

SRP16 ISA Specification

v0.1.0

Vishnu Shankar B

Contents

| | |
|---|----|
| Ch 1. Architecture Overview..... | 4 |
| 1.1. SRP16 Features..... | 4 |
| 1.2. General Purpose Registers..... | 4 |
| 1.3. Accumulator Register..... | 4 |
| 1.4. Flag Bit..... | 4 |
| 1.5. Memory Pointer Register or MPTR..... | 4 |
| 1.6. Stack Pointer..... | 4 |
| 1.7. Program Counter..... | 4 |
| 1.8. Memory..... | 4 |
| Ch 2. Instruction Set..... | 5 |
| 2.1. Instruction Encoding Types..... | 5 |
| 2.2. Register IDs..... | 5 |
| 2.3. Immediate Values..... | 5 |
| 2.4. E-type Load Instructions..... | 6 |
| 2.4.1. LDR Instruction..... | 6 |
| 2.4.2. LDRU Instruction..... | 6 |
| 2.4.3. LD@MPTR Instruction..... | 6 |
| 2.4.4. ST@MPTR Instruction..... | 6 |
| 2.4.5. LDB@MPTR Instruction..... | 6 |
| 2.4.6. STB@MPTR Instruction..... | 7 |
| 2.5. T-type Load Instructions..... | 7 |
| 2.5.1. LDA Instruction..... | 7 |
| 2.5.2. LDMPTR Instruction..... | 7 |
| 2.5.3. LDMPTRU Instruction..... | 7 |
| 2.6. R-type Load Instruction..... | 8 |
| 2.6.1. LDAU Instruction..... | 8 |
| 2.7. R-type Move Instructions..... | 8 |
| 2.7.1. MOV Instruction..... | 8 |
| 2.7.2. MOV Instruction (from PC)..... | 8 |
| 2.7.3. JMP Instruction..... | 8 |
| 2.8. T-type Jump Instructions..... | 9 |
| 2.8.1. SJMP Instruction..... | 9 |
| 2.8.2. SJMPF Instruction..... | 9 |
| 2.9. R-type Stack Instructions..... | 9 |
| 2.9.1. POP Instruction..... | 9 |
| 2.9.2. PUSH Instruction..... | 10 |
| 2.10. R-type Increment/Decrement Instructions..... | 10 |
| 2.10.1. INC Instruction..... | 10 |
| 2.10.2. DEC Instruction..... | 10 |
| 2.11. E-type Immediate Arithmetic Instructions..... | 11 |
| 2.11.1. ADDI Instruction..... | 11 |
| 2.11.2. ADCI Instruction..... | 11 |
| 2.11.3. SBBI Instruction..... | 11 |
| 2.11.4. ANDI Instruction..... | 11 |
| 2.11.5. ORI Instruction..... | 11 |
| 2.11.6. XORI Instruction..... | 12 |
| 2.12. R-type Immediate Arithmetic Instructions..... | 12 |

| | |
|--|----|
| 2.12.1. SLAI Instruction..... | 12 |
| 2.12.2. SRAI Instruction..... | 12 |
| 2.12.3. SLLI Instruction..... | 12 |
| 2.12.4. SRLI Instruction..... | 12 |
| 2.13. R-type Arithmetic Instructions..... | 13 |
| 2.13.1. ADD Instruction..... | 13 |
| 2.13.2. SUB Instruction..... | 13 |
| 2.13.3. ADC Instruction..... | 13 |
| 2.13.4. SBB Instruction..... | 13 |
| 2.13.5. AND Instruction..... | 13 |
| 2.13.6. OR Instruction..... | 14 |
| 2.13.7. XOR Instruction..... | 14 |
| 2.13.8. SLA Instruction..... | 14 |
| 2.13.9. SRA Instruction..... | 14 |
| 2.13.10. SLL Instruction..... | 14 |
| 2.13.11. SRL Instruction..... | 14 |
| 2.14. E-type Compare Immediate Instructions..... | 15 |
| 2.14.1. CLI Instruction..... | 15 |
| 2.14.2. CGI Instruction..... | 15 |
| 2.14.3. CEI Instruction..... | 15 |
| 2.15. R-type Compare Instructions..... | 16 |
| 2.15.1. CL Instruction..... | 16 |
| 2.15.2. CG Instruction..... | 16 |
| 2.15.3. CE Instruction..... | 16 |
| 2.16. R-type Flag Instruction..... | 17 |
| 2.16.1. NOTF Instruction..... | 17 |
| 2.17. SRP16 Opcode Table..... | 18 |
| 2.18. SRP16 Instruction Set Quick Reference..... | 20 |

Ch 1. Architecture Overview

1.1. SRP16 Features

- 16-bit RISC processor
- 32 general purpose registers
- Up to 64 KiB of memory
- Fixed length instruction width (2-bytes)
- Little endian

1.2. General Purpose Registers

There are 32 general purpose registers, R0-R31. Only R0-R15 can be accessed by Load instructions.

