TOY ISA 5

Instruction Set Architecture Specification

Proteus Lab

Generated on: 2025-09-19

Version: 1.5

Table of Contents

NOR

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	001101

Assembler

```
NOR rd, rs, rt
```

Semantics

```
X[rd] ← X[rs] nor X[rt]
```

LDP

Encoding

31:26	25:21	20:16	15:11	10:0
111100	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).

CBIT

Encoding

31:26	25:21	20:16	15:11	10:0
111001	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.

BDEP

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rd	rs1	rs2	00000	110011

Assembler

```
BDEP rd, rs1, rs2
```

Semantics

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

Notes

Places the least significant bits of `X[rs1]` into the positions specified by the mask `X[rs2]`. The remaining bits are filled with zeros.

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]
```

SSAT

Encoding

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

Assembler

```
SSAT rd, rs, #imm5
```

Semantics

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

Notes

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

ST

Encoding

31:26	25:21	20:16	15:0
110010	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).

CLZ

Encoding

31:26	25:21	20:16	15:6	5:0
000000	rd	rs	0000000000	110101

Assembler

```
CLZ rd, rs
```

Semantics

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

Notes

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

BNE

Encoding

31:26	25:21	20:16	15:0
000110	rs	rt	offset

Assembler

```
BNE rs, rt, #offset
```

Semantics

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

```
cond ← (X[rs] != X[rt])
```

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

LD

Encoding

31:26	25:21	20:16	15:0
001010	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).

XOR

Encoding

31:26	25:21	20:16	15:11	10:6	5:0
000000	rs	rt	rd	00000	101001

Assembler

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

Semantics

```
X[rd] ← X[rs] xor X[rt]
```

SYSCALL

Encoding

31:26	25:6	5:0
000000	code	111000

Assembler

```
SYSCALL
```

Semantics

```
SigException(SystemCall)
```

Notes

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

BEQ

Encoding

31:26	25:21	20:16	15:0
011110	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
```

J

Encoding

31:26	25:0
110110	index

Assembler

```
J target
```

Semantics

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