Shader.

If there is no data area or program in a source file, "-1" is set for the position information, and 0 is set for the size value. If the external data size or the input/output D$ numbers are unknown, 0 is set.

For input/output D$ numbers, D$ numbers 0 to 7 are associated with the least significant bit of data to the most significant bit of data, in this order, and data in which the bits associated with D$ numbers are set is set. For example, if 0 and 2 are specified for input D$ numbers, 00000101)2= 0x5 is set for the input D$ numbers.

The addresses of the \_main label and \_sini label are respective addresses in the Shader Program, not offset values in the object file. For each address value, "-1" is set if the label does not exist.

|  |  |  |
| --- | --- | --- |
| # | Field | Description |
| 1 | char magic[4] | Magic number "IMP-" |
| 2 | int revision | File format Revision: 5 |
| 3 | int uniformOffset | Position of the uniform area in the file |
| 4 | int uniformSize | Size of the uniform area |
| 5 | int vtxOffset | Position of the VTX Shader Program in the file |
| 6 | int vtxSize | Size of the VTX Shader Program |
| 7 | int pxOffset | Position of the PX Shader Program in the file |
| 8 | int pxSize | Size of the PX Shader Program |
| 9 | int argSize | External data size (rev.2) |
| 10 | int srcImage | Input D$ number (only the lower 8 bits being used) (rev.2) |
| 11 | int dstImage | Output D$ number (only the lower 8 bits being used) (rev.2) |
| 12 | int vtxMainAddr | Address of the \_main label of the VTX Shader Program (rev.3) |
| 13 | int vtxSiniAddr | Address of the \_sini label of the VTX Shader Program (rev.3) |
| 14 | int pxMainAddr | Address of the \_main label of the PX Shader Program (rev.3) |
| 15 | int pxSiniAddr | Address of the \_sini label of the PX Shader Program (rev.3) |
| 16 | int unifiedOffset | Position of the unified section in the file (rev.4) |
| 17 | int unifiedSize | Size of the unified section (rev.4) |
| 18 | int unifiedMainAddr | Address of the \_main label of the unified section (rev.4) |
| 19 | int unifiedSiniAddr | Address of the \_sini label of the unified section (rev.4) |
| 20 | int cpuType | Shader processor type (rev.5)  31 16 15 0  Core type  Series  1: X4 1: VTX  2: PX  2: X5 1: H3  2: V3M  3: V3M2  4: V3H  3: X6 1: V3U |

|  |  |  |
| --- | --- | --- |
| # | Field | Description |
| 21 | int textOffset | Position of the .text section in the file (rev.5) |
| 22 | int textSize | Size of the .text section (rev.5) |
| 23 | int textMainAddr | Address of the \_main label of the .text section (rev.5) |
| 24 | int textSiniAddr | Address of the \_sini label of the .text section (rev.5) |
| 25 | int lwmOffset | Position of the LWM (workmemory) section in the file (rev.5) |
| 26 | int lwmSize | Size of the LWM (workmemory) section (total size of lwm and lwm\_comm areas) (rev.5) |
| 27 | int lwmComSize | Size of the LWM (workmemory) section (lwm\_comm area size) (rev.5) |
| 28 | int lwmNonComAlign | Alignment of the LWM (workmemory) section (alignment of the start of the lwm area) (rev.5) |
| 29 | int lwmComAddress | Address of the LWM (workmemory) section (start address of lwm\_comm) (rev.5) |
| 30 | int lwmNonComAddress | Address of the LWM (workmemory) section (start address of lwm) (rev.5) |
| 31 | int gwmOffset | Position of the GWM (global workmemory) section in the file (rev.5) |
| 32 | int gwmSize | Size of the GWM (global workmemory) section (total size of gwm and gwm\_comm areas) (rev.5) |
| 33 | int gwmComSize | Size of the GWM (global workmemory) section (gwm\_comm area size) (rev.5) |
| 34 | int gwmNonComAlign | Alignment of the GWM (global workmemory) section (alignment of the start of the gwm area) (rev.5) |
| 35 | int gwmComAddress | Address of the GWM (global workmemory) section (start address of gwm\_comm) (rev.5) |
| 36 | int gwmNonComAddress | Address of the GWM (global workmemory) section (start address of gwm) (rev.5) |
| 37 | int impcOffset | Position of the IMPC (stack) section in the file (rev.5) |
| 38 | int impcSize | Size of the IMPC (stack) section (total size of impc and impc\_comm areas) (rev.5) |
| 39 | int impcComSize | Size of the IMPC (stack) section (impc\_comm area size) (rev.5) |
| 40 | int impcNonComAlign | Alignment of the IMPC (stack) section (alignment of the start of the impc area) (rev.5) |
| 41 | int impcComAddress | Address of the IMPC (stack) section (start address of impc\_comm) (rev.5) |
| 42 | int impcNonComAddress | Address of the IMPC (stack) section (start address of impc) (rev.5) |

## ELF Format

For details of the ELF format, refer to the CVengine Compiler (ccimp) - ELF Object File Specifications.

# Assembly Language Specifications

## Program Elements

### Source statements

The configuration of a source statement is as described below.

Write a source statement with ASCII characters.

Write a source statement on a single line, without inserting a line feed code.

[label][Δoperation[Δoperand[,...]]][Δ][comment]

Example)

|  |  |  |  |
| --- | --- | --- | --- |
| Label | Operation | Operand | Comment |
| LABEL: | MOV | R1,R2 | # Comment |

###### Label

Write a symbol as the nameplate assigned to the source statement.

Labels are classified into two types: standard labels and local labels.

* Standard label:

This label can be referenced between sections.

* Local label:

This label can be defined and referenced only within the program section.

A local label must be specified with a single digit (0 to 9).

Multiple labels having the same name can be specified within a section.

When a label is referenced, the nearest label from the position of reference is referred to, and the direction of reference must be specified with an alphabetic character of "f" (forward) or "b" (backward).

Write a label as described below.

* Start writing a label at any column.

And

* Place a colon (:) after the label name at any column.

To refer a local label, specify the direction of reference (forward or backward).

Example:

.section unified

1: # ○

calla pcr0, 1b Branches to label 1 at # ○ (backward).

calla pcr0, 1f Branches to label 1 at # □ (forward).

1: # □

###### Operation

Write an instruction, assembler control instruction, or extended instruction.

Write an operation as described below.

* If no label is written: Start writing it at any column.
* If a label is written: Start writing it at any column.

###### Operand

Write the target on which to execute the operation.

Start writing it by leaving one or more spaces or tabs from the operation.

To write more than one operand, delimit each by a comma (,).

###### Comment

Write a hash symbol (#) first and then start writing a comment.

In a comment, kanji and kana characters in the Shift\_JIS character code can be written.

The assembler regards the string of characters from the hash symbol to the end of the line as a comment.

### Keywords

Keywords are not case-sensitive.

Note, however, that a string containing both upper- and lower-case characters is not recognized as a keyword. Such a string is recognized as a symbol.

(MOV and mov are recognized as keywords, while Mov is recognized as a symbol.)

* Registers: R0 to R31 for V3M, V3M2 and V3H. R0 to R47 for V3U.
* Control registers: CR0 to CR31
* PC registers: PCR0 to PCR7
* Status register: SR
* Operators: HIGHW, LOWW
* Section names: UNIFORM, VTX, PX, UNIFIED, .TEXT, WORKMEMORY, GLOBALWORKMEMORY, STACK, LWM, LWM\_COMM, GWM, GWM\_COMM, IMPC, IMPC\_COMM
* Instructions: MOV, CALL, ADD, etc.
* Extended instructions: MOVU.F, MOVU.I, etc.
* Control instructions: .SECTION, etc.

### Symbols

##### Role of symbols

Symbols have the following use.

* Address symbol

Example:

* Address symbols

CALL R16, SUB1

...

SUB1:

##### Naming a symbol

###### Characters that can be used for a symbol

* The following ASCII characters can be used:
* Upper- and lower-case alphabetic characters
* Digits
* Underscore (\_)
* Period (.)
* Symbol for a local label (for definition)
* A single digit (0 to 9)
* Symbol for a local label (for reference)
* A digit (0 to 9) for the first character
* "b" or "f" for the second character

The assembler is case-sensitive to symbols.

###### First character of a symbol

The following ASCII characters can be used:

* Symbol for a standard label
* Upper- or lower-case alphabetic character
* Underscore (\_)
* Period (.)

A symbol starting with a period (.) is an internal label.

When a file is output in the ELF format, the symbols starting with a period (.) that are written in a source statement are not output. Note that the .STACKSIZE symbol which is generated by the assembler is output.

* Symbol for a local label
* A single digit (0 to 9)

###### Maximum number of characters (symbol for a standard label)

No limit on the number of characters.

###### Names that cannot be used as symbols

* Keywords
* Instructions

### Character strings

Cannot be used.

### Constants

The size of constants placed in the uniform section differs between Shaders as follows.

IMP-X4-VTX, IMP-X4-PX, and IMP-X5-H3: 4 bytes

IMP-X5-V3M, IMP-X5-V3M2, and IMP-X5-V3H: 1, 2, or 4 bytes

IMP-X6-V3U: 1, 2, or 4 bytes

The bit width of a constant written for an operand depends on the instruction.

##### Integer constants

Decimal and hexadecimal integer constants are supported. A hexadecimal integer constant is expressed by adding "0x".

* Decimal notation

[{ + | - }] n

If neither the + nor - sign is specified, + is assumed.

n Specify the integer part in decimal notation.

Example:

.DATA -1234

* Hexadecimal notation

0xzzzz

0x Radix indicating hexadecimal notation

zzzz Specify the constant in hexadecimal notation.

If the specification is shorter than the data length, the constant is right-aligned, with the left side being padded with the necessary number of zeros.

Example:

.DATA 0xABCD = .DATA 0x0000ABCD

##### Floating-point constants

* Value range: 1.40129846e-45 to 3.40282347e+38
* Either the following floating-point notation (a) or (b)

(a): [{ + | - }]n.[m][{e|E}[{ + | - }]zz][{f|F}]

(b): [{ + | - }]n{e|E}[{ + | - }]zz[{f|F}]

If neither the + nor - sign is specified, + is assumed.

n Specify the integer part in decimal notation.

m Specify the fraction part in decimal notation.

zz Specify the value of the exponent part (exponential in decimal) in decimal notation.

If none is specified, a power of zero is assumed. If neither the + nor - sign is specified, + is assumed.

Example:

0.5E-2 = 0.5 x 10(-2) = 0.005 = 0x3767AE

##### Label constants

If a label is specified as a constant, the address value is set as the constant.

A colon (:) is not necessary immediately after the label name.

* For the VTX Shader area, the PX Shader area, or the Unified Shader area: Program counter (PC)
* For other areas:

IMP-X4-VTX, IMP-X4-PX, and IMP-X5-H3: Index value from the beginning (32-bit word address)

IMP-X5-V3M, IMP-X5-V3M2, and IMP-X5-V3H: Byte address

IMP-X6-V3U: Byte address

Examples:

.section uniform

LABEL0:

LABEL0 # uniform index = 0

LABEL1 # PC = 0

.section vtx

LABEL1:

##### Access to constant data

Access to the constant data placed in the uniform section differs between Shaders.

The following shows output object examples.

Examples: Output object

[For Unified Shader (IMP-X5-V3M, IMP-X5-V3M2, IMP-X5-V3H and IMP-X6-V3U)]

* The address of a label or data in the uniform section is the absolute value of the byte address in the address space (hereafter called Addr).

- For an instruction that refers to the uniform section, the address specified in the LABEL in the instruction is an index value relative to 0 in 32-bit (word) units.

- For an instruction that does not refer to the uniform section, the address specified in the LABEL in the instruction is the absolute value of the byte address in the address space.

[Input assembler source]

ADDU R16, R17, 0 # R16 = R17 + 2 (Immediate value is specified as the index.)

ADDU R16, R17, LABEL # R16 = R17 + 10 (Label is specified as the index.)

ADDU.I R16, R17, 10 # R16 = R17 + 10 (Extended instruction is used.)

MOVI R16, LABEL # R16 = Address of LABEL

.SECTION uniform

.DCW 0x00000002 # User-specified constant data (Addr = 0xfff00000 (Index = 0))

LABEL: # (Addr = 0xfff00004 (Index = 1))

[Output object]

.SECTION uniform

.DCW 0x00000002 # User-specified constant data (Addr = 0xfff00000 (Index = 0))

.DCW 0x0000000A # Reserves constant data in the uniform section

# (Addr = 0xfff00004 (Index = 1))

:

ADDU R16, R17, 0 # Outputs the ADDU instruction (Adds to the data at index = 0)

ADDU R16, R17, 1 # Outputs the ADDU instruction (Adds to the data at index = 1)

ADDU R16, R17, 1 # Outputs the ADDU instruction (Adds to the data at index = 1)

MOVI R16, 4 # (Sets the value of the byte address of LABEL)

# \*The MOVI instruction uses the lower 16 bits of the specified

# value.

[For VTX Shader, PX Shader, and Unified Shader (IMP-X5-H3)]

* The address of a label or data in the uniform section is an index value relative to 0 in

[Input assembler source]

ADDU R16, R17, 0 # R16 = R17 + 2 (Immediate value is specified as the index.)

