Effektiv Kode med C og C++

Forelesning 2, vår 2014

Alfred Bratterud

Agenda:

* Plan for kurset
* Datatyper: fra byte til objekt
* Pekere og Type safety
* Roulette shootout

Plan for kurset

* Oblig1
* Oblig2
* Prøve
* Prosjektoppgave

Datatyper fra byte til objekt

Viktig viktig pensum til prøven - som må sitte i fingrene!

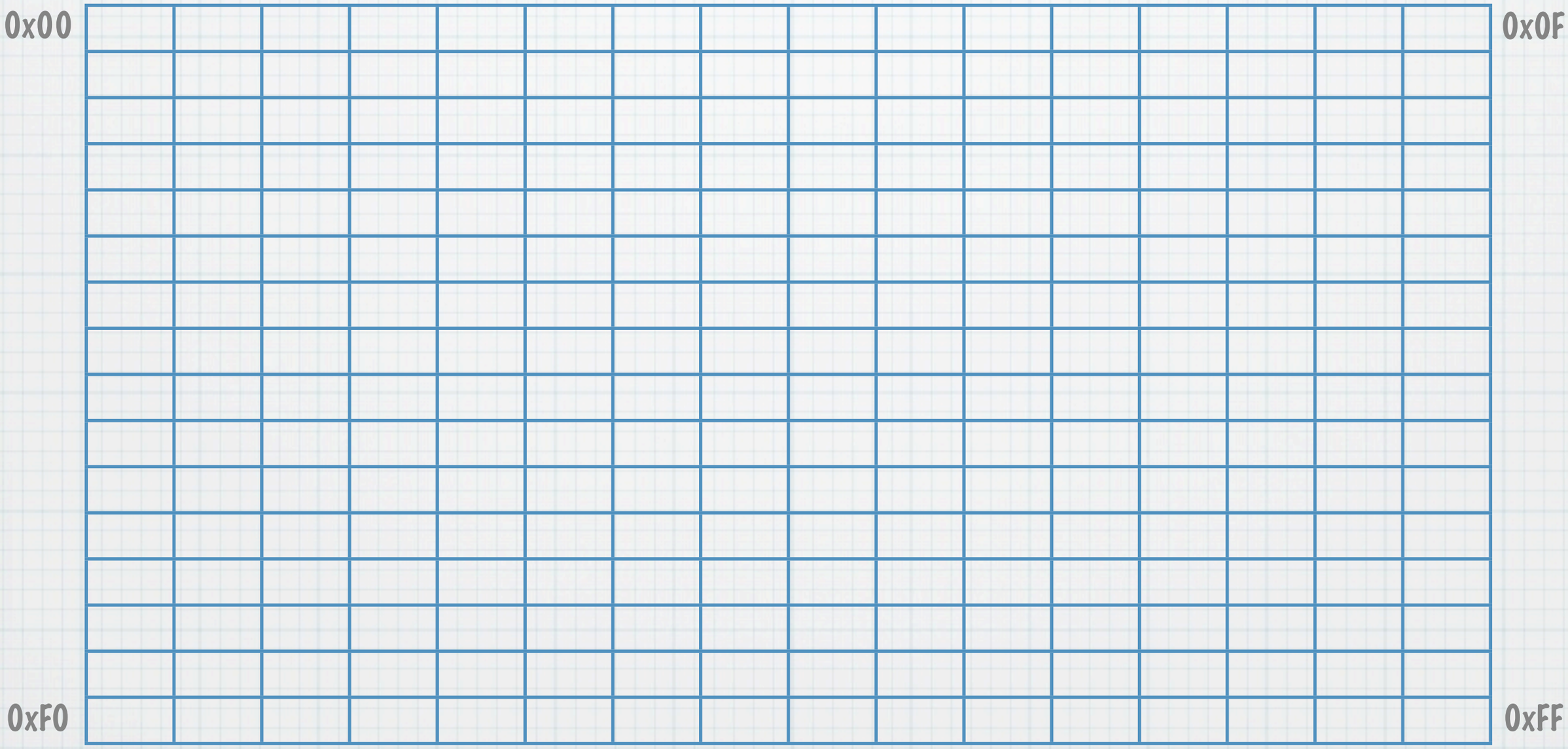
Pensum:

Kap.3 (Og alle oppgaver herfra)

