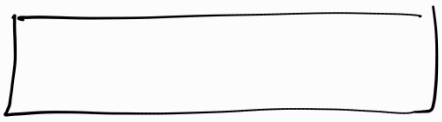
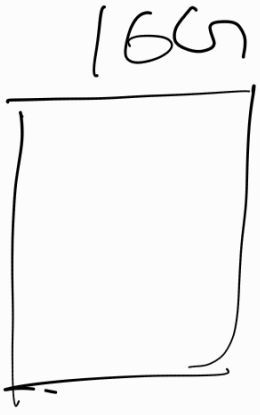
5/3/25

ch 13

challenge

How to address 

3 opcodes and their dests ?



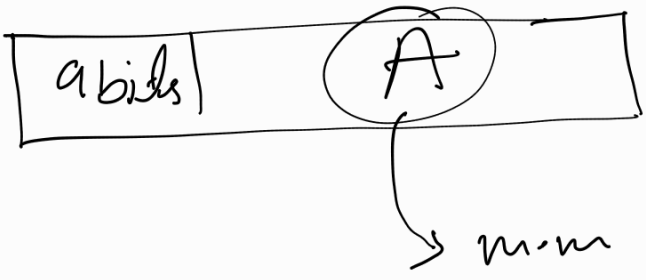
$= 2^{34}$

$= 34$ bit

We need addressing modes

1. Immediate Add. (ते पाहिजे)

2. Direct



diff

Immediate has → operands add

Direct " → mm add.

In direct

the opcode tells u to do

task

u fetch the value from m.m.

Notation

A = memory Address

R = Register "

EA = effective "

(address where your operand address.)

∴ EA = A (cause operand is stored in A)