ADDU R16, R17, LABEL # R16 = R17 + 10 (Label is specified as the index.)

ADDU.I R16, R17, 10 # R16 = R17 + 10 (Extended instruction is used.)

MOVI R16, LABEL # R16 = Address of LABEL

.SECTION uniform

.DCW 0x00000002 # User-specified constant data (Index = 0)

LABEL: # (Index = 1)

[Output object]

.SECTION uniform

.DCW 0x00000002 # User-specified constant data (Index = 0)

.DCW 0x0000000A # Reserves constant data in the uniform section (Index = 1)

:

ADDU R16, R17, 0 # Outputs the ADDU instruction (Adds to the data at index = 0)

ADDU R16, R17, 1 # Outputs the ADDU instruction (Adds to the data at index = 1)

ADDU R16, R17, 1 # Outputs the ADDU instruction (Adds to the data at index = 1)

MOVI R16, 1 # (Sets the value of the byte address of LABEL)

### Expressions and Operators

Combinations of constants and operators can be used as expressions.

The following three types of combination are allowed.

* Expression of integer constants

= Operation between integer constants

* Expression of an integer constant and a label constant

= Operation between an integer constant and a label constant

* Expression of floating-point constants

= Operation between floating-point constants

Multiple operators can be used in combination.

##### Expression of integer constants and expression of an integer constant and a label constant

* Expressions are operated in a 64-bit signed type.

No error will be output even if the value range of the specified type is exceeded during evaluation of an expression.

* An error will be output if the divisor is zero.
* An expression of an integer constant and a label constant is allowed only when the final form of the expression becomes as follows.

<label constant> {+|-} <integer constant>

* Therefore, multiple label constants are not allowed in an expression, and even when only a single label constant is used, operation that is neither addition nor subtraction is not allowed in an expression.

###### Operators

|  |  |  |
| --- | --- | --- |
| Type | Operator | Function |
| Arithmetic operator | + | Adds the lvalue and the rvalue. |
| - | Subtracts the rvalue from the lvalue. |
| \* | Multiplies the lvalue with the rvalue. |
| / | Divides the lvalue with the rvalue. |
| %(MOD) | Obtains the remainder of division of the lvalue by the rvalue. |
| +(sign) | Handles the value specified after this operator as a positive value. |
| -(sign) | Handles the value specified after this operator as a negative value. |
| Bitwise logical operator | ~ | Obtains the logical negation of the value specified after this operator. |
| & | Obtains the bitwise logical AND of the lvalue and the rvalue. |
| | | Obtains the bitwise OR of the lvalue and the rvalue. |
| ^ | Obtains the bitwise exclusive OR of the lvalue and the rvalue. |
| Shift operator | >> | Shifts the lvalue to the right by the number of bits specified in the rvalue. |
| << | Shifts the lvalue to the left by the number of bits specified in the rvalue. |
| 2-byte separation operator | HIGHW | Obtains the upper 16 bits of the value specified after this operator. |
| LOWW | Obtains the lower 16 bits of the value specified after this operator. |
| Other operator | ( ) | Gives precedence to the operation enclosed in ( ). |

The above operators are classified into unary operators and binary operators.

|  |  |
| --- | --- |
| Unary operator | +(sign), -(sign), ~, HIGHW, LOWW |
| Binary operator | +, -, \*, /, %(MOD), &, |, ^, >>, << |

###### Expression evaluation precedence

|  |  |
| --- | --- |
| Precedence | Operator |
| 1 | ( ) |
| 2 | +(sign), -(sign), ~, HIGHW, LOWW |
| 3 | \*, /, %(MOD) |
| 4 | +, - |
| 5 | >>, << |
| 6 | & |
| 7 | ^ |
| 8 | | |

##### Expression of floating-point constants

* Expressions are operated in the single-precision floating-point type.

No error will be output even if the value range of the type is exceeded during evaluation of an expression.

* An error will be output if the devisor is zero.
* An expression of floating-point constants is allowed only for four arithmetic operations (addition, subtraction, multiplication, and division).

###### Operators

|  |  |  |
| --- | --- | --- |
| Type | Operator | Function |
| Arithmetic operator | + | Adds the lvalue and the rvalue. |
| - | Subtracts the rvalue from the lvalue. |
| \* | Multiplies the lvalue with the rvalue. |
| / | Divides the lvalue with the rvalue. |
| +(sign) | Handles the value specified after this operator as a positive value. |
| -(sign) | Handles the value specified after this operator as a negative value. |
| Other operator | ( ) | Gives precedence to the operation enclosed in ( ). |

The above operators are classified into unary operators and binary operators.

|  |  |
| --- | --- |
| Unary operator | +(sign), -(sign) |
| Binary operator | +, -, \*, / |

###### Expression evaluation precedence

|  |  |
| --- | --- |
| Precedence | Operator |
| 1 | ( ) |
| 2 | +(sign), -(sign) |
| 3 | \*, / |
| 4 | +, - |

## Instructions

The following lists the instructions that can be specified in this assembly language.

For each instruction, Sn (n = 0, 1) and D0 represent registers.

CR indicates a control register.

PCR indicates a PC register.

IMM indicates an integer constant or an expression of integer constants.

LABEL indicates a label constant or an expression of an integer constant and a label constant.

SR indicates a status register.

× indicates an unsupported instruction.

○ indicates a supported instruction.

● indicates a uniform reference instruction that can be specified as an extended instruction. Adding .F or .I at the end of an instruction specifies an extended instruction. (For extended instructions, refer to section 4.4.)

The access mode of the uniform reference instruction of V3M2, V3H and V3U differs depending on whether a dollar sign ($) is prefixed to the LABEL or IMM operand. If a dollar sign ($) is prefixed, R6-based relative access is performed. If a dollar sign ($) is not prefixed, 0-based relative access is performed.

\*1 indicates an instruction to be replaced. The assembler replaces this instruction with a supported instruction (for the instructions to be replaced, refer to section 4.2.4).

Details of each control instruction may differ depending on the Shader.

Which Shader executes the specified instruction is determined according to the name of the section where the control instruction belongs and the specifications of the register and #IMM are checked.

[Supplementary note] About 0-based relative access and R6-based relative access of the uniform reference instruction

* For VTX and PX, only 0-based relative access can be specified. For Unified and V3M, only R6-based relative access can be specified. For V3M2, V3H and V3U, both 0-based relative access and R6-based relative access can be specified.

Example 1: For VTX, PX, Unified, and V3M, the following can be specified:

MOVU R16, 1

Example 2: For V3M2, V3H and V3U, the following can be specified:

MOVU R16, 1 # 0-based relative access

MOVU R16, $1 # R6-based relative access

### Control instructions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Control Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| ACTPS S0 | ○ | × | ○ | × | × | × | × |
| ACTST S0 | × | × | × | ○ | ○ | ○ | ○ |
| BF LABEL | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| BFR S0 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| BRA LABEL | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| BRAR S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| BRCID | ○ | ○ | ○ | × | × | × | × |
| BRCID IMM | × | × | × | ○ | ○ | ○ | ○ |
| BRTID IMM | × | × | × | ○ | ○ | ○ | ○ |
| BT LABEL | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| BTR S0 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| BTST D0, S0, S1 | × | × | × | × | × | × | ○ |
| BTSTU D0, S0, IMM | × | × | × | × | × | × | ● |
| BTSTI D0, S0, IMM | × | × | × | × | × | × | ○ |
| CALL D0, LABEL | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| CALLA PCR, LABEL | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| CALLR D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| CMP.EQ D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.EQ S0, S1 | ○ | ○ | ○ | × | × | × | × |
| CMP.EQ8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.EQ16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.GE S0, S1 | ○ | ○ | ○ | × | × | × | × |
| CMP.GT S0, S1 | × | × | ○ | × | × | × | × |
| CMP.LE S0, S1 | × | × | ○ | × | × | × | × |
| CMP.LT S0, S1 | ○ | ○ | ○ | × | × | × | × |
| CMP.NEQ D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.NEQ S0, S1 | ○ | ○ | ○ | × | × | × | × |
| CMP.SGE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.SGT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.SLE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.SLE8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.SLE16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.SLT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CMP.ULE8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.ULE16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.NEQ8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMP.NEQ16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMPU.EQ D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.EQ D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| CMPU.EQ S0, IMM | ● | ● | ● | × | × | × | × |
| CMPU.EQ S0, LABEL | ● | ● | ● | × | × | × | × |
| CMPU.GE S0, IMM | ● | ● | ● | × | × | × | × |
| CMPU.GE S0, LABEL | ● | ● | ● | × | × | × | × |
| CMPU.GT S0, IMM | × | × | ● | × | × | × | × |
| CMPU.GT S0, LABEL | × | × | ● | × | × | × | × |
| CMPU.LE S0, IMM | × | × | ● | × | × | × | × |
| CMPU.LE S0, LABEL | × | × | ● | × | × | × | × |
| CMPU.LT S0, IMM | ● | ● | ● | × | × | × | × |
| CMPU.LT S0, LABEL | ● | ● | ● | × | × | × | × |
| CMPU.NEQ D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.NEQ D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| CMPU.NEQ S0, IMM | ● | ● | ● | × | × | × | × |
| CMPU.NEQ S0, LABEL | ● | ● | ● | × | × | × | × |
| CMPU.SGE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.SGE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| CMPU.SGT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.SGT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Control Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| CMPU.SLE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.SLE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| CMPU.SLT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| CMPU.SLT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| DMAWAITS IMM | × | × | × | × | × | × | ○ |
| FBFR S0 | × | × | ○ | × | × | × | × |
| FBRAR S0 | × | × | ○ | × | × | × | × |
| FBTR S0 | × | × | ○ | × | × | × | × |
| FCALLR D0, S0 | × | × | ○ | × | × | × | × |
| FCMP.EQ S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.GE S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.GT S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.LE S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.LT S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.NEQ S0, S1 | × | × | ○ | × | × | × | × |
| FCMP.O D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.OEQ D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.OGE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.OGT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.OLE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.OLT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.ONE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.UEQ D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.UGE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.UGT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.ULE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.ULT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.UNE D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMP.UO D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FCMPU.EQ S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.EQ S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.GE S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.GE S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.GT S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.GT S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.LE S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.LE S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.LT S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.LT S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.NEQ S0, IMM | × | × | ● | × | × | × | × |
| FCMPU.NEQ S0, LABEL | × | × | ● | × | × | × | × |
| FCMPU.O D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.O D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.OEQ D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.OEQ D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.OGE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.OGE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Control Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| FCMPU.OGT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.OGT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.OLE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.OLE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.OLT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.OLT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.ONE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.ONE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.UEQ D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.UEQ D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.UGE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.UGE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.UGT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.UGT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.ULE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.ULE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.ULT D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.ULT D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.UNE D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.UNE D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| FCMPU.UO D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| FCMPU.UO D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| LPE | × | × | × | × | × | × | ○ |
| LPS S0, IMM | × | × | × | × | × | × | ○ |
| LMBD D0, S0 | × | × | × | × | × | × | ○ |
| INT | × | × | × | ○ | ○ | ○ | ○ |
| PCMPAGT D0, S0, S1 | × | × | × | × | × | × | ○ |
| PCMPALT D0, S0, S1 | × | × | × | × | × | × | ○ |
| PCMPOGT D0, S0, S1 | × | × | × | × | × | × | ○ |
| PCMPOLT D0, S0, S1 | × | × | × | × | × | × | ○ |
| RET D0 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| RETA PCR | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| RMDB D0, S0 | × | × | × | × | × | × | ○ |
| SYNC | ○ | ○ | × | × | × | × | × |
| SYNCG | × | × | × | ○ | ○ | ○ | ○ |
| SYNCP IMM | × | × | × | ○ | ○ | ○ | ○ |
| TRAP | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| WAIT | ○ | ○ | × | × | × | × | × |
| WAITDMA | × | × | × | ○ | ○ | ○ | ○ |

### Data transfer instructions

Notes: \*1 LDHSI, LDHUI, and STHI instructions: These instructions use the IMM operand value shifted to the right by one bit.

\*2 LDI and STI instructions: These instructions use the IMM operand value shifted to the right by two bits.

\*3 LDLLI and STLLI instructions: These instructions use the IMM operand value shifted to the right by three bits.

\*4 MOVI instruction: This instruction uses the lower 16 bits of the IMM or LABEL operand value.

