

# ***SRP16 Assembler Guide***

*v0.1.0*

*Vishnu Shankar B*

# Contents

Ch 1. Assembler Preprocessors.....	3
1.1. Comments.....	3
1.2. Labels.....	3
1.3. .byte Preprocessor.....	3
1.4. .hex Preprocessor.....	3
1.5. .string Preprocessor.....	3
1.6. .equ Preprocessor.....	3
1.7. .org Preprocessor.....	3
1.8. .include Preprocessor.....	3
Ch 2. SRP16 Instruction Set Summary.....	4
2.1. Registers.....	4
2.2. Instruction Set.....	4
Ch 3. Command Line Usage.....	6
3.1. Usage.....	6
3.2. List of available OPTIONS.....	6
3.3. Example Usage.....	6
3.3.1. Intel Hex.....	6
3.3.2. For ISA Simulator.....	6
3.3.3. For Verilog Simulation.....	6

# Ch 1. Assembler Preprocessors

## 1.1. Comments

Anything that begins with ';' - semicolon are comments. Comments are ignored.

## 1.2. Labels

Anything that ends with ':' character is a label.

## 1.3. .byte Preprocessor

Lets you define a byte.

*Example:*

```
.byte 0x08
```

## 1.4. .hex Preprocessor

Lets you define array of bytes.

*Example:*

```
.hex "AABBCC"
```

## 1.5. .string Preprocessor

Lets of define an array of character bytes.

*Example:*

```
.string "Hello world"  
.byte 0x00 ;Null character
```

## 1.6. .equ Preprocessor

.equ lets you define constants.

*Example:*

```
.equ "zero", 0 ;"zero" is now 0
```

## 1.7. .org Preprocessor

.org Lets you align data or instructions to a particular address.

*Example:*

```
.org 0x08  
.byte 0x01 ;This byte will be at address 0x08
```

## 1.8. .include Preprocessor

.include Lets you include code from other files.

*Example:*

```
.include "code.asm" ;Copy paste code from code.asm
```

# Ch 2. SRP16 Instruction Set Summary

## 2.1. Registers

- General Purpose Registers R0-R15 (Accessible by Load-Store Instructions)
- General Purpose Registers R16-R31 (Not Accessible by Load-Store Instructions)
- Accumulator Register (R60)
- Memory Pointer Register or MPTR (R61)
- Stack Pointer or SP (R62)
- Program Counter or PC (R63)
- POP, PUSH, INC, DEC instructions can only access General Purpose Registers R0-R31

## 2.2. Instruction Set

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	$Rx \leftarrow \text{memory}[\text{MPTR}]$ $\text{MPTR} \leftarrow \text{MPTR} + \text{offset}$
ST@MPTR Rx, 8-bit-signed-offset	$\text{memory}[\text{MPTR}] \leftarrow Rx$ $\text{MPTR} \leftarrow \text{MPTR} + \text{offset}$
LDB@MPTR Rx, 8-bit-signed-offset	$Rx[7:0] \leftarrow \text{memory}[\text{MPTR}]$ $\text{MPTR} \leftarrow \text{MPTR} + \text{offset}$
STB@MPTR Rx, 8-bit-signed-offset	$\text{memory}[\text{MPTR}] \leftarrow Rx[7:0]$ $\text{MPTR} \leftarrow \text{MPTR} + \text{offset}$
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	$\text{MPTR} \leftarrow \text{immediate}$
LDMPTRU 12-bit-signed-immediate	$\text{MPTR}[15:12] \leftarrow \text{immediate}[3:0]$
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	$Rx \leftarrow \text{memory}[\text{SP}]$ $\text{SP} \leftarrow \text{SP} + 1$
PUSH Rx	$\text{SP} \leftarrow \text{SP} - 1$ $\text{memory}[\text{SP}] \leftarrow Rx$
INC Rx	$Rx \leftarrow Rx + 1$
DEC Rx	$Rx \leftarrow Rx - 1$

Instruction	Operation
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	$A \leftarrow A$
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	$A \leftarrow A \gg \text{immediate}$
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	$A \leftarrow A$
XOR Rx	$A \leftarrow A \wedge Rx$
SLA Rx	$A \leftarrow A \ll Rx$
SRA Rx	$A \leftarrow A \gg Rx$
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 $\leftarrow 0$

# ***Ch 3. Command Line Usage***

## **3.1. Usage**

`srp16asm INPUTFILE OPTION OUTPUTFILE`

## **3.2. List of available OPTIONS**

- **-o** : Intel hex file format
- **-h** : Hex file for verilog simulation
- **-s** : Hex file with debug symbols, for ISA simulator

## **3.3. Example Usage**

### **3.3.1. Intel Hex**

`srp16asm input.asm -o output.hex`

### **3.3.2. For ISA Simulator**

`srp16asm input.asm -s output.dhex`

### **3.3.3. For Verilog Simulation**

`srp16asm input.asm -h output.dat`