(X) = content of X

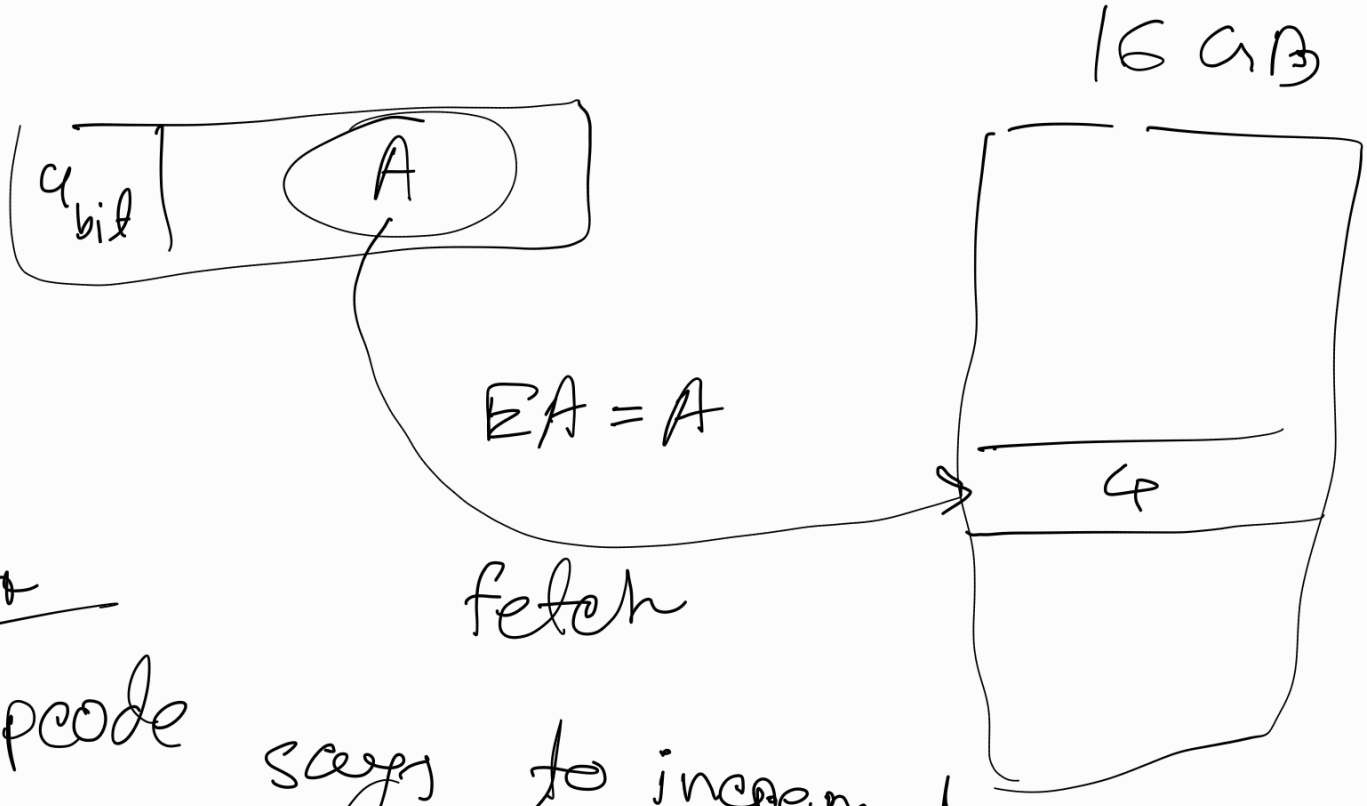
X could be A
X could be R

the number

say in add A, 4 is stored

so (A) = 4.

Direct Add



Ex

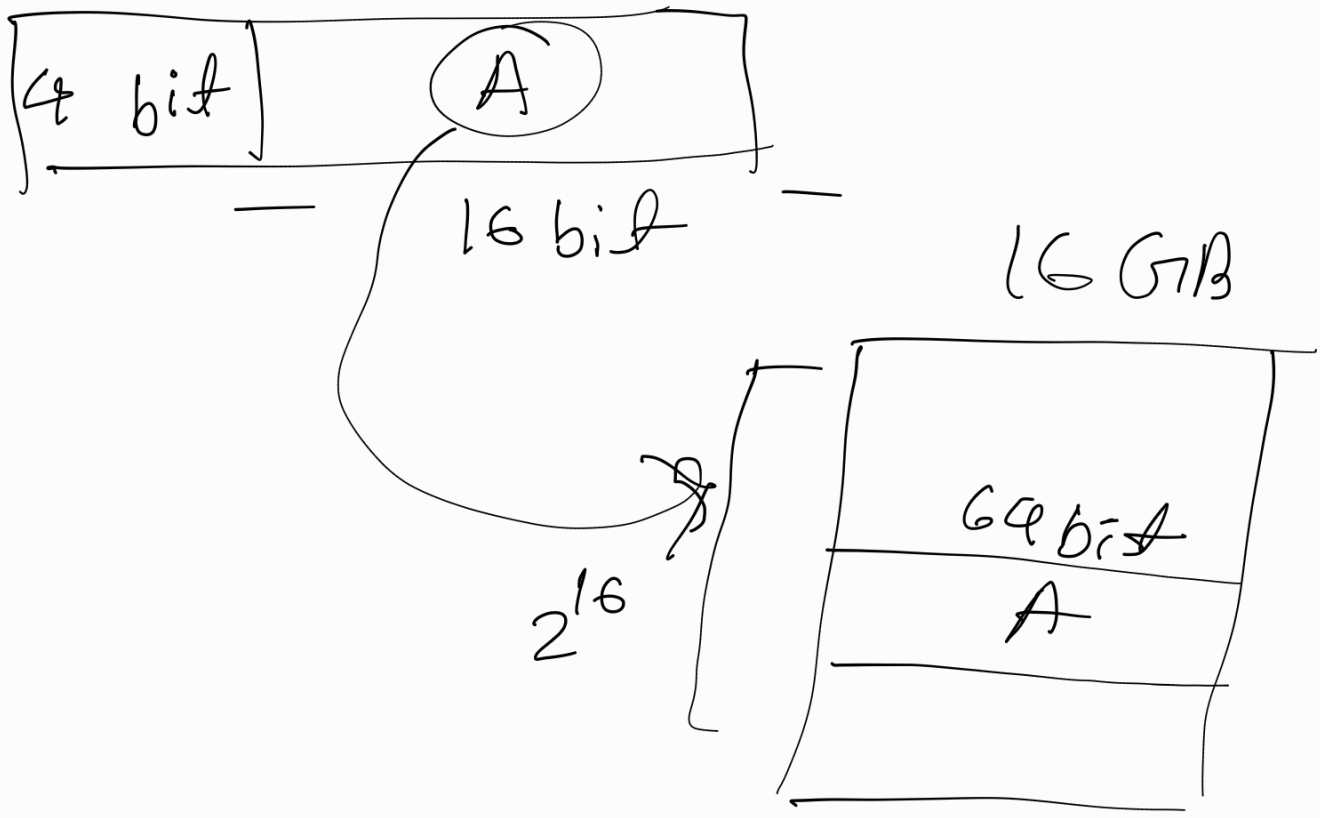
fetch opcode says to increment

calculate EA, = A

fetch data, then increment if

Result = 4 + 1 = 5.

