SpartanMC

Instruction Set Architecture

Table of Contents

List of Figures

List of Tables

Instruction Set Architecture

The SpartanMC uses two register addresses per instruction. The first operand (RD register) is automatically used as the destination of the operation. This slightly reduces the effectiveness of the compiler, but it is a reasonable decision with respect to the very limited instruction bit width of 18 bit.

The code efficiency is improved with an additional condition code register which is used to store the result of compare instructions (used for branches).

1. Instruction Types

The instruction set is composd of fixed 18 bit instructions grouped in the four types:

- R-Type (register)
- I-Type (immidiate)
- M-Type (memory)
- J-Type (jump)

1.1. R-Type

R-Type instructions are used for operations which takes two register values and computes a result, which is stored back into operand one.

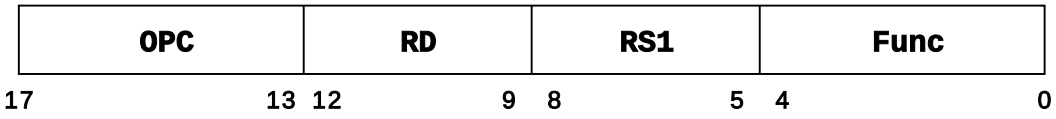


Figure 1: R-Type instruction

1.2. I-Type

This group includes all operations which take one register value and a constant to carry out an operation.

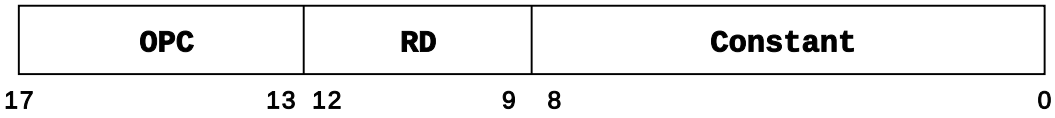


Figure 2: I-Type instruction

1.3. M-Type

This group is used for memory access operations. All load and store operations are available as half word (9 bit) or full word (18 bit) operation.

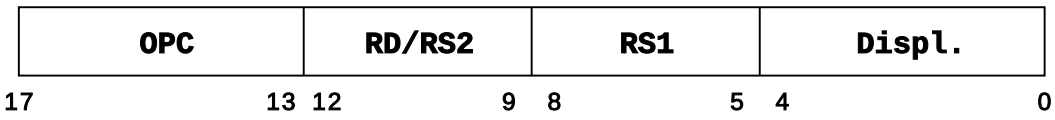


Figure 3: M-Type instruction

1.4. J-Type

This group includes the jump instruction and two branch instructions. The branch instructions interpret the condition code flag (see registers) to decide either to branch or not.

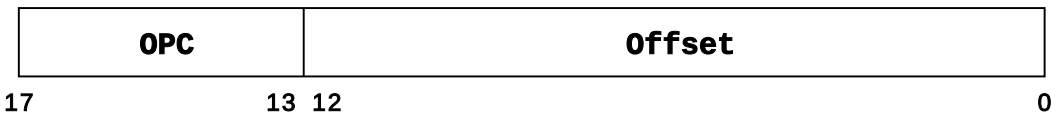


Figure 4: J-Type Instruction

2. Instruction Coding Matrices

The following table shows the instruction coding used on the SpartanMC.

Table 1: Main Matrix using IR 17-13

IR 17-13	..000	..001	..010	..011	..100	..101	..110	..111
00..	Special 1	Special 2	J	JALS	BEQZ	BNEZ	BEQZC	BNEZC
01..	ADDI	MOVI	LHI	SIGEX	ANDI	ORI	XORI	MULI
10..	L9	S9	L18	S18	SLLI	*	SRLI	SRAI
11..	SEQI	SNEI	SLTI	SGTI	SLEI	SGEI	IFADDUI	IFSUBUI

Table 2: Submatrix Special 1 using IR 4-0

IR 4-0	..000	..001	..010	..011	..100	..101	..110	..111
00..	orcc	andcc	*	*	SLL	MOV	SRL	SRA
01..	SEQU	SNEU	SLTU	SGTU	SLEU	SGEU	*	*
10..	*	*	*	*	*	*	CBITS	SBITS
11..	*	*	*	*	*	*	*	NOT

Table 3: Submatrix Special 2 using IR 4-0

IR 4-0	..000	..001	..010	..011	..100	..101	..110	..111
00..	RFE	TRAP	JR	JALR	JRS	JALRS	*	*
01..	*	*	*	*	*	*	*	*
10..	ADD	ADDU	SUB	SUBU	AND	OR	XOR	MUL
11..	SEQ	SNE	SLT	SGT	SLE	SGE	MOVI2S	MOVS2I

Note: * Code not used

Instructions written in lower case are currently not supported.

3. Register Window

The SpartanMC uses 16 addressable 18 bit registers which are stored in a 1k x 18 bit FPGA BlockRAM. The memory block is fully utilized through a sliding window technique. Registers 0 to 3 are used as global registers, registers 8 to 11 are local registers. The registers 4 to 7 are used as function input window for parameter transfer from the calling function. It equals registers 12 to 15 of the calling function which allows up to

four parameters for a function call without external memory. Each shift consumes eight positions in the block memory which results in a total of 127 call levels. Register 11 is reserved for the return address of subroutines or interupt service routines (ISR).

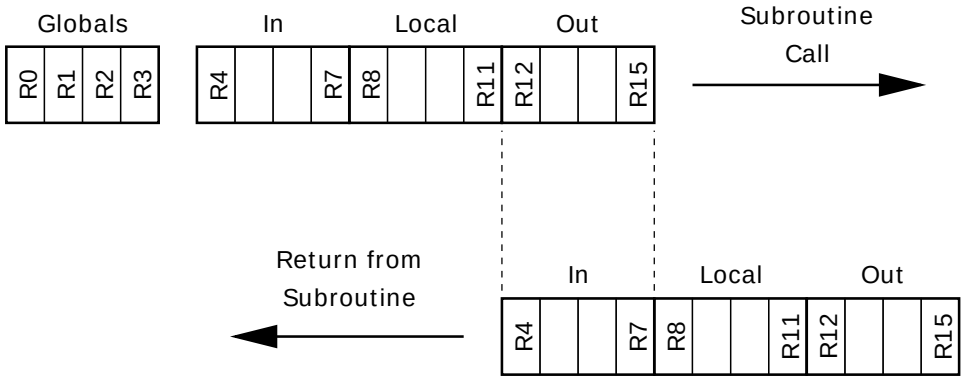


Figure 5: SpartanMC register window

4. Special Function Registers

For special purposes the SpartanMC contains special function registers (SFR). These registers could be modified via SBITS/CBITS instructions.

Note: The contents of all SFRs remain constant until the next access to the corresponding register value.

4.1. Status Register (SFR_STATUS)

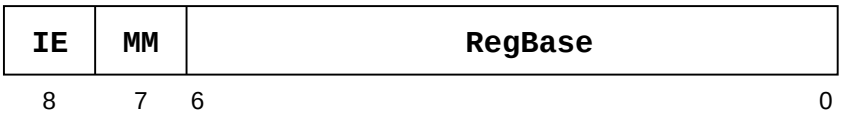


Figure 6: Status Register

SFR-Name: SFR_STATUS

SFR-Nr.: 0

SFR_STATUS [6:0]: Register Base (RegBase) - It contains the number of the current register window. The first window starts at 0. Each subroutine call increments the register by one up to the maximum value of 126.

SFR_STATUS [7]: Memory Management (MM) - This bit is set to 1 if the most significant address bit (address bit nr. 17) is used for memory access (see Address Management).

SFR_STATUS [8]: Interrupt Enable (IE) - If this bit is set to 0, the hardware interrupts are disabled. Setting IE to 1 enables the hardware interrupts.

4.2. LED Register (SFR_LEDS)

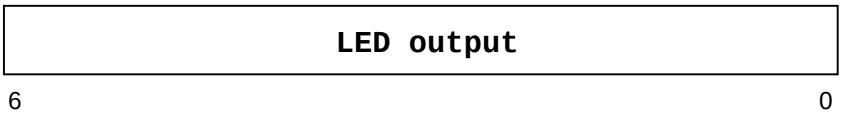


Figure 7: LED Register

SFR-Name: SFR_LEDS

SFR-Nr.: 1

SFR_LEDS [6:0]: This register is usable for custom status outputs.