1.3. Accumulator Register

The Accumulator register is used to accumulate/store results of arithmetic and logic instructions.

1.4. Flag Bit

The Flag Bit is used by compare instructions. After execution of a compare instruction, the flag bit will be set if the comparison is true or cleared otherwise.

1.5. Memory Pointer Register or MPTR

SRP16 is a load-store architecture, any instruction that reads from or writes to memory (except for instructions that deal with the stack) will use this register as pointer to a location in memory.

1.6. Stack Pointer

Holds the address of top of the stack. It decrements when you push an item onto the stack and increments when you pop an item off the stack.

1.7. Program Counter

The Program Counter holds the address of the current instruction being fetched/executed.

1.8. Memory

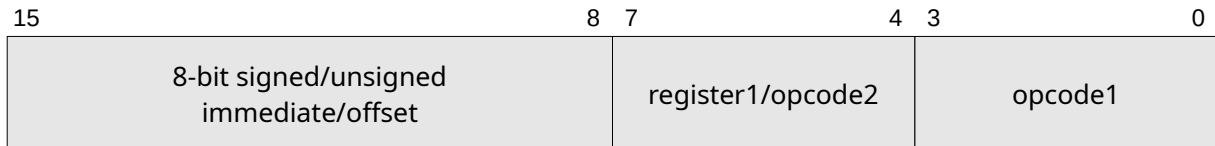
SRP16 can access up to 64 KiB of memory. This includes program memory, code memory and the stack.

Ch 2. Instruction Set

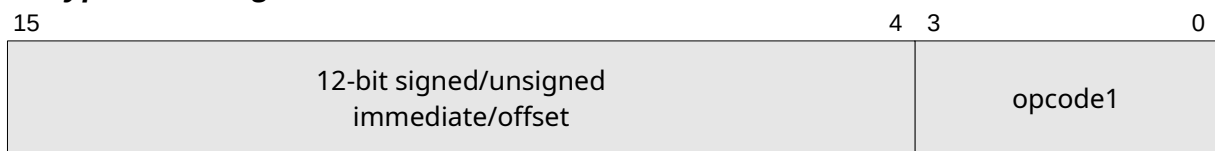
2.1. Instruction Encoding Types

SRP16 has three Instruction Encoding types as shown in Fig 1. All are 2 bytes in length.

E-type encoding:



T-type encoding:



R-type encoding:

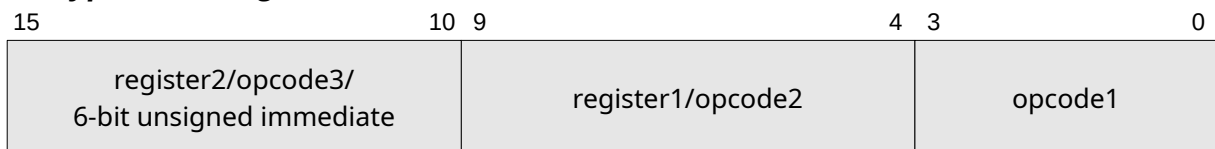


Fig 1: SRP16 Instruction Encoding Types

2.2. Register IDs

Each register has been given a unique 6-bit ID that will be used in the instruction encoding to specify which register to use as the operand.

- R0-R15: 000000 to 001111
- R16-R31: 010000 to 011111
- Accumulator (R60): 111100
- MPTR (R61): 111101
- SP (R62): 111110
- PC (R63): 111111

E-type instructions can only access R0-R15 since it has only four bits for the register field.

2.3. Immediate Values

All 8-bit or 12-bit signed immediate values are sign extended to 16-bits. All 8-bit or 12-bit unsigned immediate values are zero extended to 16-bits.

