TOY ISA 2

Instruction Set Architecture Specification

Proteus Lab

Generated on: 2025-09-19

Version: 1.2

Table of Contents

USAT

Encoding

31:26	25:21	20:16	15:11	10:0
100010	rd	rs	imm5	00000000000

Assembler

```
USAT rd, rs, #imm5
```

Semantics

```
X[rd] ← saturate_unsigned(X[rs], imm5)
```

Notes

Saturation of the unsigned value in `X[rs]` to N bits, where N = `#imm`

BEQ

Encoding

31:26	25:21	20:16	15:0
010011	rs	rt	offset

Assembler

```
BEQ rs, rt, #offset
```

Semantics

```
target ← sign_extend(offset || 0b00)
```

```
cond ← (X[rs] == X[rt])
```

```
PC ← if (cond) PC + target else PC + 4
```

SLTI

Encoding

31:26	25:21	20:16	15:0
111011	rs	rt	imm

Assembler

```
SLTI rt, rs, #imm
```

Semantics

```
X[rt] ← X[rs] < sign_extend(imm)
```

MOVZ

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	000100

Assembler

```
MOVZ rd, rs, rt
```

Semantics

```
if (X[rt] == 0) X[rd] ← X[rs]
```

STP

Encoding

31:26	25:21	20:16	15:11	10:0
111001	base	rt1	rt2	offset

Assembler

```
STP rt1, rt2, offset(base)
```

Semantics

```
addr ← X[base] + sign_extend(offset)
```

```
memory[addr] ← X[rt1]
```

```
memory[addr + 4] ← X[rt2]
```

Notes

The lowest 2 bits of the `#offset` field must be zero. If they are not, the result of the instruction is undefined (misaligned access).

SELC

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rd	rs1	rs2	00000	000011

Assembler

```
SELC rd, rs1, rs2
```

Semantics

```
X[rd] ← selc(X[rs1], X[rs1])
```

Notes

Selects the maximum value from `X[rs1]` and `X[rs2]` without using flags.

SYSCALL

Encoding

31:26	25:6	5:0
000000	code	011001

Assembler

```
SYSCALL
```

Semantics

```
SigException(SystemCall)
```

Notes

`X8` — system call number, `X0` - `X7` — args, `X0` — result, see man syscall

SUB

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	011111

Assembler

```
SUB rd, rs, rt
```

Semantics

```
X[rd] ← X[rs] - X[rt]
```

J

Encoding

31:26	25:0
010110	index

Assembler

```
J target
```

Semantics

```
PC ← PC[31:28] || instr_index || 0b00
```

ADD

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	011010

Assembler

```
ADD rd, rs, rt
```

Semantics

```
X[rd] ← X[rs] + X[rt]
```

RBIT

Encoding

31:26	25:21	20:16	15:6	5:0
000000	rd	rs	0000000000	011101

Assembler

```
RBIT rd, rs
```

Semantics

```
X[rd] ← reverse_bits(X[rs])
```

Notes

Reverses the order of the bits in the `X[rs1]` register.

LD

Encoding

31:26	25:21	20:16	15:0
100011	base	rt	offset

Assembler

```
LD rt, offset(base)
```

Semantics

```
X[rt] ← memory[X[base] + sign_extend(offset)]
```

Notes

The lowest 2 bits of the `#offset` field must be zero. If they are not, the result of the instruction is undefined (misaligned access).

RORI

Encoding

31:26	25:21	20:16	15:11	10:0
001100	rd	rs	imm5	00000000000

Assembler

```
RORI rd, rs, #imm5
```

Semantics

```
X[rd] ← rotate_right(X[rs], imm5)
```

Notes

Rotates the bits of the `X[rs]` register to the right by `#imm` bits

ST

Encoding

31:26	25:21	20:16	15:0
100101	base	rt	offset

Assembler

```
ST rt, offset(base)
```

Semantics

```
memory[X[base] + sign_extend(offset)] ← X[rt]
```

Notes

The lowest 2 bits of the `#offset` field must be zero. If they are not, the result of the instruction is undefined (misaligned access).