\*5 SETHI instruction: This instruction uses the upper 16 bits of the IMM or LABEL operand value.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Transfer Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| CMV8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| CMV16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| FLDC D0, IMM | × | × | ○ | × | × | × | × |
| FLDCI D0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| FLDRCI D0, IMM, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| FLDRCIU D0, IMM, IMM | × | × | × | ○ | ○ | ○ | ○ |
| FLDRXY D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FLDRXYU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| FLDS D0, S0 | × | × | ○ | × | × | × | × |
| FLDW D0, S0, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| FSTC CR, S0 | × | × | ○ | ○ | ○ | ○ | ○ |
| FSTRCI D0, IMM, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| FSTRCIU D0, IMM, IMM | × | × | × | ○ | ○ | ○ | ○ |
| FSTRXY D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FSTRXYU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LD D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDBS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDBSI D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| LDBU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDBUI D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| LDC D0, IMM | ○ | ○ | ○ | × | × | × | × |
| LDCI D0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| LDHS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDHSI D0, S0, IMM | × | × | × | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| LDHU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDHUI D0, S0, IMM | × | × | × | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| LDI D0, S0, IMM | × | × | × | ○\*2 | ○\*2 | ○\*2 | ○\*2 |
| LDLL D0, S0, S1 | × | × | × | × | ○ | ○ | ○ |
| LDLLI D0, S0, IMM | × | × | × | × | ○\*3 | ○\*3 | ○\*3 |
| LDLRB D0, S0, S1 | × | × | × | × | × | × | ○ |
| LDLRBC D0, S0 | × | × | × | × | × | × | ○ |
| LDLRBI D0, S0, IMM | × | × | × | × | × | × | ○ |
| LDRCI D0, IMM, IMM | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| LDRCIU D0, IMM, IMM | × | × | × | ○ | ○ | ○ | ○ |
| LDRHP D0, S0, IMM | ○ | × | ○ | × | × | × | × |
| LDRXY D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| LDRXYU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| LDSRB D0, S0, S1 | × | × | × | × | × | × | ○ |
| LDSRBC D0, S0 | × | × | × | × | × | × | ○ |
| LDSRBI D0, S0, IMM | × | × | × | × | × | × | ○ |
| LDS D0, S0 | ○ | ○ | ○ | × | × | × | × |
| LDSI D0, S0, IMM | × | × | ○ | × | × | × | × |
| LDW D0, S0, IMM | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| LLROT D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| MOV D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MOVI D0, IMM | ○ | × | ○ | ○\*4 | ○\*4 | ○\*4 | ○\*4 |
| MOVI D0, LABEL | ○ | × | ○ | ○\*4 | ○\*4 | ○\*4 | ○\*4 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Transfer Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| MOVU D0, IMM | ● | ● | ● | ● | ● | ● | ● |
| MOVU D0, LABEL | ● | ● | ● | ● | ● | ● | ● |
| PLSB8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| PLSB16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| RXCH D0, S0 | × | × | × | × | × | × | ○ |
| SBOCRD D0, IMM, IMM | × | ○ | ○ | ○ | ○ | ○ | ○ |
| SETHI D0, IMM | × | × | ○ | ○\*5 | ○\*5 | ○\*5 | ○\*5 |
| SETHI D0, LABEL | × | × | × | ○\*5 | ○\*5 | ○\*5 | ○\*5 |
| ST D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| STB D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| STBI D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| STC CR, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| STCI CR, IMM | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| STH D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| STHI D0, S0, IMM | × | × | × | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| STI D0, S0, IMM | × | × | × | ○\*2 | ○\*2 | ○\*2 | ○\*2 |
| STLL D0, S0, S1 | × | × | × | × | ○ | ○ | ○ |
| STLLI D0, S0, IMM | × | × | × | × | ○\*3 | ○\*3 | ○\*3 |
| STLRB D0, S0, S1 | × | × | × | × | × | × | ○ |
| STLRBC D0, S0 | × | × | × | × | × | × | ○ |
| STLRBI D0, S0, IMM | × | × | × | × | × | × | ○ |
| STRCI D0, IMM, IMM | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| STRCIU D0, IMM, IMM | × | × | × | ○ | ○ | ○ | ○ |
| STRHP D0, IMM, IMM | × | ○ | ○ | × | × | × | × |
| STRXY D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| STRXYU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| STSRB D0, S0, S1 | × | × | × | × | × | × | ○ |
| STSRBC D0, S0 | × | × | × | × | × | × | ○ |
| STSRBI D0, S0, IMM | × | × | × | × | × | × | ○ |
| STS D0, S0 | ○ | ○ | ○ | × | × | × | × |
| STSI D0, S0, IMM | × | × | ○ | × | × | × | × |
| STW D0, S0, IMM | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| VPEBS D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| VPEBU D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| VPEHS D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |
| VPEHU D0, S0, IMM | × | × | × | ○ | ○ | ○ | ○ |

### Arithmetic/logical instructions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic/Logical Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| ABS D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| ABS8 D0, S0 | × | × | × | × | × | × | ○ |
| ABS16 D0, S0 | × | × | × | × | × | × | ○ |
| ADD D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| ADDS D0, S0, S1 | × | × | × | × | × | × | ○ |
| ADDSS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| ADDSS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| ADDSU D0, S0, IMM | × | × | × | × | × | × | ● |
| ADDSI D0, S0, IMM | × | × | × | × | × | × | ○ |
| ADDDBL D0, S0, S1 | × | × | × | × | ○ | ○ | ○ |
| ADDDBLU D0, S0, IMM | × | × | × | × | ● | ● | ● |
| ADDDBLU D0, S0, LABEL | × | × | × | × | ● | ● | ● |
| ADDI D0, S0, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| ADDI D0, S0, LABEL | × | × | ○ | ○ | ○ | ○ | ○ |
| ADDU D0, S0, IMM | ● | ● | ● | ● | ● | ● | ● |
| ADDU D0, S0, LABEL | ● | ● | ● | ● | ● | ● | ● |
| ADDUS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| ADDUS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| AND D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| AND64 D0, S0, S1 | × | × | × | × | × | × | ○ |
| ANDU D0, S0, IMM | ● | ● | ● | ● | ● | ● | ● |
| ANDU D0, S0, LABEL | ● | ● | ● | ● | ● | ● | ● |
| ASM D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| ASMU D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| ASMU D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| CNT0 D0, S0 | ○ | × | ○ | ○ | ○ | ○ | ○ |
| CNT1 D0, S0 | ○ | × | ○ | ○ | ○ | ○ | ○ |
| CSUSS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| CSUSU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| EOR D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| EOR64 D0, S0, S1 | × | × | × | × | × | × | ○ |
| EORU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| EORU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| F2I D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| FABS D0, S0 | × | × | ○ | ○ | ○ | ○ | ○ |
| FADD D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FADDU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FADDU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FAND D0, S0, S1 | × | × | ○ | × | × | × | × |
| FANDU D0, S0, IMM | × | × | ● | × | × | × | × |
| FANDU D0, S0, LABEL | × | × | ● | × | × | × | × |
| FEOR D0, S0, S1 | × | × | ○ | × | × | × | × |
| FEORU D0, S0, IMM | × | × | ● | × | × | × | × |
| FEORU D0, S0, LABEL | × | × | ● | × | × | × | × |
| FFLR D0, S0 | × | × | ○ | ○ | ○ | ○ | ○ |
| FFRC D0, S0 | × | × | ○ | ○ | ○ | ○ | ○ |
| FINV D0, S0 | × | × | ○ | × | × | × | × |
| FMAD D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FMADU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FMADU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FMAX D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FMAXU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FMAXU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FMIN D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FMINU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FMINU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FMP D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic/Logical Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| FMSU D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FMSUU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FMSUU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FMUL D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FMULU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FMULU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| FOR D0, S0, S1 | × | × | ○ | × | × | × | × |
| FORU D0, S0, IMM | × | × | ● | × | × | × | × |
| FORU D0, S0, LABEL | × | × | ● | × | × | × | × |
| FRC D0, S0 | × | ○ | × | × | × | × | × |
| FSEQ D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSEQU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSEQU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSGE D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSGEU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSGEU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSGT D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSGTU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSGTU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSLE D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSLEU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSLEU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSLT D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSLTU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSLTU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSNEQ D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| FSNEQU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSNEQU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| FSUB D0, S0, S1 | × | × | ○ | ○ | ○ | ○ | ○ |
| FSUBU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| FSUBU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| GADDBS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GADDBU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GADDHS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GMADHSL D0, S0, S1 | x | x | x | x | x | x | O |
| GADDHU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GMADHUL D0,S0, S1 | x | x | x | x | x | x | O |
| GMADBS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GMADBSL D0, S0, S1 | x | x | x | x | x | x | O |
| GMADBU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GMADBUL D0, S0, S1 | x | x | x | x | x | x | O |
| GMADHS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GMADHU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSADBS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSADBU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSADHS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSADHU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSUBBS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSUBBU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| GSUBHS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic/Logical Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| GSUBHU D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| I2F D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| INV D0, S0 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| INV64 D0, S0 | × | × | × | × | × | × | ○ |
| L2F D0, S0 | × | × | × | ○ | ○ | ○ | ○ |
| MACS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| MAD D0, S0, S1 | × | ○ | × | × | × | × | × |
| MADU D0, S0, IMM | × | ● | × | × | × | × | × |
| MADU D0, S0, LABEL | × | ● | × | × | × | × | × |
| MAX D0, S0, S1 | ○ | ○ | ○ | × | × | × | × |
| MAXS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| MAXS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MAXS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MAXSU D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| MAXSU D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| MAXU D0, S0, IMM | ● | ● | ● | × | × | × | × |
| MAXU D0, S0, LABEL | ● | ● | ● | × | × | × | × |
| MAXU8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MAXU16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MIN D0, S0, S1 | ○ | ○ | ○ | × | × | × | × |
| MINS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| MINS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MINS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MINSU D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| MINSU D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| MINU D0, S0, IMM | ● | ● | ● | × | × | × | × |
| MINU8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MINU16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MINU D0, S0, LABEL | ● | ● | ● | × | × | × | × |
| MP D0, S0, S1 | × | ○ | × | × | × | × | × |
| MSU D0, S0, S1 | × | ○ | × | × | × | × | × |
| MSUU D0, S0, IMM | × | ● | × | × | × | × | × |
| MSUU D0, S0, LABEL | × | ● | × | × | × | × | × |
| MUL D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| MULS D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULSS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULSS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULSU D0, S0, IMM | × | × | × | × | × | × | ● |
| MULLS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| MULLS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULLS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULLSU D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| MULLSU D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| MULLU8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULLU16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULU D0, S0, IMM | ● | ● | ● | ● | ● | ● | ● |
| MULU D0, S0, LABEL | ● | ● | ● | ● | ● | ● | ● |
| MULUS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| MULUS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| NCSCS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| NOP | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| OR D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| OR64 D0, S0, S1 | × | × | × | × | × | × | ○ |
| ORU D0, S0, IMM | × | × | ● | ● | ● | ● | ● |
| ORU D0, S0, LABEL | × | × | ● | ● | ● | ● | ● |
| PACKB0 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PACKB1 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PACKB2 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PACKB3 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PACKH0 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PACKH1 D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| PCMPAEQ D0, S0, S1 | × | × | × | × | × | × | ○ |
| PCMPOEQ D0, S0, S1 | × | × | × | × | × | × | ○ |
| SCSCS D0, S0, S1 | × | × | × | ○ | ○ | ○ | ○ |
| SEQ D0, S0, S1 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SEQU D0, S0, IMM | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SEQU D0, S0, LABEL | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SGE D0, S0, S1 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SGEU D0, S0, IMM | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Arithmetic/Logical Instruction | VTX | PX | Unified | V3M | V3M2 | V3H | V3U |
| SGEU D0, S0, LABEL | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SGT D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SGTU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SGTU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SHA8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SHA16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SHAI8 D0, S0, IMM | × | × | × | × | × | × | ○ |
| SHAI16 D0, S0, IMM | × | × | × | × | × | × | ○ |
| SHL D0, S0, S1 | ○ | × | ○ | ○ | ○ | ○ | ○ |
| SHL8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SHL16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SHLI D0, S0, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| SHLI8 D0, S0, IMM | × | × | × | × | × | × | ○ |
| SHLI16 D0, S0, IMM | × | × | × | × | × | × | ○ |
| SHLU D0, S0, IMM | × | × | ● | × | × | × | × |
| SHLU D0, S0, LABEL | × | × | ● | × | × | × | × |
| SHRA D0, S0, S1 | ○ | × | ○ | ○ | ○ | ○ | ○ |
| SHRAI D0, S0, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| SHRAU D0, S0, IMM | × | × | ● | × | × | × | × |
| SHRAU D0, S0, LABEL | × | × | ● | × | × | × | × |
| SHRL D0, S0, S1 | ○ | × | ○ | ○ | ○ | ○ | ○ |
| SHRLI D0, S0, IMM | × | × | ○ | ○ | ○ | ○ | ○ |
| SHUF D0, S0, S1 | × | × | × | × | × | × | ○ |
| SHRLU D0, S0, IMM | × | × | ● | × | × | × | × |
| SHRLU D0, S0, LABEL | × | × | ● | × | × | × | × |
| SLE D0, S0, S1 | × | × | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SLEU D0, S0, IMM | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SLEU D0, S0, LABEL | × | × | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SLT D0, S0, S1 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SLTU D0, S0, IMM | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SLTU D0, S0, LABEL | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SNEQ D0, S0, S1 | ○ | ○ | ○ | ○\*1 | ○\*1 | ○\*1 | ○\*1 |
| SNEQU D0, S0, IMM | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SNEQU D0, S0, LABEL | ● | ● | ● | ●\*1 | ●\*1 | ●\*1 | ●\*1 |
| SUB D0, S0, S1 | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| SUBS D0, S0, S1 | × | × | × | × | × | × | ○ |
| SUBSS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SUBSS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SUBRU D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| SUBRU D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| SUBSRU D0, S0, IMM | × | × | × | × | × | × | ● |
| SUBSUR D0, S0, IMM | × | × | × | × | × | × | ● |
| SUBUR D0, S0, IMM | × | × | × | ● | ● | ● | ● |
| SUBUR D0, S0, LABEL | × | × | × | ● | ● | ● | ● |
| SUBUS8 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SUBUS16 D0, S0, S1 | × | × | × | × | × | × | ○ |
| SUBU D0, S0, IMM | ● | ● | ● | × | × | × | × |
| SUBU D0, S0, LABEL | ● | ● | ● | × | × | × | × |
| UPLSB8 D0, S0 | × | × | × | × | × | × | ○ |
| UPLSB16 D0, S0 | × | × | × | × | × | × | ○ |