2.4. E-type Load Instructions

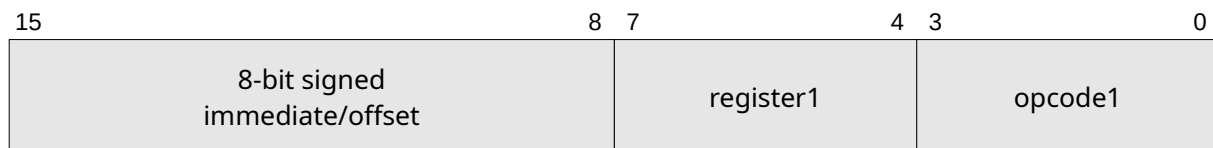


Fig 2: E-type Load Instructions' Encoding

2.4.1. LDR Instruction

- Description: Loads 8-bit signed immediate value to register
- Syntax: LDR Rx, 8-bit-signed-immediate
- Operation: $Rx \leftarrow \text{immediate}$
- Opcode1: 0000

2.4.2. LDRU Instruction

- Description: Loads upper 8-bits of register
- Syntax: LDRU Rx, 8-bit-unsigned-immediate
- Operation: $Rx[15:8] \leftarrow \text{immediate}$
- Opcode1: 0001

2.4.3. LD@MPTR Instruction

- Description: Loads 16-bit value from memory to register
- Syntax: LD@MPTR Rx, 8-bit-signed-offset
- Operation: $Rx \leftarrow \text{memory}[MPTR], MPTR \leftarrow MPTR + \text{offset}$
- Opcode1: 0010

2.4.4. ST@MPTR Instruction

- Description: Stores contents of register into memory
- Syntax: ST@MPTR Rx, 8-bit-signed-offset
- Operation: $\text{memory}[MPTR] \leftarrow Rx, MPTR \leftarrow MPTR + \text{offset}$
- Opcode1: 0011

2.4.5. LDB@MPTR Instruction

- Description: Loads a byte from memory to lower 8-bits of register
- Syntax: LDB@MPTR Rx, 8-bit-signed-offset
- Operation: $Rx[7:0] \leftarrow \text{memory}[MPTR], MPTR \leftarrow MPTR + \text{offset}$
- Opcode1: 0100

2.4.6. STB@MPTR Instruction

- Description: Stores lower 8-bits of register into memory
- Syntax: STB@MPTR Rx, 8-bit-signed-offset
- Operation: $\text{memory}[\text{MPTR}] \leftarrow \text{Rx}[7:0], \text{MPTR} \leftarrow \text{MPTR} + \text{offset}$
- Opcode1: 0101

2.5. T-type Load Instructions

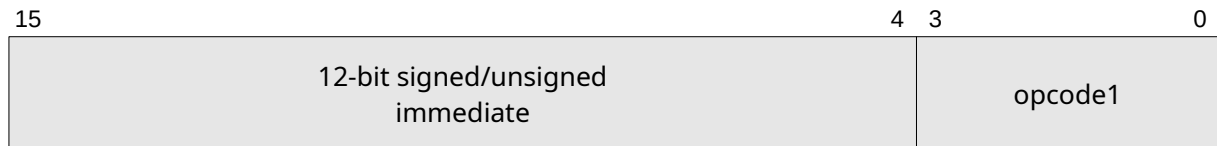


Fig 3: T-type Load Instructions' Encoding

2.5.1. LDA Instruction

- Description: Loads 12-bit signed immediate value to accumulator
- Syntax: LDA 12-bit-signed-immediate
- Operation: $A \leftarrow \text{immediate}$
- Opcode1: 0110

2.5.2. LDMPTR Instruction

- Description: Loads 12-bit unsigned immediate to MPTR
- Syntax: LDMPTR 12-bit-unsigned-immediate
- Operation: $\text{MPTR} \leftarrow \text{immediate}$
- Opcode1: 0111

2.5.3. LDMPTRU Instruction

- Description: Loads upper 4-bits of MPTR register
- Syntax: LDMPTRU 12-bit-unsigned-immediate
- Operation: $\text{MPTR}[15:12] \leftarrow \text{immediate}[3:0]$
- Opcode1: 1000

2.6. R-type Load Instruction

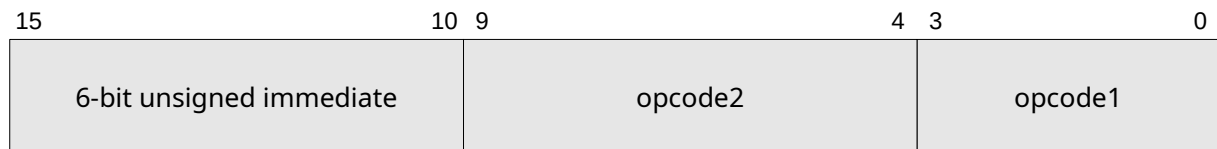


Fig 4: R-type Load Instruction's Encoding

2.6.1. LDAU Instruction

- Description: Loads upper 4-bits of accumulator
- Syntax: LDAU 6-bit-unsigned-immediate
- Operation: $A[15:12] \leftarrow \text{immediate}[3:0]$
- Opcode1: 1100
- Opcode2: 111011