4.3. MUL Register (SFR_MUL)

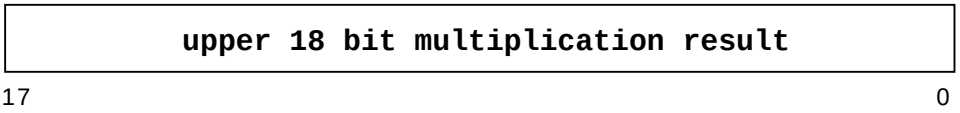


Figure 8: MUL Register

SFR-Name: SFR_MUL

SFR-Nr.: 2

SFR_MUL [17:0]: This register contains the upper 18 bit part [35:18] of a 36 bit result after a multiplication of two 18 bit values.

4.4. Condition Code Register (SFR_CC)

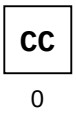


Figure 9: CC Register

SFR-Name: SFR_CC

SFR-Nr.: 3

SFR_CC [0]: Condition Code (CC) - The CC bit is used to store jump conditions. Furthermore it is used to signal an overflow after a signed arithmetic operation.

4.5. Interrupt Vector Register (SFR_IV)

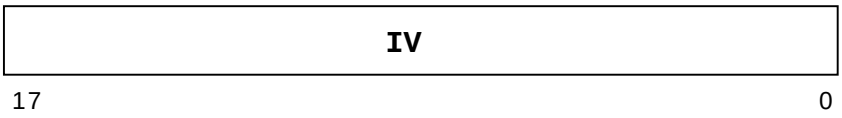


Figure 10: IV Register

SFR-Name: SFR_IV

SFR-Nr.: 4

SFR_IV [17:0]: Interrupt Vector (IV) - This Register contains the start address for the interrupt handling code (context switch and interrupt table lookup). After system reset this address is set to the value defined in the system configuration generated from jConfig. The start address of the interrupt handler can be changed by writing this register. This technique allows the usage of different interrupt service code for identical interrupts. It is recommended to disable the interrupts (set SFR_STATUS [8] to 0) befor writing SFR_IV.

4.6. Trap Vector Register (SFR_TR)

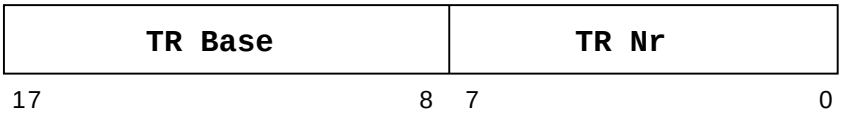


Figure 11: TR Register

SFR-Name: SFR_TR

SFR-Nr.: 5

This register contains the start address for trap service routines.

SFR_TR [17:8]: Trap (TR) - The upper 10 bits contain the base address of the trap table.

SFR_TR [7:0]: Trap (TR) - The lower 8 bits contain the number of the trap (read only - return 0x00 on read request). These bits are set via `trap` instruction.

4.7. Hardware Debugging Registers (SFR_DBG_IDX, SFR_DBG_DAT)

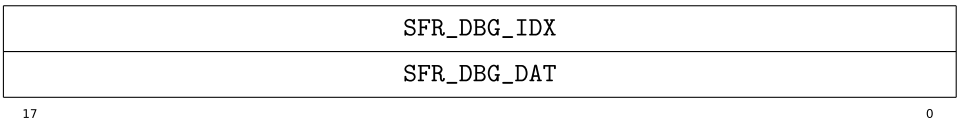


Figure 12: DBG Registers

SFR-Name: SFR_DBG_IDX, SFR_DBG_DAT

SFR-Nr.: 6,7

These registers allow indirect addressing of all Hardware Debugging registers

Both registers will be 0 if the Core was synthesized without hardware debugging support

See Hardware Debugging Support for more details on the indirect access

5. Instruction Set Details

This section is a reference to the entire SpartanMC instruction set.

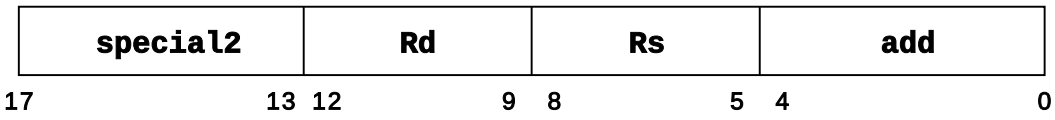
Each of the following pages covers a single SpartanMC instruction. They are organized alphabetically by instruction mnemonic.

add add

add

Mnemonic

add Rd, Rs



Pseudocode

Rd ← Rd + Rs

CC ← OV

Description

The content of GPR Rd and the content of GPR Rs are arithmetically added and form an 18 bit two's complement result, which is written to GPR Rd. If the result of the addition is greater than 2^{17}-1 (i.e.: = 0x1FFFF) or lower than -2^{17}(i.e.: 0x20000), an overflow occurs and CC is set to 1.

Comments

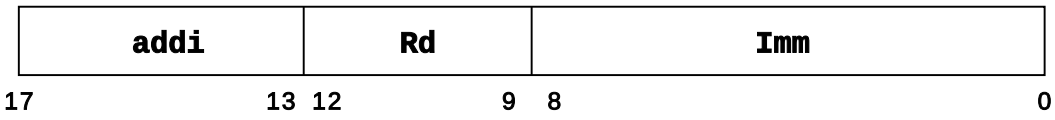
R-Typ

addi addi

add immediate

Mnemonic

```
addi        Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd + IR_8^9 \,\#\#\, IR_{8:0}$

Description

The content of GPR Rd and the immediate Imm are arithmetically added and form an 18 bit two's complement result, which is written to GPR Rd.

Comments

I-Typ

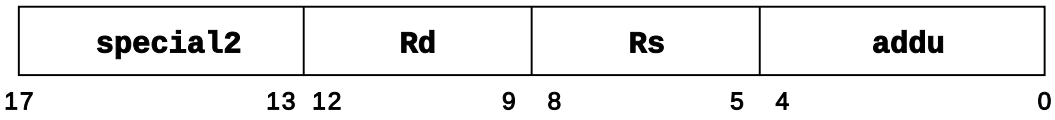
$IR_8^9 \#\# IR_{8:0}$performs a sign extension for the 9 bit immediate.

addu addu

add unsigned

Mnemonic

addu Rd, Rs



Pseudocode

Rd ← Rd + Rs

Description

The content of GPR Rd and the content of GPR Rs are arithmetically added and form an 18-bit two's complement result which is written to GPR Rd.

Comments

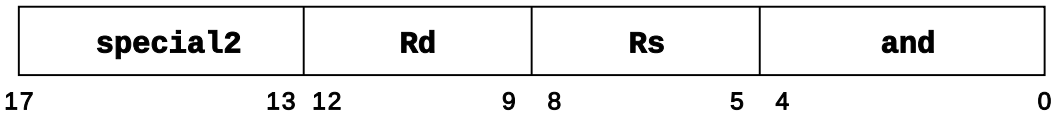
R-Typ

and and

and

Mnemonic

and Rd, Rs



Pseudocode

Rd ← Rd and Rs

Description

The content of GPR Rd is combined with the content of GPR Rs in a bitwise logical AND operation. The result is written to GPR Rd.

Comments

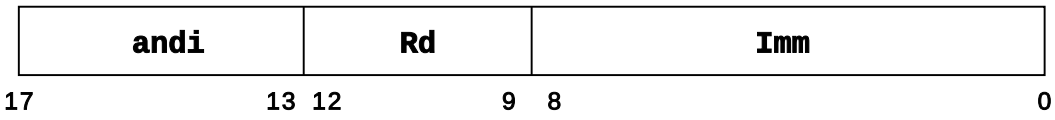
R-Typ

andi andi

and immediate

Mnemonic

```
andi        Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd \text{ and } 0^{9} \# \# IR_{8:0}$

Description

The zero extended 9 bit immediate is combined with the content of GPR Rd in a bitwise logical AND operation. The result is written to GPR Rd.

