Laboratory Exercise 6

Array and Pointer

Goals

After this laboratory Exercise, you should understand how array and pointer are represented and you will be able to differentiate index and pointer using in stepping through an array of list.

Literature

Patterson, Henessy (COD): section 2.8, 2.13

Preparation

Before you start the exercise, you should review the textbook, section 6.1 and read this laboratory carefully.

Array and Pointer

In a wide variety of programming tasks, it becomes necessary to step through an array or list, examining each of its elements in turn. For example, to determine the largest value in a list of integers, every element of the list mut be examined. There are two basic ways of accomplishing this:

1. Index: use a register that holds the index i and increment the register in each step to effect moving from element i of the list to element i+1
2. Pointer: use a register that points to (holds the address of) the list elements being examined and update it in each step to point to the next element.

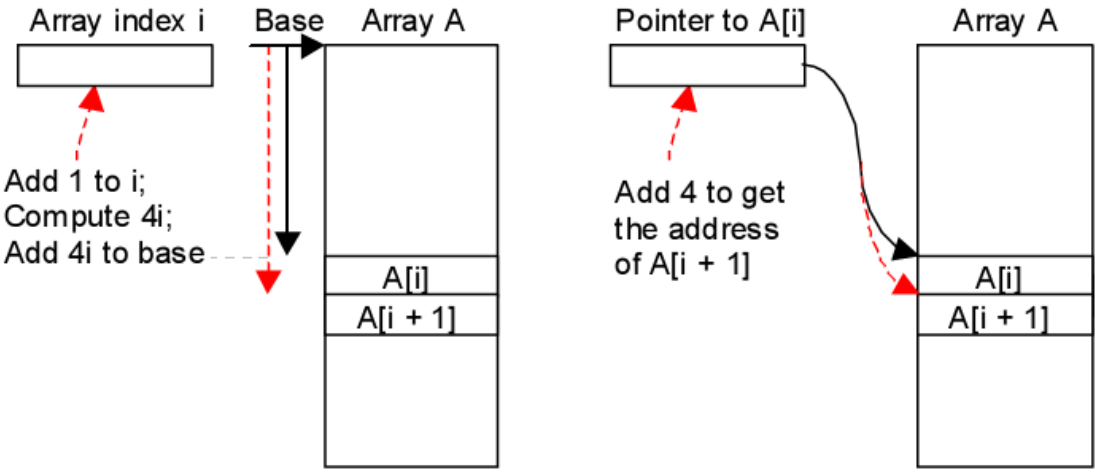
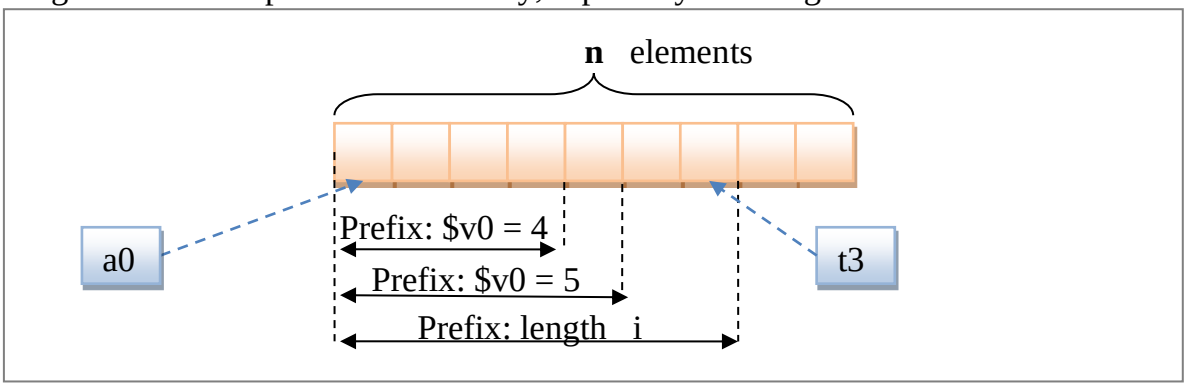


Figure 1. Using the indexing method and the pointer updating method to step through the elements of an array

Assignments at Home and at Lab

Home Assignment 1

Consider a list of integers of length n. A prefix of length I for the given list consists of the first i integers in the list, where 0 <= i <=n. A maximum-sum prefix, as the name implies, is a prefix for which the sum of elements is the largest among all prefixes. For example, if the list is (2,-3,2,5,-4), its maximum-sum prefix consists of the first four elements and the associated sum is 2-3+2+5=6; no other prefix of the given list has a larger sum. The following procedure uses an indexing method to find the maximum-sum prefix in a list of integers. Read this procedure carefully, especially indexing method.

