



OVP Guide to Using Processor Models

Model specific information for Andes_AX25F

Imperas Software Limited
Imperas Buildings, North Weston
Thame, Oxfordshire, OX9 2HA, U.K.
docs@imperas.com



Author	Imperas Software Limited
Version	20191106.0
Filename	OVP_Model_Specific_Information_andes_riscv_AX25F.pdf
Created	7 November 2019
Status	OVP Standard Release

Copyright Notice

Copyright (c) 2019 Imperas Software Limited. All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the readers responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Contents

1	Overview	1
1.1	Description	1
1.2	Licensing	1
1.3	Extensions	1
1.3.1	Available (But Not Enabled) Extensions	2
1.4	General Features	2
1.5	Floating Point Features	3
1.6	Interrupts	4
1.7	Debug Mask	4
1.8	Integration Support	5
1.8.1	CSR Register External Implementation	5
1.8.2	LR/SC Active Address	5
1.9	Limitations	5
1.10	Verification	6
1.11	References	6
2	Andes-Specific Extensions	7
2.1	Andes-Specific Parameters	7
2.1.1	Parameter andesExtensions/mmsc_cfg	7
2.1.2	Parameter andesExtensions/micm_cfg	8
2.1.3	Parameter andesExtensions/mdcm_cfg	8
2.1.4	Parameter andesExtensions/uitb	8
2.1.5	Parameter andesExtensions/milmb	8
2.1.6	Parameter andesExtensions/milmbMask	9
2.1.7	Parameter andesExtensions/mdlmb	9
2.1.8	Parameter andesExtensions/mdlmbMask	9
2.2	Hardware Stack Protection	9
2.3	Performance Throttling	9
2.4	CSRs for CCTL Operations	9
2.5	Andes-Specific Instructions	9
2.5.1	Performance Extension Instructions	10
2.5.1.1	ADDIGP	10
2.5.1.2	BBC	10
2.5.1.3	BBS	10
2.5.1.4	BEQC	10
2.5.1.5	BNEC	10
2.5.1.6	BFOS	11

2.5.1.7	BFOZ	11
2.5.1.8	LEA.h	11
2.5.1.9	LEA.w	11
2.5.1.10	LEA.d	11
2.5.1.11	LEA.b.ze	11
2.5.1.12	LEA.h.ze	12
2.5.1.13	LEA.w.ze	12
2.5.1.14	LEA.d.ze	12
2.5.1.15	LBGP	12
2.5.1.16	LBUGP	12
2.5.1.17	LHGP	12
2.5.1.18	LHUGP	13
2.5.1.19	LWGP	13
2.5.1.20	LWUGP	13
2.5.1.21	LDGP	13
2.5.1.22	SBGP	13
2.5.1.23	SHGP	14
2.5.1.24	SWGp	14
2.5.1.25	SDGP	14
2.5.1.26	FFB	14
2.5.1.27	FFZMISM	14
2.5.1.28	FFMISM	15
2.5.1.29	FLMISM	15
2.5.2	CodeDense Instructions	15
2.5.2.1	EXEC.IT	15
2.5.2.2	EX9.IT	15
2.5.3	Half-Precision Floating-Point Instructions	15
2.5.3.1	FLHW	15
2.5.3.2	FSHW	15
3	Configuration	16
3.1	Location	16
3.2	GDB Path	16
3.3	Semi-Host Library	16
3.4	Processor Endian-ness	16
3.5	QuantumLeap Support	16
3.6	Processor ELF code	16
4	All Variants in this model	17
5	Bus Master Ports	18
6	Bus Slave Ports	19
7	Net Ports	20
8	FIFO Ports	21
9	Formal Parameters	22

9.1	Extension Parameters	23
9.2	Parameters with enumerated types	23
9.2.1	Parameter user_version	23
9.2.2	Parameter priv_version	23
9.2.3	Parameter mstatus_fs_mode	24
10	Execution Modes	25
11	Exceptions	26
12	Hierarchy of the model	27
12.1	Level 1: Hart	27
13	Model Commands	28
13.1	Level 1: Hart	28
13.1.1	dumpTLB	28
13.1.1.1	Argument description	28
13.1.2	isync	28
13.1.3	itrace	28
14	Registers	29
14.1	Level 1: Hart	29
14.1.1	Core	29
14.1.2	Floating_point	30
14.1.3	User_Control_and_Status	30
14.1.4	Supervisor_Control_and_Status	31
14.1.5	Machine_Control_and_Status	31
14.1.6	Integration_support	34

Chapter 1

Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance. Most models are provided as a binary shared object and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model.

The models are run through an extensive QA and regression testing process and most model families are validated using technology provided by the processor IP owners. There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

1.1 Description

RISC-V AX25F 64-bit processor model

1.2 Licensing

This Model is released under the Open Source Apache 2.0

1.3 Extensions

The model has the following architectural extensions enabled, and the following bits in the misa CSR Extensions field will be set upon reset:

misa bit 0: extension A (atomic instructions)

misa bit 2: extension C (compressed instructions)
misa bit 3: extension D (double-precision floating point)
misa bit 5: extension F (single-precision floating point)
misa bit 8: RV32I/64I/128I base ISA
misa bit 12: extension M (integer multiply/divide instructions)
misa bit 18: extension S (Supervisor mode)
misa bit 20: extension U (User mode)
misa bit 23: extension X (non-standard extensions present)

To specify features that can be dynamically enabled or disabled by writes to the misa register in addition to those listed above, use parameter “add_Extensions_mask”. This is a string parameter containing the feature letters to add; for example, value “DV” indicates that double-precision floating point and the Vector Extension can be enabled or disabled by writes to the misa register.

Legacy parameter “misa_Extensions_mask” can also be used. This Uns32-valued parameter specifies all writable bits in the misa Extensions field, replacing any value defined in the base variant.

Note that any features that are indicated as present in the misa mask but absent in the misa will be ignored. See the next section.

1.3.1 Available (But Not Enabled) Extensions

The following extensions are supported by the model, but not enabled by default in this variant:

misa bit 4: RV32E base ISA (NOT ENABLED)
misa bit 13: extension N (user-level interrupts) (NOT ENABLED)
misa bit 21: extension V (vector instructions) (NOT ENABLED)

To add features from this list to the base variant, use parameter “add_Extensions”. This is a string parameter containing the feature letters to add; for example, value “DV” indicates that double-precision floating point and the Vector Extension should be enabled, if they are absent.

Legacy parameter “misa_Extensions” can also be used. This Uns32-valued parameter specifies the reset value for the misa CSR Extensions field, replacing any value defined in the base variant.

