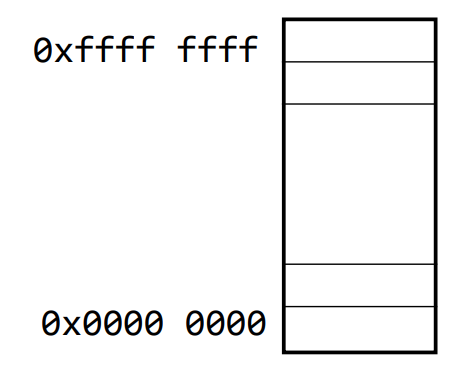
Outline:

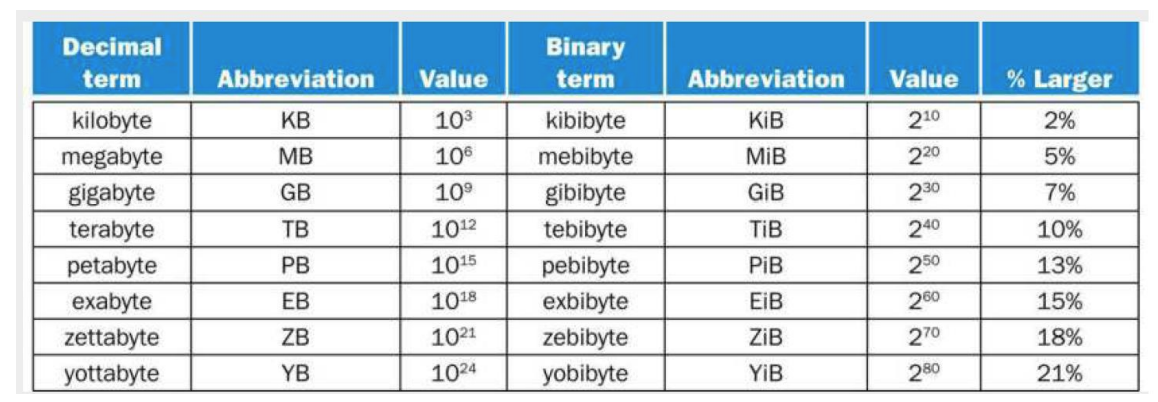
* Memory
* Load/Store instructions
  + Move data between registers and memory
* Data of other types
  + Words, halfwords, and bytes
  + ASCII strings
* Address alignment
* Endianness

Memory

* An Array of bytes
* Each byte is numbered. The number is the address
* Each address identifies a byte
  + If a data item is larger than one byte, its address is the first byte in memory
* A 32-bit address space supports 4Gib
  + A 64-address space supports 16 EiB (exbibytes)



