Chapter 2 (Instruction Set)

Operands

Low level language (Assembly Language):

- An operand is a value (an argument) on which the instruction operates. The operand may be a processor register, a memory address, or a literal constant.

ADD X0, X1, X2 . // X0 = X1 + X2

operaton

operands .

High level language (C/C++/Java):

- Operands are the constants or variables which the operators operate upon.

operands .

c = a + b ;

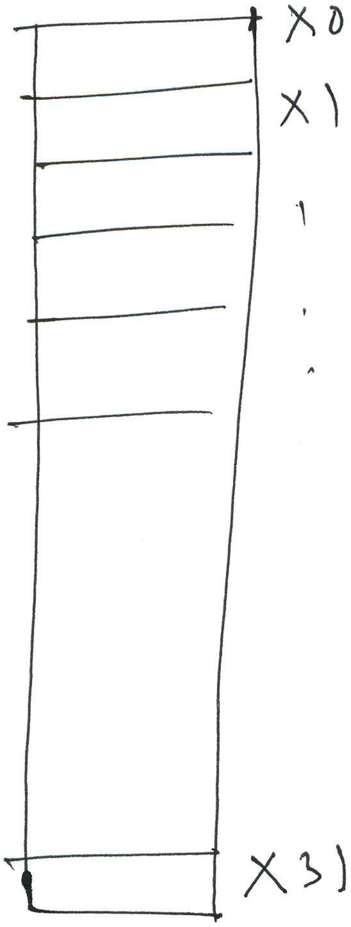
X0 X1 X2 .

Addition operator .

64 bit ARMV8 → Architecture.

→ **LEGv8 Registers**

Name	Register number	Usage
X0–X7	0–7	Arguments/Results
X8	8	Indirect result location register
X9–X15	9–15	Temporaries
X16 (IP0)	16	May be used by linker as a scratch register; other times used as temporary register
X17 (IP1)	17	May be used by linker as a scratch register; other times used as temporary register
X18	18	Platform register for platform independent code; otherwise a temporary register
X19–X27	19–27	Saved
X28 (SP)	28	Stack Pointer
X29 (FP)	29	Frame Pointer
X30 (LR)	30	Link Register (return address)
XZR	31	The constant value 0



Register
64 bit

Arithmetic Instructions

ADD-

- The ADD instruction adds two operands and place the result on destination register. Both operands are register.

operand 2

ADD X0, X1, X2 // X0 = X1 + X2

Destination register operand 1

ADDI –

- The ADDI instruction adds two operands and place the result on destination register. Operand 1 is a register, operand 2 is an immediate value.

ADDI X0, X1, #4 // X0 = X1 + 4

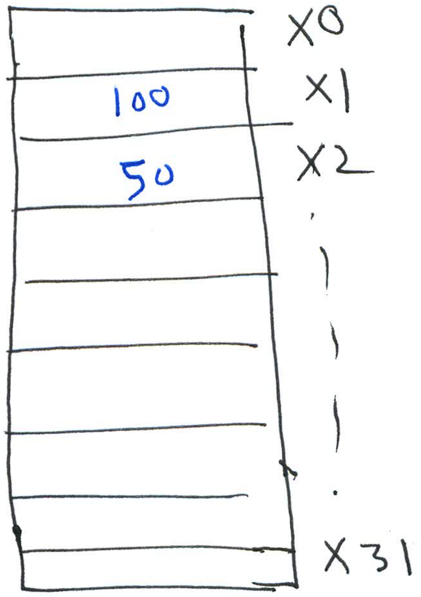
SUB-

- The SUB instruction subtracts operand 2 from operand 1 and place the result on destination register. Both operands are register.

SUB X0, X1, X2 // X0 = X1 - X2

100 - 50 = 50.

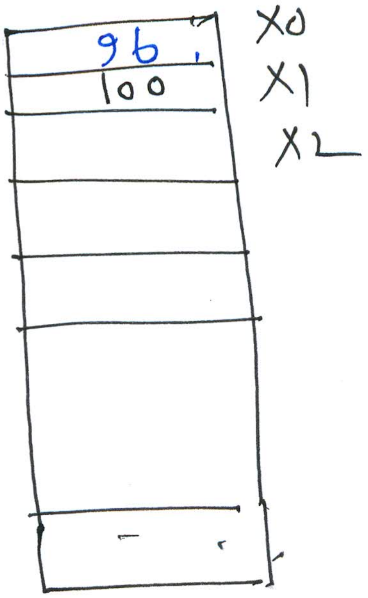
operand1 operand 2.



SUBI-

- The SUBI instruction subtracts operand 2 from operand 1 and place the result on destination register. Operand 1 is a register, operand 2 is an immediate value.

SUBI X0, X1, #4 // X0 = X1 - 4.