Limitation of Direct Add.

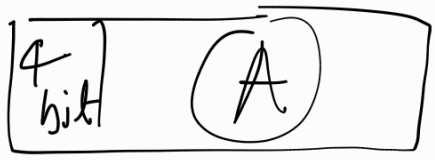


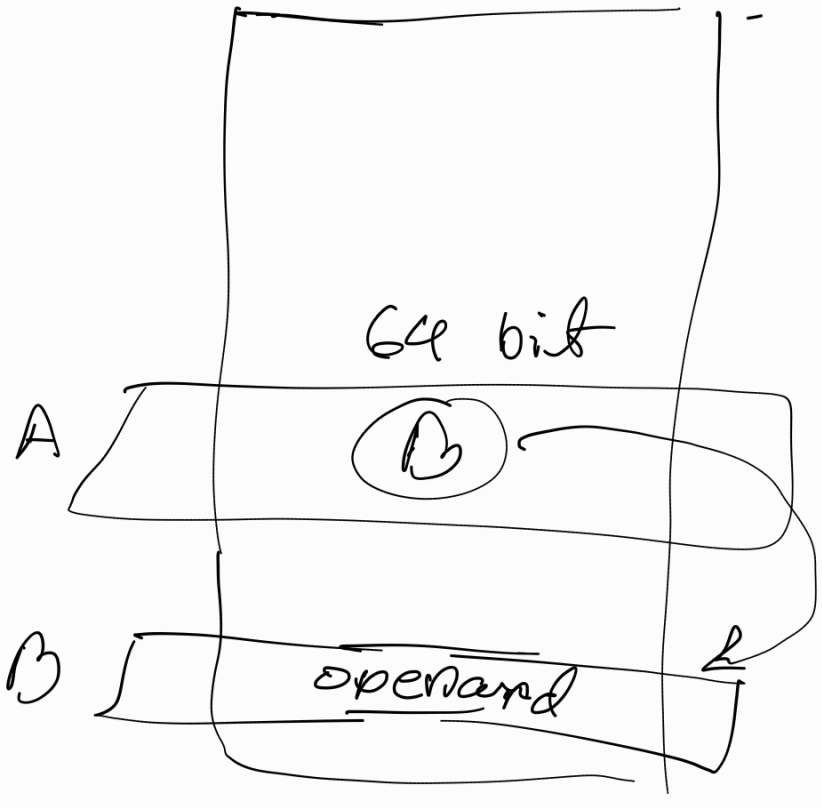
we can access to 2^{16} add for Direct Add.

Benefit

with 16 bit, I can go to the 64 u

Indirect addressing Mode





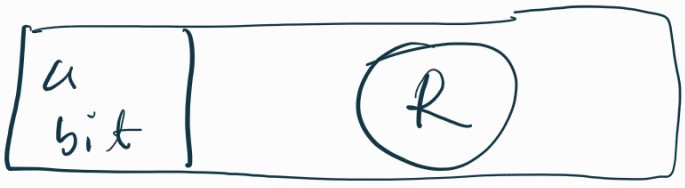
EA = (A)

= Address of B

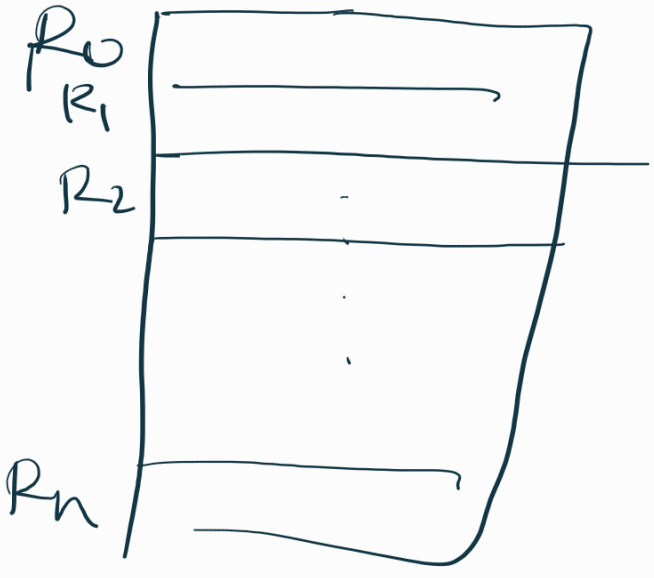
Benefit

previously A can access only 2^{16}, now we can do 2^{64}.

