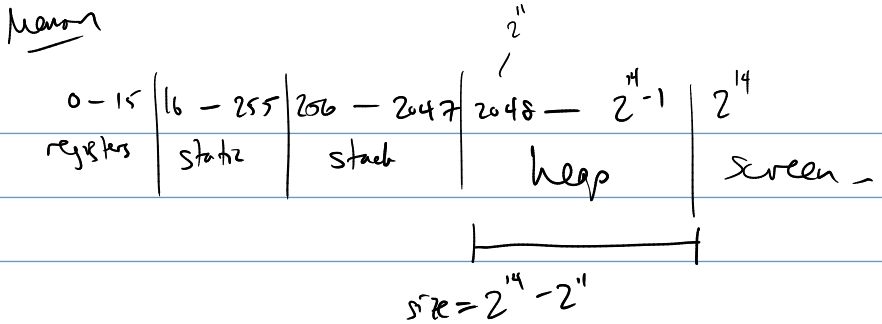
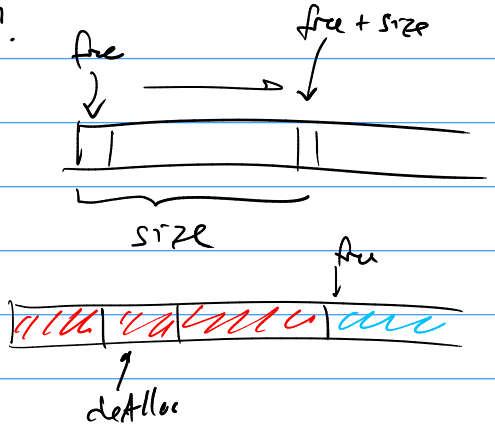
Memory allocation

Memory.alloc(size)
Memory.deAlloc(addr)



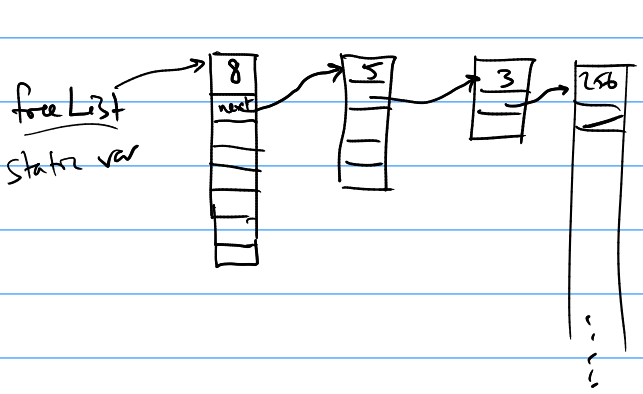
(1) Stupid way.

- keep track of address 'free' — init to 2^{11}.
- alloc(size):
  add size to free
  return the old value of free.
- deAlloc(addr):
  do nothing.

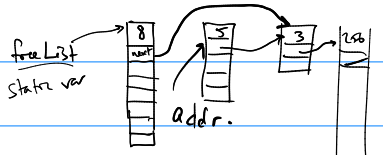


(2) Better way: linked list of free segments.

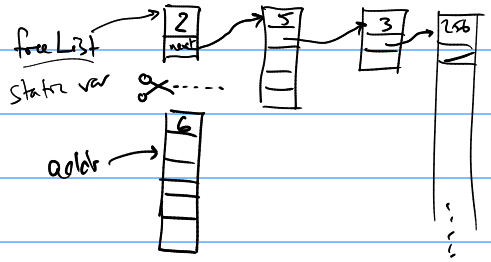
- initialize: one giant free segment.



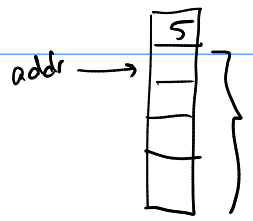
Scenario 1: request size 4.



Scenario 2: request size 5.

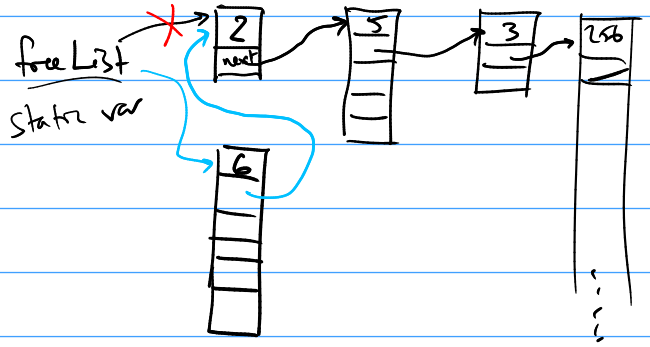


- alloc(size): find a free segment that is "big enough"
  - first fit: scan through + find first
  - best fit: find the smallest that is big enough.



- Return addr to the code that called alloc
- Ensure the location before addr has the size of the segment.

dealloc: add deallocied segment to start of freeList.



Randomness.

What is randomness?

- sufficiently unpredictable (seems unpredictable?)
- not feasible to predict.
- typically we talk about randomness of a sequence of numbers.

- sequence where we can't predict next based on previous.

Applications?

- Games — random events, procedural generation.
- Online gambling. ←
- Random sampling.
- Simulate coins/dice etc.
- Cryptography. ←

Sources of randomness

① Hardware / environmental / physical sources.

Suitable for cryptography, online gambling etc.

- using an unpredictable physical process
  dice, coins, interstellar radio signals, lava lamps, thermal noise, etc.
  embedded HW devices for generating randomness.

② Pseudo-random number generation (PRNG).

Sometimes we want sequence that looks unpredictable but is actually deterministic/repeatable.

One simple kind of PRNG is a linear congruential generator (LCG):

$$X_{n+1} = (a \cdot X_n + c) \bmod m.$$

Java Random uses an LCG with

$$m = 2^{48}, \quad c = 11, \quad a = 5DEECE66D_{16}$$

Python uses Mersenne Twister.