Comments

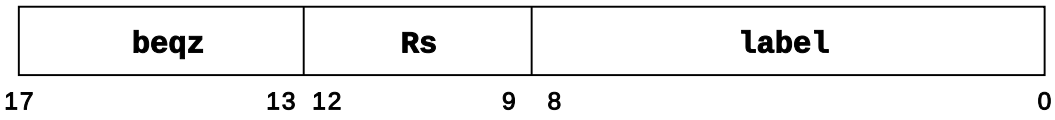
I-Typ

beqz beqz

branch equal zero

Mnemonic

```
beqz        Rs, displacement
```



Pseudocode

```
IF Rs=0; PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

Sets the program counter to PC + displacement, if GPR Rs equals zero. Note, that in contrast to all other relative jumps/branches the displacement has only a size of 9 bit instead of the usual 13 bit.

Comments

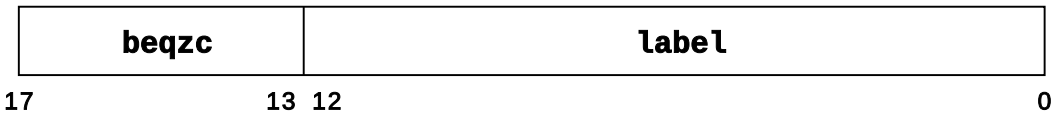
I-Typ

beqzc beqzc

branch equal zero condition bit

Mnemonic

```
beqzc       displacement
```



Pseudocode

```
IF CC=0; PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

Sets the program counter to PC + displacement if CC has a value of zero.

Comments

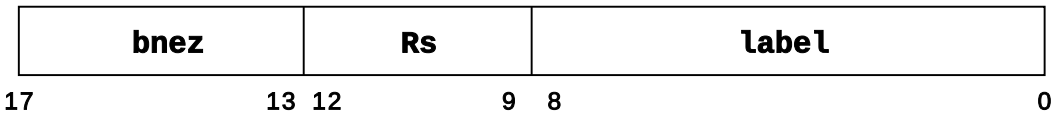
J-Typ

bnez bnez

branch not equal zero

Mnemonic

```
bnez        Rs, displacement
```



Pseudocode

```
IF Rs!=0; PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

Sets the programm counter to PC + displacement, if GPR Rs is unequal to zero. Note, that in contrast to all other relative jumps/branches the displacement has only a size of 9 bit instead of the usual 13 bit.

Comments

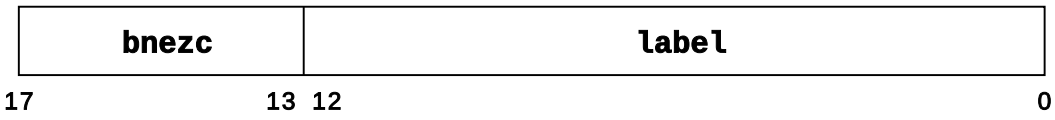
I-Typ

bnezc bnezc

branch not equal zero condition bit

Mnemonic

```
bnezc       displacement
```



Pseudocode

```
IF CC!=0; PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

Sets the program counter to the PC + displacement if CC is unequal to zero.

Comments

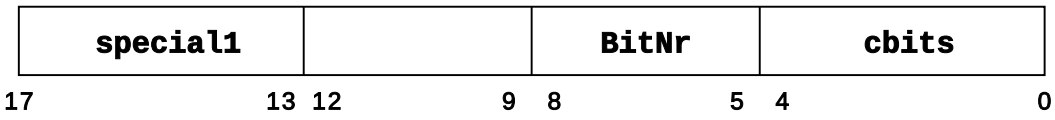
J-Typ

cbits cbits

clears bit at SFR

Mnemonic

```
cbits       BitNr
```



Pseudocode

```
SFR_Status_Bit ← 0
BitNr.: 0 = clears SFR_CC (CC)
BitNr.: 1 = clears SFR_STATUS7(MM)
BitNr.: 2 = clears SFR_STATUS8(IE)
```

Description

Clears a SFR bit according to the given BitNr. A BitNr of zero sets the CC bit to zero, a BitNr of one sets the MM bit to zero and a BitNr of two sets the IE bit to zero.

Comments

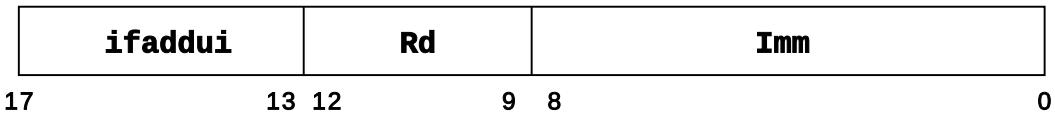
R-Typ

ifaddui ifaddui

conditional addition with an unsigned immediate

Mnemonic

```
ifaddui     Rd, Imm
```



Pseudocode

$$\texttt{IF CC = 1; Rd} \leftarrow \texttt{Rd + } 0^{9} \,\#\#\, \texttt{IR}_{8:0}$$

Description

If the value of CC is one, the addition of the zero extended 9 bit immediate with the content of GPR Rd is carried out. The unsigned 18 bit result is written to GPR Rd. Otherwise GPR Rd remains unmodified.

Comments

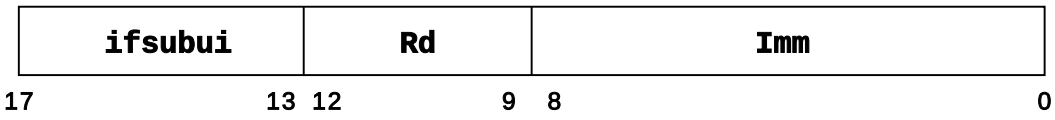
I-Typ

ifsubui ifsubui

conditional subtraction with an unsinged immediate

Mnemonic

```
ifsubui     Rd, Imm
```



Pseudocode

$$\texttt{IF CC=1; Rd} \leftarrow \texttt{Rd - } 0^{9} \,\#\#\, \texttt{IR}_{8:0}$$

Description

If the value of CC is one, the subtraction of the zero extended 9 bit immediate from the content of GPR Rd is carried out. The unsigned 18 bit result is written to GPR Rd.

Comments

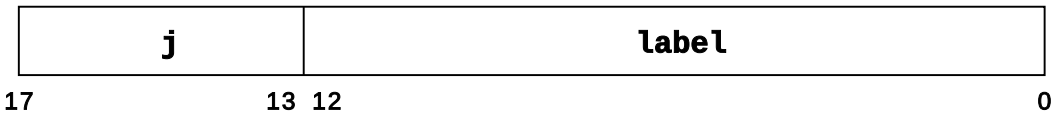
I-Typ

j j

jump

Mnemonic

```
j           displacement
```



Pseudocode

```
PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

Sets the PC unconditionally to the target address given with the value PC + displacement.

Comments

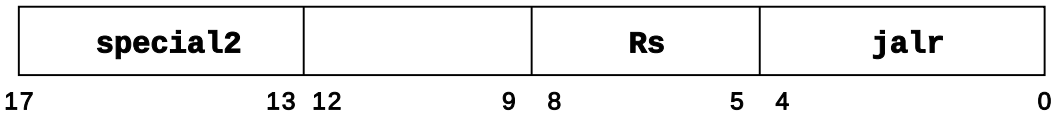
J-Typ

jalr jalr

jump and link register

Mnemonic

```
jalr        Rs
```



Pseudocode

```
R11 ← PC + 1
PC ← Rs
```

Delay Slots

1 unconditional delay slot

Description

Sets the program counter (PC) to the value of GPR Rs. The address of the instruction after the delay slot is written to GPR R11.

Comments

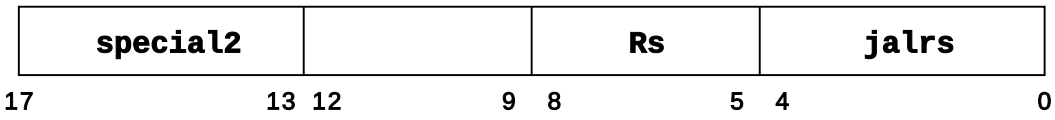
R-Typ

jalrs jalrs

jump and link and shift register window

Mnemonic

```
jalrs       Rs
```



Pseudocode

```
RegBase ← RegBase + 1
R11 ← PC + 1
PC ← Rs
```

Delay Slots

1 unconditional delay slot

Description

This instruction performs a shift of the register window for eight register positions. This is used for subroutine calls. The current PC is incremented and stored in R11 of the new register window. R11 is used to store the return address of the calling function. The value for RegBase which holds the current subroutine call level ($SFR\_STATUS_{6:0}$) is also incremented. Finally, the PC is set to the given address in GPR Rs.

Comments

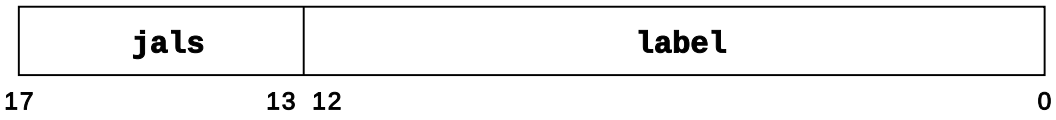
R-Typ

jals jals

jump and link and shift register window

Mnemonic