2.7. R-type Move Instructions

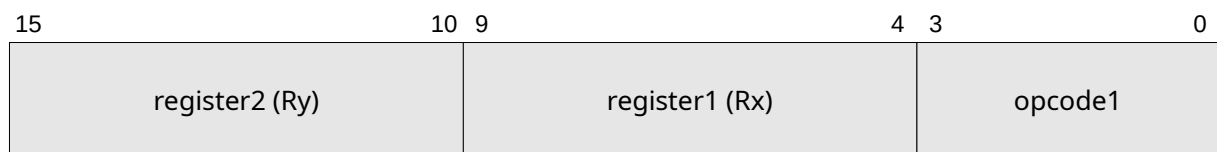


Fig 5: R-type Move Instructions' Encoding

2.7.1. MOV Instruction

- Description: Copies contents of source register to destination register
- Syntax: MOV Rx, Ry
- Operation: $Rx \leftarrow Ry$
- Opcode1: 1001

2.7.2. MOV Instruction (from PC)

- Description: Copies PC+4 to destination register
- Syntax: MOV Rx, PC
- Operation: $Rx \leftarrow PC+4$
- Opcode1: 1001
- register2: 111111 (PC register ID)

2.7.3. JMP Instruction

- Description: Branches to address specified by register
- Syntax: JMP Ry or MOV PC, Ry
- Operation: $PC \leftarrow Ry$
- Opcode1: 1001
- register1: 111111 (PC register ID)

2.8. T-type Jump Instructions



Fig 6: T-type Jump Instructions' Encoding

2.8.1. SJMP Instruction

- Description: Branches relative to PC
- Syntax: SJMP 12-bit-signed-offset
- Operation: $PC \leftarrow PC + \text{offset}$
- Opcode1: 1010

2.8.2. SJMPF Instruction

- Description: Branches relative to PC if flag is set
- Syntax: SJMPF 12-bit-signed-offset
- Operation: if(flag): $PC \leftarrow PC + \text{offset}$
- Opcode1: 1011

2.9. R-type Stack Instructions



Fig 7: R-type Stack Instructions' Encoding

2.9.1. POP Instruction

- Description: Pops an item off the stack and copies it to register.
- Constraints: Can only access general purpose registers R0-R31.
- Syntax: POP Rx
- Operation: $Rx \leftarrow \text{memory}[SP], SP \leftarrow SP + 2$
- Opcode1: 1100
- Opcode2: 111100

2.9.2. PUSH Instruction

- Description: Pushes contents of register to stack
- Constraints: Can only access general purpose registers R0-R31.
- Syntax: PUSH Rx
- Operation: $SP \leftarrow SP-2$, $memory[SP] \leftarrow Rx$
- Opcode1: 1100
- Opcode2: 111101

2.10. R-type Increment/Decrement Instructions

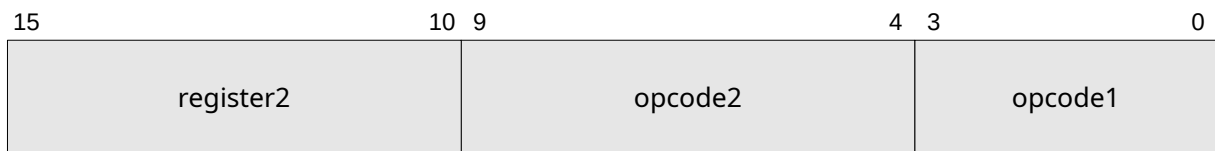


Fig 8: R-type Increment/Decrement Instructions' Encoding

2.10.1. INC Instruction

- Description: Increments contents of register
- Constraints: Can only access general purpose registers R0-R31.
- Syntax: INC Rx
- Operation: $Rx \leftarrow Rx+1$
- Opcode1: 1100
- Opcode2: 111110

2.10.2. DEC Instruction

- Description: Decrements contents of register
- Constraints: Can only access general purpose registers R0-R31.
- Syntax: DEC Rx
- Operation: $Rx \leftarrow Rx-1$
- Opcode1: 1100
- Opcode2: 111111

2.11. E-type Immediate Arithmetic Instructions

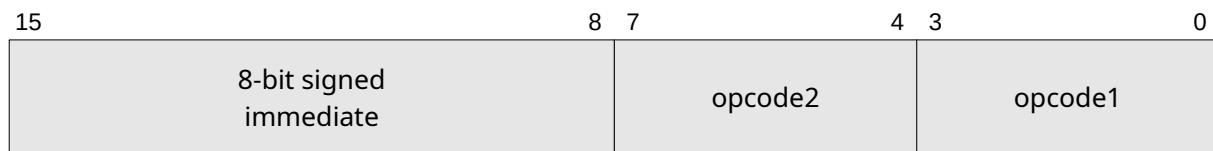


Fig 9: E-type Arithmetic Immediate Instructions' Encoding

2.11.1. ADDI Instruction

- Description: Adds 8-bit signed immediate value to accumulator
- Syntax: ADDI 8-bit-signed-immediate
- Operation: $A \leftarrow A + \text{immediate}$
- Opcode1: 1101
- Opcode2: 0001