1.4 General Features

On this variant, the Machine trap-vector base-address register (mtvec) is writable. It can instead be configured as read-only using parameter “mtvec_is_ro”.

Values written to “mtvec” are masked using the value 0xffffffffffffd. A different mask of writable bits may be specified using parameter “mtvec_mask” if required. In addition, when Vectored interrupt mode is enabled, parameter “tvec_align” may be used to specify additional hardware-enforced base address alignment. In this variant, “tvec_align” defaults to 0, implying no alignment

constraint.

The initial value of “mtvec” is 0x0. A different value may be specified using parameter “mtvec” if required.

Values written to “stvec” are masked using the value 0xffffffffffffd. A different mask of writable bits may be specified using parameter “stvec_mask” if required. parameter “tvec_align” may be used to specify additional hardware-enforced base address alignment in the same manner as for the “mtvec” register, described above.

On reset, the model will restart at address 0x0. A different reset address may be specified using parameter “reset_address” if required.

On an NMI, the model will restart at address 0x0. A different NMI address may be specified using parameter “nmi_address” if required.

WFI will halt the processor until an interrupt occurs. It can instead be configured as a NOP using parameter “wfi_is_nop”. WFI timeout wait is implemented with a time limit of 0 (i.e. WFI causes an Illegal Instruction trap in Supervisor mode when mstatus.TW=1).

The “cycle” CSR is implemented in this variant. Set parameter “cycle_undefined” to True to instead specify that “cycle” is unimplemented and reads of it should trap to Machine mode.

The “time” CSR is implemented in this variant. Set parameter “time_undefined” to True to instead specify that “time” is unimplemented and reads of it should trap to Machine mode. Usually, the value of the “time” CSR should be provided by the platform - see notes below about the artifact “CSR” bus for information about how this is done.

The “instret” CSR is implemented in this variant. Set parameter “instret_undefined” to True to instead specify that “instret” is unimplemented and reads of it should trap to Machine mode.

A 0-bit ASID is implemented. Use parameter “ASID_bits” to specify a different implemented ASID size if required.

This variant supports address translation modes 0, 8 and 9. Use parameter “Sv_modes” to specify a bit mask of different modes if required.

Unaligned memory accesses are not supported by this variant. Set parameter “unaligned” to “T” to enable such accesses.

Unaligned memory accesses are not supported for AMO instructions by this variant. Set parameter “unalignedAMO” to “T” to enable such accesses.

A PMP unit is not implemented by this variant. Set parameter “PMP_registers” to indicate that the unit should be implemented with that number of PMP entries.

LR/SC instructions are implemented with a 1-byte reservation granule. A different granule size may be specified using parameter “lr_sc_grain”.

1.5 Floating Point Features

The D extension is enabled in this variant independently of the F extension. Set parameter “d_requires_f” to “T” to specify that the D extension requires the F extension to be enabled.

By default, the processor starts with floating-point instructions disabled (`mstatus.FS=0`). Use parameter “`mstatus_FS`” to force `mstatus.FS` to a non-zero value for floating-point to be enabled from the start.

The specification is imprecise regarding the conditions under which `mstatus.FS` is set to Dirty state (3). Parameter “`mstatus_fs_mode`” can be used to specify the required behavior in this model, as described below.

If “`mstatus_fs_mode`” is set to “`always_dirty`” then the model implements a simplified floating point status view in which `mstatus.FS` holds values 0 (Off) and 3 (Dirty) only; any write of values 1 (Initial) or 2 (Clean) from privileged code behave as if value 3 was written.

If “`mstatus_fs_mode`” is set to “`write_1`” then `mstatus.FS` will be set to 3 (Dirty) by any explicit write to the `fflags`, `frm` or `fcsr` control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion to integer/unsigned that signals a floating point exception. Floating point compare or conversion to integer/unsigned instructions that do not signal an exception will not set `mstatus.FS`.

If “`mstatus_fs_mode`” is set to “`write_any`” then `mstatus.FS` will be set to 3 (Dirty) by any explicit write to the `fflags`, `frm` or `fcsr` control registers, or by any executed instruction that writes an FPR, or by any executed floating point compare or conversion even if those instructions do not signal a floating point exception.

In this variant, “`mstatus_fs_mode`” is set to “`write_1`”.

1.6 Interrupts

The “`reset`” port is an active-high reset input. The processor is halted when “`reset`” goes high and resumes execution from the reset address specified using the “`reset_address`” parameter when the signal goes low. The “`mcause`” register is cleared to zero.

The “`nmi`” port is an active-high NMI input. The processor is halted when “`nmi`” goes high and resumes execution from the address specified using the “`nmi_address`” parameter when the signal goes low. The “`mcause`” register is cleared to zero.

All other interrupt ports are active high.

1.7 Debug Mask

It is possible to enable model debug messages in various categories. This can be done statically using the “`override_debugMask`” parameter, or dynamically using the “`debugflags`” command. Enabled messages are specified using a bitmask value, as follows:

Value 0x002: enable debugging of PMP and virtual memory state;

Value 0x004: enable debugging of interrupt state.

All other bits in the debug bitmask are reserved and must not be set to non-zero values.

1.8 Integration Support

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

1.8.1 CSR Register External Implementation

If parameter “enable_CSR_bus” is True, an artifact 16-bit bus “CSR” is enabled. Slave callbacks installed on this bus can be used to implement modified CSR behavior (use `opBusSlaveNew` or `icmMapExternalMemory`, depending on the client API). A CSR with index `0xABC` is mapped on the bus at address `0xABC0`; as a concrete example, implementing CSR “time” (number `0xC01`) externally requires installation of callbacks at address `0xC010` on the CSR bus.

1.8.2 LR/SC Active Address

Artifact register “LRSCAddress” shows the active LR/SC lock address. The register holds all-ones if there is no LR/SC operation active.

1.9 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. `fence.i`) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous. Data barrier instructions (e.g. `fence`) are treated as NOPs, with the exception of any Illegal Instruction behavior, which is modeled.

Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

The processor fully supports the architecturally-specified floating-point instructions.

Hardware Performance Monitor and Debug registers are not implemented and hardwired to zero.

The TLB is architecturally-accurate but not device accurate. This means that all TLB maintenance and address translation operations are fully implemented but the cache is larger than in the real device.

This variant is under development. It includes Supervisor mode and associated standard RISC-V features, but some Andes-specific CSRs are not yet implemented.

Andes-specific cache, local memory and ECC behavior is not yet implemented, except for CSR state. Andes Performance and Code Dense instructions and associated CSR state are implemented, but the EXEC.IT instruction supports in-memory table mode using the `uitb` CSR only (not hardwired mode).