```
jals        displacement
```



Pseudocode

```
RegBase ← RegBase + 1
R11 ← PC + 1
PC ← PC + displacement
```

Delay Slots

1 unconditional delay slot

Description

This instruction performs a shift of the register window for eight register positions. This is used for subroutine calls. The current PC is incremented and stored in R11 of the new register window. R11 is used to store the return address of the calling function. The value for RegBase which holds the current function call level ($SFR\_STATUS_{6:0}$) is also incremented. Finally, the PC is set to the PC + displacement.

Comments

J-Typ

The subroutine must have at least one instruction and the return code `jrs R11` at its end.

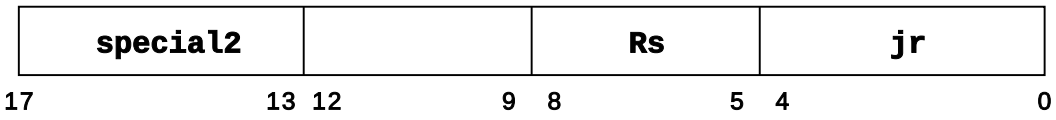
jr

jr

jump register

Mnemonic

```
jr          Rs
```



Pseudocode

```
PC ← Rs
```

Delay Slots

1 unconditional delay slot

Description

Set the PC unconditinally to the content of GPR Rs.

Comments

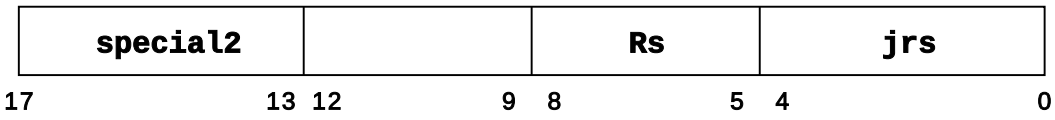
R-Typ

jrs jrs

jump register shift register window (return subroutine)

Mnemonic

```
jrs          Rs
```



Pseudocode

```
PC ← Rs
RegBase ← RegBase - 1
```

Delay Slots

1 unconditional delay slot

Description

This instruction performs the return from a subroutine by a back-shift of the register window for eight register positions. The program counter (PC) is set to the content of GPR RS.

Comments

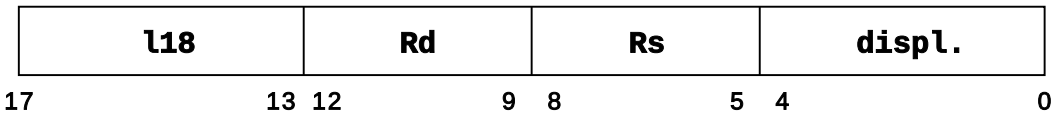
R-Typ

l18 l18

load 18 bit from memory

Mnemonic

```
l18         Rd, disp(Rs)
```



Pseudocode

```
Rd ← M[disp+Rs] ## M[disp+Rs+1]
```

Description

This instruction loads a sequence of two 9 bit words to an 18 bit register. The 5 bit displacement (disp) is zero-extended and added to the content of GPR Rs to form an unsigned 18 bit address. The 9 bit content of this address and the successor address is written to GPR Rd.

Comments

M-Typ

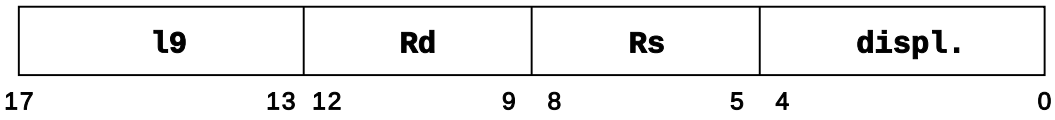
The given address must be even.

l9 l9

load 9 bit from memory

Mnemonic

```
l9          Rd, disp(Rs)
```



Pseudocode

$Rd \leftarrow 0^9 \,\#\#\, M[disp+Rs]$

Description

The 5 bit displacement (disp) is zero-extended and added to the content of GPR Rs to form an unsigned 18 bit address. The 9 Bit content of this address is written to GPR Rd.

Comments

M-Typ

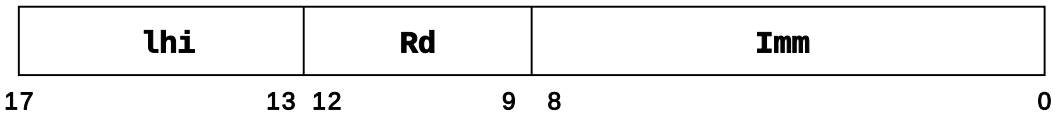
The given address can be even or odd.

lhi lhi

load high immediate

Mnemonic

```
lhi         Rd, Imm
```



Pseudocode

$Rd \leftarrow IR_{8:0} \,\#\#\, 0^{9}$

Description

This instruction writes the upper 9 bit part of GPR Rd. Therefore, the 9 bit immediate is concatenated with a 9 bit zero value and written to GPR Rd.

Comments

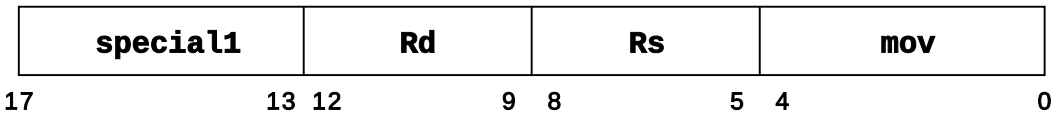
I-Typ

mov mov

move

Mnemonic

```
mov         Rd, Rs
```



Pseudocode

Rd ← Rs

Description

The content of GPR Rs is written to GPR Rd.

Comments

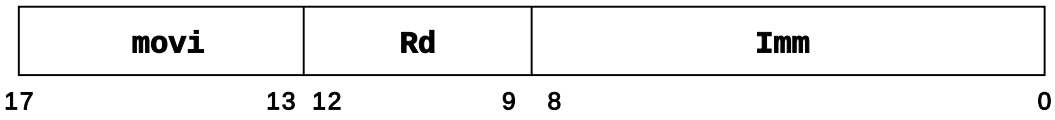
R-Typ

movi movi

move immediate

Mnemonic

```
movi        Rd, Imm
```



Pseudocode

$Rd \leftarrow 0^{9} \,\#\#\, IR_{8:0}$

Description

The content of a zero-extended 9 bit immediate is written to GPR Rd.

Comments

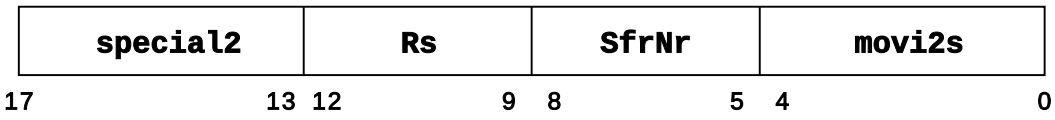
I-Typ

movi2s movi2s

move integer to special

Mnemonic

```
movi2s      SfrNr, Rs
```



Pseudocode

SFR ← Rs

Description

The content of GPR Rs is written to the SFR with the given SfrNr.

If the destination SFR is SFR_Status, a register window change will only take effect after the next instruction. This is the same behaviour as with delay slots in Function Calls, where the delay slot still uses the old register window.

Comments

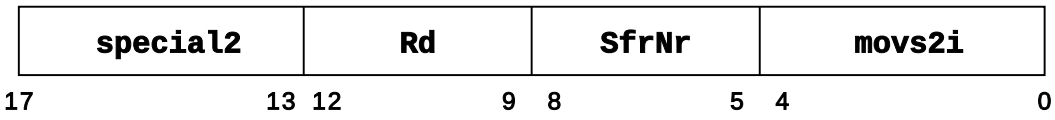
R-Typ

movs2i movs2i

move from special register to integer

Mnemonic

```
movs2i      Rd, SfrNr
```



Pseudocode

Rd ← SFR

Description

The content of the SFR with SfrNr is written to GPR Rd.

Comments

R-Typ

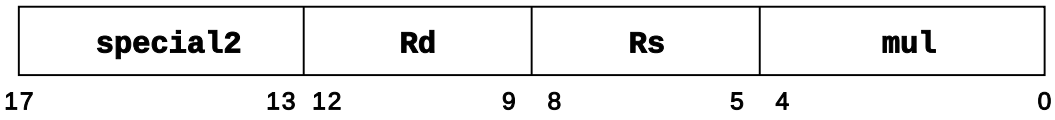
In this instruction code, $IR_{8:5}$holds the number of the SFR which is used as destination register for this instruction. The source register is given in $IR_{12:9}$.

mul mul

multiply

Mnemonic