2.11.2. ADCI Instruction

- Description: Adds 8-bit signed immediate value to accumulator with carry
- Syntax: ADCI 8-bit-signed-immediate
- Operation: $A \leftarrow A + \text{immediate} + \text{carry}$
- Opcode1: 1101
- Opcode2: 1101

2.11.3. SBBI Instruction

- Description: Subtracts 8-bit signed immediate value from accumulator with borrow
- Syntax: SBBI 8-bit-signed-immediate
- Operation: $A \leftarrow A - \text{immediate} - \text{carry}$
- Opcode1: 1101
- Opcode2: 1110

2.11.4. ANDI Instruction

- Description: Performs bitwise AND on accumulator and 8-bit signed immediate value
- Syntax: ANDI 8-bit-signed-immediate
- Operation: $A \leftarrow A \& \text{immediate}$
- Opcode1: 1101
- Opcode 2: 0111

2.11.5. ORI Instruction

- Description: Performs bitwise OR on accumulator and 8-bit signed immediate value
- Syntax: ORI 8-bit-signed-immediate
- Operation: $A \leftarrow A | \text{immediate}$
- Opcode1: 1101
- Opcode 2: 1000

2.11.6. XORI Instruction

- Description: Performs bitwise XOR on accumulator and 8-bit signed immediate value
- Syntax: XORI 8-bit-signed-immediate
- Operation: $A \leftarrow A \wedge \text{immediate}$
- Opcode1: 1101
- Opcode 2: 1001

2.12. R-type Immediate Arithmetic Instructions

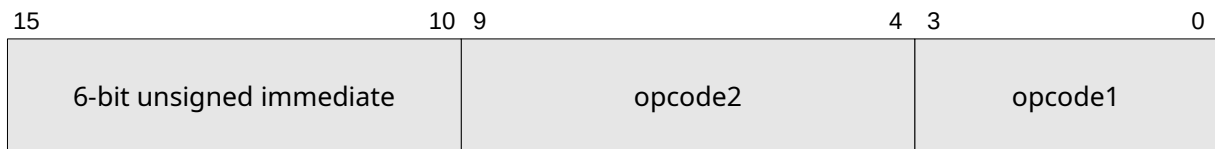


Fig 10: R-type Immediate Arithmetic Instructions' Encoding

2.12.1. SLAI Instruction

- Description: Performs arithmetic left shift on accumulator
- Syntax: SLAI 6-bit-unsigned-immediate
- Operation: $A \leftarrow A \ll \text{immediate}$
- Opcode1: 1100
- Opcode2: 100011

2.12.2. SRAI Instruction

- Description: Performs arithmetic right shift on accumulator
- Syntax: SRAI 6-bit-unsigned-immediate
- Operation: $A \leftarrow A \gg \text{immediate}$
- Opcode1: 1100
- Opcode2: 100100

2.12.3. SLLI Instruction

- Description: Performs logical left shift on accumulator
- Syntax: SLLI 6-bit-unsigned-immediate
- Operation: $A \leftarrow A \ll \text{immediate}$
- Opcode1: 1100
- Opcode2: 100101

2.12.4. SRLI Instruction

- Description: Performs logical right shift on accumulator
- Syntax: SRLI 6-bit-unsigned-immediate
- Operation: $A \leftarrow A \gg \text{immediate}$
- Opcode1: 1100
- Opcode2: 100110

2.13. R-type Arithmetic Instructions

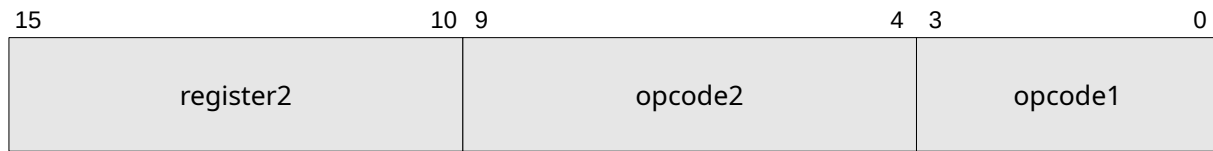


Fig 11: R-type Arithmetic Instructions' Encoding

2.13.1. ADD Instruction

- Description: Adds contents of register to accumulator
- Syntax: ADD Rx
- Operation: $A \leftarrow A + Rx$
- Opcode1: 1100
- Opcode2: 000001