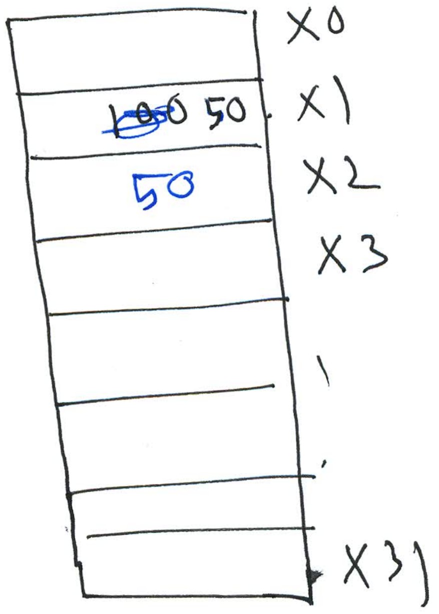
a = 10

c = a ;

MOV-

- This instruction loads a 64-bit value into the destination register from another register.

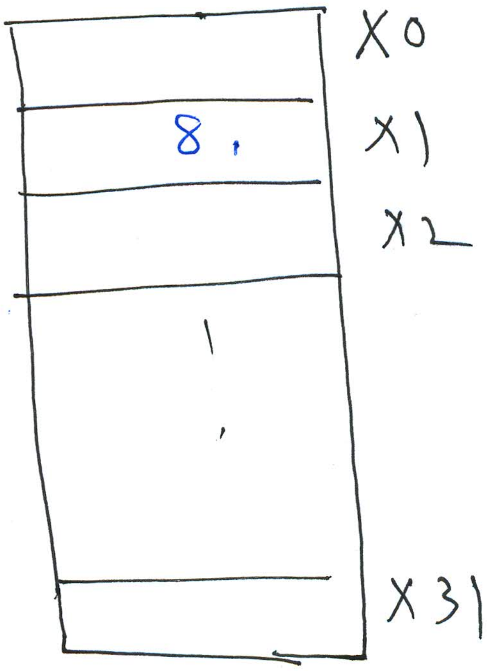
MOV X1, X2 // X1 = X2 .



MOVI

- This instruction loads a 64-bit value into the destination register from an immediate value.

MOVI X1 , #8



a X b .

LSL – Logical shift left

- LSL instruction effectively multiply the contents of a register by 2^i.
- Each bit of the register is shifted left, the MSB is removed and empty bits are filled with zeros.

LSL X0 , X2 , #2 // $X0 = X2 * 2^2$

i

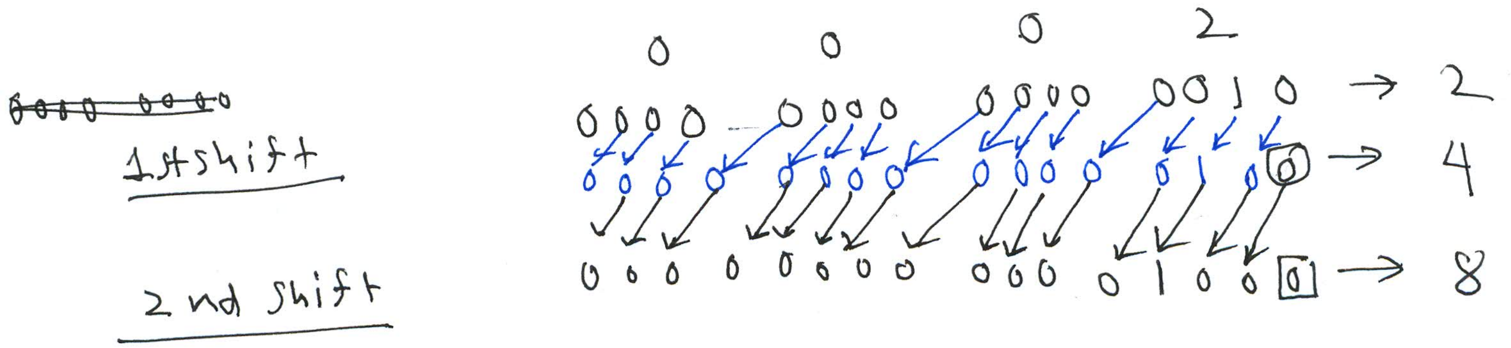
$= X2 * 4$.

Destination Register .

Decimal . Hex

$2 X 2^1$

Problem: Assume that X2 = 2 (0X0000000000000002 in hexadecimal). What will be the value of X0 after running the following instruction: LSL X0, X2, #2



LSL X0 , X1 , #5 → $X0 = X1 * 2^5$.

LSL X3 , X4 , #1 → $X3 = X4 * 2^1$

LSR

LSR

LSR X0, X1, #4 → $X0 = \frac{X1}{2^4}$

- LSR instruction divide the contents of a Register by 2^i.
- Each bit of the register is shifted right, LSB is removed and empty bits are filled with zeros.

LSR X0, X2, #3 ⓘ. // $X0 = X2/2^3 = \frac{X2}{8}$

Problem Asiume that X2 = 24. What will be the value of X0 after running the following instruction: 24/8.

LSR X0, X2, #3

0001
1000

24 = 0x 0000 0000 0000 0018

0 0 1 8

0000 0000 0001 100~~0~~ → 24

$\frac{24}{2^1} = 12$ — 0000 0000 0000 110~~0~~ ← 1st shift (12)

$\frac{24}{2^2} = 6$ — 0000 0000 0000 011~~0~~ ← 2nd shift (6)

$\frac{24}{2^3} = 3.$ → 0000 0000 0000 0011 ← 3rd shift (3)