```
mul         Rd, Rs
```



Pseudocode

```
SFR_MUL ## Rd ← Rd  *  Rs
```

Description

The content of GPR Rd and the content of GPR Rs are arithmetically multiplied, treating both operands as 18 bit two's complements values, and form a 36 bit two's complements result. The upper 18 bit part is written to SFR_MUL, the lower 18 bit part is written to GPR Rd.

Comments

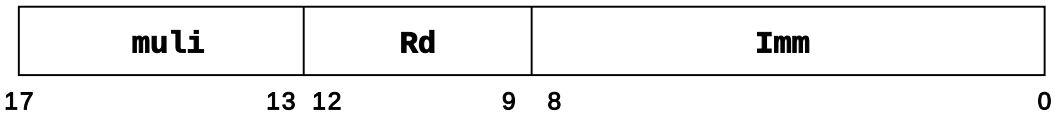
R-Typ

muli muli

multiply immediate

Mnemonic

```
muli        Rd, Imm
```



Pseudocode

$SFR\_MUL \,\#\#\, Rd \leftarrow Rd \times IR_8^9 \,\#\#\, IR_{8:0}$

Description

The sign-extended 9 bit immediate and the content of GPR Rd are arithmetically multiplied, treating both operants as 18 bit two's complement values, and form a 36-bit two's complement result. The upper 18 bit part is written to SFR_MUL, the lower 18 bit part is written to GPR Rd.

Comments

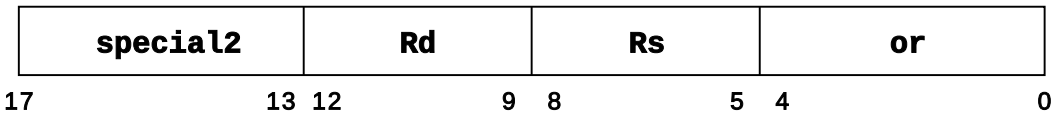
I-Typ

nop nop

no operation

Mnemonic

nop Rd



Pseudocode

Rd ← Rd or Rd

Description

Convenience Instruction. Is really an `or Rd, Rd` , but much more recognizable. Since there are 16 possible no-op combinations, this allows encoding extra information into the nop. This is useful only for debugging purposes, as it allows manipulating bypass logic and makes the instruction more distinguishable in assembler. Using no parameter defaults to 0.

Comments

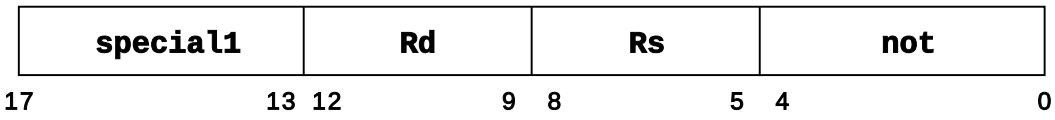
R-Typ

not not

not

Mnemonic

```
not         Rd, Rs
```



Pseudocode

```
Rd ← !Rs
```

Description

The content of GPR Rs is negated bitwise and the results is written to GPR Rd.

Comments

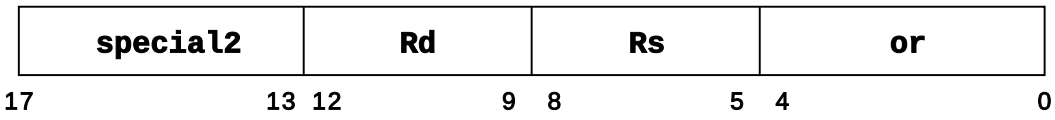
R-Typ

or or

or

Mnemonic

```
or          Rd, Rs
```



Pseudocode

```
Rd ← Rd or Rs
```

Description

The content of GPR Rs is combined with the content of GPR Rd in a bitwise logical OR operation, and the result is written to GPR Rd.

Comments

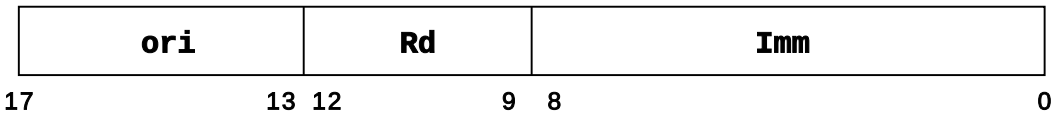
R-Typ

ori ori

or immediate

Mnemonic

```
ori         Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd \text{ or } 0^{9} \# \# IR_{8:0}$

Description

The zero-extended 9 bit immediate is combined with the content of GPR Rd in a bitwise OR operation, and the result is written to GPR Rd.

Comments

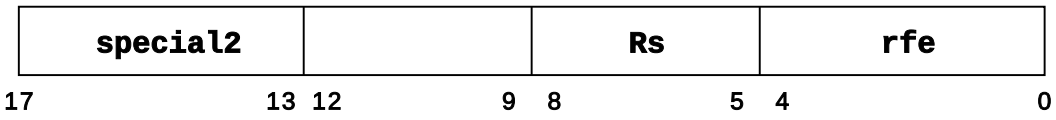
I-Typ

rfe rfe

return from exception

Mnemonic

```
rfe          Rs
```



Pseudocode

PC ← Rs

RegBase ← RegBase - 1

Delay Slots

1 unconditional delay slot

Description

This instruction performs the return from interrupt handling by a back-shift of the register window for eight register positions. The program counter (PC) is set to the content of GPR Rd and the interrupt is acknowledged.

This instruction is internally identical to `jsr` except that this will raise `ir_return` for a single cycle.

Comments

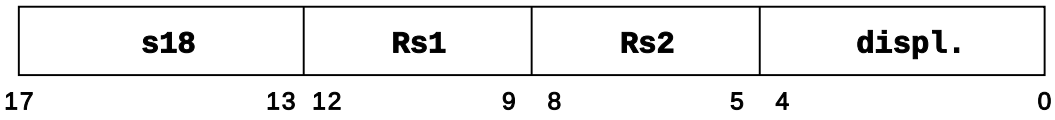
R-Typ

s18 s18

store 18 bit to memory

Mnemonic

```
s18         disp(Rs2), Rs1
```



Pseudocode

```
M[disp+Rs2] ## M[disp+Rs2+1] ← Rs1
```

Description

The zero-extended 5 bit displacement (disp) is added to the content of GPR Rs2 to form an unsigned 18 bit address. The content of GPR Rs1 is stored at this address.

Comments

M-Typ

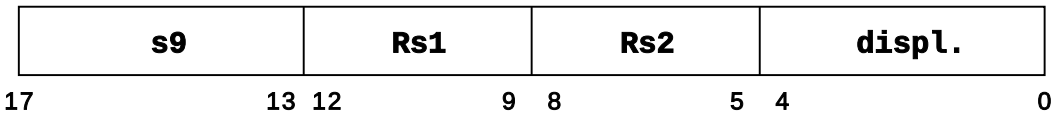
The 18 bit address must be even.

s9 s9

store 9 Bit to memory

Mnemonic

```
s9          disp(Rs2), Rs1
```



Pseudocode

```
M[disp+Rs2] ← Rs1
```

Description

The 5 bit displacement (disp) is zero-extended and added to the content of GPR Rs2 to form an unsigned 18 bit address. The lower 9 bit part of GPR Rs1 is stored at this address.

Comments

M-Typ

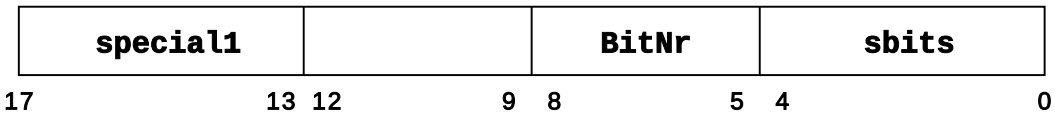
The given address can be even or odd.

sbits sbits

set bit at SFR-Register

Mnemonic

```
sbits       BitNr
```



Pseudocode

```
SFR_Status_Bit ← 1
BitNr.: 0 = sets SFR_CC
BitNr.: 1 = sets SFR_STATUS7(MM)
BitNr.: 2 = sets SFR_STATUS8(IE)
```

Description

This instruction sets a SFR bit according to the given BitNr. A BitNr of zero sets the CC bit to one, a BitNr of one sets the MM bit to one and a BitNr of two sets the IE bit to one.