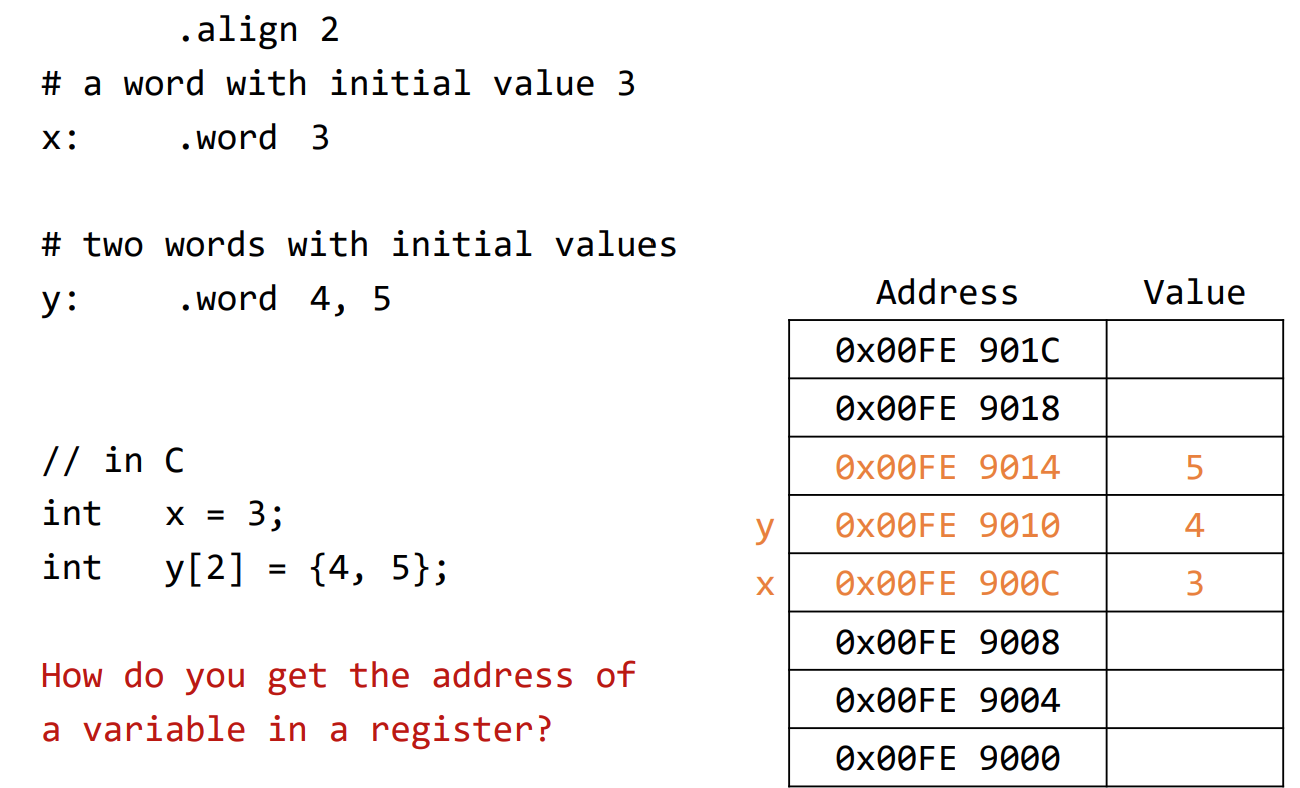
“Address of each byte is written in hex”



Using data in memory:

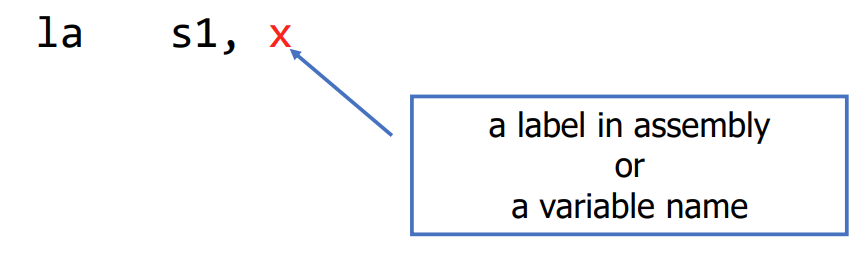
* Many ISAs like RISC-V cannot compute on data in memory directly
  + Must load data into a register first
    - x1,x2
* Two kinds of instructions to exchange data between registers and memory
  + Load: memory to register
  + Store: register to memory
* Need to know the address to read/write memory
  + You need an address to save/fetch items

Variables defined in your program:

