RISC-V Instruction Set Architecture

why clear output buffer?

what is gcc?

what is compiler?

what is assembler?

What is the design flow

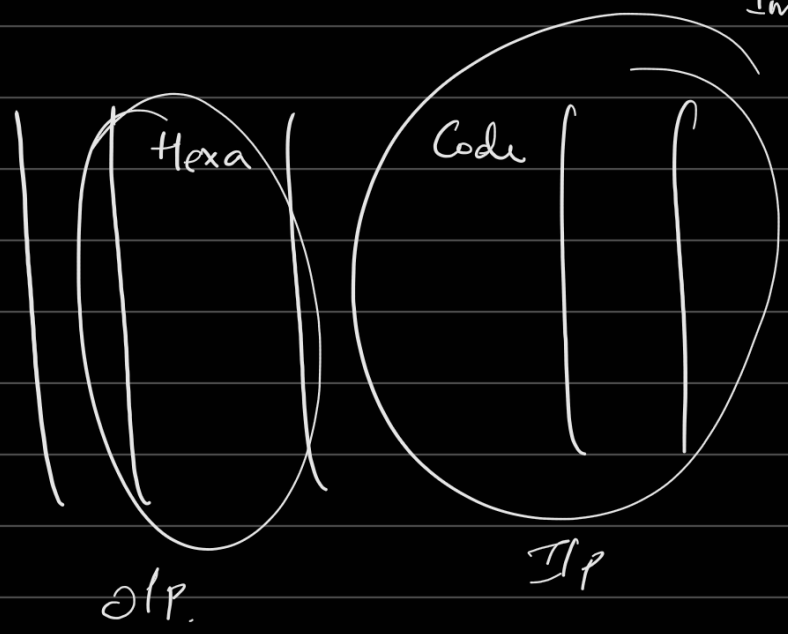
RISC Arch

↓

RTL

↓

Implementation.



Pseudo Instruction.

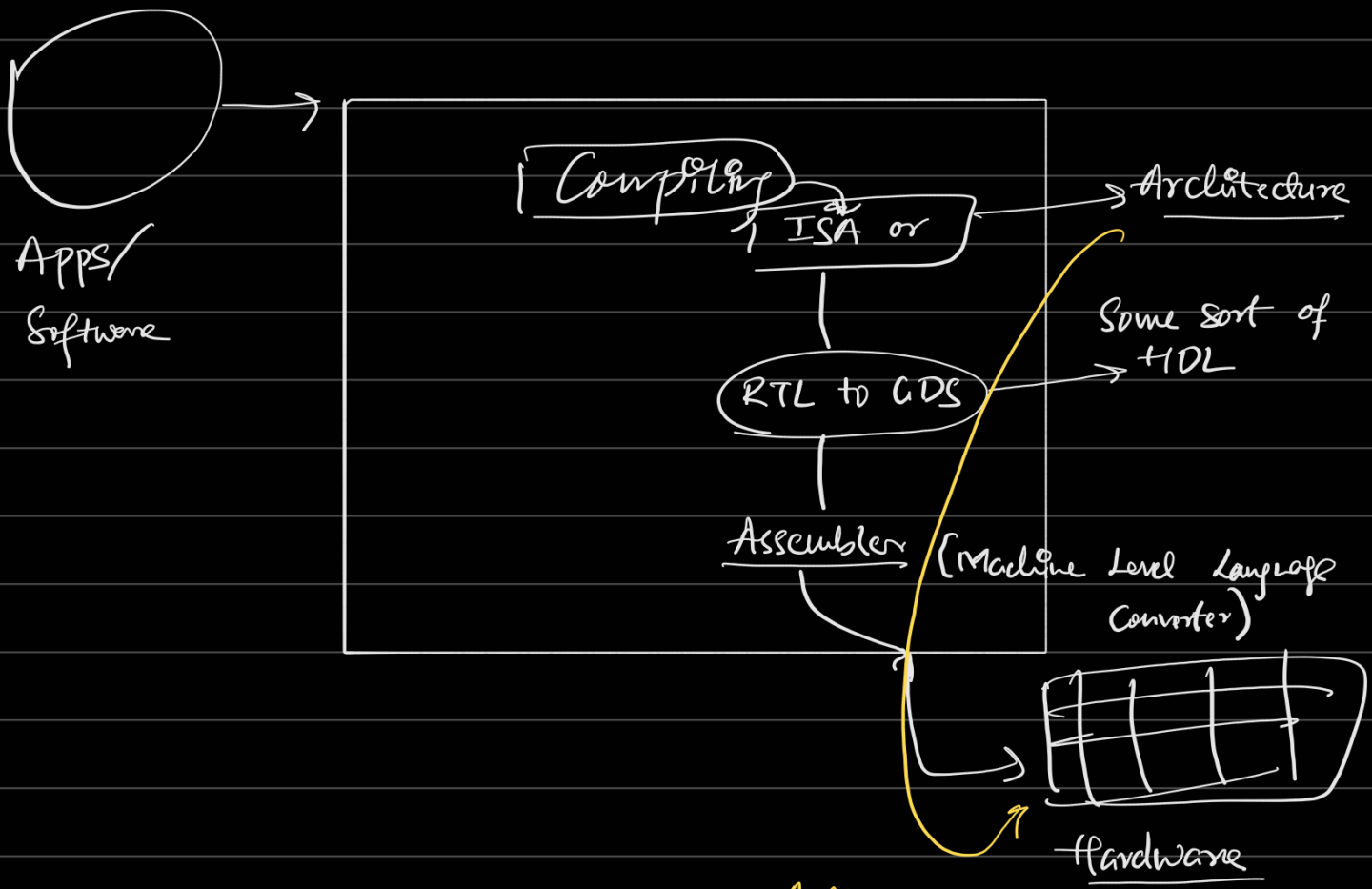
Base Integer Instr.ns. RV-64I

RISC-V 64-bit Integers ?

any CPU must contain this.

mulw
divw
} Multiply extension RV64M

Together called RV64IM CPU Core.