### Instructions to be replaced

These instructions are valid only when V3M, V3M2, V3H or V3U is specified for the -c option.

The assembler replaces these instructions with the supported instructions as shown in the following table.

|  |  |
| --- | --- |
| Instruction before Replacement | Instruction after Replacement |
| BF LABEL | BRA LABEL, !SR |
| BFR S0 | BRAR S0, !SR |
| BT LABEL | BRA LABEL, SR |
| BTR S0 | BRAR S0, SR |
| FSEQ D0, S0, S1 | FCMP.OEQ D0, S0, S1 |
| FSEQU D0, S0, IMM | FCMPU.OEQ D0, S0, IMM |
| FSEQU D0, S0, LABEL | FCMPU.OEQ D0, S0, LABEL |
| FSGE D0, S0, S1 | FCMP.UGE D0, S0, S1 |
| FSGEU D0, S0, IMM | FCMPU.UGE D0, S0, IMM |
| FSGEU D0, S0, LABEL | FCMPU.UGE D0, S0, LABEL |
| FSGT D0, S0, S1 | FCMP.OGT D0, S0, S1 |
| FSGTU D0, S0, IMM | FCMPU.OGT D0, S0, IMM |
| FSGTU D0, S0, LABEL | FCMPU.OGT D0, S0, LABEL |
| FSLE D0, S0, S1 | FCMP.ULE D0, S0, S1 |
| FSLEU D0, S0, IMM | FCMPU.ULE D0, S0, IMM |
| FSLEU D0, S0, LABEL | FCMPU.ULE D0, S0, LABEL |
| FSLT D0, S0, S1 | FCMP.OLT D0, S0, S1 |
| FSLTU D0, S0, IMM | FCMPU.OLT D0, S0, IMM |
| FSLTU D0, S0, LABEL | FCMPU.OLT D0, S0, LABEL |
| FSNEQ D0, S0, S1 | FCMP.UNE D0, S0, S1 |
| FSNEQU D0, S0, IMM | FCMPU.UNE D0, S0, IMM |
| FSNEQU D0, S0, LABEL | FCMPU.UNE D0, S0, LABEL |
| RET D0 | BRAR S0 |
| SEQ D0, S0, S1 | CMP.EQ D0, S0, S1 |
| SEQU D0, S0, IMM | CMPU.EQ D0, S0, IMM |
| SEQU D0, S0, LABEL | CMPU.EQ D0, S0, LABEL |
| SGE D0, S0, S1 | CMP.SGE D0, S0, S1 |
| SGEU D0, S0, IMM | CMPU.SGE D0, S0, IMM |
| SGEU D0, S0, LABEL | CMPU.SGE D0, S0, LABEL |
| SGT D0, S0, S1 | CMP.SGT D0, S0, S1 |
| SGTU D0, S0, IMM | CMPU.SGT D0, S0, IMM |
| SGTU D0, S0, LABEL | CMPU.SGT D0, S0, LABEL |
| SLE D0, S0, S1 | CMP.SLE D0, S0, S1 |
| SLEU D0, S0, IMM | CMPU.SLE D0, S0, IMM |
| SLEU D0, S0, LABEL | CMPU.SLE D0, S0, LABEL |
| SLT D0, S0, S1 | CMP.SLT D0, S0, S1 |
| SLTU D0, S0, IMM | CMPU.SLT D0, S0, IMM |
| SLTU D0, S0, LABEL | CMPU.SLT D0, S0, LABEL |
| SNEQ D0, S0, S1 | CMP.NEQ D0, S0, S1 |
| SNEQU D0, S0, IMM | CMPU.NEQ D0, S0, IMM |
| SNEQU D0, S0, LABEL | CMPU.NEQ D0, S0, LABEL |

### Instructions using register pairs

A register pair is two general registers having sequential register numbers. A register pair is used when an instruction needs a 64-bit register.

As an operand, specify only the first one of the desired pair of general registers.

Only an even-numbered general register is allowed for an operand, and the assembler checks the specified register number.

Example:

MACS R16, R18, R19 # R16 specifies a register pair.

# R16 and R17 are used as a 64-bit register.

The following table shows the instructions that use register pairs. ● indicates the operand for a register pair.

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Operand | | |
| First | Second | Third |
| ABS16 | D0● | S0● | - |
| ABS8 | D0● | S0● | - |
| ADDDBL | D0● | S0● | S1 |
| ADDDBLU | D0● | S0● | IMM |
| ADDDBLU | D0● | S0● | LABEL |
| ADDSS16 | D0● | S0● | S1● |
| ADDSS8 | D0● | S0● | S1● |
| ADDUS16 | D0● | S0● | S1● |
| ADDUS8 | D0● | S0● | S1● |
| AND64 | D0● | S0● | S1● |
| ASM | D0● | S0● | S1● |
| ASMU | D0● | S0● | IMM |
| ASMU | D0● | S0● | LABEL |
| CMP.EQ16 | D0● | S0● | S1● |
| CMP.EQ8 | D0● | S0● | S1● |
| CMP.NEQ16 | D0● | S0● | S1● |
| CMP.NEQ8 | D0● | S0● | S1● |
| CMP.SLE16 | D0● | S0● | S1● |
| CMP.SLE8 | D0● | S0● | S1● |
| CMP.SLT16 | D0● | S0● | S1● |
| CMP.SLT8 | D0● | S0● | S1● |
| CMP.ULE16 | D0● | S0● | S1● |
| CMP.ULE8 | D0● | S0● | S1● |
| CMP.ULT16 | D0● | S0● | S1● |
| CMP.ULT8 | D0● | S0● | S1● |
| CMV16 | D0● | S0● | S1 |
| CMV8 | D0● | S0● | S1 |
| EOR64 | D0● | S0● | S1● |
| GADDBS | D0 | S0● | S1● |
| GADDBU | D0 | S0● | S1● |
| GADDHS | D0 | S0● | S1● |
| GADDHU | D0 | S0● | S1● |
| GMADBS | D0 | S0● | S1● |
| GMADBSL | D0● | S0● | S1● |
| GMADBU | D0 | S0● | S1● |
| GMADBUL | D0● | S0● | S1● |
| GMADHS | D0 | S0● | S1● |
| GMADHSL | D0● | S0● | S1● |
| GMADHU | D0 | S0● | S1● |
| GMADHUL | D0● | S0● | S1● |
| GSADBS | D0 | S0● | S1● |
| GSADBU | D0 | S0● | S1● |
| GSADHS | D0 | S0● | S1● |
| GSADHU | D0 | S0● | S1● |
| GSUBBS | D0 | S0● | S1● |
| GSUBBU | D0 | S0● | S1● |
| GSUBHS | D0 | S0● | S1● |
| GSUBHU | D0 | S0● | S1● |
| INV64 | D0● | S0● | - |
| L2F | D0 | S0● | - |
| LDLL | D0● | S0 | S1 |
| LDLLI | D0● | S0 | IMM |
| LLROT | D0● | S0● | S1 |
| MACS | D0● | S0 | S1 |
| MAXS16 | D0● | S0● | S1● |
| MAXS8 | D0● | S0● | S1● |
| MAXU16 | D0● | S0● | S1● |
| MAXU8 | D0● | S0● | S1● |
| MINS16 | D0● | S0● | S1● |
| MINS8 | D0● | S0● | S1● |
| MINU16 | D0● | S0● | S1● |
| MINU8 | D0● | S0● | S1● |
| MULLS | D0● | S0 | S1 |
| MULLS16 | D0● | S0 | S1 |
| MULLS8 | D0● | S0 | S1 |
| MULLSU | D0● | S0 | IMM |
| MULLSU | D0● | S0 | LABEL |
| MULLU16 | D0● | S0 | S1 |
| MULLU8 | D0● | S0 | S1 |
| MULSS16 | D0● | S0● | S1● |
| MULSS8 | D0● | S0● | S1● |
| MULUS8 | D0● | S0● | S1● |
| MULUS16 | D0● | S0● | S1● |
| OR64 | D0● | S0● | S1● |
| PACKB0 | D0 | S0● | S1● |
| PACKB1 | D0 | S0● | S1● |
| PACKB2 | D0 | S0● | S1● |
| PACKB3 | D0 | S0● | S1● |
| PCMPAEQ | D0 | S0● | S1● |
| PCMPAGT | D0 | S0● | S1● |
| PCMPALT | D0 | S0● | S1● |
| PCMPOEQ | D0 | S0● | S1● |
| PCMPOGT | D0 | S0● | S1● |
| PCMPOLT | D0 | S0● | S1● |
| RXCH | D0● | S0● | - |
| SHA16 | D0● | S0● | S1● |
| SHA8 | D0● | S0● | S1● |
| SHL16 | D0● | S0● | S1● |
| SHL8 | D0● | S0● | S1● |
| SHUF | D0● | S0● | S1● |
| STLL | D0● | S0 | S1 |
| STLLI | D0● | S0 | IMM |
| SUBSS16 | D0● | S0● | S1● |
| SUBSS8 | D0● | S0● | S1● |
| SUBUS16 | D0● | S0● | S1● |
| SUBUS8 | D0● | S0● | S1● |

### Conditional instruction execution

For all instructions, it is possible to control the execution/non-execution of the instruction by referencing the status register SR.

By writing "SR" at the end of an operand, the instruction is executed if the status register is true, and the instruction is regarded as the NOP instruction if the status register is false.

By writing "!SR" at the end of an operand, the instruction is executed if the status register is false, and the instruction is regarded as the NOP instruction if the status register is true.

Examples:

CMP.EQ R16, R17

ADD R20, R21, R22, SR # When R16 == R17, execute ADD.

SUB R20, R21, R22, !SR # When R16 != R17, execute SUB.

## Assembler Control Instructions

### List

|  |  |
| --- | --- |
| Control instruction | Description |
| .SECTION | Declares a section. |
| .DATA | Reserves 4-byte constant data. |
| .DCW | Reserves 4-byte integer-type constant data. |
| .DCH | Reserves 2-byte integer-type constant data. |
| .DCB | Reserves 1-byte integer-type constant data. |
| .DCF | Reserves 4-byte floating-point type constant data. |
| .DS | Reserves a data area in 4-byte units. |
| .DSB | Reserves a data area in 1-byte units. |
| .ARGSIZE | Specifies an external data size. |
| .SRC | Specifies an input D$ number. |
| .DST | Specifies an output D$ number. |
| .ENTRY | Specifies the ENTRY symbol. |
| .ALIGN | Specifies the address alignment condition. |
| .LINE | Specifies the compiler source information. |
| .\_LINE\_TOP/.\_LINE\_END | Specifies inline\_asm. |
| .PUBLIC | Specifies the PUBLIC symbol. |
| .EXTERN | Specifies the EXTERN symbol. |
| \_\_\_stack/.DS/\_\_\_stackEnd  (three consecutive instructions) | Reserves a data area in the stack and defines labels to indicate the start and end of the given area. |

### Details

##### SECTION

[Format]

.SECTIONΔ<section name>

<section name>: { px | vtx | unified | uniform | .text | stack | workmemory | globalworkmemory

| lwm | lwm\_comm | gwm | gwm\_comm | impc | impc\_comm }

Example:

.SECTION vtx # Declare the vtx section.