Register Addressing Mode



EA = R



fetch operands from registers

Benefit

(1) very fast, cause no need to access M.M.

(2) bit required is 3 – 5

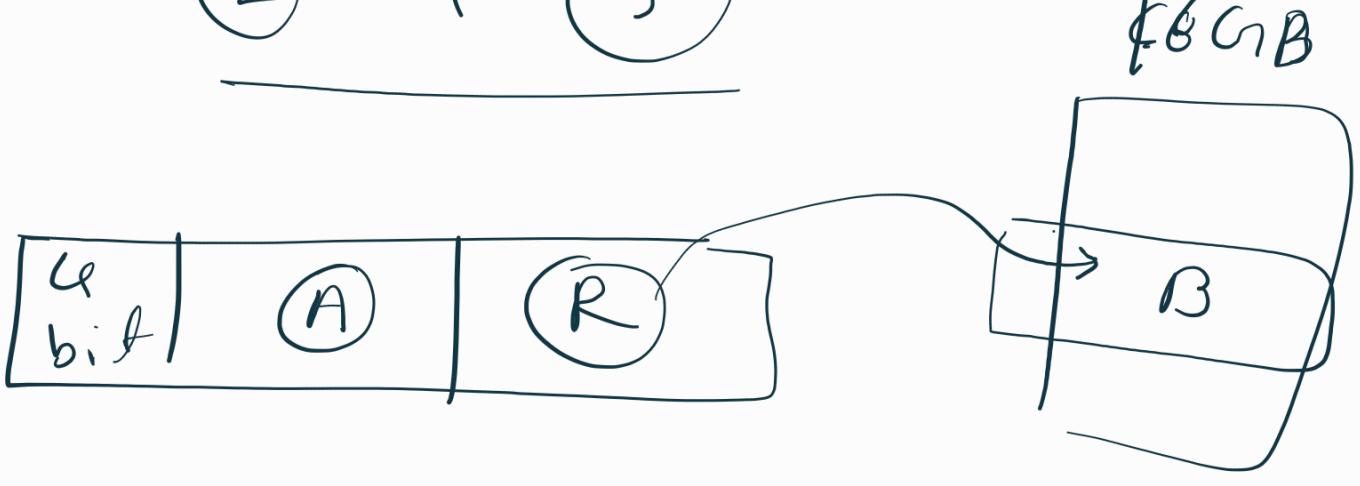
Register indirect

EA = (R)

R is not operand but the add. of the operand

Displacement (very powerful)

(2) + (5)



EA = A + B

= A + (R)

Displacement → Locality of reference

→ Relative ⟶ A + (PC)

→ Base - peg → A + (Base)
↓
offset

→ Indexing ⟶ u know the first add of the array.

So A + (Index Reg.)
↓ ↓
array index offset

Stack

There will be ZERO bits allocated for addressing.

You will just use the top 2 of stacks

How will processor know which type of Addressing Mode is used

---

We need 3 bits (cause 7 types)

