References for this chapter

Computer Organization and Design – The Hardware/Software Interface
Chapter 2
Elements of equal size sequentially in memory

Maybe be good to

Data Structures

Array addressing via element pointer

Array addressing

Array addressing via index register

Calculate \( \sum \)

Replacing indices with offsets

Replacing multiplication with shifted index register

Replacing offsets with addresses
253

Summary

- Arrays
  - Structure
  - Alignment
  - Addressing
  - Iterators
  - Copy procedures

249

Array Slices

numbers = [0, 1, 2, 3, 4, 5]
numbersSlice = numbers[1:3]  # numbersSlice equals [1, 2, 3]

type Naturals is array (Integer range <>) of Natural;
Numbers: constant Naturals (-50 .. 50) := (others => Random (Generator));
Numbers_Slice_1: constant Naturals := Numbers (-10 .. 10);
Numbers_Slice_2: constant Naturals := Numbers (11 .. 20);
Numbers_Slice_3: constant Naturals := Numbers (-20 .. 50);

begin
for n of Numbers_Slice_3 loop
  n := n + 1;
end loop;
end;

Are those copy or reference affairs?

250

Copy array slice

; r0 base address for array a
; r1 from array index
; r2 to array index >= from index
; r3 base address for array b

lsl r1, r1, #2; translate from index to offset
lsl r2, r2, #2; translate to index to offset
add r1, r0; translate from index to address -> i_addr
add r2, r0; translate to index to address -> to_addr

for_copy:
  ldr r4, [r1], #4; a[i] := [i_addr]; i_addr += 4
  str r4, [r3], #4; [j_addr] := a[i]; j_addr += 4
  cmp r1, r2; i_addr <= to_addr
  ble for_copy
end_for_copy:

b[] := a[from .. to]

Moving blocks of memory can be done even/much faster with special hardware.

DMA controllers