2.13.2. SUB Instruction

- Description: Subtracts contents of register from accumulator
- Syntax: SUB Rx
- Operation: $A \leftarrow A - Rx$
- Opcode1: 1100
- Opcode2: 000010

2.13.3. ADC Instruction

- Description: Adds contents of register to accumulator with carry
- Syntax: ADC Rx
- Operation: $A \leftarrow A + Rx + \text{carry}$
- Opcode1: 1100
- Opcode2: 001101

2.13.4. SBB Instruction

- Description: Subtracts contents of register from accumulator with borrow
- Syntax: SBB Rx
- Operation: $A \leftarrow A - Rx - \text{carry}$
- Opcode1: 1100
- Opcode2: 001110

2.13.5. AND Instruction

- Description: Performs bitwise AND on accumulator and register
- Syntax: AND Rx
- Operation: $A \leftarrow A \& Rx$
- Opcode1: 1100
- Opcode 2: 000111

2.13.6. OR Instruction

- Description: Performs bitwise OR on accumulator and register
- Syntax: OR Rx
- Operation: $A \leftarrow A | Rx$
- Opcode1: 1100
- Opcode 2: 001000

2.13.7. XOR Instruction

- Description: Performs bitwise XOR on accumulator and register
- Syntax: XOR Rx
- Operation: $A \leftarrow A \wedge Rx$
- Opcode1: 1100
- Opcode 2: 001001

2.13.8. SLA Instruction

- Description: Performs arithmetic left shift by shift amount held in register
- Syntax: SLA Rx
- Operation: $A \leftarrow A \ll Rx$
- Opcode1: 1100
- Opcode2: 000011

2.13.9. SRA Instruction

- Description: Performs arithmetic right shift by shift amount held in register
- Syntax: SRA Rx
- Operation: $A \leftarrow A \gg Rx$
- Opcode1: 1100
- Opcode2: 000100

2.13.10. SLL Instruction

- Description: Performs logical left shift by shift amount held in register
- Syntax: SLL Rx
- Operation: $A \leftarrow A \ll Rx$
- Opcode1: 1100
- Opcode2: 000101

2.13.11. SRL Instruction

- Description: Performs logical right shift by shift amount held in register
- Syntax: SRL Rx
- Operation: $A \leftarrow A \gg Rx$
- Opcode1: 1100
- Opcode2: 000110

2.14. E-type Compare Immediate Instructions

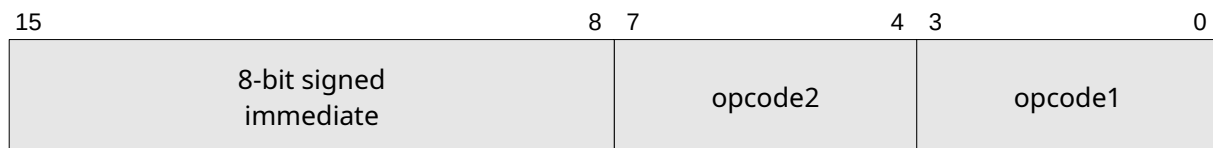


Fig 12: E-type Compare Immediate Instructions' Encoding

2.14.1. CLI Instruction

- Description: Sets flag bit if accumulator is less than 8-bit signed immediate
- Syntax: CLI 8-bit-signed-immediate
- Operation: if(A<immediate): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1101
- Opcode2: 1010

2.14.2. CGI Instruction

- Description: Sets flag bit if accumulator is greater than 8-bit signed immediate
- Syntax: CGI 8-bit-signed-immediate
- Operation: if(A>immediate): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1101
- Opcode2: 1011

2.14.3. CEI Instruction

- Description: Sets flag bit if accumulator is equal to 8-bit signed immediate
- Syntax: CEI 8-bit-signed-immediate
- Operation: if(A==immediate): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1101
- Opcode2: 1100

2.15. R-type Compare Instructions

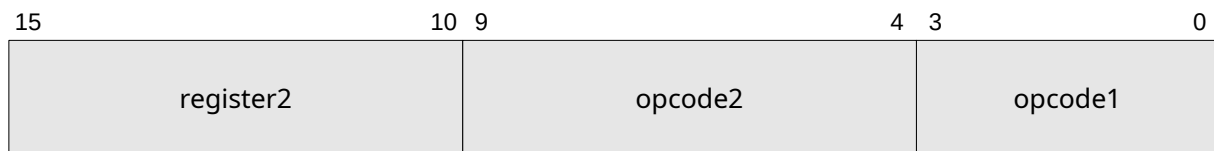


Fig 13: R-type Compare Instructions' Encoding

2.15.1. CL Instruction

- Description: Sets flag bit if accumulator is less than register
- Syntax: CL Rx
- Operation: if(A<Rx): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1100
- Opcode2: 001010

