Godson 3A1000 Processor User Manual

Volume II

GS464 processor core V1.4

2015 Nian ***10*** Yue

Loongson Zhongke Technology Co., Ltd.

Copyright Notice

Disclaimer

This document only provides periodic information, and the content can be updated at any time according to the actual situation of the product without notice. Ruin

The company does not assume any responsibility for direct or indirect losses caused by improper use of documents.

Loongson Zhongke Technology Co., Ltd.

Loongson Technology Corporation Limited

Address: Building 2, Longxin Industrial Park, Zhongguancun Environmental Protection Technology Demonstration Park, Haidian District, Beijing

Building No. 2, Loongson Industrial Park,

Zhongguancun Environmental Protection Park, Haidian District, Beijing

Telephone (Tel): 010-62546668

Fax: 010-62600826

Godson 3A1000 Processor User Manual • Next

Reading guide

"Godson 3A1000 Processor User Manual" is divided into the first and second volumes.

"Loongson 3A1000 Processor User Manual" is divided into two parts, the first part (Chapter 1 ~ Chapter 10) introduces Loongson

3A1000 multi-core processor architecture and register description, on chip system architecture, function and configuration of main modules, registers

Lists and bit fields are explained in detail; the second part (Chapter 11 ~ Chapter 16) is the system software programming guide

Special presentations on common problems in the operating system development process.

The second volume of the "Loongson 3A1000 Processor User Manual" introduces in detail the adoption of Loongson 3A1000 from the perspective of system software developers

GS464 high-performance processor core.

Page 4

Godson 3A1000 Processor User Manual • Next

revise history

Document update record

Document name:	Loongson 3A1000 processor user hand book --Volume II
version number	V1.4
founder:	R & D Center
Creation Date:	2015-10-08

Update history

Serial number	Updated	updater	version number	update content
1	2011-06-24	R & D Center	V1.0	The first draft is completed
2	2011-10-18	R & D Center	V1.1	Review and proofreading
3	2011-11-24	R & D Center	V1.2	Edit cover
4	2012-04-28	R & D Center	V1.3	Fix the 48-bit address space related error in Chapter 5 "Memory Management" error
5	2015-10-08	R & D Center	V1.4	Modify the instruction description in Chapter 2 and Chapter 7

Manual feedback: service@loongson.cn

Page 5

table of Contents

Figure catalog

Table directory

Godson 3A1000 Processor User Manual • Next

Godson 3A1000 Processor User Manual • Next

XI

1 Structure Overview

GS464 is a general-purpose RISC processor IP that implements the 64-bit MIPS64 instruction set. GS464's instruction pipeline

It takes four instructions to decode in one clock cycle, and is dynamically transmitted to five fully-pipelined functional components. Although the instructions are in

Under the premise of guaranteeing the dependency relationship, out-of-order execution is performed, but the submission of instructions is still in accordance with the original order c

The order of external and memory access is executed sequentially.

The super-scalar structure of the four launches makes the instruction and data related problems in the instruction pipeline very prominent. GS464 uses out of order

Implement technology and aggressive storage system design to improve the efficiency of the pipeline.

Out-of-order execution technologies include register renaming technology, dynamic scheduling technology, and branch prediction technology. Register rename solution

WAR (write after read) and WAW (write after write) are related and used for accurate on-site recovery caused by exception and error transfer prediction,

GS464 renames fixed-point and floating-point registers through 64-item physical register file respectively. Dynamic scheduling according to instructions

The order in which the operands are prepared rather than the order in which the instructions appear in the program to execute the instructions, reducing the RAW (read after write) pl

GS464 has a 16-item fixed-point reservation station and a 16-item floating-point reservation station for out of order transmission

And through a 64-item Reorder queue (referred to as ROQ) to achieve out-of-order execution instructions submitted in the order of the program.

Branch prediction reduces the blocking caused by control related by predicting whether the branch instruction jumps successfully, GS464 uses 16

Branch Target Buffer (BTB) for items, branch history table for 2K items (Branch

History Table (BHT), 9-bit Global Histiry Registor (GHR),

And 4 items of Return Address Stack (Return Address Stack, RAS for short) for branch prediction.

GS464 advanced storage system design can effectively improve the efficiency of the pipeline. The first-level cache of GS464 consists of 64KB

The instruction cache and the 64KB data cache are composed of four-way group structure. There are 64 items of GS464 TLB,

With a fully associative structure, each item can map an odd page and an even page, and the page size is variable from 4KB to 16MB. GS464

Address dependency is dynamically resolved through the 24-item memory access queue and the 8-item memory access queue (Miss Queue).

Out-of-order execution of current memory access operations, non-blocking cache, load speculation execution (Load Speculation)

化 技术。 Technology. GS464 supports 128-bit memory access operation, and its virtual address and physical address are both 48 bits.

GS464 has two fixed-point features and two floating-point features. Each floating-point component can execute 64

Double-precision floating-point multiply-add operation, and perform 32-bit and 64-bit fixed-point instructions through the expansion of the fmt field of floating-point instructions.

GS464 supports the EJTAG debugging specification of MIPS, adopts the standard AXI interface, and its instruction cache realizes

Parity check, data Cache implements ECC check. The above features can increase the scope of application of GS464.

The basic pipeline of GS464 includes fetching, pre-decoding, decoding, register renaming, scheduling, transmitting, reading registers,

12

Page 15

Godson 3A1000 Processor User Manual • Next

9 levels of execution and submission, each level of pipeline includes the following operations.

- ◆ Use the value of the program counter PC to access the instruction cache and instruction TLB, if the instruction cache
  If both the instruction and the TLB are hit, then four new instructions are fetched into the instruction register IR.
- ◆ The pre-decoding pipeline stage mainly decodes the branch instruction and predicts the direction of the jump.
- ◆ The decoding pipeline stage converts the four instructions in IR into the internal instruction format of GS464 and sends them to the register rename mode
  Piece.
- ◆ Register renaming pipeline level allocates a new physical register for the logical target register and registers the logical source
  The device is mapped to the physical register most recently allocated to the logical register.
- ◆ The dispatch pipeline assigns the renamed instructions to fixed-point or floating-point reserved stations for execution and sends them to ROQ
  In order to submit after execution; in addition, transfer instructions and memory access instructions are also sent to the transfer queue and
  Save the queue.
- ◆ The launch pipeline stage selects one operand from the fixed-point or floating-point reservation station
  Instructions; the instructions whose operands are not prepared when renaming, by listening to the result bus and forward bus, etc.
  Wait until its operand is ready.
- ◆ The read register pipeline stage reads the corresponding source operand from the physical register file for the issued instruction and sends it to the corresponding
  Functional parts.
- ◆ The execution pipeline executes instructions according to the type of instructions and writes the calculation results back to the register file;
  Retain the station and register renaming table to inform that the corresponding register value is available.
- ◆ The commit pipeline stage submits the executed instructions in the order of the programs recorded in the Reorder queue, GS464 has the most
  You can submit four instructions per beat, and the submitted instructions are sent to the register rename table for confirmation of its purpose.
  The renaming relationship of the device and release the physical register originally allocated to the same logical register, and send it to the memory access queue
  Allow those submitted instructions to write to the cache or memory.

The above is the pipeline level of basic instructions. For some more complicated instructions, such as fixed-point multiply and divide instructions, floating-point instructions a

The memory access instruction requires multiple shots during the execution phase. The basic structure of GS464 is shown in the figure below.

13

Page 16

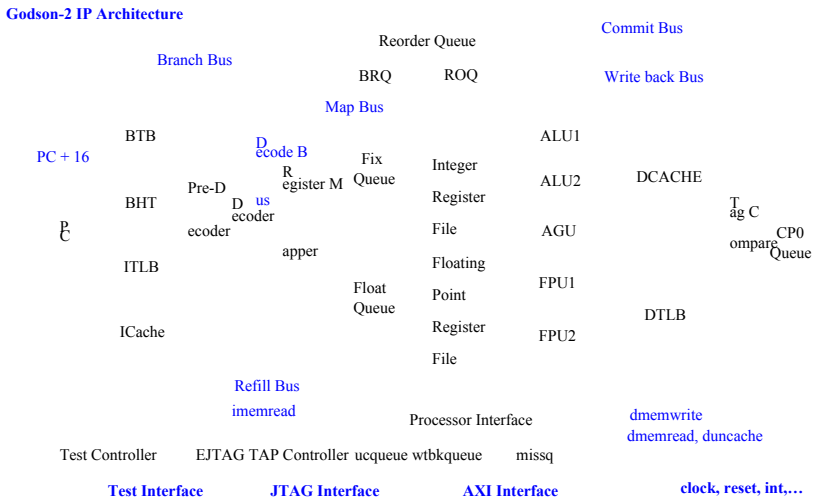


Figure 1-1 Basic structure of GS464 processor

14

Godson 3A1000 Processor User Manual • Next

2 Loongson GS464 processor core instruction set overview

Loongson GS464 processor core is compatible with MIPS64 R2 architecture, which implements the full definition of MIPS64 R2 specification A set of required instructions, as well as some custom extended instructions.

Godson GS464 core processor implemented in the MIPS64 R2 compatible instructions include 2 section 5.1 in. Because of the space, The detailed definition of this part of the instruction is not given in this document. Readers really need to understand the detailed definition of the relevant instructions, it is recomme MIPS Architecture Specification Version 2.50 Volume I and Volume II. The implementation-related content of this part of the instruction will be in section 2.2 In the article, there are several MIPS64 R2 instructions GS464 processor core implementation and MIPS architecture specifications are not the same This is also explained in Section 2.2 .

Godson custom extension instructions implemented by the processor core GS464 recited in 2. Section 1.3. Due to space limitations, this The detailed definition of the sub-command is not given in this document. Readers really need to understand the detailed definitions of relevant directives Command System Manual. "Godson Command System Manual" is currently only available to authorized customers.

2.1 MIPS64 compatible instruction list

According to the instruction function, the MIPS64 compatible instructions implemented by the GS464 processor core can be divided into the following groups:

- ◆ Memory access instruction
- ◆ Operation instruction
- ◆ Branch and jump instructions
- ◆ Coprocessor instructions
- ◆ Other instructions

2.1.1 Memory access instruction

The MIPS architecture uses a load / store architecture. All operations are performed on the register, only the memory access instruction can Access the stored data. Memory access instructions include reading and writing of various width data, unsigned reading, unaligned memory access and atomic Save etc.

Table 2-1 CPU instruction set: memory access instruction

Instruction mnemonic	Instruction function brief	ISA compatibility level
LB	Fetch bytes	MIPS32
LBU	Take unsigned byte	MIPS32

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief	ISA compatibility level
LH	Take half word	MIPS32
LHU	Take unsigned halfword	MIPS32
LW	Fetch word	MIPS32
LWU	Take unsigned word	MIPS32
LWL	Take the left part of the word	MIPS32
LWR	Take the right part of the word	MIPS32

LD	Take double word	MIPS64
LDL	Take the left part of the double word	MIPS64
LDR	Take the right part of the double word	MIPS64
LL	Take the address of the sign	MIPS32
LLD	Take the double word address of the mark	MIPS64
SB	Save byte	MIPS32
SH	Half word	MIPS32
SW	Save word	MIPS32
SWL	Save the left part	MIPS32
SWR	Save the right part	MIPS32
SD	Save double word	MIPS64
SDL	Save double word left	MIPS64
SDR	Save double word right	MIPS64
SC	Save under the conditions	MIPS32
SCD	Save double word	MIPS64

2.1.2 Operation instructions

Operational instructions complete the arithmetic, logic, shift, multiplication, and division of register values. Operational instructions include Register instruction format (R-type, operands and operation results are saved in registers) and immediate instruction format (I-type, One of the operands is a 16-bit immediate)

Table 2-2 CPU instruction set: arithmetic instructions (ALU immediate)

Instruction mnemonic	Instruction function brief	ISA compatibility level
ADDI	Add immediate	MIPS32
DADDI	Add double word immediately	MIPS64

16

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief	ISA compatibility level
ADDIU	Add unsigned immediate	MIPS32
DADDIU	Add unsigned double word immediately	MIPS64
SLTI	Less than immediate setting	MIPS32
SLTIU	Unsigned less than immediate setting	MIPS32
ANDI	And immediate	MIPS32
ORI	Or immediate	MIPS32
XORI	XOR immediate	MIPS32
LUI	Take immediate to high	MIPS32

Table 2-3 CPU instruction set: arithmetic instructions (3 operands)

Instruction mnemonic	Instruction function brief	ISA compatibility level
ADD	plus	MIPS32
DADD	Double word plus	MIPS64
ADDU	Unsigned plus	MIPS32
DADDU	Unsigned double word plus	MIPS64

SUB	Less	MIPS32
DSUB	Double word minus	MIPS64
SUBU	Unsigned minus	MIPS32
DSUBU	Unsigned double word subtraction	MIPS64
SLT	Less than setting	MIPS32
SLTU	Unsigned less than setting	MIPS32
AND	versus	MIPS32
OR	or	MIPS32
XOR	XOR	MIPS32
NOR	NOR	MIPS32

Table 2-4 CPU instruction set: arithmetic instructions (2 operands)

Instruction mnemonic	Instruction function brief	ISA compatibility level
CLO	Leading 1 number	MIPS32
DCLO	Leading 1 word	MIPS64
CLZ	Leading 0 number	MIPS32
DCLZ	Leading double words	MIPS64

WSBH	Halfword swap	MIPS32 R2
DSHD	Half word exchange	MIPS64 R2
SEB	Byte sign extension	MIPS32 R2
SEH	Half-character extension	MIPS32 R2
INS	Bit insertion	MIPS32 R2
EXT	Bit extraction	MIPS32 R2
DINS	Double word insertion	MIPS64 R2
DINSM	Double word insertion	MIPS64 R2
DINSU	Double word insertion	MIPS64 R2
DEXT	Double word bit extraction	MIPS64 R2
DEXTM	Double word bit extraction	MIPS64 R2
DEXTU	Double word bit extraction	MIPS64 R2

Table 2-5 CPU instruction set: multiplication and division instructions

Instruction mnemonic	Instruction function brief	ISA compatibility level
MUL	Multiply to general register	MIPS32
MULT	Multiply	MIPS32
DMULT	Double word multiplication	MIPS64
MULTU	Unsigned multiplication	MIPS32
DMULTU	Unsigned double word multiplication	MIPS64
MADD	Multiply-add	MIPS32
MADDU	Unsigned multiply-add	MIPS32
MSUB	Multiplication and subtraction	MIPS32
MSUBU	Unsigned multiplication and subtraction	MIPS32
DIV	except	MIPS32
DDIV	Double word division	MIPS64

DIVU	Unsigned division	MIPS32
DDIVU	Unsigned double word division	MIPS64
MFHI	Fetch data from hi register to general register	MIPS32
MTHI	Store data from general register to hi register	MIPS32
MFLO	Fetch from lo register to general register	MIPS32
MTLO	Store data from general register to lo register	MIPS32

Table 2-6 CPU instruction set: shift instruction

Instruction mnemonic	Instruction function brief	ISA compatibility level
SLL	Logical shift left	MIPS32
SRL	Logical shift right	MIPS32
SRA	Arithmetic shift right	MIPS32
SLLV	Variable logical shift left	MIPS32
SRLV	Variable logical shift right	MIPS32
SRAV	Variable arithmetic shift right	MIPS32
ROTR	Rotate right	MIPS32 R2
ROTRV	Variable loop right shift	MIPS32 R2
DSLL	Double word logical shift left	MIPS64
DSRL	Double-word logical shift right	MIPS64
DSRA	Double-word arithmetic shift right	MIPS64
DSLLV	Variable double word logical shift left	MIPS64
DSRLV	Variable double word logical shift right	MIPS64
DSRAV	Variable double word arithmetic shift right	MIPS64
DSLL32	Double word logic shift left +32	MIPS64
DSRL32	Double word logical shift right +32	MIPS64
DSRA32	Double word arithmetic shift right +32	MIPS64
DROTR	Double word rotation right	MIPS64 R2
DROTR32	Double word rotate right +32	MIPS64 R2
DROTRV	Double word variable circular right shift	MIPS64 R2

2.1.3 Branch and jump instructions

Branch and jump instructions can change the control flow of the program, including the following four types:

◆ PC relative conditional branch

◆ PC unconditional jump

◆ Register absolute jump

◆ Procedure call

In the definition of MIPS, all transfer instructions are followed by a delay slot instruction. Likely the delay slot of the transfer instruction is only in Executed when the transfer is successful, non-Likely transfer instruction delay slot instructions are always executed. Return address of procedure call instruction It is stored in the No. 31 register by default. Jumping according to the No. 31 register will be considered to return from the called process.

Table 2-7 CPU instruction set: jump and branch instructions

Instruction mnemonic	Instruction function brief	ISA compatibility level
J	Jump	MIPS32
JAL	Immediate call procedure	MIPS32
JR	Jump to the instruction pointed to by the register	MIPS32
JR.HB	Jump to the instruction pointed to by the register	MIPS32 R2
JALR	Register call process	MIPS32
JALR.HB	Register call process	MIPS32 R2
BEQ	Jump if equal	MIPS32
BNE	Jump if not equal	MIPS32
BLEZ	Less than or equal to 0	MIPS32
BGTZ	Greater than 0 jump	MIPS32
BLTZ	Less than 0 jump	MIPS32
BGEZ	Greater than or equal to 0	MIPS32
BLTZAL	Less than 0 calling procedure	MIPS32
BGEZAL	Greater than or equal to 0 calling procedure	MIPS32
BEQL	Likely jumps Likely	MIPS32
BNEL	If not equal, Likely jump	MIPS32
BLEZL	Less than or equal to 0 then Likely jump	MIPS32
BGTZL	If it is greater than 0, Likely jumps	MIPS32
BLTZL	If less than 0, Likely jumps	MIPS32
BGEZL	Greater than or equal to 0, Likely jump	MIPS32
BLTZALL	If less than 0, Likely calls the procedure	MIPS32
BGEZALL	Greater than or equal to 0 then Likely calls the procedure	MIPS32

2.1.4 Coprocessor instructions

Coprocessor instructions complete operations within the coprocessor. Loongson GS464 processor core has two coprocessors: No. 0 coprocessor Processor (system processor) and coprocessor 1 (floating point coprocessor).

Coprocessor No. 0 (CP0) manages memory and handles exceptions through CP0 registers. These instructions are listed in Table 2-8 in.

The MIPS architecture specification clearly defines floating-point instructions in the No. 1 coprocessor (CP1) instruction. Godson

The specific implementation of the floating point instruction in the coprocessor 1 instruction in the GS464 processor core will be introduced separately in Chapter 7.

Table 2-8 CPU instruction set: CP0 instruction

Instruction mnemonic	Instruction function brief	ISA compatibility level
DMFC0	Fetch double word from CP0 register	MIPS64

DMTC0	Write double word to CP0 register	MIPS64
MFC0	Get from CP0 register	MIPS32
MTC0	Write to CP0 register	MIPS32
TLBR	Read indexed TLB entries	MIPS32
TLBWI	Indexed TLB entry	MIPS32
TLBWR	Write random TLB entries	MIPS32
TLBP	Search for matches in TLB	MIPS32
CACHE	Cache operation	MIPS32
ERET	Exception return	MIPS32
DI	Disable interrupt	MIPS32 R2
EI	Allow interrupt	MIPS32 R2

2.1.5 Other instructions

In MIPS64, there are other instructions besides those listed above. For details, see Table 2 -9 to

Table 2-9 CPU instruction set: special instructions

Instruction mnemonic	Instruction function brief	ISA compatibility level
SYSCALL	System call	MIPS32
BREAK	Breakpoint	MIPS32
SYNC	Synchronize	MIPS32
SYNCI	Synchronous instruction cache	MIPS32 R2

Table 2-10 CPU instruction set: abnormal instruction

Instruction mnemonic	Instruction function brief	ISA compatibility level
TGE	Greater than or equal to	MIPS32
TGEU	Unsigned number is greater than or equal to trap	MIPS32
TLT	Less than trapped	MIPS32

Instruction mnemonic	Instruction function brief	ISA compatibility level
TLTU	Unsigned number is less than trapped	MIPS32
TEQ	Equal to falling into	MIPS32
TNE	Wait for	MIPS32
TGEI	Greater than or equal to immediate	MIPS32
TGEIU	Greater than or equal to unsigned immediate	MIPS32
TLTI	Less than immediate	MIPS32
TLTIU	Immediate number less than unsigned	MIPS32
TEQI	Equal to immediate	MIPS32
TNEI	Not equal to immediate	MIPS32

Table 2-11 CPU instruction set: conditional move instructions

Instruction mnemonic	Instruction function brief	ISA compatibility level
MOVF	Conditional movement when floating-point conditional false	MIPS32
MOVN	Conditional move when general register is not 0	MIPS32

MOVT	Conditional movement when floating-point condition is true	MIPS32
MOVZ	Conditional move when general register is 0	MIPS32

Table 2-12 CPU instruction set: other instructions

Instruction mnemonic	Instruction function brief	ISA compatibility level
PREF	Prefetch instruction	MIPS32
PREFX	Prefetch instruction	MIPS32
NOP	No operation	MIPS32
SSNOP	Single launch air operation	MIPS32

2.2 MIPS64 compatible instruction implementation related instructions

2.2.1 There are instructions to achieve differences

Godson GS464 processor core supports all MIPS64 R2 instructions, but some implementation-related instructions

Redefinition has been made, see Table 2-13 .