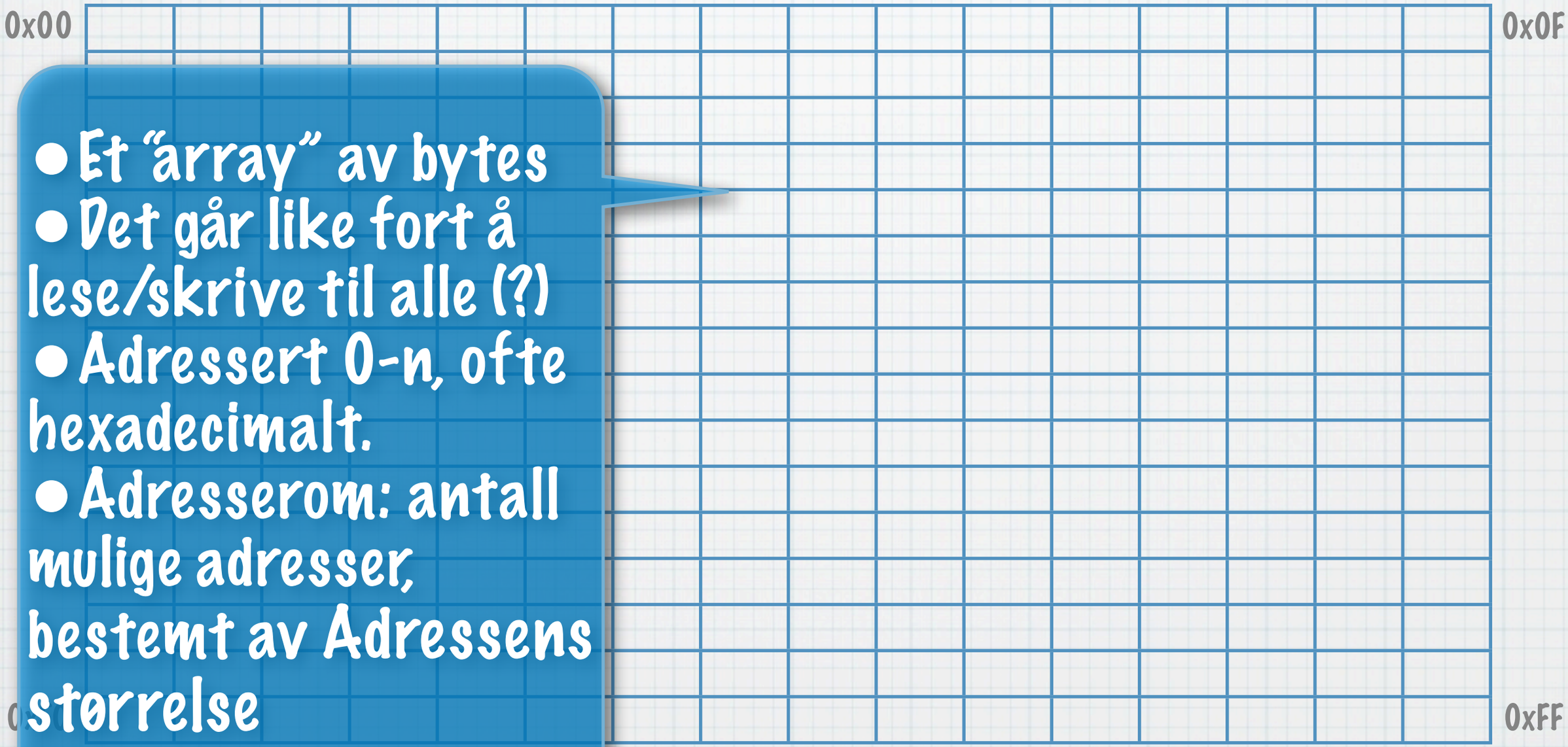
Kap. A.5.7

Kap. A.8

Random Access Memory



Random Access Memory



- Et "array" av bytes
- Det går like fort å lese/skrive til alle (?)
- Adressert 0-n, ofte hexadecimalt.
- Adresserom: antall mulige adresser, bestemt av Adressens størrelse