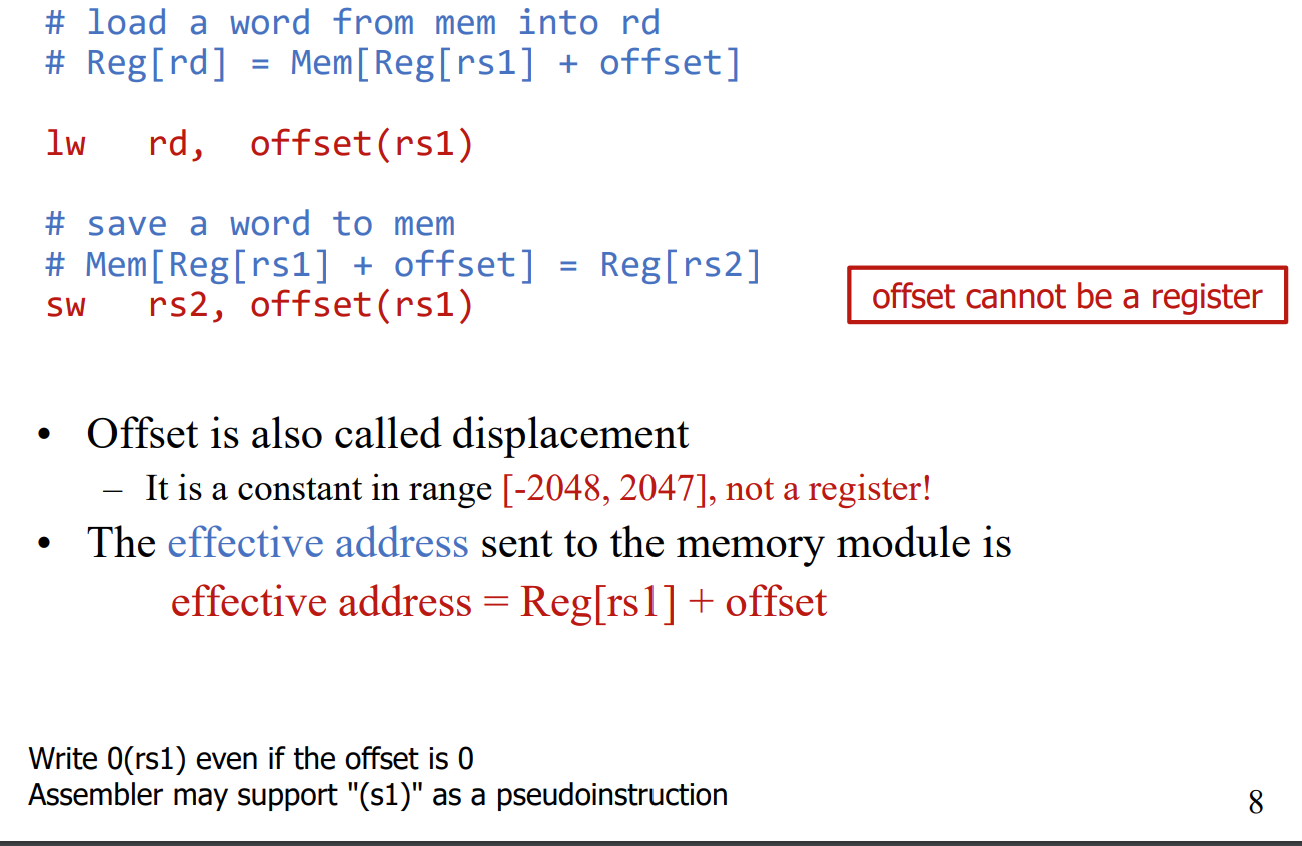


Getting the address of a variable in a register?

* Load a 32-bit constant in a register
* We can also use a pseudo instruction LA
  + It is converted into a couple of real instructions
  + We will learn the real instructions later



Load/Store word instruction



Example:

Each row in the table is a byte

Assume a’s address is in s1. Write RISC-V instructions to do

Int a, b;