Table 2-13 Instructions with implementation differences

Instruction mnemonic	Instruction function brief	Specific implementation description
PREF	Prefetch instruction	Treated as NOP instruction processing, no prefetching effect. Software pre-fetching can be achieved through Load to register 0.
PREFX	Prefetch instruction	Treated as NOP instruction processing, no prefetching effect. Software pre-fetching can be achieved through Load to register 0.
SSNOP	Single launch air operation	Treated as a NOP instruction. All data correlation and control correlation (CP0 Hazards) in GS464 are maintained by hardware, No software control is required.
EHB	Isolated execution	Treated as a NOP instruction. All data correlation and control correlation (CP0 Hazards) in GS464 are maintained by hardware, No software control is required.
WAIT	Enter wait state	Treated as NOP instruction processing, no stop pipeline effect. Avoid using this instruction in code involving low power management, not only can not achieve soft The design of the device may even cause increased power consumption.
RDPGPR	Read shadow register	GS464 does not implement shadow registers, so the source and target registers are both The current register group.
WRPGPR	Write shadow register	GS464 does not implement shadow registers, so the source and target registers are both The current register group.
SYNC	Synchronize	GS464 only implements the SYNC instruction with stype = 0, it will be reported when stype is other values The exception to the reserved instructions. This instruction acts as a memory barrier and is used to Ensure that the memory access operation before SYNC has been determined to be completed (for example, the data of the store instruction Read and write to dcache, unchaced is completed, load has retrieved the value to the register At the same time, and ensure that the memory access operation after the SYNC instruction has not begun to execute. SYNC The instruction requires CU [0], only kernel mode can be used.

twenty three

Instruction mnemonic	Instruction function brief	Specific implementation description
		The differences between the CACHE instruction implemented by GS464 and the MIPS64 specification are mainly There are two points: **. 1** , **Index** class **CACHE** address instruction analytically The index CACHE instruction implemented by GS464 uses the lowest 2 bits of the virtual address as the It is the selection signal of the Cache, instead of the slave defined in the MIPS64 specification. Intercept in the middle of the virtual address. **2** , **CACHE28** , **CACHE29** , **CACHE30** , **CACHE31** instructions include Righteousness In the processor, CACHE28, CACHE29, CACHE30 and CACHE31 refer to The meaning of the command (ie, op [4: 2] = 0b111 Cache instruction) is different from the MIPS64 specification. These instructions in the Godson processor are Index Store Data operations, while the MIPS64 regulations Fan Zhong is Fetch and Lock operation.
CACHE	cache operation	The effective Cache operation types in the GS464 processor core are listed as follows: **op [4: 0]** — Functional description b00000 — Invalid I-Cache line based on index b01000 — Write I-Cache line tag according to index b11100 — Write I-Cache row data according to index b00001 — According to the invalid index and write back the D-Cache line b00101 — Read D-Cache line tag according to index b01001 — Write D-Cache line tag according to index b10001 — Invalid D-Cache line based on hit b10101 — According to the invalid hit and write back the D-Cache line b11001 — Read D-Cache row Data according to index b11101 — Write D-Cache row data according to index b00011 — According to the invalid index and write back the S-Cache line b00111 — Read S-Cache line tags based on index b01011 — Write S-Cache line Tag according to index b10011 — Based on invalid hit and write back S-Cache line b11011 — Read S-Cache row Data according to index b11111 — Write S-Cache row Data according to index

2.2.2 Disable instruction

The GS464 processor core disables the following instructions, see Table 2-14 :

twenty four

Page 27

Godson 3A1000 Processor User Manual • Next

Table 2-14 Disable instruction

Instruction mnemonic	Instruction function brief	ISA compatibility level
DI	Disable interrupt	MIPS32 R2
EI	Allow interrupt	MIPS32 R2

2.3 Custom extended instructions

The custom extended instructions implemented by Loongson GS464 processor are divided into the following categories according to functions:

◆ Memory access instruction

◆ Multiplication and division instructions

◆ **X86** binary acceleration instruction

◆ **64** -bit multimedia commands

◆ Miscellaneous instructions

2.3.1 Custom extended memory access instruction

Table 2-15 Custom extended memory access instructions

Instruction mnemonic	Instruction function brief
GSLE	Exception if less than or equal to wrong address
GSGT	Exception if greater than wrong address
GSLBLE	Fetch bytes with out-of-bounds check
GSLBGT	Fetch byte with lower bound check
GSLHLE	Take half-word with cross-border inspection
GSLHGT	Take half word with cross-border check
GSLWLE	Take the word out of check
GSLWGT	Take the word with cross-border inspection
GSLDLE	Take double words with cross-border inspection
GSLDGT	Take double word with cross-border check
GSLQ	Double target register takes fixed-point four words
GSLBX	Fetch byte with offset
GSLHX	Halfword with offset

25

Page 28

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief
GSLWX	Fetch word with offset
GSLDX	Double word with offset
GSSBLE	Stored bytes with out-of-bounds check

GSSBGT	Stored bytes with lower bound check
GSSHLE	Save half-word with cross-border inspection
GSSHGT	Save half word with cross-border check
GSSWLE	Bring the deposit with cross-border inspection
GSSWGT	Save the word with cross-border inspection
GSSDLE	Save double word with cross-border inspection
GSSDGT	Save double word with cross-border check
GSSQ	Dual source registers store fixed-point four words
GSSBX	Byte with offset
GSSHX	Halfword with offset
GSSWX	Store word with offset
GSSDX	Double word with offset

2.3.2 Custom extended multiplication and division operation instructions

Table 2-16 User-defined extended multiplication and division instructions

Instruction mnemonic	Instruction function brief
GSMULT	Signed word multiplication, result is written to general register
GSDMULT	Signed double word multiplication, write result to general register
GSMULTU	Unsigned word multiplication, the result is written to the general register
GSDMULTU	Unsigned double word multiplication, the result is written to the general register
GSDIV	Signed word division, quotient write general register
GSDDIV	Signed double word division, quotient write general register
GSDIVU	Unsigned word division, quotient to write general register
GSDDIVU	Unsigned double word division, quotient write general register
GSMOD	Divide signed words, write remainder to general register
GSDMOD	Signed double word division, write remainder to general register
GSMODU	Unsigned word division, remainder write to general register

Instruction mnemonic	Instruction function brief
GSDMODU	Unsigned double word division, write remainder to general register

2.3.3 Custom extended X86 binary translation acceleration instruction

Table 2-17 Custom extended X86 binary translation acceleration instructions

Instruction mnemonic	Instruction function brief
X86AND	Set only the logical bits of EFLAG in x86 mode
X86OR	Only set the logical bits of EFLAG in x86 mode or
X86XOR	Only set the logical bit XOR of EFLAG in x86 mode
X86DADD	Set only double word plus of EFLAG in x86 mode
X86ADD	Only set the word addition of EFLAG in x86 mode
X86DADDU	Set only EFLAG's no exception double word plus in x86

X86ADDU	In x86 mode, only set the EFLAG without exception word plus
X86DSUB	Only set the double word subtraction of EFLAG in x86 mode
X86SUB	Only set the word subtraction of EFLAG in x86 mode
X86DSUBU	Set only EFLAG's no exception double word subtraction in x86 mode
X86SUBU	Set only EFLAG's no exception word subtraction in x86 mode
X86INC	Only set the double word increment of EFLAG in x86 mode 1
X86DEC	Only set the double word decrement of EFLAG by 1 in x86
X86DSLL	Set only double-word left shift of EFLAG in x86 mode
X86SLL	Set only the left shift of EFLAG in x86 mode
X86DSLL32	Set the shift amount of EFLAG plus the double-word logical left shift of 32 in x86 mode
X86DSLLV	Set the double-word variable shift amount of EFLAG to the left by x86
X86SLLV	Set the variable shift amount of the word EFLAG only to the left in x86 mode
X86DSRL	Set the double-word logical right shift of EFLAG only in x86 mode
X86SRL	Set only the right logical shift of EFLAG in x86 mode
X86DSRL32	In x86 mode, only set the shift amount of EFLAG plus 32 double-word logic right shift
X86DSRLV	Set only double-word variable shift logic right shift of EFLAG in x86 mode
X86SRLV	Set only the right shift of the word variable shift amount of EFLAG in x86 mode
X86DSRA	Set only double-word arithmetic right shift of EFLAG in x86 mode
X86SRA	Set only the word arithmetic right shift of EFLAG in x86 mode

27

Page 30

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief
X86DSRA32	In x86 mode, only set the shift amount of EFLAG plus 32 double-word logical arithmetic right shift
X86DSRAV	Set the double-word variable shift arithmetic right shift of EFLAG only in x86 mode
X86SRAV	Arithmetic right shift of only variable word shift amount of EFLAG in x86 mode
X86DROTR	Set only double-word circular right shift of EFLAG in x86 mode
X86ROTR	Set only the EFLAG word rotation right in x86 mode
X86DROTR32	In x86 mode, only set the shift amount of EFLAG plus 32 double-word logical circular right shift
X86DROTRV	Double-word circular right shift with only variable displacement of EFLAG set in x86 mode
X86ROTRV	Set the variable shift amount of EFLAG only in x86 mode
X86MFFLAG	Extract the value of the EFLAG flag in x86
X86MTFLAG	Modify the value of the EFLAG flag in x86
X86J	Jump based on EFLAG value in x86
X86LOOP	Cycle through EFLAG values in x86 mode
SETTM	Set in x86 floating point stack mode
CLRTM	x86 floating-point stack mode clear
INCTOP	x86 floating point stack top pointer plus 1
DECTOP	x86 floating point stack top pointer minus 1
MTTOP	Write x86 floating point stack top pointer
MFTOP	Read x86 floating point stack top pointer
SETTAG	Judge and set register

2.3.4 Custom extended 64-bit multimedia acceleration instructions

Table 2-18 Custom extended 64-bit multimedia acceleration instructions

Instruction mnemonic	Instruction function brief
PADDSH	Four 16-bit signed integer additions, signed saturation
PADDUSH	Four 16-bit unsigned integer addition, unsigned saturation
PADDH	Four 16-digit plus
PADDW	Two 32-digit plus
PADDSB	Eight 8-bit signed integer additions, signed saturation
PADDUSB	Eight 8-bit unsigned integer additions, unsigned saturation
PADDB	Eight 8-digit plus

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief
PADDD	64 digit plus
PSUBSH	Four 16-bit signed integer subtraction, signed saturation
PSUBUSH	Four 16-bit unsigned integers, unsigned saturation
PSUBH	Four 16-digit minus
PSUBW	Two 32-bit minus
PSUBSB	Eight 8-bit signed integer subtraction, signed saturation
PSUBUSB	Eight 8-bit unsigned integers minus, unsigned saturation
PSUBB	Eight 8-digit minus
PSUBD	64 digit minus
PSHUFH	Shuffle four 16-digit numbers
PACKSSWH	Convert 32-bit signed integer to 16-bit, signed saturation
PACKSSHB	16-bit signed integer converted to 8-bit, signed saturation
PACKUSHB	16-bit signed integer converted to 8-bit, unsigned saturation
PANDN	fs is negated and ft bitwise and
PUNPCKLHW	16 digits lower unpacking
PUNPCKHHW	16 digits high unpacking
PUNPCKLBH	8 digits lower unpacking
PUNPCKHBH	8 digits high unpacking
PINSRH_0	The lower 16 digits of ft are inserted into the lower 16 digits of fs
PINSRH_1	The lower 16 digits of ft are inserted into the lower 16 digits of fs
PINSRH_2	The lower 16 digits of ft are inserted into the lower 16 digits of fs
PINSRH_3	The lower 16 digits of ft are inserted into the lower 16 digits of fs
PAVGH	Four 16-bit unsigned integers are averaged
PAVGB	Eight 8-bit unsigned integers are averaged
PMAXSH	Four 16-bit signed integers, whichever is greater
PMINSH	Four 16-bit signed integers take the smaller value
PMAXUB	Eight 8-bit unsigned integers, whichever is greater
PMINUB	Eight 8-bit unsigned integers take the smaller value
PCMPEQW	Two 32-digit equal comparisons
PCMPGTW	Two 32-bit signed integers are greater than the comparison
PCMPEQH	Four 16-digit equality comparisons
PCMPGTH	Four 16-bit signed integers are greater than the comparison
PCMPEQB	Eight 8-digit equality comparisons

29

Page 32

Instruction mnemonic	Instruction function brief
PCMPGTB	Eight 8-bit signed integers are greater than the comparison
PSLLW	Two 32-digit logical shifts to the left
PSLLH	Four 16-digit logical shift left
PMULLH	Multiply four 16-bit signed integers and take the lower 16 bits
PMULHH	Multiply four 16-bit signed integers and take the result as high 16 bits
PMULUW	Multiply the lower 32-bit unsigned integers and store the 64-bit result
PMULHUH	Multiply four 16-bit unsigned integers and take the result as high 16 bits
PSRLW	Two 32-bit logical shift right
PSRLH	Four 16-digit logical shift right
PSRAW	Two 32-digit arithmetic shift right
PSRAH	Four 16-digit arithmetic shift right
PUNPCKLWD	The lower 32-bit array synthesizes 64 digits
PUNPCKHWD	The upper 32-bit array synthesizes 64 digits
PASUBUB	Eight 8-bit unsigned integers are subtracted and take the absolute value
PEXTRH	fs 16 bits are copied to fd lower 16 bits, fd higher bits are filled with 0
PMADDHW	Four 16-bit signed numbers are multiplied and the low and high bits are accumulated separately
BIADD	Multi-byte accumulation
PMOVMSKB	Conditional byte shift
GSXOR	logical OR of fs and ft
GSNOR	fs and ft logical bit NOR
GSAND	fs and ft logical bitwise AND
GSADDU	fs and ft fixed-point unsigned word plus
GSOR	fs and ft fixed-point logical OR
GSADD	fs and ft fixed-point words plus
GSDADD	fs and ft fixed-point double word addition
GSSEQU	fs and ft fixed-point number equal comparison
GSSEQ	fs and ft fixed-point number equal comparison
GSSUBU	fs and ft fixed-point unsigned word subtraction
GSSUB	fs and ft fixed-point subtraction
GSDSUB	fs and ft fixed-point double word subtraction
GSSLTU	fs and ft fixed-point unsigned fixed-point number is less than comparison
GSSLT	fs and ft fixed-point fixed-point number is less than comparison
GSSLL	fs and ft fixed-point logical shift left

30

Page 33

Instruction mnemonic	Instruction function brief
GSDSLL	fs and ft fixed-point logical double shift left
GSSRL	fs and ft fixed-point logical shift right

GSDSRL	fs and ft fixed-point logical shift right double word
GSSRA	fs and ft fixed-point arithmetic shift right
GSDSRA	fs and ft fixed-point arithmetic right shift double word
GSSLEU	fs and ft fixed-point unsigned fixed-point number less than or equal to the comparison
GSSLE	fs and ft fixed-point fixed-point number is less than or equal to the comparison

2.3.5 Custom extended miscellaneous instructions

Table 2-19 Custom extended miscellaneous instructions

Instruction mnemonic	Instruction function brief
CAMPV	Query the lookup table and return the content of the hit
CAMPI	Query the lookup table and return the index of the hit
CAMWI	Write the specified item in the lookup table
RAMRI	Read the contents of the specified item in the lookup table

3 CP0 control register

This chapter describes the operation of Coprocessor 0 (Coprocessor 0, referred to as CP0). The main contents include CP0 register definition and CP0 instruction implemented by Loongson No. 3 processor. The CP0 register is used to control the state change of the processor and report the current state of the processor. These registers are read by the MFC0 / DMFC0 instruction or written by the MTC0 / DMTC0 instruction. CP0 register as shown in Table 3-1 by Show.

When the processor is running in core mode or the 28th bit (CU0) in the status register (Status register) is set, you can To use the CP0 instruction. Otherwise, executing the CP0 instruction will result in "Coprocessor Unavailable Exception".

Table 3-1 CP0 Register

Register number		Register name	description
Total number	Subnumber		
0	0	Index	Writable register, used to specify TLB entries that need to be read / written
1	0	Random	Pseudo-random counter for TLB replacement

2	0	EntryLo0	The content of the lower half of the TLB entry corresponding to the even virtual page (mainly Physical page number)
3	0	EntryLo1	The content corresponding to odd virtual pages in the lower half of TLB entries (mainly Physical page number)
4	0	Context	Virtual page translation table (PTE) pointing to the kernel in 32-bit addressing mode
5	0	Page Mask	Set the mask value of the TLB page size
5	1	Page Grain	Whether the logo supports large page addresses
6	0	Wired	Number of fixed-line TLB entries (refers to those that are not used for random replacement Low-end TLB entries)
7	0	Hwrena	Hardware register enable
8	0	BadVaddr	Wrong virtual address
9	0	Count	counter
10	0	EntryHi	The upper half of TLB entries (virtual page number and ASID)
11	0	Compare	Counter comparison
12	0	Status	Processor status register
12	1	IntCtl	Extended interrupt control register
12	2	SRSCtl	Shadow register bank control register
13	0	Cause	Cause of the last exception
14	0	EPC	Exception program counter
15	0	PRid	Processor revision identification number
15	1	EBase	Exception vector base address
16	0	Config	Configuration register

16	1	Config1	Configuration register 1
16	2	Config2	Configuration register 2
16	3	Config3	Configuration register 3
17	0	LLAddr	Link read memory address
18			Keep
19			Keep
20	0	Xcontext	Virtual page conversion table (PTE) pointing to the kernel in 64-bit addressing mode
twenty one			Keep
twenty two	0	Diagnose	Enable / disable BTB, RAS and clear ITLB table
twenty three	0	Debug	EJTAG debug register
twenty four	0	DEPC	EJTAG debug exception program counter
25	0/1/2/3	PerfCnt	Performance counter
26	0	ErrCtl	Parity / ECC check control and status
27	0	CacheErr	Cache ECC check control and status
28	0	TagLo	The lower half of the CACHE TAG register
28	1	DataLo	Used to interact with and diagnose cache data queues
29	0	TagHi	The upper half of the CACHE TAG register
29	1	DataHi	Used to interact with and diagnose cache data queues
30	0	ErrorEPC	Error exception program counter
31	0	DESAVE	Scratchpad for Debug exception handling

3.1 Index register (0, 0)

The Index register is a 32-bit readable / writable register, and the last six bits of the value are used to index TLB entries. Register

The high bit indicates whether the TLB probing (TLBP) instruction was successfully executed.

The value of the Index register indicates the y TLB read (TLBR) and TLB index write (TLBWI) instructions.

Figure 3-1 shows the format of the Index reg lescribes the meaning of each field of the Index register.

31 30		6 5	0
P	0	Index	
1	25	6	

Figure 3-1 Index register

Table 3-2 Field description of Index register

area	description
33	
P	Probe failed. Set to 1 when the last TLB probe command (TLBP) was unsuccessful
Index	The index value of the TLB entry indicating the operation of the TLB read instruction and TLB index write instruction
0	Reserved. It must be written as 0 and returns 0 when read.

3.2 Random register (1, 0)

The Random register is a read-only register, in which the lower six bits index TLB entries. After each instruction is executed, the register Value minus 1. At the same time, the register value floats between an upper bound and a lower bound. The upper and lower bounds are specifically:

- The lower bound is equal to the number of TLB entries reserved for the operating system (that is, the contents of the Wired register).
- The upper bound is equal to the number of items in the entire TLB minus 1 (maximum is 64-1).

The Random register indicates the TLB entry that will be operated by the TLB random write instruction. For this purpose, there is no need to read this register. But this register is readable to verify that the corresponding operation of the processor is correct.

To simplify testing, the Random register is set to the upper bound when the system restarts. In addition, when the Wired register is written, the register The device should also be set to the upper bound.

Figure 3-2 shows the format of the Random registe ibes the meaning of each field of the Random register.

31		6 5	0
	0	Random	
	26	6	

Figure 3-2 Random register

Table 3-3 Random register fields

area	description
Random	Random TLB index value
0	Reserved. It must be written as 0 and returns 0 when read.

3.3 EntryLo0 (2, 0) and EntryLo1 (3, 0) registers

The EntryLo register includes two registers of the same format:

- EntryLo0 is used for even virtual pages
- EntryLo1 is used for odd and virtual pages

The EntryLo0 and EntryLo1 registers are both read / write registers. When performing TLB read and write operations, they include TLB respectively The physical page number (PFN) of the parity page in the entry. Figure 3-3 shows the format of these registers.

34

63	42 41	6 5 3 2 1 0	
0	PFN	C	DVG
2	36	3	1 1 1

Figure 3-3 EntryLo0 and EntryLo1 registers

The PFN field of the EntryLo0 and EntryLo1 registers is the upper 36 bits (47:12) of the 48-bit physical address.

Table 3-4 EntryLo Register Field

area	description
PFN	The page number is the high-order bit of the physical address.
C	Cache consistency property of TLB page.
D	Dirty bit. If this bit is set, the page is marked as dirty, that is, writable. Actually this one is soft It is used as a write protection to prevent data from being changed.
V	Valid bit. When this bit is set, it means that the TLB entry is valid, otherwise it will generate a TLBL or The exception is TLBS.
G	Global bit. When the G bits in EntryLo0 and EntryLo1 are both set to 1, the processor will be in TLB Ignore ASID when searching.
0	Reserved. It must be written as 0 and returns 0 when read.

There is only one global bit in each TLB entry, which is written according to the values of EntryLo0 [0] and EntryLo1 [0] in the TLB write operation.

3.4 Context (4, 0)

The Context register is a read / write register that contains a pointer to an item in the page table. The page table is an operating system number
According to the structure, the conversion of the virtual address to the physical address is stored.

When a TLB exception occurs, the CPU will load the TLB from the page table based on the failed conversion. In general, the operating system uses
The Context register addresses the mapping of the current page in the page table. The Context register copies part of the information in the BadVAddr register, but
This information is arranged in a form that facilitates the are TLB exception handlers.

Figure 3-4 shows the format of the Context registe s the context register fields.

63	4 3	0
P EBase	BadVPN2	0
41	19	4

Figure 3-4 Context register

Table 3-5 Context register field

area	description
BadVPN2	This field is written by hardware when TLB is an exception. It contains the virtual page number of the virtual address that could not be effectively converted recently

35

(VPN).

This field is a read / write field used by the operating system. The value written in this field allows the operating system to send the Context

PTEBase	The memory serves as a pointer to the current page table in memory.
0	Reserved. It must be written as 0 and returns 0 when read.

The 19-bit BadVPN2 field contains the 31:13 bits of the virtual address that caused the TLB exception; the 12th bit was excluded because of a single

Of TLB entries are mapped to a parity page pair. For a page size of 4K bytes, this format can directly address PTE table entries as 8.

Page tables that are long in bytes and organized by pairs. For pages and PTEs of other sizes, moving and masking this value can generate a suitable address.

3.5 PageMask register (5, 0)

The PageMask register is a readable and writable register, which is used in t ıg and writing TLB;

A different page size TLB entry set, as shown in Table 3- 6 The format of this reg ;ure 3-5.

TLB read and write operations use this register as a source or destination; w ual and real address translation, the TLB corresponds to PageMask

The corresponding bits in the register indicate which of the virtual address bits 24:13 are used for comparison. When the value of the MASK field is not the value in Table 3-6 , the TLB

The operation is undefined. The 0 field is reserved and must be written as 0, and returns 0 when read.

31	25	24	13	12	0
0		Mask		0	
7		12		13	

Figure 3-5 PageMask Register

Table 3-6 Mask values for different page sizes

Page size	Bit											
	twenty four	twenty three	twenty two	twenty one	20	19	18	17	16	15	14	13
4Kbytes	0	0	0	0	0	0	0	0	0	0	0	0
16 Kbytes	0	0	0	0	0	0	0	0	0	0	1	1
64 Kbytes	0	0	0	0	0	0	0	0	1	1	1	1
256 Kbytes	0	0	0	0	0	0	1	1	1	1	1	1
1 Mbytes	0	0	0	0	1	1	1	1	1	1	1	1
4 Mbytes	0	0	1	1	1	1	1	1	1	1	1	1
16M bytes	1	1	1	1	1	1	1	1	1	1	1	1

3.6 PageGrain Register (5, 1)

The PageGrain register is a readable and writable register. Godson 3 only defines the 29th bit of this register: ELPA (Enable

Large Physical Adress), the remaining bits are reserved as 0.

When ELPA = 1, Loongson No. 3 supports 48-bit physical address; when ELPA = 0, Loongson No. 3 only supports 40-bit physical address.

Whether the ELPA bit can be written depends on the LPA fie egister. When the LPA bit of Config3 is 0, PageGrain's ELPA

The bit is set to 0. The format of this register is shown in Figure 3-6 ields are shown in Table 3-7.

31	30	29	28	0
	ELPA			
2	1			29

Figure 3-6 PageGrain Register

Table 3-7 PageGrain Register Field

area	description
ELPA	Whether this field is set indicates whether it supports large physical addresses
0	Reserved. It must be written as 0 and returns 0 when read.

3.7 Wired register (6, 0)

The Wired register is a readable / writable register. The value of this register specifies the fixed and random entries in the TLB.

The limits are shown in Figure 3-7 . Wired entries are fixed and irreplaceable entries. The contents of these entries will not be written by TLB modify. The contents of random entries can be modified.

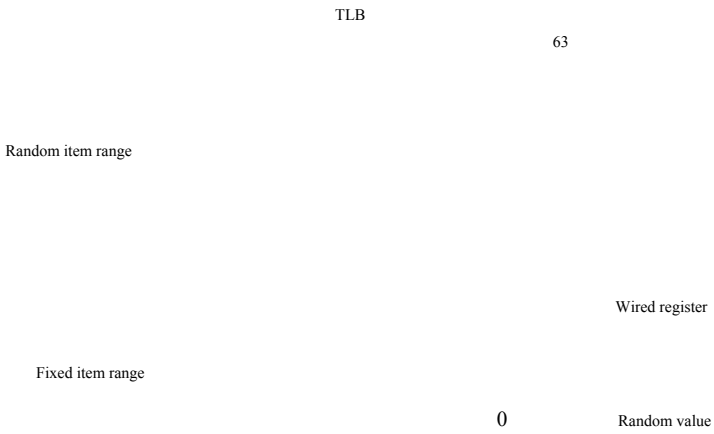


Figure 3-7 Wired register limits

The Wired register is set to 0 at system reset. When writing to this register, set the Random register value to the upper limit (see Random register).

Figure 3-8 shows the format of the Wired regi bes the fields of this register.

37

Figure 3-8 Wired register

Table 3-8 Wired Register Domain

area	description
Wired	TLB fixed entry boundary
0	Reserved. It must be written as 0 and returns 0 when read.