[Description]

Declares a section.

[Cautionary notes]

* A restart of a section cannot be specified. That is, there can be only one section with the same section name in a single file.
* Section names in output files and messages are not output as the names with which the sections were declared but as the section names after conversion to lower-case English characters.
* A file cannot contain multiple program sections together. Only one of vtx, px, unified, and .text sections can be specified in a single file.
* The section that can be specified differs depending on the processor type specified in the -c option.

|  |  |  |  |
| --- | --- | --- | --- |
| Shader Type | Program Section | Data Section | Stack Section |
| IMP-X4-VTX | vtx | workmemory,  globalworkmemory | stack |
| IMP-X4-PX | px | workmemory,  globalworkmemory | stack |
| IMP-X5-H3 | unified | workmemory,  globalworkmemory | stack |
| IMP-X5-V3M  IMP-X5-V3M2  IMP-X5-V3H  IMP-X6-V3U | .text | lwm,  lwm\_comm,  gwm,  gwm\_comm | impc,  impc\_comm |

[Difference between the common section and the non-common (thread local) section]

This difference applies only when V3M, V3M2, V3H, or V3U is specified for the -c option.

* Section name

lwm: Local working memory (non-common/thread local)

lwm\_comm: Local working memory (common)

gwm: Global working memory (non-common/thread local)

gwm\_comm: Global working memory (common)

impc: impc (non-common/thread local)

impc\_comm: impc (common)

* Common section

This is a section for the common area where the constant data and data areas are reserved for the size specified through the control instructions.

The common area is used when data used in common by multiple threads should be stored or accessed.

* Non-common section

This is a section for the non-common area where the constant data and data areas are reserved for the number of threads × the size specified through the control instructions.

The non-common area is used when data should be stored and accessed independently for each thread.

The number of threads is 8 for lwm and 32 for gwm and impc.

Example: Data access in the non-common section - for lwm

<Conceptual diagram>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Thread | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| (\*) Address | 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 | 5 |  |  |  | 2 |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  | 1 |  |
| 5 |  |  |  |  |  |  |  |  |

(\*): Absolute address in the address space (word units)

* The non-common area is accessed as if it is divided into strips of the number of threads, each of which has a width of one word.

- When thread 0 accesses address 2, 5 is referenced.

- When thread 4 accesses address 2, 2 is referenced.

In this manner, the access destination differs depending on the thread, and separate data is stored for each thread.

Therefore, the non-common section needs an area for the number of threads × the size specified through the control instruction.

* Allocation of common and non-common data

Common and non-common data are allocated in order to contiguous areas of physical memory.

Example: For DS

<Assembler source>

.SECTION lwm\_comm

a: .DS 16

.SECTION lwm

b: .DS 1

c: .DS 1

<Physical memory image> (object file output image）

0000: a: ds 16 ; 16 words = 64 bytes

0040: b: ds 1 \* 8 ; 1 word \* 8 threads = 32 bytes

0060: c: ds 1 \* 8

0080: ...

<Address map>

LWM (common) : 0xf0000000 - 0xf0003ffc

LWM (non-common): 0xf8000000 - 0xf80007fc

When the areas allocated to the above addresses, the label values are as follows.

Value of label a: 0xf0000000

Value of label b: 0xf8000008

Value of label c: 0xf800000c

<Conceptual diagram>

.section lwm

b: ds 1

c: ds 1

.section lwm\_comm

a: ds 16

64 bytes = 16 \* 32 bits

32 bytes = 32 bits \* 8

32 bytes = 32 bits \* 8

a

b

c

LWM (physical memory)

Address map

a: 16 words

(\*)For common area

b: 1 word

c: 1 word

Specified LWM (common) address

Specified LWM (non-common) address

LWM (common) window

LWM (non-common) window

0xf0000000

0xf80000080

0xf800000c0

0xf0000000

0xf8000008

* How to determine non-common section addresses

The non-common data output size is the specified size × the number of threads, but the non-common addresses are determined without multiplication with the number of threads.

Therefore, when calculating the start address of the non-common area by adding the size of "(\*) For common area" in the above diagram, first obtain the common area size that is not multiplied with the number of threads (divide the common area size by the number of threads), and then add it to the specified non-common start address to determine the actual non-common start address.

* The following describes the access symbol addresses assuming that the start address of LWM (physical memory) is 0x1000.

- Access from thread 2

(1): Address map (absolute value of the byte address in the address space)

(2): LWM (physical memory) address

Value of label a is (1): f0000000 and (2): 0x00001000

Value of label b is (1): f8000008 and (2): 0x00001048

Value of label c is (1): f800000c and (2): 0x00001068

For label a in LWM (common access)

* Specified LWM (common) address (0xf0000000)

+ total size of LWM (common) data output until label a is reached (0x0)

= (1): 0xf0000000

* Start address of LWM (physical memory) (0x1000)

+ total size of LWM (common) data output until label a is reached (0x0)

= (2): 0x00001000

For label c in LWM (non-common access)

* Specified LWM (non-common) address (0xf8000000)

+ (LWM data output size in LWM (common) (0x40)

÷ number of threads in LWM (non-common) (0x8))

+ (total size of LWM (non-common) data output until label c is reached (0x20)

÷ number of threads in LWM (non-common) (0x8)) = (1): 0xf800000c

* Start address of LWM (physical memory) (0x1000)

+ LWM data output size in LWM (common) (0x40)

+ total size of LWM (non-common) data output until label c is reached (0x20)

+ (word index for access for thread 2 (0x2)

× byte (0x4)) = (2): 0x00001068

* Information output by the -M option (label information output)

The label addresses in the address map (absolute values of the byte addresses in the address space) are output.

[Notes]

* The start address of the non-common section is aligned with a 4-byte boundary. Accordingly, the beginning of the physical memory is aligned with a 32-byte boundary (4 bytes x 8 threads) in LWM or a 128-byte boundary (4 bytes x 32 threads) in GWM and IMPC.
* If the section size is not a multiple of 4 when V3M, V3M2, V3H, or V3U is specified for the -c option, the unused area is filled with 0 so that the size becomes a multiple of 4.

##### DATA

[Format]

[symbol:][Δ.DATA] Δ<constant data>[[Δ]"<character string>"]

Examples:

.DATA 1

.DATA 2.0

.DATA 0x7FFF "character string"

[Description]

Reserves memory for constant data.

The constant data is treated as 4-byte data.

Data is aligned with a 4-byte boundary.

.DATA can be omitted.

The character strings that can be written after the constant data do not influence the object file. They influence only the log file that is output when the -d option is specified.

The constant data is output to an object file by the Shader as described below.

* For the VTX Shader

Integer constants and expressions of integer constants are output as integer type data.

Floating-point constants and expressions of floating-point constants are output as floating-point type data.

Label constants and expressions of an integer constant and a label constant are output as integer type data.

* For the PX Shader

All constants are output as floating-point type data.

* For the Unified Shader

Integer constants and expressions of integer constants are output as integer type data.

Floating-point constants and expressions of floating-point constants are output as floating-point type data.

Label constants and expressions of an integer constant and a label constant are output as floating-point type data.

Examples: Label constant LABEL＝1

[For the VTX Shader]

.DATA 1 # 0x00000001

.DATA 0x00000001 # 0x00000001

.DATA 0x00000001+1 # 0x00000002

.DATA 1.0 # 0x3F800000

.DATA 1.0+1.0 # 0x40000000

.DATA LABEL # 0x00000001

.DATA LABEL+1 # 0x00000002

[For the PX Shader]

.DATA 1 # 0x3F800000

.DATA 0x00000001 # 0x3F800000

.DATA 0x00000001+1 # 0x40000000

.DATA 1.0 # 0x3F800000

.DATA 1.0+1.0 # 0x40000000

.DATA LABEL # 0x3F800000

.DATA LABEL+1 # 0x40000000

[For the Unified Shader]

.DATA 1 # 0x00000001

.DATA 0x00000001 # 0x00000001

.DATA 0x00000001+1 # 0x00000002

.DATA 1.0 # 0x3F800000

.DATA 1.0+1.0 # 0x40000000

.DATA LABEL # 0x3F800000

.DATA LABEL+1 # 0x40000000

[Cautionary notes]

* The type of Shader is determined by the -c option and section declaration (.SECTION).
* If neither the vtx nor px section is present when X4 is specified for the -c option, the PX Shader is assumed. When X5, V3M, V3M2, V3H, or V3U is specified for the -c option, the Unified Shader is assumed. The PX Shader is also assumed if only the vtx or px section declaration is present and there is no instruction when X4 is specified for the -c option.
* This control instruction is not allowed in the workmemory, globalworkmemory, or stack section.
* When V3M, V3M2, V3H, or V3U is specified for the -c option, this control instruction can be specified in any section that can be used.

##### DCW

[Format]

[symbol:] Δ.DCW Δ<constant data>

Examples:

.DCW 1

.DCW LABEL

.DCW 1+1

.DCW LABEL+1

[Description]

Reserves memory for integer type constant data.

The constant data is treated as 4-byte data.

An integer constant, a label constant, an expression of integer constants, or an expression of an integer constant and a label constant can be written as constant data.

Data is aligned with a 4-byte boundary.

[Cautionary notes]

* This control instruction is not allowed in the workmemory, globalworkmemory, or stack section.
* When V3M, V3M2, V3H, or V3U is specified for the -c option, this control instruction can be specified in any section that can be used.

##### DCH

[Format]

[symbol:] Δ.DCH Δ<constant data>

Examples:

.DCH 1

.DCH LABEL

.DCH 1+1

.DCH LABEL+1

[Description]

Reserves memory for integer type constant data.

The constant data is treated as 2-byte data.

An integer constant, a label constant, an expression of integer constants, or an expression of an integer constant and a label constant can be written as constant data.

Data is aligned with a 2-byte boundary.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction can be specified in any section that can be used.

##### DCB

[Format]

[symbol:] Δ.DCB Δ<constant data>

Examples:

.DCB 1

.DCB LABEL

.DCB 1+1

.DCB LABEL+1

[Description]

Reserves memory for integer type constant data.

The constant data is treated as 1-byte data.

An integer constant, a label constant, an expression of integer constants, or an expression of an integer constant and a label constant can be written as constant data.

Data is aligned with a 1-byte boundary.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction can be specified in any section that can be used.

##### DCF

[Format]

[symbol:] Δ.DCF Δ<constant data>

Examples:

.DCF 1.0

.DCF LABEL

.DCF 1.0+1.0

[Description]

Reserves memory for floating-point type constant data.

The constant data is treated as 4-byte data.

A floating-point constant, a label constant, or an expression of floating-point constants can be written as constant data.

Data is aligned with a 4-byte boundary.

[Cautionary notes]

* This control instruction is not allowed in the workmemory, globalworkmemory, or stack section.
* When V3M, V3M2, V3H, or V3U is specified for the -c option, this control instruction can be specified in any section that can be used.

##### DS

[Format]

[symbol:] Δ.DS Δ<area size>

Example:

.DS 10

[Description]

Reserves a specified area size.

An integer constant within the range from 0 to1073741823 (0x3fffffff) can be written as an area size.

The size of an area is 32 bits (4 bytes).

Data is aligned with a 4-byte boundary.

When V3M, V3M2, V3H, or V3U is specified for the -c option, the reserved area is filled with 0.

[Caution]

* This control instruction can be written only in the workmemory, globalworkmemory, stack, lwm, lwm\_comm, gwm, gwm\_comm, impc, or impc\_comm section.

##### DSB

[Format]

[symbol:] Δ.DSB Δ<area size>

Example:

.DSB 10

[Description]

Reserves a specified area size.

An integer constant within the range from 0 to1073741823 (0x3fffffff) can be written as an area size.

The size of an area is 8 bits (1 byte).

Data is aligned with a 1-byte boundary.

The reserved area is filled with 0.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction can be written only in the lwm, lwm\_comm, gwm, gwm\_comm, impc, or impc\_comm section.

##### ARGSIZE

[Format]

.ARGSIZEΔ<external data size>

[Description]

Specifies an external data size in bytes.

An integer constant can be written as an external data size.

[Caution]

* There can be only one ARGSIZE specification in a single file.

##### SRC

[Format]

.SRCΔ<D$ number>[, …]

Example:

.SRC 0,2 # Specify 0 and 2 as input D$ numbers.

[Description]

Specifies a specified D$ number(s) as an input D$ number(s).

Integer constants of 0 to 7 can be written as D$ numbers.

[Caution]

* There can be only one SRC specification in a single file.

##### DST

[Format]

.DSTΔ<D$ number>[, …]

Example:

.DST 7 # Specify 7 as an output D$ number.

[Description]

Specifies a specified D$ number(s) as an output D$ number(s).