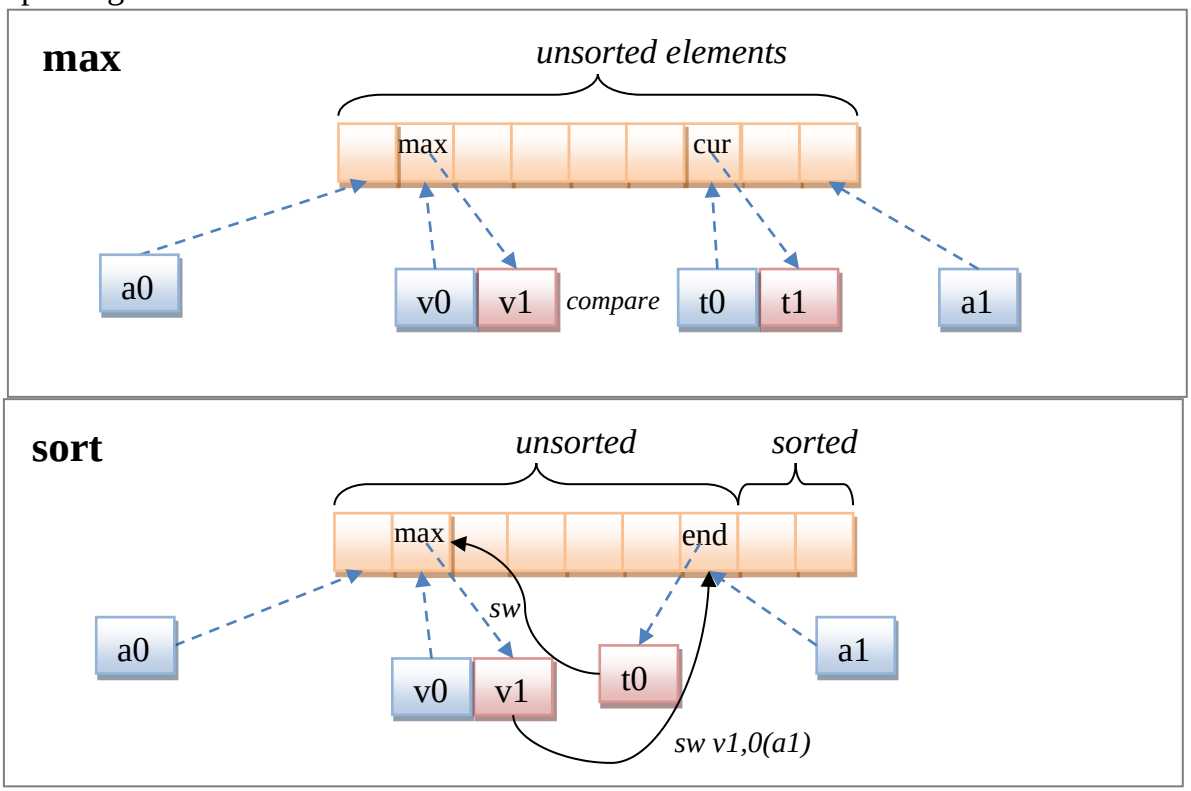



```
.data
A: .word -2, 6, -1, 3, -2

.text
main:       la    $a0,A
            li    $a1,5
            j     mspfx
            nop
continue:
lock:       j  lock
            nop
end_of_main:

#---------------------------------------------------------------
#Procedure mspfx
# @brief    find the maximum-sum prefix in a list of integers
# @param[in] a0   the base address of this list(A) need to be
processed
# @param[in] a1   the number of elements in list(A)
# @param[out] v0  the length of sub-array of A in which max sum
reachs.
# @param[out] v1  the max sum of a certain sub-array
#---------------------------------------------------------------
#Procedure mspfx
#function: find the maximum-sum prefix in a list of integers
#the base address of this list(A) in $a0 and the number of
#elements is stored in a1
mspfx:  addi      $v0,$zero,0 #initialize length in $v0 to 0
        addi      $v1,$zero,0 #initialize max sum in $v1to 0
        addi      $t0,$zero,0 #initialize index i in $t0 to 0
        addi      $t1,$zero,0 #initialize running sum in $t1 to 0
loop: add   $t2,$t0,$t0       #put 2i in $t2
      add   $t2,$t2,$t2       #put 4i in $t2
      add   $t3,$t2,$a0       #put 4i+A (address of A[i]) in $t3
```

```
      lw    $t4,0($t3)          #load A[i] from mem(t3) into $t4
      add   $t1,$t1,$t4         #add A[i] to running sum in $t1
      slt   $t5,$v1,$t1         #set $t5 to 1 if max sum < new sum
      bne   $t5,$zero,mdfy      #if max sum is less, modify results
      j     test                #done?
mdfy: addi  $v0,$t0,1           #new max-sum prefix has length i+1
      addi  $v1,$t1,0           #new max sum is the running sum
test: addi  $t0,$t0,1           #advance the index i
      slt   $t5,$t0,$a1         #set $t5 to 1 if i<n
      bne   $t5,$zero,loop      #repeat if i<n
done: j     continue
mspfx_end:
```