1.10 Verification

All instructions have been extensively tested by Imperas, using tests generated specifically for this model and also reference tests from <https://github.com/riscv/riscv-tests>.

Also reference tests have been used from various sources including:

<https://github.com/riscv/riscv-tests>

<https://github.com/ucb-bar/riscv-torture>

The Imperas OVPsim RISC-V models are used in the RISC-V Foundations Compliance Framework as a functional Golden Reference:

<https://github.com/riscv/riscv-compliance>

where the simulated model is used to provide the reference signatures for compliance testing. The Imperas OVPsim RISC-V models are used as reference in both open source and commercial instruction stream test generators for hardware design verification, for example:

<http://valtrix.in/sting/> from Valtrix

<https://github.com/google/riscv-dv> from Google

The Imperas OVPsim RISC-V models are also used by commercial and open source RISC-V Core RTL developers as a reference to ensure correct functionality of their IP.

1.11 References

The Model details are based upon the following specifications:

RISC-V Instruction Set Manual, Volume I: User-Level ISA (User Architecture Version 2.2)

RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Privileged Architecture Version 1.10)

— AndesCore_AX25_DS142_V1.1 DS142-11

— AndeStar V5 Instruction Extension Specification (UMxxx-0.4, 2018-05-30)

— AndeStar V5 Architecture and CSR Definitions (UM164-12, 2018-06-14)

Chapter 2

Andes-Specific Extensions

Andes processors add various custom extensions to the basic RISC-V architecture. This model implements the following:

- 1: Hardware Stack Protection (if `mmisc_cfg.HSP=1`);
- 2: Performance Throttling (register interface only, if `mmisc_cfg.PFT=1`);
- 3: CSRs for CCTL Operations (register interface only, if `mmisc_cfg.CCTLCSR=1`);
- 4: Performance Extension instructions (if `mmisc_cfg.EV5MPE=1`);
- 5: CodeDense instructions (if `mmisc_cfg.ECD=1`);
- 6: Half-Precision Floating-Point instructions (if `mmisc_cfg.EFHW=1`).

Other Andes-specific extensions are not currently modeled. The exact set of supported extensions can be configured using parameter “`andesExtensions/mmisc_cfg`”, which overrides the default value of the `mmisc_cfg` register (see detailed description below).

2.1 Andes-Specific Parameters

In addition to the base model RISC-V parameters, this model implements parameters allowing Andes-specific model features to be controlled. These parameters are documented below.

2.1.1 Parameter `andesExtensions/mmisc_cfg`

This parameter allows the value of the read-only `mmisc_cfg` register to be specified. Bits that affect behavior of the model are:

bit 3 (ECD): enables CodeDense instructions and uitb CSR.

bit 4 (PFT): determines presence of `mpft_ctl` register and affects implemented fields in `mxstatus`.

bit 5 (HSP): enables HW Stack protection, relevant CSRs and affects implemented fields in `mxstatus`.

bit 13 (EV5PE): enables Performance Extension support.

bit 16 (CCTLCSR): enables CCTL CSRs.

Other bits can be set or cleared but do not affect model behavior.

Example: `-override iss/cpu0/andesExtensions/mmsc_cfg=0x2028`

2.1.2 Parameter `andesExtensions/micm_cfg`

This parameter allows the value of the read-only `micm_cfg` register to be specified. Bits that affect behavior of the model are:

bits 8:6 (ISZ): enables `mcache_ctl` CSR if non-zero.

bits 14:12 (ILMB): enables `milmb` CSR if non-zero.

Other bits can be set or cleared but do not affect model behavior, except that if any bit is non zero then IME/PIME bits in `mxstatus` are modeled.

Example: `-override iss/cpu0/andesExtensions/micm_cfg=0`

2.1.3 Parameter `andesExtensions/mdcm_cfg`

This parameter allows the value of the read-only `mdcm_cfg` register to be specified. Bits that affect behavior of the model are:

bits 8:6 (DSZ): enables `mcache_ctl` CSR if non-zero.

bits 14:12 (DLMB): enables `mdlmb` CSR if non-zero.

Other bits can be set or cleared but do not affect model behavior, except that if any bit is non zero then DME/DIME bits in `mxstatus` are modeled.

Example: `-override iss/cpu0/andesExtensions/mdcm_cfg=0`

2.1.4 Parameter `andesExtensions/uitb`

This parameter allows the value of the `uitb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/uitb=0`

2.1.5 Parameter `andesExtensions/milmb`

This parameter allows the value of the `milmb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/milmb=0`

2.1.6 Parameter `andesExtensions/milmbMask`

This parameter allows the mask of writable bits in the `milmb` register to be specified. The default value for this variant is `0xe` (`RWECC` and `ECCEN` are writable, all other bits are read-only).

Example: `-override iss/cpu0/andesExtensions/milmbMask=0xe`

2.1.7 Parameter `andesExtensions/mdlmb`

This parameter allows the value of the `mdlmb` register to be specified.

Example: `-override iss/cpu0/andesExtensions/mdlmb=0`

2.1.8 Parameter `andesExtensions/mdlmbMask`

This parameter allows the mask of writable bits in the `mdlmb` register to be specified. The default value for this variant is `0xe` (`RWECC` and `ECCEN` are writable, all other bits are read-only).

Example: `-override iss/cpu0/andesExtensions/mdlmbMask=0xe`

2.2 Hardware Stack Protection

Hardware Stack Protection is present on this variant (`mmisc_cfg.HSP=1`). Registers `mhsp_ctl`, `misp_bound` and `misp_base` are implemented.

2.3 Performance Throttling

Performance Throttling registers are present on this variant (`mmisc_cfg.PFT=1`). Register `mpft_ctl` is present but has no behavior except for the effects on `mxstatus`, which are modeled.

2.4 CSRs for CCTL Operations

CSRs for CCTL Operation are not present on this variant (`mmisc_cfg.CCTLCSR=0`).

2.5 Andes-Specific Instructions

This section describes Andes-specific instructions implemented by this variant. Refer to Andes reference documentation for more information.

2.5.1 Performance Extension Instructions

2.5.1.1 ADDIGP

31	30	21	20	19	17	16	15		
imm[17]		imm[10:1]		imm[11]		imm[14:12]		imm[16:15]	
14	13	12	11	7	6	0			
imm[0]		01		Rd		Custom0 0001011			

Add the content of the implied GP (x3) register with a signed constant.

2.5.1.2 BBC

31	30	29	25	24	20	19	15		
imm[10]		0		imm[9:5]		cimm[4:0]		Rs1	
14	12	11	8	7	6	0			
111		imm[4:1]		cimm[5]		Custom2 1011011			

Branch on bit is clear/zero.