3.8 HWREna register (7, 0)

The HWREna register is a readable and writable register. Godson 3 only defines the Mask field of this register, which is used to indicate instructions Source hardware register of RDHWR. The 0 field is reserv :n as 0, and returns 0 when read.

Figure 3-9 shows the format of the HWREna registe : the correspondence between the Mask field and the hardware register.

Figure 3-9 HWREna register

Table 3-9 Correspondence between Mask fields and hardware registers

Hardware register	Bit			
	3	2	1	0
CPUnum	0	0	0	1
SYNCI_Step	0	0	1	0
CC	0	1	0	0
CCRes	1	0	0	0

3.9 BadVAddr register (8, 0)

The error virtual address register (BadVAddr) is a read-only register that records the most recent example of a TLB or addressing error Virtual address outside. Unless a software reset occurs, and NMI or Cache errors are exceptions, the BadVAddr register will remain unchanged. no Then this register is undefined.

Figure 3-10 shows the format of the wrong virtual address register.

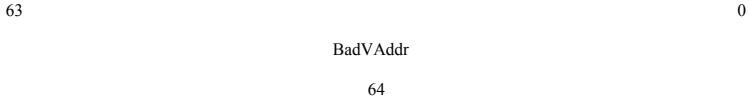


Figure 3-10 BadVAddr register

3.10 Count register (9, 0) and Compare register (11, 0)

The Count register and Compare register are both 32-bit read-write registers.

The Count register works as a real-time timer, incrementing every two clock cycles.

The Compare register is used to generate an interrupt at a specific time. This register is written with a value and is continuously Compare the values in the registers. Once these two values are equal, an interrupt request is generated. The TI and IP [7] in the Cause register are set. When the Compare register is written again, the TI bit c eared.

Figure 3-11 shows the format of the Count regist : format of the Compare register.

Figure 3-11 Count register

Figure 3-12 Compare register

3.11 EntryHi Register (10, 0)

The EntryHi register is used to store the high-order bits of TLB entries during TLB reading and writing.

The EntryHi register can be used by TLB Probe, TLB Write Random, TLB Write Indexed, and TLB Read Indexed Instruction access.

Figure 3-13 EntryHi Register

Table 3-10 EntryHi Register Field

39

area	description
VPN2	The virtual page number is divided by 2 (mapped to double pages); the high-order bits of the virtual address.
ASID	The address space identifies the domain. An 8-bit field; used to allow multiple processes to share the TLB; for the same Virtual page number, each process has a different mapping with other processes.
R	Regional bits. (00-> user, 01-> super user, 11-> core) for matching vAddr63 ... 62
0	Reserved. It must be written as 0 and returns 0 when read.

The VPN2 domain contains 61:13 bits of 64-bit virtual addresses.

When a TLB Refill, TLB Invalid, or TLB Modified exception occurs, there is no virtual address that matches the TLB entry The virtual page number (VPN2) and ASID will be loaded into the EntryHi register.

3.12 Status register (12, 0)

The Status Register (SR) is a read-write register that includes the operation mode, interrupt enable, and processor status diagnosis. List below Describes some of the more important Status register fields : Figure 3-14 shows the format of the entire register, including the description of the fields. Among them The required domains are:

- The 8-bit interrupt mask (IM) field controls the enable of 8 interrupt conditions. The interrupt must be enabled before being triggered. The corresponding bits of the interrupt mask field of the register and the interrupt pending field of the Cause register should be set. For more information, please refer to Cause Register interrupt pending (IP) field.
- The 4-bit coprocessor availability (CU) field controls the availability of 4 possible coprocessors. Regardless of how the CU0 bit is set, CP0 is always available in kernel mode.

Status register format

Figure 3-14 shows the format of the Status regist ibes the fields of the Status register.

31 28 27 26 25 24 23 22 21 20 19 18 16 15 8 7 6 5 4 2 1 0

CU 0 FR 0 PX BEV 0 SR NMI 0 IM7-IM0 KX SX UX KSU ERL EXL IE

(cu3: cu0)

4 1 1 2 1 1 1 1 1 2 8 1 1 1 1 1 1

Figure 3-14 Status register

Table 3-11 Status Register Field

area	description
CU	Control the availability of 4 coprocessor units. Regardless of how the CU0 bit is set, CP0 is always available in kernel mode of. 1- Available 0- Not available The initial value of the CU field is 0011
0	Reserved. It must be written as 0 and returns 0 when read.

40

FR	Enable additional floating-point registers s 0-16 registers 1-32 registers
PX	Enable 64-bit operation in user mode (64-bit operation in other modes does not need to be enabled) 1-enable 0-not enabled (at this time, whether the 64-bit operation is available in user mode needs to judge the UX bit)
BEV	Control the entry address of the exception vector 0-normal 1-Start
SR	1 means a soft reset exception occurred
NMI	Whether an NMI exception occurred. Note that the software cannot write this bit from 0 to 1.
IM	Interrupt mask: control the enable of each external, internal and software interrupt. If the interrupt is enabled, it will be allowed to trigger, At the same time, the corresponding bit in the interrupt pending field of the Cause register is set. 0-prohibited 1-Allow
KSU	Mode bit 11 . Undefined 10 . general user 01 . root 00 . core
KX	1 – Enable 64-bit Kernel segment access; use XTLB Refill vector. 0 – No access to 64-bit Kernel segment; use TLB Refill vector
SX	1 – Enable 64-bit Supervisor segment access; use XTLB Refill vector. 0 – cannot access 64-bit Supervisor segment; use TLB Refill vector
UX	1 – Enable 64-bit User segment access; use XTLB Refill vector. 0 – cannot access the 64-bit User segment; use TLB Refill vector
ERL	Error level. The processor will reset this bit when a reset, software reset, NMI or Cache error occurs. 0 . normal 1 . error
EXL	Exceptional level. When an exception that is not caused by a reset, software reset, or cache error occurs, the processor will set That bit.
IE	The interrupt is enabled. 0 . Disable all interrupts 1 . Enable all interrupts

Status register mode and access status

The fields used to set the mode and access status in the Status register are described below:

● Interrupt enable: When the following conditions are met, the interrupt is enabled:

IE = 1 and

41

Page 44

Godson 3A1000 Processor User Manual • Next

EXL = 0 and

ERL = 0.

If these conditions are encountered, the setting of the IM bit allows interruption.

● Operation mode: When the processor is in normal user, kernel and super user mode, the following bit fields need to be set.

When KSU = 10 2 , EXL = 0 and ERL = 0, the processor is in normal user mode.

When KSU = 01 2 , EXL = 0 and ERL = 0, the processor is in super user mode.

When KSU = 00 2 , or EXL = 1 or ERL = 1, the processor is in kernel mode.

● Kernel address space access: When the processor is in kernel mode, the kernel address space can be accessed.

● Super user address space access: When the processor is in kernel mode or super user mode, you can access the super user address space.

● User address space access: The processor can access the user address space in all three operating modes.

Status register reset

At reset, the value of the Status register is 0x30c000e4.

3.13 IntCtl register (12, 1)

The IntCtl register is a 32-bit register that can be read and written. It manages the extended interrupts in the Release2 system. Loongson 3 realization Vector interrupt. Use the VS field of the IntCtl register to represent the vector space between interrupt vectors. 1 field is not writable, read as 1; 0 field To keep, it must be written as 0, and it returns 0 when r 5 bits represent HW5 shared by the clock interrupt and the Performance Counter interrupt.

Figure 3-15 shows the format of the IntCtl regist bes the correspondence between the VS field and the vector space.

31	26 25		10 9	5 4	0
	1	0	VS	0	
	6	16	5	5	

Figure 3-15 IntCtl register

Table 3-12 Correspondence between VS domain coding and vector space

coding	Vector space (**16** hex)	Vector space (**10** decimal)
0x00	0x000	0
0x01	0x020	32
0x02	0x040	64
0x04	0x080	128
0x08	0x100	256
0x10	0x200	512

3.14 SRSCtl register (12, 2)

The SRSCtl register is a 32-bit readable and writable register. It controls the shadow register set in the processor. Since Godson 3 only A group of general registers, no shadow registers, so the shadow of general registers is the general register itself, SRSCtl in Godson 3 The register only implements two fields: ESS and PSS.

Figure 3-16 shows the format of the SRSCtl register, Table 3-13Describes the fields of the SRSCtl register.

31		15 12 11 10		9 6 5		0
16						
	0	ESS	0	PSS	0	
	16	4		4	6	

Figure 3-16 SRSCtl register

Table 3-13 SRSCtl Register Field

area	description
ESS	The shadow register set used for exceptions. Only 0 in Godson 3
PSS	The previous shadow register set. Only 0 in Godson 3
0	Reserved. It must be written as 0 and returns 0 when read.

3.15 Cause register (13, 0)

The 32-bit readable and writable Cause regis f the most recent exception.

Figure 3-17 shows the format of this register es the fields of the Cause register. A 5-digit exception code (ExcCode)

One of the reasons is pointed out, as shown in Tabl

31 30 29 28 27 26 25 24 23 22 16 15								8 7 6			2 1 0
BD	TI	CE		C PCI		0	IV	0	IP7-IP0	0 Exc Code	0
1	1	2	1	1	2	1	7	8	1	5	2

Figure 3-17 Cause register

Table 3-14 Cause Register Field

area	description
BD	Indicate whether the last adopted exception is in the branch delay slot. 1-Delay slot 0-normal
TI	Clock interrupt indication 0-no clock interrupt 1-There is time to wait for processing

CE	The unit number of the coprocessor when an exception of coprocessor unavailability occurs.
DC	Disable count register 0-count register available 1-Count register is disabled
PCI	Performance counter interrupt indication 0-No performance counter interrupt 1-There is a performance counter interrupt pending
IV	Interrupt exception entry 0-Use generic exception vector (0x180) 1-Use special interrupt vector (0x200)
IP	Point out the waiting interrupt. This bit will remain unchanged until the interrupt is removed. IP0 ~ IP1 are soft interrupt bits, which can be set by software With clear. 1-Interrupt wait 0-no interruption
ExcCode	Exception code field (see Table 3-15)
0	Reserved. It must be written as 0 and returns 0 when read.

Table 3-15 ExcCode field of Cause register

Exception code	Mnemonic	description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (read or fetch instruction)
3	TLBS	TLB exception (storage)
4	AdEL	Address error exception (read or fetch instruction)
5	AdES	Address error exception (storage)
6	IBE	Bus error exception (fetch instruction)
7	DBE	Bus error exception (data reference: read or store)
8	Sys	System call exception
9	Bp	Breakpoint exception
10	RI	Exceptions to reserved orders
11	CpU	Coprocessor unavailable exception
12	Ov	Arithmetic overflow exception
13	Tr	Trap exception

14	-	Keep
15	FPE	Floating point exception
16	IS	Stack exception
17-18	-	Keep
19	DIB	Debug command exception

20	DDBS	Debug save data exception
twenty one	DDBL	Debug fetch data exception
twenty two	-	Keep
twenty three	WATCH	WATCH exception
24-25	-	Keep
26	DBP	Debug breakpoint exception
27	DINT	Debug Exception
28	DSS	Debug single step exception
29	-	Keep
30	CACHERROR	cache error exception
31	-	Keep

3.16 Exception Program Counter register (14, 0)

Exception Program Counter (EPC) is a read / write register, which includes exception handling

After the end, continue to process the address.

For synchronization exceptions, the content of the EPC register is one of the following:

- Instruction virtual address, which is the direct cause of the exception, or
- The previous branch or jump instruction (when the instruction is in the branch delay slot, the instruction delay bit is set in the Cause register)

Virtual address.

When the EXL bit in the Status register is set, the processor does not write to the EPC register.

Figure 3-18 shows the format of the EPC register.

63

0

EPC

64

Figure 3-18 EPC register

3.17 Processor Revision Identifier (PRID) register (15, 0)

The PRID register is a 32 read-only register, which con version and revision of the calibration processor and CP0 version

Version information. Figure 3-19 shows the format of this reg s the fields of this register.

31	15	8 7	0
16			
0	Imp	Rev	
16	8	8	

Figure 3-19 Processor Revision Identifier register

Table 3-16 PRId Register Field

area	description
IMP	Implementation version number
REV	Revision number
0	Reserved. It must be written as 0 and returns 0 when read.

The lower bits (7: 0) of the PRID register can be used as the revision number, and the higher bits (15: 8) can be used as the implementation version number. The implementation version number of Godson 3 is 0x63, and the revised version number is 0x03.

The format of the version number is YX, where Y (7: 4 digits) is the major version number and X (3: 0 digits) is the minor version number.

The version number can distinguish the version of some processors, but there is no guarantee that any changes to the processor will be reflected in the PRID register. In other words, there is no guarantee that changes to the version number must reflect changes to the processor. For this reason, the value of the register is not given, and Software cannot also rely on the version number in the PRID register to identify the processor.

3.18 EBase register (15, 1)

The EBase register is a readable and writable register that contains the exception vector base address and a read-only CPU number.

When BEV = 0 in the status register, the base add

Figure 3-20 shows the format of the EBase regist

Table 3-17 describes the fields of the EBase regis

31 30 29		12 11 10 9		0
1	0	Exception Base	0	CPUNum
1		18	2	10

Figure 3-20 Ebase register

Table 3-17 Ebase register field

area	description
1	Not write-only read, read as 1
0	Reserved. It must be written as 0 and returns 0 when read.
Exception Base	Jointly specify the base address of the exception vector with 31 bits and 30 bits

CPUNum	In a multi-core system, used to indicate the processor number

3.19 Config register (16, 0)

The Config register specifies various configuration options in the Loongson 3 processor; Table 3-18 lists these options.

Some configuration options defined by bits 31: 3 of the Config register are set by the hardware at reset and are read-only status bits Included in the Config register for software access. Other configuration options (Bits 2: 0 of the Config register) are readable / writeable and Controlled by software. These fields are undefined at reset.

The configuration of the Config register is limited. The Config register should be initialized by software before Cache is used, and,

The cache should be re-initialized after making any chan

Figure 3-21 shows the format of the Config registe :he fields of the Config register. Initial value of Config register

It is 0x00030932.

1 3		16 15 14 13 12 10 9 7 6 4 3 2 0					
M	0	BE AT		AR	MT	0	V K0
1	15	4	3	3	1	1	1 3

Figure 3-21 Config register

Table 3-18 Config register field

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
M	Whether there is a config1 register, set to 1 means there is.
BE	Indicate the end type
	1-big end
	0-little end
AT	Specify the architecture type
	0 – MIPS32
	1-MIPS64, can only access the 32-bit address segment
	2-MIPS64, can access all address segments
	3-reserved
AR	Specified version
	0 – Release 1
	1 – Release 2
	2-7-Reserved
MT	Indicate the type of memory management unit
	0-no mapping
	1-Standard TLB
	2-7-Reserved

47

Page 50

Godson 3A1000 Processor User Manual • Next

VI	Indicate whether there is a virtual instruction cache
	0 – instruction cache is not virtual
	1 – The instruction cache is virtual
K0	Kseg0 consistency algorithm (Cache consistency algorithm)
	2 – Uncached
	3 – Cacheable
	The rest-reserved

3.20 Config1 register (16, 1)

The Config1 register specifies the cache configuration of the Godson 3 processor.

The Config1 register is an additional content of the

Figure 3-22 shows the format of the Config1 registe

Table 3-19 describes the fields of the Config1 regist

31 30 25 24 22 21 19 18 16 15 13 12 10 9 7 6 5 4 3 2 1 0																			
M	MMU Size-1			IS			IL			IA			DS			DL	DA	C2	MD	PC	WR	A	EP	F
1	6			3			3			3			3			3	1	1	1	1	1	1	1	1

Figure 3-22 Config1 register

Table 3-19 Config1 register field

area	description
M	If there is a config2 register, set to 1 to indicate existence.
MMU Size-1	Number of TLB entries minus 1
IS	Icache groups per channel Code meaning 0 64 1 128 2 56 3 12 4 1024 5 2048 6 4096 7 Keep
IL	Icache size per group Code meaning 0 No Icache 1 4 bytes 2 8 bytes 3 16 bytes 4 32 bytes 5 64 bytes 6 128 bytes 7 Keep
IA	Icache connection method Code meaning 0 Directly connected 1 2 way 2 3-way connection 3 4-way connection 4 5 channels connected 5 6 channels 6 7 Ways 7 8 channels
DS	Dcache groups per channel Edit meaning 64 1 128 2 256 3 512 4 1024 5 2048 6 4096 7 Keep
DL	Dcache size per group Code meaning 0 No Dcache 1 4 bytes 2 8 bytes 3 16 bytes 4 32 bytes 5 64 bytes

48

Page 51

Godson 3A1000 Processor User Manual • Next

	6	128 bytes
	7	Keep
	Dcache connection method	
	Code meaning	
DA	0	Directly connected
	1	2 way
	2	3-way connection

	3	4-way connection
	4	5 channels connected
	5	6 channels
	6	7 Ways
	7	8 channels
	Whether No. 2 coprocessor is implemented	
C2	0-not implemented	
	1-Implementation	
	Whether MDMX ASE is implemented	
MD	0-not implemented	
	1-Implementation	
	Whether the performance count register is implemented	
PC	0-not implemented	
	1-Implementation	
	Whether the watch register is implemented	
WR	0-not implemented	
	1-Implementation	
	Whether MIPS16e is realized	
CA	0-not implemented	
	1-Implementation	
	Whether EJTAG is implemented	
EP	0-not implemented	
	1-Implementation	
	Whether the FPU is realized	
FP	0-not implemented	
	1-Implementation	

3.21 Config 2 register (16, 2)

The Config2 register specifies the configuration of the secondary cache in the Godson 3 processor.

As the Config2 register, all contents are read-only a set.

Figure 3-23 shows the format of the Config1 registe he fields of the Config2 register. Config2 register

The initial value is 0x80001643.

31	30 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
M	TU	TS	TL	TA	SU	SS	SL	A
1	3	4	4	4	4	4	4	4

Godson 3A1000 Processor User Manual • Next

Figure 3-23 Config2 register

Table 3-20 Config2 register field

area	description
M	If there is a config3 register, set to 1 to indicate existence.
TU	Three-level cache control or status bit
TS	Number of groups per level of cache Code meaning 0 6 1 128 2 256 3 512 4 1024 2048 6 4096 7 8192 8-15 Reserved
TL	Level 3 cache per group size Code meaning 0 No Icache 1 4 bytes 2 8 bytes 3 16 bytes 4 32 bytes 5 64 bytes 6 128 bytes 7 56 bytes 8-15 Reserved
TA	Three-level cache connection Code meaning 0 Directly connected 1 2 way 2 3-way connection 3 4-way connection 4 5 channels connected 5 6 channels 6 7 Ways 7 8 channels -15 Reserved
SU	Secondary cache control or status bit
SS	Number of secondary cache groups

Godson 3A1000 Processor User Manual • Next

Code meaning

0 64

1 128

2 56

3 512

4 1024

5 2048

6 4096

7 8192

	8-15 Reserved
SL	The size of each group of secondary cache
	Code meaning
	0 No Icache
	1 4 bytes
	2 8 bytes
	3 16 bytes
	4 32 bytes
	5 64 bytes
	6 128 bytes
	7 256 ytes
	8-15 Reserved
SA	Secondary cache connection method
	Code meaning
	0 Directly connected
	1 2 way
	2 3-way connection
	3 4-way connection
	4 5 channels connected
	5 6 channels
	6 7 way phase
	8 channels
	8-15 Reserved

3.22 Config 3 registers (16, 3)

The Config3 register marks whether some functions ntents are read-only, and are automatically set at reset.

Figure 3-24 shows the format of the Config1 registe he fields of the Config1 register. Config1 register

The initial value is 0x000000a0.

31	30 1	10 9 8 7 6 5 4	3 2 1 0
M	0	DSPP 0 LPA VEIC Vint SP 0 MT SM TL	
1	20	1 1 1 1	1 1 1 1 1 1

Figure 3-24 Config3 register

Table 3-21 Config3 register field

area	description
M	Keep
0	Reserved. It must be written as 0 and returns 0 when read.
	Whether MIPS DSPASE is implemented
DSPP	0-not implemented
	1-Implementation
	Whether the large physical address is realized
LPA	0-not implemented
	1-Implementation
	Whether the external interrupt controller is implemented
VEIC	0-not implemented

	1-Implementation
	Whether vector interrupt is realized
Vint	0-not implemented
	1-Implementation
	Whether small page support is realized
SP	0-not implemented
	1-Implementation
	Whether MIPS MTASE is realized
MT	0-not implemented
	1-Implementation
	Whether SmartMIPS ASE is realized
SM	0-not implemented
	1-Implementation
	Whether Trace Logic is implemented
TL	0-not implemented
	1-Implementation

3.23 Load Linked Address (LLAddr) register (17, 0)

The LLAddr register is a 64-bit read-only register. The LLAddr register is used to store the address of the most recent load-linked instruction Page number PFN, when an exception is returned (when an eret instruction occurs), the LLAddr register is cleared. The format of this register in Godson 3 As shown in Figure 3-25 .

63	48 47	12 11	0
0	PFN	0	
16	36	12	

Figure 3-25 LLAddr Register

3.24 XContext register (20, 0)

The readable and writable XContext register contains a pointer to an entry in the operating system's page table. When a TLB case occurs Outside, the operating system loads the TLB from the page table based on the failed conversion.

The XContext register is used for XTLB refill processing, handles the loading of TLB entries in the 64-bit address space, and is only used by the operating system use. The operating system sets the PTEBase field in the regi

Figure 3-26 shows the format of the XContext registe he fields of the XContext register.

63	41 40 39 38	4 3	0
PTEBase	R	BadVPN	0
twenty three	2	35	4

Figure 3-26 XContext register

The 35-bit BadVPN2 domain contains 47:13 bits of the virtual address that caused the TLB exception, because a single TLB entry maps to a Parity page pairs, so the 12th bit is not included. When the page size is 4K bytes, this format can directly address PTE The table entry is an 8-byte long page table organized by pairs. For other pages and PTE sizes, the shift and mask can get the correct address.

Table 3-22 XContext register domain

area	description
BadVPN2	When a TLB exception occurs, the hardware will write this field, which contains the virtual page number of the most recent invalid virtual address except 2.
R	This field contains the 63:62 bits of the virtual address. 00 . general user 01 . root

	11 . Kernel
0	Reserved. It must be written as 0 and returns 0 when read.
PTEBase	Read / write field, this value allows the operating system to use the XContext register as a pointer to memory Pointer to the current page table.

3.25 Diagnostic register (22, 0)

The 64-bit registers unique to the Loongson processor are ma internal queues and special operations of the processor. Figure 3-27 The Diagnostic register shows the format of the Diagnostic register fields of the Diagnostic register.

63	8 7	6	5	4	3 2	1	0
0	W-CAC W-ISS		S-ISS	S-FET	0 ITLB BTB RAS		
56	1	1	1	1	1	1	1	1

Figure 3-27 Diagnostic Register

Table 3-23 Diagnostic Register Field

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
W-CAC	Remove the limit of wait-cache operation
W-ISS	Remove the limit of wait-issue operation
S-ISS	Remove restrictions on store-issue operations
S-FET	Remove restrictions on store-fetch operations
ITLB	Clear ITLB when writing 1
BTB	Clear BTB when writing 1
RAS	The use of RAS is prohibited when writing 1.

3.26 Debug register (23, 0)

The Debug register is a 32-bit readable and writable register. The Debug register contains the most recent debug exception or debug The reason for the exception that occurs in mode, it also controls single-step interrupts. This register indicates debug resources and other internal states. only There are LSNM field and SSt field can be written. When reading Debug register in non-debug mode, only DM bit and EJTAGver field can be read. Loongson 3 did not implement the power saving mode ir tion. At reset, the initial value of the Debug register is: 0x02018000.

Figure 3-28 shows the format of the Debug registe he fields of the Debug register.