Comments

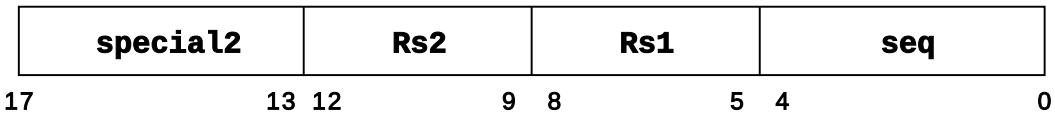
R-Typ

seq seq

set equal

Mnemonic

```
seq         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If both values are equal, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

R-Typ

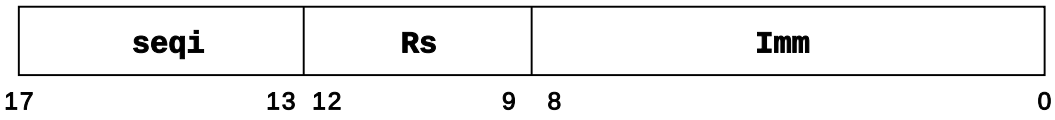
seqi

seqi

set equal immediate

Mnemonic

```
seqi        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \# \# IR_{8:0}$

Description

This instruction compares the content of GPR Rs and the 9 bit immediate $IR_8^9 \# \# IR_{8:0}$. If both values are equal, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

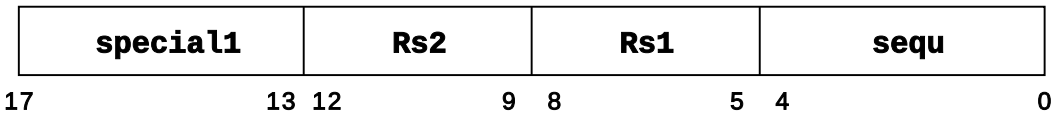
I-Typ

sequ sequ

set equal unsigned

Mnemonic

```
sequ        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If both values are equal, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{17}-1 and greater than or equal to zero.

Comments

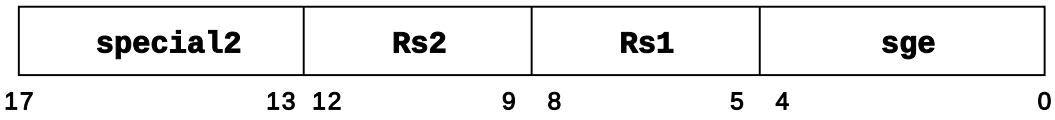
R-Typ

sge sge

set greater than or equal

Mnemonic

```
sge         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is equal to or greater than the value of Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

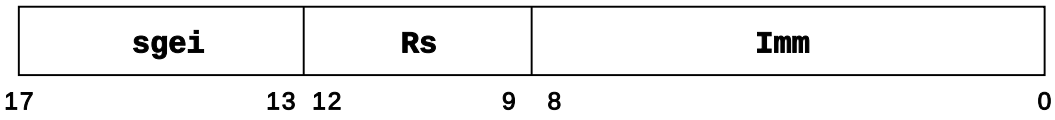
R-Typ

sgei sgei

set greater than or equal immediate

Mnemonic

```
sgei        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \,\#\#\, IR_{8:0}$

Description

This instruction compares the content of GPR Rs and the content of a 9 bit immediate. If the value of Rs is equal to or greater than the immediate, the result will be one, otherwise the reslt will be zero. The 18-bit result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

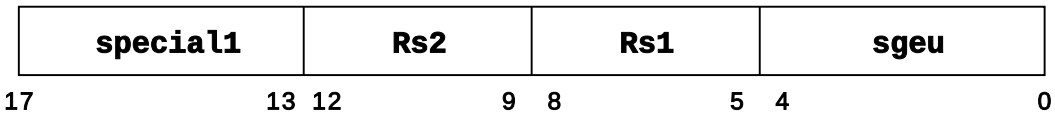
I-Typ

sgeu sgeu

set greater than or equal unsigned

Mnemonic

```
sgeu        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is equal to or greater than the value of Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{18}-1 and greater than or equal to zero.

Comments

R-Typ

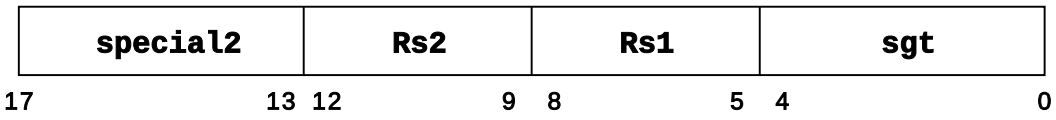
In this instruction -1 = 0x3FFFF is bigger than Ox1FFFF.

sgt sgt

set greater than

Mnemonic

```
sgt         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of GPR Rs2 is greater than the value of GPR Rs1, the result will be one, otherwise, the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than 2^{17}-1 and greater than -2^{17}.

Comments

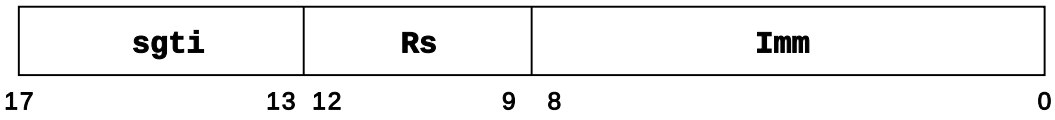
R-Typ

sgti sgti

set greater than immediate

Mnemonic

```
sgti        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \,\#\#\, IR_{8:0}$

Description

This instruction compares the content of GPR Rs and a 9 bit immediate. If the value of GPR Rs is greater than the immediate, the result will be one, otherwise the result will be zero. This result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

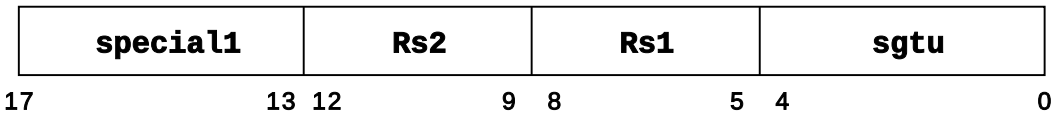
I-Typ

sgtu sgtu

set greather than unsigned

Mnemonic

```
sgtu        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is greater than the value of Rs1, the result will be one, otherwise, the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{18}-1 and greater than or equal to zero.

Comments

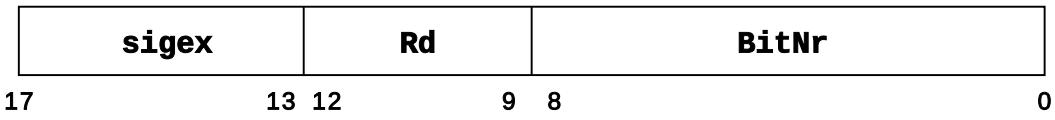
R-Typ

sigex sigex

signum extention

Mnemonic

```
sigex       Rd, BitNr
```



Pseudocode

$$Rd \leftarrow Rd_{BitNr-1}^{17-BitNr-1} \,\#\#\, Rd_{(BitNr-1):0}$$

Description

This instruction expands the content of Rd to an 18 bit value using the value of Rd at the given bit number (BitNr).

Comments

I-Typ

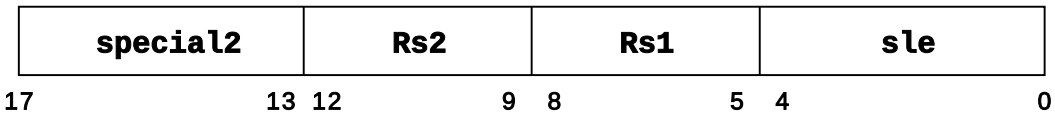
The allowed values for BitNr are 8, 9 or 16. Other values will be treated as 8.

sle sle

set less than or equal

Mnemonic

```
sle         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is equal to or less than the value of Rs2, the result will be one, otherwise, the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

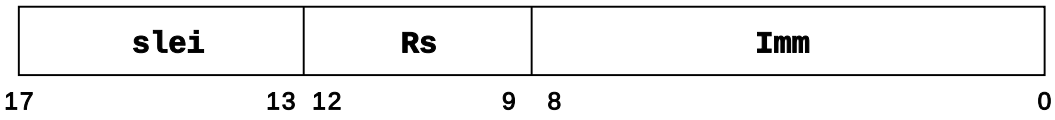
R-Typ

slei slei

set less than or equal immediate

Mnemonic

```
slei        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \,\#\#\, IR_{8:0}$

Description

This instruction compares the content of GPR Rs and a 9 bit immediate. If the value of Rs is equal to or less than the immediate, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

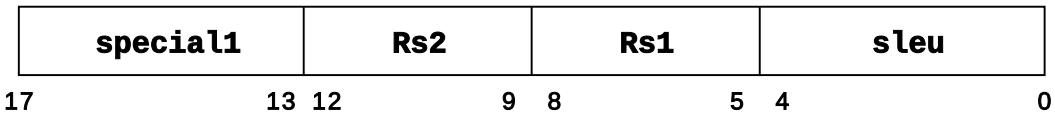
I-Typ

sleu sleu

set less than or equal unsigned

Mnemonic

```
sleu        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is equal to or less than the value of Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{18}-1 and greater than or equal to zero.

