Lec 28/01/25

⇒ 32-bit MIPS ISA

- 32-bit = register size.
  - → max virtual addr size = 32 bits
  - ✗ → instr size. (but has 32 bit instrs)
- 64 bit MIPS → 32 bit instr.
  64 bit regs
  = 64 bit VA.
- mult / div : Hi Lo → quotient
  → remainder
- arithmetic → sign extension
  logical → zero extension
  comparison → dep on signed / unsigned.
  (int x, unsigned y ⇒ both signed treated)
- add, addi, addu, addiu, sub, subu, slt, slti,
  → all signed extension and treated signed (2's com)
- sltu, sltiu } treats as unsigned.
  but does sign extension.
  then treats both operands as unsigned.
- slt $3, $2, $1 ≡ $3 = ($2 < $1) ? 1 : 0;
- srl → shift in 0

sra → "shift in signed bit."

- lui $3, 40 — $3 | 40 | 0 (← 16 → ← 16 →)

→ FP Instr

- cvt (typecasting) float → int
- mfc / mtc move from / to coprocessor ⋯ → FP unit/ regs.

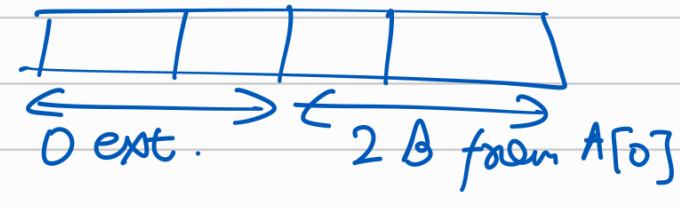
So we move b/w IP & FP regs.

mtc ⇒ use of FP constants.
we have to prepare the FP constants in the integer side only and then move it to FP in IEEE format.
float x, y; y = x + 5.2;

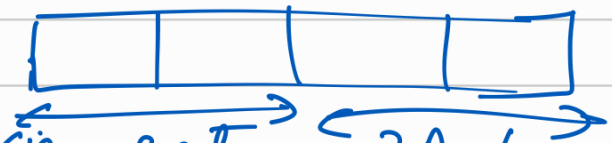
mfc ⇒ int z; z = (int) x; z += 1;
z: in FP unit; x: needs to be moved from F → I

→ Load / Store

unsigned short x;
x = A[0];



short x;
x = A[0];



sign ext 2B from A[0]

sh ⇒ pick 2 LS Bytes & store at given addr following Big Endian

add.s lf.s
add.d lf.d

Predicated Calls

if (x > y) c = a + b ;

```
cmp  z, x, y
jle
add  c, a, b
```

→ control dependence (pipeline hazards)

```
cmp  z, x, y
add  c, a, b, z
```

→ data dependence

if no jump then it will become NOP and processor will still execute one instr.

but memory operation of load/store of a, b, c won't happen. So we use predicated calls, so that which all is true will only be counted.