b = a

lw t0, 0(sl) #To <- a

sw t0, 4(sl) #To -> b

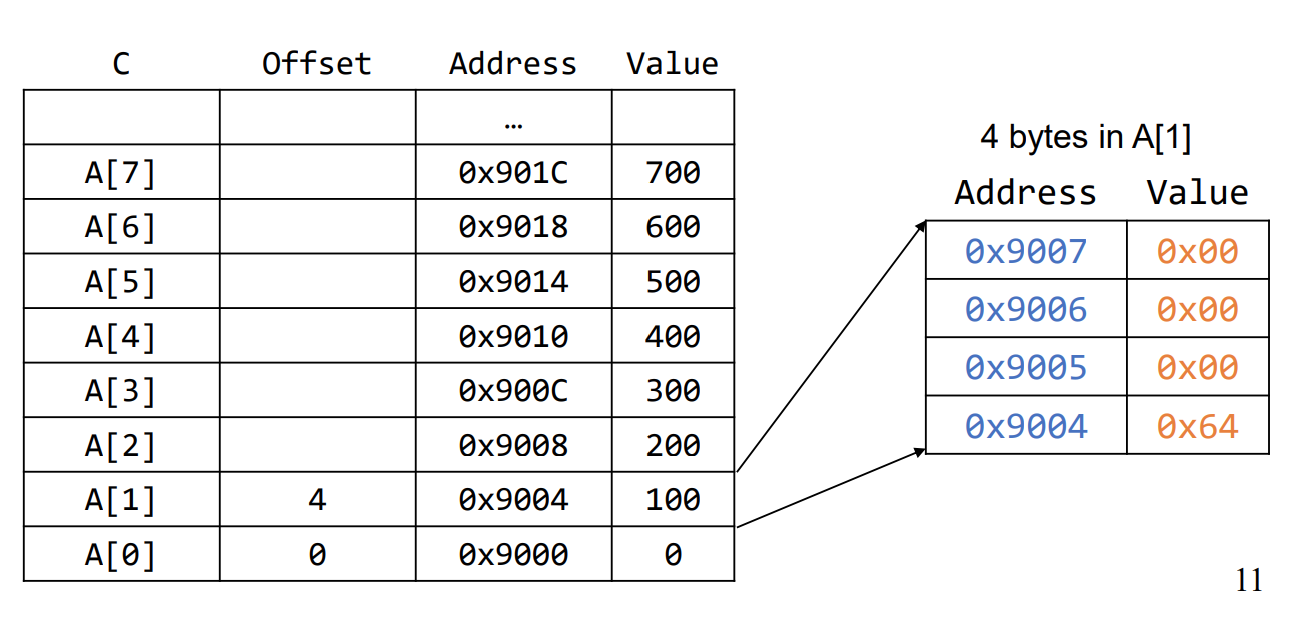
Array access

Suppose word array A starts from 0x9000 (stored in s1)

A[0], A[1], A[2], … A[i]

We need to specify an address in load/store instructions.

What are the addresses of these words?



Offset is 4\*i(index)

If you know of the offset and beginning of array

If index is a small constant, (smaller than 2047)

Asume word array A’s starting (base) address is in register s1.

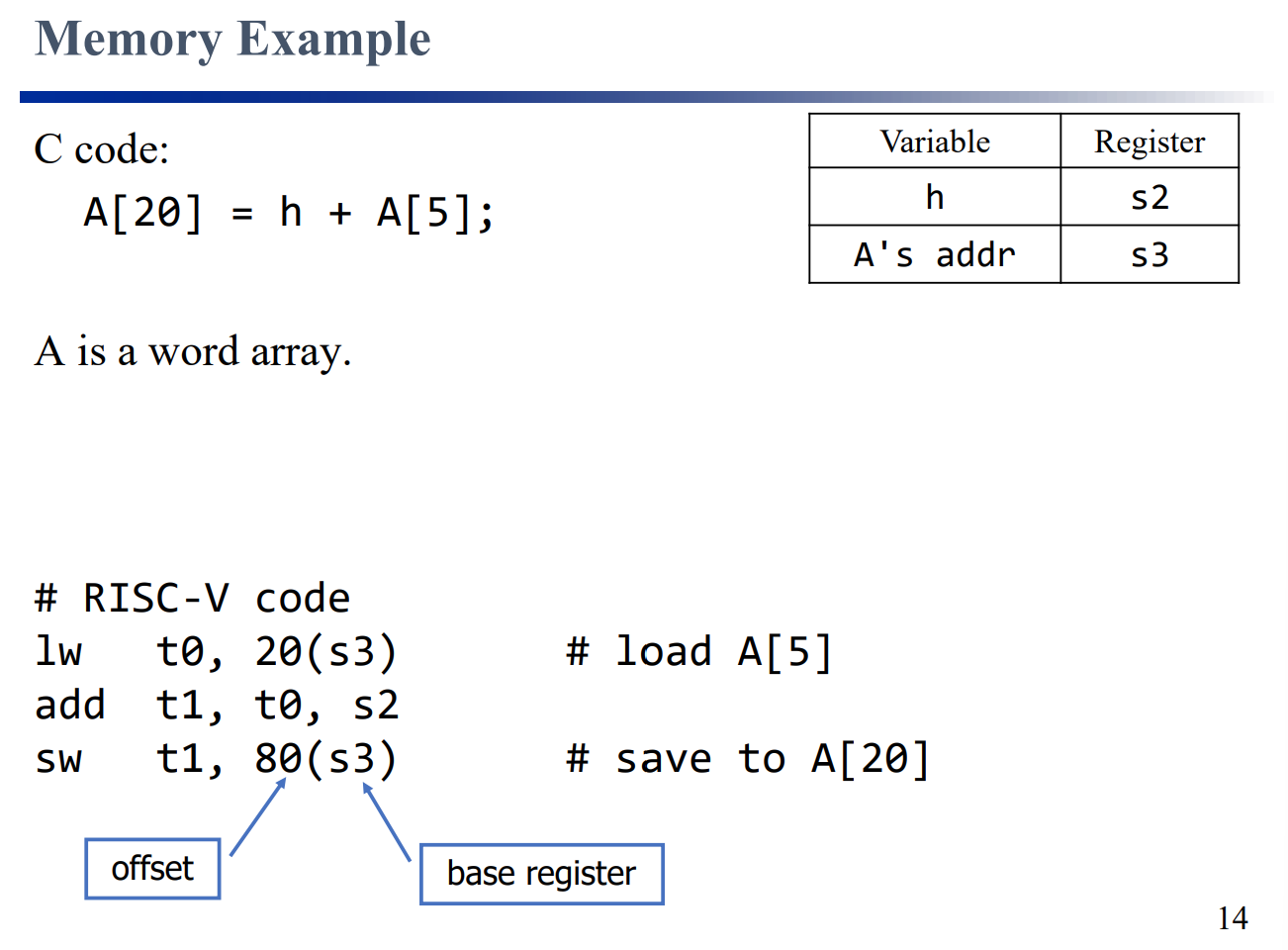
lw t0, 0(s1) #A[0]