31	30	29	28	27	26	25	twenty four	17	16	15	14	10	9	8	7	6	5	4	3	2	1	0
DBD	DM	NoDCR	LSNM	0	Count M	0	EJTAGver			DExcCode		No St	SSt	0	DINT		DIB	DDBS	DDBL	DBp		DSS
1	1	1	1	2		1	7			3		5	1	1	2	1	1	1		1		1

Figure 3-28 Debug register

Table 3-24 Debug register fields

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
DBD	Indicate whether the last debug exception occurred in the delay slot. 0-not 1-yes
DM	Indicates whether the processor is in Debug mode 0 – Non-Debug Mode 1 – Debug Mode
NoDCR	Indicate whether the dseg segment exists 1-does not exist 0-exists
LSNM	When the dseg segment exists, specify the addresses where loads / stores are available 0 – dseg segment 1-System memory
CountDM	The working state of Count register when entering DM 0 – stopped 1 – runnig
EJTAGver	EJTAG version 0 – version 1 and 2.0 1-Version 2.5 2 – Version 2.6 3-Version 3.1 4-reserved
DexcCode	Specify the reason for the last exception in Debug mode
NoSSt	Indicate whether to support single-step interrupt 0-support 1-Not supported
SSt	Single-step interrupt enable bit 0-not available

	1-enable
DINT	Set to indicate that a Debug interrupt exception occurred Automatically cleared when entering Debug mode
DIB	Set to indicate that a Debug instruction interrupt exception occurred Automatically cleared when entering Debug mode
DDBS	Set to indicate that a Debug data interrupt exception occurred Automatically cleared when entering Debug mode

DBp	Set to indicate that a Debug breakpoint exception occurred Automatically cleared when entering Debug mode
DSS	Set to indicate that a Debug single-step interrupt exception occurred Automatically cleared when entering Debug mode

The bits or fields in the Debug register are only updated when an exception occurs in debug exception or debug mode.

3.27 Debug Exception Program Counter register (24, 0)

Debug Exception Program Counter (DEPC) is a 64-bit read / write register, which includes the continuation after the exception processing is completed Address. This register is updated by hardware in debug exception or exception in debug mode.

For precise debug exceptions and precise debug mode exceptions, the contents of the DEPC register is one of the following:

- Instruction virtual address, which is the direct cause of the exception, or
- The previous branch or jump instruction (when the instruction is in the branch delay slot, the instruction delay bit DBD is changed in the Debug register Set) virtual address.

Figure 3-29 shows the format of the DEPC register.

63 0

DEPC

64

Figure 3-29 DEPC register

3.28 Performance Counter Register (25, 0/1/2/3)

Loongson No. 3 processor defines four (two groups) performance counters, which are mapped to sel 0 of CP0 register No. 24, sel 1, sel 2 and sel 3.

When Godson 3 resets, the initial values assigned to the two control registers of the PerfCnt register are:

PerfCnt, select 0 = 0xc0000000

PerfCnt, select 2 = 0x40000000

The purpose of these four registers is shown in Table 3-25 As shown, the fo hown in Figure 3-30 (the two groups have the same format), control The definition of the enable bit of the register is shown in Table 3-26 :

57

Table 3-25 Performance counter list

Performance counter	sel	Use description
0	select 0	Control register 0
	select 1	Count register 0
1	select 2	Control register 1
	select 3	Count register 1

Each counter is a 64-bit read / write register and increments every time a countable event occurs in the associated control domain. Per count Both devices can independently count an event.

Control register sel 0

31	30	29	11	10 5	4	3 2 1 0
M	W	0		Event	IE	USK EXL
1	1	19		6	1	1 1 1 1

Count register sel 0

6 1

Event Count

64

Figure 3-30 Performance Counter Register

When the first bit (63 bits) of the counter becomes 1 (counter overflows), the counter will trigger an interrupt and the PCI register in the Cause register The bit is set to one (if there are multiple sets of counters, the

Table 3-26 Count enable bit definition

Count enable bit	**Count Qualifier (CP0 Status** register field)
M	Is there another set of counters
	1-yes
	0-no
W	Count register bit width
	0 – 32 bits
	1 – 64 bits

K	KSU = 0 (kernel mode), EXL = 0, ERL = 0
S	KSU = 1 (Super User Mode), EXL = 0, ERL = 0
U	KSU = 2 (normal user mode), EXL = 0, ERL = 0
EXL	EXL = 1, ERL = 0

Table 3-27 Counter 0 events

event	signal	description
0000	Cycles	cycle
0001	Brbus.valid	Branch instruction
0010	Jrcount	JR Directive
0011	Jr31count	JR instruction and the field rs = 31
0100	Imemread.valid & imemread_allow	Level 1 I-cache is missing
0101	Rissuebus0.valid	Alu1 operation has been launched
0110	Rissuebus2.valid	Mem operation has been launched
0111	Rissuebus3.valid	Falu1 operation has been launched
1000	Brbus_bht	BHT guess instruction
1001	Mreadreq.valid & Mreadreq_allow	Read from main memory
1010	Fxqfull	Fixed the number of times the launch queue is full
1011	Roqfull	The number of times the queue is full
1100	Cp0qfull	CP0 queue full times
1101	Exbus.ex & excode = 34,35	Tlb refill exception
1110	Exbus.ex & Excode = 0	exception
1111	Exbus.ex & Excode = 63	Internal exception

Table 3-28 Counter 1 events

event	signal	description
0000	Cmtbus? .Valid	Submit operation
0001	Brbus.brerr	Branch prediction failed
0010	Jrmiss	JR prediction failed
0011	Jr31miss	JR and rs = 31 prediction failure
0100	Dmemread.valid & Dmemread_allow	First-level D-cache is missing
0101	Rissuebus1.valid	Alu2 operation has been launched

0110	Rissuebus4.valid	Falu2 operation has been launched
0111	Duncache_valid & Duncache_allow	Access is not cached
1000	Brbus_bhtmiss	BHT guess error
1001	Mwritereq.valid & Mwritereq_allow	Write to main memory
1010	Ftqfull	The number of times the floating-point pointer queue is full
1011	Brqfull	The number of times the branch queue is full
1100	Exbus.ex & Op == OP_TLBPI	Itlb is missing
1101	Exbus.ex	Total exceptions
1110	Mispec	Loading speculation is missing
1111	CP0fwd_valid	CP0 queue is loaded forward

3.29 ECC register (26, 0)

Godson 3 uses the optional ErrCtl register 26 in tl CC verification.

Figure 3-31 shows the format of the DEPC regist ıe fields of the ECC register.

Figure 3-31 ECC register

Table 3-29 ECC register fields

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
ECC	A double-word check code for the relevant cache

3.30 CacheErr register (27, 0/1)

Loongson No. 3 will complete the ECC check of Loongson No. 3 in the MIPS64 standard by software and hardware. The hardware is only responsible for checking errors. After detecting data errors, the content is saved in control rc

The CacheErr1 register field.

60

63 4 5 twenty one 0

0 ECCWay ECCType

58 4 2

Figure 3-32 CacheErr register

Table 3-30 CacheErr Register Field

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
ECCWay	Different codes indicate different errors of Cache in case of Cache check error
ECCType	00-Incorrect instruction cache
	01-Data cache is wrong
	10-Secondary cache error
	11-The chip interface bus is wrong

63 0

ECCAddr

64

Figure 3-33 CacheErr1 register

Table 3-31 CacheErr1 Register Field

area	description
ECCAddr	The virtual address where the verification error occurred

3.31 TagLo (28) and TagHi (29) registers

The TagLo and TagHi registers are 32-bit read / write registers, used to store the tags and status of the primary / secondary cache, use The CACHE and MTC0 instructions write to the Tag register.

Figure 3-34 shows the format of these registers for primary cache (P-Cache) operatic o and TagHi Definition of the domain in the register.

61

TagLo register

31 8 7 6 5 4 3 0

PTAG [23: 0] CS SCSETI 0

twenty four 2 2 4

TagHi register

Figure 3-34 TagLo and TagHi registers (P-Cache)

Table 3-32 Cache Tag register field

area	description
PTAG	Specify the 39:12 bits of the physical address.
CS	Specifies the status of Cache.
SCSETI	The group number of the corresponding Cache line in the secondary cache (the field of the secondary cache is 0)
0	Reserved. It must be written as 0 and returns 0 when read.

3.32 DataLo (28, 1) and DataHi (29, 1) registers

DataLo and DataHi are read-only registers, used only for interaction and diagnosis with cache data queues. CACHE instruction The IndexLoadTag operation will read the corresponding data to the DataLo or DataHi register . Figure 3-35 lists the DataLo register and DataHi register format.

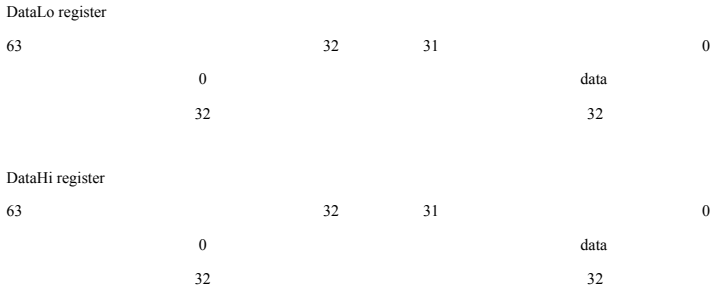


Figure 3-35 DataLo and DataHi registers

62

Page 65

Godson 3A1000 Processor User Manual • Next

3.33 ErrorEPC register (30, 0)

With the exception of ECC and parity errors, the ErrorEPC register is similar to the EPC register. It is used in reset, software reset Bit, and non-maskable interrupt (NMI) exceptions store program counter.

ErrorEPC is a read-write register that contains the virtual address where the instruction resumes execution after processing an error. FIG 3-36 significant The format of the ErrorEPC register is shown.

Figure 3-36 ErrorEPC register

3.34 DESAVE register (31, 0)

The DESAVE register is a 64-bit register that can be read and written. His function is a simple scratchpad for debugging exceptions Save the value of a general register during processing so that this general register retains other contexts.

Figure 3-37 shows the format of the DESAVE register .

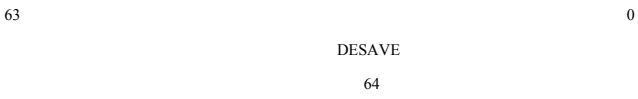


Figure 3-37 DESAVE register

3.35 CP0 instruction

Table 3-33 lists the CP0 instructions defined by Godson-2 processor.

Table 3-33 CP0 instructions

OpCode	Description	MIPS ISA
DMFC0	Fetch double word from CP0 register	III
DMTC0	Write double word to CP0 register	III
MFC0	Get from CP0 register	I
MTC0	Write to CP0 register	I
TLBR	Press Index to read TLB entries	III
TLBWI	Press Index to write TLB entries	III
TLBWR	Press Random to write TLB entries	III
TLBP	Find the pairing index in TLB	III

63

CACHE	Cache operation	III
ERET	Exception return	III
DERET	Debug returns	EJTAG
DI	Close interrupt	MIPS32 R2
EI	Open interrupt	MIPS32 R2
RDHWR	Read hardware registers	MIPS32 R2
RDPGPR	Read from shadow register	MIPS32 R2
WRPGPR	Write to shadow register	MIPS32 R2
SDBBP	Software breakpoint	EJTAG

Related

Loongson processor can handle the pipeline correlation in hardware, including CP0 correlation and memory access correlation, so CP0 instruction does not need NOP instruction to correct the instruction sequence.

64

4 CACHE organization and operation

Loongson 3 uses three independent caches:

First-level instruction cache: a total of 64KB, the cache line size is 32 bytes, using a four-way group structure.

First-level data cache: a total of 64KB, the cache line size is 32 bytes, a four-way group structure is used, and a write-back strategy is adopted.

Level 2 Hybrid Cache: On-chip Cache, which communicates with the processor core through a 128-bit AXI bus, and has a total of 4 level 2 Cache modules. Each secondary Cache module is globally addressed, with a Cache line size of 32 bytes and a capacity of 1M. It uses a four-way group structure and uses write-back Strategy.

4.1 Cache overview

Fixed-point access to the primary cache requires 3 clock cycles, and floating-point access to the primary cache requires 4 clock cycles. Each First-level caches have their own data paths, so that two caches can be accessed simultaneously. Read, write and refill of first-level cache The roads are all 128 bits.

The secondary cache uses a 256-bit data path, and it is only accessed when the primary cache fails. Secondary cache and one Level 1 Cache cannot be accessed at the same time. When Level 1 Cache fails, access to Level 2 Cache will increase the cost of failure by at least 14 cycles (in The processor core and secondary cache of Godson 3 need to communicate through the crossbar switch, so an additional 6 beats are required). Secondary Cache The speed of 128-bit data per clock cycle backfills the primary cache.

The first-level cache uses virtual address indexes and physical address marks, while the second-level cache indexes and signs use physical addresses. The virtual address index may cause inconsistencies. At present, Godson 3 uses the operating system to ensure that it will not cause inconsistencies.

Loongson 3 uses a directory-based cache consistency protocol to ensure that each processor core and IO write can be used by other processor cores And IO correctly observed. A directory is maintained in the secondary cache. For each cache line in the secondary cache, the directory uses 32 bits The bit vector of records whether each level 1 Cache (including instruction and data Cache) has a backup of the Cache line Utilization of Godson 3 The hardware maintains the consistency between the first-level instruction cache, the first-level data cache, the second-level cache, and the HT external device. The software does not You need to use the Cache command to flash the Cache to maintain consistency.

4.1.1 Non-blocking Cache

Loongson 3 implements non-blocking Cache technology. The non-blocking Cache is to allow the Cache to fail to access the memory behind the multiple Cache failure or hit memory access operations continue to improve the overall performance of the system.

In a blocking Cache design, when a Cache fails, the processor will suspend subsequent memory access operations. at this time, The processor starts a storage cycle, fetches the requested data, fills it in the cache, and then resumes execution. This operation process Will occupy more clock cycles, depending on the design of the memory system.

However, in a non-blocking Cache design, Cache does not pause on a failure. Godson-3 supports hits under multiple failures, It can support up to 24 cache failures.

When the primary cache fails, the processor checks the secondary cache to see if the required data is in it. If the secondary cache is still lost To be effective, you need to access the main memory.

65

Page 68

Godson 3A1000 Processor User Manual • Next

The non-blocking cache structure in Loongson 3 can use loop unrolling and software pipelining more effectively. In order to maximize
To take advantage of Cache, before using the instructions to access the data, the corresponding Load operation should be performed as early as possible.

For those I / O systems that require sequential access, the default setting of Loongson 3 is the blocking Uncached access method.

Godson No. 3 provides prefetch instructions, which can be prefetched to the first level data by loading to the fixed point register No. 0
Cache. In addition, the DSP engine in Loongson 3 can prefetch the data in memory or IO into the secondary cache.

4.1.2 Replacement strategy

Both the first-level cache and the second-level cache use random replacement algorithms. But the secondary cache provides a locking mechanism. Send via configuration lock windc
Memory to ensure that up to 4 locked areas are not replaced with secondary caches (for specific configuration methods, please refer to "Godson 3A1000 Processing
Chapter 4 on the user manual of the

4.1.3 Cache parameters

Table 4-1 gives some parameters of the three caches

Table 4-1 Cache parameters

parameter	Instruction **Cache**	Data **Cache**	Secondary **Cache**
Cache size	64KB	64KB	1MB (total 4MB)
Degree of association	4 groups connected	4 groups connected	4 groups connected
Replacement strategy	Random method	Random method	Random method (lockable)
Line size	32 bytes	32 bytes	32 bytes
Index	Virtual address 13: 5 bits	Virtual address 13: 5 bits	Physical address 17: 5 bit 1
Tag	Physical address 47: 12 bits	Physical address 47: 12 bits	Physical address 47: 12 bits
Write strategy	Not writable	Write-back	Write-back
Reading strategy	Non-blocking (2 simultaneous)	Non-blocking (24 simultaneous)	Non-blocking (8 simultaneous)
Reading order	Keyword first	Keyword first	Keyword first
Write order	Not writable	Sequential	
Verification method	Parity check	ECC check	ECC check

4.2 Level 1 instruction cache

The size of the first-level instruction cache is 64KB, and the four-way group structure is used. Cache block size (also commonly referred to as Cache
Line) is 32 bytes and can store 8 instructions. Since Godson 3 uses a 128-bit read path, it can be taken every clock cycle
Four instructions are sent to the superscalar scheduling unit.

The first-level instruction cache implements parity checking. When a parity error occurs when reading the first-level instruction cache, the hardware will automatically respond acco
Will be invalidated and the correct value will be obtained from the secondary cache. The entire process does not require software intervention.

1 Which one of the four secondary cache modules a physical address falls on is determined by the routing configuration register of the crossbar.

66

Page 69

Godson 3A1000 Processor User Manual • Next

4.2.1 Organization of instruction cache

Figure 4-1The organizational structure of the first-level instruction cache is given. The Cache uses a four-way group linked mapping method, where each group includes
512 index entries. Select the corresponding tags (Tag) and data (Data) according to the index (Index). After reading the tag from the cache, it is used

Compare with the translated part of the virtual address to determine the group containing the correct data.

When the first-level instruction Cache is indexed, the four groups will return their corresponding Cache line, the Cache line size is 32 bytes,

The cache line uses 34 bits as a flag and 1 bit as a valid bit.

Figure 4-1 Organization of instruction cache

4.2.2 Access to the instruction cache

Loongson No. 3 instruction Cache adopts a four-way link structure of virtual address index and physical address mark. As shown in Figure 4-2, the address

The lower 14 bits are used as the index of the instruction cache. 13: 5 bits are used to index 512 items. Each item contains four 64-bit

, Use 4: 3 bits to select among these four double words.

When indexing the Cache, the Data and corresponding physical address tags in the four blocks are taken from the Cache.

Through instruction TLB (Instruction Translation Look-aside Buffer, referred to as ITLB) for conversion, the converted address and the retrieved

The tags in the four groups are compared, and if there is a tag matching it, the data in the group is used. This is called "one at a time

Level Cache Hit (Hit) ". If none of the four groups of tags match it, then abort the operation and start accessing the Level 2 Cache. This is

It is called "level 1 cache failure (miss)".

Figure 4-2 Instruction Cache access

4.3 Level 1 Data Cache

The data cache has a capacity of 64KB and uses a four-way group structure. The Cache block size is 32 bytes, that is, it can store 8

word. The data cache read and write data paths are all 128 bits.

Data Cache uses virtual address index and physical address mark. The operating system needs to address page coloring that may be caused by virtual addresses

Consistency issues. The data cache is non-blocking, which means that a failure in the data cache will not cause pipeline stalls.

The write strategy adopted by the data cache is the write-back method, that is, the operation of writing data to the first-level cache will not cause the second-level cache and the main

Update. The write-back strategy reduces the amount of communication from the first-level cache to the second-level cache, thereby improving global performance. Only in the data cache

When the row is replaced, the data is written to the secondary cache.

The first-level data cache implements ECC verification. When an ECC check error occurs in reading the primary data cache, the hardware will automatically

Correct the result read by Cache and update the content in Cache to the corrected value. The entire process does not require software intervention. When reading a level

According to the Cache, when a two-bit ECC check error occurs, an exception will be left for the software to handle.

4.3.1 Organization of Data Cache

Figure 4- 3 shows the organization structure of the data cache. This is a four-way group-associated cache, which contains 512 index entries. When indexing the Cache, access Tag and Data in the four groups at the same time. Then combine the tags in the four groups with the converted physical addresses Partially compare to determine which data row hit.

When indexing the data cache, each of the four groups will return their respective cache lines. The Cache block size is 32 bytes, The cache line uses 34 bits as the physical flag address, 1 bit as the dirty bit and 2 bits as the status bit (including INV, SHD and EXC Three states). The INV state indicates that the cache line is invalid, the SHD state indicates that the cache line is readable, and the EXC state indicates that the cache Line is readable and writable.

Figure 4-3 Organization structure of data cache

4.3.2 Data Cache access

Loongson No. 3 Data Cache adopts a four-way link structure of virtual address index and physical address mark. Figure 4-4Gives a visit once How to decompose the virtual address during data cache.

Figure 4-4 Data Cache access

As shown in Figure 4-4 , the lower 14 bits of the address are used as an index into the data cache. 13 of them are used for indexing 512 items, of which Each entry includes four 64-bit double words. Use 4: 3 bits to select four double words, 2: 0 bits to select eight words of a double word A byte in the section.

Data Cache access invalid instruction (access instruction misses or write instruction hits Cache line in SHD state), then access level 2 Cache. If the secondary cache hits, the cache block retrieved from the secondary cache is sent back to the primary cache. If the secondary cache If it fails, the memory is accessed, and the secondary cache and data cache are filled with the values retrieved from within.

4.4 Secondary Cache

Loongson 3 includes 4 on-chip secondary cache modules. The capacity of each secondary cache module is 1MB, a total of 4MB. each The size of each cache line is 32 bytes. The main features of the secondary cache module include: 128-bit AXI interface, four-way group connection, 8-item Cache access queue, keyword priority, fastest 8 beats to receive read invalid request to return data, support Cache consistency through directory Protocol, it can be used for on-chip multi-core structure (it can also be directly connected with single processor IP), and the soft IP level can be configured with Small (512KB / 1MB), using a 4-way associative structure, which can be dynamically closed during operation, supports ECC check, and supports DMA consistency Read and write and prefetch read, support 16 kinds of secondary cache hashes, support secondary cache by window lock, to ensure that read data returns atomicity.

The secondary cache also maintains a directory of each cache line to record whether each primary cache contains a backup of the cache line.

The write strategy adopted by the secondary cache is the write-back method. The write-back strategy reduces the amount of communication on the bus, thereby improving the overall perfc

Only when the secondary cache line is replaced, the data will be written to the memory.

The secondary cache implements ECC verification. When a one-bit ECC check error occurs in reading the secondary cache, the hardware will automatically correct the cache

Read the result and update the content in the Cache to the corrected value. The entire process does not require software intervention. When reading the secondary data cache

When a two-bit ECC check error occurs, an exception will be left for the software to handle.

4.4.1 Organization of secondary cache

The secondary cache is a hybrid cache, which includes both instructions and data. The second-level Cache module supports the Cache consistency protocol.

All the on-chip secondary caches in Loongson 3 are addressed uniformly, and each secondary cache block has a fixed home node. According to Cache one

Consistent requirements, the second-level cache of Loongson 3 has two roles: for the first-level cache, it is home, and for memory

It's Cache. When accessing the secondary cache, four groups of data and tags are accessed at the same time.

The high-order part of the address is compared to determine whether the data still resides in the cache.

Each Cache line contains a 32-byte data, 31-bit physical address flag, 1 Cache status bit (indicating the corresponding

Whether the cache line is valid in the second-level cache, a 1-bit directory status bit (indicating whether the corresponding Cache block is in a first-level cache

In the exclusive or shared state) and 1 W bit (indicating whether the line has been written).

4.4.2 Secondary Cache access

Only when the primary cache fails, the secondary cache is accessed. The secondary cache uses the physical address index to physically

Address mark. As shown in Figure 4-5 , the lower address is used to index the secondary cache. The four groups will return their corresponding Cache line.

16: 5 bits are used as the index for the secondary cache. Each indexed item contains four 64-bit double-word data. Use 4: 3 bits in 4 pairs

Word selection. Bits 2: 0 are used to select 8 bytes in a double word.

Figure 4-5 Secondary Cache access

4.5 Cache algorithm and Cache consistency properties

Loongson 3 implements the Cache algorithm and Cache consistency attributes shown in Table 4-2 .

Table 4-2 Consistency attributes of Godson 3 Cache

Attribute classification	Consistent code
Keep	0
Keep	1
Uncached	2
Cacheable coherent	3
Keep	4
Keep	5
Keep	6
Uncached Accelerated	7

70

4.5.1 Uncached (Uncached, consistency code 2)

If a page uses a non-cache algorithm, then for any Load or Store operation anywhere on the page, the processor

Directly transmit a double word, partial double word, word, partial word read or write request to the main memory without passing any level of cache Non-high speed

The caching algorithm is implemented in a blocking manner.

4.5.2 **Coherent** cache (Cacheable coherent, coherent code 3)

A row with this attribute can reside in the cache, and the corresponding storage and access operations only access the first-level cache. When the first level

When the cache fails, the processor checks the secondary cache to see if it contains the requested address. If the secondary cache hits, then

Data is filled in the secondary cache. If the secondary cache does not hit, then take the data from the main memory and write it to the secondary cache and the primary cache

Level Cache.