Random Access Memory

0x00

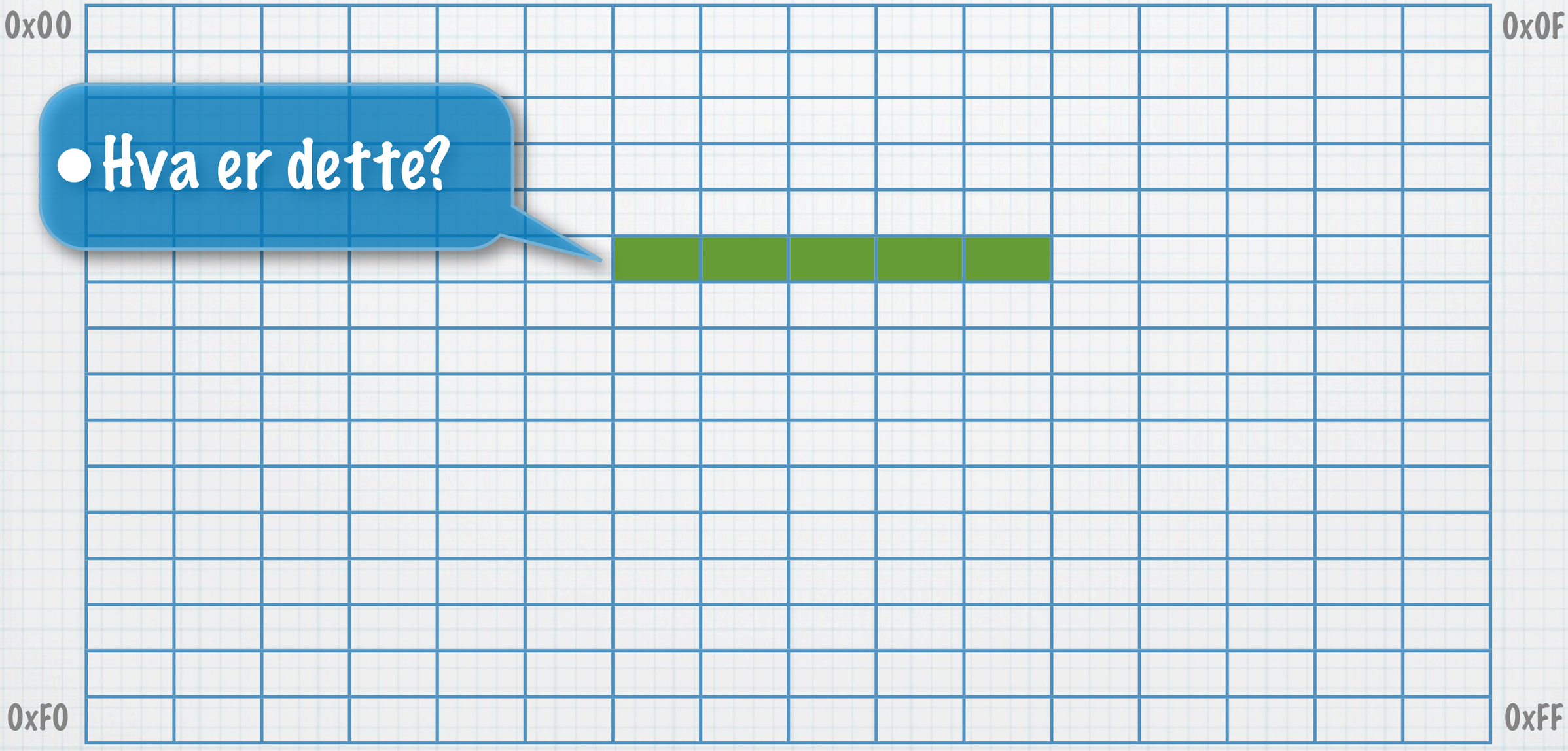
0x0F

- Et "array" av bytes
- Det går like fort å lese/skrive til alle (?)
- Adressert 0-n, ofte hexadecimalt.
- Adresserom: antall mulige adresser, bestemt av Adressens størrelse

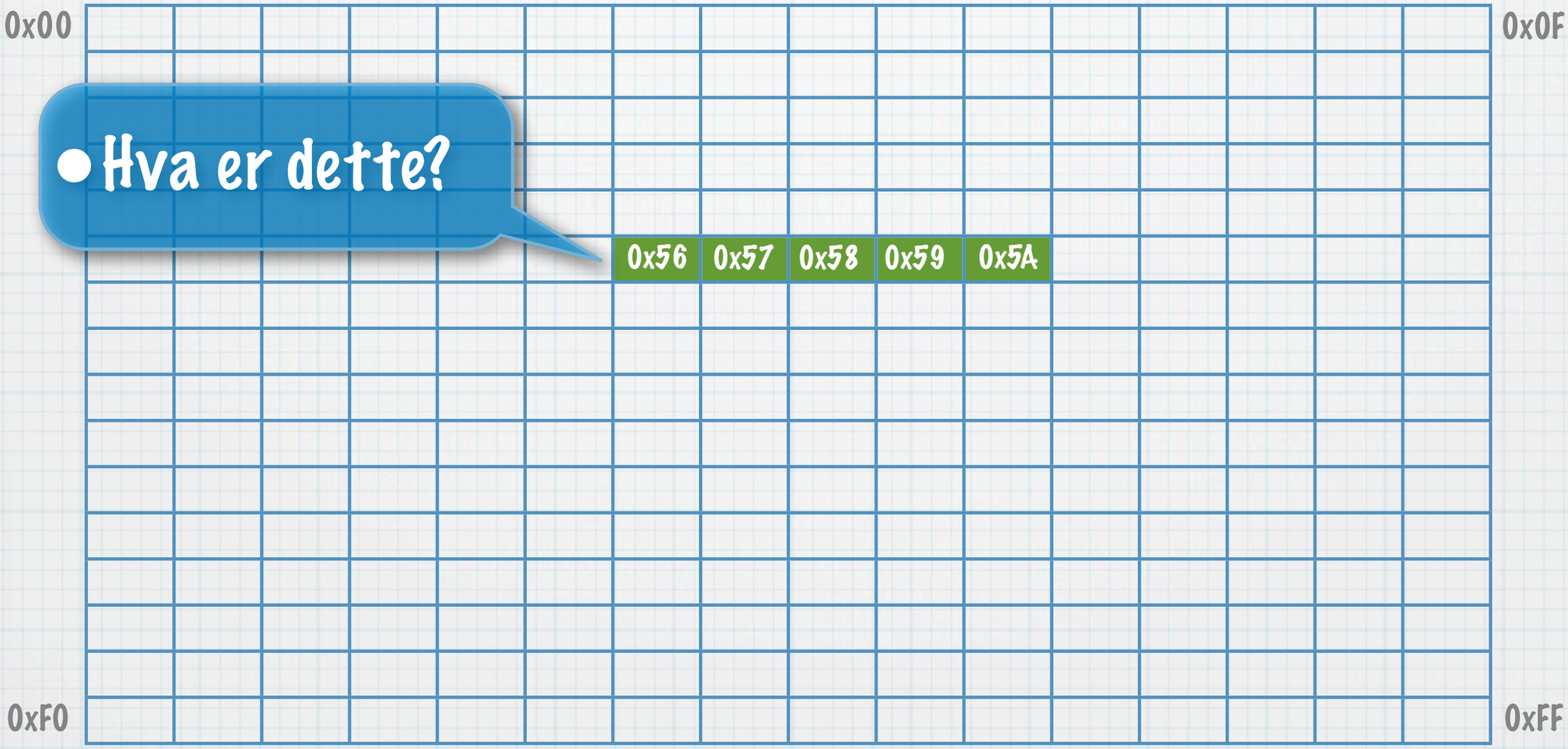
- Hvorfor hex?
- Bare "convenience"
- 16*16=256
- $=2^8$
- =1 byte
- 1 byte adresserom gir 256 adresser
- hver adresse, to "siffer" (hex)