2.5.1.3 BBS

31	30	29	25	24	20	19	15		
imm[10]		1		imm[9:5]		cimm[4:0]		Rs1	
14	12	11	8	7	6	0			
111		imm[4:1]		cimm[5]		Custom2 1011011			

Branch on bit is set/non-zero.

2.5.1.4 BEQC

31	30	29	25	24	20	19	15		
imm[10]		cimm[6]		imm[9:5]		cimm[4:0]		Rs1	
14	12	11	8	7	6	0			
101		imm[4:1]		cimm[5]		Custom2 1011011			

Branch on equal to a constant.

2.5.1.5 BNEC

31	30	29	25	24	20	19	15		
imm[10]		cimm[6]		imm[9:5]		cimm[4:0]		Rs1	
14	12	11	8	7	6	0			
110		imm[4:1]		cimm[5]		Custom2 1011011			

Branch on not-equal to a constant.

2.5.1.6 BFOS

31	26	25	20	19	15	14	12	11	7	6	0
msb[5:0]		lsb[5:0]		Rs1		011		Rd		Custom2 1011011	

Sign-extended bit-field extract or insert operation.

2.5.1.7 BFOZ

31	26	25	20	19	15	14	12	11	7	6	0
msb[5:0]		lsb[5:0]		Rs1		010		Rd		Custom2 1011011	

Zero-extended bit-field extract or insert operation.

2.5.1.8 LEA.h

31	25	24	20	19	15	14	12	11	7	6	0
0000101		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with a half-word-aligned offset from an offset register.

2.5.1.9 LEA.w

31	25	24	20	19	15	14	12	11	7	6	0
0000110		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with a word-aligned offset from an offset register.

2.5.1.10 LEA.d

31	25	24	20	19	15	14	12	11	7	6	0
0000111		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with a double-word-aligned offset from an offset register.

2.5.1.11 LEA.b.ze

31	25	24	20	19	15	14	12	11	7	6	0
0001000		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with an unsigned 32-bit byte offset from an offset register.

2.5.1.12 LEA.h.ze

31	25	24	20	19	15	14	12	11	7	6	0
0001001		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with an unsigned 32-bit half-word offset from an offset register.

2.5.1.13 LEA.w.ze

31	25	24	20	19	15	14	12	11	7	6	0
0001010		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with an unsigned 32-bit word offset from an offset register.

2.5.1.14 LEA.d.ze

31	25	24	20	19	15	14	12	11	7	6	0
0001011		Rs2		Rs1		000		Rd		Custom2 1011011	

Add a base register with an unsigned 32-bit double-word offset from an offset register.

2.5.1.15 LBGP

31	30	21	20	19	17	16	15		
imm[17]		imm[10:1]		imm[11]		imm[14:12]		imm[16:15]	
14	13	12	11	7	6	0			
imm[0]		00		Rd		Custom0 0001011			

Load a sign-extended 8-bit byte from memory into a general register.

2.5.1.16 LBUGP

31	30	21	20	19	17	16	15		
imm[17]		imm[10:1]		imm[11]		imm[14:12]		imm[16:15]	
14	13	12	11	7	6	0			
imm[0]		10		Rd		Custom0 0001011			

Load a zero-extended 8-bit byte from memory into a general register.

2.5.1.17 LHGP

31	30	21	20	19	17		
imm[17]		imm[10:1]		imm[11]		imm[14:12]	
16	15	14	12	11	7	6	0
imm[16:15]		001		Rd		Custom1 0101011	

Load a sign-extended 16-bit half-word from memory into a general register.

2.5.1.18 LHUGP

31	30	21	20	19	17
imm[17]	imm[10:1]		imm[11]	imm[14:12]	
16	15	14	12	11	7
imm[16:15]	101		Rd	Custom1 0101011	

Load a zero-extended 16-bit half-word from memory into a general register.

2.5.1.19 LWGP

31	30	22	21	20	19	17
imm[18]	imm[10:2]		imm[17]	imm[11]	imm[14:12]	
16	15	14	12	11	7	6
imm[16:15]	010		Rd	Custom1 0101011		

Load a sign-extended 32-bit word from memory into a general register.

2.5.1.20 LWUGP

31	30	22	21	20	19	17
imm[18]	imm[10:2]		imm[17]	imm[11]	imm[14:12]	
16	15	14	12	11	7	6
imm[16:15]	110		Rd	Custom1 0101011		

Load a zero-extended 32-bit word from memory into a general register.

2.5.1.21 LDGP

31	30	23	22	21	20	19	17
imm[19]	imm[10:3]		imm[18:17]		imm[11]	imm[14:12]	
16	15	14	12	11	7	6	0
imm[16:15]	011		Rd	Custom1 0101011			

Load a 64-bit double-word from memory into a general register.

2.5.1.22 SBGP

31	30	25	24	20	19	17	16	15
imm[17]	imm[10:5]			Rs2	imm[14:12]		imm[16:15]	
14	13	12	11	8	7	6	0	0
imm[0]	11		imm[4:1]		imm[11]	Custom0 0001011		

Store an 8-bit byte from a general register into a memory location.

2.5.1.23 SHGP

31	30	25	24	20	19	17	16	15	
imm[17]		imm[10:5]		Rs2		imm[14:12]		imm[16:15]	
14	12	11	8	7	6	0			
000		imm[4:1]		imm[11]		Custom1 0101011			

Store a 16-bit half-word from a general register into a memory location.

2.5.1.24 SWGP

31	30	25	24	20	19	17	16	15	
imm[18]		imm[10:5]		Rs2		imm[14:12]		imm[16:15]	
14	12	11	9	8	7	6	0		
100		imm[4:2]		imm[17]		imm[11]		Custom1 0101011	

Store a 32-bit word from a general register into a memory location.

2.5.1.25 SDGP

31	30	25	24	20	19	17	16	15	
imm[19]		imm[10:5]		Rs2		imm[14:12]		imm[16:15]	
14	12	11	10	9	8	7	6	0	
111		imm[4:3]		imm[18:17]		imm[11]		Custom1 0101011	

Store a 64-bit double-word from a general register into a memory location.

2.5.1.26 FFB

31	25	24	20	19	15	14	12	11	7	6	0
0010000		Rs2		Rs1		000		Rd		Custom2 1011011	

Find the first byte in a first register that matches a value in a second register.

2.5.1.27 FFZMISM

31	25	24	20	19	15	14	12	11	7	6	0
0010001		Rs2		Rs1		000		Rd		Custom2 1011011	

Find the first byte in a register that is zero or fails a corresponding byte comparison.