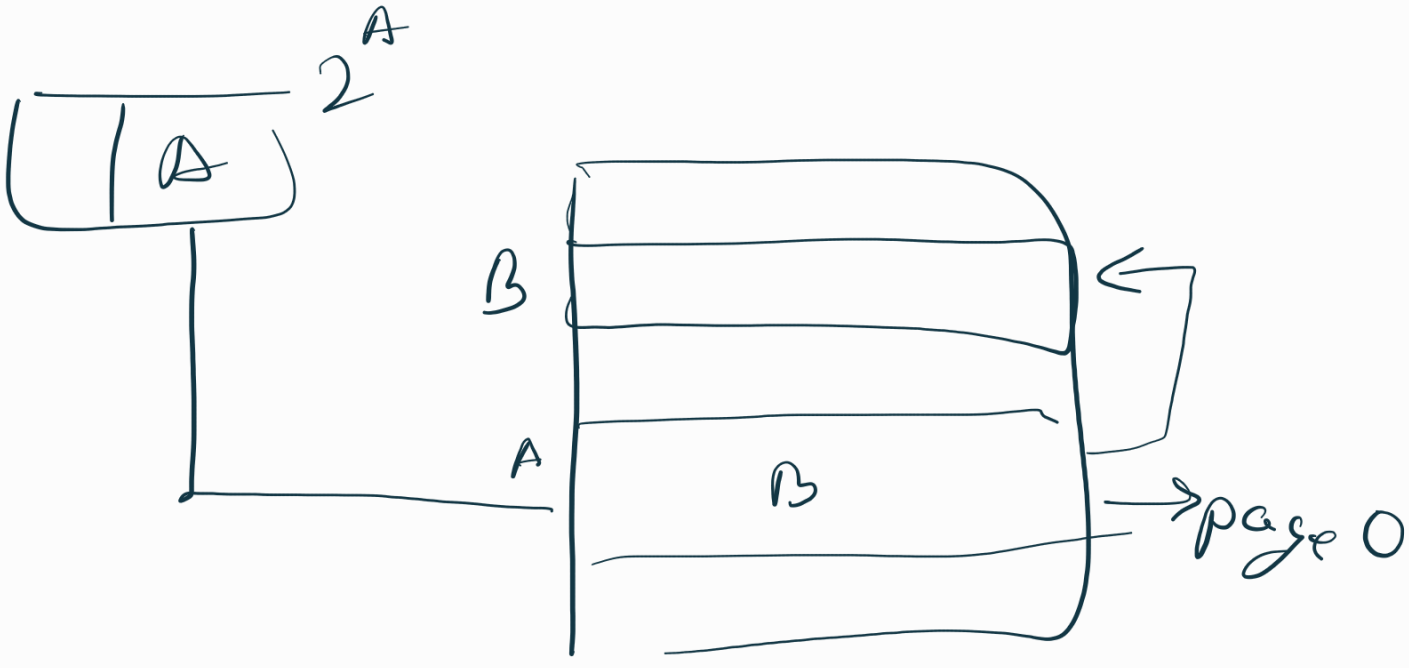
it's called mode fill

bits wasted, need mapping

---

we will always use Immediate add. for t_

For Indirect Add.



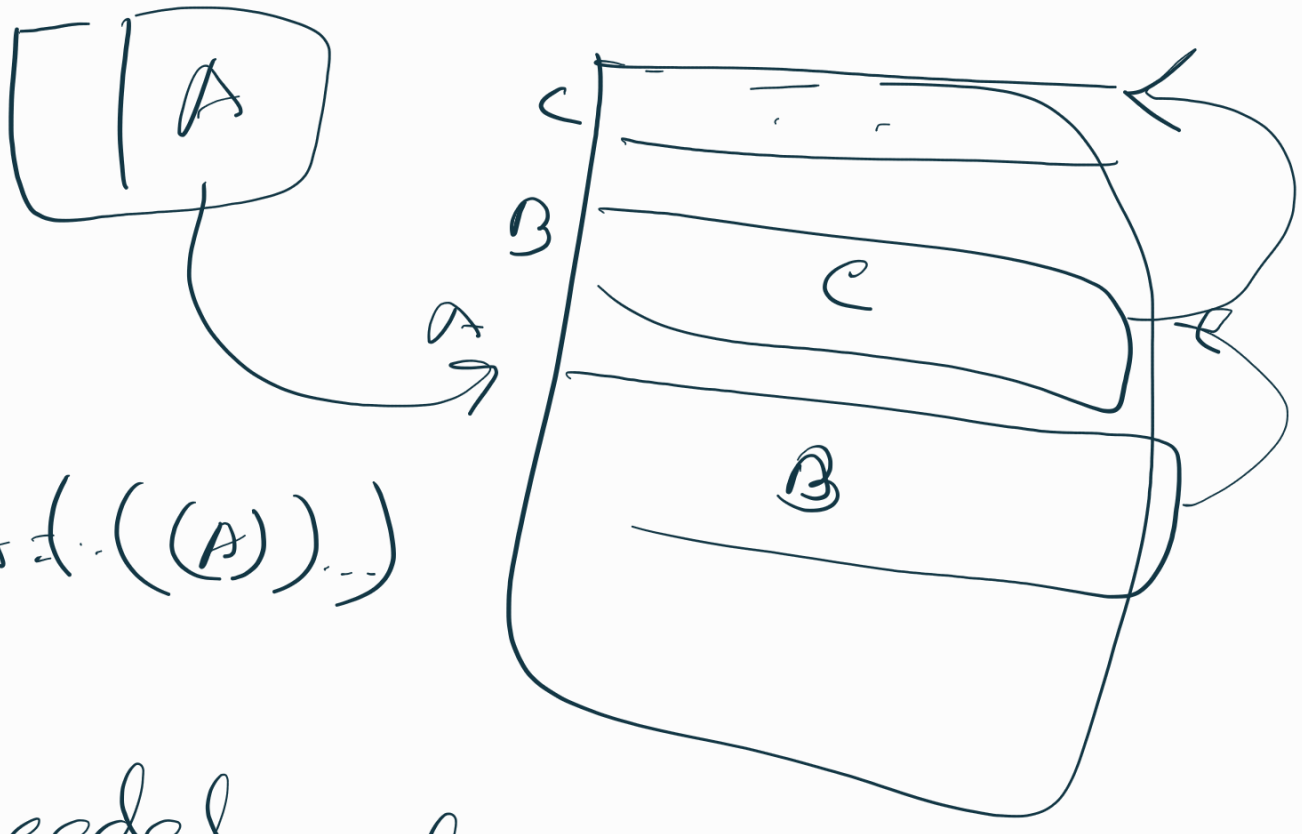
how many cache miss?