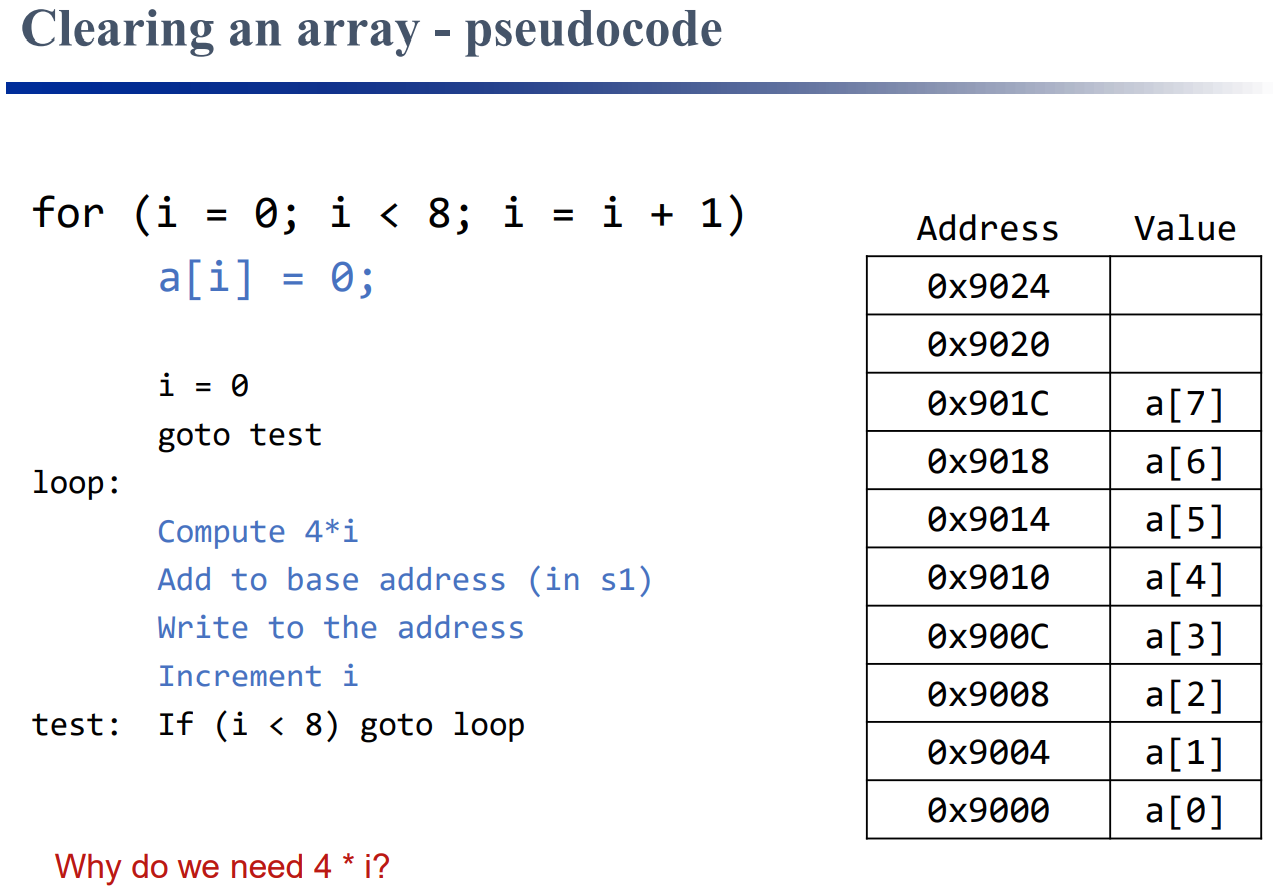
lw t1, 4(s1) #A[1]