Comments

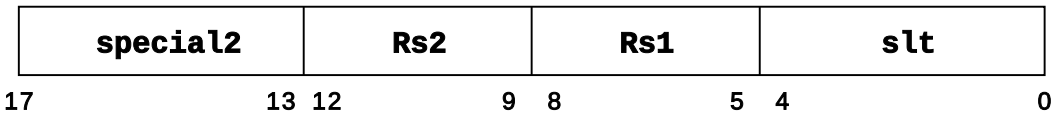
R-Typ

slt slt

set less than

Mnemonic

```
slt         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of GPR Rs2 is less than the value of GPR Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than 2^{17}-1 and greater than -2^{17}.

Comments

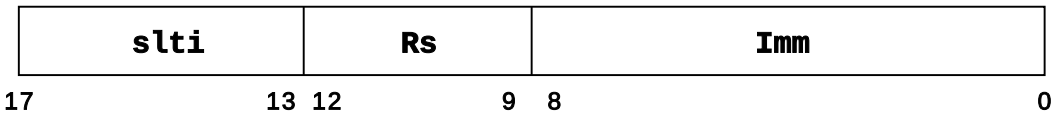
R-Typ

slti slti

set less than immediate

Mnemonic

```
slti        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \,\#\#\, IR_{8:0}$

Description

This instruction compares the content of GPR Rs and a 9 bit immediate. If the value of Rs is less than the immediate, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

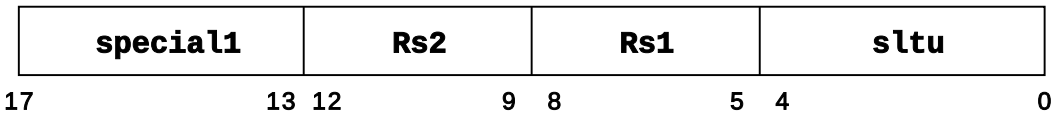
I-Typ

sltu sltu

set less than unsigned

Mnemonic

```
sltu        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is less than the value of Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{18}-1 and greater than or equal to zero.

Comments

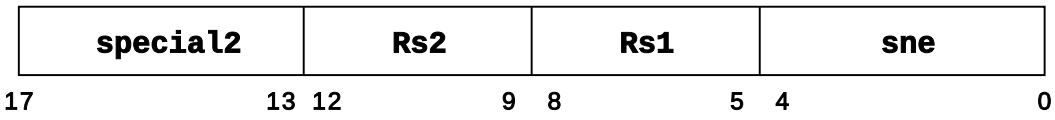
R-Typ

sne sne

set not equal

Mnemonic

```
sne         Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of GPR Rs2 is lower or greater than the value of GPR Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than 2^{17}-1 and greater than -2^{17}.

Comments

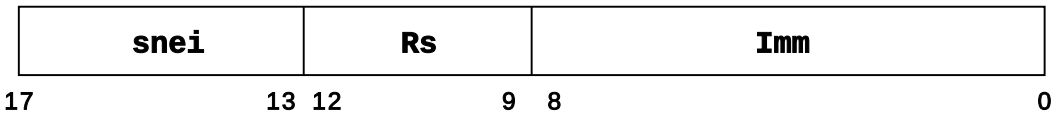
R-Typ

snei snei

set not equal immediate

Mnemonic

```
snei        Rs, Imm
```



Pseudocode

$CC \leftarrow Rs - IR_8^9 \,\#\#\, IR_{8:0}$

Description

This instruction compares the content of GPR Rs and the content of 9 bit immediate. If the value of Rs is greater or lower than the immediate, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The content of GPR Rs is lower than or equal to 2^{17}-1 and greater than or equal to -2^{17}.

Comments

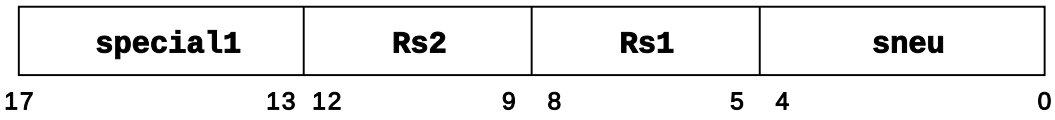
I-Typ

sneu sneu

set not equal unsigned

Mnemonic

```
sneu        Rs2, Rs1
```



Pseudocode

```
CC ← Rs2 - Rs1
```

Description

This instruction compares the content of GPR Rs2 and GPR Rs1. If the value of Rs2 is lower or greater than the value of Rs1, the result will be one, otherwise the result will be zero. The result is written to SFR_CC. The contents of GPR Rs2 and GPR Rs1 are lower than or equal to 2^{18}-1 and greater than or equal to zero.

Comments

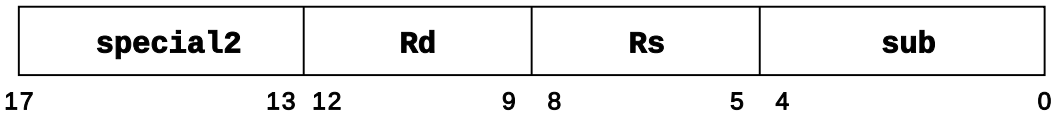
R-Typ

sub sub

subtract

Mnemonic

```
sub         Rd, Rs
```



Pseudocode

```
Rd ← Rd  -  Rs
cc ← ov
```

Description

The content of GPR Rs is arithmetically subtracted from the content of GPR Rd and forms an 18 bit two's complement result, which is written to GPR Rd. If the result of the subtraction is greater than 2^{17}-1 (i.e.: = 0x1FFFF) or lower than -2^{17}(i.e.: 0x20000), an overflow occurs and CC is set to 1.

Comments

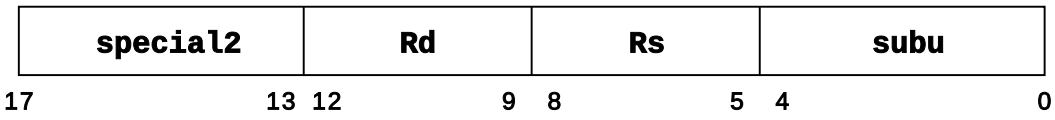
R-Typ

subu subu

subtract unsigned

Mnemonic

subu Rd, Rs



Pseudocode

Rd ← Rd - Rs

Description

The content of GPR Rs is arithmetically subtracted from the content of GPR Rd and forms an 18 bit unsigned result which is written to GPR Rd. This instruction can not produce an overflow exception, which is the only difference between this instruction and the sub instruction.

Comments

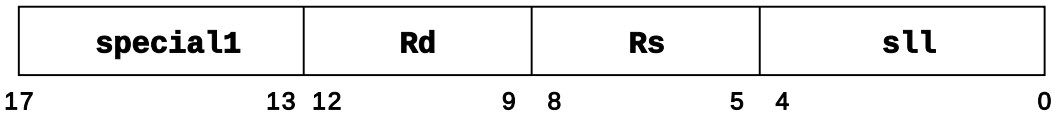
R-Typ

sll sll

shift left logical

Mnemonic

```
sll         Rd, Rs
```



Pseudocode

```
Rd ← CC ## Rd « Rs
```

Description

This instruction performs a left shift operation of GPR Rd. The shift width is set to the value of GPR Rs. The free bit positions are filled with zeros. The value of the highest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