Since there are multiple processor cores and IO devices in Loongson 3 that can access the main memory, Loongson 3 hardware implements Cache 1

Consistent agreement, no need to use the Cache instruction through software to actively maintain the consistency of the Cache.

4.5.3 Uncached Accelerated (Consistency code 7)

The non-cache acceleration attribute is used to optimize a series of sequences of the same type that are completed in a continuous address space

Uncached storage operation. The optimization method is to collect the storage operation of this attribute by setting a buffer. As long as the buffer is not full,

You can store the data of these operations in the buffer. The buffer is the same size as a cache line. Store data in the buffer

Is the same as storing in Cache. When the buffer is full, block writing begins. In the process of collecting sequential storage instructions, if

If there are other types of Uncached storage instructions inserted, the collection will be terminated, and the data stored in the buffer will be output in byte write mode.

The non-cache acceleration attribute can speed up sequential Uncached access, which is suitable for fast output access to display device storage.

4.6 Cache consistency

Loongson 3 achieves directory-based cache consistency, and the hardware guarantees first-level instruction cache, first-level data cache, and second-level cache

The consistency of data between Cache, memory, and IO devices from HT does not require software to use Cache instructions to force Cache refresh.

Each cache line in Godson 3 has a fixed host secondary cache module. The directory information of the Cache line is on the host two

Maintained in the level Cache module. The directory uses 32-bit bit vectors to record the first-level cache (including a

Level instruction cache and level one data cache). Each level 1 cache block has three possible states: INV (inactive state), SHD (total

Shared state, readable) and EXC (exclusive state, readable and writable). The transition of the three states is shown in Figure 4-6. When reading instructions or fetching instructions

When the first-level cache fails, the processor core sends a Reqread request to the second-level cache module, which is sent back to the second-level cache module

After the Repread response, the first-level cache of the processor core obtains a cache backup of the SHD state; when the write instruction occurs, the first-level cache

When the Cache fails, the processor core sends a Reqwrite request to the secondary Cache module, and gets the Repwrite returned by the secondary Cache module

After answering, the first-level cache of the processor core obtains an EXC state cache backup;

When changing, the secondary cache module is written back through Reqreplace, and the secondary cache module informs the processor core to replace through the Repreplace response

The request has been processed. The second-level cache module can invalidate a first-level SHD state by sending a Reqinv request to the processor core

Cache backup, the processor core changes the first-level Cache backup to INV state and responds to the second-level Cache module through Repinv;

The Cache module can write back an EXC level 1 Cache backup by sending a Reqwtbk request to the processor core, the processor

71

The core changes the primary cache backup to the EXC state and responds to the secondary cache module through Repwtbk; the secondary cache module can communicate

By sending a Reqinvwtbk request to the processor core to write it back and invalidate an EXC level primary cache backup, the processor core will

The level Cache backup becomes the INV state and responds to the level two Cache module through Repinvwtbk.

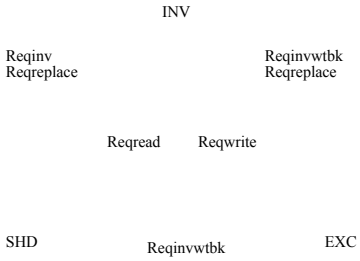


Figure 4-6 Loongson No. 3 cache state transition

Godson 3A1000 Processor User Manual • Next

5 memory management

Godson GS464 processor core provides a full-featured memory management unit (MMU), which uses on-chip TLB to implement virtual Conversion of the intended address to the physical address.

This chapter describes the virtual address and physical address space of the processor core, the conversion of virtual address to physical address, TLB These conversion operations, cache, and system control coprocessor (CP0) registers that provide software interfaces for TLBs.

5.1 Quick Lookup Table TLB

Mapping virtual addresses to physical addresses is usually implemented by TLB (there are also virtual address translations that do not go through TLB, such as The CKSEG0 and CKSEG1 kernel address space segments (see Figure 5-5) do not perform page mapping, and the physical addresses are virtual The address is obtained by subtracting a base address.). The first level of TLB is JTLB, which is also used as data TLB. In addition, Godson GS464 The processor core contains independent instruction TLB to ease competition for JTLB.

5.1.1 JTLB

In order to quickly map the virtual address to the physical address, the Godson GS464 processor core uses a larger, fully connected

The TLB and JTLB of the mapping mechanism are used for address mapping of instructions and data, and their process numbers and virtual addresses are used for indexing.

JTLB is organized in pairs of odd / even entries, mapping virtual address space and address space identifiers to 256T physical address space.

By default, JTLB has 64 pairs of odd / even entries, allowing 128 pages to be mapped.

There are two mechanisms to assist in controlling the size of the mapping space and the replacement strategy for different areas of memory.

First, the page size can be 4KB to 16MB, but it must be increased by 4 times. The CP0 register PageMask is used to record images

The size of the projected page, and this record is loaded into the TLB while writing a new entry. Godson GS464 processor core can be in the same

Supports pages of different sizes at runtime, allowing the operating system to generate specific-purpose mappings: for example, frame buffering in video codec processing

The area can be used for memory mapping with only one entry.

Second, the Godson GS464 processor core can use the random replacement strategy to select the TLB to be replaced when the TLB is missing

Table entry. The operating system can also reside a certain number of pages in the TLB without being randomly replaced, this mechanism is beneficial

To enable the operating system to improve performance and avoid deadlock. This mechanism also makes it easier for real-time systems to provide specific entries for a key software.

In addition, for each page, JTLB also maintains the Cache consistency attribute of the page, each page has a specific bit to mark: no

After Cache (Uncached), non-uniform Cache (Cacheable Noncoherent), or non-Cache acceleration (Uncached

Accelerated).

5.1.2 Instruction TLB

Loongson GS464 processor core instruction TLB (ITLB) has 16 entries, which minimizes the capacity of JTLB and passes a

The large associative array shortens the time-critical path during mapping and reduces power. Each ITLB entry can only map one page, page size

It is specified by the PageMask register. ITLB instruction address mapping and data address mapping can be performed in parallel, thereby improving performance.

When the entry in ITLB fails, find the corresponding entry from JTLB and randomly select an ITLB entry for replacement. ITLB 's

The operation is completely transparent to the user. The processor guarantees the consistency of ITLB and JTLB. When the JTLB is modified using core state instructions, ITLB

Will be automatically emptied.

5.1.3 Hits and failures

If the virtual address is the same as the virtual address of an entry in the TLB (that is, the TLB hits), the physical page number is taken from the TLB

Out and connected with the offset to form a physical address.

If the virtual address is not consistent with the virtual address of any entry in the TLB (that is, the TLB is invalid), the CPU generates an exception,

And the software refills the TLB according to the page table stored in the memory. The software can rewrite the specified TLB entry, or use the hardware

The provided mechanism rewrites any TLB entry.

5.1.4 Multiple hits

The Godson GS464 processor checks that the virtual address in the TLB is not only consistent with the virtual address of an entry and does not provide any detection

And disabling mechanism, this is not like the design of early MIPS processors. Multiple hits do not physically destroy the TLB, so multiple hits

The detection mechanism in is unnecessary. However, the situation of multiple hits is not defined, so the software should control not to allow multiple hits

The situation occurs.

5.2 Processor mode

Loongson GS464 processor core has three working modes, but unlike other MIPS processors, Loongson GS464 processor core only

Support one address mode, one instruction set mode and one end mode.

5.2.1 Working mode of the processor

The processor priority of the following three modes is reduced in order:

- Kernel mode (highest system priority): In this mode, the processor can access and change any register, the operating system is the most

  The inner kernel runs in kernel mode;

- Management mode: The priority of the processor is reduced, and some less critical parts of the operating system run in this mode;
- User mode (lowest system priority): This mode prevents different users from interfering with each other.

The three modes are switched by the operating system (in kernel mode) setting the corresponding bit in the KSU field of the status register. Be out

When an error (ERL bit is set) or an exception (EXL bit is set) occurs, the processor is forced to switch to kernel mode. Table 5-1

It lists the setting of KSU, EXL and ERL when switching between the three modes. Empty entries can be ignored.

Table 5-1 Working modes of the processor

KSU	ERL	EXL	description
4: 3	2	1	
10	0	0	User mode
01	0	0	Management Mode
00	0	0	Kernel mode
	0	1	Exception level
	1		Error level

5.2.2 Address mode

Godson GS464 processor core only supports 64-bit virtual address mode, and the hardware is guaranteed to be compatible with 32-bit address mode.

5.2.3 Instruction set mode

Loongson GS464 processor core implements a complete MIPS64R2 instruction set, and also adds some integer and floating-point instructions.

See Appendix A and Appendix B for additional instructions.

5.2.4 End mode

Godson GS464 processor core only works in small-endian mode.

5.3 Address space

This section describes the virtual address space, physical address space, and the method of performing virtual and real address translation through TLB.

5.3.1 Virtual address space

Loongson GS464 processor core has three virtual address spaces: user address space, management address space and kernel address space, each

The space is 64 bit ome discontinuous address space segments, the largest segment is 256T (2 48) bytes.

5.3 Section 5) describe these three address spaces.

5.3.2 Physical address space

By using 48-bit addresses, the physical address space of the processor is 256T (2 48) bytes. The following section will detail the transfer of virtual and real addresses

Change the method.

5.3.3 Virtual and real address translation

When performing virtual-real address translation, first compare the virtual address given by the processor with the virtual address stored in the TLB. When the virtual page number (

VPN domain equal to a TLB entry, and if either of the following two conditions holds:

- The Global bit of the TLB entry is 1
- The ASID fields of the two virtual addresses are the same.

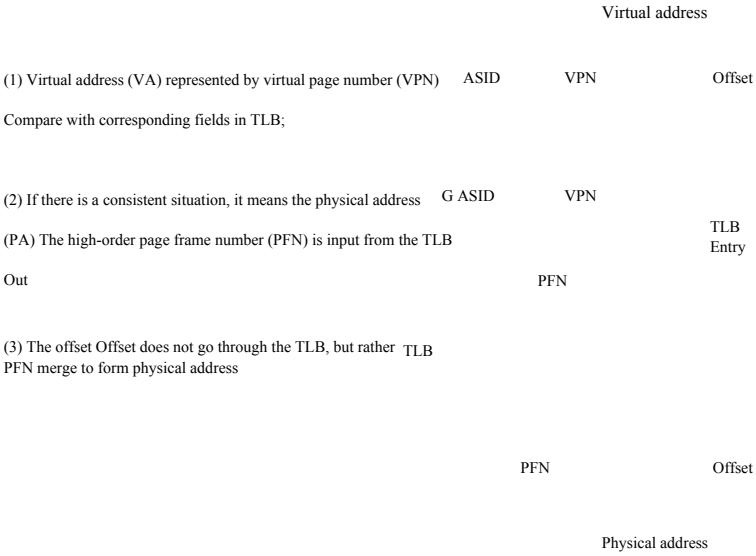
TLB hit. If the above conditions are not met, then the CPU will generate a TLB failure exception, so that the software can

Fill in the TLB again in the page table stored in.

If the TLB hits, the physical page number will be taken from the TLB and merged with the offset within the page Offset to form a physical address.

In-page offset Offset does not go through the TLB during the conversion of virtual and real addresses.

75



Reduced the frequency of TLB refresh when context switching. The ASID is stored in the CP0 EntryHi register. Global position (G) in

In the corresponding TLB entry.

Figure 5- 2 shows the virtual and real address translation process in 64-bit mode, this figure shows the maximum page 16MB and the minimum page 4KB

Happening.

The upper part of the figure shows the case where the page size is 4K bytes. The offset within the page takes 12 bits in the virtual address.

The remaining 36 bits in the proposed address are virtual page number VPNs, which are used to index 64G page table entries;

The lower part of the figure shows the case where the page size is 16M bytes, and the offset within the page takes 24 bits in the virtual address.

The remaining 24 bits in the virtual address are the virtual page number VPN, which is used to index 16M page table entries.

76

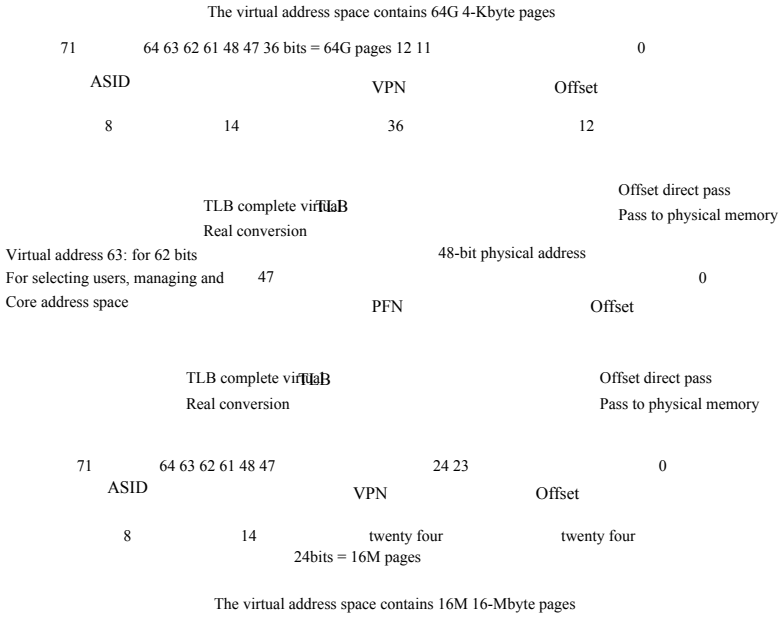


Figure 5-2 64-bit mode virtual address translation

5.3.4 User address space

In user mode, there is only a single, unified virtual address space called User Segment, whose size is

The user segment starts at address 0, and the currently active user process resides in this segment (XUSEG). In different modes, TLB The mapping process for XUSEG segments is the same, and controls whether the cache can be accessed.

When the value of the Status register of the processor satisfies three conditions: KSU = 10 2 , EXL = 0, the ERL = 0, the processor operates at In user mode.

77

Godson 3A1000 Processor User Manual • Next

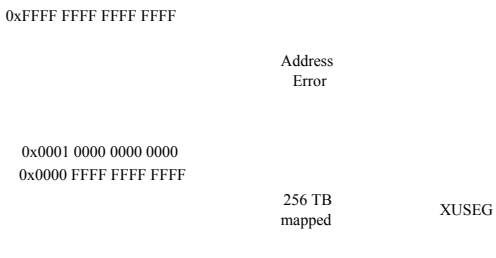


Figure 5-3 Overview of user virtual address space in user mode

In all available user modes, bits 63 to 48 of the virtual address must be 0, and access to any one of bits 63 to 40

Addresses with bits not all 0 will result in an address error exception. If the TLB in the XUSEG address segment is missing, use the XTLB refill vector. Dragon

The XTLB refill vector of the core GS464 processor core has the same exception entry address as the TLB refill vector in 32-bit mode.

5.3.5 Manage address space

The management mode is designed for a layered operating system. In a hierarchical operating system, the real kernel runs in kernel mode

In mode, the rest of the operating system runs in management mode. The management address space provides codes and data for program access in management mode

According to space. The missing TLB that manages the address space is handled by the XTLB refill processor.

Both management mode and kernel mode can access the management address space.

When the value of the Status register of the processor satisfies three conditions: KSU = 01 2 , EXL = 0, the ERL = 0, the processor operates at

Under management mode. Figure 5-4 shows the overview of users and management address space in management mode.

78

Figure 5-4 User space and management space in management mode

- 64-bit management mode, user address space (XSUSEG)

In management mode, when accessing the user address space and the highest two bits of the 64-bit address (bits 63 and 62) is 00 2 ,
The program uses a virtual address space named XSUSEG, XSUSEG covers the entire current user address space 2 48 (1T)
byte. At this time, the virtual address is expanded, and the 8-bit ASID field is added to form a unique virtual address in the system. This address space starts from
0x0000 0000 0000 0000 starts and ends at 0x0000 FFFF FFFF FFFF.

- 64-bit management mode, current management address space (XSSEG)

In the management mode, when the highest two bits of the 64-bit address (bits 63 and 62) are 01 2 , the program uses a name XSSEG
'S currently managed virtual address space. At this time, the virtual address is expanded, and the 8-bit ASID field is added to form a unique virtual in the system
address. This address space starts at 0x4000 0000 0000 0000 and ends at 0x4000 FFFF FFFF FFFF.

- 64-bit management mode, independent management address space (CSSEG)

In management mode, when the highest two bits of the 64-bit address (bits 63 and 62) are 11 2 , the program uses a name CSSEG
Independently manages the virtual address space. Addressing in CSSEG is compatible with addressing in SSEG in 32-bit mode. Virtual
The address is expanded, and the 8-bit ASID field is added to form a unique virtual address in the system. This address space starts from 0xFFFF FFFF C000
0000 starts and ends at 0xFFFF FFFF DFFF FFFF.

Godson 3A1000 Processor User Manual • Next

5.3.6 Kernel address space

When the value of the processor's Status register meets the following conditions: KSU = 00 2 or EXL = 1 or ERL = 1, the processor works in
In nuclear mode.

Whenever the processor detects an exception, it enters kernel mode and remains until the execution of the exception return instruction (ERET). ERET
The instruction restores the processor to the mode it was in before the exception occurred.

According to different high-order virtual addresses, the kernel-mode virtual address space is divided into different areas, as shown in Figure 5-5 .

- 64-bit kernel mode, user address space (XKUSEG)

In kernel mode, when accessing user space and the highest two digits of a 64-bit virtual address is 00 2 , the program uses a name
XKUSEG's virtual address space, XKUSEG covers the current user address space. At this time, the virtual address is expanded, plus 8 bits
The ASID field forms a unique virtual address in the system.

- 64-bit kernel mode, current management address space (XKSSEG)

In kernel mode, when accessing the management space and the highest two bits of the 64-bit address is 01 2 , the program uses a name
XKSSEG's virtual address space, XKSSEG is the currently managed virtual address space. At this time, the virtual address is expanded, plus 8 bits
The ASID field forms a unique virtual address in the system.

- 64-bit kernel mode, physical address space (XKPHY)

In kernel mode, when the highest two bits of a 64-bit address are 10 2 , the program uses a virtual address space named XKPHY,
XKPHY is a collection of eight 2 48- byte kernel physical address spaces. Access any memory card whose addresses 58 to 48 are not 0
Yuan will cause an address error. The access to XKPHY does not perform address conversion through TLB, but the 47th to 0th of the virtual address
Bit as a physical address. The 61st to 59th bits of the virtual address control whether to pass the Cache and Cache consistency attributes, as shown in Table 3-2
The C bit value of the described TLB page has the same meaning.

- 64-bit kernel mode, kernel address space (XKSEG)

In kernel mode, when the top two bits of 64-bit address is 11 2 , the program address space of one of the following two:

The kernel virtual address space XKSEG, at this time the virtual address is extended, plus the 8-bit ASID field, forming a system
The only virtual address in the system;

The four 32-bit cores are compatible with the address space, as detailed in the next section.

- 64-bit kernel mode, compatible with address space (CKSEG1: 0, CKSSEG, CKSEG3)

In kernel mode, when the highest two bits of a 64-bit address are 11 2 and all bits from bits 61 to 31 of the virtual address are equal to 1,
One of the following four 512M byte address spaces used by the program, which one is determined according to bits 30 and 29:

80

Figure 5-5 Overview of user, management, and kernel address space in kernel mode

CKSEG0: The 64-bit virtual address space does not go through the TLB and is compatible with KSEG0 in 32-bit mode. Config The K0 field of the register controls whether to pass the consistency attribute of Cache and Cache,

CKSEG1: The 64-bit virtual address space does not go through TLB or Cache, and KSEG1 in 32-bit mode compatible.

CKSSEG: The 64-bit virtual address space is the currently managed virtual address space, and KSSEG in 32-bit mode

81

compatible.

CKSEG3: The 64-bit virtual address space is the kernel virtual address space, compatible with KSEG3 in 32-bit mode.

5.4 System Control Coprocessor

The system control coprocessor (CP0) is responsible for supporting storage management, virtual and real address translation, exception handling, and some privileged operations. Dr

The core GS464 processor core has 26 CP0 registers and a 64-item TLB, each register has a unique register number. below

The chapter will give an overview of the registers related to memory management.

5.4.1 Format of TLB entries

Figure 5-6 shows the format of the TLB entry, each field in the entry is in EntryHi, EntryLo0, EntryLo1, PageMask register

There are corresponding fields in.

The format of EntryHi, EntryLo0, EntryLo1, and PageMask registers and TLB entries are similar. The or

The TLB entry has a Global field (G bit), which is not in the EntryHi register, but appears as a reserved field. F : 5-9

The fields of the TLB entry in Figure 5-6 are shown separately.

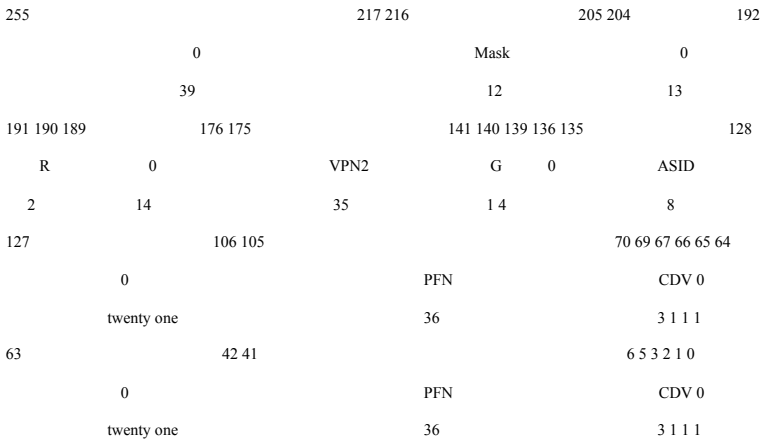
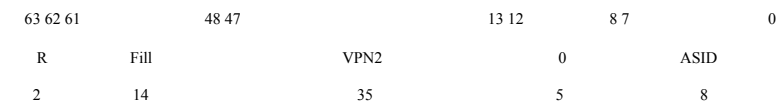


Figure 5-6 TLB entries

Mask … page comparison mask

0 …… .reserved. Write must be 0, and return 0 when reading.

Figure 5-7 PageMask Register



VPN2 …. Virtual page number divided by 2 (map 2 pages).

ASID …. Address space ID field. An 8-bit field used to allow multiple processes to share the TLB; for the same virtual page number as other processes, each Processes have different mappings.

R …… area. (00-> User, 01-> Management, 11-> Core) The user matches the 63:62 bits of the virtual address.

Fill ..... Keep. Read as 0, write ignore.

0 ... reserved. Write must be 0, and return 0 when reading.

Figure 5-8 EntryHi Register

63	42 41	6 5	3 2 1 0
0	PFN	CDVG	
twenty two	36	3 1 1 1	
63	42 41	6 5	3 2 1 0
0	PFN	CDVG	
twenty two	36	3 1 1 1	

PFN … page frame number; high order of physical address.

C ..... Specify the consistency attributes of the TLB page; see Table 3-2.

D ..... Dirty bit. If the bit value is set to 1, the corresponding page is marked as dirty, so it can be written. This bit is actually a write-protected bit, software is available This bit protects data changes.

V ..... Effective bit. The setting of this bit indicates that the TLB entry is valid; otherwise, TLBL / TLBS will be invalid.

G ..... Global position. If the corresponding bits in Lo0 and Lo1 are both set to 1, the processor ignores the ASID during TLB lookup.

0 …… .. reserved. Write must be 0, and return 0 when reading.

Figure 5-9 EntryLo0 and EntryLo1 registers

The TLB page consistency attribute bit (C) specifies whether to pass the cache when accessing the page.

Cache consistency properties. Table 5-2Indicates the Cache consistency attribute corresponding to the C bit.

Table 5-2 Value of C bit on TLB page

C (5: 3) value	**Cache** consistency properties
0	Keep
1	Keep
2	Uncached
3	Cacheable Noncoherent
4	Keep
5	Keep
6	Keep
7	Uncached Accelerated

5.4.2 CP0 register

Table 5- 3 lists the CP0 registers related to memory management. Chapter 1 provides a complete description of the CP0 registers.