0xFF

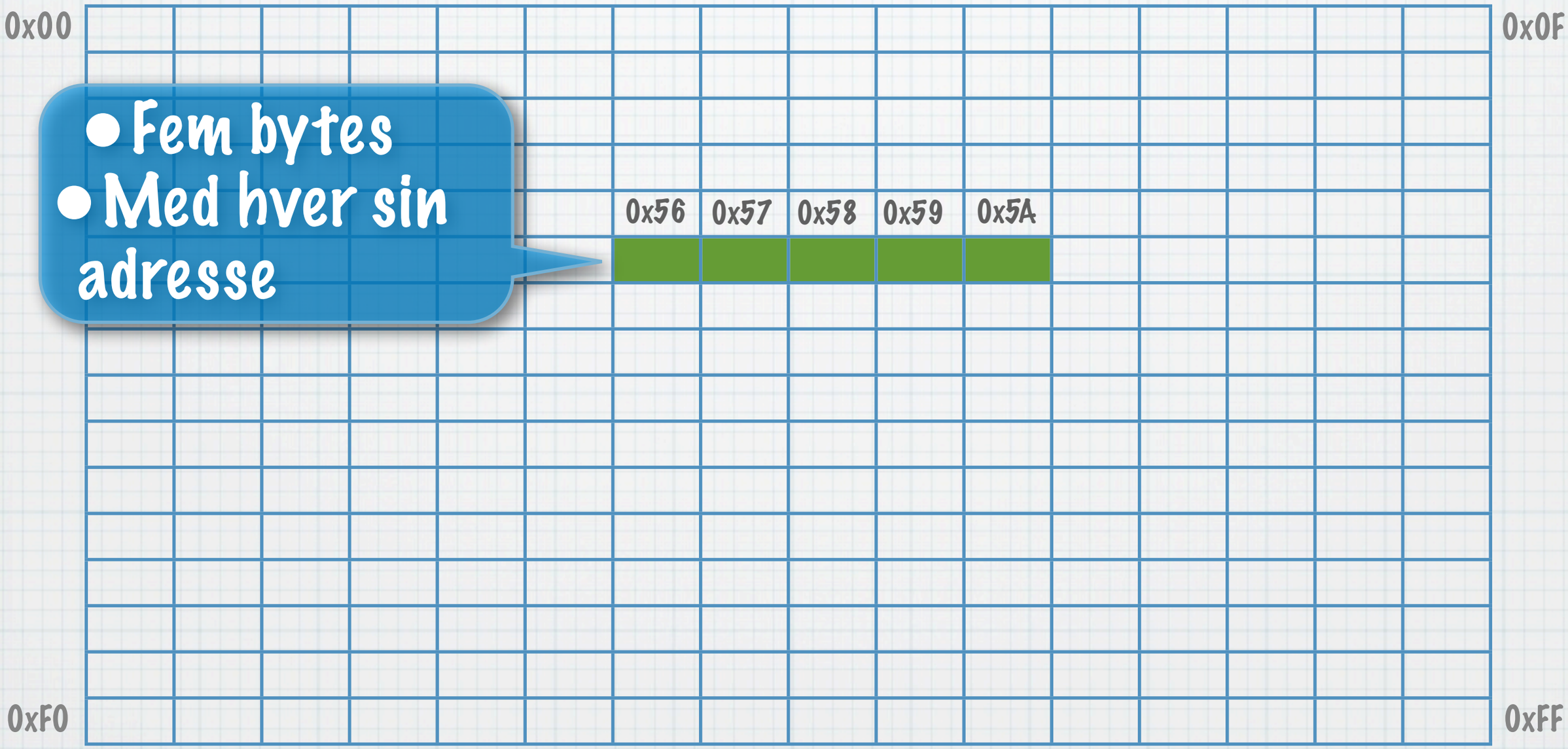
Random Access Memory



Random Access Memory

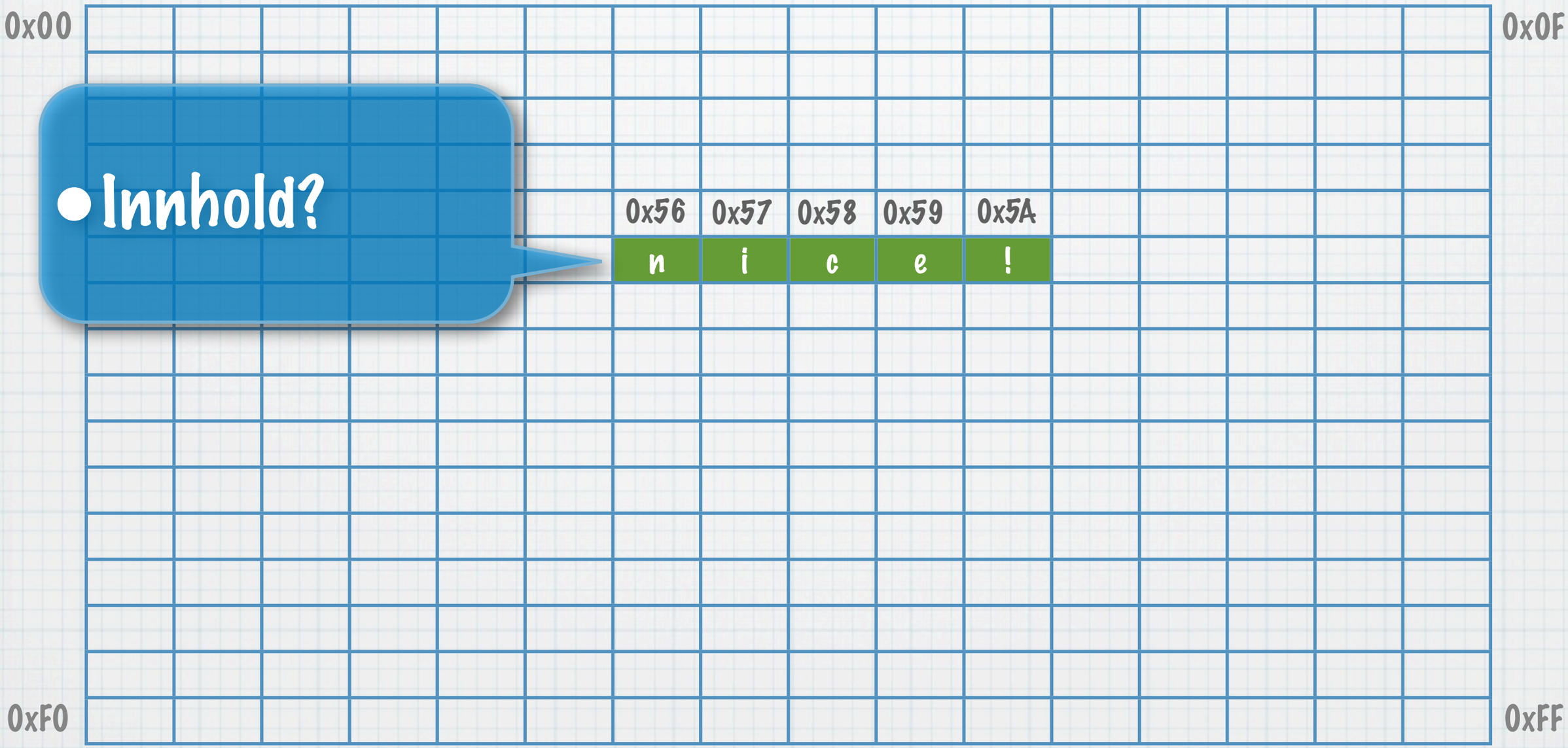


Random Access Memory

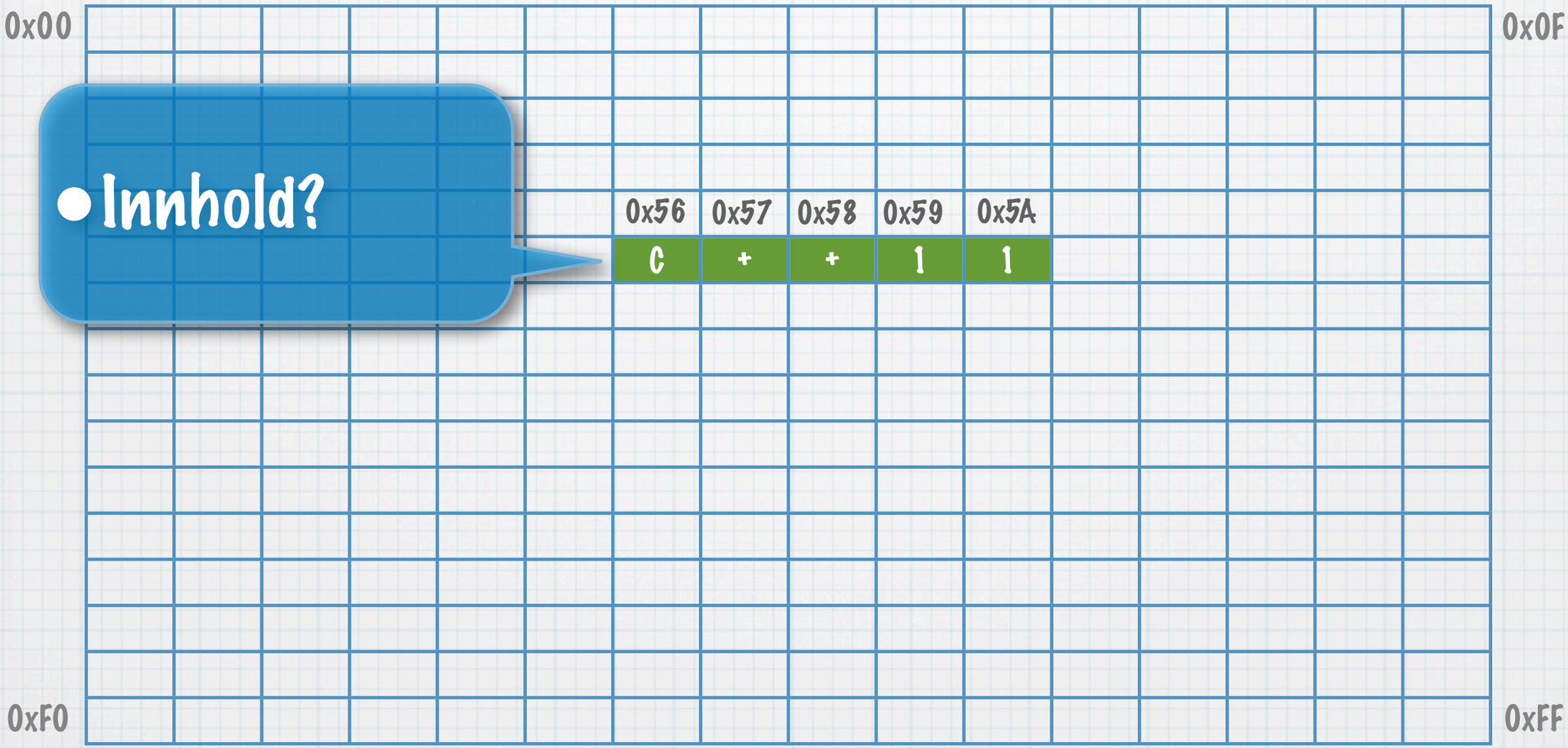


- Fem bytes
- Med hver sin adresse

Random Access Memory



Random Access Memory



Hva er en datatype?

* Én av flere måter å lagre data i minne på
* Et antall bytes sett som en helhet
  - Primitive typer
* En sammensetting av primitive typer, sett som en helhet
  - Kompositte typer
* For (primitive) typer med lik størrelse? "compile-time" regler for "oppførsel"

Primitive (Built-in) typer

Navn	Beskrivelse	Størrelse*
char	'a', 'b', 'c', etc.	1 byte
short (int)	Lite tall, 0-65535*	2 byte*
int	"vanlig tall"	4 byte*
long (int)	Stort tall,	4 byte*
bool	true / false	1 byte
float	kommatall	4 byte*
double	Dobbel presisjon float	8 byte*
long double	"Stor" double	8 byte*

* Gjelder 32-bit system. Flere detaljer: http://cplusplus.com/doc/tutorial/variables/

Noen begreper

* En datatype er en samling bytes eller "mulige verdier" (og operasjoner, for objekter)
* Et objekt er et område i minne, med en verdi av en bestemt type
* En variabel er et navngitt objekt
* En deklarasjon gir navn til objektet
* En definisjon setter av minne til et objekt

Eksempler på typer

* int, float, char - primitive typer
* Struct, class - kompositte typer
* Pekere og referanser - minneadresser - også typer
* Strømmer, Containere etc. - objekter
* sizeof(type) gir størrelse i minnet
* Alle funksjoner må ha en returtype - og typede argumenter
* Er en funksjon en type?

