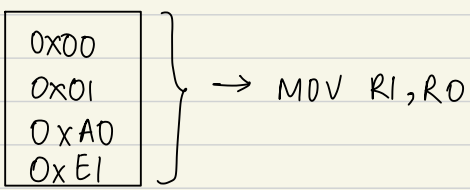
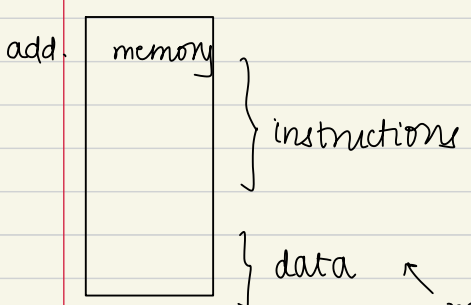
1. Characteristics of Instructions

- instructions are also stored in binary patterns called machine code
- for us, they are represented as mneumonics
- ARM (32-bit CPU) eg.



- instructions must be saved sequentially



Von neumann executive model
or
Turing machine

↖ memory is partitioned into separate code and data segments

100
011
111100
C
A
B

2. ARM Instruction Format

this is an example study

- MOV ≡ move instruction
  ↓
  takes 2 operand ; copies source to destination

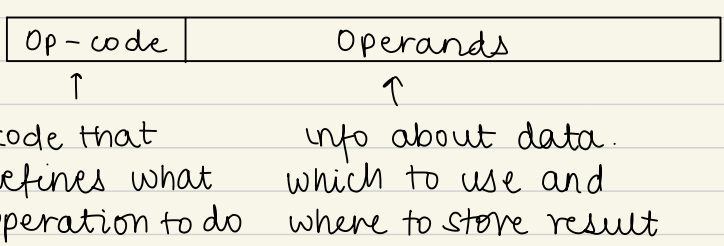
∴ MOV R1, R0
  ↑ dest ↑ source

R0	0x1234	→ R0	0x1234
R1	0x000	R1	0x1234

- here, both operands are registers

3. Instruction Organisation in memory

- machine code is executable, unlike data
- converting instructions into machine code is called encoding
  CPU reads the encoded instructions
- instructions format



- some operations simply change program flow or flag status rather than produce a storable result

4. Opcode

- n instructions need $\log_2 n$ bits to encode it
- more the variety of instructions, longer is the length of each instruction
- its a tradeoff b/w software and hardware
  = a multiplier will take up a lot of hardware
- the method by which an operand is specified is called addressing mode (encoding the operand.)
  ↳ a lot of ways

R0 - R12 are general purpose
R13
R14 is
R15 is
R14 is
R15 is
R14 is

is the link
is

R11 → FP
R13 → SP
R14 → LR
R15 → PC

Program Counter

- R15
- keeps it