Home Assignment 2

A given list of n numbers can be sorted in ascending order as follows. Find the largest number in the list (there may be more than one) and swap it with the last element in the list. The new last element is now in its proper position in sorted order. Now, sort the remaining n-1 elements using the same step repeatedly. When only one element is left, sorting is complete. This method is known as selection sort. This example demonstrates this kind of sorting using pointer updating method. Read this procedure carefully and pay attention to pointer updating method.




```
.data
A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
Aend: .word

.text
main:       la   $a0,A           #$a0 = Address(A[0])
```

```
            la   $a1,Aend
            addi $a1,$a1,-4      #$a1 = Address(A[n-1])
            j    sort            #sort
after_sort: li   $v0, 10         #exit
            syscall
end_main:
#------------------------------------------------------------
#procedure sort (ascending selection sort using pointer)
#register usage in sort program
#$a0 pointer to the first element in unsorted part
#$a1 pointer to the last element in unsorted part
#$t0 temporary place for value of last element
#$v0 pointer to max element in unsorted part
#$v1 value of max element in unsorted part
#------------------------------------------------------------
sort:       beq     $a0,$a1,done    #single element list is sorted
            j       max             #call the max procedure
after_max:  lw      $t0,0($a1)      #load last element into $t0
            sw      $t0,0($v0)      #copy last element to max location
            sw      $v1,0($a1)      #copy max value to last element
            addi    $a1,$a1,-4      #decrement pointer to last element
            j       sort            #repeat sort for smaller list
done:       j       after_sort

#------------------------------------------------------------------------
---
#Procedure max
#function: fax the value and address of max element in the list
#$a0 pointer to first element
#$a1 pointer to last element
#------------------------------------------------------------------------
---
max:
      addi  $v0,$a0,0         #init max pointer to first element
      lw    $v1,0($v0)        #init max value to first value
      addi  $t0,$a0,0         #init next pointer to first
loop:
      beq   $t0,$a1,ret       #if next=last, return
      addi  $t0,$t0,4         #advance to next element
      lw    $t1,0($t0)        #load next element into $t1
      slt   $t2,$t1,$v1       #(next)<(max) ?
      bne   $t2,$zero,loop    #if (next)<(max), repeat
      addi  $v0,$t0,0         #next element is new max element
      addi  $v1,$t1,0         #next value is new max value
      j     loop              #change completed; now repeat
ret:
      j     after_max
```

Assignment 1

Create a new project to implement procedure in Home Assignment 1. Add code to the main program and initialize data for the integer list. Compile and upload to simulator. Run this program step by step, observe the process of exploring each element of the integer list using the indexing method.

Assignment 2

Create a new project to implement procedure in Home Assignment 2. Add code to the main program and initialize data for the integer list. Compile and upload to simulator. Run this program step by step, observe the process of exploring each element of the integer list using pointer updating method.

Assignment 3

Write a procedure to implement bubble sort algorithm.

Assignment 4

Write a procedure to implement insertion sort algorithm.

Conclusions

Before you pass the laboratory exercise, think about the questions below:

- What the advantage and disadvantage of two methods: indexing and updating pointer?