Integer constants of 0 to 7 can be written as D$ numbers.

[Caution]

* There can be only one DST specification in a single file.

##### ENTRY

[Format]

.ENTRYΔ<symbol>[, {MASTER | SLAVE}]

Example:

.ENTRY \_main, MASTER #\_main is specified as a MASTER ENTRY symbol.

[Description]

Specifies a symbol as an ENTRY symbol.

MASTER and SLAVE must be specified in all uppercase or lowercase letters.

When neither MASTER nor SLAVE is specified, MASTER is assumed.

Multiple symbols can be specified for each of MASTER and SLAVE.

There is no limit on the number of specified symbols.

[Cautionary notes]

* Local symbols are not allowed.
* A symbol starting with a period (.) is not allowed.
* The symbols \_\_stack and \_\_stackEnd are not allowed.
* Only the symbols belonging to the .TEXT section can be specified.
* This control instruction is valid only when -O ELF is specified. When -O IMP is specified, this control instruction is ignored.

##### ALIGN

[Format]

. ALIGNΔ<alignment condition>[, <offset>]

Example:

.section lwm\_comm

.dcb 0x11 # 0x0 relative to section

.align 2 # 0x2 relative to section, 1-byte padding

LABEL: # 0x2 relative to section

.dcb 0x22

[Description]

Adjusts the address (offset value relative to the section) with the specified alignment condition.

The alignment condition must be an integer constant of a power of 2 within the range from 2 (inclusive) to 31 (exclusive).

The <offset> must be an integer greater than or equal to 0.

The unused area generated by this address adjustment is filled with 0.

When -T\_REL is specified, a maximum value greater than 4 as an alignment value (alignment condition) specified with the .ALIGN control instruction for the same section is set for sh\_addralign in the section header of the ELF file.

If <offset> is presented, the current location is aligned to next lowest address of the form:

<offset> + <n> \* <alignment condition>

In which, <n> is any integer which the assembler selects to minimize padding.

If <offset> is not presented, value of <offset> is 0.

If <alignment condition>, ALIGN sets the current location to the next word (four byte) boundary.

The unused area generated by this address adjustment is filled with 0.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction can be specified in any section that can be used.

Example:

* <n>=0

ADDRESS ADDRESS

0x0000 ADD R16, R17, R18 0x0000 ADD R16, R17, R18

0x0001 .ALIGN 16, 12 0x0001 NOP

0x0001 LPS R16, 1 0x0002 NOP

0x0002 LoopHeadBB: 0x0003 LPS R16, 1

0x0002 LDI R17, R18, 0 0x0004 LoopHeadBB:

0x0003 ADD R16, R17, R18 0x0004 LDI R17, R18, 0

0x0005 ADD R16, R17, R18

ADDRESS ADDRESS

0x0000 ADD R16, R17, R18 0x0000 ADD R16, R17, R18

0x0001 .ALIGN 16, 8 0x0001 NOP

0x0001 LPS R16, 1 0x0002 LPS R16, 1

0x0002 LoopHeadBB: 0x0003 LoopHeadBB:

0x0002 LDI R17, R18, 0 0x0003 LDI R17, R18, 0

0x0003 ADD R16, R17, R18 0x0004 ADD R16, R17, R18

* <n> != 0

ADDRESS ADDRESS

0x0000 NOP 0x0000 NOP

0x0001 NOP 0x0001 NOP

0x0002 ADD R16, R17, R18 0x0002 ADD R16, R17, R18

0x0003 .ALIGN 16, 4 0x0003 NOP

0x0003 LPS R16, 1 0x0004 NOP

0x0004 LoopHeadBB: 0x0005 LPS R16, 1

0x0005 LDI R17, R18, 0 0x0006 LoopHeadBB:

0x0006 ADD R16, R17, R18 0x0006 LDI R17, R18, 0

0x0007 ADD R16, R17, R18

ADDRESS ADDRESS

0x0000 NOP 0x0000 NOP

0x0001 NOP 0x0001 NOP

0x0002 ADD R16, R17, R18 0x0002 ADD R16, R17, R18

0x0003 .ALIGN 16, 8 0x0003 NOP

0x0003 LPS R16, 1 0x0004 NOP

0x0004 LoopHeadBB: 0x0005 NOP

0x0005 LDI R17, R18, 0 0x0006 LPS R16, 1

0x0006 ADD R16, R17, R18 0x0007 LoopHeadBB:

0x0007 LDI R17, R18, 0

0x0008 ADD R16, R17, R18

##### .LINE

[Format]

.LINEΔ"<file name>",line number

Example:

C source

1 #pragma entry main master

2 const int a=1,b=2;

3 int c;

4 int func(int p1, int p2)

5 {

6 return p1+p2;

7 }

8

9 void main()

10 {

11 c = func(a,b);

12 }

Assembly source

.section .text

\_func:

# individual function stack size: 0

.LINE "C:/RSO/work/\_IMP-X5/TP/TP14/01.c", 4

add r16, r16, r17

.LINE "C:/RSO/work/\_IMP-X5/TP/TP14/01.c", 7

brar r31

\_main:

.entry \_main, master

# individual function stack size: 0

.LINE "C:/RSO/work/\_IMP-X5/TP/TP14/01.c", 9

movi r2, 0x00000020

movu r17, \_b

movu r16, \_a

call r31, \_func

movi r17, \_c

sethi r17, \_c

sti r16, r17, 0x00000000

trap

.section uniform

\_a:

.dcw 0x00000001

\_b:

.dcw 0x00000002

.section lwm\_comm

\_c:

.dsb 4

.section impc

\_\_\_stack:

.ds 8

\_\_\_stackEnd:

[Description]

When the compiler source information is specified with a file name and a line number, the C-language source file can be referred to during debugging.

For a file name, specify the C-language source file with a full path.

Change the file name and the line number (output the line information) which are referenced during debugging at the position where .LINE has appeared.

Change the file name and the line number (output the line information) only when either of them is different from .LINE that has appeared last.

Before first .LINE is output, the file is recognized as the first file of file\_names in .debug\_line.

.LINE is only enabled when the .db2 file is specified with the -b option and the debug\_line section exists in the .db2 file. In other cases, .LINE is ignored.

If there is no file name of .LINE for file\_names in .debug\_line, an internal error will occur.

Line numbers 0 to 4,294,967,295 can be specified for .LINE.

If 0 is specified for the line number of .LINE, it will mean that the line number is unknown and line information will not be output.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction is valid only when the -d option is specified.
* This control instruction is valid only when the -b option is specified.
* This control instruction can be specified only in the .text section.
* No labels are allowed before this control instruction.

##### .\_LINE\_TOP/.\_LINE\_END

The .\_LINE\_TOP and .\_LINE\_END directives are information for a function specified with the #pragma inline\_asm directive of the compiler.

[Format]

.\_LINE\_TOPΔINLINE\_ASM

.\_LINE\_ENDΔINLINE\_ASM

Example:

C source

int a;

#pragma inline\_asm asm\_func

void asm\_func(void)

{

movu.i r16, \_a

movi r17, 0x00000002

sti r17, r16, 0x00000000

}

int f1(void)

{

a=1;

asm\_func();

return 0;

}

Assembly source

.section .text

\_asm\_func:

# individual function stack size: 0

.line\_top inline\_asm

movu.i r16, \_a

movi r17, 0x00000002

sti r17, r16, 0x00000000

.line\_end inline\_asm

brar r31

\_f1:

# individual function stack size: 0

.line "C:\RSO\work\\_IMP-X5\Test\_sasm\#6441\UT\PCL010\02.c", 11

movu.i r16, \_a

movi r17, 0x00000001

sti r17, r16, 0x00000000

.line "C:\RSO\work\\_IMP-X5\Test\_sasm\#6441\UT\PCL010\02.c", 12

.line\_top inline\_asm

movu.i r16, \_a

movi r17, 0x00000002

sti r17, r16, 0x00000000

.line\_end inline\_asm

movi r16, 0x00000000

.line "C:\RSO\work\\_IMP-X5\Test\_sasm\#6441\UT\PCL010\02.c", 14

brar r31

.section lwm\_comm

.align 4

\_a:

.dsb 4

.section impc

\_\_\_stack:

.ds 8

\_\_\_stackEnd:

[Description]

* These control instructions are information for a function specified with #pragma inline\_asm in a C source program.
* The .\_LINE\_TOP and .\_LINE\_END instructions indicate the first and last instruction strings of a function specified with inline\_asm.
* Assembler control instructions other than the following are not allowed in a function specified as inline\_asm. If other control instructions are included, an error message will be output.
* ata definition instructions: .DATA, .DCB, .DCH, .DCW, and .DCF
* Alignment condition instruction: .ALIGN
* If a label other than a local label is defined for a function specified as inline\_asm, since the label with the same name will appear at the position where the inline\_asm function has been called the function cannot be specified; if it is specified, an error message will be output.

[Cautionary notes]

* This control instruction is valid only when V3M, V3M2, V3H, or V3U is specified for the -c option.
* This control instruction is valid only when the -d option is specified.
* This control instruction is valid only when the -b option is specified.
* This control instruction can be specified only in the .text section.
* No label is allowed to be written before this control instruction.
* Labels in an inline\_asm function are not output as debugging information.

##### PUBLIC

[Format]

.PUBLICΔ<symbol>

Example:

.PUBLIC \_func

[Description]

Declares the specified symbol to allow reference to it by other modules.

[Cautionary notes]

* Declaring a local symbol name (a single numeral) is not allowed.\*
* A symbol starting with a period (.) is not allowed.\*
* The symbols \_\_stack and \_\_stackEnd are not allowed.\*
* If no symbol is defined, a warning message will be output.
* This control instruction is only valid when -O ELF is specified. With -O IMP, it is ignored.
* This control instruction is only valid when -T REL is specified. With -T ABS, it is ignored and the symbol is output as a global label.
* This control instruction can be written anywhere in a source program.
* Even if the same symbol is specified multiple times, no error message will be output.

\*Note: If this is specified, the message F0551028 will be output.

##### EXTERN

[Format]

.EXTERNΔ<symbol>

Example:

.EXTERN \_func

[Description]

Declares a symbol of another module for reference by the given module.

[Cautionary notes]

* Declaring a local symbol name (a single numeral) is not allowed.\*
* A symbol starting with a period (.) is not allowed.\*
* Even if a symbol has been defined, no error message will be output.
* This control instruction is only valid when -O ELF is specified. With -O IMP, it is ignored.
* This control instruction is only valid when -T REL is specified. With -T ABS, it is ignored.
* This control instruction can be written anywhere in a source program.
* Even in case of reference to symbols for other modules without their having been declared as .EXTERN, no error message will be output.
* Even if the same symbol is specified multiple times, no error message will be output.

\*Note: If this is specified, the message F0551028 will be output.

##### \_\_\_stack/.DS/\_\_\_stackEnd (three consecutive instructions)

[Format]

\_\_\_stack:

.DS <size of an area in the stack>

\_\_\_stackEnd:

or

\_\_\_stack: .DS <size of an area in the stack>

\_\_\_stackEnd:

Example:

C source

#pragma stacksize 16 // A value is specified in bytes.

Assembly source

.section impc

\_\_\_stack:

.ds 4 # A value is specified in words.

\_\_\_stackEnd:

Image of dumped object code when -T REL is specified

[Symbol Table]

Info Value Size Section/Name

l- 0x00000000 0x00000000 <UNDEF>

l-file 0x00000000 0x00000000 <ABS>/test.s

l- 0x00000000 0x00000010 <UNDEF>/.STACKSIZE

--------------------------------------------------

[Disassemble List]

# No stack area is output.

--------------------------------------------------

Image of dumped object code when -T ABS is specified

[Symbol Table]

Info Value Size Section/Name

l- 0x00000000 0x00000000 <UNDEF>

l-file 0x00000000 0x00000000 <ABS>/test.s

l-sect 0xed800000 0x00000000 impc/impc

g- 0xed800000 0x00000000 impc/\_\_\_stack

g- 0xed800010 0x00000000 impc/\_\_\_stackEnd

--------------------------------------------------

[Disassemble List]

.section impc

\_\_\_stack:

.DCW 0x00000000

.DCW 0x00000000

.DCW 0x00000000

.DCW 0x00000000

\_\_\_stackEnd:

--------------------------------------------------

[Description]

* The three consecutive instructions, \_\_\_stack, .DS, and .\_\_\_stackEnd, are information for a function specified with #pragma stacksize for the compiler.
* The three consecutive instructions, \_\_\_stack, .DS, and .\_\_\_stackEnd, are only recognized as setting up space in the stack area when -T REL is specified.

- When the size of the areas in the stack specified by .DS is not zero, an internal symbol (local undefined label) ".STACKSIZE" is generated between the assembler and linker.