Table 5-3 CP0 registers related to memory management

Register number	Register name
0	Index
1	Random
2	EntryLo0
3	EntryLo1

5	PageMask
6	Wired
10	EntryHi
15	PRID
16	Config
17	LLAddr
28	TagLo
29	TagHi

5.4.3 Conversion process from virtual address to physical address

During virtual address to physical address translation, the CPU converts the 8-bit ASID of the virtual address (if the global bit G is not set) and the TLB entry

The ASIDs are compared to see if they match. While comparing ASID, the virtual address needs to be changed according to the value of the page mask (PageMask)

The upper 15 ~ 27 digits of the data are compared with the virtual page number of the TLB item. If there is a TLB entry match, remove the physical ground from the matching TLB entry

Address and access control bits (C, D and V). For a valid address translation, the V bit of the matching TLB entry must be set, but

The value of the V bit is not considered in the match comparison. Figure 5-10 shows the TLB address translation process.

Figure 5-10 TLB address translation

5.4.4 TLB failure

If no TLB entry matches the virtual address, a TLB miss exception is raised. If the access control bits (D and V) indicate

The access is not legal and raises a TLB modification or TLB invalid exception. If the C bit is equal to 011 2 , the physical address retrieved

Godson 3A1000 Processor User Manual • Next

Access memory through Cache, otherwise not through Cache.

5.4.5 TLB instruction

Table 5-4 lists all the instructions provided by the CPU for TLB operation.

Table 5-4 TLB instructions

Opcode	Instruction description
TLBP	Search for matches in TLB
TLBR	Read indexed TLB entries
TLBWI	Indexed TLB entry
TLBWR	Write random TLB entries

5.4.6 Code examples

The first example is how to configure a TLB entry to map a pair of 4KB pages. Most real-time system kernels do this, this kind of

A simple kernel MMU is only used for memory protection, so static mapping is sufficient, all TLBs in all statically mapped systems

Exceptions are treated as error conditions (not accessible).

```
1.  mtc0 r0, C0_WIRED                    # make all entries available to random replacement
2.  li r2, (vpn2 << 13) | (asid & 0xff);
3.  mtc0 r2, C0_ENHI                     # set the virtual address
4.  li r2, (epfn << 6) | (coherency << 3) | (Dirty << 2) | Valid << 1 | Global)
5.  mtc0 r2, C0_ENLO0 # set the physical address for the even page
6.  li r2, (opfn << 6) | (coherency << 3) | (Dirty << 2) | Valid << 1 | Global)
7.  mtc0 r2, C0_ENLO1 # set the physical address for the odd page
8.  li r2, 0                             # set the page size to 4KB
9.  mtc0 r2, C0_PAGEMASK
10. li r2, index_of_some_entry # needed for tlbwi only
11. mtc0 r2, C0_INDEX                    # needed for tlbwi only
tlbwr                                    # or tlbwi
```

A complete virtual storage operating system (such as UNIX), using MMU for memory protection, and main storage and mass storage

Form feed of the device. This mechanism allows programs to access larger storage devices than just the physically allocated space of the system. This

The mechanism that relies on request paging requires dynamic page mapping. Dynamic mapping is implemented through a series of different types of MMU exceptions, TLB refill

It is the most common exception in such systems. The following is a possible TLB refill exception control.

```
12. refill_exception:
    13. mfc0 k0, C0_CONTEXT
```

Godson 3A1000 Processor User Manual • Next

```
14. sra k0, k0,1                # index into the page table
15. lw k1,0 (k0)                # read page table
16. lw k0,4 (k0)
17. sll k1, k1,6
18. srl k1, k1,6
19. mtc0 k1, C0_TLBLO0
20. sll k0, k0,6
21. srl k0, k0,6
22. mtc0 k0, C0_TLBLO1
23. tlbwr                       # write a random entry
eret
```

This exception control process is very simple, because its frequent execution will affect the system performance, which is the TLB refill exception allocation alone
The reason for the exception vector. This code assumes that the required mapping has been established in the main memory page table. If not established
Then, TLB invalid exception will occur after ERET instruction. TLB failure exceptions rarely occur, which is beneficial because it must
The desired mapping must be calculated, and some page tables may need to be read from the back-up memory. TLB modification exception is used to implement read-only pages and tags
Remember the process to clear the page where the code needs to be modified. In order to protect different processes and users from mutual interference, the virtual storage operating syster
Execute the user program in user mode. The following example shows how to enter user mode from kernel mode.

```
24. mtc0 r10, C0_EPC                    # assume r10 holds desired usermode address
25. mfc0 r1, C0_SR                      # get current value of Status register
26. and r1, r1, ~ (SR_KSU || SR_ERL)    # clear KSU and ERL field
27. or r1, r1, (KSU_USERMODE || SR_EXL) # set usermode and EXL bit
28. mtc0 r1, C0_SR
eret                                    # jump to user mode
```

5.5 Physical address space distribution

The address space of Godson 3 is evenly distributed to each node according to the high-order address. The upper 4 bits [47:44] of the 48-bit address correspond to the address space
The node number where the time is located, each node has a fixed 44-bit address space. The 44-bit address space within the node is further divided
There are 8 41-bit address spaces. The 41-bit space is mainly used because one port may be connected to two HT controllers.
HT requires a 40-bit address space.

Godson 3A1000 Processor User Manual • Next

6 processor exception

This chapter introduces the exception of Godson GS464 processor core, including: the generation and return of exception, the location and location of exception vector
Supported exception types. Among them, the types of exceptions supported by each category, the introduction includes the reasons for exceptions, treatment and services.

6.1 Generation and return of exceptions

When the processor starts to handle an exception, the EXL bit of the status register is set to 1, which means that the system is running

In kernel mode. After saving the appropriate on-site state, the exception handler usually sets the KSU field of the status register

Set to kernel mode and return the EXL position to 0 at the same time. When the on-site state is restored and executed again, the processing procedure is

The KSU field will be restored to the previous value, and the EXL bit will be set to 1.

Returning from an exception will also set the EXL position to 0.

6.2 Exception vector position

Cold reset, soft reset, and non-maskable interrupt (NMI) exception vector addresses are dedicated reset exception vector addresses

0xFFFFFFFFBFC00000, this address is neither accessed through Cache nor address mapping. In addition, EJTAG

The entry of the debug interrupt is selected according to whether the ProbeTrap bit in its control register is 0 or 1.

0xFFFFFFFFBFC00480 and 0xFFFFFFFFF200200. All other exception vector addresses are in the form of base addresses

Add the vector offset. When the BEV bit in the status register is 0, the user can define the base address of the exception vector, see Table 6-1 for details

Exception vector base address.

Table 6-1 Exception vector base address

exception	BEV = 0	BEV = 1
Reset, Soft Reset, NMI	0xFFFFFFFF BFC00000	
EJTAG Debug (ProbEn = 0)	0xFFFFFFFF BFC00480	
EJTAG Debug (ProbEn = 1)	0xFFFFFFFF FF200200	
Cache Error	0xFFFFFFFF || EBase31..30 || 1 || EBase28..12 || 0x000	0xFFFFFFFF BFC00300
Others	0xFFFFFFFF || EBase31..12 || 0x000	0xFFFFFFFF BFC00200

88

Page 91

Table 6-2 lists the offset of the exception vector in the Godson GS464 processor core.

Table 6-2 Exception vector offset

exception	Exception vector offset
TLB Refill, EXL = 0	0x000
XTLB Refill, EXL = 0	0x080
Cache error	0x100
Other common exceptions	0x180
Interrupted and CauseIV = 1	0x200
Reset, Soft Reset, NMI	None (use base address)

For external interrupts (including clock and performance counter interrupts), the traditional approach is to use a shared exception entry, which is

Responsible for distribution to corresponding services. Godson GS464 processor core supports Vectored Interrupt mode (Vectored Interrupt), the mode

It is selected by the IV bit of Cause register. In vectored interrupt mode, the interrupt priority level decreases from IP7 to IP0 and

There are special exception entrances. The VS field of the IntCtl register controls the amount of space occupied by these exception handling codes, each

The entry offset corresponding to the interrupt can be calculated by the following formula (where the vector number starts from zero):

Vector interrupt offset = 0x200 + vector number * IntCtlVS

6.3 Exception priority

The remainder of this chapter will introduce the exceptions in the order of priority given in Table 6-3

(For example, TLB exceptions and command / data exceptions are introduced together for convenience). When an instruction generates more than one instance at the same time

Outside, only the highest priority exception is reported to the processor. Some exceptions are not caused by the instructions being executed at the time

, And some exceptions may be postponed. For more details, please see the separate introduction of each exception in this chapter.

Table 6-3 Exception priority

Exception priority
Cold reset (highest priority)
Non-maskable interrupt (NMI)
Incorrect address-fetch instructions
TLB refilling-fetching instructions
TLB is invalid
Cache error-fetch instructions

Bus error-fetch instructions
Integer overflow, traps, system calls, breakpoints, reserved instructions, coprocessor not available, floating point exception
EJTAG interrupt
Address error-data access
TLB refill-data access
TLB is invalid-data access
TLB modification-write data
EJTAG data breakpoint
Cache error-data access
Bus error-data access
Interrupt (lowest priority)

In general, the exceptions introduced in the following sections are first handled by hardware and then serviced by software.

6.4 Cold reset exception

the reason

When the system is first powered on or cold reset, a cold reset exception is generated. This exception cannot be masked.

deal with

The CPU provides a special interrupt vector for this exception:

- 0xBFC0 0000 in 32-bit mode
- 0xFFFF FFFF BFC0 0000 in 64-bit mode

The cold reset vector address belongs to the CPU address space that does not require address mapping and does not access data through the cache.

This exception is not necessary to initialize TLB or Cache. This also means that even if Cache and TLB are in an uncertain state,

The processor can also fetch and execute instructions.

When an exception occurs, the contents of all registers in the CPU are undefined, except for the following register fields:

● The initial value of the Status register is 0x30c000e4, the SR bit is cleared to 0, and the ERL bit and BEV bit are Set to 1.

● The initial value of the Config register is 0x80034482.

● Random (Random) register is initialized to its maximum value.

● The Wired register is initialized to 0.

● The ErroEPC register is initialized to the value of PC.

90

Godson 3A1000 Processor User Manual • Next

● The Event bit of the Performance Count register is initialized to 0.

● All breakpoints and external interrupts are cleared.

service

Cold reset exception services include:

● Initialize all processor registers, coprocessors, cache and storage systems.

● Perform diagnostic tests.

● Boot the boot operating system.

6.5 NMI exception

the reason

NMIn is low and generates an NMI exception. This exception cannot be masked.

deal with

When an NMI exception occurs, the SR bit of the status register is set to 1 to distinguish between cold reset.

NMI exceptions can only be extracted at the boundary of the instruction. It does not abandon the state of any machine, but retains the processor 's Status is used for diagnosis. The contents of the Cause register remain unchanged, and the system jumps to the beginning of the NMI exception handler.

The NMI exception retains all register values except the following registers:

● ErrorEPC register containing PC value.

● Set the ERL bit in the status register to 1.

● Soft reset or NMI is set to 1, and cold reset is set to 0 in the status register SR bit.

● Set the BEV bit in the status register to 1.

● The PC register is reset to 0xFFFF FFFF BFC0 0000

service

The NMI exception can be used in situations other than "resetting the processor while maintaining the contents of the cache and memory." E.g, When a power failure is detected, the system can immediately and controllably shut down the system through an NMI exception.

Since the NMI exception occurred in another error exception, it is usually unlikely to continue to execute after returning from the exception 行 程序。 Line procedures.

91

Godson 3A1000 Processor User Manual • Next

6.6 Address error exception

the reason

An address error exception occurs when the following conditions are performed:

- Reference to illegal address space.
- The super user address space is referenced in user mode.
- Reference the kernel address space in user or superuser mode.
- Take (Load) or store (Store) a double word, but the double word is not aligned on the double word boundary.
- Fetch (Load, Fetch) or store (Store) a word, but the word is not aligned to the boundary of the word.
- Get or save a halfword, but the halfword is not aligned with the halfword boundary.

This exception cannot be masked.

deal with

The common exception vector is used for address error exceptions. The value of the ExcCode field of the Cause register is set to ADEL or

The ADES coded value, together with the BD bit of the EPC register and Cause register, indicates the instruction and example that caused the exception

The external cause is instruction reference, fetch operation instruction or save operation instruction.

When an exception occurs, the BadVAddr register holds a virtual address that is not properly aligned, or a protected address space

Virtual address.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's

Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

At this time, the running process that caused the exception will receive the UNIX SIGSEGV (segment violation) signal. This error

It is usually fatal to the process.

6.7 TLB exception

Three TLB exceptions may occur:

- When there is no entry in the TLB that matches the address of the mapped address space to be referenced, it will cause the TLB refill exception.
- When the virtual address reference matches an item in the TLB, but the item is marked as invalid, the TLB invalid exception occurs.
- When the virtual address reference of the write memory operation matches an item in the TLB, but the item is not marked as "dirty"

Godson 3A1000 Processor User Manual • Next

(Indicating that this item cannot be written), TLB modification exception occurs.

The following three sections describe these TLB exceptions.

Note: The selection of TLB refill vectors has been introduced earlier in this chapter. For specific chapters, see 6.8 " Exceptions to TLB Refill".

6.8 TLB refill exception

the reason

When there is no entry in the TLB that matches the reference address in the mapped address space, a TLB refill exception occurs, which is not possible Shielded.

deal with

For this exception, there are two special exception vectors in the MIPS architecture: one for the 32-bit address space The other is used for the 64-bit address space. The exception vector offset is 0x000 when the reference address is in the 32-bit address space, The exception vector offset is 0x080 when the reference address is in the 64-bit address space.

When the EXL bit in the status register is set to 0, all address references use these exception vectors. This exception Set the value of the ExcCode field in the Cause register to TLBL or TLBS encoding. This code and the EPC register Together with the BD of the Cause register, specify the instruction that caused the exception and the cause of the exception is the instruction reference, fetch operation instruction Still save the operation instruction.

When this exception occurs, the BadVAddr, Context, XContext, and EntryHi registers hold the address transfer Change the failed virtual address. The EntryHi register also holds the ASID when the conversion fails. Random registers are usually saved The legal location for placing the replaced TLB item. The content of the EntryLo register is undefined. If an exception is raised If the instruction is not in the branch delay slot, then the EPC register holds the address of the instruction that caused the exception; otherwise, The EPC register holds the address of the previous branch instruction, and the BD bit in the Cause register is set to 1.

service

To serve this exception, the contents of the Contex or XContext register are used as virtual addresses to obtain certain memory bits These locations contain the physical page address and access control bits for a pair of TLB entries. This pair of TLB items was placed EntryLo0 / EntryLo1 register; EntryHi and EntryLo registers are written to TLB.

The virtual address used to obtain the physical address and access control information may be located on a page that does not reside in the TLB on. If this happens, another TLB refill exception is allowed in the TLB refill handler to resolve it. due to The EXL bit of the Status register is set to 1, and the second TLB refill exception is passed the common exception vector.

Godson 3A1000 Processor User Manual • Next

6.9 TLB invalid exception

the reason

When a virtual address reference matches a TLB entry marked as invalid (the TLB valid bit is cleared), the TLB has no Exceptions occur. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception. The value of the ExcCode field of the Cause register is set to TLBL or TLBS, Together with the BD bit of the EPC register and Cause register, specify the instruction that caused the exception and the cause of the exception Order reference, fetch operation instruction or save operation instruction.

When this exception occurs, the BadVAddr, Context, XContext, and EntryHi registers hold the address transfer Change the failed virtual address. The EntryHi register also holds the ASID when the conversion fails. Random registers are usually saved The legal location for placing the replaced TLB item. The content of the EntryLo register is undefined.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction ’s Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

When one of the following occurs, the TLB entry is marked as invalid:

- Virtual address does not exist
- The virtual address exists, but it is not in the main memory (missing page)
- Referencing this page causes a trap (for example, maintaining the reference bit)

After servicing the cause of TLB invalid exceptions, locate the TLB item through the TLBP instruction (detect the TLB to find the match Matching item), and then replace the TLB item with the item with the valid flag.

6.10 TLB modification exception

the reason

When the virtual address reference of the write memory operation matches an item in the TLB, but the item is not marked as "dirty", so the When the item is not writable, a TLB modification exception occurs. This exception cannot be masked.

deal with

The common exception vector is used to handle this exception, and the ExcCode field value in the Cause register is set to

94

Godson 3A1000 Processor User Manual • Next

MOD.

When this exception occurs, the BadVAddr, Context, XContext, and EntryHi registers hold the address transfer Change the failed virtual address. The EntryHi register also holds the ASID when the conversion fails. The content of the EntryLo register is not definite.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

The kernel uses the failed virtual address or virtual page number to identify the corresponding access control information. The identified page may allow or Write access is not allowed; if write access is not allowed, then a write protection violation occurs.

If write access is allowed, the kernel marks the page as writable within its own data structure. TLBP instruction The index of the TLB item that must be changed is placed in the Index register. Contains physical pages and access control bits (D bit is set) The word of is taken out into the EntryLo register, and then, the EntryHi and EntryLo registers are written to the TLB.

6.11 Cache error exception

the reason

When the processor fetches instructions or accesses memory and an internal Cache verification error occurs, a Cache error exception occurs. The exception is not shield.

deal with

The Cache error exception entry with offset 0x100 is used to handle Cache error exceptions. At this time the base address of the exception entrance is set In the address segment that does not go through the cache. The value of the ExcCode field of the Cache register is set to CacheErr, together with the EPC The memory and the BD bit of the Cause register together indicate the instruction that caused the exception and whether the cause of the exception is instruction reference or access

Save operation instructions. The CacheErr register records the type of error and its position in the group's associated cache. CacheErr1 register

Record the erroneous instruction virtual address or memory physical address, see section **3.30** CacheErr and CacheErr1 register description

Narrate.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's

Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

Godson 3A1000 Processor User Manual • Next

service

Loongson GS464 processor checks the Cache error and implements the hardware self-correction function.

Exception returns.

If there is an error in the instruction cache, the cache line in error will be invalid; if there is an error in the data cache and there is only one error

The erroneous data will be automatically corrected; if the data cache is wrong and there are two errors, the operating system should consider the bit of the erroneous data block

Set the processing method.

6.12 Bus error exception

the reason

When the processor performs a data block read, update or double word / single word / half word read request, the external ERR completion response

Answer signal, bus error exception occurred. This exception cannot be masked.

deal with

The common interrupt vector is used to handle bus error exceptions. The value of the ExcCode field of the Cause register is set to IBE or

DBE, together with the BD bit of the EPC register and Cause register, indicates the instruction that caused the exception and the cause of the exception

Because it is instruction reference, fetch operation instruction, or save operation instruction.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's

Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

The physical address where the error occurred can be calculated from the information in the CP0 register.

If the value of the ExcCode field in the Cause register is set to the IBE code (indicating that it is for guidance), then

The virtual address of the instruction where the exception occurred is stored in the EPC register (if the BD bit of the Cause register is set to 1, the

Let the virtual address be the content of the EPC register plus 4).

If the value of the ExcCode field in the Cause register is set to the DBE code (meaning read or store reference),

Then the virtual address of the instruction that caused the exception is stored in the EPC register (if the BD bit of the Cause register is set to 1,

Then the virtual address of the instruction is the content of the EPC register plus 4).

Thus, reading and storing the referenced virtual address can be obtained by interpreting this instruction. And the physical address can be passed

TLBP instruction and read EntryLo register content to calculate the physical page number to obtain. Running cause the exception

The process will receive a UNIX SIGBUS (bus error) signal, which is usually fatal to the process.

6.13 Integer overflow exception

the reason

When an ADD, ADDI, SUB, DADD, DADDI, or DSUB instruction is executed, the resulting two's complement overflow

When out, an integer overflow exception occurs. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to OV encoding

value.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's

Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

The executing process that caused the exception will receive a UNIX SIGFPE / FPE_INTOVE_TRAP (floating point)

Exception / integer overflow) signal. For this process, this error is usually fatal.

6.14 Trap exception

the reason

When TGE, TGUE, TLT, TLTU, TEQ, TNE, TGEI, TGEUI, TLTI, TLTUI, TEQI,

When the TNEI instruction is executed and a conditional result is true, a trap exception occurs. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to the TR code value.

If the instruction that caused the exception is not an instruction located in the branch delay slot, then the EPC register holds the instruction 's

Address; otherwise, the EPC register holds the address of the previous branch instruction, and the BD bit of the Cause register is set to

1.

service

The executing process that caused the exception will receive a UNIX SIGFPE / FPE_INTOVE_TRAP (floating point)

97

Exception / integer overflow) signal. For this process, this error is usually fatal.

6.15 System call exception

the reason

When the SYSCALL instruction is executed, an exception to the system call occurs. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to SYS encoding value.

If the SYSCALL instruction is not in the branch delay slot, the EPC register holds the address of the instruction; otherwise, Save the address of the previous branch instruction.

If the SYSCALL instruction is in a sub-delay slot, the BD bit in the status register is set to 1, otherwise the bit is cleared 0.

service

When this exception occurs, control is transferred to the appropriate system routine. Further system call differentiation can be analyzed The Code field of the SYSCALL instruction (bits 25: 6) and the content of the instruction loaded into the address stored in the EPC register.

In order to resume the execution of the process, the contents of the EPC register must be changed so that the SYSCALL instruction will not be repeated Is executed; this can be done by adding 4 to the value of the EPC register before returning.

If the SYSCALL instruction is in the branch delay slot, a more complex algorithm is required, which is beyond the capability of this section Describe the scope.

6.16 Breakpoint exception

the reason

When a BREAK instruction is executed, a breakpoint exception occurs. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to the BP code value.

If the BREAK instruction is not in the branch delay slot, the EPC register holds the address of the instruction; otherwise, Save the address of the previous branch instruction.

If the BREAK instruction is in a sub-delay slot, the BD bit in the status register is set to 1, otherwise the bit is cleared.

service

When this exception occurs, control is transferred to the appropriate system routine. Further distinction can be analyzedBREAK refers to Command Code field (bit 25: 6), and the content of the instruction loaded into the address stored in the EPC register. If this means If it is in the branch delay slot, then the contents of the EPC register must be increased by 4 to locate the instruction.

In order to resume the execution of the process, the contents of the EPC register must be changed so that the BREAK instruction will not be Execute; this can be done by adding 4 to the value of the EPC register before returning.

If the BREAK instruction is in the branch delay slot, then in order to resume the execution of the process, this branch needs to be explained instruction.

6.17 Reserved instruction exception

the reason

When trying to execute an instruction that is not defined in MIPS64 Release2 and is not customized by Godson, retain the instruction example Happen outside. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to the RI code value.

If the reserved instruction instruction is not in the branch delay slot, the EPC register holds the address of the instruction; otherwise,

Save the address of the previous branch instruction.

service

At this time, no instruction is interpreted and executed. The process being executed that caused the exception will receive UNIX

SIGILL / ILL_RESOP_FAULT (illegal instruction / reserved operation error) signal. For this process, this error

It is often fatal.

6.18 Coprocessor unavailable exception

the reason

Attempting to execute any of the following coprocessor instructions will result in an unavailable coprocessor exception:

- The corresponding coprocessor unit (CP1 or CP2) is not marked as available.
- The CP0 unit is not marked as available, and the process executes CP0 in user or superuser mode

instruction.

Godson 3A1000 Processor User Manual • Next

This exception is unmaskable.

Loongson custom extended instruction coprocessor unavailable exception trigger conditions are as

- Custom extended memory access instruction (Table 2-15) , custom extended 64-bit multimedia ı (Table 2-18),

The custom extended floating-point fetch instruction (Table 7-5) triggers the coprocessor unavailable exception when CP2 is not marked as available.

- Custom extended floating-point fetch instruction (Table 7-5) when CP1 is not marked as available to trigger the coprocessor

With exceptions.