2.5.1.28 FFMISM

31	25	24	20	19	15	14	12	11	7	6	0
0010010		Rs2		Rs1		000		Rd		Custom2 1011011	

Find the first byte in a register that fails a corresponding byte comparison.

2.5.1.29 FLMISM

31	25	24	20	19	15	14	12	11	7	6	0
0010011		Rs2		Rs1		000		Rd		Custom2 1011011	

Find the last byte in a register that fails a corresponding byte comparison.

2.5.2 CodeDense Instructions

2.5.2.1 EXEC.IT

15	13	12	9	8	7	6	2	1	0		
100		imm[10 4:3 8]		imm[11]		0		imm[7:6 2 9 5]		00	

Execute an instruction fetched from the instruction table.

2.5.2.2 EX9.IT

15	13	12	9	8	7	6	2	1	0
100		imm[10 4:3 8]		00		imm[7:6 2 9 5]		00	

Execute an instruction fetched from the instruction table.

2.5.3 Half-Precision Floating-Point Instructions

2.5.3.1 FLHW

31	20	19	15	14	12	11	7	6	0
imm[11:0]		Rs1		000		FRd		LOAD-FP 0000111	

Load half-precision floating-point value from memory and convert it to a single-precision value.

2.5.3.2 FSHW

31	25	24	20	19	15	14	12	11	7	6	0
imm[11:5]		FRs2		Rs1		000		imm[4:0]		STORE-FP 0100111	

Store half-precision floating-point data to memory. The half-precision floating-point value is converted from a single-precision value source.

Chapter 3

Configuration

3.1 Location

This model's VLVN is `andes.ovpworld.org/processor/riscv/1.0`.

The model source is usually at:

`$IMPERAS_HOME/ImperasLib/source/andes.ovpworld.org/processor/riscv/1.0`

The model binary is usually at:

`$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib/andes.ovpworld.org/processor/riscv/1.0`

3.2 GDB Path

The default GDB for this model is: `$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/riscv-none-embed-gdb`.

3.3 Semi-Host Library

The default semi-host library file is `riscv.ovpworld.org/semihosting/pk/1.0`

3.4 Processor Endian-ness

This is a LITTLE endian model.

3.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

3.6 Processor ELF code

The ELF code supported by this model is: `0xf3`.

Chapter 4

All Variants in this model

This model has these variants

Variant	Description
N25	
NX25	
N25F	
NX25F	
A25	
AX25	
A25F	
AX25F	(described in this document)

Table 4.1: All Variants in this model

Chapter 5

Bus Master Ports

This model has these bus master ports.

Name	min	max	Connect?	Description
INSTRUCTION	32	56	mandatory	Instruction bus
DATA	32	56	optional	Data bus

Table 5.1: Bus Master Ports

Chapter 6

Bus Slave Ports

This model has no bus slave ports.

Chapter 7

Net Ports

This model has these net ports.

Name	Type	Connect?	Description
reset	input	optional	Reset
nmi	input	optional	NMI
SSWInterrupt	input	optional	Supervisor software interrupt
MSWInterrupt	input	optional	Machine software interrupt
STimerInterrupt	input	optional	Supervisor timer interrupt
MTimerInterrupt	input	optional	Machine timer interrupt
SExternalInterrupt	input	optional	Supervisor external interrupt
MExternalInterrupt	input	optional	Machine external interrupt

Table 7.1: Net Ports

Chapter 8

FIFO Ports

This model has no FIFO ports.

Chapter 9

Formal Parameters

Name	Type	Description
variant	Enumeration	Selects variant (either a generic UISA or a specific model)
user_version	Enumeration	Specify required User Architecture version (2.2, 2.3 or 20190305)
priv_version	Enumeration	Specify required Privileged Architecture version (1.10, 1.11 or 20190405)
mstatus_fs_mode	Enumeration	Specify conditions causing update of mstatus.FS to dirty (write_1, write_any or always_dirty)
verbose	Boolean	Specify verbose output messages
updatePTEA	Boolean	Specify whether hardware update of PTE A bit is supported
updatePTED	Boolean	Specify whether hardware update of PTE D bit is supported
unaligned	Boolean	Specify whether the processor supports unaligned memory accesses
unalignedAMO	Boolean	Specify whether the processor supports unaligned memory accesses for AMO instructions
wfi_is_nop	Boolean	Specify whether WFI should be treated as a NOP (if not, halt while waiting for interrupts)
mtvec_is_ro	Boolean	Specify whether mtvec CSR is read-only
tvec_align	Uns32	Specify hardware-enforced alignment of mtvec/stvec/utvec when Vectored interrupt mode enabled
mtvec_mask	Uns64	Specify hardware-enforced mask of writable bits in mtvec register
stvec_mask	Uns64	Specify hardware-enforced mask of writable bits in stvec register
tval_ii_code	Boolean	Specify whether mtval/stval contain faulting instruction bits on illegal instruction exception
cycle_undefined	Boolean	Specify that the cycle CSR is undefined (reads to it are emulated by a Machine mode trap)
time_undefined	Boolean	Specify that the time CSR is undefined (reads to it are emulated by a Machine mode trap)
instret_undefined	Boolean	Specify that the instret CSR is undefined (reads to it are emulated by a Machine mode trap)
enable_CSR_bus	Boolean	Add artifact CSR bus port, allowing CSR registers to be externally implemented
d_requires_f	Boolean	If D and F extensions are separately enabled in the misa CSR, whether D is enabled only if F is enabled
xret_preserves_lr	Boolean	Whether an xRET instruction preserves the value of LR
ASID_bits	Uns32	Specify the number of implemented ASID bits
lr_sc_grain	Uns32	Specify byte granularity of ll/sc lock region (constrained to a power of two)
reset_address	Uns64	Override reset vector address
nmi_address	Uns64	Override NMI vector address
PMP_grain	Uns32	Specify PMP region granularity, G (0 =>4 bytes, 1 =>8 bytes, etc)
PMP_registers	Uns32	Specify the number of implemented PMP address registers
Sv_modes	Uns32	Specify bit mask of implemented Sv modes (e.g. 1<<8 is Sv39)
local_int_num	Uns32	Specify number of supplemental local interrupts
endian	Endian	Model endian

misa_MXL	Uns32	Override default value of misa.MXL
misa_MXL_mask	Uns32	Override mask of writable bits in misa.MXL
misa_Extensions	Uns32	Override default value of misa.Extensions
add_Extensions	String	Add extensions specified by letters to misa.Extensions (for example, specify “VD” to add V and D features)
misa_Extensions_mask	Uns32	Override mask of writable bits in misa.Extensions
add_Extensions_mask	String	Add extensions specified by letters to mask of writable bits in misa.Extensions (for example, specify “VD” to add V and D features)
mvendorid	Uns64	Override mvendorid register
marchid	Uns64	Override marchid register
mimpid	Uns64	Override mimpid register
mhartid	Uns64	Override mhartid register
mtvec	Uns64	Override mtvec register
mstatus_FS	Uns32	Override default value of mstatus.FS (initial state of floating point unit)