- For the size of the .STACKSIZE symbol, the value is set to the size of the stack area specified by .DS, but in bytes (the same value as was specified for #pragma stacksize).

- No data area is reserved and the labels \_\_\_stack and \_\_\_stackEnd are not defined.

* When -T ABS is specified, processing is the same as that for normal defined labels and the .DS control instruction. The following cautionary notes thus do not apply.

[Cautionary notes]

* Definitions of the \_\_\_stack and \_\_\_stackEnd labels cannot be written in sequences or locations other than the three consecutive instructions in order, \_\_\_stack, .DS, and \_\_\_stackEnd.\*
* The three consecutive instructions, \_\_\_stack, .DS, and .\_\_\_stackEnd, cannot be written multiple times.\*
* The three consecutive instructions, \_\_\_stack, .DS, and .\_\_\_stackEnd, can only be written in the lwm, lwm\_comm, gwm, gwm\_comm, impc, or impc\_comm section.\*
* .STACKSIZE which may have been written in the source code is an internal label. It differs from .STACKSIZE which is generated by these three consecutive instructions. Even if .STACKSIZE is written in the source code, no error message will be output.

\* If this does not apply, the message F0551010 will be output.

[Supplementary note]

* The linker uses ".STACKSIZE" generated from these three consecutive instructions, calculates the maximum value from the symbol size of ".STACKSIZE" in all input files, and generates a stack area with the required size (maximum value).

For details, refer to the CVengine Compiler (ccimp) - Linker System Specifications.

## Extended Instructions

This function automatically generates, from a single extended instruction, an instruction for reserving constant data in the uniform section and referencing it.

[Supplement]

* The Shader does not offer any instructions that enable one to write 32-bit constant data directly for an operand. For this reason, performing an operation with a 32-bit constant requires that constant data be reserved in the uniform section and an instruction (such as ADDU) referencing the data be written, or MOVI and SETHI be used in combination to load a register with constant data and then an operation be performed.

Note that MOVI and SETHI can be used in combination to load a register with constant data only when V3M, V3M2, V3H, or V3U is specified for the -c option.

[Extended instructions that can be used with the VTX Shader]

CMPU.EQ.I S0, <constant data (int)>

CMPU.GE.I S0, <constant data (int)>

CMPU.LT.I S0, <constant data (int)>

CMPU.NEQ.I S0, <constant data (int)>

MOVU.I D0, <constant data (int)>

ADDU.I D0, S0, <constant data (int)>

ANDU.I D0, S0, <constant data (int)>

MAXU.I D0, S0, <constant data (int)>

MINU.I D0, S0, <constant data (int)>

MULU.I D0, S0, <constant data (int)>

SEQU.I D0, S0, <constant data (int)>

SGEU.I D0, S0, <constant data (int)>

SLTU.I D0, S0, <constant data (int)>

SNEQU.I D0, S0, <constant data (int)>

SUBU.I D0, S0, <constant data (int)>

[Extended instructions that can be used with the PX Shader]

CMPU.EQ.F S0, <constant data (float)>

CMPU.GE.F S0, <constant data (float)>

CMPU.LT.F S0, <constant data (float)>

CMPU.NEQ.F S0, <constant data (float)>

MOVU.F D0, <constant data (float)>

ADDU.F D0, S0, <constant data (float)>

MAXU.F D0, S0, <constant data (float)>

MINU.F D0, S0, <constant data (float)>

MADU.F D0, S0, <constant data (float)>

MSUU.F D0, S0, <constant data (float)>

MULU.F D0, S0, <constant data (float)>

SEQU.F D0, S0, <constant data (float)>

SGEU.F D0, S0, <constant data (float)>

SLTU.F D0, S0, <constant data (float)>

SNEQU.F D0, S0, <constant data (float)>

SUBU.F D0, S0, <constant data (float)>

ANDU.F D0, S0, <constant data (float)>

[Extended instructions that can be used with the Unified Shader (IMP-X5-H3)]

For the Unified Shader (IMP-X5-H3), access to the uniform section is done with R6 + #IMM and therefore the assembler cannot determine the index in the uniform section. When an extended instruction is specified, the assembler handles R6 as 0 to reserve constant data in the uniform section and generate an instruction.

CMPU.EQ.I S0, <constant data (int)>

CMPU.GE.I S0, <constant data (int)>

CMPU.LT.I S0, <constant data (int)>

CMPU.NEQ.I S0, <constant data (int)>

CMPU.GT.I S0, <constant data (int)>

CMPU.LE.I S0, <constant data (int)>

FCMPU.EQ.F S0, <constant data (float)>

FCMPU.GE.F S0, <constant data (float)>

FCMPU.LT.F S0, <constant data (float)>

FCMPU.NEQ.F S0, <constant data (float)>

FCMPU.GT.F S0, <constant data (float)>

FCMPU.LE.F S0, <constant data (float)>

MOVU.I D0, <constant data (int)>

MOVU.F D0, <constant data (float)>

ADDU.I D0, S0, <constant data (int)>

FADDU.F D0, S0, <constant data (float)>

FMADU.F D0, S0, <constant data (float)>

FMAXU.F D0, S0, <constant data (float)>

FMINU.F D0, S0, <constant data (float)>

FMSUU.F D0, S0, <constant data (float)>

FANDU.F D0, S0, <constant data (float)>

FEORU.F D0, S0, <constant data (float)>

FORU.F D0, S0, <constant data (float)>

ANDU.I D0, S0, <constant data (int)>

EORU.I D0, S0, <constant data (int)>

MAXU.I D0, S0, <constant data (int)>

MINU.I D0, S0, <constant data (int)>

MULU.I D0, S0, <constant data (int)>

ORU.I D0, S0, <constant data (int)>

FMULU.F D0, S0, <constant data (float)>

SEQU.I D0, S0, <constant data (int)>

SGEU.I D0, S0, <constant data (int)>

SLTU.I D0, S0, <constant data (int)>

SNEQU.I D0, S0, <constant data (int)>

SGTU.I D0, S0, <constant data (int)>

SLEU.I D0, S0, <constant data (int)>

FSEQU.F D0, S0, <constant data (float)>

FSGEU.F D0, S0, <constant data (float)>

FSLTU.F D0, S0, <constant data (float)>

FSNEQU.F D0, S0, <constant data (float)>

FSGTU.F D0, S0, <constant data (float)>

FSLEU.F D0, S0, <constant data (float)>

SUBU.I D0, S0, <constant data (int)>

FSUBU.F D0, S0, <constant data (float)>

SHLU.I D0, S0, <constant data (int)>

SHRAU.I D0, S0, <constant data (int)>

SHRLU.I D0, S0, <constant data (int)>

[Extended instructions that can be used with the Unified Shader (IMP-X5-V3M)]

For the Unified Shader (IMP-X5-V3M), access to the uniform section is done with R6 + #IMM and therefore the assembler cannot determine the index in the uniform section. When an extended instruction is specified, the assembler handles R6 as 0 to reserve constant data in the uniform section and generate an instruction.

ADDU.I D0, S0, <constant data (int)>

SUBRU.I D0, S0, <constant data (int)>

SUBUR.I D0, S0, <constant data (int)>

MULU.I D0, S0, <constant data (int)>

ANDU.I D0, S0, <constant data (int)>

EORU.I D0, S0, <constant data (int)>

ORU.I D0, S0, <constant data (int)>

MAXSU.I D0, S0, <constant data (int)>

MINSU.I D0, S0, <constant data (int)>

MULLSU.I D0, S0, <constant data (int)>

SGTU.I D0, S0, <constant data (int)>

SLEU.I D0, S0, <constant data (int)>

SGEU.I D0, S0, <constant data (int)>

SLTU.I D0, S0, <constant data (int)>

SEQU.I D0, S0, <constant data (int)>

SNEQU.I D0, S0, <constant data (int)>

CMPU.SGT.I D0, S0, <constant data (int)>

CMPU.SLE.I D0, S0, <constant data (int)>

CMPU.SGE.I D0, S0, <constant data (int)>

CMPU.SLT.I D0, S0, <constant data (int)>

CMPU.EQ.I D0, S0, <constant data (int)>

CMPU.NEQ.I D0, S0, <constant data (int)>

MOVU.I D0, <constant data (int)>

MOVU.F D0, <constant data (float)>

ASMU.I D0, S0, <constant data (int)>

FADDU.F D0, S0, <constant data (float)>

FSUBU.F D0, S0, <constant data (float)>

FMULU.F D0, S0, <constant data (float)>

FMAXU.F D0, S0, <constant data (float)>

FMINU.F D0, S0, <constant data (float)>

FSGTU.F D0, S0, <constant data (float)>

FSLEU.F D0, S0, <constant data (float)>

FSEQU.F D0, S0, <constant data (float)>

FSGEU.F D0, S0, <constant data (float)>

FSLTU.F D0, S0, <constant data (float)>

FSNEQU.F D0, S0, <constant data (float)>

FCMPU.OGT.F D0, S0, <constant data (float)>

FCMPU.UGT.F D0, S0, <constant data (float)>

FCMPU.OLE.F D0, S0, <constant data (float)>

FCMPU.ULE.F D0, S0, <constant data (float)>

FCMPU.OEQ.F D0, S0, <constant data (float)>

FCMPU.UEQ.F D0, S0, <constant data (float)>

FCMPU.OGE.F D0, S0, <constant data (float)>

FCMPU.UGE.F D0, S0, <constant data (float)>

FCMPU.OLT.F D0, S0, <constant data (float)>

FCMPU.ULT.F D0, S0, <constant data (float)>

FCMPU.ONE.F D0, S0, <constant data (float)>

FCMPU.UNE.F D0, S0, <constant data (float)>

FCMPU.O.F D0, S0, <constant data (float)>

FCMPU.UO.F D0, S0, <constant data (float)>

FMADU.F D0, S0, <constant data (float)>

FMSUU.F D0, S0, <constant data (float)>

[Extended instructions that can be used with the Unified Shader (IMP-X5-V3M2 or IMP-X5-V3H)]

For the Unified Shader (IMP-X5-V3M2 or IMP-X5-V3H) extended instruction, the assembler handles access to the uniform section as 0-based relative access to reserve constant data in the uniform section and generate an instruction.

ADDDBLU.I D0, S0, <constant data (int)>

ADDU.I D0, S0, <constant data (int)>

SUBRU.I D0, S0, <constant data (int)>

SUBUR.I D0, S0, <constant data (int)>

MULU.I D0, S0, <constant data (int)>

ANDU.I D0, S0, <constant data (int)>

EORU.I D0, S0, <constant data (int)>

ORU.I D0, S0, <constant data (int)>

MAXSU.I D0, S0, <constant data (int)>

MINSU.I D0, S0, <constant data (int)>

MULLSU.I D0, S0, <constant data (int)>

SGTU.I D0, S0, <constant data (int)>

SLEU.I D0, S0, <constant data (int)>

SGEU.I D0, S0, <constant data (int)>

SLTU.I D0, S0, <constant data (int)>

SEQU.I D0, S0, <constant data (int)>

SNEQU.I D0, S0, <constant data (int)>

CMPU.SGT.I D0, S0, <constant data (int)>

CMPU.SLE.I D0, S0, <constant data (int)>

CMPU.SGE.I D0, S0, <constant data (int)>

CMPU.SLT.I D0, S0, <constant data (int)>

CMPU.EQ.I D0, S0, <constant data (int)>

CMPU.NEQ.I D0, S0, <constant data (int)>

MOVU.I D0, <constant data (int)>

MOVU.F D0, <constant data (float)>

ASMU.I D0, S0, <constant data (int)>

FADDU.F D0, S0, <constant data (float)>

FSUBU.F D0, S0, <constant data (float)>

FMULU.F D0, S0, <constant data (float)>

FMAXU.F D0, S0, <constant data (float)>

FMINU.F D0, S0, <constant data (float)>

FSGTU.F D0, S0, <constant data (float)>

FSLEU.F D0, S0, <constant data (float)>

FSEQU.F D0, S0, <constant data (float)>

FSGEU.F D0, S0, <constant data (float)>

FSLTU.F D0, S0, <constant data (float)>

FSNEQU.F D0, S0, <constant data (float)>

FCMPU.OGT.F D0, S0, <constant data (float)>

FCMPU.UGT.F D0, S0, <constant data (float)>

FCMPU.OLE.F D0, S0, <constant data (float)>

FCMPU.ULE.F D0, S0, <constant data (float)>

FCMPU.OEQ.F D0, S0, <constant data (float)>

FCMPU.UEQ.F D0, S0, <constant data (float)>

FCMPU.OGE.F D0, S0, <constant data (float)>

FCMPU.UGE.F D0, S0, <constant data (float)>

FCMPU.OLT.F D0, S0, <constant data (float)>

FCMPU.ULT.F D0, S0, <constant data (float)>

FCMPU.ONE.F D0, S0, <constant data (float)>

FCMPU.UNE.F D0, S0, <constant data (float)>

FCMPU.O.F D0, S0, <constant data (float)>

FCMPU.UO.F D0, S0, <constant data (float)>

FMADU.F D0, S0, <constant data (float)>

FMSUU.F D0, S0, <constant data (float)>

[Extended instructions that can be used with the Unified Shader (IMP-X6-V3U)]

For the Unified Shader (IMP-X6-V3U) extended instruction, the assembler handles access to the uniform section as 0-based relative access to reserve constant data in the uniform section and generate an instruction.

