TOY ISA 12

Instruction Set Architecture Specification

Proteus Lab

Generated on: 2025-09-19

Version: 1.12

Table of Contents

J

Encoding

31:26	25:0
010111	index

Assembler

```
J target
```

Semantics

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

CBIT

Encoding

31:26	25:21	20:16	15:11	10:0
111011	rd	rs	imm5	00000000000

Assembler

```
CBIT rd, rs, #imm5
```

Semantics

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

Notes

Clears exactly one bit at position `#imm` in the `X[rs]` register to 0. All other bits remain unchanged.

SUBI

Encoding

31:26	25:21	20:16	15:0
001111	rs	rt	imm

Assembler

```
SUBI rt, rs, #imm
```

Semantics

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

MOVN

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	000101

Assembler

```
MOVN rd, rs, rt
```

Semantics

```
if (X[rt] != 0) X[rd] ← X[rs]
```

BEQ

Encoding

31:26	25:21	20:16	15:0
100110	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
```

BEXT

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rd	rs1	rs2	00000	110110

Assembler

```
BEXT rd, rs1, rs2
```

Semantics

```
X[rd] ← bit_extract(X[rs1], X[rs1])
```

Notes

Extracts and packs bits from register `X[rs1]` into the positions where the corresponding bits in mask `X[rs2]` are set. The extracted bits will place the least significant bits of the result Rd.

LDP

Encoding

31:26	25:21	20:16	15:11	10:0
001110	base	rt1	rt2	offset

Assembler

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

Semantics

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

```
X[rt1] ← memory[addr]
```

```
X[rt2] ← memory[addr + 4]
```

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).

ADD

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	000110

Assembler

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

Semantics

```
X[rd] ← X[rs] + X[rt]
```

LD

Encoding

31:26	25:21	20:16	15:0
100010	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).

CLS

Encoding

31:26	25:21	20:16	15:6	5:0
000000	rd	rs	0000000000	101101

Assembler

```
CLS rd, rs
```

Semantics

```
X[rd] ← count_leading_signs(X[rs])
```

Notes

Counts the number of leading ones bits in the `X[rs1]` register.

RORI

Encoding

31:26	25:21	20:16	15:11	10:0
111111	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
111010	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).

XOR

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	011011

Assembler

```
XOR rd, rs, rt
```

Semantics

```
X[rd] ← X[rs] xor X[rt]
```

SYSCALL

Encoding

31:26	25:6	5:0
000000	code	000111

Assembler

```
SYSCALL
```

Semantics

```
SigException(SystemCall)
```

Notes

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