1. Loads:
2. Load Byte
3. Load Word

LI A0,2794287 // 0x2AA32F

ADDI SP,SP,-16 // Add space to the stack

SW A0,0(SP) // 0x2AA32F -> stack

LB A1,1(SP) // Sign extend 0xA3 into X11

LB A2,2(SP) // Save 0x2A = 42 into A2

LW A3,0(SP) // Save 0x2AA32F into A3

ADDI SP,SP,16 // Restore the stack

Non-zero registers:

X2 = 1024

X10 = 2794287

X11 = 0xFFFFFFA3 = 4294967203 // I’m getting 163 = 0xA3 (No sign extend)

X12 = 42

X13 = 2794287

1. Stores:
2. Store Byte
3. Store Word

LI A0,163 // 0xA3

ADDI SP,SP,-16 // Add space to the stack

SB SP,A0,6 // Save 0xA3 into 6(SP)

LW A1,4(SP) // Save 0xA30000 = 10682368 into A1

LB A2,SP,6 // Save 0xA3 = into A2

ADDI SP,SP,16 // Restore the stack

Non-zero registers:

X2 = 1024

X10 = 163

X11 = 10682368

X12 = 163

1. Arithmetic:
2. Add
3. Add imm
4. Sub

LI T1,65535 // Save 0xFFFF into T1

LI T2,2 // Save 2 into T2

ADD T3,T1,T2 // Add 0xFFFF + 0x2 = 0x10001 = 65537 into T3

ADDI T4,T2,4095 // Save 4097 into T4

SUB T5,T3,T4 // 0x10001 – 0x1001 = 61440 into T5

SUB T6,T2,T1 // 2 – 65535 = -65533 = 0xFFFF0003 = 4294901763

Non-zero registers:

X2 = 1024

X6 = 65535

X7 = 2

X28 = 65537

X29 = 4097

X30 = 61440

X31 = 4294901763

1. Bitwise operators:
2. Exclusive OR
3. Exclusive OR imm
4. Shift left imm
5. Shift right imm

LI S2,2863311530 // Load 0xAAAAAAAA into S2

LI S3,-1 // Load 0xFFFFFFFF into S3

XOR S4,S2,S3 // Given that 0xA ^ 0xF = 0x5, save 0x55555555 = 1431655765 into S4

XORI S5,S2,4095 // Save 0xAAAAA555 = 2863310165 into S5

SLLI S6,S2,3 // Save 0x55555550 = 1431655760 into S6

SRLI S7,S3,9 // Save 0x7FFFFF = 8388607 into S7

Non-zero registers:

X2 = 1024

X18 = 2863311530

X19 = 4294967295

X20 = 1431655765

X21 = 2863310165

X22 = 1431655760

X23 = 8388607

1. Pseudo-instructions:
2. Move
3. Load Immediate
4. Negate
5. Ones complement:

LI X0,20 // Should not overwrite X0

MV FP,SP // Save SP to FP

NEG S10,FP // Save -1024 = 0xFFFFFC00 = 4294966272

NOT S11,S10 // Save 0x3FF = 1023 into S11

ADDI A0,S11,1 // Save 1024 into S12

Non-zero registers:

X2 = 1024

X8 = 1024

X10 = 1024

X26 = 4294966272

X27 = 1023

1. Jump offset:
2. Jump And Link
3. Jump
4. Jump offset + reg:
5. Jump And Link Reg
6. Jump Register

JALR X0,X0,20 // PC = 20

LI A7,20 // Load 20 into A7 (This should not happen)

ADDI RA,RA,4 // RA = 24+4 = 28

JR RA // Overwrite PC with RA

LI A1,10 // Load 10 into A1 (This should not happen)

JAL RA,-12 // RA = 24 and PC = 20-12 = 8

LI A0,20 // Load 20 into A0 (This should not happen)

ADDI A6,X0,1 // Add A6 = A7 + 1

J 4 // Jump to the next instruction

Non-zero registers:

X1 = 28

X2 = 1024

X16 = 1