Mer finmaskede typer

* "signed" og "unsigned" kan settes foran primitive typer for å avgjøre om de skal ha fortegn (+/-)
  * Bruker en bit
* Long og double, angir størrelse på hhv. int og float
  * Som oftest "dobbelt så stor" men ikke alltid
  * I C++11 garanterer "long long" 64 bit
* "Const" gjør en variabel "read only"
* sizeof(x) gir "størrelse i bytes" av x... noen ganger

Typekonvertering

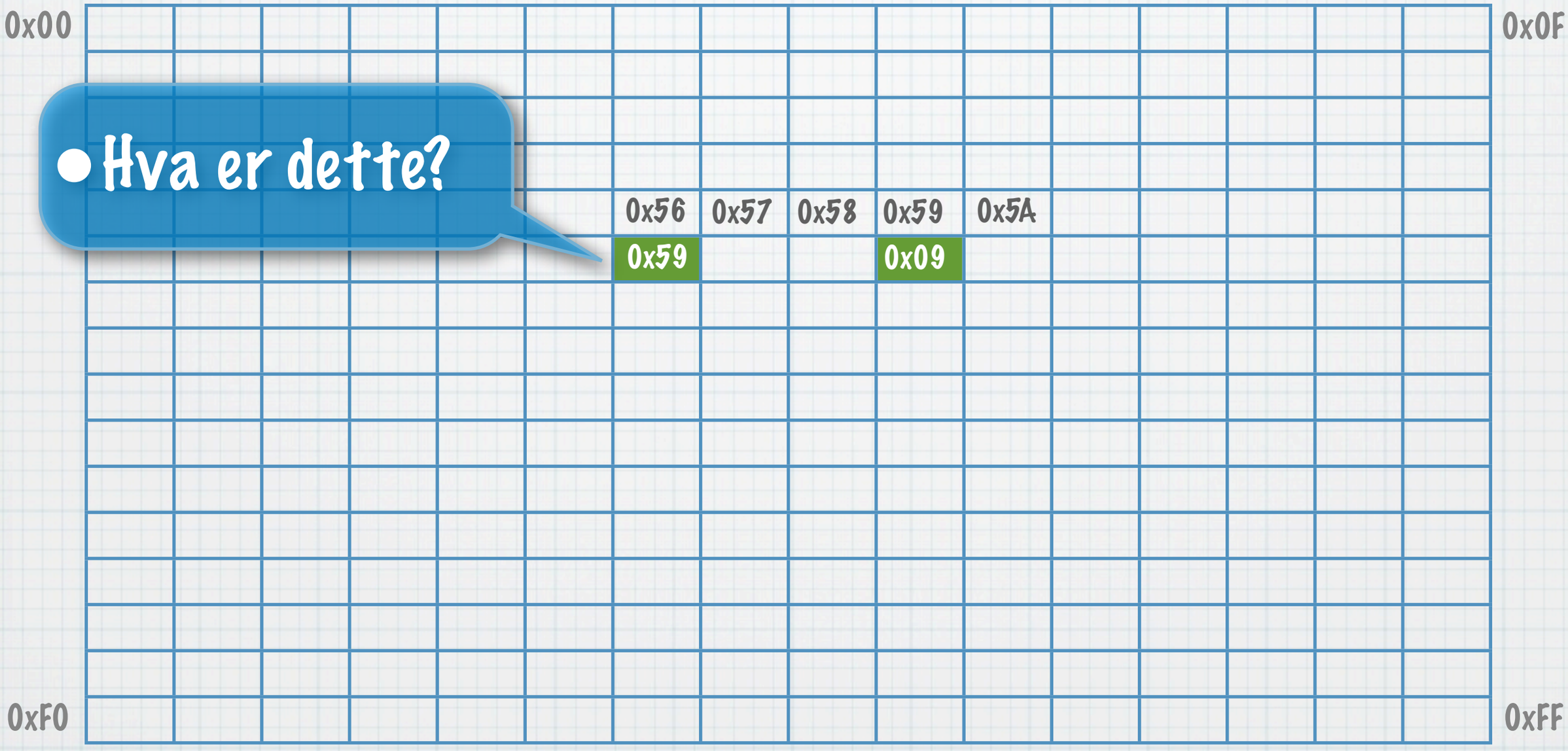
- "Implicit conversion" betyr at man kan endre type uten "å si fra eksplisitt":
  - int x=8;
  - float y=x; //OK
  - y+=0.1; //OK
  - x=y; //OK ... men hva nå?
- "Explicit conversion" krever en "type cast".
  - c-style cast, eller "type coersion" har syntax som i Java:
  - C++ har flere "trygge" varianter (senere)