Table 9.1: Parameters that can be set in: Hart

9.1 Extension Parameters

Name	Type	Description
milmb	Uns64	Override milmb register
mdlmb	Uns64	Override mdlmb register
mmsc_cfg	Uns64	Override mmsc_cfg register
micm_cfg	Uns64	Override micm_cfg register
mdcm_cfg	Uns64	Override mdcm_cfg register
uitb	Uns64	Override uitb register
milmbMask	Uns64	Override milmb register writable bit mask
mdlmbMask	Uns64	Override mdlmb register writable bit mask

Table 9.2: Extension Parameters

9.2 Parameters with enumerated types

9.2.1 Parameter user_version

Set to this value	Description
2.2	User Architecture Version 2.2
2.3	Deprecated and equivalent to 20190305
20190305	User Architecture Version 20190305-Base-Ratification

Table 9.3: Values for Parameter user_version

9.2.2 Parameter priv_version

Set to this value	Description
1.10	Privileged Architecture Version 1.10
1.11	Deprecated and equivalent to 20190405
20190405	Privileged Architecture Version 20190405-Priv-MSU-Ratification

Table 9.4: Values for Parameter priv_version

9.2.3 Parameter `mstatus_fs_mode`

Set to this value	Description
<code>write_1</code>	Any non-zero flag result sets <code>mstatus.fs</code> dirty
<code>write_any</code>	Any write of flags sets <code>mstatus.fs</code> dirty
<code>always_dirty</code>	<code>mstatus.fs</code> is either off or dirty

Table 9.5: Values for Parameter `mstatus_fs_mode`

Chapter 10

Execution Modes

Mode	Code	Description
User	0	User mode
Supervisor	1	Supervisor mode
Machine	3	Machine mode

Table 10.1: Modes implemented in: Hart

Chapter 11

Exceptions

Exception	Code	Description
InstructionAddressMisaligned	0	Fetch from unaligned address
InstructionAccessFault	1	No access permission for fetch
IllegalInstruction	2	Undecoded, unimplemented or disabled instruction
Breakpoint	3	EBREAK instruction executed
LoadAddressMisaligned	4	Load from unaligned address
LoadAccessFault	5	No access permission for load
StoreAMOAddressMisaligned	6	Store/atomic memory operation at unaligned address
StoreAMOAccessFault	7	No access permission for store/atomic memory operation
EnvironmentCallFromUMode	8	ECALL instruction executed in User mode
EnvironmentCallFromSMode	9	ECALL instruction executed in Supervisor mode
EnvironmentCallFromMMode	11	ECALL instruction executed in Machine mode
InstructionPageFault	12	Page fault at fetch address
LoadPageFault	13	Page fault at load address
StoreAMOPageFault	15	Page fault at store/atomic memory operation address
SSWInterrupt	65	Supervisor software interrupt
MSWInterrupt	67	Machine software interrupt
STimerInterrupt	69	Supervisor timer interrupt
MTimerInterrupt	71	Machine timer interrupt
SExternalInterrupt	73	Supervisor external interrupt
MExternalInterrupt	75	Machine external interrupt
HSP_OVF	32	Stack overflow
HSP_UDF	33	Stack underflow

Table 11.1: Exceptions implemented in: Hart

Chapter 12

Hierarchy of the model

A CPU core may be configured to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.

OVP processor models can be written to include SMP blocks and to have many levels of hierarchy. Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

12.1 Level 1: Hart

This level in the model hierarchy has 3 commands.

This level in the model hierarchy has 6 register groups:

Group name	Registers
Core	33
Floating_point	32
User_Control_and_Status	36
Supervisor_Control_and_Status	10
Machine_Control_and_Status	112
Integration_support	1

Table 12.1: Register groups

This level in the model hierarchy has no children.

Chapter 13

Model Commands

A Processor model can implement one or more **Model Commands** available to be invoked from the simulator command line, from the OP API or from the Imperas Multiprocessor Debugger.

13.1 Level 1: Hart

13.1.1 dumpTLB

13.1.1.1 Argument description

show TLB contents

13.1.2 isync

specify instruction address range for synchronous execution

Argument	Type	Description
-addresshi	Uns64	end address of synchronous execution range
-addresslo	Uns64	start address of synchronous execution range

Table 13.1: isync command arguments

13.1.3 itrace

enable or disable instruction tracing

Argument	Type	Description
-after	Uns64	apply after this many instructions
-enable	Boolean	enable instruction tracing
-instructioncount	Boolean	include the instruction number in each trace
-off	Boolean	disable instruction tracing
-on	Boolean	enable instruction tracing
-registerchange	Boolean	show registers changed by this instruction
-registers	Boolean	show registers after each trace

Table 13.2: itrace command arguments

Chapter 14

Registers

14.1 Level 1: Hart

14.1.1 Core

Registers at level:1, type:Hart group:Core

Name	Bits	Initial-Hex	RW	Description
zero	64	0	r-	
ra	64	0	rw	
sp	64	0	rw	stack pointer
gp	64	0	rw	
tp	64	0	rw	
t0	64	0	rw	
t1	64	0	rw	
t2	64	0	rw	
s0	64	0	rw	
s1	64	0	rw	
a0	64	0	rw	
a1	64	0	rw	
a2	64	0	rw	
a3	64	0	rw	
a4	64	0	rw	
a5	64	0	rw	
a6	64	0	rw	
a7	64	0	rw	
s2	64	0	rw	
s3	64	0	rw	
s4	64	0	rw	
s5	64	0	rw	
s6	64	0	rw	
s7	64	0	rw	
s8	64	0	rw	
s9	64	0	rw	
s10	64	0	rw	
s11	64	0	rw	
t3	64	0	rw	
t4	64	0	rw	
t5	64	0	rw	
t6	64	0	rw	
pc	64	0	rw	program counter

