PC logic :- Identify instruction CPU will execute. Executed sequentially; default behaviour is to increment.

fetch :- instruction memory (IMem) holds inst^r to execute.

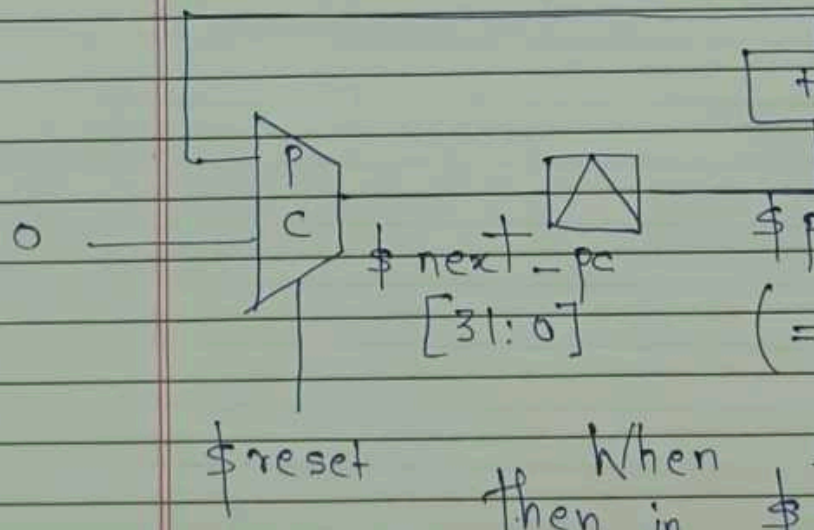
decode logic : inst^r broken to fields based on type, tell which register to read.

ALU :- does the operation

Register-file write : Result from ALU written to destination register

DMem :- Written to by store instructions and read from by load instructions.

PC : byte address references first byte of inst^r in IMem
4 byte long.



When $reset. address then in $next_pc : assigned to

Computer Architecture

Abstraction

software ← → hardware

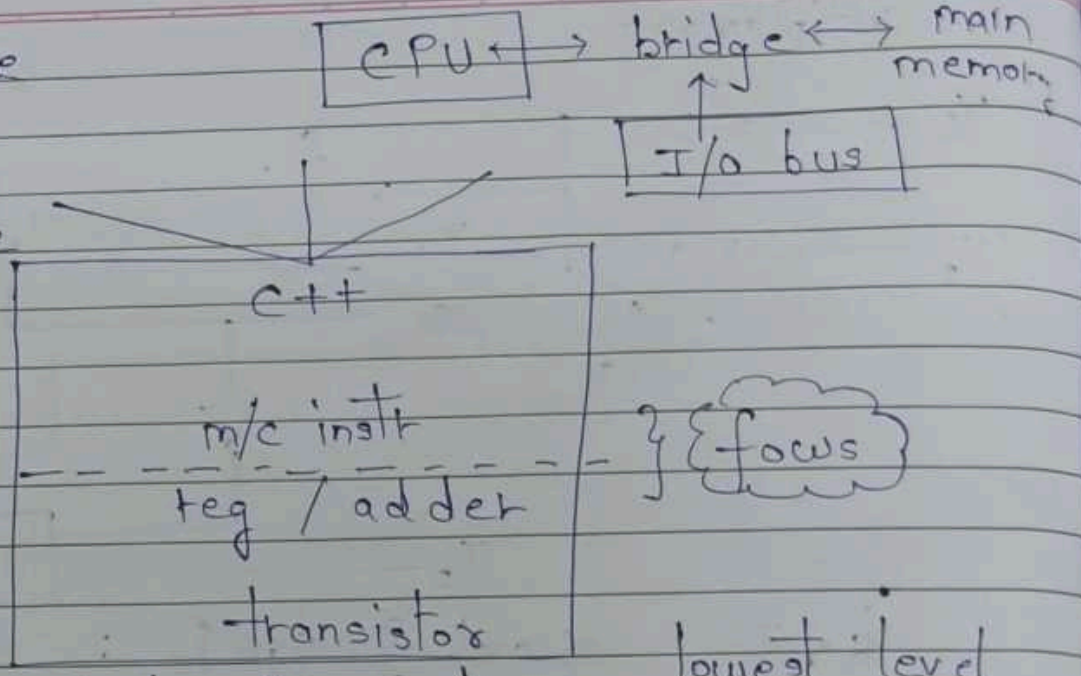
high-level code

↓

assembly

↓

machine code



architecture → instruction set : lowest level visible to coder

→ micro : fills gap btw instructions & logic module

Application

OS compiler

ISA

CPU design

↓

circuit design

↓

chip layout

MIPS architecture

— all instruction have 3 operands.

— operand order is fixed

code : A = B + C

MIPS : add $s0, $s1, $s2

32 operands allowed = 32 registers

32 bit address :-

2^{32} bytes : byte add. : 0 to $2^{32} - 1$

2^{30} words : byte add. : 0, 4, 8, $2^{32} - 4$.

Big endian :

0	1	2	3
4	5	6	7

3	2	1	0
7	6	5	4

little endian :

3	2	1	4

: non-aligned word.

Data transfer btw register & memory

load and store

C code : A[8] = h + A[8]

```
lw   $t0, 32($s3)
add  $t0, $s2, $t0
sw   $t0, 32($s3)
```

machine language :-

add $t0, $s1, $s2

6	5	5	5	5	6
000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

32 bit

- op: handle instructions of operation but max = $2^6 = 64$
- rs: source
- rt: third
- rd: register - some will have same destination.
- shamt: unused (shift amt)
- funct: extension of op

lw / sw : new format — I - type

e.g. lw $t0, 32($s2)

35	18	8	32
op	rs	rt	16 bit number

only 6 bit used to define function

Conditional branch instructions

```
if(i==j)
h = i+j
```

⟹
```
bne $s0, $s1, Label
add $s3, $s0, $s1
```

```
if (i != j)
    h = i+j;
else
    h = i-j;
```

```
beq $s4, $s5, Lab1
add $s3, $s4, $s5
j Lab2
```

Comparison less or greater

set if less than

```
slt $t0, $s1, $s2
```

if $s1 < $s2 then

$t0 = 1

else

$t0 = 0

beq, bne — I-format

op rs rt 16 bit number.

j — I-format

op 26 bit number.

slt — R-format

op rs rt rd shmt funct.

Constants :-

I format

```
addi $29, $29, 4
slti $8, $18, 10
andi $29, $29, 6
```

'0' : special constant
↳ $0 (hard-wired into register)

add $s2, $s1, $zero } means move from $s1 to $s

for 32 bit constants] 2 instructions required : using

lui → load upper intermediate → then → ori → move them right