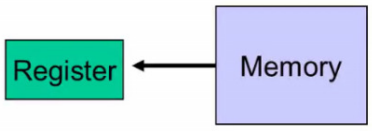
MIPS Architecture

- Multiplication and division instructions, which run asynchronously from other instructions.
- A pair of 32-bit registers, **HI** and **LO**, are provided.
- The program counter has 32 bits.
- The two low-order bits always contain zero. Why?
- MIPS I instructions are 32 bits long and are aligned to their natural word boundaries.

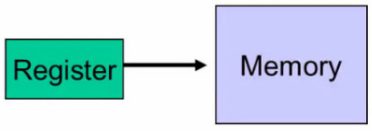
MIPS Architecture

- Values must be fetched from memory before (add and sub) instructions can operate on them

Load word:
lw $t0, memory-address

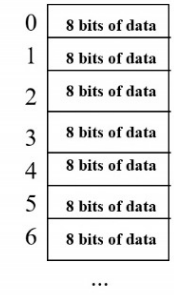


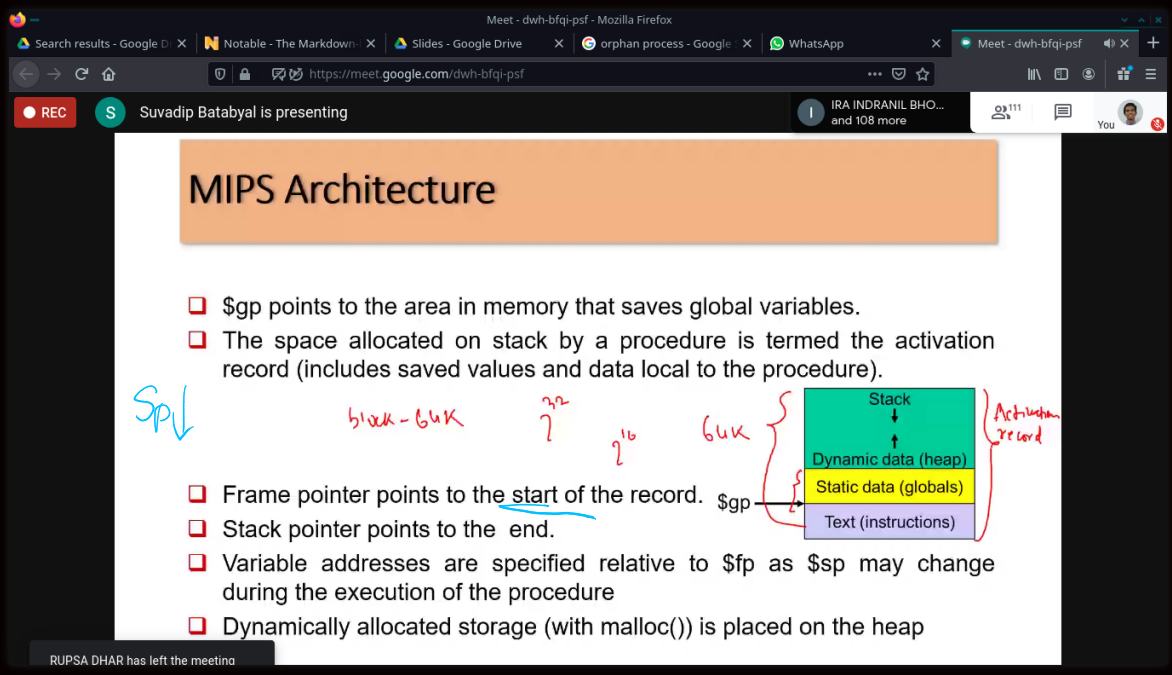
Store word:
sw $t0, memory-address



MIPS Architecture – Memory Organization

- Viewed as a large single-dimension array with access by address
- A memory address is an index into the memory array
- Byte addressing means that the index points to a byte of memory, and that the unit of memory accessed by a load/store is a byte
- Bytes are load/store units, but most data items use larger words
- For MIPS, a word is 32 bits or 4 bytes.
- 2^{32} bytes with byte addresses from 0 to 2^{32}-1
- 2^{30} words with byte addresses 0, 4, 8, ... 2^{32}-4
  - words are aligned
  - what are the least 2 significant bits of a word address?





R / I / J formats

Regs / immediate / jump

Type						
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	immediate (16)			
J	opcode (6)	address (26)				

rs - register source | rt - reg. target | rd = reg. destination

shamt = Shift amount | funct - function field (select variant)

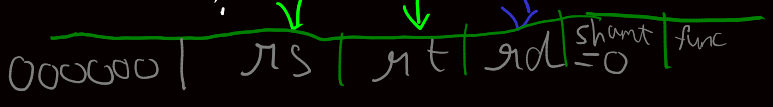
(function code #)

All R type instructions have 000000 as opcode

Operand code is fixed

C-Code a = b + c;

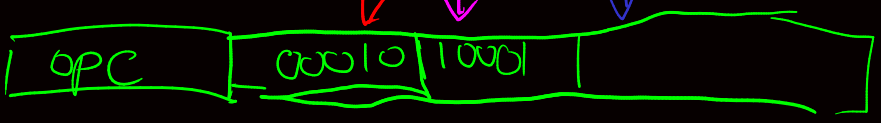
add $t0, $t1, $t2



I type: opc (6) src reg (5) dest (5) imm (16)

add – R type
addi = I-type

addi $17, $2, 1



J type

To convert 26 bits to 32 bit

→ Pad the end with 00 (alignment) each instruction is word aligned

→ The first 4 are the PC → Why (?)

first 4 bits of

why specifically PC?
form of segmentation

4($2) :- 4 bytes offset from value in $2 (256 megs)

↳ (goes to shamt i think for R-type)

32 bit immediate value in a register →

li $t0, 0x12345678

li is a pseudo instruction
lui $t0, 0x1234 -> load upper immediate
ori $t0, 0x5678

unconditional jumps : 26 - bits
conditional jumps - 16 bits -> program must fit into 16 bits
Use PC-Relative then

Conditional value: (address-PC)/4
Unconditional value : Address/4

bne is at 80012 , Exit is at 800024
so the address is supposed to be 12/4 = 3

jump is unconditional however -> it's jumpting to 80000 -> address is 80000/4 = 20000