Table 14.1: Registers at level 1, type:Hart group:Core

14.1.2 Floating_point

Registers at level:1, type:Hart group:Floating_point

Name	Bits	Initial-Hex	RW	Description
ft0	64	0	rw	
ft1	64	0	rw	
ft2	64	0	rw	
ft3	64	0	rw	
ft4	64	0	rw	
ft5	64	0	rw	
ft6	64	0	rw	
ft7	64	0	rw	
fs0	64	0	rw	
fs1	64	0	rw	
fa0	64	0	rw	
fa1	64	0	rw	
fa2	64	0	rw	
fa3	64	0	rw	
fa4	64	0	rw	
fa5	64	0	rw	
fa6	64	0	rw	
fa7	64	0	rw	
fs2	64	0	rw	
fs3	64	0	rw	
fs4	64	0	rw	
fs5	64	0	rw	
fs6	64	0	rw	
fs7	64	0	rw	
fs8	64	0	rw	
fs9	64	0	rw	
fs10	64	0	rw	
fs11	64	0	rw	
ft8	64	0	rw	
ft9	64	0	rw	
ft10	64	0	rw	
ft11	64	0	rw	

Table 14.2: Registers at level 1, type:Hart group:Floating_point

14.1.3 User_Control_and_Status

Registers at level:1, type:Hart group:User_Control_and_Status

Name	Bits	Initial-Hex	RW	Description
fflags	64	0	rw	Floating-Point Flags
frm	64	0	rw	Floating-Point Rounding Mode
fcsr	64	0	rw	Floating-Point Control and Status
uitb*	64	0	rw	Instruction Table Base Address
cycle	64	0	r-	Cycle Counter
time	64	0	r-	Timer
instret	64	0	r-	Instructions Retired

hpmcounter3	64	0	r-	Performance Monitor Counter 3
hpmcounter4	64	0	r-	Performance Monitor Counter 4
hpmcounter5	64	0	r-	Performance Monitor Counter 5
hpmcounter6	64	0	r-	Performance Monitor Counter 6
hpmcounter7	64	0	r-	Performance Monitor Counter 7
hpmcounter8	64	0	r-	Performance Monitor Counter 8
hpmcounter9	64	0	r-	Performance Monitor Counter 9
hpmcounter10	64	0	r-	Performance Monitor Counter 10
hpmcounter11	64	0	r-	Performance Monitor Counter 11
hpmcounter12	64	0	r-	Performance Monitor Counter 12
hpmcounter13	64	0	r-	Performance Monitor Counter 13
hpmcounter14	64	0	r-	Performance Monitor Counter 14
hpmcounter15	64	0	r-	Performance Monitor Counter 15
hpmcounter16	64	0	r-	Performance Monitor Counter 16
hpmcounter17	64	0	r-	Performance Monitor Counter 17
hpmcounter18	64	0	r-	Performance Monitor Counter 18
hpmcounter19	64	0	r-	Performance Monitor Counter 19
hpmcounter20	64	0	r-	Performance Monitor Counter 20
hpmcounter21	64	0	r-	Performance Monitor Counter 21
hpmcounter22	64	0	r-	Performance Monitor Counter 22
hpmcounter23	64	0	r-	Performance Monitor Counter 23
hpmcounter24	64	0	r-	Performance Monitor Counter 24
hpmcounter25	64	0	r-	Performance Monitor Counter 25
hpmcounter26	64	0	r-	Performance Monitor Counter 26
hpmcounter27	64	0	r-	Performance Monitor Counter 27
hpmcounter28	64	0	r-	Performance Monitor Counter 28
hpmcounter29	64	0	r-	Performance Monitor Counter 29
hpmcounter30	64	0	r-	Performance Monitor Counter 30
hpmcounter31	64	0	r-	Performance Monitor Counter 31

Table 14.3: Registers at level 1, type:Hart group:User_Control_and_Status

* Registers marked with an asterisk are part of the processor extension library.

14.1.4 Supervisor_Control_and_Status

Registers at level:1, type:Hart group:Supervisor_Control_and_Status

Name	Bits	Initial-Hex	RW	Description
sstatus	64	2 00000000	rw	Supervisor Status
sie	64	0	rw	Supervisor Interrupt Enable
stvec	64	0	rw	Supervisor Trap-Vector Base-Address
scounteren	64	0	rw	Supervisor Counter Enable
sscratch	64	0	rw	Supervisor Scratch
sepc	64	0	rw	Supervisor Exception Program Counter
scause	64	0	rw	Supervisor Cause
stval	64	0	rw	Supervisor Trap Value
sip	64	0	rw	Supervisor Interrupt Pending
satp	64	0	rw	Supervisor Address Translation and Protection

Table 14.4: Registers at level 1, type:Hart group:Supervisor_Control_and_Status

14.1.5 Machine_Control_and_Status

Registers at level:1, type:Hart group:Machine_Control_and_Status

Name	Bits	Initial-Hex	RW	Description
mstatus	64	a 00001800	rw	Machine Status
misa	64	80000000 0094112d	rw	ISA and Extensions
medeleg	64	0	rw	Machine Exception Delegation
mideleg	64	0	rw	Machine Interrupt Delegation
mie	64	0	rw	Machine Interrupt Enable
mtvec	64	0	rw	Machine Trap-Vector Base-Address
mcounteren	64	0	rw	Machine Counter Enable
mhpmevent3	64	0	rw	Machine Performance Monitor Event Select 3
mhpmevent4	64	0	rw	Machine Performance Monitor Event Select 4
mhpmevent5	64	0	rw	Machine Performance Monitor Event Select 5
mhpmevent6	64	0	rw	Machine Performance Monitor Event Select 6
mhpmevent7	64	0	rw	Machine Performance Monitor Event Select 7
mhpmevent8	64	0	rw	Machine Performance Monitor Event Select 8
mhpmevent9	64	0	rw	Machine Performance Monitor Event Select 9
mhpmevent10	64	0	rw	Machine Performance Monitor Event Select 10
mhpmevent11	64	0	rw	Machine Performance Monitor Event Select 11
mhpmevent12	64	0	rw	Machine Performance Monitor Event Select 12
mhpmevent13	64	0	rw	Machine Performance Monitor Event Select 13
mhpmevent14	64	0	rw	Machine Performance Monitor Event Select 14
mhpmevent15	64	0	rw	Machine Performance Monitor Event Select 15
mhpmevent16	64	0	rw	Machine Performance Monitor Event Select 16
mhpmevent17	64	0	rw	Machine Performance Monitor Event Select 17
mhpmevent18	64	0	rw	Machine Performance Monitor Event Select 18
mhpmevent19	64	0	rw	Machine Performance Monitor Event Select 19
mhpmevent20	64	0	rw	Machine Performance Monitor Event Select 20
mhpmevent21	64	0	rw	Machine Performance Monitor Event Select 21
mhpmevent22	64	0	rw	Machine Performance Monitor Event Select 22
mhpmevent23	64	0	rw	Machine Performance Monitor Event Select 23
mhpmevent24	64	0	rw	Machine Performance Monitor Event Select 24
mhpmevent25	64	0	rw	Machine Performance Monitor Event Select 25
mhpmevent26	64	0	rw	Machine Performance Monitor Event Select 26
mhpmevent27	64	0	rw	Machine Performance Monitor Event Select 27
mhpmevent28	64	0	rw	Machine Performance Monitor Event Select 28
mhpmevent29	64	0	rw	Machine Performance Monitor Event Select 29
mhpmevent30	64	0	rw	Machine Performance Monitor Event Select 30
mhpmevent31	64	0	rw	Machine Performance Monitor Event Select 31
mscratch	64	0	rw	Machine Scratch
mepc	64	0	rw	Machine Exception Program Counter
mcause	64	0	rw	Machine Cause
mtval	64	0	rw	Machine Trap Value
mip	64	0	rw	Machine Interrupt Pending
pmpcfg0	64	0	rw	Physical Memory Protection Configuration 0
pmpcfg2	64	0	rw	Physical Memory Protection Configuration 2
pmpaddr0	64	0	rw	Physical Memory Protection Address 0
pmpaddr1	64	0	rw	Physical Memory Protection Address 1
pmpaddr2	64	0	rw	Physical Memory Protection Address 2
pmpaddr3	64	0	rw	Physical Memory Protection Address 3
pmpaddr4	64	0	rw	Physical Memory Protection Address 4
pmpaddr5	64	0	rw	Physical Memory Protection Address 5
pmpaddr6	64	0	rw	Physical Memory Protection Address 6
pmpaddr7	64	0	rw	Physical Memory Protection Address 7
pmpaddr8	64	0	rw	Physical Memory Protection Address 8
pmpaddr9	64	0	rw	Physical Memory Protection Address 9
pmpaddr10	64	0	rw	Physical Memory Protection Address 10