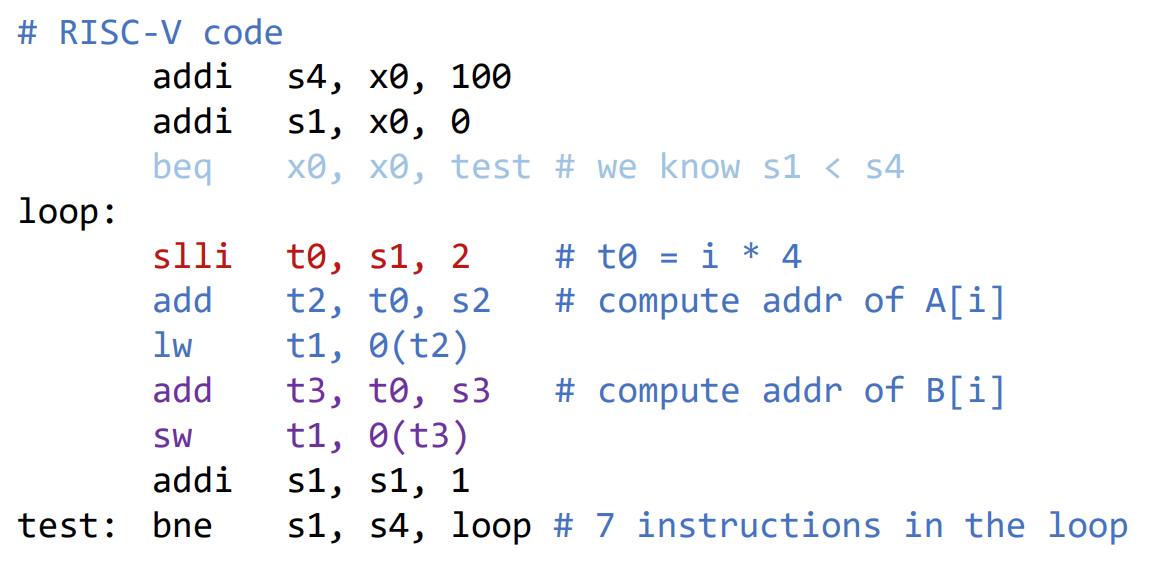
lw t2, 40(s1) #A[10]

S2 << 2

4\*i







Address alignment

Data items address is a multiple of its size

Address words os a multiple of 4

Address of half words is a multiple of 2

Data address do not have to be aligned in RISC-V, but

Misalignment will cause poor performance

Must be aligned in this course

Align the address of next variable to 2^2 =4

.align 2