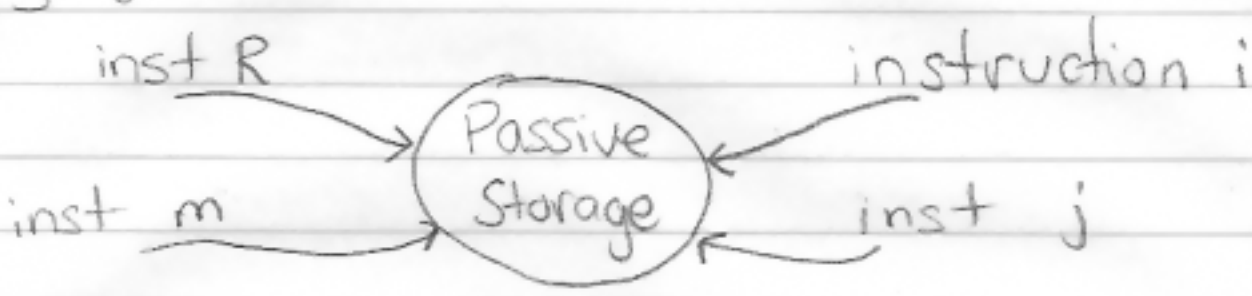
CS 3725 Mar. 15/12

Falling Into The "Address Pit"

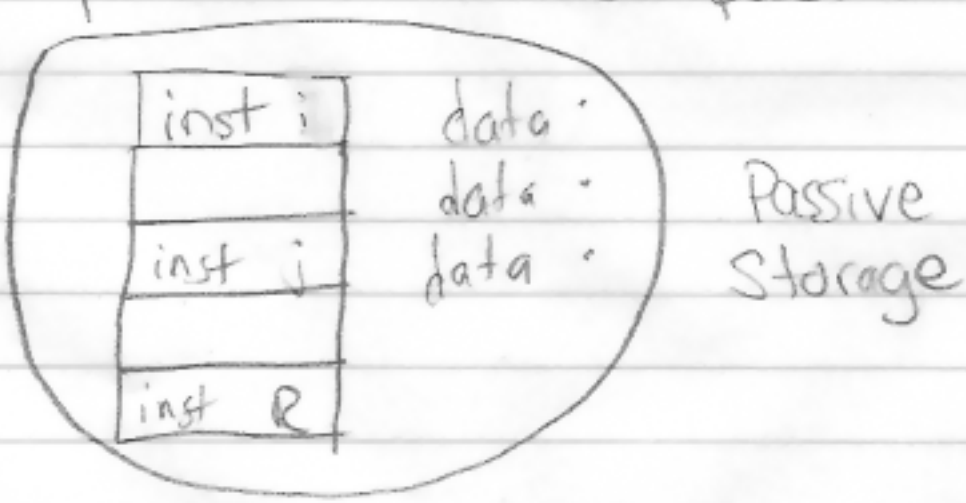
[A] "Need address to find an address and so on"

(I) Fundamental idea in Vi N computation is:



instructions $i, j, r, m \in K$ will access the passive storage and will modify the content of the passive storage.

(II) But, the instructions themselves are part of the passive storage



Instructions of this is that
- instructions and data in the passive storage indistinguishable
- which in turn implies that an instruction can modify another instruction! (self modifying program)
- So we need to find the address of the instruction which in turn has to find the address of the data which it has (to read/to modify)

- Now, the address of the instruction is to be found by the PC. But PC is also a part of the passive storage.
- So we need to find the address of the PC. And so on...

(IV) Steps one has to take to go thrugh to successfully complete an instruction and then get to the next
a) The address of the next instruction ≡ the content of the PC
b) But more importantly the content of the PC is formed as an <u>after-effect</u> of the behaviour of the current instruction.
c) To reduce the "overhead", we force one setup to place instructions in subsequent locations. But physically, consecutive instructions do not mean that they are logically consecutive.

<u>Bottom Line in V. N</u>
- Passive storage
- Sequential control of instruction execution
- One-word-at-a-time transfer between the storage and the rest of the system

[B] Relationship among Language Construct Programming Machine

Level Ø

Binding: Addresses (name) to storage cells

- Language Declarations
  ie. var A : Array [1...10] of Integer
- Machine Variable "A" defines a subspace of passive storage and
  implied: - How the storage locations are physically connected
  Does not have any meaningful (mathematical notion of the values in the structure)

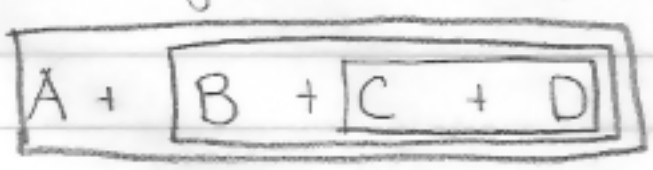
Imperative Program

- The domain and no range of the program: storage
- Purpose of the "program" is simply store-to-store transformation

Level 1

Binding
Locations to Values

- Language EXPRESSIONS
  eg. X[i] + B * C
- Machine Function of the ALU
  implied: Sequential Computation



eg. Is A+B ≡ B+A ? what about context!
"x+f(x) ≡ f(x)+x" f(x) could change x!

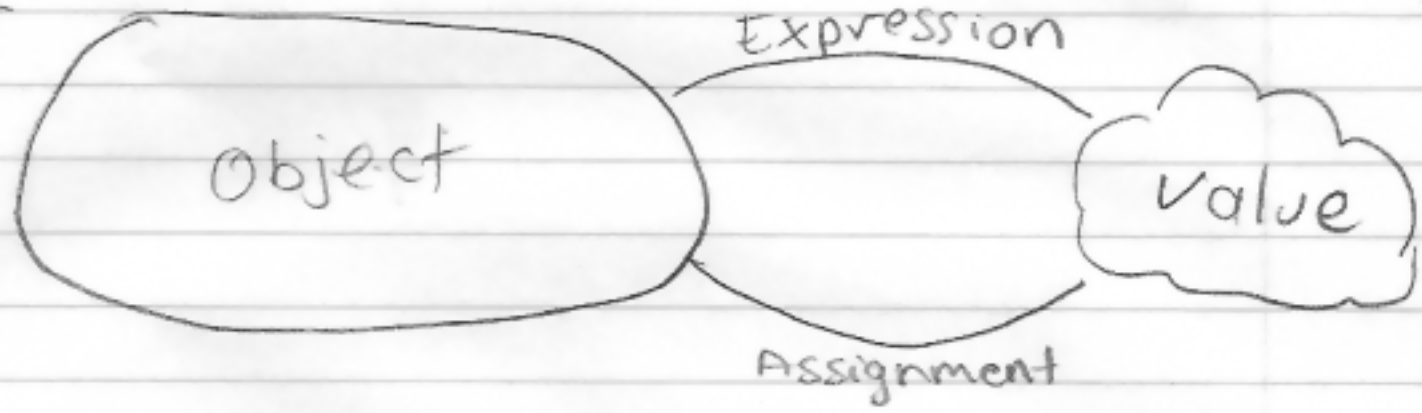
Level 2

Binding values to location

Language — Assignment statement
ie. C := A[i] + B + C

Machine — Modifing the Storage
ie. Change the current state to the next state

Program



Hence, this dictate that such modifications of state are done in strict order.

Consider

A[i] := B+C
...
A[k] := D+F

} are they data dependent?

We don't know if i==k.
Not known by static analysis.
Needs dynamic (execution time) analysis.
Hard to figure out parallelism!