pmpaddr11	64	0	rw	Physical Memory Protection Address 11
pmpaddr12	64	0	rw	Physical Memory Protection Address 12
pmpaddr13	64	0	rw	Physical Memory Protection Address 13
pmpaddr14	64	0	rw	Physical Memory Protection Address 14
pmpaddr15	64	0	rw	Physical Memory Protection Address 15
tselect	64	-	rw	Debug/Trace Trigger Register Select (not implemented)
tdata1	64	-	rw	Debug/Trace Trigger Data 1 (not implemented)
tdata2	64	-	rw	Debug/Trace Trigger Data 2 (not implemented)
tdata3	64	-	rw	Debug/Trace Trigger Data 3 (not implemented)
dcsr	64	-	rw	Debug Control and Status (not implemented)
dpc	64	-	rw	Debug PC (not implemented)
dscratch	64	-	rw	Debug Scratch (not implemented)
mnvec*	64	0	rw	NMI Vector Base Address
mxstatus*	64	0	rw	Machine Extended Status
mpft_ctl*	64	0	rw	Performance Throttling Control
mhsp_ctl*	64	0	rw	Machine Hardware Stack Protection Control
mssp_bound*	64	ffffff fffffff	rw	Machine SP Bound
mssp_base*	64	ffffff fffffff	rw	Machine SP Base
mdcause*	64	0	rw	Machine Detailed Trap Cause
mmisc_ctl*	64	0	rw	Machine Miscellaneous Control
mcycle	64	0	rw	Machine Cycle Counter
minstret	64	0	rw	Machine Instructions Retired
mhpmcounter3	64	0	rw	Machine Performance Monitor Counter 3
mhpmcounter4	64	0	rw	Machine Performance Monitor Counter 4
mhpmcounter5	64	0	rw	Machine Performance Monitor Counter 5
mhpmcounter6	64	0	rw	Machine Performance Monitor Counter 6
mhpmcounter7	64	0	rw	Machine Performance Monitor Counter 7
mhpmcounter8	64	0	rw	Machine Performance Monitor Counter 8
mhpmcounter9	64	0	rw	Machine Performance Monitor Counter 9
mhpmcounter10	64	0	rw	Machine Performance Monitor Counter 10
mhpmcounter11	64	0	rw	Machine Performance Monitor Counter 11
mhpmcounter12	64	0	rw	Machine Performance Monitor Counter 12
mhpmcounter13	64	0	rw	Machine Performance Monitor Counter 13
mhpmcounter14	64	0	rw	Machine Performance Monitor Counter 14
mhpmcounter15	64	0	rw	Machine Performance Monitor Counter 15
mhpmcounter16	64	0	rw	Machine Performance Monitor Counter 16
mhpmcounter17	64	0	rw	Machine Performance Monitor Counter 17
mhpmcounter18	64	0	rw	Machine Performance Monitor Counter 18
mhpmcounter19	64	0	rw	Machine Performance Monitor Counter 19
mhpmcounter20	64	0	rw	Machine Performance Monitor Counter 20
mhpmcounter21	64	0	rw	Machine Performance Monitor Counter 21
mhpmcounter22	64	0	rw	Machine Performance Monitor Counter 22
mhpmcounter23	64	0	rw	Machine Performance Monitor Counter 23
mhpmcounter24	64	0	rw	Machine Performance Monitor Counter 24
mhpmcounter25	64	0	rw	Machine Performance Monitor Counter 25
mhpmcounter26	64	0	rw	Machine Performance Monitor Counter 26
mhpmcounter27	64	0	rw	Machine Performance Monitor Counter 27
mhpmcounter28	64	0	rw	Machine Performance Monitor Counter 28
mhpmcounter29	64	0	rw	Machine Performance Monitor Counter 29
mhpmcounter30	64	0	rw	Machine Performance Monitor Counter 30
mhpmcounter31	64	0	rw	Machine Performance Monitor Counter 31
mvendorid	64	31e	r-	Vendor ID
marchid	64	10000000 00008a25	r-	Architecture ID
mimpid	64	10	r-	Implementation ID
mhartid	64	0	r-	Hardware Thread ID

micm_cfg*	64	0	r-	Instruction Cache/Memory Configuration
mdcm_cfg*	64	0	r-	Data Cache/Memory Configuration
mmsc_cfg*	64	22038	r-	Miscellaneous Configuration

Table 14.5: Registers at level 1, type:Hart group:Machine_Control_and_Status

* Registers marked with an asterisk are part of the processor extension library.

14.1.6 Integration_support

Registers at level:1, type:Hart group:Integration_support

Name	Bits	Initial-Hex	RW	Description
LRSCAddress	64	ffffff fffffff	rw	LR/SC active lock address

Table 14.6: Registers at level 1, type:Hart group:Integration_support