2.15.2. CG Instruction

- Description: Sets flag bit if accumulator is greater than register
- Syntax: CG Rx
- Operation: if(A>Rx): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1100
- Opcode2: 001011

2.15.3. CE Instruction

- Description: Sets flag bit if accumulator is equal to register
- Syntax: CE Rx
- Operation: if(A==Rx): flag \leftarrow 1 else: flag \leftarrow 0
- Opcode1: 1100
- Opcode2: 001100

2.16. R-type Flag Instruction

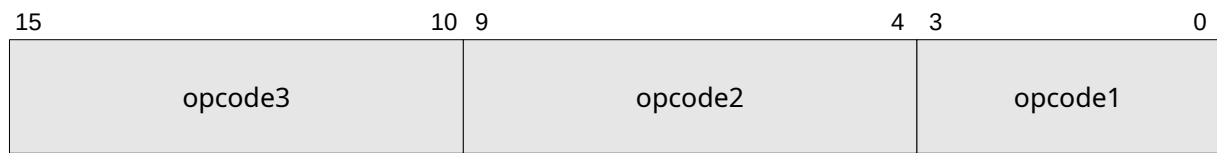


Fig 14: R-type Flag Instruction's Encoding

2.16.1. NOTF Instruction

- Description: Inverts flag bit
- Syntax: NOTF
- Operation: $\text{flag} \leftarrow !\text{flag}$
- Opcode1: 1100
- Opcode2: 010000
- Opcode3: 000000 or xxxxxx

2.17. SRP16 Opcode Table

| Instruction | Encoding Type | Opcode2 | Opcode1 |
|-------------|---------------|---------|---------|
| LDR | E-type | - | 0000 |
| LDRU | E-type | - | 0001 |
| LD@MPTR | E-type | - | 0010 |
| ST@MPTR | E-type | - | 0011 |
| LDB@MPTR | E-type | - | 0100 |
| STB@MPTR | E-type | - | 0101 |
| LDA | T-type | - | 0110 |
| LDMPTR | T-type | - | 0111 |
| LDMPTRU | T-type | - | 1000 |
| MOV | R-type | - | 1001 |
| SJMP | T-type | - | 1010 |
| SJMPF | T-type | - | 1011 |
| ADD | R-type | 000001 | 1100 |
| SUB | R-type | 000010 | 1100 |
| SLA | R-type | 000011 | 1100 |
| SRA | R-type | 000100 | 1100 |
| SLL | R-type | 000101 | 1100 |
| SRL | R-type | 000110 | 1100 |
| AND | R-type | 000111 | 1100 |
| OR | R-type | 001000 | 1100 |
| XOR | R-type | 001001 | 1100 |
| CL | R-type | 001010 | 1100 |
| CG | R-type | 001011 | 1100 |
| CE | R-type | 001100 | 1100 |
| ADC | R-type | 001101 | 1100 |
| SBB | R-type | 001110 | 1100 |
| NOTF | R-type | 010000 | 1100 |
| SLAI | R-type | 100011 | 1100 |
| SRAI | R-type | 100100 | 1100 |
| SLLI | R-type | 100101 | 1100 |
| SRLI | R-type | 100110 | 1100 |
| LDAU | R-type | 111011 | 1100 |
| POP | R-type | 111100 | 1100 |

| Instruction | Encoding Type | Opcode2 | Opcode1 |
|-------------|---------------|---------|---------|
| PUSH | R-type | 111101 | 1100 |
| INC | R-type | 111110 | 1100 |
| DEC | R-type | 111111 | 1100 |
| ADDI | E-type | 0001 | 1101 |
| ANDI | E-type | 0111 | 1101 |
| ORI | E-type | 1000 | 1101 |
| XORI | E-type | 1001 | 1101 |
| CLI | E-type | 1010 | 1101 |
| CGI | E-type | 1011 | 1101 |
| CEI | E-type | 1100 | 1101 |
| ADCI | E-type | 1101 | 1101 |
| SBBI | E-type | 1110 | 1101 |