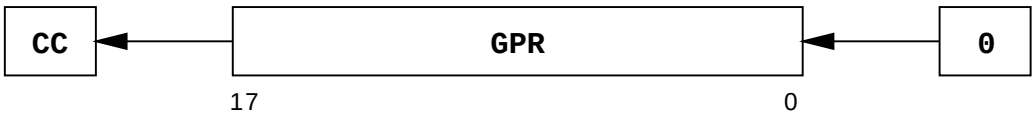


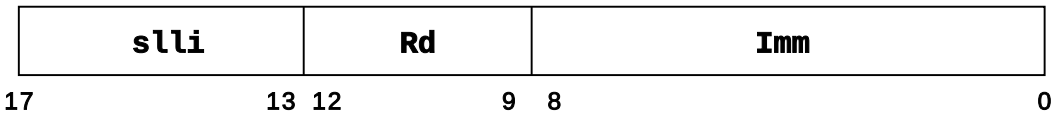
Figure 13: shift left logical

slli slli

shift left logical immediate

Mnemonic

```
slli        Rd, Imm
```



Pseudocode

$Rd \leftarrow (CC \mathbin{\#\#} Rd) \ll (0^{9} \mathbin{\#\#} IR_{8:0})$

Description

This instruction performs a left shift operation of GPR Rd. The shift width is set by a zero-extended 9 bit immediate. The free bit positions are filled with zero. The value of the highest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

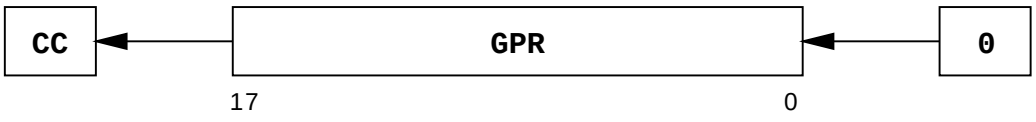


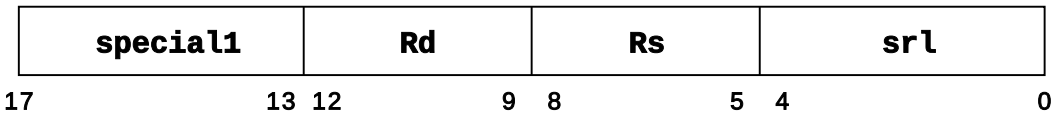
Figure 14: shift left logical immediate

srl srl

shift right logical

Mnemonic

```
srl         Rd, Rs
```



Pseudocode

```
Rd ← (Rd ## CC) » Rs
```

Description

This instruction performs a right shift operation of GPR Rd. The shift width is set to the value of GPR Rs. The free bit positions are filled with zeros. The value of the lowest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

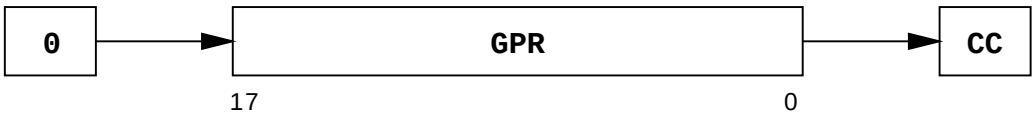


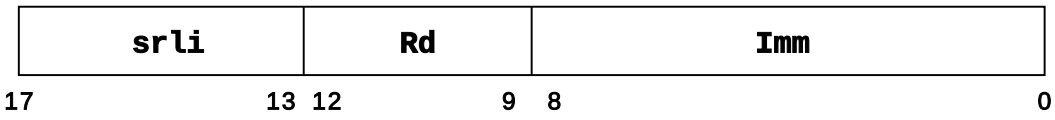
Figure 15: shift right logical

srli srli

shift right logical immediate

Mnemonic

```
srli        Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd \mathbin{\#\#} CC \gg 0^{9} \mathbin{\#\#} IR_{8:0}$

Description

This instruction performs a right shift operation of GPR Rd. The shift width is set by a zero-extended 9 bit immediate. The free bit positions are filled with zero. The value of the lowest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

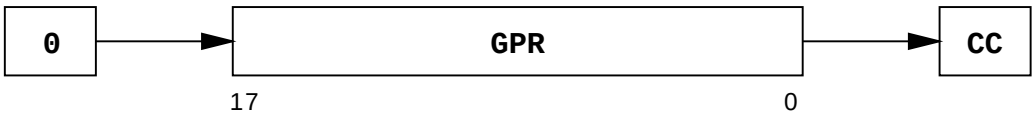


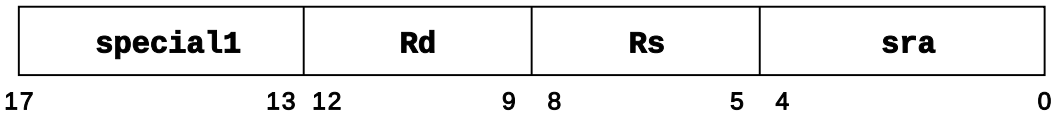
Figure 16: shift right logical immediate

sra sra

shift right arithmetic

Mnemonic

```
sra         Rd, Rs
```



Pseudocode

$Rd \leftarrow (Rd \mathbin{\#\#} CC) \gg_a Rs$

Description

This instruction performs a right shift operation of GPR Rd. The shift width is set to the value of GPR Rs. The free bit positions are filled with the highest bit of GPR Rd. The value of the lowest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

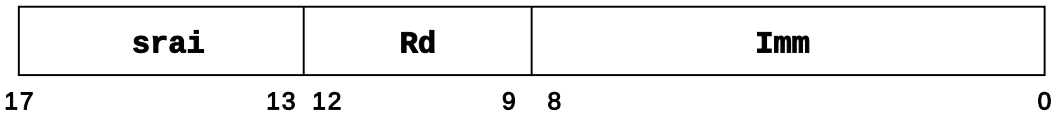
Figure 17: shift right arithmetic

srai srai

shift right arithmetic immediate

Mnemonic

```
srai        Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd \mathbin{\#\#} CC \gg_a 0^9 \mathbin{\#\#} IR_{8:0}$

Description

This instruction performs a right shift operation of GPR Rd. The shift width is set by a zero-extended 9 bit immediate. The free bit positions are filled with the highest bit of GPR Rd. The value of the lowest bit in GPR Rd is written to SFR_CC. The result is written to GPR Rd.

Comments

R-Typ

The value in GPR Rs will be ignored and the shift width will be always one if single shift is configured.

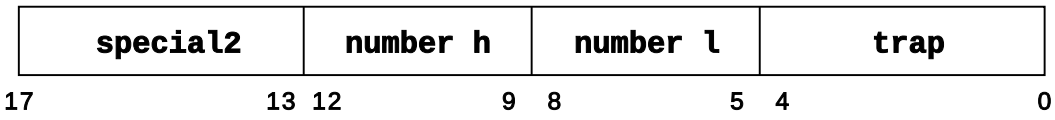
Figure 18: shift right arithmetic immediate

trap trap

trap

Mnemonic

trap Number



Pseudocode

RegBase ← RegBase + 1

R11 ← PC + 1

PC ← SFR_TR$_{17:8}$ ## Number

Description

This instruction performs a shift of the register window for eight register positions. The current PC is incremented and stored in R11 of the new register window. R11 is used to store the return address of the calling function. The value for RegBase holding the current subroutine call level (SFR_STATUS$_{6:0}$) is also incremented. Finally, the PC is set to the 10 bit value of SFR_TR$_{17:8}$ and to the 8 bit value of number (number h ## number l).

Comments

R-Typ

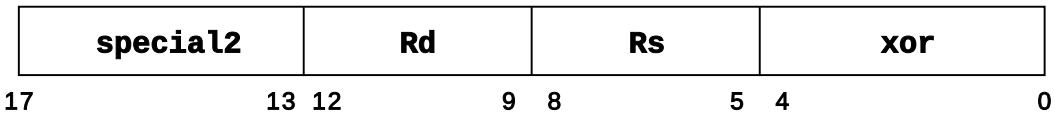
number = number h ## number l

xor xor

exclusive or

Mnemonic

```
xor         Rd, Rs
```



Pseudocode

```
Rd ← Rd xor Rs
```

Description

The content of GPR Rd is combined with the content of GPR Rs in a bitwise logical XOR operation, and the result is written to GPR Rd.

Comments

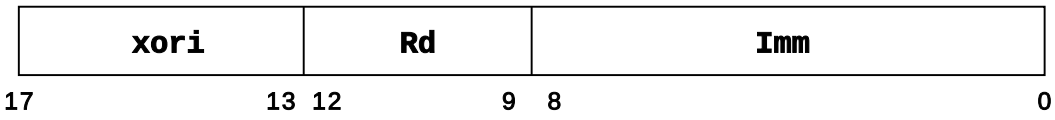
R-Typ

xori xori

exclusvie or immediate

Mnemonic

```
xori        Rd, Imm
```



Pseudocode

$Rd \leftarrow Rd \text{ xor } 0^{9} \# \# IR_{8:0}$

Description

The zero-extended 9 bit immediate is combined with the content of GPR Rd in a bitwise logical XOR operation, and the result is written to GPR Rd.

Comments

I-Typ