=> single.

the processor doesn't know page

So, we need page 0 > 2^A

multilevel Indirection



EA = (((A)))

needed for pointers.

H.W

locality → spatial
locality → temporal

(Internal memory)

in case of Indexing Displacement

array

(A) + (IR)

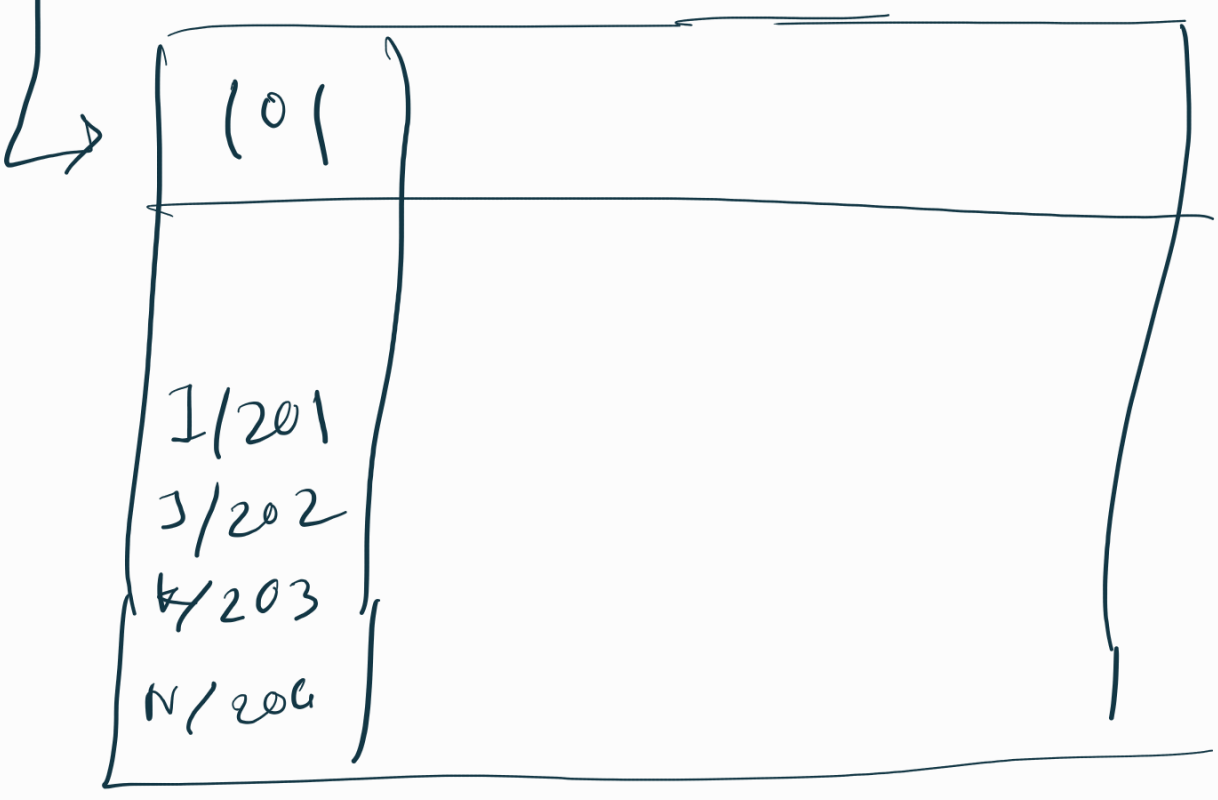
what if, not Operand but address?

A will be calculated first

EA =

এইটা বইয়ে আছে, বই পড়বি.

$N = I + J + K$



Low level instruction

Direct Add.

Load 0010 I → AC

Add AC, J

Add AC, K

Store AC → N