2.18. SRP16 Instruction Set Quick Reference

| Instruction | Operation |
|-----------------------------------|--|
| LDR Rx, 8-bit-signed-immediate | $Rx \leftarrow \text{immediate}$ |
| LDRU Rx, 8-bit-unsigned-immediate | $Rx[15:8] \leftarrow \text{immediate}$ |
| LD@MPTR Rx, 8-bit-signed-offset | $Rx \leftarrow \text{memory}[MPTR]$ $MPTR \leftarrow MPTR + \text{offset}$ |
| ST@MPTR Rx, 8-bit-signed-offset | $\text{memory}[MPTR] \leftarrow Rx$ $MPTR \leftarrow MPTR + \text{offset}$ |
| LDB@MPTR Rx, 8-bit-signed-offset | $Rx[7:0] \leftarrow \text{memory}[MPTR]$ $MPTR \leftarrow MPTR + \text{offset}$ |
| STB@MPTR Rx, 8-bit-signed-offset | $\text{memory}[MPTR] \leftarrow Rx[7:0]$ $MPTR \leftarrow MPTR + \text{offset}$ |
| LDA 12-bit-signed-immediate | $A \leftarrow \text{immediate}$ |
| LDAU 6-bit-unsigned-immediate | $A[15:12] \leftarrow \text{immediate}[3:0]$ |
| LDMPTR 12-bit-unsigned-immediate | $MPTR \leftarrow \text{immediate}$ |
| LDMPTRU 12-bit-signed-immediate | $MPTR[15:12] \leftarrow \text{immediate}[3:0]$ |
| MOV Rx, Ry | $Rx \leftarrow Ry$ |
| MOV Rx, PC | $Rx \leftarrow PC + 4$ |
| JMP Ry or MOV PC, Ry | $PC \leftarrow Ry$ |
| SJMP 12-bit-signed-offset | $PC \leftarrow PC + \text{offset}$ |
| SJMPF 12-bit-signed-offset | if(flag): $PC \leftarrow PC + \text{offset}$ |
| NOTF | $\text{flag} \leftarrow \neg \text{flag}$ |
| POP Rx | $Rx \leftarrow \text{memory}[SP]$ $SP \leftarrow SP + 1$ |
| PUSH Rx | $SP \leftarrow SP - 1$ $\text{memory}[SP] \leftarrow Rx$ |
| INC Rx | $Rx \leftarrow Rx + 1$ |
| DEC Rx | $Rx \leftarrow Rx - 1$ |
| ADDI 8-bit-signed-immediate | $A \leftarrow A + \text{immediate}$ |
| ADCI 8-bit-signed-immediate | $A \leftarrow A + \text{immediate} + \text{carry}$ |
| SBBI 8-bit-signed-immediate | $A \leftarrow A - \text{immediate} - \text{carry}$ |
| ANDI 8-bit-signed-immediate | $A \leftarrow A \& \text{immediate}$ |
| ORI 8-bit-signed-immediate | $A \leftarrow A \text{immediate}$ |
| XORI 8-bit-signed-immediate | $A \leftarrow A \wedge \text{immediate}$ |
| SLAI 6-bit-unsigned-immediate | $A \leftarrow A \ll \text{immediate}$ |
| SRAI 6-bit-unsigned-immediate | $A \leftarrow A \gg \text{immediate}$ |

| Instruction | Operation |
|-------------------------------|---|
| SLLI 6-bit-unsigned-immediate | $A \leftarrow A \ll \text{immediate}$ |
| SRLI 6-bit-unsigned-immediate | $A \leftarrow A \gg \text{immediate}$ |
| ADD Rx | $A \leftarrow A + Rx$ |
| SUB Rx | $A \leftarrow A - Rx$ |
| ADC Rx | $A \leftarrow A + Rx + \text{carry}$ |
| SBB Rx | $A \leftarrow A - Rx - \text{carry}$ |
| AND Rx | $A \leftarrow A \& Rx$ |
| OR Rx | $A \leftarrow A Rx$ |
| XOR Rx | $A \leftarrow A \wedge Rx$ |
| SLA Rx | $A \leftarrow A \ll Rx$ |
| SRA Rx | $A \leftarrow A \ggg Rx$ |
| SLL Rx | $A \leftarrow A \ll Rx$ |
| SRL Rx | $A \leftarrow A \gg Rx$ |
| CLI 8-bit-signed-immediate | if($A < \text{immediate}$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |
| CGI 8-bit-signed-immediate | if($A > \text{immediate}$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |
| CEI 8-bit-signed-immediate | if($A == \text{immediate}$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |
| CL Rx | if($A < Rx$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |
| CG Rx | if($A > Rx$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |
| CE Rx | if($A == Rx$): flag $\leftarrow 1$ else: flag $\leftarrow 0$ |