It should be noted that the three custom extended floating-point format conversion instructions CVT.D.LD, CVT.LD.D, and CVT.UD.D

Even if CP1 is not marked as available, the coprocessor unavailability exception will not be triggered.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to the CPU code

value. The CE field of the Cause register indicates which of the four coprocessors is referenced. If this instruction is not delayed in the branch

In the slot, the EPC register holds the address of the unusable coprocessor instruction; otherwise, the EPC register holds the previous

The address of the branch instruction.

service

There are the following situations:

If the process is authorized to access the coprocessor and the coprocessor is marked as available, then the corresponding user state is restored to

Then the coprocessor executes.

If the process is authorized to access the coprocessor, but the coprocessor does not exist or is faulty, you need to explain / simulate this

A coprocessor instruction.

If the BD bit in the Cause register is set, the branch instruction must be interpreted; then the coprocessor instruction is simulation. When the exception returns, the coprocessor instruction skipping the exception continues to execute.

If the process is not authorized to access the coprocessor, then the executing process receives UNIX SIGILL / ILL_PRIVIN_FAULT (illegal instruction / privileged instruction error) signal. This error is usually fatal.

6.19 Floating point exception

the reason

Floating-point coprocessors use floating-point exceptions. This exception is unmaskable.

deal with

The common exception vector is used to handle this exception, and the ExcCode field of the Cause register is set to FPE encoding value.

The content of the floating point control / status register indicates the cause of this exception.

service

This exception can be cleared by clearing the appropriate bit in the floating-point / status register.

6.20 EJTAG exception

When certain EJTAG related conditions are met, an EJTAG exception is triggered. Detailed description see Section **X** Chapter

6.21 Interrupt exception

the reason

When one of the eight interrupt conditions triggers, an interrupt exception occurs. The importance of these interruptions depends on the specific system implementation Now.

By clearing the corresponding bit in the Interrupt-Mask (IM) field in the status register, any of the eight interrupts All can be masked, and by clearing the IE bit of the status register, all eight interrupts can be masked at once.

deal with

The ExcCode field of the Cause register is set to the INT code value. Based on the current configuration, the processor uses traditional Use exception vector processing or use the vector exception mode to select the entry corresponding to the highest priority interrupt number for processing.

The IP field in the Cause register indicates the current interrupt request. More than one interrupt bit may be set at the same time (e.g. If the interrupt is triggered and is cancelled before the register is read, even no bit is set).

There are three sources of IP [7] interrupt. Except interrupt line 5, the content of Count register is equal to that of Compare register Or when the CP0 performance counter overflows. Clock interrupt and performance counter overflow interrupt are caused by Cause register TI and PCI bit indication.

If the vectored interrupt mode is not used, the software needs to query each possible interrupt source to determine the original cause of the interrupt Due to (an interrupt may have multiple sources at the same time).

service

If the interrupt is caused by one of the two software exceptions, set the corresponding bit in the Cause register, IP [1:

Page 104

Godson 3A1000 Processor User Manual • Next

0], set to 0 to clear the interrupt condition.

Software interrupts are not precise. Once the software interrupt is triggered, the program may continue to execute several times before the exception is handled

Instructions. The timer interrupt is cleared by writing values to the Compare register. Clearance of performance counter interrupt

It is realized by writing 0 to the overflow bit of the counter, that is, bit 31.

Cold reset and soft reset will clear all outstanding external interrupt requests, IP [2] to IP [6].

If the interrupt is generated by hardware, then cancel the condition that caused the interrupt pin that triggered the interrupt condition to clear the interrupt condition.

Page 105

Godson 3A1000 Processor User Manual • Next

7 floating point coprocessor

This chapter describes the characteristics of Loongson 3A1000 processor Floating Point Unit (FPU),

Including programming model, instruction set and instruction format, instruction pipeline and exception. Loongson 3A1000 floating point coprocessor and

The relevant system software fully complies with the ANSI / IEEE 754-1985 binary floating point arithmetic standard.

7.1 Overview

The FPU, as a coprocessor of the CPU, is called CP1 (Coprocessor 1). By extending the instruction set of the CPU

Complete floating-point arithmetic operations.

The FPU consists of the following two functional units:

- FALU1 unit
- FALU2 unit

The FALU1 module can perform all floating-point operations except floating-point memory access and floating-point and fixed-point data transfer, including

Floating point addition (subtraction), floating point multiplication, floating point multiplication and addition (subtraction), floating point division, floating square root, floating point

Reverse after rooting, floating-point and fixed-point conversion, floating-point precision conversion, floating-point comparison, transfer judgment, and other simple logic. In additio

The FALU1 module performs SIMD media operations by expanding and multiplexing the FMT field in the instruction code.

FALU2 performs floating-point multiply-add operations (computing floating-point multiply, add, and floating-point multiply-add instructions), and media instruction

Make. At the same time, the FPU of Loongson 3A1000 supports parallel single precision (Paired-Single,

Referred to as PS) floating-point instructions. Figure 7-1 illustrates the organization of functional units in Loongson 3A1000 architecture

Bright.

Figure 7-1 Organization of functional units in Loongson 3A1000 architecture

The floating-point queue can issue 1 instruction to the FALU1 unit and 1 instruction to the FALU2 unit every clock cycle. Floating point register file provides three dedicated read ports and one dedicated write for FALU1 unit and FALU2 unit port.

7.2 FPU register

This section describes the FPU register set and their data organization structure. Loongson 3A1000 FPU register and The FPU registers of MIPS64 are compatible. The FPU registers of MIPS64 include floating-point registers and floating-point control registers. its Middle floating-point control registers include FIR (No. 1), FCSR (No. 31), FCCR (No. 25), FEXR (No. 26), FENR (No. 28) etc.

7.2.1 Floating-point registers

The floating-point registers of Godson 3A1000 follow the usage of R10000, which is slightly different from MIPS64. Send in Status Control

When the FR bit of the memory is 1, there are 32 64-bit floating-point registers, as shown in the following figure; in the Status control register When the FR bit is 0, R10000 has only 16 32-bit or 64-bit floating-point registers, and MIPS64 means there are 32 32 Bit floating point register or 16 64 bit floating point registers.

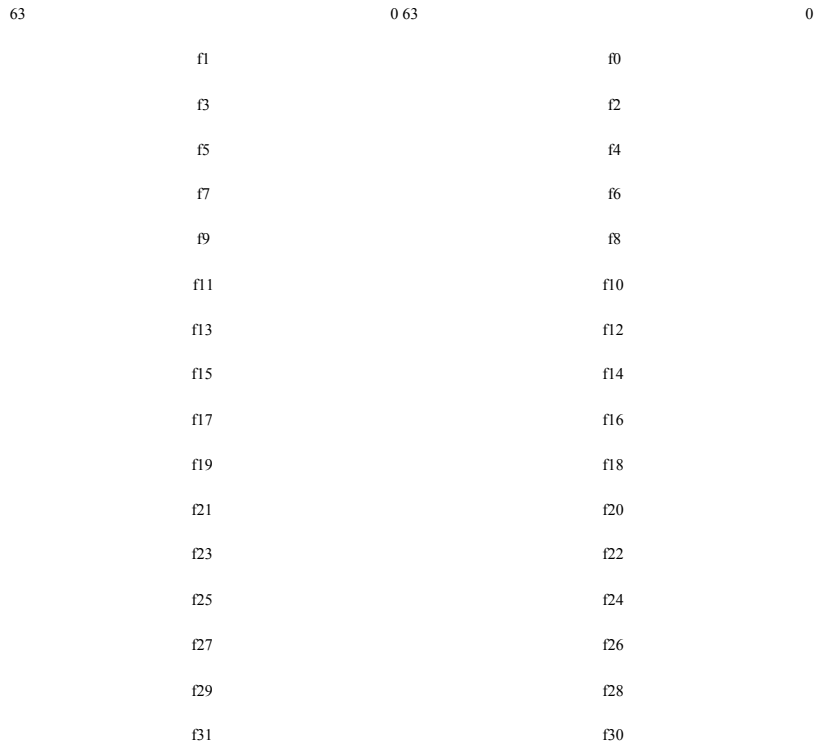


Figure 7-2 Floating-point register format

7.2.2 FIR register (CP1, 0)

FIR is a 32-bit read-only register, which contains the functions implemented by the floating-point unit, such as processor ID, revision number, etc. information. The initial value of FIR in Godson 3A

Figure 7-3 shows the format of the FIR regist cribes the fields of this register .

105

Figure 7-3 FIR register

Table 7-1 FIR register field

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
Impl	Implementation related
F64	Whether the floating-point data path is 64-bit 0-32 bit 1-64 bit
L	Whether the long word (64-bit) fixed-point data type is implemented 0-not implemented 1-realized
W	Whether the word (32-bit) fixed-point data type is implemented 0-not implemented 1-realized
3D	Whether MIPS-3D ASE is realized 0-not implemented 1-realized
PS	Whether floating point implements data types 0-not implemented 1-realized
D	Whether the double-precision floating-point data type is implemented 0-not implemented 1-realized

106

Page 109

area	description
S	Whether the single-precision floating-point data type is implemented 0-not implemented 1-realized
ProcessorID	Floating-point processor identification
Revision	Revision number of floating point unit

7.2.3 FCSR register (CP1, 31)

The FCSR register is used to control the operation of the floatir :ate some states. The initial value of FCSR in GS464 0x00000F80 . The format of the FCSR register is shown in Figure 7- the fields of the FCSR register . its Middle E, V, Z, O, U, I represent unimplemented operation, invalid operation, division by zero, overflow, underflow and inaccuracy

31 25 24 23 2221 20 18 17 12 11 7 6 2 1 0

FCC FS FCC Impl 0 Cause Enables Flags RM

7 6 5 4 3 2 1 1 0 5 EVZOUIVZOUIVZOUI 2

Figure 7-4 FCSR register

Table 7-2 FCSR Register Field

area	description
0	Reserved. It must be written as 0 and returns 0 when read.
FCC	Floating point condition codes. Record floating point comparison results for conditional jumps or transitions.
FS	Flush to 0. When this bit is set, the result of abnormal operation will be set to 0 instead of an exception.
Impl	Implementation related, GS464 uses FSCR [21] as top_mode, this bit is used to indicate whether to use it when decoding The TOP register of X86 renames the floating-point register number.
Cause	When an exception to floating-point arithmetic is generated, the corresponding bit is set.
Enables	Whether to allow exceptions for the corresponding conditions.
Flags	Are there IEEE floating point exceptions? (For example, the corresponding bit is not opened in Enables to view this field)
RM	Whether the double-precision floating-point data type is implemented 0-not implemented 1-realized

107

Page 110

Control / status register condition (**CC0**) bit

When a floating-point comparison operation occurs, the result is stored in the CC0 bit, the condition bit. If the comparison result is true,

Then the CC0 bit is set; otherwise, it is set to 0. The CC0 bit can only be modified by floating-point comparison instructions and CTC1 instructions.

Control / status register cause **(Causes)** field

Bits 17:12 of the control / status register are the Causes field. These bits reflect the result of the most recently executed instruction.

The Causes field is a logical extension of the Cause register of coprocessor 0. These bits indicate that the cause was caused by the last floating-point operation

Exceptions, and an interrupt or exception is generated if the corresponding enable bit (Enable) is set. If one

There are more than one exception in each instruction, and each corresponding exception causes the bit to be set.

The Causes field can be rewritten by each floating-point operation instruction (excluding Load, Store, and Move operations). Where if

If software simulation is required, the unimplemented operation bit (E) of the operation is set to 1, otherwise it remains at 0. Other positions are based on

According to the IEEE754 standard, see if the corresponding exception is generated and set to 1 or 0 respectively.

When a floating-point exception occurs, no result will be stored, and the only affected state is the Causes field.

Control / Status Register Enable **(Enables)** field

At any time, when the Cause bit and the corresponding enable bit (Enable) are both 1, a floating-point exception will be generated. Such as

If the floating-point operation sets a Cause bit that is allowed to be activated (the corresponding enable bit is 1), the processor will immediately generate a

With one exception, this has the same effect as setting the Cause bit and Enable bit to 1 with the CTC1 instruction.

There is no corresponding enable bit for unimplemented operation (E), if unimplemented operation is set, it will always generate a

Exceptions.

Before returning from a floating point exception, the software must first use a CTC1 instruction to clear the activated Cause

To prevent repeated execution of interrupts. Therefore, programs running in user mode will never observe the enabled Cause

The value of the bit is 1; if the user-mode handler needs to obtain this information, the contents of the Cause bit must be passed

Somewhere instead of in the status register.

If the floating-point operation only sets the Cause bit that is not enabled (the corresponding enable bit is 0), no exception occurs.

The default results defined by the IEEE754 standard are written back. In this case, the exception caused by the previous floating-point instruction can

It can be determined by reading the value in the Causes field.

Control / Status Register Flags **(Flags)** field

The flag bit is cumulative and indicates that an exception has occurred since the last time it was explicitly reset. If an IEEE754 exception

Is generated, then the corresponding Flag bit is set to 1, otherwise it remains unchanged, so for floating-point operations these bits will never be

Clear. But we can write a new value to the status register through the CTC1 control instruction to realize the setting of the Flag bit

Set or clear.

When a floating-point exception occurs, the Flag bit is not set by the hardware; the software that handles floating-point exceptions is responsible for calling

These bits are set before the user program.

Control / status register rounding mode **(RM)** field

The 0th and 1st bits in the control / status register form the rounding mode (RM) field. As shown in Table 7-3 As shown in the FPU

According to the rounding mode specified by these bits, all floating-point operations are rounded accordingly.

Table 7-3 Round mode decoding

Rounding mode RM (1: 0)	Mnemonic	description
0	RN	Round the result to the direction closest to the representable number, when the two closest representable numbers are as close as the result, Then it is rounded to the nearest number with the lowest bit as 0.
1	RZ	Round towards 0: Round the result to the number closest to it and not greater than it in absolute value.
2	RP	Round towards positive infinity: round the result to the number closest to it and not less than it

3	RM	Round towards negative infinity: round the result to the number closest to it and not greater than it

7.2.4 FCCR register (CP1, 25)

The FCCR register is another way to access the FCC field, and its content is exactly the same as the FCC bit in the FCSR. The difference is that the FCC bits in this register are continuous. Figure 7-5 shows the format of the FCCR register.

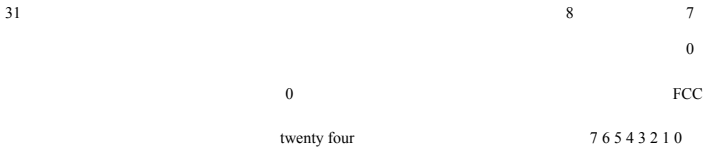


Figure 7-5 FCCR register

7.2.5 FEXR register (CP1, 26)

The FEXR register is another way to access the Cause and Flags fields, and its contents are the same as the corresponding fields in the FCSR Exactly the same. Figure 7-6 shows the format of the FEXR register.

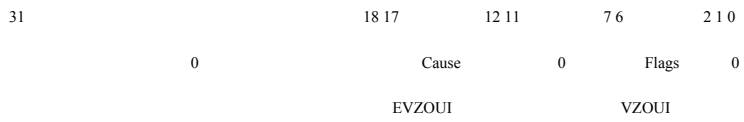


Figure 7-6 FEXR register

7.2.6 FENR register (CP1, 28)

The FENR register is another way to access the Enable, FS and RM fields, and its content is the same as the corresponding word in the FCSR The segments are exactly the same. Figure 7-7 shows the format of the FENR register.

Figure 7-7 FENR register

7.3 Floating-point instructions

7.3.1 MIPS64 compatible floating point instruction list

GS464 implements all data types of the FPU part in MIPS64, including S, D, W, L, and optional PS.

Table 7-4 lists the FPU instructions in the MIPS64 part of GS464.

Table 7-4 MIPS64 FPU instruction set

OpCode	Description	MIPS ISA
	Arithmetic instruction	
ABS.fmt	Absolute value	MIPS32
ADD.fmt	addition	MIPS32
DIV.fmt	division	MIPS32
MADD.fmt	Multiply-add	MIPS64

MSUB.fmt	Multiplication and subtraction	MIPS64
MUL.fmt	multiplication	MIPS32
NEG.fmt	Negate	MIPS32
NMADD.fmt	Invert after multiply-add	MIPS64
NMSUB.fmt	Multiply and subtract	MIPS64
RECIP.fmt	Find the reciprocal	MIPS64
RSQRT.fmt	Find the reciprocal after the square root	MIPS64
SQRT.fmt	Square root	MIPS32
SUB.fmt	Subtraction	MIPS32
	Branch jump instruction	
BC1F	Floating-point false jump	MIPS32
BC1FL	Likely jump when floating point false	MIPS32
BC1T	Floating point real time jump	MIPS32
BC1TL	Floating point realtime Likely jump	MIPS32
	Compare instructions	
C.cond.fmt	Compare floating-point values and set flags	MIPS32
	Conversion instructions	
CEIL.L.fmt	Floating point conversion to 64-bit fixed point, rounded up	MIPS64
CEIL.W.fmt	Floating point conversion to 32-bit fixed point, rounded up	MIPS64
CVT.D.fmt	Floating or fixed point conversion to double precision floating point	MIPS32
CVT.L.fmt	Convert floating-point values to 64-bit fixed-point	MIPS64
CVT.PS.S	Convert two floating-point values to floating-point pairs	MIPS64
CVT.S.PL	Converts the low-order bits of floating-point pairs to single-precision floating-point	MIPS64
CVT.S.PL	Convert high-order bits of floating-point pairs to single-precision floating-point	MIPS64
CVT.S.fmt	Floating or fixed point conversion to single precision floating point	MIPS32
CVT.W.fmt	Convert floating-point values to 32-bit fixed-point	MIPS32
FLOOR.L.fmt	Floating point conversion to 64-bit fixed point, round down	MIPS64
FLOOR.W.fmt	Floating point conversion to 32-bit fixed point, round down	MIPS64
PLL.PS	Merge the lower bits of two floating-point pairs into a new floating-point pair	MIPS64
PLU.PS	Merge the low and high bits of two floating-point pairs into a new floating-point pair	MIPS64
PUL.PS	Combine the high and low bits of two floating-point pairs into a new floating-point pair	MIPS64

PUU.PS	Merge the high-order bits of two floating-point pairs into a new floating-point pair	MIPS64

ROUND.L.fmt	Round floating-point numbers to 64-bit fixed point	MIPS64
ROUND.W.fmt	Round floating-point numbers to 32-bit fixed-point	MIPS32
TRUNC.L.fmt	Round floating-point numbers to 64-bit fixed points with small absolute values	MIPS64
TRUNC.W.fmt	Round floating-point numbers to 32-bit fixed points with small absolute values	MIPS32
	Fetch instruction	
LDC1	Access double words from within	MIPS32
LDXC1	Access double words from within by index	MIPS64
LUXC1	Access double words from within by unaligned index	MIPS64
LWC1	Access word	MIPS32
LWXC1	Access words by index	MIPS64
SDC1	Save double word to memory	MIPS32
SDXC1	Save doubleword to memory by index	MIPS64
SUXC1	Save doublewords to memory by unaligned index	MIPS64
SWC1	Save word to memory	MIPS32
SWXC1	Save words to memory by index	MIPS64
	MOVE instruction	
CFC1	Read floating point control register to GPR	MIPS32
CTC1	Write floating point control register to GPR	MIPS32
DMFC1	Copy double words from FPR to GPR	MIPS64
DMTC1	Copy double words from GPR to FPR	MIPS64
MFC1	Copy low words from FPR to GPR	MIPS32
MFHC1	Copy high words from FPR to GPR	MIPS32 R2
ALNV.PS	Variable floating point alignment	MIPS64
MOV.fmt	Copy FPR	MIPS32
MOVF.fmt	FPR when floating-point fake	MIPS32
MOVN.fmt	Copy FPR when GPR is not 0	MIPS32
MOVT.fmt	Floating point real time copy FPR	MIPS32
MOVZ.fmt	Copy FPR when GPR is 0	MIPS32
MTC1	Copy low words from GPR to FPR	MIPS32
MTHC1	Copy high words from GPR to FPR	MIPS32 R2

7.3.2 MIPS64 compatible floating-point instruction implementation related instructions

GS464 is compatible with the MIPS64 Release 2 version, and functionally implements all the requirements of the MIPS64 architecture FPU instructions, but some instructions have subtle implementations that do not affect compatibility but are more important differences. The following two The point is worth the programmer's attention.

(1) Multiply-add, multiply-subtract instructions. After executing MADD.fmt, MSUB.fmt, NMADD.fmt, NMSUB.fmt this For four sets of instructions, the calculation result of GS464 is slightly different from that of MIPS64 processor, because GS464 is doing multiply and add The calculation only rounds precision at the final result (so-called fused-mulitply-add), while the MIPS64 processor is multiply Rounding is performed twice after the operation and after the addition operation, the difference between the two rounding processing methods leads to the final result in some cases The lowest bit differs by 1.

(2) Single precision arithmetic instruction. When the FR bit of the Status control register is 0, abs.s, add.s, ceil.wd, ceil.ws, div.s, floor.wd, floor.ws, mul.s, neg.s, round.wd, round.ws, sqrt.s, sub.s, trunc.wd, trunc.ws, mov.s, cvt.ds, cvt.dw, cvt.sd, cvt.sw, cvt.wd, cvt.ws, movf.s, movn.s, movt.s, movz.s
The instruction cannot use odd-numbered registers, but the MIPS64 architecture processor can. At this point, Loongson continues to use The practices of MIPS R4000 and MIPSR10000 are slightly different from those of MIPS64. (Early MIPS processors The middle FR bit indicates whether the floating-point register is 16 or 32, and the MFR64 middle FR bit indicates whether the floating-point register is 32-bit or 64-bit).

7.3.3 Godson custom extended floating-point instructions

Table 7-5 Custom extended floating-point memory access instructions

Instruction mnemonic	Instruction function brief
GSSQC1	Dual source registers store fixed-point four words
GSSWLEC1	Save word from floating-point register with out-of-bounds check
GSLWXC1	Floating-point word with offset
GSLQC1	Double target register fetches floating point four words
GSLWLEC1	Fetch word with floating-point check to floating-point register
GSLWLC1	Take the left part of the word to the floating point register
GSLWRC1	Take the right part of the word to the floating point register
GSLDLC1	Take the left part of the double word to the floating-point register
GSLDRC1	Take the right part of the double word to the floating point register

113

Godson 3A1000 Processor User Manual • Next

Instruction mnemonic	Instruction function brief
GSLWGTC1	Fetch word with floating point check to floating point register
GSLDLEC1	Take double word with floating-point check to floating point register
GSLDGTC1	Take double word to floating-point register with lower out-of-bounds check
GSLDXC1	Floating double word with offset
GSSWLC1	Save the left part of the word from the floating-point register
GSSWRC1	Save the right part of the word from the floating-point register
GSSDLC1	Save double word left from floating point register
GSSDRC1	Save double word right from floating point register
GSSWGTC1	Save word from floating-point register with lower out-of-bounds check
GSSDLEC1	Save double word from floating-point register with cross-border check

GSSDGTC1	Save double word from floating-point register with lower out-of-bounds check
GSSWXC1	Floating-point word with offset
GSSDXC1	Floating double word with offset

Table 7-6 Custom extended floating-point format conversion instructions

Instruction mnemonic	Instruction function brief
CVT.D.LD	Convert double precision to double precision
CVT.LD.D	Conversion of double precision to extended double precision low
CVT.UD.D	Conversion of double precision to extended double precision high

7.4 Floating-point component format

7.4.1 Floating point format

The FPU can perform 32-bit (single-precision) or 64-bit (double-precision) IEEE-compliant floating-point numbers operating. The 32-bit single-precision format includes a 24-bit fractional field (F + S) in sign-magnitude and an 8 Bit exponent field (E); 64-bit double-precision format includes a 53-bit sign-magnitude field (F + S) And an 11-bit exponent field (E); the 64-bit double-precision (PS) format contains two single-precision floating-point formats. Respectively, as shown in 7-8 .