Pekere

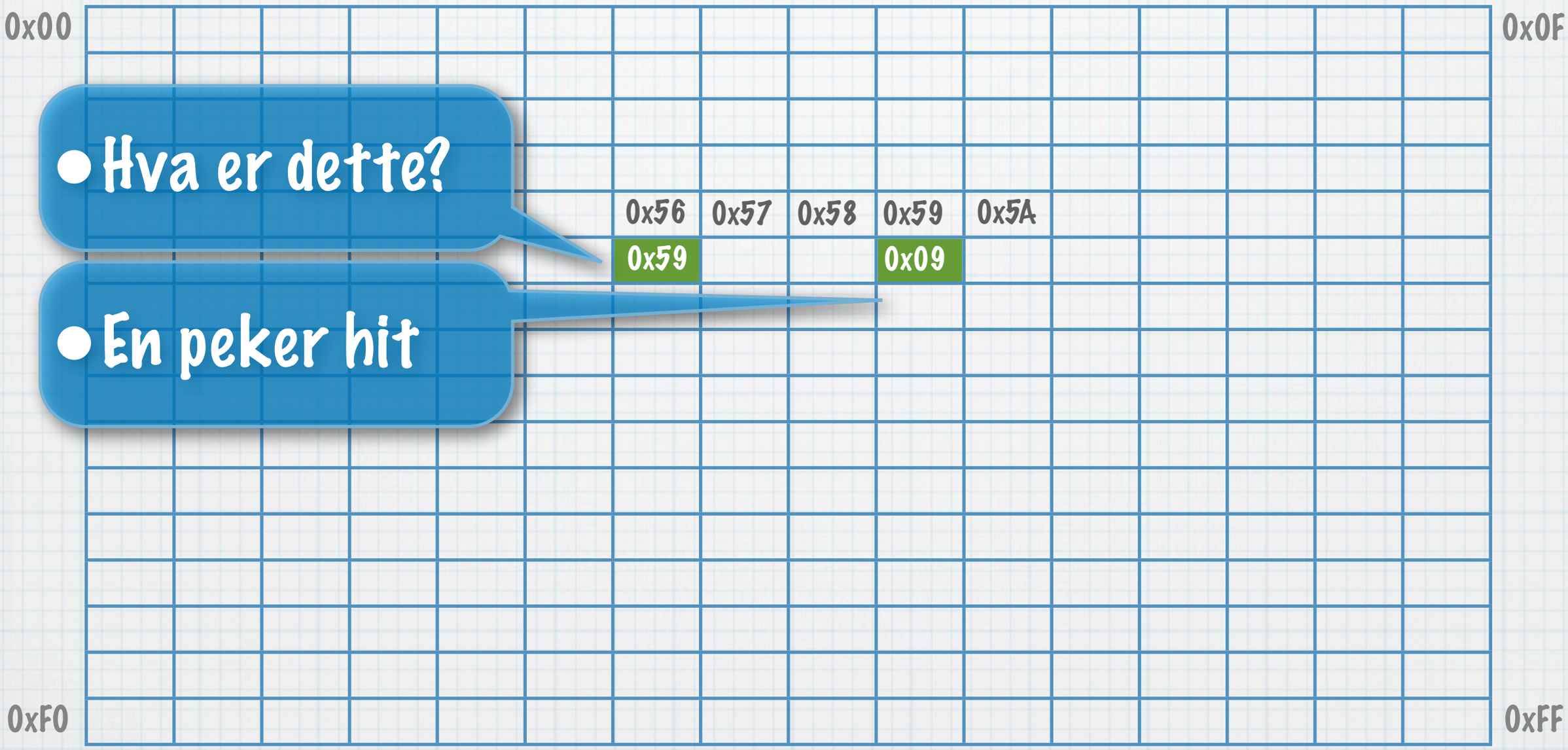
* For alle typer t gir " t* myData" en peker til et dataobjekt av typen t.

* Operatoren "*myData" (dereference) henter ut det myData peker på
* Operatoren "&myInt" (reference) henter ut adressen til der myInt ligger

Random Access Memory



Random Access Memory



Pekere (Forts.)

* Et array av typen "t" er nettopp en peker til første element
* [i] (index operator) må brukes i initialisering, og aksessering
* char* c er spesiell - en cstring
* Hvordan vet vi lengden på et array?

Type Safety

* Alle pekere kan endre type, feks. med c-style cast:
  int* a;
  char* b=(char*)a;
* Pekere kan også regnes med - de er jo bytes.
  int* i=myInt[5];
  i++;
  i==myInt[6];
* Dette gir mye kraft - og kan gjøre mye skade.