ADDDBLU.I D0, S0, <constant data (int)>

ADDU.I D0, S0, <constant data (int)>

ADDSU.I D0, S0, <constant data (int)>

BTSTU.I D0, S0, <constant data (int)>

SUBRU.I D0, S0, <constant data (int)>

SUBUR.I D0, S0, <constant data (int)>

SUBSRU D0, S0, <constant data (int)>

SUBSUR D0, S0, <constant data (int)>

MULU.I D0, S0, <constant data (int)>

ANDU.I D0, S0, <constant data (int)>

EORU.I D0, S0, <constant data (int)>

ORU.I D0, S0, <constant data (int)>

MAXSU.I D0, S0, <constant data (int)>

MINSU.I D0, S0, <constant data (int)>

MULLSU.I D0, S0, <constant data (int)>

MULSU.I D0, S0, <constant data (int)>

SGTU.I D0, S0, <constant data (int)>

SLEU.I D0, S0, <constant data (int)>

SGEU.I D0, S0, <constant data (int)>

SLTU.I D0, S0, <constant data (int)>

SEQU.I D0, S0, <constant data (int)>

SNEQU.I D0, S0, <constant data (int)>

CMPU.SGT.I D0, S0, <constant data (int)>

CMPU.SLE.I D0, S0, <constant data (int)>

CMPU.SGE.I D0, S0, <constant data (int)>

CMPU.SLT.I D0, S0, <constant data (int)>

CMPU.EQ.I D0, S0, <constant data (int)>

CMPU.NEQ.I D0, S0, <constant data (int)>

MOVU.I D0, <constant data (int)>

MOVU.F D0, <constant data (float)>

ASMU.I D0, S0, <constant data (int)>

FADDU.F D0, S0, <constant data (float)>

FSUBU.F D0, S0, <constant data (float)>

FMULU.F D0, S0, <constant data (float)>

FMAXU.F D0, S0, <constant data (float)>

FMINU.F D0, S0, <constant data (float)>

FSGTU.F D0, S0, <constant data (float)>

FSLEU.F D0, S0, <constant data (float)>

FSEQU.F D0, S0, <constant data (float)>

FSGEU.F D0, S0, <constant data (float)>

FSLTU.F D0, S0, <constant data (float)>

FSNEQU.F D0, S0, <constant data (float)>

FCMPU.OGT.F D0, S0, <constant data (float)>

FCMPU.UGT.F D0, S0, <constant data (float)>

FCMPU.OLE.F D0, S0, <constant data (float)>

FCMPU.ULE.F D0, S0, <constant data (float)>

FCMPU.OEQ.F D0, S0, <constant data (float)>

FCMPU.UEQ.F D0, S0, <constant data (float)>

FCMPU.OGE.F D0, S0, <constant data (float)>

FCMPU.UGE.F D0, S0, <constant data (float)>

FCMPU.OLT.F D0, S0, <constant data (float)>

FCMPU.ULT.F D0, S0, <constant data (float)>

FCMPU.ONE.F D0, S0, <constant data (float)>

FCMPU.UNE.F D0, S0, <constant data (float)>

FCMPU.O.F D0, S0, <constant data (float)>

FCMPU.UO.F D0, S0, <constant data (float)>

FMADU.F D0, S0, <constant data (float)>

FMSUU.F D0, S0, <constant data (float)>

[Description]

* An extended instruction outputs an instruction for defining constants and referencing them.
* (If ADDU.I is written, the instruction ADDU is actually output.)
* An integer constant, a label constant, an expression of integer constants, or an expression of an integer constant and a label constant can be written as <constant data (int)>.
* For a label constant and an expression of an integer constant and a label constant, its address value is recognized as an integer value.
* An integer constant, a floating-point constant, an expression of floating-point constants, or a label constant can be written as <constant data (float)>.
* An integer constant is recognized as an internal representation value of a floating-point number.
* For a label constant, its address value is recognized as a floating-point value.
* By writing "SR" or "!SR" at the end of an operand, an extended instruction can be written as being subject to conditional instruction execution.

Example 1: Output object file example from an extended instruction

[Input assembler source]

ADDU.I R16, R17, 10 # R16 = R17 + 10

[Output object file]

.SECTION uniform

.DCW 0x0000000A # Reserve constant data in uniform (Index=0).

:

ADDU R16, R17, 0 # Output the ADDU instruction

# (addition to the Index=0 data).

Example 2: Constant data output example

For a label constant LABEL = 1

[For the VTX Shader]

# uniform data

Integer value (decimal): ADDU.I R16, R17, 10 # 0x0000000A

Integer value (hexadecimal): ADDU.I R16, R17, 0xA # 0x0000000A

Floating-point value: ADDU.I R16, R17, 1.0 # Error

Label constant : ADDU.I R16, R17, LABEL # 0x00000001

[For the PX Shader]

# uniform data

Integer value (decimal): ADDU.F R16, R17, 10 # 0x0000000A

Integer value (hexadecimal): ADDU.F R16, R17, 0xA # 0x0000000A

Floating-point value: ADDU.F R16, R17, 1.0 # 0x3F800000(1.0)

Label constant : ADDU.F R16, R17, LABEL # 0x3F800000(1.0)

[Commonalizing constant data]

If there are several extended instructions referencing the same constant data, commonalize the constant data and reserve it in the uniform section. Only the constant data generated from extended instructions can be commonalized, and it cannot be commonalized with the constant data reserved with the .DCW control instruction and others.

For label constants, commonalize those of labels with the same name. They cannot be commonalized with constants that match address values or with the constants of labels with different names that indicate the same address value.

Examples:

[Input assembler source code]

.SECTION uniform

.DCF 1.0

LABEL1:

LABEL2:

.SECTION px

MOVU.F R16, 1.0 # (1)

MOVU.F R17, 1.0 # (2)

MOVU.F R16, LABEL1 # (3)

MOVU.F R17, LABEL1 # (4)

MOVU.F R17, LABEL2 # (5)

[Output object file]

.SECTION uniform

.DCF 1.0

LABEL1:

LABEL2:

.DCF 1.0 # Constant 1.0 in (1) and (2)

.DCF 1.0 # Label constant LABEL1 (Index=1) in (3) and (4)

.DCF 1.0 # Label constant LABEL2 in (5)

.SECTION px

MOVU R16, 1 # (1)

MOVU R17, 1 # (2)

MOVU R16, 2 # (3)

MOVU R17, 2 # (4)

MOVU R17, 3 # (5)

# Assembler Errors

## Message Format

[<file name>(<line number>):]<level>055<message number>:<message>

* <file name>(<line number>) is displayed if the name of the file and the number of the line in which the error occurred can be identified.
* <level> is one of the alphabetic characters below.

|  |  |  |
| --- | --- | --- |
| Character | Level | Remarks |
| C | Internal error | Processing is terminated (suspended). |
| F | Fatal error | Processing is terminated (suspended). |
| E | Error | Processing continues. |
| W | Warning | Processing continues. |

* "055" is a fixed value assigned as part of an error number of the assembler.

("05" is a component number = build phase, and "5" is an occurrence phase = assembler)

Example 1) If a message contains the file name and line number

file.s(123): F0551004:Syntax error.

Example 2) If a message does not contain the file name or line number

F0551002: Cannot open source file "file.s".

## Messages

|  |  |  |
| --- | --- | --- |
| C0554001 | Message | Internal Error. |
| Description | An internal error occurred. Contact your sales representative. |
| E0552001 | Message | Source file is multiple specified. |
| Description | Multiple source files cannot be specified. |
| E0552002 | Message | Illegal register number "register number". |
| Description | The register number is illegal. |
| E0552003 | Message | Immediate value is out of range "immediate value". |
| Description | The immediate value is out of range. |
| E0552004 | Message | Invalid instruction in "section" section. |
| Description | Instruction that cannot be used in the specified section. |
| E0552005 | Message | Label is undefined "label". |
| Description | The undefined label is referenced. |
| E0552006 | Message | "keyword" multiple definition. |
| Description | The keyword is defined more than once. |
| E0552007 | Message | Duplicate label "label". |
| Description | The label is defined more than once. |
| E0552008 | Message | Cannot create file "file name". |
| Description | The output file cannot be opened. |
| E0552009 | Message | "section 1" cannot be combined with "section 2". |
| Description | The VTX, PX, and UNIFIED cannot be used together. |
| E0552010 | Message | Illegal character. |
| Description | An unusable character appeared. |

|  |  |  |
| --- | --- | --- |
| E0552011 | Message | Invalid instruction "instruction". |
| Description | The instruction cannot be used. **(Not used)** |
| E0552012 | Message | Cannot write file "file name". |
| Description | It is not possible to write to the output file. |
| E0552013 | Message | "memory" size overflow. |
| Description | The section size exceeds the memory size. |
| E0552014 | Message | Invalid entry label "label". |
| Description | The label cannot be specified as an ENTRY label. |
| E0552015 | Message | Cannot open file "file name". |
| Description | The file cannot be opened. |
| E0552016 | Message | Cannot read file "file name". |
| Description | The file cannot be read. |
| E0552017 | Message | Illegal file format "file name". |
| Description | The file format is illegal. |
| E0552018 | Message | Syntax error in expression. |
| Description | An operand or an expression of an operand has an error. |
| E0552019 | Message | The 1st term should be label. |
| Description | The first term of an expression of an operand including a label must be a label. |
| E0552020 | Message | Invalid address value specified in option "option" : "address". |
| Description | "address" specified in “option” is an invalid value. |
| E0552021 | Message | Invalid instruction in inline\_asm. |
| Description | An instruction that cannot be used in inline\_asm has been specified. |
| E0552022 | Message | Invalid "label" specified in "control instruction". |
| Description | An invalid label has been specified for the given control instruction. |
| E0552023 | Message | The "option" option must be specified. |
| Description | Specify "option". |
| F0551001 | Message | No source files specified. |
| Description | No source file is specified. |
| F0551002 | Message | Cannot open source file "file name". |
| Description | The source file cannot be opened. |
| F0551003 | Message | Invalid option "option". |
| Description | The invalid option is specified. |
| F0551004 | Message | Syntax error. |
| Description | Syntax error. |
| F0551005 | Message | Division by zero. |
| Description | Division by 0 is found. |
| F0551006 | Message | Syntax error. (The instruction is not supported for current processor) |
| Description | The instruction is not supported for current processor. |
| F0551007 | Message | Syntax error. (The instruction is not supported for current program section) |
| Description | The instruction is not supported for current program section. |
| F0551008 | Message | Syntax error. (Only master or slave can be set for entry) |
| Description | Only master or slave can be set for entry. |
| F0551009 | Message | Syntax error. (The section is not supported for current processor) |
| Description | The section is not supported for current processor. |
| F0551010 | Message | Syntax error. (The section is not supported for ELF format) |
| Description | The section is not supported for ELF format. |
| F0551011 | Message | Syntax error. (Statements of setting stack size are not correct) |
| Description | Statements of setting stack size are not correct. |
| F0551012 | Message | Syntax error. (The definition of label must be set inside section) |
| Description | The definition of label must be set inside section. |
| F0551013 | Message | Syntax error. (The constant data must be set in section) |
| Description | The constant data must be set in section. |
| F0551014 | Message | Syntax error. (Appearance order of control instructions is not correct) |
| Description | Appearance order of control instructions is not correct. |
| F0551015 | Message | Syntax error. (R6 bit offset is not supported for current processor) |
| Description | R6 bit offset is not supported for current processor. |
| W0553001 | Message | Invalid register number "register name". |
| Description | "register name" is illegal. It is not possible to write to the "register name". |
| W0553002 | Message | Invalid register number "register number". |
| Description | "register number" is illegal. It is not possible to read from the "register number". |
| W0553003 | Message | Invalid register number "PCR register". |
| Description | The invalid PCR register is specified. (**Not used**) |
| W0553004 | Message | Invalid register number "register". |
| Description | The register number is invalid. |
| W0553005 | Message | Invalid immediate value "immediate value". |
| Description | The invalid immediate value is specified. |
| W0553006 | Message | Invalid option "option". |
| Description | An invalid option has been specified. |
| W0553007 | Message | "label" referenced in "control instruction" is undefined. |
| Description | The label referred to by the control instruction is undefined. |

## Error Messages for Determining Licenses

When errors related to licenses occur, processing will be terminated.

Error messages for licenses are sent to the standard error output.