Single precision format Double precision format

PS format (double single precision)

Double single precision format

Figure 7-8 Floating-point format

As shown in Figure 7-8 , the format of floating-point numbers consists of the following three fields:

- Symbol field, S
- Index field with offset, E = E 0 + Bias, E 0 is the index without offset
- Decimal field, F = .b 1 b 2 .... b p–1

The range of the index E 0 is an integer between E min and E max , plus the following two guarantees Residual value:

- E min -1 (used to encode 0 and subnormal numbers)
- E max +1 (used to encode ∞ and NaN [Not a Number])

For single-precision or double-precision formats, each non-zero number that can be represented has a unique code corresponding to it. The value V corresponding to the code is determined by the equation in Table 7-7 .

Table 7-7 Formulas for calculating floating-point values in single-precision and double-precision formats

NO.	formula
(1)	if E0 = Emax + 1 and F ≠ 0, then V = NaN, regardless of s
(2)	if E0 = Emax + 1 and F = 0, then V = (–1) S∞
(3)	if Emin≤E0≤Emax, then V = (–1) S2 E0 (1.F)
(4)	if E0 = Emin–1 and F ≠ 0, then V = (–1) S2Emin (0.F)
(5)	if E0 = Emin–1 and F = 0, then V = (–1) S0

For all floating-point formats, if V is a NaN, then the highest bit of F determines that the number is Signaling

NaN or Quiet NaN: If the highest bit of F is set, then V is Signaling NaN, otherwise V is Quiet Na

Table 7-8 defines the values of some related parameters in floating point format; the maxim lues of floating point are given in Table 7-9.

Table 7-8 Floating-point format parameter values

parameter	format	
	Single precision	Double precision
Emax	+127	+1203
Emin	-126	-1022
Exponential offset	+127	+1023
Exponent width	8	11
Integer bits	Hidden	Hidden
F (decimal width)	twenty four	53
Format total width	32	64

Table 7-9 Floating-point values for maximum and minimum numbers

Types of	value
Single-precision floating-point minimum	1.401129846e–45
Single precision floating point minimum normal number	1.175494351e–38
Single precision floating point maximum	3.402823347e + 38
Double precision floating point minimum	4.940656458412465 4e–324
Double precision floating point minimum normal number	2.2250738585072014e–308
Double precision floating point maximum	1.797693134862315 7e + 308

7.5 FPU instruction pipeline overview

The FPU provides an instruction pipeline parallel to the CPU instruction pipeline. It shares the basic 9-stage pipeline with the CPU Architecture, but according to different floating point operations, the execution pipeline level is subdivided into 2 to 6 pipeline levels. Each FPU instruction is One of the two floating-point functional units is executed: FALU1 or FALU2. FALU1 can perform all floating-point arithmetic operations Work and media operations. FALU2 only performs floating point addition and subtraction, multiplication, multiply-add operations, and all media operations.

Each FALU unit can receive 1 instruction per cycle and can send one to the floating-point register file result. In each FALU unit, floating point addition and subtraction, floating point multiplication, floating point multiplication and addition operations require 6 execution cycles; fixed

The format conversion operation between floating point and floating point requires 4 execution cycles; floating point division requires 4 ~ 16 executions depending on the operand

Cycle; the square root of floating point needs 4 ~ 31 execution cycles according to the different operands, other floating point operations require 2 executions

cycle. In each FALU unit, if two instructions with different execution cycles output the result on the same beat, in this way

In this case, the instruction with a shorter execution cycle has priority to output the result to the bus. Among them floating point except floating point division and floating point root

Operations and all media operations are fully pipelined. If there are two floating-point division instructions or two floating-point square root instructions

Order in FALU1, then FALU1 unit will send a pause signal to the first water level, and FALU1 unit is in

No new instructions can be received until the division or square root instruction is written back.

7.6 Floating point exception handling

This section describes exceptions for floating point calculations. Floating point exceptions occur when the FPU cannot handle operands in the usual way or

When the result of floating-point calculation, the FPU generates a corresponding exception to start the corresponding software trap or set the status flag.

The control and status registers of the FPU contain an enable bit for each exception. The enable bit determines whether an exception is

Can it cause the FPU to initiate an exception trap or set a status flag.

If a trap is started, the FPU keeps the state where the operation started, and the software exception processing path is started; if there is no trap

At startup, an appropriate value is written to the FPU target register, and the calculation continues.

The FPU supports five IEEE754 exceptions:

- Inexact (I)
- Underflow (U)
- Overflow (O)
- Division by Zero (Z)
- Invalid Operation (V)

And the sixth exception:

- Unimplemented Operation (E)

Unimplemented operation exceptions are used when the FPU cannot execute the standard MIPS floating-point structure, including the FPU cannot be determined correctly

Of exceptional behavior. This exception indicates the execution of software exception processing. Unimplemented operation exception without enable signal and

Flag bit, when this exception occurs, a corresponding unimplemented exception trap occurs.

The five exceptions of IEEE754 (V, Z, O, U, I) all correspond to an exception trap controlled by the user.

When one of the enable bits is set, the corresponding exception trap is allowed to occur. When an exception occurs, the corresponding cause (Cause)

The bit is set. If the corresponding Enable bit is not set, the Exception flag is set. If enabled

Bit is set, then the flag bit is not set, and the FPU generates an exception to the CPU. Subsequent exception handling allows the

An exception trap occurs.

When there is no exception trap signal, the floating-point processor adopts the default method for processing, providing a floating-point calculation exception junction

The replacement value of the fruit. Different exception types determine different default values. Table 7-10 lists the FPU for each IEEE example

Default processing outside.

Table 7-10 Default handling of exceptions

area	description	Rounding mode	Default action
I	Inexact exception	Any	Provide rounded results
U	Underflow exception	RN	Set the result to 0 according to the sign of the intermediate result
		RZ	Set the result to 0 according to the sign of the intermediate result
		RP	Correct positive underflow to the smallest positive number, and negative underflow to -0
		RM	Correct negative underflow to the smallest negative number, and positive underflow to +0
O	Overflow exception	RN	Set the result to infinity according to the sign of the intermediate result
		RZ	Set the result to the maximum number according to the sign of the intermediate result
		RP	Correct negative underflow to the largest negative number, and positive underflow to + ∞
		RM	Correct positive underflow to the largest integer and negative underflow to -∞
Z	Divide by 0	Any	Provide a corresponding signed infinity number
V	Illegal operation	Any	Provide a Quiet Not a Number (QNaN)

The conditions that cause each exception to the FPU are described below, and the FPU guides each exception

Conditional response.

Inexact exception **(I)**

The FPU produces an inaccurate exception when the following occurs:

- Rounding result is not precise
- Rounding result overflow
- The rounding result underflows, and neither the underflow nor the inaccurate enable bit is set, and the FS bit is set.

Trap enabled result: If a non-exact exception trap is enabled, the result register is not modified, and the source

The register is reserved. Because this execution mode affects performance, imprecise exception traps are only used when necessary

Enable.

The result of the trap not being enabled: if no other software trap occurs, the rounding or overflow result is sent to the target

register.

Illegal operation exception **(V)**

Godson 3A1000 Processor User Manual • Next

When two operands of an executable operation or one of the operands is illegal, an illegal operation exception is sent

Notice. If the exception is not caught, MIPS defines this result as a Quiet Not a Number (QNaN). non-

Legal operations include:

- Addition or subtraction: infinite subtraction. For example: (+ ∞) + (-∞) or (-∞)-(-∞)
- Multiplication: 0 × ∞, for all positive and negative numbers
- Division: 0/0, ∞ / ∞, for all positive and negative numbers
- When the operand of the comparison operation that does not handle Unordered is Unordered
- Perform floating-point comparison or conversion on an indicator signal NaN
- Any mathematical operation on SNaN (Signaling NaN). When one of the operands is SNaN or two

Both result in this exception when SNaN (MOV operations are not considered to be mathematical operations, but ABS and NEG are considered to be Mathematical operations)

- Prescription: X , when X is less than 0

The software can simulate the exception of illegal operations of other given source operands. For example, using software in IEEE754 to implement The current specific function: X REM Y, here when Y is 0 or X is infinity; or when the floating-point number is converted to decimal Overflow occurs during control, which is infinite or NaN; or a priori function such as ln (5) or cos -1 (3).

The result of the trap being enabled: the value of the source operand is not sent.

The result of trap disabling: If no other exception occurs, QNaN is sent to the target register.

Except for zero exception **(Z)**

In division operations, when the divisor is 0 and the dividend is a finite non-zero data, the division by zero exception is signaled. Benefit The software can be used to simulate the exception of zero when generating signed infinite values for other operations, such as: ln (0), sin (π / 2), cos (0), Or 0 -1 .

Trap is enabled: the result register is not modified, the source register is retained.

Trap is not enabled: If no trap occurs, the result is a signed infinite value.

Overflow exception **(O)**

When the amplitude of the floating-point result after rounding is expressed by an exponent without limits, the target mode greater than Limit data, overflow notification signal. (This exception sets both inexact exceptions and flags)

Trap is enabled: the result register is not modified, the source register is retained.

Trap is not enabled: if no trap occurs, the final result comes from the rounding mode and the sign of the intermediate result Decide.

Underflow exception **(U)**

119

Page 122

Godson 3A1000 Processor User Manual • Next

Two related events led to underflow exceptions:

- A very small non-zero result between ± 2 Emin , because the result is very small, it will cause subsequent Overflow exception.
- Use subnormal data (Denormalized Number) to approximate the seriousness of these two small data The data is distorted.

IEEE754 allows many different methods to detect these events, but requires the same method for all operations Detection. Small data can be detected using one of the following methods:

- After rounding (if a non-zero data is calculated without exponential range, it should be strictly Between ± 2 Emin)
- Before rounding (if a non-zero data is calculated without exponential and precision bounds, it should be Strictly between ± 2 Emin)

The structure of MIPS requires small data to be detected after rounding. Accuracy distortion can be detected by one of the following methods:

- Distortion of subnormal data (when the result produced is different from the exponent without calculation)
- Inaccurate data (when the result produced is different from the exponent and the accuracy range, the calculated result is different)

The MIPS structure requires that precision distortion be detected to produce inaccurate results.

Trap is enabled: if an underflow or inaccurate exception is enabled, or the FS bit is not set, an unrealized The current operation is exceptional, and the result register is not modified.

Trap is not enabled: if an underflow or inaccurate exception is not enabled, and the FS bit is set, the final result

The result is determined by the rounding mode and the sign bit of the immediate result.

Unimplemented operation exception **(E)**

When executing any operation code or operation format instruction reserved for future definition, FPU control / status registration

Unimplemented operations in the device cause the bit to be set and generate a trap. The source operand and destination register remain unchanged, while the instruction is

Simulation in the software. Any exception in IEEE754 can be generated from the simulation operation, and these exceptions can in turn be

simulation. In addition, when the hardware cannot correctly perform some rare operations or result conditions, it will also generate unimplemented instruction examples

outer. These include:

- Subnormal operand (Denormalized Operand), except for comparison instructions
- Quite Not a Number operand (QNaN), except for comparison instructions
- Subnormal data or underflow, and when the underflow or inaccurate enable signal is set and the FS bit is not set

Set

Note **:** Subnormal and NaN operations only enter traps in conversion or calculation instructions, not traps in MOV instructions

120

Godson 3A1000 Processor User Manual • Next

trap.

Trap is enabled: the original operation data is not sent.

Trap is not enabled: this trap cannot be disabled.

121

8 Performance analysis and optimization

This chapter provides some information related to software performance optimization in Loongson 3A1000 architecture, including instruction delay and

The instruction loop interval, extended instruction, instruction stream, and storage access processing can be used by compilers and other software developers for reference.

8.1 Delay and cycle interval of user instructions

Table 8-1 shows the delay and cycle of all user instructions executed in the ALU1 / 2, MEM, FALU1 / 2 functional units

Ring interval, excluding kernel instructions and control instructions. The instruction delay here is that the instruction is issued until its result can be the next instruction

Make the required number of beats (one processor cycle is one beat). For example, most of the ALU instruction delay is 2, this table

The result of the ALU instruction can only be used by subsequent instructions after one beat. Therefore, a related cycle of the form i = i + 1 (below

A cycle depends on the result of the previous cycle). And the cycle interval of an instruction refers to the function

The frequency with which the component accepts such commands, 1 means that it can accept more than one command of the same kind per shot, and n means that the functional cor

After this instruction, you need to wait for n-1 beats before accepting the same kind of instruction. The instruction cycle interval of the full pipeline function unit is 1.

Table 8-1 Loongson 3A1000 instruction delay

Instruction type	Executive part	delay	Cycle interval
Integer operation			
add / sub / logical / shift / lui / cmp	ALU1 / 2	2	1
trap / branch	ALU1	2	1
MF / MT HI / LO	ALU1 / 2	2	1
(D) MULT (U)	ALU2	5	2
(D) MULT (U) G	ALU2	5	1
(D) DIV (U)	ALU2	2-38	10-76
(D) DIV (U) G	ALU2	2-38	4-37
(D) MOD (U) G	ALU2	2-38	4-37
load	MEM	5	1
store	MEM	-	1
Floating point operation			
(D) MTC1 / (D) MFC1	MEM	5	1
abs / neg / C.cond / bc1t / bc1f / move / cvt *	FALU1	3	1
round / trunc / ceil / floor / cvt *	FALU1	5	1
add / sub / mul / madd / msub / nmadd / nmsub	FALU1 / 2	7	1
div.s	FALU2	5-11	4-10
div.d	FALU2	5-18	4-17
sqrt..s	FALU2	5-17	4-16
sqrt..d	FALU2	5-32	4-31
lwc1, ldc1	MEM	5	1

122

Instruction type	Executive part	delay	Cycle interval
swc1, sdc1	MEM	-	1

For Table 1, there are the following points:

The cycle interval of the load / store operation here does not include LL / SC. LL / SC is waiting for launch operation, only when

They are at the head of the reorder queue, and can only be launched when the cp0 queue is empty.

There are no special restrictions on the use of HI / LO registers. They are used like other general-purpose registers. This one

The table does not include CTC1 / CFC1. They are serialized like many other control instructions.

Multimedia commands are also not included in this table. Because they are done by extending the format of ordinary floating-point instructions, it

Our functional units and delays are the same as the extended instructions.

8.2 Precautions for instruction expansion and use

Loongson 3A1000 has completed the following instructions expansion:

Write only one result to the fixed-point multiplication and division of the general register. Includes 12 instructions:

(D) MULTG, (D) MULTUG, (D) DIVG

(D) DIVUG, (D) MODG, (D) MODUG

In the standard MIPS instruction set, multiplication and division need to write two special result registers in one operation

(HI / LO), they are difficult to achieve in the RISC pipeline. In order to use these results, you will have to use additional

Let it be taken out from HI / LO and sent to the general register. Even more troublesome is that, due to pipeline problems, many MIPS

The processor has some restrictions on the use of these instructions. These new instructions execute faster and are easier to use.

Due to the implementation of register renaming, on the Loongson 3A1000 processor, the standard MIPS instruction set involves

HI / LO register operation related instructions must wait until all the instructions in the processor instruction queue before these instructions are submitted

The execution can only be started afterwards, which will cause the pipeline to stop. But when using the above extension instructions, there is no HI / LO in the pipeline

Register renaming problem, these instructions will not be blocked. Therefore, when writing programs using assembly instructions, you should try to

Use the above extended instructions. If you really need to use the HI / LO operation result, for example, you need to get the multiplication after 64-bit multiplication

The higher 64 bits of the method result, then the programmer should clearly know that using the standard MIPS multiply instruction will cause the pipeline

Impact.

Fixed-point operations use floating-point data paths:

During the execution of fixed-point programs, floating-point data paths are often idle

Use them to further increase the degree of instruction parallelism.

123

8.3 Compiler usage tips

The open source compiler suite GCC currently supports Loongson 3A1000 processor architecture tuning options. in

In GCC 4.6.0 and above, use -march = loongson3A to use the pipeline description for the processor to

Code scheduling, the generated code can also make full use of Godson 3A1000's instruction expansion. In most cases, make

With this option, performance can be improved on the Loongson 3A1000 processor.

In addition, in the process of exploring the compiler tuning space for the SPEC CPU2000 benchmark test set, we concluded

The following GCC compilation options may improve the performance of Godson 3A1000 processor. Fine tuning the program

In the process, the following options have certain reference significance.

-fdefer-pop	-fcaller-saves
-fno-move-loop-invariants	-fno-cprop-registers
-funroll-all-loops	-fno-early-inlining
-ffunction-cse	-floop-optimize
-fno-optimize-register-move	-fno-peephole
-freorder-blocks	-fno-peephole2
-ftracer	-fprefetch-loop-arrays
-ftree-fre	-fsched-spec-load-dangerous
-fno-cse-follow-jumps	-fschedule-insns2
-fno-math-errno	-fsignaling-nans
-fno-optimize-sibling-calls	-fno-strength-reduce
-fno-peel-loops	-fthread-jumps
-fsingle-precision-constant	-fno-tree-copyrename
-ftree-loop-optimize	-ftree-dominator-opts
-fno-branch-count-reg	-ftree-vect-loop-version

8.4 Instruction flow

Loongson 3A1000 is a multi-launch highly parallel processor, which may process the instruction stream which is essentially serial

This has a significant impact on program performance. This section discusses issues such as instruction alignment, branch instructions, and instruction scheduling.

8.4.1 Instruction alignment

In one cycle, Loongson 3A1000 can fetch four instructions from a cache line, but these four instructions are not

Can cross the boundary of the cache line. We should align the basic blocks that are executed frequently to avoid

Cross the boundary of the cache line. In addition, if there is a transfer instruction among the four instructions fetched at once, it will also affect the fetch instruction

Efficiency. If the first branch instruction is a branch and the branch prediction is successful, then the last two instructions will be discarded.

If the last one is a transfer instruction, even if the transfer is successful, the processor has to take another cache line to get it

Instruction in the delay slot. Godson 3A1000 can only decode a branch instruction in one cycle.

In the case of two branch instructions, it will take two cycles to complete the decoding, which means that the fetch unit will be blocked for one cycle.

8.4.2 Processing of branch instructions

In the Godson 3A1000 processor, an unexpected change in the instruction stream address would waste about 10 fingers

Time. "Unexpected" can be caused by a successful branch instruction, or it can be caused by a branch prediction error. Correct

As far as the current Loongson 3A1000 is concerned, even a correctly predicted and predicted successful transfer instruction is better than the sequential code

Slow, it will waste a cycle, because for ordinary conditional branch instructions, the branch target buffer (BTB) cannot be given under

A correct program counter PC value.

The compiler can reduce the overhead caused by branch instructions in the following ways:

Godson 3A1000's branch instruction prediction method is different from other high-performance processors, and different versions have a

Some slight differences. Based on the execution profile, the compiler can perform code location based on the actual transfer frequency

Rearrange to get better prediction results.

Make the basic block as large as possible. A better optimization result is that between the two branch instructions where the branch is successful

There are an average of 20 instructions. In order to have at least 20 instructions between them, this needs to be unrolled in a loop

The 20-command subroutine is directly expanded inline. Loongson 3A1000 implements conditional movement commands, which can be used to reduce

The number of branch instructions. Reorganizing the code by performing profiling also helps this optimization.

On Loongson 3A1000, different transfer instructions are predicted in different ways:

Static prediction

For like-like transfer instructions and direct jump instructions.

G-share predictor

A 9-bit global history register GHR, and a pattern history table PHT with 4K entries. Used for conditions

Transfer instructions.

BTB (transfer target cache)

There are 16 items of fully associative cache. Used to predict the target address of a register jump instruction.

RAS (Return Address Stack)

4 items, used to predict the target address returned by the function.

There are a few points to note about software:

On the Godson 3A1000 processor, you need to use the like-like transfer instruction with great care. Despite likely class transfer

Godson 3A1000 Processor User Manual • Next

Instructions may be effective for simple static scheduling of sequential scalar processors, but they are not the same for modern high-performance processors

Kind of effective. Because the transition prediction hardware of modern high-performance processors is relatively complex, they are usually more than 90% correct

Forecast rate. (For example, Godson 3A1000 can correctly predict the transfer of 85% -100%, with an average of 95% conditional transfer

Direction) In this case, the compiler should not use like-like branch instructions with a low prediction rate. In fact, I

We found that gcc with the -mno-branch-likely option usually works better.

The fetch decoding unit is divided into 3 pipeline stages, where the target address of the branch is calculated in the third stage. Transfer into

The transfer instruction of the work will cause a pause of two cycles, that is, if a transfer instruction is taken out in cycle 0,

In cycle 1, the instruction with the address of PC + 16 is fetched, and in cycle 2 the instruction with the address of PC + 32 is fetched. In cycle 3,

Only then will the target address of the branch instruction be obtained. So reducing the number of successful transfer instructions will be more helpful.

The BTB in Loongson 3A1000 is only used for register jump instructions (jr instructions without jr31 and jalr exceptions).

The target address of the jr31 instruction is predicted by a 4-item RAS. The validity of the predictions returned by the function depends on those

Software that uses the jr31 instruction as a function return instruction.

8.4.3 Improvement of instruction stream density

The compiler should use execution profiling as much as possible to ensure that those bytes called into the instruction cache are executed. This is about

Find the target address of the jump instruction to be aligned, and move the code that is rarely executed out of the Cache line.

8.4.4 Instruction scheduling

Loongson 3A1000 has a relatively large instruction window for dynamic instruction scheduling, but due to the various

This kind of resources is limited and the optimal scheduling cannot be achieved. The compiler can assist the processor to better schedule to a certain extent.

Modern compilers (such as gcc) have instruction scheduling support.

The delay situation (see Table 1) tells the compiler that it can perform better scheduling.

8.5 Memory access

The execution of the load-store instruction has a great impact on the performance of the entire system. If the primary data cache includes all

Content, then these instructions can be executed quickly. If the data is only in the secondary cache, it is slightly slower, if only

There will be a large delay in the main memory. However, out-of-order execution and non-blocking cache can reduce the delay caused by these delays

Performance loss.

Loongson 3A1000 includes 4 on-chip secondary cache modules, each secondary cache module size is 1MB, a total of 4MB.

Each module is organized as a four-way group. Loongson 3A1000 built-in DDR memory controller, minimized

Godson 3A1000 Processor User Manual • Next

Memory access latency. The memory frequency and processor working frequency of Loongson 3A1000 system can be configured separately.

High memory frequency, reducing the gap with the processing frequency, is very beneficial to improve the performance of most applications. About memory control

For more information about the controller, please refer to the relevant part of the 3A1000 processor documentation.

Loongson 3A1000 provides prefetch instructions, which can be prefetched by loading to No. 0 fixed-point register

To the first-level data cache. In addition, the DSP engine in Loongson 3 can prefetch the data in memory or IO to the second level

In the Cache, please refer to the relevant parts of the manual.

The compiler should minimize unnecessary storage access. The current Loongson 3A1000 processor has a delay in storing instructions

Large (even a cache hit requires 4 cycles), and the instruction window is not large enough to tolerate tens of weeks

Period of access delay.

The software should pay special attention to the problem of data alignment. The assembly (array, some records, subroutine stack frame) should be

Allocated on the aligned cache line boundary, so that the cache line can be used to align the data path, and can also reduce the high

The number of cache lines filled. In those aggregates that force misalignment (such as the packed attribute of gcc) (records,

In the ordinary block) project, a warning message at compile time should be generated. Normal in Godson 3A1000

The load / store instruction has alignment requirements, and instructions that do not meet the requirements must be implemented through kernel emulation. For example, from non-fi

Taking a word (four bytes) at an aligned address triggers an exception, which is handled by the operating system; usually the operating system requires thousands

The processor cycle can complete this task. Therefore, users need to know that these warning messages indicate that the performance of the code may be very low.

The code of the compilation parameters defaults to the alignment of these parameters. Scalars that are frequently used should reside in registers.

8.6 Other tips

Use all floating-point registers. Although O32 ABI only opened 16 for users, but Godson 3A1000

Provides 32 64-bit floating-point registers. Using the N32 or N64 ABI helps to maximize the performance of the processor.

Use performance counters. The performance counter of Loongson 3A1000 can be used to monitor the real-time performance parameters of the program. Compile

Developers and software developers can improve their code by analyzing this result.