Architecture

based on the type of hardware
it is of (ARM, x86, RISC etc)

Single and Double precision floating point extension

RV64F

flw
fadd.s
fmul.s
fsd

RV64D

fcvt.d.s
fmv.x.d

Together called RVG4IMFD

(sp) a0, r0, s0 etc are Application Binary Interface (ABI)
used alot a.k.a., registers we used in ARM.
in example.

Memory allocation and stack pointer.

64 bit rep. of signed and unsigned integer.

number of inputs to be given.

"riscv64-unknown-elf-gcc" command not found

unkown

lab 2 of the course ('GCC Compile and disassemble')

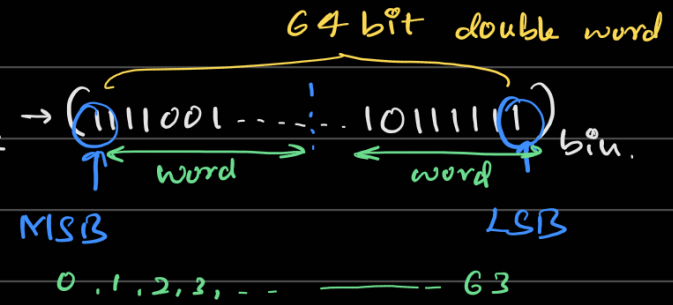
This error arises when the RISC-V GCC compiler is not installed or not in your system's PATH

↑ SPELLING MISTAKE IT WAS!!!

Number Representations

(DO1TO4)

- Entire 64 bit binary number is called the **Doubleword**
- 32 bit halves are called **Word**



- Group of 8 bits is byte. In a 64 bit archi., each instruction /doubleword consists of 8 bytes.

8	bits	→ 1 byte
4	bytes	→ 1 word
2	words	→ 1 doubleword

↑ These are very important for the implementation of efficient codes in our RISC-V Archi.

In the 64 bit representation, total number of patterns which we can represent are : 2^{64}

i.e., 0 to $(2^{64}-1)$

Range of 64 bit in decimal is → [0, 18,446,744,073,709,551,615]

Numbers exceeding this range will cause overflow flag to be set

∴ The no. of unsigned patterns or numbers (positive numbers) we can represent in RV64 are (0 to $2^{64}-1$)

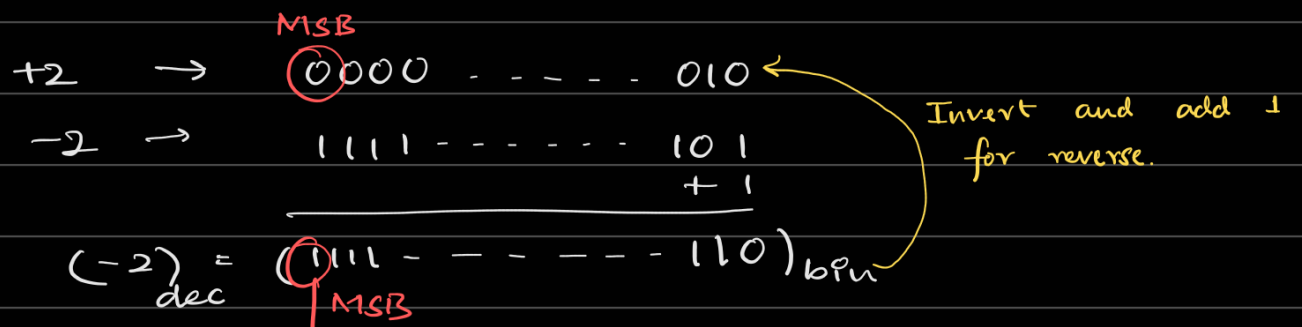
[001TO5]

For negative numbers or signed integers, we need to use 2's complement for that.

i.e., Take the positive bit rep.

Invert the 1s and 0s and add 1 to it

↑ 2's complement.



Note:-

Signed Numbers.

⊕ve nos → MSB = '0'

⊖ve nos → MSB = '1'

Range of signed range is $[-2^{63}, 2^{63}-1]$

eg. for 3 bit it is $-2^{3-1}, 2^{3-1}-1$

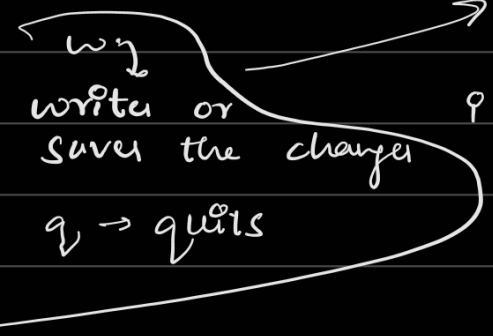
$= [-4, 3]$

	0,1,2,3
0	0 0 0 0 1 1 1 0 2 1 1 3
1	0 0 -4 0 1 -3 1 0 -2 1 1 -1

[-4, -3, -2, -1, 0, 1, 2, 3]

vim → used to open a c program file in the terminal

↳ :wq → exit out of the c program file and back to the terminal.



↳ i → switch from normal to insert mode where you can edit the c program file

↳ esc → to go back to normal mode.

:q! → quit w/o saving.

Even after changing the power of $2^{(n)}$, the max value remains same as ours is a 64 bit arch. and/or our data type doesn't go beyond that

[unsigned long long int]

	bytes	specifier	
unsigned int	4	%u	Word
int	4	%d	
unsigned long long int	8	%llu	Double word.
long long int	8	%lld	